IBM COBOL for Linux en x86 1.2

Referencia de idioma

SC28-3117-01

Nota

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general en “Avisos” en la página 589.

Segunda edición

Esta edición se aplica a la versión 1.2 de IBM® COBOL for Linux® en x86 (número de programa 5737-L11) y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones. Asegúrese de que está utilizando la edición correcta para el nivel del producto.

Puede ver o descargar publicaciones en copia software de forma gratuita en COBOL for Linux en la biblioteca x86.

Contenido

Tablas

Prefacio

Acerca de esta información

Esta información describe el lenguaje COBOL soportado por IBM COBOL for Linux en x86, a la que se hace referencia en esta información como COBOL para Linux.

Consulte la publicación *IBM COBOL for Linux en x86 Programming Guide* para obtener información y ejemplos que le ayudarán a escribir, compilar y depurar programas COBOL.

Cómo leer los diagramas de sintaxis

En todo el documento, los diagramas ilustran la sintaxis COBOL para Linux .

Utilice la descripción siguiente para leer los diagramas de sintaxis de este documento:

- Lea los diagramas de sintaxis de izquierda a derecha, de arriba abajo, siguiendo la ruta de la línea.

  El símbolo ▶▶— indica el principio de un diagrama de sintaxis.

  El símbolo —▶ indica que el diagrama de sintaxis continúa en la línea siguiente.

  El símbolo ▶— indica que el diagrama de sintaxis continúa desde la línea anterior.

  El símbolo —▶◀ indica el final de un diagrama de sintaxis.

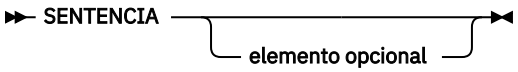
  Los diagramas de unidades sintácticas que no sean sentencias completas empiezan con el símbolo ▶— y terminan con el símbolo —▶ .
- Los elementos necesarios aparecen en la línea horizontal (la ruta principal).

  Formato

- Los elementos opcionales aparecen debajo de la ruta principal.

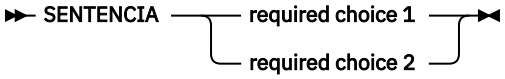
  Formato

- Cuando puede elegir entre dos o más elementos, aparecen verticalmente, en una pila.

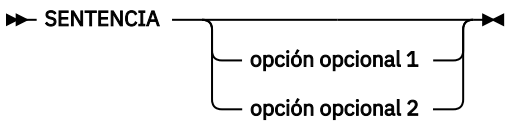
  Si *debe* elegir uno de los elementos, aparecerá un elemento de la pila en la vía de acceso principal.

  Formato

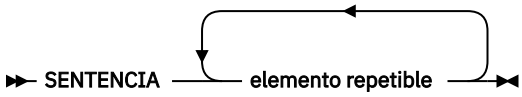
  Si elegir uno de los elementos es opcional, toda la pila aparece debajo de la ruta principal.

Formato



- Una flecha que vuelve a la izquierda sobre la línea principal indica un elemento que se puede repetir.

Formato

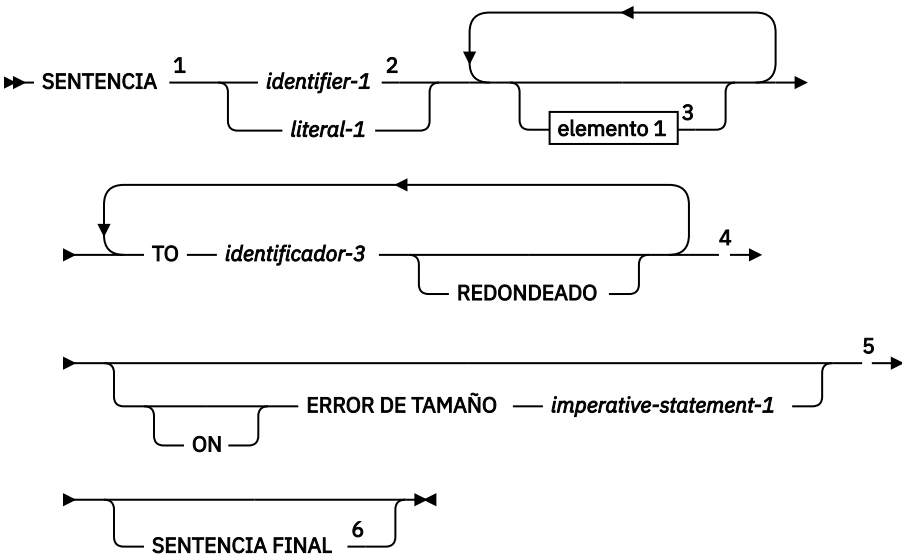


Una flecha de repetición sobre una pila indica que puede realizar más de una opción de los elementos apilados, o repetir una sola opción.

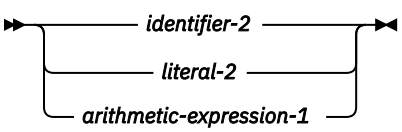
- Las variables aparecen en minúsculas en cursiva (por ejemplo, *parmx*). Representan nombres o valores proporcionados por el usuario.
- Si se muestran signos de puntuación, paréntesis, operadores aritméticos u otros símbolos de este tipo, se deben especificar como parte de la sintaxis.

El ejemplo siguiente muestra cómo se utiliza la sintaxis.

Formato



elemento 1



Notas:

[1] La palabra clave STATEMENT debe especificarse y codificarse tal como se muestra.
[2] Este operando es obligatorio. Debe codificarse *identifier-1* o *literal-1* .
[3] El fragmento de elemento 1 es opcional; se puede codificar o no, según lo requiera la aplicación. Si el elemento 1 está codificado, se puede repetir con cada entrada separada por uno o más

separadores COBOL. Las selecciones de entrada permitidas para este fragmento se describen en la parte inferior del diagrama.
[4] El operando *identifier-3* y la palabra clave TO asociada son necesarios y se pueden repetir con uno o más separadores COBOL que separan cada entrada. A cada entrada se le puede asignar la palabra clave REDONDEADA.
[5] La frase ON SIZE ERROR con *imperative-statement-1* asociada es opcional. Si la frase ON SIZE ERROR está codificada, la palabra clave ON es opcional.
[6] La palabra clave END-STATEMENT puede codificarse para finalizar la sentencia. No es un delimitador necesario.

Cómo utilizar ejemplos

Los ejemplos de código de programa en esta información se escriben en minúsculas, mayúsculas o mayúsculas y minúsculas para demostrar que puede escribir los programas de cualquiera de estas maneras.

Para separar más claramente algunos ejemplos del texto explicativo, se presentan en a `monospace font`.

Las palabras clave COBOL y las opciones de compilador que aparecen en el texto se muestran generalmente en SMALL UPPERCASE. Otros términos como nombres de variables de programa se muestran a veces en *una fuente en cursiva* para mayor claridad.

Si copia y pega ejemplos de la documentación en formato PDF, asegúrese de que los espacios de los ejemplos (si los hay) están en su lugar; es posible que tenga que añadir manualmente algunos espacios que faltan para asegurarse de que el texto de origen COBOL se alinea con las columnas necesarias según el formato de referencia COBOL. De forma alternativa, puede copiar y pegar ejemplos de la documentación de formato HTML y los espacios ya deben estar en su lugar.

IBM

Las extensiones IBM generalmente añaden características, sintaxis o reglas que no están especificadas en los estándares ANSI e ISO COBOL listados en Apéndice G, “Especificaciones de la industria”, en la página 587. En este documento, el término 85 COBOL Estándar hace referencia a dichos estándares.

Las extensiones van desde la relajación menor de las reglas hasta las prestaciones principales, como el soporte XML, el soporte Unicode, la interoperatividad y el manejo de caracteres DBCS.

El resto de este documento describe el idioma completo sin identificar extensiones. Tendrá que revisar Apéndice A, “Extensiones de IBM”, en la página 541 y *Opciones de compilador* en *COBOL for Linux en x86 Guía de programación* si desea utilizar sólo elementos de lenguaje estándar.

Elementos de lenguaje obsoletos

Los elementos de lenguaje obsoletos son elementos que se clasifican como *obsoletos* en 85 COBOL Estándar. Estos elementos no forman parte de Estándar COBOL 2002.

Esto *no* implica que IBM eliminará los elementos obsoletos 85 COBOL Estándar de un futuro release de COBOL para Linux.

Los siguientes elementos de lenguaje están categorizados como obsoletos por 85 COBOL Estándar:

- Sentencia ALTER
- AUTOR, párrafo
- Entrada de comentario
- cláusula DATA RECORDS

- DATE-Párrafo COMPILADO
- DATE-Párrafo ESCRITO
- Registro especial DEBUG-ITEM
- Secciones de depuración
- ENTER, sentencia
- GO TO sin un nombre de procedimiento especificado
- Párrafo de INSTALACIÓN
- cláusula LABEL RECORDS
- cláusula MEMORY SIZE
- cláusula MULTIPLE FILE TAPE
- cláusula RERUN
- frase REVERSED
- Párrafo SEGURIDAD
- Módulo de segmentación
- Formato STOP *literal* de la sentencia STOP
- Declarativo USE FOR DEBUGGING
- cláusula VALUE OF
- La constante figurativa ALL *literal* con una longitud mayor que uno, cuando la constante figurativa está asociada a un elemento numérico o numérico editado

notación DBCS

Las series de juego de caracteres de doble byte (DBCS) en literales, comentarios y palabras definidas por el usuario están delimitadas por caracteres de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico.

En este documento, los caracteres DBCS se muestran con este formato: D1D2D3. Los caracteres alfabéticos latinos en la representación DBCS se muestran en este formato: .A.B.C.

Notas:

- En los datos DBCS EBCDIC que contienen caracteres mixtos de un solo byte y de doble byte, las series de caracteres de doble byte están delimitadas por caracteres de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico. En este documento, el delimitador de desplazamiento a teclado ideográfico se representa pictóricamente mediante el carácter < y el carácter de desplazamiento a teclado ideográfico se representa pictóricamente mediante el carácter >. Los códigos EBCDIC de un solo byte para los delimitadores de desplazamiento a teclado ideográfico y de desplazamiento a teclado estándar son X'0E'y X'0F', respectivamente. El símbolo < > indica caracteres de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico contiguos. El símbolo > < indica caracteres de desplazamiento a teclado ideográfico y a teclado ideográfico contiguos.
- En los datos DBCS ASCII que contienen caracteres mixtos de un solo byte y de doble byte, las series de caracteres de doble byte no están delimitadas por caracteres de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico.

Acuse de recibo

El siguiente extracto del Formulario de la Oficina de Impresión del Gobierno 1965-0795689 se presenta para la información y orientación del usuario:

> Cualquier organización interesada en reproducir el informe COBOL y las especificaciones en su totalidad o en parte, utilizando ideas tomadas de este informe como base para un manual de instrucciones o para cualquier otro fin, es libre de hacerlo. Sin embargo, se solicita a todas estas organizaciones que reproduzcan esta sección como parte de la introducción al documento. A

los que utilizan un pasaje corto, como en una revisión de libro, se les solicita que mencionen COBOL en acuse de recibo de la fuente, pero no es necesario que citen toda esta sección.

COBOL es un lenguaje de la industria y no es propiedad de ninguna empresa o grupo de empresas, ni de ninguna organización o grupo de organizaciones.

Ningún colaborador ni el Comité COBOL ofrecen ninguna garantía, expresa o implícita, en cuanto a la precisión y el funcionamiento del sistema y el lenguaje de programación. Por otra parte, ninguna responsabilidad es asumida por ningún contribuyente, o por la comisión, en relación con ello.

Se han establecido procedimientos para el mantenimiento de COBOL. Las investigaciones relativas a los procedimientos para proponer cambios deben dirigirse al Comité Ejecutivo de la Conferencia sobre Lenguas de los Sistemas de Datos.

Los autores y titulares de derechos de autor del material con derechos de autor:

- FLOW-MATIC (Trademark of Sperry Rand Corporation), Programming for the UNIVAC (R) I and II, Data Automation Systems copyright en 1958, 1959, por Sperry Rand Corporation
- IBM Commercial Translator, Formulario No. F28-8013, copyright en 1959 por IBM
- FACT, DSI 27A5260-2760, copyright en 1960 de Minneapolis-Honeywell

han autorizado específicamente el uso de este material en su totalidad o en parte, en las especificaciones COBOL. Dicha autorización se extiende a la reproducción y uso de especificaciones COBOL en manuales de programación o publicaciones similares.

Nota: la Conferencia sobre idiomas de sistemas de datos (CODASYL), mencionada anteriormente, ya no existe.

Accesibilidad

Las características de accesibilidad ayudan a los usuarios que tienen una discapacidad, como por ejemplo movilidad restringida o visión limitada, a utilizar correctamente los productos de tecnología de la información.

Funciones de accesibilidad

IBM COBOL for Linux en x86 utiliza el estándar W3C más reciente, WAI-ARIA 1.0, para garantizar la conformidad con US Section 508 y Web Content Accessibility Guidelines (WCAG) 2.0. Para aprovechar las características de accesibilidad, utilice el release más reciente del lector de pantalla en combinación con el navegador web más reciente soportado por este producto.

Navegación mediante teclado

Este producto utiliza teclas de navegación estándar.

Información de interfaz

Puede utilizar el software de reconocimiento de voz como una herramienta de texto a voz (TTS) para ver la salida generada por el producto.

La documentación del producto en línea está disponible en la documentación de IBM , que se puede ver desde un navegador web estándar.

Los archivos PDF tienen un soporte de accesibilidad limitado. Con la documentación en PDF, puede utilizar la ampliación de fuentes opcional, los valores de visualización de alto contraste y puede navegar solo con el teclado.

Para que el lector de pantalla pueda leer con precisión los diagramas de sintaxis, los ejemplos de código fuente y el texto que contiene los símbolos PICTURE de punto o coma, debe establecer el lector de pantalla para que hable todos los signos de puntuación.

Información de accesibilidad relacionada

Además de los sitios web de soporte y servicio de asistencia técnica estándar de IBM , IBM ha establecido un servicio telefónico TTY para que los clientes sordos o con dificultades auditivas puedan acceder a los servicios de soporte y ventas:

Servicio de teletipo 800-IBM-3383 (800-426-3383) (en Norteamérica)

IBM y accesibilidad

Para obtener más información sobre el compromiso que IBM tiene con la accesibilidad, consulte IBM Accesibilidad.

Cómo enviar sus comentarios

Sus comentarios son importantes para ayudarnos a proporcionar información precisa y de alta calidad. Si tiene comentarios sobre este documento o cualquier otra documentación para este producto, envíe sus comentarios a: compinfo@cn.ibm.com.

Asegúrese de incluir el nombre del documento, el número de publicación del documento, la versión del producto y, si procede, la ubicación específica (por ejemplo, número de página o cabecera de sección) del texto que está comentando.

Cuando envía información a IBM, concede a IBM un derecho no exclusivo para utilizar o distribuir la información de cualquier forma que IBM considere adecuada sin incurrir por ello en ninguna obligación con el usuario.

Parte 1. Estructura de lenguaje COBOL

Capítulo 1. Caracteres

La unidad más básica e indivisible del lenguaje COBOL es el *carácter*. El juego de caracteres básico incluye las letras del alfabeto latino, los dígitos y los caracteres especiales.

En el lenguaje COBOL, los caracteres individuales se unen para formar *series de caracteres* y *separadores*. Las cadenas de caracteres y separadores, entonces, se utilizan para formar las palabras, literales, frases, cláusulas, sentencias y oraciones que forman el idioma.

Los caracteres básicos utilizados para formar series de caracteres y separadores en el código fuente se muestran en Tabla 1 en la página 3.

Para determinados elementos de lenguaje, el juego de caracteres básico se amplía con el juego de caracteres de doble byte (DBCS) juegos de caracteres siguientes, en función de la *página de códigos* utilizada en el tiempo de compilación:

- Juego de caracteres de doble byte ASCII (DBCS). Los caracteres DBCS ocupan 2 bytes adyacentes para representar un carácter. Los caracteres representados en varios bytes en el código fuente (incluidos los caracteres DBCS) se conocen en este documento como *caracteres multibyte*. Una serie de caracteres que contiene sólo caracteres DBCS también se denomina *serie de caracteres DBCS* o *serie de caracteres de doble byte*.
- UTF-8, un formato de codificación del juego de caracteres Unicode. Los caracteres UTF-8 ocupan de uno a cuatro bytes por carácter. Los caracteres UTF-8 que ocupan 2 o más bytes se conocen en este documento como *caracteres de varios bytes*.
- Código UNIX ampliado (EUC). Los caracteres EUC ocupan de 1 byte a 4 bytes por carácter (o de 1 byte a 3 bytes, en función de la página de códigos). Los caracteres EUC que ocupan 2 o más bytes se conocen en este documento como *caracteres multibyte.*

Los caracteres Multibyte se pueden utilizar para formar palabras definidas por el usuario.

El contenido de literales alfanuméricos, líneas de comentario y entradas de comentario puede incluir cualquiera de los caracteres del juego de caracteres de tiempo de compilación del sistema, y puede incluir caracteres de un solo byte y multibyte .

Los datos de tiempo de ejecución pueden incluir cualquier carácter del juego de caracteres de tiempo de ejecución del sistema. El juego de caracteres de tiempo de ejecución del sistema puede incluir caracteres alfanuméricos, multibyte caracteres y caracteres nacionales. Los caracteres nacionales se representan en UTF-16, un formato de codificación de 16 bits de Unicode.

Cuando la opción de compilador NSYMBOL (NATIONAL) está en vigor, los literales identificados por el delimitador de apertura N " o N' son literales nacionales y pueden contener cualquier carácter de un solo byte o de multibyte , o ambos, que son válidos para la página de códigos de tiempo de compilación de página de códigos de tiempo de compilación. Los caracteres contenidos en literales nacionales se representan como caracteres nacionales en tiempo de ejecución.

Para obtener detalles, consulte “Palabras definidas por el usuario con multibyte caracteres” en la página 12, “Literales DBCS” en la página 33y “Literales nacionales” en la página 35.

Tabla 1. ***Conjunto de caracteres COBOL básico****.* Esta tabla lista el juego de caracteres COBOL básico.

Carácter	Significado	Uso	Ejemplo
	Espacio	Carácter de puntuación	`01 WS-A PIC X(10).`

Tabla 1. ***Conjunto de caracteres COBOL básico****.* Esta tabla lista el juego de caracteres COBOL básico. *(continuación)*

Carácter	Significado	Uso	Ejemplo
+	Signo más	Operador aritmético	`COMPUTE WS-A = WS-B + WS-C.`
		Carácter de edición	`01 WS-A PIC +9(3).`
-	Signo menos o guión	Operador aritmético	`COMPUTE WS-A = WS-B - WS-C.`
		Carácter de edición	`01 WS-A PIC -9(3).`
		Carácter de continuación	`  01 WS-VAR  PIC X(27) VALUE -      'THIS MULTI- LINE TEXT'.`
		Elemento de palabra COBOL	`01 WS-A PIC 9(3).`
*	Asterisco	Operador aritmético	`COMPUTE WS-A = WS-B * WS-C.`
		Carácter de edición	`01 WS-A PIC **9.`
		Carácter de comentario	`* THIS IS COMMENT LINE.`
/	Barra inclinada o solidus	Operador aritmético	`COMPUTE WS-A = WS-B / WS-C.`
		Carácter de edición	`01 WS-DATE PIC 99/99/99.`
		Carácter de continuación	`/01 WS-VAR  PIC X(27) VALUE /     'THIS MULTI- LINE TEXT'.`
=	Signo igual	Carácter de asignación	`COMPUTE WS-A = WS-B / WS-C.`
		Carácter de relación	`IF WS-A = 10`
$	Signo de moneda	Carácter de edición	`01 WS-DATE PIC $$99.`

Tabla 1. ***Conjunto de caracteres COBOL básico****.* Esta tabla lista el juego de caracteres COBOL básico. *(continuación)*

Carácter	Significado	Uso	Ejemplo
,	Coma	Carácter de edición	`01 WS-DATE PIC 99,999.`
		Carácter de puntuación	`MOVE 10 TO WS-A, WS-B.`
;	Punto y coma	Carácter de puntuación	`MOVE 10 TO WS-A; WS-B.`
.	Coma decimal o punto	Carácter de edición	`01 WS-DATE PIC 99.999.`
		Carácter de puntuación	`MOVE 10 TO WS-A, WS-B.`
"	Comillas	Carácter de puntuación	`01 WS-VAR PIC X(5) VALUE "HELLO".`
'	Apóstrofo	Carácter de puntuación	`01 WS-VAR PIC X(5) VALUE 'HELLO'.`
(	Paréntesis izquierdo	Carácter de puntuación	`IF (WS-A = 10) AND (WS-B = 5)`
)	Paréntesis derecho	Carácter de puntuación	`IF (WS-A = 10) AND (WS-B = 5)`
>	Mayor que	Carácter de relación	`IF WS-A > 10`
<	Menor que	Carácter de relación	`IF WS-A < 10`
:	Dos puntos	Carácter de relación	`MOVE WS-VAR(1:10) TO WS-VAR1.`
_	Subrayado	Elemento de palabra definida por el usuario	`01 WS_VAR PIC X(10).`
A-Z	Alfabeto (mayúsculas)	Caracteres alfabéticos	/
a-z	Alfabeto (minúsculas)	Caracteres alfabéticos	/
0 - 9	Caracteres numéricos	Caracteres numéricos	/

Capítulo 2. Juegos de caracteres y páginas de códigos

Un *juego de caracteres* es un conjunto de letras, números, caracteres especiales y otros elementos utilizados para representar información. El término *página de códigos* hace referencia a un juego de caracteres codificado.

Un juego de caracteres es independiente de una representación codificada. Un *juego de caracteres codificado* es la representación codificada de un conjunto de caracteres, donde a cada carácter se le asigna una posición numérica, denominada *elemento de código*, en el esquema de codificación. ASCII y EBCDIC son ejemplos de tipos de juegos de caracteres codificados. Cada variación de ASCII o EBCDIC es un juego de caracteres codificado específico.

Cada página de códigos que define IBM se identifica mediante un *nombre de página de códigos*, por ejemplo IBM-1252, y un *identificador de juego de caracteres codificado* (CCSID), por ejemplo 1252.

Página de códigos de tiempo de compilación

La página de códigos en tiempo de compilación puede ser una página de códigos ASCII de un solo byte o ASCII de doble byte, una página de códigos EUC o UTF-8. La página de códigos específica se indica mediante el entorno local de tiempo de compilación, o la variable de entorno en vigor.

El programa fuente (incluidas las palabras definidas por el usuario y el contenido de literales alfanuméricos, DBCS y nacionales) se codifica en la página de códigos indicada por el entorno local o variable de entorno en vigor en el momento de la compilación.

Página de códigos de tiempo de ejecución

La página de códigos utilizada en tiempo de ejecución viene determinada por una combinación de la cláusula USAGE de un elemento de datos, las opciones de compilador en vigor y el entorno local (o valor de variable de entorno) en vigor.

Cuando la opción de compilador CHAR (NATIVE) está en vigor, los elementos de datos descritos con USAGE DISPLAY o USAGE DISPLAY-1 se codificada en una página de códigos ASCII, EUC o UTF-8 tal como indica el entorno local de ejecución.

Cuando la opción de compilador CHAR (EBCDIC) está en vigor, los elementos de datos descritos con USAGE DISPLAY o USAGE DISPLAY-1 se codifican en una página de códigos EBCDIC, excepto cuando se especifica la frase NATIVE en la cláusula USAGE del elemento. Si se especifica la frase NATIVE, la página de códigos utilizada es la Página de códigos ASCII, EUC o UTF-8 indicada por el entorno local de ejecución.

Para EBCDIC, la página de códigos se determina a partir de la variable de entorno EBCDIC_CODEPAGE, si se ha establecido. Si la variable de entorno EBCDIC_CODEPAGE no está establecida, se utiliza la página de códigos EBCDIC predeterminada asociada con el entorno local de ejecución actual. La página de códigos EBCDIC predeterminada asociada con cada entorno local soportado se identifica en *Entornos locales y páginas de códigos soportadas* en *COBOL for Linux en x86 Guía de programación*.

Para DBCS, la página de códigos viene determinada por la variable de entorno DBCS_CODEPAGE, si se ha establecido. Si la variable de entorno DBCS_CODEPAGE no está establecida, se utiliza la página de códigos DBCS predeterminada asociada con el entorno local de ejecución actual.

La página de códigos de predeterminado para elementos de datos descritos con USAGE NATIONAL y los literales nacionales es UTF-16LE (little endian), CCSID 1200. La representación de texto de origen de literales nacionales se convierte en tiempo de ejecución de la página de códigos en tiempo de compilación a UTF-16LE.

Para cambiar la representación de endianness para elementos de datos descritos con USAGE NATIONAL y literales nacionales, consulte la opción de compilador UTF16 .

Una referencia a *UTF-16* en este documento es una referencia a UTF-16LE.

Unidades de codificación de caracteres

Una *unidad de codificación de caracteres* (o *unidad de codificación*) es la unidad de datos que COBOL trata como un único carácter en tiempo de ejecución. En esta información, los términos *carácter* y *posición de carácter* hacen referencia a una sola unidad de codificación.

El tamaño de una unidad de codificación para elementos de datos y literales depende de la cláusula USAGE del elemento de datos o de la categoría del literal como se indica a continuación:

- Para los elementos de datos descritos con USAGE DISPLAY y para literales alfanuméricos, una unidad de codificación es de 1 byte, independientemente de la página de códigos utilizada y del número de bytes utilizados para representar un *carácter gráfico*determinado.
- Para elementos de datos descritos con USAGE DISPLAY-1 (elementos de datos DBCS) y para literales DBCS, una unidad de codificación es de 2 bytes.
- Para elementos de datos descritos con USAGE NATIONAL y para literales nacionales, una unidad de codificación es de 2 bytes.

La relación entre un carácter gráfico y una unidad de codificación depende del tipo de página de códigos utilizada para el elemento de datos o literal. Consulte los siguientes tipos de páginas de códigos de tiempo de ejecución:

- ASCII o EBCDIC de un solo byte
- Multibyte basado en ASCII (ASCII DBCS, EUC o UTF-8) o EBCDIC DBCS multibyte
- Unicode UTF-16

Consulte las secciones siguientes para obtener los detalles de cada tipo de página de códigos.

Consulte también la sección *Especificación de la página de códigos para datos de caracteres* en *COBOL for Linux en x86 Guía de programación*.

Páginas de códigos de un solo byte

Puede utilizar una página de códigos de un solo byte en elementos de datos descritos con USAGE DISPLAY y en literales de categoría alfanumérica. Una unidad de codificación es de 1 byte y cada carácter gráfico se representa en 1 byte. Para estos elementos de datos y literales, no es necesario que se ocupe de las unidades de codificación.

Páginas de códigos de varios bytes

VISUALIZACIÓN DE USO

Puede utilizar datos codificados en una página de códigos EBCDIC o basada en ASCII de varios bytes en elementos de datos descritos con USAGE DISPLAY (categoría alfanumérica) y en literales de categoría alfanumérica. Una unidad de codificación es de 1 byte y el tamaño de un carácter gráfico varía de 1 byte a 4 bytes, en función de la página de códigos.

Cuando los elementos de datos alfanuméricos o literales contienen datosde varios bytes, los programadores son responsables de asegurarse de que las operaciones no separan involuntariamente las varias unidades de codificación que forman un carácter gráfico. Se debe tener cuidado con la modificación de referencia, y se debe evitar el truncamiento durante los movimientos. El sistema de tiempo de ejecución COBOL no comprueba si hay una división entre las unidades de codificación que forman un carácter gráfico.

Para evitar problemas, puede convertir literales alfanuméricos y elementos de datos descritos con el uso DISPLAY a datos nacionales (UTF-16) moviendo los elementos de datos o literales a elementos de datos descritos con el uso NATIONAL o utilizando la función intrínseca NATIONAL-OF. A continuación, puede realizar operaciones en los datos nacionales con menos preocupación por dividir los caracteres gráficos. Puede volver a convertir los datos a USAGE DISPLAY utilizando la función intrínseca DISPLAY-OF.

Cuando los datos de origen que convierte son UTF-8, debe preocuparse por dividir los caracteres gráficos en el elemento de datos NATIONAL de uso de destino porque los datos nacionales resultantes pueden

contener pares sustitutos UTF-16 o combinar secuencias de caracteres como se describe en “Unicode UTF-16” en la página 9.

USO DISPLAY-1

Puede utilizar caracteres de doble byte de una página de códigos ASCII multibyte DBCS o EBCDIC DBCS en elementos de datos descritos con USAGE DISPLAY-1 y en literales de la categoría DBCS. Una unidad de codificación es de 2 bytes y cada carácter gráfico se representa en una sola unidad de codificación de 2 bytes. Para estos elementos de datos y literales, no es necesario que se ocupe de las unidades de codificación.

No puede utilizar UTF-8 o EUC en los elementos de datos descritos con USAGE DISPLAY-1.

Unicode UTF-16

Puede utilizar UTF-16 en elementos de datos descritos con USAGE NATIONAL. Los literales nacionales se almacenan como caracteres UTF-16 independientemente de la página de códigos utilizada para el programa de origen. Una unidad de codificación para elementos de datos de uso NACIONAL y literales nacionales es de 2 bytes.

Para la mayoría de los caracteres en UTF-16, un carácter gráfico es una unidad de codificación. Los caracteres convertidos a UTF-16 desde una página de códigos EBCDIC, ASCII o EUC se representan en una unidad de codificación UTF-16 . Algunos de los otros caracteres gráficos en UTF-16 se representan mediante un *par sustituto* o una *secuencia de caracteres de combinación*. Un par sustituto consta de dos unidades de codificación (4 bytes). Una secuencia de caracteres de combinación consta de un carácter base y una o más *marcas de combinación* o una secuencia de una o más marcas de combinación (4 bytes o más, en incrementos de 2 bytes). En elementos de datos de uso NATIONAL, cada unidad de codificación de 2 bytes se trata como un carácter.

Cuando los datos nacionales contienen pares sustitutos o secuencias de caracteres de combinación, los programadores son responsables de asegurarse de que las operaciones en caracteres nacionales no separan involuntariamente las múltiples unidades de codificación que forman un carácter gráfico. Se debe tener cuidado con la modificación de referencia, y se debe evitar el truncamiento durante los movimientos. El sistema de tiempo de ejecución COBOL no comprueba si hay una división entre las unidades de codificación que forman un carácter gráfico.

Capítulo 3. Series de caracteres

Una *serie de caracteres* es un carácter o una secuencia de caracteres contiguos que forman una palabra COBOL, un literal, una serie de caracteres PICTURE o una entrada de comentario. Una serie de caracteres está delimitada por separadores.

Un *separador* es una serie de caracteres contiguos que se utiliza para delimitar las series de caracteres. Los separadores se describen en detalle en Capítulo 4, “Separadores”, en la página 39.

Las series de caracteres y determinados separadores forman *palabras de texto*. Una palabra de texto es un carácter o una secuencia de caracteres contiguos (posiblemente continuados entre líneas) entre las posiciones de caracteres 8 y 72 en formato de origen fijo, o las posiciones 8 y 252 en formato de origen ampliado inclusive en texto de origen, texto de biblioteca o pseudotexto. Para obtener más información sobre el pseudotexto, consulte “Pseudotexto” en la página 51.

El texto de origen, el texto de biblioteca y el pseudo-texto se pueden escribir en un solo byte y, para algunas series de caracteres, caracteres de varios bytes del ASCII, UTF-8o la página de códigos EUC indicada por el entorno local de tiempo de compilación.

Puede utilizar series de caracteres de byte único y multibyte para formar los elementos siguientes:

- Palabras COBOL
- Literales
- Texto de comentario

Sólo puede utilizar caracteres de un solo byte para formar series de caracteres PICTURE.

Palabras COBOL con caracteres de un solo byte

Una *palabra* COBOL es una serie de caracteres que forma una palabra definida por el usuario, un nombre de sistema o una palabra reservada. El tamaño máximo de una palabra COBOL definida por el usuario es de 30 bytes. El número de caracteres que se pueden especificar depende de la página de códigos indicada por el entorno local de tiempo de compilación.

A excepción de los operadores aritméticos y los caracteres de relación, cada carácter de una palabra COBOL se selecciona del conjunto siguiente:

- Letras mayúsculas latinas de la A a la Z
- Letras latinas minúsculas de la a a la z
- dígitos del 0 al 9
- -(guión)
- _ (subrayado)

El guión no puede aparecer como el primer o último carácter en tales palabras. El subrayado no puede aparecer como el primer carácter en estas palabras. La mayoría de las palabras definidas por el usuario (todas excepto los nombres de sección, los nombres de párrafo, los números de prioridad y los números de nivel) deben contener al menos un carácter alfabético. Los números de prioridad y los números de nivel no necesitan ser exclusivos; una especificación dada de un número de prioridad o número de nivel puede ser idéntica a cualquier otro número de prioridad o número de nivel.

En palabras COBOL (pero no en el contenido de caracteres alfanuméricos, DBCS, y nacionales), cada letra alfabética de un solo byte en minúsculas se considera equivalente a su correspondiente letra alfabética en mayúsculas de un solo byte.

Las reglas siguientes se aplican a todas las palabras COBOL:

- Una palabra reservada no se puede utilizar como una palabra definida por el usuario o como un nombre de sistema.

- Sin embargo, se puede utilizar la misma palabra COBOL como una palabra definida por el usuario y como un nombre de sistema. La clasificación de una aparición específica de una palabra COBOL viene determinada por el contexto de la cláusula o frase en la que se produce.

Palabras definidas por el usuario con multibyte caracteres

Cuando se utiliza en el contexto de palabras definidas por el usuario, el término *multibyte* hace referencia a tres tipos de palabras.

Los tres tipos de palabras son:

- Palabras formadas por caracteres DBCS, posiblemente combinadas con caracteres de un solo byte
- Palabras formadas por caracteres UTF-8 que se componen de uno o más bytes
- Palabras formadas por caracteres EUC que se componen de uno o más bytes

Estas son las reglas para formar palabras definidas por el usuario con caracteres multibyte :

Caracteres contenidos
Una palabra definida por el usuario puede constar de caracteres de un solo byte y de varios bytes. Si existe un carácter en formato de un solo byte y de varios bytes, sus representaciones de un solo byte y de varios bytes no son equivalentes.

Los caracteres de un solo byte de la palabra definida por el usuario están limitados a los siguientes caracteres:

- Letras latinas en mayúsculas de la A a la Z
- Letras latinas en minúsculas de la a a la z
- dígitos del 0 al 9
- -(guión)
- _ (subrayado)

El guión codificado de un solo byte no puede aparecer como el primer o último carácter en estas palabras.

El subrayado codificado de un solo byte no puede aparecer como el primer carácter de estas palabras.

Letras en mayúsculas y minúsculas
En palabras COBOL, cada carácter codificado en un solo byte en minúsculas "a" a "z" se considera equivalente a su carácter en mayúsculas codificado en un solo byte correspondiente. Las letras en mayúsculas y minúsculas codificadas en varios bytes no son equivalentes.

Rango de valores
Los rangos de valores válidos para los caracteres multibyte dependen de la página de códigos específica que se esté utilizando.

Longitud máxima
30 bytes. El número de caracteres que puede especificar en 30 bytes varía en función de la página de códigos fuente y de los caracteres utilizados en la palabra definida por el usuario.

Continuación
Las palabras formadas con multibyte caracteres no se pueden continuar entre líneas.

Uso de caracteres de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico
Sólo es aplicable cuando la opción de compilador SOSI (Shift-out/shift-in) ficticia está en vigor. Consulte *SSI* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener detalles de la opción de compilador SOSI.

Palabras definidas por el usuario

Una palabra definida por el usuario es una palabra COBOL que debe proporcionar el usuario para satisfacer el formato de una cláusula o sentencia.

Se da soporte a los siguientes conjuntos de palabras definidas por el usuario. La segunda columna indica si se permiten los caracteres de multibyte en palabras de un conjunto determinado.

Palabra definida por el usuario	**¿Se permiten Multibyte caracteres?**
Nombre alfabético	Sí
Nombre-asignación	No
Nombre de clase (de datos)	Sí
Nombre-condición	Sí
Nombre de datos	Sí
Nombre de archivo	Sí
Nombre de índice	Sí
Números de nivel: 01–49, 66, 77, 88	No
Nombre de biblioteca	No
Nombre mnemotécnico	Sí
Nombre de párrafo	Sí
Números de prioridad: 00–99	No
Nombre de programa	No
Nombre-registro	Sí
Nombre-clave-registro **Restricción:** *nombre-clave-registro* sólo está soportado para el sistema de archivos STL.	Sí
Nombre-sección	Sí
Carácter simbólico	Sí
Nombre-texto	No

La longitud máxima de una palabra definida por el usuario es de 30 bytes, excepto para los números de nivel y los números de prioridad. Los números de nivel y los números de prioridad deben ser cada uno de un dígito o un entero de dos dígitos.

Una palabra determinada definida por el usuario sólo puede pertenecer a uno de estos conjuntos, excepto que un número determinado puede ser un número de prioridad y un número de nivel. Cada palabra definida por el usuario dentro de un conjunto debe ser exclusiva, excepto para los números de prioridad y los números de nivel y excepto tal como se especifica en Capítulo 8, “Referencia a nombres de datos, bibliotecas de copia y nombres de PROCEDURE DIVISION”, en la página 57.

Las sentencias y entradas del programa en el que se declara la palabra definida por el usuario pueden hacer referencia a los siguientes tipos de palabras definidas por el usuario:

- Nombre de párrafo
- Nombre-sección

Cualquier programa COBOL puede hacer referencia a los siguientes tipos de palabras definidas por el usuario, siempre que el sistema de compilación dé soporte a la biblioteca asociada u otro sistema y que las entidades referenciadas sean conocidas por dicho sistema:

- Nombre de biblioteca
- Nombre-texto

Las sentencias y entradas del programa que contiene la sección de configuración o cualquier programa contenido en dicho programa pueden hacer referencia a los siguientes tipos de nombres, cuando se declaran dentro de una sección de configuración:

- Nombre alfabético
- Nombre-condición
- Nombre mnemotécnico
- Carácter simbólico

La función de cada palabra definida por el usuario se describe en la cláusula o sentencia en la que aparece.

Referencias relacionadas
Apéndice E, “Palabras sensibles al contexto”, en la página 583

Nombres de sistema

Un *nombre-sistema* es una serie de caracteres que tiene un significado específico para el sistema.

Existen tres tipos de nombres de sistema:

- Nombre de sistema
- Idioma-nombre
- nombre-implementador

Existen tipos de nombres de implementador:

- Nombre de entorno
- Nombre-clase-externa
- Nombre-asignación

El significado de cada nombre de sistema se describe con el formato en el que aparece.

Se permiten series de caracteres de varios bytes para nombres de sistema.

Nombres de función

Un *nombre-función* especifica el mecanismo proporcionado para determinar el valor de una función intrínseca.

La misma palabra, en un contexto diferente, puede aparecer en un programa como una palabra definida por el usuario o un nombre de sistema. Para obtener una lista de nombres de función y sus definiciones, consulte Tabla 59 en la página 458.

Palabras reservadas

Una *palabra reservada* es una serie de caracteres con un significado predefinido en una unidad de origen COBOL.

Las palabras reservadas se listan en Apéndice D, “Palabras reservadas”, en la página 565. Hay cinco tipos de palabras reservadas:

- Palabras clave

- Palabras opcionales
- Constantes figurativas
- Palabras de caracteres especiales
- Registros especiales

Palabras clave
Las palabras clave son palabras reservadas que son necesarias dentro de una cláusula, entrada o sentencia determinada. Dentro de cada formato, estas palabras aparecen en mayúsculas en la ruta principal.

Palabras opcionales
Las palabras opcionales son palabras reservadas que se pueden incluir en el formato de una cláusula, entrada o sentencia para mejorar la legibilidad. No tienen ningún efecto en la ejecución del programa.

Constantes figurativas
Consulte "Constantes figurativas" en la página 15.

Palabras de caracteres especiales
Existen cinco tipos de *palabras de caracteres especiales*, que se reconocen como caracteres especiales sólo cuando se representan en caracteres de un solo byte:

- **Operadores aritméticos:** +-/* **

  Consulte "Expresiones aritméticas" en la página 246.
- **Operadores relacionales**: < > = < = > =

  Consulte "Expresiones condicionales" en la página 250.
- **Indicadores de comentario flotante**: * >

  Consulte "Indicadores de comentario flotante (* >)" en la página 50.
- **Delimitadores de pseudo-texto en sentencias COPY y REPLACE**: ==

  Consulte "sentencia COPY" en la página 508 y "sentencia REPLACE" en la página 520.
- **Indicadores de directiva de compilador**: > >

  Consulte Capítulo 22, "Directivas de compilador", en la página 529.

Registros especiales
Consulte "Registros especiales" en la página 17.

Constantes figurativas

Las *constantes figurativas* son palabras reservadas que nombran y hacen referencia a valores constantes específicos. Las palabras reservadas para las constantes figurativas y sus significados se listan en esta sección.

CERO, CEROS, CEROS
Representa el valor numérico cero (0), una o más apariciones del carácter alfanumérico cero, o el valor booleano B "0", en función del contexto.

Cuando se utiliza la constante figurativa ZERO, ZEROS o ZEROES en un contexto que requiere un carácter alfanumérico, se utiliza un carácter alfanumérico cero. Cuando el contexto requiere un carácter nacional cero, se utiliza un carácter nacional cero (valor NX'3000'). Cuando no se puede determinar el contexto, se utiliza un carácter alfanumérico cero.

ESPACIO, ESPACIOS
Representa uno o más espacios en blanco o espacios. SPACE se trata como un literal alfanumérico cuando se utiliza en un contexto que requiere un carácter alfanumérico, como un literal DBCS cuando se utiliza en un contexto que requiere un carácter DBCS, como un literal nacional cuando se utiliza en un contexto que requiere un carácter nacional.

HIGH-VALUE, HIGH-VALUES
Representa una o más apariciones del carácter que tiene la posición ordinal más alta en el orden de clasificación utilizado.

HIGH-VALUE se trata como un literal alfanumérico en un contexto que requiere un carácter alfanumérico. Para datos alfanuméricos con la secuencia de clasificación EBCDIC, el valor es X'FF '. Para otros datos alfanuméricos, el valor depende de la secuencia secuencia indicada por el entorno local. Para obtener más información sobre los entornos locales, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

HIGH-VALUE se trata como un literal nacional cuando se utiliza en un contexto que requiere un literal nacional. El valor es el carácter nacional NX'FFFF '. HIGH-VALUE se puede utilizar en un contexto que requiere un literal nacional sólo cuando la opción de compilador NCOLLSEQ (BIN) está en vigor.

Cuando no se puede determinar el contexto, se asume un contexto alfanumérico y se utiliza el valor X'FF '.

Nota de uso: No debe utilizar HIGH-VALUE (o un valor asignado desde HIGH-VALUE) de una forma que dé como resultado la conversión entre una representación de datos y otra. X'FF 'no representa un carácter EBCDIC o ASCII válido y NX'FFFF' no representa un carácter nacional válido. La conversión de la representación ALTA-VALUE alfanumérica o nacional a otra representación da como resultado un carácter de sustitución. Por ejemplo, la conversión de X'FF 'a UTF-16 daría un carácter de sustitución, no NX'FFFF'.

LOW-VALUE, LOW-VALUES
Representa una o más apariciones del carácter que tiene la posición ordinal más baja en el orden de clasificación utilizado.

LOW-VALUE se trata como un literal alfanumérico en un contexto que requiere un carácter alfanumérico. Para datos alfanuméricos con la secuencia de clasificación EBCDIC, el valor es X'00 '. Para otros datos alfanuméricos, el valor depende de la secuencia secuencia indicada por el entorno local. Para obtener más información sobre los entornos locales, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

LOW-VALUE se trata como un literal nacional cuando se utiliza en un contexto que requiere un literal nacional. El valor es el carácter nacional NX'0000 '. LOW-VALUE se puede utilizar en un contexto que requiere un literal nacional sólo cuando la opción de compilador NCOLLSEQ (BIN) está en vigor.

Cuando no se puede determinar el contexto, se asume un contexto alfanumérico y se utiliza el valor X'00 '.

QUOTE, QUOTE
Representa una o más apariciones de:

- El carácter de comillas ("), si la opción de compilador QUOTE está en vigor
- El carácter de apóstrofo ('), si la opción de compilador APOST está en vigor

QUOTE o QUOTE representa un carácter alfanumérico cuando se utiliza en un contexto que requiere un carácter alfanumérico, representa un carácter nacional cuando se utiliza en un contexto que requiere un carácter nacional. El valor de carácter nacional de comillas es NX'2200'. El valor de carácter nacional del apóstrofo es NX'2700'.

QUOTE y QUOTE no se pueden utilizar en lugar de una comilla o un apóstrofo para encerrar un literal alfanumérico.

ALL *literal*
literal puede ser un literal alfanumérico, un literal DBCS, un literal nacional, o una constante figurativa distinta del literal ALL.

Cuando *literal* no es una constante figurativa, TODOS los *literales* representan una o más apariciones de la serie de caracteres que componen el literal.

Cuando *literal* es una constante figurativa, la palabra ALL no tiene ningún significado y sólo se utiliza para la legibilidad.

La constante figurativa ALL *literal* no debe utilizarse con las sentencias CALL, INSPECT, STOP o STRING.

carácter-simbólico
Representa uno o más de los caracteres especificados como un valor de *carácter simbólico* en la cláusula SYMBOLIC CHARACTERS del párrafo SPECIAL-NAMES.

carácter-simbólico siempre representa un carácter alfanumérico; se puede utilizar en un contexto que requiera un carácter nacional sólo cuando se define la conversión implícita de caracteres alfanuméricos a nacionales . (Se puede utilizar, por ejemplo, en una sentencia MOVE donde el elemento receptor es de clase nacional porque la conversión implícita se define cuando el elemento emisor es alfanumérico y el elemento receptor es nacional .)

No puede especificar la cláusula SYMBOLIC CHARACTERS si una página de códigos de varios bytes se indica mediante el valor de entorno local en tiempo de compilación. Para obtener más información sobre los entornos locales, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

NULL, NULLS
Representa un valor utilizado para indicar que los elementos de datos definidos con USAGE POINTER, USAGE PROCEDURE-POINTER, USAGE FUNCTION-POINTER, o el registro especial ADDRESS OF no contienen una dirección válida. NULL sólo se puede utilizar cuando se permite explícitamente en los formatos de sintaxis. NULL tiene el valor cero.

Las formas singular y plural de NULL, ZERO, SPACE, HIGH-VALUE, LOW-VALUE y QUOTE se pueden utilizar indistintamente. Por ejemplo, si DATA-NAME-1 es un elemento de datos de cinco caracteres, cada una de las sentencias siguientes mueve cinco espacios a DATA-NAME-1:

```
MOVE SPACE       TO  DATA-NAME-1
MOVE SPACES      TO  DATA-NAME-1
MOVE ALL SPACES  TO  DATA-NAME-1
```

Cuando las reglas de COBOL permiten cualquier ortografía de un nombre de constante figurativa, se puede especificar cualquier ortografía alternativa de ese nombre de constante figurativa.

Puede utilizar una constante figurativa siempre que aparezca *literal* en un diagrama de sintaxis, excepto cuando esté explícitamente prohibido. Cuando aparece un literal numérico en un diagrama de sintaxis, sólo se puede utilizar la constante figurativa CERO (o ZEROS o ZEROES). Las constantes figurativas no se permiten como argumentos de función excepto en una expresión aritmética, donde la expresión es un argumento para una función donde la expresión es un argumento para una función. La constante figurativa ZERO se puede utilizar como un literal booleano.

La longitud de una constante figurativa depende del contexto de su uso. Se aplican las reglas siguientes:

- Cuando se especifica una constante figurativa en una cláusula VALUE o se asocia con un elemento de datos (por ejemplo, cuando se mueve a o se compara con otro elemento), la longitud de la serie de caracteres de constante figurativa es igual a 1 o el número de posiciones de caracteres en el elemento de datos asociado, lo que sea mayor.
- Cuando una constante figurativa, distinta del literal ALL, no está asociada con otro elemento de datos (por ejemplo, en una sentencia CALL, STOP, STRING o UNSTRING), la longitud de la serie de caracteres es de un carácter.

Registros especiales

Los *registros especiales* son palabras reservadas que nombran áreas de almacenamiento generadas por el compilador. Su uso principal es almacenar información producida a través de características COBOL específicas. Cada área de almacenamiento de este tipo tiene un nombre fijo y no debe definirse dentro del programa.

Para programas con el atributo RECURSIVE, para los siguientes registros especiales se asigna por invocación:

- DIRECCIÓN DE
- CÓDIGO-RETORNO
- CONTROL DE SORT
- TAMAÑO-NÚCLEO-CLASIFICACIÓN
- SORT-FILE-SIZE
- ORDENAR-MENSAJE
- TAMAÑO-MODALIDAD-CLASIFICACIÓN
- ORDENAR-RETORNO
- RECUENTO
- CÓDIGO XML
- XML-EVENT
- XML-NTEXT
- XML-TEXTO

Para la primera llamada a un programa después de una cancelación del programa, el compilador inicializa los campos de registro especial a sus valores iniciales.

Para los siguientes dos casos:

- Programas que tienen especificada la cláusula INITIAL
- Programas que tienen especificada la cláusula RECURSIVE

los siguientes registros especiales se restablecen a su valor inicial en cada programa entrada:

- CÓDIGO-RETORNO
- CONTROL DE SORT
- TAMAÑO-NÚCLEO-CLASIFICACIÓN
- SORT-FILE-SIZE
- ORDENAR-MENSAJE
- TAMAÑO-MODALIDAD-CLASIFICACIÓN
- ORDENAR-RETORNO
- RECUENTO
- CÓDIGO XML
- XML-EVENT

Además, en los dos casos anteriores, los valores establecidos en los registros especiales ADDRESS OF persisten sólo para la extensión del programa invocaciónen particular.

En todos los demás casos, los registros especiales no se restablecerán; no se modificarán con respecto al valor contenido en la CALLanterior.

A menos que se restrinja explícitamente de otro modo, se puede utilizar un registro especial siempre que se pueda utilizar un nombre de datos o identificador que tenga la misma definición que la definición implícita del registro especial. Las definiciones implícitas, si procede, se dan en la especificación de cada registro especial.

Puede especificar un registro especial alfanumérico en una función siempre que se permita un argumento alfanumérico en una función, a menos que esté específicamente prohibido.

Si se permite la calificación, los registros especiales pueden ser calificados como necesarios para proporcionar exclusividad. (Para obtener más información, consulte “Cualificación” en la página 57.)

DIRECCIÓN DE

El registro especial ADDRESS OF hace referencia a la dirección de un elemento de datos en LINKAGE SECTION, LOCAL-STORAGE SECTION o WORKING-STORAGE SECTION.

Para los elementos de nivel 01 y 77 en la SECCIÓN DE ENLACE, el registro especial ADDRESS OF puede utilizarse como elemento de envío o elemento de recepción. Para todos los demás operandos, el registro especial ADDRESS OF sólo se puede utilizar como elemento de envío.

El registro especial ADDRESS OF se define implícitamente como USAGE POINTER.

No se permite un identificador de función como operando del registro especial ADDRESS OF.

ELEMENTO DE DEPURACIÓN

El registro especial DEBUG-ITEM proporciona información para un procedimiento declarativo de depuración sobre las condiciones que provocan la ejecución de la sección de depuración.

DEBUG-ITEM tiene la siguiente descripción implícita:

```
01  DEBUG-ITEM.
    02   DEBUG-LINE     PICTURE IS X(6).
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-NAME     PICTURE IS X(30).
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-SUB-1    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-SUB-2    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-SUB-3    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-CONTENTS PICTURE IS X(n).
```

Antes de que se ejecute cada sección de depuración, DEBUG-ITEM se rellena con espacios. El contenido de los subcampos DEBUG-ITEM se actualiza de acuerdo con las reglas de la sentencia MOVE, con una excepción: DEBUG-CONTENTS se actualiza como si el movimiento fuera un movimiento elemental alfanumérico a alfanumérico sin conversión de datos de una forma de representación interna a otra.

Después de actualizar, el contenido de los subcampos DEBUG-ITEM son:

LÍNEA DE DEPURACIÓN
El número de secuencia de sentencia de origen (o el número de secuencia generado por el compilador, en función de la opción de compilador elegida) que ha provocado la ejecución de la sección de depuración.

NOMBRE_DEPURACIÓN
Los primeros 30 caracteres del nombre que han causado la ejecución de la sección de depuración. Los calificadores están separados por la palabra 'OF'.

DEBUG-SUB-1, DEBUG-SUB-2, DEBUG-SUB-3
Establecer siempre en espacios. Estos subcampos se documentan para la compatibilidad con productos COBOL anteriores.

CONTENIDO DE DEPURACIÓN
Los datos se mueven a DEBUG-CONTENTS, tal como se muestra en la tabla siguiente.

Tabla 2. ***Contenido del subcampo DEBUG-ITEM***

Causa de la ejecución de la sección de depuración	**Sentencia a la que se hace referencia en DEBUG-LINE**	**Contenido de DEBUG-NAME**	**Contenido de DEBUG-CONTENTS**
procedure-name-1 Referencia de ALTER	Sentencia ALTER	*procedure-name-1*	*procedure-name-n* en la frase TO CONTINUAR
GO TO *nombre-procedimiento-n*	sentencia GO TO	*nombre-procedimiento-n*	Espacios
procedure-name-n en el procedimiento de entrada/salida SORT o MERGE	sentencia SORT o MERGE	*nombre-procedimiento-n*	"SORT INPUT", "SORT OUTPUT" o "MERGE OUTPUT" (según corresponda)
Transferencia de control de sentencia PERFORM	Esta sentencia PERFORM	*nombre-procedimiento-n*	"REALIZAR BUCLE"
procedure-name-n en un procedimiento USE	Sentencia que provoca la ejecución del procedimiento USE	*nombre-procedimiento-n*	"UTILIZAR PROCEDIMIENTO"
Transferencia implícita desde un procedimiento secuencial anterior	Sentencia anterior ejecutada en el procedimiento secuencial anterior[1]	*nombre-procedimiento-n*	"CAER HASTA"
Primera ejecución del primer procedimiento no declarativo	Número de línea del primer nombre de procedimiento no declarativo	Nombre del primer procedimiento no declarativo	"INICIAR PROGRAMA"

1. Si este procedimiento va precedido de una cabecera de sección y el control se pasa a través de la cabecera de sección, el número de sentencia hace referencia a la cabecera de sección.

FORMATO DE

La frase FORMAT OF de PROCEDURE DIVISION crea un registro especial implícito, denominado registro especial FORMAT OF, cuyo contenido es igual al literal FORMAT del elemento de datos al que hace referencia el identificador.

El registro especial FORMAT OF sólo puede especificarse para elementos de datos de la clase fecha-hora. La longitud de este registro especial depende del literal o entorno local especificado en la frase FORMAT para el elemento de datos.

El registro especial FORMAT OF tiene la definición implícita:

```
USAGE DISPLAY, PICTURE X(n)
where n equals the number of bytes of the implicit or explicit
FORMAT literal.
```

Por ejemplo, considere la siguiente entrada de descripción de datos para el elemento de datos de fecha `date2`:

```
05 date2 FORMAT DATE IS '%d,%m,%y'.
```

La siguiente sentencia MOVE utiliza la función intrínseca CONVERT-DATE-TIME para convertir el elemento de datos de fecha `date3` al formato del elemento de datos de fecha `date2`. La frase FORMAT OF crea un registro especial implícito cuyo contenido sería `%d,%m,%y`.

```
MOVE FUNCTION CONVERT-DATE-TIME(date3, DATE, FORMAT OF date2)
          TO alpha-num-date.
```

La longitud del registro especial en este ejemplo es 8.

Se aplican las reglas siguientes:

- El registro especial FORMAT OF no se puede modificar y sólo se puede especificar en PROCEDURE DIVISION, donde se permite un literal no numérico FORMAT.
- Existe un registro especial FORMAT OF independiente para cada identificador al que se hace referencia con la frase FORMAT OF

LENGTH OF

El registro especial LENGTH OF contiene el número de bytes utilizados por un elemento de datos.

LENGTH OF crea un registro especial implícito que contiene la longitud de byte actual del elemento de datos al que hace referencia el identificador.

Para los elementos de datos descritos con el uso DISPLAY-1 (elementos de datos DBCS) y los elementos de datos descritos con el uso NATIONAL, cada carácter ocupa 2 bytes de almacenamiento.

LENGTH OF se puede utilizar en PROCEDURE DIVISION en cualquier lugar en el que se pueda utilizar un elemento de datos numérico que tenga la misma definición que la definición implícita del registro especial LENGTH OF.

Cuando se especifica la opción de compilador ADDR (32), LENGTH OF tiene la definición implícita:

```
USAGE IS BINARY PICTURE 9(9).
```

Cuando se especifica la opción de compilador ADDR (64), el registro especial LENGTH OF tiene la definición implícita:

```
USAGE IS BINARY PICTURE 9(18).
```

Si el elemento de datos al que hace referencia el identificador contiene la cláusula GLOBAL, el registro especial LENGTH OF es un elemento de datos global.

El registro especial LENGTH OF puede aparecer dentro de la posición del carácter inicial o de las expresiones de longitud de una especificación de modificación de referencia. Sin embargo, el registro especial LENGTH OF no se puede aplicar a ningún operando modificado por referencia.

El operando LENGTH OF no puede ser una función, pero se permite el registro especial LENGTH OF en una función donde se permite un argumento entero.

Si se utiliza el registro especial LENGTH OF como argumento para la función LENGTH, el resultado es siempre 4 cuando se especifica la opción de compilador **ADDR(32)** o siempre 8 cuando se especifica la opción de compilador **ADDR(64)** , independientemente del argumento especificado para LENGTH OF.

Si se utiliza el registro especial ADDRESS OF como argumento para la función LENGTH, el resultado es siempre 4 cuando se especifica la opción de compilador **ADDR(32)** o siempre 8 cuando se especifica la opción de compilador **ADDR(64)** , independientemente del argumento especificado para ADDRESS OF.

LENGTH OF no puede ser uno de los siguientes elementos:

- Un elemento de datos de recepción
- Un subíndice

Cuando se utiliza el registro especial LENGTH OF como parámetro en una sentencia CALL, debe pasarse BY CONTENT o BY VALUE.

Cuando se especifica un elemento de tabla, el registro especial LENGTH OF contiene la longitud en bytes de una aparición. Al hacer referencia a un elemento de tabla, no es necesario que el nombre de elemento esté subíndice.

Se devuelve un valor para cualquier identificador cuya longitud pueda determinarse, incluso si el área referenciada por el identificador no está disponible actualmente para el programa.

Existe un registro especial LENGTH OF separado para cada identificador al que se hace referencia con la frase LENGTH OF. Por ejemplo:

```
MOVE LENGTH OF A TO B
DISPLAY LENGTH OF A, A
ADD LENGTH OF A TO B
CALL "PROGX" USING BY REFERENCE A BY CONTENT LENGTH OF A
```

La función intrínseca LENGTH también se puede utilizar para obtener la longitud de un elemento de datos. Para elementos de datos de uso NATIONAL, la longitud devuelta por la función LENGTH es el número de posiciones de caracteres nacionales, en lugar de bytes; por lo tanto, el registro especial LENGTH OF y la función intrínseca LENGTH tienen resultados diferentes para elementos de datos de uso NATIONAL. Para todos los demás elementos de datos, el resultado es el mismo.

La función intrínseca LENGTH, cuando se aplica a un literal alfanumérico terminado en nulo, devuelve el número de bytes en el literal antes de, pero sin incluir, el nulo de terminación. (El registro especial LENGTH no soporta operandos literales.) Para obtener detalles sobre los literales alfanuméricos terminados en nulo, consulte “Literales alfanuméricos terminados en nulo” en la página 32.

LINAGE-COUNTER

Se genera un registro especial LINAGE-COUNTER independiente para cada entrada FD que contiene una cláusula LINAGE. Cuando se genera más de uno, debe calificar cada referencia a un LINAGE-COUNTER con su nombre de archivo relacionado.

La descripción implícita del registro especial LINAGE-COUNTER se encuentra en uno de los casos siguientes:

- Si la cláusula LINAGE especifica un nombre de datos, LINAGE-COUNTER tiene los mismos PICTURE y USAGE que ese nombre de datos.
- Si la cláusula LINAGE especifica un entero, LINAGE-COUNTER es un elemento binario con el mismo número de dígitos que ese entero.

Para obtener más información, consulte “cláusula LINAGE” en la página 163.

El valor en LINAGE-COUNTER en cualquier momento dado es el número de línea en el que el dispositivo está situado dentro de la página actual. Se puede hacer referencia a LINAGE-COUNTER en las sentencias PROCEDURE DIVISION; no deben modificarlo.

LINAGE-COUNTER se inicializa en 1 cuando se ejecuta una sentencia OPEN para su archivo asociado.

LINAGE-COUNTER se modifica automáticamente mediante cualquier sentencia WRITE para este archivo. (Consulte “sentencia WRITE” en la página 423.)

Si la entrada de descripción de archivo para un archivo secuencial contiene la cláusula LINAGE y la cláusula EXTERNAL, el elemento de datos LINAGE-COUNTER es un elemento de datos externo. Si la entrada de descripción de archivo para un archivo secuencial contiene la cláusula LINAGE y la cláusula GLOBAL, el elemento de datos LINAGE-COUNTER es un elemento de datos global.

Puede especificar el registro especial LINAGE-COUNTER siempre que se permita un argumento entero en una función.

UBICACIÓN DE

El registro especial LOCALE OF devuelve el equivalente de un nombre mnemotécnico de entorno local asociado con el elemento de datos especificado.

Si el elemento de datos no tiene un entorno local asociado, se devuelve la palabra clave COBOL. El registro especial LOCALE OF no se puede modificar y sólo se puede especificar en PROCEDURE DIVISION donde se permite un nombre mnemotécnico de entorno local.

Se puede utilizar un elemento de datos de fecha y hora en expresiones utilizando el registro especial LOCALE OF.

CÓDIGO-RETORNO

El registro especial RETURN-CODE puede utilizarse para pasar un código de retorno al programa o sistema operativo de llamada cuando finaliza el programa COBOL actual.

Cuando un programa COBOL finaliza:

- Si el control vuelve al sistema operativo, el valor del registro especial RETURN-CODE se pasa al sistema operativo como un código de retorno de usuario. Los valores de código de retorno de usuario soportados los determina el sistema operativo y es posible que no incluyan el rango completo de valores de registro especial RETURN-CODE. Para obtener información sobre los valores de código de retorno de usuario en Linux, consulte *Obtención de comentarios de servicios invocables de fecha y hora* en *COBOL for Linux en x86 Guía de programación*.
- Si el control vuelve a un programa de llamada, el valor del registro especial RETURN-CODE se pasa al programa de llamada. Si el programa de llamada es un programa COBOL, el registro especial RETURN-CODE del programa de llamada se establece en el valor del registro especial RETURN-CODE del programa llamado.

El registro especial RETURN-CODE tiene la definición implícita:

```
01  RETURN-CODE GLOBAL PICTURE S9(4) USAGE BINARY VALUE ZERO.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con la cláusula GLOBAL en el programa más externo.

Los ejemplos siguientes muestran cómo establecer el registro especial RETURN-CODE:

- `COMPUTE RETURN-CODE = 8.`
- `MOVE 8 to RETURN-CODE.`

El registro especial RETURN-CODE no devuelve un valor de un programa que utiliza CALL ... DEVOLVIENDO. Para obtener más información, consulte “sentencia CALL” en la página 300.

Puede especificar el registro especial RETURN-CODE en una función siempre que se permita un argumento entero.

El registro especial RETURN-CODE no devuelve información de una llamada de servicio de servicio invocable de fecha/hora. Para obtener más información, consulte *Manipulación de fechas y hora* en la publicación *COBOL for Linux en x86 Guía de programación*.

SHIFT-OUT y SHIFT-IN

Puede especificar los registros especiales SHIFT-OUT y SHIFT-IN en una función siempre que se permita un argumento alfanumérico.

Los registros especiales SHIFT-OUT y SHIFT-IN sólo están soportados cuando se compila con la opción de compilador CHAR (EBCDIC). Sin embargo, sus valores no se reconocen como delimitadores para caracteres de doble byte en las páginas de códigos soportadas para COBOL para Linux.

Los registros especiales SHIFT-OUT y SHIFT-IN se definen implícitamente como elementos de datos alfanuméricos con el formato:

```
01  SHIFT-OUT GLOBAL PICTURE X(1) USAGE DISPLAY VALUE X"0E".
01  SHIFT-IN  GLOBAL PICTURE X(1) USAGE DISPLAY VALUE X"0F".
```

Cuando se utilizan en programas anidados, estos registros especiales se definen implícitamente con el atributo global en el programa más externo.

Estos registros especiales representan caracteres de control de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico EBCDIC, que son caracteres no imprimibles.

Estos registros especiales no pueden ser artículos de recepción. SHIFT-OUT y SHIFT-IN no se pueden utilizar en lugar de los caracteres de control del teclado cuando se definen palabras definidas por el usuario de multibyte o se especifican literales DBCS EBCDIC.

CONTROL DE SORT

El registro especial SORT-CONTROL es el nombre de un elemento de datos alfanumérico.

El registro especial SORT-CONTROL tiene la definición implícita:

```
01 SORT-CONTROL GLOBAL PICTURE X(160) VALUE "file name".
```

donde "nombre de archivo" es el nombre de archivo utilizado por la clasificación como fuente para opciones adicionales de clasificación/fusión.

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

Puede especificar el registro especial SORT-CONTROL en una función siempre que se permita un argumento alfanumérico.

El registro especial SORT-CONTROL no es necesario para una operación de clasificación o fusión satisfactoria.

El archivo de control de ordenación tiene prioridad sobre los registros especiales SORT.

TAMAÑO-NÚCLEO-CLASIFICACIÓN

El registro especial SORT-CORE-SIZE es el nombre de un elemento de datos binario que puede utilizar para especificar el número de bytes de almacenamiento disponibles para el programa de utilidad de clasificación.

El registro especial SORT-CORE-SIZE tiene la definición implícita:

```
01  SORT-CORE-SIZE GLOBAL PICTURE S9(8) USAGE BINARY VALUE ZERO.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

La cantidad de almacenamiento indicada en el registro especial SORT-CORE-SIZE no incluye las áreas de memoria necesarias para las funciones de biblioteca COBOL no relacionadas con la función SORT o MERGE. Tampoco incluye la cantidad fija de áreas de memoria (módulos, bloques de control, áreas de trabajo de tamaño fijo) necesarias para la implementación de clasificación y fusión.

Puede especificar el registro especial SORT-CORE-SIZE en una función siempre que se permita un argumento entero.

SORT-FILE-SIZE

El registro especial SORT-FILE-SIZE es el nombre de un elemento de datos binarios que puede utilizar para especificar el número estimado de registros en el archivo de entrada de clasificación, *file-name-1*.

El registro especial SORT-FILE-SIZE tiene la definición implícita:

```
01  SORT-FILE-SIZE GLOBAL PICTURE S9(8) USAGE BINARY VALUE ZERO.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

El compilador resuelve las referencias al registro especial SORT-FILE-SIZE; sin embargo, el valor del registro especial no tiene ningún efecto en la ejecución de una sentencia SORT o MERGE.

Puede especificar el registro especial SORT-FILE-SIZE en una función siempre que se permita un argumento entero.

ORDENAR-MENSAJE

El registro especial SORT-MESSAGE es el nombre de un elemento de datos alfanumérico que está disponible para los programas de clasificación y fusión.

El compilador resuelve las referencias al registro especial SORT-MESSAGE; sin embargo, el valor del registro especial no tiene ningún efecto en la ejecución de una sentencia SORT o MERGE.

El registro especial SORT-MESSAGE tiene la definición implícita:

```
01  SORT-MESSAGE GLOBAL PICTURE X(8) USAGE DISPLAY VALUE "SYSOUT".
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

Puede especificar el registro especial SORT-MESSAGE en una función siempre que se permita un argumento alfanumérico.

TAMAÑO-MODALIDAD-CLASIFICACIÓN

El registro especial SORT-MODE-SIZE es el nombre de un elemento de datos binarios que puede utilizar para especificar la longitud de los registros de longitud variable que se producen con más frecuencia.

El registro especial SORT-MODE-SIZE tiene la definición implícita:

```
01  SORT-MODE-SIZE GLOBAL PICTURE S9(5) USAGE BINARY VALUE ZERO.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

El compilador resuelve las referencias al registro especial SORT-MODE-SIZE; sin embargo, el valor del registro especial no tiene ningún efecto en la ejecución de una sentencia SORT o MERGE.

Puede especificar el registro especial SORT-MODE-SIZE en una función siempre que se permita un argumento entero.

ORDENAR-RETORNO

El registro especial SORT-RETURN es el nombre de un elemento de datos binario y está disponible para los programas de clasificación y fusión.

El registro especial SORT-RETURN tiene la definición implícita:

```
01  SORT-RETURN GLOBAL PICTURE S9(4) USAGE BINARY VALUE ZERO.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

El registro especial SORT-RETURN contiene un código de retorno de 0 (satisfactorio) o 16 (no satisfactorio) al finalizar una operación de clasificación o fusión. Si la clasificación o fusión no es

satisfactoria y no hay ninguna referencia a este registro especial en ninguna parte del programa, se visualiza un mensaje en el terminal.

Puede establecer el registro especial SORT-RETURN en 16 en un procedimiento declarativo de error o de entrada/salida para terminar una operación de clasificación o fusión antes de que se procesen todos los registros. La operación termina en la siguiente función de entrada o salida para la operación de clasificación o fusión.

Puede especificar el registro especial SORT-RETURN en una función siempre que se permita un argumento entero.

RECUENTO

El registro especial ALLY es el nombre de un elemento de datos binarios.

Consulte la definición siguiente de un elemento de datos binarios:

```
01  TALLY GLOBAL PICTURE 9(5) USAGE BINARY VALUE ZERO.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

Puede hacer referencia o modificar el contenido de RECUENTO.

Puede especificar el registro especial CONTABILIDAD en una función siempre que se permita un argumento entero.

WHEN-COMPILADO

El registro especial WHEN-COMPILE contiene la fecha al principio de la compilación.

WHEN-COMPILE es un elemento de datos alfanumérico que tiene la definición implícita:

```
01  WHEN-COMPILED GLOBAL PICTURE X(16) USAGE DISPLAY.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

El registro especial WHEN-COMPILE tiene el formato:

```
MM/DD/YYhh.mm.ss (MONTH/DAY/YEARhour.minute.second)
```

Por ejemplo, si la compilación empezara a las 14:04 horas del 15 de octubre de 2007, WHEN-COMPILADO contendría el valor 10 /15/0714.04.00.

WHEN-COMPILE sólo se puede utilizar como campo de envío en una sentencia MOVE.

Los datos de registro especial WHEN-COMPILE no se pueden modificar por referencia.

También se puede acceder a la fecha y hora de compilación con la función intrínseca WHEN-COMPILE (consulte “WHEN-COMPILADO” en la página 499). Esta función admite valores de año de cuatro dígitos y proporciona información adicional.

CÓDIGO XML

El registro especial XML-CODE se utiliza para comunicar el estado entre el analizador XML y el procedimiento de proceso que se ha identificado en una sentencia XML PARSE, y para indicar que una sentencia XML GENERATE se ha ejecutado correctamente o que se ha producido una excepción durante la generación de XML.

El registro especial XML-CODE tiene la definición implícita:

```
01  XML-CODE PICTURE S9(9) USAGE BINARY VALUE 0.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

Cuando el analizador XML encuentra un suceso XML, establece XML-CODE y, a continuación, pasa el control al procedimiento de proceso. Para todos los sucesos excepto un suceso EXCEPTION, XML-CODE contiene cero cuando el procedimiento de proceso recibe el control.

Para un suceso EXCEPTION, el analizador establece XML-CODE en un código de excepción que indica la naturaleza de la excepción. Los códigos de excepción XML PARSE se describen en *Manejo de excepciones XML PARSE* en la publicación *COBOL for Linux en x86 Guía de programación*.

Puede establecer XML-CODE antes de volver al analizador, como se indica a continuación:

- A -1, después de un suceso normal, para indicar que el analizador debe terminar inmediatamente sin procesar ningún texto de documento XML restante, y sin provocar un suceso EXCEPTION.
- A 0, después de un suceso EXCEPTION para el que se permite la continuación, para indicar que el analizador va a continuar el proceso. El analizador intenta continuar procesando el documento XML, pero los resultados no están definidos.
- Puede establecer XML-CODE en un identificador de página de códigos después de una excepción de conflicto de codificación en algunos casos. Consulte *Manejo de excepciones XML PARSE* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener más detalles.

Si establece XML-CODE en cualquier otro valor antes de volver al analizador, los resultados no están definidos.

Cuando el analizador devuelve el control a la sentencia XML PARSE, XML-CODE contiene el valor más reciente establecido por el procedimiento de proceso o el analizador. En algunos casos, el analizador altera temporalmente el valor establecido por el procedimiento de proceso.

Al finalizar una sentencia XML GENERATE, XML-CODE contiene cero, lo que indica una finalización satisfactoria de la generación XML, o un código de error distinto de cero, lo que indica que se ha producido una excepción durante la generación XML. Los códigos de excepción XML GENERATE se detallan en *Excepciones XML GENERATE* en *COBOL for Linux en x86 Guía de programación*.

Conceptos relacionados
XML-CODE (*COBOL for Linux en x86 Guía de programación*)

Tareas relacionadas
Manejo de excepciones XML PARSE (*COBOL for Linux en x86 Guía de programación*)

Referencias relacionadas
Excepciones XML GENERATE (*COBOL for Linux en x86 Guía de programación*)

XML-EVENT

El registro especial XML-EVENT se utiliza para comunicar información de sucesos del analizador XML al procedimiento de proceso que se ha identificado en la sentencia XML PARSE.

Antes de pasar el control al procedimiento de proceso, el analizador XML establece el registro especial XML-EVENT en el nombre del suceso XML, tal como se describe en Tabla 3 en la página 28.

XML-EVENT tiene la definición implícita:

```
01  XML-EVENT USAGE DISPLAY PICTURE X(30) VALUE SPACE.
```

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

El contenido de XML-EVENT se codifica en la página de códigos EBCDIC o ASCII en vigor en tiempo de ejecución, en función del valor de la opción de compilador CHAR (EBCDIC, NATIVE o S390).

XML-EVENT no se puede utilizar como elemento de datos receptor.

Tabla 3. ***Contenido de los registros especiales XML-EVENT y XML-TEXT o XML-NTEXT***

Suceso XML (contenido de XML-EVENT)	**Contenido de XML-TEXT o XML-NTEXT**
CARÁCTER DE ATRIBUTO	El carácter único que se corresponde con la referencia de entidad predefinida en el valor de atributo
ATRIBUTO-CARACTERES	El valor entre comillas o apóstrofos. Puede ser una subserie del valor de atributo si el valor incluye una referencia de entidad.
ATRIBUTO-NAME	El nombre de atributo; la serie a la izquierda del signo igual
ATRIBUTO-CARÁCTER NACIONAL	Independientemente del tipo de documento XML especificado por *identifier-1* en la sentencia XML PARSE, XML-TEXT está vacío con la longitud cero y XML-NTEXT contiene el único carácter nacional que corresponde con la referencia de carácter numérico.
COMENTARIO	El texto del comentario entre la secuencia de caracteres de apertura "<! --" y la secuencia de caracteres de cierre "-- >"
CARÁCTER-CONTENIDO	El carácter único que se corresponde con la referencia de entidad predefinida en el contenido del elemento
CARACTERES DE CONTENIDO	El contenido de caracteres del elemento entre las etiquetas de inicio y de finalización. Puede ser una subserie del contenido de caracteres si el contenido incluye una referencia de entidad u otro elemento.
CONTENIDO-CARÁCTER NACIONAL	Independientemente del tipo de documento XML especificado por *identifier-1* en la sentencia XML PARSE, XML-TEXT está vacío con longitud cero y XML-NTEXT contiene el único carácter nacional que corresponde con la referencia de carácter numérico.[1]
DOCUMENTO-TIPO-DECLARACIÓN	Toda la declaración de tipo de documento, incluidas las secuencias de caracteres de apertura y cierre " <!DOCTYPE "y ">"
DECLARACIÓN DE CODIFICACIÓN	El valor, entre comillas o apóstrofos, de la declaración de codificación en la declaración XML
END-OF-CDATA-SECCIÓN	La serie "]] >"
FIN-DE-DOCUMENTO	Vacío con longitud cero
END-OF-ELEMENT	El nombre del código de elemento final o el código de elemento vacío
EXCEPCIÓN	La parte del documento que se ha explorado correctamente, hasta el punto en el que se ha detectado la excepción inclusive.[2] El registro especial XML-CODE contiene el código de error exclusivo que identifica la excepción.
PROCESANDO-INSTRUCCIÓN-DATOS	El resto de la instrucción de proceso (después del nombre de destino), sin incluir la secuencia de cierre, "?>", pero incluyendo los caracteres de espacio en blanco finales y no iniciales

Tabla 3. ***Contenido de los registros especiales XML-EVENT y XML-TEXT o XML-NTEXT*** *(continuación)*

Suceso XML (contenido de XML-EVENT)	**Contenido de XML-TEXT o XML-NTEXT**
PROCESO-INSTRUCCIÓN-DESTINO	El nombre de destino de instrucción de proceso, que se produce inmediatamente después de la secuencia de apertura de instrucción de proceso, "<?"
AUTÓNOMO-DECLARACIÓN	El valor, entre comillas o apóstrofos ("yes" o "no"), de la declaración autónoma en la declaración XML
START-OF-CDATA-SECCIÓN	La serie " <![CDATA ["
INICIO-DE-DOCUMENTO	Todo el documento
INICIO-DE-ELEMENTO	El nombre del código de elemento de inicio o el código de elemento vacío, también conocido como tipo de elemento
DESCONOCIDO-REFERENCIA-EN-CONTENIDO	El nombre de referencia de entidad, sin incluir "&" y delimitadores ";"
DESCONOCIDO-REFERENCIA-EN-ATRIBUTO	El nombre de referencia de entidad, sin incluir "&" y delimitadores ";"
VERSIÓN-INFORMACIÓN	El valor, entre comillas o apóstrofos, de la información de versión de la declaración XML

1. Caracteres nacionales con valores escalares mayores que 65.535 (NX "FFFF") se representan utilizando dos unidades de codificación (un "par suplente"). Los programadores son responsables de asegurarse de que las operaciones en el contenido de XML-NTEXT no dividen el par de unidades de codificación que juntas forman un carácter gráfico, formando así datos no válidos.
2. Las excepciones para los conflictos de codificación se señalan antes de que empiece el análisis. Para estas excepciones, XML-TEXT o XML-NTEXT tiene una longitud cero o sólo contiene el valor de declaración de codificación del documento. Consulte *Excepciones de XML GENERATE* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener información sobre los códigos de excepción XML.

XML-NTEXT

El registro especial XML-NTEXT se define durante el análisis XML para contener fragmentos de documento representados en el uso NATIONAL.

XML-NTEXT es un elemento de datos elemental de categoría nacional de la longitud del fragmento de documento XML contenido. La longitud de XML-NTEXT puede variar de 0 a 2.000.000 de *posiciones de caracteres nacionales*. La longitud máxima de bytes es 4.000.000.

No hay ninguna entrada de descripción de datos COBOL equivalente.

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

El analizador establece XML-NTEXT en el fragmento de documento asociado con un suceso antes de transferir el control al procedimiento de proceso en estos casos:

- Cuando el operando de la sentencia XML PARSE es un elemento de datos de la categoría nacional
- Para el suceso ATTRIBUTE-NATIONAL-CHARACTER
- Para el evento CONTENT-NATIONAL-CHARACTER

Cuando se establece XML-NTEXT, el registro especial XML-TEXT tiene una longitud de cero. En un momento determinado, sólo uno de los dos registros especiales XML-NTEXT y XML-TEXT tiene una longitud distinta de cero.

Utilice la función LENGTH para determinar el número de caracteres nacionales que contiene XML-NTEXT. Utilice el registro especial LENGTH OF para determinar el número de bytes, en lugar del número de caracteres nacionales, que contiene XML-NTEXT.

XML-NTEXT no se puede utilizar como elemento receptor.

XML-TEXTO

El registro especial XML-TEXT se define durante el análisis XML para contener fragmentos de documento representados en DISPLAY de uso.

XML-TEXT es un elemento de datos elemental de categoría alfanumérica de la longitud del fragmento de documento XML contenido. La longitud de XML-TEXT puede variar de 0 a 2,147,483,646 bytes.

No hay ninguna entrada de descripción de datos COBOL equivalente.

Cuando se utiliza en programas anidados, este registro especial se define implícitamente con el atributo global en el programa más externo.

El contenido de XML-TEXT tiene la codificación del documento XML de origen: ASCII o UTF-8 si la opción de compilador CHAR (NATIVE) está en vigor; EBCDIC si la opción de compilador CHAR (EBCDIC) está en vigor.

El analizador establece XML-TEXT en el fragmento de documento asociado con un suceso antes de transferir el control al procedimiento de proceso cuando el operando de la sentencia XML PARSE es un elemento de datos alfanumérico, excepto para el suceso ATTRIBUTE-NATIONAL-CHARACTER y el suceso CONTENT-NATIONAL-CHARACTER.

Cuando se establece XML-TEXT, el registro especial XML-NTEXT tiene una longitud de cero. En un momento determinado, sólo uno de los dos registros especiales XML-NTEXT y XML-TEXT tiene una longitud distinta de cero.

Utilice la función LENGTH o el registro especial LENGTH OF para XML-TEXT para determinar el número de bytes que contiene XML-TEXT.

XML-TEXT no se puede utilizar como elemento receptor.

Literales

Un *literal* es una serie de caracteres cuyo valor se especifica mediante los caracteres de los que se compone o mediante el uso de una constante figurativa.

Para obtener más información sobre las constantes figurativas, consulte “Constantes figurativas” en la página 15.

Para obtener descripciones de los distintos tipos de literales, consulte los temas siguientes:

- “Literales alfanuméricos” en la página 30
- “Literales booleanos” en la página 33
- “Literales DBCS” en la página 33
- “Literales nacionales” en la página 35
- “Literales numéricos” en la página 34

Literales alfanuméricos

COBOL para Linux proporciona varios formatos de literales alfanuméricos.

Los formatos de literales alfanuméricos son:

- Formato 1: “Literales alfanuméricos básicos” en la página 31
- Formato 2: “Notación hexadecimal para literales alfanuméricos” en la página 31
- Formato 3: “Literales alfanuméricos terminados en nulo” en la página 32

Literales alfanuméricos básicos

Los literales alfanuméricos básicos pueden contener sólo caracteres de un solo byte o de varios bytes.

El formato siguiente es para un literal alfanumérico básico:

Formato 1: Literales alfanuméricos básicos

```
"single-byte or multibyte characters"
'single-byte or multibyte characters'
```

Las comillas o apóstrofos delimitadores se excluyen del literal cuando se compila el programa.

Las comillas o apóstrofos incluidos deben estar representados por un par de comillas (" ") o un par de apóstrofos (' '), respectivamente, cuando es el carácter utilizado como delimitador de apertura. Por ejemplo:

```
"THIS ISN""T WRONG"
'THIS ISN''T WRONG'
```

El carácter delimitador utilizado como delimitador de apertura para un literal debe utilizarse como delimitador de cierre para dicho literal. Por ejemplo:

```
'THIS IS RIGHT'
"THIS IS RIGHT"
'THIS IS WRONG"
```

Puede utilizar apóstrofos o comillas como delimitadores literales independientes de la opción de compilador APOST/QUOTE.

Los caracteres de puntuación incluidos en un literal alfanumérico forman parte del valor del literal.

"*caracteres de un solo byte o de varios bytes*" puede ser cualquier carácter representado en la página de códigos indicada por el entorno local en vigor en el momento de la compilación.

La longitud máxima de un literal alfanumérico es de 160 bytes. La longitud mínima es de 1 byte.

Los literales alfanuméricos están en la categoría y clase de datos alfanuméricos. (Las clases y categorías de datos se describen en “Clases y categorías de datos” en la página 146.)

Consulte *SSI* en la publicación *COBOL for Linux en x86 Guía de programación* que describe el tratamiento especial de los valores X'1E'y X'1F' en literales alfanuméricos y restricciones de programación adicionales para literales alfanuméricos que contienen estos caracteres cuando la opción de compilador SOSI está en vigor.

Nota de uso: Utilice la notación hexadecimal para expresar los caracteres de control X'00 'a X'1F' dentro de un literal alfanumérico. Los resultados son imprevisibles si especifica estos caracteres de control en un literal alfanumérico básico.

Notación hexadecimal para literales alfanuméricos

La notación hexadecimal se puede utilizar para literales alfanuméricos.

La notación hexadecimal tiene el formato siguiente:

Formato 2: Notación hexadecimal para literales alfanuméricos

```
X"hexadecimal-digits"
X'hexadecimal-digits'
```

X" o X'
Delimitador de apertura para la notación hexadecimal de un literal alfanumérico.

" o '
El delimitador de cierre para la notación hexadecimal de un literal alfanumérico. Si se utiliza una comilla en el delimitador de apertura, se debe utilizar una comilla como delimitador de cierre. De forma similar, si se utiliza un apóstrofo en el delimitador de apertura, se debe utilizar un apóstrofo como delimitador de cierre.

Los dígitos hexadecimales son caracteres en el rango de '0' a '9', 'a' a 'f' y 'A' a 'F', ambos inclusive. Dos dígitos hexadecimales representan un carácter de un juego de caracteres de un solo byte (EBCDIC o ASCII). Cuatro dígitos hexadecimales representan un carácter en un juego de caracteres DBCS. Para UTF-8 o EUC, el número de dígitos hexadecimales que representan un carácter depende de la longitud de byte del carácter. Se debe especificar un número par de dígitos hexadecimales. La longitud máxima de un literal hexadecimal es de 320 dígitos hexadecimales.

Las reglas de continuación son las mismas que para cualquier literal alfanumérico. El delimitador de apertura (X" o X') no se puede dividir entre líneas.

Un literal alfanumérico en notación hexadecimal tiene clase de datos y categoría alfanumérica. El compilador convierte la notación hexadecimal en los caracteres normales de un literal alfanumérico. La notación hexadecimal para literales alfanuméricos se puede utilizar en cualquier lugar donde se puedan utilizar literales alfanuméricos.

Sin embargo, los literales alfanuméricos en notación hexadecimal se interpretan como EBCDIC si la opción de compilador CHAR (EBCDIC) está en vigor, mientras que los literales alfanuméricos básicos siempre se interpretan en la página de códigos (basada en ASCI) del entorno local actual. Consulte *CAR* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener más información sobre la opción de compilador CHAR (EBCDIC).

Nota de uso: Utilice la notación hexadecimal para expresar los caracteres de control X'00 'a X'1F' dentro de un literal alfanumérico. Los resultados son imprevisibles si especifica estos caracteres de control en un literal alfanumérico básico.

Consulte también “Notación hexadecimal para literales nacionales” en la página 36.

Literales alfanuméricos terminados en nulo

Los literales alfanuméricos pueden terminar en nulo.

El formato de los literales alfanuméricos terminados en nulo es:

Formato 3: literales alfanuméricos terminados en nulo

```
Z"mixed-characters"
Z'mixed-characters'
```

Z" o Z'
El delimitador de apertura para un literal alfanumérico terminado en nulo. Ambos caracteres del delimitador de apertura (Z" o Z') deben estar en la misma línea de origen.

" o '
El delimitador de cierre para un literal alfanumérico terminado en nulo.

Si se utiliza una comilla en el delimitador de apertura, se debe utilizar una comilla como delimitador de cierre. De forma similar, si se utiliza un apóstrofo en el delimitador de apertura, se debe utilizar un apóstrofo como delimitador de cierre.

caracteres mixtos
Puede ser cualquiera de los caracteres siguientes:

- Sólo caracteres de un solo byte

- Caracteres mixtos de un solo byte y multibyte
- Sólo caracteres de multibyte
- Caracteres UTF-8
- Caracteres EUC
- Caracteres DBCS

  Sin embargo, no puede especificar el carácter de un solo byte con el valor X'00 '. X'00 ' es el carácter nulo que se añade automáticamente al final del literal. De lo contrario, el contenido del literal está sujeto a las mismas reglas y restricciones que un literal alfanumérico con multibyte caracteres (formato 1).

La longitud de la serie de caracteres en el contenido literal puede ser de 0 a 159 bytes. La longitud real del literal incluye el carácter nulo de terminación y tiene un máximo de 160 bytes.

Un literal alfanumérico terminado en nulo tiene una clase de datos y una categoría alfanuméricas. Se puede utilizar en cualquier lugar donde se pueda utilizar un literal alfanumérico, excepto que los literales terminados en nulo no están soportados en TODAS las constantes figurativas *literales* .

La función intrínseca LENGTH, cuando se aplica a un literal alfanumérico terminado en nulo, devuelve el número de bytes en el literal antes de, pero sin incluir, el nulo de terminación. (El registro especial LENGTH no soporta operandos literales.)

Literales booleanos

Un *literal booleano* es una serie de caracteres delimitada a la izquierda por el separador *B "* y a la derecha por el separador de comillas. La serie de caracteres sólo consta del carácter 0 o 1. El valor de un literal booleano es el propio carácter, excluyendo los separadores delimitadores.

Literales DBCS

Los formatos y reglas para los literales DBCS se listan en esta sección.

Formato para literales DBCS

```
G"DBCS-characters"
G'DBCS-characters'
N"DBCS-characters"
N'DBCS-characters'
```

G", G', N", o N'
Delimitadores de apertura.

N" y N' identifican un literal DBCS cuando la opción de compilador NSYMBOL (DBCS) está en vigor. Identifican un literal nacional cuando la opción de compilador NSYMBOL (NATIONAL) está en vigor y se aplican las reglas especificadas en “Literales nacionales” en la página 35 .

" o '
El delimitador de cierre. Si se utiliza una comilla en el delimitador de apertura, se debe utilizar una comilla como delimitador de cierre. De forma similar, si se utiliza un apóstrofo en el delimitador de apertura, se debe utilizar un apóstrofo como delimitador de cierre.

Caracteres DBCS
Los Cualquier carácter DBCS.

Longitud máxima
La longitud máxima está limitada por el espacio disponible en una línea fuente.

Reglas de continuación
No se puede continuar entre líneas

Literales DBCS con la opción de compilador SOSI

Cuando la opción de compilador SOSI está en vigor, los caracteres de control de desplazamiento a teclado ideográfico (SO) y desplazamiento a teclado estándar (SI) de la estación de trabajo delimitan los caracteres DBCS en el texto de origen. La siguiente sección es para literales DBCS con delimitadores de desplazamiento a teclado ideográfico y de desplazamiento a teclado estándar:

Formato para literales DBCS

```
G"<DBCS-characters>"
G'<DBCS-characters>'
N"<DBCS-characters>"
N'<DBCS-characters>'
```

<
Representa el carácter de control de desplazamiento a teclado ideográfico (X'1E')

>
Representa el carácter de control de desplazamiento a teclado estándar (X'1F')

Las reglas para caracteres DBCS, delimitadores literales, longitud máxima y continuación son las mismas que para los literales DBCS sin la opción de compilador SOSI. Consulte *SSI* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener detalles de la opción de compilador SOSI.

Donde se pueden utilizar literales DBCS

Los literales DBCS se pueden utilizar en los siguientes lugares:

- DIVISIÓN DE DATOS
  - En la cláusula VALUE de las entradas de descripción de datos que definen un elemento de datos de clase DBCS.
  - En la cláusula VALUE OF de las entradas de descripción de archivo.
- DIVISIÓN DE PROCEDIMIENTO
  - En una condición de relación cuando el comparand es un elemento de datos DBCS, un elemento de datos elemental de clase nacional, un elemento de grupo nacional o un elemento de grupo alfanumérico
  - Como argumento pasado BY CONTENT en una sentencia CALL
  - En las sentencias DISPLAY y EVALUATE
  - En las sentencias siguientes:
    - INITIALIZE; para obtener detalles, consulte “Sentencia INITIALIZE” en la página 329.
    - INSPECT; para obtener detalles, consulte “sentencia INSPECT” en la página 333.
    - MOVE; para obtener detalles, consulte “sentencia MOVE” en la página 347.
    - STRING; para obtener detalles, consulte “sentencia STRING” en la página 409.
    - UNSTRING, para obtener detalles, consulte “Sentencia UNSTRING” en la página 416.
  - En constante figurativa ALL
  - Como argumento para la función intrínseca NATIONAL-OF
- Sentencias de dirección de compilador COPY, REPLACE y TITLE

Literales numéricos

Un *literal numérico* es una serie de caracteres cuyos caracteres se seleccionan entre los dígitos del 0 al 9, un carácter de signo (+ o-) y la coma decimal.

Si el literal no contiene ninguna coma decimal, es un entero. (En esta documentación, la palabra *entero* que aparece en un formato representa un literal numérico de valor distinto de cero que no contiene ningún signo ni ninguna coma decimal, excepto cuando se incluyen otras reglas con la descripción del formato.) Se aplican las reglas siguientes:

- Si la opción de compilador ARITH (COMPAT) está en vigor, se permiten de uno a 18 dígitos. Si la opción de compilador ARITH (EXTEND) está en vigor, se permiten de uno a 31 dígitos.
- Sólo se permite un carácter de signo. Si se incluye, debe ser el carácter situado más a la izquierda del literal. Si el literal no está firmado, es un valor positivo.
- Sólo se permite una coma decimal. Si se incluye una coma decimal, se trata como una coma decimal asumida (es decir, como si no tomara una posición de carácter en el literal). La coma decimal puede aparecer en cualquier lugar dentro del literal excepto como el carácter situado más a la derecha.

El valor de un literal numérico es la cantidad algebraica expresada por los caracteres del literal. El tamaño de un literal numérico es igual al número de dígitos especificado por el usuario.

Los literales numéricos pueden ser números de coma fija o coma flotante.

Los literales numéricos están en la clase y categoría de datos numéricos. (Las clases y categorías de datos se describen en "Clases y categorías de datos" en la página 146.)

Reglas para valores literales de coma flotante

El formato y las reglas para literales de coma flotante se listan a continuación.

Formato

- El signo es opcional antes de la mantisa y el exponente; si omite el signo, el compilador asume un número positivo.
- La mantissa puede contener entre uno y 16 dígitos. Debe incluirse una coma decimal en la mantisa.
- El exponente se representa mediante una E seguida de un signo opcional y uno o dos dígitos.
- La magnitud de un valor literal de coma flotante debe estar entre:
  - Representación de 32 bits: 1,175 (10^{-38}) a 3,403 (10^{38})

Literales nacionales

Los formatos literales nacionales que proporciona COBOL para Linux son literales nacionales básicos y notación hexadecimal para literales nacionales.

Para obtener más información sobre los formatos, consulte "Literales nacionales básicos" en la página 35 y "Notación hexadecimal para literales nacionales" en la página 36.

Literales nacionales básicos

En esta sección se enumeran el formato y las reglas de los literales nacionales básicos.

Formato 1: Literales nacionales básicos

```
N"character-data"
N'character-data'
```

Cuando la opción de compilador NSYMBOL (NATIONAL) está en vigor, el delimitador de apertura N" o N' identifica un literal nacional. Un literal nacional es de la clase y categoría nacional.

Cuando la opción de compilador NSYMBOL (DBCS) está en vigor, el delimitador de apertura N" o N' identifica un literal DBCS y se aplican las reglas especificadas en "Literales DBCS" en la página 33 .

N" o N'
Delimitadores de apertura. El delimitador de apertura debe codificarse como caracteres de un solo byte. No se puede dividir entre líneas.

" o '
El delimitador de cierre. El delimitador de cierre debe codificarse como un carácter de un solo byte. Si se utiliza una comilla en el delimitador de apertura, debe utilizarse como delimitador de cierre. De forma similar, si se utiliza un apóstrofo en el delimitador de apertura, debe utilizarse como delimitador de cierre.

Para incluir las comillas o apóstrofos utilizados en el delimitador de apertura en el contenido del literal, especifique un par de comillas o apóstrofos, respectivamente. Ejemplos:

```
N'This literal''s content includes an apostrophe'
N'This literal includes ", which is not used in the opening delimiter'
N"This literal includes "", which is used in the opening delimiter"
```

datos-caracteres
La representación de texto fuente del contenido del literal nacional. Los *datos de caracteres* pueden incluir cualquier combinación de caracteres de un solo byte y de varios bytes representados en la página de códigos en vigor para el código fuente.

Los caracteres DBCS en el contenido del literal pueden estar delimitados por caracteres de control de desplazamiento a teclado ideográfico y de desplazamiento a teclado estándar de la estación de trabajo, tal como se describe para la opción de compilador SOSI. Consulte *SSI* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener detalles de la opción de compilador SOSI.

Longitud máxima
La longitud máxima de un literal nacional es de 80 caracteres, excluyendo los delimitadores de apertura y cierre. Si el contenido de origen del literal contiene uno o más caracteres de multibyte , la longitud máxima está limitada por el espacio disponible en el Área B de una sola línea de origen.

El literal debe contener al menos un carácter. Cada carácter de un solo byte del literal cuenta como una posición de carácter y cada carácter de multibyte del literal cuenta como una posición de carácter. No se cuentan los delimitadores de desplazamiento a teclado ideográfico de estación de trabajo ni los delimitadores de desplazamiento a teclado ideográfico para caracteres DBCS.

Reglas de continuación
Cuando el contenido del literal incluye caracteres de multibyte , el literal no puede continuar. Cuando el contenido del literal no incluye caracteres de multibyte , se aplican las reglas de continuación normales.

La representación de texto de origen de *datos-caracteres* se convierte automáticamente a UTF-16 para su uso en tiempo de ejecución (por ejemplo, cuando el literal se mueve a o se compara con un elemento de datos de categoría nacional).

Notación hexadecimal para literales nacionales

El formato y las reglas para el formato de notación hexadecimal de literales nacionales se listan en esta sección.

Formato 2: notación hexadecimal para literales nacionales

```
NX"hexadecimal-digits"
NX'hexadecimal-digits'
```

El formato de notación hexadecimal de literales nacionales no se ve afectado por la opción de compilador NSYMBOL.

NX" o NX'
Delimitadores de apertura. El delimitador de apertura debe estar representado en caracteres de un solo byte. No debe dividirse entre líneas.

" o '
El delimitador de cierre. El delimitador de cierre debe representarse como un carácter de un solo byte.

Si se utiliza una comilla en el delimitador de apertura, se debe utilizar una comilla como delimitador de cierre. De forma similar, si se utiliza un apóstrofo en el delimitador de apertura, se debe utilizar un apóstrofo como delimitador de cierre.

dígitos hexadecimales
Dígitos hexadecimales en el rango de '0' a '9', 'a'-f 'y' A 'a' F ', ambos incluidos. Cada grupo de cuatro dígitos hexadecimales representa un único carácter nacional y debe representar un punto de código válido en UTF-16. El número de dígitos hexadecimales debe ser un múltiplo de cuatro.

Longitud máxima
La longitud de un literal nacional en notación hexadecimal debe ser de cuatro a 320 dígitos hexadecimales, excluyendo los delimitadores de apertura y cierre. La longitud debe ser un múltiplo de cuatro.

Reglas de continuación
Se aplican las reglas de continuación normales.

El contenido de un literal nacional en notación hexadecimal se almacena como caracteres nacionales. El contenido resultante tiene el mismo significado que un literal nacional básico que especifica los mismos caracteres nacionales.

Un literal nacional en notación hexadecimal tiene clase de datos y categoría nacional y se puede utilizar en cualquier lugar donde se pueda utilizar un literal nacional básico.

Dónde pueden utilizarse los literales nacionales

Los literales nacionales se pueden utilizar de varias maneras.

Se pueden utilizar literales nacionales:

- En una cláusula VALUE asociada con un elemento de datos de clase nacional o una cláusula VALUE asociada con un nombre-condición para una variable condicional definida con el uso NATIONAL
- En constante figurativa ALL
- En una condición de relación
- En la frase WHEN de una sentencia format-2 SEARCH (búsqueda binaria)
- En la frase ALL, INICIAL o FIRST de una sentencia INSPECT
- En la frase BEFORE o AFTER de una sentencia INSPECT
- En la frase DELIMITED BY de una sentencia STRING
- En la frase DELIMITED BY de una sentencia UNSTRING
- Como argumento pasado BY CONTENT en la sentencia CALL
- Como argumento pasado BY VALUE en una sentencia CALL
- En las sentencias DISPLAY y EVALUATE
- Como elemento remitente en las siguientes sentencias de procedimiento:
  - INICIALIZAR
  - INSPECT
  - MOVE
  - Serie

  - UNSTRING
- En la lista de argumentos a las siguientes funciones intrínsecas:

  DISPLAY-OF, LENGTH, LOWER-CASE, MAX, MIN, ORD-MAX, ORD-MIN, REVERSE, UPPER-CASE, USUPPLEMENTARY y UVALID

  Nota: Los literales DBCS no se pueden utilizar en las funciones USUPPLEMENTARY y UVALID.
- En las sentencias de dirección de compilador COPY, REPLACE y TITLE

Un literal nacional sólo puede utilizarse tal como se especifica en las normas detalladas del presente documento.

Series de caracteres PICTURE

Una *serie de caracteres PICTURE* se compone del símbolo de moneda y de determinadas combinaciones de caracteres en el juego de caracteres COBOL. Carácter PICTURE: las series están delimitadas sólo por el espacio separador, la coma separador, el punto y coma separador o el punto y coma separador.

Aparece un gráfico de símbolos de cláusula PICTURE en Tabla 17 en la página 192.

Comentarios

Un *comentario* es una serie de caracteres que puede contener cualquier combinación de caracteres del juego de caracteres del sistema.

No tiene ningún efecto en la ejecución del programa. Hay tres formas de comentarios:

Entrada de comentario (IDENTIFICACIÓN DIVISION)
Este formulario se describe en “Párrafos opcionales” en la página 90.

Línea de comentario (cualquier división)
Este formulario se describe en “Líneas de comentario” en la página 50.

Comentarios en línea (cualquier división)
Un comentario en línea se identifica mediante un indicador de comentario flotante (* >) precedido por una o más series de caracteres en el área de texto de programa, y se puede escribir en cualquier línea de un grupo de compilación. Todos los caracteres que siguen al indicador de comentario flotante hasta el final del área B son texto de comentario.

Las series de caracteres que forman comentarios pueden contener cualquier carácter de un solo byte o de varios bytes de la página de códigos en vigor para la compilación.

Se permiten varias líneas de comentario que contienen series multibyte . La inclusión de caracteres de multibyte en una línea de comentario se debe realizar línea a línea. Las palabras que contienen estos caracteres no pueden continuar en una línea siguiente. No se proporciona ninguna comprobación de sintaxis para las series válidas en las líneas de comentario.

Capítulo 4. Separadores

Un *separador* es un carácter o una serie de dos o más caracteres contiguos que delimitan series de caracteres.

Los separadores se muestran en la tabla siguiente.

Tabla 4. ***Separadores***

Separador	**Significado**
b[1]	Espacio
,*b*[1]	Coma
.*b*[1]	Período
;*b*[1]	Punto y coma
(	Paréntesis izquierdo
)	Paréntesis derecho
:	Dos puntos
"*b*[1]	Comillas
'*b*[1]	Apóstrofo
B"	Delimitador de apertura para literal booleano
X"	Delimitador de apertura para un literal alfanumérico de formato hexadecimal
X'	Delimitador de apertura para un literal alfanumérico de formato hexadecimal
Z"	Delimitador de apertura para un literal alfanumérico terminado en nulo
Z'	Delimitador de apertura para un literal alfanumérico terminado en nulo
N"	Delimitador de apertura para un literal nacional[2]
N'	Delimitador de apertura para un literal nacional[2]
NX"	Delimitador de apertura para un literal nacional de formato hexadecimal
NX'	Delimitador de apertura para un literal nacional de formato hexadecimal
G"	Delimitador de apertura para un literal DBCS
G'	Delimitador de apertura para un literal DBCS
==	Delimitador de pseudotexto

1. *b* representa un espacio en blanco.
2. N" y N' son el delimitador de apertura para un literal DBCS cuando la opción de compilador NSYMBOL (DBCS) está en vigor.

Reglas para separadores

Un separador es una serie de uno o más caracteres de puntuación.

En la descripción siguiente, {} (llaves) encierre cada separador y *b* representa un espacio. En cualquier lugar en el que se utilice un espacio como separador o como parte de un separador, se puede utilizar más de un espacio.

Espacio {*b*}
Un espacio puede preceder o seguir inmediatamente cualquier separador excepto:

- El delimitador de pseudotexto de apertura, donde es necesario el espacio anterior.
- Entre comillas. Los espacios entre comillas se consideran parte del literal alfanumérico; no se consideran separadores.

Periodo {.*b*}, Comma {,*b*}, Semicolon {;*b*}
Una coma de separador se compone de una coma seguida de un espacio. Un punto de separación se compone de un punto seguido de un espacio. Un punto y coma de separador se compone de un punto y coma seguido de un espacio.

El punto de separación sólo debe utilizarse para indicar el final de una frase, o como se muestra en los formatos. La coma de separador y el punto y coma de separador se pueden utilizar en cualquier lugar donde se utilice el espacio de separador.

- En la DIVISIÓN DE IDENTIFICACIÓN, cada párrafo debe terminar con un punto de separación.
- En ENVIRONMENT DIVISION, los párrafos SOURCE-COMPUTER, OBJECT-COMPUTER, SPECIAL-NAMES e I-O-CONTROL deben terminar cada uno con un punto de separación. En el párrafo FILE-CONTROL, cada entrada de control de archivo debe finalizar con un punto separador.
- En DATA DIVISION, las entradas de archivo (FD), archivo de clasificación/fusión (SD) y descripción de datos deben finalizar cada una con un punto de separación.
- En PROCEDURE DIVISION, las comas separadoras o los signos de punto y coma separadores pueden separar sentencias dentro de una frase y operandos dentro de una sentencia. Cada frase y cada procedimiento deben terminar con un punto de separación.

Paréntesis { (} ... {) }
Excepto en pseudotexto, los paréntesis sólo pueden aparecer en pares equilibrados de paréntesis izquierdo y derecho. Delimitan subíndices, una lista de argumentos de función, modificadores de referencia, expresiones aritméticas o condiciones.

Dos puntos { : }
Los dos puntos son un separador y son necesarios cuando se muestran en formatos generales.

comillas {"} ... {"}
Una comilla de apertura debe ir inmediatamente precedida de un espacio o un paréntesis izquierdo. Una comilla de cierre debe ir seguida inmediatamente de un espacio separador, coma, punto y coma, punto, paréntesis derecho o delimitador de pseudotexto. Las comillas deben aparecer como pares equilibrados. Delimitan literales alfanuméricos, excepto cuando el literal continúa (consulte "Líneas de continuación" en la página 48).

Apóstrofos {'} ... {'}
Un apóstrofo de apertura debe ir inmediatamente precedido de un espacio o un paréntesis izquierdo. Un apóstrofo de cierre debe ir inmediatamente seguido de un espacio separador, coma, punto y coma, punto, paréntesis derecho o delimitador de pseudotexto. Los apóstrofos deben aparecer como pares equilibrados. Delimitan literales alfanuméricos, excepto cuando el literal continúa (consulte "Líneas de continuación" en la página 48).

Delimitadores literales terminados en nulo {Z"} ... {"}, {Z'} ... {'}
El delimitador de apertura debe ir inmediatamente precedido de un espacio o un paréntesis izquierdo. El delimitador de cierre debe ir inmediatamente seguido de un espacio separador, coma, punto y coma, punto, paréntesis derecho o pseudo-delimitador de texto.

Delimitadores literales DBCS {G"} ... {"}, {G'} ... {'}, {N"} ... {"}, {N'} ... {'}
El delimitador de apertura debe ir inmediatamente precedido de un espacio o un paréntesis izquierdo. El delimitador de cierre debe ir inmediatamente seguido de un espacio separador, coma, punto y coma, punto, paréntesis derecho o pseudo-delimitador de texto. `N"` y `N'` son delimitadores literales DBCS cuando la opción de compilador NSYMBOL (DBCS) está en vigor.

Delimitadores literales nacionales {N"} ... {"}, {N'} ... {'}, {NX"} ... {"}, {NX'} ... {'}
El delimitador de apertura debe ir inmediatamente precedido de un espacio o un paréntesis izquierdo. El delimitador de cierre debe ir inmediatamente seguido de un espacio separador, coma, punto y

coma, punto, paréntesis derecho o pseudo-delimitador de texto. `N"` y `N'` son delimitadores literales DBCS cuando la opción de compilador NSYMBOL (DBCS) está en vigor.

Delimitadores de pseudo-texto {*b*==} ... {==*b*}
Un delimitador de pseudotexto de apertura debe ir inmediatamente precedido de un espacio. Un delimitador de pseudotexto de cierre debe ir inmediatamente seguido de un espacio de separador, coma, punto y coma o punto. Los delimitadores de pseudotexto deben aparecer como pares equilibrados. Delimitan el pseudo-texto. (Consulte “sentencia COPY” en la página 508.)

Cualquier carácter de puntuación incluido en una serie de caracteres PICTURE, una serie de caracteres de comentario o un literal alfanumérico no se considera un carácter de puntuación, sino que forma parte de la serie de caracteres o literal.

Capítulo 5. Secciones y párrafos

Las secciones y párrafos definen un programa. Las secciones y párrafos se subdividen en frases, sentencias y entradas.

Las frases se subdividen en sentencias, y las sentencias se subdividen en frases. Las entradas se subdividen en cláusulas.

Para obtener más detalles, consulte:

- “Frases, sentencias y entradas” en la página 43
- “Sentencias” en la página 44
- “Frases” en la página 44
- “Cláusulas” en la página 44

Para obtener más información sobre secciones, párrafos y sentencias, consulte “Procedimientos” en la página 245.

Frases, sentencias y entradas

A menos que las reglas asociadas indiquen explícitamente lo contrario, cada cláusula o sentencia necesaria debe escribirse en la secuencia que se muestra en su formato. Si se utilizan cláusulas o sentencias opcionales, deben escribirse en la secuencia que se muestra en sus formatos. Estas reglas son ciertas incluso para cláusulas y sentencias tratadas como comentarios.

La jerarquía sintáctica sigue este formato:

- DIVISIÓN DE IDENTIFICACIÓN
  - Párrafos
    - Entradas
      - Cláusulas
- DIVISIÓN DE MEDIO AMBIENTE
  - Secciones
    - Párrafos
      - Entradas
        - Cláusulas
          - Frases
- DIVISIÓN DE DATOS
  - Secciones
    - Entradas
      - Cláusulas
        - Frases
- DIVISIÓN DE PROCEDIMIENTO
  - Secciones
    - Párrafos
      - Frases
        - Sentencias
          - Frases

Entradas

Una *entrada* es una serie de cláusulas que termina con un punto separador. Las entradas se construyen en las divisiones de identificación, entorno y datos.

Cláusulas

Una *cláusula* es un conjunto ordenado de series de caracteres COBOL consecutivas que especifica un atributo de una entrada. Las cláusulas se construyen en las divisiones de identificación, entorno y datos.

Frases

Una *frase* es una secuencia de una o más sentencias que termina con un punto separador. Las frases se construyen en la DIVISION PROCEDURE.

Sentencias

Una *sentencia* especifica una acción que debe realizar el programa. Las sentencias se construyen en PROCEDURE DIVISION.

Para obtener descripciones de los distintos tipos de sentencias, consulte:

- “Declaraciones imperativas” en la página 275
- “Sentencias condicionales” en la página 277
- Capítulo 7, “Ámbito de nombres”, en la página 53
- Capítulo 21, “Sentencias de direccionamiento de compilador”, en la página 505

Frases

Cada cláusula o sentencia de un programa se puede subdividir en unidades más pequeñas denominadas *frases*.

Capítulo 6. Formato de referencia

El texto de origen COBOL debe escribirse en COBOL *formato de referencia*, que puede ser *formato de origen fijo* o *formato de origen ampliado*.

El formato Origen fijo consta de las siguientes áreas en una línea de 72 caracteres. El formato de origen ampliado consta de las áreas siguientes en una línea de 252 caracteres. En el formato de origen fijo y ampliado, el área de texto de programa empieza en la posición de caracteres 8 y continúa hasta el final del área B.

Área de número de secuencia
Columnas 1 a 6

Área de indicador
Columna 7

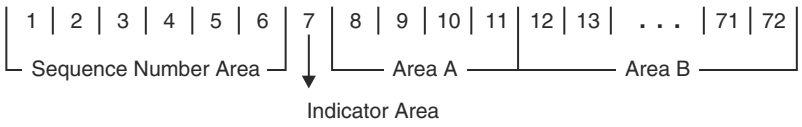
Área A
Columnas 8 a 11

Área B
Columnas 12 a 72 en formato de origen fijo

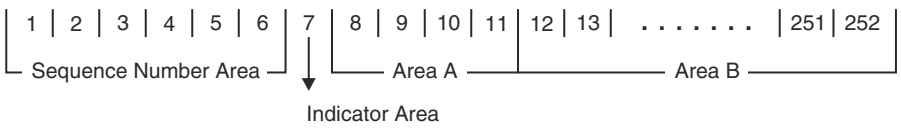
Columnas 12 a 252 en formato de origen ampliado

Recuerde: En formato de origen fijo y formato de origen ampliado, las líneas más cortas que la longitud máxima se amplían con espacios hasta la longitud máxima.

Esta figura ilustra el formato de de origen fijopara una línea de origen COBOL.



Esta figura ilustra el formato de origen ampliado para una línea de origen COBOL.



Para obtener información sobre cómo indicar el formato de origen, consulte *SRCFORMAT* en la publicación *COBOL for Linux en x86 Guía de programación*.

Los temas siguientes proporcionan detalles sobre estas áreas:

- "Área de número de secuencia" en la página 45
- "Área de indicador" en la página 46
- "Área A" en la página 46
- "Área B" en la página 47
- "Área A o Área B" en la página 50
- "Programa de utilidad de conversión de origen (scu)" en la página 51

Área de número de secuencia

El área de número de secuencia se puede utilizar para etiquetar una línea de sentencia de origen. El contenido de esta área puede constar de cualquier carácter del juego de caracteres del sistema.

Área de indicador

Utilice el área de indicador para especificar la continuación de palabras o literales alfanuméricos de la línea anterior en la línea actual, el tratamiento del texto como documentación y las líneas de depuración.

Consulte "Líneas de continuación" en la página 48, "Líneas de comentario" en la página 50y "Depuración de líneas" en la página 51.

El área de indicador se puede utilizar para el formato de listado fuente. Una barra inclinada (/) colocada en la columna de indicador hace que el compilador inicie una nueva página para el listado de origen y que el registro de origen correspondiente se trate como un comentario. El efecto puede depender de la opción de compilador LINECOUNT. Para obtener información sobre la opción de compilador LINECOUNT, consulte *LINECOUNT* en la publicación *COBOL for Linux en x86 Guía de programación*.

Área A

Determinados elementos deben empezar en el área A.

Estos elementos son:

- Cabeceras de división
- "Cabeceras de sección" en la página 46
- Cabeceras de párrafo o nombres de párrafo
- Indicadores de nivel o números de nivel (01 y 77)
- DECLARATIVES y END DECLARATIVES
- Marcadores END PROGRAM

Cabeceras de división

Una cabecera de división es una combinación de palabras, seguida de un punto de separación para indicar el principio de una división.

Consulte las siguientes cabeceras de división:

- DIVISION DE IDENTIFICACION.
- DIVISION DE AMBIENTE.
- DIVISION DE DATOS.
- DIVISION DE PROCEDIMIENTO.

Una cabecera de división (excepto cuando se especifica una frase USING con una cabecera PROCEDURE DIVISION) debe ir seguida inmediatamente de un punto separador. A excepción de la frase USING, no puede aparecer ningún texto en la misma línea.

Cabeceras de sección

En las divisiones de entorno y procedimiento, una cabecera de sección indica el principio de una serie de párrafos.

Por ejemplo:

```
INPUT-OUTPUT SECTION.
```

En DATA DIVISION, una cabecera de sección indica el principio de una entrada; por ejemplo:

```
FILE SECTION.
LINKAGE SECTION.
LOCAL-STORAGE SECTION.
```

```
WORKING-STORAGE SECTION.
```

Una cabecera de sección debe ir seguida inmediatamente de un punto de separación.

Cabeceras de párrafo o nombres de párrafo

Una cabecera de párrafo o un nombre de párrafo indica el principio de un párrafo.

En ENVIRONMENT DIVISION, un párrafo consta de una cabecera de párrafo seguida de una o más entradas. Por ejemplo:

```
OBJECT-COMPUTER. computer-name.
```

En PROCEDURE DIVISION, un párrafo consiste en un nombre de párrafo seguido de una o más frases.

Indicadores de nivel (FD y SD) o números de nivel (01 y 77)

Un indicador de nivel puede ser FD o SD.

Un indicador de nivel debe empezar en el área A y estar seguido por un espacio. (Consulte "SECCIÓN DE ARCHIVO" en la página 158.) Un número de nivel que debe empezar en el Área A es un entero de uno o dos dígitos con un valor de 01 o 77. Debe ir seguido de un espacio o un punto separador.

DECLARATIVES y END DECLARATIVES

DECLARATIVES y END DECLARATIVES son palabras clave que inician y finalizan la parte declarativa de la unidad de origen.

En PROCEDURE DIVISION, cada una de las palabras clave DECLARATIVES y END DECLARATIVES deben comenzar en el Área A y ser seguidas inmediatamente por un punto separador; ningún otro texto puede aparecer en la misma línea. Después de las palabras clave END DECLARATIVES, no puede aparecer ningún texto antes de la cabecera de sección siguiente. (Consulte "Declaraciones" en la página 244.)

Finalizar marcadores PROGRAM

Un marcador END PROGRAM indica el final de un programa COBOL.

Por ejemplo:

```
END PROGRAM program-name.
```

Para programas
nombre-programa debe ser idéntico al *nombre-programa* del párrafo PROGRAM-ID correspondiente. Cada programa COBOL, excepto un programa externo que no contiene programas anidados y no va seguido de otro programa por lotes, debe finalizar con un marcador END PROGRAM.

Área B

Determinados elementos deben comenzar en el área B.

Estos elementos son:

- Entradas, frases, sentencias y cláusulas
- Líneas de continuación

Entradas, frases, sentencias, cláusulas

La primera entrada, frase, sentencia o cláusula empieza en la misma línea que la cabecera o el nombre de párrafo que sigue, o en el Área B de la siguiente línea no en blanco que no es una línea de comentario. Las

frases o entradas sucesivas empiezan en el Área B de la misma línea que la frase o entrada anterior, o en el Área B de la siguiente línea no en blanco que no es una línea de comentario.

Dentro de una entrada o frase, las líneas sucesivas del Área B pueden tener el mismo formato o pueden ser sangradas para aclarar la lógica del programa. El listado de salida sólo se sangra si las sentencias de entrada están sangradas. La sangría no afecta al significado del programa. El programador puede elegir la cantidad de sangría, sujeta sólo a las restricciones sobre el ancho del área B. Consulte también Capítulo 5, “Secciones y párrafos”, en la página 43.

Líneas de continuación

Cualquier frase, entrada, cláusula o frase que requiera más de una línea puede continuar en el Área B de la siguiente línea que no sea ni una línea de comentario ni una línea en blanco.

La línea que continúa es una *línea continua*; las líneas siguientes son *líneas de continuación*. El área A de una línea de continuación debe estar en blanco.

Si no hay ningún guión (-) en el área de indicador (columna 7) de una línea, se supone que el último carácter de la línea anterior va seguido de un espacio.

Los elementos siguientes no pueden continuar:

- Palabras definidas por el usuario multibyte
- Literales DBCS
- Literales alfanuméricos que contienen caracteres multibyte
- Literales nacionales que contienen caracteres multibyte

Sin embargo, los literales alfanuméricos y los literales nacionales en notación hexadecimal pueden continuar independientemente del tipo de caracteres expresados en notación hexadecimal.

Todos los caracteres que forman un delimitador literal de apertura deben estar en la misma línea. Por ejemplo, `Z"`, `G"`, `N"`, `NX"` o `X"`.

Ambos caracteres que forman el separador de delimitador de pseudo-texto, ==, el indicador de comentario flotante, * >, o el indicador de directiva de compilador, > >, deben estar en la misma línea.

Una directiva de compilador o frase de directiva de compilador, que empieza por > >, debe especificarse en la misma línea.

Si hay un guión en el área de indicador de una línea, el primer carácter no en blanco de la línea de continuación sigue inmediatamente al último carácter no en blanco de la línea de continuación sin un espacio intermedio.

Continuación de literales alfanuméricos y nacionales

Los literales alfanuméricos y nacionales sólo pueden continuar cuando no hay caracteres multibyte en el contenido del literal.

Las reglas siguientes se aplican a los literales alfanuméricos y nacionales que no contienen caracteres multibyte :

- Si la línea continua contiene un literal alfanumérico o nacional sin comillas de cierre, todos los espacios al final de la línea continua (a través de la columna 72 en formato de origen fijo, o a través de la columna 252 en formato de origen ampliado) se consideran parte del literal. La línea de continuación debe contener un guión en el área de indicador, y el primer carácter no en blanco debe ser una comilla. La continuación del literal empieza con el carácter que sigue inmediatamente a las comillas.
- Si un literal alfanumérico o nacional que debe continuar en la línea siguiente tiene como último carácter una comilla en la columna 72 en formato de origen fijo, o en la columna 252 en formato de origen ampliado, la línea de continuación debe empezar con dos comillas consecutivas. Esto dará como resultado una comilla como parte del valor del literal.

Si el último carácter de la línea continua de un literal alfanumérico o nacional es una comilla en el Área B, la línea de continuación puede empezar con una comilla. Esto dará como resultado dos literales consecutivos en lugar de un literal continuado.

Las reglas son las mismas cuando se utiliza un apóstrofo en lugar de una comilla en delimitadores.

Si desea continuar un literal de tal manera que las líneas continuas y las líneas de continuación formen parte de un literal:

- Codifique un guión en el área de indicador de cada línea de continuación.
- Codifique el valor literal utilizando todas las columnas de cada línea continua, hasta la columna 72 en formato de origen fijo e incluida, o la columna 252 en formato de origen ampliado. (No termine las líneas continuadas con una comilla seguida de un espacio.)
- Codifique una comilla antes del primer carácter del literal en cada línea de continuación.
- Termine la última línea de continuación con una comilla seguida de un espacio.

En los siguientes ejemplos de formato de origen fijo , el número y el tamaño de los literales creados se indican a continuación del ejemplo:

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000001               "AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE
      -              "GGGGGGGGGGHHHHHHHHHHIIIIIIIIIIJJJJJJJJJJKKKKKKKKKK
      -    "LLLLLLLLLLMMMMMMMMMM"
```

- El literal 000001 se interpreta como un literal alfanumérico con una longitud de 120 bytes. Cada carácter entre las comillas iniciales y hasta la columna 72 inclusive de líneas continuas se cuenta como parte del literal.

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000003             N"AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE
      -              "GGGGGGGGGG"
```

- El literal 000003 se interpreta como un literal nacional que tiene una longitud de 60 posiciones de caracteres nacionales (120 bytes). Cada carácter entre las comillas iniciales y las comillas finales en la línea continua se cuenta como parte del literal. Aunque se especifican caracteres de un solo byte, el valor de los literales se almacena como caracteres nacionales.

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000005     "AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE
      -    "GGGGGGGGGGHHHHHHHHHHIIIIIIIIIIJJJJJJJJJJKKKKKKKKKK
      -    "LLLLLLLLLLMMMMMMMMMM"
```

- El literal 000005 se interpreta como un literal con una longitud de 140 bytes. Los espacios en blanco al final de cada línea continua se cuentan como parte del literal porque las líneas continuas no terminan con una comilla.

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000010     "AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE"
      -    "GGGGGGGGGGHHHHHHHHHHIIIIIIIIIIJJJJJJJJJJKKKKKKKKKK"
      -    "LLLLLLLLLLMMMMMMMMMM"
```

- El literal 000010 se interpreta como tres literales separados que tienen longitudes de 50, 50 y 20, respectivamente. Las comillas con el siguiente espacio terminan la línea de continuación. Sólo los caracteres entre comillas se cuentan como parte de los literales. El literal 000010 no es válido como literal de cláusula VALUE para elementos de datos non-level-88 .

Para codificar un literal continuado donde la longitud de cada parte continuada del literal es menor que la longitud del Área B, ajuste la columna inicial de forma que el último carácter de la parte continuada esté en la columna 72 en formato de origen fijo, o en la columna 252 en formato de origen ampliado.

Área A o Área B

Determinados elementos pueden empezar en el Área A o en el Área B.

Estos elementos son:

- Números de nivel
- Líneas de comentario
- Indicadores de comentario flotante (* >)
- Sentencias de direccionamiento de compilador
- Directivas de compilador
- Depuración de líneas
- Pseudotexto
- Líneas en blanco

Nivel-números

Un número de nivel que puede empezar en el área A o B es un entero de uno o dos dígitos con un valor de 02 a 49, 66 u 88.

Un número de nivel que debe empezar en el Área A es un entero de uno o dos dígitos con un valor de 01 o 77. Un número de nivel debe ir seguido de un espacio o un punto separador. Para obtener más información, consulte "Nivel-números" en la página 170.

Líneas de comentario

Una *línea de comentario* es cualquier línea con un asterisco (*) o una barra inclinada (/) en el área de indicador (columna 7) de la línea, o con un indicador de comentario flotante (* >) como la primera serie de caracteres en el área de texto del programa (Área A más Área B).

El comentario se puede escribir en cualquier lugar del área de texto del programa de esa línea, y puede constar de cualquier combinación de caracteres del juego de caracteres del sistema.

Las líneas de comentario se pueden colocar en cualquier lugar de un programa. Las líneas de comentario colocadas antes de la cabecera IDENTIFICATION DIVISION deben seguir a las tarjetas de control (por ejemplo, PROCESS o CBL).

Importante: Los comentarios mezclados con tarjetas de control podrían anular algunas de las tarjetas de control y hacer que se diagnosticaran como errores.

Se permiten varias líneas de comentario. Cada uno debe empezar por asterisco (*) o una barra inclinada (/) en el área de indicador, o con un indicador de comentario flotante (* >).

Para obtener más información sobre los indicadores de comentario flotante, consulte "Indicadores de comentario flotante (* >)" en la página 50.

Se imprime una línea de comentario de asterisco (*) en la siguiente línea disponible en el listado de salida. El efecto puede depender de la opción de compilador LINECOUNT. Para obtener información sobre la opción de compilador LINECOUNT, consulte *LINECOUNT* en la publicación *COBOL for Linux en x86 Guía de programación*. Se imprime una línea de comentario de barra inclinada (/) en la primera línea de la página siguiente y se expulsa la página actual del listado de salida.

El compilador trata una línea de comentario como documentación y no la comprueba sintácticamente.

Indicadores de comentario flotante (* >)

Además de los indicadores fijos que sólo se pueden especificar en el área de indicador del formato de referencia de origen, se puede especificar un indicador de comentario flotante (* >) en cualquier lugar del área de texto de programa para indicar una línea de comentario o un comentario en línea.

Un indicador de comentario flotante indica una línea de comentario si es la primera serie de caracteres en el área de texto de programa (Área A más Área B), o indica un comentario en línea si está detrás de una o más series de caracteres en el área de texto de programa.

Estas son las reglas para los indicadores de comentario flotante:

- Ambos caracteres (* y >) que forman el indicador flotante de varios caracteres deben ser contiguos y estar en la misma línea.
- El indicador de comentario flotante para un comentario en línea debe ir precedido de un espacio separador y se puede especificar siempre que se pueda especificar un espacio separador.
- Todos los caracteres que siguen al indicador de comentario flotante hasta el final del Área B son texto de comentario.

Sentencias de direccionamiento de compilador

La mayoría de las sentencias de direccionamiento de compilador, incluidas COPY y REPLACE, pueden empezar en el Área A o en el Área B.

Las sentencias BASIS, PROCESS (CBL), *CBL (*CONTROL), DELETE, EJECT, INSERT, SKIP1, SKIP2, SKIP3y TITLE también pueden empezar en el Área A o en el Área B.

Directivas de compilador

Las directivas de compilador deben empezar en el Área B.

Para obtener más información, consulte Capítulo 22, “Directivas de compilador”, en la página 529.

Depuración de líneas

Una *línea de depuración* es cualquier línea con una D (o d) en el área de indicador de la línea.

Las líneas de depuración se pueden escribir en ENVIRONMENT DIVISION (después del párrafo OBJECT-COMPUTER), DATA DIVISION y PROCEDURE DIVISION. Si una línea de depuración sólo contiene espacios en el Área A y el Área B, se considera una línea en blanco.

Consulte "WITH DEBUGGING MODE" en “Párrafo SOURCE-COMPUTER” en la página 95.

Pseudotexto

Las series de caracteres y separadores que componen el *pseudo-texto* pueden empezar en el Área A o en el Área B.

Sin embargo, si hay un guión en el área de indicador (columna 7) de una línea que sigue al delimitador de pseudo-texto de apertura, el área A de la línea debe estar en blanco y las reglas para las líneas de continuación se aplican a la formación de palabras de texto. Consulte “Líneas de continuación” en la página 48 para obtener más detalles.

Líneas en blanco

Una *línea en blanco* no contiene más que espacios en la columna 7 a la columna 72 en formato de origen fijo, o en la columna 7 a la columna 252 en formato de origen ampliado. Una línea en blanco puede estar en cualquier lugar de un programa.

Programa de utilidad de conversión de origen (scu)

El programa de utilidad de conversión de origen (scu) es un programa Linux autónomo que ayuda en la conversión de programas fuente COBOL desde formatos de origen que no son deIBM o de formato libre a un formato que puede ser compilado por COBOL para Linux.

Para obtener más información sobre scu, consulte "Programa de utilidad de conversión de origen (scu)" en la *Guía de migración*.

Capítulo 7. Ámbito de nombres

Una palabra definida por el usuario nombra un recurso de datos o un elemento de programación COBOL. Ejemplos de recursos de datos con nombre son un archivo, un elemento de datos o un registro. Los ejemplos de elementos de programación con nombre son un programa o un párrafo.

Las secciones siguientes definen los tipos de nombres en COBOL y explican dónde se puede hacer referencia a los nombres:

- "Tipos de nombres" en la página 53
- "Recursos externos e internos" en la página 55
- "Resolución de nombres" en la página 56

Tipos de nombres

Además de identificar un recurso, un nombre puede tener atributos globales o locales. Algunos nombres siempre son globales, algunos nombres siempre son locales y algunos nombres son locales o globales en función de las especificaciones del programa en el que los nombres están definidos.

Para programas

Se puede utilizar un *nombre global* para hacer referencia al recurso con el que está asociado:

- Desde dentro del programa en el que el nombre global está definido
- Desde cualquier otro programa contenido en el programa que define el nombre global

Utilice la cláusula GLOBAL en la entrada de descripción de datos para indicar que un nombre es global. Para obtener más información sobre cómo utilizar la cláusula GLOBAL, consulte "cláusula GLOBAL" en la página 159.

Un *nombre local* sólo se puede utilizar para hacer referencia al recurso con el que está asociado desde dentro del programa en el que el nombre local está definido.

De forma predeterminada, si un nombre de datos, un nombre de archivo, un nombre de registro, un nombre de clave de registro, o una definición de nombre de condición en una entrada de descripción de datos no incluye la cláusula GLOBAL, el nombre es local.

Restricción: Las reglas específicas a veces prohíben especificar la cláusula GLOBAL para determinadas entradas de descripción de datos, descripción de archivo o descripción de registro.

La lista siguiente indica los nombres que puede utilizar y si el nombre puede ser local o global:

nombre-datos

nombre-datos asigna un nombre a un elemento de datos.

Un nombre de datos es global si se especifica la cláusula GLOBAL en la entrada de descripción de datos que define el nombre de datos o en otra entrada a la que está subordinada dicha entrada de descripción de datos.

nombre-archivo

nombre-archivo asigna un nombre a un conector de archivo.

Un nombre de archivo es global si se especifica la cláusula GLOBAL en la entrada de descripción de archivo para ese nombre de archivo.

nombre-registro

nombre-registro asigna un nombre a un registro.

Un nombre de registro es global si se especifica la cláusula GLOBAL en la descripción de registro que define el nombre de registro, o en el caso de entradas de descripción de registro en FILE SECTION, si se especifica la cláusula GLOBAL en la entrada de descripción de archivo para el nombre de archivo asociado a la entrada de descripción de registro.

nombre-clave-registro
record-key-name asigna un nombre a una clave asociada con un archivo indexado.

Puede definir el nombre de clave de registro utilizando la frase SOURCE en la cláusula ALTERNATE RECORD KEY o utilizando la cláusula RECORD KEY de la entrada de control de archivo para un archivo indexado. Un nombre de clave de registro es global si se especifica la cláusula GLOBAL en la entrada de descripción de archivo para ese archivo.

Restricción: *nombre-clave-registro* sólo está soportado para el sistema de archivos STL.

nombre-condición
nombre-condición asocia un valor con una variable condicional.

Un nombre de condición definido en una entrada de descripción de datos es global si dicha entrada está subordinada a otra entrada que especifica la cláusula GLOBAL.

Un nombre de condición definido dentro de la sección de configuración siempre es global.

nombre-programa
nombre-programa asigna un nombre a un programa externo o interno (anidado). Para obtener más información, consulte "Convenios para nombres de programa" en la página 82.

Un nombre de programa no es local ni global. Para obtener más información, consulte "Convenios para nombres de programa" en la página 82.

nombre-sección
nombre-sección asigna un nombre a una sección de PROCEDURE DIVISION.

Un nombre de sección siempre es local.

nombre-párrafo
nombre-párrafo asigna un nombre a un párrafo en PROCEDURE DIVISION.

Un nombre de párrafo es siempre local.

nombre-base
nombre-base especifica el nombre del texto de origen que el compilador incluye en la unidad de origen. Para obtener detalles, consulte "sentencia BASIS" en la página 505.

nombre-biblioteca
nombre-biblioteca especifica la biblioteca COBOL que utiliza el compilador para incluir texto COPY. Para obtener detalles, consulte "sentencia COPY" en la página 508.

nombre-texto
nombre-texto especifica el nombre del texto COPY que debe incluir el compilador en la unidad de origen. Para obtener detalles, consulte "sentencia COPY" en la página 508.

nombre-alfabeto
nombre_alfabeto asigna un nombre a un juego de caracteres específico o a una secuencia de clasificación, o a ambos, en el párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION.

Un nombre de alfabeto siempre es global.

***nombre-clase* (de datos)**
nombre-clase asigna un nombre a la proposición en el párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION para el que se puede definir un valor de verdad.

Un nombre de clase siempre es global.

nombre-mnemotécnico
mnemonic-name asigna una palabra definida por el usuario a un nombre de implementador.

Un nombre nemotécnico es siempre global.

carácter-simbólico
carácter simbólico especifica una constante figurativa definida por el usuario.

Un carácter simbólico es siempre global.

nombre-índice
nombre-índice asigna un nombre a un índice asociado con una tabla específica.

Si un elemento de datos que posee el atributo global incluye una tabla a la que se accede con un índice, dicho índice también posee el atributo global. Además, el ámbito de ese nombre-índice es idéntico al ámbito del nombre-datos que incluye la tabla.

nombre-tipo
nombre-tipo nombra un tipo de datos definido por el usuario que se puede utilizar en una cláusula TYPE para definir un elemento de datos.

Un nombre de tipo es global si se especifica la cláusula GLOBAL en la entrada de descripción de datos por la que se declara el nombre-tipo. El atributo GLOBAL de un nombre-tipo está restringido al nombre-tipo y no lo adquiere un elemento de datos definido utilizando el nombre-tipo en una cláusula TYPE.

Recursos externos e internos

El almacenamiento asociado con un elemento de datos o un conector de archivo puede ser *externo* o *interno* para el programa en el que se declara el recurso.

Un elemento de datos o conector de archivo es externo si el almacenamiento asociado con ese recurso está asociado con la unidad de ejecución en lugar de con cualquier programa en particular dentro de la unidad de ejecución. Cualquier programa de la unidad de ejecución que describe el recurso puede hacer referencia a un recurso externo. Las referencias a un recurso externo de distintos programas que utilizan descripciones separadas del recurso siempre están en el mismo recurso. En una unidad de ejecución, sólo hay una representación de un recurso externo.

Un recurso es interno si el almacenamiento asociado con dicho recurso sólo está asociado con el programa que describe el recurso.

Los recursos externos e internos pueden tener nombres globales o locales.

Un registro de datos descrito en WORKING-STORAGE SECTION recibe el atributo externo por la presencia de la cláusula EXTERNAL en su entrada de descripción de datos. Cualquier elemento de datos descrito por una entrada de descripción de datos subordinada a una entrada que describe un registro externo también alcanza el atributo externo. Si un registro o elemento de datos no tiene el atributo externo, forma parte de los datos internos del programa en el que se describe.

Dos programas en una unidad de ejecución pueden hacer referencia al mismo conector de archivo en las circunstancias siguientes:

- Se puede hacer referencia a un conector de archivo externo desde cualquier programa que describa dicho conector de archivo.
- Si un programa está contenido en otro programa, ambos programas pueden hacer referencia a un conector de archivo global haciendo referencia a un nombre de archivo global asociado en el programa contenedor o en cualquier programa que contenga directa o indirectamente el programa contenedor.

Dos programas en una unidad de ejecución pueden hacer referencia a datos comunes en las circunstancias siguientes:

- Se puede hacer referencia al contenido de datos de un registro de datos externo desde cualquier programa siempre que el programa haya descrito dicho registro de datos.
- Si un programa está contenido en otro programa, ambos programas pueden hacer referencia a datos que posean el atributo global en el programa o en cualquier programa que contenga directa o indirectamente el programa que lo contiene.

Los registros de datos descritos como subordinados a una entrada de descripción de archivo que no contiene la cláusula EXTERNAL o a una entrada de descripción de archivo de fusión de clasificación, así como cualquier elemento de datos descrito como subordinado a las entradas de descripción de datos para dichos registros, son siempre internos para el programa que describe el nombre de archivo. Si la cláusula EXTERNAL se incluye en la entrada de descripción de archivo, los registros de datos y los elementos de datos alcanzan el atributo externo.

Resolución de nombres

Cuando un programa, el programa B, está directamente contenido en otro programa, el programa A, ambos programas pueden definir un nombre de condición, un nombre de datos, un nombre de archivo, un nombre de clave de registro o un nombre de registro utilizando la misma palabra definida por el usuario. Cuando se hace referencia a un nombre duplicado de este tipo en el programa B, los pasos siguientes determinan el recurso referenciado:

1. El recurso referenciado se identifica a partir del conjunto de todos los nombres definidos en el programa B y todos los nombres globales definidos en el programa A y en cualquier programa que contenga directa o indirectamente el programa A. Las reglas normales de cualificación y cualquier otra regla de exclusividad de referencia se aplican a este conjunto de nombres hasta que se identifiquen uno o varios recursos.
2. Si sólo se identifica un recurso, es el recurso al que se hace referencia.
3. Si se identifica más de un recurso, no más de un recurso puede tener un nombre local para el programa B. Si cero o uno de los recursos tiene un nombre local para el programa B, se aplican las reglas siguientes:
   - Si el nombre se declara en el programa B, el recurso del programa B es el recurso al que se hace referencia.
   - Si el nombre no está declarado en el programa B, el recurso al que se hace referencia es:
     – El recurso del programa A si el nombre se declara en el programa A
     – El recurso del programa contenedor si el nombre está declarado en el programa que contiene el programa A

     Esta regla se aplica a programas que contienen más hasta que se encuentra un recurso válido.

Capítulo 8. Referencia a nombres de datos, bibliotecas de copia y nombres de PROCEDURE DIVISION

Se pueden hacer referencias a recursos externos e internos. Las referencias a datos y procedimientos pueden ser explícitas o implícitas.

Para obtener más información sobre las reglas de cualificación y sobre las referencias de datos explícitas e implícitas, consulte los temas siguientes:

- "Exclusividad de referencia" en la página 57
- "Especificación de atributo de datos" en la página 70

Exclusividad de referencia

Cada nombre definido por el usuario en un programa COBOL es asignado por el usuario para nombrar un recurso para resolver un problema de proceso de datos. Para utilizar un recurso, una sentencia de un programa COBOL debe contener una referencia que identifique de forma exclusiva ese recurso.

Para garantizar la exclusividad de referencia, se puede calificar un nombre definido por el usuario. Se necesita un subíndice para una referencia exclusiva a un elemento de tabla, excepto tal como se especifica en "Suscripción" en la página 63. Un nombre de datos o un nombre de función, cualquier subscript y el modificador de referencia especificado hacen referencia de forma exclusiva a un elemento de datos definido por la modificación de referencia.

Cuando se ha asignado el mismo nombre en programas separados a dos o más apariciones de un recurso de un tipo determinado, y cuando la calificación por sí misma no permite que las referencias de uno de esos programas diferencien entre los recursos con nombres idénticos, se aplican ciertas convenciones que limitan el ámbito de los nombres. Los convenios garantizan que el recurso identificado es el que se describe en el programa que contiene la referencia. Para obtener más información sobre la resolución de nombres de programa, consulte "Resolución de nombres" en la página 56.

A menos que las reglas de una sentencia especifiquen lo contrario, los subíndices y la modificación de referencia se evalúan sólo una vez como el primer paso para ejecutar dicha sentencia.

Cualificación

Un nombre que existe dentro de una jerarquía de nombres se puede hacer exclusivo especificando uno o más nombres de nivel superior en la jerarquía. Los nombres de nivel superior se denominan *calificadores*, y el proceso mediante el cual dichos nombres se hacen exclusivos se denomina *calificación*.

La calificación se especifica colocando una o más frases después de un nombre especificado por el usuario, con cada frase formada por la palabra IN o OF seguida de un calificador. (IN y OF son lógicamente equivalentes.)

En cualquier jerarquía, el nombre de datos asociado con el nivel más alto debe ser exclusivo si se hace referencia a él y no se puede calificar.

Debe especificar suficiente calificación para que el nombre sea exclusivo; sin embargo, no siempre es necesario especificar todos los niveles de la jerarquía. Por ejemplo, si hay más de un archivo cuyos registros contienen el campo `EMPLOYEE-NO`, pero sólo uno de los archivos tiene un registro denominado `MAIN-RECORD`:

- `EMPLOYEE-NO OF MAIN-RECORD` califica suficientemente `EMPLOYEE-NO`.
- `EMPLOYEE-NO OF MAIN-RECORD OF MAIN-FILE` es válido pero no es necesario.

Reglas de cualificación

Las reglas para calificar un nombre son:

- Un nombre se puede calificar aunque no necesite calificación excepto en una cláusula REDEFINES, en cuyo caso no se debe calificar.
- Cada calificador debe tener un nivel superior al nombre que califica y debe estar dentro de la misma jerarquía.
- Si hay más de una combinación de calificadores que garantiza la exclusividad, se puede utilizar cualquiera de estas combinaciones.

Nombres idénticos

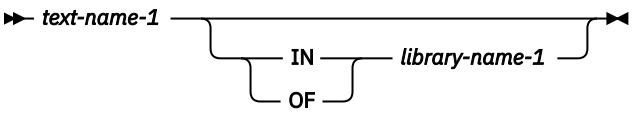
Cuando los programas están contenidos directa o indirectamente dentro de otros programas, cada programa puede utilizar palabras definidas por el usuario idénticas para nombrar recursos.

Un programa hace referencia a los recursos que el programa describe en lugar de a los recursos con el mismo nombre descritos en otro programa, incluso si los nombres son tipos diferentes de palabras definidas por el usuario.

Referencias a bibliotecas COPY

Si no se especifica *library-name-1* , se presupone que SYSLIB es el nombre de biblioteca.

Formato

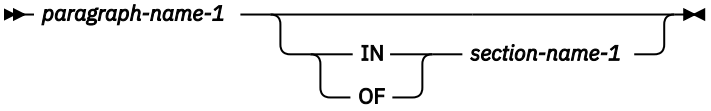


Para ver las reglas sobre cómo hacer referencia a bibliotecas COPY, consulte “sentencia COPY” en la página 508.

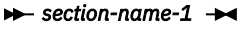
Referencias a nombres de PROCEDURE DIVISION

Los nombres de PROCEDURE DIVISION a los que se hace referencia explícitamente en un programa deben ser exclusivos dentro de una sección.

Formato 1



Formato 2



Un nombre de sección es el calificador más alto y único disponible para un nombre de párrafo y debe ser exclusivo si se hace referencia a él. (Los nombres de sección se describen en “Procedimientos” en la página 245.)

Si se hace referencia de forma explícita, un nombre de párrafo no debe duplicarse dentro de una sección. Cuando un nombre de párrafo está calificado por un nombre de sección, la palabra SECTION no debe aparecer. No será necesario calificar un nombre de párrafo cuando se haga referencia a él en la sección en

la que aparece. Un nombre de párrafo o un nombre de sección que aparece en un programa no se puede referenciar desde ningún otro programa.

Referencias a nombres DATA DIVISION

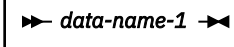
En esta sección se tratan los siguientes tipos de referencias.

- "Referencia de datos simple" en la página 59
- "Identificadores" en la página 59

Referencia de datos simple

El método más básico para hacer referencia a elementos de datos en un programa COBOL es la *referencia de datos simple*, que es *data-name-1* sin calificación, suscripción o modificación de referencia. La referencia de datos simple se utiliza para hacer referencia a un único elemento elemental o de grupo.

Formato



data-name-1

Puede ser cualquier entrada de descripción de datos.

data-name-1 debe ser exclusivo en un programa.

Identificadores

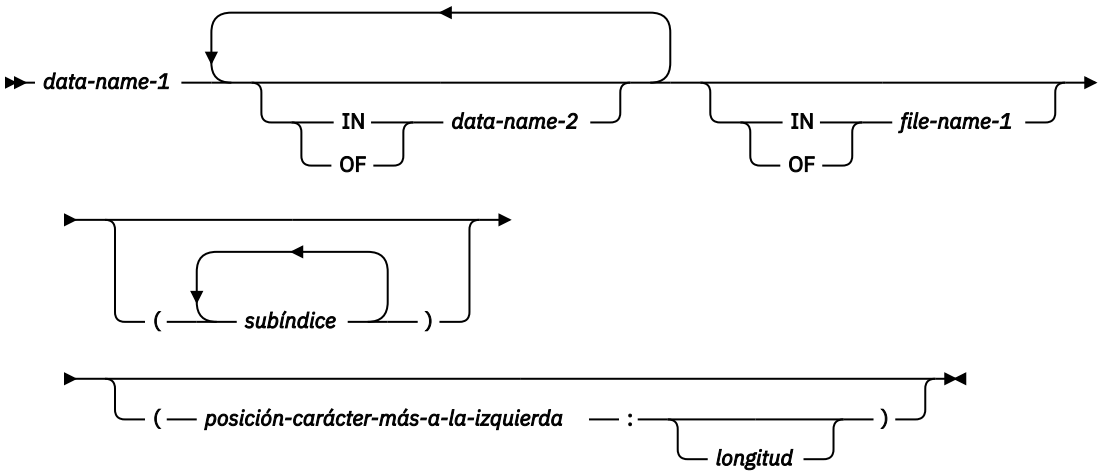
Cuando se utiliza en un diagrama de sintaxis en esta información, el término *identificador* hace referencia a una combinación válida de un nombre de datos o identificador de función con sus calificadores, subíndices y modificadores de referencia según sea necesario para la exclusividad de referencia.

Sin embargo, las reglas para los identificadores asociados a un formato pueden prohibir específicamente la calificación, la suscripción o la modificación de referencia.

El término *nombre-datos* hace referencia a un nombre que no debe ser calificado, subtitulado o de referencia modificado a menos que lo permitan específicamente las reglas para el formato.

- Para obtener una descripción de la cualificación, consulte "Cualificación" en la página 57.
- Para obtener una descripción de la suscripción, consulte "Suscripción" en la página 63.
- Para obtener una descripción de la modificación de referencia, consulte "Modificación de referencia" en la página 65.

Formato 1

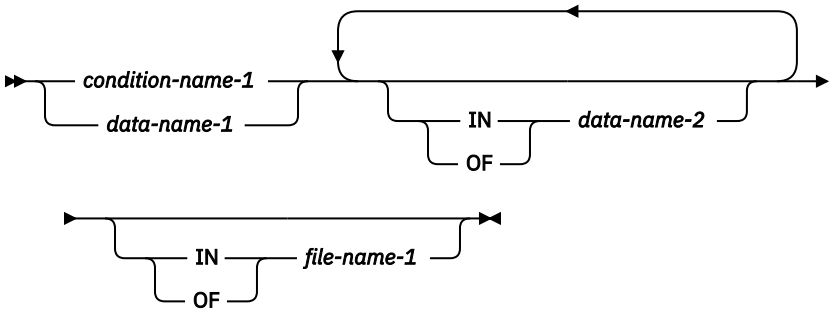


data-name-1 , data-name-2
Puede ser un nombre de registro.

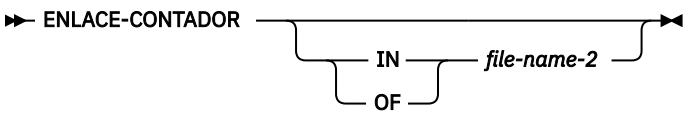
file-name-1
Debe identificarse mediante una entrada FD o SD en DATA DIVISION.

file-name-1 debe ser exclusivo dentro de este programa.

Formato 2



Formato 3



data-name-1 , data-name-2
Puede ser un nombre de registro.

condition-name-1
Las sentencias y entradas pueden hacer referencia a ellas en el programa que contiene la sección de configuración o en un programa contenido en dicho programa.

file-name-1
Debe identificarse mediante una entrada FD o SD en DATA DIVISION.

Debe ser exclusivo dentro de este programa.

ENLACE-CONTADOR
Debe calificarse cada vez que se haga referencia a ella si se ha especificado más de una entrada de descripción de archivo que contiene una cláusula LINAGE en la unidad fuente.

file-name-2
Debe identificarse mediante la entrada FD o SD en DATA DIVISION. *file-name-2* debe ser exclusivo dentro de este programa.

La duplicación de nombres de datos no debe producirse en aquellos lugares en los que los nombres de datos no pueden ser exclusivos por cualificación.

En el mismo programa, el nombre de datos especificado como sujeto de la entrada cuyo número de nivel es 01 que incluye la cláusula EXTERNAL no debe ser el mismo nombre de datos especificado para cualquier otra entrada de descripción de datos que incluya la cláusula EXTERNAL.

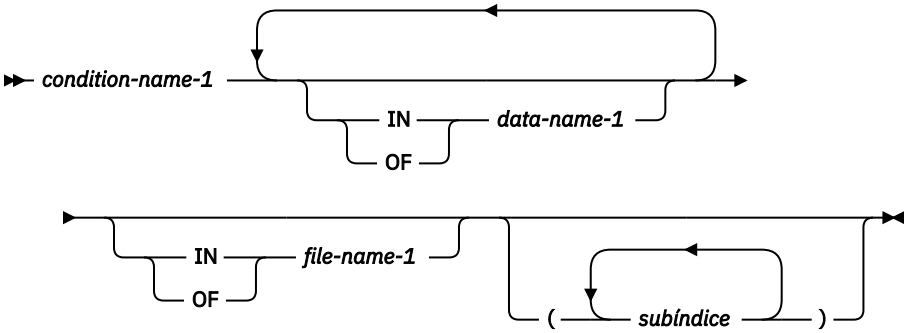
En la misma DATA DIVISION, las entradas de descripción de datos para dos elementos de datos cualesquiera para los que se ha especificado el mismo nombre de datos no deben incluir la cláusula GLOBAL.

Los nombres de DATA DIVISION a los que se hace referencia explícitamente deben estar definidos de forma exclusiva o ser exclusivos a través de la calificación. No es necesario que los elementos de datos no referenciados se definan de forma exclusiva. El nivel más alto de una jerarquía de datos (un elemento de datos asociado con un indicador de nivel (FD o SD en FILE SECTION) o con el número de nivel 01) debe tener un nombre exclusivo si se hace referencia a él. Los elementos de datos asociados con los números de nivel 02 a 49 son sucesivamente niveles inferiores de la jerarquía.

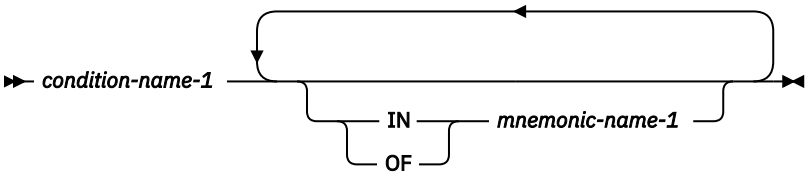
Nombre-condición

Consulte la sintaxis y la descripción para obtener más detalles.

Formato 1: nombre-condición en división de datos



Formato 2: nombre-condición en el párrafo SPECIAL-NAMES



condition-name-1
Las sentencias y entradas pueden hacer referencia a ellas en el programa que contiene la definición de *condition-name-1*, o en un programa contenido en dicho programa.

Si se hace referencia de forma explícita, un nombre de condición debe ser exclusivo o debe ser exclusivo a través de la calificación o subscripción (o ambos), excepto cuando el ámbito de los nombres por sí mismo garantiza la exclusividad de la referencia.

Si se utiliza la cualificación para hacer que un nombre de condición sea exclusivo, la variable condicional asociada se puede utilizar como primer calificador. Si se utiliza la cualificación, se debe utilizar la jerarquía de nombres asociada con la propia variable condicional para que el nombre de condición sea exclusivo.

Si las referencias a una variable condicional requieren subscripción, la referencia a cualquiera de sus nombres de condición también requiere la misma combinación de subscripción.

En esta información, *nombre-condición* hace referencia a un nombre-condición calificado o subtitulado, según sea necesario.

data-name-1
Puede ser un nombre de registro.

file-name-1
Debe identificarse mediante una entrada FD o SD en DATA DIVISION.

file-name-1 debe ser exclusivo dentro de este programa.

mnemonic-name-1
Para obtener información sobre los valores aceptables para *mnemonic-name-1*, consulte "Párrafo SPECIAL-NAMES" en la página 97.

Nombre de índice

Un nombre de índice identifica un índice. Un índice puede considerarse como un registro especial privado que el compilador genera para trabajar con una tabla. Puede nombrar un índice especificando la frase INDEXED BY en la cláusula OCCURS que define una tabla.

Puede utilizar un nombre de índice sólo en los siguientes elementos de lenguaje:

- Sentencias SET
- Sentencias PERFORM
- sentencias SEARCH
- Subíndices
- Condiciones de relación

Un nombre de índice no es el mismo que el nombre de un elemento de datos de índice, y un nombre de índice no se puede utilizar como un nombre de datos.

Elemento de datos de índice

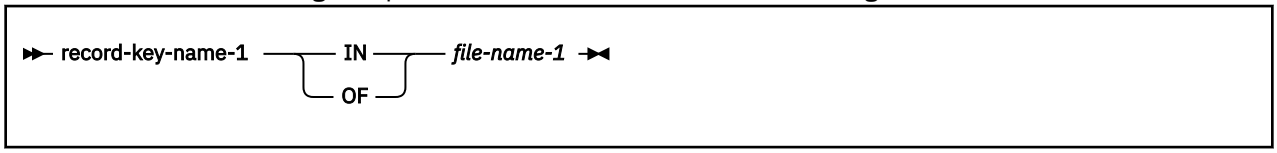
Un elemento de datos de índice es un elemento de datos que puede contener el valor de un índice.

Puede definir un elemento de datos de índice especificando la cláusula USAGE IS INDEX en una entrada de descripción de datos. El nombre de un elemento de datos de índice es un nombre de datos. Un elemento de datos de índice se puede utilizar en cualquier lugar donde se pueda utilizar un nombre de datos o un identificador, a menos que se indique lo contrario en las reglas de una sentencia determinada. Puede utilizar la sentencia SET para guardar el valor de un índice (al que hace referencia el nombre de índice) en un elemento de datos de índice.

Nombre-clave-registro

Un nombre de clave de registro es una palabra definida por el usuario que nombra una clave asociada a un archivo indexado.

Un nombre de clave de registro puede calificarse utilizando la sintaxis siguiente:

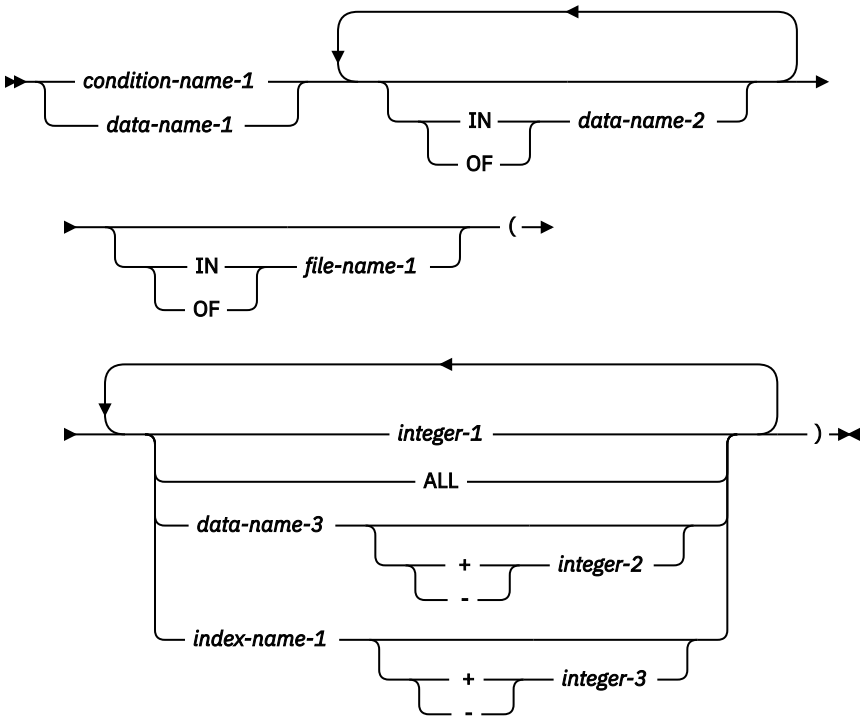


Restricción: El nombre de clave de registro sólo está soportado para el sistema de archivos STL.

Suscripción

Suscripción es un método para proporcionar referencias de tabla mediante el uso de subíndices. Un *subíndice* es un entero positivo cuyo valor especifica el número de aparición de un elemento de tabla.

Formato



condition-name-1
La variable condicional para *condition-name-1* debe contener una cláusula OCCURS o debe estar subordinada a una entrada de descripción de datos que contenga una cláusula OCCURS.

data-name-1
Debe contener una cláusula OCCURS o debe estar subordinada a una entrada de descripción de datos que contenga una cláusula OCCURS.

data-name-2 , file-name-1
Debe nombrar elementos de datos o registros que contengan *data-name-1*.

integer-1
Se puede firmar. Si se firma, debe ser positivo.

all
No se especificará si se especifica *condition-name-1* .

Sólo se debe utilizar cuando se utiliza el identificador de subscripción como argumento de función intrínseca o para identificar una tabla en una sentencia SORT de formato 2 (sentencia SORT de tabla).

Si se especifica ALL, el subíndice es todos los valores posibles de un subíndice para la tabla asociada tal como se especifica en las reglas para las funciones para las que se permite el subíndice ALL.

data-name-3
Debe ser un elemento elemental numérico que represente un entero.

data-name-3 se puede calificar. *data-name-3* no puede ser un campo de fecha con ventana.

index-name-1
Corresponde a una entrada de descripción de datos en la jerarquía de la tabla a la que se hace referencia que contiene una frase INDEXED BY que especifica ese nombre.

integer-2 , integer-3
No se puede firmar.

Los subíndices, entre paréntesis, se escriben inmediatamente después de cualquier calificación para el nombre del elemento de tabla. El número de subíndices de una referencia de este tipo debe ser igual al número de dimensiones de la tabla a cuyo elemento se hace referencia. Es decir, debe haber un subíndice para cada cláusula OCCURS en la jerarquía que contiene el nombre de datos incluido el propio nombre de datos.

Cuando se necesita más de un subíndice, se escriben en el orden de las dimensiones menos inclusivas de la organización de datos. Si una tabla multidimensional se considera como una serie de tablas anidadas y la tabla más inclusiva o más externa del nido se considera la tabla principal, siendo la tabla más interna o menos inclusiva la tabla menor, los subíndices se escriben de izquierda a derecha en el orden mayor, intermedio y menor.

Por ejemplo, si TABLE-THREE se define como:

```
01  TABLE-THREE.
    05  ELEMENT-ONE OCCURS 3 TIMES.
        10  ELEMENT-TWO OCCURS 3 TIMES.
            15  ELEMENT-THREE OCCURS 2 TIMES    PIC X(8).
```

una referencia de suscripción válida a TABLE-THREE es:

```
ELEMENT-THREE (2 2 1)
```

Las referencias con subíndice también se pueden modificar con referencia. Consulte el tercer ejemplo en “Ejemplos de modificación de referencia” en la página 67. Una referencia a un elemento no debe tener subíndice a menos que el elemento sea un elemento de tabla o un elemento o un nombre de condición asociado a un elemento de tabla.

Cada referencia de elemento de tabla debe tener un subíndice, excepto cuando aparezca dicha referencia:

- En una sentencia USE FOR DEBUGGING
- Como asunto de una sentencia SEARCH
- En una cláusula REDEFINES
- En la frase KEY IS de una cláusula OCCURS
- En una sentencia SORT de formato 2 (sentencia SORT de tabla)
- En una cláusula LIKE

En una sentencia SORT de formato 2, se puede especificar la suscripción con el subíndice situado más a la derecha que es la palabra ALL.

El número de aparición más bajo permitido representado por un subíndice es 1. El número de aparición más alto permitido en cualquier caso concreto es el número máximo de apariciones del elemento tal como se especifica en la cláusula OCCURS.

Suscripción utilizando nombres de datos

Cuando se utiliza un nombre de datos para representar un subíndice, se puede utilizar para hacer referencia a elementos dentro de distintas tablas. Estas tablas no necesitan tener elementos del mismo tamaño. El mismo nombre de datos puede aparecer como el único subíndice con un elemento y como uno de dos o más subíndices con otro elemento. Un subíndice de nombre de datos puede calificarse;

no puede ser subíndice o indexado. Por ejemplo, las referencias con subíndice válidas a TABLE-THREE, suponiendo que SUB1, SUB2y SUB3 son todos elementos subordinados a SUBSCRIPT-ITEM, incluyen:

```
ELEMENT-THREE (SUB1 SUB2 SUB3)

ELEMENT-THREE IN TABLE-THREE (SUB1 OF SUBSCRIPT-ITEM,
    SUB2 OF SUBSCRIPT-ITEM, SUB3 OF SUBSCRIPT-ITEM)
```

Suscripción utilizando nombres de índice (indexación)

La indexación permite operaciones como la búsqueda de tablas y la manipulación de elementos específicos. Para utilizar la indexación, asocie uno o más nombres de índice con un elemento cuya entrada de descripción de datos contenga una cláusula OCCURS.

Un índice asociado con un nombre de índice actúa como un subíndice, y su valor corresponde a un número de aparición para el elemento al que está asociado el nombre de índice.

La frase INDEXED BY, mediante la cual se identifica el nombre-índice y se asocia con su tabla, es una parte opcional de la cláusula OCCURS. No hay ninguna entrada separada para describir el índice asociado con el nombre-índice. En tiempo de ejecución, el contenido del índice corresponde a un número de aparición para esa dimensión específica de la tabla con la que está asociado el índice.

El valor inicial de un índice en tiempo de ejecución no está definido y el índice debe inicializarse antes de que se utilice como subíndice. Se asigna un valor inicial a un índice con una de las sentencias siguientes:

- La sentencia PERFORM con la frase VARYING
- La sentencia SEARCH con la frase ALL
- La sentencia SET

El uso de un entero o nombre de datos como un subíndice que hace referencia a un elemento de tabla o a un elemento dentro de un elemento de tabla no provoca la alteración de ningún índice asociado a dicha tabla.

Se puede utilizar un nombre de índice para hacer referencia a cualquier tabla. Sin embargo, la longitud de elemento de la tabla a la que se hace referencia y de la tabla con la que está asociado el nombre-índice debe coincidir. De lo contrario, la referencia no será al mismo elemento de tabla en cada tabla, y es posible que obtenga errores de tiempo de ejecución.

A menudo se busca en los datos que se organizan en forma de tabla. La sentencia SEARCH proporciona recursos para producir búsquedas en serie y no en serie. Se utiliza para buscar un elemento de tabla que satisfaga una condición específica y para ajustar el valor del índice asociado para indicar ese elemento de tabla.

Para que sea válido durante la ejecución, un valor de índice debe corresponder a una aparición de elemento de tabla de no menos de uno, ni mayor que el número de aparición más alto permitido.

Para obtener más información sobre los nombres de índice, consulte “Nombre de índice” en la página 62 y “frase INDEXED BY” en la página 187.

Subscripción relativa

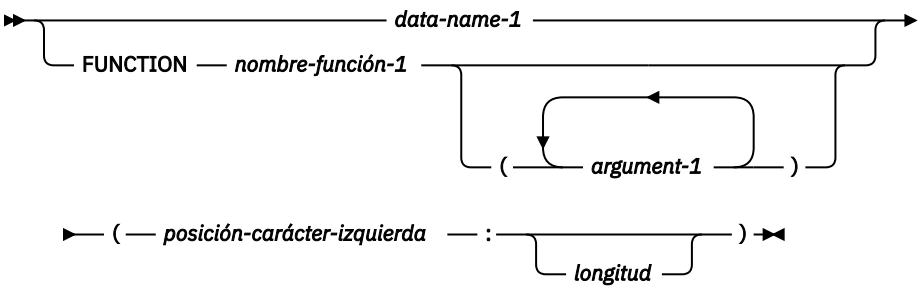
En *suscripciones relativas*, el nombre de un elemento de tabla va seguido de un subíndice con el formato nombre-datos o nombre-índice seguido del operador + o-, y de un literal entero positivo o sin signo.

Los operadores + y-deben ir precedidos y seguidos de un espacio. El valor del subíndice utilizado es el mismo que si el nombre de índice o el nombre de datos se hubiera configurado o desactivado por el valor del entero. El uso de la indexación relativa no hace que el programa altere el valor del índice.

Modificación de referencia

Modificación de referencia define un elemento de datos especificando un carácter situado más a la izquierda y una longitud opcional para el elemento de datos.

Formato: modificación de referencia



data-name-1
Debe hacer referencia a un elemento de datos descrito explícita o implícitamente con el uso DISPLAY, DISPLAY-1o NATIONAL. Un elemento de grupo nacional se procesa como un elemento de datos elemental de categoría nacional.

data-name-1 puede estar calificado o subtitulado. *data-name-1* no puede ser un campo de fecha con ventana.

data-name-1 no debe hacer referencia a un elemento de datos booleano.

data-name-1 no debe hacer referencia a un elemento definido utilizando la cláusula TYPE.

posición-carácter-más-a-la-izquierda
Debe ser una expresión aritmética. La evaluación de *posición de carácter más a la izquierda* debe dar como resultado un entero no cero positivo que sea menor o igual que el número de caracteres en el elemento de datos al que hace referencia *data-name-1*.

La evaluación de la *posición de carácter más a la izquierda* no debe dar como resultado un campo de fecha con ventana.

longitud
Debe ser una expresión aritmética.

La evaluación de *longitud* debe dar como resultado un entero positivo distinto de cero.

La evaluación de *longitud* no debe dar como resultado un campo de fecha con ventana.

La suma de *posición de carácter más a la izquierda* y *longitud* menos el valor 1 debe ser menor o igual que el número de posiciones de carácter en *data-name-1*. Si se omite *length* , la longitud utilizada será igual al número de posiciones de caracteres en *data-name-1* más 1, menos *posición-carácter-izquierda*.

function-name-1
Debe hacer referencia a una función alfanumérica o nacional.

Para los usos DISPLAY-1 y NATIONAL, cada posición de carácter ocupa 2 bytes. La modificación de referencia opera en posiciones de caracteres enteros y no en los bytes individuales de los caracteres en los usos DISPLAY-1 y NATIONAL. Para el uso DISPLAY, la modificación de referencia funciona como si cada carácter fuera un carácter de un solo byte.

A menos que se especifique lo contrario, se permite la modificación de referencia en cualquier lugar en el que se permita un identificador o identificador de función que haga referencia a un elemento o función de datos con el mismo uso que el elemento de datos modificado por referencia.

A cada posición de carácter referenciada por *data-name-1* o *function-name-1* se le asigna un número ordinal que se incrementa en uno desde la posición más a la izquierda a la posición más a la derecha. A la posición más a la izquierda se le asigna el número ordinal uno. Si la entrada de descripción de datos para *data-name-1* contiene una cláusula SIGN IS SEPARATE, a la posición de signo se le asigna un número ordinal dentro de ese elemento de datos.

Si *data-name-1* se describe con el uso DISPLAY y la categoría numérica, numérica editada, alfabética, alfanumérica editada o externa de coma flotante, *data-name-1* se utiliza con fines de modificación de referencia como si se hubiera redefinido como un elemento de datos de categoría alfanumérica con el mismo tamaño que el elemento de datos al que hace referencia *data-name-1*.

Si *data-name-1* se describe con el uso NATIONAL y la categoría numérico, editado-numérico, editado-nacional o coma flotante externo, *data-name-1* se utiliza para fines de modificación de referencia como si se hubiera redefinido como un elemento de datos de categoría nacional con el mismo tamaño que el elemento de datos al que hace referencia *data-name-1*.

Si *data-name-1* es un elemento de grupo nacional, *data-name-1* se procesa como un elemento de datos elemental de categoría nacional.

Si *data-name-1* es un campo de fecha expandido, el resultado de la modificación de referencia es una no fecha.

La modificación de referencia crea un elemento de datos exclusivo que es un subconjunto de *data-name-1* o un subconjunto del valor al que hace referencia *function-name-1* y sus argumentos, si los hay. Este elemento de datos exclusivo se considera un elemento de datos elemental sin la cláusula JUSTIFIC.

Cuando una función se modifica por referencia, el elemento de datos exclusivo tiene clase, categoría y uso nacional si el tipo de la función es nacional; de lo contrario, tiene clase y categoría alfanumérica y visualización de uso.

Cuando *data-name-1* se modifica por referencia, el elemento de datos exclusivo tiene la misma clase, categoría y uso que el definido para el elemento de datos al que hace referencia *data-name-1* , excepto que:

- Si *data-name-1* tiene la categoría nacional editada, el elemento de datos exclusivo tiene la categoría nacional.
- Si *data-name-1* tiene uso de NATIONAL y de categoría numérica editada, numérica o de coma flotante externa, el elemento de datos exclusivo tiene categoría nacional.
- Si *data-name-1* tiene el uso DISPLAY y la categoría numérica editada, alfanumérica editada, numérica o de coma flotante externa, el elemento de datos exclusivo tiene la categoría alfanumérica.
- Si *data-name-1* hace referencia a un elemento de grupo alfanumérico, se considera que el elemento de datos exclusivo tiene el uso DISPLAY y la categoría alfanumérica.
- Si *data-name-1* hace referencia a un elemento de grupo nacional, el elemento de datos exclusivo tiene el uso NATIONAL y la categoría nacional.

Si no se especifica *length* , el elemento de datos exclusivo creado se extiende desde e incluye la posición de carácter identificada por *posición de carácter más a la izquierda* hasta la posición de carácter más a la derecha del elemento de datos referenciado por *data-name-1*.

Evaluación de operandos

La modificación de referencia para un operando se evalúa de la siguiente manera:

- Si se especifica la suscripción para el operando, la modificación de referencia se evalúa inmediatamente después de la evaluación del subíndice.
- Si no se especifica la suscripción para el operando, la modificación de referencia se evalúa en el momento en que se evaluaría la suscripción si se hubieran especificado subíndices.

Ejemplos de modificación de referencia

Las sentencias de los ejemplos transfieren los primeros 10 caracteres del elemento de datos al que hace referencia WHOLE-NAME al elemento de datos al que hace referencia FIRST name.

```
77  WHOLE-NAME  PIC X(25).
77  FIRST-NAME  PIC X(10).

77  START-P     PIC 9(4) BINARY VALUE 1.
```

```
77  STR-LENGTH  PIC 9(4) BINARY VALUE 10.

...
    MOVE WHOLE-NAME(1:10) TO FIRST-NAME.
    MOVE WHOLE-NAME(START-P:STR-LENGTH) TO FIRST-NAME.
```

La sentencia siguiente transfiere los últimos 15 caracteres del elemento de datos referenciado por WHOLE-NAME al elemento de datos referenciado por LAST-NAME.

```
77  WHOLE-NAME  PIC X(25).
77  LAST-NAME   PIC X(15).
...
    MOVE WHOLE-NAME(11:) TO LAST-NAME.
```

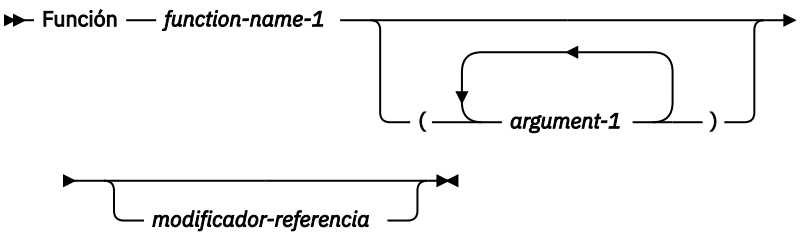
La sentencia siguiente transfiere los caracteres cuarto y quinto de la tercera aparición de TAB a la variable SUFFIX.

```
01  TABLE-1.
    02  TAB  OCCURS 10 TIMES  PICTURE X(5).
77  SUFFIX                   PICTURE X(2).
...
    MOVE TAB OF TABLE-1 (3) (4:2) TO SUFFIX.
```

Identificador-función

Un *identificador-función* es una secuencia de series de caracteres y separadores que hace referencia de forma exclusiva al elemento de datos que resulta de la evaluación de una función.

Formato



argument-1
Debe ser un identificador, literal (que no sea una constante figurativa) o expresión aritmética.

Para obtener más información, consulte Capítulo 20, “Funciones intrínsecas”, en la página 449.

function-name-1
function-name-1 debe ser uno de los nombres de función intrínseca.

modificador-referencia
Sólo se puede especificar para funciones de tipo alfanumérico o nacional.

Un identificador de función que haga referencia a una función alfanumérica o nacional puede especificarse en cualquier lugar en el que pueda hacerse referencia a un elemento de datos de categoría alfanumérica o de categoría nacional, respectivamente, y en el que las referencias a funciones no estén específicamente prohibidas, excepto las siguientes:

- Como operando receptor de cualquier sentencia
- Cuando un elemento de datos deba tener características particulares (tales como clase y categoría, tamaño, signo y valores permisibles) y la evaluación de la función de acuerdo con su definición y los argumentos particulares especificados no tendrían estas características

Un identificador de función que hace referencia a una función entera o numérica se puede utilizar siempre que se pueda utilizar una expresión aritmética.

Referencias a elementos de fecha-hora

Una función de fecha-hora puede especificarse en cualquier lugar de los formatos generales en los que se permite un identificador de fecha-hora, y en los que las normas asociadas a los formatos generales no prohíben específicamente la referencia a las funciones, excepto las siguientes:

- Como operando receptor de cualquier sentencia
- Cuando las normas asociadas a los formatos generales requieran que el elemento de datos al que se hace referencia tenga características particulares (tales como clase y categoría, uso, tamaño y valores permisibles) y la evaluación de la función de acuerdo con su definición y los argumentos específicos especificados no tendrían estas características.

Se puede hacer referencia a una función de fecha-hora como argumento para una función que permite un argumento de fecha-hora.

Referencias a elementos booleanos

Una función booleana se puede especificar en cualquier lugar de los formatos generales en los que se permite un identificador booleano, y en los que las reglas asociadas con los formatos generales no prohíben específicamente la referencia a funciones, excepto de la siguiente manera:

- Como operando receptor de cualquier sentencia
- Cuando las normas asociadas a los formatos generales requieran que el elemento de datos al que se hace referencia tenga características particulares (tales como clase y categoría, uso, tamaño y valores permisibles) y la evaluación de la función de acuerdo con su definición y los argumentos específicos especificados no tendrían estas características.

Se puede hacer referencia a una función booleana como argumento para una función que permite un argumento booleano.

Tipos de datos definidos por el usuario

Un tipo de datos definido por el usuario (o nombre de tipo) es un elemento elemental o de grupo de nivel 01 que contiene la cláusula TYPEDEF. No se ha asignado almacenamiento para un artículo de este tipo. Se puede considerar como una plantilla que describe un nombre de datos y sus elementos subordinados.

A continuación, se puede utilizar un nombre de tipo para definir un nombre de datos (u otro nombre de tipo) especificándolo dentro de una cláusula TYPE. El nombre de datos definido tendrá las características del nombre de tipo especificado en la cláusula TYPE. Si el nombre de tipo es un elemento de grupo, el nombre de datos definido será un elemento de grupo con elementos subordinados que tengan los mismos nombres, jerarquía y características que los elementos subordinados al nombre de tipo.

Al definir un nombre de datos (o nombre de tipo) utilizando un tipo de datos definido por el usuario en una cláusula TYPE, sólo se pueden utilizar las cláusulas siguientes junto con la cláusula TYPE para completar la descripción del nombre de datos:

- cláusula EXTERNAL
- cláusula GLOBAL
- cláusula OCCURS
- cláusula TYPEDEF
- Cláusula VALUE.

Las reglas de ámbito para los nombres de tipo son las mismas que para los nombres de datos.

Para obtener más información sobre las cláusulas TYPE y TYPEDEF, consulte “cláusula TYPE” en la página 223 y “cláusula TYPEDEF” en la página 224.

cláusula TYPE

La cláusula TYPE permite utilizar un tipo de datos definido por el usuario (o nombre de tipo) para definir un elemento de datos. Esto se realiza especificando el nombre de tipo (que se declara utilizando la cláusula TYPEDEF) en una cláusula TYPE. Si el nombre de tipo es un elemento de grupo, el elemento de datos definido también será un elemento de grupo: sus entradas subordinadas corresponderán en nombre, jerarquía y características a las subordinadas al nombre de tipo.

Formato

type-name-1
El nombre del nombre de tipo que se va a utilizar para definir el nombre de datos de asunto.

cláusula TYPEDEF

La cláusula TYPEDEF declara que un elemento de datos elemental o de grupo es un tipo de datos definido por el usuario (o nombre de tipo). Una vez que se ha definido el nombre de tipo, se puede utilizar (en una cláusula TYPE) para definir otros elementos de datos.

Formato

Especificación de atributo de datos

Los *atributos de datos explícitos* son atributos de datos que se especifican en la codificación COBOL. Los *atributos de datos implícitos* son valores predeterminados. Si no codifica explícitamente un atributo de datos, el compilador asume un valor predeterminado.

Por ejemplo, no es necesario especificar la USAGE de un elemento de datos. Si se omite USAGE y no se especifica el símbolo N en la cláusula PICTURE, el valor predeterminado es USAGE DISPLAY, que es el atributo de datos implícito. Cuando se utiliza el símbolo N de PICTURE, USAGE DISPLAY-1 es el valor por omisión cuando la opción de compilador NSYMBOL (DBCS) está en vigor; USAGE NATIONAL es el valor por omisión cuando la opción de compilador NSYMBOL (NATIONAL) está en vigor. Estos son atributos de datos implícitos.

Capítulo 9. Transferencia de control

En PROCEDURE DIVISION, a menos que haya una transferencia de control *explícita* o no haya una siguiente sentencia ejecutable, el flujo de programa transfiere el control de sentencia a sentencia en el orden en el que se escriben las sentencias. Este flujo de programa normal es una transferencia de control *implícita* .

Además de las transferencias implícitas de control entre sentencias consecutivas, la transferencia implícita de control también se produce cuando el flujo normal se altera sin la ejecución de una sentencia de ramificación de procedimiento. Los ejemplos siguientes muestran transferencias de control *implícitas* , alterando temporalmente la transferencia de control de sentencia a sentencia:

- Después de la ejecución de la última sentencia de un procedimiento que se ejecuta bajo el control de otra sentencia COBOL, el control transfiere implícitamente. (Las sentencias COBOL que controlan la ejecución de procedimientos son, por ejemplo, MERGE, PERFORM, SORT y USE.) Además, si un párrafo se ejecuta bajo el control de una sentencia PERFORM que provoca la ejecución iterativa, y ese párrafo es el primer párrafo del rango de dicha sentencia PERFORM, se produce una transferencia implícita de control entre el mecanismo de control asociado a dicha sentencia PERFORM y la primera sentencia de dicho párrafo para cada ejecución iterativa del párrafo.
- Durante la ejecución de sentencias SORT o MERGE, el control se transfiere implícitamente a un procedimiento de entrada o salida.
- Durante la ejecución de la sentencia XML PARSE, el control se transfiere implícitamente a un procedimiento de proceso.
- Durante la ejecución de cualquier sentencia COBOL que provoque la ejecución de un procedimiento declarativo, el control se transfiere implícitamente a dicho procedimiento.
- Al final de la ejecución de cualquier procedimiento declarativo, el control se transfiere implícitamente de nuevo al mecanismo de control asociado con la sentencia que causó su ejecución.

COBOL también proporciona transferencias de control *explícitas* a través de la ejecución de cualquier ramificación de procedimiento, llamada de programa o sentencia condicional. (Las listas de ramificación de procedimiento y sentencias condicionales están contenidas en “Categorías de sentencia” en la página 275.)

Definición: El término *siguiente sentencia ejecutable* hace referencia a la siguiente sentencia COBOL a la que se transfiere el control, de acuerdo con las reglas indicadas anteriormente. No hay ninguna siguiente sentencia ejecutable en las siguientes circunstancias:

- Cuando el programa no contiene PROCEDURE DIVISION
- Después de la última sentencia en una sección declarativa cuando el párrafo en el que aparece no se está ejecutando bajo el control de alguna otra sentencia COBOL
- Después de la última sentencia en un programa cuando el párrafo en el que aparece no se está ejecutando bajo el control de alguna otra sentencia COBOL en ese programa
- Después de la última sentencia en una sección declarativa cuando la sentencia está en el rango de una sentencia PERFORM activa ejecutada en una sección diferente y esta última sentencia de la sección declarativa no es también la última sentencia del procedimiento que es la salida de la sentencia PERFORM activa
- Después de una sentencia STOP RUN o EXIT PROGRAM que transfiere el control fuera del programa COBOL
- Seguimiento de una sentencia GOBACK que transfiere el control fuera del programa COBOL
- El marcador de programa final

Cuando no hay una siguiente sentencia ejecutable y el control no se transfiere fuera del programa COBOL, el flujo de control del programa no está definido a menos que la ejecución del programa esté en la parte

de procedimientos no declarativos de un programa bajo el control de una sentencia CALL, en cuyo caso se ejecuta una sentencia EXIT PROGRAM implícita.

Capítulo 10. Millennium Language Extensions y campos de fecha

Las extensiones de lenguaje del milenio están diseñadas para resolver el problema del milenio causado por el uso de dos dígitos para representar el año en los campos de fecha de algunas aplicaciones, y para dar soporte a las operaciones más comunes en los campos de fecha.

Muchas aplicaciones utilizan dos dígitos en lugar de cuatro dígitos para representar el año en los campos de fecha, y asumen que estos valores representan años de 1900 a 1999. Este formato de fecha compacto funciona bien para 1900s, pero no funciona para el año 2000 y más allá porque estas aplicaciones interpretan "00" como 1900 en lugar de 2000, produciendo resultados incorrectos.

Las extensiones de lenguaje millennium están diseñadas para permitir que las aplicaciones que utilizan años de dos dígitos sigan funcionando correctamente en el año 2000 y más allá, con una modificación mínima del código existente. Esto se consigue utilizando una técnica conocida como *windowing*, que elimina la suposición de que todos los campos de año de dos dígitos representan años de 1900 a 1999. En su lugar, la ventana permite que los campos de año de dos dígitos representen los años dentro de un rango de 100 años conocido como una *ventana de siglo*.

Por ejemplo, si un campo de año de dos dígitos contiene el valor 15, muchas aplicaciones interpretarían el año como 1915. Sin embargo, con una ventana de siglo de 1960–2059, el año se interpretaría como 2015.

Las extensiones de lenguaje del milenio proporcionan soporte para las operaciones más comunes en los campos de fecha: comparaciones, movimiento y almacenamiento, y aumento y disminución. Este soporte está limitado a campos de fecha de determinados formatos; para obtener detalles, consulte “cláusula DATE FORMAT” en la página 172.

Para obtener información sobre las operaciones y restricciones soportadas al utilizar campos de fecha, consulte “Restricciones en el uso de campos de fecha” en la página 173.

Sintaxis de Millennium Language Extensions

Las extensiones de lenguaje millennium introducen elementos de lenguaje de la cláusula DATE FORMAT y algunas funciones intrínsecas.

- La cláusula DATE FORMAT en las entradas de descripción de datos, que define los elementos de datos como campos de fecha.
- Las siguientes funciones intrínsecas:

  FECHA DE FECHA
  Convierte una no fecha en un campo de fecha.

  FECHA
  Convierte un campo de fecha en un campo que no es de fecha.

  VENTANA
  Devuelve el primer año de la ventana de siglo especificada por la opción de compilador YEARWINDOW.

Para obtener detalles sobre cómo utilizar las extensiones de lenguaje del milenio en las aplicaciones, consulte *Millennium Language Extensions (MLE)* en la publicación *COBOL for Linux en x86 Guía de programación*.

Las extensiones de lenguaje millennium no tienen ningún efecto a menos que el programa se compile utilizando la opción de compilador DATEPROC y la ventana de siglo se especifique mediante la opción de compilador YEARWINDOW.

Términos y conceptos

Vea los siguientes términos cuando el documento se refiere a las extensiones de lenguaje del milenio.

- "Campo de fecha" en la página 74
- "No fecha" en la página 75
- "Ventana de siglo" en la página 75

Campo de fecha

Un *campo de fecha* puede ser uno de varios elementos.

Estos son los posibles campos de fecha:

- Elemento de datos cuya entrada de descripción de datos incluye una cláusula DATE FORMAT
- Un valor devuelto por una de las siguientes funciones intrínsecas:
  - FECHA-DE-ENTERO
  - DATE-TO-AAAAMMDD
  - DATEVAL
  - DÍA-OF-INTEGER
  - DAY-TO-AAAADDD
  - AÑO-A-AAAA
  - VENTANA
- Los elementos de datos conceptuales DATE, DATE YYYYMMDD, DAY o DAY YYYYDDD de la sentencia ACCEPT
- El resultado de determinadas operaciones aritméticas (para obtener detalles, consulte "Aritmética con campos de fecha" en la página 248)

El término *campo de fecha* hace referencia a los *campos de fecha expandidos* y a los *campos de fecha con ventana*.

Campo de fecha con ventana

Un *campo de fecha con ventana* es un campo de fecha que contiene un año con ventana. Un *año con ventana* consta de dos dígitos, que representan un año dentro de la ventana de siglo.

Campo de fecha expandida

Un *campo de fecha expandida* es un campo de fecha que contiene un año expandido. Un *año expandido* consta de cuatro dígitos.

El uso principal de los campos de fecha ampliados es proporcionar resultados correctos cuando se utilizan en combinación con campos de fecha con ventanas; por ejemplo, cuando la migración a fechas de año de cuatro dígitos no está completa. Si todas las fechas de una aplicación utilizan años de cuatro dígitos, no es necesario utilizar las extensiones de lenguaje del milenio.

Año-Última fecha, campo

Un *campo de fecha de último año* es un campo de fecha cuya cláusula DATE FORMAT especifica una o más X antes de AA o AAAA. Los campos de fecha-última fecha están soportados en un número limitado de operaciones, que normalmente implican otra fecha con el mismo formato de fecha (año-última) o una fecha distinta.

Formato de fecha

Formato de fecha hace referencia al patrón de fecha de un campo de fecha, especificado:

- Explícitamente, mediante la cláusula DATE FORMAT o la función intrínseca DATEVAL *argument-2*
- Implícitamente, por sentencias y funciones intrínsecas que devuelven campos de fecha.

Campo de fecha compatible

El significado del término *compatible*, cuando se aplica a campos de fecha, depende de la división COBOL en la que se produce el campo de fecha:

División de datos

Dos campos de fecha son compatibles si tienen USAGE idéntico y cumplen al menos una de las condiciones siguientes:

- Tienen el mismo formato de fecha.
- Ambos son campos de fecha con ventana, donde uno está formado únicamente por un año con ventana, DATE FORMAT YY.
- Ambos son campos de fecha expandidos, donde uno solo consta de un año expandido, DATE FORMAT YYYY.
- Uno tiene DATE FORMAT YYXXXX, el otro, YYXX.
- Uno tiene DATE FORMAT YYYYXXXX, el otro, YYYYXX.

Un campo de fecha con ventana puede estar subordinado a un elemento de datos de grupo de fechas expandido. Los dos campos de fecha son compatibles si el campo de fecha subordinado tiene USAGE DISPLAY, inicia 2 bytes después del inicio del campo de fecha expandida de grupo y los dos campos cumplen al menos una de las condiciones siguientes:

- El campo de fecha subordinado tiene un patrón DATE FORMAT con el mismo número de X que el patrón DATE FORMAT del campo de fecha de grupo.
- El campo de fecha subordinada tiene DATE FORMAT YY.
- El campo de fecha de grupo tiene DATE FORMAT YYYYXXXX y el campo de fecha subordinado tiene DATE FORMAT YYXX.

División de procedimiento

Dos campos de fecha son compatibles si tienen el mismo formato de fecha excepto para la parte del año, que puede ser con ventana o expandida. Por ejemplo, un campo de fecha con ventana con DATE FORMAT YYXXX es compatible con:

- Otro campo de fecha con ventana con DATE FORMAT YYXXX
- Un campo de fecha ampliado con DATE FORMAT YYYYXXX

No fecha

Una *no fecha* puede ser cualquiera de varios elementos.

Estas son las posibles no fechas:

- Un elemento de datos cuya entrada de descripción de fecha no incluye la cláusula DATE FORMAT
- Un campo de fecha que se ha convertido utilizando la función UNDATE
- Un literal
- Un campo de fecha de modificación de referencia
- El resultado de determinadas operaciones aritméticas que pueden incluir operandos de campo de fecha; por ejemplo, la diferencia entre dos campos de fecha compatibles

Ventana de siglo

Una *ventana de siglo* es un intervalo de 100 años dentro del cual cualquier año de dos dígitos es exclusivo.

Hay varias formas de especificar una ventana de siglo en un programa COBOL:

- Para campos de fecha con ventana, se especifica una ventana de siglo mediante la opción de compilador YEARWINDOW.
- Para las funciones intrínsecas de ventana DATE-TO-YYYYMMDD, DAY-TO-YYYYDDD y YEAR-TO-YYYY, se especifica una ventana de siglo mediante *argument-2*.

Parte 2. Estructura de unidad de origen COBOL

Capítulo 11. Estructura de programa COBOL

Un programa fuente COBOL es un conjunto sintácticamente correcto de sentencias COBOL.

Programas anidados
Un *programa anidado* es un programa contenido en otro programa. Los programas contenidos pueden hacer referencia a algunos de los recursos de los programas que los contienen. Si el programa B está contenido en el programa A, está *directamente* contenido si no hay ningún programa contenido en el programa A que también contenga el programa B. El programa B está *indirectamente* contenido en el programa A si existe un programa contenido en el programa A que también contiene el programa B. Para obtener más información sobre programas anidados, consulte *Programas anidados* en *COBOL for Linux en x86 Guía de programación*.

Programa de objeto
Un *programa objeto* es un conjunto o grupo de instrucciones de lenguaje de máquina ejecutable y otro material diseñado para interactuar con los datos para proporcionar soluciones de problemas. Un programa objeto es generalmente el resultado de lenguaje de máquina de la operación de una compilación COBOL en un programa fuente.

unidad de ejecución
Una *unidad de ejecución* es uno o más programas de objeto que interactúan entre sí y que funcionan en tiempo de ejecución como una entidad para proporcionar soluciones de problemas.

Programa hermano
Los *programas hermanos* son programas que están contenidos directamente en el mismo programa.

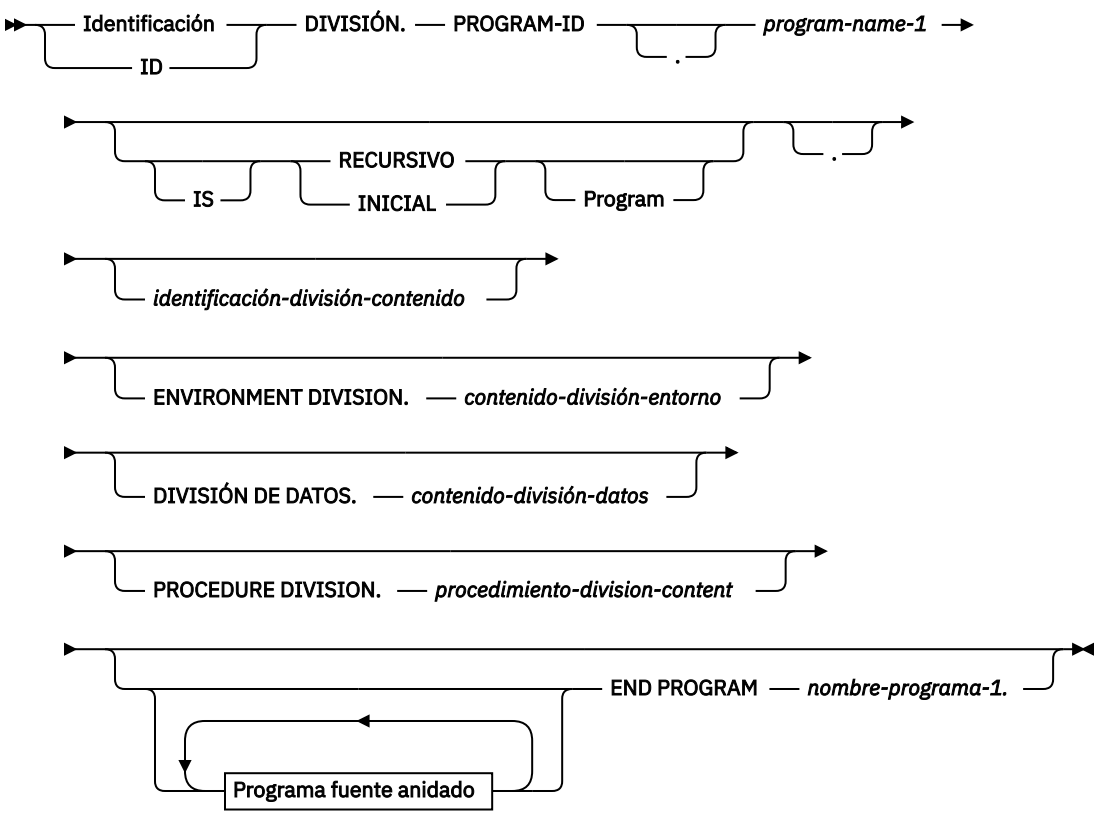
Con la excepción de las sentencias COPY y REPLACE y el marcador de programa final, las sentencias, entradas, párrafos y secciones de un programa fuente COBOL se agrupan en las cuatro divisiones siguientes:

- DIVISIÓN DE IDENTIFICACIÓN
- DIVISIÓN DE MEDIO AMBIENTE
- DIVISIÓN DE DATOS
- DIVISIÓN DE PROCEDIMIENTO

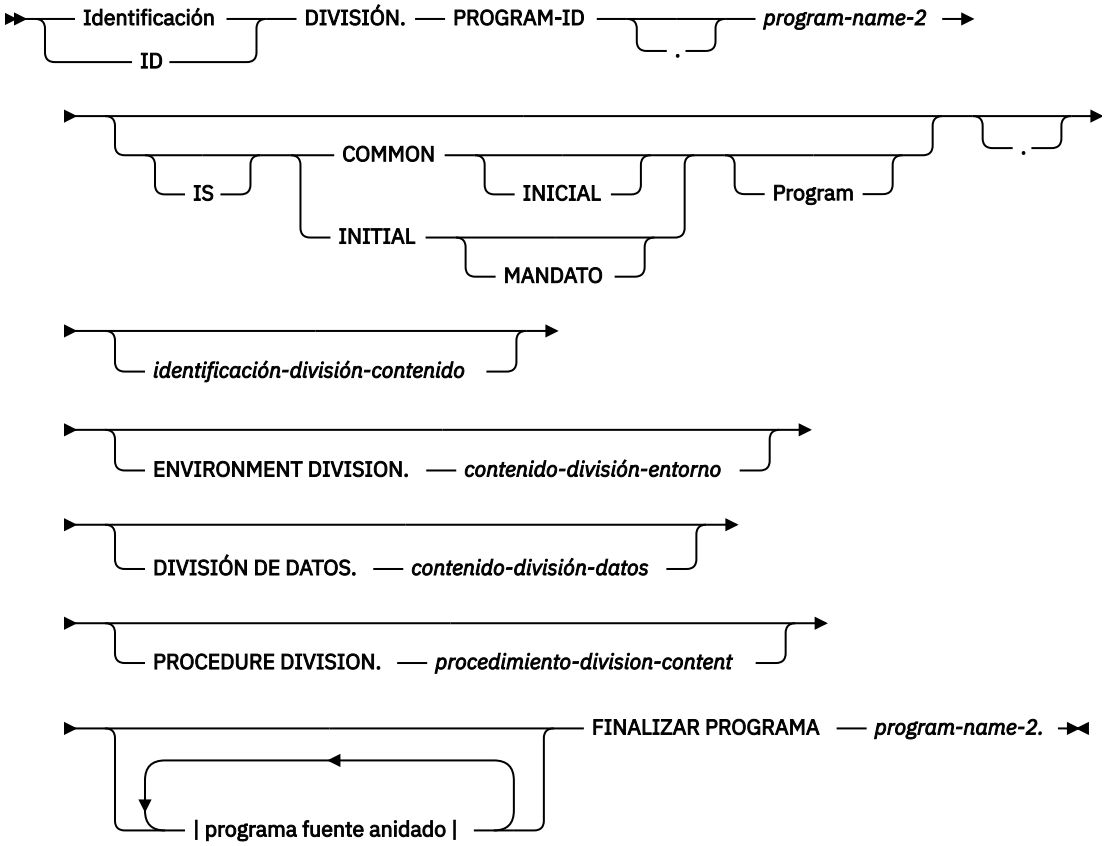
El final de un programa fuente COBOL se indica mediante el marcador END PROGRAM. Si no hay programas anidados, la ausencia de líneas de programa fuente adicionales también indica el final de un programa COBOL.

El formato siguiente es para las entradas y sentencias que constituyen un programa fuente COBOL compilado por separado.

Formato: Programa fuente COBOL

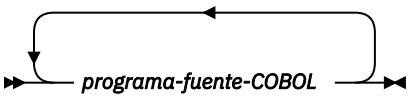


programa fuente anidado



Una secuencia de programas COBOL separados también se puede especificar en el compilador. El formato siguiente es para las entradas y sentencias que constituyen una secuencia de programas fuente (compilación por lotes).

Formato: secuencia de programas fuente COBOL



END PROGRAM ***nombre-programa***

Un marcador de programa final separa cada programa en la secuencia de programas. *nombre-programa* debe ser idéntico a un nombre-programa declarado en un párrafo de ID-programa anterior.

nombre-programa se puede especificar como una palabra definida por el usuario o en un literal alfanumérico. De cualquier forma, *nombre-programa* debe seguir las reglas para formar nombres de programa. *nombre-programa* no puede ser una constante figurativa. Las letras minúsculas del literal se convierten a mayúsculas.

Un marcador de programa final es opcional para el último programa de la secuencia sólo si dicho programa no contiene ningún programa fuente anidado.

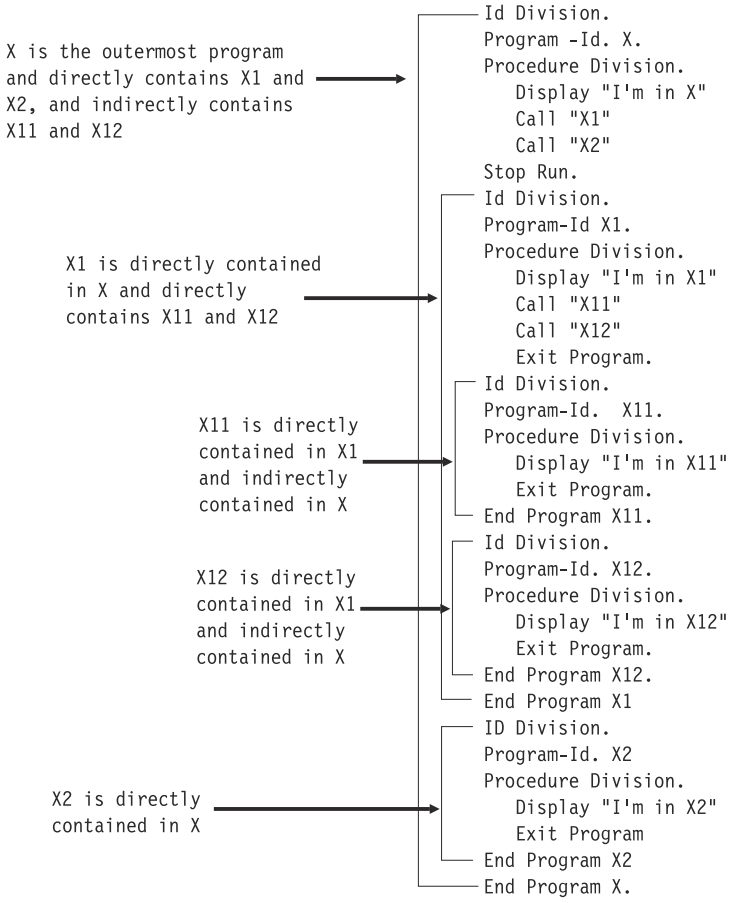
Programas anidados

Un programa COBOL puede contener otros programas COBOL, que a su vez pueden contener otros programas COBOL. Estos programas contenidos se denominan *programas anidados*. Los programas anidados pueden estar *directamente* o *indirectamente* contenidos en el programa contenedor.

En el fragmento de código siguiente, el programa `Outer-program` *directamente* contiene el programa `Inner-1`. El programa `Inner-1` *directamente* contiene el programa `Inner-1`ay `Outer-program` *indirectamente* contiene `Inner-1`a:

```
Id division.
Program-id. Outer-program.
  Procedure division.
    Call "Inner-1".
    Stop run.
Id division.
  Program-id. Inner-1
  ...
    Call Inner-1a.
    Stop run.
  Id division.
  Program-id. Inner-1a.
  ...
  End Inner-1a.
  End Inner-1.
End Outer-program.
```

La figura siguiente describe una estructura de programa anidado más compleja con programas contenidos directa e indirectamente.



Convenios para nombres de programa

El nombre de programa de un programa se especifica en el párrafo PROGRAM-ID de la DIVISIÓN DE IDENTIFICACIÓN del programa. Sólo se puede hacer referencia a un nombre de programa mediante la sentencia CALL, la sentencia CANCEL, la sentencia SET o el marcador END PROGRAM.

Los nombres de los programas que constituyen una unidad de ejecución no son necesariamente exclusivos, pero cuando dos programas de una unidad de ejecución tienen un nombre idéntico, al menos uno de los programas debe estar contenido directa o indirectamente dentro de otro programa compilado por separado que no contenga el otro de esos dos programas.

Un programa compilado por separado y todos sus programas contenidos directa e indirectamente deben tener nombres de programa exclusivos dentro de ese programa compilado por separado.

Reglas para nombres de programa

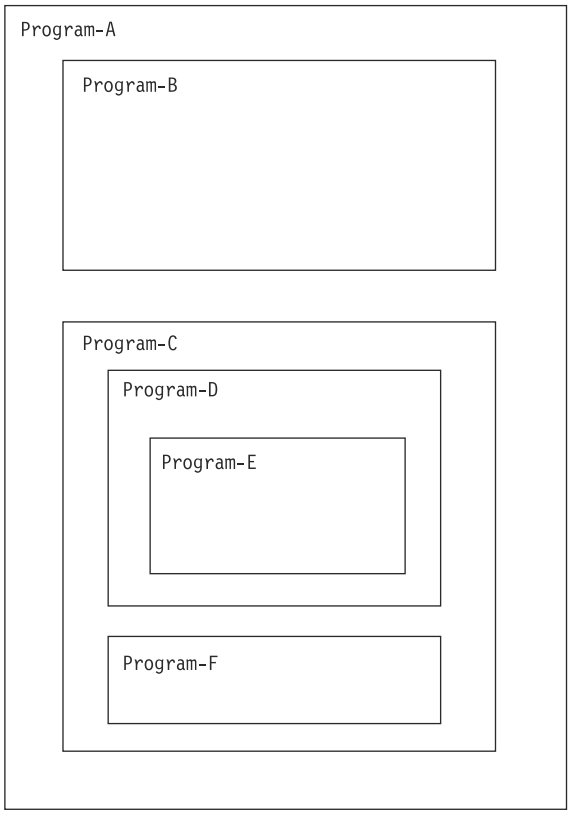
Las reglas siguientes definen el ámbito de un nombre de programa:

- Si el nombre de programa es el de un programa que no posee el atributo COMMON y ese programa está directamente contenido en otro programa, sólo se puede hacer referencia a ese nombre de programa mediante sentencias incluidas en el programa que lo contiene.
- Si el nombre de programa es el de un programa que posee el atributo COMMON y ese programa está contenido directamente dentro de otro programa, sólo se puede hacer referencia a ese nombre de programa mediante sentencias incluidas en el programa que lo contiene y cualquier programa contenido directa o indirectamente dentro de ese programa que lo contiene, excepto ese programa que posee el atributo COMMON y cualquier programa contenido en él.
- Si el nombre de programa es el de un programa que se compila por separado, se puede hacer referencia a ese nombre de programa mediante sentencias incluidas en cualquier otro programa de la unidad de ejecución, excepto programas que contenga directa o indirectamente.

El mecanismo utilizado para determinar a qué programa llamar es el siguiente:

- Si uno de los dos programas que tienen el mismo nombre que el especificado en la sentencia CALL está directamente contenido en el programa que incluye la sentencia CALL, se llama a ese programa.
- Si uno de los dos programas que tienen el mismo nombre que el especificado en la sentencia CALL posee el atributo COMMON y está directamente contenido dentro de otro programa que contiene directa o indirectamente el programa que incluye la sentencia CALL, se llama a ese programa común a menos que el programa de llamada esté contenido dentro de ese programa común.
- De lo contrario, se llama al programa compilado por separado.

Las reglas siguientes se aplican a la referencia a un nombre de programa de un programa contenido en otro programa. Para este debate, el programa A contiene el programa B y el programa C; el programa C contiene el programa D y el programa F; y el programa D contiene el programa E.



Si Program-D no posee el atributo COMMON, sólo se puede hacer referencia a Program-D mediante el programa que contiene directamente Program-D, es decir, Program-C.

Si Program-D posee el atributo COMMON, Program-D puede ser referenciado por Program-C (porque Program-C contiene Program-D) y por cualquier programa contenido en Program-C excepto para los programas contenidos en Program-D. En otras palabras, si Program-D posee el atributo COMMON, Program-D puede ser referenciado en Program-C y Program-F pero no por sentencias en Program-E, Program-A o Program-B.

Parte 3. División de identificación

Capítulo 12. DIVISIÓN DE IDENTIFICACIÓN

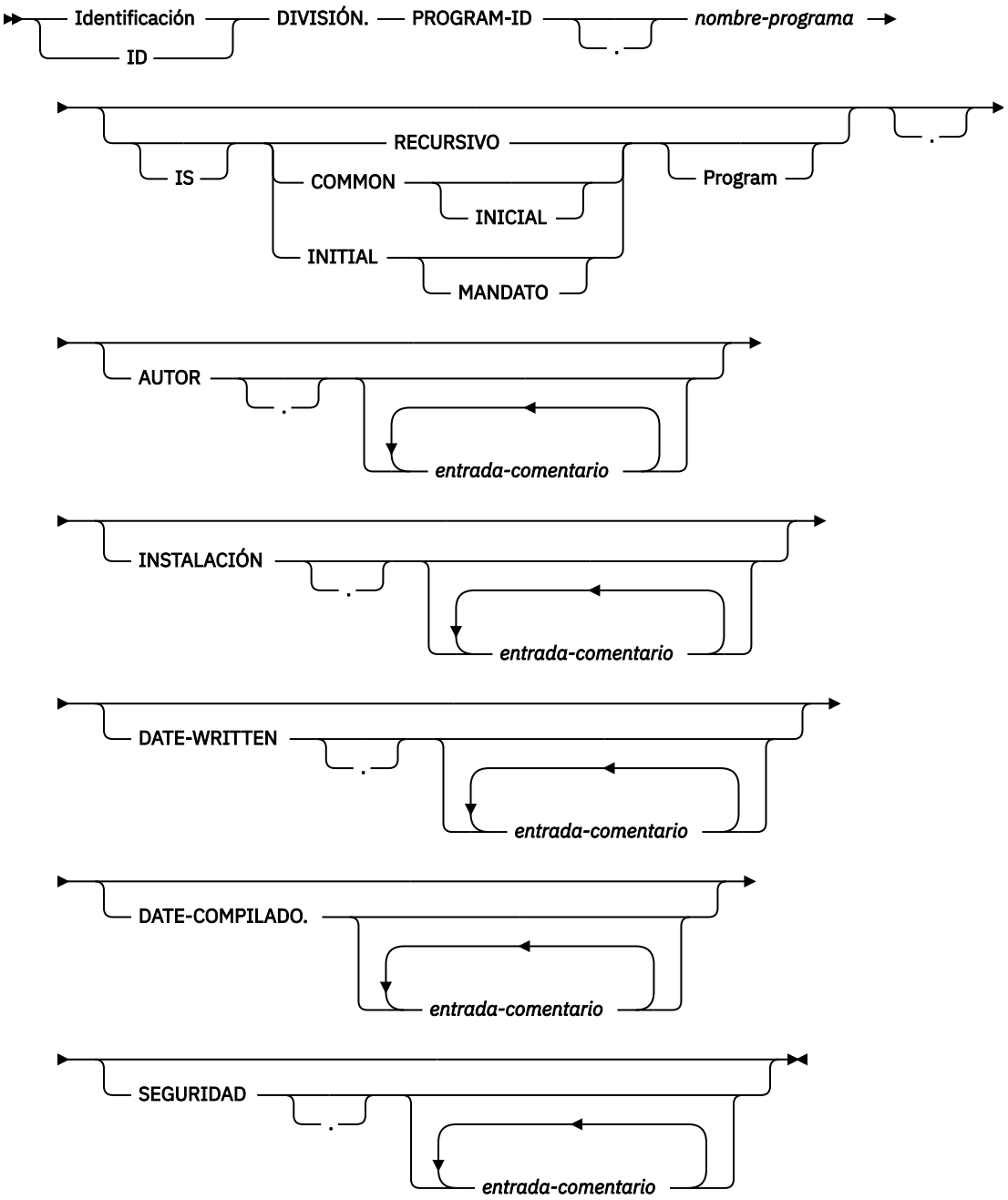
La DIVISIÓN DE IDENTIFICACIÓN debe ser la primera división en cada programa fuente COBOL. La DIVISIÓN DE IDENTIFICACIÓN puede incluir la fecha en que se ha escrito un programa , la fecha de compilación y otra información documental de este tipo.

DIVISIÓN DE IDENTIFICACIÓN DE PROGRAMA

Para un programa, el primer párrafo de la DIVISIÓN DE IDENTIFICACIÓN debe ser el párrafo PROGRAM-ID. Los otros párrafos son opcionales y pueden aparecer en cualquier orden.

El siguiente formato es para un programa DIVISION DE IDENTIFICACION.

Formato: división de identificación de programa



Párrafo PROGRAM-ID

El párrafo PROGRAM-ID especifica el nombre por el que se conoce el programa y asigna los atributos de programa seleccionados a dicho programa. Es obligatorio y debe ser el primer párrafo de la DIVISIÓN DE IDENTIFICACIÓN.

nombre-programa
Palabra definida por el usuario o literal alfanumérico, pero no una constante figurativa, que identifica el programa. Debe seguir las siguientes reglas de formación, en función del valor de la opción de compilador PGMNAME:

PGMNAME (LONGUPPER)
Si *nombre-programa* es una palabra definida por el usuario, puede tener hasta 30 caracteres de longitud.

Si *nombre-programa* es un literal alfanumérico, el literal puede tener hasta 160 caracteres de longitud. El literal no puede ser una constante figurativa.

Sólo se permiten el guión, el subrayado de , los dígitos del 0 al 9 de y los caracteres alfabéticos en el nombre cuando el nombre se especifica como una palabra definida por el usuario.

Al menos un carácter debe ser alfabético.

El guión no puede ser el primer o último carácter.

Si *nombre-programa* es un literal alfanumérico, el carácter de subrayado puede ser el primer carácter.

Los nombres de programa externos se procesan con caracteres alfabéticos en mayúsculas.

PGMNAME (LONGMIXED)
nombre-programa debe especificarse como un literal alfanumérico, que puede tener hasta 160 caracteres de longitud. El literal no puede ser una constante figurativa.

Siempre que se permitan caracteres alfabéticos, puede utilizar caracteres de varios bytes.

Para obtener información sobre la opción de compilador PGMNAME y cómo el compilador procesa los nombres, consulte *PGMNAME* en *COBOL for Linux en x86 Guía de programación*.

RECURSIVO
Cláusula opcional que permite que los programas COBOL se vuelvan a especificar de forma recursiva.

Puede especificar la cláusula RECURSIVE sólo en el programa más externo de una unidad de compilación. Los programas recursivos no pueden contener subprogramas anidados.

Si se especifica la cláusula RECURSIVE, *nombre-programa* se puede volver a especificar de forma recursiva mientras una invocación anterior sigue activa. Si no se especifica la cláusula RECURSIVE, no se puede volver a entrar de forma recursiva un programa activo.

La sección WORKING-STORAGE SECTION de un programa recursivo define el almacenamiento que se asigna e inicializa estáticamente en la primera entrada a un programa y está disponible en un estado de última utilización para cualquiera de las invocaciones recursivas.

La sección LOCAL-STORAGE SECTION de un programa recursivo (así como un programa no recursivo) define el almacenamiento que se asigna, inicializa y desasigna automáticamente por invocación.

Los conectores de archivo internos que corresponden a un FD en FILE SECTION de un programa recursivo se asignan estáticamente. El estado de los conectores de archivos internos forma parte del último estado utilizado de un programa que persiste entre invocaciones.

Los siguientes elementos de lenguaje no están soportados en un programa recursivo:

- ALTER
- GO TO sin un nombre de procedimiento especificado
- RERUN
- LÍMITE DE SEGMENTO
- USO PARA DEPURACIÓN

MANDATO
Especifica que el programa denominado por *nombre-programa* está contenido (es decir, anidado) dentro de otro programa y se puede llamar desde los hermanos del programa común y los programas contenidos en ellos. La cláusula COMMON sólo se puede utilizar en programas anidados. Para obtener más información sobre los convenios para nombres de programa, consulte “Convenios para nombres de programa” en la página 82.

INICIAL
Especifica que cuando se llama a *nombre-programa* , *nombre-programa* y los programas contenidos (anidados) en él se colocan en su estado inicial.

Un programa está en el estado inicial:

- La primera vez que se llama al programa en una unidad de ejecución
- Cada vez que se llama al programa, si posee el atributo inicial
- La primera vez que se llama al programa después de la ejecución de una sentencia CANCEL que hace referencia al programa o una sentencia CANCEL que hace referencia a un programa que contiene directa o indirectamente el programa
- La primera vez que se llama al programa después de la ejecución de una sentencia CALL que hace referencia a un programa que posee el atributo inicial y que contiene directa o indirectamente el programa

Cuando un programa está en el estado inicial:

- Se inicializan los datos internos del programa contenidos en WORKING-STORAGE SECTION. Si se utiliza una cláusula VALUE en la descripción del elemento de datos, el elemento de datos se inicializa con el valor definido. Si una cláusula VALUE no está asociada con un elemento de datos, el valor inicial del elemento de datos no está definido.
- Los archivos con conectores de archivo internos asociados con el programa no están en modalidad abierta.
- Los mecanismos de control para todas las sentencias PERFORM contenidas en el programa se establecen en sus estados iniciales.
- Una sentencia GO TO modificada contenida en el programa se establece en su estado inicial.

Para ver las reglas que rigen los nombres de programa no exclusivos, consulte "Reglas para nombres de programa" en la página 82.

Párrafos opcionales

Algunos párrafos opcionales de la DIVISIÓN DE IDENTIFICACIÓN pueden omitirse.

Los párrafos opcionales son:

AUTOR
Nombre del autor del programa.

instalación
Nombre de la empresa o ubicación.

FECHA ESCRITA
Fecha en la que se ha escrito el programa.

DATE-COMPILADO
El párrafo DATE-COMPILE proporciona la fecha de compilación en el listado fuente. Si se especifica una entrada de comentario, la entrada completa se sustituye por la fecha actual, incluso si la entrada abarca líneas. Si se omite la entrada de comentario, el compilador añade la fecha actual a la línea en la que se imprime DATE-COMPILE. Por ejemplo:

```
DATE-COMPILED. 06/30/10.
```

Seguridad
Nivel de confidencialidad del programa.

La *entrada de comentario* en cualquiera de los párrafos opcionales puede ser cualquier combinación de caracteres del juego de caracteres del sistema. La entrada de comentario se escribe en el Área B en una o más líneas.

Las entradas de comentarios sólo sirven como documentación; no afectan al significado del programa. No se permite un guión en el área de indicador (columna 7) en entradas de comentario.

Puede incluir caracteres de varios bytes, así como caracteres de un solo byte de un EUC, UTF-8o DBCS en entradas de comentario en la DIVISIÓN DE IDENTIFICACIÓN del programa. Se permiten varias líneas en una entrada de comentario que contiene caracteres de varios bytes.

Parte 4. División de entorno

Capítulo 13. Sección de configuración

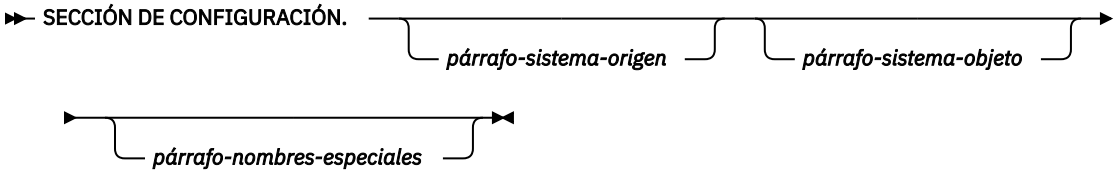
La sección de configuración es una sección opcional para programas, y puede describir el entorno del sistema en el que se compila y ejecuta el programa o clase.

Sección de configuración del programa

La sección de configuración sólo se puede especificar en ENVIRONMENT DIVISION del programa más externo de un programa fuente COBOL.

No debe especificar la sección de configuración en un programa contenido en otro programa. Las entradas especificadas en la sección de configuración de un programa se aplican a cualquier programa contenido en ese programa.

Formato:



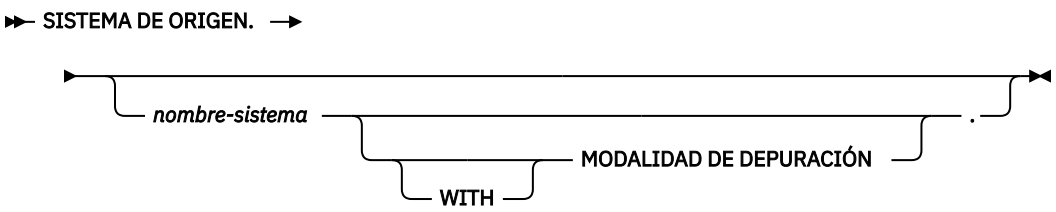
La sección de configuración puede:

- Relacionar nombres de entorno definidos por IBMcon nombres nemotécnicos definidos por el usuario
- Especifique el orden de clasificación
- Especifique un valor de signo de moneda y el símbolo de moneda utilizado en la cláusula PICTURE para representar el valor de signo de moneda
- Intercambiar las funciones de la coma y el punto en cláusulas PICTURE y literales numéricos
- Relacionar nombres alfabéticos con juegos de caracteres o secuencias de clasificación
- Especificar caracteres simbólicos
- Especificar los formatos predeterminados para un tipo de datos de fecha u hora

Párrafo SOURCE-COMPUTER

El párrafo SOURCE-COMPUTER describe el sistema en el que se va a compilar el texto de origen.

Formato



nombre-sistema

Un nombre de sistema. Por ejemplo:

```
IBM-system
```

CON MODALIDAD DE DEPURACIÓN
Activa un conmutador de tiempo de compilación para depurar líneas escritas en el texto de origen.

Una *línea de depuración* es una sentencia que se compila sólo cuando se activa el conmutador de tiempo de compilación. La depuración de líneas le permite, por ejemplo, comprobar el valor de un nombre de datos en determinados puntos de un procedimiento.

Para especificar una línea de depuración en el programa, codifique una D en la columna 7 (área de indicador). Puede incluir líneas de depuración sucesivas, pero cada una debe tener una D en la columna 7, y no puede dividir series de caracteres entre líneas.

Todas las líneas de depuración deben escribirse para que el programa sea sintácticamente correcto, tanto si las líneas de depuración se compilan como si se tratan como comentarios.

La presencia o ausencia de la cláusula DEBUGGING MODE se determina lógicamente después de que se hayan procesado todas las sentencias COPY y REPLACE.

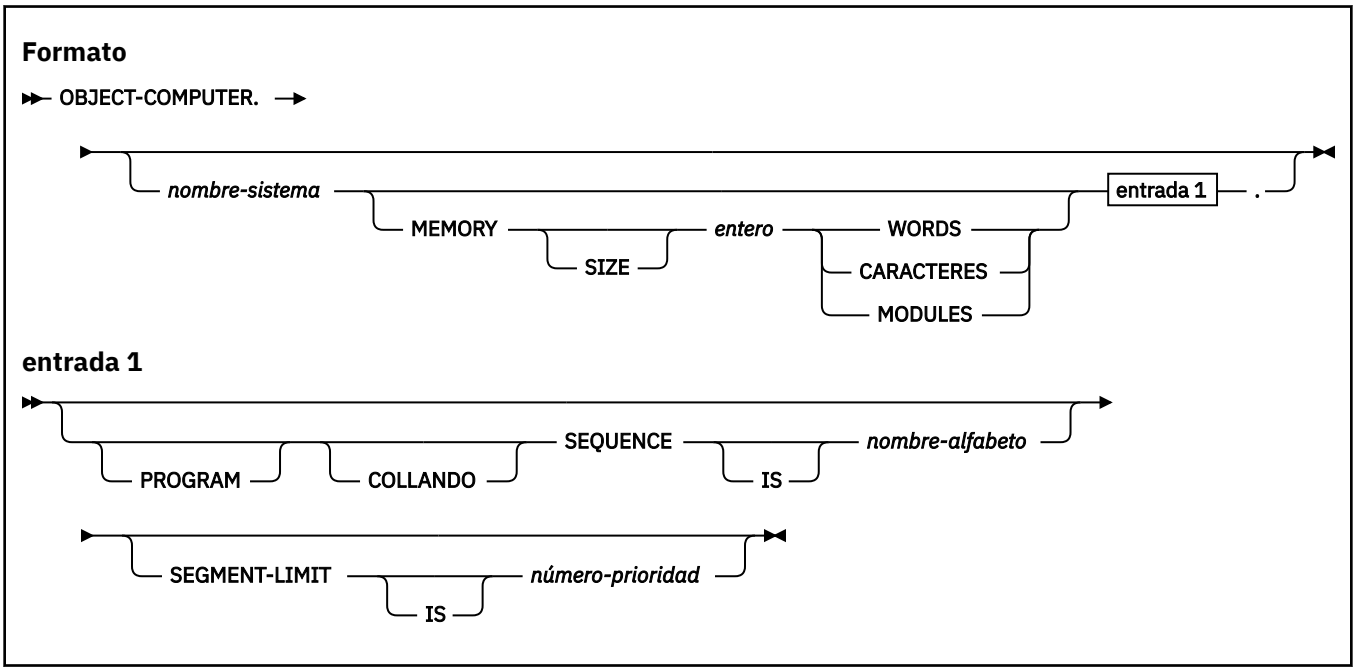
Puede codificar líneas de depuración en ENVIRONMENT DIVISION (después del párrafo OBJECT-COMPUTER) y en las divisiones de datos y procedimientos.

Si una línea de depuración contiene sólo espacios en el área A y en el área B, la línea de depuración se trata igual que una línea en blanco.

Se comprueba la sintaxis de todo el párrafo SOURCE-COMPUTER, pero sólo la cláusula WITH DEBUGGING MODE tiene un efecto en la ejecución del programa.

Párrafo OBJECT-COMPUTER

El párrafo OBJECT-COMPUTER especifica el sistema para el que se designa el programa objeto.



nombre-sistema
Un nombre de sistema. Por ejemplo:

```
IBM-system
```

MEMORY SIZE *entero*
integer especifica la cantidad de almacenamiento principal necesaria para ejecutar el programa objeto, en palabras, caracteres o módulos. La cláusula MEMORY SIZE se comprueba con la sintaxis, pero no tiene ningún efecto en la ejecución del programa.

PROGRAM COLLATING SEQUENCE IS *nombre-alfabeto*

La secuencia de clasificación utilizada en este programa es la secuencia de clasificación asociada con el *nombre-alfabético*especificado.

La secuencia de clasificación pertenece a este programa y a cualquier programa que este programa pueda contener.

PROGRAM COLLATING SEQUENCE determina el valor de verdad de las siguientes comparaciones alfanuméricas:

- Las especificadas explícitamente en condiciones de relación
- Los especificados explícitamente en condiciones de nombre de condición

La cláusula PROGRAM COLLATING SEQUENCE también se aplica a las claves de fusión o clasificación descritas con el uso DISPLAY, a menos que se especifique la frase COLLATING SEQUENCE en la sentencia MERGE o SORT.

La cláusula PROGRAM COLLATING SEQUENCE no se aplica a los elementos de datos a los elementos de datos de uso NATIONAL.

No puede especificar la cláusula PROGRAM COLLATING SEQUENCE si la página de códigos fuente es una página de códigos de varios bytes.

Si se omite la cláusula PROGRAM COLLATING SEQUENCE, se utiliza la secuencia de clasificación EBCDIC de La opción de compilador COLLSEQ indica el orden de clasificación utilizado. Por ejemplo, si se especifica COLLSEQ (EBCDIC) y no se especifica la cláusula PROGRAM COLLATING SEQUENCE (o es NATIVE), se utiliza la secuencia de clasificación EBCDIC.

SEGMENT-LIMIT ES

La cláusula SEGMENT-LIMIT se comprueba con la sintaxis pero no tiene ningún efecto en la ejecución del programa.

número-prioridad

Un entero comprendido entre 1 y 49. Este valor se comprueba con la sintaxis pero no tiene ningún efecto en la ejecución del programa.

Se comprueba la sintaxis de todo el párrafo OBJECT-COMPUTER, pero sólo la cláusula PROGRAM COLLATING SEQUENCE tiene un efecto en la ejecución del programa.

Párrafo SPECIAL-NAMES

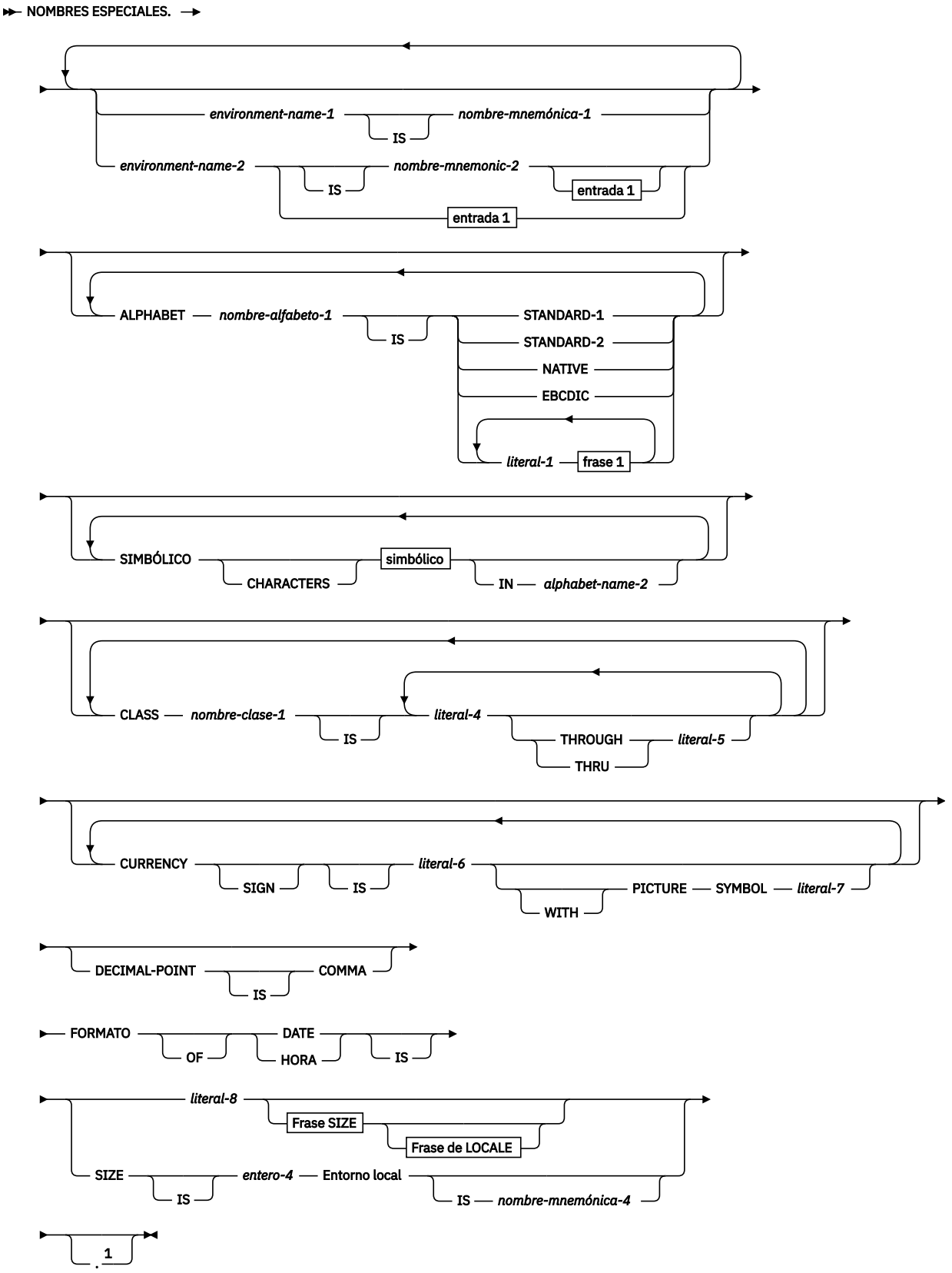
El párrafo SPECIAL-NAMES es el nombre de un párrafo ENVIRONMENT DIVISION en el que los nombres de entorno están relacionados con nombres mnemónicos especificados por el usuario.

El párrafo SPECIAL-NAMES:

- Relaciona nombres de entorno especificados por IBMcon nombres mnemotécnicos definidos por el usuario
- Relaciona nombres de alfabeto con juegos de caracteres o secuencias de clasificación
- Especifica caracteres simbólicos
- Relaciona nombres de clase con conjuntos de caracteres
- Especifica uno o más valores de signo de moneda y define un símbolo de imagen para representar cada valor de signo de moneda en las cláusulas PICTURE
- Especifica que las funciones de coma y coma decimal deben intercambiarse en cláusulas PICTURE y literales numéricos
- Especifica los formatos predeterminados para un tipo de datos de fecha y/o hora
- Relaciona nombres de objeto de entorno local y su biblioteca asociada con nombres nemotécnicos definidos por el usuario

Las cláusulas del párrafo SPECIAL-NAMES pueden aparecer en cualquier orden.

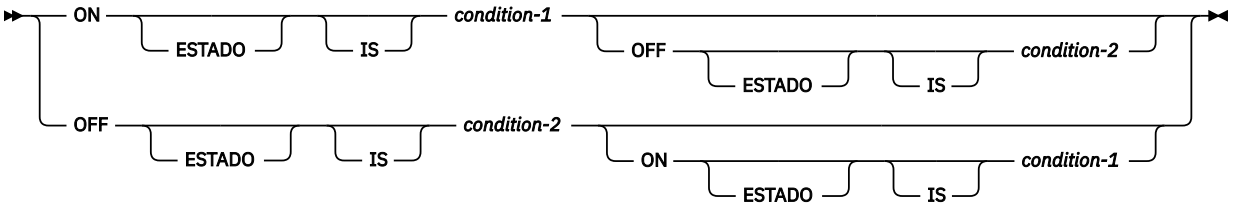
Formato: párrafo SPECIAL-NAMES



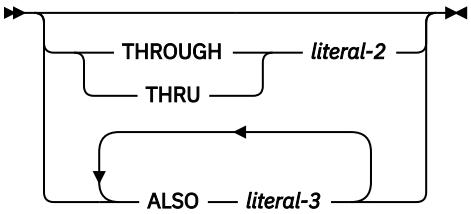
Notas:

[1] Este punto de separación es opcional cuando no se selecciona ninguna cláusula. Si utiliza alguna cláusula, debe codificar el periodo después de la última cláusula.

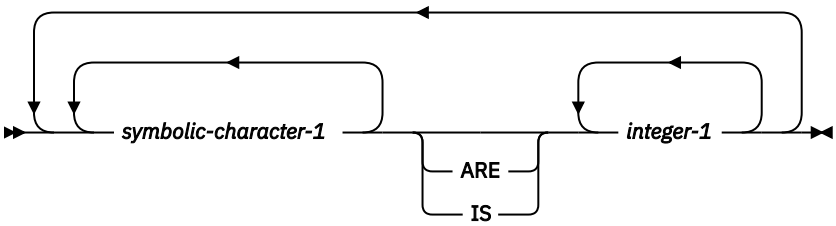
Fragmentos
entrada 1



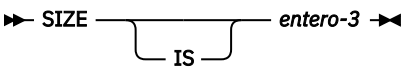
frase 1



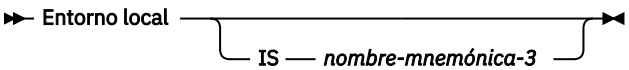
simbólica



Frase SIZE



Frase LOCALE



Cuando la página de códigos fuente es una página de códigos de varios bytes, no se pueden especificar las cláusulas siguientes:

- cláusula ALPHABET
- cláusula CLASS
- cláusula SYMBOLIC characters

environment-name-1
Dispositivos del sistema o acciones estándar del sistema realizadas por el compilador.

En la tabla siguiente se muestran las especificaciones válidas para *environment-name-1* .

Tabla 5. ***Significados de los nombres de entorno***		
entorno- name-1	**Significado**	**Permitido en**
SYSIN SYSIPT	Unidad de entrada lógica del sistema	ACEPTAR
SYSOUT SYSLISTA SYSLST	Unidad de salida lógica del sistema	VISUALIZAR
SYSPUNCH SYSPCH	Dispositivo de perforación del sistema	VISUALIZAR
CONSOLA	Consola	ACCEPT y DISPLAY
C01 a C12	Saltar al canal 1 a través del canal 12, respectivamente	AVANCE DE ESCRITURA Con C01 a C12, una línea es avanzada.
PEC	Suprimir espaciado	AVANCE DE ESCRITURA
S01 a S05	Selección de bolsillo de 1 a 5 en dispositivos punch	AVANCE DE ESCRITURA Con S01 a S05, una línea es avanzada.
AFP-5A	Impresión de función avanzada	AVANCE DE ESCRITURA

environment-name-2
Conmutador de indicador de estado programable por el usuario (UPSI) de 1 byte. Las especificaciones válidas para *environment-name-2* son UPSI-0 a UPSI-7.

mnemonic-name-1 , mnemonic-name-2
mnemonic-name-1 y *mnemonic-name-2* siguen las reglas de formación para nombres definidos por el usuario. *mnemonic-name-1* se puede utilizar en sentencias ACCEPT, DISPLAY y WRITE. Sólo se puede hacer referencia a *mnemonic-name-2* en la sentencia SET. *mnemonic-name-2* puede calificar *condition-1* o *condition-2* nombres.

No es necesario que los nombres mnemotécnicos y los nombres de entorno sean exclusivos. Si elige un nombre-mnemónico que también sea un nombre-entorno, su definición como un nombre-mnemónico tendrá prioridad sobre su definición como un nombre-entorno.

EL ESTADO ES, EL ESTADO DESACTIVADO ES
UPSI cambia condiciones especiales de proceso dentro de un programa, como el proceso de inicio de año o de fin de año. Por ejemplo, al principio de la PROCEDURE DIVISION, se puede probar un conmutador UPSI; si está ON, se toma la rama especial. (Consulte el apartado "Condición de estado de conmutador" en la página 268.)

condition-1, condition-2
Los nombres de condición siguen las reglas para los nombres definidos por el usuario. Al menos un carácter debe ser alfabético. El valor asociado con el nombre-condición se considera alfanumérico. Un nombre de condición puede estar asociado con el estado activado o desactivado de cada conmutador UPSI especificado.

En PROCEDURE DIVISION, el estado del conmutador UPSI se prueba a través del nombre de condición asociado. Cada nombre-condición es el equivalente de un elemento level-88 ; el nombre-nemotécnico asociado, si se especifica, se considera la variable condicional y se puede utilizar para la calificación.

Los nombres de condición especificados en el párrafo SPECIAL-NAMES de un programa contenedor pueden estar referenciados en cualquier programa contenido.

cláusula ALPHABET

La cláusula ALPHABET proporciona un medio para relacionar un nombre de alfabeto con un juego de códigos de caracteres o secuencia de clasificación especificados.

El conjunto de códigos de caracteres relacionados o la secuencia de clasificación se pueden utilizar para datos alfanuméricos, pero no para datos nacionales.

ALPHABET *alphabet-name-1* IS

alphabet-name-1 especifica un *orden de clasificación* cuando se utiliza en:

- Cláusula PROGRAM COLLATING SEQUENCE del párrafo object-computer
- La frase COLLATING SEQUENCE de la sentencia SORT o MERGE

alphabet-name-1 especifica un juego de códigos de caracteres cuando se utiliza en:

- Cláusula CODE-SET de entrada FD
- Cláusula SIMBÓLICO CHARACTERS

No puede especificar la cláusula ALPHABET si la página de códigos fuente en vigor es una página de códigos de varios bytes. Para obtener más detalles, consulte *Especificación del orden de clasificación* en el *COBOL for Linux en x86 Guía de programación*.

STANDARD-1
Especifica que la secuencia de clasificación se basa en los valores de código binario de los caracteres, ignorando el valor de entorno local.

STANDARD-2
Especifica que la secuencia de clasificación se basa en los valores de código binario de los caracteres, ignorando el valor de entorno local.

NATIVA
Especifica el juego de códigos de caracteres nativos. Si se omite la cláusula ALPHABET, se asume el nombre del alfabeto está asociado con ASCII, UTF-8, o EUC juego de caracteres indicado por el entorno local en vigor.

EBCDIC
Especifica el juego de caracteres EBCDIC.

literal-1* , *literal-2* , *literal-3
Especifica que el programa determina la secuencia de clasificación para los datos alfanuméricos, de acuerdo con las reglas siguientes:

- El orden en el que aparecen los literales especifica el número ordinal, en secuencia ascendente, de los caracteres de este orden de clasificación.
- Cada literal numérico especificado debe ser un entero sin signo.
- Cada literal numérico debe tener un valor que corresponda a una posición ordinal válida dentro del orden de clasificación en vigor.

  Consulte Apéndice B, “Secuencias de clasificación EBCDIC y ASCII”, en la página 555 para ver los números ordinales de los caracteres de las secuencias de clasificación EBCDIC y ASCII de un solo byte.
- Cada carácter de un literal alfanumérico representa ese carácter real en el juego de caracteres. (Si el literal alfanumérico contiene más de un carácter, a cada carácter, empezando por el situado más a la izquierda, se le asigna una posición sucesivamente ascendente dentro de esta secuencia de clasificación.)
- Los caracteres que no se especifican explícitamente asumen posiciones en este orden de clasificación superiores a cualquiera de los caracteres especificados explícitamente. El orden relativo dentro de la secuencia de clasificación de estos caracteres no especificados es su orden relativo en la secuencia de clasificación indicada por la opción de compilador COLLSEQ.

- Dentro de una cláusula de nombre de alfabeto, un carácter determinado no debe especificarse más de una vez.
- Cada literal alfanumérico asociado con una frase THROUGH o ALSO debe tener un carácter de longitud.
- Cuando se especifica la frase THROUGH, los caracteres contiguos del juego de caracteres nativo que empiezan con el carácter especificado por *literal-1* y terminan con el carácter especificado por *literal-2* se asignan sucesivamente posiciones ascendentes en esta secuencia de clasificación.

  Esta secuencia puede ser ascendente o descendente dentro del juego de caracteres nativo original. Es decir, si se especifica "Z" THROUGH "A", los valores ascendentes, de izquierda a derecha, para las letras mayúsculas son:

  ```
  ZYXWVUTSRQPONMLKJIHGFEDCBA
  ```

- Cuando se especifica la frase ALSO, los caracteres especificados como *literal-1*, *literal-3*, ... se asignan a la misma posición en este orden de clasificación. Por ejemplo, si especifica:

  ```
  "D" ALSO "N" ALSO "%"
  ```

  los caracteres D, N y% se consideran todos en la misma posición en el orden de clasificación.
- Cuando se especifica la frase ALSO y se hace referencia a *alphabet-name-1* en una cláusula SIMBÓLICO CHARACTERS, solo se utiliza *literal-1* para representar el carácter del juego de caracteres.
- El carácter que tiene la posición ordinal más alta en esta secuencia de clasificación se asocia con la constante figurativa HIGH-VALUE. Si más de un carácter tiene la posición más alta debido a la especificación de la frase ALSO, el último carácter especificado (o que se toma por omisión cuando no se especifica explícitamente ningún carácter) se considera el carácter HIGH-VALUE para sentencias de procedimiento como DISPLAY y como campo de envío en una sentencia MOVE. (Si el ejemplo de frase ALSO indicado anteriormente se especificara como los caracteres de orden superior de esta secuencia de clasificación, el carácter HIGH-VALUE sería%.)
- El carácter que tiene la posición ordinal más baja en esta secuencia de clasificación está asociado con la constante figurativa LOW-VALUE. Si más de un carácter tiene la posición más baja debido a la especificación de la frase ALSO, el primer carácter especificado es el carácter LOW-VALUE. (Si el ejemplo de frase ALSO indicado anteriormente se especificara como los caracteres de orden inferior de la secuencia de clasificación, el carácter LOW-VALUE sería D.)

Cuando se especifica *literal-1*, *literal-2*o *literal-3* , No se debe hacer referencia al nombre de alfabeto en una cláusula CODE-SET (consulte “cláusula CODE-SET” en la página 165).

literal-1, *literal-2*y *literal-3* deben ser literales alfanuméricos o numéricos. Todos deben tener la misma categoría. No se debe especificar un literal de coma flotante, un literal nacional, un literal DBCS o una constante figurativa de carácter simbólico.

cláusula CLASS

La cláusula CLASS proporciona un medio para relacionar un nombre con el conjunto especificado de caracteres listados en dicha cláusula.

No puede especificar la cláusula CLASS si la página de códigos fuente en vigor es una página de códigos de varios bytes.

CLASS *class-name-1* IS

Proporciona un medio para relacionar un nombre con el conjunto especificado de caracteres listados en esa cláusula. Sólo se puede hacer referencia a *class-name-1* en una condición de clase. Los caracteres especificados por los valores de los literales de esta cláusula definen el conjunto exclusivo de caracteres de los que consta esta clase.

literal-4*, *literal-5

Debe ser de categoría numérica o alfanumérica, y ambos deben ser de la misma categoría.

Si es numérico, *literal-4* y *literal-5* deben ser enteros sin signo y deben tener un valor mayor o igual que 1 y menor o igual que el número de caracteres del alfabeto especificado. Cada número corresponde a la posición ordinal de cada carácter en la secuencia de clasificación EBCDIC o ASCII de un solo byte. No se pueden especificar como literales de coma flotante o como literales DBCS.

Si es alfanumérico, *literal-4* y *literal-5* son un carácter EBCDIC de un solo byte o ASCII de un solo byte real.

literal-4 y *literal-5* no deben especificar una constante figurativa de carácter simbólico. Si el valor del literal alfanumérico contiene varios caracteres, cada carácter del literal se incluye en el conjunto de caracteres identificados por el nombre de clase.

Si el literal alfanumérico está asociado con una frase THROUGH, el literal debe tener un carácter de longitud.

A TRAVÉS, A TRAVÉS

THROUGH y THRU son equivalentes. Si se especifica THROUGH, el nombre de clase incluye los caracteres que empiezan con el valor de *literal-4* y que terminan con el valor de *literal-5*. Además, los caracteres especificados por una frase THROUGH pueden estar en orden ascendente o descendente.

cláusula CURRENCY SIGN

La cláusula CURRENCY SIGN afecta a los elementos de datos editados numéricos cuyas series de caracteres PICTURE contienen un *símbolo de moneda*.

Un símbolo de moneda representa un *valor de signo de moneda* que es:

- Insertado en estos elementos de datos cuando se utilizan como elementos de recepción
- Eliminados de estos elementos de datos cuando se utilizan como envío de elementos para un receptor numérico o editado numérica-editado

Normalmente, los valores de signo de moneda identifican las unidades monetarias almacenadas en un elemento de datos. Por ejemplo: '$', 'EUR ',' CHF ',' JPY ',' HK$ ',' HKD 'o X'9F' (punto de código hexadecimal en algunas páginas de códigos EBCDIC para €, el signo de moneda euro). Para más detalles sobre las técnicas de programación para manejar el euro, consulte *Utilización de signos de moneda* en el *COBOL for Linux en x86 Guía de programación*.

La cláusula CURRENCY SIGN especifica un valor de signo de moneda y el símbolo de moneda utilizado para representar dicho valor de signo de moneda en una cláusula PICTURE.

El párrafo SPECIAL-NAMES puede contener varias cláusulas CURRENCY SIGN. Cada cláusula CURRENCY SIGN debe especificar un símbolo de moneda diferente. A diferencia de todos los demás símbolos de cláusula PICTURE, los símbolos de moneda distinguen entre mayúsculas y minúsculas. Por ejemplo, 'D' y' d' especifican diferentes símbolos de moneda.

CURRENCY SIGN IS *literal-6*

literal-6 debe ser un literal alfanumérico. *literal-6* no debe ser una constante figurativa o un literal terminado en nulo. *literal-6* no debe contener un carácter multibyte.

Si no se especifica la frase PICTURE SYMBOL, *literal-6*:

- Especifica un valor de signo de moneda y el símbolo de moneda para este valor de signo de moneda
- Debe ser un único carácter
- No debe contener ninguno de los dígitos o caracteres siguientes:
  - Dígitos del 0 al 9
  - Caracteres alfabéticos A, B, C, D, E, G, N, P, R, S, V, X, Z, sus equivalentes en minúsculas o el espacio

- Caracteres especiales +-,. */; () " = ' (signo más, signo menos, coma, punto, asterisco, barra inclinada, punto y coma, paréntesis izquierdo, paréntesis derecho, comillas, signo igual, apóstrofo)

- Puede ser uno de los siguientes caracteres alfabéticos en minúsculas: f, h, i, j, k, l, m, o, q, t, u, w, y

Si se especifica la frase PICTURE SYMBOL, *literal-6*:

- Especifica un valor de signo de moneda. *literal-7* en la frase PICTURE SYMBOL especifica el símbolo de moneda para este valor de signo de moneda.
- Puede constar de uno o más caracteres.
- No debe contener ninguno de los dígitos o caracteres siguientes:
  - Dígitos del 0 al 9
  - Caracteres especiales +-.,

PICTURE SYMBOL *literal-7*
Especifica un símbolo de moneda que se puede utilizar en una cláusula PICTURE para representar el valor de signo de moneda especificado por *literal-6*.

literal-7 debe ser un literal alfanumérico que conste de un carácter de un solo byte. *literal-7* no debe contener ninguno de los dígitos o caracteres siguientes:

- Una constante figurativa
- Dígitos del 0 al 9
- Caracteres alfabéticos A, B, C, D, E, G, N, P, R, S, V, X, Z, sus equivalentes en minúsculas o el espacio
- Caracteres especiales +-,. */; () " = '

Si se especifica la cláusula CURRENCY SIGN, se ignoran las opciones de compilador CURRENCY y NOCURRENCY. Si no se especifica la cláusula CURRENCY SIGN y la opción de compilador NOCURRENCY está en vigor, se utiliza el signo de dólar ($) como valor de signo de moneda predeterminado y símbolo de moneda. Para obtener más información sobre las opciones de compilador CURRENCY y NOCURRENCY, consulte *MONEDA* en la publicación *COBOL for Linux en x86 Guía de programación*.

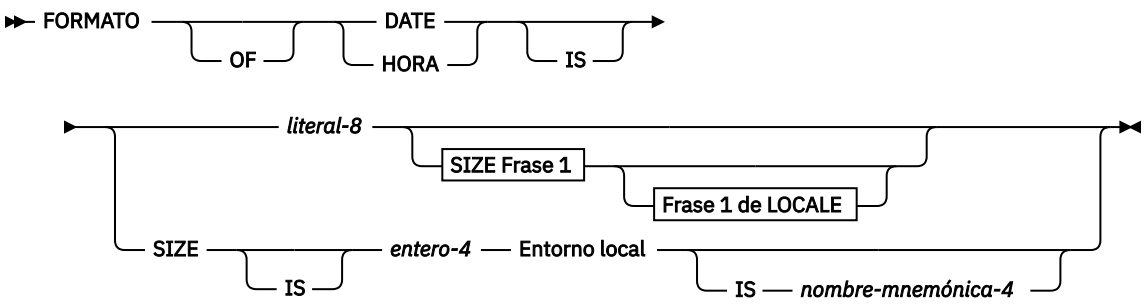
Cláusula DECIMAL-POINT IS COMA

La cláusula DECIMAL-POINT IS COMA intercambia las funciones del punto y la coma en series de caracteres PICTURE y en literales numéricos.

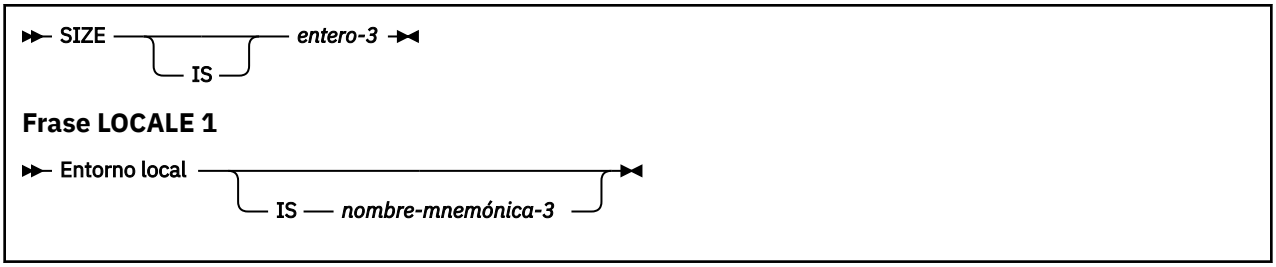
cláusula FORMAT

La cláusula FORMAT se utiliza para especificar un formato predeterminado para un elemento de fecha u hora DATA DIVISION. La cláusula de formato también puede especificar el formato de fecha u hora predeterminado para una función intrínseca.

Cláusula FORMAT-Formato



Frase 1 de TAMAÑO



literal-8

Especifica el formato predeterminado de un elemento de fecha u hora. *literal-8* debe ser un literal alfanumérico de al menos 2 caracteres de longitud. *literal-8* debe contener uno o más especificadores de conversión y cero o más separadores. Para obtener más información sobre los efectos de *literal-8* en la frase LOCALE, consulte "frase LOCALE" en la página 107. Para obtener una lista de los especificadores de conversión que se pueden utilizar en *literal-8*, consulte Tabla 6 en la página 105.

Se aplican las reglas siguientes:

- Cuando no se especifica ninguna frase LOCALE con *literal-8*, las especificaciones de conversión se sustituyen por valores basados en el entorno local COBOL. Para obtener más información sobre el entorno local COBOL, consulte *COBOL for Linux en x86 Guía de programación*.
- Para un elemento de fecha, *literal-8* debe contener un especificador de conversión que dará como resultado el día del año. Si *literal-8* contiene una especificación de conversión de año y mes, pero no una especificación de conversión de día, se asume el primer día del mes.
- Si no se especifica ninguna cláusula FORMAT para un elemento de fecha, el formato de elemento de fecha predeterminado es ISO.
- Si no se especifica *literal-8* , se debe especificar la frase LOCALE.
- Para un elemento de tiempo, *literal-8* debe contener una especificación de conversión de hora y minuto. Si no se especifica ningún segundo (o milisegundos), se presupone un valor de 0.
- Si no se especifica ninguna cláusula FORMAT para un elemento de tiempo, el formato de elemento de tiempo predeterminado es ISO.

La tabla siguiente listalos especificadores de conversión que se pueden utilizar en *literal-8*.

Tabla 6. Especificadores de conversión que se pueden utilizar en literal-8

Especificador	Descripción	Longitud	Permitido para
@C	Sustituido por el siglo como un entero [0, 9] (0edad 20th)	1 bytes	D
%d	Se sustituye por el día del mes como un entero [01,31]	2 bytes	D
%D	Igual que %m/%d/%y	8 bytes	D
%H	Se sustituye por la hora (reloj de 24 horas) como un entero [00,23]	2 bytes	T
%I	Se sustituye por la hora (reloj de 12 horas) como un entero [01,12]	2 bytes	T
%j	Se sustituye por el día del año como un entero [001,366]	3 bytes	D
%m	Se sustituye por el mes como un entero [01,12]	2 bytes	D
%M	Sustituido por el minuto como un entero [00,59]	2 bytes	T
%p	Se sustituye por el equivalente del entorno local de a.m. o p.m.	Entorno local	T
@p	AM y PM pueden ser cualquier mezcla de mayúsculas y minúsculas	2 bytes	T

Tabla 6. Especificadores de conversión que se pueden utilizar en literal-8 (continuación)

Especificador	Descripción	Longitud	Permitido para
%r	Se sustituye por la hora de la mañana. y p.m. notación; en el entorno local POSIX es equivalente a %I: %M: %S %p	entorno local, al menos 8 bytes	T
%R	Sustituido por la hora en la notación de 24 horas [%H: %M]	5 bytes	T
%S	Sustituido por el segundo como un entero [00,61]	2 bytes	T
@Sh	Sustituido por las centésimas de segundo como un entero [00,99]	2 bytes	T
@Sm	Se sustituye por las millonésimas de segundo como un entero [000000,999999]	6 bytes	T
@So	Se sustituye por las milésimas de segundo como un entero [000,999]	3 bytes	T
@Sp	Se sustituye por las trillonésimas de segundo como un entero [000000000000, 999999999999]	12 bytes	T
@St	Sustituido por las décimas de segundo como un entero [0, 9]	1 bytes	T
%y	Sustituido por el año sin siglo como un entero [00,99]	2 bytes	D
%Y	Sustituido por el año con el siglo como un entero	normalmente 4 bytes	D
@Y	Sustituido por el año con el siglo como un entero	4 bytes	D
%%	Sustituido por un%	1 byte	D, T
@@	Sustituido por @	1 byte	D, T

Notas:

1. Los especificadores de conversión distinguen entre mayúsculas y minúsculas.
2. Los símbolos de columna permitidos para tienen el significado siguiente:
   - D-FECHA elemento
   - elemento T-TIME
3. La columna Longitud se basa en el entorno local COBOL predeterminado.
4. De forma predeterminada, un valor de cero representa el siglo XX (de 1900 a 1999).

frase SIZE

La frase SIZE especifica el tamaño total del elemento de fecha u hora en número de dígitos. El número de dígitos debe ser mayor o igual que el tamaño del literal de formato. El tamaño del literal de formato se determina sustituyendo los especificadores de conversión por su valor más grande y realizando conversiones, si es necesario, al CCSID de tiempo de ejecución.

La frase SIZE debe especificarse para un elemento de fecha u hora cuando la longitud de ese elemento no puede determinarse en el momento de la compilación. El compilador no puede determinar el tamaño de un elemento de fecha u hora cuando:

- Se especifican tanto *literal-8* como la frase LOCALE, lo que significa que la longitud real del elemento de fecha u hora se determinará parcialmente en tiempo de ejecución a partir del entorno local especificado.

- *literal-8* se especifica sin una frase LOCALE, y una de las especificaciones de conversión dentro de *literal-8* puede dar como resultado un elemento de longitud variable.
- *literal-8 no* se ha especificado, lo que significa que la longitud real del elemento de fecha u hora se determinará completamente en tiempo de ejecución a partir del entorno local especificado.

integer-3, integer-4
integer-3 y *integer-4* especifican el tamaño del elemento de fecha u hora predeterminado en número de dígitos. Se debe especificar *integer-3* o *integer-4* si el tamaño del elemento de fecha u hora no se puede determinar en el momento de la compilación. Para un elemento de fecha y hora, *integer-3* y *integer-4* deben ser iguales o mayores que 4. El tamaño máximo de un elemento de la clase fecha-hora es 256, si el elemento tiene un USAGE de DISPLAY, o 31 para un USAGE de PACKED-DECIMAL.

frase LOCALE

La frase LOCALE se utiliza para especificar el entorno local culturalmente específico que se va a utilizar para formatear elementos de fecha y hora.

Cuando se especifica la frase LOCALE *sin literal-8*, el formato y separador del elemento de fecha u hora se basa completamente en un entorno local. Cuando se especifica la frase LOCALE *con literal-8*, *literal-8* determina el formato del elemento, pero el valor utilizado para sustituir cualquier especificador de conversión que dependa de un entorno local para su representación exacta (por ejemplo, %p) se basará en el entorno local.

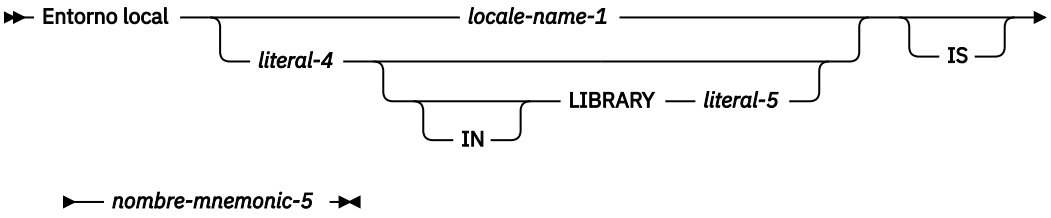
mnemonic-name-3, mnemonic-name-4
Si se especifica *mnemonic-name-3* o *mnemonic-name-4* , el entorno local utilizado para el elemento de fecha u hora es el que está asociado con *mnemonic-name-3* o *mnemonic-name-4* en el párrafo SPECIAL-NAMES. Si no se especifica *mnemonic-name-3* o *mnemonic-name-4* , se utiliza el entorno local actual. Para determinar el entorno local actual, consulte *CEELOCT: get current local date or time* en la publicación *COBOL for Linux en x86 Guía de programación*.

mnemonic-name-3 y *mnemonic-name-4* deben ser nombres mnemónicos de entorno local. Los nombres nemotécnicos de entorno local se especifican con la cláusula LOCALE del párrafo SPECIAL-NAMES, consulte “cláusula LOCALE” en la página 107.

cláusula LOCALE

La cláusula LOCALE se utiliza para definir nombres mnemónicos de entorno local y su biblioteca y nombre de objeto de entorno local equivalente.

Cláusula LOCALE-Formato



locale-name-1
Especifica un nombre específico del sistema que hace referencia a un objeto de entorno local. Para COBOL para Linux, el único *locale-name-1* soportado es POSIX.

literal-4
literal-4 debe ser un nombre de objeto de entorno local. Debe ser un literal no numérico con una longitud máxima de 10 caracteres.

literal-5
literal-5 se utiliza para especificar el nombre de la biblioteca del sistema operativo en la que se encuentra el objeto de entorno local. Debe ser un literal no numérico con una longitud máxima de 10 caracteres. Si se omite la frase LIBRARY, se utiliza la lista de bibliotecas del trabajo para buscar el objeto de entorno local.

mnemonic-name-5

mnemonic-name-5 proporciona una referencia al entorno local identificado por *locale-name-1* o los valores especificados para *literal-4* y *literal-5*. Sólo se puede utilizar en una cláusula FORMAT, cláusula PICTURE, formato 9 de la sentencia SET o en la lista de argumentos de algunas funciones intrínsecas.

cláusula SIMBÓLICO CHARACTERS

La cláusula SYMBOLIC CHARACTERS sólo es aplicable a los juegos de caracteres de un solo byte. Cada carácter representado es un carácter alfanumérico.

CARACTERES SIMBÓLICOS *symbolic-character-1*

Proporciona un medio para especificar uno o más caracteres simbólicos. *symbolic-character-1* es una palabra definida por el usuario y debe contener al menos un carácter alfabético. El mismo carácter simbólico sólo puede aparecer una vez en una cláusula SYMBOLIC CHARACTERS.

No puede utilizar la cláusula SYMBOLIC CHARACTERS si la página de códigos de texto fuente es una página de códigos de varios bytes.

La representación interna de *symbolic-character-1* es la representación interna del carácter que se representa en el juego de caracteres especificado. Se aplican las reglas siguientes:

- La relación entre cada *symbolic-character-1* y el *integer-1* correspondiente se establece por su posición en la cláusula SYMBOLIC-CHARACTER-1. El primer *symbolic-character-1* se empareja con el primer *entero-1*; el segundo *symbolic-character-1* se empareja con el segundo *entero-1*; y así sucesivamente.
- Debe haber una correspondencia de uno a uno entre las apariciones de *symbolic-character-1* y las apariciones de *integer-1* en una cláusula SYMBOLIC-CHARACTER-1.
- Si se especifica la frase IN, *integer-1* especifica la posición ordinal del carácter que se representa en el juego de caracteres denominado *alphabet-name-2*. Esta posición ordinal debe existir.
- Si no se especifica la frase IN, *symbolic-character-1* representa el carácter cuya posición ordinal en el juego de caracteres nativo se especifica mediante *integer-1*.

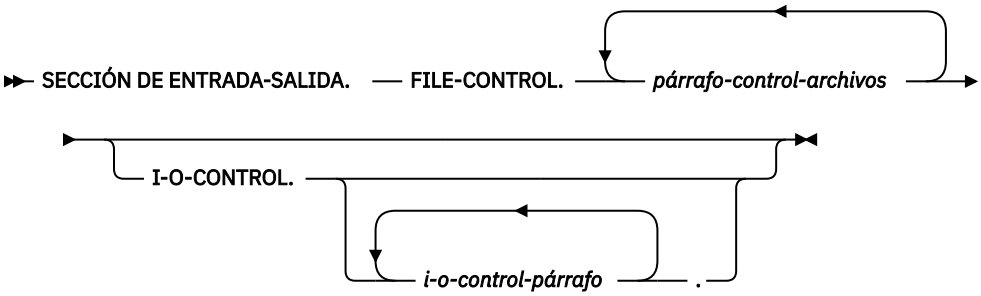
  Las posiciones ordinales se numeran a partir de 1.

Capítulo 14. Sección Entrada-Salida

La sección de entrada-salida de ENVIRONMENT DIVISION contiene el párrafo FILE-CONTROL y el párrafo I-O-CONTROL.

El contenido exacto de la sección de entrada-salida depende de la organización del archivo y de los métodos de acceso utilizados. Consulte “cláusula ORGANIZATION” en la página 121 y “cláusula ACCESS MODE” en la página 124.

Formato: sección de entrada-salida



FILE-CONTROL
La palabra clave FILE-CONTROL identifica el párrafo de control de archivos. Esta palabra clave sólo puede aparecer una vez, al principio del párrafo FILE-CONTROL. Debe empezar en el Área A y estar seguido por un punto separador.

La palabra clave FILE-CONTROL y el punto pueden omitirse si no se especifica ningún *párrafo-control-archivo* y no hay archivos definidos en el programa.

párrafo-control-archivos
Nombra los archivos y los asocia con los archivosexternos.

Debe empezar en el Área B con una cláusula SELECT. Debe finalizar con un punto de separación. Consulte “Párrafo FILE-CONTROL” en la página 109.

párrafo-control-archivo se puede omitir si no hay archivos definidos en el programa, incluso si se especifica la palabra clave FILE-CONTROL.

I-O-CONTROL
La palabra clave I-O-CONTROL identifica el párrafo I-O-CONTROL.

i-o-control-párrafo
Especifica la información necesaria para una transmisión eficaz de datos entre el externo y el programa COBOL. La serie de entradas debe finalizar con un punto de separación. Consulte “Párrafo I-O-CONTROL” en la página 132.

Párrafo FILE-CONTROL

El párrafo FILE-CONTROL asocia cada archivo del programa COBOL con un de conjunto de datosexterno y especifica la organización de archivos, la modalidad de acceso y otra información.

Los formatos siguientes son para el párrafo FILE-CONTROL:

- Entradas de archivo secuencial
- Entradas de archivo indexado
- Entradas de archivo relativas

- Entradas de archivo secuencial de línea

La tabla siguiente lista los distintos tipos de archivos disponibles para los programas.

Tabla 7. ***Tipos de archivos***

Organización de archivo	**Sistema de archivos**
Secuencial	SFS, STL, RSD[1], Db2, QSAM[2], MONGO
Relativo	SFS, STL, Db2, MONGO
Indexado	SFS, STL, Db2, MONGO
secuencia de línea	LSQ

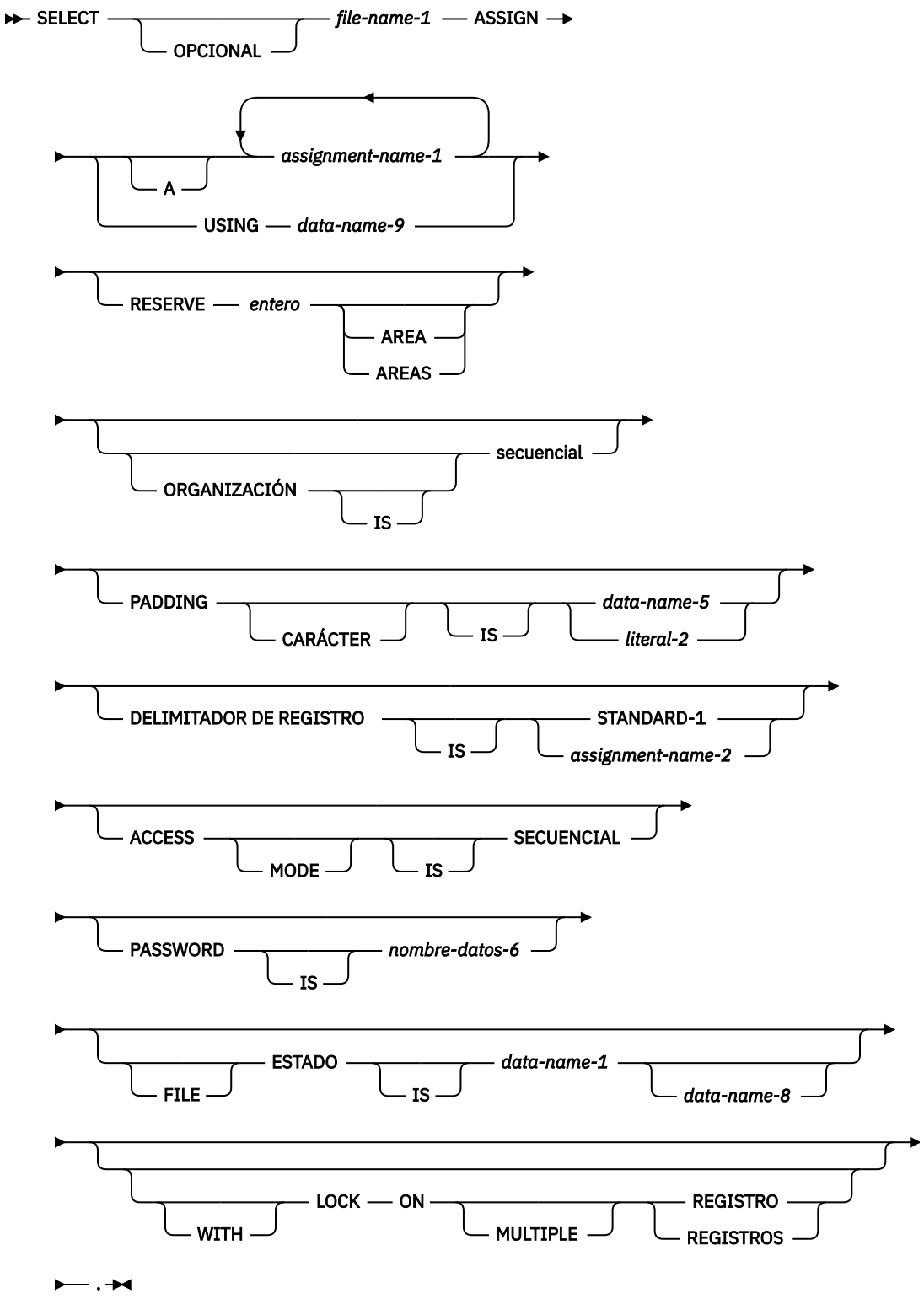
1. El sistema de archivos RSD soporta archivos secuenciales de registro de longitud fija o variable.
2. El sistema de archivos QSAM soporta registros fijos, variables y expandido.
3. Si especifica el tipo de archivo SdU , se manejará como si hubiera especificado el archivo STL.

El párrafo FILE-CONTROL comienza con la palabra FILE-CONTROL seguida de un punto separador. Debe contener una y sólo una entrada para cada archivo descrito en una entrada FD o SD en DATA DIVISION.

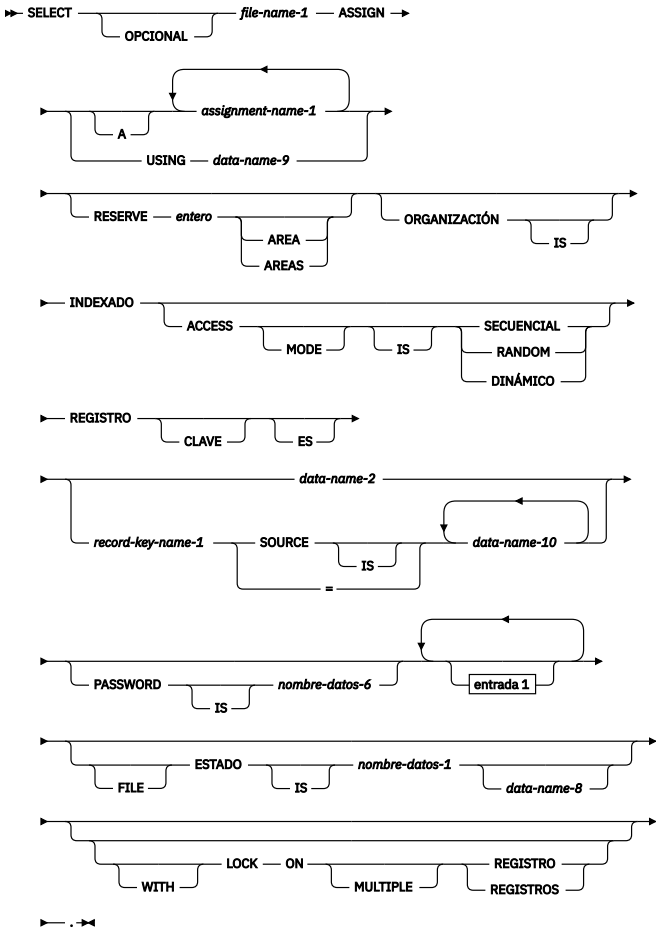
Dentro de cada entrada, la cláusula SELECT debe aparecer en primer lugar. Las otras cláusulas pueden aparecer en cualquier orden.

El subrayado está permitido en el componente *nombre-archivo-externo* de *assignment-name-1*.

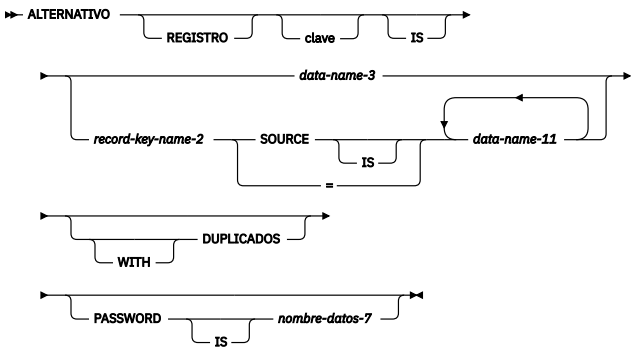
Formato 1: entrada-control-archivo-secuencial



Formato 2: entrada-control-archivo-indexado

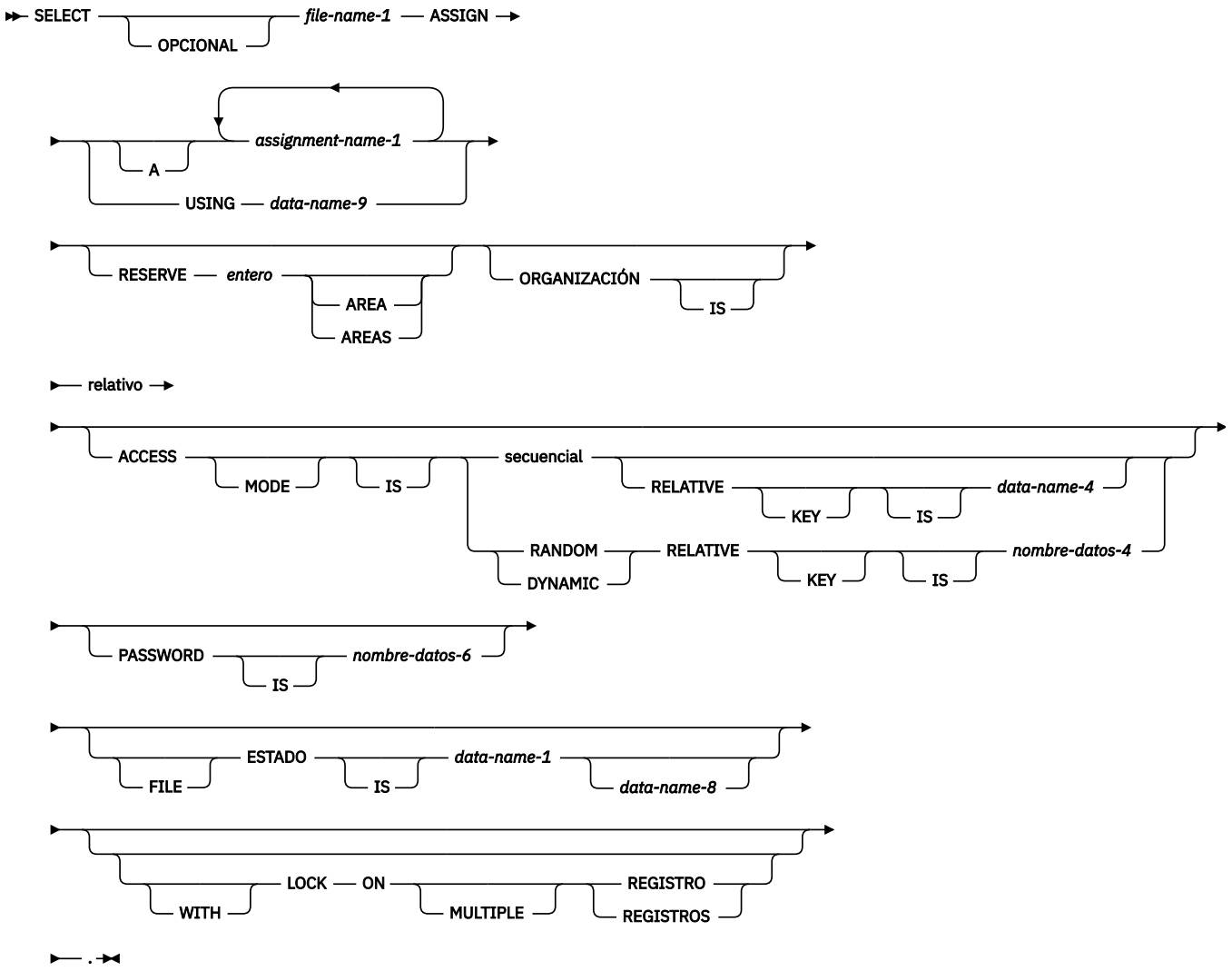


entrada 1

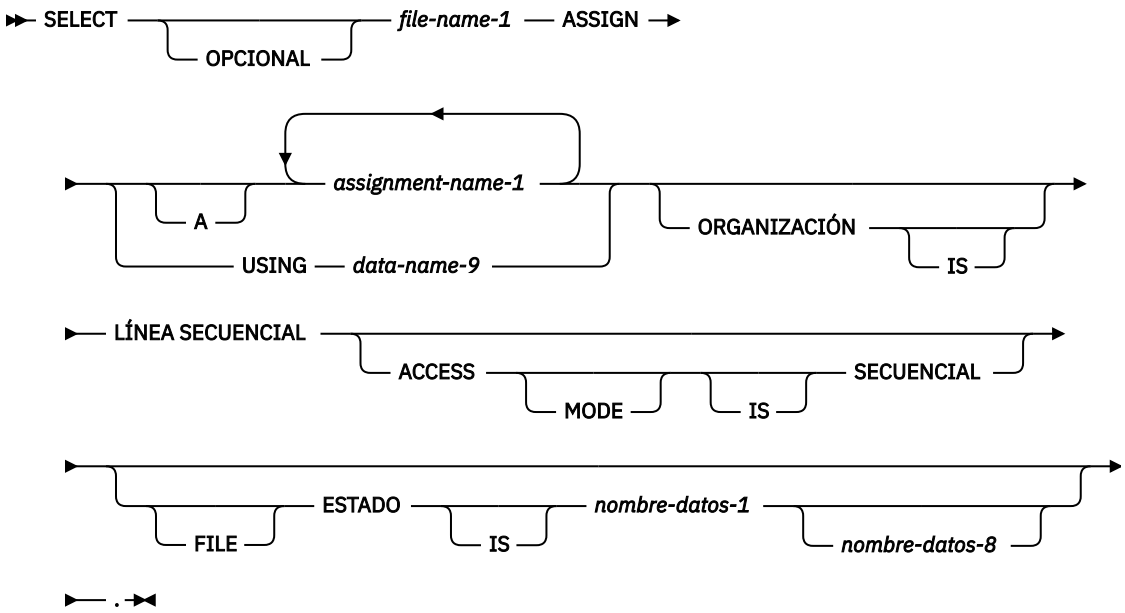


Restricción: *nombre-clave-registro* sólo está soportado para el sistema de archivos STL.

Formato 3: entrada-control-archivo-relativo



Formato 4: entrada-control-archivo-secuencial-línea



cláusula SELECT

La cláusula SELECT identifica un archivo en el programa COBOL que se va a asociar con un de conjunto de datosexterno.

SELECT OPCIONAL
Sólo se puede especificar para archivos abiertos en la modalidad de entrada, I-O o de ampliación. Debe especificar SELECT OPTIONAL para los archivos de entrada que no están necesariamente disponibles cada vez que se ejecuta el programa objeto. Para obtener más información, consulte el apartado "Notas de la sentencia OPEN" en la página 359.

file-name-1
Debe identificarse mediante una entrada FD o SD en DATA DIVISION. Un nombre de archivo debe cumplir las reglas para un nombre definido por el usuario COBOL, debe contener al menos un carácter alfabético y debe ser exclusivo dentro de este programa.

Cuando *file-name-1* especifica una clasificación o un archivo de fusión, sólo la cláusula ASSIGN puede seguir a la cláusula SELECT.

Si el conector de archivo al que hace referencia *file-name-1* es un conector de archivo externo, todas las entradas de control de archivo de la unidad de ejecución que hacen referencia a este conector de archivo deben tener la misma especificación para la frase OPTIONAL.

cláusula ASSIGN

La cláusula ASSIGN asocia un nombre de archivo interno en un programa COBOL con un nombre de archivo del sistema.

assignment-name-1

Se puede especificar como una palabra definida por el usuario o como un literal alfanumérico. Ambas formas constan de hasta tres componentes, separados por guiones. De izquierda a derecha:

1. Una serie de comentario opcional
2. Un ID de sistema de archivos opcional

3. Un nombre de archivo externo necesario si se ha especificado *assignment-name-1* como palabra definida por el usuario; un nombre de archivo del sistema necesario si se ha especificado *assignment-name-1* como literal

Para obtener más detalles sobre cómo se determina el componente de nombre de archivo externo de *assignment-name-1* , consulte "Nombre de asignación para palabras y literales definidos por el usuario" en la página 115.

Palabra definida por el usuario

assignment-name-1 debe seguir las reglas para una palabra COBOLy pueden tener hasta 30 caracteres de un solo byte de longitud. La palabra definida por el usuario no puede ser la misma que una palabra reservada, pero puede ser la misma que cualquier otra palabra definida por el usuario en el programa, incluido el nombre de un elemento de datos. El componente de nombre de archivo externo se trata en tiempo de ejecución inicialmente como el nombre de una variable de entorno; a continuación, si la variable de entorno no se establece, o se establece en la serie vacía, el componente se trata directamente como el nombre de archivo del sistema:

- **Nombre de archivo externo:** cada vez que se abre un archivo COBOL, se utiliza el nombre de archivo externo como nombre de una variable de entorno. Si la variable de entorno se establece en un valor no vacío, el valor se trata como el nombre de archivo del sistema o como el nombre de archivo del sistema precedido por un ID de sistema de archivos. Para obtener detalles, consulte "Nombre de asignación para nombres de datos y variables de entorno" en la página 118. Si se especifica un ID de sistema de archivos en la palabra definida por el usuario y también en la variable de entorno, la variable de entorno tiene prioridad. Para obtener más información sobre la prioridad del sistema de archivos, consulte *Prioridad de determinación de sistema de archivos* en la publicación *COBOL for Linux en x86 Guía de programación*.
- **Nombre de archivo del sistema:** Si la variable de entorno indicada por el nombre de archivo externo no está establecida, la palabra definida por el usuario se trata como:
  - El nombre de archivo del sistema
  - El nombre de archivo del sistema precedido por un ID de sistema de archivos
  - El nombre de archivo del sistema precedido por un ID de sistema de archivos precedido por un comentario

  Para obtener detalles, consulte "Nombre de asignación para palabras y literales definidos por el usuario" en la página 115.

Literal

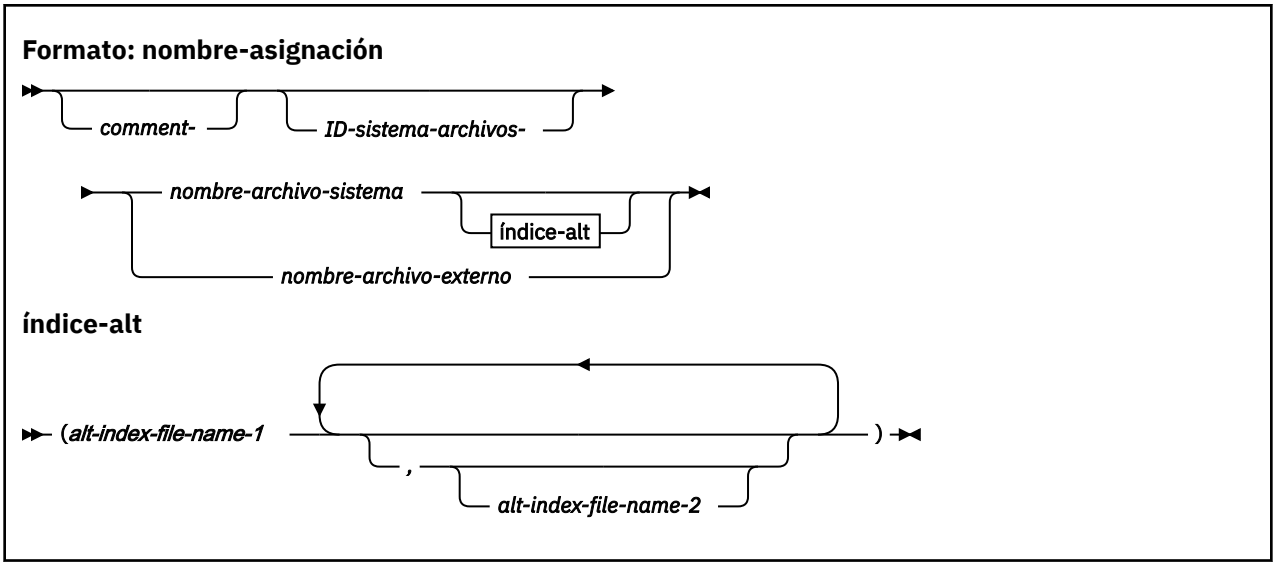
assignment-name-1 debe seguir las reglas para un literal alfanumérico COBOL. Todos los caracteres especificados dentro de los delimitadores literales se utilizan sin ninguna correlación. No se realiza ninguna comprobación en tiempo de ejecución para la existencia de una variable de entorno. Para obtener detalles, consulte "Nombre de asignación para palabras y literales definidos por el usuario" en la página 115.

USING *data-name-9*

Debe definirse en la sección de almacenamiento de trabajo como un elemento de datos de categoría alfanumérica y no debe estar subordinado a la descripción de archivo para *file-name-1*. El contenido de *data-name-9* se evalúa cada vez que se abre el archivo para determinar la información de asignación. En cuanto a un valor de variable de entorno, la información de asignación se trata como el nombre de archivo del sistema, o como el nombre de archivo del sistema precedido por un ID de sistema de archivos. Para obtener detalles, consulte "Nombre de asignación para nombres de datos y variables de entorno" en la página 118.

Nombre de asignación para palabras y literales definidos por el usuario

Si no se especifica USING, la palabra o literal definidos por el usuario que se especifica para el nombre de asignación se procesa tal como se describe a continuación.



Si la palabra definida por el usuario no contiene guiones, toda la serie se trata como el nombre de archivo externo. Si el literal no contiene ningún guión, toda la serie se trata como *información de nombre de archivo*. La información de nombre de archivo puede ser un único nombre de archivo del sistema, o un único nombre de archivo del sistema seguido de una lista de nombres de archivo de índice alternativos, o una concatenación de nombres de archivo del sistema separados por dos puntos. Para cada una de estas alternativas, cada nombre de archivo del sistema se puede calificar mediante un nombre de vía de acceso. De lo contrario, la palabra o literal definidos por el usuario se particiona en un máximo de tres componentes separados por guiones:

comentario
Los caracteres a la izquierda del ID de sistema de archivos si se especifica, o los caracteres a la izquierda del nombre de archivo externo o del nombre de archivo del sistema si no hay ningún ID de sistema de archivos válido se interpretan como un comentario.

ID-sistema-archivos

Serie de tres caracteres que especifica el sistema de archivos en el que se almacena un archivo y a través del cual se accede a él.

Si el componente situado más a la derecha cumple todos los criterios siguientes, los tres primeros caracteres de la serie, doblados a mayúsculas, se interpretan como el ID del sistema de archivos:

- La serie consta de tres o más caracteres.
- Los tres primeros caracteres (los más a la izquierda) son alfanuméricos (A-Z, a-z o 0-9).
- El primer carácter es alfabético (A-Z o a-z).

Si la serie no cumple todos estos criterios, se trata como parte del comentario.

***nombre-archivo-externo* o información de nombre de archivo**
El componente situado más a la derecha es el nombre de archivo externo o una de las tres formas de información de nombre de archivo.

Debido a que el guión se utiliza como separador, un nombre de archivo externo o nombre de archivo del sistema que contiene uno o más guiones no se puede especificar directamente con una palabra definida por el usuario o un literal. Para especificar un nombre de archivo externo o un nombre de archivo del sistema que contenga uno o más guiones, utilice uno de los métodos siguientes:

- Establezca la variable de entorno indicada por el nombre de archivo externo en un valor adecuado en tiempo de ejecución.
- Especifique el formato USING *data-name-9* de la cláusula ASSIGN y mueva un valor adecuado a *data-name-9* antes de abrir el archivo.

Si se especifica *assignment-name-1* como una palabra definida por el usuario, *nombre-archivo-externo* se interpreta en tiempo de ejecución como el nombre de una variable de entorno establecida en un valor no vacío o, si no existe dicha variable de entorno, directamente como el nombre-archivo-sistema.

Si se especifica *assignment-name-1* como un literal, el componente situado más a la derecha siempre se interpreta como información de nombre de archivo directamente.

Ejemplos:

Tabla 8. ***Nombre de asignación para palabras definidas por el usuario***

Palabra definida por el usuario	Comentario	ID de sistema de archivos	Nombre de archivo externo
`Read-Only-STL-Orders`	`Read-Only`	`STL`	`ORDERS`
`This-is-my-file`	`This-is-my`	VSAM (Valor predeterminado = STL)	`FILE`
`Comment-STL--file`	`Comment-STL-`	VSAM (Valor predeterminado = STL)	`FILE`
`Watch-for-this-file`	`Watch-for`	`THI` (no válido)	`FILE`

Tabla 9. ***Nombre de asignación para literales***

Literal	Comentario	ID de sistema de archivos	Nombre de archivo del sistema
`Read-Only-STL-Orders`	`Read-Only`	`STL`	`Orders`
`Eh?-What's this?`	`Eh?`	VSAM (Valor predeterminado = STL)	`What's this?`
`This-is-a-Db2-CICS.FILE`	`This-is-a`	`Db2`	`FILE` (bajo esquema `CICS`)
`I-Like-STL!-`	`I-Like`	`STL`	(Nulo)
`vsa-/.:/cics/sfs/svr/W123`	(Ninguno)	`SFS`	`W123` (en el servidor SFS `svr`)

Especificación de índices alternativos: el compilador normalmente asigna los nombres de archivo de índices alternativos adecuados; sin embargo, debe proporcionar nombres de archivo de índices alternativos para:

- Archivos SFS indexados que tienen índices alternativos. Cada nombre de archivo de índice tiene este formato:

```
/.:/cics/sfs/sfsServer/base-file-name;index-file-name
```

- Archivos SdU indexados que tienen archivos de índice alternativos con nombres que no son los valores predeterminados del compilador COBOL. Por ejemplo, un archivo que se crea a través de un lenguaje diferente, como por ejemplo PL/I.

Los nombres de archivo de índice alternativos, si se especifican, deben especificarse en el mismo orden en que se especifican las claves de registro alternativas en el programa fuente. Puede omitir nombres de archivo de índice alternativos, pero cualquier otro nombre de archivo de índice alternativo debe corresponder a la posición en la definición de archivo. El ejemplo siguiente muestra cómo especificar el primer y tercer nombre de archivo de índice alternativo:

```
base-file-name(first-index-file-name,,third-index-file-name)
```

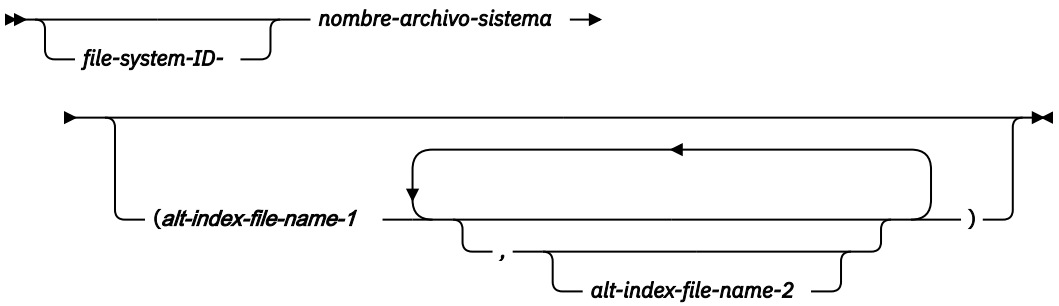
En este ejemplo, el compilador asigna un nombre de archivo predeterminado para el segundo archivo de índice alternativo.

Los nombres de archivo de índice alternativos se ignoran para sistemas de archivos que no sean SdU o SFS.

Nombre de asignación para nombres de datos y variables de entorno

Si se especifica una variable de entorno o un nombre de datos para el nombre de asignación, el valor del nombre de datos o la variable de entorno se procesa tal como se describe a continuación.

Formato: variable de entorno y valores de nombre de datos



El valor de *data-name-9* o la variable de entorno se evalúa cada vez que se ejecuta una sentencia OPEN para el archivo COBOL correspondiente. Si el valor no contiene ningún guión, toda la serie se trata como *información de nombre de archivo*. La información de nombre de archivo puede ser un único nombre de archivo del sistema, o un único nombre de archivo del sistema seguido de una lista de nombres de archivo de índice alternativos, o una concatenación de nombres de archivo del sistema separados por dos puntos. Para cada una de estas alternativas, cada nombre de archivo del sistema se puede calificar mediante un nombre de vía de acceso. Si el valor contiene uno o más guiones con al menos un carácter después del guión situado más a la izquierda, y la serie a la izquierda del guión situado más a la izquierda cumple todos los criterios siguientes, el valor se interpreta como *file-system-ID*, seguido de la información de nombre de archivo:

- La serie consta de tres o más caracteres.
- Los tres primeros caracteres (los más a la izquierda) son alfanuméricos (A-Z, a-z o 0-9).
- El primer carácter es alfabético (A-Z o a-z).

ID-sistema-archivos
Los tres primeros caracteres de *file-system-ID*, doblados a mayúsculas, se interpretan como el ID del sistema de archivos, sujeto a validación en tiempo de ejecución. El valor de un ID de sistema de archivos que se especifica en tiempo de ejecución altera temporalmente cualquier ID de sistema de archivos que se especifique en la palabra definida por el usuario de *assignment-name-1*.

información de nombre de archivo
Una de las tres formas de información de nombre de archivo. Para obtener información sobre cómo especificar índices alternativos, consulte "Especificación de índices alternativos" en “Nombre de asignación para palabras y literales definidos por el usuario” en la página 115.

Los ejemplos de la tabla siguiente presuponen que la opción de tiempo de ejecución FILESYS se ha establecido en STLy, por lo tanto, el ID del sistema de archivos toma como valor predeterminado STL.

Tabla 10. ***Nombre de asignación para nombres de datos y variables de entorno***

Valor de variable de entorno o valor de nombre de datos	ID de sistema de archivos	Nombre de archivo del sistema
`my-fyle`	STL	`my-fyle`
`abc/my-fyle`	ABC (no válido)	`fyle`

Tabla 10. ***Nombre de asignación para nombres de datos y variables de entorno*** *(continuación)*		
Valor de variable de entorno o valor de nombre de datos	**ID de sistema de archivos**	**Nombre de archivo del sistema**
`payroll-fyle`	PAY (no válido)	`fyle`
`rsd-payroll-file`	RSD	`payroll-file`
`\\Strange (...) name!`	STL	`\Strange (...) name!`
`Db2-File1`	Db2	`File1`
`./Db2-File2`	STL	`./Db2-File2`
`sfs-/.:/cics/sfs/svr/F1`	SFS	`F1` (en el servidor SFS `svr`)
`STL-/.:/cics/sfs/svr/F1`	SFS	(Vía de acceso no válida)
`stl-file1:file2`	STL	`file1` y `file2` (concatenados)

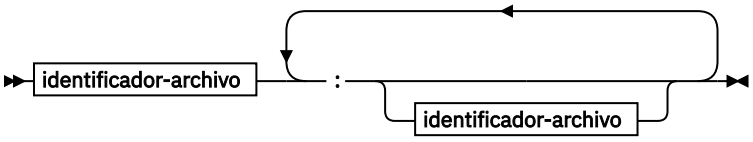
Para obtener más información sobre la identificación de archivos, consulte *Identificación de archivos* en la publicación *COBOL for Linux en x86 Guía de programación*.

Para obtener más información sobre el concepto de archivo y la terminología, consulte *Conceptos y terminología de archivos* en la publicación *COBOL for Linux en x86 Guía de programación*.

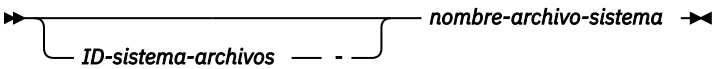
Concatenación de archivos

COBOL para Linux da soporte a la asignación de una concatenación de archivos a un archivo interno (*file-name-1* en la cláusula SELECT). La concatenación de archivos se especifica mediante varios identificadores de archivo separados por dos puntos (:).

Formato: identificador de archivo concatenado



identificador-archivo



ID-sistema-archivos
El sistema de archivos en el que existe *nombre-archivo-sistema* si *ID-sistema-archivos* consta de al menos tres caracteres alfanuméricos, y el primer carácter es alfabético. De lo contrario, el identificador de archivo completo se interpreta como el nombre del archivo en el sistema de archivos predeterminado.

nombre-archivo-sistema
El nombre del archivo en el sistema de archivos especificado si se especifica *ID-sistema-archivos* ; de lo contrario, el identificador de archivo completo se interpreta como el nombre del archivo en el sistema de archivos predeterminado. *nombre-archivo-sistema* puede incluir la información de unidad y vía de acceso para el archivo.

Por ejemplo:

```
export MYFILE='STL-/home/user1/file1:STL-/home/user1/file2'
```

En este ejemplo, MYFILE es el resultado de la concatenación de archivos de `/home/user1/file1` y `/home/user1/file2`, ambos en el sistema de archivos STL.

Un identificador de archivo no puede contener dos puntos (:) porque el identificador de archivo se interpretaría entonces como una concatenación de dos archivos separados.

Se pueden especificar hasta 256 identificadores de archivo en una concatenación. No es necesario que los identificadores de archivo sean exclusivos. Por ejemplo, una concatenación puede constar de 256 apariciones del mismo identificador de archivo. Los dos puntos adyacentes se tratan como un único dos puntos.

La concatenación de archivos está soportada para el valor de la variable de entorno, el contenido del elemento de datos ASSIGN USING y la forma literal del nombre de asignación.

Un archivo interno COBOL asignado a una concatenación de archivos debe cumplir los criterios siguientes:

- FILE ORGANIZATION es SECUENCIAL o LINE SEQUENTIAL.
- ACCESS MODE es SECUENCIAL.
- La modalidad de cualquier sentencia OPEN es INPUT.

Estos criterios se comprueban en tiempo de ejecución, incluso si la concatenación se especifica en el formato literal del nombre de asignación.

Si un signo de dos puntos precede o sigue a un único identificador de archivo, el archivo se convierte en miembro de una concatenación trivial y es de sólo lectura en cualquier programa que utilice el archivo, independientemente de sus permisos de archivo.

La concatenación está soportada para todos los sistemas de archivos. El ID del sistema de archivos puede especificarse en cualquiera o en todos los identificadores de archivo de una concatenación determinada. Sin embargo, todos los identificadores de archivo de una concatenación deben especificar o tomar por omisión el mismo sistema de archivos en tiempo de ejecución, y los archivos deben tener atributos coherentes.

Una concatenación puede incluir archivos de generación individuales o grupos de datos de generación completos (GDG). Si se especifica un GDG como miembro de una concatenación, los archivos individuales del grupo se leen en orden de generación, es decir, desde la generación más actual a la generación más antigua. Para obtener más información sobre grupos de datos de generación o archivos de generación, consulte *Grupos de datos de generación* en la publicación *COBOL for Linux en x86 Guía de programación*.

Una concatenación también puede incluir archivos vacíos, es decir, archivos que no contienen registros. Cuando se leen archivos vacíos, devuelven inmediatamente el final del archivo internamente y, por lo tanto, son invisibles para el programa que lee los archivos concatenados. Si la concatenación sólo contiene archivos vacíos, la primera operación READ devuelve el final del archivo y existe la condición AT END.

Si se define un archivo COBOL con la frase OPTIONAL en una cláusula SELECT, la concatenación puede incluir archivos no disponibles. Un archivo *no está disponible* si no existe o los permisos de archivo no permiten la operación solicitada. Un archivo no disponible se trata como un archivo vacío, por lo tanto es invisible para el programa.

Para obtener más información sobre la disponibilidad de archivos, consulte Tabla 52 en la página 359.

Para obtener más información sobre la concatenación de archivos, consulte *Concatenación de archivos* en la publicación *COBOL for Linux en x86 Guía de programación*.

cláusula RESERVE

La cláusula RESERVE es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

cláusula ORGANIZATION

La cláusula ORGANIZATION identifica la estructura lógica del archivo. La estructura lógica se establece en el momento en que se crea el archivo y no se puede cambiar posteriormente.

Puede encontrar una descripción de las distintas formas en las que se pueden organizar los datos y de los distintos métodos de acceso que puede utilizar para recuperar los datos en "Organización de archivos y modalidades de acceso" en la página 126.

ORGANIZATION IS SEQUENTIAL (formato 1)
Una relación predecesor-sucesor entre los registros del archivo se establece por el orden en el que los registros se colocan en el archivo cuando se crea o se amplía.

ORGANIZATION IS INDEXED (formato 2)
La posición de cada registro lógico en el archivo está determinada por los índices creados con el archivo y mantenidos por el sistema. Los índices se basan en claves incorporadas dentro de los registros del archivo.

ORGANIZATION IS RELATIVE (formato 3)
La posición de cada registro lógico en el archivo viene determinada por su número de registro relativo.

ORGANIZATION IS LINE SEQUENTIAL (formato 4)
Una relación predecesor-sucesor entre los registros del archivo se establece por el orden en el que se colocan los registros en el archivo cuando se crea o se amplía. Un registro de un archivo LINE SEQUENTIAL sólo puede constar de caracteres imprimibles.

Si omite la cláusula ORGANIZATION, el compilador presupone que ORGANIZATION ES SECUENCIAL.

Si el conector de archivo al que hace referencia *file-name-1* en la cláusula SELECT es un conector de archivo externo, se debe especificar la misma organización para todas las entradas de control de archivo en la unidad de ejecución que hacen referencia a este conector de archivo.

Organización de archivo

Establece la organización de los datos cuando crea un archivo. Una vez que se ha creado el archivo, puede expandirlo, pero no puede cambiar la organización.

Organización secuencial

El orden físico en el que se colocan los registros en el archivo determina la secuencia de registros. Las relaciones entre los registros del archivo no cambian, excepto que el archivo se puede ampliar. Los registros pueden ser de longitud fija o de longitud variable; no hay claves.

Cada registro del archivo excepto el primero tiene un registro predecesor exclusivo; y cada registro excepto el último tiene un registro sucesor exclusivo.

Organización indexada

Cada registro del archivo tiene una o más claves incorporadas (a las que se hace referencia como *elementos de datos de clave*); cada clave está asociada a un índice. Un índice proporciona una vía de acceso lógica a los registros de datos de acuerdo con el contenido de los elementos de datos de clave de registro incorporados asociados. Los archivos indexados deben ser archivos de almacenamiento de acceso directo. Los registros pueden ser de longitud fija o de longitud variable.

Cada registro de un archivo indexado debe tener un elemento de datos de clave principal incorporado. Cuando los registros se insertan, actualizan o suprimen, se identifican únicamente por los valores de sus claves principales. Por lo tanto, el valor de cada elemento de datos de clave principal debe ser exclusivo y no debe cambiarse cuando se actualiza el registro. Indique a COBOL el nombre del elemento de datos de clave principal en la cláusula RECORD KEY del párrafo de control de archivos.

Además, cada registro de un archivo indexado puede contener uno o más elementos de datos clave alternativos incorporados. Cada clave alternativa proporciona otro medio de identificar qué registro

recuperar. Indique a COBOL el nombre de cualquier elemento de datos de clave alternativo en la cláusula ALTERNATIVA RECORD KEY del párrafo de control de archivos.

La clave utilizada para cualquier solicitud de entrada-salida específica se conoce como *clave de referencia*.

Organización relativa

Piense en el archivo como una serie de áreas de registro, cada una de las cuales contiene un único registro. Cada área de registro se identifica mediante un número de registro relativo; el método de acceso almacena y recupera un registro basándose en su número de registro relativo. Por ejemplo, la primera área de registro se direcciona mediante el número de registro relativo 1 y la 10th se direcciona mediante el número de registro relativo 10. La secuencia física en la que se colocaron los registros en el archivo no tiene ninguna relación con el área de registro en la que se almacenan y, por lo tanto, no tiene ningún efecto en el número de registro relativo de cada registro. Los archivos relativos deben ser archivos de acceso directo. Los registros pueden ser de longitud fija o de longitud variable.

Organización secuencial de línea

En un archivo secuencial de línea, cada registro contiene una secuencia de caracteres que termina con un delimitador de registro. El delimitador no se cuenta en la longitud del registro.

Cuando se escribe un registro, los espacios en blanco finales se eliminan antes de añadir el delimitador de registro. Los caracteres del área de registro desde el primer carácter hasta el delimitador de registro añadido, inclusive, constituyen un registro y se graban en el archivo.

Cuando se lee un registro, los caracteres se leen de uno en uno en el área de registro hasta:

- Se ha encontrado el primer delimitador de registro. El delimitador de registro se descarta y el resto del registro se rellena con espacios.
- Toda el área de registro se rellena con caracteres. Si el primer carácter no leído es el delimitador de registro, se descarta. De lo contrario, el primer carácter no leído se convierte en el primer carácter leído por la siguiente sentencia READ.
- Se ha encontrado un final de archivo. El resto del área de registro se rellena con espacios.

La longitud de registro está determinada por uno de los siguientes:

- La longitud de lectura solicitada
- La ubicación del carácter de nueva línea (*n*)

El único carácter especial de un archivo secuencial de línea es el carácter de nueva línea (*n*). Cualquier otro carácter que incluya NULL (0x00) es válido y no provocará errores de lectura o grabación.

Los registros escritos en archivos secuenciales de línea deben constar de elementos de datos descritos como USAGE DISPLAY o DISPLAY-1 o una combinación de elementos DISPLAY y DISPLAY-1 . Si la opción de compilador CHAR (EBCDIC) está en vigor, se puede codificar un elemento DISPLAY o DISPLAY-1 en ASCII o EBCDIC, en función de la presencia o ausencia de la frase NATIVE en la cláusula USAGE del elemento de datos. Un elemento de datos decimal con zona debe estar sin signo o, si está firmado, debe declararse con la frase SEPARADAS CHARACTER.

Un archivo secuencial de línea puede contener caracteres imprimibles y caracteres de control. No obstante, tenga en cuenta que si el archivo contiene un carácter de salto de línea (X'0A'), el carácter de salto de línea funcionará como delimitador de registro.

Las cláusulas siguientes no están soportadas para los archivos secuenciales de línea:

- Cláusula APPLY WRITE-ONLY
- cláusula CODE-SET
- cláusula DATA RECORDS
- cláusula LABEL RECORDS
- cláusula LINAGE

- Frase I-O de la sentencia OPEN
- cláusula PADDING CHARACTER
- Cláusula RECORD CONTAINS 0
- RECORD CONTAINS formato de cláusula 2 (por ejemplo: RECORD CONTAINS 100 a 200 CHARACTERS)
- cláusula RECORD DELIMITER
- cláusula REGISTRO MODE
- cláusula RERUN
- cláusula RESERVE
- Frase REVERSED de la sentencia OPEN
- Sentencia REWRITE
- Cláusula VALUE OF de entrada de descripción de archivo
- WRITE ... DESPUÉS DE AVANZAR *nombre-mnemotécnico*
- WRITE ... AT END-OF-PAGE
- WRITE ... ANTES DE AVANZAR

Elementos de lenguaje tratados como comentarios

Para otros archivos (secuenciales, relativos e indexados), los siguientes elementos de lenguaje son comprobación de sintaxis, pero no tienen ningún efecto en la ejecución del programa:

- Cláusula APPLY WRITE-ONLY
- CERRAR ... PARA ELIMINACIÓN
- CERRAR ... SIN REBOBINAR
- cláusula CODE-SET
- cláusula DATA RECORDS
- cláusula LABEL RECORDS
- cláusula MULTIPLE FILE TAPE
- Abrir ... Inverso
- cláusula PADDING CHARACTER
- cláusula PASSWORD
- Cláusula RECORD CONTAINS 0
- cláusula RECORD DELIMITER
- Cláusula REGISTRO MODE (para archivos relativos e indexados)
- cláusula RERUN
- cláusula RESERVE
- cláusula SAME AREA
- cláusula SAME SORT AREA
- Cláusula SAME SORT-MERGE AREA
- Cláusula VALUE OF de entrada de descripción de archivo

No se generan mensajes de error (con la excepción de la opción de nombre de datos para LABEL RECORDS, USE ... DESPUÉS ... LABEL PROCEDURE y GO TO MORE-LABELS).

cláusula PADDING CHARACTER

La cláusula PADDING CHARACTER especifica un carácter que debe utilizarse para el relleno de bloques en archivos secuenciales.

data-name-5
Debe definirse en DATA DIVISION como un elemento de datos de un carácter de categoría alfabético, alfanumérico o nacional, y no debe definirse en FILE SECTION. *data-name-5* puede calificarse.

literal-2
Debe ser un literal alfanumérico de un carácter o un literal nacional.

Para archivos externos, *data-name-5*, si se especifica, debe hacer referencia a un elemento de datos externo.

La cláusula PADDING CHARACTER se comprueba con la sintaxis, pero no tiene ningún efecto en la ejecución del programa.

cláusula RECORD DELIMITER

La cláusula RECORD DELIMITER indica el método para determinar la longitud de un registro de longitud variable en un medio externo. Sólo se puede especificar para registros de longitud variable.

STANDARD-1
Si se especifica STANDARD-1 , el soporte externo debe ser un archivo de cinta magnética.

assignment-name-2
Puede ser cualquier palabra COBOL.

La cláusula RECORD DELIMITER se comprueba con la sintaxis, pero no tiene ningún efecto en la ejecución del programa.

cláusula ACCESS MODE

La cláusula ACCESS MODE define la forma en que los registros del archivo están disponibles para su proceso. Si no se especifica la cláusula ACCESS MODE, se presupone el acceso secuencial.

Para los archivos relativos a los que se accede secuencialmente, la cláusula ACCESS MODE no tiene que preceder a la cláusula RELATIVE KEY.

LA MODALIDAD DE ACCESO ES SECUENCIAL
Se puede especificar en todos los formatos.

Formato 1: secuencial
Se accede a los registros del archivo en la secuencia establecida cuando se crea o se amplía el archivo. El formato 1 sólo da soporte al acceso secuencial.

Formato 2: indexado
Se accede a los registros del archivo en la secuencia de valores de clave de registro ascendente de acuerdo con la secuencia de clasificación del archivo.

Formato 3: relativo
Se accede a los registros del archivo en la secuencia ascendente de números de registro relativos de registros existentes en el archivo.

Formato 4: secuencial de línea
Se accede a los registros del archivo en la secuencia establecida cuando se crea o se amplía el archivo. El formato 4 sólo da soporte al acceso secuencial.

A continuación se muestra la sintaxis de la modalidad de acceso secuencial:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT file-name ASSIGN TO dd-name
   ORGANIZATION IS SEQUENTIAL
   ACCESS MODE IS SEQUENTIAL


ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT file-name ASSIGN TO dd-name
   ORGANIZATION IS INDEXED
```

```
   ACCESS MODE IS SEQUENTIAL
   RECORD KEY IS rec-key1
   ALTERNATE RECORD KEY IS rec-key2


ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT file-name ASSIGN TO dd-name
   ORGANIZATION IS RELATIVE
   ACCESS MODE IS SEQUENTIAL
   RELATIVE KEY IS rec-key1
```

LA MODALIDAD DE ACCESO ES ALEATORIA

Sólo se puede especificar en los formatos 2 y 3.

Formato 2: indexado

El valor colocado en un elemento de datos de clave de registro especifica el registro al que se va a acceder.

Formato 3: relativo

El valor colocado en un elemento de datos de clave relativa especifica el registro al que se va a acceder.

A continuación se muestra la sintaxis de la modalidad de acceso aleatorio:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT file-name ASSIGN TO dd-name
   ORGANIZATION IS INDEXED
   ACCESS MODE IS RANDOM
   RECORD KEY IS rec-key1
   ALTERNATE RECORD KEY IS rec-key2


ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT file-name ASSIGN TO dd-name
   ORGANIZATION IS RELATIVE
   ACCESS MODE IS RANDOM
   RELATIVE KEY IS rec-key1
```

LA MODALIDAD DE ACCESO ES DINÁMICA

Sólo se puede especificar en los formatos 2 y 3.

Formato 2: indexado

Se puede acceder a los registros del archivo de forma secuencial o aleatoria, en función del formato de la sentencia de entrada-salida específica utilizada.

Formato 3: relativo

Se puede acceder a los registros del archivo de forma secuencial o aleatoria, en función del formato de la solicitud de entrada-salida específica.

A continuación se muestra la sintaxis de la modalidad de acceso dinámico:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT file-name ASSIGN TO dd-name
   ORGANIZATION IS SEQUENTIAL
   ACCESS MODE IS DYNAMIC
   RECORD KEY IS rec-key1
   ALTERNATE RECORD KEY IS rec-key2


ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
   FILE-CONTROL.
   SELECT file-name ASSIGN TO dd-name
   ORGANIZATION IS RELATIVE
   ACCESS MODE IS DYNAMIC
   RELATIVE KEY IS rec-key1
```

Organización de archivos y modalidades de acceso

Organización de archivos es la estructura lógica permanente del archivo. Indique al sistema cómo recuperar registros del archivo especificando la *modalidad de acceso* (secuencial, aleatoria o dinámica).

Para obtener detalles sobre los métodos de acceso y la organización de datos, consulte Tabla 7 en la página 110.

Sólo se puede acceder secuencialmente a los datos organizados secuencialmente; sin embargo, se puede acceder a los datos que tienen una organización indexada o relativa en cualquiera de las tres modalidades de acceso.

Modalidades de acceso

Consulte las descripciones de los siguientes tipos de modalidades de acceso.

Modalidad de acceso secuencial
Permite leer y grabar registros de un archivo de forma seriada; el orden de referencia viene determinado implícitamente por la posición de un registro en el archivo.

Modalidad de acceso aleatorio
Permite leer y escribir registros de una forma especificada por el programador; el control de referencias sucesivas al archivo se expresa mediante claves definidas específicamente suministradas por el usuario.

Modalidad de acceso dinámico
Permite que la sentencia de entrada-salida específica determine la modalidad de acceso. Por lo tanto, los registros se pueden procesar secuencial o aleatoriamente o ambos.

Para los archivos externos, cada entrada de control de archivos de la unidad de ejecución asociada con ese archivo externo debe especificar la misma modalidad de acceso. Además, para las entradas de archivo relativas, *data-name-4* debe hacer referencia a un elemento de datos externo, y la frase RELATIVE KEY de cada entrada de control de archivo asociada debe hacer referencia a ese mismo elemento de datos externo.

Relación entre organizaciones de datos y modalidades de acceso

Esta sección describe qué modalidades de acceso son válidas para cada tipo de organización de datos.

Archivos secuenciales
Sólo se puede acceder secuencialmente a los archivos con organización secuencial. La secuencia en la que se accede a los registros es el orden en el que se escribieron originalmente los registros.

Archivos secuenciales de línea
Igual que para los archivos secuenciales (descritos anteriormente).

Archivos indexados
Se permiten las tres modalidades de acceso.

En la modalidad de acceso secuencial, la secuencia en la que se accede a los registros es el orden ascendente (o, opcionalmente, descendente) del valor de clave de registro. El orden de recuperación dentro de un conjunto de registros que tienen valores de clave de registro alternativos duplicados es el orden en el que se han grabado los registros en el conjunto.

En la modalidad de acceso aleatorio, puede controlar la secuencia en la que se accede a los registros. Se accede a un registro específico colocando el valor de su clave o claves en el elemento de datos RECORD KEY (y el elemento de datos ALTERNATE RECORD KEY). Si un conjunto de registros tiene valores de clave de registro alternativo duplicados, sólo está disponible el primer registro grabado.

En la modalidad de acceso dinámico, puede cambiar según sea necesario de acceso secuencial a acceso aleatorio utilizando los formatos adecuados de sentencias de entrada-salida.

Archivos relativos
Se permiten las tres modalidades de acceso.

En la modalidad de acceso secuencial, la secuencia en la que se accede a los registros es el orden ascendente (o, opcionalmente, descendente) de los números de registro relativos de todos los registros que existen en el archivo.

En la modalidad de acceso aleatorio, puede controlar la secuencia en la que se accede a los registros. Se accede a un registro específico colocando su número de registro relativo en el elemento de datos RELATIVE KEY; RELATIVE KEY no debe definirse dentro de la entrada de descripción de registro para el archivo.

En la modalidad de acceso dinámico, puede cambiar según sea necesario de acceso secuencial a acceso aleatorio utilizando los formatos adecuados de sentencias de entrada-salida.

cláusula RECORD KEY

La cláusula RECORD KEY (formato 2) especifica el elemento de datos dentro del registro que es la RECORD KEY principal para un archivo indexado. Los valores contenidos en el elemento de datos RECORD KEY primo deben ser exclusivos entre los registros del archivo.

data-name-2

El elemento de datos CLAVE DE REGISTRO principal.

data-name-2 debe describirse dentro de una entrada de descripción de registro asociada con el archivo. La clave puede tener cualquiera de las siguientes categorías de datos:

- Alfanumérico
- Numérico
- Numérico editado (con uso DISPLAY o NATIONAL)
- Alfanumérico-editado
- Alfabético
- Coma flotante externa (con uso DISPLAY o NATIONAL)
- Coma flotante interna
- DBCS
- Nacional
- Nacional-editado

Independientemente de la categoría del elemento de datos clave, la clave se trata como un elemento alfanumérico. El orden de clasificación de la clave viene determinado por el orden de valor binario del artículo cuando se utiliza la clave para localizar un registro o para establecer el indicador de posición de archivo asociado con el archivo.

data-name-2 no puede ser un campo de fecha con ventana.

data-name-2 no debe hacer referencia a un elemento de datos de longitud variable. *data-name-2* puede calificarse.

Si el archivo indexado contiene registros de longitud variable, no es necesario que *data-name-2* esté contenido en el tamaño mínimo de registro especificado para el archivo. Es decir, *data-name-2* puede superar el tamaño mínimo de registro, pero no se recomienda.*nombre-datos-2* estará contenido en los primeros *n* bytes del registro, donde *n* es igual al tamaño mínimo de registro especificado para el archivo. *data-name-2* puede sobrepasar el tamaño mínimo de registro, pero no se recomienda. Para obtener más información, consulte el apartado “cláusula RECORD” en la página 160.

La descripción de datos de *data-name-2* y su ubicación relativa dentro del registro deben ser las mismas que las utilizadas cuando se definió el archivo.

Si el archivo tiene más de una entrada de descripción de registro, *data-name-2* debe describirse sólo en una de esas entradas de descripción de registro. Las posiciones de caracteres idénticas a las que hace referencia *data-name-2* en cualquier entrada de descripción de registro se referencian implícitamente como claves para todas las demás entradas de descripción de registro para ese archivo.

Para los archivos definidos con la cláusula EXTERNAL, todas las entradas de descripción de archivo de la unidad de ejecución que están asociadas con el archivo deben tener entradas de descripción de datos para *data-name-2* que especifiquen la misma ubicación relativa en el registro y la misma longitud.

record-key-name-1

record-key-name-1 tiene la clase y la categoría de *data-name-10*.

Restricción: *nombre-clave-registro* sólo está soportado para el sistema de archivos STL.

data-name-10

data-name-10 debe describirse dentro de una entrada de descripción de registro asociada con el archivo. La clave puede tener cualquiera de las siguientes categorías de datos:

- Alfanumérico
- Numérico
- Numérico editado (con uso DISPLAY o NATIONAL)
- Alfanumérico-editado
- Alfabético
- Coma flotante externa (con uso DISPLAY o NATIONAL)
- Coma flotante interna
- DBCS
- Nacional
- Nacional-editado

Independientemente de la categoría del elemento de datos clave, la clave se trata como un elemento alfanumérico. El orden de clasificación de la clave viene determinado por el orden de valor binario del artículo cuando se utiliza la clave para localizar un registro o para establecer el indicador de posición de archivo asociado con el archivo.

Todas las apariciones de *data-name-10* deben ser de la misma categoría.

data-name-10 no puede ser un campo de fecha con ventana.

data-name-10 no debe hacer referencia a un elemento de datos de longitud variable. *data-name-10* se puede calificar.

Si el archivo indexado contiene registros de longitud variable, *data-name-10* debe estar contenido en los primeros *n* bytes del registro, donde *n* es igual al tamaño mínimo de registro especificado para el archivo. Para obtener más información, consulte el apartado “cláusula RECORD” en la página 160.

Si el conector de archivo al que hace referencia *file-name-1* en la cláusula SELECT es un conector de archivo externo, todas las entradas de control de archivos de la unidad de ejecución que hacen referencia a este conector de archivos deben tener la misma entrada de descripción de datos para *data-name-2* y cada *data-name-10* , así como su ubicación relativa dentro del registro asociado. Para obtener detalles sobre el conector de archivo externo, consulte “cláusula EXTERNAL” en la página 176.

Cláusula ALTERNATIVA RECORD KEY

La cláusula ALTERNATIVE RECORD KEY (formato 2) especifica un elemento de datos dentro del registro que proporciona una vía de acceso alternativa a los datos de un archivo indexado.

data-name-3

Un elemento de datos CLAVE DE REGISTRO ALTERNATIVO.

data-name-3 debe describirse dentro de una entrada de descripción de registro asociada con el archivo. La clave puede tener cualquiera de las siguientes categorías de datos:

- Alfanumérico
- Numérico
- Numérico editado (con uso DISPLAY o NATIONAL)
- Alfanumérico-editado
- Alfabético
- Coma flotante externa (con uso DISPLAY o NATIONAL)
- Coma flotante interna
- DBCS
- Nacional
- Nacional-editado

Independientemente de la categoría del elemento de datos clave, la clave se trata como un elemento alfanumérico. El orden de clasificación de la clave viene determinado por el orden de valor binario del artículo cuando se utiliza la clave para localizar un registro o para establecer el indicador de posición de archivo asociado con el archivo.

data-name-3 no puede ser un campo de fecha con ventana.

data-name-3 no debe hacer referencia a un elemento de grupo que contenga un elemento de datos de ocurrencia variable. *data-name-3* se puede calificar.

data-name-3 no debe hacer referencia a los siguientes elementos de datos:

- Un elemento de datos de longitud variable.
- Elemento cuya posición de byte más a la izquierda corresponde a la posición de byte más a la izquierda de la clave de registro principal o de otra clave de registro alternativa. Esta restricción no se aplica si se especifica alguna de las claves utilizando la frase SOURCE.

Si el archivo indexado contiene registros de longitud variable, no es necesario que *data-name-3* esté incluido en el tamaño mínimo de registro especificado para el archivo. Es decir, *data-name-3* puede superar el tamaño mínimo de registro, pero no se recomienda.

Si el archivo indexado contiene registros de longitud variable, no es necesario que *data-name-3* esté incluido en el tamaño mínimo de registro especificado para el archivo. Es decir, *data-name-3* puede superar el tamaño mínimo de registro, pero no se recomienda.

Si el archivo indexado contiene registros de longitud variable, *data-name-3* se incluirá en los primeros *x* bytes del registro, donde *x* es igual al tamaño mínimo de registro especificado para el archivo. *data-name-3* puede superar el tamaño mínimo de registro, pero no se recomienda. Para obtener más información, consulte el apartado "cláusula RECORD" en la página 160.

La descripción de datos de *data-name-3* y su ubicación relativa dentro del registro deben ser las mismas que las utilizadas cuando se definió el archivo. El número de claves de registro alternativas para el archivo también debe ser el mismo que el utilizado cuando se creó el archivo.

La posición de carácter más a la izquierda de *data-name-3* no debe ser la misma que la posición de carácter más a la izquierda de la clave de registro principal o de otra clave de registro alternativa.

record-key-name-2

record-key-name-2 tiene la clase y la categoría de *data-name-11*.

Restricción: *nombre-clave-registro* sólo está soportado para el sistema de archivos STL.

data-name-11

data-name-11 debe describirse dentro de una entrada de descripción de registro asociada con el archivo. La clave puede tener cualquiera de las siguientes categorías de datos:

- Alfanumérico
- Numérico
- Numérico editado (con uso DISPLAY o NATIONAL)

- Alfanumérico-editado
- Alfabético
- Coma flotante externa (con uso DISPLAY o NATIONAL)
- Coma flotante interna
- DBCS
- Nacional
- Nacional-editado

Independientemente de la categoría del elemento de datos clave, la clave se trata como un elemento alfanumérico. El orden de clasificación de la clave viene determinado por el orden de valor binario del artículo cuando se utiliza la clave para localizar un registro o para establecer el indicador de posición de archivo asociado con el archivo.

Todas las apariciones de *data-name-11* deben ser de la misma categoría.

data-name-11 no puede ser un campo de fecha con ventana.

data-name-11 no debe hacer referencia a un elemento de datos de longitud variable. *data-name-11* puede calificarse.

Si el archivo indexado contiene registros de longitud variable, *data-name-11* debe estar contenido en los primeros *x* bytes del registro, donde *x* es igual al tamaño mínimo de registro especificado para el archivo. Para obtener más información, consulte el apartado “cláusula RECORD” en la página 160.

Si no se especifica la frase DUPLICATES, los valores contenidos en el elemento de datos ALTERNATIVA RECORD KEY deben ser exclusivos entre los registros del archivo.

Si se especifica la frase DUPLICATES, los valores contenidos en el elemento de datos ALTERNATIVA RECORD KEY pueden duplicarse dentro de cualquier registro del archivo. En el acceso secuencial, los registros con claves duplicadas se recuperan en el orden en el que se colocaron en el archivo. En el acceso aleatorio, sólo se puede recuperar el primer registro escrito en una serie de registros con claves duplicadas.

Para los archivos definidos con la cláusula EXTERNAL, todas las entradas de descripción de archivo de la unidad de ejecución que están asociadas con el archivo deben tener entradas de descripción de datos para *data-name-3* que especifiquen la misma ubicación relativa en el registro y la misma longitud. Las entradas de descripción de archivo deben especificar el mismo número de claves de registro alternativas y la misma frase DUPLICATES.

Si el conector de archivo al que hace referencia *file-name-1* en la cláusula SELECT es un conector de archivo externo, todas las entradas de control de archivos de la unidad de ejecución que hacen referencia a este conector de archivo deben tener la misma entrada de descripción de datos para *data-name-3* y cada *data-name-11* , así como su ubicación relativa dentro del registro asociado, el mismo número de claves de registro alternativas y la misma frase DUPLICATES. Para obtener detalles sobre el conector de archivo externo, consulte “cláusula EXTERNAL” en la página 176.

cláusula RELATIVE KEY

La cláusula RELATIVE KEY (formato 3) identifica un nombre de datos que especifica el número de registro relativo para un registro lógico específico dentro de un archivo relativo.

data-name-4
Debe definirse como un elemento de datos entero sin signo cuya descripción no contenga el símbolo P. *data-name-4* no debe estar definido en una entrada de descripción de registro asociada con este archivo relativo. Es decir, la CLAVE RELATIVA no forma parte del registro. *data-name-4* se puede calificar.

data-name-4 no puede ser un campo de fecha con ventana.

data-name-4 es necesario para ACCESS IS SEQUENTIAL sólo cuando se va a utilizar la sentencia START. Siempre es necesario para ACCESS IS RANDOM y ACCESS IS DYNAMIC. Cuando se ejecuta la

sentencia START, el sistema utiliza el contenido del elemento de datos RELATIVE KEY para determinar el registro en el que va a empezar el proceso secuencial.

Si un valor se coloca en *data-name-4*y no se ejecuta una sentencia START, el valor se ignora y el proceso empieza con el primer registro del archivo.

Si una sentencia START debe hacer referencia a un archivo relativo, debe especificar la cláusula RELATIVE KEY para ese archivo.

Para archivos externos, *data-name-4* debe hacer referencia a un elemento de datos externo, y la frase RELATIVE KEY de cada entrada de control de archivos asociada debe hacer referencia a ese mismo elemento de datos externo en cada caso.

La cláusula ACCESS MODE IS RANDOM no debe especificarse para los nombres de archivo especificados en la frase USING o GIVING de una sentencia SORT o MERGE.

cláusula PASSWORD

La cláusula PASSWORD es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

cláusula FILE STATUS

La cláusula FILE STATUS supervisa la ejecución de cada operación de entrada-salida para el archivo.

Cuando se especifica la cláusula FILE STATUS, el sistema mueve un valor al elemento de datos de clave de estado de archivo después de cada operación de entrada-salida que hace referencia explícita o implícitamente a este archivo. El valor indica el estado de ejecución de la sentencia. (Consulte la descripción de la clave de estado de archivo en “Recursos de proceso comunes” en la página 284.)

data-name-1
El elemento de datos de clave de estado de archivo se puede definir en WORKING-STORAGE, LOCAL-STORAGE o LINKAGE SECTION como uno de los elementos siguientes:

- Un elemento de datos de dos caracteres de categoría alfanumérica
- Un elemento de datos de dos caracteres de categoría nacional
- Un elemento de datos de dos dígitos de categoría numérica con uso DISPLAY o NATIONAL (un elemento de datos decimal externo)

data-name-1 no debe contener el símbolo PICTURE 'P'.

data-name-1 puede calificarse.

El elemento de datos de clave de estado de archivo no debe estar ubicado de forma variable; es decir, el elemento de datos no puede ir a continuación de un elemento de datos que contenga una cláusula OCCURS DEPENDING ON.

data-name-8
Representa la información devuelta por el sistema de archivos. Puesto que las definiciones son específicas de los sistemas de archivos y plataformas, es posible que las aplicaciones que dependen de los valores específicos de *data-name-8* no sean portables entre plataformas.

La forma en que defina *data-name-8* depende del sistema de archivos que esté utilizando.

Sistemas de archivos LSQ, MONGO, QSAM, RSD, SFS y STL

Debe definir *data-name-8* con los atributos PICTURE 9 (6) y USAGE DISPLAY. Sin embargo, puede definir un campo adicional con PICTURE X (*n*). El sistema de archivos define los valores de comentarios, que se convierten a la representación decimal externa de seis dígitos con ceros iniciales cuando el valor de comentarios del sistema de archivos es inferior a 100000. Si ha definido un campo adicional utilizando PICTURE X (*n*), entonces X (*n*) contiene información adicional que describe cualquier código

de comentarios distinto de cero. (Para la mayoría de programas, un valor *n* de 100 debería ser adecuado para mostrar el texto completo del mensaje.)

Sistema de archivosDb2

Debe definir *data-name-8* como un elemento de grupo como el ejemplo siguiente:

```
01 FileStatus2.
  02 FS2-SQLCODE COMP PICTURE S9(9).
  02 FS2-SQLSTATE PICTURE X(5).
```

Los valores de tiempo de ejecución en FS2-SQLCODE y FS2-SQLSTATE representan información de comentarios de SQL para la operación que se ha completado anteriormente.

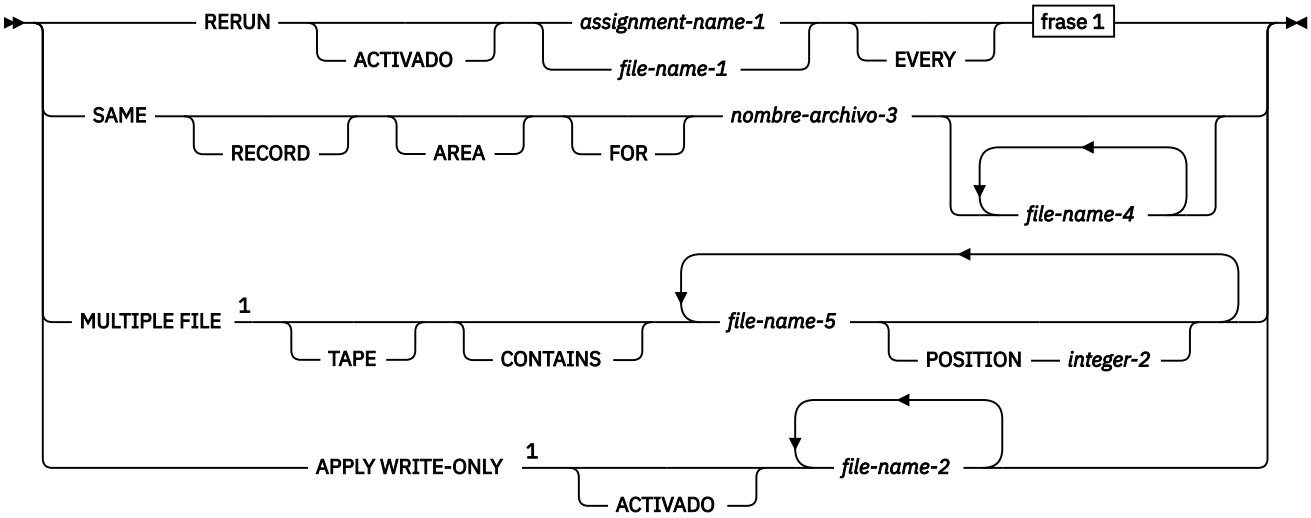
Párrafo I-O-CONTROL

El párrafo I-O-CONTROL de la sección de entrada-salida especifica cuándo deben tomarse los puntos de control y las áreas de almacenamiento que deben compartir los distintos archivos. Este párrafo es opcional en un programa COBOL.

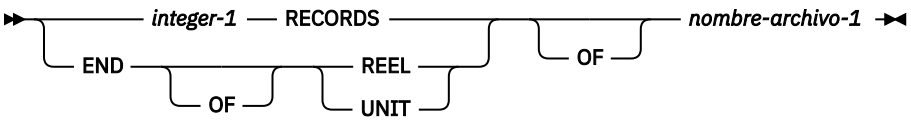
La palabra clave I-O-CONTROL sólo puede aparecer una vez, al principio del párrafo. La palabra I-O-CONTROL debe empezar en el Área A y debe ir seguida de un punto de separación.

El orden en que se escriben las cláusulas del párrafo I-O-CONTROL no es significativo. El párrafo I-O-CONTROL termina con un punto de separación.

Formato: secuencial-i-o-control-entry



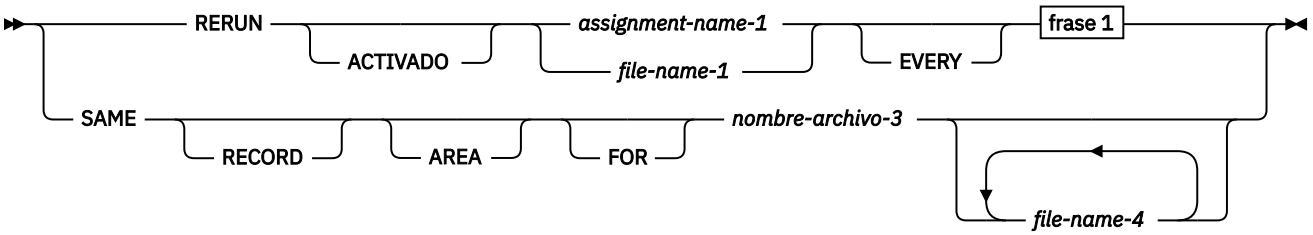
frase 1



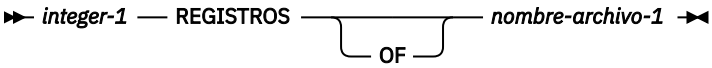
Notas:

[1] La cláusula MULTIPLE FILE y la cláusula APPLY WRITE-ONLY comprobación de sintaxis, pero no tienen ningún efecto en la ejecución del programa.

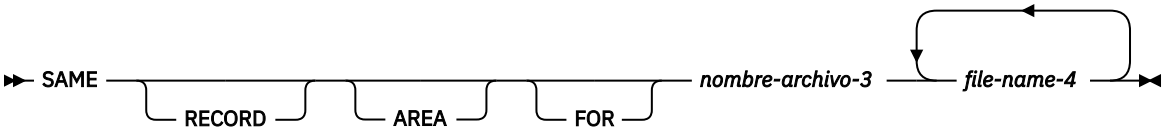
Formato: relative-and-indexed-i-o-control-entry



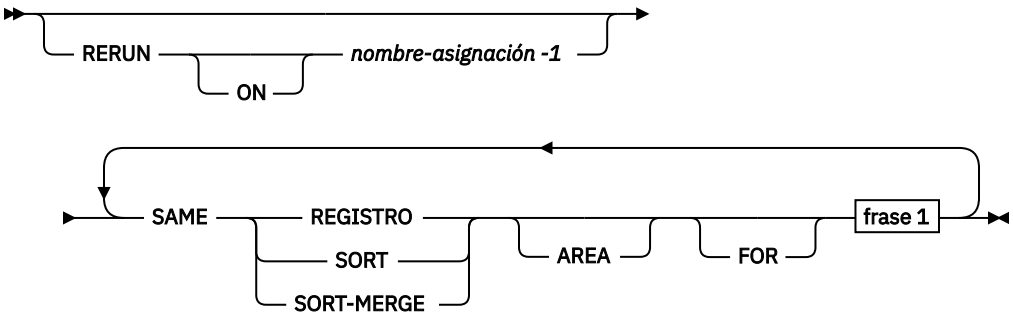
frase 1



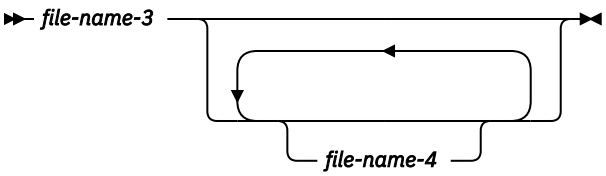
Formato: entrada-control-i-o-secuencial-línea



Formato: sort/merge-i-o-control-entry



frase 1



cláusula RERUN

La cláusula RERUN especifica que se deben tomar los registros de punto de comprobación. Sujeto a las restricciones dadas con cada frase, se puede especificar más de una cláusula RERUN.

La cláusula RERUN se comprueba con la sintaxis, pero no tiene ningún efecto en la ejecución de programas.

No utilice la cláusula RERUN:

- Para archivos descritos con la cláusula EXTERNAL
- En programas con la cláusula RECURSIVE especificada

file-name-1
Debe ser un archivo organizado secuencialmente.

assignment-name-1
El externo para el archivo de punto de comprobación. No debe ser el mismo nombre de asignación que el especificado en cualquier cláusula ASSIGN en todo el programa, incluidos los programas contenidos y que los contienen.

Consideraciones sobre SORT/MERGE:

Cuando se especifica la cláusula RERUN en el párrafo I-O-CONTROL, los registros de punto de comprobación se escriben a intervalos lógicos determinados por el programa de clasificación/fusión durante la ejecución de cada sentencia SORT o MERGE del programa. Cuando se omite la cláusula RERUN, los registros de punto de comprobación no se escriben.

Sólo puede haber un párrafo SORT/MERGE I-O-CONTROL en un programa, y no puede especificarse en programas contenidos. Tendrá un efecto global en todas las sentencias SORT y MERGE de la unidad de programa.

EVERY *integer-1* RECORDS
Se debe escribir un registro de punto de comprobación para cada *integer-1* registros en *file-name-1* que se procesen.

Cuando se especifican varias frases *integer-1* RECORDS, no hay dos de ellas que puedan especificar el mismo valor para *file-name-1*.

Si especifica la frase *integer-1* RECORDS, debe especificar *assignment-name-1*.

CADA FIN DE REEL/UNIDAD
Un registro de punto de comprobación debe grabarse siempre que se produzca el fin de volumen para *file-name-1* . Los términos REEL y UNIT son intercambiables.

Cuando se especifican varias frases END OF REEL/UNIT, no hay dos de ellas que puedan especificar el mismo valor para *file-name-1*.

La frase END OF REEL/UNIT sólo se puede especificar si *file-name-1* es un archivo organizado secuencialmente.

cláusula SAME AREA

La cláusula SAME AREA es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

cláusula SAME RECORD AREA

La cláusula SAME RECORD AREA especifica que dos o más archivos deben utilizar la misma área de almacenamiento principal para procesar el registro lógico actual.

Los archivos nombrados en una cláusula SAME RECORD AREA no necesitan tener la misma organización o acceso.

file-name-3 , file-name-4
Debe especificarse en el párrafo de control de archivos del mismo programa. *file-name-3* y *file-name-4* no deben hacer referencia a un archivo definido con la cláusula EXTERNAL.

Todos los archivos se pueden abrir al mismo tiempo. Un registro lógico en el área de almacenamiento compartido se considera que son los dos siguientes:

- Un registro lógico de cada archivo de salida abierto en la cláusula SAME RECORD AREA
- Un registro lógico del archivo de entrada leído más recientemente en la cláusula SAME RECORD AREA

Puede incluirse más de una cláusula SAME RECORD AREA en un programa. Sin embargo:

- Un nombre de archivo específico no debe aparecer en más de una cláusula SAME RECORD AREA.

- Si uno o más nombres de archivo de una cláusula SAME AREA aparecen en una cláusula SAME RECORD AREA, todos los nombres de archivo de esa cláusula SAME AREA deben aparecer en esa cláusula SAME RECORD AREA. Sin embargo, la cláusula SAME RECORD AREA puede contener nombres de archivo adicionales que no aparecen en la cláusula SAME AREA.
- La regla de que en la cláusula SAME AREA sólo un archivo puede estar abierto a la vez tiene prioridad sobre la regla SAME RECORD AREA de que todos los archivos pueden estar abiertos al mismo tiempo.
- Si se especifica la cláusula SAME RECORD AREA para varios archivos, las entradas de descripción de registro o las entradas de descripción de archivo para estos archivos no deben incluir la cláusula GLOBAL.
- La cláusula SAME RECORD AREA no debe especificarse cuando se especifica la cláusula RECORD CONTAINS 0 CHARACTERS.

No es necesario que los archivos especificados en la cláusula SAME RECORD AREA tengan la misma organización o acceso.

cláusula SAME SORT AREA

La cláusula SAME SORT AREA está comprobada en la sintaxis pero no tiene ningún efecto en la ejecución del programa.

file-name-3 , file-name-4
Debe especificarse en el párrafo de control de archivos del mismo programa. *file-name-3* y *file-name-4* no deben hacer referencia a un archivo definido con la cláusula EXTERNAL.

Cuando se especifica la cláusula SAME SORT AREA, al menos un nombre de archivo especificado debe nombrar un archivo de clasificación. También se pueden especificar los archivos que no son archivos de ordenación. Se aplican las reglas siguientes:

- Puede especificarse más de una cláusula SAME SORT AREA. Sin embargo, un archivo de clasificación determinado no debe nombrarse en más de una cláusula de este tipo.
- Si un archivo que no es un archivo de ordenación se nombra en una cláusula SAME AREA y en una o más cláusulas SAME SORT AREA, todos los archivos de la cláusula SAME AREA también deben aparecer en esa cláusula SAME SORT AREA.
- Los archivos nombrados en una cláusula SAME SORT AREA no necesitan tener la misma organización o acceso.
- Los archivos nombrados en una cláusula SAME SORT AREA que no son archivos de clasificación no comparten almacenamiento entre sí a menos que se denominen en una cláusula SAME AREA o SAME RECORD AREA.
- Durante la ejecución de una sentencia SORT o MERGE que hace referencia a un archivo de clasificación o fusión especificado en esta cláusula, los archivos no de clasificación o no fusión asociados con nombres de archivo especificados en esta cláusula no deben estar en la modalidad abierta.

Cláusula SAME SORT-MERGE AREA

La cláusula SAME SORT-MERGE AREA es equivalente a la cláusula SAME SORT AREA.

Para obtener más detalles, consulte “cláusula SAME SORT AREA” en la página 135.

cláusula MULTIPLE FILE TAPE

La cláusula MULTIPLE FILE TAPE (formato 1) especifica que dos o más archivos comparten el mismo carrete físico de cinta.

Esta cláusula es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Cláusula APPLY WRITE-ONLY

La cláusula APPLY WRITE-ONLY es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Parte 5. División de datos

Capítulo 15. Visión general de DATA DIVISION

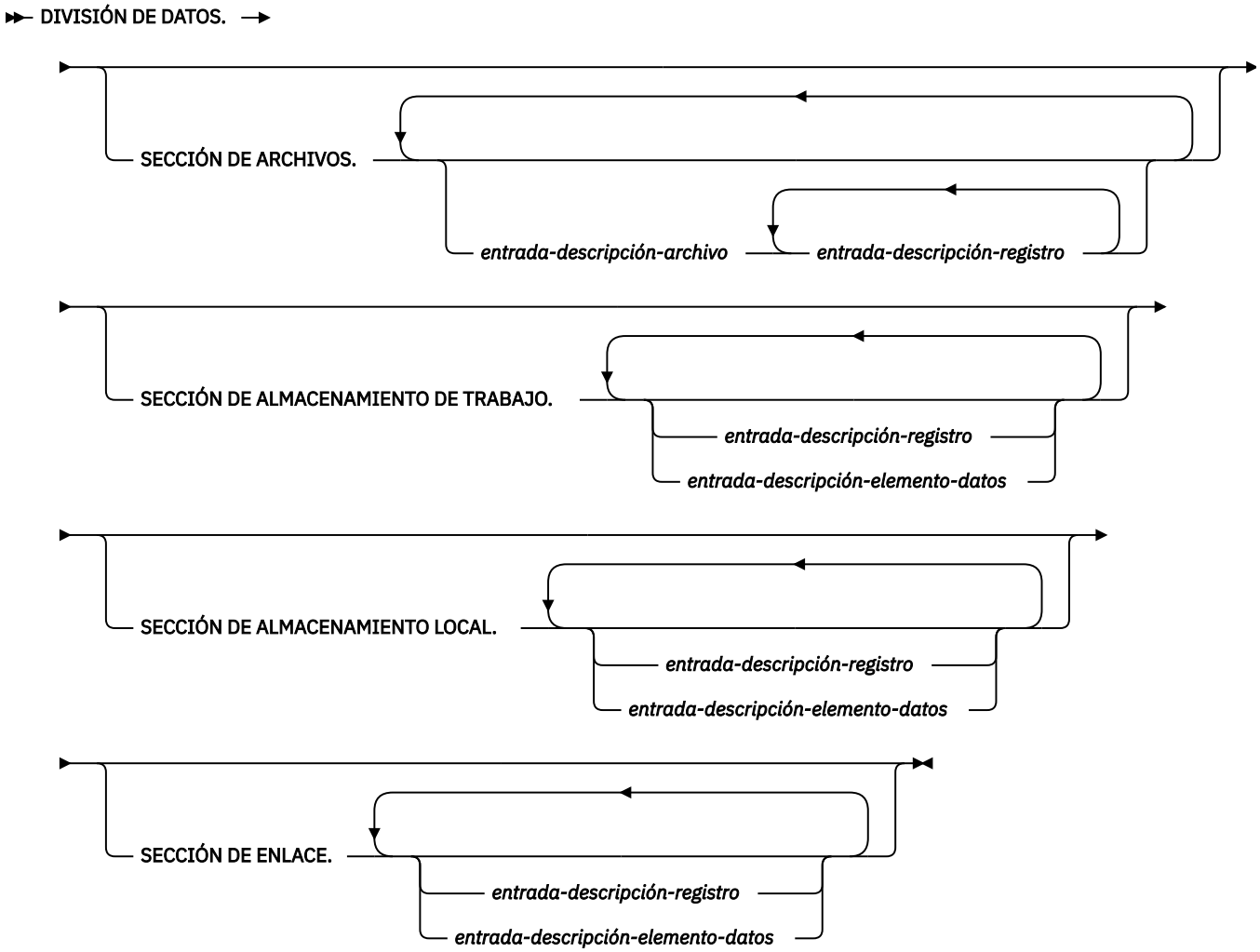
Esta visión general describe la estructura de DATA DIVISION para programas.

Cada sección de DATA DIVISION tiene una función lógica específica dentro de un programa COBOL y se puede omitir cuando esa función lógica no es necesaria. Si se incluyen, las secciones deben estar escritas en el orden mostrado. DATA DIVISION es opcional.

División de datos de programa

La división DATA DIVISION de un programa fuente COBOL describe, de forma estructurada, todos los datos que debe procesar el programa.

Formato: división de datos de programa



SECCIÓN DE ARCHIVO

FILE SECTION define la estructura de los archivos de datos. FILE SECTION debe empezar con la cabecera FILE SECTION, seguida de un punto separador.

entrada-descripción-archivo
Representa el nivel más alto de organización en FILE SECTION. Proporciona información sobre la estructura física y la identificación de un archivo, y proporciona los nombres de registro asociados con ese archivo. Para ver el formato y las cláusulas necesarias en una entrada de descripción de archivo, consulte Capítulo 16, “DATA DIVISION -- entradas de descripción de archivo”, en la página 153.

entrada-descripción-registro
Un conjunto de entradas de descripción de datos (descritas en Capítulo 17, “DATA DIVISION -- entrada de descripción de datos”, en la página 167) que describen los registros concretos contenidos dentro de un archivo determinado, o describen un nombre-tipo (utilizando la cláusula TYPEDEF).

Un registro en FILE SECTION debe describirse como un elemento de grupo alfanumérico, un elemento de grupo nacional o un elemento de datos elemental de clase alfabético, alfanumérico, DBCS, nacional o numérico.

Se puede especificar más de una entrada de descripción de registro; cada entrada que no describe un nombre de tipo es una descripción alternativa de la misma área de almacenamiento de registro.

Las áreas de datos descritas en FILE SECTION no están disponibles para el proceso a menos que el archivo que contiene el área de datos esté abierta. Los nombres de tipo definidos en FILE SECTION pueden utilizarse en WORKING-STORAGE, LOCAL-STORAGE o LINKAGE SECTION para definir otros elementos de datos.

SECCIÓN "WORKING-STORAGE"

WORKING-STORAGE SECTION describe los registros de datos que no forman parte de los archivos de datos pero que un programadesarrolla y procesa. La sección WORKING-STORAGE SECTION también describe los elementos de datos cuyos valores se asignan en el programa fuente y no cambian durante la ejecución del programa objeto.

La WORKING-STORAGE SECTION debe empezar por la cabecera de sección WORKING-STORAGE SECTION, seguida de un punto de separación. Los nombres de tipo se pueden definir en la SECCIÓN WORKING-STORAGE.

ALMACENAMIENTO DE TRABAJO DE PROGRAMA
La WORKING-STORAGE SECTION para programas también puede describir registros de datos externos, que son compartidos por programas en toda la unidad de ejecución. Todas las cláusulas que se utilizan en las descripciones de registro en FILE SECTION y también las cláusulas VALUE y EXTERNAL (que podrían no especificarse en las entradas de descripción de registro en FILE SECTION) pueden utilizarse en las descripciones de registro en WORKING-STORAGE SECTION.

La sección WORKING-STORAGE SECTION contiene entradas de descripción de registro y entradas de descripción de datos para elementos de datos independientes, denominadas *entradas de descripción de elemento de datos*.

entrada-descripción-registro
Las entradas de datos en la WORKING-STORAGE SECTION que tienen una relación jerárquica definida entre sí deben agruparse en registros estructurados por número de nivel. Consulte Capítulo 17, “DATA DIVISION -- entrada de descripción de datos”, en la página 167 para obtener más información.

entrada-descripción-elemento-datos
Los elementos independientes de la SECCIÓN WORKING-STORAGE que no tienen ninguna relación jerárquica entre sí no es necesario agruparlos en registros siempre que no sea necesario subdividirlos más. En cambio, se clasifican y definen como ítems elementales independientes. Cada uno se define

en una entrada de descripción de elemento de datos independiente que empieza por el número de nivel 77 o 01. Consulte Capítulo 17, “DATA DIVISION -- entrada de descripción de datos”, en la página 167 para obtener más información.

SECCIÓN DE ALMACENAMIENTO LOCAL

La sección LOCAL-STORAGE SECTION define el almacenamiento que se asigna y libera por invocación. LOCAL-STORAGE SECTION utiliza memoria de pila.

En cada invocación, los elementos de datos definidos en LOCAL-STORAGE SECTION se reasignan. Cada elemento de datos que tiene una cláusula VALUE se inicializa con el valor especificado en dicha cláusula.

Para programas anidados, los elementos de datos definidos en LOCAL-STORAGE SECTION se asignan en cada invocación del programa externo que los contiene. Sin embargo, cada elemento de datos se reinicializa al valor especificado en su cláusula VALUE cada vez que se invoca el programa anidado.

Los elementos de datos definidos en LOCAL-STORAGE SECTION no pueden especificar la cláusula EXTERNAL.

La sección LOCAL-STORAGE SECTION debe empezar con la cabecera LOCAL-STORAGE SECTION, seguida de un punto de separación.

Puede especificar la sección LOCAL-STORAGE SECTION en programas recursivos y no recursivos.

SECCIÓN DE ENLACE

La sección LINKAGE SECTION describe los datos disponibles de otro programa.

entrada-descripción-registro
Consulte “SECCIÓN "WORKING-STORAGE"” en la página 140 para obtener una descripción.

entrada-descripción-elemento-datos
Consulte “SECCIÓN "WORKING-STORAGE"” en la página 140 para obtener una descripción.

Las entradas de descripción de registro y las entradas de descripción de elemento de datos de LINKAGE SECTION proporcionan nombres y descripciones, pero el almacenamiento dentro del programa no está reservado porque el área de datos existe en otro lugar. *Los nombres de tipo* se pueden definir en LINKAGE SECTION.

Se puede utilizar cualquier cláusula de descripción de datos para describir elementos en LINKAGE SECTION con las excepciones siguientes:

- No puede especificar la cláusula VALUE para elementos que no sean elementos de level-88 .
- No puede especificar la cláusula EXTERNAL.

Puede especificar la cláusula GLOBAL en LINKAGE SECTION.

Unidades de datos

Los datos se agrupan en las unidades conceptuales tal como se listan en el tema.

- Datos de archivo
- Datos de programa

Datos de archivo

Los datos de archivo están contenidos en archivos. Un *archivo* es una colección de registros de datos que existen en algún dispositivo de entrada-salida. Un archivo se puede considerar como un grupo de registros físicos; también se puede considerar como un grupo de registros lógicos. DATA DIVISION describe la relación entre registros físicos y lógicos.

Para obtener más información, consulte “SECCIÓN DE ARCHIVO” en la página 158.

Un *registro físico* es una unidad de datos que se trata como una entidad cuando se mueve dentro o fuera del almacenamiento. El tamaño de un registro físico lo determina el dispositivo de entrada-salida concreto en el que se almacena. El tamaño no tiene necesariamente una relación directa con el tamaño o el contenido de la información lógica contenida en el archivo.

Un *registro lógico* es una unidad de datos cuyas subdivisiones tienen una relación lógica. Un registro lógico puede ser por sí mismo un registro físico (es decir, estar completamente contenido en una unidad física de datos); varios registros lógicos pueden estar contenidos en un registro físico, o un registro lógico puede extenderse a través de varios registros físicos.

Las *entradas de descripción de archivo* especifican los aspectos físicos de los datos (como la relación de tamaño entre registros físicos y lógicos, el tamaño y los nombres de los registros lógicos, la información de etiquetado, etc.).

Las *entradas de descripción de registro* describen los registros lógicos del archivo (incluida la categoría y el formato de los datos dentro de cada campo del registro lógico), los valores diferentes que se pueden asignar a los datos, etc.

Después de que se haya establecido la relación entre los registros físicos y lógicos, sólo se ponen a su disposición los registros lógicos. Por esta razón, una referencia en esta información a "registros" significa registros lógicos, a menos que se utilice el término "registros físicos".

Datos de programa

Un programa crea datos de programa en lugar de leerlos de un archivo.

El concepto de registros lógicos se aplica a los datos de programa, así como a los datos de archivo. Por lo tanto, los datos de programa se pueden agrupar en registros lógicos y se pueden definir mediante una serie de entradas de descripción de registro. Los elementos que no necesitan estar agrupados se pueden definir en entradas de descripción de datos independientes (denominadas *entradas de descripción de elemento de datos*).

Relaciones de datos

Las relaciones entre todos los datos que se van a utilizar en un programa se definen en DATA DIVISION a través de un sistema de indicadores de nivel y números de nivel.

Un *indicador de nivel*, con su entrada descriptiva, identifica cada archivo de un programa. Los indicadores de nivel representan el nivel más alto de cualquier jerarquía de datos con la que estén asociados. FD es el indicador de nivel de descripción de archivo y SD es el indicador de nivel de descripción de archivo de fusión de clasificación.

Un *número de nivel*, con su entrada descriptiva, indica las propiedades de datos específicos. Los números de nivel se pueden utilizar para describir una jerarquía de datos; pueden indicar que estos datos tienen un propósito especial. Aunque se pueden asociar con (y subordinar a) indicadores de nivel, también se pueden utilizar de forma independiente para describir datos internos o datos comunes a dos o más programas. (Consulte “Nivel-números” en la página 170 para ver las reglas de número de nivel.)

Niveles de datos

Una vez que se ha definido un registro, se puede subdividir para proporcionar referencias de datos más detalladas.

Por ejemplo, en un archivo de cliente para un almacén por departamento, un registro completo podría contener todos los datos que pertenecen a un cliente. Las subdivisiones dentro de ese registro podrían ser, por ejemplo, nombre de cliente, dirección de cliente, número de cuenta, número de departamento de venta, cantidad unitaria de venta, cantidad en dólares de venta, saldo anterior y otra información pertinente.

Las subdivisiones básicas de un registro (es decir, los campos que no se subdividen más) se denominan *elementos elementales*. Por lo tanto, un registro puede estar formado por una serie de elementos elementales o puede ser por sí mismo un elemento elemental.

Es posible que sea necesario hacer referencia a un conjunto de elementos elementales; por lo tanto, los elementos elementales se pueden combinar en *elementos de grupo*. Los grupos también se pueden combinar en un grupo más inclusivo que contenga uno o más subgrupos. Por lo tanto, dentro de una jerarquía de elementos de datos, un elemento elemental puede pertenecer a más de un elemento de grupo.

Un sistema de números de nivel especifica la organización de elementos elementales y de grupo en registros. Los números de nivel especial también se utilizan para identificar los elementos de datos utilizados para fines especiales.

Niveles de datos en una entrada de descripción de registro

Cada grupo y elemento elemental de un registro requiere una entrada separada, y a cada uno se le debe asignar un número de nivel.

Un número de nivel es un entero de un dígito o dos dígitos entre 01 y 49, o uno de los tres números de nivel especiales: 66, 77 u 88. Los siguientes números de nivel se utilizan para estructurar registros:

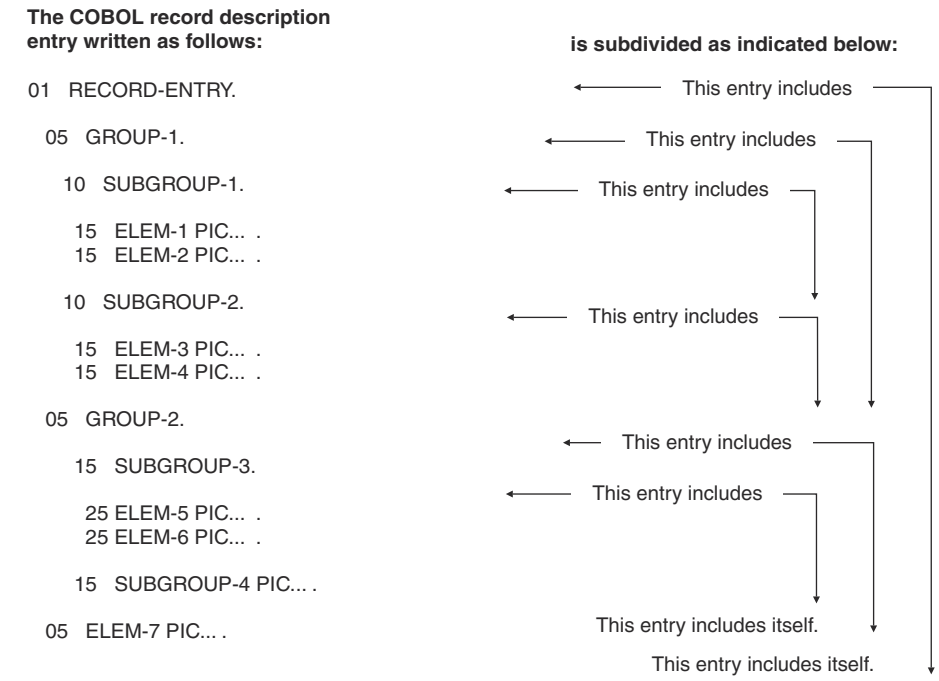
01
Este número de nivel especifica el propio registro y es el número de nivel más inclusivo posible. Una entrada de level-01 puede ser un elemento de grupo alfanumérico, un elemento de grupo nacional o un elemento elemental. El número de nivel debe empezar en el Área A. Los nombres de tipo (definidos utilizando la cláusula TYPEDEF) deben ser elementos level-01 .

02 a 49
Estos números de nivel especifican elementos de grupo y elementales dentro de un registro. Pueden comenzar en el Área A o en el Área B. A los elementos de datos menos inclusivos se les asignan números de nivel más altos (no necesariamente consecutivos) en esta serie.

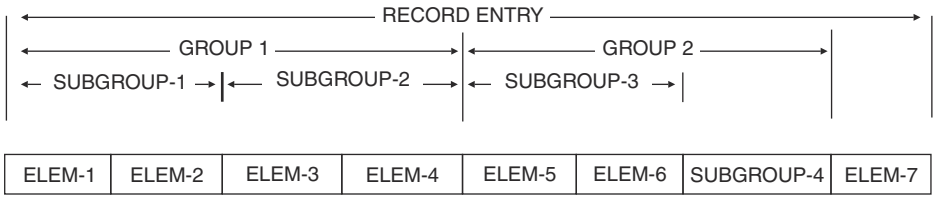
La relación entre los números de nivel dentro de un elemento de grupo define la jerarquía de datos dentro de ese grupo.

Un elemento de grupo incluye todos los elementos de grupo y elementales que le siguen hasta que se encuentra un número de nivel menor o igual que el número de nivel de ese grupo.

La figura siguiente ilustra un grupo en el que todos los grupos subordinados inmediatamente a la entrada level-01 tienen el mismo número de nivel.



The storage arrangement of the record description entry is illustrated below:



También puede definir grupos con elementos subordinados que tengan distintos números de nivel para el mismo nivel en la jerarquía. Por ejemplo, `05 EMPLOYEE-NAME` y `04 EMPLOYEE-ADDRESS` en `EMPLOYEE-RECORD` a continuación definen el mismo nivel en la jerarquía. El compilador vuelve a numerar los niveles de forma relativa, tal como se muestra en la salida de MAP.

```
01   EMPLOYEE-RECORD.
     05  EMPLOYEE-NAME.
         10  FIRST-NAME PICTURE  X(10).
         10  LAST-NAME  PICTURE  X(10).
     04  EMPLOYEE-ADDRESS.
         08  STREET     PICTURE  X(10).
         08  CITY       PICTURE  X(10).
```

La siguiente entrada de descripción de registro define la misma jerarquía de datos que la entrada de descripción de registro anterior:

```
01   EMPLOYEE-RECORD.
     02  EMPLOYEE-NAME.
         03  FIRST-NAME PICTURE  X(10).
         03  LAST-NAME  PICTURE  X(10).
     02  EMPLOYEE-ADDRESS.
         03  STREET     PICTURE  X(10).
         03  CITY       PICTURE  X(10).
```

Los elementos elementales se pueden especificar en cualquier nivel dentro de la jerarquía.

Número de nivel especial

Los números de nivel especial identifican elementos que no estructuran un registro.

Los números de nivel especiales son:

66
Identifica elementos que deben contener una cláusula RENAMES; dichos elementos reagrupan elementos de datos definidos anteriormente. (Para obtener detalles, consulte “cláusula RENAMES” en la página 215.)

77
Identifica entradas de descripción de elemento de datos que son elementos WORKING-STORAGE, LOCAL-STORAGE o LINKAGE SECTION independientes; no son subdivisiones de otros elementos y no se subdividen ellos mismos. Los elementos de Level-77 deben empezar en el Área A.

88
Identifica cualquier entrada de nombre de condición que esté asociada con un valor determinado de una variable condicional. (Para obtener detalles, consulte “cláusula VALUE” en la página 232.)

Level-77 y level-01 en las entradas WORKING-STORAGE, LOCAL-STORAGE, o LINKAGE SECTION a los que se hace referencia en un programa deben proporcionarse nombres de datos exclusivos porque las entradas level-77 y level-01 no se pueden calificar. Los nombres de datos subordinados a los que se hace referencia en el programa deben estar definidos de forma exclusiva o ser exclusivos a través de la calificación. No es necesario que los nombres de datos no referenciados estén definidos de forma exclusiva.

Sangría

Las entradas de descripción de datos sucesivas pueden empezar en la misma columna que las entradas anteriores, o pueden sangrarse.

La sangría es útil para la documentación, pero no afecta a la acción del compilador.

Clases y categorías de elementos de grupo

COBOL para Linux tiene dos tipos de grupos: grupos alfanuméricos y grupos nacionales.

Los grupos que no especifican una cláusula GROUP-USAGE son grupos alfanuméricos. Un grupo alfanumérico tiene una clase y una categoría alfanuméricas y se trata como si su uso fuera DISPLAY, independientemente de la representación de los elementos de datos elementales contenidos en el grupo. En muchas operaciones, como movimientos y comparaciones, los grupos alfanuméricos se tratan como si fueran elementos elementales de la categoría alfanumérica, excepto que no se realiza ninguna edición o conversión de la representación de datos. En otras operaciones, como MOVE CORRESPONDIENTE y ADD CORRESPONDIENTE, los elementos de datos subordinados se procesan como elementos elementales separados.

El contenido de un grupo alfanumérico se trata como si estuviera representado en caracteres nativos de un solo byte cuando se utiliza la opción de compilador CHAR (NATIVE) y como caracteres EBCDIC de un solo byte cuando se utiliza la opción de compilador CHAR (EBCDIC).

Los grupos nacionales se definen mediante una cláusula GROUP-USAGE con la frase NATIONAL a nivel de grupo. Todos los elementos de datos subordinados deben describirse explícita o implícitamente con el uso NATIONAL, y los grupos subordinados deben describirse explícita o implícitamente con GROUP-USAGE NATIONAL.

A menos que se indique lo contrario, un elemento de grupo nacional se procesa exactamente como si se tratara de un elemento de datos elementales de uso nacional, de clase y de categoría nacional, descrito con PICTURE N (m), donde m es la longitud del grupo en posiciones de carácter nacional. Puesto que los grupos nacionales sólo contienen caracteres nacionales, los datos se convierten según sea necesario para movimientos y comparaciones. El compilador garantiza el truncamiento y el relleno adecuados. En otras operaciones, como MOVE CORRESPONDIENTE y ADD CORRESPONDIENTE, los elementos de datos subordinados se procesan como elementos elementales separados. Consulte “cláusula GROUP-USAGE” en la página 179 para obtener más detalles.

La tabla siguiente resume las clases y categorías de elementos de grupo.

Tabla 11. ***Clases y categorías de elementos de grupo***				
Descripción de grupo	**Clase de grupo**	**Categoría del grupo**	**USO de elementos elementales dentro de un grupo**	**USO de un grupo**
Sin una cláusula GROUP-USAGE	Alfanumérico	Alfanumérico (aunque los elementos elementales del grupo pueden tener cualquier categoría)	any	Tratado como DISPLAY cuando el uso es relevante
Con cláusula GROUP-USAGE explícita o implícita	Nacional	Nacional	NACIONAL	NACIONAL

Clases y categorías de datos

La mayoría de los datos y todos los literales utilizados en un programa COBOL se dividen en clases y categorías. Las clases de datos son agrupaciones de categorías de datos. Las categorías de datos se determinan mediante los atributos de las entradas de descripción de datos o las definiciones de función.

Para obtener más información sobre las categorías de datos, consulte "Descripciones de categorías" en la página 148.

Los siguientes elementos de datos elementales no tienen una clase ni una categoría:

- Elementos de datos de índice
- Elementos descritos con USAGE POINTER, USAGE FUNCTION-POINTER, o USAGE PROCEDURE-POINTER

Todos los demás tipos de elementos de datos elementales tienen una clase y una categoría tal como se muestra en la Tabla 12 en la página 146.

Una función hace referencia a un elemento de datos elemental y pertenece a la clase de datos y categoría asociadas con el tipo de la función, tal como se muestra en la Tabla 13 en la página 147.

Los literales tienen una clase y una categoría tal como se muestra en la Tabla 14 en la página 148.
Las constantes figurativas (excepto NULL) tienen una clase y categoría que depende del literal o valor representado por la constante figurativa en el contexto de su uso. Para obtener detalles, consulte "Constantes figurativas" en la página 15.

Todos los elementos de grupo tienen una clase y una categoría, incluso si los elementos elementales subordinados pertenecen a otra clase y categoría. Para la clasificación de elementos de grupo, consulte "Clases y categorías de elementos de grupo" en la página 145.

Tabla 12. ***Clase, categoría y uso de elementos de datos elementales***		
Clase de elementos de datos elementales[2]	**Categoría**	**Uso**
Alfabético	Alfabético	VISUALIZAR
Alfanumérico	Alfanumérico	VISUALIZAR
	Alfanumérico-editado	VISUALIZAR
	Numérico-editado	VISUALIZAR

Tabla 12. ***Clase, categoría y uso de elementos de datos elementales*** *(continuación)*		
Clase de elementos de datos elementales[2]	**Categoría**	**Uso**
Booleano[1]	Booleano[1]	VISUALIZAR

Tabla 12. ***Clase, categoría y uso de elementos de datos elementales***		
Fecha y hora[1]	Fecha[1] Hora[1]	VISUALIZAR DECIMAL EMPAQUETADO
	Indicación de fecha y hora[1]	VISUALIZAR
DBCS[1]	DBCS[1]	DISPLAY-1
Nacional[1]	Nacional[1]	NACIONAL
	Nacional-editado[1]	NACIONAL
	Numérico-editado[1]	NACIONAL
Numérico	Numérico	DISPLAY (tipo decimal con zona)
		NATIONAL (tipo decimal nacional)
		PACKED-DECIMAL (tipo decimal interno)
		COMP-3 (tipo decimal interno)
		BINARIO
		COMP
		COMP-4
		COMP-5
	Coma flotante interna[1]	COMP-1
		COMP-2
	Coma flotante externa[1]	VISUALIZAR
		NACIONAL

1. Extensión de IBM
2. La clase de elementos de grupo es alfanumérica para todas las categorías.

Tabla 13. ***Clases y categorías de funciones***	
Tipo de función	**Clase y categoría**
Alfanumérico	Alfanumérico
Nacional	Nacional
Entero	Numérico
Numérico	Numérico

Tabla 14. ***Clases y categorías de literales***	
Literal	**Clase y categoría**
Alfanumérico (incluidos los formatos hexadecimales)	Alfanumérico
DBCS	DBCS
Nacional (incluidos los formatos hexadecimales)	Nacional
Numérico (punto fijo y coma flotante)	Numérico

Descripciones de categorías

La categoría de un elemento de datos se establece mediante los atributos de su entrada de descripción de datos (como su serie de caracteres PICTURE o cláusula USAGE) o mediante su definición de función.

El significado de cada categoría se indica a continuación.

Alfabético

Un elemento de datos se describe como alfabético de categoría mediante su serie de caracteres PICTURE. Para obtener detalles de serie de caracteres PICTURE, consulte "Elementos alfabéticos" en la página 199.

Un elemento de datos de categoría alfabética se conoce como elemento de datos alfabéticos.

Alfanumérico

Cada uno de los siguientes elementos de datos es de categoría alfanumérica:

- Elemento de datos elemental descrito como alfanumérico por su serie de caracteres PICTURE. Para detalles de serie de caracteres PICTURE, consulte "Elementos alfanuméricos" en la página 200.
- Un elemento de grupo alfanumérico.
- Una función alfanumérica.
- Los siguientes registros especiales:
  - ELEMENTO DE DEPURACIÓN
  - MAYÚS-OUT
  - MAYÚS-IN
  - CONTROL DE SORT
  - ORDENAR-MENSAJE
  - WHEN-COMPILADO
  - XML-EVENT
  - XML-TEXTO

Alfanumérico-editado

Un elemento de datos se describe como de categoría alfanumérica-editado por su serie de caracteres PICTURE. Para obtener detalles de serie de caracteres PICTURE, consulte "Elementos alfanuméricos editados" en la página 200.

Un elemento de datos de categoría alfanumérico editado se conoce como elemento de datos alfanumérico editado.

Booleano

Los datos booleanos son una extensión de IBM que proporciona un medio para modificar y pasar los valores de los indicadores asociados con los formatos de pantalla de visualización y los archivos de impresora descritos externamente. Un valor booleano de 0 es el estado *off* del indicador y un valor booleano de 1 es el estado *on* del indicador.

Un literal booleanocontiene un único 0 o 1, está entre comillas y va inmediatamente precedido de una B identificativa. Un literal booleano se define como B "0" o B "1".

Un carácter booleano ocupa un byte.

Cuando la constante figurativa ZERO está asociada con un elemento de datos booleano o un literal booleano, representa el literal booleano B "0".

La palabra reservada ALL es válida con un literal booleano.

Para detalles de serie de caracteres PICTURE, consulte "Elementos booleanos" en la página 199.

Un elemento de datos de categoría booleana se conoce como elemento de datos booleano.

Fecha, hora, indicación de fecha y hora

Un elemento de datos se describe como categoría fecha, hora o indicación de fecha y hora mediante su cláusula FORMAT. Para obtener detalles sobre la cláusula FORMAT, consulte "cláusula FORMAT" en la página 177.

DBCS

Un elemento de datos se describe como categoría DBCS mediante su serie de caracteres PICTURE y la opción de compilador NSYMBOL (DBCS) o mediante una cláusula USAGE DISPLAY-1 explícita. Para obtener detalles de serie de caracteres PICTURE, consulte "Elementos DBCS" en la página 201.

Un elemento de datos de categoría DBCS se conoce como elemento de datos DBCS.

Coma flotante externa

Un elemento de datos se describe como de coma flotante externo de categoría mediante su serie de caracteres PICTURE. Para detalles de serie de caracteres PICTURE, consulte "Elementos de coma flotante externos" en la página 201. Un elemento de datos de coma flotante externo puede describirse con USAGE DISPLAY o USAGE NATIONAL.

Cuando el uso es DISPLAY, se hace referencia al elemento como un elemento de datos de coma flotante de visualización.

Cuando el uso es NACIONAL, el elemento se conoce como elemento de datos de coma flotante nacional.

Un elemento de datos de coma flotante externo es de clase numérica y, a menos que se excluya específicamente, se incluye en una referencia a un elemento de datos numérico.

Coma flotante interna

Un elemento de datos se describe como de coma flotante interno de categoría mediante una cláusula USAGE con la frase COMP-1 o COMP-2 .

Un elemento de datos de punto flotante interno de categoría se conoce como elemento de datos de punto flotante interno. Un elemento de datos de coma flotante interno es de clase numérica y, a menos que se excluya específicamente, se incluye en una referencia a un elemento de datos numérico.

Nacional

Cada uno de los siguientes elementos de datos de es de la categoría nacional :

- Elemento de datos que se describe como categoría nacional mediante su serie de caracteres PICTURE y la opción de compilador NSYMBOL (NATIONAL) o mediante una cláusula USAGE NATIONAL explícita. Para detalles de serie de caracteres PICTURE, consulte “Partidas nacionales” en la página 202.
- Elemento de grupo descrito explícita o implícitamente con una cláusula GROUP-USAGE NATIONAL.
- Una función nacional.
- El registro especial XML-NTEXT.

Nacional-editado

Un elemento de datos se describe como de categoría nacional editado por su serie de caracteres PICTURE. Para obtener detalles de serie de caracteres PICTURE, consulte “Elementos editados a nivel nacional” en la página 203.

Un elemento de datos de categoría editada a nivel nacional se denomina elemento de datos editado a nivel nacional.

Numérico

Cada uno de los siguientes elementos de datos es de categoría numérica:

- Elemento de datos elemental descrito como numérico por su serie de caracteres PICTURE y no descrito con una cláusula BLANK WHEN ZERO. Para detalles de serie de caracteres PICTURE, consulte “Elementos numéricos” en la página 204.
- Un elemento de datos elemental descrito con uno de los usos siguientes:
  - BINARY, COMPUTATIONAL, COMPUTATIONAL-4, COMPUTATIONAL-5, COMP, COMP-4o COMP-5
  - PACKED-DECIMAL, COMPUTATIONAL-3o COMP-3
- Un registro especial de tipo numérico:
  - LONGITUD DE
  - LINAGE-COUNTER
  - CÓDIGO-RETORNO
  - TAMAÑO-NÚCLEO-CLASIFICACIÓN
  - SORT-FILE-SIZE
  - TAMAÑO-MODALIDAD-CLASIFICACIÓN
  - ORDENAR-RETORNO
  - RECUENTO
  - CÓDIGO XML
- Una función numérica.
- Una función de entero.

Un elemento de datos de categoría numérica se conoce como un elemento de datos numérico.

Numérico-editado

Cada uno de los siguientes elementos de datos es de categoría numérica-editada:

- Elemento de datos descrito como numérico editado por su serie de caracteres PICTURE. Para detalles de serie de caracteres PICTURE, consulte “Elementos editados numérica-editados” en la página 205.
- Elemento de datos descrito como numérico por su serie de caracteres PICTURE y descrito con una cláusula BLANK WHEN ZERO.

Reglas de alineación

Las reglas de alineación estándar para colocar datos en un artículo elemental dependen de la categoría de un artículo receptor.

Un elemento de recepción es un elemento al que se mueven los datos. Para obtener más detalles sobre un artículo de recepción, consulte "Movimientos elementales" en la página 349.

Numérico
Para los elementos de recepción numéricos, se aplican las reglas siguientes:

1. Los datos se alinean en la coma decimal asumida y, si es necesario, se truncan o se rellenan con ceros. (Un *punto decimal asumido* es uno que tiene un significado lógico pero que no existe como un carácter real en los datos.)
2. Si no se especifica explícitamente una coma decimal asumida, el elemento receptor se trata como si se especificara una coma decimal asumida inmediatamente a la derecha del campo. A continuación, los datos se tratan de acuerdo con la regla anterior.

Numérico editado
Los datos se alinean en la coma decimal, y (si es necesario) se truncan o se rellenan con ceros en cualquier extremo, excepto cuando la edición provoca la sustitución de los ceros iniciales.

coma flotante interna
Se asume una coma decimal inmediatamente a la izquierda del campo. A continuación, los datos se alinean en la posición de dígito más a la izquierda que sigue a la coma decimal, con el exponente ajustado en consecuencia.

coma flotante externa
Los datos se alinean en la posición de dígito más a la izquierda; el exponente se ajusta en consecuencia.

Alfanumérico, alfanumérico-editado, alfabético, DBCS
Para estos artículos de recepción, se aplican las reglas siguientes:

1. Los datos se alinean en la posición más a la izquierda del carácter, y (si es necesario) se truncan o rellenan con espacios a la derecha.
2. Si se especifica la cláusula JUSTIFIC para este elemento receptor, la regla anterior se modifica tal como se describe en "cláusula JUSTIFIC" en la página 181.

Nacional, edición nacional
Para estos artículos de recepción, se aplican las reglas siguientes:

1. Los datos se alinean en la posición de carácter situada más a la izquierda y (si es necesario) se truncan o se rellenan con espacios Unicode predeterminados (NX'2000') a la derecha. El truncamiento se produce en el límite de una posición de carácter nacional.
2. Si se especifica la cláusula JUSTIFIC para este elemento receptor, la regla anterior se modifica tal como se describe en "cláusula JUSTIFIC" en la página 181.

Fecha, hora, indicación de fecha y hora
Para estos artículos de recepción, se aplican las reglas siguientes:

1. Paraelementos de fecha y hora de clase con un USAGE de DISPLAY, los datos se alinean en la posición más a la izquierda del carácter y (si es necesario) se rellenan con espacios a la derecha.
2. Para los elementos de fecha y hora de clase con un USAGE de PACKED-DECIMAL, los datos se alinean en la posición más a la derecha del dígito y (si es necesario) se rellenan con ceros a la izquierda.

Serie de caracteres y tamaño de elemento

Para los elementos descritos con una cláusula PICTURE, el tamaño de un elemento elemental se expresa en el código fuente mediante el número de posiciones de caracteres descritas en la serie de caracteres PICTURE y una cláusula SIGN (si procede). Sin embargo, el tamaño de almacenamiento viene

determinado por el número real de bytes que ocupa el elemento tal como lo determina la combinación de su serie de caracteres PICTURE, la cláusula SIGN IS SEPARATE (si se especifica) y la cláusula USAGE.

Para los elementos descritos con USAGE DISPLAY (categorías alfabéticas, alfanuméricas, editadas alfanuméricas, editadas numéricas, numéricas y de coma flotante externa), se reserva 1 byte de almacenamiento para cada posición de carácter descrita por la serie de caracteres PICTURE del elemento y la cláusula SIGN IS SEPARATE (si procede).

Para los elementos descritos con USAGE DISPLAY-1 (categoría DBCS), se reservan 2 bytes de almacenamiento para cada posición de carácter descrita por la serie de caracteres PICTURE del elemento.

Para los elementos descritos con USAGE NATIONAL (categorías nacionales, editadas a nivel nacional, numéricas, numéricas y de coma flotante externa), se reservan 2 bytes de almacenamiento para cada posición de carácter descrita por la serie de caracteres PICTURE del elemento y la cláusula SIGN IS SEPARATE (si se especifica).

Para elementos de coma flotante internos, el tamaño del elemento en almacenamiento viene determinado por su cláusula USAGE. USAGE COMPUTATIONAL-1 reserva 4 bytes de almacenamiento para el elemento; USAGE COMPUTATIONAL-2 reserva 8 bytes de almacenamiento.

Normalmente, cuando un elemento aritmético se mueve de un campo más largo a uno más corto, el compilador trunca los datos al número de dígitos representados en la serie de caracteres PICTURE del elemento más corto truncando los dígitos iniciales. Por ejemplo, si un campo de envío con PICTURE S99999 que contiene el valor +12345 se mueve a un campo de recepción BINARY con PICTURE S99, los datos se truncan a +45. Para obtener información adicional, consulte “cláusula USAGE” en la página 225.

La opción de compilador TRUNC puede afectar al valor de un elemento numérico binario. Para obtener información sobre TRUNC, consulte *TRUNC* en la publicación *COBOL for Linux en x86 Guía de programación*.

Datos firmados

Hay dos categorías de signos algebraicos utilizados en COBOL: signos operativos y signos de edición.

Signos operativos

Los signos operativos se asocian con elementos numéricos con signo e indican sus propiedades algebraicas.

La representación interna de un signo algebraico depende de la cláusula USAGE del elemento, de su cláusula SIGN (si está presente) y del entorno operativo. (Para obtener más detalles sobre la representación interna, consulte *Ejemplos: datos numéricos y representación interna* en *COBOL for Linux en x86 Guía de programación*.)

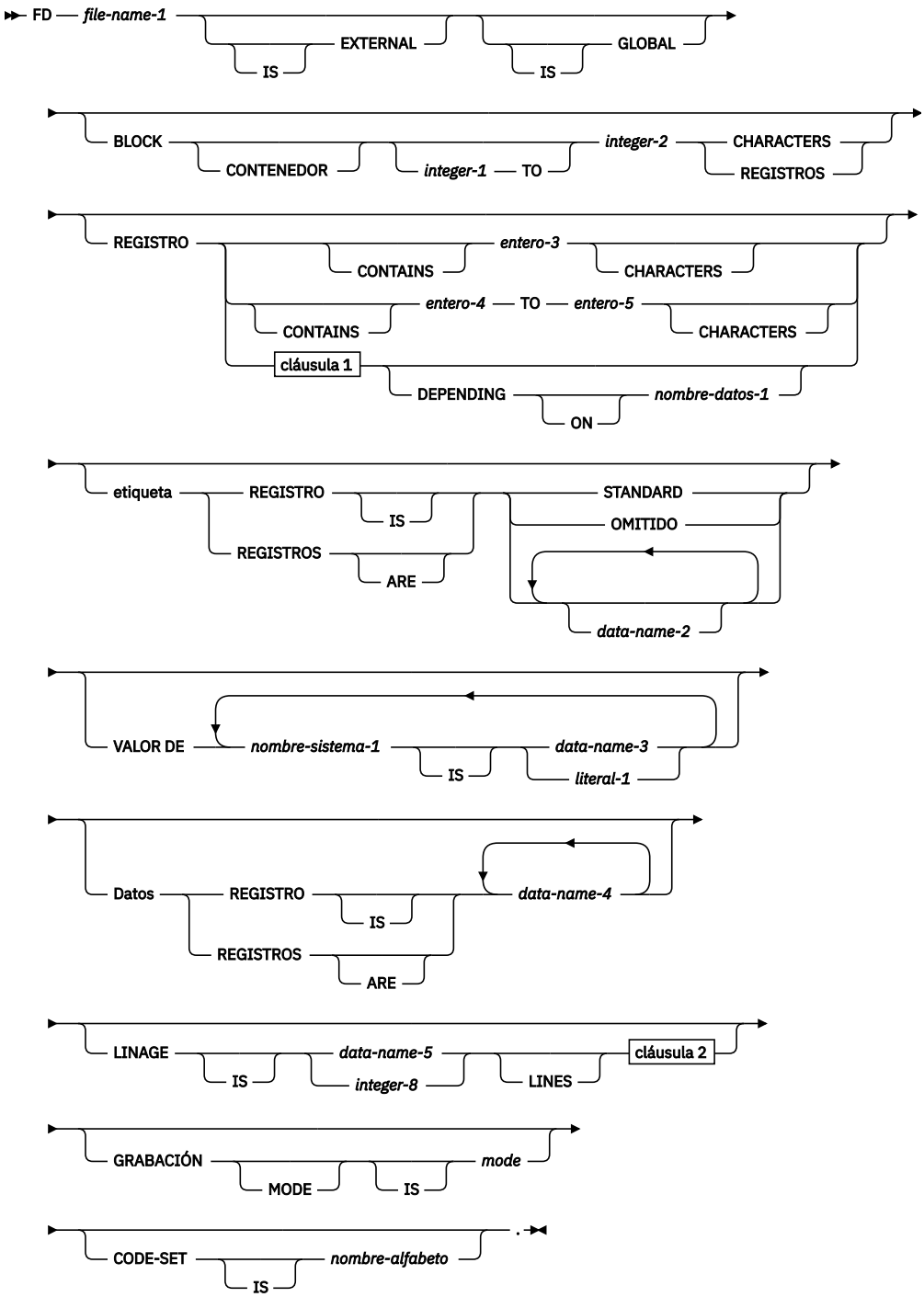
Edición de signos

Los signos de edición están asociados con elementos editados numérica-. Los signos de edición son símbolos PICTURE que identifican el signo del elemento en la salida editada.

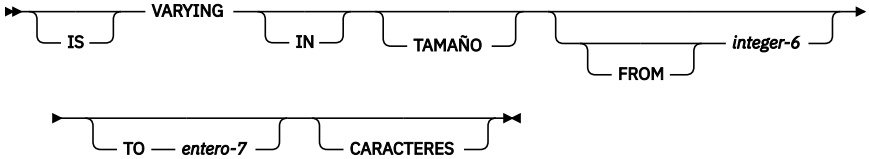
Capítulo 16. DATA DIVISION -- entradas de descripción de archivo

En un programa COBOL, la *Entrada de descripción de archivo (FD)* (o *Ordenar entrada de descripción de archivo (SD)* para los archivos de clasificación/fusión) representa el nivel más alto de organización en FILE SECTION. El orden en el que las cláusulas opcionales siguen a la entrada FD o SD no es importante.

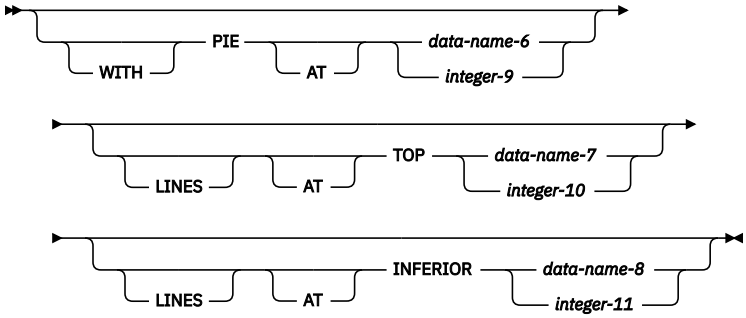
Formato 1: entrada de descripción de archivo secuencial



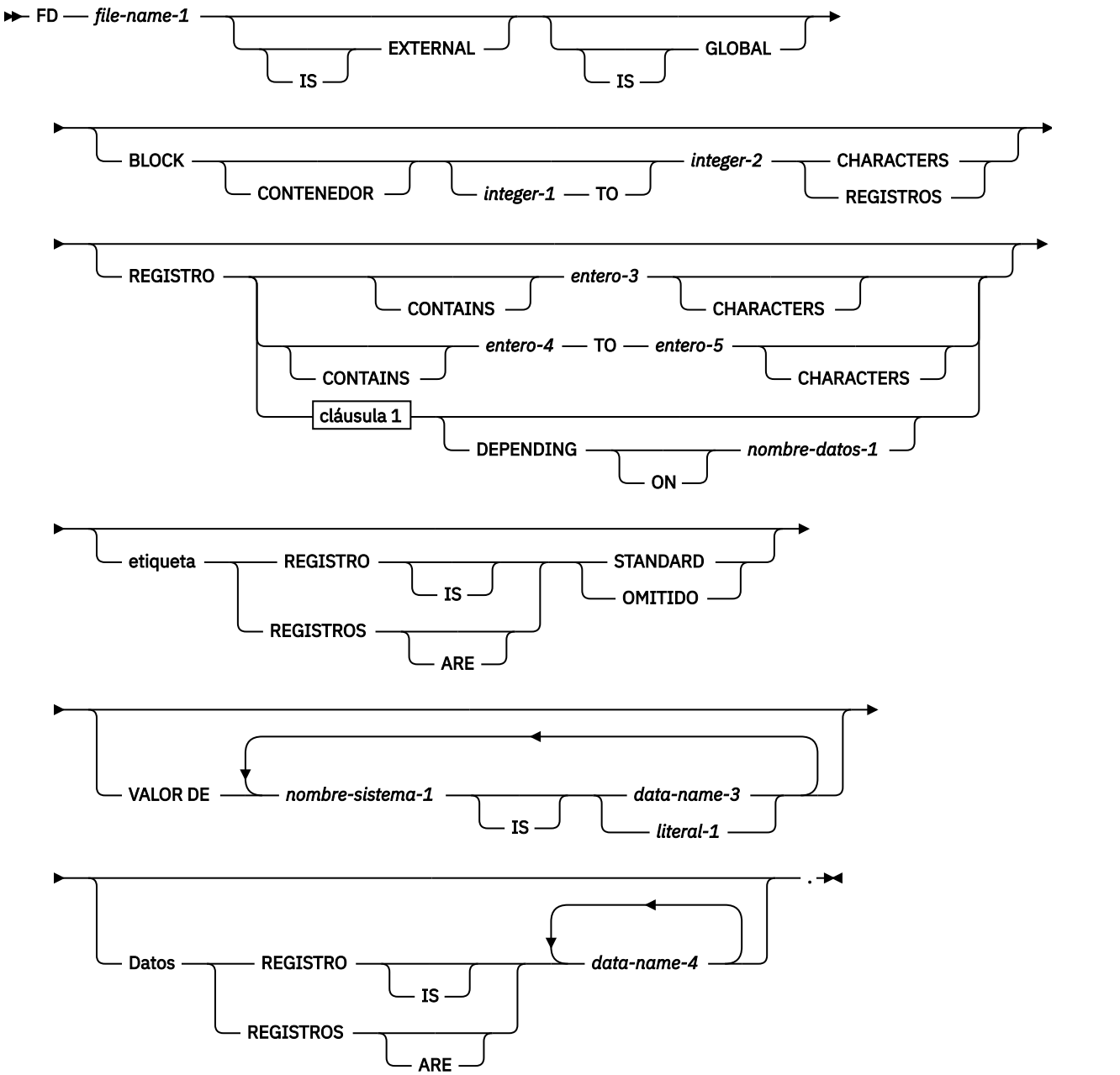
cláusula 1



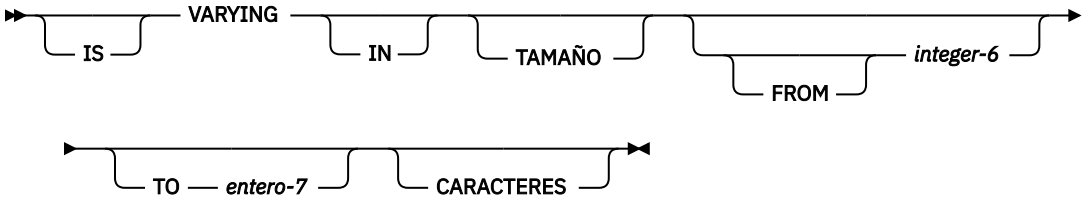
cláusula 2



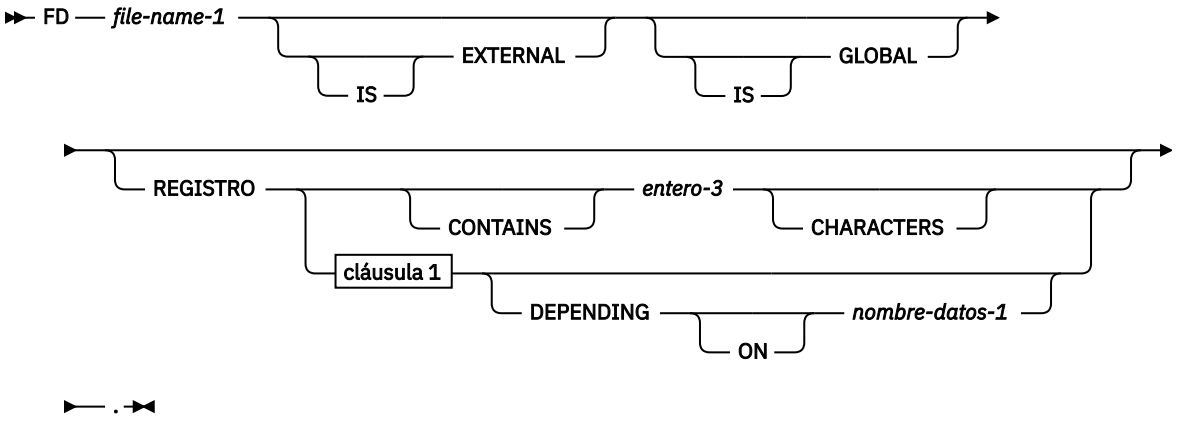
Formato 2: entrada de descripción de archivo relativa o indexada



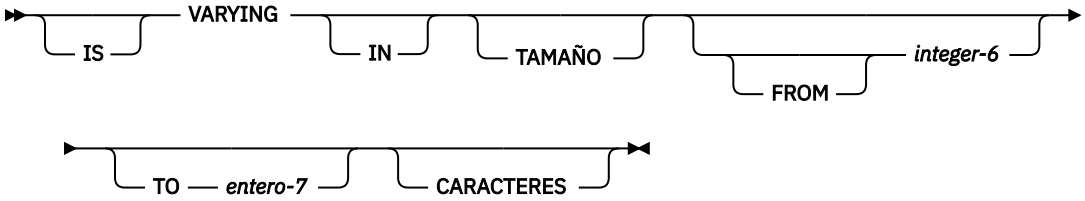
cláusula 1



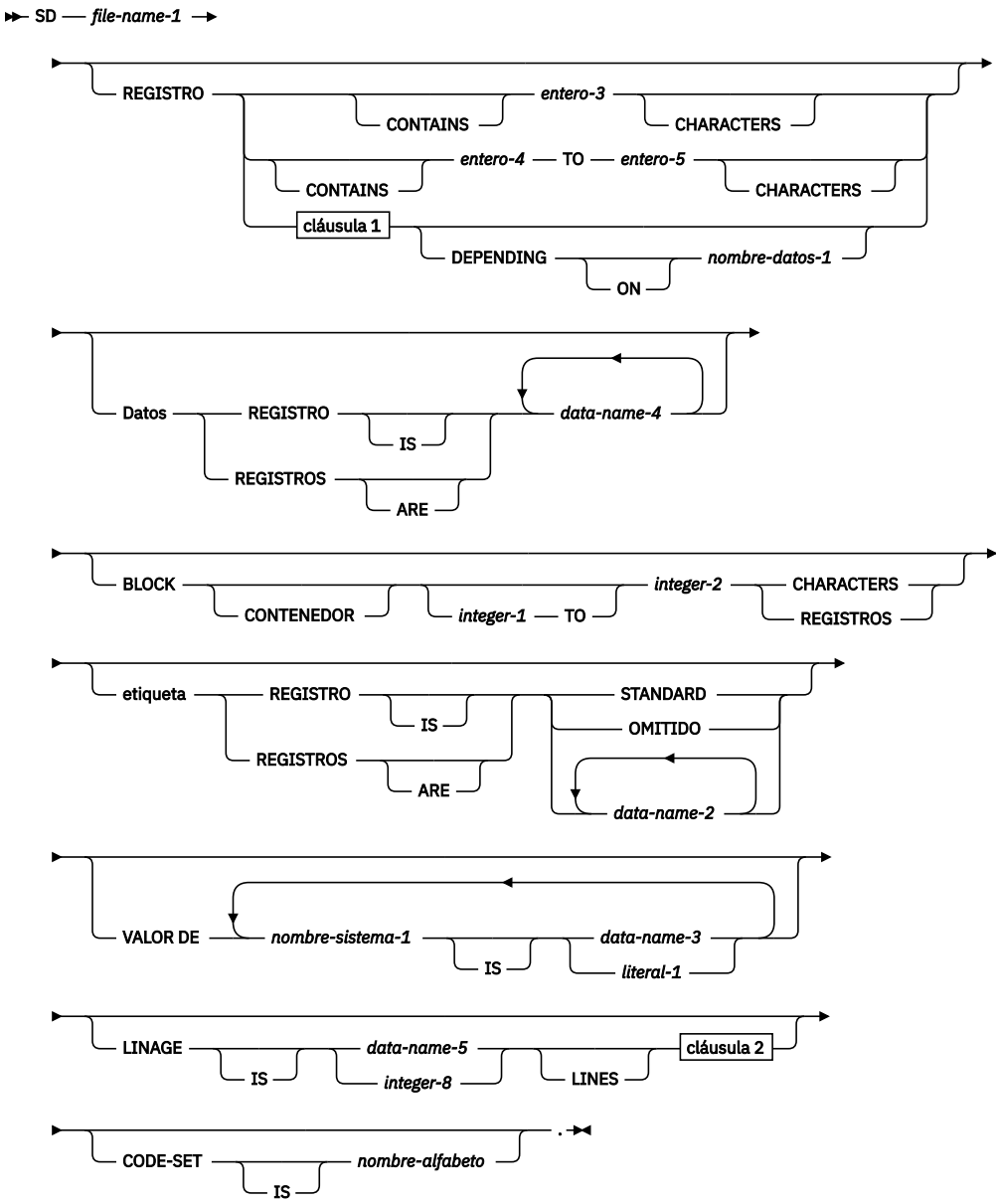
Formato 3: entrada de descripción de archivo secuencial de línea



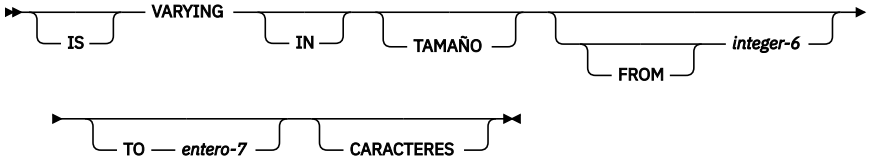
cláusula 1



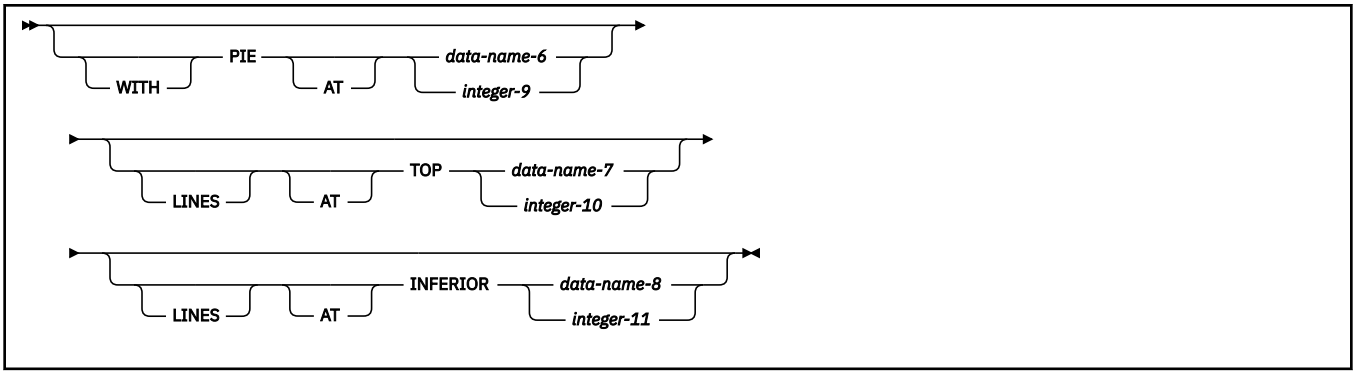
Formato 4: ordenar/fusionar entrada de descripción de archivo



cláusula 1



cláusula 2



SECCIÓN DE ARCHIVO

FILE SECTION debe contener un indicador de nivel para cada archivo de entrada y salida. Para todos los archivos excepto los archivos de clasificación o fusión, FILE SECTION debe contener una entrada FD. Para cada archivo de clasificación o fusión, FILE SECTION debe contener una entrada SD.

nombre-archivo
Debe seguir el indicador de nivel (FD o SD) y debe ser el mismo que el especificado en la cláusula SELECT asociada. *nombre-archivo* debe cumplir las reglas de formación para una palabra definida por el usuario; al menos un carácter debe ser alfabético. *nombre-archivo* debe ser exclusivo dentro de este programa.

Una o más entradas de descripción de registro deben seguir a *nombre-archivo*. Una entrada de descripción de registro puede describir un *nombre-tipo*. Cada entrada, que no es un nombre de tipo, implica una redefinición de la misma área de almacenamiento.

Las cláusulas que siguen a *nombre-archivo* son opcionales y pueden aparecer en cualquier orden.

FD (formatos 1, 2 y 3)
La última cláusula de la entrada FD debe ir seguida inmediatamente de un punto separador.

SD (formato 4)
Debe escribirse una entrada SD para cada archivo de clasificación o fusión del programa. La última cláusula de la entrada SD debe ir seguida inmediatamente de un punto separador.

El ejemplo siguiente ilustra las entradas FILE SECTION necesarias para un archivo de clasificación o fusión:

```
SD  SORT-FILE.
01  SORT-RECORD  PICTURE X(80).
```

Un registro en FILE SECTION debe describirse como un elemento de grupo alfanumérico, un elemento de grupo nacional o un elemento elemental de clase alfabética, alfanumérica, DBCS, nacional o numérica.

cláusula EXTERNAL

La cláusula EXTERNAL especifica que un conector de archivo es externo y permite la comunicación entre dos programas mediante el uso compartido de archivos.

Un conector de archivo es externo si el almacenamiento asociado con ese archivo está asociado con la unidad de ejecución en lugar de con cualquier programa en particular dentro de la unidad de ejecución. Cualquier programa de la unidad de ejecución que describa el archivo puede hacer referencia a un archivo externo. Las referencias a un archivo externo de distintos programas que utilizan descripciones separadas del archivo siempre están en el mismo archivo. En una unidad de ejecución, sólo hay un representante de un archivo externo.

En FILE SECTION, la cláusula EXTERNAL sólo puede especificarse en entradas de descripción de archivo.

Los registros que aparecen en la entrada de descripción de archivo no necesitan tener el mismo nombre en las entradas de descripción de archivo externo correspondientes. Además, no es necesario que el número de estos registros sea el mismo en las entradas de descripción de archivo correspondientes.

El uso de la cláusula EXTERNAL no implica que el nombre de archivo asociado sea un nombre global. Consulte *Compartición de datos utilizando la cláusula EXTERNAL* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener información específica sobre el uso de la cláusula EXTERNAL. La cláusula TYPEDEF no se puede especificar en la misma entrada de descripción de datos que la cláusula EXTERNAL; sin embargo, la cláusula TYPE sí.

cláusula GLOBAL

La cláusula GLOBAL especifica que el conector de archivo nombrado por un nombre de archivo es un nombre global. Un nombre de archivo global está disponible para el programa que lo declara y para cada programa que está contenido directa o indirectamente en ese programa.

La cláusula GLOBAL puede especificarse en la misma entrada de descripción de datos que la cláusula TYPEDEF. El ámbito de la cláusula se aplica sólo al nombre-tipo, y no a ningún elemento de datos definido utilizando un nombre-tipo global con una cláusula TYPE.

Un nombre de archivo es global si se especifica la cláusula GLOBAL en la entrada de descripción de archivo para ese nombre de archivo. Un nombre de registro es global si se especifica la cláusula GLOBAL en la entrada de descripción de registro por la que se declara el nombre de registro o, en el caso de entradas de descripción de registro en FILE SECTION, si se especifica la cláusula GLOBAL en la entrada de descripción de archivo para el nombre de archivo asociado a la entrada de descripción de registro. Estas entradas de descripción de registro pueden describir un nombre de tipo. Para obtener detalles sobre cómo utilizar la cláusula GLOBAL, consulte *Utilización de datos en operaciones de entrada y salida* y *Ámbito de nombres* en *COBOL for Linux en x86 Guía de programación*.

Dos programas en una unidad de ejecución pueden hacer referencia a conectores de archivos globales en las circunstancias siguientes:

- Se puede hacer referencia a un conector de archivo externo desde cualquier programa que describa dicho conector de archivo.
- Si un programa está contenido en otro programa, ambos programas pueden hacer referencia a un conector de archivo global haciendo referencia a un nombre de archivo global asociado en el programa contenedor o en cualquier programa que contenga directa o indirectamente el programa contenedor.

cláusula BLOCK CONTAINS

La cláusula BLOCK CONTAINS se comprueba con la sintaxis pero no tiene ningún efecto en la ejecución del programa.

integer-1 , integer-2
Debe ser distinto de cero enteros sin signo. Especifican:

CARACTERES
Especifica el número de bytes necesarios para almacenar el registro físico, independientemente de la USAGE que tengan los elementos de datos en el registro de datos.

Si sólo se especifica *integer-2* , especifica el número exacto de bytes en el registro físico. Cuando se especifican *integer-1* y *integer-2* , ambos representan el número mínimo y máximo de bytes en el registro físico, respectivamente.

integer-1 y *integer-2* deben incluir los bytes de control y el relleno contenidos en el registro físico. (los registros lógicos no incluyen el relleno.)

La frase CHARACTERS es el valor predeterminado. Se debe especificar CHARACTERS cuando:

- El registro físico contiene relleno.

- Los registros lógicos se agrupan para que se pueda implicar un tamaño de registro físico inexacto. Por ejemplo, supongamos que describe un registro de longitud variable de 100 bytes, pero cada vez que escribe un bloque de 4, se escribe un registro de 50 bytes seguido de tres registros de 100 bytes. Si se especificara la frase RECORDS, el compilador calcularía el tamaño de bloque como 420 bytes en lugar del tamaño real, 370 bytes. (Este cálculo incluye descriptores de bloque y de registro.)

registros
Especifica el número de registros lógicos contenidos en cada registro físico.

El compilador presupone que el tamaño de bloque debe proporcionar *integer-2* registros de tamaño máximo y proporciona cualquier espacio adicional necesario para los bytes de control.

cláusula RECORD

Cuando se utiliza la cláusula RECORD, el tamaño de registro debe especificarse como el número de bytes necesarios para almacenar el registro internamente, independientemente de la USAGE de los elementos de datos contenidos en el registro.

Por ejemplo, si tiene un registro con 10 caracteres DBCS, la cláusula RECORD debería decir `RECORD CONTAINS 20 CHARACTERS`. Para un registro con 10 caracteres nacionales, la cláusula RECORD debe decir lo mismo, `RECORD CONTAINS 20 CHARACTERS`.

El tamaño de un registro se determina de acuerdo con las reglas para obtener el tamaño de un elemento de grupo. (Consulte "cláusula USAGE" en la página 225 y "cláusula SYNCHRONIZED" en la página 218.)

Cuando se omite la cláusula RECORD, el compilador determina las longitudes de registro a partir de las descripciones de registro. Cuando una de las entradas de una descripción de registro contiene una cláusula OCCURS DEPENDING ON, el compilador utiliza el valor máximo del elemento de longitud variable para calcular el número de bytes necesarios para almacenar el registro internamente.

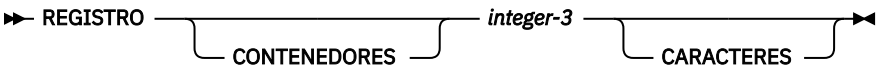
Si el conector de archivo asociado es un conector de archivo externo, todas las entradas de descripción de archivo de la unidad de ejecución que están asociadas con ese conector de archivo deben especificar el mismo número máximo de bytes.

Las secciones siguientes describen los formatos de la cláusula RECORD:

Formato 1

El formato 1 especifica el número de bytes para registros de longitud fija.

Formato 1



integer-3
Debe ser un entero sin signo que especifique el número de bytes contenidos en cada registro del archivo.

La cláusula RECORD CONTAINS 0 caracteres es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

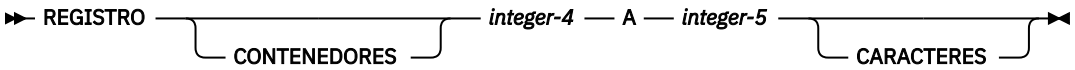
No especifique la cláusula RECORD CONTAINS 0 para una entrada SD.

Formato 2

El formato 2 especifica el número de bytes para registros de longitud fija o de longitud variable.

Los registros de longitud fija se obtienen cuando todas las longitudes de entrada de descripción de registro 01 son iguales. La cláusula format-2 RECORD CONTAINS nunca es necesaria, porque las longitudes de registro mínima y máxima se determinan a partir de las entradas de descripción de registro.

Formato 2

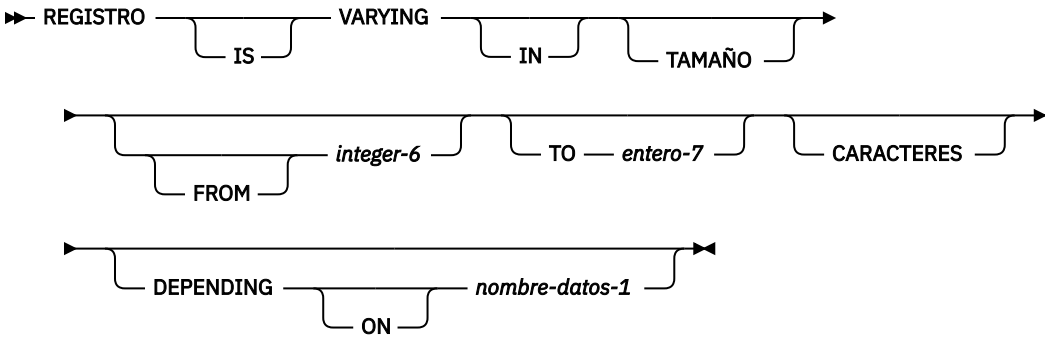


integer-4* , *integer-5
Deben ser enteros sin signo. *integer-4* especifica el tamaño del registro de datos más pequeño y *integer-5* especifica el tamaño del registro de datos más grande.

Formato 3

El formato 3 se utiliza para especificar registros de longitud variable.

Formato 3



integer-6
Especifica el número mínimo de bytes que debe contener cualquier registro del archivo. Si no se especifica *integer-6* , el número mínimo de bytes que debe contener cualquier registro del archivo es igual al número mínimo de bytes descrito para un registro en dicho archivo.

integer-7
Especifica el número máximo de bytes en cualquier registro del archivo. Si no se especifica *integer-7* , el número máximo de bytes que debe contener cualquier registro del archivo es igual al número máximo de bytes descrito para un registro en ese archivo.

El número de bytes asociados a una descripción de registro viene determinado por la suma del número de bytes en todos los elementos de datos elementales (excluyendo las redefiniciones y las renombraciones), más cualquier FILLER implícito debido a la sincronización. Si se especifica una tabla:

- El número mínimo de elementos de tabla descritos en el registro se utiliza en la suma anterior para determinar el número mínimo de bytes asociados a la descripción de registro.
- El número máximo de elementos de tabla descritos en el registro se utiliza en la suma anterior para determinar el número máximo de bytes asociados con la descripción de registro.

Si se especifica *data-name-1* :

- *data-name-1* debe ser un entero elemental sin signo.
- *data-name-1* no puede ser un campo de fecha con ventana.
- El número de bytes del registro debe colocarse en el elemento de datos al que hace referencia *data-name-1* antes de ejecutar cualquier sentencia RELEASE, REWRITE o WRITE para el archivo.

- La ejecución de una sentencia DELETE, RELEASE, REWRITE, START o WRITE o la ejecución no satisfactoria de una sentencia READ o RETURN no altera el contenido del elemento de datos al que hace referencia *data-name-1*.
- Después de la ejecución satisfactoria de una sentencia READ o RETURN para el archivo, el contenido del elemento de datos al que hace referencia *data-name-1* indica el número de bytes del registro que se acaba de leer.

Durante la ejecución de una sentencia RELEASE, REWRITE o WRITE, el número de bytes del registro viene determinado por las condiciones siguientes:

- Si se especifica *data-name-1* , por el contenido del elemento de datos al que hace referencia *data-name-1*
- Si no se especifica *data-name-1* y el registro no contiene un elemento de datos de aparición variable, por el número de posiciones de bytes en el registro
- Si no se especifica *data-name-1* y el registro contiene un elemento de datos de aparición variable, por la suma de la posición fija y la parte de la tabla descrita por el número de apariciones en el momento de la ejecución de la sentencia de salida

Durante la ejecución de un READ ... INTO o RETURN ... Sentencia INTO, el número de bytes del registro actual que participan como elementos de datos de envío en la sentencia MOVE implícita viene determinado por las condiciones siguientes:

- Si se especifica *data-name-1* , por el contenido del elemento de datos al que hace referencia *data-name-1*
- Si no se especifica *data-name-1* , por el valor que se habría movido al elemento de datos al que hace referencia *data-name-1* si se hubiera especificado *data-name-1*

cláusula LABEL RECORDS

La cláusula LABEL RECORDS es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Se emite un mensaje de aviso si utiliza alguno de los siguientes elementos de lenguaje:

- LABEL RECORD IS nombre-datos
- USE ... DESPUÉS ... PROCEDIMIENTO DE ETIQUETA

La cláusula LABEL RECORDS indica la presencia o ausencia de etiquetas. Si no se especifica para un archivo, los registros de etiqueta para ese archivo deben ajustarse a las especificaciones de etiqueta del sistema.

Standard
Existen etiquetas que se ajustan a las especificaciones del sistema para este archivo.

STANDARD está permitido para dispositivos de almacenamiento masivo y dispositivos de cinta.

OMITIDO
No existen etiquetas para este archivo.

OMITIDO está permitido para dispositivos de cinta.

data-name-2
Las etiquetas de usuario están presentes además de las etiquetas estándar. *data-name-2* especifica el nombre de un registro de etiqueta de usuario. *data-name-2* debe aparecer como el asunto de una entrada de descripción de registro asociada con el archivo.

cláusula VALUE OF

La cláusula VALUE OF describe un elemento en los registros de etiqueta asociados con el archivo.

data-name-3
Debe calificarse cuando sea necesario, pero no se puede suscribir. Debe describirse en la SECCIÓN DE TRABAJO-ALMACENAMIENTO. No se puede describir con la cláusula USAGE IS INDEX.

literal-1
Puede ser numérico o alfanumérico, o una constante figurativa de categoría numérica o alfanumérica. No puede ser un literal de coma flotante.

La cláusula VALUE OF es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

cláusula DATA RECORDS

La cláusula DATA RECORDS se comprueba con la sintaxis, pero sólo sirve como documentación para los nombres de los registros de datos asociados con el archivo.

data-name-4
Los nombres de las entradas de descripción de registro asociadas con el archivo. *data-name-4* no debe ser un nombre-tipo.

El nombre de datos no necesita tener una descripción de registro de número de nivel 01 asociada con el mismo nombre.

cláusula LINAGE

La cláusula LINAGE especifica la profundidad de una página lógica en número de líneas. Opcionalmente, también especifica el número de línea en el que empieza el área de pie y los márgenes superior e inferior de la página lógica. (La página lógica y la página física no pueden tener el mismo tamaño.)

La cláusula LINAGE es efectiva para los archivos secuenciales abiertos como OUTPUT o EXTEND.

Todos los enteros deben estar sin signo. Todos los nombres de datos deben describirse como elementos de datos enteros sin signo.

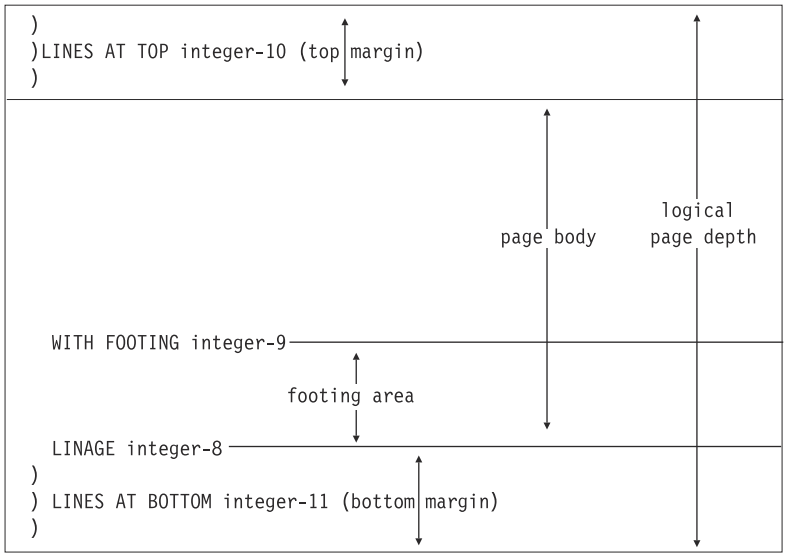
data-name-5 , integer-8
El número de líneas que se pueden escribir o espaciar en esta página lógica. El área de la página que representan estas líneas se denomina *cuerpo de página*. El valor debe ser mayor que cero.

CON PIE EN
integer-9 o el valor del elemento de datos en *data-name-6* especifica el primer número de línea del área de pie dentro del cuerpo de página. El número de línea de pie debe ser mayor que cero y no mayor que la última línea del cuerpo de página. El área de pie se extiende entre esas dos líneas.

LÍNEAS EN LA PARTE SUPERIOR
integer-10 o el valor del elemento de datos en *data-name-7* especifica el número de líneas en el margen superior de la página lógica. El valor puede ser cero.

LÍNEAS EN LA PARTE INFERIOR
integer-11 o el valor del elemento de datos en *data-name-8* especifica el número de líneas en el margen inferior de la página lógica. El valor puede ser cero.

La figura siguiente ilustra el uso de cada frase de la cláusula LINAGE.



El tamaño de página lógica especificado en la cláusula LINAGE es la suma de todos los valores especificados en cada frase excepto la frase CABECERA. Si se omite la frase LINES AT TOP, el valor asumido para el margen superior es cero. De forma similar, si se omite la frase LINES AT BOTTOM, el valor asumido para el margen inferior es cero. Cada página lógica sigue inmediatamente a la página lógica anterior, sin que se haya proporcionado ningún espaciado adicional.

Si se omite la frase INICIAL, su valor asumido es igual al del cuerpo de la página (*integer-8* o *data-name-5*).

En el momento en que se ejecuta una sentencia OPEN OUTPUT, los valores de *integer-8*, *integer-9*, *integer-10*y *integer-11*, si se especifica, se utilizan para determinar el cuerpo de página, la primera línea de pie de página, el margen superior y el margen inferior de la página lógica para este archivo. (Véase la figura anterior.) A continuación, estos valores se utilizan para todas las páginas lógicas impresas para este archivo durante una ejecución determinada del programa.

En el momento en que se ejecuta una sentencia OPEN con la frase OUTPUT para el archivo, *data-name-5*, *data-name-6*, *data-name-7*y *data-name-8* determinan el cuerpo de la página, la primera línea de pie, el margen superior y el margen inferior sólo para la primera página lógica.

En el momento en que se ejecuta una sentencia WRITE con la frase AVANZANDO PÁGINA o se produce una condición de desbordamiento de página, los valores de *data-name-5*, *data-name-6*, *data-name-7*y *data-name-8* si se especifica, se utilizan para determinar el cuerpo de la página, la primera línea de pie de página, el margen superior y el margen inferior de la siguiente página lógica.

Si un conector de archivo externo está asociado con esta entrada de descripción de archivo, todas las entradas de descripción de archivo de la unidad de ejecución que están asociadas con este conector de archivo deben tener:

- Una cláusula LINAGE, si alguna entrada de descripción de archivo tiene una cláusula LINAGE
- Los mismos valores correspondientes para *integer-8*, *integer-9*, *integer-10*, y *integer-11*, si se especifica
- Los mismos elementos de datos externos correspondientes a los que hace referencia *data-name-5*, *data-name-6*, *data-name-7*y *data-name-8*

Consulte “Frase AVANZANDO” en la página 425 para ver el comportamiento de los caracteres de control de carro en archivos externos.

Una cláusula LINAGE bajo una SD es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

LINAGE-Registro especial COUNTER

Para obtener información sobre el registro especial LINAGE-COUNTER, consulte “LINAGE-COUNTER” en la página 22.

cláusula REGISTRO MODE

La cláusula RECORD MODE para los archivos secuenciales de registro se trata de la siguiente manera.

F
Las descripciones de registro se validan como fijas. No especifique RECORD MODE F si las descripciones de registro son variables.

V
Se asume el formato de registro de longitud variable (incluso si las descripciones de registro son fijas).

u
Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa.

S

- Para archivos no QSAM, tratados igual que V.
- Para los archivos QSAM, los registros pueden ser de longitud fija o de longitud variable, y pueden ser más grandes que un bloque. Si un registro es mayor que el espacio restante en un bloque, se graba un segmento del registro para rellenar el bloque. El resto del registro se almacena en el siguiente bloque (o bloques, si es necesario). Sólo los registros completos se ponen a su disposición. Cada segmento de un registro en un bloque, incluso si es el registro completo, incluye un campo de descriptor de segmento, y cada bloque incluye un campo de descriptor de bloque. Estos campos no se describen en la DIVISIÓN DE DATOS; se establece automáticamente para ellos. Estos campos no están disponibles para usted.

cláusula CODE-SET

La cláusula CODE-SET es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Capítulo 17. DATA DIVISION -- entrada de descripción de datos

Una *entrada de descripción de datos* especifica las características de un elemento de datos. En las secciones siguientes, los conjuntos de entradas de descripción de datos se denominan *entradas de descripción de registro*. El término *entrada de descripción de datos* hace referencia a entradas de descripción de datos y registros.

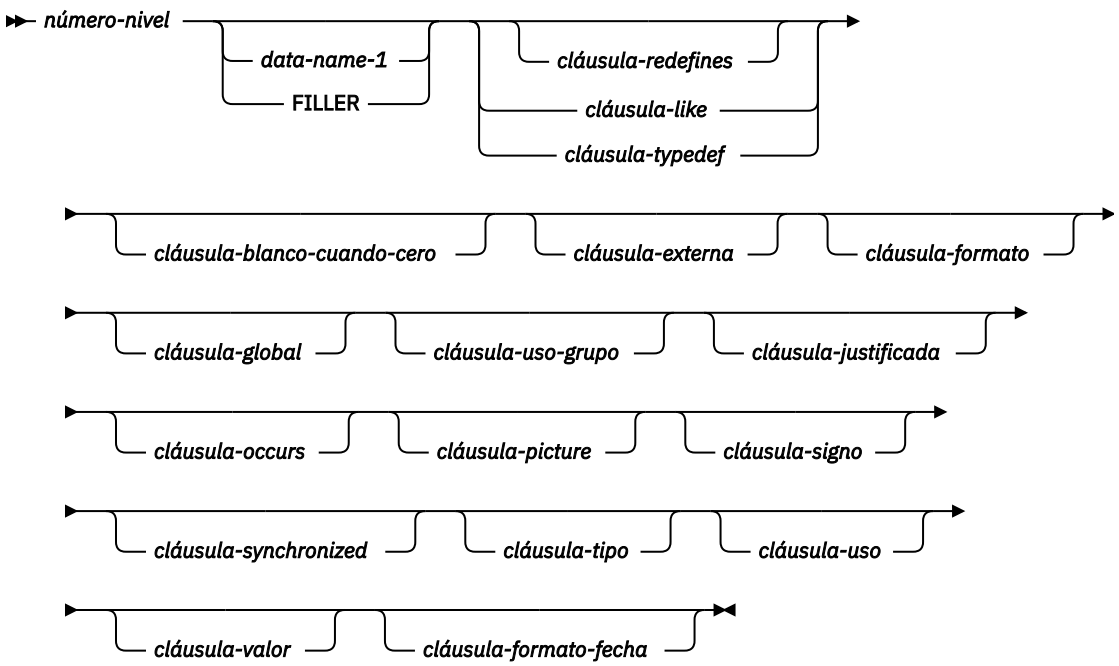
Las entradas de descripción de datos que definen elementos de datos independientes no forman un registro. Estas entradas se conocen como *entradas de descripción de elemento de datos*.

Las entradas de descripción de datos tienen tres formatos generales y todas las entradas de descripción de datos deben finalizar con un punto separador.

Formato 1

El formato 1 se utiliza para las entradas de descripción de datos en todas las secciones DATA DIVISION.

Formato 1: entrada de descripción de datos



Las cláusulas se pueden escribir en cualquier orden, con las siguientes excepciones:

- *data-name-1* o FILLER, si se especifica, debe ir inmediatamente después del número de nivel.
- Cuando se especifica la cláusula REDEFINES, debe ir inmediatamente después de *data-name-1* o FILLER, si se especifica alguna. Si no se especifica *data-name-1* o FILLER, la cláusula REDEFINES debe ir inmediatamente después del número de nivel.
- Cuando se especifica la cláusula TYPEDEF, debe ser la primera entrada después de *data-name-1*. La cláusula TYPEDEF no se puede especificar con FILLER. La cláusula TYPEDEF y la cláusula REDEFINES no se pueden especificar a la vez para *data-name-1*.

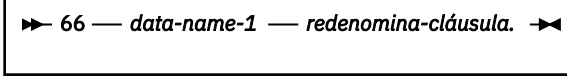
El número de nivel en formato 1 puede ser cualquier número en el rango 01–49 o 77.

Un espacio, una coma o un punto y coma deben separar las cláusulas.

Formato 2

El formato 2 reagrupa los elementos definidos anteriormente.

Formato 2: renombra



Una entrada level-66 no puede renombrar otra entrada level-66 , ni puede renombrar una entrada level-01, level-77o level-88 .

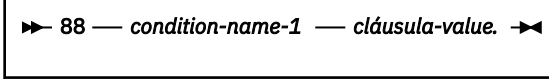
Todas las entradas de level-66 asociadas con un registro deben ir inmediatamente después de la última entrada de descripción de datos de ese registro.

Consulte “cláusula RENAMES” en la página 215 para obtener más detalles.

Formato 3

El formato 3 describe los nombres de condición.

Formato 3: nombre-condición



condition-name-1
Nombre especificado por el usuario que asocia un valor, un conjunto de valores o un rango de valores con una variable condicional.

Las entradas de Level-88 deben ir inmediatamente después de la entrada de descripción de datos para la variable condicional con la que están asociados los nombres de condición.

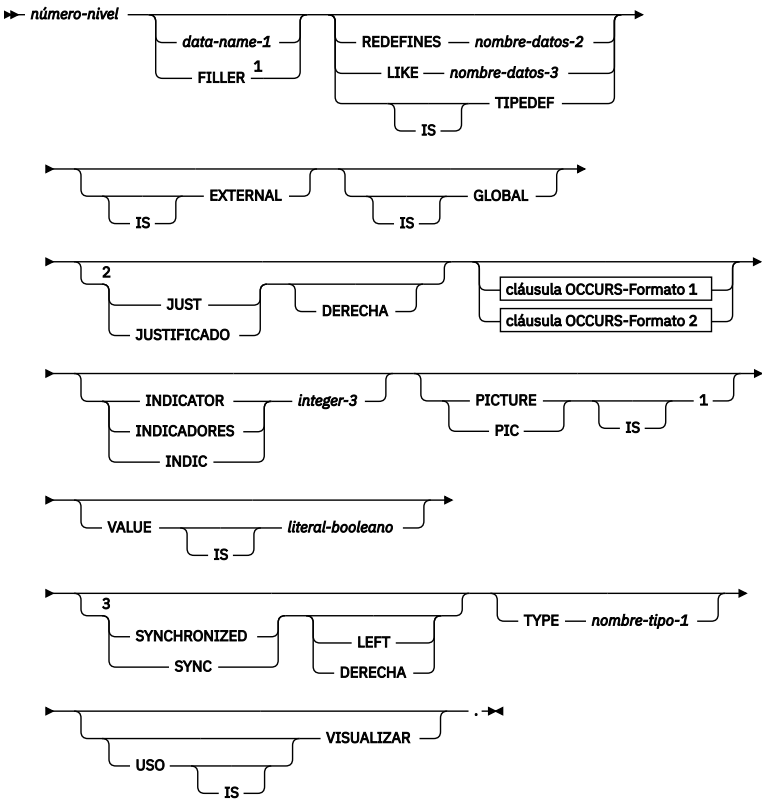
El formato 3 se puede utilizar para describir elementos elementales, elementos de grupo nacional o elementos de grupo alfanuméricos. Puede encontrar información adicional sobre las entradas de nombre de condición en “cláusula VALUE” en la página 232 y “Condición de nombre de condición” en la página 253.

Formato 4

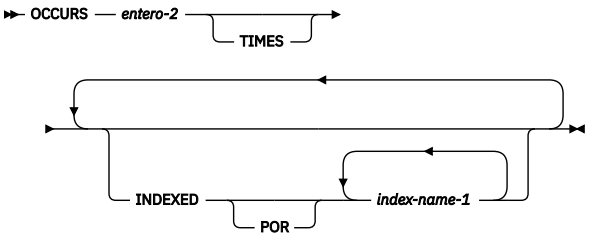
Este formato describe los datos booleanos. Los elementos de datos booleanos son elementos que están limitados a un valor de 1 o 0.

Nota: Cuando utilice indicadores en un programa COBOL, debe describirlos como elementos de datos booleanos utilizando la entrada de descripción de datos para datos booleanos.

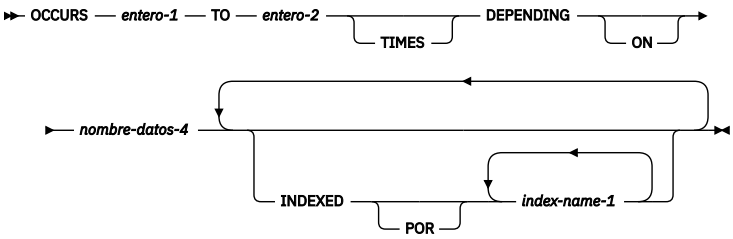
Formato 4: datos booleanos



Cláusula OCCURS-Formato 1



cláusula OCCURS-Formato 2



Notas:

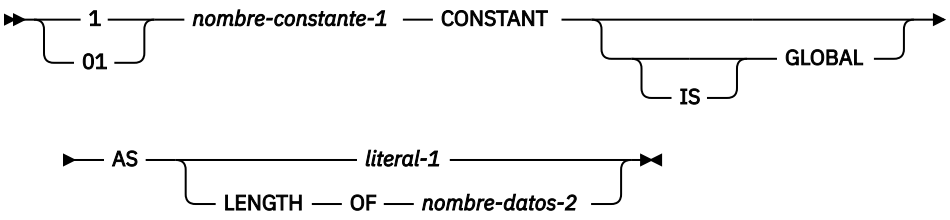
[1] No se puede utilizar con la cláusula TYPEDEF
[2] Sólo comprobación de sintaxis
[3] Sólo comprobación de sintaxis

A continuación se muestran las consideraciones especiales para las cláusulas utilizadas con los datos booleanos. Todas las demás reglas para cláusulas son las mismas que para otros datos.

Formato 5

El formato 5 describe los nombres de constante. El nombre de constante sólo se puede describir como una entrada level-01 .

Formato 5: nombre-constante



La cláusula CONSTANT se utiliza para asociar un nombre de constante con un literal. A continuación, se puede utilizar el nombre de constante en lugar de un literal. La cláusula CONSTANT sólo se puede especificar para las entradas level-01 para el nombre de constante elemental. La cláusula CONSTANT también se puede definir como otro nombre de constante definido anteriormente.

Es necesario definir un nombre de constante en una cláusula CONSTANT antes de su uso. Se puede utilizar en DATA DIVISION y PROCEDURE DIVISION donde se permite literal o entero, excepto en las sentencias de dirección de compilador, como por ejemplo la sentencia COPY y la sentencia TITLE.

constant-name-1 se puede utilizar en cualquier lugar en el que un formato especifique un literal de la clase y categoría de *constant-name-1*. La clase y la categoría de *constant-name-1* es la misma que la de *literal-1* o es un entero si se especifica la frase LENGTH OF. Si *constant-name-1* es un entero, también se puede utilizar para especificar la repetición en una serie de imagen.

Literal-1 no puede ser una constante figurativa.

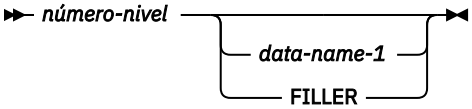
Si se especifica la frase LENGTH OF, el valor de *constant-name-1* se determina tal como se especifica en la función intrínseca LENGTH con la excepción de que cuando *data-name-2* es un elemento de datos de longitud variable descrito con la cláusula OCCURS DEPENDING ON, se utiliza el tamaño máximo del elemento de datos.

Nivel-números

El número de nivel especifica la jerarquía de datos dentro de un registro e identifica entradas de datos especiales. Un número de nivel empieza una entrada de descripción de datos, un elemento redenominado o redefinido, o una entrada de nombre de condición.

Un número de nivel tiene un valor entero entre 1 y 49, inclusive, o uno de los valores de número de nivel especial 66, 77 u 88.

Formato



número-nivel

01 y 77 deben empezar en el Área A y deben ir seguidos de un punto de separación o de un espacio seguido de su nombre de datos asociado, FILLER o la cláusula de descripción de datos adecuada.

Los números de nivel 02 a 49 pueden empezar en las áreas A o B y deben ir seguidos de un espacio o un punto de separación.

Los números de nivel 66 y 88 pueden empezar en las áreas A o B y deben ir seguidos de un espacio.

Los números de nivel 1 a 9 de un solo dígito se pueden sustituir por los números de nivel 01 a 09.

Las entradas de descripción de datos sucesivas pueden empezar en la misma columna que la primera entrada o pueden sangrarse según el número de nivel. La sangría no afecta a la magnitud de un número de nivel.

Cuando los números de nivel están sangrados, cada nuevo número de nivel puede comenzar cualquier número de espacios a la derecha del Área A. La extensión de la sangría a la derecha sólo está limitada por la anchura del área B.

Para obtener más información, consulte "Niveles de datos" en la página 142.

data-name-1
Identifica explícitamente los datos que se describen.

data-name-1, si se especifica, identifica un elemento de datos utilizado en el programa. *data-name-1* debe ser la primera palabra después del número de nivel.

El elemento de datos puede cambiarse durante la ejecución del programa.

data-name-1 debe especificarse para los elementos level-66 y level-88 . También debe especificarse para cualquier entrada que contenga la cláusula GLOBAL o EXTERNAL, y para las entradas de descripción de registro asociadas con entradas de descripción de archivo que tengan las cláusulas GLOBAL o EXTERNAL.

FILLER
Elemento de datos al que no se hace referencia explícitamente en un programa. La palabra clave FILLER es opcional. Si se especifica, FILLER debe ser la primera palabra después del número de nivel.

La palabra clave FILLER se puede utilizar con una variable condicional si nunca se hace referencia explícita a la variable condicional, sino sólo a los valores que puede asumir. FILLER no se puede utilizar con un nombre de condición.

En una sentencia MOVE CORRESPONDIENTE o en una sentencia ADD CORRESPONDIENTE o RESTAR CORRESPONDIENTE, los elementos FILLER se ignoran.

En una sentencia INITIALIZE:

- Cuando no se especifica la frase FILLER, se ignoran los elementos FILLER elementales.
- Cuando se especifica la frase FILLER, se inicializarán los elementos de datos elementales de recepción que tienen una cláusula FILLER explícita o implícita.

Si se omite *data-name-1* o la cláusula FILLER, el elemento de datos que se describe se trata como si se hubiera especificado FILLER.

Cláusula BLANK WHEN ZERO

La cláusula BLANK WHEN ZERO especifica que un elemento sólo contiene espacios cuando su valor es cero.

Formato

La cláusula BLANK WHEN ZERO sólo puede especificarse para un elemento elemental descrito por su serie de caracteres de imagen como de categoría numérica editada o numérica, sin el símbolo de imagen S o *. Estos elementos deben describirse, implícita o explícitamente, como USAGE DISPLAY o USAGE NATIONAL.

Una cláusula BLANK WHEN ZERO que se especifica para un elemento definido como numérico por su serie de caracteres de imagen define el elemento como de categoría editada numérica-.

No debe especificarse la cláusula BLANK WHEN ZERO para los campos de fecha.

cláusula DATE FORMAT

La cláusula DATE FORMAT especifica que un elemento de datos es un campo de fecha con ventana o expandido.

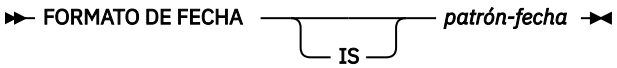
Campos de fecha con ventana
Contiene un año con ventana (dos dígitos), especificado por una cláusula DATE FORMAT que contiene AA.

Campos de fecha expandidos
Contiene un año expandido (de cuatro dígitos), especificado por una cláusula DATE FORMAT que contiene AAAA.

Si la opción de compilador NODATEPROC está en vigor, la cláusula DATE FORMAT se comprueba con la sintaxis pero no tiene ningún efecto en la ejecución del programa. NODATEPROC inhabilita el proceso de fecha. Las reglas y restricciones descritas en esta referencia para la cláusula DATE FORMAT y los campos de fecha sólo se aplican si la opción de compilador DATEPROC está en vigor.

La cláusula DATE FORMAT no debe especificarse para un elemento de datos descrito con USAGE NATIONAL.

Formato



patrón-fecha es una serie de caracteres, como por ejemplo YYXXXX, que representa un año con ventanas o expandido, opcionalmente seguido o precedido de uno a cuatro caracteres que representan otras partes de una fecha como, por ejemplo, el mes y el día:

Serie de patrón de fecha	Especifica que el elemento de datos contiene
AA	Un año con ventana (dos dígitos)
AAAA	Un año expandido (cuatro dígitos)
X	Un único carácter; por ejemplo, un dígito que representa un semestre o trimestre (1–4)
XX	Dos caracteres; por ejemplo, dígitos que representan un mes (01–12)
XXX	Tres caracteres; por ejemplo, dígitos que representan un día del año (001–366)
XXXX	Cuatro caracteres; por ejemplo, dos dígitos que representan un mes (01–12) y dos dígitos que representan un día del mes (01–31)

Para obtener una introducción a los campos de fecha y términos relacionados, consulte Capítulo 10, “Millennium Language Extensions y campos de fecha”, en la página 73. Para obtener detalles sobre el uso de campos de fecha en aplicaciones, consulte *Millennium Language Extensions (MLE)* en *COBOL for Linux en x86 Guía de programación*.

Semántica de campos de fecha con ventanas

Los campos de fecha con ventanas experimentan una expansión automática relativa a la ventana de siglo cuando se utilizan como operandos en expresiones aritméticas o sentencias aritméticas. Sin embargo, el resultado de incrementar o disminuir una fecha de ventana se sigue tratando como una fecha de ventana para su posterior cálculo, comparación y almacenamiento.

Cuando se utilizan en las situaciones siguientes, los campos de fecha con ventanas se tratan como si se hubieran convertido al formato de fecha expandido:

- Operandos en restas en los que el otro operando es una fecha ampliada
- Operandos en condiciones de relación
- Envío de campos en sentencias aritméticas o MOVE

Los detalles de la conversión al formato de fecha expandida dependen de si el campo de fecha con ventana es numérico o alfanumérico.

Dado un año de inicio de ventana de siglo de 19*nn*, la parte de año (*aa*) de un campo de fecha con ventana numérico se trata como si se expandiera de la siguiente manera:

- Si *yy* es menor que *nn*, añada 2000 a *yy*.
- Si *yy* es igual o mayor que *nn*, añada 1900 a *yy*.

Para campos de fecha con ventanas numéricas con signo, esto significa que puede haber dos representaciones de algunos años. Por ejemplo, los valores de año con ventana 99 y -01 se tratan como 1999, desde 1900 + 99 = 2000 + -01.

Los campos de fecha con ventana alfanuméricos se tratan de forma similar, pero utilizan un prefijo de 19 o 20 en lugar de añadir 1900 o 2000.

Por ejemplo, cuando se utiliza como operando de una condición de relación, un campo de fecha con ventana definido por:

```
01  DATE-FIELD  DATE FORMAT YYXXXX PICTURE 9(6)
                VALUE IS 450101.
```

se trata como si se tratara de un campo de fecha ampliado con un valor de:

- 19450101, si la ventana del siglo que comienza el año es 1945 o anterior
- 20450101, si la ventana del siglo que comienza el año es posterior a 1945

Restricciones en el uso de campos de fecha

Existen varias restricciones sobre el uso de campos de fecha en contextos diferentes.

Los contextos son:

- Combinación de la cláusula DATE FORMAT con otras cláusulas
- Elementos de grupo que son campos de fecha
- Elementos de idioma que tratan los campos de fecha como no fechas
- Elementos de lenguaje que no aceptan campos de fecha con ventanas como argumentos
- Elementos de lenguaje que no aceptan campos de fecha como argumentos

Para ver las restricciones sobre el uso de campos de fecha en otros contextos, consulte:

- “Aritmética con campos de fecha” en la página 248
- “Comparación de campos de fecha” en la página 264
- “sentencia ADD” en la página 297
- “SUBTRACT, sentencia” en la página 413
- “sentencia MOVE” en la página 347

Combinación de la cláusula DATE FORMAT con otras cláusulas

Existen restricciones sobre la combinación de la cláusula DATE FORMAT con otras cláusulas.

Las frases siguientes son las únicas frases de la cláusula USAGE que se pueden combinar con la cláusula DATE FORMAT:

- BINARIO
- COMPUTATIONAL[1]
- COMPUTATIONAL-3
- COMPUTATIONAL-4
- VISUALIZAR
- DECIMAL EMPAQUETADO

[1]USAGE COMPUTATIONAL no se puede combinar con la cláusula DATE FORMAT si la opción de compilador TRUNC (BIN) está en vigor.

La serie de caracteres PICTURE debe especificar el mismo número de caracteres o dígitos que la cláusula DATE FORMAT. Para los campos de fecha alfanuméricos, los únicos símbolos de serie de caracteres PICTURE permitidos son A, 9 y X, con al menos una X. Para campos de fecha numéricos, los únicos símbolos de serie de caracteres PICTURE permitidos son 9 y S.

Las cláusulas siguientes no están permitidas para un elemento de datos definido con DATE FORMAT:

- BLANCO CUANDO CERO
- JUSTIFICADO
- Frase SEPARADAS CHARACTER de la cláusula SIGN

La cláusula EXTERNAL no está permitida para un campo de fecha con ventana o un elemento de grupo que contiene un elemento subordinado de campo de fecha con ventana.

Se aplican algunas restricciones al combinar las cláusulas siguientes con DATE FORMAT:

- "cláusula REDEFINES" en la página 211
- "cláusula VALUE" en la página 232

Agrupar elementos que son campos de fecha

Si un elemento de grupo se define con una cláusula DATE FORMAT, se aplican ciertas restricciones.

Las restricciones son:

- Todos los elementos elementales del grupo deben ser USAGE DISPLAY.
- La longitud del elemento de grupo debe ser el mismo número de caracteres que el *patrón-fecha* en la cláusula DATE FORMAT.
- Si el grupo consta únicamente de un campo de fecha con USAGE DISPLAY, y tanto el grupo como el único elemento subordinado tienen cláusulas DATE FORMAT, las cláusulas DATE FORMAT deben ser idénticas.
- Si el elemento de grupo contiene elementos subordinados que subdividen el grupo, se aplican las restricciones siguientes:
  - Si un elemento subordinado con nombre (no FILLER) consta exactamente de la parte de año del campo de fecha de elemento de grupo y tiene una cláusula DATE FORMAT, la cláusula DATE FORMAT debe ser YY o YYYY con el mismo número de caracteres de año que el elemento de grupo.
  - Si el elemento de grupo es una fecha gregoriana con una cláusula DATE FORMAT de YYXXXX, YYYYXXXX, XXXXYY o XXXXYYYY, y un elemento de datos de fecha subordinado con nombre consta de la parte de año y mes de la fecha gregoriana, su cláusula DATE FORMAT debe ser YYXX, YYYYYYXX, XXYY o XXYYYY, respectivamente (o, para un formato de fecha de grupo de YYYYXXXX, un formato de fecha subordinado de YYXX tal como se describe a continuación).
  - Un campo de fecha con ventana puede estar subordinado a un elemento de grupo de campos de fecha expandida si el elemento subordinado empieza dos caracteres después del elemento de grupo, ninguna de las fechas está en el último formato de año, y el formato de fecha del elemento subordinado no tiene Xs o tiene el mismo número de Xs después de los Ys que el elemento de grupo, o es YYXX bajo un formato de fecha de grupo de YYYYXXXX.

– Los únicos elementos subordinados que pueden tener una cláusula DATE FORMAT son los que definen la parte del año del elemento de grupo, la parte con ventana de un elemento de grupo de campos de fecha expandido o la parte del año y mes de un elemento de grupo de fechas gregoriano, tal como se describe en las restricciones anteriores.

El ejemplo siguiente define un elemento de grupo válido:

```
01  YYMMDD     DATE FORMAT YYXXXX.
    02  YYMM   DATE FORMAT YYXX.
        03  YY DATE FORMAT YY PICTURE 99.
        03                    PICTURE 99.
    02  DD                    PICTURE 99.
```

Elementos de idioma que tratan los campos de fecha como no fechas

Si los campos de fecha se utilizan en los siguientes elementos de lenguaje, se tratan como no fechas. Es decir, se ignora DATE FORMAT y el contenido del elemento de datos de fecha se utiliza sin experimentar una expansión automática.

- En el párrafo de control de archivos de división de entorno:
  - SELECT ... ASSIGN USING nombre-datos
  - SELECT ... PASSWORD IS nombre-datos
  - SELECT ... FILE STATUS IS nombre-datos
- En entradas de división de datos:
  - LABEL RECORD IS nombre-datos
  - LABEL RECORDS ARE nombre-datos
  - LINAGE IS nombre-datos PIE nombre-datos TOP nombre-datos BOTTOM nombre-datos
- En condiciones de clase
- En condiciones de firma
- En sentencias DISPLAY

Elementos de lenguaje que no aceptan campos de fecha con ventanas como argumentos

Existen restricciones sobre el uso de campos de fecha con ventanas como argumentos.

Los campos de fecha con ventana no se pueden utilizar como:

- Un nombre de datos en los siguientes formatos del párrafo de control de archivos de división de entorno:
  - SELECT ... LA CLAVE DE REGISTRO ES
  - SELECT ... LA CLAVE DE REGISTRO ALTERNATIVA ES
  - SELECT ... LA CLAVE RELATIVA ES
- Un nombre de datos en la cláusula RECORD IS VARIABLE DEPENDING ON de una entrada de descripción de archivo de división de datos (FD) o descripción de clasificación (SD)
- El objeto de una cláusula OCCURS DEPENDING ON de una entrada de definición de datos de división de datos
- La clave de una frase ASCENDING KEY o DESCENDING KEY de una cláusula OCCURS de una entrada de definición de datos de división de datos
- Cualquier nombre de datos o identificador en las sentencias siguientes:
  - CANCEL
  - Ir a ... DEPENDIENDO DE

  - INSPECT
  - SET
  - SORT
  - Serie
  - UNSTRING
- En la sentencia CALL, como identificador que contiene el nombre de programa
- Identificadores en las frases TIMES y VARYING de la sentencia PERFORM (los campos de fecha con ventana *están* permitidos en las condiciones PERFORM)
- Un identificador en la frase VARYING de una sentencia SEARCH de serie (format-1) o cualquier identificador en una sentencia SEARCH binaria (format-2) (los campos de fecha con ventana *están* permitidos en las condiciones SEARCH)
- Un identificador en la frase AVANZANDO de la sentencia WRITE
- Argumentos para las funciones intrínsecas, excepto la función intrínseca UNDATE

Los campos de fecha con ventana no se pueden utilizar como claves ascendentes o descendentes en sentencias MERGE o SORT.

Elementos de idioma que no aceptan campos de fecha como argumentos

Existen restricciones sobre el uso de campos de fecha con ventanas y campos de fecha ampliados.

No se pueden utilizar campos de fecha con ventanas ni campos de fecha expandidos:

- En la sentencia DIVIDE, excepto como identificador en la cláusula CEDER o RESTO
- En la sentencia MULTIPLY, excepto como identificador en la cláusula CEDER

(los campos de fecha no se pueden utilizar como operandos en división o multiplicación.)

cláusula EXTERNAL

La cláusula EXTERNAL especifica que el almacenamiento asociado a un elemento de datos está asociado a la unidad de ejecución en lugar de a cualquier programa concreto dentro de la unidad de ejecución.

Cualquier programa de la unidad de ejecución que describe el elemento de datos puede hacer referencia a un elemento de datos externo. Las referencias a un elemento de datos externo de distintos programas que utilizan descripciones separadas del elemento de datos siempre están en el mismo elemento de datos. En una unidad de ejecución, sólo hay un representante de un elemento de datos externo.

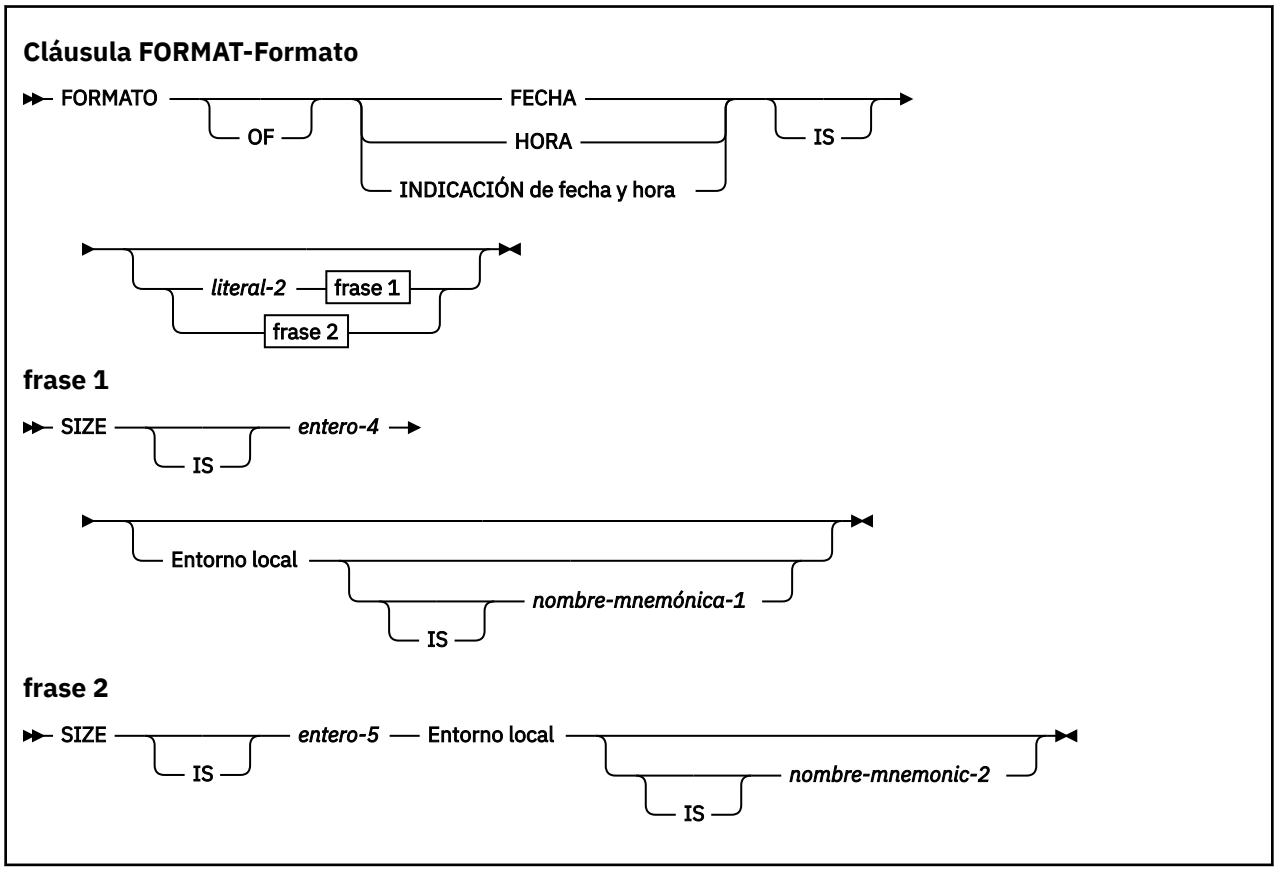
La cláusula EXTERNAL sólo se puede especificar en entradas de descripción de datos cuyo número de nivel sea 01. Sólo se puede especificar en entradas de descripción de datos que estén en la SECCIÓN WORKING-STORAGE de un programa. No se puede especificar en las entradas de descripción de datos LINKAGE SECTION, LOCAL-STORAGE SECTIONo FILE SECTION. Cualquier elemento de datos descrito por una entrada de descripción de datos subordinada a una entrada que describe un registro externo también alcanza el atributo externo. Los índices de un registro de datos externo no poseen el atributo externo.

Los datos contenidos en el registro especificado por la cláusula de nombre de datos son externos y pueden ser accedidos y procesados por cualquier programa de la unidad de ejecución que los describe y, opcionalmente, los redefine. Estos datos están sujetos a las reglas siguientes:

- Si dos o más programas dentro de una unidad de ejecución describen el mismo registro de datos externo, cada nombre de registro de las entradas de descripción de registro asociadas debe ser el mismo, y los registros deben definir el mismo número de bytes. Sin embargo, un programa que describe un registro externo puede contener una entrada de descripción de datos que incluya la cláusula REDEFINES que redefine el registro externo completo, y esta redefinición completa no tiene que producirse de forma idéntica en otros programas de la unidad de ejecución.
- El uso de la cláusula EXTERNAL no implica que el nombre de datos asociado sea un nombre global.

cláusula FORMAT

La cláusula FORMAT especifica las características generales y los requisitos de edición de un elemento elemental de fecha, hora o indicación de fecha y hora.



La cláusula FORMAT debe especificarse para cada elemento de fecha, hora o indicación de fecha y hora elemental, excepto el asunto de una cláusula RENAMES.

Si no se especifica la frase SIZE para un elemento de indicación de fecha y hora, el tamaño predeterminado es 26. Si se especifica, debe tener un valor de 19 o un valor entre 21 y 32.

literal-2 y la frase LOCALE no se pueden especificar para un elemento de indicación de fecha y hora. Una indicación de fecha y hora tiene un formato fijo, que depende del tamaño del elemento de indicación de fecha y hora.

- Cuando no se especifica la frase SIZE, el formato es equivalente a un valor de *literal-2* de "@Y-%m-%d-%H.%M.%S.@Sm".
- Cuando se especifica la frase SIZE con un valor de 19, el formato es equivalente a un valor *literal-2* de "@Y-%m-%d-%H.% M.% S".
- Cuando la frase SIZE se especifica como un valor entre 21 y 32, el formato es equivalente a un valor *literal-2* de "@Y-%m-%d-%H.% M.% S." seguido de las fracciones de segundo en la indicación de fecha y hora. Por ejemplo, una indicación de fecha y hora con un tamaño de 25 podría tener el valor "2014-01-23-01.02.03.12345".

Si no se especifica *literal-2* o la frase LOCALE para un elemento de fecha u hora, el formato del elemento se determina a partir de la cláusula SPECIAL-NAMES FORMAT.

No se puede hacer referencia a un elemento de datos de la clase fecha-hora.

Cuando se especifica la cláusula FORMAT, no se pueden especificar las cláusulas siguientes:

- Cláusula PICTURE.
- Cláusula SIGN.

- Cláusula BLANK WHEN ZERO.
- Cláusula JUSTIFIC.
- Cláusula LIKE. Sin embargo, se puede utilizar una cláusula LIKE para definir el FORMAT de un elemento de datos. No puede cambiar el tamaño de un elemento de fecha, hora o indicación de fecha y hora con una cláusula LIKE. Cuando una cláusula LIKE hace referencia a un elemento de fecha, hora o indicación de fecha y hora, se genera un comentario con la información de cláusula FORMAT adecuada que se hereda
- Tipocláusula.

Se aplican las siguientes normas generales:

- Un nombre de condición se puede asociar con un elemento de fecha-hora. La cláusula VALUE del nombre de condición puede especificarse con una frase THRU.
- Una cláusula SYNCHRONIZED se trata como documentación.
- Las cláusulas OCCURS, REDEFINES y RENAMES se pueden asociar con elementos de fecha, hora o indicación de fecha y hora.
- Si se especifica una cláusula LIKE, no se puede especificar una cláusula FORMAT.
- Cualquier cláusula VALUE asociada debe especificar un literal no numérico. El literal se trata exactamente como se ha especificado; no se realiza ningún formateo.

literal-2
Especifica el formato de un elemento de fecha u hora. *literal-2* debe ser un literal no numérico, de al menos 2 caracteres de longitud. El contenido de *literal-2* se compone de separadores y especificadores de conversión. Para obtener una lista de especificadores de conversión válidos, consulte la tabla *Especificadores de conversión que se pueden utilizar en literal-8* . Para obtener más información sobre el contenido de *literal-2*, consulte la descripción de la cláusula FORMAT utilizada en el párrafo SPECIAL-NAMES en “cláusula FORMAT” en la página 104.

frase SIZE

En esta sección se describen los parámetros que se especifican para esta frase SIZE.

Para obtener una descripción más detallada de la frase SIZE, consulte “frase SIZE” en la página 106.

integer-3*, *integer-4
integer-3 y *integer-4* determinan el tamaño del elemento de fecha u hora en número de dígitos. Se debe especificar *integer-3* o *integer-4* si el tamaño del elemento de fecha u hora no se puede determinar en el momento de la compilación. Para un elemento de fecha u hora, los valores de *integer-3* y *integer-4* deben ser iguales o mayores que 4.

mnemonic-name-1*, *mnemonic-name-2
Para obtener más información sobre *mnemonic-name-1* o *mnemonic-name-2*, consulte la descripción en “frase LOCALE” en la página 179 y “frase LOCALE” en la página 107.

USAGE para un elemento de fecha y hora de clase

Si no se especifica ninguna cláusula USAGE para un elemento de la clase fecha-hora, se asume USAGE DISPLAY.

Se puede especificar explícitamente un USAGE de DISPLAY o PACKED-DECIMAL (COMP-3) para un elemento de fecha-hora. Se puede especificar una USAGE de PACKED-DECIMAL para un elemento de la clase fecha-hora, si *literal-2* contiene sólo especificadores de conversión, y estos especificadores darán como resultado dígitos numéricos.

Similitudes entre la cláusula FORMAT y la cláusula PICTURE

Una cláusula FORMAT define una cláusula PICTURE implícita.

Aunque no hay ninguna serie de caracteres PICTURE que pueda describir fácilmente un elemento de fecha u hora, para algunos formatos, existe una definición aproximada. Por ejemplo, un elemento de

fecha con un FORMAT '%y, %m,%d' es similar al PICTURE 99/99/99, donde el símbolo '/' PICTURE se sustituye por un', '.

frase LOCALE

La frase LOCALE especifica el formato culturalmente adecuado del elemento de fecha, hora o indicación de fecha y hora.

Cuando se especifica la frase LOCALE, sin *literal-2*, el formato y el separador utilizados para los elementos de fecha y hora se basan completamente en el entorno local.

Cuando la frase LOCALE se especifica con *literal-2*, *literal-2* determina el formato del elemento, pero las especificaciones de conversión se sustituyen por elementos basados en el entorno local.

mnemonic-name-1*, *mnemonic-name-2
Si se especifica un nombre mnemotécnico, el entorno local utilizado para el elemento de fecha u hora es uno asociado con el nombre mnemotécnico en la cláusula LOCALE del párrafo SPECIAL-NAMES. Si no se especifica un nombre nemotécnico, se utiliza el entorno local actual. Para obtener más información sobre cómo determinar el entorno local actual, consulte *CEELOCT: get current local date or time* en *COBOL for Linux en x86 Guía de programación*.

cláusula GLOBAL

La cláusula GLOBAL especifica que un nombre de datos o un nombre de constante está disponible para cada programa contenido en el programa que lo define , siempre que el programa contenido no tenga por sí mismo una definición para ese nombre. Todos los nombres de datos o nombres de constante subordinados o nombres de condición o índices asociados a un nombre global son nombres globales.

Un nombre de datos o un nombre de constante es global si la cláusula GLOBAL se especifica en la entrada de descripción de datos por la que el nombre de datos o el nombre de constante está definido o en otra entrada a la que está subordinada dicha entrada de descripción de datos. La cláusula GLOBAL puede especificarse en WORKING-STORAGE SECTION, FILE SECTION, LINKAGE SECTION y LOCAL-STORAGE SECTION, pero sólo en las entradas de descripción de datos cuyo número de nivel sea 01.

En la misma DATA DIVISION, las entradas de descripción de datos para dos elementos de datos cualesquiera para los que se ha especificado el mismo nombre de datos o nombre de constante no deben incluir la cláusula GLOBAL.

Una sentencia de un programa contenida directa o indirectamente en un programa que describe un nombre global puede hacer referencia a ese nombre sin volver a describirlo.

Si la cláusula TYPEDEF se especifica con la cláusula GLOBAL, el ámbito de la cláusula GLOBAL se aplica al nombre-tipo y a cualquier elemento de datos subordinado al nombre-tipo. El atributo global no lo adquiere un elemento de datos definido utilizando un nombre-tipo global dentro de una cláusula TYPE.

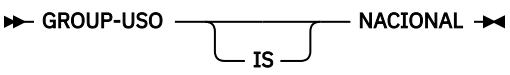
Dos programas en una unidad de ejecución pueden hacer referencia a datos comunes en las circunstancias siguientes:

- Se puede hacer referencia al contenido de datos de un registro de datos externo desde cualquier programa que describa el registro de datos como externo.
- Si un programa está contenido en otro programa, ambos programas pueden hacer referencia a datos que posean el atributo global en el programa contenedor o en cualquier programa que contenga directa o indirectamente el programa contenedor.

cláusula GROUP-USAGE

Una cláusula GROUP-USAGE con la frase NATIONAL especifica que el elemento de grupo definido por la entrada es un elemento de grupo nacional. Un elemento de grupo nacional contiene caracteres nacionales en todos los elementos de datos subordinados y elementos de grupo subordinados.

Formato



Cuando se especifica GROUP-USAGE NATIONAL:

- El asunto de la entrada es un elemento de grupo nacional. La clase y la categoría de un grupo nacional son nacionales.
- No debe especificarse una cláusula USAGE para el asunto de la entrada. Está implícita una cláusula USAGE NATIONAL.
- Una cláusula USAGE NATIONAL está implícita para los elementos de datos elementales subordinados que no se describen con una cláusula USAGE NATIONAL.
- Todos los elementos de datos elementales subordinados deben describirse explícita o implícitamente con USAGE NATIONAL.
- Los elementos de datos numéricos con signo deben describirse con la cláusula SIGN IS SEPARATE.
- Una cláusula GROUP-USAGE NATIONAL está implícita para cualquier elemento de grupo subordinado que no esté descrito con una cláusula GROUP-USAGE NATIONAL.
- Todos los elementos de grupo subordinados deben describirse explícita o implícitamente con una cláusula GROUP-USAGE NATIONAL.
- No debe especificarse la cláusula JUSTIFIC.

A menos que se indique lo contrario, un elemento de grupo nacional se procesa como si fuera un elemento de datos elemental de uso nacional, de clase y de categoría nacional, descrito con PICTURE N (*m*), donde *m* es la longitud del grupo en posiciones de caracteres nacionales.

Nota de uso: Cuando se utilizan grupos nacionales, el compilador puede garantizar el truncamiento y el relleno adecuados de elementos de grupo para sentencias como MOVE e INSPECT. Los grupos definidos sin una cláusula GROUP-USAGE NATIONAL son grupos alfanuméricos. El contenido de los grupos alfanuméricos, incluidos los caracteres nacionales, se trata como datos alfanuméricos, lo que puede dar lugar a un truncamiento no válido o a un mal manejo de los datos de caracteres nacionales.

La tabla siguiente resume los casos en los que un elemento de grupo nacional se procesa como elemento de grupo.

Tabla 15. ***Donde los elementos de grupo nacional se procesan como grupos***

Característica de idioma	**Tratamiento de los grupos nacionales**
Cualificación de nombre	El nombre de un elemento de grupo nacional puede utilizarse para calificar los nombres de elementos de datos elementales y elementos de grupo subordinados en el grupo nacional. Las normas de cualificación para un grupo nacional son las mismas que las normas de cualificación para un grupo alfanumérico.
cláusula RENAMES	Las reglas para un elemento de grupo nacional especificado en la frase THROUGH son las mismas que las reglas para un elemento de grupo alfanumérico especificado en la frase THROUGH. El resultado es un elemento de grupo alfanumérico.
Frase CORRESPONDIENTE	Un elemento de grupo nacional se procesa como un grupo de acuerdo con las reglas de la frase CORRESPONDIENTE. Los elementos de datos elementales dentro de un grupo nacional se procesan de la misma forma que si se definieran dentro de un grupo alfanumérico.

Tabla 15. ***Donde los elementos de grupo nacional se procesan como grupos*** *(continuación)*	
Característica de idioma	**Tratamiento de los grupos nacionales**
Sentencia INITIALIZE	Un elemento de grupo nacional se procesa como un grupo de acuerdo con las reglas de la sentencia INITIALIZE. Los elementos elementales dentro del grupo nacional se inicializan de la misma manera que si se definieran dentro de un grupo alfanumérico.
sentencia XML GENERATE	Un elemento de grupo nacional especificado en la frase FROM se procesa como un grupo de acuerdo con las reglas de la sentencia XML GENERATE. Los elementos elementales dentro del grupo nacional se procesan de la misma manera que lo harían si se definieran dentro de un grupo alfanumérico.

cláusula JUSTIFIC

La cláusula JUSTIFIC altera temporalmente las reglas de posicionamiento estándar para recibir artículos de categoría alfabética, alfanumérica, DBCS o nacional.

Formato

Puede especificar la cláusula JUSTIFIC sólo en el nivel elemental. JUST es una abreviatura de JUSTIFIC, y tiene el mismo significado.

No puede especificar la cláusula JUSTIFIC:

- Para elementos de datos de categoría numérica, editada numérica, editada alfanumérica o editada nacional
- Para elementos DBCS editados
- Para elementos de datos de índice
- Para elementos descritos como USAGE FUNCTION-POINTER, USAGE POINTER, o USAGE PROCEDURE-POINTER
- Para elementos de coma flotante externa o de coma flotante interna
- Para campos de fecha
- Con entradas level-66 (RENAMES) y level-88 (condition-name)
- Para elementos con la cláusula TYPE

Cuando se especifica la cláusula JUSTIFIC para un elemento de recepción, los datos se alinean en la posición de carácter más a la derecha en el elemento de recepción. Además:

- Si el elemento de envío es mayor que el elemento de recepción, las posiciones de carácter más a la izquierda se truncan.
- Si el elemento de envío es menor que el elemento de recepción, las posiciones de carácter no utilizadas de la izquierda se rellenan con espacios. Para un elemento DBCS, cada posición no utilizada se rellena con un espacio DBCS espacio; para un elemento descrito con el uso NATIONAL, cada puesto no utilizado se rellena con el espacio Unicode predeterminado (NX'2000'); de lo contrario, cada posición no utilizada se rellena con un espacio alfanumérico.

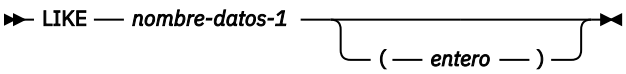
Si omite la cláusula JUSTIFIC, se siguen las reglas para la alineación estándar (consulte “Reglas de alineación” en la página 151).

La cláusula JUSTIFIC no afecta a los valores iniciales tal como los determina la cláusula VALUE.

Cláusula LIKE

La cláusula LIKE le permite definir las características PICTURE, USAGE, SIGN y FORMAT de un elemento de datos copiándolos de un elemento de datos definido anteriormente. También le permite hacer que la longitud del elemento de datos que define sea diferente de la longitud del elemento original.

Formato



data-name-1
Puede hacer referencia a un elemento elemental, un elemento de grupo, un nombre de índice o un nombre de tipo. El elemento al que hace referencia *data-name-1* se conoce como *objeto* de la cláusula LIKE.

Entero
Especifica la diferencia de longitud entre los elementos nuevos y los existentes.

Se puede firmar.

Si un espacio en blanco o un signo + precede al entero, el nuevo elemento es más largo. Si a-precede al entero, el nuevo elemento es más corto.

No puede utilizar la opción de entero para:

- Cambiar la longitud de un elemento editado
- Cambiar la longitud de un elemento de índice, puntero o puntero de procedimiento
- Cambiar el número de posiciones decimales en un elemento de datos
- Cambiar la longitud de un elemento de datos de coma flotante interno o externo
- Cambiar la longitud de un elemento de fecha, hora o indicación de fecha y hora

Nota: Un elemento cuyos atributos incluyen BLANK WHEN ZERO se trata como un elemento editado.

La cláusula LIKE hace que el nuevo elemento de datos herede características específicas del elemento de datos existente. Estas características son los atributos PICTURE, USAGE, SIGN, BLANK WHEN ZERO y FORMAT del elemento existente.

El compilador genera comentarios para identificar las características del nuevo elemento. Estos comentarios aparecen después de la sentencia que contiene la cláusula LIKE.

Tenga en cuenta que las características USAGE IS DISPLAY y SIGN IS TRAILING predeterminadas no se imprimen como comentarios.

Las características FORMAT que se pueden heredar incluyen:

- La categoría del artículo: fecha, hora o indicación de fecha y hora
- Un literal FORMAT
- Una frase SIZE y una frase LOCALE.

Para obtener más información sobre la cláusula FORMAT, consulte “cláusula FORMAT” en la página 177.

Comentarios generados en función de las características de USAGE heredadas

Las diferentes cláusulas USAGE que puede especificar para el elemento original dan como resultado un número limitado de comentarios.

Tabla 16. Comentarios generados en función de las características de USAGE heredadas	
Cláusula USAGE heredada	**Comentario generado**
DECIMAL EMPAQUETADO COMPUTATIONAL COMPUTATIONAL-3	* EL USO ES DECIMAL EMPAQUETADO
COMP-1 COMUTATIONAL-1	* USAGE IS COMPUTATIONAL-1
COMP-2 COMUTATIONAL-2	* USAGE IS COMPUTATIONAL-2
BINARIO COMP-4 COMPUTATIONAL-4	* EL USO ES BINARIO
COMP-5 COMPUTATIONAL-5	* USO COMP-5
ÍNDICE	*USO ES ÍNDICE
NACIONAL	*EL USO ES NACIONAL
VISUALIZAR	Este es el uso predeterminado, por lo que no se genera un comentario.
DISPLAY-1	* USAGE IS DISPLAY-1
PUNTERO	* EL USO ES PUNTERO
PROCEDIMIENTO-PUNTERO	* USAGE IS PROCEDURE-POINTER

Las características del elemento de datos que defina utilizando la cláusula LIKE se muestran en el listado del programa compilado.

Reglas y restricciones

Puede utilizar la cláusula LIKE en los números de nivel 01 a 49 y en el número de nivel 77.

Si especifica entradas de nombre de datos o FILLER, puede colocar la cláusula LIKE en cualquier posición después de ellas. De lo contrario, puede colocarlo en cualquier posición después del número de nivel.

Puede especificar una o más cláusulas antes o después de la cláusula LIKE:

- JUSTIFICADO
- SINCRONIZADO
- BLANCO CUANDO CERO
- Valor
- APARICIONES

Tenga en cuenta que puede especificar BLANK WHEN ZERO sólo si no se ha heredado anteriormente.

No puede utilizar la cláusula LIKE con las cláusulas siguientes:

- REDEFINE
- SIGNO
- Uso

- IMAGEN
- Formato
- Tipo
- TYPEDEF

Si especifica alguna cláusula heredada en la cláusula LIKE, se producirá un error de duplicación.

Para elementos numéricos, el número total de caracteres numéricos en el nuevo elemento no puede ser cero. Pero si el elemento contiene decimales, el número de caracteres en la parte entera puede ser cero.

Si una cláusula PICTURE especifica una mezcla de caracteres alfabéticos, numéricos o alfanuméricos, y la cláusula LIKE tiene una modificación de longitud, la nueva cláusula PICTURE especifica caracteres alfanuméricos.

No puede utilizar la cláusula LIKE para definir un elemento que esté subordinado al elemento que nombre en la cláusula.

Elobjeto de una cláusula LIKE no puede contener la cláusula TYPE en su descripción de datos. Si el objeto de una cláusula LIKE es un elemento de grupo, ninguno de los elementos subordinados a este grupo se puede definir utilizando la cláusula TYPE. Si el objeto de una cláusula LIKE está subordinado a un elemento de grupo (level-01) y un elemento que está subordinado al elemento de grupo level-01 contiene una cláusula TYPE, el nombre de tipo al que se hace referencia en la cláusula TYPE debe estar totalmente definido en el punto de DATA DIVISION cuando se utiliza la cláusula LIKE.

Ejemplos de codificación

Para crear el elemento de datos DEPTH con los mismos atributos que el elemento de datos HEIGHT, simplemente escriba:

```
DEPTH LIKE HEIGHT
```

Para crear el elemento de datos PROVINCIA con los mismos atributos que el elemento de datos STATE, excepto un byte más largo, escriba:

```
PROVINCE LIKE STATE (+1)
```

cláusula OCCURS

Los elementos de lenguaje DATA DIVISION utilizados para el manejo de tablas son la cláusula OCCURS y la frase INDEXED BY.

Para obtener la descripción de la frase INDEXED BY, consulte “frase INDEXED BY” en la página 187.

La cláusula OCCURS especifica tablas a cuyos elementos se puede hacer referencia mediante indexación o suscripción. También elimina la necesidad de entradas separadas para elementos de datos repetidos.

Los formatos de la cláusula OCCURS incluyen tablas de longitud fija y tablas de longitud variable.

El *asunto* de una cláusula OCCURS es el nombre de datos del elemento de datos que contiene la cláusula OCCURS. A excepción de la propia cláusula OCCURS, las cláusulas de descripción de datos utilizadas con el asunto se aplican a cada aparición del elemento descrito.

Siempre que se haga referencia al asunto de una cláusula OCCURS o a cualquier elemento de datos subordinado al mismo, debe estar subindexado o indexado, con las excepciones siguientes:

- Cuando el asunto de la cláusula OCCURS se utiliza como asunto de una sentencia SEARCH.
- Cuando el asunto de la cláusula OCCURS se utiliza como asunto de una sentencia SORT de formato 2.
- Cuando el sujeto o un elemento de datos subordinado es el objeto de la frase ASCENDING/ DESCENDING KEY.
- Cuando el elemento de datos subordinado es el objeto de la cláusula REDEFINES.

- Cuando el elemento de datos subordinado se utiliza como sujeto de un registro especial LENGTH OF. Para obtener más información, consulte "LENGTH OF" en la página 21.

Cuando se subindexa o se indexa, el sujeto hace referencia a una aparición dentro de la tabla, a menos que se utilice el subíndice ALL en una función intrínseca.

La cláusula OCCURS no puede especificarse en una entrada de descripción de datos que:

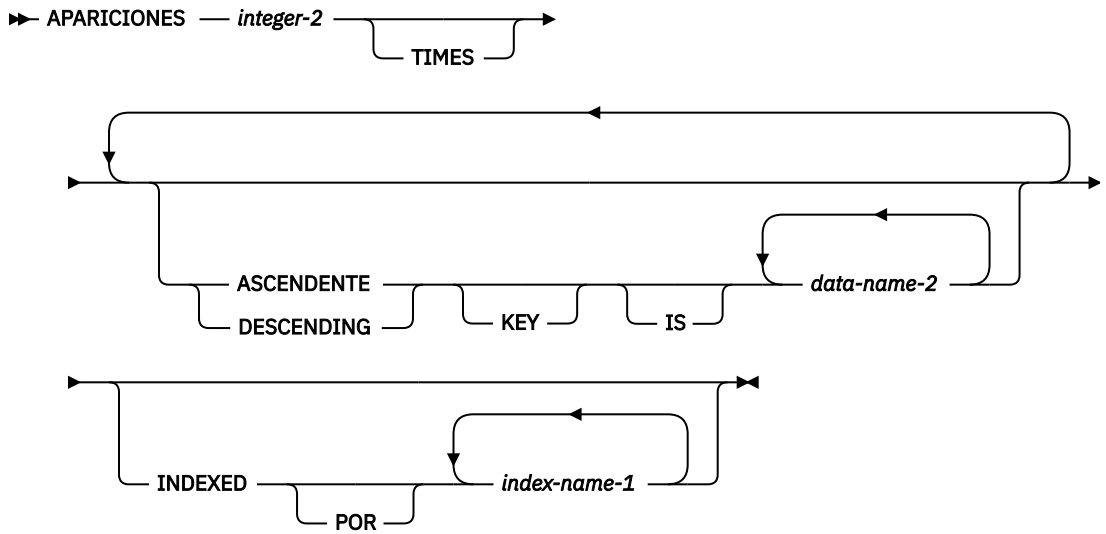
- Tiene un número de nivel de 01, 66, 77 o 88.
- Describe un elemento de datos redefinido. (Sin embargo, un elemento redefinido puede estar subordinado a un elemento que contiene una cláusula OCCURS.) Consulte "cláusula REDEFINES" en la página 211.

Tablas de longitud fija

Las tablas de longitud fija se especifican utilizando la cláusula OCCURS.

Puesto que se permiten siete subíndices o índices, se permiten seis niveles anidados y un nivel más externo de la cláusula format-1 OCCURS. La cláusula format-1 OCCURS puede especificarse como subordinada a la cláusula OCCURS DEPENDING ON. De esta forma, se puede especificar una tabla de hasta siete dimensiones.

Formato 1: tablas de longitud fija



integer-2
El número exacto de apariciones. *integer-2* debe ser mayor que cero.

Frases ASCENDING KEY y DESCENDING KEY

Los datos se organizan en orden ascendente o descendente, en función de la palabra clave especificada, según los valores contenidos en *data-name-2*. Los nombres de datos se listan en su orden descendente de significación.

El orden viene determinado por las reglas para la comparación de operandos (consulte "Condiciones de relación" en la página 255). Los elementos de datos ASCENDING KEY y DESCENDING KEY se utilizan en las cláusulas OCCURS, las sentencias SEARCH ALL para una búsqueda binaria del elemento de tabla y las sentencias SORT de formato 2. Como alternativa, las claves se pueden especificar con las sentencias SORT de formato 2.

data-name-2
Debe ser el nombre de la entrada de asunto o el nombre de una entrada subordinada a la entrada de asunto. *data-name-2* no puede ser un campo de fecha con ventana. *data-name-2* puede calificarse.

Si *data-name-2* nombra la entrada de asunto, esa entrada entera se convierte en CLAVE ASCENDENTE o CLAVE DESCENDENTE y es la única clave que se puede especificar para este elemento de tabla.

Si *data-name-2* no da nombre a la entrada del asunto, *data-name-2*:

- Debe estar subordinado al asunto de la propia entrada de tabla
- *No* debe estar subordinado a, o seguir, cualquier otra entrada que contenga una cláusula OCCURS
- No debe contener una cláusula OCCURS

data-name-2 no debe tener elementos subordinados que contengan cláusulas OCCURS DEPENDING ON.

Cuando se especifica la frase ASCENDING KEY o DESCENDING KEY, se aplican las reglas siguientes:

- Las claves deben listarse en orden decreciente de significación.
- El número total de claves para un elemento de tabla determinado no debe exceder de 12.
- Los datos de la tabla deben estar ordenados en orden ascendente o descendente de acuerdo con el orden de clasificación en uso.
- La clave debe describirse con uno de los usos siguientes:
  - BINARIO
  - VISUALIZAR
  - DISPLAY-1
  - NACIONAL
  - DECIMAL EMPAQUETADO
  - COMPUTATIONAL
  - COMPUTATIONAL-1
  - COMPUTATIONAL-2
  - COMPUTATIONAL-3
  - COMPUTATIONAL-4
  - COMPUTATIONAL-5
- Una clave descrita con el uso NACIONAL puede tener una de las categorías siguientes: nacional, nacional editada, numérica editada, numérica o de coma flotante externa.
- La suma de las longitudes de todas las claves asociadas con un elemento de tabla no debe exceder de 256.
- Si se especifica una clave sin calificadores y no es un nombre exclusivo, la clave se calificará implícitamente con el sujeto de la cláusula OCCURS y todos los calificadores del sujeto de la cláusula OCCURS.

El ejemplo siguiente ilustra la especificación de elementos de datos ASCENDING KEY:

```
WORKING-STORAGE SECTION.
01  TABLE-RECORD.
    05  EMPLOYEE-TABLE OCCURS 100 TIMES
        ASCENDING KEY IS WAGE-RATE EMPLOYEE-NO
        INDEXED BY A, B.
      10  EMPLOYEE-NAME                    PIC X(20).
      10  EMPLOYEE-NO                      PIC 9(6).
      10  WAGE-RATE                        PIC 9999V99.
      10  WEEK-RECORD OCCURS 52 TIMES
          ASCENDING KEY IS WEEK-NO INDEXED BY C.
        15  WEEK-NO                        PIC 99.
        15  AUTHORIZED-ABSENCES            PIC  9.
        15  UNAUTHORIZED-ABSENCES          PIC  9.
        15  LATE-ARRIVALS                  PIC  9.
```

Las claves para EMPLOYEE-TABLE están subordinadas a dicha entrada, y la clave para WEEK-RECORD está subordinada a dicha entrada subordinada.

En el ejemplo anterior, los registros de EMPLOYEE-TABLE deben estar ordenados en orden ascendente de WAGE-RATE y en orden ascendente de EMPLOYEE-NO dentro de WAGE-RATE. Los registros en WEEK-RECORD deben estar ordenados en orden ascendente de WEEK-NO. Si no lo son, los resultados de cualquier sentencia SEARCH ALL son imprevisibles.

frase INDEXED BY

La frase INDEXED BY especifica los índices que se pueden utilizar con una tabla. Se puede hacer referencia a una tabla sin una frase INDEXED BY mediante la indexación utilizando un nombre de índice asociado con otra tabla.

Para obtener más información sobre cómo utilizar la indexación, consulte "Suscripción utilizando nombres de índice (indexación)" en la página 65.

Los índices normalmente se asignan en memoria estática asociada con el programa que contiene la tabla. Por lo tanto, los índices se encuentran en el último estado utilizado cuando se vuelve a especificar un programa. Sin embargo, en los casos siguientes, los índices se asignan por invocación. Por lo tanto, debe establecer el valor del índice en cada entrada para índices en tablas en las secciones siguientes:

- LOCAL-STORAGE SECTION
- La SECCIÓN DE ENLACE de:
  - Programas compilados con la cláusula RECURSIVE

Los índices especificados en un registro de datos externo no poseen el atributo externo.

index-name-1
Cada nombre de índice especifica un índice que debe crear el compilador para que lo utilice el programa. Estos nombres de índice *no* son nombres de datos y no se identifican en ningún otro lugar del programa COBOL; en su lugar, se pueden considerar como registros especiales privados sólo para el uso de este programa objeto. No son datos y no forman parte de ninguna jerarquía de datos.

No es necesario que los nombres de índice no referenciados se definan de forma exclusiva.

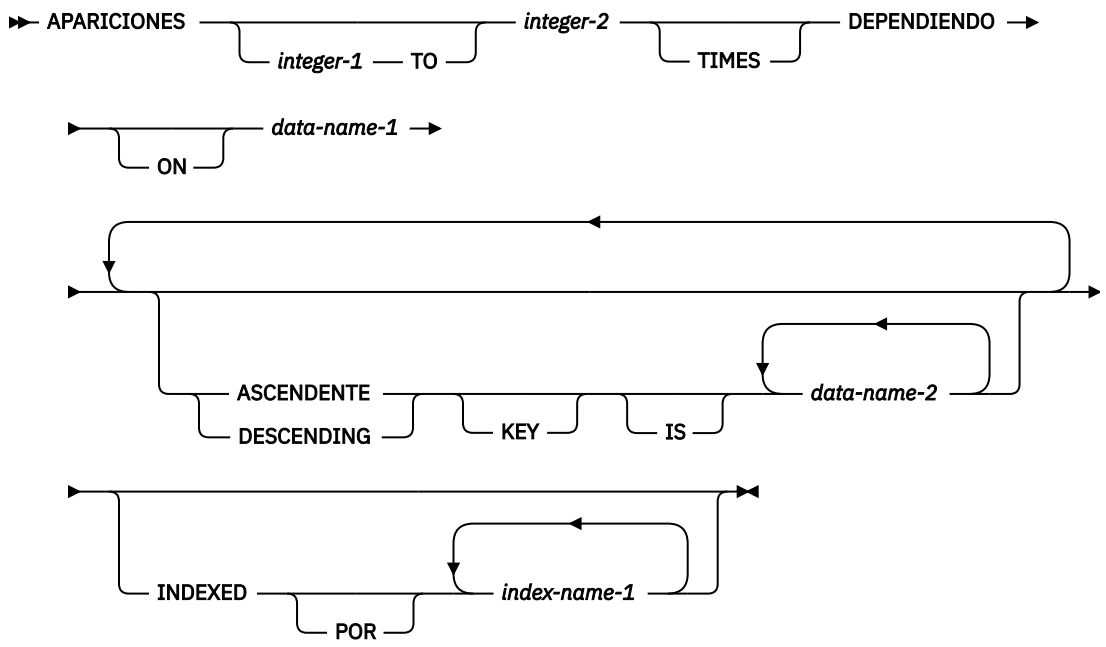
En una entrada de tabla, se pueden especificar hasta 12 nombres de índice.

Si un elemento de datos que posee el atributo global incluye una tabla a la que se accede con un índice, dicho índice también posee el atributo global. Por lo tanto, el ámbito de un nombre de índice es el mismo que el del nombre de datos que nombra la tabla en la que se define el índice.

Tablas de longitud variable

Puede especificar tablas de longitud variable utilizando la cláusula OCCURS DEPENDING ON.

Formato 2: tablas de longitud variable



integer-1
El número mínimo de apariciones.

El valor de *integer-1* debe ser mayor o igual que cero, y también debe ser menor que el valor de *integer-2*.

Si se omite *integer-1* , se asume un valor de 1 y también se debe omitir la palabra clave TO.

integer-2
El número máximo de apariciones.

integer-2 debe ser mayor que *integer-1*.

La *longitud* del elemento de asunto es fija. Sólo el *número de repeticiones* del elemento de asunto es variable.

Cláusula OCCURS DEPENDING ON

La cláusula OCCURS DEPENDING ON especifica tablas de longitud variable.

data-name-1
Identifica el *objeto* de la cláusula OCCURS DEPENDING ON; es decir, el elemento de datos cuyo valor actual representa el número actual de apariciones del elemento *subject* . El contenido de los elementos cuyos números de aparición exceden el valor del objeto no está definido.

El objeto de la cláusula OCCURS DEPENDING ON (*data-name-1*) debe describir un elemento de datos entero. El objeto no puede ser un campo de fecha con ventana.

El objeto de la cláusula OCCURS DEPENDING ON no debe ocupar ninguna posición de almacenamiento dentro del rango de la tabla (es decir, cualquier posición de almacenamiento desde la primera posición de carácter de la tabla hasta la última posición de carácter de la tabla).

El objeto de la cláusula OCCURS DEPENDING ON no se puede localizar de forma variable; el objeto no puede ir a continuación de un elemento que contiene una cláusula OCCURS DEPENDING ON.

Si se especifica la cláusula OCCURS en una entrada de descripción de datos incluida en una entrada de descripción de registro que contiene la cláusula EXTERNAL, *data-name-1*, si se especifica, debe hacer referencia a un elemento de datos que posea el atributo externo. *data-name-1* debe describirse en la misma DATA DIVISION que el asunto de la entrada.

Si se especifica la cláusula OCCURS en una entrada de descripción de datos subordinada a una que contiene la cláusula GLOBAL, *data-name-1*, si se especifica, debe ser un nombre global. *data-name-1* debe describirse en la misma DATA DIVISION que el asunto de la entrada.

Todos los nombres de datos utilizados en la cláusula OCCURS pueden calificarse; no pueden tener subíndices ni indexarse.

En el momento en que se hace referencia al elemento de grupo, o a cualquier elemento de datos que contenga un elemento subordinado OCCURS DEPENDING ON o que no esté subordinado al elemento OCCURS DEPENDING ON, el valor del objeto de la cláusula OCCURS DEPENDING ON debe estar dentro del rango de *integer-1* a *integer-2* .

El comportamiento no está definido si el valor del objeto está fuera del rango de *integer-1* a *integer-2*.

Cuando se hace referencia a un elemento de grupo que contiene un elemento subordinado OCCURS DEPENDING ON, la parte del área de tabla utilizada en la operación se determina de la forma siguiente:

- Si el objeto está fuera del grupo, sólo se utiliza la parte del área de tabla especificada por el objeto al principio de la operación.
- Si el objeto se incluye en el mismo grupo y se hace referencia al elemento de datos de grupo como un elemento de envío, sólo se utiliza en la operación la parte del área de tabla especificada por el valor del objeto al inicio de la operación.
- Si el objeto se incluye en el mismo grupo y se hace referencia al elemento de datos de grupo como elemento receptor, se utiliza la longitud máxima del elemento de grupo en la operación.

Las sentencias siguientes se ven afectadas por la regla de longitud máxima:

- ACCEPT *identificador* (formato 1 y 2)
- CALL ... *Identificador* USING BY REFERENCE
- MOVE ... TO *identificador*
- LEA ... INTO *identificador*
- RELEASE *identificador* FROM ...
- DEVOLVER ... INTO *identificador*
- REWRITE *identificador* FROM ...
- CADENA ... INTO *identificador*
- UNSTRING ... INTO *identificador* DELIMITER IN *identificador*
- WRITE *identificador* FROM ...

Si un elemento de grupo de longitud variable no va seguido de un elemento no subordinado, se utiliza la longitud máxima del grupo cuando aparece como identificador en CALL ... *Identificador*USING BY REFERENCE. Por lo tanto, no es necesario establecer el objeto de la cláusula OCCURS DEPENDING ON a menos que el grupo esté ubicado de forma variable.

Si el elemento de grupo va seguido de un elemento no subordinado, se utiliza la longitud real, en lugar de la longitud máxima. En el momento en que se hace referencia al asunto de entrada, o se hace referencia a cualquier elemento de datos subordinado o superior al asunto de entrada, el objeto de la cláusula OCCURS DEPENDING ON debe estar dentro del rango de *integer-1* a *integer-2*.

Determinados usos de la cláusula OCCURS DEPENDING ON dan como resultado elementos *complex OCCURS DEPENDING ON* (ODO). Los siguientes elementos constituyen elementos ODO complejos:

- Un elemento de datos descrito con una cláusula OCCURS DEPENDING ON seguida de un elemento de datos elemental no subordinado, descrito con o sin una cláusula OCCURS
- Un elemento de datos descrito con una cláusula OCCURS DEPENDING ON que va seguido de un elemento de grupo no subordinado
- Un elemento de grupo que contiene uno o más elementos subordinados descritos con una cláusula OCCURS DEPENDING ON

- Un elemento de datos descrito con una cláusula OCCURS o una cláusula OCCURS DEPENDING ON que contiene un elemento de datos subordinado descrito con una cláusula OCCURS DEPENDING ON (una tabla que contiene elementos de longitud variable)
- Un nombre de índice asociado con una tabla que contiene elementos de longitud variable

El objeto de una cláusula OCCURS DEPENDING ON no puede ser un elemento no subordinado que sigue a un elemento ODO complejo.

Cualquier elemento no subordinado que sigue a un elemento descrito con una cláusula OCCURS DEPENDING ON es un *elemento de ubicación variable*. Es decir, su ubicación se ve afectada por el valor del objeto OCCURS DEPENDING ON.

Cuando se utiliza la redefinición implícita en una entrada de Descripción de archivo (FD), los elementos de nivel subordinado pueden contener cláusulas OCCURS DEPENDING ON.

La frase INDEXED BY puede especificarse para una tabla que tiene un elemento subordinado que contiene una cláusula OCCURS DEPENDING ON.

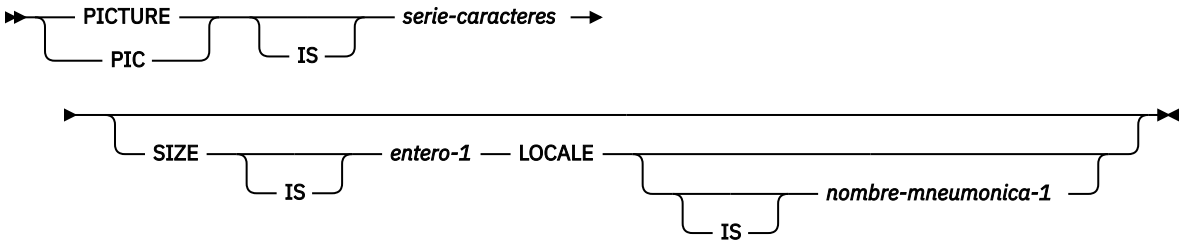
Para obtener más información sobre OCCURS DEPENDING ON complejo, consulte *SE PRODUCE COMPLEJO EN FUNCIÓN DE* en la publicación *COBOL for Linux en x86 Guía de programación*.

La frase ASCENDING KEY, la frase DESCENDING KEY y la cláusula INDEXED BY se describen en "Tablas de longitud fija" en la página 185.

cláusula PICTURE

La cláusula PICTURE especifica las características generales y los requisitos de edición de un elemento elemental.

Formato



PICTURE o PIC

La cláusula PICTURE debe especificarse para cada elemento elemental excepto los siguientes:

- Elementos de datos de índice
- El asunto de la cláusula cláusula LIKE, RENAMES o TYPE
- Elementos descritos con USAGE POINTER, USAGE FUNCTION-POINTER, o USAGE PROCEDURE-POINTER
- Elementos de datos de coma flotante internos
- Elementos de datos de fecha, hora o indicación de fecha y hora

En estos casos, se prohíbe el uso de la cláusula PICTURE.

La cláusula PICTURE sólo se puede especificar en el nivel elemental.

PIC es una abreviatura de PICTURE y tiene el mismo significado.

serie-caracteres

serie-caracteres se compone de determinados caracteres COBOL utilizados como símbolos de imagen. Las combinaciones permitidas determinan la categoría del elemento de datos elemental. Las combinaciones permitidas determinan la categoría del elemento de datos elemental, excepto cuando se especifica la frase de entorno local. Una frase LOCALE en una cláusula PICTURE define un elemento editado numérico de categoría.

serie-caracteres puede contener un máximo de 50 caracteres.

Entorno local

Consulte el apartado "frase LOCALE" en la página 191.

frase LOCALE

Cuando se especifica la frase LOCALE en la cláusula PICTURE, la edición se lleva a cabo de acuerdo con las especificaciones de entorno local. Se aplican las reglas siguientes:

- Una cláusula BLANK WHEN ZERO tiene prioridad sobre la edición del entorno local.
- Cuando se especifica mnemonic-name-1 , el entorno local utilizado para editar y deseditar el elemento es el asociado con mnemonic-name-1 en el párrafo SPECIAL-NAMES. De lo contrario, se utiliza el entorno local actual.

  Nota: La conmutación de entornos locales entre las etapas de edición y desedición de un elemento puede provocar resultados imprevisibles. Debe asegurarse de que el entorno local utilizado para editar un elemento es el mismo que el entorno local utilizado para deseditar un elemento.
- Si se especifica un símbolo de signo de moneda (cs) en la serie de imagen, la posición, la longitud y la serie de caracteres utilizados para el signo de moneda se determinan a partir del entorno local.
- El separador decimal, el separador de millares y la agrupación están determinados por el entorno local.
- La alineación de coma decimal y la sustitución de cero tienen lugar tal como se describe en "Reglas de alineación" en la página 151.
- Si se especifica + en la serie de caracteres PICTURE, la forma en que se representan los números positivos y negativos viene determinada por un entorno local.
- Los datos de envío se alinean en la coma decimal y (si es necesario) se truncan o rellenan con ceros en cualquiera de los extremos dentro de las posiciones de caracteres de recepción del elemento de datos de recepción. Los datos también están justificados por la derecha, con la agrupación y los separadores aplicados de acuerdo con la especificación de entorno local. Los ceros iniciales se sustituyen por espacios en blanco.

Si, después del formateo, el número de posiciones de dígito especificado en la serie de caracteres PICTURE no cabe en el elemento receptor, y hay un exceso de dígitos en el elemento emisor, los dígitos se truncan a la izquierda y se emite un mensaje de escape del sistema operativo.

Símbolos utilizados en la cláusula PICTURE

Cualquier carácter de puntuación que aparezca dentro de la serie de caracteres PICTURE no se considera un carácter de puntuación, sino que es un símbolo de serie de caracteres PICTURE.

Cuando se especifica en el párrafo SPECIAL-NAMES, DECIMAL-POINT IS COMA intercambia las funciones del punto y la coma en series de caracteres PICTURE y en literales numéricos.

Las letras minúsculas que corresponden a las letras mayúsculas que representan los siguientes símbolos PICTURE son equivalentes a sus representaciones en mayúsculas en una serie de caracteres PICTURE:

```
A, B, E, G, N, P, S,  V, X, Z, CR, DB
```

Todas las demás letras en minúsculas no son equivalentes a sus representaciones en mayúsculas correspondientes.

El significado de cada símbolo de cláusula PICTURE se define en las tablas siguientes:

- Si no se especifica la frase LOCALE, consulte Tabla 17 en la página 192.
- Si se especifica la frase LOCALE, consulte Tabla 18 en la página 194.

La cabecera *Tamaño* indica cómo se cuenta el elemento al determinar el número de posiciones de caracteres en el elemento. El tipo de las posiciones de carácter depende de la cláusula USAGE especificada para el elemento.

Uso	**Tipo de posiciones de caracteres**	**Número de bytes por carácter**
VISUALIZAR	Alfanumérico	1
DISPLAY-1	DBCS	2
NACIONAL	Nacional	2
Todos los demás	Conceptuales	No aplicable

Tabla 17. ***Significados de símbolo de cláusula PICTURE cuando no se especifica la frase LOCALE***

Símbolo	**Significado**	**Tamaño**
A	Posición de carácter que sólo puede contener una letra del alfabeto latino o un espacio.	Cada 'A' se cuenta como una posición de carácter en el tamaño del elemento de datos.
B	Para DISPLAY de uso, una posición de carácter en la que se inserta un espacio alfanumérico. Para el uso DISPLAY-1, una posición de carácter en la que se inserta un espacio DBCS. Para uso NATIONAL, una posición de carácter en la que se inserta un espacio nacional.	Cada 'B' se cuenta como una posición de carácter en el tamaño del elemento de datos.
E	Marca el inicio del exponente en un elemento de coma flotante externo. Para obtener detalles adicionales de elementos de coma flotante externos, consulte “Categorías de datos y reglas PICTURE” en la página 199.	Cada 'E' se cuenta como una posición de carácter en el tamaño del elemento de datos.
G	Una posición de carácter DBCS.	Cada 'G' se cuenta como una posición de carácter en el tamaño del elemento de datos.
N	Una posición de carácter DBCS cuando se especifica con el uso DISPLAY-1 o cuando no se especifica el uso y la opción de compilador NSYMBOL (DBCS) está en vigor. Para la categoría nacional, una posición de carácter nacional cuando se especifica con el uso NATIONAL o cuando no se especifica el uso y la opción de compilador NSYMBOL (NATIONAL) está en vigor. Para la categoría nacional editada, una posición de carácter nacional.	Cada ' N' se cuenta como una posición de carácter en el tamaño del elemento de datos.

Tabla 17. ***Significados de símbolo de cláusula PICTURE cuando no se especifica la frase LOCALE*** *(continuación)*

Símbolo	Significado	Tamaño
P	Una posición de escalado decimal asumida. Se utiliza para especificar la ubicación de un punto decimal supuesto cuando el punto no está dentro del número que aparece en el elemento de datos. Consulte "Símbolo P" en la página 197 para obtener más detalles.	No se cuenta en el tamaño del elemento de datos. Los caracteres de posición de escalado se cuentan al determinar el número máximo de posiciones de dígitos en elementos editados numéricos o en elementos que se utilizan como operandos aritméticos. El tamaño del valor es el número de posiciones de dígito representadas por la serie de caracteres PICTURE.
S	Un indicador de la presencia (pero no la representación, y no necesariamente la posición) de un signo operativo. Un signo operativo indica si el valor de un elemento implicado en una operación es positivo o negativo.	No se cuenta en el tamaño del elemento elemental, a menos que una cláusula SIGN asociada especifique la frase SEPARAR CARÁCTER (que se contaría como una posición de carácter).
V	Indicador de la ubicación de la coma decimal asumida. No representa una posición de carácter. Cuando la coma decimal asumida está a la derecha del símbolo situado más a la derecha de la serie, la V es redundante.	No se cuenta en el tamaño del elemento elemental.
X	Posición de carácter que puede contener cualquier carácter permitido del juego de caracteres alfanuméricos del sistema.	Cada ' X' se cuenta como una posición de carácter en el tamaño del elemento de datos.
Z	Una posición de carácter numérico inicial. Cuando esa posición contiene un cero, un carácter de espacio sustituye al cero.	Cada 'Z' se cuenta como una posición de un carácter en el tamaño del elemento de datos.
9	Posición de carácter que contiene un número.	Cada nueve especifica un dígito decimal en el valor del elemento. Para los usos DISPLAY y NATIONAL, cada nueve se cuenta como una posición de carácter en el tamaño del elemento de datos.
0	Posición de carácter en la que se inserta el número cero.	Cada cero se cuenta como una posición de carácter en el tamaño del elemento de datos.
1	Posición de carácter que contiene un valor booleano de B "1" o B "0". El uso debe definirse explícita o implícitamente como DISPLAY.	Cada valor booleano se cuenta como una posición de carácter en el tamaño del elemento de datos.
/	Posición de carácter en la que se inserta el carácter de barra inclinada.	Cada carácter de barra inclinada se cuenta como una posición de carácter en el tamaño del elemento de datos.
,	Posición de carácter en la que se inserta una coma.	Cada coma se cuenta como una posición de carácter en el tamaño del elemento de datos.

Tabla 17. ***Significados de símbolo de cláusula PICTURE cuando no se especifica la frase LOCALE*** *(continuación)*

Símbolo	Significado	Tamaño
.	Símbolo de edición que representa la coma decimal a efectos de alineación. Si el carácter de inserción de punto es el último símbolo de la serie de caracteres PICTURE, la cláusula PICTURE debe ser la última cláusula de esa entrada de descripción de datos y debe ir inmediatamente seguida del punto separador. El carácter de coma decimal utilizado en tiempo de ejecución se toma del entorno local. **Nota:** Para un programa determinado, las funciones del punto y la coma se intercambian si la cláusula DECIMAL-POINT IS COMA se especifica en el párrafo SPECIAL-NAMES. En este intercambio, las reglas para el punto se aplican a la coma y las reglas para la coma se aplican al punto donde aparecen en una cláusula PICTURE.	Cada punto se cuenta como una posición de carácter en el tamaño del elemento de datos.
+ - CR BD	Cómo modificar símbolos de control de signo. Cada uno representa la posición de carácter en la que se coloca el símbolo de control de signo de edición.	Cada carácter utilizado en el símbolo de signo de edición se cuenta como una posición de carácter en el tamaño del elemento de datos.
*	Un símbolo de protección de comprobación: una posición de carácter numérico inicial en la que se coloca un asterisco cuando dicha posición contiene un cero.	Cada asterisco se cuenta como una posición de carácter en el tamaño del elemento.
cs	*cs* puede ser cualquier símbolo de moneda válido. Un símbolo de moneda representa una posición de carácter en la que se coloca un valor de signo de moneda. El símbolo de moneda predeterminado es el carácter asignado al valor X'24' en la página de códigos en vigor en el momento de la compilación. En este documento, el símbolo de moneda predeterminado se representa mediante el signo de dólar ($) y *cs* representa cualquier símbolo de moneda válido. Para obtener detalles, consulte "Símbolo de moneda" en la página 198.	La primera aparición de un símbolo de moneda añade el número de caracteres en el valor de signo de moneda al tamaño del elemento de datos. Cada aparición posterior añade una posición de carácter al tamaño del elemento de datos.

Tabla 18. ***Significa el símbolo de cláusula PICTURE cuando se especifica la frase LOCALE***

Símbolo	Significado	Tamaño
9	Posición de carácter que contiene un número y se cuenta en el número de números que pueden aparecer en el elemento editado.	Cada nueve especifica un dígito decimal en el valor del elemento. Para los usos DISPLAY y NATIONAL, cada nueve se cuenta como una posición de carácter en el tamaño del elemento de datos.

Tabla 18. ***Significa el símbolo de cláusula PICTURE cuando se especifica la frase LOCALE*** *(continuación)*

Símbolo	Significado	Tamaño
.	Símbolo de edición que representa la coma decimal a efectos de alineación. Si el carácter de inserción de punto es el último símbolo de la serie de caracteres PICTURE, la cláusula PICTURE debe ser la última cláusula de esa entrada de descripción de datos y debe ir inmediatamente seguida del punto separador. El carácter de coma decimal utilizado en tiempo de ejecución se toma del entorno local. **Nota:** Para un programa determinado, las funciones del punto y la coma se intercambian si la cláusula DECIMAL-POINT IS COMA se especifica en el párrafo SPECIAL-NAMES. En este intercambio, las reglas para el punto se aplican a la coma y las reglas para la coma se aplican al punto donde aparecen en una cláusula PICTURE.	Cada punto se cuenta como una posición de carácter en el tamaño del elemento de datos.
+	Edición del símbolo de control de signo. El signo + indica que el elemento editado se debe firmar de acuerdo con el entorno local especificado. Si no se especifica +, el elemento editado no estará firmado.	Cada + se cuenta como una posición de carácter en el tamaño del elemento de datos.
cs	El símbolo de moneda en la serie de caracteres indica que el elemento editado debe incluir la serie de moneda asociada con el entorno local especificado.	La primera aparición de un símbolo de moneda añade el número de caracteres en el valor de signo de moneda al tamaño del elemento de datos. Cada aparición posterior añade una posición de carácter al tamaño del elemento de datos.

La Figura 1 en la página 196 muestra la secuencia en la que deben especificarse los símbolos de cláusula PICTURE si no se especifica la frase LOCALE. Consulte las notas al final de la figura. La Figura 2 en la página 197 muestra la secuencia en la que deben especificarse los símbolos de cláusula PICTURE si se especifica la frase LOCALE.

First Symbol / Second Symbol		Non-floating Insertion Symbols										Floating Insertion Symbols						Other Symbols								
		B	0	/	,	.	+ -	+ -	CR DB	$	E	Z *	Z *	+ -	+ -	$	$	9	A X	S	V	P	P	1	G	N
Non-floating Insertion Symbols	B	X	X	X	X	X	X			X		X	X	X	X	X	X	X	X		X		X		X	
	0	X	X	X	X	X	X			X		X	X	X	X	X	X	X	X		X		X			
	/	X	X	X	X	X	X			X		X	X	X	X	X	X	X	X		X		X			
	,	X	X	X	X	X	X			X		X	X	X	X	X	X	X			X		X			
	.	X	X	X	X		X			X		X		X		X		X								
	+ -																									
	+ -	X	X	X	X	X				X	X	X	X			X	X	X			X	X	X			
	CR DB	X	X	X	X	X				X		X	X			X	X	X			X	X	X			
	$						X																			
	E				X	X												X			X					
Floating Insertion Symbols	Z *	X	X	X	X		X			X		X														
	Z *	X	X	X	X	X	X			X		X	X								X		X			
	+ -	X	X	X	X					X				X												
	+ -	X	X	X	X	X				X				X	X						X		X			
	$	X	X	X	X		X									X										
	$	X	X	X	X	X	X									X	X				X		X			
Other Symbols	9	X	X	X	X	X	X			X		X		X		X		X	X	X	X		X			
	A X	X	X	X														X	X							
	S																									
	V	X	X	X	X		X			X		X		X		X		X		X		X				
	P	X	X	X	X		X			X		X		X		X		X		X		X				
	P						X			X										X	X		X			
	1																									
	G	X																							X	
	N																									X

Figura 1. Secuencia de símbolos de cláusula PICTURE cuando no se especifica la frase LOCALE

Notas a Figura 1 en la página 196:

1. Una X en una intersección indica que los símbolos en la parte superior de la columna pueden, en una serie de caracteres determinada, aparecer en cualquier lugar a la izquierda de los símbolos a la izquierda de la fila.
2. El carácter $, sin embargo, se representa en el juego de caracteres adecuado, es el valor predeterminado para el símbolo de moneda.

3. Al menos uno de los símbolos A, X, Z, 9 o *, o al menos dos de los símbolos +,-o $deben estar presentes en una serie PICTURE.
4. Los símbolos G o N pueden aparecer solos en la serie de caracteres PICTURE.
5. Los símbolos de inserción no flotante + y-, los símbolos de inserción flotante Z, *, +,-y $, y el símbolo P aparecen dos veces en la tabla de prioridad de caracteres PICTURE anterior. La columna situada más a la izquierda y la fila situada más arriba de cada símbolo representa su uso a la izquierda de la posición de coma decimal. El segundo aspecto del símbolo en la tabla representa su uso a la derecha de la posición de coma decimal. ({ }) indican elementos que se excluyen mutuamente.
6. Las llaves ({}) indican elementos que se excluyen mutuamente.

First Symbol / Second Symbol		Symbols			
		9	CS	.	+
Symbols	9	X	X	X	X
	CS				X
	.	X	X		X
	+				

Figura 2. Secuencia de símbolos de cláusula PICTURE cuando se especifica la frase LOCALE

Símbolo P

El símbolo P especifica una posición de escalado e implica un punto decimal asumido (a la izquierda de los Ps si los Ps son caracteres PICTURE más a la izquierda; a la derecha de los Ps si los Ps son caracteres PICTURE más a la derecha).

El símbolo de coma decimal asumido V es redundante como el carácter situado más a la izquierda o más a la derecha dentro de una descripción PICTURE de este tipo.

El símbolo P sólo se puede especificar como una serie continua de Ps en las posiciones de dígitos situados más a la izquierda o más a la derecha dentro de una serie de caracteres PICTURE.

En determinadas operaciones que hacen referencia a un elemento de datos cuya serie de caracteres PICTURE contiene el símbolo P, se utiliza el valor algebraico del elemento de datos en lugar de la representación de caracteres real del elemento de datos. Este valor algebraico asume la coma decimal en la ubicación prescrita y cero en lugar de la posición de dígito especificada por el símbolo P. El tamaño del valor es el número de posiciones de dígito representadas por la serie de caracteres PICTURE. Estas operaciones son cualquiera de las siguientes:

- Cualquier operación que requiera un operando de envío numérico
- Una sentencia MOVE donde el operando emisor es numérico y su serie de caracteres PICTURE contiene el símbolo P
- Una sentencia MOVE donde el operando emisor es numérico editado y su serie de caracteres PICTURE contiene el símbolo P, y el operando receptor es numérico o numérico editado
- Una operación de comparación en la que ambos operandos son numéricos

En todas las demás operaciones, las posiciones de dígito especificadas con el símbolo P se ignoran y no se cuentan en el tamaño del operando.

Símbolo de moneda

El símbolo de moneda de una serie de caracteres de imagen se representa mediante el símbolo de moneda predeterminado $o mediante un único carácter especificado en la opción de compilador CURRENCY o en la cláusula CURRENCY SIGN del párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION.

Si se especifica la cláusula CURRENCY SIGN, se ignoran las opciones de compilador CURRENCY y NOCURRENCY. Si no se especifica la cláusula CURRENCY SIGN y la opción de compilador NOCURRENCY está en vigor, se utiliza el signo de dólar ($) como valor de signo de moneda predeterminado y símbolo de moneda. Para obtener más información sobre la cláusula CURRENCY SIGN, consulte "cláusula CURRENCY SIGN" en la página 103. Para obtener más información sobre las opciones de compilador CURRENCY y NOCURRENCY, consulte *MONEDA* en la publicación *COBOL for Linux en x86 Guía de programación*.

Un símbolo de moneda puede repetirse dentro de la serie de caracteres PICTURE para especificar la inserción flotante. No deben utilizarse símbolos de moneda diferentes en la misma serie de caracteres PICTURE.

A diferencia de todos los demás símbolos de imagen, los símbolos de moneda distinguen entre mayúsculas y minúsculas. Por ejemplo, 'D' y' d' especifican diferentes símbolos de moneda.

Un símbolo de moneda sólo se puede utilizar para definir un elemento editado numéricamente con USAGE DISPLAY.

Representación de serie de caracteres

El tema lista los símbolos que pueden aparecer una o más veces en la serie de caracteres PICTURE.

Símbolos que pueden aparecer más de una vez

Los símbolos siguientes pueden aparecer más de una vez en una serie de caracteres PICTURE:

```
A  B  G  N  P  X    Z  9  0  /  ,  +  –  *  cs
```

Al menos uno de los símbolos A, G, N, X, Z, 9 o *, o al menos dos de los símbolos +, –o *cs* deben estar presentes en una serie PICTURE.

Un entero no cero sin signo entre paréntesis inmediatamente después de cualquiera de estos símbolos especifica el número de apariciones consecutivas de ese símbolo.

Ejemplo: Las dos especificaciones de cláusula PICTURE siguientes son equivalentes:

```
PICTURE IS $99999.99CR
PICTURE IS $9(5).9(2)CR
```

Símbolos que solo pueden aparecer una vez

Los símbolos siguientes sólo pueden aparecer una vez en una serie de caracteres PICTURE:

```
E  S  V  .  CR  DB  1
```

Excepto para el símbolo PICTURE V, cada aparición de cualquiera de los símbolos anteriores en una serie de caracteres PICTURE dada representa una aparición de ese carácter o conjunto de caracteres permitidos en el elemento de datos.

Si se especifica la frase LOCALE, sólo el símbolo 9 puede aparecer más de una vez. Si se especifica la frase LOCALE, los símbolos siguientes sólo pueden aparecer una vez en una serie de caracteres PICTURE:

```
. + cs
```

Categorías de datos y reglas PICTURE

Las combinaciones permitidas de símbolos PICTURE determinan la categoría de datos del artículo.

Las categorías de datos son:

- Alfabético
- Alfanumérico
- Alfanumérico-editado
- Booleano
- DBCS
- Coma flotante externa
- Nacional
- Nacional-editado
- Numérico
- Numérico-editado

Nota:

- La coma flotante interna de categoría se define mediante una cláusula USAGE que especifica la frase COMP-1 o COMP-2 .
- Si la frase LOCALE se especifica en una cláusula PICTURE , la categoría de datos definida por dicha cláusula PICTURE sólo se edita numéricamente.

Elementos alfabéticos

La serie de caracteres PICTURE sólo puede contener el símbolo A.

El contenido del elemento debe estar formado únicamente por letras del alfabeto latino y el carácter de espacio.

Otras cláusulas

USAGE DISPLAY debe estar especificado o implícito.

Cualquier cláusula VALUE asociada debe especificar un literal alfanumérico que contenga sólo caracteres alfabéticos, SPACE o un carácter simbólico como valor de una constante figurativa.

El entorno local de ejecución en vigor debe indicar una página de códigos que incluya caracteres DBCS. Para obtener información sobre entornos locales, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

No incluya un carácter de un solo byte en un elemento de datos DBCS.

Cuando el relleno es necesario para un elemento de datos DBCS, se aplican las reglas siguientes:

- El relleno se realiza utilizando caracteres de espacio de doble byte hasta que se rellena el área de datos (basándose en el número de posiciones de caracteres de doble byte asignadas para el elemento de datos).
- El relleno se realiza utilizando caracteres de espacio de un solo byte cuando el relleno necesario no es un número par de bytes (por ejemplo, cuando un elemento de grupo alfanumérico se mueve a un elemento de datos DBCS).

Elementos booleanos

Se aplican las reglas siguientes:

- La serie de caracteres PICTURE sólo puede contener el símbolo 1.
- Sólo puede especificarse un carácter 1.
- La USAGE de un elemento sólo puede ser DISPLAY.
- Una cláusula VALUE asociada debe especificar un literal booleano (B "1" o B "0") o cero.
- No se pueden especificar las cláusulas siguientes para un elemento booleano:
  - cláusula SIGN
  - Cláusula BLANK WHEN ZERO
  - Cláusula ASCENDING/DESCENDING KEY.

Elementos alfanuméricos

La serie de caracteres PICTURE debe constar de determinados símbolos.

Los símbolos son:

- Una o más apariciones del símbolo X.
- Combinaciones de los símbolos A, X y 9. (Una serie de caracteres que contiene todos los valores As o all 9s no define un elemento alfanumérico.)

El elemento se trata como si la serie de caracteres sólo contuviera el símbolo X.

El contenido del elemento en formato de datos estándar puede ser cualquier carácter permitido del juego de caracteres del sistema.

Otras cláusulas

USAGE DISPLAY debe estar especificado o implícito.

Cualquier cláusula VALUE asociada debe especificar un literal alfanumérico o una de las siguientes constantes figurativas:

- CERO
- ESPACIO
- PRESUPUESTO
- ALTO VALOR
- LOW-VALUE
- *carácter-simbólico*
- ALL *literal-alfanumérico*

Elementos alfanuméricos editados

La serie de caracteres PICTURE puede contener los símbolos siguientes: `A X 9 B 0 /`.

La serie debe contener al menos una A o X, y al menos una B o 0 (cero) o/.

El contenido del elemento en formato de datos estándar debe estar formado por dos o más caracteres del juego de caracteres del sistema.

Otras cláusulas

USAGE DISPLAY debe estar especificado o implícito.

Cualquier cláusula VALUE asociada debe especificar un literal alfanumérico o o una de las siguientes constantes figurativas:

- CERO
- ESPACIO
- PRESUPUESTO
- ALTO VALOR

- LOW-VALUE
- *carácter-simbólico*
- ALL *literal-alfanumérico*

El literal se trata exactamente como se ha especificado; no se realiza ninguna edición.

Elementos DBCS

La serie de caracteres PICTURE puede contener los símbolos G, G y Bo N. Cada G, Bo N representa una única posición de carácter DBCS.

Cualquier cláusula VALUE asociada debe contener un literal DBCS, la constante figurativa SPACE o la constante figurativa ALL *literal-DBCS*.

Otras cláusulas

El entorno local de ejecución en vigor debe indicar una página de códigos que incluya caracteres DBCS. Para obtener información sobre entornos locales, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

No incluya un carácter de un solo byte en un elemento de datos DBCS.

Cuando el relleno es necesario para un elemento de datos DBCS, se aplican las reglas siguientes:

- El relleno se realiza utilizando caracteres de espacio de doble byte hasta que se rellena el área de datos (basándose en el número de posiciones de caracteres de doble byte asignadas para el elemento de datos).
- El relleno se realiza utilizando caracteres de espacio de un solo byte cuando el relleno necesario no es un número par de bytes (por ejemplo, cuando un elemento de grupo alfanumérico se mueve a un elemento de datos DBCS).

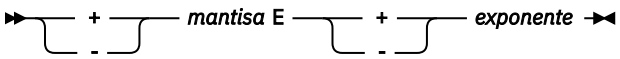
Cuando se utiliza el símbolo G de PICTURE, se debe especificar USAGE DISPLAY-1 . Cuando se utiliza el símbolo N de PICTURE y la opción de compilador NSYMBOL (DBCS) está en vigor, USAGE DISPLAY-1 está implícito si se omite la cláusula USAGE.

Elementos de coma flotante externos

Un elemento de datos se describe como de coma flotante externo de categoría mediante su serie de caracteres PICTURE.

A continuación se describen los detalles de serie de caracteres PICTURE.

Formato



+ o-

Un carácter de signo debe preceder inmediatamente a la mantisa y al exponente.

Un signo + indica que se utilizará un signo positivo en la salida para representar valores positivos y que un signo negativo representará valores negativos.

Un signo-indica que se utilizará un espacio en blanco en la salida para representar valores positivos y que un signo negativo representará valores negativos.

Cada posición de signo ocupa un byte de almacenamiento.

mantisa

La mantisa puede contener los símbolos:

```
9 . V
```

Una coma decimal real se puede representar con un punto (.) mientras que una coma decimal asumida se representa con una V.

Una coma decimal real o una coma decimal asumida debe estar presente en la mantisa; la coma decimal puede ser inicial, incorporada o final.

La mantisa puede contener de 1 a 16 caracteres numéricos.

E
Indica el exponente.

exponente
El exponente debe constar del símbolo 99.

Ejemplo: Pic -9v9(9)E-99

La frase DISPLAY de la cláusula USAGE y una serie de caracteres de imagen de coma flotante definen el elemento como un *elemento de datos de coma flotante de visualización*.

La frase NATIONAL de la cláusula USAGE y una serie de caracteres de imagen de coma flotante definen el elemento como un *elemento de datos de coma flotante nacional.*

Para los elementos definidos con el uso DISPLAY, cada símbolo de imagen excepto V define una posición de carácter alfanumérico en el elemento.

Para los elementos definidos con el uso NATIONAL, cada símbolo de imagen excepto V define una posición de carácter nacional en el elemento.

Otras cláusulas

La frase DISPLAY o la frase NATIONAL de la cláusula USAGE debe especificarse o implicarse.

Las cláusulas LIKE, OCCURS, REDEFINES, RENAMES y TYPEDEF se pueden asociar con elementos de coma flotante externos.

La cláusula SIGN se acepta como documentación y no tiene ningún efecto en la representación del signo.

La cláusula SYNCHRONIZED se trata como documentación.

Las cláusulas siguientes no son válidas con elementos de coma flotante externos:

- BLANCO CUANDO CERO
- JUSTIFICADO
- Valor

Partidas nacionales

La serie de caracteres PICTURE puede contener una o más apariciones del símbolo de imagen N.

Estas reglas se aplican cuando la opción de compilador NSYMBOL (NATIONAL) está en vigor o cuando se especifica la cláusula USAGE NATIONAL. En ausencia de una cláusula USAGE NATIONAL, si la opción de compilador NSYMBOL (DBCS) está en vigor, el símbolo de imagen N representa un carácter DBCS y se aplican las normas de la cláusula PICTURE para un elemento DBCS.

Cada N representa una única posición de carácter nacional.

Cualquier cláusula VALUE asociada debe especificar un literal alfanumérico, un literal nacional o una de las siguientes constantes figurativas:

- CERO
- ESPACIO
- PRESUPUESTO
- ALTO VALOR
- LOW-VALUE

- *carácter-simbólico*
- ALL *literal-alfanumérico*
- ALL *literal-nacional*

Otras cláusulas

Sólo se puede especificar la frase NATIONAL en la cláusula USAGE. Cuando se utiliza el símbolo N de PICTURE y la opción de compilador NSYMBOL (NATIONAL) está en vigor, USAGE NATIONAL está implícito si se omite la cláusula de uso.

Se pueden utilizar las cláusulas siguientes:

- JUSTIFICADO
- externo
- GLOBAL
- APARICIONES
- REDEFINE
- RENAMES
- SINCRONIZADO
- TYPEDEF

No se pueden utilizar las cláusulas siguientes:

- BLANCO CUANDO CERO
- FORMATO DE FECHA
- Formato
- SIGNO
- Tipo

Elementos editados a nivel nacional

La serie de caracteres PICTURE debe contener al menos un símbolo N, y al menos una instancia de uno de estos símbolos: B 0 (cero) o / (barra inclinada).

Cada símbolo representa una única posición de carácter nacional.

Cualquier cláusula VALUE asociada debe especificar un literal alfanumérico, un literal nacional o una de las siguientes constantes figurativas:

- CERO
- ESPACIO
- PRESUPUESTO
- ALTO VALOR
- LOW-VALUE
- *carácter-simbólico*
- ALL *literal-alfanumérico*
- ALL *literal-nacional*

El literal se trata exactamente como se ha especificado; no se realiza ninguna edición.

La opción de compilador NSYMBOL (NATIONAL) no tiene ningún efecto en la definición de un elemento de datos de la categoría editada a nivel nacional.

Otras cláusulas

Debe especificarse o implicarse USAGE NATIONAL.

Se pueden utilizar las cláusulas siguientes:

- JUSTIFICADO
- externo
- GLOBAL
- APARICIONES
- REDEFINE
- RENAMES
- SINCRONIZADO
- TYPEDEF

No se pueden utilizar las cláusulas siguientes:

- BLANCO CUANDO CERO
- FORMATO DE FECHA
- Formato
- SIGNO
- Tipo

Elementos numéricos

Existen varios tipos de elementos numéricos.

Los tipos son:

- Binario
- Decimal empaquetado (decimal interno)
- Decimal con zona (decimal externo)
- Decimal nacional (decimal externo)

El tipo de un elemento numérico se define mediante la cláusula de uso tal como se muestra en la tabla siguiente.

Tabla 19. ***Tipos numéricos***

Tipo	**cláusula USAGE**
Binario	BINARY, COMP, COMP-4o COMP-5
Decimal interno	PACKED-DECIMAL, COMP-3
Decimal con zona (decimal externo)	VISUALIZAR
Decimal nacional (decimal externo)	NACIONAL

Para campos de fecha numéricos, la serie de caracteres PICTURE sólo puede contener los símbolos 9 y S. Para todos los otros campos numéricos, la serie de caracteres PICTURE sólo puede contener los símbolos 9, P, S y V.

El símbolo S sólo se puede escribir como el carácter situado más a la izquierda en la serie de caracteres PICTURE.

El símbolo V sólo se puede escribir una vez en una serie de caracteres PICTURE determinada.

Para elementos binarios, el número de posiciones de dígito debe oscilar entre 1 y 18, ambos inclusive. Para elementos decimales empaquetados y decimales con zona, el número de posiciones de dígitos debe estar comprendido entre 1 y 18, inclusive, cuando la opción de compilador ARITH (COMPAT) está en vigor, o entre 1 y 31, inclusive, cuando la opción de compilador ARITH (EXTEND) está en vigor.

Para campos de fecha numéricos, el número de posiciones de dígito debe coincidir con el número de caracteres especificado por la cláusula DATE FORMAT.

Si no está firmado, el contenido del elemento en formato de datos estándar debe contener una combinación de los números arábigos 0-9. Si está firmado, también puede contener un signo +,-u otra representación del signo operativo.

Ejemplos de rangos válidos

```
 PICTURE   Valid range of values
   9999     0 through 9999
    S99    -99 through +99
 S999V9    -999.9 through +999.9
 PPP999     0 through .000999
S999PPP    -1000 through -999000 and
           +1000 through +999000 or zero
```

Otras cláusulas

La USAGE del elemento puede ser DISPLAY, NATIONAL, BINARY, COMPUTATIONAL, PACKED-DECIMAL, COMPUTATIONAL-3, COMPUTATIONAL-4o COMPUTATIONAL-5.

Para los elementos numéricos con signo descritos con el uso NATIONAL, debe especificarse o implicarse la cláusula SIGN IS SEPARATE.

La opción de compilador TRUNC pueden afectar al uso de elementos de datos numéricos. Para obtener detalles, consulte *TRUNC* en *COBOL for Linux en x86 Guía de programación*.

Elementos editados numérica-editados

La serie de caracteres PICTURE puede contener determinados símbolos.

Los símbolos son:

```
B  P  V  Z  9  0  /  ,  .  +  -  CR  DB  *  cs
```

Las combinaciones de símbolos permitidos se determinan a partir del orden de símbolos de la cláusula PICTURE permitido (consulte la figura en "Símbolos utilizados en la cláusula PICTURE" en la página 191) y las reglas de edición (consulte "Edición de cláusula PICTURE" en la página 206).

Se aplican las reglas siguientes:

- Debe especificarse la cláusula BLANK WHEN ZERO para el elemento o la serie debe contener al menos uno de los símbolos siguientes:

  ```
  B  /  Z  0  ,  .  *  +  -  CR  DB  cs
  ```

- Sólo uno de los símbolos siguientes puede escribirse en una serie de caracteres PICTURE determinada:

  ```
  +  -  CR  DB
  ```

- Si la opción de compilador ARITH (COMPAT) está en vigor, el número de posiciones de dígito representadas en la serie de caracteres debe estar en el rango de 1 a 18, ambos inclusive. Si la opción de compilador ARITH (EXTEND) está en vigor, el número de posiciones de dígito representadas en la serie de caracteres debe estar en el rango de 1 a 31, ambos inclusive.
- El número total de posiciones de caracteres en la serie (incluidas las posiciones de caracteres de edición) no debe exceder de 127.
- El contenido de esas posiciones de caracteres que representan dígitos en formato de datos estándar debe ser uno de los 10 números arábigos.

Otras cláusulas

Debe especificarse o implicarse USAGE DISPLAY o NATIONAL.

Si el uso del elemento es DISPLAY, cualquier cláusula VALUE asociada debe especificar un literal alfanumérico o una constante figurativa. El valor se asigna sin editar.

Si el uso del elemento es NATIONAL, cualquier cláusula VALUE asociada debe especificar un literal alfanumérico, un literal nacional o una constante figurativa. El valor se asigna sin editar.

Edición de cláusula PICTURE

Hay dos métodos generales de edición en una cláusula PICTURE, la edición de inserción y la edición de supresión y sustitución.

La edición de inserción incluye los siguientes tipos de edición:

- Inserción simple
- Inserción especial
- Inserción fija
- Inserción flotante

La edición de supresión y sustitución incluye los siguientes tipos de edición:

- Supresión de ceros y sustitución por asteriscos
- Supresión y sustitución de ceros por espacios

El tipo de edición permitido para un elemento depende de su *categoría de datos*. El tipo de edición que es válido para cada categoría se muestra en la tabla siguiente. *cs* indica cualquier símbolo de moneda válido.

Tabla 20. ***Categorías de datos***

Categoría de datos	**Tipo de edición**	**Símbolo de inserción**
Alfabético	Ninguna	Ninguna
Alfanumérico	Ninguna	Ninguna
Alfanumérico-editado	Inserción simple	B 0/
Booleano	Ninguna	Ninguna
DBCS	Inserción simple	B
Coma flotante externa	Inserción especial	.
Nacional	Ninguna	Ninguna
Nacional-editado	Inserción simple	B 0/
Numérico	Ninguna	Ninguna
Numérico-editado	Inserción simple Inserción especial Inserción fija Inserción flotante Supresión de ceros Sustitución	B 0/, . *cs* +-BD CR *cs* +- Z * Z * +- *cs*

Los tipos de edición se describen en las secciones siguientes:

- “Edición de inserción simple” en la página 206
- “Edición de inserción especial” en la página 207
- “Edición de inserción fija” en la página 207
- “Edición de inserción flotante” en la página 209
- “Supresión de ceros y edición de sustitución” en la página 210

Edición de inserción simple

Este tipo de edición es válido para elementos editados alfanuméricos, editados numéricos y DBCS.

Cada símbolo de inserción se cuenta en el tamaño del elemento y representa la posición dentro del elemento donde se va a insertar el carácter equivalente. Para los elementos DBCS editados, cada símbolo de inserción (B) se cuenta en el tamaño del elemento y representa la posición dentro del elemento donde se va a insertar el espacio DBCS.

Por ejemplo:

IMAGEN	Valor de los datos	Resultado editado
X (10) /XX	ALPHANUMER01	ALPHANUMER/01
X (5) BX (7)	ALPHANUMERIC	ALFANUMÉRICO
99,B999,B000	1234	01,*b*234,*b*000[1]
99,999	12345	12,345
GGBBGG	D1D2D3D4	D1D2*bbbb*D3D4[1]
Notas: 1. El símbolo *b* representa un espacio.		

Edición de inserción especial

Este tipo de edición es válido para elementos editados numérica-o elementos de coma flotante externos.

El punto (.) es el símbolo de inserción especial; también representa la coma decimal real a efectos de alineación.

Nota: Si se especifica la cláusula DECIMAL-POINT IS COMA, se utilizará una coma en lugar del punto.

El símbolo de inserción de periodo se cuenta en el tamaño del elemento y representa la posición dentro del elemento donde se inserta el punto decimal real.

La coma decimal real o el símbolo V como la coma decimal asumida, pero no ambos, deben especificarse en una serie de caracteres PICTURE.

Por ejemplo:

IMAGEN	Valor de los datos	Resultado editado
`999.99`	`1.234`	`001.23`
`999.99`	`12.34`	`012.34`
`999.99`	`123.45`	`123.45`
`999.99`	`1234.5`	`234.50`
`+999.99E+99`	`12345`	`+123.45E+02`

Edición de inserción fija

La edición de inserción fija sólo es válida para elementos editados numéricos.

Se utilizan los siguientes símbolos de inserción:

- *cs*
- +-CR DB (símbolos de control de signo de edición)

En la edición de inserción fija, sólo se puede especificar un símbolo de moneda y un símbolo de control de signo de edición en una serie de caracteres PICTURE.

A menos que esté precedido por un símbolo + o-, el símbolo de moneda debe ser el primer carácter de la serie de caracteres.

Cuando se utiliza + o-como símbolo, debe ser el primer o último carácter de la serie de caracteres.

Cuando se utiliza CR o DB como símbolo, debe ocupar las posiciones de dos caracteres más a la derecha en la serie de caracteres. Si estas dos posiciones de caracteres contienen los símbolos CR o DB, las letras mayúsculas son los caracteres de inserción.

La edición de símbolos de control de signos genera resultados que dependen del valor del elemento de datos, tal como se muestra a continuación:

Símbolo de edición en serie de caracteres PICTURE	**Resultado: elemento de datos positivo o cero**	**Resultado: elemento de datos negativo**
+	+	-
-	espacio	-
CR	2 espacios	CR
BD	2 espacios	BD

Por ejemplo:

IMAGEN	**Valor de los datos**	**Resultado editado**
`999.99+`	`+6555.556`	`555.55+`
`+9999.99`	`-6555.555`	`-6555.55`
`9999.99`	`+1234.56`	`1234.56`
`$999.99`	`-123.45`	`$123.45`
`-$999.99`	`-123.456`	`-$123.45`
`-$999.99`	`+123.456`	`$123.45`
`$9999.99CR`	`+123.45`	`$0123.45`
`$9999.99CR`	`-123.45`	`$0123.45CR`

Edición de inserción flotante

La edición de inserción flotante sólo es válida para elementos editados numérica-.

Se utilizan los símbolos siguientes:

```
cs + -
```

Dentro de una serie de caracteres PICTURE, estos símbolos se excluyen mutuamente como caracteres de inserción flotante.

La edición de inserción flotante se especifica utilizando una serie de al menos dos de los símbolos de inserción flotante permitidos para representar las posiciones de caracteres más a la izquierda en las que se pueden insertar los caracteres reales.

El símbolo de inserción flotante situado más a la izquierda de la serie de caracteres representa el límite situado más a la izquierda en el que el carácter real puede aparecer en el elemento de datos. El símbolo de inserción flotante situado más a la derecha representa el límite situado más a la derecha en el que puede aparecer el carácter real.

El segundo símbolo de inserción flotante situado más a la izquierda de la serie de caracteres representa el límite situado más a la izquierda en el que pueden aparecer datos numéricos dentro del elemento de datos. Los datos numéricos distintos de cero pueden sustituir todos los caracteres en o a la derecha de este límite.

Cualquier símbolo de inserción simple (B 0/,) dentro o a la derecha inmediata de la serie de símbolos de inserción flotante se considera parte de la serie de caracteres flotante. Si el símbolo de inserción especial de punto (.) se incluye dentro de la serie flotante, se considera que forma parte de la serie de caracteres.

Para evitar el truncamiento, el tamaño mínimo de la serie de caracteres PICTURE debe ser:

- El número de posiciones de carácter en el elemento de envío, más
- El número de símbolos de inserción no flotantes en el elemento de recepción, más
- Una posición de carácter para el símbolo de inserción flotante

Representación de la edición de inserción flotante

En una serie de caracteres PICTURE, hay dos formas de representar la edición de inserción flotante y, por lo tanto, dos formas en las que se realiza la edición:

1. Cualquiera o todas las posiciones de caracteres numéricos iniciales a la izquierda de la coma decimal se representan mediante el símbolo de inserción flotante. Cuando se realiza la edición, se coloca un único carácter de inserción flotante a la izquierda inmediata del primer dígito distinto de cero en los datos, o de la coma decimal, el que esté más a la izquierda. Las posiciones de carácter a la izquierda del carácter insertado se rellenan con espacios.

   Si todas las posiciones de caracteres numéricos de la serie de caracteres PICTURE están representadas por el carácter de inserción, al menos uno de los caracteres de inserción debe estar a la izquierda de la coma decimal.
2. Todas las posiciones de caracteres numéricos se representan mediante el símbolo de inserción flotante. Cuando se realiza la edición, entonces:
   - Si el valor de los datos es cero, el elemento de datos completo contendrá espacios.
   - Si el valor de los datos es distinto de cero, el resultado es el mismo que en la regla 1.

Por ejemplo:

IMAGEN	Valor de los datos	Resultado editado
$$$$.99	.123	$.12

IMAGEN	Valor de los datos	Resultado editado
$$$9.99	.12	$0.12
$,$$$,999.99	-1234.56	$1,234.56
+,+++,999.99	-123456.789	-123,456.78
$$,$$$,$$$.99CR	-1234567	$1,234,567.00CR
++,+++,+++.+++	0000.00	

Supresión de ceros y edición de sustitución

La supresión de ceros y la edición de sustitución sólo es válida para elementos editados numéricos.

En la edición de supresión de ceros, se utilizan los símbolos Z y *. Estos símbolos son mutuamente excluyentes en una serie de caracteres PICTURE.

Los símbolos siguientes se excluyen mutuamente como símbolos de sustitución flotante en una serie de caracteres PICTURE:

Z * +- *cs*

Especifique la supresión de ceros y la edición de sustitución con una serie de uno o más de los símbolos permitidos para representar las posiciones de caracteres más a la izquierda en las que se puede realizar la supresión de ceros y la edición de sustitución.

Los símbolos de inserción simples (B 0/,) dentro o a la derecha inmediata de la serie de símbolos de edición flotante se consideran parte de la serie. Si el símbolo de inserción especial de punto (.) se incluye dentro de la serie de edición flotante, se considera que forma parte de la serie de caracteres.

Representación de la supresión de ceros

En una serie de caracteres PICTURE, hay dos formas de representar la supresión de ceros, y dos formas en las que se realiza la edición:

1. Cualquiera o todas las posiciones de caracteres numéricos iniciales a la izquierda de la coma decimal se representan mediante símbolos de supresión. Cuando se realiza la edición, el carácter de sustitución sustituye cualquier cero inicial en los datos que aparecen en la misma posición de carácter que un símbolo de supresión. La supresión se detiene en el carácter situado más a la izquierda:
   - Que no corresponde a un símbolo de supresión
   - Contiene datos distintos de cero
   - Es decir, la coma decimal
2. Todas las posiciones de caracteres numéricos de la serie de caracteres PICTURE se representan mediante los símbolos de supresión. Cuando se realiza la edición y el valor de los datos es distinto de cero, el resultado es el mismo que en la regla anterior. Si el valor de los datos es cero, entonces:
   - Si se ha especificado Z, todo el elemento de datos contendrá espacios.
   - Si se ha especificado *, todo el elemento de datos excepto la coma decimal real contendrá asteriscos.

Por ejemplo:

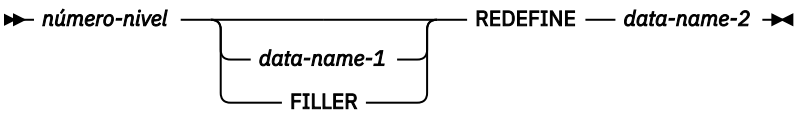
IMAGEN	Valor de los datos	Resultado editado
****.**	0000.00	****.**
ZZZZ.ZZ	0000.00	
ZZZZ.99	0000.00	.00
****.99	0000.00	****.00
ZZ99.99	0000.00	00.00
Z,ZZZ.ZZ+	+123.456	123.45+
*,***.**+	-123.45	**123.45-
,*,***.**+	+12345678.9	12,345,678.90+
$Z,ZZZ,ZZZ.ZZCR	+12345.67	$ 12,345.67
$B*,***,***.**BBDB	-12345.67	$ ***12,345.67 DB

No especifique el asterisco (*) como símbolo de supresión y la cláusula BLANK WHEN ZERO para la misma entrada.

cláusula REDEFINES

La cláusula REDEFINES le permite utilizar distintas entradas de descripción de datos para describir la misma área de almacenamiento del sistema.

Formato



(*número-nivel*, *data-name-1*y FILLER no forman parte de la cláusula REDEFINES y se incluyen en el formato sólo para mayor claridad.)

Cuando se especifica, la cláusula REDEFINES debe ser la primera entrada después de *data-name-1* o FILLER. Si no se especifica *data-name-1* o FILLER, la cláusula REDEFINES debe ser la primera entrada después del número de nivel.

***data-name-1*, FILLER**
Identifica una descripción alternativa para el área de datos identificada por *data-name-2*; *data-name-1* es el elemento *redefine* o el asunto REDEFINES .

Ni *data-name-1* ni ninguna de sus entradas subordinadas puede contener una cláusula VALUE o TYPE. *data-name-1* no debe contener una cláusula TYPEDEF.

data-name-2
Identifica el elemento *redefinido* o el objeto REDEFINES .

La entrada de descripción de datos para *data-name-2* puede contener una cláusula REDEFINES.

La entrada de descripción de datos para *data-name-2* no puede contener una cláusula OCCURS. Sin embargo, *data-name-2* puede estar subordinado a un elemento cuya entrada de descripción de datos contenga una cláusula OCCURS; en este caso, la referencia a *data-name-2* en la cláusula REDEFINES no debe tener subíndices.

La entrada redefinida y las entradas subordinadas no deben contener una cláusula TYPE.

Ni *data-name-1* ni *data-name-2* pueden contener una cláusula OCCURS DEPENDING ON.

data-name-1 y *data-name-2* deben tener el mismo nivel en la jerarquía; sin embargo, no es necesario que los números de nivel sean los mismos. Ni *data-name-1* ni *data-name-2* se pueden definir con el número de nivel 66 u 88.

data-name-1 y *data-name-2* se pueden describir con cualquier uso.

La redefinición empieza en *data-name-1* y finaliza cuando se encuentra un número de nivel menor o igual que el de *data-name-1* . Entre estas entradas no puede aparecer ninguna entrada que tenga un número de nivel numéricamente inferior a los de *data-name-1* y *data-name-2* . En el ejemplo siguiente:

```
05  A PICTURE X(6).
05  B REDEFINES A.
    10 B-1          PICTURE X(2).
    10 B-2          PICTURE 9(4).
05  C               PICTURE 99V99.
```

A es el elemento redefinido y B es el elemento redefinido. La redefinición empieza por B e incluye los dos elementos subordinados B-1 y B-2. La redefinición finaliza cuando se encuentra el elemento level-05 C .

Si se utiliza la cláusula GLOBAL en la entrada de descripción de datos que contiene la cláusula REDEFINES, sólo *data-name-1* (el elemento de redefinición) posee el atributo global. Por ejemplo, en la descripción siguiente, sólo el elemento B1 posee el atributo GLOBAL:

```
01 A1 PICTURE X(6).
01 B1 REDEFINES A1 GLOBAL PICTURE X(4).
```

La cláusula EXTERNAL no debe especificarse en la misma entrada de descripción de datos que una cláusula REDEFINES.

Si el elemento de datos redefinido (*data-name-2*) se declara como un registro de datos externo, el tamaño del elemento de datos redefinido (*data-name-1*) no debe ser mayor que el tamaño del elemento de datos redefinido. Si el elemento de datos redefinido no se declara como un registro de datos externo, no hay ninguna restricción de este tipo.

El ejemplo siguiente muestra que el elemento de redefinición, B2, puede ocupar más almacenamiento que el elemento redefinido, A2. El tamaño de almacenamiento para la cláusula REDEFINED se determina

en número de bytes. El elemento A2 ocupa 6 bytes de almacenamiento y el elemento B2, un elemento de datos de categoría nacional, ocupa 8 bytes de almacenamiento.

```
01 A2 PICTURE X(6).
01 B2 REDEFINES A2 GLOBAL PICTURE N(4).
```

Se permiten una o más redefiniciones de la misma área de almacenamiento. Las entradas que proporcionan las nuevas descripciones del área de almacenamiento deben ir inmediatamente después de la descripción del área redefinida sin intervenir entradas que definan nuevas posiciones de caracteres. Varias redefiniciones pueden, pero no es necesario, utilizar el nombre de datos de la entrada original que ha definido esta área de almacenamiento. Por ejemplo:

```
05  A               PICTURE 9999.
05  B REDEFINES A   PICTURE 9V999.
05  C REDEFINES A   PICTURE 99V99.
```

Además, varias redefiniciones pueden utilizar el nombre de la definición anterior tal como se muestra en el ejemplo siguiente:

```
05  A               PICTURE 9999.
05  B REDEFINES A   PICTURE 9V999.
05  C REDEFINES B   PICTURE 99V99.
```

Cuando se escribe más de una entrada level-01 subordinada a una entrada FD, se produce una condición conocida como *redefinición implícita* . Es decir, la segunda entrada level-01 redefine implícitamente el almacenamiento asignado para la primera entrada. En tales entradas level-01 , no debe especificarse la cláusula REDEFINES.

Cuando se escribe más de una entrada level-01 subordinada a una entrada FD (y la entrada level-01 no es un nombre de tipo), se produce una condición conocida como *redefinición implícita* . Es decir, la segunda entrada level-01 redefine implícitamente el almacenamiento asignado para la primera entrada. En tales entradas level-01 , no se puede especificar la cláusula REDEFINES ni la cláusula TYPE. Además, la cláusula TYPE no debe especificarse en ningún elemento subordinado a ninguna de las entradas level-01 .

Cuando el elemento de datos redefine implícitamente varios registros de nivel 01 en una entrada de descripción de archivo (FD), los elementos subordinados al elemento redefinido o redefinido pueden contener una cláusula OCCURS DEPENDING ON.

Consideraciones sobre la cláusula REDEFINES

El tema enumera consideraciones sobre el uso de la cláusula REDEFINES.

Cuando se redefine un área, todas las descripciones del área siempre están en vigor; es decir, la redefinición no reemplaza a una descripción anterior. Por lo tanto, si se ha especificado `B REDEFINES C` , cualquiera de las dos sentencias de procedimiento `MOVE X TO B` o `MOVE Y TO C` podría ejecutarse en cualquier punto del programa. En el primer caso, el área descrita como B recibiría el valor y el formato de X. En el segundo caso, la misma área física (descrita ahora como C) recibiría el valor y el formato de Y. Tenga en cuenta que si la segunda sentencia se ejecuta inmediatamente después de la primera, el valor de Y sustituye el valor de X en un área de almacenamiento.

No es necesario que el uso de un elemento de datos de redefinición sea el mismo que el de un elemento redefinido. Sin embargo, esto no provoca ningún cambio en el formato o contenido de los datos existentes. Por ejemplo:

```
05  B                   PICTURE  99 USAGE DISPLAY VALUE 8.
05  C REDEFINES B       PICTURE S99 USAGE COMPUTATIONAL-4.
05  A                   PICTURE S99 USAGE COMPUTATIONAL-4.
```

La redefinición de B no cambia la configuración de bits de los datos en el área de almacenamiento. Por lo tanto, las dos sentencias siguientes producen resultados diferentes:

```
ADD B TO A
ADD C TO A
```

En el primer caso, el valor 8 se añade a A (porque B tiene USAGE DISPLAY). En la segunda sentencia, el valor -3848 se añade a A (porque C tiene USAGE COMPUTATIONAL-4) y la configuración de bits del área de almacenamiento tiene el valor binario -3848. Este ejemplo muestra cómo el uso inadecuado de la redefinición puede dar resultados inesperados o incorrectos.

Ejemplos de la cláusula REDEFINES

La cláusula REDEFINES se puede especificar para un elemento dentro del ámbito de (subordinado a) un área que se redefine.

En el ejemplo siguiente, WEEKLY-PAY redefine SEMI-MONTHLY-PAY (que está dentro del ámbito de REGULAR-EMPLOYEE, mientras que REGULAR-EMPLOYEE se redefine mediante TEMPORARY-EMPLOYEE).

```
05  REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  GRADE                         PICTURE X(4).
  10  SEMI-MONTHLY-PAY              PICTURE 9999V99.
  10  WEEKLY-PAY REDEFINES SEMI-MONTHLY-PAY
                                    PICTURE 999V999.
05  TEMPORARY-EMPLOYEE REDEFINES REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  FILLER                        PICTURE X(6).
  10  HOURLY-PAY                    PICTURE 99V99.
```

La cláusula REDEFINES también se puede especificar para un elemento subordinado a un elemento de redefinición, tal como se muestra para CODE-H REDEFINES HOURLY-PAY en el ejemplo siguiente:

```
05  REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  GRADE                         PICTURE X(4).
  10  SEMI-MONTHLY-PAY              PICTURE 999V999.
05  TEMPORARY-EMPLOYEE REDEFINES REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  FILLER                        PICTURE X(6).
  10  HOURLY-PAY                    PICTURE 99V99.
  10  CODE-H REDEFINES HOURLY-PAY    PICTURE 9999.
```

Los elementos de datos dentro de un área se pueden redefinir sin cambiar sus longitudes. Por ejemplo:

```
05  NAME-2.
  10  SALARY              PICTURE XXX.
  10  SO-SEC-NO           PICTURE X(9).
  10  MONTH               PICTURE XX.
05  NAME-1 REDEFINES NAME-2.
  10  WAGE                PICTURE XXX.
  10  EMP-NO              PICTURE X(9).
  10  YEAR                PICTURE XX.
```

Las longitudes y tipos de elementos de datos también se pueden volver a especificar dentro de un área. Por ejemplo:

```
05  NAME-2.
  10  SALARY              PICTURE XXX.
  10  SO-SEC-NO           PICTURE X(9).
  10  MONTH               PICTURE XX.
05  NAME-1 REDEFINES NAME-2.
  10  WAGE                PICTURE 999V999.
  10  EMP-NO              PICTURE X(6).
  10  YEAR                PICTURE XX.
```

Los elementos de datos también se pueden volver a especificar con una longitud mayor que la longitud del elemento redefinido. Por ejemplo:

```
05  NAME-2.
  10  SALARY              PICTURE XXX.
  10  SO-SEC-NO           PICTURE X(9).
  10  MONTH               PICTURE XX.
05  NAME-1 REDEFINES NAME-2.
  10  WAGE                PICTURE 999V999.
  10  EMP-NO              PICTURE X(6).
  10  YEAR                PICTURE X(4).
```

Esto no cambia la longitud del elemento redefinido `NAME-2`.

Resultados no definidos

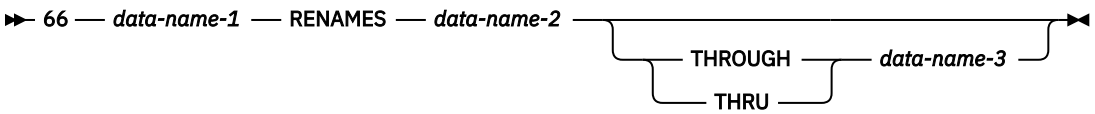
Los resultados no definidos pueden producirse en las condiciones tal como se listan en el tema.

- Un elemento de redefinición se mueve a un elemento redefinido (es decir, si se ejecuta `B REDEFINES C` y la sentencia `MOVE B TO C`).
- Un elemento redefinido se mueve a un elemento de redefinición (es decir, si se ejecuta `B REDEFINES C` y la sentencia `MOVE C TO B`).

cláusula RENAMES

La cláusula RENAMES especifica agrupaciones alternativas y posiblemente solapadas de elementos de datos elementales.

Formato



Debe especificarse el número de nivel especial 66 para las entradas de descripción de datos que contienen la cláusula RENAMES. (El número de nivel 66 y *data-name-1* no forman parte de la cláusula RENAMES y se incluyen en el formato sólo para mayor claridad.)

Se pueden escribir una o más entradas RENAMES para un registro lógico. Todas las entradas RENAMES asociadas a un registro lógico deben ir inmediatamente después de la última entrada de descripción de datos de ese registro.

data-name-1
Identifica una agrupación alternativa de elementos de datos.

Una entrada level-66 no puede renombrar una entrada level-01, level-77, level-88u otra entrada level-66 .

data-name-1 no se puede utilizar como calificador; sólo se puede calificar mediante los nombres de las entradas de indicador de nivel o las entradas level-01 .

data-name-2, data-name-3
Identifique la agrupación original de elementos de datos elementales; es decir, deben nombrar elementos elementales o de grupo dentro de la entrada level-01 asociada y no deben ser el mismo nombre de datos. Ambos nombres de datos se pueden calificar.

data-name-2 y *data-name-3* pueden hacer referencia a cualquiera de los elementos siguientes:

- Un elemento de datos elemental
- Un elemento de grupo alfanumérico
- Un elemento de grupo nacional

Cuando *data-name-2* o *data-name-3* hace referencia a un elemento de grupo nacional, el elemento referenciado se procesa como un grupo (no como un elemento de datos elemental de categoría nacional).

La cláusula OCCURS no debe especificarse en las entradas de datos para *data-name-2* y *data-name-3*, o para cualquier entrada de grupo a la que estén subordinadas. Además, la cláusula OCCURS DEPENDING no debe especificarse para ningún elemento definido entre *data-name-2* y *data-name-3*.

La cláusula TYPE no debe especificarse en las descripciones de datos de *data-name-2*, *data-name-3*, y elementos definidos entre *data-name-2*, *data-name-3*o cualquier subordinado de estos elementos. Si *data-name-2*, *data-name-3*, o cualquier elemento definido entre *data-name-2* y *data-name-3* está subordinado a un elemento de grupo definido mediante la cláusula TYPE, entonces *data-name-1* debe estar subordinado al mismo elemento de grupo.

Las palabras clave THROUGH y THRU son equivalentes.

Cuando se especifica la frase THROUGH:

- *data-name-1* define un elemento de grupo alfanumérico que incluye todos los elementos elementales que:
  - Empiece por *data-name-2* si es un elemento elemental, o el primer elemento elemental dentro de *data-name-2* si es un elemento de grupo
  - Finalice con *data-name-3* si es un elemento elemental, o el último elemento elemental dentro de *data-name-3* si es un elemento de grupo alfanumérico o elemento de grupo nacional
- El área de almacenamiento ocupada por el elemento inicial a través del elemento final se convierte en el área de almacenamiento ocupada por *data-name-1*.

Nota de uso: El grupo definido con la frase THROUGH puede incluir elementos de datos de uso NATIONAL.

La posición de carácter más a la izquierda en *data-name-3* no debe preceder a la posición de carácter más a la izquierda en *data-name-2*, y la posición del carácter situado más a la derecha en *data-name-3* no debe preceder a la posición del carácter situado más a la derecha en *data-name-2*. Esto significa que *data-name-3* no puede estar totalmente subordinado a *data-name-2*.

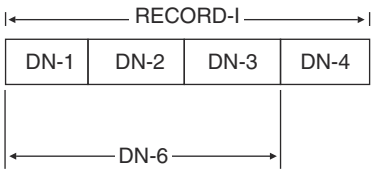
Cuando no se especifica la frase THROUGH:

- El área de almacenamiento ocupada por *data-name-2* se convierte en el área de almacenamiento ocupada por *data-name-1*.
- Todos los atributos de datos de *data-name-2* se convierten en los atributos de datos de *data-name-1*. Es decir:
  - Cuando *data-name-2* es un elemento de grupo alfanumérico, *data-name-1* es un elemento de grupo alfanumérico.
  - Cuando *data-name-2* es un elemento de grupo nacional, *data-name-1* es un elemento de grupo nacional.
  - Cuando *data-name-2* es un elemento elemental, *data-name-1* es un elemento elemental.

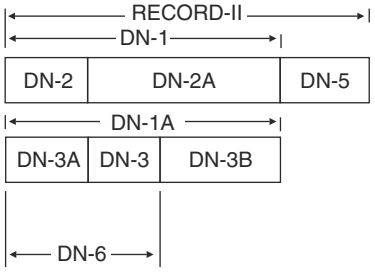
La figura siguiente ilustra las especificaciones de cláusula RENAMES válidas y no válidas.

COBOL Specifications **Storage Layouts**

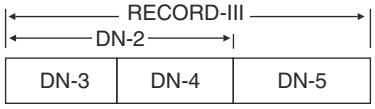
Example 1 (Valid)
```
01   RECORD-I.
     05 DN-1... .
     05 DN-2... .
     05 DN-3... .
     05 DN-4... .
66   DN-6 RENAMES DN-1 THROUGH DN-3.
```



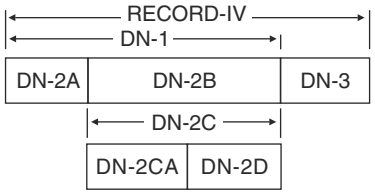
Example 2 (Valid)
```
01   RECORD-II.
     05 DN-1.
        10 DN-2... .
        10 DN-2A... .
     05 DN-1A REDEFINES DN-1.
        10 DN-3A... .
        10 DN-3... .
        10 DN-3B... .
     05 DN-5... .
66   DN-6 RENAMES DN-2 THROUGH DN-3.
```



Example 3 (Invalid)
```
01   RECORD-III.
     05 DN-2.
        10 DN-3... .
        10 DN-4... .
     05 DN-5... .
66   DN-6 RENAMES DN-2 THROUGH DN-3.    DN-6 is indeterminate
```



Example 4 (Invalid)
```
01   RECORD-IV.
     05 DN-1.
        10 DN-2A... .
        10 DN-2B... .
        10 DN-2C REDEFINES DN-2B.
           15 DN-2CA... .
     15 DN-2D... .
     05 DN-3... .
66   DN-4 RENAMES DN-1 THROUGH DN-2CA.  DN-4 is indeterminate
```

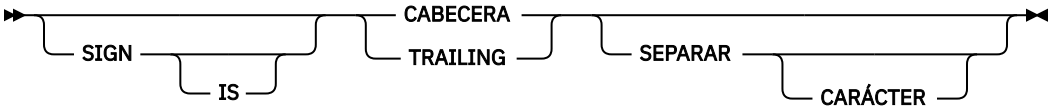


cláusula SIGN

La cláusula SIGN especifica la posición y la modalidad de representación del signo operativo para el elemento numérico con signo al que se aplica.

La cláusula SIGN sólo es necesaria cuando es necesaria una descripción explícita de las propiedades o posición del signo operativo.

Formato



La cláusula SIGN sólo se puede especificar para los elementos siguientes:

- Un elemento de datos numéricos elementales de uso DISPLAY o NATIONAL que se describe con una S en su serie de caracteres de imagen, o
- Un elemento de grupo que contiene al menos una entrada elemental como un elemento subordinado

Cuando se especifica la cláusula SIGN a nivel de grupo, dicha cláusula SIGN sólo se aplica a los elementos de datos elementales numéricos con signo subordinados de uso DISPLAY o NATIONAL. Un grupo de este tipo también puede contener elementos que no se ven afectados por la cláusula SIGN. Si se

especifica la cláusula SIGN para un grupo o una entrada elemental que está subordinada a un elemento de grupo que tiene una cláusula SIGN, la cláusula SIGN para la entrada subordinada tiene prioridad para dicha entrada subordinada.

La cláusula SIGN se trata como documentación para elementos de coma flotante externos.

Cuando se especifica la cláusula SIGN sin la frase SEPARADA, USAGE DISPLAY debe especificarse explícita o implícitamente. Cuando se especifica SIGN IS SEPARATE, se puede especificar USAGE DISPLAY o USAGE NATIONAL.

Si especifica la cláusula CODE-SET en una entrada FD, las entradas de descripción de datos numéricos con signo asociadas a esa entrada de descripción de archivo deben describirse con la cláusula SIGN IS SEPARATE.

Si no se especifica la frase SEPARADAS CHARACTER, entonces:

- Se supone que el signo operativo está asociado con la posición de dígito INICIAL o TRAILING, la que se especifique, del elemento de datos numéricos elementales. (En este caso, la especificación de SIGN IS TRAILING es el equivalente a la acción estándar del compilador.)
- El carácter S de la serie de caracteres PICTURE no se cuenta al determinar el tamaño del elemento (en términos de caracteres de formato de datos estándar).

Si se especifica la frase SEPARADAS CHARACTER, entonces:

- Se presupone que el signo operativo es la posición de carácter INICIAL o TRAILING, la que se especifique, del elemento de datos numérico elemental. Esta posición de carácter no es una posición de dígito.
- El carácter S de la serie de caracteres PICTURE se cuenta al determinar el tamaño del elemento de datos (en términos de caracteres de formato de datos estándar).
- + es el carácter utilizado para el signo operativo positivo.
- -es el carácter utilizado para el signo operativo negativo.

La frase SEPARADAS CHARACTER no se puede especificar para un campo de fecha.

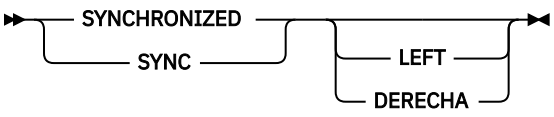
No se puede especificar la cláusula SIGN si se especifica la cláusula FORMAT.

La cláusula TYPE no puede especificarse en la misma entrada de descripción de datos que la cláusula SIGN.

cláusula SYNCHRONIZED

La cláusula SYNCHRONIZED especifica la alineación de un elemento elemental en un límite natural en el almacenamiento.

Formato



SYNC es una abreviatura de SYNCHRONIZED y tiene el mismo significado.

La cláusula SYNCHRONIZED nunca es necesaria, pero puede mejorar el rendimiento en algunos sistemas para los elementos binarios utilizados en la aritmética.

La cláusula SYNCHRONIZED puede especificarse para elementos elementales y para elementos de grupo level-01 , en cuyo caso cada elemento elemental del elemento de grupo se sincroniza.

LEFT
Especifica que el elemento elemental debe situarse de forma que empiece en la posición de carácter izquierdo del límite natural en el que se coloca el elemento elemental.

RIGHT
Especifica que el elemento elemental debe situarse de forma que termine en la posición de carácter derecho del límite natural en el que se ha colocado.

Cuando se especifica, las frases LEFT y RIGHT se comprueban con la sintaxis pero no tienen ningún efecto en la ejecución del programa.

La longitud de un elemento elemental no se ve afectada por la cláusula SYNCHRONIZED.

La tabla siguiente lista el efecto de la cláusula SYNCHRONIZE en otros elementos de lenguaje.

Tabla 21. ***Efecto de la cláusulaSYNCHRONIZE en otros elementos de lenguaje***

Elemento de idioma	**Comentarios**
cláusula OCCURS	Cuando se especifica para un elemento dentro del ámbito de una cláusula OCCURS, cada aparición del elemento se sincroniza.
USAGE DISPLAY o PACKED-DECIMAL	Se comprueba la sintaxis de cada elemento, pero la cláusula SYNCHRONIZED no tiene ningún efecto en la ejecución.
USO NACIONAL	Se comprueba la sintaxis de cada elemento, pero la cláusula SYNCHRONIZED no tiene ningún efecto en la ejecución.
USAGE BINARY o COMPUTATIONAL	Cuando el elemento es el primer elemento elemental subordinado a un elemento que contiene una cláusula REDEFINES, el elemento no debe requerir la adición de posiciones de caracteres no utilizadas. Cuando no se especifica la cláusula sincronizada para un elemento de datos subordinado (uno con un número de nivel de 02 a 49): • El elemento se alinea en un desplazamiento que es un múltiplo de 2 relativo al principio del registro si su USAGE es BINARY y su PICTURE está en el rango de S9 a S9(4). • El elemento se alinea en un desplazamiento que es un múltiplo de 4 relativo al principio del registro si su USAGE es BINARY y su PICTURE está en el rango de S9(5) a S9(18), o su USAGE es INDEX. Cuando no se especifica SYNCHRONIZED para elementos binarios, no hay espacio reservado para bytes de holgura.
USAGE POINTER, PROCEDURE-POINTER, FUNCTION-POINTER	Los datos se alinean en un límite de palabra completa cuando se especifica la opción de compilador **ADDR(32)** . Los datos se alinean en un límite de doble palabra cuando se especifica la opción de compilador **ADDR(64)** .
USO COMPUTATIONAL-1	Los datos se alinean en un límite de palabra completa.
USO COMPUTATIONAL-2	Los datos se alinean en un límite de doble palabra.
USO COMPUTATIONAL-3	Los datos se tratan igual que la cláusula SYNCHRONIZED para un elemento PACKED-DECIMAL.
USO COMPUTATIONAL-4	Los datos se tratan igual que la cláusula SYNCHRONIZED para un elemento COMPUTATIONAL.
USO COMPUTATIONAL-5	Los datos se tratan igual que la cláusula SYNCHRONIZED para un elemento COMPUTATIONAL.
DBCS y elementos de coma flotante externos	Se comprueba la sintaxis de cada elemento, pero la cláusula SYNCHRONIZED no tiene ningún efecto en la ejecución.

*Tabla 21. **Efecto de la cláusulaSYNCHRONIZE en otros elementos de lenguaje** (continuación)*	
Elemento de idioma	**Comentarios**

cláusula REDEFINES	Para un elemento que contiene una cláusula REDEFINES, el elemento de datos que se redefine debe tener la alineación de límite adecuada para el elemento de datos que lo redefine. Por ejemplo, si escribe lo siguiente, asegúrese de que el elemento de datos A empieza en un límite de palabra completa:
	`02 A              PICTURE X(4).`
	`02 B REDEFINES A  PICTURE S9(9) BINARY SYNC.`

cláusula FORMAT	Se comprueba la sintaxis de cada elemento, pero la cláusula SYNCHRONIZED no tiene ningún efecto en la ejecución.

En FILE SECTION, el compilador presupone que todos los registros level-01 que contienen elementos SYNCHRONIZED están alineados en límites de palabra doble en el almacenamiento intermedio. Debe proporcionar los bytes de holgura necesarios entre registros para garantizar la alineación cuando hay varios registros en un bloque.

En la WORKING-STORAGE SECTION, el compilador alinea todas las entradas level-01 en un límite de palabra doble.

Para alinear elementos binarios en la SECCIÓN LINKAGE, se supone que todos los elementos level-01 empiezan en límites de doble palabra. Por lo tanto, si emite una sentencia CALL, dichos operandos de cualquier frase USING dentro de ella deben alinearse de forma correspondiente.

La cláusula SYNCHRONIZED no puede especificarse en la misma entrada de descripción de datos que la cláusula TYPE.

Bytes de holgura

Hay dos tipos de bytes de holgura.

- Bytes de holgura *dentro* de registros: posiciones de caracteres no utilizadas que preceden a cada elemento sincronizado del registro
- Bytes de holgura *entre* registros: posiciones de caracteres no utilizadas añadidas entre registros lógicos bloqueados

Bytes de holgura dentro de registros

Para cualquier descripción de datos que tenga elementos binarios que no estén en sus límites naturales, el compilador inserta bytes de holgura dentro de un registro para asegurarse de que todos los elementos SYNCHRONIZED están en sus límites adecuados.

Puesto que es importante que conozca la longitud de los registros de un archivo, debe determinar si se necesitan bytes de holgura y, si es así, cuántos bytes añadirá el compilador. El algoritmo que utiliza el compilador es el siguiente:

- El número total de bytes ocupados por todos los elementos de datos elementales que preceden al elemento binario se añaden juntos, incluidos los bytes de holgura que se han añadido anteriormente.
- Esta suma se divide por *m*, donde:
  - *m* = 2 para elementos binarios de longitud de cuatro dígitos o menos
  - *m* = 4 para elementos binarios de longitud de cinco dígitos o más y para elementos de datos COMPUTATIONAL-1
  - *m* = 4 u 8 para elementos de datos descritos con USAGE INDEX, USAGE POINTER, USAGE PROCEDURE-POINTER, o USAGE FUNCTION-POINTER
  - *m* = 8 para elementos de datos COMPUTATIONAL-2

- Si el resto (*r*) de esta división es igual a cero, no se necesitan bytes de holgura. Si el resto no es igual a cero, el número de bytes de holgura que se deben añadir es igual a *m* - *r*.

Estos bytes de holgura se añaden a cada registro inmediatamente después del elemento de datos elemental que precede al elemento binario. Se definen como si constituyeran un elemento con un número de nivel igual al del elemento elemental que precede inmediatamente al elemento binario SYNCHRONIZED y se incluyen en el tamaño del grupo que los contiene.

Por ejemplo:

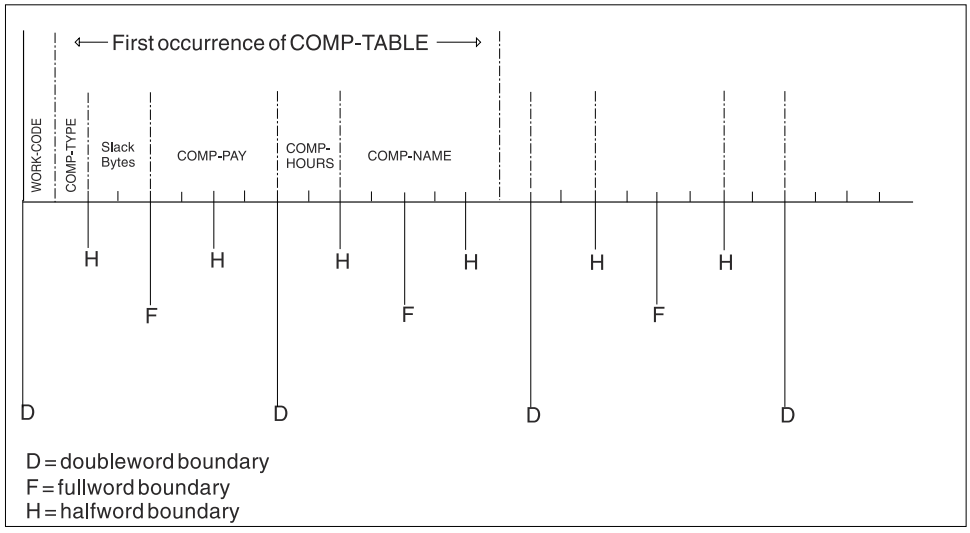
```
01  FIELD-A.
    05  FIELD-B                    PICTURE X(5).
    05  FIELD-C.
        10  FIELD-D                PICTURE XX.
       [10  SLACK-BYTES             PICTURE X.  INSERTED BY COMPILER]
        10  FIELD-E  COMPUTATIONAL  PICTURE S9(6) SYNC.
01  FIELD-L.
    05  FIELD-M                    PICTURE X(5).
    05  FIELD-N                    PICTURE XX.
   [05  SLACK-BYTES                 PICTURE X.  INSERTED BY COMPILER]
    05  FIELD-O.
        10  FIELD-P COMPUTATIONAL   PICTURE S9(6) SYNC.
```

El compilador también puede añadir bytes de holgura cuando un elemento de grupo se define con una cláusula OCCURS y contiene un elemento de datos binarios SYNCHRONIZED. Para determinar si se van a añadir bytes de holgura, se realiza la siguiente acción:

- El compilador calcula el tamaño del grupo, incluidos todos los bytes de holgura necesarios dentro de un registro.
- Esta suma se divide por la mayor *m* necesaria para cualquier elemento elemental del grupo.
- Si *r* es igual a cero, no se necesitan bytes de holgura. Si *r* no es igual a cero, se deben añadir *m* - *r* bytes de holgura.

Los bytes de holgura se insertan al final de cada aparición del elemento de grupo que contiene la cláusula OCCURS. Por ejemplo, un registro definido como se indica a continuación aparece en el almacenamiento, tal como se muestra, en la figura posterior al registro:

```
01  WORK-RECORD.
    05  WORK-CODE                  PICTURE X.
    05  COMP-TABLE OCCURS 10 TIMES.
        10  COMP-TYPE              PICTURE X.
       [10  SLACK-BYTES             PIC XX.  INSERTED BY COMPILER]
        10  COMP-PAY               PICTURE S9(4)V99 COMP SYNC.
        10  COMP-HOURS             PICTURE S9(3) COMP SYNC.
        10  COMP-NAME              PICTURE X(5).
```

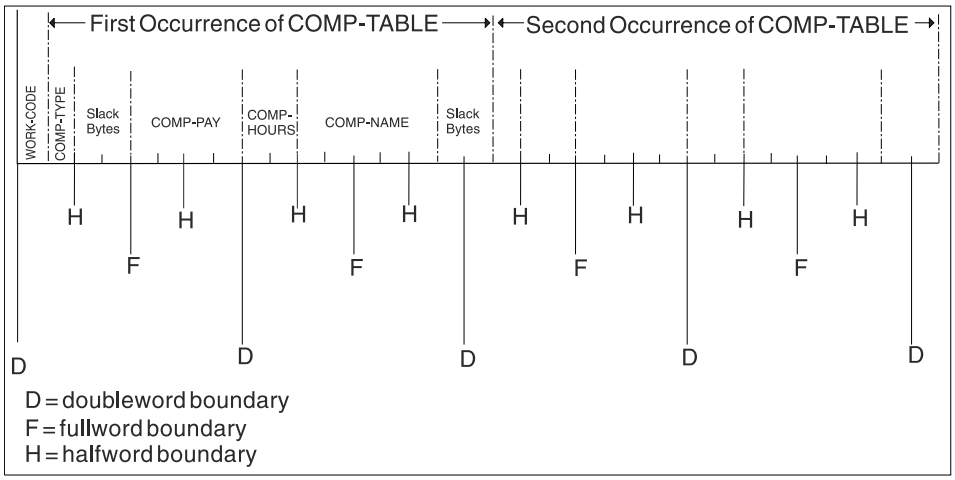


Para alinear COMP-PAY y COMP-HOURS en sus límites adecuados, el compilador ha añadido 2 bytes de holgura dentro del registro.

En el ejemplo anterior, sin ningún ajuste adicional, la segunda aparición de COMP-TABLE empezaría 1 byte antes de un límite de palabra doble, y la alineación de COMP-PAY y COMP-HOURS no sería válida para ninguna aparición de la tabla después de la primera. Por lo tanto, el compilador debe añadir bytes de holgura al final del grupo, como si el registro se hubiera escrito de la siguiente manera:

```
01  WORK-RECORD.
    05  WORK-CODE                  PICTURE X.
    05  COMP-TABLE OCCURS 10 TIMES.
        10  COMP-TYPE              PICTURE X.
       [10  SLACK-BYTES             PIC XX.  INSERTED BY COMPILER]
        10  COMP-PAY               PICTURE S9(4)V99 COMP SYNC.
        10  COMP-HOURS             PICTURE S9(3)  COMP SYNC.
        10  COMP-NAME              PICTURE X(5).
       [10  SLACK-BYTES             PIC XX.  INSERTED BY COMPILER]
```

En este ejemplo, la segunda y cada aparición siguiente de COMP-TABLE empieza 1 byte más allá de un límite de palabra doble. El diseño de almacenamiento para la primera aparición de COMP-TABLE ahora aparece como se muestra en la figura siguiente:



Cada aparición siguiente dentro de la tabla empezará ahora en la misma posición relativa que la primera.

Bytes de holgura entre registros

Se añaden las longitudes de todos los elementos de datos elementales del registro, incluidos todos los bytes de holgura. A continuación, el total se divide por el valor más alto de *m* para cualquiera de los elementos elementales del registro.

Si *r* (el resto) es igual a cero, no se necesitan bytes de holgura. Si *r* no es igual a cero, se necesitan *m* - *r* bytes de holgura. Estos bytes de holgura se pueden especificar escribiendo un FILLER level-02 al final del registro.

Tenga en cuenta la siguiente descripción de registro:

```
01  COMP-RECORD.
    05  A-1    PICTURE X(5).
    05  A-2    PICTURE X(3).
    05  A-3    PICTURE X(3).
    05  B-1    PICTURE S9999  USAGE COMP SYNCHRONIZED.
    05  B-2    PICTURE S99999 USAGE COMP SYNCHRONIZED.
    05  B-3    PICTURE S9999  USAGE COMP SYNCHRONIZED.
```

El número de bytes en A-1, A-2y A-3 totaliza 11. B-1 es un elemento COMPUTATIONAL de cuatro dígitos y, por lo tanto, se debe añadir 1 byte de holgura antes de B-1. Con este byte añadido, el número de bytes

que preceden a B-2 totaliza 14. Puesto que B-2 es un elemento COMPUTATIONAL de cinco dígitos de longitud, se deben añadir 2 bytes de holgura antes de él. No se necesitan bytes de holgura antes de B-3.

La entrada de descripción de registro revisada ahora aparece como:

```
01  COMP-RECORD.
    05  A-1            PICTURE X(5).
    05  A-2            PICTURE X(3).
    05  A-3            PICTURE X(3).
   [05  SLACK-BYTE-1    PICTURE X.   INSERTED BY COMPILER]
    05  B-1            PICTURE S9999 USAGE COMP SYNCHRONIZED.
   [05  SLACK-BYTE-2    PICTURE XX.  INSERTED BY COMPILER]
    05  B-2            PICTURE S99999 USAGE COMP SYNCHRONIZED.
    05  B-3            PICTURE S9999  USAGE COMP SYNCHRONIZED.
```

Hay un total de 22 bytes en COMP-RECORD, pero de las reglas anteriores, parece que *m* = 4 y *r* = 2. Por lo tanto, para lograr una alineación adecuada para los registros bloqueados, debe añadir 2 bytes de holgura al final del registro.

La entrada de descripción de registro final aparece como:

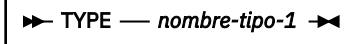
```
01  COMP-RECORD.
    05  A-1             PICTURE X(5).
    05  A-2             PICTURE X(3).
    05  A-3             PICTURE X(3).
   [05  SLACK-BYTE-1    PICTURE X.   INSERTED BY COMPILER]
    05  B-1             PICTURE S9999 USAGE COMP SYNCHRONIZED.
   [05  SLACK-BYTE-2    PICTURE XX.  INSERTED BY COMPILER]
    05  B-2             PICTURE S99999 USAGE COMP SYNCHRONIZED.
    05  B-3             PICTURE S9999  USAGE COMP SYNCHRONIZED.
    05  FILLER          PICTURE XX.  [SLACK BYTES YOU ADD]
```

cláusula TYPE

La cláusula TYPE indica que la descripción de datos del asunto de la entrada se especifica mediante un tipo de datos definido por el usuario.

El tipo de datos definido por el usuario se define utilizando la cláusula TYPEDEF, que se describe en "cláusula TYPEDEF" en la página 224.

Formato



Se aplican las siguientes normas generales:

- Si *type-name-1* (definido utilizando la cláusula TYPEDEF) describe un elemento de grupo, el asunto de la cláusula TYPE es un elemento de grupo cuyos elementos subordinados tienen los mismos nombres, descripciones, y jerarquías como los elementos subordinados de *type-name-1*.

  Nota: Puesto que el asunto de la cláusula TYPE puede tener un número de nivel tan alto como 49 y *type-name-1* puede ser un elemento de grupo con 49 niveles, el número de niveles de esta jerarquía puede superar los 49. De hecho, puesto que las descripciones de los nombres de tipo pueden hacer referencia a otros nombres de tipo, no hay ningún límite en el número de niveles de esta jerarquía.
- Si se especifica una cláusula VALUE en la descripción de datos del asunto de la cláusula TYPE, cualquier cláusula VALUE especificada en la descripción de *type-name-1* se ignora para esta entrada.
- Las reglas de ámbito para nombres de tipo son similares a las reglas de ámbito para nombres de datos.
- No se permite la modificación de referencia para un elemento elemental que es objeto de una cláusula TYPE.
- La descripción de *type-name-1*, incluidos sus elementos de datos subordinados, no puede contener una cláusula LIKE que haga referencia al asunto de la cláusula TYPE (que hace referencia a *type-name-1*), o cualquier elemento de grupo al que esté subordinado el sujeto de la cláusula TYPE.

- La descripción de *type-name-1*, incluidos sus elementos de datos subordinados, no puede contener una cláusula TYPE que haga referencia al registro al que está subordinado el sujeto de la cláusula TYPE (que hace referencia a *type-name-1*)

  Por ejemplo, A es un elemento de grupo definido utilizando la cláusula TYPEDEF. B es también un elemento de grupo definido utilizando la cláusula TYPEDEF, pero que también incluye un elemento subordinado de TYPE A. Siendo este el caso, la definición de tipo para A no puede incluir elementos de TYPE B.
- El asunto de una cláusula TYPE no se puede renombrar en su totalidad o en parte.
- El asunto de una cláusula TYPE no se puede redefinir explícita o implícitamente.
- Si el asunto de una cláusula TYPE está subordinado a un elemento de grupo, la descripción de datos del elemento de grupo no puede contener la cláusula USAGE.
- La cláusula TYPE no puede aparecer en una entrada de descripción de datos con la cláusula BLANK WHEN ZERO, FORMAT, JUSTIFIC, LIKE, PICTURE, REDEFINES, RENAMES, SIGN, SYNCHRONIZED o USAGE.
- La cláusula TYPE puede especificarse en una entrada de descripción de datos con las cláusulas EXTERNAL, GLOBAL, OCCURS, TYPEDEF y VALUE.

cláusula TYPEDEF

La cláusula TYPEDEF se utiliza para crear un nuevo tipo de datos definido por el usuario, nombre-tipo. El nombre del nuevo tipo de datos definido por el usuario es el sujeto de la cláusula TYPEDEF.

data-name-1 debe especificarse con la cláusula TYPEDEF: no se puede utilizar FILLER. La cláusula TYPEDEF debe ir inmediatamente después de *data-name-1*. Después de definir un nuevo tipo de datos utilizando la cláusula TYPEDEF, los elementos de datos pueden declararse como este nuevo tipo de datos utilizando la cláusula TYPE. Para obtener más información sobre la cláusula TYPE, consulte "cláusula TYPE" en la página 223.

Formato

La cláusula TYPEDEF sólo se puede especificar para entradas de nivel 01, que también pueden ser elementos de grupo. Si se especifica un elemento de grupo, todos los elementos subordinados del grupo pasan a formar parte de la declaración de tipo. No se ha asignado ningún almacenamiento para una declaración de tipo.

La cláusula TYPEDEF no se puede especificar en la misma entrada de descripción de datos que las cláusulas siguientes:

- externo
- REDEFINE
- SIMILAR

Todas las demás cláusulas de descripción de datos, si se especifican, son asumidas por cualquier elemento de datos que se haya definido utilizando el tipo de datos definido por el usuario (dentro de la cláusula TYPE).

TYPEDEF no se puede utilizar con OCCURS DEPENDING ON complejos. Esto significa que no puede especificar una cláusula OCCURS DEPENDING ON dentro de una tabla que forme parte de TYPEDEF. Para obtener más información, consulte *APARICIONES COMPLEJAS EN FUNCIÓN DE* en la publicación *COBOL for Linux en x86 Guía de programación*.

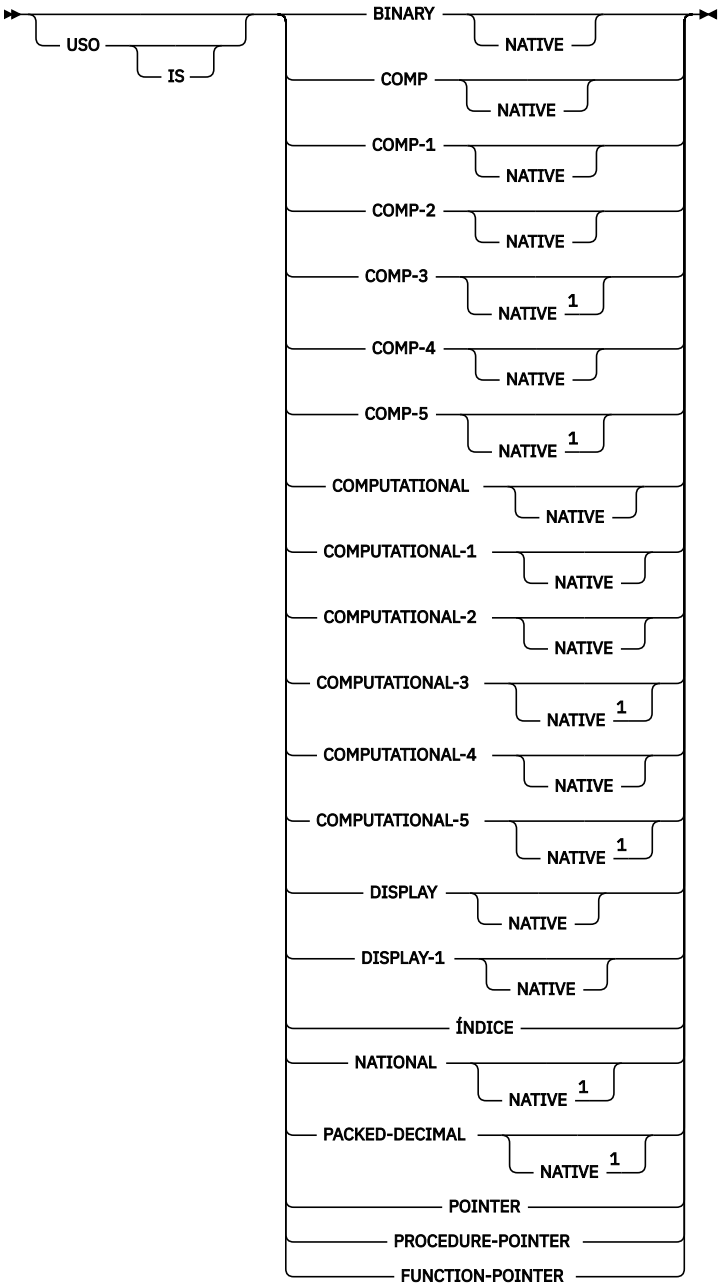
La cláusula TYPEDEF sólo puede especificarse en las secciones WORKING-STORAGE, LOCAL-STORAGE, LINKAGE o FILE de un programa.

La cláusula TYPE puede especificarse en la misma entrada de descripción de datos que la cláusula TYPEDEF.

cláusula USAGE

La cláusula USAGE especifica el formato en el que se representan los datos en el almacenamiento.

Formato 1



Notas:

[1] La frase NATIVE se trata como un comentario para los elementos de datos COMP-3, COMPUTATIONAL-3, COMP-5, COMPUTATIONAL-5, NATIONAL y PACKED-DECIMAL.

La cláusula USAGE puede especificarse para una entrada de descripción de datos con cualquier número de nivel que no sea 66 u 88.

Cuando se especifica a nivel de grupo, la cláusula USAGE se aplica a cada elemento elemental del grupo. El uso de elementos elementales no debe contradecir el uso de un grupo al que pertenecen los elementos elementales.

No debe especificarse una cláusula USAGE en una entrada de nivel de grupo para la que se ha especificado una cláusula GROUP-USAGE NATIONAL.

Cuando se especifica o está implícita una cláusula GROUP-USAGE NATIONAL para una entrada de nivel de grupo, se debe especificar o implicar USAGE NATIONAL para cada elemento elemental del grupo. Para obtener detalles, consulte "cláusula GROUP-USAGE" en la página 179.

Cuando no se especifica la cláusula USAGE en el nivel de grupo o elemental, se implica una cláusula de uso con:

- Uso DISPLAY cuando la cláusula PICTURE contiene sólo símbolos distintos de G o N
- Utilizar NATIONAL cuando la cláusula PICTURE contiene sólo uno o más del símbolo N y la opción de compilador NSYMBOL (NATIONAL) está en vigor
- Uso DISPLAY-1 cuando la cláusula PICTURE contiene uno o más del símbolo N y la opción de compilador NSYMBOL (DBCS) está en vigor

Para los elementos de datos definidos con la cláusula DATE FORMAT, solo se permite el uso DISPLAY y COMP-3 (o sus equivalentes, COMPUTATIONAL-3 y PACKED-DECIMAL). Para obtener detalles, consulte "Combinación de la cláusula DATE FORMAT con otras cláusulas" en la página 173.

La cláusula TYPE no puede especificarse en la misma entrada de descripción de datos que la cláusula USAGE.

Las entradas de descripción de datos con una cláusula TYPE no pueden estar subordinadas a una entrada de descripción de datos que contenga una cláusula USAGE. Por ejemplo, lo siguiente no está permitido:

```
01 FLAGS   USAGE   DISPLAY.
   05 F-STATUS    TYPE CHAR.
   05 FLAG-ACTIVE TYPE CHAR.
```

Elementos computacionales

Un elemento computacional es un valor utilizado en operaciones aritméticas. Debe ser numérico. Si un elemento de grupo se describe con un uso computacional, los elementos elementales dentro del grupo tienen ese uso.

La longitud máxima de un elemento de cálculo es de 18 dígitos decimales, excepto para un elemento PACKED-DECIMAL. Si la opción de compilador ARITH (COMPAT) está en vigor, la longitud máxima de un elemento PACKED-DECIMAL es de 18 dígitos decimales. Si la opción de compilador ARITH (EXTEND) está en vigor, la longitud máxima de un elemento PACKED-DECIMAL es de 31 dígitos decimales.

La PICTURE de un elemento computacional sólo puede contener:

9
Una o más posiciones de caracteres numéricos

S
Un signo operativo

V
Una coma decimal implícita

P
Una o más posiciones de escalado decimal

Los elementos COMPUTATIONAL-1 y COMPUTATIONAL-2 (coma flotante interna) no pueden tener series PICTURE.

BINARIO
Especificado para elementos de datos binarios. Estos elementos tienen un equivalente decimal que consta de los dígitos decimales del 0 al 9, más un signo. Los números negativos se representan como el complemento de los dos del número positivo con el mismo valor absoluto.

La cantidad de almacenamiento ocupada por un elemento binario depende del número de dígitos decimales definidos en su cláusula PICTURE:

Dígitos en cláusula PICTURE	Almacenamiento ocupado
1 a 4	2 bytes (media palabra)
5 a 9	4 bytes (palabra completa)
10 a 18	8 bytes (palabra doble)

De forma predeterminada, los datos binarios utilizan el formato de representación binaria nativo de la plataforma. En sistemas Linux , la representación binaria nativa tiene el formato *little-endian* (dígito menos significativo en la dirección más baja). Consulte la opción de compilador BINARY si desea modificar el formato de endianness para elementos de datos binarios.

Los elementos de datos BINARY, COMPUTATIONAL y COMPUTATIONAL-4 pueden verse afectados por las opciones de compilador BINARY y TRUNC de opción de compilador TRUNC. Para obtener información sobre el efecto de esta opción de compilador, , consulte *TRUNC* en *COBOL for Linux en x86 Guía de programación*.

DECIMAL EMPAQUETADO
Se especifica para elementos decimales internos. Este elemento aparece en el almacenamiento en formato decimal empaquetado. Hay dos dígitos para cada posición de carácter, excepto para la posición de carácter final, que está ocupada por el dígito de orden inferior y el signo. Un elemento de este tipo puede contener cualquiera de los dígitos del 0 al 9, más un signo, que representa un valor que no supera los 18 dígitos decimales.

La representación de signo utiliza la misma configuración de bits que la representación de signo de 4 bits en campos decimales con zona. Para obtener detalles, consulte *Representación de signo de datos con zona y decimal empaquetado* en la publicación *COBOL for Linux en x86 Guía de programación*.

También se puede especificar PACKED-DECIMAL para elementos de fecha y hora cuyo literal FORMAT contiene sólo especificadores de conversión. Estos especificadores de conversión sólo pueden contener dígitos numéricos.

COMPUTATIONAL o COMP (binario)
Es el equivalente a BINARY. La frase COMPUTATIONAL es sinónimo de BINARY.

COMPUTATIONAL-1 o COMP-1 (coma flotante)
Especificado para elementos de coma flotante internos (precisión única). Los elementos COMP-1 tienen una longitud de 4 bytes.

Los elementos de datos COMP-1 se ven afectados por la opción de compilador FLOAT (NATIVE | BE | LE) . Para obtener detalles, consulte *FLOAT* en la publicación *COBOL for Linux en x86 Guía de programación*.

COMPUTATIONAL-2 o COMP-2 (coma flotante larga)
Especificado para elementos de coma flotante internos (precisión doble). Los elementos COMP-2 tienen una longitud de 8 bytes.

Los elementos de datos COMP-2 se ven afectados por la opción de compilador FLOAT (NATIVE | BE | LE) . Para obtener detalles, consulte *FLOAT* en la publicación *COBOL for Linux en x86 Guía de programación*.

COMPUTATIONAL-3 o COMP-3 (decimal interno)
Es el equivalente de PACKED-DECIMAL.

COMPUTATIONAL-4 o COMP-4 (binario)
Es el equivalente a BINARY.

COMPUTATIONAL-5 o COMP-5 (binario nativo)

Estos elementos de datos se representan en el almacenamiento como datos binarios. Los elementos de datos pueden contener valores hasta la capacidad de la representación binaria nativa (2, 4 u 8 bytes), en lugar de limitarse al valor implícito por el número de líneas en la imagen para el elemento (como es el caso de los datos USAGE BINARY). Cuando los datos numéricos se mueven o almacenan en un elemento COMP-5 , el truncamiento se produce en el tamaño del campo binario en lugar de en el límite de tamaño de imagen COBOL. Cuando se hace referencia a un elemento COMP-5 , se utiliza el tamaño de campo binario completo en la operación.

La opción de compilador TRUNC (BIN) hace que todos los elementos de datos binarios (USAGE BINARY, COMP, COMP-4) se manejen como si se hubieran declarado USAGE COMP-5.

La tabla siguiente muestra varias series de caracteres de imagen, la representación de almacenamiento resultante y el rango de valores para los elementos de datos descritos con USAGE COMP-5.

Imagen	Representación de almacenamiento	Valores numéricos
S9(1) a S9(4)	Media palabra binaria (2 bytes)	-32768 a +32767
S9(5) a S9(9)	Palabra completa binaria (4 bytes)	-2.147.483.648 a +2.147.483.647
S9(10) a S9(18)	Palabra doble binaria (8 bytes)	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
9 (1) a 9 (4)	Media palabra binaria (2 bytes)	0 a 65535
9 (5) a 9 (9)	Palabra completa binaria (4 bytes)	0 a 4.294.967.295
9 (10) a 9 (18)	Palabra doble binaria (8 bytes)	0 a 18.446.744.073.709.551.615

La imagen de un elemento de datos COMP-5 puede especificar un factor de escala (es decir, posiciones decimales o posiciones enteras implícitas). En este caso, las capacidades máximas listadas en la tabla anterior deben escalarse adecuadamente. Por ejemplo, un elemento de datos descrito con PICTURE S99V99 COMP-5 se representa en el almacenamiento como una media palabra binaria y soporta un rango de valores de -327.68 a +327.67.

NOTA DE USO: Cuando se utiliza la frase ON SIZE ERROR en una sentencia aritmética y se define un receptor con USAGE COMP-5, el valor máximo que puede contener el receptor es el valor implícito en la serie de caracteres PICTURE decimal del elemento. Cualquier intento de almacenar un valor mayor que este máximo dará como resultado una condición de error de tamaño.

frase DISPLAY

El elemento de datos se almacena en formato de caracteres, un carácter para cada byte de 8 bits. Corresponde al formato utilizado para la salida impresa. DISPLAY puede ser explícito o implícito.

USAGE IS DISPLAY es válido para los siguientes tipos de elementos:

- Alfabético
- Alfanumérico
- Alfanumérico-editado
- Booleano
- Fecha, hora e indicación de fecha y hora
- Numérico-editado
- Coma flotante externa

- Decimal externo

Los elementos alfabéticos, alfanuméricos, editados alfanuméricos, booleanos y editados numéricos se describen en "Categorías de datos y reglas PICTURE" en la página 199.

Los elementos decimales externos con USAGE DISPLAY a veces se conocen como elementos *decimales con zona* . Cada dígito de un número se representa mediante un solo byte. Los 4 bits de orden superior de cada byte son bits de zona; los 4 bits de orden superior del byte de orden inferior representan el signo del elemento. Los 4 bits de orden inferior de cada byte contienen el valor del dígito.

Si la opción de compilador ARITH (COMPAT) está en vigor, la longitud máxima de un elemento decimal externo es de 18 dígitos. Si la opción de compilador ARITH (EXTEND) está en vigor, la longitud máxima de un elemento decimal externo será de 31 dígitos.

La serie de caracteres PICTURE de un elemento decimal externo sólo puede contener:

- Uno o más del símbolo 9
- El signo operativo, S
- La coma decimal asumida, V
- Uno o más del símbolo P

Efecto de la opción de compilador CHAR (EBCDIC)

Los elementos de datos definidos con la frase DISPLAY o DISPLAY-1 se tratan como EBCDIC cuando se utiliza la opción de compilador CHAR (EBCDIC), a menos que los datos de tipo carácter se definan con la frase NATIVE.

Frase DISPLAY-1

La frase DISPLAY-1 define un elemento como DBCS. El elemento de datos se almacena en formato de caracteres, con cada carácter ocupando 2 bytes de almacenamiento.

Frase FUNCTION-POINTER

La frase FUNCTION-POINTER define un elemento como un *elemento de datos de puntero de función*. Un elemento de datos de puntero de función puede contener la dirección de un descriptor para un punto de entrada de procedimiento.

Un puntero de función es un elemento elemental de 4 bytes o un elemento elemental de 8 bytes en función de si la opción de compilador **ADDR(32)** o **ADDR(64)** está en vigor. Los punteros de función tienen las mismas prestaciones que los punteros de procedimiento. Los punteros de función son fácilmente interoperables con punteros de función C.

Un puntero de función puede apuntar a un descriptor de función para uno de los siguientes o puede contener NULL:

- El punto de entrada primario de un programa COBOL, definido por el párrafo PROGRAM-ID del programa más externo
- Un punto de entrada alternativo de un programa COBOL, definido por una sentencia ENTRY de COBOL
- Un punto de entrada en un programa no COBOL

Una cláusula VALUE para un elemento de datos de puntero de función sólo puede contener NULL o NULLS.

Un puntero de función se puede utilizar en los mismos contextos que un puntero de procedimiento, tal como se define en "Frase PROCEDURE-POINTER" en la página 231.

frase INDEX

Un elemento de datos definido con la frase INDEX es un *elemento de datos de índice*.

Un *elemento de datos de índice* es un elemento elemental de 4 bytes o un elemento elemental de 8 bytes en función de si la opción de compilador **ADDR(32)** o **ADDR(64)** está en vigor, que se puede utilizar para guardar valores de nombre de índice para una referencia futura. Un *elemento de datos de índice* no está necesariamente conectado con ninguna tabla específica. Mediante una sentencia SET, a un elemento de datos de índice se le puede asignar un valor de nombre de índice. Este valor corresponde al número de aparición de una tabla.

Las referencias directas a un elemento de datos de índice sólo pueden realizarse en una sentencia SEARCH, una sentencia SET, una condición de relación, la frase USING de la cabecera PROCEDURE DIVISION o la frase USING de la sentencia CALL o ENTRY.

Un elemento de datos de índice puede formar parte de un elemento de grupo alfanumérico al que se hace referencia en una sentencia MOVE o una sentencia de entrada/salida.

Un elemento de datos de índice guarda valores que representan apariciones de tabla, pero no está necesariamente definido como parte de ninguna tabla. No hay conversión de valores cuando se hace referencia a un elemento de datos de índice en las circunstancias siguientes:

- directamente en una sentencia SEARCH o SET
- indirectamente en una sentencia MOVE
- indirectamente en una sentencia de entrada o salida

Un elemento de datos de índice no puede ser una variable condicional.

DATE FORMAT, JUSTIFIC, PICTURE, BLANK WHEN ZERO, VALUE, TYPE, o las cláusulas FORMAT no se pueden utilizar para describir un elemento de grupo o elementos elementales descritos con la cláusula USAGE IS INDEX.

SYNCHRONIZED se puede utilizar con USAGE IS INDEX para obtener un uso eficiente del elemento de datos de índice.

Frase NACIONAL

La frase NATIONAL define un elemento cuyo contenido se representa en el almacenamiento en UTF-16 (CCSID 1200). La clase y la categoría del elemento de datos dependen de los símbolos de imagen especificados en la cláusula PICTURE asociada.

Frase POINTER

Un elemento de datos definido con USAGE IS POINTER es un *elemento de datos de puntero*. Un *elemento de datos de puntero* es un elemento elemental de 4 bytes o un elemento elemental de 8 bytes en función de si la opción de compilador **ADDR(32)** o **ADDR(64)** está en vigor.

Puede utilizar elementos de datos de puntero para realizar un direccionamiento base limitado. Los elementos de datos de puntero pueden compararse para la igualdad o moverse a otros elementos de puntero.

Un elemento de datos de puntero sólo se puede utilizar:

- En una sentencia SET (formato 5 y formato 8 solamente)
- En una condición de relación
- En la frase USING de una sentencia CALL, una sentencia ENTRY o la cabecera PROCEDURE DIVISION

Los elementos de datos de puntero pueden formar parte de un grupo alfanumérico al que se hace referencia en una sentencia MOVE o una sentencia de entrada/salida. Sin embargo, si un elemento de datos de puntero forma parte de un grupo, no hay ninguna conversión de valores cuando se ejecuta la sentencia.

Un elemento de datos de puntero puede ser el asunto u objeto de una cláusula REDEFINES.

SYNCHRONIZED se puede utilizar con USAGE IS POINTER para obtener un uso eficiente del elemento de datos de puntero.

Una cláusula VALUE para un elemento de datos de puntero sólo puede contener NULL o NULLS.

Un elemento de datos de puntero no puede ser una variable condicional.

Un elemento de datos de puntero no pertenece a ninguna clase o categoría.

La tabla siguiente lista las cláusulas que pueden o no pueden utilizarse para describir elementos de grupo o elementales definidos con USAGE IS POINTER.

Tabla 22. Cláusulas que pueden o no pueden utilizarse con USAGE IS POINTER

Se puede utilizar con USAGE IS POINTER	**No se puede utilizar con USAGE IS POINTER**
cláusula GLOBAL cláusula EXTERNAL cláusula OCCURS cláusula LIKE	Cláusula DATE FORMAT Cláusula JUSTIFIC cláusula PICTURE Cláusula BLANK WHEN ZERO cláusula TYPE cláusula FORMAT

Los elementos de datos de puntero se ignoran en el proceso de una frase CORRESPONDIENTE.

Un elemento de datos de puntero se puede escribir en un , pero al leer posteriormente el registro que contiene el puntero, es posible que la dirección contenida ya no represente un puntero válido.

USAGE IS POINTER se especifica implícitamente para el registro especial ADDRESS OF. Para obtener más información, consulte *Utilización de tablas (matrices) y punteros* en la publicación *COBOL for Linux en x86 Guía de programación*.

Frase PROCEDURE-POINTER

La frase PROCEDURE-POINTER define un elemento como un *elemento de datos de puntero de procedimiento*. Un elemento de datos de puntero de procedimiento puede contener la dirección de un descriptor para un punto de entrada de procedimiento.

Un elemento de datos de puntero de procedimiento es un elemento elemental de 4 bytes o un elemento elemental de 8 bytes en función de si la opción de compilador **ADDR(32)** o **ADDR(64)** está en vigor.

Un puntero de procedimiento puede apuntar a un descriptor de función para uno de los siguientes o puede contener NULL:

- El punto de entrada primario de un programa COBOL tal como lo define el párrafo de ID de programa del programa más externo de una unidad de compilación
- Un punto de entrada alternativo de un programa COBOL tal como lo define una sentencia ENTRY de COBOL
- Un punto de entrada en un programa no COBOL

Un elemento de datos de puntero de procedimiento sólo se puede utilizar:

- En una sentencia SET (sólo formato 6)
- En una sentencia CALL
- En una condición de relación
- En la frase USING de una sentencia ENTRY o la cabecera PROCEDURE DIVISION

Los elementos de datos de puntero de procedimiento pueden compararse para la igualdad o moverse a otros elementos de datos de puntero de procedimiento.

Los elementos de datos de puntero de procedimiento pueden formar parte de un grupo al que se hace referencia en una sentencia MOVE o una sentencia de entrada/salida. Sin embargo, no hay conversión de valores cuando se ejecuta la sentencia. Si un elemento de datos de puntero de procedimiento se graba en un , la lectura posterior del registro que contiene el puntero de procedimiento puede dar como resultado un valor no válido en el puntero de procedimiento.

Un elemento de datos de puntero de procedimiento puede ser el sujeto u objeto de una cláusula REDEFINES.

SYNCHRONIZED puede utilizarse con USAGE IS PROCEDURE-POINTER para obtener una alineación eficiente del elemento de datos de puntero de procedimiento.

Las cláusulas GLOBAL, EXTERNAL, OCCURSy LIKE se pueden utilizar con USAGE IS PROCEDURE-POINTER.

Una cláusula VALUE para un elemento de datos de puntero de procedimiento sólo puede contener NULL o NULLS.

Las cláusulas DATE FORMAT, JUSTIFIC, PICTURE, FORMAT, TYPE, y BLANK WHEN ZERO no se pueden utilizar para describir elementos de grupo o elementales definidos con la cláusula USAGE IS PROCEDURE-POINTER.

Un elemento de datos de puntero de procedimiento no puede ser una variable condicional.

Un elemento de datos de puntero de procedimiento no pertenece a ninguna clase o categoría.

Los elementos de datos de puntero de procedimiento se ignoran en las operaciones CORRESPONDIENTE.

Frase NATIVE

Utilizando la frase NATIVE, puede mezclar caracteres y datos de coma flotante tal como se representan en las plataformas z/OS o OS/390 y Linux . La frase NATIVE altera temporalmente las opciones de compilador CHAR (EBCDIC) y FLOAT (BE) , que indican los usos de tipos de datos de host.

El uso de tipos de datos de host y nativos dentro de un programa (ASCII y EBCDIC, y coma flotante IEEE) sólo es válido para los elementos de datos definidos específicamente con la frase NATIVE.

La especificación de NATIVE no cambia la clase ni la categoría del elemento de datos.

Los elementos de datos numéricos se procesan en operaciones aritméticas (comparaciones numéricas, expresiones aritméticas, asignación a destinos numéricos, sentencias aritméticas) basándose en sus valores numéricos lógicos, independientemente de sus representaciones internas.

Los caracteres se convierten a la representación del elemento de destino antes de una asignación.

Las comparaciones se realizan basándose en las reglas de secuencia de clasificación aplicables a los operandos. Si se comparan los caracteres alfanuméricos o DBCS nativos y no nativos, la comparación se basa en la opción COLLSEQ en vigor.

cláusula VALUE

La cláusula VALUE especifica el contenido inicial de un elemento de datos o los valores asociados a un nombre de condición. El uso de la cláusula VALUE difiere en función de la sección DATA DIVISION en la que se especifica.

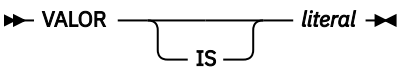
Una cláusula VALUE que se utiliza en FILE SECTION o LINKAGE SECTION en una entrada que no sea una entrada de nombre de condición es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

En WORKING-STORAGE SECTION y LOCAL-STORAGE SECTION, la cláusula VALUE puede utilizarse en entradas de nombre de condición o al especificar el valor inicial de cualquier elemento de datos. El elemento de datos asume el valor especificado al principio de la ejecución del programa. Si el valor inicial no se especifica explícitamente, el valor es imprevisible.

Formato 1

El formato 1 especifica el valor inicial de un elemento de datos. La inicialización es independiente de cualquier cláusula BLANK WHEN ZERO o JUSTIFIC que se especifique.

Formato 1: valor literal



Una cláusula format-1 VALUE especificada en una entrada de descripción de datos que contiene o está subordinada a una cláusula OCCURS hace que a cada aparición del elemento de datos asociado se le asigne el valor especificado. Se supone que cada estructura que contiene la frase DEPENDING ON de la cláusula OCCURS contiene el número máximo de apariciones para la inicialización de VALUE.

La cláusula VALUE no debe especificarse para una entrada de descripción de datos que contenga o esté subordinada a una entrada que contenga una cláusula EXTERNAL o REDEFINES. Esta regla no se aplica a las entradas de nombre de condición.

Se puede especificar una cláusula format-1 VALUE para un elemento de datos elemental o para un elemento de grupo. Cuando se especifica la cláusula VALUE a nivel de grupo, el área de grupo se inicializa sin tener en cuenta las entradas subordinadas del grupo. Además, no debe especificarse una cláusula VALUE para las entradas subordinadas dentro del grupo.

Para elementos de grupo, no se debe especificar la cláusula VALUE si alguna entrada subordinada contiene una cláusula JUSTIFIC o SYNCHRONIZED.

Si se especifica la cláusula VALUE para un grupo alfanumérico, todos los elementos subordinados deben describirse explícita o implícitamente con USAGE DISPLAY.

La cláusula VALUE no debe entrar en conflicto con otras cláusulas en la entrada de descripción de datos o en la descripción de datos de la jerarquía de dicha entrada.

Las funciones de los caracteres de edición en una cláusula PICTURE se ignoran al determinar el valor inicial del elemento descrito. Sin embargo, los caracteres de edición se incluyen al determinar el tamaño del elemento. Por lo tanto, los caracteres de edición deben incluirse en el literal. Por ejemplo, si el elemento se define como PICTURE +999.99 y el valor es +12.34, la cláusula VALUE debe especificarse como VALUE "+ 012.34".

No se puede especificar una cláusula VALUE para elementos de coma flotante externos.

Puede especificarse una cláusula VALUE en la entrada de descripción de datos para un nombre-tipo. Dicha cláusula VALUE se utiliza para inicializar cualquier nombre de datos (que no es un nombre-tipo), que se define utilizando una cláusula TYPE que hace referencia a dicho nombre-tipo. Si se especifica una cláusula VALUE en la descripción de datos del asunto de una cláusula TYPE, cualquier cláusula VALUE especificada en la descripción del nombre-tipo asociado se ignora para esta entrada.

Un elemento de datos no puede contener una cláusula VALUE si el elemento de datos anterior contiene una cláusula OCCURS con la frase DEPENDING ON.

Una cláusula VALUE asociada con un elemento de fecha, hora o indicación de fecha y hora debe ser un literal no numérico. El literal se alinea de acuerdo con las reglas de alineación. No se realiza ningún formateo del literal para que coincida con los especificadores de conversión o la definición LOCALE, excepto si la USAGE del elemento es PACKED-DECIMAL, en cuyo caso el literal no numérico se convierte en empaquetado.

Reglas para valores literales

- Siempre que se especifique un literal, se puede sustituir una constante figurativa, de acuerdo con las reglas especificadas en “Constantes figurativas” en la página 15.
- Si el elemento es numérico de clase, el literal de cláusula VALUE debe ser numérico. Si el literal define el valor de un elemento WORKING-STORAGE o un elemento LOCAL-STORAGE, el literal se alinea de acuerdo con las reglas para movimientos numéricos, con una restricción adicional: El literal no debe tener un valor que requiera el truncamiento de dígitos distintos de cero. Si el literal está firmado, la serie de caracteres PICTURE asociada debe contener un símbolo de signo.

- Con algunas excepciones, los literales numéricos en una cláusula VALUE deben tener un valor dentro del rango de valores indicado por la cláusula PICTURE para el elemento. Por ejemplo, para PICTURE 99PPP, el literal debe ser cero o estar dentro del rango de 1000 a 99000. Para PICTURE PPP99, el literal debe estar dentro del rango de 0.00000 a 0.00099.

  Las excepciones son las siguientes:

  – Elementos de datos descritos con el uso COMP-5 que no tienen un símbolo de imagen P en su cláusula PICTURE.
  – Cuando la opción de compilador TRUNC (BIN) está en vigor, los elementos de datos descritos con el uso BINARY, COMP o COMP-4 que no tienen un símbolo de imagen P en su cláusula PICTURE.

    Una cláusula VALUE para estos elementos puede tener un valor hasta la capacidad de la representación binaria nativa.
- Si se especifica la cláusula VALUE para un elemento alfabético elemental, alfanumérico, editado alfanumérico o editado numérico descrito con el uso DISPLAY, el literal de cláusula VALUE debe ser un literal alfanumérico o una constante figurativa. El literal se alinea de acuerdo con las reglas de alineación alfanuméricas, con una restricción adicional: el número de caracteres en el literal no debe exceder el tamaño del elemento.
- Si se especifica la cláusula VALUE para un elemento elemental de edición nacional, nacional o numérico descrito con el uso NATIONAL, el literal de cláusula VALUE debe ser un literal nacional o alfanumérico o una constante figurativa tal como se especifica en “Constantes figurativas” en la página 15. El valor de un literal alfanumérico se convierte de su representación de código fuente a la representación UTF-16 . El literal se alinea de acuerdo con las reglas de alineación nacional, con una restricción adicional: el número de caracteres en el literal no debe exceder el tamaño, en posiciones de caracteres, del elemento.
- Si se especifica la cláusula VALUE a nivel de grupo para un grupo alfanumérico, el literal debe ser un literal alfanumérico o una constante figurativa tal como se especifica en “Constantes figurativas” en la página 15, que no sea ALL *literal-nacional* . El tamaño del literal no debe superar el tamaño del elemento de grupo.
- Si se especifica la cláusula VALUE a nivel de grupo para un grupo nacional, el literal puede ser un literal alfanumérico, un literal nacional o una de las constantes figurativas ZERO, SPACE, COMILLAS, HIGH-VALUE, LOW-VALUE, *carácter simbólico*, ALL *literal-nacional*o ALL *-literal*. El valor de un literal alfanumérico se convierte de su representación de código fuente a la representación UTF-16 . Cada constante figurativa representa un valor de carácter nacional. El tamaño del literal no debe superar el tamaño del elemento de grupo.
- Una cláusula VALUE asociada a un elemento DBCS debe contener un literal DBCS, la constante figurativa SPACE o la constante figurativa ALL *literal-DBCS*. La longitud del literal no debe exceder el tamaño indicado por la cláusula PICTURE del elemento de datos.
- Una cláusula VALUE que especifica un literal nacional sólo se puede asociar con un elemento de datos de clase nacional.
- Una cláusula VALUE que especifica un literal DBCS sólo puede asociarse con un elemento de datos de clase DBCS.
- Una cláusula VALUE asociada con un elemento COMPUTATIONAL-1 o COMPUTATIONAL-2 (coma flotante interna) debe especificar un literal de coma flotante. Además, la constante figurativa CERO y las formas entera y decimal del literal cero se pueden especificar en una cláusula VALUE de coma flotante.

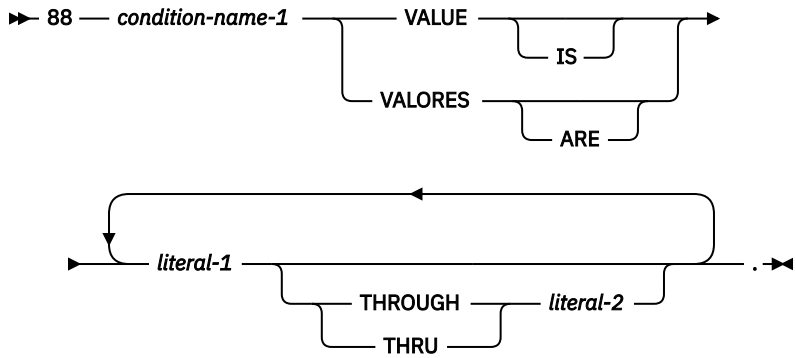
  No puede especificar un literal numérico de formato de coma flotante en la cláusula VALUE de un elemento numérico de punto fijo.

  Para obtener información sobre los valores literales de coma flotante, consulte “Reglas para valores literales de coma flotante” en la página 35.
- Si el elemento es booleano, la cláusula VALUE debe ser un literal booleano.

Formato 2

Este formato asocia un valor, valores o rangos de valores con un nombre de condición. Cada nombre de condición requiere una entrada level-88 independiente. El número de nivel 88 y el nombre de condición no forman parte de la propia cláusula VALUE format-2 . Se incluyen en el formato sólo para mayor claridad.

Formato 2: valor de nombre-condición



condition-name-1
Nombre especificado por el usuario que asocia un valor con una variable condicional. Si la variable condicional asociada requiere subíndices o índices, cada referencia de procedimiento al nombre-condición debe estar subindexada o indexada según sea necesario para la variable condicional.

Los nombres de condición se prueban de forma procedimental en condiciones de nombre de condición (consulte “Expresiones condicionales” en la página 250).

literal-1
Asocia el nombre de condición con un único valor.

La clase de *literal-1* debe ser una clase válida para la asignación a la variable condicional asociada.

literal-1* THROUGH *literal-2
Asocia el nombre de condición con al menos un rango de valores. Cuando se utiliza la frase THROUGH, *literal-1* debe ser menor que *literal-2*, a menos que el elemento de datos asociado sea un campo de fecha de última ventana que no sea un año. Para obtener detalles, consulte “Reglas para entradas de nombre de condición” en la página 236.

literal-1 y *literal-2* deben ser de la misma clase. La clase de *literal-1* y *literal-2* debe ser una clase válida para la asignación a la variable condicional asociada.

El rango de literales alfanuméricos, literales nacionales o literales DBCS especificados para la frase THROUGH se basa en la secuencia de clasificación en vigor para la variable condicional asociada. Para obtener más información sobre la clasificación de secuencias, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

Si la variable condicional asociada es de clase DBCS, *literal-1* y *literal-2* deben ser literales DBCS. Se puede especificar la constante figurativa SPACE o la constante figurativa ALL *DBCS-literal* .

Si la variable condicional asociada es de la clase NATIONAL, *literal-1* y *literal-2* deben ser ambos literales nacionales o ambos literales alfanuméricos para un nombre de condición determinado. Se pueden especificar las constantes figurativas ZERO, SPACE, QUOTE, HIGH-VALUE, LOW-VALUE, *carácter-simbólico*, ALL *literal-nacional*o ALL *literal* .

Reglas para entradas de nombre de condición

Existen determinadas reglas para las entradas de nombre de condición.

Las reglas son:

- La cláusula VALUE es necesaria en una entrada de nombre de condición y debe ser la única cláusula de la entrada. Cada entrada de nombre de condición está asociada con una variable condicional anterior. Por lo tanto, cada entrada level-88 siempre debe ir precedida de la entrada para la variable condicional o de otra entrada level-88 cuando se aplican varios nombres de condición a una variable condicional. Cada entrada de level-88 de este tipo tiene implícitamente las características PICTURE de la variable condicional.
- Un espacio, una coma de separador o un punto y coma de separador deben separar los operandos sucesivos.

  Cada entrada debe finalizar con un punto separador.
- Las palabras clave THROUGH y THRU son equivalentes.
- Las entradas de nombre de condición asociadas a una variable condicional determinada deben ir inmediatamente después de la entrada de variable condicional. La variable condicional puede ser cualquier entrada de descripción de datos elemental excepto las siguientes:
  - Otro nombre-condición
  - Una cláusula RENAMES (elementolevel-66)
  - Un elemento descrito con USAGE IS INDEX
  - Un elemento descrito con USAGE POINTER, USAGE PROCEDURE-POINTER o USAGE FUNCTION-POINTER
- Los nombres de condición se pueden especificar tanto a nivel de grupo como a nivel subordinado dentro de un grupo alfanumérico o un grupo nacional.
- Cuando se especifica el nombre-condición para una entrada de descripción de datos de grupo alfanumérico:
  - El valor de *literal-1* (o *literal-1* y *literal-2*) debe especificarse como literal alfanumérico o constante figurativa.
  - El grupo puede contener elementos de cualquier uso.
- Cuando se especifica el nombre de condición para una entrada de descripción de datos de grupo nacional:
  - El valor de *literal-1* (o *literal-1* y *literal-2*) debe especificarse como un literal alfanumérico, un literal nacional o una constante figurativa.
  - El grupo solo puede contener elementos de uso nacional, tal como se especifica para "cláusula GROUP-USAGE" en la página 179.
- Cuando el nombre de condición está asociado con una entrada de descripción de datos de grupo alfanumérico o una entrada de descripción de datos de grupo nacional :
  - El tamaño de cada valor literal no debe sobrepasar la suma de los tamaños de todos los elementos elementales del grupo.
  - Ningún elemento del grupo puede contener una cláusula JUSTIFIC o SYNCHRONIZED.
- Las pruebas de relación implícitas en la definición de un nombre de condición se realizan de acuerdo con las reglas a las que se hace referencia en la tabla siguiente.

Tabla 23. ***Referencias de prueba de relación para nombres de condición***

Tipo de variable condicional	**Reglas de condición de relación**
Elemento de grupo alfanumérico	"Comparaciones de grupos" en la página 263

Tabla 23. ***Referencias de prueba de relación para nombres de condición*** *(continuación)*	
Tipo de variable condicional	**Reglas de condición de relación**
Elemento de grupo nacional (tratado como elemento de datos elementales de clase nacional)	“Comparaciones nacionales” en la página 262
Elemento de datos elementales de clase alfanumérico	“Comparaciones alfanuméricas” en la página 260
Elemento de datos elementales de clase nacional	“Comparaciones nacionales” en la página 262
Elemento de datos elementales de clase numérica	“Comparaciones numéricas” en la página 263
Elemento de datos elemental de clase DBCS	“Comparaciones DBCS” en la página 262

- Una cláusula VALUE que especifica un literal nacional puede asociarse con un nombre de condición definido sólo para un elemento de datos de clase nacional.
- Una cláusula VALUE que especifica un literal DBCS puede asociarse con un nombre de condición definido sólo para un elemento de datos de clase DBCS.
- Los literales de una entrada de nombre de condición para un elemento de datos elemental de clase nacional o un elemento de grupo nacional deben ser literales nacionales o literales alfanuméricos, y *literal-1* y *literal-2* deben ser de la misma clase. Para grupos alfanuméricos o elementos de datos elementales de otras clases, el tipo de literal debe ser coherente con el tipo de datos de la variable condicional. En el ejemplo siguiente:
  - CITY-COUNTY-INFO, COUNTY-NO y CITY son variables condicionales.

    La PICTURE asociada con COUNTY-NO limita el valor de nombre de condición a un literal numérico de dos dígitos.

    La PICTURE asociada con CITY limita el valor de nombre-condición a un literal alfanumérico de tres caracteres.
  - Los nombres de condición asociados son entradas de level-88 .

    Los valores de los nombres de condición asociados con CITY-COUNTY-INFO no pueden superar los cinco caracteres.

    Puesto que se trata de un elemento de grupo alfanumérico, el literal debe ser alfanumérico.

```
05  CITY-COUNTY-INFO.
    88  BRONX                  VALUE "03NYC".
    88  BROOKLYN               VALUE "24NYC".
    88  MANHATTAN              VALUE "31NYC".
    88  QUEENS                 VALUE "41NYC".
    88  STATEN-ISLAND          VALUE "43NYC".
  10  COUNTY-NO                PICTURE 99.
      88  KINGS                VALUE 24.
      88  NEW-YORK             VALUE 31.
      88  RICHMOND             VALUE 43.
  10  CITY                     PICTURE X(3).
      88  BUFFALO              VALUE "BUF".
      88  NEW-YORK-CITY        VALUE "NYC".
      88  POUGHKEEPSIE         VALUE "POK".
05  POPULATION...
```

- Un nombre de condición se puede asociar con un elemento de fecha, hora o indicación de fecha y hora. En este caso:
  - El valor de nombre-condición debe especificarse como un literal no numérico.
  - Cada nombre-condición tiene implícitamente las características FORMAT de la variable condicional. Por lo tanto, cualquier prueba de relación que implique este nombre-condición se realiza de acuerdo con las reglas para comparar elementos de la clase fecha-hora.

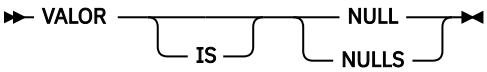
  - Se puede especificar una frase THROUGH cuando una variable condicional es de clase fecha-hora. En este caso, la hora o fecha de *literal-1* debe ser menor que *literal-2*.
- Si el artículo es un campo de fecha con ventana, se aplican las restricciones siguientes:
  - Para variables condicionales alfanuméricas:
    - Tanto *literal-1* como *literal-2* (si se especifica) deben ser literales alfanuméricos de la misma longitud que la variable condicional.
    - Los literales no deben especificarse como constantes figurativas.
    - Si se especifica *literal-2* , ambos literales deben contener sólo dígitos decimales.
  - Si la opción de compilador YEARWINDOW se especifica como un entero negativo, no se debe especificar *literal-2* .
  - Si se especifica *literal-2* , *literal-1* debe ser menor que *literal-2* después de aplicar la ventana de siglo especificada por la opción de compilador YEARWINDOW. Es decir, el valor de fecha expandida de *literal-1* debe ser menor que el valor de fecha expandida de *literal-2*.

Para obtener más información sobre cómo utilizar nombres de condición con campos de fecha con ventanas, consulte Condiciones de nombre de condición y comparaciones de campos de fecha con ventanas.

Formato 3

Este formato asigna una dirección no válida como valor inicial de un elemento definido como USAGE POINTER, USAGE PROCEDURE-POINTER o USAGE FUNCTION-POINTER.

Formato 3: valor NULL



VALUE IS NULL sólo puede especificarse para elementos elementales descritos implícita o explícitamente como USAGE POINTER, USAGE PROCEDURE-POINTER o USAGE FUNCTION-POINTER.

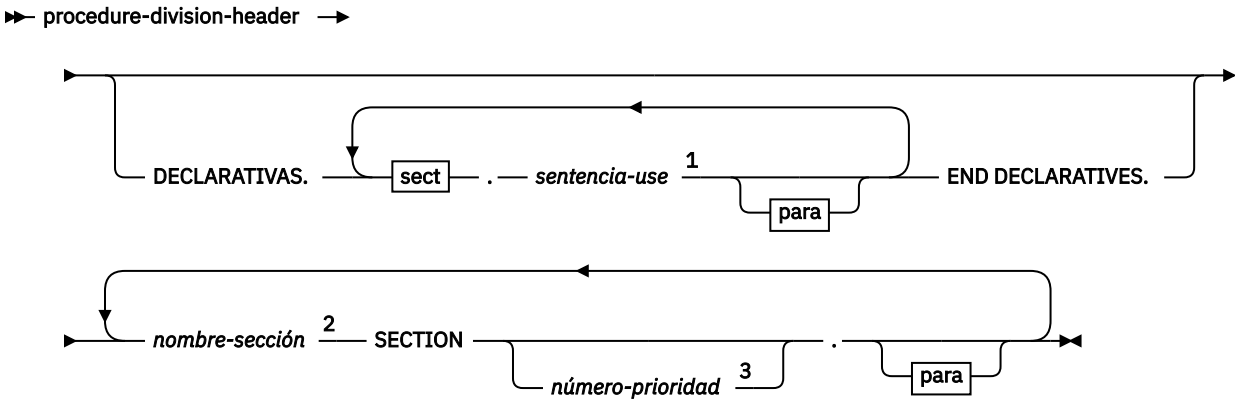
Parte 6. PROCEDURE DIVISION

Capítulo 18. Estructura de división de procedimiento

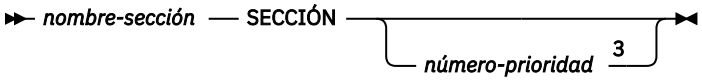
La PROCEDURE DIVISION es una división opcional.

División de procedimiento de programa
La división de procedimientos de programa consta de declaraciones opcionales y procedimientos que contienen secciones, párrafos, frases y sentencias.

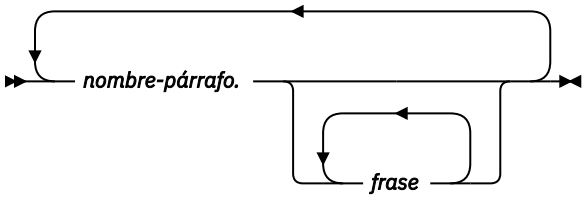
Formato: división de procedimiento



secta



para



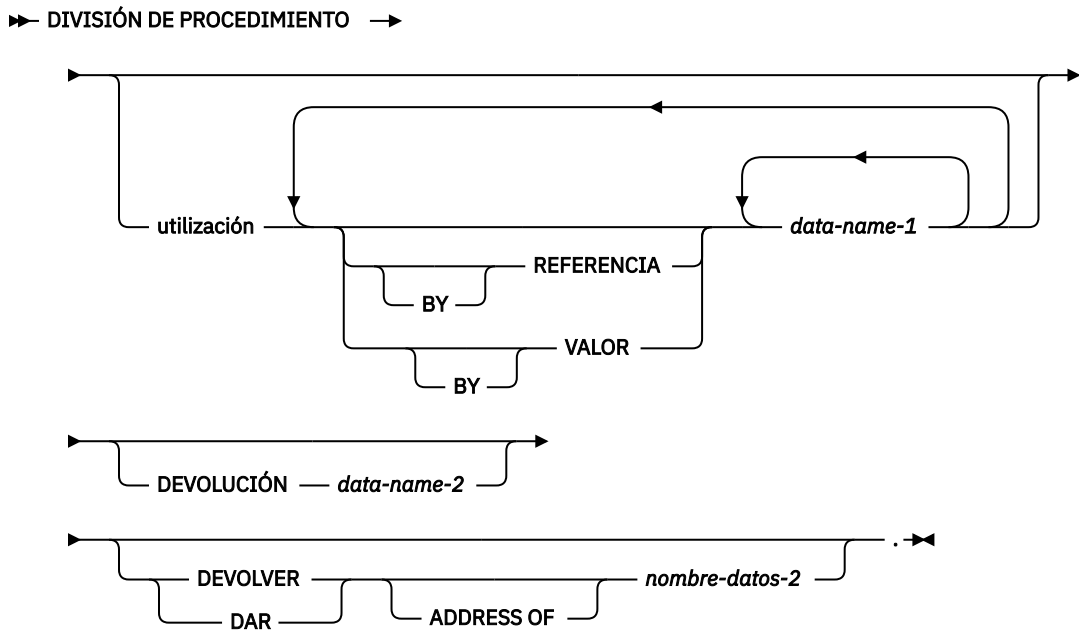
Notas:

[1] La sentencia USE se describe en "USE, sentencia" en la página 525.
[2] *nombre-sección* se puede omitir. Si omite *nombre-sección*, *nombre-párrafo* se puede omitir.
[3] Los números de prioridad no son válidos para métodos programas recursivos.

Cabecera PROCEDURE DIVISION

PROCEDURE DIVISION, si se especifica, se identifica mediante la siguiente cabecera de división de procedimiento de programa.

Formato: cabecera de división de procedimiento de programa



La frase USING

La frase USING especifica los parámetros que recibe un programa cuando se llama al programa.

La frase USING es válida en la cabecera PROCEDURE DIVISION de un subprograma llamado especificado al principio de la parte no declarativa. Cada identificador USING debe definirse como un elemento level-01 o level-77 en la LINKAGE SECTION del subprograma llamado.

En un subprograma llamado especificado en la primera sentencia ejecutable después de una sentencia ENTRY, la frase USING es válida en la sentencia ENTRY. Cada identificador USING debe definirse como un elemento level-01 o level-77 en la LINKAGE SECTION del subprograma llamado.

Sin embargo, un elemento de datos especificado en la frase USING de la sentencia CALL puede ser un elemento de datos de cualquier nivel en DATA DIVISION del programade COBOL de llamada.

Un elemento de datos de la frase USING de la cabecera puede tener una cláusula REDEFINES en su entrada de descripción de datos.

Es posible llamar a programas COBOL desde programas no COBOL o pasar parámetros de usuario desde un mandato del sistema a un programa principal COBOL.

Los argumentos de línea de mandatos siempre se pasan como tipos de datos nativos. Si especifica las opciones de compilador de tipo de datos de host CHAR (EBCDIC) o FLOAT (BE), debe especificar la frase NATIVE en la descripción de los argumentos con tipos de datos afectados por dichas opciones de compilador.

El orden de aparición de los identificadores USING en los subprogramas de llamada y de llamada determina la correspondencia de conjuntos únicos de datos disponibles para ambos. La correspondencia es posicional y no por nombre. Para llamar y llamar a subprogramas, los identificadores correspondientes deben contener el mismo número de bytes, aunque sus descripciones de datos no tienen que ser las mismas.

Para los nombres de índice, no se establece ninguna correspondencia. Los nombres de índice en los programas de llamada y llamada siempre hacen referencia a índices separados.

Los identificadores especificados en una sentencia CALL USING nombran los elementos de datos disponibles para el programa de llamada a los que se puede hacer referencia en el programa llamado. Estos elementos se pueden definir en cualquier sección DATA DIVISION.

Un identificador determinado puede aparecer más de una vez en una frase USING. Se utiliza el último valor que le ha pasado una sentencia CALL .

La frase BY REFERENCE o BY VALUE se aplica a todos los parámetros que siguen hasta que se altere temporalmente con otra frase BY REFERENCE o BY VALUE.

BY REFERENCE
Cuando se pasa un argumento BY CONTENT o BY REFERENCE, BY REFERENCE debe especificarse o implicarse para el parámetro formal correspondiente en la frase PROCEDURE o ENTRY USING.

BY REFERENCE es el valor predeterminado si no se especifica BY REFERENCE ni BY VALUE.

Si la referencia al elemento de datos correspondiente en la sentencia CALL declara que el parámetro debe pasarse BY REFERENCE (explícito o implícito), el programa se ejecuta como si cada referencia a un identificador USING en el subprograma llamado se sustituyera por una referencia al identificador USING correspondiente en el programa de llamada.

Si la referencia al elemento de datos correspondiente en la sentencia CALL declara que el parámetro se debe pasar BY CONTENT, el valor del elemento se mueve cuando se ejecuta la sentencia CALL y se coloca en un elemento de almacenamiento definido por el sistema que posee los atributos declarados en LINKAGE SECTION para *data-name-1*. La descripción de datos de cada parámetro en la frase BY CONTENT de la sentencia CALL debe ser la misma, lo que significa que no hay conversión, extensión o truncamiento, que la descripción de datos del parámetro correspondiente en la frase USING de la cabecera.

POR VALOR
Cuando se pasa un argumento BY VALUE, se pasa el valor del argumento, no una referencia al elemento de datos de envío. El subprograma o método de recepción sólo tiene acceso a una copia temporal del elemento de datos de envío. Las modificaciones realizadas en los parámetros formales que corresponden a un argumento pasado BY VALUE no afectan al argumento. Consulte *Pasar datos* en la publicación *COBOL for Linux en x86 Guía de programación* para ver ejemplos que ilustran estos conceptos.

data-name-1
data-name-1 debe ser un elemento level-01 o level-77 en LINKAGE SECTION.

Frase GIVING/DEVOLVER

La frase GIVING/DEVOLVER especifica un elemento de datos que va a recibir el resultado del programa . DEVOLVER y DAR son equivalentes.

data-name-2
data-name-2 es el elemento de datos DEVOLUCIÓN. *data-name-2* debe ser un elemento level-01 o level-77 en la LINKAGE SECTION.

El elemento de datos DEVOLUCIÓN es un parámetro de sólo salida.

No utilice la frase PROCEDURE DIVISION DEVOLVER en:

- Programas que contienen la sentencia ENTRY.
- Programas anidados.
- Programas principales: los resultados de especificar PROCEDURE DIVISION DEVOLVER en un programa principal no están definidos. Debe especificar la frase PROCEDURE DIVISION DEVOLVER sólo en los subprogramas llamados. Para los programas principales, utilice el registro especial RETURN-CODE para devolver un valor al entorno operativo.

Registro especial ADDRESS OF
Para obtener información sobre este registro, consulte “DIRECCIÓN DE” en la página 19.

Referencias a elementos en la SECCIÓN DE ENLACE

Se puede hacer referencia a los elementos de datos definidos en la LINKAGE SECTION del programa llamado dentro de la PROCEDURE DIVISION de dicho programa sólo si cumplen una de las condiciones enumeradas en el tema.

- Son operandos de la frase USING de la cabecera PROCEDURE DIVISION o de la sentencia ENTRY.
- Son operandos de SET ADDRESS OF, o CALL ... POR DIRECCIÓN DE REFERENCIA DE.
- Se definen con una cláusula REDEFINES o RENAMES, cuyo objeto satisface las condiciones anteriores.
- Son elementos subordinados a cualquier elemento que satisfaga la condición de las reglas anteriores.
- Son nombres de condición o nombres de índice asociados a elementos de datos que satisfacen cualquiera de las condiciones anteriores.

Declaraciones

Los declarativos proporcionan una o más secciones especiales que se ejecutan cuando se produce una condición excepcional.

Cuando se especifiquen las secciones declarativas, éstas se agruparán al principio de la división del procedimiento y toda la DIVISIÓN DEL PROCEDIMIENTO se dividirá en secciones.

Cada sección declarativa empieza con una sentencia USE que identifica la función de la sección. La serie de procedimientos siguientes especifican las acciones que se deben realizar cuando se produce la condición excepcional. Cada sección declarativa termina con otro nombre de sección seguido de una sentencia USE, o con las palabras clave END DECLARATIVES.

Todo el grupo de secciones declarativas va precedido por la palabra clave DECLARATIVES escrita en la línea después de la cabecera PROCEDURE DIVISION. El grupo va seguido de las palabras clave END DECLARATIVES. Las palabras clave DECLARATIVES y END DECLARATIVES deben empezar cada una en el Área A y deben ir seguidas de un punto separador. No puede aparecer ningún otro texto en la misma línea.

En la parte declarativa de PROCEDURE DIVISION, cada cabecera de sección debe ir seguida de un punto separador y debe ir seguida de una sentencia USE seguida de un punto separador. No puede aparecer ningún otro texto en la misma línea.

La sentencia USE tiene los formatos siguientes:

- "EXCEPTION/ERROR declarativo" en la página 525
- "Declarativo DEBUGGING" en la página 527

La propia sentencia USE nunca se ejecuta; en su lugar, la sentencia USE define las condiciones que ejecutan los párrafos de procedimiento siguientes, que especifican las acciones que se deben realizar. Una vez ejecutado el procedimiento, el control se devuelve a la rutina que lo ha activado.

Un procedimiento declarativo se puede realizar a partir de un procedimiento no declarativo.

Un procedimiento no declarativo puede realizarse a partir de un procedimiento declarativo.

Se puede hacer referencia a un procedimiento declarativo en una sentencia GO TO en un procedimiento declarativo.

Se puede hacer referencia a un procedimiento no declarativo en una sentencia GO TO en un procedimiento declarativo.

Puede incluir una sentencia que ejecute un procedimiento USE llamado anteriormente que todavía esté en control. Sin embargo, para evitar un bucle infinito, debe estar seguro de que hay una salida eventual en la parte inferior.

Se sale del procedimiento declarativo cuando se ejecuta la última sentencia del procedimiento.

Procedimientos

Dentro de PROCEDURE DIVISION, un *procedimiento* consta de una *sección* o un grupo de secciones, y un *párrafo* o grupo de párrafos.

Un *nombre-procedimiento* es un nombre definido por el usuario que identifica una sección o un párrafo.

Sección
Una *cabecera de sección* seguida opcionalmente de uno o más párrafos.

Cabecera de sección
Un *nombre-sección* seguido de la palabra clave SECTION, seguido de un punto separador.

Las cabeceras de sección son opcionales después de las palabras clave END DECLARATIVES o si no hay declarativos.

nombre-sección
Palabra definida por el usuario que identifica una sección. Un nombre de sección referenciado, porque no se puede calificar, debe ser exclusivo dentro del programa en el que está definido.

Una sección finaliza inmediatamente antes de la cabecera de sección siguiente, o al final de PROCEDURE DIVISION, o, en la parte declarativa, en las palabras clave END DECLARATIVES.

Párrafo
Un *nombre-párrafo* seguido de un punto de separación, opcionalmente seguido de una o más frases.

Los párrafos deben ir precedidos de un punto porque los párrafos siempre van a continuación de la cabecera IDENTIFICACIÓN DIVISION, una sección u otro párrafo, todos los cuales deben terminar con un punto.

Nombre de párrafo
Palabra definida por el usuario que identifica un párrafo. Un nombre de párrafo, porque puede estar calificado, no necesita ser exclusivo.

Si no hay declaraciones (formato 2), no se requiere un nombre de párrafo en PROCEDURE DIVISION.

Un párrafo termina inmediatamente antes del siguiente nombre de párrafo o cabecera de sección, o al final de PROCEDURE DIVISION, o, en la parte declarativa, en las palabras clave END DECLARATIVES.

No es necesario que todos los párrafos estén contenidos en secciones, incluso si uno o más párrafos están contenidos.

Frase
Una o más *sentencias* terminadas por un punto de separación.

Sentencia
Una combinación válida sintácticamente de *identificadores* y símbolos (literales, operadores relacionales, etc.) que empiezan por una sentencia COBOL.

Identificador
La palabra o palabras necesarias para hacer referencia exclusiva a un elemento de datos, incluyendo opcionalmente la calificación, la suscripción, la indexación y la modificación de referencia. En cualquier referencia PROCEDURE DIVISION (excepto la prueba de clase), el contenido de un identificador debe ser compatible con la clase especificada a través de su cláusula PICTURE; de lo contrario, los resultados son imprevisibles.

La ejecución empieza por la primera sentencia de PROCEDURE DIVISION, excluyendo los declarativos. Las sentencias se ejecutan en el orden en el que se presentan para la compilación, a menos que las reglas de sentencia dicten algún otro orden de ejecución.

El final de la DIVISIÓN DE PROCEDIMIENTO se indica mediante uno de los siguientes elementos:

- Una cabecera IDENTIFICATION DIVISION que indica el inicio de un programa fuente anidado
- Un marcador END PROGRAM

- El final físico de un programa; es decir, la posición física en un programa fuente después de la cual no se producen más líneas de programa fuente

Expresiones aritméticas

Las expresiones aritméticas se utilizan como operandos de determinadas sentencias condicionales y aritméticas.

Una expresión aritmética puede constar de cualquiera de los elementos siguientes:

1. Un identificador descrito como un elemento elemental numérico (incluidas las funciones numéricas)
2. Un literal numérico
3. La constante figurativa CERO
4. Identificadores y literales, tal como se definen en los elementos 1, 2 y 3, separados por operadores aritméticos
5. Dos expresiones aritméticas, tal como se definen en los elementos 1, 2, 3 o 4, separadas por un operador aritmético
6. Una expresión aritmética, tal como se define en los elementos 1, 2, 3, 4 o 5, entre paréntesis

Cualquier expresión aritmética puede ir precedida de un operador unario.

Los identificadores y literales que aparecen en expresiones aritméticas deben representar elementos elementales numéricos o literales numéricos en los que se puede realizar la aritmética.

Si una expresión exponencial se evalúa como un número positivo y un número negativo, el resultado es siempre el número positivo. Por ejemplo, la raíz cuadrada de 4:

```
4 ** 0.5
```

se evalúa como + 2 y -2. COBOL para Linux siempre devuelve +2.

Si el valor de una expresión que se va a elevar a una potencia es cero, el exponente debe tener un valor mayor que cero. De lo contrario, la condición de error de tamaño existe. En cualquier caso en el que no exista un número real como resultado de una evaluación, la condición de error de tamaño existe.

Operadores aritméticos

Se pueden utilizar cinco operadores aritméticos binarios y dos operadores aritméticos unarios en expresiones aritméticas. Estos operadores se representan mediante caracteres específicos que deben ir precedidos y seguidos de un espacio.

Estos operadores aritméticos binarios y unarios se muestran en Tabla 24 en la página 246.

Tabla 24. ***Operadores binarios y unarios***

Operador binario	**Significado**	**Operador unario**	**Significado**
+	Adición	+	Multiplicación por + 1
-	Resta	-	Multiplicación por -1
*	Multiplicación		
/	División		
**	Exponenciación		

Limitación: los exponentes en expresiones exponenciales de punto fijo no pueden contener más de nueve dígitos. El compilador truncará cualquier exponente con más de nueve dígitos. En el caso de truncamiento, el compilador emitirá un mensaje de diagnóstico si el exponente es un literal o constante;

si el exponente es una variable o un nombre de datos, se emitirá un mensaje de diagnóstico en tiempo de ejecución.

Los paréntesis se pueden utilizar en expresiones aritméticas para especificar el orden en el que se van a evaluar los elementos.

Las expresiones entre paréntesis se evalúan en primer lugar. Cuando las expresiones están contenidas entre paréntesis anidados, la evaluación continúa desde el conjunto menos inclusivo al conjunto más inclusivo.

Cuando no se utilizan paréntesis, o las expresiones entre paréntesis están en el mismo nivel de inclusión, se implica el siguiente orden jerárquico:

1. Operador unario
2. Exponenciación
3. Multiplicación y división
4. Suma y resta

Los paréntesis eliminan las ambigüedades en la lógica donde las operaciones consecutivas aparecen en el mismo nivel jerárquico, o modifican la secuencia jerárquica normal de ejecución cuando es necesario. Cuando el orden de las operaciones consecutivas en el mismo nivel jerárquico no se especifica completamente mediante paréntesis, el orden es de izquierda a derecha.

Una expresión aritmética sólo puede empezar con un paréntesis izquierdo, un operador unario o un operando (es decir, un identificador o un literal). Sólo puede terminar con un paréntesis derecho o un operando. Una expresión aritmética debe contener al menos una referencia a un identificador o un literal.

Debe haber una correspondencia de uno a uno entre paréntesis izquierdo y derecho en una expresión aritmética, con cada paréntesis izquierdo colocado a la izquierda de su paréntesis derecho correspondiente.

Si el primer operador de una expresión aritmética es un operador unario, debe ir inmediatamente precedido de un paréntesis izquierdo si dicha expresión aritmética va inmediatamente después de un identificador u otra expresión aritmética.

La tabla siguiente muestra los pares de símbolos aritméticos permitidos. Un par de símbolos aritméticos es la combinación de dos de estos símbolos en secuencia. En la tabla:

Sí
Indica un emparejamiento permitido.

no
Indica que el emparejamiento no está permitido.

Tabla 25. ***Pares de símbolos aritméticos válidos***

	Identificador o segundo símbolo literal	***/** +- segundo símbolo**	**Símbolo Unary + o unary-second**	**(segundo símbolo**	**) segundo símbolo**
Identificador o primer símbolo literal	No	Sí	No	No	Sí
*** / ** + - primer símbolo**	Sí	No	Sí	Sí	No
Símbolo Unary + o unary-first	Sí	No	No	Sí	No

*Tabla 25. **Pares de símbolos aritméticos válidos** (continuación)*					
	Identificador o segundo símbolo literal	***/** +- segundo símbolo**	**Símbolo Unary + o unary- second**	**(segundo símbolo**	**) segundo símbolo**
(primer símbolo	Sí	No	Sí	Sí	No
) primer símbolo	No	Sí	No	No	Sí

Aritmética con campos de fecha

Las operaciones aritméticas que incluyen un campo de fecha están restringidas a añadir una no fecha a un campo de fecha, restar una no fecha de un campo de fecha y restar un campo de fecha de un campo de fecha compatible.

Los operandos de campo de fecha son compatibles si tienen el mismo formato de fecha excepto para la parte de año, que se puede ampliar o ampliar.

Las operaciones siguientes no están permitidas:

- Cualquier operación entre fechas incompatibles
- Adición de dos campos de fecha
- Sustracción de un campo de fecha de un campo que no es de fecha
- Unario menos aplicado a un campo de fecha
- División, exponenciación o multiplicación de o por un campo de fecha
- Expresiones aritméticas que especifican un campo de fecha de último año
- Sentencias aritméticas que especifican un campo de fecha de último año, excepto como elemento de datos de recepción cuando el campo de envío no es una fecha

Las secciones siguientes describen el resultado de utilizar campos de fecha en las operaciones de suma y resta soportadas.

Para obtener más información sobre el uso de campos de fecha en operaciones aritméticas, consulte:

- “sentencia ADD” en la página 297
- “COMPUTE, sentencia” en la página 311
- “SUBTRACT, sentencia” en la página 413

Notas de uso

- Las operaciones aritméticas tratan los campos de fecha como elementos numéricos; no reconocen ninguna estructura interna específica de fecha. Por ejemplo, si se añade 1 a un campo de fecha con ventana que contiene el valor 991231 (que se puede utilizar en una aplicación para representar el 31 de diciembre de 1999), se genera el valor 991232, no 000101.
- Cuando se utilizan como operandos en expresiones aritméticas o sentencias aritméticas, los campos de fecha con ventana se expanden automáticamente según la ventana de siglo especificada por la opción de compilador YEARWINDOW..

Adición que implica campos de fecha

La tabla siguiente muestra el resultado de utilizar un campo de fecha con un operando compatible en una adición.

Tabla 26. ***Resultados de la utilización de campos de fecha además***		
	Segundo operando sin fecha	**Segundo operando de campo de fecha**
Primer operando sin fecha	No fecha	Campo de fecha
Primer operando de campo de fecha	Campo de fecha	No permitido

Para obtener detalles sobre cómo se almacena un resultado en un campo de recepción, consulte "Almacenamiento de resultados aritméticos que implican campos de fecha" en la página 249.

Resta que implica campos de fecha

Esta sección describe los resultados del uso de campos de fecha en la resta.

La tabla siguiente muestra el resultado de utilizar un campo de fecha con un operando compatible en la resta:

primer operando - segundo operando

En una sentencia SUBTRACT, estos operandos aparecen en el orden inverso:

SUBTRACT *segundo operando* FROM *primer operando*

Tabla 27. ***Resultados de la utilización de campos de fecha en la resta***		
	Segundo operando sin fecha	**Segundo operando de campo de fecha**
Primer operando sin fecha	No fecha	No permitido
Primer operando de campo de fecha	Campo de fecha	No fecha

Almacenamiento de resultados aritméticos que implican campos de fecha

Las sentencias ADD, COMPUTE, DIVIDE, MULTIPY y SUBTRACT realizan la aritmética y, a continuación, almacenan el resultado, o el campo de envío, en uno o más campos de recepción.

En una sentencia MULTIPLY, sólo los identificadores CEDER pueden ser campos de fecha. En una sentencia DIVIDE, sólo los identificadores CEDER o el identificador RESTO pueden ser campos de fecha.

Los campos de fecha con ventanas que son operandos de la expresión aritmética o sentencia se tratan como si se hubieran expandido antes de utilizar, tal como se describe en "Semántica de campos de fecha con ventanas" en la página 172.

Si el campo de envío es un campo de fecha, el campo de recepción debe ser un campo de fecha compatible. Es decir, ambos campos deben tener el mismo formato de fecha, excepto para la parte de año, que puede ser con ventana o expandida.

Si no se especifica la cláusula ON SIZE ERROR en la sentencia, la operación de almacenamiento sigue las reglas COBOL existentes para la sentencia y continúa como si los campos de recepción y envío (después de cualquier expansión automática de los operandos de campo de fecha con ventana o resultado) no fueran fechas.

Tabla 28 en la página 250 muestra cómo estas sentencias almacenan el valor de un campo de envío en un campo de recepción, donde cualquiera de los campos puede ser un campo de fecha. La sección utiliza los términos siguientes para describir cómo se realiza la tienda:

Sin ventanas
La sentencia realiza la tienda sin ningún proceso especial de error de tamaño sensible a la fecha, tal como se describe en "Frases de ERROR DE TAMAÑO" en la página 281.

Campo de envío con ventana sin fecha
El campo de envío sin fecha se trata como un campo de fecha con ventana compatible con el campo de recepción de fecha con ventana, pero con la parte de año que representa el número de años desde 1900. (Esta representación es similar a un campo de fecha con ventana con un año base de 1900, excepto que la parte del año no está limitada a un número positivo de como máximo dos dígitos.) La tienda continúa como si este supuesto año parte del campo de envío se ampliara añadiendo 1900 a él.

Con ventana con campo de envío de fecha
La tienda continúa como si todos los operandos de campo de fecha con ventana se hubieran expandido según sea necesario, de modo que el campo de envío es un campo de fecha expandido compatible.

Proceso de error de tamaño: para ambos tipos de campo de envío, si la parte del año supuesto o real del campo de envío se encuentra dentro de la ventana de siglo, el campo de envío se almacena en el campo de recepción después de eliminar el componente de siglo de la parte de año. Es decir, se conservan los dos dígitos más bajos o más a la derecha de la parte de año ampliada y se descartan los dos dígitos más altos o más a la izquierda.

Si la parte de año no se encuentra dentro de la ventana de siglo, el campo de recepción no se modifica y la sentencia de error de tamaño se ejecuta cuando se completan las operaciones aritméticas restantes.

Por ejemplo:

```
77  DUE-DATE PICTURE 9(5) DATE FORMAT YYXXX.
77  IN-DATE  PICTURE 9(8) DATE FORMAT YYYYXXX VALUE 1995001.
    ...
    COMPUTE DUE-DATE = IN-DATE + 10000
      ON SIZE ERROR imperative-statement
    END-COMPUTE
```

El campo de envío es un campo de fecha ampliado que representa el 1 de enero de 2005. Suponiendo que 2005 esté dentro de la ventana de siglo, el valor almacenado en DUE-DATE es 05001; es decir, el valor de envío de 2005001 sin el componente de siglo 20.

Tabla 28. ***Almacenamiento de resultados aritméticos que implican campos de fecha cuando se especifica ON SIZE ERROR***

	Campo de envío sin fecha	**Campo de envío de campo de fecha**
No fecha de recepción, campo	Sin ventanas	No permitido
Campo de recepción de campo de fecha con ventana	Con ventana	Con ventana
Campo de recepción de campo de fecha ampliado	Sin ventanas	Sin ventanas

Expresiones condicionales

Una expresión condicional hace que el programa objeto seleccione vías de control alternativas, en función del valor de verdad de una prueba. Las expresiones condicionales se especifican en las sentencias EVALUATE, IF, PERFORM y SEARCH.

Una expresión condicional se puede especificar en condiciones simples o en condiciones complejas. Tanto las condiciones simples como las complejas se pueden incluir entre cualquier número de paréntesis emparejados; los paréntesis no cambian si la condición es simple o compleja.

Condiciones simples

Hay cinco condiciones simples.

Las condiciones simples son:

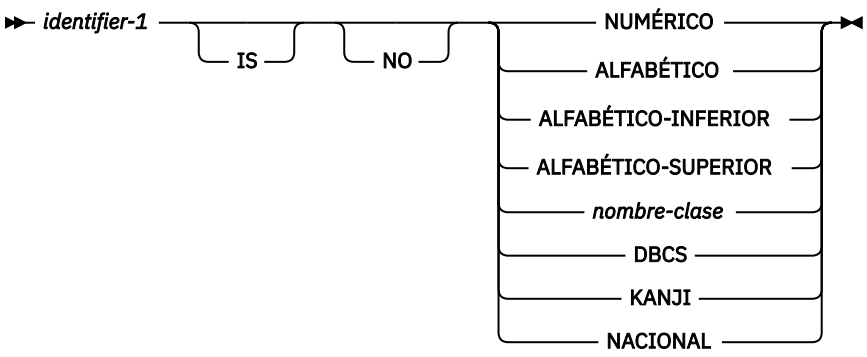
- Condición de clase
- Condición de nombre de condición
- Condición de relación
- Condición de firma
- Condición de estado de conmutador

Una condición simple tiene un valor verdadero de verdadero o falso.

Condición de clase

La condición de clase determina si el contenido de un elemento de datos es alfabético, alfabético-inferior, alfabético-superior, numérico, DBCS, KANJI, o contiene sólo los caracteres del conjunto de caracteres especificados por la cláusula CLASS tal como se define en el párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION.

Formato



identifier-1
Debe hacer referencia a un elemento de datos descrito con uno de los usos siguientes:

- DISPLAY, NATIONAL, COMPUTATIONAL-3o PACKED-DECIMAL cuando se especifica NUMERIC
- DISPLAY-1 cuando se especifica DBCS o KANJI
- DISPLAY o NATIONAL cuando se especifica ALPHABETIC, ALPHABETIC-UPPER o ALPHABETIC-LOWER
- DISPLAY cuando se especifica el nombre de clase

No debe ser de clase alfabética cuando se especifica NUMERIC.

No debe ser de clase numérica cuando se especifica ALPHABETIC, ALPHABETIC-UPPER o ALPHABETIC-LOWER.

Tabla 29 en la página 253 lista los formatos de condición de clase que son válidos para cada tipo de identificador.

Si *identifier-1* es un identificador de función, debe hacer referencia a una función alfanumérica, función nacional o función de fecha-hora.

Un elemento de grupo alfanumérico se puede utilizar en una condición de clase donde se puede utilizar un elemento alfanumérico elemental, *excepto* que la condición de clase NUMERIC no se puede utilizar si el grupo contiene uno o más elementos elementales con signo.

NACIONAL
identifier-1 consta totalmente de caracteres NACIONALES, con las reglas siguientes:

- Para elementos de datos NACIONALES, el identificador que se está probando debe describirse explícita o implícitamente como USAGE NATIONAL.
- Se realiza una comprobación de rango en la parte de datos del elemento para una representación de caracteres NATIONAL válida.

NO
Cuando se utiliza, NOT y la siguiente palabra clave definen la prueba de clase que se va a ejecutar para el valor de verdad. Por ejemplo, NOT NUMERIC es una prueba de verdad para determinar que el resultado de una prueba de clase NUMERIC es falso (en otras palabras, el elemento contiene datos que no son numéricos).

NUMÉRICO
identifier-1 consta por completo de los caracteres del 0 al 9, con o sin un signo operativo.

Si su PICTURE no contiene un signo operativo, se determina que el identificador que se está probando es numérico sólo si el contenido es numérico y no está presente un signo operativo.

Si su PICTURE contiene un signo operativo, se determina que el identificador que se está probando es numérico sólo si el elemento es un elemento elemental, el contenido es numérico y hay un signo operativo válido.

ALFABÉTICO
identifier-1 consta totalmente de cualquier combinación de los caracteres alfabéticos latinos en minúscula o en mayúscula de la A a la Z y el espacio.

ALFABÉTICO-INFERIOR
identifier-1 consta completamente de cualquier combinación de los caracteres alfabéticos latinos en minúsculas de la a a la z y el espacio.

ALFABÉTICO-SUPERIOR
identifier-1 consta completamente de cualquier combinación de los caracteres alfabéticos latinos en mayúsculas de la A a la Z y el espacio.

nombre-clase
identifier-1 consta completamente de los caracteres listados en la definición de nombre de clase en el párrafo SPECIAL-NAMES.

DBCS
identifier-1 consta totalmente de caracteres DBCS. *identifier-1* contiene caracteres DBCS que corresponden a caracteres DBCS EBCDIC válidos.

Se realiza una comprobación de rango en el elemento para una representación de caracteres válida. El rango válido es de X'41 'a X'FE' para ambos bytes de cada carácter DBCS y X'4040 ' para el espacio en blanco DBCS. (Estos rangos son para la representación de caracteres DBCS equivalente para Enterprise COBOL for z/OS, no para los rangos de valores de caracteres DBCS reales de los caracteres DBCS de la estación de trabajo.)

KANJI
identifier-1 contiene caracteres DBCS que corresponden a caracteres DBCS EBCDIC válidos.

Se realiza una comprobación de rango en el elemento para una representación de caracteres válida. El rango válido es de X'41 'a X'7E' para el primer byte, de X'41 'a X'FE' para el segundo byte y X'4040 ' para el DBCS en blanco. (Estos rangos son para la representación de caracteres DBCS equivalente para Enterprise COBOL for z/OS, no para los rangos de valores de caracteres DBCS reales de los caracteres DBCS de la estación de trabajo.)

Tabla 29. ***Formularios válidos de la condición de clase para distintos tipos de elementos de datos***		
Tipo de elemento de datos al que hace referencia *identifier-1*	**Formatos válidos de la condición de clase**	
Alfabético	ALFABÉTICO ALFABÉTICO-INFERIOR ALFABÉTICO-SUPERIOR *nombre-clase*	NO ALFABÉTICO NO ALFABÉTICO-INFERIOR NO ALFABÉTICO-SUPERIOR NOT *nombre-clase*
Alfanumérico, alfanumérico editado o numérico editado	ALFABÉTICO ALFABÉTICO-INFERIOR ALFABÉTICO-SUPERIOR NUMÉRICO *nombre-clase*	NO ALFABÉTICO NO ALFABÉTICO-INFERIOR NO ALFABÉTICO-SUPERIOR NO NUMÉRICO NOT *nombre-clase*
Decimal externo o decimal-interno	NUMÉRICO	NO NUMÉRICO
DBCS	DBCS KANJI	NO DBCS NO KANJI
Nacional	NUMÉRICO ALFABÉTICO ALFABÉTICO-INFERIOR ALFABÉTICO-SUPERIOR NACIONAL	NO NUMÉRICO NO ALFABÉTICO NO ALFABÉTICO-INFERIOR NO ALFABÉTICO-SUPERIOR NO NACIONAL
Numérico	NUMÉRICO *nombre-clase*	NO NUMÉRICO NOT *nombre-clase*
Fecha-hora	NUMÉRICO *nombre-clase*	NO NUMÉRICO NOT *nombre-clase*

Condición de nombre de condición

Una condición de nombre de condición prueba una variable condicional para determinar si su valor es igual a cualquier valor asociado con el nombre de condición.

Formato

Un nombre de condición se utiliza en condiciones como abreviatura de la condición de relación. Las reglas para comparar una variable condicional con un valor de nombre-condición son las mismas que las especificadas para las condiciones de relación.

Si *condition-name-1* se ha asociado con un rango de valores (o con varios rangos de valores), la variable condicional se prueba para determinar si su valor se encuentra dentro de los rangos, incluidos los valores finales. El resultado de la prueba es verdadero si uno de los valores que corresponde al nombre-condición es igual al valor de su variable condicional asociada.

Se permiten nombres de condición para elementos de datos alfanuméricos,booleanos, de fecha y hora, DBCS, nacional, y de coma flotante, así como otros, tal como se define para el formato de nombre de condición de la cláusula VALUE.

El ejemplo siguiente ilustra el uso de variables condicionales y nombres de condición:

```
01  AGE-GROUP         PIC  99.
    88  INFANT        VALUE 0.
    88  BABY          VALUE 1, 2.
    88  CHILD         VALUE 3 THRU 12.
    88  TEENAGER      VALUE 13 THRU 19.
```

AGE-GROUP es la variable condicional; BABY, BABY, CHILD y TEENAGER son nombres de condición. Para registros individuales en el archivo, sólo puede estar presente uno de los valores especificados en las entradas de nombre de condición.

Las sentencias IF siguientes se pueden añadir al ejemplo anterior para determinar el grupo de edad de un registro específico:

```
IF INFANT...         (Tests for value 0)
IF BABY...           (Tests for values 1, 2)
IF CHILD...          (Tests for values 3 through 12)
IF TEENAGER...       (Tests for values 13 through 19)
```

En función de la evaluación de la condición de nombre de condición, el programa objeto toma vías de acceso alternativas de ejecución.

Condiciones de nombre de condición y comparaciones de campo de fecha con ventana

Si la variable condicional es un campo de fecha con ventana, los valores asociados con sus nombres de condición se tratan como valores del campo de fecha con ventana. Es decir, se tratan como si se hubieran convertido al formato de fecha expandida.

Para obtener más información sobre la semántica de los campos de fecha con ventanas, consulte “Semántica de campos de fecha con ventanas” en la página 172.

Por ejemplo, a partir de YEARWINDOW (1945), una ventana de siglo de 1945–2044 y la definición siguiente:

```
05  DATE-FIELD   PIC 9(6) DATE FORMAT YYXXXX.
    88  DATE-TARGET       VALUE 051220.
```

un valor de 051220 en DATE-FIELD haría que se cumpla la condición siguiente:

```
IF DATE-TARGET...
```

porque el valor asociado con DATE-TARGET y el valor de DATE-FIELD se tratarían como si estuvieran precedidos por "20" antes de la comparación.

Sin embargo, la condición siguiente sería falsa:

```
IF DATE-FIELD = 051220...
```

porque en una comparación con un campo de fecha con ventana, los literales se tratan como si estuvieran precedidos por "19" independientemente de la ventana del siglo. Así que la condición anterior se convierte efectivamente:

```
IF 20051220 = 19051220...
```

Para obtener más información sobre cómo utilizar campos de fecha con ventanas en expresiones condicionales, consulte “Comparación de campos de fecha” en la página 264.

Condiciones de relación

Una condición de relación especifica la comparación de dos operandos. El operador relacional que une los dos operandos especifica el tipo de comparación. La condición de relación es verdadera si la relación especificada existe entre los dos operandos; la condición de relación es falsa si la relación especificada no existe.

Las comparaciones se definen para los casos siguientes:

- Dos operandos de clase alfabética
- Dos operandos de clase alfanumérica
- Dos operandos de clase DBCS
- Dos operarios de clase nacional
- Dos operandos de clase numérica
- Dos operandos donde uno es un entero numérico y el otro es de clase alfanumérica o nacional
- Dos operandos donde uno es de clase DBCS y el otro es de clase nacional
- Comparaciones que implican índices o elementos de datos de índice
- Dos operandos de puntero de datos
- Dos operandos de puntero de procedimiento
- Dos operandos de puntero de función
- Un grupo alfanumérico y cualquier operando que tenga el uso DISPLAY, DISPLAY-1o NATIONAL
- Dos operandos de fecha-hora

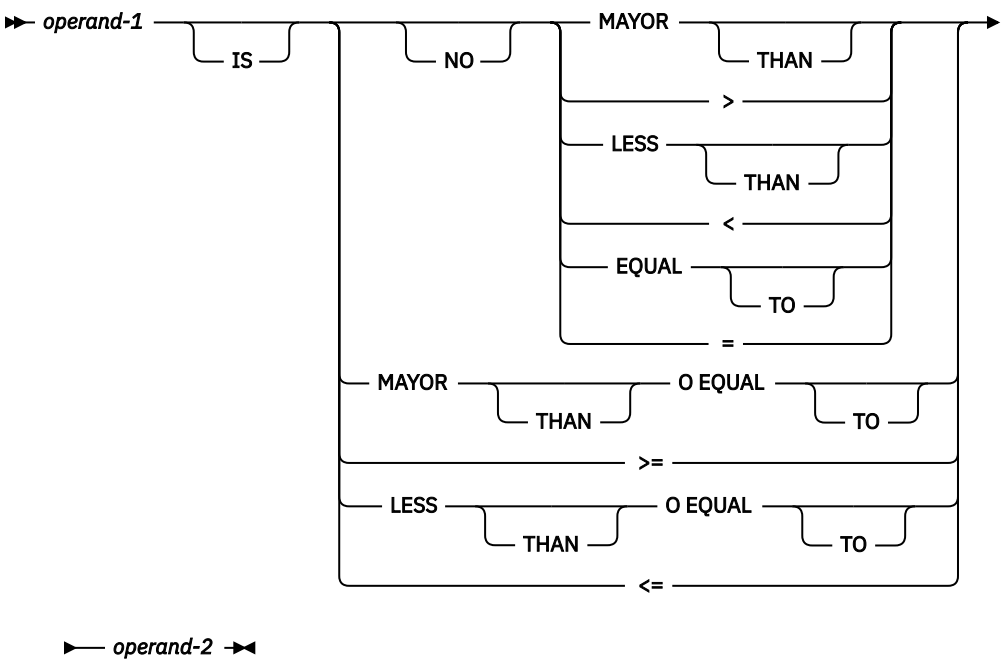
Se definen los siguientes formatos de condición de relación:

- Una condición de relación general, para comparaciones que implican sólo elementos de datos, literales, nombres de índice o elementos de datos de índice. Para obtener detalles, consulte “Condiciones generales de relación” en la página 255.
- Una condición de relación de puntero de datos. Para obtener detalles, consulte “Condiciones de relación de puntero de datos” en la página 265.
- Condición de relación de puntero de programa, para la comparación de punteros de procedimiento o punteros de función. Para obtener detalles, consulte “Condiciones de relación de puntero de procedimiento y puntero de función” en la página 267.

Condiciones generales de relación

Una condición de relación general compara dos operandos, cualquiera de los cuales puede ser un identificador, literal, expresión aritmética o nombre de índice.

Formato 1: condición de relación general



operand-1
El sujeto de la condición de relación. Puede ser un identificador, literal, identificador de función, expresión aritmética o nombre de índice.

operand-2
El objeto de la condición de relación. Puede ser un identificador, literal, identificador de función, expresión aritmética o nombre de índice.

Un literal alfanumérico puede estar entre paréntesis dentro de una condición de relación.

La condición de relación debe contener al menos una referencia a un identificador.

Los operadores relacionales, que se muestran en Tabla 30 en la página 256, especifican el tipo de comparación que se va a realizar. Cada operador relacional debe ir precedido y seguido de un espacio. Los dos caracteres de los operadores relacionales >= y <= no deben tener un espacio entre ellos.

Tabla 30. ***Operadores relacionales y sus significados***

Operador relacional	**Se puede escribir**	**Significado**
ES MAYOR QUE	IS **>**	Mayor que
NO ES MAYOR QUE	NO ES **>**	No mayor que
ES MENOR QUE	IS **<**	Menor que
NO ES MENOR QUE	NO ES **<**	No menor que
ES IGUAL A	IS **=**	Igual a
NO ES IGUAL A	NO ES **=**	No es igual a
ES MAYOR O IGUAL QUE	IS **> =**	Es mayor o igual que
ES MENOR O IGUAL QUE	IS **< =**	Es menor o igual que

En una condición de relación general, los elementos de datos, literales y constantes figurativas de clase alfabética, alfanumérica, DBCS, nacional, y numérica se comparan utilizando los siguientes tipos de comparación:

Tipo de comparación	Significado
Alfanumérico	Comparación del valor de carácter alfanumérico de dos operandos
Booleano	Comparación del valor booleano de dos operandos
Fecha-hora	Comparación de los valores de fecha u hora de dos operandos.
DBCS	Comparación del valor de caracteres DBCS de dos operandos
Nacional	Comparación del valor de carácter nacional de dos operandos
Numérico	Comparación del valor algebraico de dos operandos
Grupo	Comparación del valor de carácter alfanumérico de dos operandos, donde uno o ambos operandos es un elemento de grupo alfanumérico

Tabla 31 en la página 258 y Tabla 32 en la página 260 muestran los pares permitidos para comparaciones con distintos tipos de operandos. El tipo de comparación se indica en la intersección de fila y columna para comparaciones permitidas, utilizando la clave siguiente:

Alph
Comparación de caracteres alfanuméricos (descritos más adelante en "Comparaciones alfanuméricas" en la página 260)

Booleano
Comparación del valor booleano de dos operandos (descrito más adelante en "Comparaciones booleanas" en la página 261)

Fecha-hora
Comparación de los valores de fecha u hora (descritos más adelante en "Comparaciones de fecha-hora" en la página 261)

DBCS
Comparación de caracteres DBCS (descritos más adelante en "Comparaciones DBCS" en la página 262)

Nat
Comparación de caracteres nacionales (se describe más adelante en "Comparaciones nacionales" en la página 262)

Núm.
Comparación del valor algebraico (descrito más adelante en "Comparaciones numéricas" en la página 263)

Grupo
Comparación de caracteres alfanuméricos que implican un grupo alfanumérico (más adelante se describe en "Comparaciones de grupos" en la página 263)

(Int)
Sólo elementos enteros (combinados con el tipo de comparación Alph, Nat, Num o Group)

En blanco
La comparación no está permitida

Para ver las reglas y restricciones para las comparaciones que implican campos de fecha de último año, consulte "Comparación de campos de fecha" en la página 264.

Para ver las reglas y restricciones para las comparaciones que implican nombres de índice y elementos de datos de índice, consulte "Comparación de nombres de índice y elementos de datos de índice" en la página 264.

Introducción a Tabla 31 en la página 258: Esta tabla se organiza de la siguiente manera:

- En la primera columna, bajo "Tipo de elemento de datos o literal", cada fila identifica un tipo de operando. En algunos casos, el tipo de operando hace referencia a una agrupación de operandos que tienen propiedades comunes para la comparación. Por ejemplo, la fila para "Elementos de caracteres alfanuméricos" hace referencia a todos los tipos de operandos que se listan en la celda, como se indica a continuación:
  - Elementos de datos de categoría:
    - Alfanumérico
    - Alfanumérico-editado
    - Numérico-editado con uso DISPLAY
  - Funciones alfanuméricas
- Las cabeceras de columna posteriores hacen referencia al tipo de un operando o una agrupación de operandos. Por ejemplo, la cabecera de columna "Elementos de caracteres alfabéticos y alfanuméricos" hace referencia a los tipos de operandos identificados como "Elementos de datos alfabéticos" y a todos los tipos de operandos que se agrupan bajo el operando titulado "Elementos de caracteres alfanuméricos".
- Los literales se listan como un tipo de operando sólo en la primera columna. No aparecen como cabeceras de columna porque los literales no se pueden utilizar como ambos operandos de una condición de relación.

Tabla 31. ***Comparaciones que implican elementos de datos y literales***

Tipo de elemento de datos o literal	**Elementos de grupo alfanuméricos**	**Elementos de caracteres alfanuméricos y alfanuméricos**	**Elementos decimales con zona**	**Elementos numéricos nativos**	**Elementos alfanuméricos de coma flotante**	**Elementos de carácter nacional**	**Elementos decimales nacionales**	**Partidas nacionales de coma flotante**	**Elementos DBCS**	**Elementos booleanos**	**Elementos de fecha-hora**
Elemento de grupo alfanumérico	Grupo	Grupo	Grupo (Int)		Grupo	Grupo	Grupo (Int)	Grupo	Grupo		
Elementos de datos alfabéticos	Grupo	Alph	Alph (Int)		Alph	Nat	Alph (Int)	Nat			
Elementos de caracteres alfanuméricos: • Elementos de datos de categoría: – Alfanumérico – Alfanumérico-editado – Numérico-editado con uso DISPLAY • Funciones alfanuméricas	Grupo	Alph	Alph (Int)		Alph	Nat	Alph (Int)	Nat			
Literales alfanuméricos	Grupo	Alph	Alph (Int)		Alph	Nat	Alph (Int)	Nat			
Literales numéricos	Grupo (Int)	Alph (Int)	Número	Número	Número	Nat (Int)	Número	Número			
Elementos de datos decimales con zona	Grupo (Int)	Alph (Int)	Número	Número	Número	Nat (Int)	Número	Número			

Tabla 31. ***Comparaciones que implican elementos de datos y literales*** *(continuación)*

Tipo de elemento de datos o literal	**Elementos de grupo alfanuméricos**	**Elementos de caracteres alfanuméricos y alfanuméricos**	**Elementos decimales con zona**	**Elementos numéricos nativos**	**Elementos alfanuméricos de coma flotante**	**Elementos de carácter nacional**	**Elementos decimales nacionales**	**Partidas nacionales de coma flotante**	**Elementos DBCS**	**Elementos booleanos**	**Elementos de fecha-hora**
Elementos numéricos nativos: • Binario • Expresión aritmética • Decimal interno • Coma flotante interna **Funciones intrínsecas numéricas y enteras**			Número	Número	Número		Número	Número			
Visualizar elementos de coma flotante	Grupo	Alph	Número	Número	Número	Nat	Número	Número			
Literales de coma flotante			Número	Número	Número		Número	Número			
Elementos de caracteres nacionales: • Elementos de datos de la categoría: – Nacional – Nacional-editado – Numérico-editado con uso NATIONAL • Funciones intrínsecas nacionales • Grupos nacionales (tratados como elemento elemental)	Grupo	Nat	Nat (Int)		Nat	Nat	Nat (Int)	Nat	Nat		
Literales nacionales	Grupo	Nat	Nat (Int)		Nat	Nat	Nat (Int)	Nat	Nat		
Elementos decimales nacionales	Grupo (Int)	Alph (Int)	Número	Número	Número	Nat (Int)	Número	Número			
Elementos de coma flotante nacionales	Grupo	Nat	Número	Número	Número	Nat	Número	Número			
elementos de datos DBCS	Grupo					Nat			DBCS		
literales DBCS	Grupo					Nat			DBCS		
Elementos de datos booleanos										Alph	
Literales booleanos										Alph	
Elementos de datos de fecha-hora	Grupo		Número (Int)	Número (Int)	Alph		Número (Int)				Fecha-hora

Tabla 32. ***Comparaciones que implican constantes figurativas***

Constante figurativa	Elementos de grupo alfanuméricos	Elementos de caracteres alfanuméricos y alfanuméricos	Elementos decimales con zona	Elementos numéricos nativos	Elementos alfanuméricos de coma flotante	Elementos de carácter nacional	Elementos decimales nacionales	Elementos de coma flotante nacionales	Elementos DBCS
CERO	Grupo	Alph	Número	Número	Número	Nat	Número	Número	
ESPACIO	Grupo	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	DBCS
ALTO VALOR, LOW-VALUE PRESUPUESTO	Grupo	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	
Carácter simbólico	Grupo	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	
TODOS los literales alfanuméricos	Grupo	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	
TODOS los literales nacionales	Grupo	Nat	Nat (Int)		Nat	Nat	Nat (Int)	Nat	Nat
ALL DBCS literal	Grupo					Nat			DBCS

Comparaciones alfanuméricas

Una comparación alfanumérica es una comparación de los valores de caracteres de un solo byte de dos operandos.

Cuando uno de los operandos no es de clase alfanumérica ni alfabética de clase, dicho operando se procesa de la forma siguiente:

- Un elemento de datos de coma flotante de visualización se trata como si fuera un elemento de datos de categoría alfanumérica, en lugar de un valor numérico.
- Un operando entero decimal con zona se trata como si se hubiera movido a un elemento de datos elemental temporal de categoría alfanumérica con una longitud igual que el número total de dígitos del número, según las reglas de la sentencia MOVE.

  Cuando la opción de compilador ZWB está en vigor, el valor sin signo del operando entero se mueve al elemento de datos temporal. Cuando se especifica la opción de compilador NOZWB, el valor firmado se mueve al elemento de datos temporal. Consulte *ZWB* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener más detalles sobre la opción de compilador ZWB (NOZWB).

  A continuación, la comparación continúa con el elemento de datos temporal de la categoría alfanumérica.

Comparación de dos operandos alfanuméricos

La secuencia de clasificación utilizada para la comparación viene determinada por la cláusula PROGRAM COLLATING SEQUENCE en el párrafo OBJECT-COMPUTER y el valor de la opción de compilador COLLSEQ.

Si la cláusula PROGRAM COLLATING SEQUENCE se especifica con un nombre de alfabeto de STANDARD-1, STANDARD-2o EBCDIC, la opción de compilador COLLSEQ se ignora. La secuencia de clasificación es la asociada con el nombre de alfabeto especificado. La comparación continúa como se describe a continuación para Comparación estándar.

Si no se especifica la cláusula PROGRAM COLLATING SEQUENCE o se especifica con un nombre de alfabeto NATIVE, el método de comparación viene determinado por la opción de compilador COLLSEQ:

- Si la opción de compilador COLLSEQ (BINARY) está en vigor, la secuencia de clasificación viene determinada por los valores binarios de los caracteres. La comparación continúa como se describe a continuación para Comparación estándar.
- Si la opción de compilador COLLSEQ (EBCDIC) está en vigor, se utiliza la secuencia de clasificación EBCDIC. La comparación continúa como se describe a continuación para Comparación estándar.

- Si la opción de compilador COLLSEQ (LOCALE) está en vigor, la secuencia de clasificación la determina el entorno local de ejecución. A efectos de comparación, los espacios finales se truncan de los operandos, excepto que un operando que consta de todos los espacios se trunca en un único espacio. La comparación basada en el entorno local no es necesariamente una comparación de caracteres por caracteres. Si el operando más corto se ha ampliado con espacios como para una comparación no basada en el entorno local, es posible que el resultado no sea el resultado esperado culturalmente. Para obtener información sobre los entornos locales, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

Si la cláusula PROGRAM COLLATING SEQUENCE se especifica con un nombre de alfabeto que hace referencia a una secuencia de clasificación definida por literales, la secuencia de clasificación viene determinada por el orden de los literales especificados y la secuencia indicada por la opción de compilador COLLSEQ, tal como se describe para “cláusula ALPHABET” en la página 101. La comparación continúa como se describe a continuación para Comparación estándar.

Comparación estándar

Una comparación estándar es cualquier comparación que no se base en un entorno local. El método de comparación estándar depende de si los operandos que se van a comparar son de longitud igual o desigual.

Si los operandos son de longitud desigual, la comparación continúa como si el operando más corto se rellenara a la derecha con caracteres de espacio adecuados para hacer que los operandos sean de longitud igual. A continuación, la comparación continúa de acuerdo con las reglas para la comparación de operandos de igual longitud.

Si los operandos son de igual longitud, la comparación continúa comparando las posiciones de caracteres correspondientes en los dos operandos, empezando desde la posición más a la izquierda, hasta que se encuentren caracteres desiguales o se alcance la posición de carácter más a la derecha, lo que ocurra primero. Se determina que los operandos son iguales si todos los caracteres correspondientes son iguales.

El primer carácter desigual encontrado en los operandos se compara para determinar la relación de los operandos. El operando que contiene el carácter con el valor de clasificación más alto es el operando más grande.

Comparaciones booleanas

Una comparación alfanumérica es una comparación de los valores de caracteres de un solo byte de dos operandos.

Los operandos booleanos sólo se utilizan en la condición de relación EQUAL TO o NOT EQUAL TO. Los operandos booleanos no se pueden comparar con los operandos no booleanos. Los elementos de datos booleanos y literales deben tener una posición de longitud. Dos operandos booleanos son iguales si ambos tienen un valor de booleano 1 o booleano 0.

Comparaciones de fecha-hora

Una comparación de fecha-hora es una comparación de los valores de fecha u hora de dos operandos.

Si un elemento de la clase fecha-hora se compara con un operando no numérico (excepto para los operandos editados numéricamente), el elemento de fecha-hora se trata como si fuera no numérico.

Durante la comparación de un elemento de la clase fecha-hora con un operando numérico editado o numérico, el elemento de fecha-hora se desedita. La desedición da como resultado un entero numérico. A continuación, este entero numérico se compara numéricamente con el otro operando.

Durante la comparación de un elemento de fecha y hora con otro, los elementos se convierten primero a un formato de fecha, hora o indicación de fecha y hora común y, a continuación, se comparan. Los caracteres que forman parte de un literal de formato, pero que no son especificadores de conversión (por ejemplo, los caracteres/o-), no tienen ningún efecto en una comparación de fecha-hora.

Al comparar un elemento de fecha con un elemento de indicación de fecha y hora, sólo se tiene en cuenta la parte de fecha de la indicación de fecha y hora. Al comparar un elemento de tiempo con un elemento de indicación de fecha y hora, sólo se tiene en cuenta la parte de hora de la indicación de fecha y hora.

Comparaciones DBCS

Una comparación DBCS es una comparación de dos operandos DBCS.

Las reglas siguientes se aplican a una comparación DBCS:

- Las comparaciones de operandos DBCS se basan en una secuencia de clasificación especificada por la opción de compilador COLLSEQ:
  - Si la opción de compilador COLLSEQ (LOCALE) está en vigor, la secuencia de clasificación la determina el entorno local de ejecución. Para obtener información sobre el entorno local, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.
  - De lo contrario, la secuencia de clasificación viene determinada por los valores binarios de los caracteres DBCS.

Comparaciones nacionales

Una comparación nacional es una comparación del valor de carácter nacional de dos operandos de clase nacional.

Cuando la condición de relación especifica un operando que no es de clase nacional, dicho operando se convierte en un elemento de datos de categoría nacional antes de la comparación. La lista siguiente describe la conversión de operandos a nacionales de categoría.

DBCS
Un operando DBCS se trata como si se trasladara a un elemento de datos temporal de categoría nacional de la misma longitud que el operando DBCS. Los caracteres DBCS se convierten a los caracteres nacionales correspondientes. La página de códigos fuente utilizada para la conversión depende de si el operando DBCS es un elemento de datos EBCDIC o ASCII. Si el operando DBCS es un elemento de datos EBCDIC (es decir, la opción de compilador CHAR (EBCDIC) está en vigor y no se especifica la frase NATIVE para el operando), se utiliza la página de códigos EBCDIC en vigor para el operando; de lo contrario, se utiliza la página de códigos indicada por el entorno local en vigor para el operando. Para obtener detalles, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

Alfabético, alfanumérico, alfanumérico editado y numérico editado con uso DISPLAY
El operando se trata como si se hubiera movido a un elemento de datos temporal de categoría nacional de la longitud necesaria para representar el número de posiciones de caracteres en ese operando. Los caracteres alfanuméricos se convierten a los caracteres nacionales correspondientes. La página de códigos fuente utilizada para la conversión depende de si el operando alfanumérico es un elemento de datos EBCDIC o ASCII. Si el operando alfanumérico es un elemento de datos EBCDIC (es decir, la opción de compilador CHAR (EBCDIC) está en vigor y la frase NATIVE no se especifica para el operando), se utiliza la página de códigos EBCDIC en vigor para el operando; de lo contrario, se utiliza la página de códigos indicada por el entorno local en vigor para el operando. Para obtener detalles, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

Entero numérico
Un operando de entero numérico se trata como si se hubiera movido a un elemento de datos temporal de categoría alfanumérica de una longitud igual al número de dígitos del entero. Se utiliza el valor sin signo. A continuación, el elemento de datos temporal resultante se convierte como un operando alfanumérico.

coma flotante externa
Un elemento de coma flotante de visualización se trata como si fuera un elemento de datos de categoría alfanumérica, en lugar de un valor numérico, y luego se convierte como un operando alfanumérico.

Un elemento de coma flotante nacional se trata como si fuera un elemento de datos de categoría nacional, en lugar de un valor numérico.

Los movimientos implícitos para las conversiones se llevan a cabo de acuerdo con las reglas de la sentencia MOVE.

El elemento de datos category national resultante se utiliza en la comparación de dos operandos nacionales.

Comparación de dos operarios nacionales

El método utilizado para la comparación viene determinado por el valor de la opción de compilador NCOLLSEQ.

Si la opción de compilador NCOLLSEQ (BINARY) está en vigor, la secuencia de clasificación viene determinada por los valores binarios de los caracteres nacionales. La comparación es la siguiente:

- Si los operandos tienen una longitud desigual, la comparación continúa como si el operando más corto se rellenara a la derecha con el carácter de espacio nacional predeterminado (NX'2000') para hacer que los operandos tengan la misma longitud. A continuación, la comparación continúa de acuerdo con las reglas para la comparación de operandos de igual longitud.
- Si los operandos son de igual longitud, la comparación continúa comparando las posiciones de carácter nacional correspondientes en los dos operandos, empezando desde la posición más a la izquierda, hasta que se encuentran caracteres nacionales desiguales o se alcanza la posición de carácter nacional más a la derecha, lo que ocurra primero. Se determina que los operandos son iguales si todos los caracteres nacionales correspondientes son iguales.
- El primer carácter nacional desigual encontrado en los operandos se compara para determinar la relación de los operandos. El operando que contiene el carácter nacional con el valor de clasificación más alto es el operando más grande.

Si la opción de compilador NCOLLSEQ (LOCALE) está en vigor, la secuencia de clasificación la determina el entorno local de ejecución. A efectos de comparación, los espacios finales se truncan de los operandos, excepto que un operando que consta de todos los espacios se trunca en un único espacio. La comparación basada en el entorno local no es necesariamente una comparación de caracteres por caracteres. Si el operando más corto se ha ampliado con espacios como para una comparación no basada en el entorno local, es posible que el resultado no sea el resultado esperado culturalmente. Para obtener información sobre entornos locales, consulte Apéndice F, “Consideraciones sobre el entorno local”, en la página 585.

La cláusula PROGRAM COLLATING SEQUENCE no tiene ningún efecto en las comparaciones de los operandos nacionales.

Comparaciones numéricas

Una comparación numérica es una comparación del valor algebraico de dos operandos de clase numérica.

Cuando se comparan los valores algebraicos de operandos numéricos:

- La longitud (número de dígitos) de los operandos no es significativa.
- El uso de los operandos no es significativo.
- Los operandos numéricos sin signo se consideran positivos.
- Todos los valores cero se comparan igual; la presencia o ausencia de un signo no afecta al resultado.

Comparaciones de grupos

Una comparación de grupos es una comparación de los valores de caracteres alfanuméricos de dos operandos de .

Para la operación de comparación , cada operando se trata como si fuera un elemento de datos elemental de categoría alfanumérica del mismo tamaño que el operando, en bytes. A continuación, la comparación continúa como para dos operandos elementales de categoría alfanumérica, tal como se describe en “Comparaciones alfanuméricas” en la página 260.

Nota de uso: No hay conversión de datos para comparaciones de grupos. La comparación opera en bytes de datos sin tener en cuenta la representación de datos. El resultado de comparar un elemento elemental u operando literal con un elemento de grupo alfanumérico es predecible cuando ese operando y el contenido del elemento de grupo tienen la misma representación de datos.

Comparación de nombres de índice y elementos de datos de índice

Las comparaciones que implican nombres de índice, elementos de datos de índice o ambos se ajustan a las reglas.

Las reglas para las comparaciones son:

- La comparación de dos nombres de índice es en realidad la comparación de los números de aparición correspondientes.
- En la comparación de un nombre de índice con un elemento de datos (que no sea un elemento de datos de índice), o en la comparación de un nombre de índice con un literal, el número de aparición que corresponde al valor del nombre de índice se compara con el elemento de datos o literal.
- En la comparación de un nombre-índice con una expresión aritmética, el número de aparición que corresponde al valor del nombre-índice se compara con la expresión aritmética.

  Puesto que se puede utilizar una función de entero siempre que se pueda utilizar una expresión aritmética, puede comparar un nombre de índice con una función de entero o numérica.
- En la comparación de un elemento de datos de índice con un nombre de índice u otro elemento de datos de índice, los valores reales se comparan sin conversión. Los resultados de cualquier otra comparación que implique un elemento de datos de índice no están definidos.

En la tabla siguiente se muestran las comparaciones válidas para los nombres de índice y los elementos de datos de índice.

Tabla 33. ***Comparaciones para nombres de índice y elementos de datos de índice***

Operandos comparados	**Nombre de índice**	**Elemento de datos de índice**	**Nombre de datos (sólo entero numérico)**	**Literal (sólo entero numérico)**	**Expresión aritmética**
Nombre de índice	Comparar número de aparición	Comparar sin conversión	Comparar el número de aparición con el contenido del elemento de datos referenciado	Comparar número de aparición con literal	Comparar número de aparición con expresión aritmética
Elemento de datos de índice	Comparar sin conversión	Comparar sin conversión	No válido	No válido	No válido

Comparación de campos de fecha

Los campos de fecha pueden ser alfanuméricos de categoría, con zona decimal o decimal interno; se aplican las reglas existentes para la validez y el tipo de comparación (numérico o alfanumérico).

Por ejemplo, un campo de fecha alfanumérico no se puede comparar con un campo de fecha decimal interno. Además de estas reglas, sólo se pueden comparar dos campos de fecha si son compatibles; deben tener el mismo formato de fecha excepto para la parte del año, que puede ser con ventana o expandida.

Para los campos de fecha de último año, las únicas comparaciones soportadas son IGUAL A y NO ES IGUAL A entre dos campos de fecha de último año con formatos de fecha idénticos, o entre un campo de fecha de último año y un campo de no fecha.

Tabla 34 en la página 265 muestra las comparaciones soportadas para los campos de fecha no-año-última. Esta tabla utiliza los términos siguientes para describir cómo se realizan las comparaciones:

Sin ventanas
La comparación se realiza sin ventanas, como si los operandos no fueran fechas.

Ventanillado
La comparación se realiza como si:

1. Cualquier campo de fecha con ventana en la relación se ha expandido de acuerdo con la ventana de siglo especificada por la opción de compilador YEARWINDOW, tal como se describe en "Semántica de campos de fecha con ventanas" en la página 172.
2. Cualquier constante figurativa alfanumérica repetitiva se amplió al tamaño del campo de fecha con ventana con el que se compara, dando una comparación de no fecha alfanumérica. Las constantes figurativas alfanuméricas repetitivas incluyen CERO (en un contexto alfanumérico), SPACE, LOW-VALUE, HIGH-VALUE, QUOTE y ALL *literal*.
3. Los operandos que no son de fecha se tratan como si tuvieran el mismo formato de fecha que el campo de fecha, pero con un año base de 1900.

A continuación, la comparación se realiza de acuerdo con las reglas normales de COBOL. Las comparaciones alfanuméricas no se cambian a comparaciones numéricas con el prefijo del valor del siglo.

Tabla 34. ***Comparaciones con campos de fecha***

	No fecha segundo operando	**con ventana campo de fecha segundo operando**	**Expandido campo de fecha segundo operando**
No fecha primer operando	Sin ventanas	Con ventana[1]	Sin ventanas
Campo de fecha con ventana primer operando	Con ventana[1]	Con ventana	Con ventana
Campo de fecha expandida primer operando	Sin ventanas	Con ventana	Sin ventanas

1. **Cuando se compara con los campos de fecha con ventanas, se presupone que las no fechas contienen un año con ventanas relativo a 1900. Para obtener más detalles, consulte el punto 3 bajo la definición de comparación "WindoMié".**

Las condiciones de relación pueden contener expresiones aritméticas. Para obtener información sobre el tratamiento de campos de fecha en expresiones aritméticas, consulte "Aritmética con campos de fecha" en la página 248.

Condiciones de relación de puntero de datos

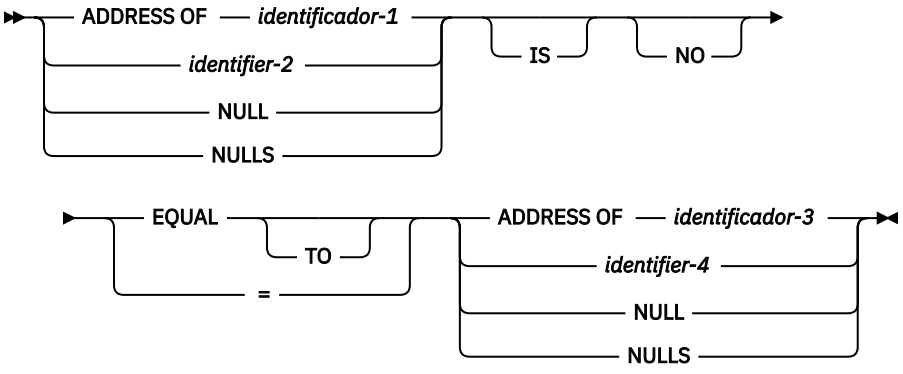
Solo se permiten EQUAL y NOT EQUAL como operadores relacionales al especificar elementos de datos de puntero.

Los elementos de datos de puntero son elementos definidos explícitamente como USAGE POINTER, o son registros especiales ADDRESS OF, que se definen implícitamente como USAGE POINTER.

Los operandos son iguales si las dos direcciones utilizadas en la comparación dan como resultado la misma ubicación de almacenamiento.

Esta condición de relación está permitida en sentencias IF, PERFORM, EVALUATE y SEARCH format-1 . No está permitido en sentencias SEARCH format-2 (SEARCH ALL) porque no hay ningún orden significativo que se pueda aplicar a elementos de datos de puntero.

Formato 2: condición de relación de puntero de datos



identifier-1 , identifier-3
Puede especificar cualquier elemento de nivel definido en LINKAGE SECTION, excepto 66 y 88.

identifier-2 , identifier-4
Debe describirse como USAGE POINTER.

NULL, NULLS
Sólo se puede utilizar si el otro operando está definido como USAGE POINTER. Es decir, NULL=NULL no está permitido.

La tabla siguiente resume las comparaciones permitidas para USAGE POINTER, NULL y ADDRESS OF.

Tabla 35. ***Comparaciones permisibles para USAGE POINTER, NULL y ADDRESS OF***

Comparaciones permitidas	**PUNTERO DE USO segundo operando**	**DIRECCIÓN DE segundo operando**	**NULL o NULLS segundo operando**
PUNTERO DE USO primer operando	Sí	Sí	Sí
DIRECCIÓN DE primer operando	Sí	Sí	Sí
NULL/NULLS primer operando	Sí	Sí	No

Sí
Comparación permitida sólo para EQUAL, NOT EQUAL

No
No se permite ninguna comparación

Condiciones de relación de puntero de procedimiento y puntero de función

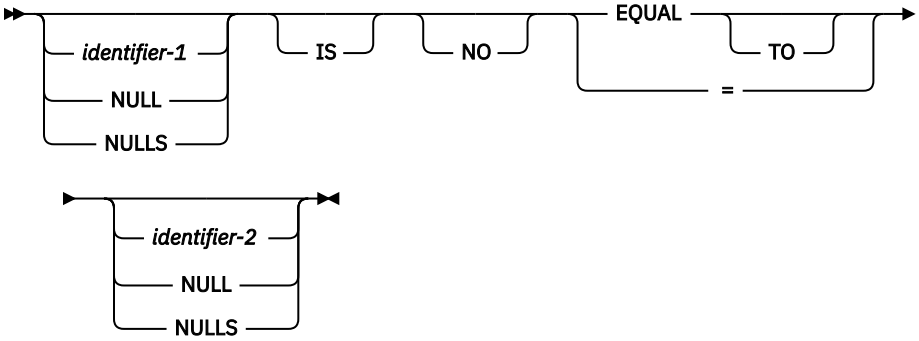
Solo se permiten EQUAL y NOT EQUAL como operadores relacionales al especificar elementos de datos de puntero de procedimiento o puntero de función en una condición de relación.

Los elementos de datos de puntero de procedimiento se definen explícitamente como USAGE PROCEDURE-POINTER. Los elementos de datos de puntero de función se definen explícitamente como USAGE FUNCTION-POINTER.

Los operandos son iguales si las dos direcciones utilizadas en la comparación dan como resultado la misma ubicación de almacenamiento.

Esta condición de relación está permitida en sentencias IF, PERFORM, EVALUATE y SEARCH format-1 . No está permitido en sentencias SEARCH format-2 (SEARCH ALL), porque no hay ningún orden significativo que se pueda aplicar a elementos de datos de puntero de procedimiento.

Formato 3: condición de relación de puntero de procedimiento y puntero de función



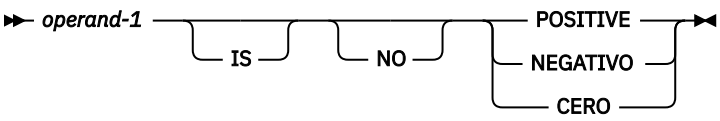
identifier-1 , identifier-2
Debe describirse como USAGE PROCEDURE-POINTER o USAGE FUNCTION-POINTER. No es necesario que *identifier-1* y *identifier-2* se describan igual.

NULL, NULLS
Sólo se puede utilizar si el otro operando está definido como USAGE FUNCTION-POINTER o USAGE PROCEDURE-POINTER. Es decir, NULL=NULL no está permitido.

Condición de firma

La condición de signo determina si el valor algebraico de un operando numérico es mayor que, menor que o igual a cero.

Formato: condición de signo



operand-1
Debe definirse como un identificador numérico o como una expresión aritmética que contenga al menos una referencia a una variable. *operand-1* se puede definir como un identificador de coma flotante.

El operando es:

- POSITIVE si su valor es mayor que cero
- NEGATIVO si su valor es menor que cero
- CERO si su valor es igual a cero

Un operando sin signo es POSITIVO o CERO.

NO
Se ejecuta una prueba algebraica para el valor de verdad de la condición de signo. Por ejemplo, NOT ZERO se considera verdadero cuando el operando probado es positivo o negativo en valor.

Campos de fecha en condiciones de signo

El operando de una condición de signo puede ser un campo de fecha, pero se trata como una no fecha para la prueba de condición de signo. Por lo tanto, si el operando es un identificador de un campo de fecha con ventana, la ventana de fecha no se realiza, por lo que la condición de signo se puede utilizar para probar un campo de fecha con ventana para un valor de todo cero.

Sin embargo, si el operando es una expresión aritmética, los campos de fecha con ventana de la expresión se expandirán durante el cálculo del resultado aritmético antes de utilizar el resultado para la prueba de condición de signo.

Por ejemplo, dado que el identificador WIN-DATE de:

- El identificador WIN-DATE se define como un campo de fecha con ventana y contiene un valor de cero
- La opción de compilador DATEPROC está en vigor
- La opción de compilador YEARWINDOW (*inicio-año*) está en vigor, con un *inicio-año* distinto de 1900

entonces esta condición de signo se evaluaría como verdadera:

```
WIN-DATE IS ZERO
```

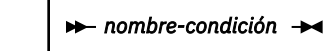
considerando que esta condición de signo se evaluaría como falsa:

```
WIN-DATE + 0 IS ZERO
```

Condición de estado de conmutador

La condición de estado de conmutador determina el estado de encendido o apagado de un conmutador UPSI.

Formato



nombre-condición
Debe definirse en el párrafo de nombres especiales como asociado con el valor activado o desactivado de un conmutador UPSI. (Consulte “Párrafo SPECIAL-NAMES” en la página 97.)

La condición de conmutación de estado prueba el valor asociado con *nombre-condición*. (El valor se considera alfanumérico.) El resultado de la prueba es verdadero si el conmutador UPSI se establece en el valor (0 o 1) correspondiente a *nombre-condición*. Para obtener detalles, consulte *UPSI* en la publicación *COBOL for Linux en x86 Guía de programación*.

Condiciones complejas

Una condición compleja se forma combinando condiciones simples, condiciones combinadas o condiciones complejas con operadores lógicos, o negando esas condiciones con la negación lógica.

Cada operador lógico debe ir precedido y seguido de un espacio. La tabla siguiente muestra los operadores lógicos y sus significados.

Tabla 36. ***Operadores lógicos y sus significados***

Operador lógico	Name	Significado
Y	Conjunción lógica	El valor de la verdad es verdadero cuando ambas condiciones son verdaderas.
O	OR inclusivo lógico	El valor de la verdad es verdadero cuando una o ambas condiciones son verdaderas.
NO	Negación lógica	Inversión del valor de la verdad (el valor de la verdad es verdadero si la condición es falsa).

A menos que se modifiquen mediante paréntesis, la siguiente lista es el orden de prioridad (de mayor a menor):

1. Operaciones aritméticas
2. Condiciones simples
3. NO
4. Y
5. O

El valor de verdad de una condición compleja (entre paréntesis o no) es el valor de verdad que resulta de la interacción de todos los operadores lógicos indicados en cualquiera de las siguientes opciones:

- Los valores de la verdad individual de las condiciones simples
- Los valores de verdad intermedios de las condiciones lógicamente combinadas o lógicamente negadas

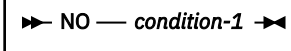
Una condición compleja puede ser una de las siguientes opciones:

- Una condición simple negada
- Una condición combinada (que se puede negar)

Condiciones simples negadas

Una condición simple se niega mediante el uso del operador lógico NOT.

Formato



La condición simple negada da el valor de verdad opuesto de la condición simple. Es decir, si el valor de la verdad de la condición simple es verdadero, entonces el valor de la verdad de esa misma condición simple negada es falso, y viceversa.

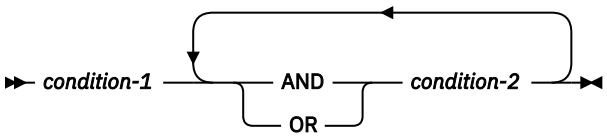
Colocar una condición simple negada entre paréntesis no cambia su valor de verdad. Es decir, las dos sentencias siguientes son equivalentes:

```
NOT A IS EQUAL TO B.
NOT (A IS EQUAL TO B).
```

Condiciones combinadas

Dos o más condiciones se pueden conectar lógicamente para formar una condición combinada.

Formato



La condición que se debe combinar puede ser cualquiera de las siguientes:

- Una condición simple
- Una condición simple negada
- Una condición combinada
- Una condición combinada negada (es decir, el operador lógico NOT seguido de una condición combinada entre paréntesis)
- Una combinación de las condiciones anteriores que se especifica de acuerdo con las reglas de la tabla siguiente

Tabla 37. ***Condiciones combinadas—secuencias de elementos permitidas***

Elemento de condición combinada	**Más a la izquierda**	**Cuando no está situado más a la izquierda, puede ir inmediatamente precedido de:**	**Más a la derecha**	**Cuando no está más a la derecha, puede ir inmediatamente seguido de:**
condición simple	Sí	O NO Y (	Sí	O Y)
O Y	No	condición simple)	No	condición simple NO (
NO	Sí	O Y (	No	condición simple (
(	Sí	O NO Y (	No	condición simple NO (
)	No	condición simple)	Sí	O Y)

Los paréntesis nunca son necesarios cuando los AND o OR (pero no ambos) se utilizan exclusivamente en una condición combinada. Sin embargo, es posible que sean necesarios paréntesis para modificar las reglas de prioridad implícitas para mantener la relación lógica correcta de operadores y operandos.

Debe haber una correspondencia de uno a uno entre paréntesis izquierdo y derecho, con cada paréntesis izquierdo a la izquierda de su paréntesis derecho correspondiente.

La tabla siguiente ilustra las relaciones entre los operadores lógicos y las condiciones C1 y C2.

Tabla 38. ***Operadores lógicos y resultados de evaluación de condiciones combinadas***

Valor para C1	**Valor para C2**	**C1 Y C2**	**C1 O C2**	**NO (C1 Y C2)**	**NO C1 Y C2**	**NO (C1 O C2)**	**NO C1 O C2**
True	True	True	True	Falso	Falso	Falso	True
Falso	True	Falso	True	True	True	Falso	True
True	Falso	Falso	True	True	Falso	Falso	Falso
Falso	Falso	Falso	Falso	True	Falso	True	True

Orden de evaluación de las condiciones

Los paréntesis, tanto explícitos como implícitos, definen el nivel de inclusión dentro de una condición compleja. Dos o más condiciones conectadas sólo por los operadores lógicos AND u OR en el mismo nivel de inclusión establecen un nivel jerárquico dentro de una condición compleja. Por lo tanto, una condición compleja completa es una estructura anidada de niveles jerárquicos, siendo toda la condición compleja el nivel jerárquico más inclusivo.

Dentro de este contexto, la evaluación de las condiciones dentro de una condición compleja completa comienza a la izquierda de la condición. Las condiciones conectadas constituyentes dentro de un nivel jerarquico son evaluadas en orden de izquierda a derecha, y la evaluacion de ese nivel jerarquico termina tan pronto como se determina un valor de verdad para el mismo, sin importar si todas las condiciones conectadas constituyentes dentro de ese nivel jerarquico han sido evaluadas.

Los valores se establecen para expresiones aritméticas y funciones si y cuando se evalúan las condiciones que las contienen. Del mismo modo, las condiciones negadas se evalúan si y cuándo es necesario evaluar la condición compleja que representan. Por ejemplo:

```
NOT A IS GREATER THAN B OR A + B IS EQUAL TO C AND D IS POSITIVE
```

se evalúa como si estuviera entre paréntesis como se indica a continuación:

```
(NOT (A IS GREATER THAN B)) OR
(((A + B) IS EQUAL TO C) AND (D IS POSITIVE))
```

Orden de evaluación:

1. `(NOT (A IS GREATER THAN B))` se evalúa, dando algún valor de verdad intermedio, *t1*. Si *t1* es true, la condición combinada es true y no se realiza ninguna evaluación adicional. Si *t1* es false, la evaluación continúa como se indica a continuación.
2. `(A + B)` se evalúa, dando algún resultado intermedio, *x*.
3. `(x IS EQUAL TO C)` se evalúa, dando algún valor de verdad intermedio, *t2*. Si *t2* es false, la condición combinada es false y no se realiza ninguna evaluación adicional. Si *t2* es true, la evaluación continúa como se indica a continuación.
4. `(D IS POSITIVE)` se evalúa, dando algún valor de verdad intermedio, *t3*. Si *t3* es false, la condición combinada es false. Si *t3* es true, la condición combinada es true.

Ejemplo

El ejemplo siguiente muestra una sentencia IF con dos condiciones combinadas con el operador AND lógico.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. COMBINE.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 N1 PIC 9(2) VALUE 30.
01 N2 PIC 9(2) VALUE 45.
01 N3 PIC 9(2) VALUE 30.

PROCEDURE DIVISION.
A000-FIRST-PARA.

   IF N1 IS LESS THAN N2 AND N1 = N3 THEN
         DISPLAY 'BOTH CONDITIONS OK'
   ELSE
         DISPLAY 'EITHER OR BOTH CONDITIONS FAILED'
   END-IF.

   STOP RUN.
```

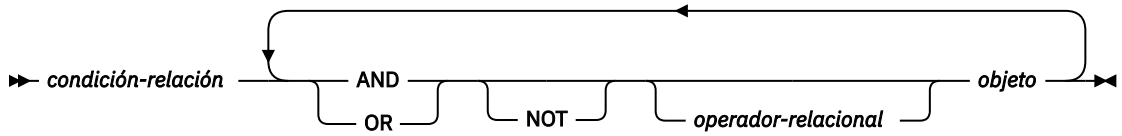
El ejemplo genera la salida siguiente:

```
BOTH CONDITIONS OK
```

Condiciones de relación combinadas abreviadas

Cuando las condiciones-relación son escritas consecutivamente, cualquier condición-relación después de la primera puede ser abreviada por omisión del sujeto, o por omisión del sujeto y operador relacional.

Formato



En cualquier secuencia consecutiva de condiciones de relación, se pueden especificar ambas formas de abreviatura. La condición abreviada se evalúa como si:

1. El último tema declarado es el tema que falta.
2. El último operador relacional indicado es el operador relacional que falta.

La condición combinada resultante debe cumplir las reglas para las secuencias de elementos en condiciones combinadas, tal como se muestra en la “Condiciones combinadas” en la página 269.

Si NOT va inmediatamente seguido de MAYOR QUE, >, MENOR QUE, <, IGUAL A o =, el NOT participa como parte del operador relacional. NOT en cualquier otra posición se considera un operador lógico (y por lo tanto da como resultado una condición de relación negada).

Utilización de paréntesis

Puede utilizar paréntesis en condiciones de relación combinadas para especificar un orden de evaluación previsto. El uso de paréntesis también puede ayudar a mejorar la legibilidad de las expresiones condicionales.

Las reglas siguientes rigen el uso de paréntesis en condiciones de relación combinadas abreviadas:

1. Se pueden utilizar paréntesis para cambiar el orden de evaluación de los operadores lógicos AND y OR.

2. La palabra NOT participa como parte del operador relacional cuando va inmediatamente seguida de MAYOR QUE, >, MENOR QUE, <, EQUAL TO o =.
3. NOT en cualquier otra posición se considera un operador lógico y, por lo tanto, da como resultado una condición de relación negada. Si utiliza NOT como operador lógico, sólo se niega la condición de relación inmediatamente después de NOT; la negación no se propaga a través de la condición de relación combinada abreviada junto con el sujeto y el operador relacional.
4. El operador NOT lógico puede aparecer dentro de una expresión entre paréntesis que sigue inmediatamente a un operador relacional.
5. Cuando aparece un paréntesis izquierdo inmediatamente después del operador relacional, el operador relacional se distribuye a todos los objetos incluidos entre paréntesis. En el caso de un operador relacional "distribuido", el sujeto y el operador relacional permanecen actuales después del paréntesis derecho que finaliza la distribución. Las tres restricciones siguientes se aplican a los casos en los que el operador relacional se distribuye por toda la expresión:
   a. Una condición simple no puede aparecer dentro del ámbito de la distribución.
   b. Otro operador relacional no puede aparecer dentro del ámbito de la distribución.
   c. El operador lógico NOT no puede aparecer inmediatamente después del paréntesis izquierdo, que define el ámbito de la distribución.
6. La evaluación procede de la condición menos inclusiva a la más inclusiva.
7. Debe haber una correspondencia de uno a uno entre paréntesis izquierdo y derecho, con cada paréntesis izquierdo a la izquierda de su paréntesis derecho correspondiente. Si los paréntesis no están equilibrados, el compilador inserta un paréntesis y emite un mensaje de nivel E. Sin embargo, si el paréntesis insertado por el compilador da como resultado el truncamiento de la expresión, recibirá un mensaje de diagnóstico de nivel S.
8. El último asunto indicado se inserta en lugar del asunto que falta.
9. El último operador relacional indicado se inserta en lugar del operador relacional que falta.
10. La inserción del sujeto omitido o del operador relacional finaliza cuando:
    a. Se ha encontrado otra condición simple.
    b. Se ha encontrado un nombre de condición.
    c. Se ha encontrado un paréntesis derecho que coincide con un paréntesis izquierdo que aparece a la izquierda del sujeto.
11. En cualquier secuencia consecutiva de condiciones de relación, puede utilizar tanto las condiciones de relación abreviadas que contienen paréntesis como las que no.
12. Los operadores NOT lógicos consecutivos se cancelan entre sí y dan como resultado un mensaje de nivel S. Sin embargo, tenga en cuenta que una condición de relación combinada abreviada puede contener dos operadores NOT consecutivos cuando el segundo NOT forma parte de un operador relacional. Por ejemplo, puede abreviar la primera condición como la segunda condición listada a continuación.

```
A = B and not A not = C
A = B and not not = C
```

La tabla siguiente resume las reglas para formar una condición de relación combinada abreviada.

Tabla 39. ***Condiciones combinadas abreviadas: secuencias de elementos permitidas***				
Elemento de condición combinada	**Más a la izquierda**	**Cuando no está situado más a la izquierda, puede ir inmediatamente precedido de:**	**Más a la derecha**	**Cuando no está más a la derecha, puede ir inmediatamente seguido de:**
Asunto	Sí	NO (	No	Operador relacional
objeto	No	Operador relacional Y O NO (	Sí	Y O)
Operador relacional	No	Asunto Y O NO	No	objeto (
Y O	No	objeto)	No	objeto Operador relacional NO (
NO	Sí	Y O (	No	Asunto objeto Operador relacional (
(	Sí	Operador relacional Y O NO (	No	Asunto objeto NO (
)	No	objeto)	Sí	Y O)

La tabla siguiente muestra ejemplos de condiciones de relación combinadas abreviadas, con y sin paréntesis, y sus equivalentes no abreviados.

Tabla 40. ***Condiciones combinadas abreviadas: equivalentes no abreviados***	
Condición de relación combinada abreviada	**Equivalente**
A = B AND NOT < C OR D	((A = B) AND (A NOT < C)) OR (A NOT < D)
A NO > B O C	(A NOT > B) O (A NOT > C)
NO A = B O C	(NOT (A = B)) O (A = C)
NOT (A = B O < C)	NOT ((A = B) O (A < C))

Tabla 40. ***Condiciones combinadas abreviadas: equivalentes no abreviados*** *(continuación)*	
Condición de relación combinada abreviada	**Equivalente**
NOT (A NOT = B AND C AND NOT D)	NOT ((((A NOT = B) AND (A NOT = C)) AND (NOT (A NOT = D))))

Categorías de sentencia

Hay cuatro categorías de sentencias COBOL: sentencias imperativas, sentencias condicionales, sentencias de ámbito delimitado y sentencias de dirección de compilador. Consulte los temas siguientes para obtener más detalles.

Declaraciones imperativas

Una *sentencia imperativa* especifica una acción incondicional que debe realizar el programa o es una sentencia condicional terminada por su terminador de ámbito explícito.

Se puede especificar una serie de sentencias imperativas siempre que se permita una sentencia imperativa. Una sentencia condicional que termina por su terminador de ámbito explícito también se clasifica como una sentencia imperativa.

Para obtener más información sobre el terminador de ámbito explícito, consulte “Sentencias de ámbito delimitado” en la página 278.

Las listas siguientes contienen las sentencias imperativas COBOL.

Aritmética

- ADD[1]
- COMPUTE[1]
- DIVIDIR[1]
- MULTIPLY[1]
- SUBTRACT[1]

1. Sin la frase ON SIZE ERROR o NOT ON SIZE ERROR.

Movimiento de datos

- ACEPTAR (FECHA, DÍA, DÍA DE LA SEMANA, TIME)
- INICIALIZAR
- INSPECT
- MOVE
- SET
- STRING[2]
- UNSTRING[2]
- XML GENERATE[3]
- XML PARSE[3]

2. Sin la frase ON OVERFLOW o NOT ON OVERFLOW.

3. Sin la frase ON EXCEPTION o NOT ON EXCEPTION.

Finalizando

- DETENER EJECUCIÓN

- EXIT PROGRAM
- GOBACK

Entrada-salida

- ACCEPT *identificador*
- CERRAR
- DELETE[4]
- VISUALIZAR
- OPEN
- LEA[5]
- REWRITE[4]
- START[4]
- STOP *literal*
- UNLOCK
- GRABACIÓN[6]

4. Sin la frase INVALID KEY o NOT INVALID KEY.

5. Sin las frases AT END o NOT AT END, y INVALID KEY o NOT INVALID KEY.

6. Sin las frases INVALID KEY o NOT INVALID KEY, y END-OF-PAGE o NOT END-OF-PAGE.

Ordenación

- Formato 1 SORT
- MERGE
- RELEASE
- RETURN[7]

7. Sin la frase AT END o NOT AT END.

Procedimiento-bifurcación

- ALTER
- CONTINUAR
- Formato 1 EXIT
- VA A
- PERFORM

Enlace de programa

- CALL[8]
- CANCEL

8. Sin la frase ON OVERFLOW, y sin la frase ON EXCEPTION o NOT ON EXCEPTION.

Manejo de tablas

- Formato 2 SORT (tabla SORT)
- SET

Sentencias condicionales

Una *sentencia condicional* especifica que el valor de verdad de una condición debe determinarse y que la acción posterior del programa objeto depende de este valor de verdad.

Para obtener más información sobre las expresiones condicionales, consulte "Expresiones condicionales" en la página 250.)

Las listas siguientes contienen sentencias COBOL que pasan a ser condicionales cuando se incluye una *condición* (por ejemplo, ON SIZE ERROR o ON OVERFLOW) y cuando la sentencia no termina por su terminador de ámbito explícito.

Aritmética

- ADD ... ERROR EN TAMAÑO
- ADD ... NO EN ERROR DE TAMAÑO
- COMPUTE ... ERROR EN TAMAÑO
- COMPUTE ... NO EN ERROR DE TAMAÑO
- DIVIDIR ... ERROR EN TAMAÑO
- DIVIDIR ... NO EN ERROR DE TAMAÑO
- MULTIPLY ... ERROR EN TAMAÑO
- MULTIPLY ... NO EN ERROR DE TAMAÑO
- SUBTRACT ... ERROR EN TAMAÑO
- SUBTRACT ... NO EN ERROR DE TAMAÑO

Movimiento de datos

- CADENA ... EN DESBORDAMIENTO
- CADENA ... NO EN DESBORDAMIENTO
- UNSTRING ... EN DESBORDAMIENTO
- UNSTRING ... NO EN DESBORDAMIENTO
- XML GENERATE ... EN EXCEPCIÓN
- XML GENERATE ... NO EN EXCEPCIÓN
- XML PARSE ... EN EXCEPCIÓN
- XML PARSE ... NO EN EXCEPCIÓN

Decisión

- SI
- EVALUAR

Entrada-salida

- DELETE ... CLAVE NO VÁLIDA
- DELETE ... CLAVE NO VÁLIDA
- LEA ... AT END
- LEA ... NO EN FIN
- LEA ... CLAVE NO VÁLIDA
- LEA ... CLAVE NO VÁLIDA
- REWRITE ... CLAVE NO VÁLIDA
- REWRITE ... CLAVE NO VÁLIDA

- INICIO ... CLAVE NO VÁLIDA
- INICIO ... CLAVE NO VÁLIDA
- GRABACIÓN ... AT END-OF-PAGE
- GRABACIÓN ... NOT AT END-OF-PAGE
- GRABACIÓN ... CLAVE NO VÁLIDA
- GRABACIÓN ... CLAVE NO VÁLIDA

Ordenación

- DEVOLVER ... AT END
- DEVOLVER ... NO EN FIN

Enlace de programa

- CALL ... EN DESBORDAMIENTO
- CALL ... EN EXCEPCIÓN
- CALL ... NO EN EXCEPCIÓN

Manejo de tablas

- BUSCAR

Sentencias de ámbito delimitado

En general, una sentencia DELIMITED SCOPE utiliza un terminador de ámbito explícito para convertir una sentencia condicional en una sentencia imperativa.

A continuación, se puede anidar la sentencia imperativa resultante. Los terminadores de ámbito explícitos también se pueden utilizar para terminar el ámbito de una sentencia imperativa. Se proporcionan terminadores de ámbito explícitos para todas las sentencias COBOL que pueden tener frases condicionales.

A menos que se especifique explícitamente lo contrario, se puede especificar una sentencia de ámbito delimitado siempre que las reglas del idioma permitan una sentencia imperativa.

Terminadores de ámbito explícitos

Un *terminador de ámbito explícito* marca el final de determinadas sentencias PROCEDURE DIVISION.

Una sentencia condicional delimitada por su terminador de ámbito explícito se considera una sentencia imperativa y debe seguir las reglas de las sentencias imperativas.

Estos son los terminadores de ámbito explícitos:

- END-ADD
- FIN-CALL
- END-COMPUTE
- FIN-DELETE
- END-DIVIDIR
- END-EVALUAR
- END-IF
- END-MULTIPLY
- FIN-PERFORM
- END-LEER
- FIN-RETORNO

- END-REWRITE
- FIN-BUSCAR
- END-START
- FIN-STRING
- END-SUBTRACT
- FIN-DESCABEZADO
- FIN-GRABACIÓN
- END-XML

Terminadores de ámbito implícitos

Al final de cualquier frase, un *terminador de ámbito implícito* es un punto de separación que termina el ámbito de todas las sentencias anteriores que todavía no han terminado.

Otra sentencia no puede contener una sentencia condicional no terminada.

Excepto para anidar sentencias condicionales dentro de sentencias IF, las sentencias anidadas deben ser sentencias imperativas y deben seguir las reglas para sentencias imperativas. No debe anidar sentencias condicionales.

Referencias relacionadas
RULES (*COBOL for Linux en x86 Guía de programación*)

Sentencias de direccionamiento de compilador

Una sentencia de direccionamiento de compilador es una sentencia que hace que el compilador realice una acción específica durante la compilación.

Para obtener más información sobre las sentencias que indican al compilador que realice una acción especificada, consulte Capítulo 21, “Sentencias de direccionamiento de compilador”, en la página 505.

Operaciones de sentencia

El tema muestra los tipos de operaciones realizadas por sentencias COBOL.

Las sentencias COBOL realizan los siguientes tipos de operaciones:

- Aritmética
- Manipulación de datos
- Entrada/salida
- Ramificación de procedimiento

Existen varias frases comunes a las sentencias aritméticas y de manipulación de datos, como por ejemplo:

- Frase CORRESPONDIENTE
- Frase CEDER
- Frase REDONDEADA
- Frases de ERROR DE TAMAÑO

Frase CORRESPONDIENTE

La frase CORRESPONDIENTE (CORR) hace que las operaciones ADD, SUBTRACT y MOVE se realicen en elementos de datos elementales con el mismo nombre si se especifica el elemento de grupo alfanumérico o el elemento de grupo nacional al que pertenecen.

Un grupo nacional se procesa como un elemento de grupo cuando se utiliza la frase CORRESPONDIENTE.

Ambos identificadores que siguen a la palabra clave CORRESPONDIENTE deben nombrar elementos de grupo. En esta discusión, estos identificadores se conocen como *identifier-1* y *identifier-2*. *identifier-1* hace referencia al elemento de grupo de envío. *identifier-2* hace referencia al elemento de grupo receptor.

Dos elementos de datos subordinados, uno de *identifier-1* y otro de *identifier-2*, se corresponden si se cumplen las condiciones siguientes:

- En una sentencia ADD o SUBTRACT, ambos elementos de datos son elementos de datos numéricos elementales. Se ignoran otros elementos de datos.
- En una sentencia MOVE, al menos uno de los elementos de datos es un elemento elemental y el movimiento está permitido por las reglas de movimiento.
- Los dos elementos subordinados tienen el mismo nombre y los mismos calificadores hasta, pero sin incluir, *identifier-1* y *identifier-2*.
- Los elementos subordinados no se identifican mediante la palabra clave FILLER.
- Ni *identifier-1* ni *identifier-2* se describe como un elemento de nivel 66, 77 o 88, y tampoco se describe como un elemento de datos de índice. Ni *identifier-1* ni *identifier-2* se pueden modificar por referencia.
- Ni *identifier-1* ni *identifier-2* se describe con USAGE POINTER, USAGE FUNCTION-POINTER, o USAGE PROCEDURE-POINTER.
- Los elementos subordinados no incluyen una cláusula REDEFINES, RENAMES, OCCURS, USAGE INDEX, USAGE POINTER, USAGE PROCEDURE-POINTER, o USAGE FUNCTION-POINTER en sus descripciones.

  Sin embargo, los propios *identifier-1* y *identifier-2* pueden contener o estar subordinados a elementos que contienen una cláusula REDEFINES o OCCURS en sus descripciones.
- El nombre de cada elemento de datos subordinado que cumple estas condiciones es exclusivo después de la aplicación de calificadores implícitos.

identifier-1, *identifier-2*, o ambos, pueden estar subordinados a un elemento FILLER.

Por ejemplo, considere dos jerarquías de datos definidas de la siguiente manera:

```
05 ITEM-1 OCCURS 6.
   10  ITEM-A PIC S9(3).
   10  ITEM-B PIC +99.9.
   10  ITEM-C PIC X(4).
   10  ITEM-D REDEFINES ITEM-C PIC 9(4).
   10  ITEM-E USAGE COMP-1.
   10  ITEM-F USAGE INDEX.
05 ITEM-2.
   10  ITEM-A PIC 99.
   10  ITEM-B PIC +9V9.
   10  ITEM-C PIC A(4).
   10  ITEM-D PIC 9(4).
   10  ITEM-E PIC 9(9) USAGE COMP.
   10  ITEM-F USAGE INDEX.
```

Si se especifica `ADD CORR ITEM-2 TO ITEM-1(x)` , `ITEM-A` y `ITEM-A(x)`, `ITEM-B` y `ITEM-B(x)`, y `ITEM-E` y `ITEM-E(x)` se consideran correspondientes y se añaden juntos. `ITEM-C` y `ITEM-C(x)` no se incluyen porque no son numéricos. `ITEM-D` y `ITEM-D(x)` no se incluyen porque `ITEM-D(x)` incluye una cláusula REDEFINES en su descripción de datos. `ITEM-F` y `ITEM-F(x)` no se incluyen porque son elementos de datos de índice. Tenga en cuenta que `ITEM-1` es válido como *identifier-1* o *identifier-2*.

Si alguna de las operaciones individuales de la sentencia ADD CORRESPONDIENTE produce una condición de error de tamaño, *imperative-statement-1* en la frase ON SIZE ERROR no se ejecuta hasta que se hayan completado todas las adiciones individuales.

Frase CEDER

Para sentencias aritméticas, el valor del identificador que sigue a la palabra CEDER se establece igual al resultado calculado de la operación aritmética. Puesto que este identificador no está implicado en el cálculo, puede ser un elemento editado numérica-editado.

Frase REDONDEADA

Después de la alineación de coma decimal, el número de lugares en la fracción del resultado de una operación aritmética se compara con el número de lugares proporcionados para la fracción del identificador resultante.

Cuando el tamaño del resultado fraccional excede el número de lugares proporcionados para su almacenamiento, se produce un truncamiento a menos que se especifique REDONDEADO. Cuando se especifica REDONDEADO, el dígito menos significativo del identificador resultante se incrementa en 1 siempre que el dígito más significativo del exceso sea mayor o igual que 5.

Cuando el identificador resultante se describe mediante una cláusula PICTURE que contiene Ps más a la derecha y cuando el número de lugares en el resultado calculado excede el número de posiciones enteras especificadas, el redondeo o truncamiento se produce en relación con la posición entera más a la derecha para la que se asigna el almacenamiento.

En una operación aritmética de coma flotante, la frase REDONDEADA no tiene ningún efecto; el resultado de una operación de coma flotante siempre se redondea. Para obtener más información sobre las expresiones aritméticas de coma flotante, consulte *Punto fijo contrastado con aritmética de coma flotante* en la publicación *COBOL for Linux en x86 Guía de programación*.

Cuando la opción de compilador ARITH (EXTEND) está en vigor, la frase REDONDEADA no está soportada para receptores aritméticos con posiciones de 31 dígitos a la derecha de la coma decimal. Por ejemplo, ni X ni Y son válidos como receptores con la frase REDONDEADA:

```
01  X PIC V31.
01  Y PIC P(30)9(1).
    . . .
    COMPUTE X ROUNDED = A + B
    COMPUTE Y ROUNDED = A - B
```

De lo contrario, la frase REDONDEADA está totalmente soportada para las sentencias aritméticas de precisión ampliada.

Frases de ERROR DE TAMAÑO

Una condición de error de tamaño puede producirse de diferentes maneras.

- Cuando el valor absoluto del resultado de una evaluación aritmética, después de la alineación de coma decimal, excede el valor más grande que puede estar contenido en el campo de resultado.
- Cuando se produce la división por cero.
- Cuando el resultado de una sentencia aritmética se almacena en un campo de fecha con ventana y el año del resultado queda fuera de la ventana del siglo. Por ejemplo, dada YEARWINDOW (1940) (que especifica una ventana de siglo de 1940–2039), la siguiente sentencia SUBTRACT provoca un error de tamaño:

  ```
  01  WINDOWED-YEAR  DATE FORMAT YY PICTURE 99
                     VALUE IS 50.
      ...
      SUBTRACT 20 FROM WINDOWED-YEAR
               ON SIZE ERROR imperative-statement
  ```

  El error de tamaño se produce porque el resultado de la resta, un campo de fecha con ventana, tiene un valor de año efectivo de 1930, que queda fuera de la ventana del siglo. Para obtener detalles sobre cómo se tratan los campos de fecha con ventanas como si se hubieran convertido al formato de fecha expandido, consulte “Resta que implica campos de fecha” en la página 249.

  Para obtener más información sobre cómo se pueden producir errores de tamaño al utilizar campos de fecha, consulte “Almacenamiento de resultados aritméticos que implican campos de fecha” en la página 249.
- En una expresión exponencial, tal como se indica en la tabla siguiente:

Tabla 41. ***Condiciones de error de tamaño de exposición***		
Error de tamaño	**Acción realizada cuando existe una cláusula SIZE ERROR**	**Acción realizada cuando no existe una cláusula SIZE ERROR**
Cero elevado a cero potencia	Se ejecuta el imperativo SIZE ERROR.	El valor devuelto es 1 y se emite un mensaje.
Cero elevado a un número negativo	Se ejecuta el imperativo SIZE ERROR.	El programa termina de forma anómala.
Un número negativo elevado a una potencia fraccional	Se ejecuta el imperativo SIZE ERROR.	Se utiliza el valor absoluto de la base y se emite un mensaje.

La condición de error de tamaño sólo se aplica a los resultados finales, no a los resultados intermedios.

Si el identificador resultante se define con el uso BINARY, COMPUTATIONAL, COMPUTATIONAL-4o COMPUTATIONAL-5, el valor más grande que puede contener el elemento de datos resultante es el valor implícito en la serie de caracteres PICTURE decimal del elemento, independientemente de la opción de compilador TRUNC en vigor.

Si se especifica la frase REDONDEADA, el redondeo tiene lugar antes de la comprobación de errores de tamaño.

Cuando se produce un error de tamaño, la acción posterior del programa depende de si se especifica la frase ON SIZE ERROR.

Si no se especifica la frase ON SIZE ERROR y se produce una condición de error de tamaño, se aplican las reglas de truncamiento y se calcula el valor del identificador resultante afectado.

Si se especifica la frase ON SIZE ERROR y se produce una condición de error de tamaño, el valor del identificador resultante afectado por el error de tamaño no se altera; es decir, los resultados del error no se colocan en el identificador de recepción. Después de completar la ejecución de la operación aritmética, se ejecuta la sentencia imperativa de la frase ON SIZE ERROR, el control se transfiere al final de la sentencia aritmética y, si se especifica, se ignora la frase NOT ON SIZE ERROR.

Para las sentencias ADD CORRESPONDIENTE y SUBTRACT CORRESPONDIENTE, si una operación aritmética individual provoca una condición de error de tamaño, la sentencia imperativa ON SIZE ERROR no se ejecuta hasta que se hayan completado todas las adiciones o restas individuales.

Si se ha especificado la frase NOT ON SIZE ERROR y, después de la ejecución de una operación aritmética, no existe una condición de error de tamaño, se ejecuta la frase NOT ON SIZE ERROR.

Cuando se especifican las frases ON SIZE ERROR y NOT ON SIZE ERROR y la sentencia de la frase que se ejecuta no contiene ninguna transferencia explícita de control, si es necesario se realiza una transferencia implícita de control después de la ejecución de la frase al final de la sentencia aritmética.

Sentencias aritméticas

Las sentencias aritméticas se utilizan para los cálculos. Las operaciones individuales se especifican mediante las sentencias ADD, SUBTRACT, MULTIPLY y DIVIDE. Estas operaciones individuales se pueden combinar simbólicamente en una fórmula que utiliza la sentencia COMPUTE.

Operandos de sentencia aritmética

No es necesario que las descripciones de datos de los operandos de una sentencia aritmética sean las mismas. A lo largo del cálculo, el compilador realiza la conversión de datos necesaria y la alineación de coma decimal.

Tamaño de operandos

Si la opción de compilador ARITH (COMPAT) está en vigor, el tamaño máximo de cada operando es de 18 dígitos decimales. Si la opción de compilador ARITH (EXTEND) está en vigor, el tamaño máximo de cada operando es de 31 dígitos decimales.

El *compuesto de operandos* es un elemento de datos hipotético resultante de alinear los operandos en la coma decimal y, a continuación, superponerlos entre sí.

Si la opción de compilador ARITH (COMPAT) está en vigor, el compuesto de operandos puede tener un máximo de 30 dígitos. Si la opción de compilador ARITH (EXTEND) está en vigor, el compuesto de operandos puede tener un máximo de 31 dígitos.

La tabla siguiente muestra cómo se determina el compuesto de operandos para sentencias aritméticas:

Tabla 42. ***Cómo se determina el compuesto de operandos***

Sentencia	**Determinación del compuesto de operandos**
SUBTRACT add	Superposición de todos los operandos de una sentencia determinada excepto los que siguen a la palabra DANDO
MULTIPLY	Superposición de todos los elementos de datos de recepción
DIVIDIR	Superposición de todos los elementos de datos de recepción excepto el elemento de datos RESTANTE
SISTEMA	La restricción no se aplica

Por ejemplo, supongamos que cada elemento se define de la forma siguiente en DATA DIVISION:

```
A  PICTURE 9(7)V9(5).
B  PICTURE 9(11)V99.
C  PICTURE 9(12)V9(3).
```

Si se ejecuta la sentencia siguiente, el compuesto de operandos consta de 17 dígitos decimales:

```
ADD A B TO C
```

Tiene la siguiente descripción implícita:

```
COMPOSITE-OF-OPERANDS PICTURE 9(12)V9(5).
```

En las sentencias ADD y SUBTRACT, si el compuesto de operandos tiene 30 dígitos o menos con la opción de compilador ARITH (COMPAT), o 31 dígitos o menos con la opción de compilador ARITH (EXTEND), el compilador garantiza que se transporten suficientes lugares para que no se pierdan dígitos significativos durante la ejecución.

En todas las sentencias aritméticas, es importante definir datos con suficientes dígitos y posiciones decimales para garantizar la precisión necesaria en el resultado final. Para obtener más información, consulte *Apéndice A. Resultados intermedios y precisión aritmética* en *COBOL for Linux en x86 Guía de programación*.

Solapamiento de operandos

Cuando los operandos de una sentencia aritmética comparten parte de su almacenamiento (es decir, cuando los operandos se solapan), el resultado de la ejecución de dicha sentencia es imprevisible.

Varios resultados

Cuando una sentencia aritmética tiene varios resultados, la ejecución continúa conceptualmente como se indica a continuación:

1. La sentencia realiza todas las operaciones aritméticas para buscar el resultado que se va a colocar en los elementos receptores y almacena el resultado en una ubicación temporal.
2. Una secuencia de sentencias transfiere o combina el valor de este resultado temporal con cada campo de recepción individual. Las sentencias se consideran escritas en el mismo orden de izquierda a derecha en el que se listan los resultados múltiples.

Por ejemplo, ejecutando la sentencia siguiente:

```
ADD A, B, C, TO C, D(C), E.
```

es equivalente a ejecutar la siguiente serie de sentencias:

```
ADD A, B, C GIVING TEMP.
ADD TEMP TO C.
ADD TEMP TO D(C).
ADD TEMP TO E.
```

En el ejemplo anterior, TEMP es un campo de resultado temporal proporcionado por el compilador. Cuando se realiza la operación de adición para D (C), el subíndice C contiene el nuevo valor de C.

Sentencias de manipulación de datos

Las siguientes sentencias COBOL mueven e inspeccionan datos: ACCEPT, INITIALIZE, INSPECT, MOVE, READ, RELEASE, RETURN, REWRITE, SET, STRING, UNSTRING, WRITE, XML PARSE y XML GENERATE.

Solapamiento de operandos

Cuando los campos de envío y recepción de una sentencia de manipulación de datos comparten una parte de su almacenamiento (es decir, cuando los operandos se solapan), el resultado de la ejecución de dicha sentencia es imprevisible.

Sentencias de entrada-salida

Las sentencias de entrada-salida COBOL transfieren datos a y desde archivos almacenados en medios externos, y también controlan los datos de bajo volumen que se obtienen o se envían a un dispositivo de entrada/salida.

En COBOL, la unidad de datos de archivo que está disponible para el programa es un registro. Usted sólo necesita estar preocupado con tales registros. Se prevén automáticamente operaciones tales como el traslado de datos a almacenamientos intermedios, almacenamiento interno, comprobación de validez, corrección de errores (cuando sea posible), bloqueo y desbloqueo, y procedimientos de conmutación de volumen.

La descripción del archivo en ENVIRONMENT DIVISION y DATA DIVISION controla qué sentencias de entrada-salida están permitidas en PROCEDURE DIVISION. Las sentencias permitidas para archivos secuenciales se muestran en Tabla 53 en la página 360 , y las sentencias permitidas para archivos indexados y archivos relativos se muestran en Tabla 54 en la página 360. Las sentencias permisibles para archivos secuenciales de línea se muestran en Tabla 55 en la página 361.

Recursos de proceso comunes

Varios recursos de proceso comunes se aplican a más de una sentencia de entrada-salida.

Las instalaciones de procesamiento comunes que se proporcionan son:

- “Clave de estado de archivo” en la página 285

- “Condición de clave no válida” en la página 290
- “Frases INTO y FROM” en la página 290
- “Indicador de posición de archivo” en la página 291

Las discusiones en las secciones siguientes utilizan los términos *volumen* y *carrete*. El término *volumen* hace referencia a todos los dispositivos de entrada/salida que no son de registro de unidad. El término *carrete* sólo se aplica a los dispositivos de cinta. El tratamiento de los dispositivos de acceso directo en el modo de acceso secuencial es lógicamente equivalente al tratamiento de los dispositivos de cinta.

Clave de estado de archivo

Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, se coloca un valor en la clave de estado de archivo especificada (el elemento de datos de dos caracteres especificado en la cláusula FILE STATUS) durante la ejecución de cualquier solicitud en dicho archivo; el valor indica el estado de dicha solicitud.

El valor se coloca en la clave de estado de archivo antes de la ejecución de cualquier declaración EXCEPTION/ERROR, frase INVALID KEY o frase AT END asociada a la solicitud.

Hay dos nombres de datos de clave de estado de archivo. *data-name-1* describe uno en la cláusula FILE STATUS de la entrada de control de archivos. Se trata de un elemento de datos de dos caracteres con el primer carácter conocido como clave de estado de archivo 1 y el segundo carácter conocido como clave de estado de archivo 2. Las combinaciones de valores posibles y sus significados se muestran en Tabla 43 en la página 286.

La otra clave de estado de archivo se describe mediante *data-name-8* en la cláusula FILE STATUS de la entrada de control de archivo. *data-name-8* no se aplica a los archivos secuenciales de línea. Para obtener más información sobre *data-name-8*, consulte “cláusula FILE STATUS” en la página 131.

Tabla 43. ***Valores y significados de clave de estado de archivo***			
Dígito de orden alto	**Significado**	**Dígito de orden inferior**	**Significado**
0	Finalización satisfactoria	0	No hay más información
		2	Este valor de estado de archivo sólo se aplica a archivos indexados con claves alternativas que permiten duplicados. La sentencia de entrada-salida se ha ejecutado correctamente, pero se ha detectado una clave duplicada. Para una sentencia READ, el valor de clave para la clave de referencia actual era igual al valor de la misma clave en el siguiente registro dentro de la clave de referencia actual. Para una sentencia REWRITE o WRITE, el registro que se acaba de grabar ha creado un valor de clave duplicada para al menos una clave de registro alternativa para la que se permiten duplicados.
		4	Se ha ejecutado correctamente una sentencia READ, pero la longitud del registro que se estaba procesando no se ajustaba a los atributos de archivo fijo para ese archivo.
		5	Se ha ejecutado satisfactoriamente una sentencia OPEN, pero el archivo opcional referenciado no estaba disponible en el momento en que se ejecutó la sentencia OPEN. El archivo se había creado si la modalidad de apertura era I-O o EXTEND.
		7	Para una sentencia CLOSE con la frase NO REWIND, REEL/UNIT o FOR REMOCIÓN o para una sentencia OPEN con la frase NO REWIND, el archivo referenciado estaba en un medio que no era de reel/unidad.
1	Condición de finalización	0	Se ha intentado una sentencia READ secuencial y no existía ningún registro lógico siguiente en el archivo porque se había alcanzado el final del archivo. O se ha intentado el primer READ en un archivo de entrada opcional que no estaba disponible.
		4	Se ha intentado una sentencia READ secuencial para un archivo relativo y el número de dígitos significativos en el número de registro relativo era mayor que el tamaño del elemento de datos de clave relativa descrito para el archivo.

Tabla 43. ***Valores y significados de clave de estado de archivo*** *(continuación)*			
Dígito de orden alto	**Significado**	**Dígito de orden inferior**	**Significado**
2	Condición de clave no válida	1	Existe un error de secuencia para un archivo indexado accedido secuencialmente. El programa ha cambiado el valor de clave de registro principal entre la ejecución satisfactoria de una sentencia READ y la ejecución de la siguiente sentencia REWRITE para dicho archivo. O se han violado los requisitos ascendentes para valores de clave de registro sucesivos.
		2	Se ha intentado escribir un registro que crearía una clave duplicada en un archivo relativo. O se ha intentado escribir o reescribir un registro que crearía una clave de registro principal duplicada o una clave de registro alternativa duplicada sin la frase DUPLICATES en un archivo indexado.
		3	Se ha intentado acceder aleatoriamente a un registro que no existe en el archivo. O se ha intentado una sentencia START o READ aleatoria en un archivo de entrada opcional que no estaba disponible.
		4	Se ha intentado escribir más allá de los límites definidos externamente de un archivo relativo o indexado. O se ha intentado una sentencia WRITE secuencial para un archivo relativo y el número de dígitos significativos en el número de registro relativo era mayor que el tamaño del elemento de datos de clave relativa descrito para el archivo.

Tabla 43. ***Valores y significados de clave de estado de archivo*** *(continuación)*			
Dígito de orden alto	**Significado**	**Dígito de orden inferior**	**Significado**
3	Condición de error permanente	3	Una sentencia OPEN no ha sido satisfactoria porque se han especificado archivos concatenados, pero no se han cumplido uno o varios de los requisitos para la concatenación. Las posibles infracciones son: • El archivo ORGANIZATION no era SEQUENTIAL ni LINE SEQUENTIAL. • La MODALIDAD DE ACCESO no era SECUENCIAL. • La modalidad OPEN no era INPUT. • Se han especificado demasiados identificadores de archivo en la concatenación, o no se ha podido obtener memoria para registrar los identificadores de archivo que se han especificado. Una sentencia OPEN o READ no ha sido satisfactoria porque no estaba disponible un archivo no opcional en una concatenación. Por ejemplo, el archivo no existía o no tenía suficientes permisos de acceso. *Data-name-8*, si se especifica, contiene el valor de estado de archivo original, el identificador de archivo anómalo y cualquier información adicional que se hubiera proporcionado en *data-name-8* para una especificación de archivo no concatenado.
		4	Existe un error permanente debido a una violación de límite; se ha intentado escribir más allá de los límites definidos externamente de un archivo secuencial.
		5	Se ha intentado una sentencia OPEN con la frase INPUT, I-O o EXTEND en un archivo no opcional que no estaba disponible.
		7	Se ha intentado una sentencia OPEN en un archivo que no soportaría la modalidad de apertura especificada en la sentencia OPEN. Las posibles infracciones son: • Se ha especificado la frase EXTEND o OUTPUT pero el archivo no soportaría operaciones de grabación. • Se ha especificado la frase I-O pero el archivo no soportaría las operaciones de entrada y salida permitidas. • Se ha especificado la frase INPUT pero el archivo no soportaría operaciones de lectura.
		8	Se ha intentado una sentencia OPEN en un archivo cerrado anteriormente con bloqueo.
		9	La sentencia OPEN no ha sido satisfactoria porque se ha detectado un conflicto entre los atributos de archivo fijo y los atributos especificados para ese archivo en el programa. Estos atributos incluyen la organización del archivo (secuencial, relativo o indexado), la clave de registro principal, las claves de registro alternativas, el conjunto de códigos, el tamaño máximo de registro, el tipo de registro (fijo o variable) y el factor de bloqueo.

Tabla 43. ***Valores y significados de clave de estado de archivo*** *(continuación)*

Dígito de orden alto	**Significado**	**Dígito de orden inferior**	**Significado**
4	Condición de error lógico	1	Se ha intentado una sentencia OPEN para un archivo en modalidad abierta.
		2	Se ha intentado una sentencia CLOSE para un archivo que no está en modalidad abierta.
		3	Para un archivo de almacenamiento masivo en modalidad de acceso secuencial, la última sentencia de entrada-salida ejecutada para el archivo asociado antes de la ejecución de una sentencia REWRITE no era una sentencia READ ejecutada correctamente. Para los archivos relativos e indexados en la modalidad de acceso secuencial, la última sentencia de entrada-salida ejecutada para el archivo antes de la ejecución de una sentencia DELETE o REWRITE no era una sentencia READ ejecutada correctamente.
		4	Existe una violación de límite porque se ha intentado volver a escribir un registro en un archivo y el registro no tenía el mismo tamaño que el registro que se estaba sustituyendo. O se ha intentado grabar o reescribir un registro que era mayor o menor que el registro más pequeño permitido por la cláusula RECORD IS VARIABLE del nombre de archivo asociado.
		6	Se ha intentado una sentencia READ secuencial en un archivo abierto en la modalidad de entrada o I-O y no se ha establecido ningún registro siguiente válido porque: • La sentencia READ anterior no ha sido satisfactoria, pero no ha provocado una condición de finalización. • La sentencia READ anterior ha provocado una condición de finalización.
		7	Se ha intentado la ejecución de una sentencia READ en un archivo no abierto en la modalidad de entrada o I-O.
		8	Se ha intentado la ejecución de una sentencia WRITE en un archivo no abierto en la modalidad de I-O, salida o ampliación.
		9	Se ha intentado la ejecución de una sentencia DELETE o REWRITE en un archivo no abierto en modalidad I-O.
9	Condición definida por implementador	0	No hay más información.
		1	Anomalía de autorización
		2	Error lógico
		3	Recurso no disponible
		4	Error de apertura simultánea
		5	Información de archivo no válida o incompleta
		6	Sistema de archivos no disponible
		8	La apertura ha fallado debido a un archivo bloqueado
		9	El acceso al registro ha fallado debido a un registro bloqueado

Condición de clave no válida

La condición de clave no válida puede producirse durante la ejecución de una sentencia START, READ, WRITE, REWRITE o DELETE. Cuando se produce una condición de clave no válida, la sentencia de entrada-salida que ha causado la condición no es satisfactoria.

Cuando se reconoce la condición de clave no válida, las acciones se realizan en el orden siguiente:

1. Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, se coloca un valor en la clave de estado de archivo para indicar una condición de clave no válida, tal como se muestra en la Tabla 43 en la página 286.
2. Si se especifica la frase INVALID KEY en la sentencia que ha causado la condición, el control se transfiere a la sentencia imperativa INVALID KEY. No se ejecuta ningún procedimiento declarativo EXCEPTION/ERROR especificado para este archivo. A continuación, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en la sentencia imperativa.
3. Si no se especifica la frase INVALID KEY en la sentencia de entrada-salida para un archivo y existe un procedimiento EXCEPTION/ERROR aplicable, se ejecuta dicho procedimiento. La frase NOT INVALID KEY, si se especifica, se ignora.

Tanto la frase INVALID KEY como el procedimiento EXCEPTION/ERROR pueden omitirse.

Si la condición de clave no válida no existe después de la ejecución de la operación de entrada-salida, la frase INVALID KEY se ignora, si se especifica, y se realizan las acciones siguientes:

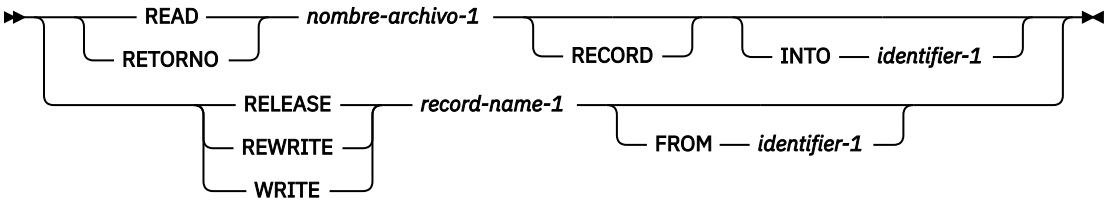
- Si existe una condición de excepción que no es una condición de clave no válida, el control se transfiere de acuerdo con las reglas de la sentencia USE después de la ejecución de cualquier procedimiento USE AFTER EXCEPTION.
- Si no existe ninguna condición de excepción, el control se transfiere al final de la sentencia de entrada-salida o a la sentencia imperativa especificada en la frase NOT INVALID KEY, si se especifica.

Frases INTO y FROM

Las frases INTO y FROM son válidas para las sentencias READ, RETURN, RELEASE, REWRITE y WRITE.

Debe especificar un identificador que sea el nombre de una entrada en WORKING-STORAGE SECTION o LINKAGE SECTION, o de una descripción de registro para otro archivo abierto anteriormente.

Formato: frases INTO y FROM de sentencias de entrada-salida



- *record-name-1* y *identifier-1* no deben hacer referencia a la misma área de almacenamiento.
- Si *record-name-1* o *identifier-1* hace referencia a un elemento de grupo nacional, el elemento se procesa como un elemento de datos elemental de categoría nacional.
- La frase INTO puede especificarse en una sentencia READ o RETURN.

  El resultado de la ejecución de una sentencia READ o RETURN con la frase INTO es equivalente a la aplicación de las reglas siguientes en el orden especificado:

  – La ejecución de la misma sentencia READ o RETURN sin la frase INTO.
  – El registro actual se mueve del área de registro al área especificada por *identifier-1* de acuerdo con las reglas para la sentencia MOVE sin la frase CORRESPONDIENTE. El tamaño del registro actual viene determinado por las reglas especificadas en la cláusula RECORD. Si la entrada de descripción de archivo contiene una cláusula RECORD IS VARIABLE, el movimiento implícito es un movimiento de

grupo. La sentencia MOVE implícita no se produce si la ejecución de la sentencia READ o RETURN no ha sido satisfactoria. Cualquier subscripción o modificación de referencia asociada con *identifier-1* se evalúa después de que el registro se haya leído o devuelto e inmediatamente antes de que se mueva al elemento de datos. El registro está disponible tanto en el área de registro como en el elemento de datos al que hace referencia *identifier-1*.

- La frase FROM puede especificarse en una sentencia RELEASE, REWRITE o WRITE.

  El resultado de la ejecución de una sentencia RELEASE, REWRITE o WRITE con la frase FROM es equivalente a la ejecución de las sentencias siguientes en el orden especificado:

  1. MOVE *identifier-1* TO *record-name-1*
  2. La misma sentencia RELEASE, REWRITE o WRITE sin la frase FROM

  Después de que se haya completado la ejecución de la sentencia RELEASE, REWRITE o WRITE, la información del área referenciada por *identifier-1* está disponible aunque la información del área referenciada por *record-name-1* no esté disponible, excepto según lo especificado por la cláusula SAME RECORD AREA.

Indicador de posición de archivo

El indicador de posición de archivo es una entidad conceptual utilizada en este documento para facilitar la especificación exacta del siguiente registro (o, alternativamente, el registro anterior) al que se puede acceder dentro de un archivo determinado durante determinadas secuencias de operaciones de entrada-salida.

El valor del indicador de posición de archivo sólo se ve afectado por las sentencias OPEN, CLOSE, READ y START. El concepto de un indicador de posición de archivo no tiene ningún significado para un archivo abierto en la modalidad de salida o ampliación.

Capítulo 19. Sentencias PROCEDURE DIVISION

Las sentencias, frases y párrafos de PROCEDURE DIVISION se ejecutan secuencialmente excepto cuando se utiliza una sentencia de ramificación de procedimiento como EXIT, GO TO, PERFORM, GOBACK o STOP.

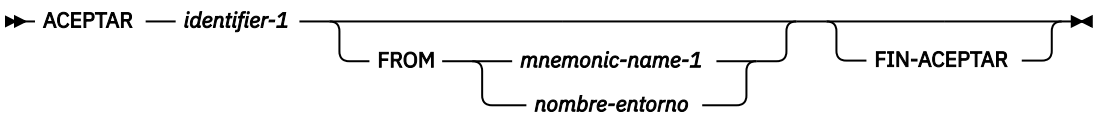
Sentencia ACCEPT

La sentencia ACCEPT transfiere datos o información relacionada con la fecha del sistema al área de datos a la que hace referencia el identificador especificado. No hay edición ni comprobación de errores de los datos entrantes.

Transferencia de datos

El formato 1 transfiere datos de un origen de entrada al elemento de datos al que hace referencia el *identifier-1* (el área de recepción). Cuando se omite la frase FROM, se presupone el dispositivo de entrada del sistema.

Formato 1: transferencia de datos



El formato 1 es útil para situaciones excepcionales en un programa cuando se requiere la intervención del operador (para proporcionar un determinado mensaje, código o indicador de excepción). Por supuesto, el operador debe recibir los mensajes adecuados con los que responder.

El archivo de entrada debe ser un archivo continuo de bytes (por ejemplo, un archivo que consta de datos de texto con registros delimitados por un terminador de registro). Puede crear un archivo continuo de bytes en el programa COBOL utilizando el archivo secuencial de línea I-O o con la sentencia DISPLAY. (La mayoría de los editores de texto también se pueden utilizar para crear un archivo continuo de bytes.)

El archivo de entrada no puede ser un archivo SdU, SFS o STL (incluidos los archivos secuenciales, relativos o indexados).

Si el origen de la sentencia ACCEPT es un archivo y el área receptora se llena sin utilizar el registro completo delimitado por el terminador de registro, el resto del registro de entrada se utiliza en la siguiente sentencia ACCEPT para el archivo. Los caracteres delimitadores de registro se eliminan de los datos de entrada antes de que los registros de entrada se muevan al área de recepción.

Si el origen de la sentencia ACCEPT es un teclado, los datos introducidos en el teclado seguidos por la tecla Intro se tratan como los datos de entrada. Si los datos de entrada son más cortos que el área de recepción, el área se rellena con espacios de la representación adecuada para el área de recepción.

identifier-1
El área receptora. Puede ser:

- Un elemento de grupo alfanumérico
- Un elemento de grupo nacional
- Un elemento de datos elemental de uso DISPLAY, DISPLAY-1o NATIONAL

Un elemento de grupo nacional se procesa como un elemento de datos elementales de categoría nacional.

Si la descripción de *identifier-1* contiene una cláusula TYPE , el nombre-tipo al que se hace referencia en dicha cláusula debe ser elemental.

Si identifier-1 es de uso NATIONAL y el origen de los datos de entrada es un teclado, la entrada se convierte de la página de códigos nativa (la página de códigos indicada por el entorno local runtime) a la representación de caracteres nacionales.

mnemonic-name-1
Especifica el dispositivo de entrada. *mnemonic-name-1* debe estar asociado en el párrafo SPECIAL-NAMES con un nombre de entorno. Consulte "Párrafo SPECIAL-NAMES" en la página 97.

nombre-entorno
Identifica el origen de los datos de entrada. Se puede especificar un nombre de entorno de los nombres proporcionados en Tabla 5 en la página 100 .

ACCEPT con un nombre de entorno utiliza el origen asociado con ese nombre de entorno por asignación de variable de entorno en el entorno operativo. Si la variable de entorno que corresponde al nombre de entorno COBOL no está establecida, ACCEPT de SYSIN o SYSIPT es del dispositivo de entrada lógico del sistema; ACCEPT de CONSOLE es del teclado del usuario. Para obtener más información sobre las variables de entorno, consulte *Ejemplo: Establecimiento y acceso a las variables de entorno* en *COBOL for Linux en x86 Guía de programación*.

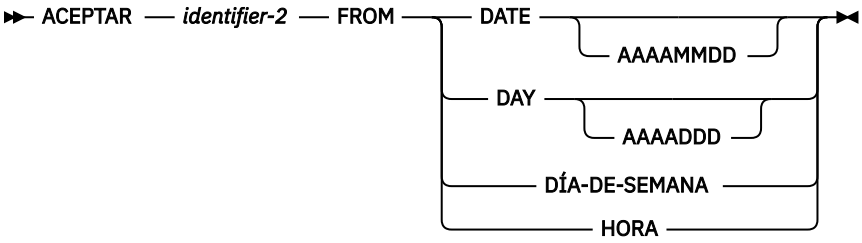
Si el dispositivo es el mismo que el utilizado para sentencias READ para un archivo LINE SEQUENTIAL, los resultados son imprevisibles.

Transferencia de información relacionada con la fecha del sistema

La información del sistema contenida en los elementos de datos conceptuales especificados DATE, DATE YYYYMMDD, DAY, DAY YYYYDDD, DAY-OF-WEEK o TIME, se puede transferir al elemento de datos al que hace referencia *identifier-2*. La transferencia debe seguir las reglas para la sentencia MOVE sin la frase CORRESPONDIENTE.

Para obtener más información, consulte "sentencia MOVE" en la página 347.

Formato 2: transferencia de información del sistema



identifier-2
El área receptora. Puede ser:

- Un elemento de grupo alfanumérico
- Un elemento de grupo nacional
- Un elemento de datos elemental de una de las categorías siguientes:
  - alfanumérico
  - alfanumérico editado
  - numérica-editada (con uso DISPLAY o NATIONAL)
  - nacional
  - editado a nivel nacional

- numérico
- coma flotante interna
- coma flotante externa (con uso DISPLAY o NATIONAL)

Un elemento de grupo nacional se procesa como un elemento de datos elementales de categoría nacional.

El formato 2 accede a la fecha actual en dos formatos: el día de la semana o la hora del día que lleva el sistema (que puede ser útil para identificar cuándo se ejecutó una ejecución concreta de un programa objeto). También puede utilizar el formato 2 para proporcionar la fecha en cabeceras y pies.

También se puede acceder a la fecha y hora actual con la función intrínseca CURRENT-DATE, que también admite valores de año de cuatro dígitos y proporciona información adicional (consulte "ACTUAL-FECHA" en la página 468).

DATE, DATE YYYYMMDD, DAY, DAY YYYYDDD, DAY-OF-WEEK y TIME

Los elementos de datos conceptuales DATE, DATE YYYYMMDD, DAY, DAY YYYYDDD, DAY-OF-WEEK y TIME tienen implícitamente USAGE DISPLAY. Puesto que se trata de elementos de datos conceptuales, no se pueden describir en el programa COBOL.

El contenido de los elementos de datos conceptuales se mueve al área de recepción utilizando las reglas de la sentencia MOVE. Si el área de recepción es de uso NATIONAL, los datos se convierten a representación de caracteres nacionales.

fecha
Tiene el PICTURE 9 (6) implícito. Si la opción de compilador DATEPROC está en vigor, el valor devuelto tiene implícito DATE FORMAT YYXXXX y *identifier-2* debe definirse con este formato de fecha.

La secuencia de elementos de datos (de izquierda a derecha) es:

```
Two digits for the year
Two digits for the month
Two digits for the day
```

Así, el 27 de abril de 2003 se expresa como 030427.

FECHA AAAAMMDD
Tiene el PICTURE 9 (8) implícito. Si la opción de compilador DATEPROC está en vigor, el valor devuelto tiene implícito DATE FORMAT YYYYXXXX y *identifier-2* debe definirse con este formato de fecha.

La secuencia de elementos de datos (de izquierda a derecha) es:

```
Four digits for the year
Two digits for the month
Two digits for the day
```

Así, el 27 de abril de 2003 se expresa como 20030427.

DÍA
Tiene el implícito PICTURE 9 (5). Si la opción de compilador DATEPROC está en vigor, el valor devuelto tiene implícito DATE FORMAT YYXXX y *identifier-2* debe definirse con este formato de fecha.

La secuencia de elementos de datos (de izquierda a derecha) es:

```
Two digits for the year
Three digits for the day
```

Así, el 27 de abril de 2003 se expresa como 03117.

DAY AAAADDD
Tiene la imagen implícita 9 (7). Si la opción de compilador DATEPROC está en vigor, el valor devuelto tiene implícito DATE FORMAT YYYYXXX y *identifier-2* debe definirse con este formato de fecha.

La secuencia de elementos de datos (de izquierda a derecha) es:

```
Four digits for the year
Three digits for the day
```

Así, el 27 de abril de 2003 se expresa como `2003117`.

DÍA DE LA SEMANA
Tiene el PICTURE 9 (1) implícito.

El elemento de datos único representa el día de la semana de acuerdo con los valores siguientes:

```
1 represents Monday           5 represents Friday
2 represents Tuesday          6 represents Saturday
3 represents Wednesday        7 represents Sunday
4 represents Thursday
```

Así el miércoles se expresa como 3.

hora
Tiene el PICTURE 9 (8) implícito.

La secuencia de elementos de datos (de izquierda a derecha) es:

```
Two digits for hour of day
Two digits for minute of hour
Two digits for second of minute
Two digits for hundredths of second
```

Por lo tanto, 2:41 PM se expresa como `14410000`.

Ejemplo de la sentencia ACCEPT

Este tema lista un ejemplo para la sentencia ACCEPT.

```
       IDENTIFICATION DIVISION.
        PROGRAM-ID. ACCPTST.
        ENVIRONMENT DIVISION.
        DATA DIVISION.
        WORKING-STORAGE SECTION.
        01 AGE                      PIC 9(3).
        01 GENDER                   PIC X(1).
        PROCEDURE DIVISION.
            ACCEPT AGE.
            ACCEPT GENDER.
            EVALUATE TRUE ALSO TRUE
               WHEN AGE > 60 ALSO GENDER = 'M'
                    DISPLAY 'THE MAN IS RETIRED   '
               WHEN AGE > 60 ALSO GENDER = 'F'
                    DISPLAY 'THE WOMAN IS RETIRED  '
               WHEN AGE <= 60 ALSO GENDER = 'M'
                    DISPLAY 'THE MAN IS NOT RETIRED   '
               WHEN AGE <= 60 ALSO GENDER = 'F'
                    DISPLAY 'THE WOMAN IS NOT RETIRED  '
               WHEN OTHER
                    DISPLAY 'INVALID INPUT      '
                    DISPLAY 'AGE =' AGE ' and GENDER =' GENDER
            END-EVALUATE.
            STOP RUN.
```

Coloque los datos en `file1.dat`:

```
 64
M
```

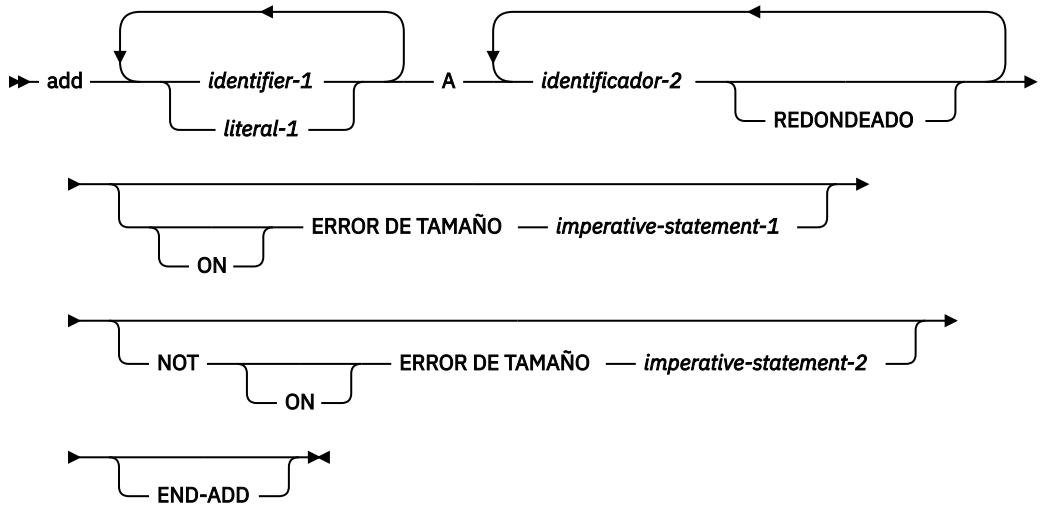
y exportar utilizando SYSIN:

```
export SYSIN=file1.dat
```

sentencia ADD

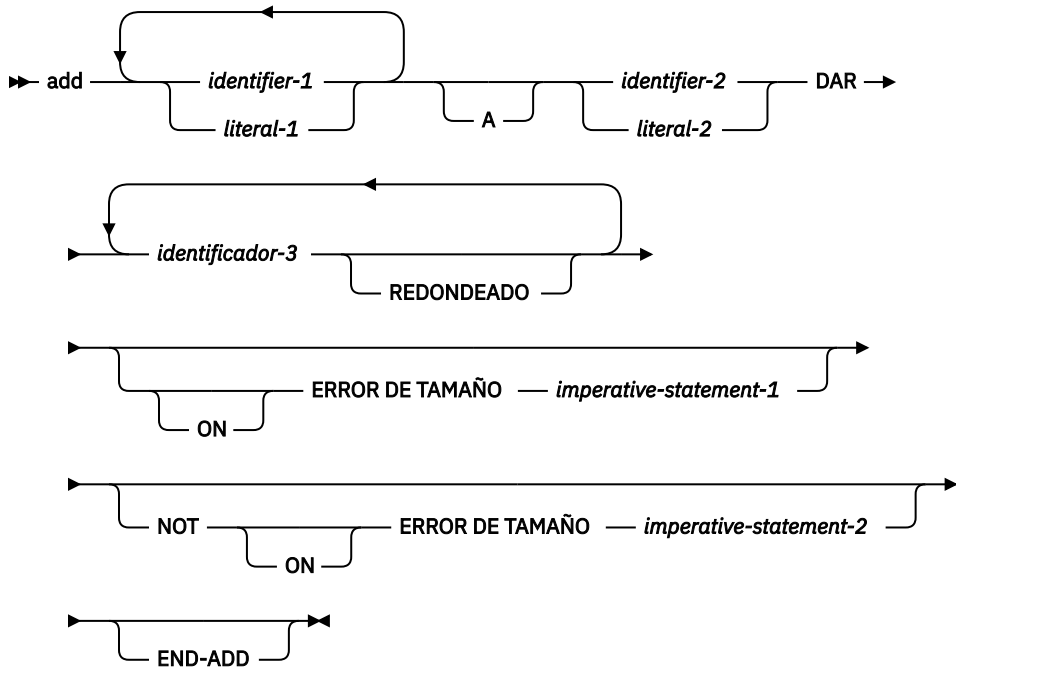
La sentencia ADD suma dos o más operandos numéricos y almacena el resultado.

Formato 1: sentencia ADD



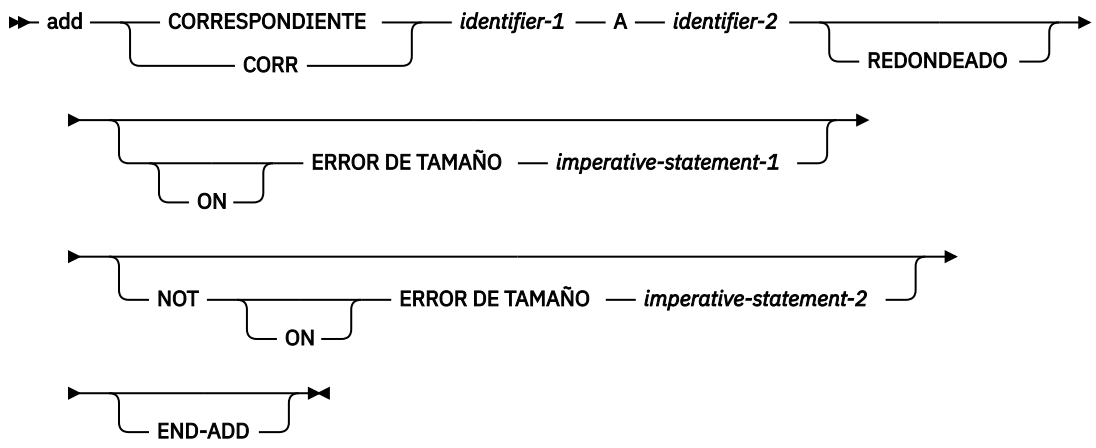
Se añaden todos los identificadores o literales que preceden a la palabra clave TO, y esta suma se añade y se almacena en *identifier-2*. Este proceso se repite para cada aparición sucesiva de *identifier-2* en el orden de izquierda a derecha en el que se especifica *identifier-2* .

Formato 2: sentencia ADD con frase CEDER



Los valores de los operandos que preceden a la palabra CEDER se añaden juntos, y la suma se almacena como el nuevo valor de cada elemento de datos al que hace referencia *identifier-3*.

Formato 3: sentencia ADD con frase CORRESPONDIENTE



Los elementos de datos elementales dentro de *identifier-1* se añaden y se almacenan en los elementos elementales correspondientes dentro de *identifier-2*.

Para todos los formatos:

identifier-1, identifier-2
En el formato 1, debe nombrar un elemento numérico elemental.

En el formato 2, debe nombrar un elemento numérico elemental excepto cuando se sigue la palabra CEDER. Cada identificador que sigue a la palabra CEDER debe nombrar un elemento numérico elemental o numérico editado.

En el formato 3, debe nombrar un elemento de grupo alfanumérico o un elemento de grupo nacional.

Las restricciones siguientes se aplican a los campos de fecha:

- En el formato 1, *identifier-2* puede especificar uno o más campos de fecha. *identifier-1* no debe especificar un campo de fecha.
- En el formato 2, el *identifier-1* o *identifier-2* (pero no ambos) puede especificar como máximo un campo de fecha. Si *identifier-1* o *identifier-2* especifica un campo de fecha, entonces cada instancia de *identifier-3* debe especificar un campo de fecha que sea compatible con el campo de fecha especificado por *identifier-1* o *identifier-2*. Es decir, deben tener el mismo formato de fecha, a excepción de la parte de año, que puede ser con ventana o expandida.

  Si ni *identifier-1* ni *identifier-2* especifica un campo de fecha, *identifier-3* puede especificar uno o más campos de fecha sin ninguna restricción en los formatos de fecha.
- En el formato 3, sólo los elementos elementales correspondientes dentro de *identifier-2* pueden ser campos de fecha. No hay ninguna restricción en el formato de estos campos de fecha.
- Un campo de fecha de último año está permitido en una sentencia ADD sólo como *identifier-1* y cuando el resultado de la adición es una no fecha.

Hay dos pasos para determinar el resultado de una sentencia ADD que implica uno o más campos de fecha:

1. Además: determine el resultado de la operación de adición, tal como se describe en "Adición que implica campos de fecha" en la página 248.
2. Almacenamiento: determina cómo se almacena el resultado en el campo de recepción. (En los formatos 1 y 3, el campo de recepción es *identifier-2*; en el Formato 3, el campo de recepción es el DANDO *identifier-3*.) Para obtener detalles, consulte "Almacenamiento de resultados aritméticos que implican campos de fecha" en la página 249.

literal
Debe ser un literal numérico.

Los elementos de datos de coma flotante y literales se pueden utilizar en cualquier lugar donde se pueda especificar un literal o un elemento de datos numérico.

Cuando la opción de compilador ARITH (COMPAT) está en vigor, el compuesto de operandos puede contener un máximo de 30 dígitos. Cuando la opción de compilador ARITH (EXTEND) está en vigor, el compuesto de operandos puede contener un máximo de 31 dígitos. Para obtener más información, consulte "Operandos de sentencia aritmética" en la página 282 y los detalles sobre los resultados intermedios aritméticos en *Apéndice A. Resultados intermedios y precisión aritmética* en *COBOL for Linux en x86 Guía de programación*.

Frase REDONDEADA

Para los formatos 1, 2 y 3, consulte "Frase REDONDEADA" en la página 281.

Frases de ERROR DE TAMAÑO

Para los formatos 1, 2 y 3, consulte "Frases de ERROR DE TAMAÑO" en la página 281.

Frase CORRESPONDIENTE (formato 3)

Consulte "Frase CORRESPONDIENTE" en la página 279.

Frase END-ADD

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia ADD. END-ADD permite que una sentencia ADD condicional se anide en otra sentencia condicional. END-ADD también se puede utilizar con una sentencia ADD imperativa.

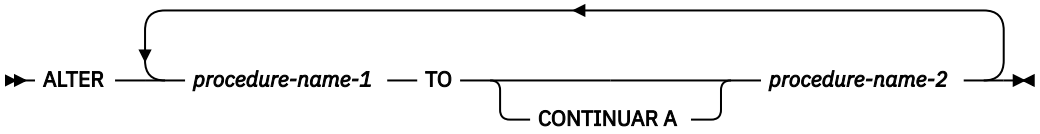
Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

Sentencia ALTER

La sentencia ALTER cambia el punto de transferencia especificado en una sentencia GO TO.

La sentencia ALTER fomenta el uso de prácticas de programación no estructuradas; la sentencia EVALUATE proporciona la misma función que la sentencia ALTER, pero ayuda a garantizar que un programa esté bien estructurado.

Formato



La sentencia ALTER modifica la sentencia GO TO en el párrafo nombrado por *procedure-name-1*. Las ejecuciones posteriores de la sentencia GO TO modificada transfieren el control a *procedure-name-2*.

procedure-name-1
Debe nombrar un párrafo PROCEDURE DIVISION que contenga sólo una frase: una sentencia GO TO sin la frase DEPENDING ON.

procedure-name-2
Debe nombrar una sección o párrafo de PROCEDURE DIVISION.

Antes de ejecutar la sentencia ALTER, cuando el control alcanza el párrafo especificado en *procedure-name-1*, la sentencia GO TO transfiere el control al párrafo especificado en la sentencia GO TO. Sin embargo, después de la ejecución de la sentencia ALTER, la próxima vez que el control alcance el párrafo

especificado en *procedure-name-1*, la sentencia GO TO transfiere el control al párrafo especificado en *procedure-name-2*.

La sentencia ALTER actúa como un conmutador de programa, permitiendo, por ejemplo, una secuencia de ejecución durante la inicialización y otra secuencia durante la mayor parte del proceso de archivos.

Las sentencias GO TO alteradas en programas con el atributo INITIAL se devuelven a sus estados iniciales cada vez que se entra en el programa.

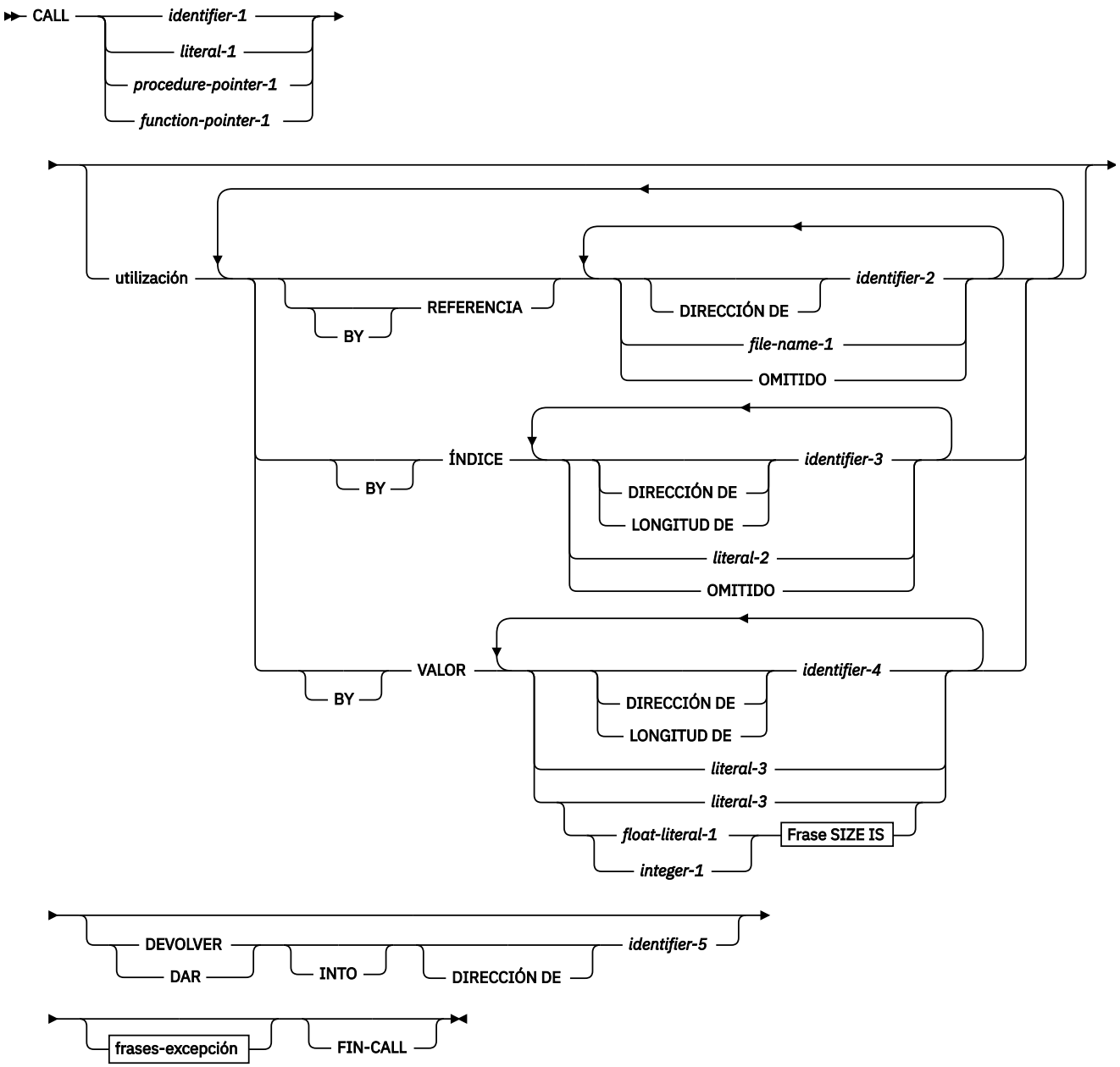
No utilice la sentencia ALTER en programas que tengan el atributo RECURSIVE.

sentencia CALL

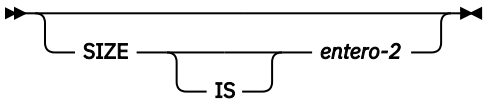
La sentencia CALL transfiere el control de un programa objeto a otro dentro de la unidad de ejecución.

El programa que contiene la sentencia CALL es el programa que llama; el programa identificado en la sentencia CALL es el subprograma llamado. Los programas llamados pueden contener sentencias CALL; sin embargo, sólo los programas definidos con la cláusula RECURSIVE pueden ejecutar una sentencia CALL que se llame directa o indirectamente a sí mismo.

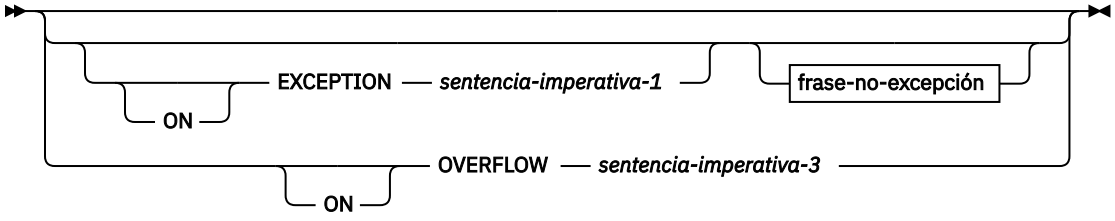
Formato



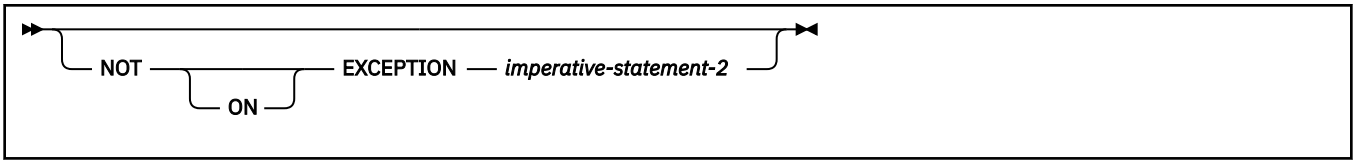
Frase SIZE IS



frases de excepción



frase-no-excepción



identifier-1, literal-1
literal-1 debe ser un literal alfanumérico. *identifier-1* debe ser un elemento de datos alfanumérico, alfabético o numérico descrito con USAGE DISPLAY de forma que su valor pueda ser un nombre de programa.

Las reglas de formación para los nombres de programa dependen de la opción de compilador PGMNAME. Para obtener detalles, consulte la descripción de los nombres de programa en "Párrafo PROGRAM-ID" en la página 88 y también la descripción de *PGMNAME* en *COBOL for Linux en x86 Guía de programación*.

identifier-1 no puede ser un campo de fecha con ventana.

Nota de uso: No especifique el nombre de una clase en la sentencia CALL.

procedure-pointer-1
Debe definirse con USAGE IS PROCEDURE-POINTER y debe establecerse en un punto de entrada de programa válido; de lo contrario, los resultados de la sentencia CALL no están definidos.

Después de que un programa haya sido cancelado por COBOL, liberado por PL/I o C, o suprimido por assembler, los punteros de procedimiento que se hayan establecido en el punto de entrada de ese programa ya no son válidos.

function-pointer-1
Debe definirse con USAGE IS FUNCTION-POINTER y debe establecerse en un punto de entrada de programa o función válido; de lo contrario, los resultados de la sentencia CALL no están definidos.

Después de que un programa haya sido cancelado por COBOL, liberado por PL/I o C, o suprimido por el ensamblador, los punteros de función que se hayan establecido en esa función o punto de entrada del programa ya no son válidos.

Cuando el subprograma llamado debe especificarse al principio de PROCEDURE DIVISION, *literal-1* o el contenido de *identifier-1* debe especificar el nombre de programa del subprograma llamado.

Cuando el subprograma llamado se especifica mediante una sentencia ENTRY, *literal-1* o el contenido de *identifier-1* debe ser el mismo que el nombre especificado en la sentencia ENTRY del subprograma llamado.

frase USING

La frase USING especifica los argumentos que se pasan al programa de destino.

Incluya la frase USING en la sentencia CALL sólo si hay una frase USING en la cabecera PROCEDURE DIVISION o la sentencia ENTRY a través de la cual se ejecuta el programa llamado. El número de operandos en cada frase USING debe ser idéntico.

Para obtener más información sobre la frase USING, consulte "Cabecera PROCEDURE DIVISION" en la página 241.

La secuencia de los operandos en la frase USING de la sentencia CALL y en la frase USING correspondiente en la cabecera PROCEDURE DIVISION del subprograma llamado o sentencia ENTRY determina la correspondencia entre los operandos utilizados por los programas de llamada y de llamada. Esta correspondencia es posicional.

Los valores de los parámetros a los que se hace referencia en la frase USING de la sentencia CALL se ponen a disposición del subprograma llamado en el momento en que se ejecuta la sentencia CALL. La descripción de los elementos de datos en el programa llamado debe describir el mismo número de posiciones de caracteres que la descripción de los elementos de datos correspondientes en el programa de llamada.

Las frases BY CONTENT, BY REFERENCE y BY VALUE se aplican a los parámetros que las siguen hasta que se encuentra otra frase BY CONTENT, BY REFERENCE o BY VALUE. Se presupone BY REFERENCE si no especifica una frase BY CONTENT, BY REFERENCE o BY VALUE antes del primer parámetro.

Frase POR REFERENCIA

Si la frase BY REFERENCE se especifica o está implícita para un parámetro, el elemento de datos correspondiente del programa de llamada ocupa la misma área de almacenamiento que el elemento de datos del programa llamado.

identifier-2
Puede ser cualquier elemento de datos de cualquier nivel en DATA DIVISION. *identifier-2* no puede ser un identificador de función.

Si está definido en LINKAGE SECTION o FILE SECTION, ya debe haber proporcionado direccionabilidad para el *identifier-2* antes de invocar la sentencia CALL. Puede hacerlo codificando uno de los elementos siguientes: `SET ADDRESS OF identifier-2 TO pointer` o `PROCEDURE/ENTRY USING`.

file-name-1
Un nombre de archivo para un archivo organizado secuencialmente . Consulte *Pasar datos* en la publicación *COBOL for Linux en x86 Guía de programación* para obtener detalles sobre cómo utilizar el nombre de archivo con la sentencia CALL.

ADDRESS OF *identifier-2*
identifier-2 debe ser un elemento level-01 o level-77 definido en LINKAGE SECTION.

OMITIDO
Indica que no se ha pasado ningún argumento.

Frase BY CONTENT

Si se especifica o se implica la frase BY CONTENT para un parámetro, el programa llamado no puede cambiar el valor de este parámetro como se hace referencia en la frase USING de la sentencia CALL, aunque el programa llamado puede cambiar el valor del elemento de datos al que hace referencia el nombre de datos correspondiente en la cabecera PROCEDURE DIVISION del programa llamado. Los cambios en el parámetro en el programa llamado no afectan al argumento correspondiente en el programa de llamada.

identifier-3
Puede ser cualquier elemento de datos de cualquier nivel en DATA DIVISION. *identifier-3* no puede ser un identificador de función.

Si está definido en LINKAGE SECTION o FILE SECTION, ya debe haber proporcionado direccionabilidad para *identifier-3* antes de invocar la sentencia CALL. Puede hacerlo codificando una de las frases siguientes:

- `SET ADDRESS OF identifier-3 TO pointer`
- `PROCEDURE DIVISION USING`
- `ENTRY USING`

literal-2
Puede ser:

- Un literal alfanumérico
- Un literal booleano
- Una constante figurativa (excepto ALL *literal* o NULL/NULLS)
- Un literal DBCS
- Un literal nacional

LENGTH OF registro especial
Para obtener información sobre el registro especial LENGTH OF, consulte "LENGTH OF" en la página 21.

ADDRESS OF *identifier-3*
identifier-3 debe ser un elemento de datos de cualquier nivel excepto 66 u 88 definido en LINKAGE SECTION, WORKING-STORAGE SECTION o LOCAL-STORAGE SECTION.

OMITIDO
Indica que no se ha pasado ningún argumento.

Para literales alfanuméricos, el subprograma llamado debe describir el parámetro como `PIC X(n) USAGE DISPLAY`, donde *n* es el número de caracteres del literal.

Para literales DBCS, el subprograma llamado debe describir el parámetro como `PIC G(n) USAGE DISPLAY-1`, o `PIC N(n)` con USAGE implícito o explícito DISPLAY-1, donde *n* es la longitud del literal.

Para literales nacionales, el subprograma llamado debe describir el parámetro como `PIC N(n)` con USAGE NATIONAL implícito o explícito, donde *n* es la longitud del literal.

Frase BY VALUE

La frase BY VALUE se aplica a todos los argumentos que siguen hasta que se alteran temporalmente por otra frase BY REFERENCE o BY CONTENT.

Si se especifica o está implícita la frase BY VALUE para un argumento, se pasa el valor del argumento, no una referencia al elemento de datos de envío. El programa llamado puede modificar el parámetro formal que corresponde al argumento BY VALUE, pero cualquier cambio de este tipo no afecta al argumento porque el programa llamado tiene acceso a una copia temporal del elemento de datos de envío.

Aunque los argumentos BY VALUE están pensados principalmente para la comunicación con programas no COBOL (como C), también se pueden utilizar para invocaciones de COBOL a COBOL. En este caso, BY VALUE debe especificarse o implicarse tanto para el argumento de la frase CALL USING como para el parámetro formal correspondiente de la frase PROCEDURE DIVISION USING.

identifier-4
Debe ser un elemento de datos elemental en DATA DIVISION. Debe ser uno de los elementos siguientes:

- Binario (USAGE BINARY, COMP, COMP-4o COMP-5)
- Coma flotante (USAGE COMP-1 o COMP-2)
- Puntero de función (USAGE FUNCTION-POINTER)
- Puntero (USAGE POINTER)
- Puntero de procedimiento (USAGE PROCEDURE-POINTER)
- Un carácter alfanumérico de un solo byte (como PIC X o PIC A)
- Un carácter nacional (PIC N), descrito como un dato elemental de la categoría nacional.

Los elementos siguientes también se pueden pasar POR VALOR:

- Elemento modificado por referencia de USAGE DISPLAY y longitud 1
- Elemento modificado por referencia de USAGE NATIONAL y longitud 1
- Registros especiales SHIFT-IN y SHIFT-OUT
- LINAGE-Registro especial COUNTER cuando es USAGE BINARY

ADDRESS OF *identifier-4*
identifier-4 debe ser un elemento de datos de cualquier nivel excepto 66 u 88 definido en LINKAGE SECTION, WORKING-STORAGE SECTION o LOCAL-STORAGE SECTION.

LENGTH OF registro especial
Un registro especial LENGTH OF pasado BY VALUE se trata como un binario PIC 9 (9) cuando se especifica la opción de compilador **ADDR(32)** , o como un binario PIC 9 (18) cuando se especifica

la opción de compilador **ADDR(64)** . Para obtener información sobre el registro especial LENGTH OF, consulte "LENGTH OF" en la página 21.

literal-3
Debe ser de uno de los tipos siguientes:

- Un literal booleano
- Un literal numérico
- Una constante figurativa CERO
- Un literal alfanumérico de un carácter
- Un literal nacional de un carácter
- Un carácter simbólico
- Una constante figurativa de un solo byte
  - SPACE
  - QUOTE
  - ALTO VALOR
  - LOW-VALUE

CERO se trata como un valor numérico; se pasa un cero binario de palabra completa.

Si *literal-3* es un literal numérico de punto fijo, debe tener una precisión de nueve o menos dígitos. En este caso, se pasa una representación binaria de palabra completa del valor literal.

Si *literal-3* es un literal numérico de coma flotante, se pasa una representación de coma flotante interna de 8 bytes (COMP-2) del valor.

literal-3 no debe ser un literal DBCS.

float-literal-1
Un literal de coma flotante se pasa como un flotante interno de 8 bytes (COMP-2), a menos que se especifique la frase SIZE. Para elementos de coma flotante, la frase de tamaño puede ser 4 u 8.

integer-1
Puede ser un entero con signo o sin signo.

integer-1 se pasa como un valor binario. Si no se especifica *integer-2* , *integer-1* se pasará como un valor binario de 4 bytes. *integer-2* especifica el tamaño de *integer-1*. Puede ser uno de 1, 2, 4 u 8.

Frase DEVOLVER

DEVOLVER y DAR son equivalentes.

ADDRESS OF *identifier-5*
identifier-5 debe ser un elemento level-01 o level-77 definido en LINKAGE SECTION.

identifier-5
El elemento de datos DEVOLUCIÓN, que puede ser cualquier elemento de datos definido en DATA DIVISION. El valor de retorno del programa llamado se almacena implícitamente en *identifier-5*.

Puede especificar la frase DEVOLUCIÓN para llamadas a funciones escritas en COBOL, C o en otros lenguajes de programación que utilizan convenios de enlace C. Si especifica la frase DEVOLUCIÓN en una CALL a un subprograma COBOL:

- El subprograma llamado debe especificar la frase DEVOLVER en su cabecera PROCEDURE DIVISION.
- *identifier-5* y el identificador PROCEDURE DIVISION DEVOLUCIÓN correspondiente en el programa de destino deben tener las mismas cláusulas PICTURE, USAGE, SIGN, SYNCHRONIZE, JUSTIZE y BLANK WHEN ZERO (excepto que los símbolos de moneda de la cláusula PICTURE pueden diferir, y los puntos y comas pueden intercambiarse debido a la cláusula DECIMAL POINT IS COMA).

Cuando se devuelve el destino, su valor de retorno se asigna a *identifier-5* utilizando las reglas para la sentencia SET si *identifier-6* es de uso INDEX, POINTER, FUNCTION-POINTER o PROCEDURE-POINTER. Cuando *identifier-5* es de cualquier otro uso, se utilizan las reglas para la sentencia MOVE.

CALL ... El elemento de datos DEVOLVER es un parámetro de sólo salida. Al entrar en el programa llamado, el estado inicial del elemento de datos PROCEDURE DIVISION REGRESANDO tiene un valor no definido e imprevisible. Debe inicializar el elemento de datos PROCEDURE DIVISION DEVUELVE en el programa llamado antes de hacer referencia a su valor. El valor que se devuelve al programa de llamada es el valor final del elemento de datos PROCEDURE DIVISION RETURNS cuando se devuelve el programa llamado.

Si se produce una excepción o un desbordamiento, *identifier-5* no se cambia. *identifier-5* no debe ser modificado por referencia.

El registro especial RETURN-CODE no se establece mediante la ejecución de sentencias CALL que incluyen la frase RETURN.

Los elementos a los que se hace referencia en la frase DEVOLVER de la sentencia CALL no pueden contener la frase TYPE.

Frase ON EXCEPTION

Se produce una condición de excepción cuando el subprograma llamado no puede estar disponible. En ese momento, se producirá una de las dos acciones siguientes:

1. Si se especifica la frase ON EXCEPTION, el control se transfiere a *imperative-statement-1*. A continuación, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en *imperative-statement-1*. Si se ejecuta una ramificación de procedimiento o una sentencia condicional que provoca una transferencia explícita de control, el control se transfiere de acuerdo con las reglas de dicha sentencia. De lo contrario, al finalizar la ejecución de *imperative-statement-1*, el control se transfiere al final de la sentencia CALL y la frase NOT ON EXCEPTION, si se especifica, se ignora.
2. Si no se especifica la frase ON EXCEPTION en la sentencia CALL, se ignora la frase NOT ON EXCEPTION, si se especifica.

Frase NO EN EXCEPCIÓN

Si no se produce una condición de excepción (es decir, el subprograma llamado puede estar disponible), el control se transfiere al programa llamado. Después de que se devuelva el control del programa llamado, el control se transfiere a:

- *imperative-statement-2*, si se especifica la frase NOT ON EXCEPTION.
- El final de la sentencia CALL en cualquier otro caso. (Si se especifica la frase ON EXCEPTION, se ignora.)

Si el control se transfiere a *imperative-statement-2*, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en *imperative-statement-2*. Si se ejecuta una ramificación de procedimiento o una sentencia condicional que provoca una transferencia explícita de control, el control se transfiere de acuerdo con las reglas de dicha sentencia. De lo contrario, al finalizar la ejecución de *imperative-statement-2*, el control se transfiere al final de la sentencia CALL.

Frase ON OVERFLOW

La frase ON OVERFLOW tiene el mismo efecto que la frase ON EXCEPTION.

frase END-CALL

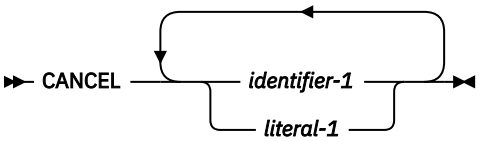
Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia CALL. END-CALL permite que una sentencia CALL condicional se anide en otra sentencia condicional. END-CALL también se puede utilizar con una sentencia CALL imperativa.

Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

sentencia CANCEL

La sentencia CANCEL garantiza que el subprograma al que se hace referencia se entre en estado inicial la próxima vez que se llame.

Formato



identifier-1, literal-1

literal-1 debe ser un literal alfanumérico. *identifier-1* debe ser un elemento de datos alfanumérico, alfabético o decimal con zona, de forma que su valor pueda ser un nombre de programa. Las reglas de formación para los nombres de programa dependen de la opción de compilador PGMNAME. Para obtener detalles, consulte la descripción de los nombres de programa en "Párrafo PROGRAM-ID" en la página 88 y la descripción de *PGMNAME* en *COBOL for Linux en x86 Guía de programación*.

identifier-1 no puede ser un campo de fecha con ventana.

literal-1 o el contenido del *identifier-1* debe ser el mismo que un literal o el contenido de un identificador especificado en una sentencia CALL asociada.

No especifique el nombre de una clase o un método en la sentencia CANCEL.

Después de que se haya ejecutado una sentencia CANCEL para un subprograma llamado, ese subprograma ya no tiene una conexión lógica con el programa. El contenido de los elementos de datos en los registros de datos externos descritos por el subprograma no se cambia cuando se cancela dicho subprograma. Si una sentencia CALL es ejecutada posteriormente por cualquier programa de la unidad de ejecución que nombre el mismo subprograma, ese subprograma se entra en su estado inicial.

Cuando se ejecuta una sentencia CANCEL, también se cancelan todos los programas contenidos en el programa al que se hace referencia en la sentencia CANCEL. El resultado es el mismo que si se ejecutara una CANCEL válida para cada programa contenido en el orden inverso en el que aparecen los programas en el programa compilado por separado.

Una sentencia CANCEL cierra todos los archivos abiertos que están asociados con un conector de archivo interno en el programa especificado en una sentencia CANCEL explícita. Los procedimientos USE asociados con esos archivos no se ejecutan.

Puede cancelar un subprograma llamado de cualquiera de las maneras siguientes:

- Haciendo referencia a él como operando de una sentencia CANCEL
- Terminando la unidad de ejecución de la que es miembro el subprograma
- Ejecutando una sentencia EXIT PROGRAM o una sentencia GOBACK en el subprograma llamado si ese subprograma posee el atributo inicial

No se realiza ninguna acción cuando se ejecuta una sentencia CANCEL si el programa especificado:

- Otro programa IBM COBOL no ha llamado dinámicamente a esta unidad de ejecución.
- Se ha llamado y posteriormente se ha cancelado

En un entorno multihebra, un programa no puede ejecutar una sentencia CANCEL nombrando un programa que esté activo en cualquier hebra. El programa especificado debe estar completamente inactivo.

Los subprogramas llamados pueden contener sentencias CANCEL. Sin embargo, un subprograma llamado no debe ejecutar una sentencia CANCEL que cancele directa o indirectamente el propio programa de llamada o que cancele cualquier programa superior a sí mismo en la jerarquía de llamadas. En tal caso, la unidad de ejecución se termina.

Un programa denominado en una sentencia CANCEL debe ser un programa al que se ha llamado y que ha ejecutado una sentencia EXIT PROGRAM o una sentencia GOBACK.

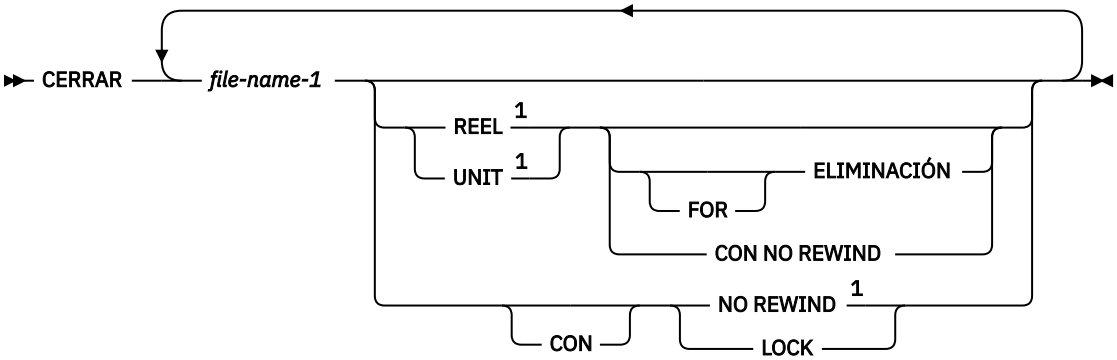
Un programa puede cancelar un programa al que no ha llamado, siempre que, en la jerarquía de llamadas, el programa que ejecuta la sentencia CANCEL sea superior o igual al programa que está cancelando. Por ejemplo:

```
A calls B and B calls C    (When A receives control, it can cancel C.)
A calls B and A calls C    (When C receives control, it can cancel B.)
```

sentencia CLOSE

La sentencia CLOSE termina el proceso de volúmenes y archivos.

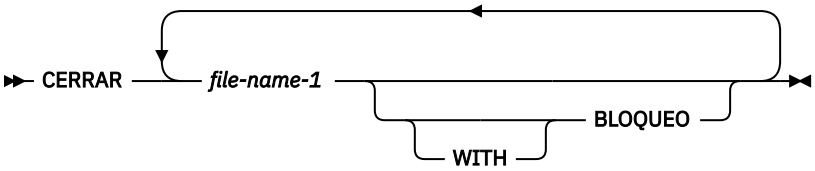
Formato 1: sentencia CLOSE para archivos secuenciales



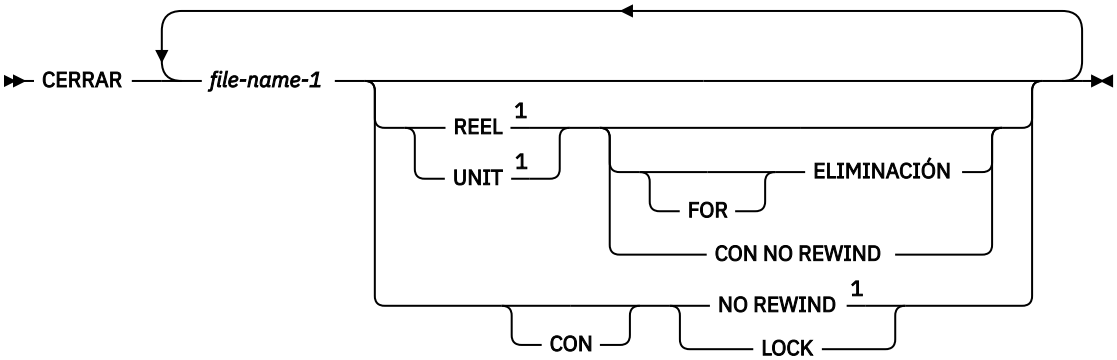
Notas:

[1] Las frases UNIT, REEL y NO REWIND se tratan como un comentario, aunque el estado del archivo se establecerá en 07, lo que indica una finalización satisfactoria de un CLOSE para un soporte no reel/unidad.

Formato 2: sentencia CLOSE para archivos indexados y relativos



Formato 3: sentencia CLOSE para archivos secuenciales de línea



Notas:

[1] Las frases UNIT, REEL y NO REWIND se tratan como un comentario, aunque el estado del archivo se establecerá en 07, lo que indica una finalización satisfactoria de un CLOSE para un soporte que no es de reel/unidad.

file-name-1
Designa el archivo sobre el que va a operar la sentencia CLOSE. Si se especifica más de un nombre de archivo, no es necesario que los archivos tengan la misma organización o acceso. *file-name-1* no debe ser un archivo de clasificación o fusión.

REEL y UNIT
REEL y UNIT son comprobación de sintaxis, pero no tienen ningún efecto en la ejecución del programa.

CON NO REWIND y PARA ELIMINACIÓN
WITH NO REWIND y FOR ELIMINAR son comprobación de sintaxis, pero no tienen ningún efecto en la ejecución del programa.

Una sentencia CLOSE solo se puede ejecutar para un archivo en modalidad abierta. Después de la ejecución satisfactoria de una sentencia CLOSE (sin la frase REEL/UNIT si se utiliza el formato 1):

- El área de registro asociada con el nombre de archivo ya no está disponible. La ejecución no satisfactoria de una sentencia CLOSE deja la disponibilidad de los datos de registro sin definir.
- Debe ejecutarse una sentencia OPEN para el archivo antes de que pueda ejecutarse cualquier otra sentencia de entrada/salida para el archivo y antes de que los datos se muevan a una entrada de descripción de registro asociada con el archivo.
- Se liberan los bloqueos de registro y los bloqueos de archivo retenidos por el conector de archivo en el archivo cerrado.

Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, la clave de estado de archivo asociada se actualiza cuando se ejecuta la sentencia CLOSE.

Si el archivo está en un estado abierto y la ejecución de una sentencia CLOSE no es satisfactoria, se ejecuta el procedimiento EXCEPTION/ERROR (si se especifica) para este archivo.

Efecto de la sentencia CLOSE en los tipos de archivo

Si se especifica la cláusula SELECT OPTIONAL en la entrada de control de archivos para un archivo, y el archivo no está disponible en tiempo de ejecución, no se realiza el proceso de fin de archivo estándar.

Los archivos se dividen en los tipos siguientes:

No reel/unidad
Archivo cuyo medio de entrada o salida es tal que el rebobinado, los carretes y las unidades no tienen ningún significado. Todos los archivos son de tipos de archivo no reel/unit.

Volumen único secuencial
Un archivo secuencial que está contenido por completo en un volumen. Este volumen puede contener más de un archivo. Todos los archivos son de un solo volumen.

Multivolumen secuencial
Archivo secuencial contenido en más de un volumen. Los archivos El concepto de volumen no tiene ningún significado.

Las combinaciones permitidas de frases de sentencia CLOSE se muestran en las tablas siguientes:

- Para archivos secuenciales: Archivos secuenciales y frases de sentencia CLOSE
- Para archivos indexados y relativos: Tabla 45 en la página 310
- Para archivos secuenciales de línea: Tabla 46 en la página 310

El significado de cada letra clave se muestra en Tabla 47 en la página 310.

Tabla 44. ***Archivos secuenciales y frases de sentencia CLOSE***			
Frases de sentencia CLOSE	**No reel/ unidad**	**Volumen único secuencial**	**Multivolumen secuencial**
CERRAR	C	C, G	A, C, G
CERRAR CON BLOQUEO	C, E	C, E, G	A, C, E, G

Tabla 45. ***Tipos de archivo indexados y relativos y frases de sentencia CLOSE***	
Frases de sentencia CLOSE	**Acción**
CERRAR	C
CERRAR CON BLOQUEO	C, E

Tabla 46. ***Tipos de archivo secuenciales de línea y frases de sentencia CLOSE***	
Frases de sentencia CLOSE	**Acción**
CERRAR	C
CERRAR CON BLOQUEO	C, E

Tabla 47. ***Significados de letras clave para tipos de archivo secuenciales***	
clave	**Acciones emprendidas**
A	**Volúmenes anteriores no afectados** **Archivos de entrada y de entrada/salida**: el proceso de conmutación de volumen estándar se realiza para todos los volúmenes anteriores (excepto los controlados por una sentencia CLOSE REEL/UNIT anterior). Los volúmenes subsiguientes no se procesan. **Archivos de salida**: el proceso de conmutación de volumen estándar se realiza para todos los volúmenes anteriores (excepto los controlados por una sentencia CLOSE REEL/UNIT anterior).
C	**Cerrar archivo** **Archivos de entrada y entrada-salida**: si el archivo está al final y se especifican registros de etiqueta, se realiza el procedimiento de etiqueta de finalización estándar. A continuación, se realizan los procedimientos de cierre del sistema estándar. Si el archivo está al final y no se especifican registros de etiqueta, el proceso de etiquetas no tiene lugar, pero se realizan procedimientos de cierre del sistema estándar. Si el archivo no está en su extremo, se realizan los procedimientos de cierre del sistema estándar, pero no hay ningún proceso de finalización de etiqueta. **Archivos de salida**: si se especifican registros de etiqueta, se realizan procedimientos de etiqueta de finalización estándar. A continuación, se realizan los procedimientos de cierre del sistema estándar. Si no se especifican registros de etiqueta, no se realizan los procedimientos de finalización de etiqueta, pero se realizan los procedimientos de cierre del sistema estándar.
E	**Bloqueo de archivo**: el compilador garantiza que este archivo no se pueda abrir de nuevo durante esta ejecución del programa objeto. Si el archivo es una unidad de cinta, se rebobinará y descargará.
G	**Rebobinar**: el volumen actual se coloca en su inicio físico.

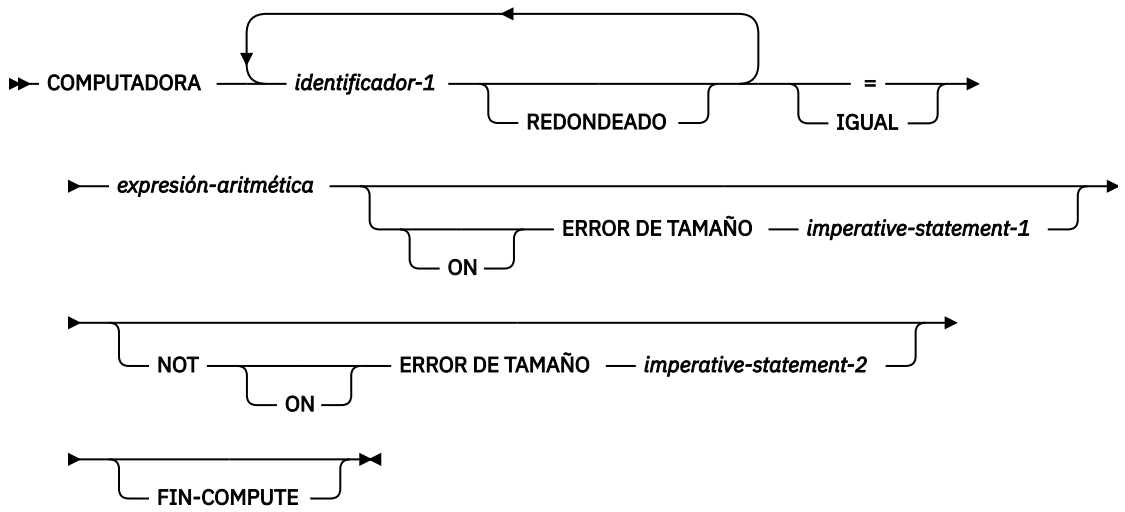
COMPUTE, sentencia

La sentencia COMPUTE asigna el valor de una expresión aritmética a uno o más elementos de datos.

Con la sentencia COMPUTE, las operaciones aritméticas se pueden combinar sin las restricciones de recepción de elementos de datos impuestas por las reglas para las sentencias ADD, RESTTRACT, MULTIPLY y DIVIDE.

Cuando se combinan operaciones aritméticas, la sentencia COMPUTE puede ser más eficiente que las sentencias aritméticas separadas escritas en una serie.

Formato



identifier-1

Debe nombrar un elemento numérico elemental o un elemento editado numérico elemental.

Puede nombrar un elemento de datos de coma flotante elemental.

Si *identifier-1* o el resultado de *expresión aritmética* (o ambos) son campos de fecha, Consulte "Almacenamiento de resultados aritméticos que implican campos de fecha" en la página 249 para obtener detalles sobre cómo se almacena el resultado en *identifier-1*. Si se especifica un campo de fecha de último año como *identifier-1*, el resultado de la *expresión aritmética* debe ser un valor distinto de fecha.

expresión-aritmética

Puede ser cualquier expresión aritmética, tal como se define en "Expresiones aritméticas" en la página 246.

Cuando se ejecuta la sentencia COMPUTE, el valor de *expresión aritmética* se calcula y se almacena como el nuevo valor de cada elemento de datos al que hace referencia el *identifier-1*.

Una expresión aritmética que consta de un único identificador, función numérica o literal permite al usuario establecer el valor de los elementos de datos a los que hace referencia *identifier-1* igual al valor de dicho identificador, función o literal.

No debe especificarse un campo de fecha de último año en la expresión aritmética.

Frase REDONDEADA

Para obtener una descripción de la frase REDONDEADA, consulte "Frase REDONDEADA" en la página 281.

Frases de ERROR DE TAMAÑO

Para obtener una descripción de las frases SIZE ERROR, consulte “Frases de ERROR DE TAMAÑO” en la página 281.

frase END-COMPUTE

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia COMPUTE. END-COMPUTE permite que una sentencia COMPUTE condicional se anide en otra sentencia condicional. END-COMPUTE también se puede utilizar con una sentencia COMPUTE imperativa.

Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

CONTINUE, sentencia

La sentencia CONTINUE no es una sentencia de operación. CONTINUE indica que no hay ninguna instrucción ejecutable presente.

Formato

sentencia DELETE

La sentencia DELETE elimina un registro de un archivo indexado o relativo. Para los archivos indexados, la clave se puede reutilizar para la adición de registros. Para los archivos relativos, el espacio está disponible para un nuevo registro con el mismo valor de RELATIVE KEY.

Cuando se ejecuta la sentencia DELETE, el archivo asociado debe estar abierto en modalidad I-O.

Formato

file-name-1
Debe definirse en una entrada FD en DATA DIVISION y debe ser el nombre de un archivo indexado o relativo.

Después de ejecutar correctamente una sentencia DELETE, el registro se elimina del archivo y ya no se puede acceder a él.

La ejecución de la sentencia DELETE no afecta al contenido del área de registro asociada con *file-name-1* o al contenido del elemento de datos al que hace referencia el nombre de datos especificado en la frase DEPENDING ON de la cláusula RECORD asociada con *file-name-1*.

Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, la clave de estado de archivo asociada se actualiza cuando se ejecuta la sentencia DELETE.

El indicador de posición de archivo no se ve afectado por la ejecución de la sentencia DELETE.

Si los bloqueos de registro están en vigor, se llevan a cabo las acciones siguientes:

- Si se especifica un bloqueo de registro único para el conector de archivo asociado con *file-name-1*:
  - Un bloqueo retenido por ese conector de archivo en el registro suprimido se libera al finalizar la ejecución satisfactoria de la sentencia DELETE.
  - Un bloqueo retenido por ese conector de archivo en otro registro se libera al principio de la ejecución de la sentencia DELETE.
- Si se especifica un bloqueo de varios registros para el conector de archivo asociado con *file-name-1*, todos los bloqueos retenidos en el registro suprimido se liberan al finalizar la ejecución satisfactoria de la sentencia DELETE.

Modalidad de acceso secuencial

Para un archivo en modalidad de acceso secuencial, la sentencia de entrada/salida anterior debe ser una sentencia READ ejecutada satisfactoriamente. Cuando se ejecuta la sentencia DELETE, el sistema elimina el registro recuperado por dicha sentencia READ.

Para un archivo en modalidad de acceso secuencial, no deben especificarse las frases INVALID KEY y NOT INVALID KEY. Se puede especificar un procedimiento EXCEPTION/ERROR.

Modo de acceso aleatorio o dinámico

En la modalidad de acceso aleatorio o dinámico, los resultados de la ejecución de la sentencia DELETE dependen de la organización del archivo: indexada o relativa.

Cuando se ejecuta la sentencia DELETE, el sistema elimina el registro identificado por el contenido del elemento de datos RECORD KEY principal para archivos indexados, o el elemento de datos RELATIVE KEY para archivos relativos. Si el archivo no contiene un registro de este tipo, existe una condición INVALID KEY. (Consulte “Condición de clave no válida” en la página 290.)

Frase INVALID KEY

Se puede omitir la frase INVALID KEY y un procedimiento EXCEPTION/ERROR aplicable.

La transferencia del control después de la ejecución satisfactoria de una sentencia DELETE, con la frase NOT INVALID KEY especificada, es a la sentencia imperativa asociada con la frase.

frase END-DELETE

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia DELETE. END-DELETE permite que una sentencia DELETE condicional se anide en otra sentencia condicional. END-DELETE también se puede utilizar con una sentencia DELETE imperativa.

Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

sentencia DISPLAY

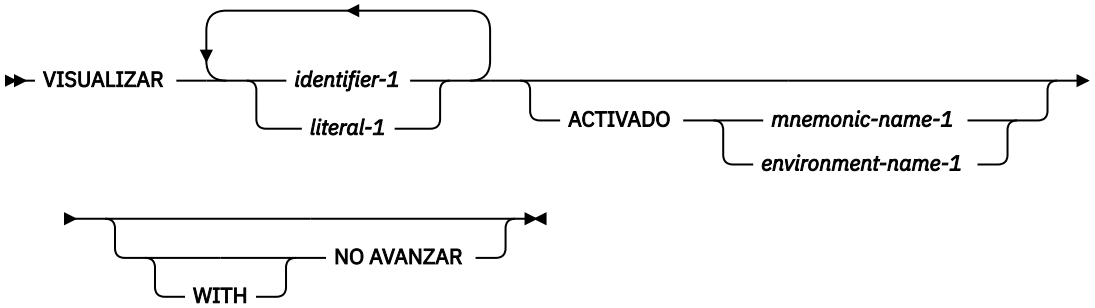
La sentencia DISPLAY transfiere el contenido de cada operando al dispositivo de salida. El contenido se visualiza en el dispositivo de salida en el orden, de izquierda a derecha, en el que se listan los operandos.

El archivo de destino se determina comprobando el nombre de entorno COBOL (CONSOLE, SYSIN, SYSIPT, SYSOUT, SYSLIST, SYSLST, SYSPUNCH y SYSPCH). Si se define una variable de entorno correspondiente al nombre de entorno COBOL, se utiliza el valor de la variable de entorno como identificador de archivo del sistema. Si la variable de entorno que corresponde al nombre de entorno COBOL no está establecida, DISPLAY en SYSOUT, SYSLIST o SYSLST está en el dispositivo de salida lógica del sistema (stdout).

Para SYSPUNCH y SYSPCH, la sentencia DISPLAY falla a menos que la variable de entorno correspondiente esté establecida para que apunte a un destino válido. Para obtener más información

sobre las variables de entorno, consulte *Ejemplo: Establecimiento y acceso a las variables de entorno* en *COBOL for Linux en x86 Guía de programación*.

Formato



identifier-1

Identifier-1 hace referencia a los datos que se van a visualizar. *Identifier-1* puede hacer referencia a cualquier elemento de datos excepto a un elemento de uso PROCEDURE-POINTER, FUNCTION-POINTER, o INDEX. *Identifier-1* no puede ser un nombre de índice.

Si *identifier-1* es un elemento de datos binario, decimal interno o de coma flotante interno , *identifier-1* se convierte automáticamente al formato externo como se indica a continuación:

- Los elementos decimales binarios e internos se convierten en decimales con zona . Los valores con signo negativo provocan un sobrescritura de signo de orden inferior.
- Los números de coma flotante internos se convierten en números de coma flotante externos para su visualización de forma que:
  - Un elemento COMP-1 se mostrará como si tuviera una cláusula PICTURE de coma flotante externa de-.9 (8)E-99.
  - Un elemento COMP-2 se mostrará como si tuviera una cláusula PICTURE de coma flotante externa de-.9 (17)E-99.

Los elementos de datos definidos con USAGE POINTER se convierten a un número decimal con zona que tiene una cláusula PICTURE implícita de PIC 9 (10).

Los elementos de datos descritos con el uso NATIONAL se convierten a la página de códigos asociada con el entorno local actual.

Ninguna otra categoría de datos requiere conversión.

Los campos de fecha se tratan como no fechas cuando se especifican en una sentencia DISPLAY. Es decir, DATE FORMAT se ignora y el contenido del elemento de datos se transfiere al dispositivo de salida tal cual.

Los elementos de datos DBCS, definidos explícita o implícitamente como USAGE DISPLAY-1, se transfieren al campo de envío del dispositivo de salida.

Los operandos DBCS y no DBCS pueden especificarse en una única sentencia DISPLAY.

literal-1

Puede ser cualquier literal o cualquier constante figurativa tal como se especifica en “Constantes figurativas” en la página 15. Cuando se especifica una constante figurativa, sólo se muestra una única aparición de dicha constante figurativa.

ACTIVADO

environment-name-1 o el nombre de entorno asociado con *mnemonic-name-1* debe estar asociado con un dispositivo de salida. Consulte “Párrafo SPECIAL-NAMES” en la página 97.

Cuando se omite la frase UPON, se asume el dispositivo lógico de salida del sistema. La lista de nombres de entorno válidos en una sentencia DISPLAY se muestra en Tabla 5 en la página 100.

CON NO AVANZAR

Cuando se especifique, la posición del dispositivo de salida no se cambiará de ninguna manera después de la visualización del último operando. Si el dispositivo de salida es capaz de posicionarse en una posición de carácter específica, permanecerá posicionado en la posición de carácter inmediatamente después del último carácter del último operando visualizado. Si el dispositivo de salida no es capaz de posicionarse en una posición de carácter específica, sólo se verá afectada la posición vertical, si procede. Esto puede provocar una sobreimpresión.

Si no se especifica la frase WITH NO AVANZANDO, después de que el último operando se haya transferido al dispositivo de salida, la posición del dispositivo de salida se restablecerá a la posición más a la izquierda de la siguiente línea del dispositivo.

La sentencia DISPLAY transfiere los datos del campo de envío al dispositivo de salida. El tamaño del campo de envío es el recuento total de bytes de todos los operandos listados. Si el dispositivo de salida es capaz de recibir datos del mismo tamaño que el elemento de datos que se está transfiriendo, se transfiere el elemento de datos. Si el dispositivo de salida no es capaz de recibir datos del mismo tamaño que el elemento de datos que se está transfiriendo, se aplica uno de los siguientes:

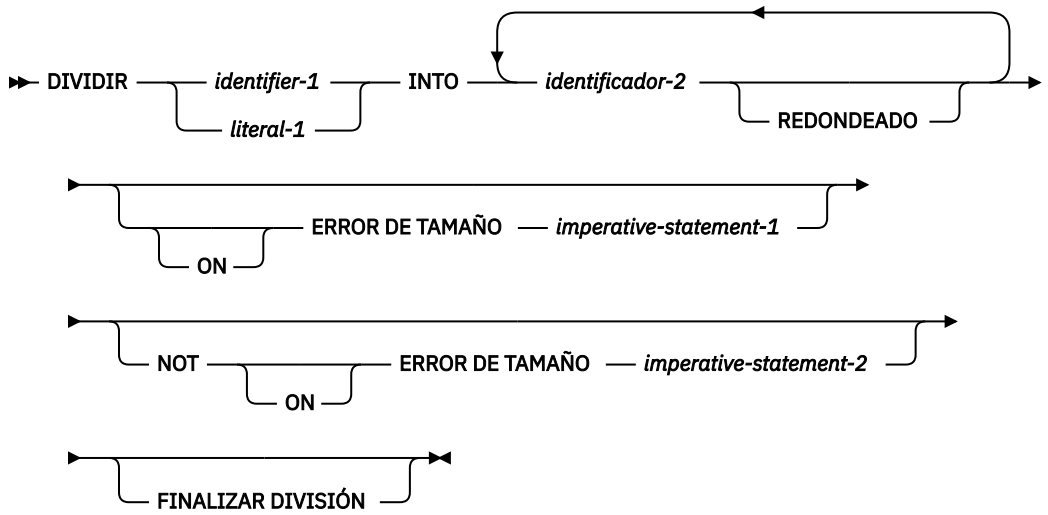
- Si el recuento total es menor que el máximo del dispositivo, las posiciones restantes más a la derecha se rellenan con espacios.
- Si el recuento total supera el máximo, se grabarán tantos registros como sean necesarios para visualizar todos los operandos. Cualquier operando que se imprima o visualice cuando se alcance el final de un registro continúa en el siguiente registro.

Si un operando DBCS debe dividirse entre varios registros, sólo se dividirá en un límite de doble byte.

sentencia DIVIDE

La sentencia DIVIDE divide un elemento de datos numérico en o por otros y establece los valores de los elementos de datos iguales al cociente y al resto.

Formato 1: sentencia DIVIDE



En el formato 1, el valor de *identifier-1* o *literal-1* se divide en el valor de *identifier-2*, y el cociente se almacena en *identifier-2*. Para cada aparición sucesiva de *identifier-2*, la división tiene lugar en el orden de izquierda a derecha en el que se especifica *identifier-2* .

Ejemplo de formato 1:

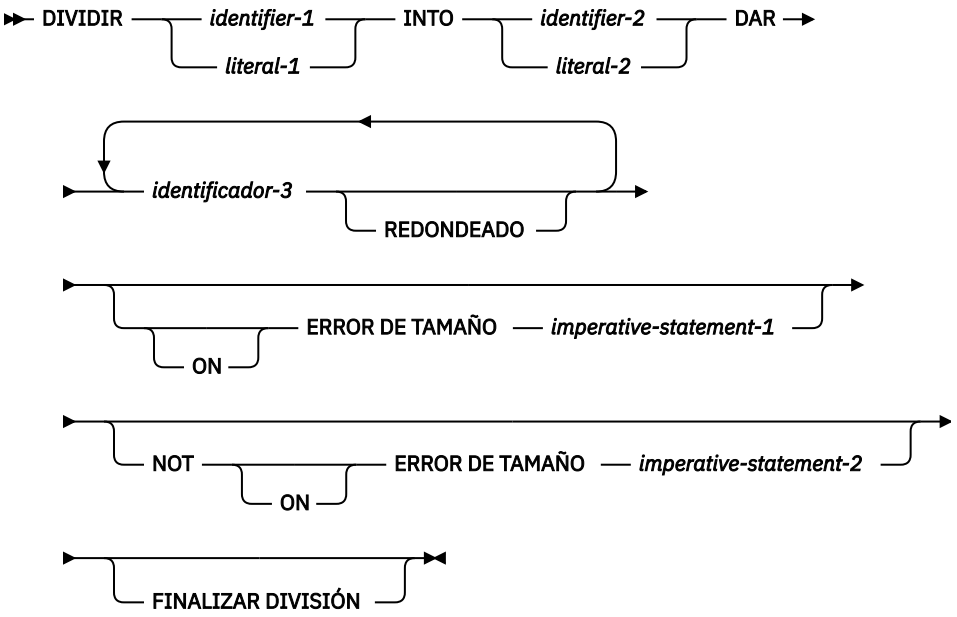
El valor en A se divide en el valor en B y el resultado se almacena en B. El valor en A no se modifica.

```
DIVIDE A INTO B
```

El valor en C se divide en el valor en D, almacenando la respuesta en D. El valor de C también se divide en E, almacenando el valor en E. El valor en C no se modifica.

```
DIVIDE C INTO D E
```

Formato 2: sentencia DIVIDE con frases INTO y DANDO



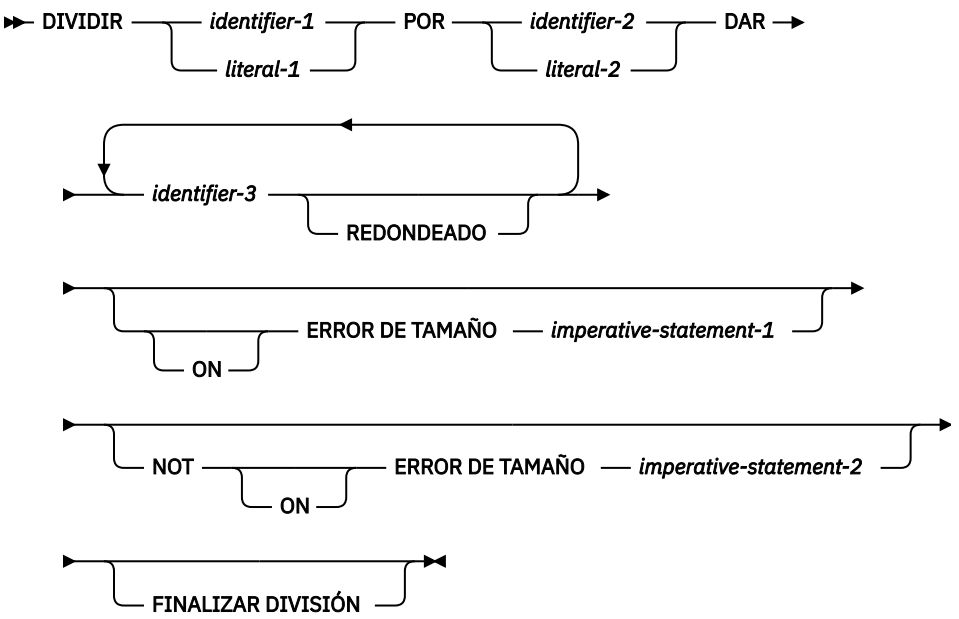
En el formato 2, el valor de *identifier-1* o *literal-1* se divide en el valor de *identifier-2* o *literal-2*. El valor del cociente se almacena en cada elemento de datos al que hace referencia *identifier-3*.

Ejemplo de formato 2:

El valor en A se divide en el valor en B y el resultado se almacena en C. Los valores de A y B no se modifican.

```
DIVIDE A INTO B GIVING C
```

Formato 3: sentencia DIVIDE con frases BY y DANDO



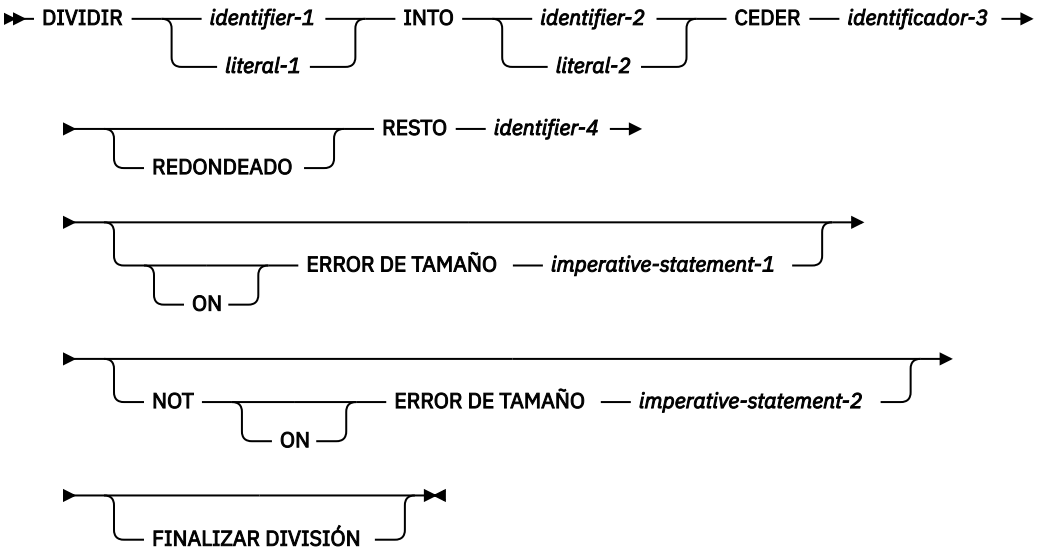
En el formato 3, el valor de *identifier-1* o *literal-1* se divide por el valor de *identifier-2* o *literal-2*. El valor del cociente se almacena en cada elemento de datos al que hace referencia *identifier-3*.

Ejemplo de formato 3:

El valor en A se divide por el valor en B y el resultado se almacena en C. Los valores de A y B no se modifican.

```
DIVIDE A BY B GIVING C
```

Formato 4: sentencia DIVIDE con frases INTO y RESTO



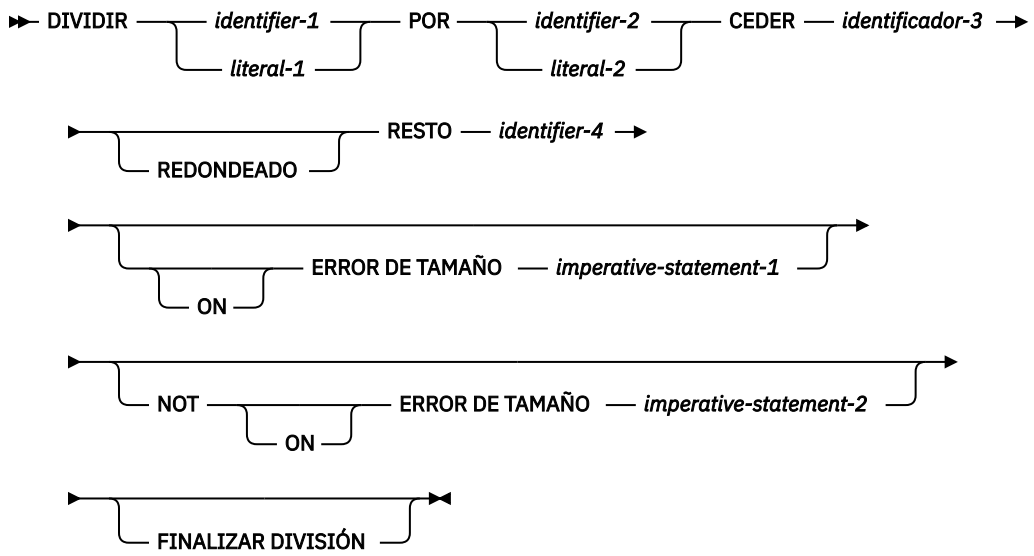
En el formato 4, el valor de *identifier-1* o *literal-1* se divide en *identifier-2* o *literal-2*. El valor del cociente se almacena en *identifier-3*, y el valor del resto se almacena en *identifier-4*.

Ejemplo de formato 4:

El valor en A se divide en el valor en B y los resultados se almacenan en C y el resto se almacena en D. Los valores de A y B no se modifican.

```
DIVIDE A INTO B GIVING C REMAINDER D
```

Formato 5: sentencia DIVIDE con frases BY y RESTO



En el formato 5, el valor de *identifier-1* o *literal-1* se divide por *identifier-2* o *literal-2*. El valor del cociente se almacena en *identifier-3*, y el valor del resto se almacena en *identifier-4*.

Ejemplo de formato 5:

El valor en A se divide por el valor en B y el resultado se almacena en C y el resto se almacena en D. Los valores de A y B no se modifican.

```
DIVIDE A BY B GIVING C REMAINDER D
```

Para todos los formatos:

identifier-1*, *identifier-2
Debe nombrar un elemento de datos numéricos elementales. *identifier-1* y *identifier-2* no pueden ser campos de fecha.

identifier-3*, *identifier-4
Debe nombrar un elemento numérico elemental o numérico editado.

Si *identifier-3* o *identifier-4* es un campo de fecha, consulte "Almacenamiento de resultados aritméticos que implican campos de fecha" en la página 249 para obtener detalles sobre cómo se almacena el cociente o el resto en *identifier-3*.

literal-1*, *literal-2
Debe ser un literal numérico.

En los formatos 1, 2 y 3, los elementos de datos de coma flotante y literales se pueden utilizar en cualquier lugar donde se pueda especificar un elemento de datos numérico o literal.

En los formatos 4 y 5, no se pueden utilizar elementos de datos de coma flotante o literales.

Frase REDONDEADA

Para los formatos 1, 2 y 3, consulte "Frase REDONDEADA" en la página 281.

Para los formatos 4 y 5, el cociente utilizado para calcular el resto se encuentra en un campo intermedio. El valor del campo intermedio se trunca en lugar de redondearse.

Frase RESTO

El resultado de restar el producto del cociente y el divisor del dividendo se almacena en *identifier-4*. Si *identifier-3*, el cociente, es un elemento editado numérico, el cociente utilizado para calcular el resto es un campo intermedio que contiene el cociente no editado.

La frase RESTO no es válida si el receptor o cualquiera de los operandos es un elemento de coma flotante.

Los subíndices para *identifier-4* en la frase RESTO se evalúan después de que el resultado de la operación de división se almacene en *identifier-3* de la frase CEDER.

Frases de ERROR DE TAMAÑO

Para los formatos 1, 2 y 3, consulte "Frases de ERROR DE TAMAÑO" en la página 281.

Para los formatos 4 y 5, si se produce un error de tamaño en el cociente, ningún cálculo restante es significativo. Por lo tanto, el contenido del campo de cociente (*identifier-3*) y el campo restante (*identifier-4*) no se modifican.

Si se produce un error de tamaño en el resto, el contenido del campo restante (*identifier-4*) no se modifica.

En cualquiera de estos casos, debe analizar los resultados para determinar qué situación se ha producido realmente.

Para obtener información sobre la frase NOT ON SIZE ERROR, consulte "Frases de ERROR DE TAMAÑO" en la página 281.

Frase END-DIVIDE

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia DIVIDE. END-DIVIDE convierte una sentencia DIVIDE condicional en una sentencia imperativa que se puede anidar en otra sentencia condicional. END-DIVIDE también se puede utilizar con una sentencia DIVIDE imperativa.

Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

ENTRY, sentencia

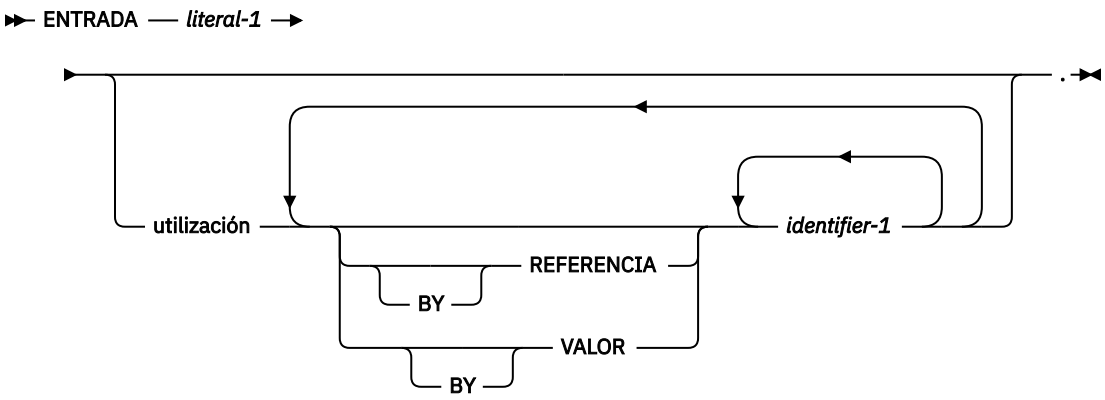
La sentencia ENTRY establece un punto de entrada alternativo en un subprograma COBOL llamado.

La sentencia ENTRY no se puede utilizar en:

- Programas que especifican un valor de retorno utilizando la frase PROCEDURE DIVISION RETURN. Para obtener más detalles, consulte la descripción de la frase DEVOLUCIÓN en "Cabecera PROCEDURE DIVISION" en la página 241.
- Programa anidado. Consulte "Programas anidados" en la página 81 para obtener una descripción de los programas anidados.

Cuando se ejecuta una sentencia CALL que especifica el punto de entrada alternativo en un programa de llamada, el control se transfiere a la siguiente sentencia ejecutable después de la sentencia ENTRY.

Formato



literal-1

Debe ser un literal alfanumérico que se ajuste a las reglas para la formación de un nombre de programa en un programa más externo (consulte "Párrafo PROGRAM-ID" en la página 88).

No debe coincidir con el ID de programa ni con ningún otro literal ENTRY de este programa.

No debe ser una constante figurativa.

La ejecución del programa llamado empieza en la primera sentencia ejecutable que sigue a la sentencia ENTRY cuyo literal corresponde al literal o identificador especificado en la sentencia CALL.

El nombre de punto de entrada en la sentencia ENTRY puede verse afectado por la opción de compilador PGMNAME. Para obtener detalles, consulte *PGMNAME* en la publicación *COBOL for Linux en x86 Guía de programación*.

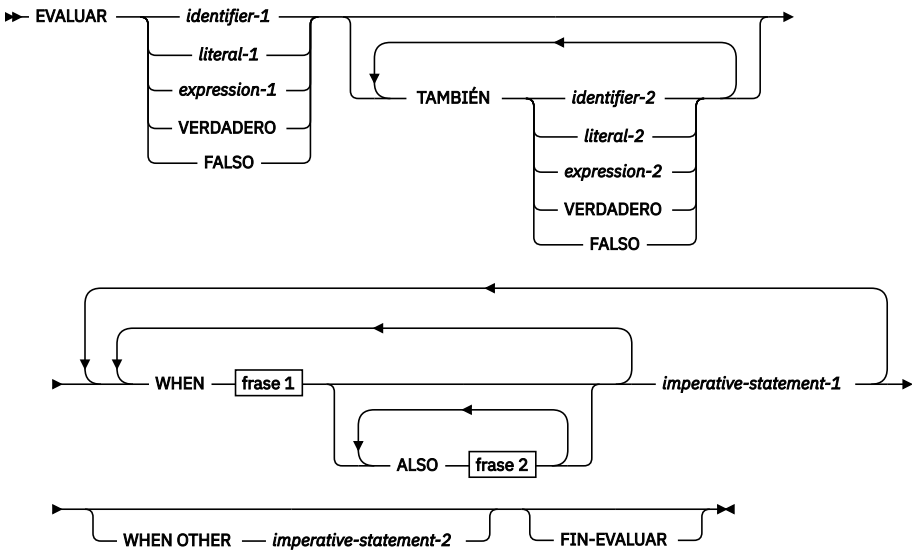
frase USING

Para ver una descripción de la frase USING, consulte "Cabecera PROCEDURE DIVISION" en la página 241.

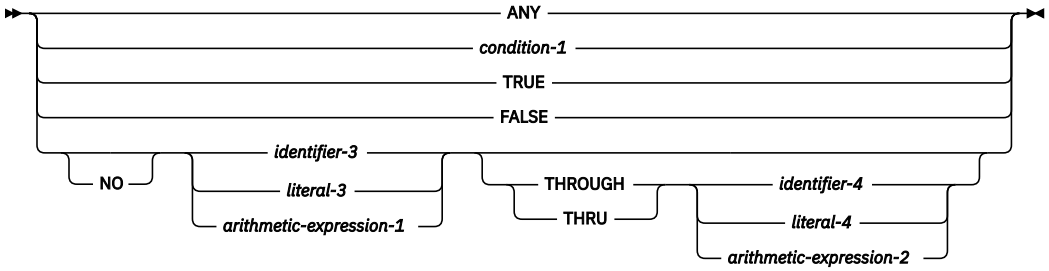
EVALUATE, sentencia

La sentencia EVALUATE proporciona una notación abreviada para una serie de sentencias IF anidadas. La sentencia EVALUATE puede evaluar varias condiciones. La acción posterior depende de los resultados de estas evaluaciones.

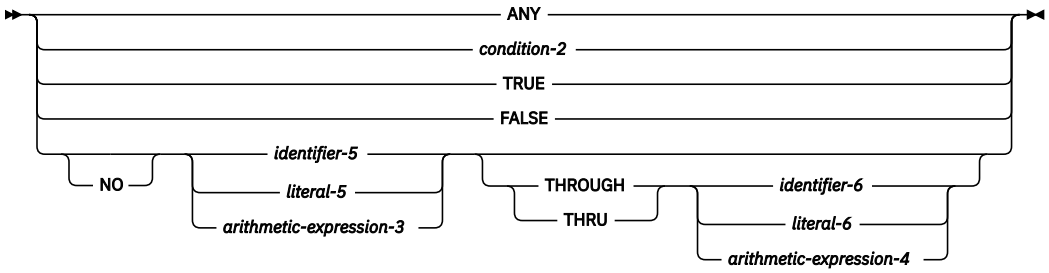
Formato



frase 1



frase 2



Operandos antes de la frase WHEN

Se interpretan de una de dos maneras, en función de cómo se especifiquen:

- Individualmente, se denominan *sujetos*de selección.
- De forma colectiva, se denominan un *conjunto* de temas de selección.

Operandos en la frase WHEN

Se interpretan de una de dos maneras, en función de cómo se especifiquen:

- Individualmente, se denominan *objetos* de selección
- Colectivamente, se denominan un *conjunto* de objetos de selección.

TAMBIÉN

Separa los sujetos de selección dentro de un conjunto de sujetos de selección; separa los objetos de selección dentro de un conjunto de objetos de selección.

THROUGH y THRU
Son equivalentes.

Dos operandos conectados mediante una frase THRU deben ser de la misma clase. Los dos operandos así conectados constituyen un único objeto de selección.

El número de objetos de selección dentro de cada conjunto de objetos de selección debe ser igual al número de sujetos de selección.

Cada objeto de selección dentro de un conjunto de objetos de selección debe corresponder al sujeto de selección que tenga la misma posición ordinal dentro del conjunto de sujetos de selección, de acuerdo con las siguientes reglas:

- Los identificadores, literales o expresiones aritméticas que aparecen dentro de un objeto de selección deben ser operandos válidos para compararlos con el operando correspondiente en el conjunto de sujetos de selección. Para las comparaciones que implican campos de fecha, consulte "Comparación de campos de fecha" en la página 264.
- *condition-1*, *condition-2*, o la palabra TRUE o FALSE que aparece como objeto de selección debe corresponder a una expresión condicional o a la palabra TRUE o FALSE en el conjunto de sujetos de selección.
- La palabra ANY puede corresponder a un asunto de selección de cualquier tipo.

Frase END-EVALUATE

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia EVALUATE. END-EVALUATE permite que una sentencia EVALUATE condicional se anide en otra sentencia condicional.

Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

Determinación de valores

La ejecución de la sentencia EVALUATE funciona como si cada asunto de selección y objeto de selección se evaluaran y se asignara un valor de carácter numérico, alfanumérico, DBCS o nacional; un rango de valores de carácter numérico, alfanumérico, DBCS o nacional; o un valor de verdad.

Estos valores se determinan de la forma siguiente:

- Cualquier asunto de selección especificado por *identifier-1*, *identifier-2*, ... y a cualquier objeto de selección especificado por *identifier-3* o *identifier-5* sin la frase NOT o THRU se les asigna el valor y la clase del elemento de datos al que hacen referencia.
- Cualquier asunto de selección especificado por *literal-1*, *literal-2*, ... y a cualquier objeto de selección especificado por *literal-3* o *literal-5* sin la frase NOT o THRU se les asigna el valor y la clase del literal especificado. Si *literal-3* o *literal-5* es la constante figurativa ZERO, QUOTE o SPACE, a la constante figurativa se le asigna la clase del asunto de selección correspondiente.
- Cualquier asunto de selección en el que *expression-1*, *expression-2*, ... se especifica como una expresión *aritmética* , y cualquier objeto de selección sin la frase NOT o THRU en la que se especifique *arithmetic-expression-1* o *arithmetic-expression-3* , se asignarán valores numéricos de acuerdo con las reglas para evaluar una expresión aritmética. (Consulte "Expresiones aritméticas" en la página 246.)
- Cualquier asunto de selección en el que *expression-1*, *expression-2*, ... se especifica como una expresión *condicional* y cualquier objeto de selección en el que se especifique *condition-1* o *condition-2* , se les asigna un valor de verdad de acuerdo con las reglas para evaluar expresiones condicionales. (Consulte "Expresiones condicionales" en la página 250.)
- A cualquier asunto de selección o cualquier objeto de selección especificado por las palabras TRUE o FALSE se le asigna un valor de verdad. El valor de verdad "true" se asigna a los elementos especificados con la palabra TRUE, y el valor de verdad "false" se asigna a los elementos especificados con la palabra FALSE.
- Cualquier objeto de selección especificado por la palabra ANY no se evalúa más.
- Si se especifica la frase THRU para un objeto de selección sin la frase NOT, el rango de valores incluye todos los valores que, cuando se comparan con el sujeto de selección, son mayores o iguales

que el primer operando y menores o iguales que el segundo operando de acuerdo con las reglas de comparación. Si el primer operando es mayor que el segundo operando, no hay valores en el rango.

- Si se especifica la frase NOT para un objeto de selección, los valores asignados a ese elemento no son todos valores iguales al valor, o rango de valores, que se habrían asignado al elemento si se hubiera omitido la frase NOT.

Comparación de sujetos y objetos de selección

A continuación, la ejecución de la sentencia EVALUATE continúa como si los valores asignados a los sujetos de selección y a los objetos de selección se compararan para determinar si alguna frase WHEN satisface el conjunto de sujetos de selección.

Esta comparación es la siguiente:

1. Cada objeto de seleccion dentro del conjunto de objetos de seleccion para la primera frase WHEN se compara con el sujeto de seleccion que tiene la misma posicion ordinal dentro del conjunto de sujetos de seleccion. Se debe cumplir una de las condiciones siguientes si se desea que se cumpla la comparación:
   a. Si a los elementos que se comparan se les asignan valores numéricos, alfanuméricos, DBCS o de caracteres nacionales, o un rango de valores numéricos, alfanuméricos, DBCS o de caracteres nacionales, la comparación se realiza si el valor, o un valor del rango de valores, asignado al objeto de selección es igual al valor asignado al sujeto de selección de acuerdo con las reglas de comparación.
   b. Si a los elementos que se comparan se les asignan valores de verdad, la comparación se satisface si a los elementos se les asignan valores de verdad idénticos.
   c. Si el objeto de selección que se compara se especifica mediante la palabra ANY, la comparación siempre se satisface, independientemente del valor del asunto de selección.
2. Si la comparacion anterior se satisface para cada objeto de seleccion dentro del conjunto de objetos de seleccion que se comparan, la frase WHEN que contiene ese conjunto de objetos de seleccion se selecciona como el que satisface el conjunto de sujetos de seleccion.
3. Si la comparación anterior no se satisface para cada objeto de selección dentro del conjunto de objetos de selección que se está comparando, ese conjunto de objetos de selección no satisface el conjunto de sujetos de selección.
4. Este procedimiento se repite para conjuntos subsecuentes de objetos de seleccion en el orden de su apariencia en el texto fuente, hasta que se seleccione una frase WHEN que satisfaga el conjunto de sujetos de seleccion o hasta que se agoten todos los conjuntos de objetos de seleccion.

Ejecución de la sentencia EVALUATE

Una vez completada la operación de comparación, continúa la ejecución de la sentencia EVALUATE.

- Si se selecciona una frase WHEN, la ejecución continúa con la primera *imperative-statement-1* después de la frase WHEN seleccionada. Tenga en cuenta que se permiten varias sentencias WHEN para un único *imperative-statement-1*.
- Si no se selecciona ninguna frase WHEN y se especifica una frase WHEN OTHER, la ejecución continúa con *imperative-statement-2*.
- Si no se selecciona ninguna frase WHEN y no se especifica ninguna frase WHEN OTHER, la ejecución continúa con la siguiente sentencia ejecutable después del delimitador de ámbito.
- El ámbito de ejecución de la sentencia EVALUATE se termina cuando la ejecución alcanza el final del ámbito de la frase WHEN o frase WHEN OTHER seleccionada, o cuando no se selecciona ninguna frase WHEN y no se especifica ninguna frase WHEN OTHER.

sentencia EXIT

La sentencia EXIT proporciona un punto final común para una serie de procedimientos. También proporciona una forma de salir de una sección, un párrafo o una sentencia PERFORM en línea.

Nota: La sentencia EXIT de formato 4, EXIT FUNCTION, todavía no está soportada.

Formato 1 (simple)

La sentencia EXIT de formato 1 proporciona un punto final común para una serie de procedimientos.

Formato 1

La sentencia EXIT de formato 1 le permite asignar un nombre de procedimiento a un punto determinado de un programa.

La sentencia EXIT de formato 1 se trata como una sentencia CONTINUE. Se ejecutan las sentencias que siguen a la sentencia EXIT.

Formato 2 (programa)

La sentencia EXIT PROGRAM especifica el final de un programa llamado y devuelve el control al programa de llamada.

Sólo puede especificar EXIT PROGRAM en la PROCEDURE DIVISION de un programa. EXIT PROGRAM no debe utilizarse en un procedimiento declarativo en el que se especifique la frase GLOBAL.

Formato 2

Si el control alcanza una sentencia EXIT PROGRAM en un programa que no posee el atributo INITIAL mientras opera bajo el control de una sentencia CALL (es decir, la sentencia CALL está activa), el control vuelve al punto de la rutina de llamada (programa) inmediatamente después de la sentencia CALL. El estado de la rutina de llamada es idéntico al que existía en el momento en que ejecutó la sentencia CALL. El contenido de los elementos de datos y el contenido de los archivos de datos compartidos entre la rutina de llamada y de llamada podría haberse cambiado. El estado del programa llamado no se modifica excepto que se considera que se han alcanzado los extremos de los rangos de todas las sentencias PERFORM ejecutadas.

La ejecución de una sentencia EXIT PROGRAM en un programa llamado que posee el atributo INITIAL es equivalente también a la ejecución de una sentencia CANCEL que hace referencia a dicho programa.

Si el control alcanza una sentencia EXIT PROGRAM y no hay ninguna sentencia CALL activa, el control pasa a través del punto de salida a la siguiente sentencia ejecutable.

Si un subprograma especifica la frase PROCEDURE DIVISION DEVOLVER, el valor del elemento de datos al que hace referencia la frase DEVOLVER se convierte en el resultado de la invocación del subprograma.

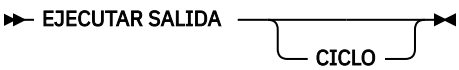
La sentencia EXIT PROGRAM debe ser la última sentencia de una secuencia de sentencias imperativas. Si no es así, las sentencias que siguen a EXIT PROGRAM no se ejecutarán si una sentencia CALL está activa.

Cuando no hay una siguiente sentencia ejecutable en un programa llamado, se ejecuta una sentencia EXIT PROGRAM implícita.

Formato 5 (ejecución en línea)

La sentencia EXIT PERFORM controla la salida de un PERFORM en línea sin utilizar una sentencia GO TO o un PERFORM ... Sentencia THROUGH.

Formato 5



Si especifica una sentencia EXIT PERFORM fuera de una sentencia PERFORM en línea, se ignora EXIT PERFORM.

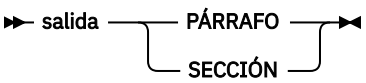
Cuando se ejecuta una sentencia EXIT PERFORM sin la frase CYCLE, el control se pasa a una sentencia CONTINUE implícita. Esta sentencia CONTINUE implícita sigue inmediatamente a la frase END-PERFORM que coincide con la sentencia PERFORM en línea más cercana y sin terminar.

Cuando se ejecuta una sentencia EXIT PERFORM con la frase CYCLE, el control se pasa a una sentencia CONTINUE implícita. Esta sentencia CONTINUE implícita precede inmediatamente a la frase END-PERFORM que coincide con la sentencia PERFORM en línea más cercana y sin terminar.

Formato 6 (procedimiento)

La sentencia EXIT PÁRRAFO controla la salida desde la mitad de un párrafo sin ejecutar ninguna sentencia siguiente dentro del párrafo. La sentencia EXIT SECTION controla la salida de una sección sin ejecutar ninguna sentencia siguiente dentro de la sección.

Formato 6



PÁRRAFO DE SALIDA

Cuando se ejecuta una sentencia EXIT EXISTENTE, el control se pasa a una sentencia CONTINUE implícita que sigue inmediatamente a la última sentencia explícita del párrafo actual. Este mecanismo de retorno reemplaza cualquier otro mecanismo de retorno que esté asociado con elementos de lenguaje, como PERFORM, SORT y USE para dicho párrafo.

SECCIÓN DE SALIDA

La sentencia EXIT SECTION sólo se puede especificar en una sección.

Cuando se ejecuta una sentencia EXIT SECTION, el control se pasa a un párrafo vacío sin nombre que sigue inmediatamente al último párrafo de la sección actual. Este mecanismo de retorno reemplaza cualquier otro mecanismo de retorno que esté asociado con elementos de lenguaje, como PERFORM, SORT y USE para esa sección.

sentencia GOBACK

La sentencia GOBACK funciona como la sentencia EXIT PROGRAM cuando se codifica como parte de un programa llamado y como la sentencia STOP RUN cuando se codifica en un programa principal.

La sentencia GOBACK especifica el final lógico de un programa llamado.

Formato

Una sentencia GOBACK debe aparecer como la única sentencia o como la última de una serie de sentencias imperativas en una frase porque cualquier sentencia que siga a GOBACK no se ejecuta. GOBACK no debe utilizarse en un procedimiento declarativo en el que se especifique la frase GLOBAL.

Si el control alcanza una sentencia GOBACK mientras una sentencia CALL está activa, el control vuelve al punto del programa de llamada inmediatamente después de la sentencia CALL, como en la sentencia EXIT PROGRAM.

Además, la ejecución de una sentencia GOBACK en un programa llamado que posee el atributo INITIAL es equivalente a ejecutar una sentencia CANCEL que hace referencia a dicho programa.

La tabla siguiente muestra la acción realizada para la sentencia GOBACK en un programa principal de programa principal o un subprograma.

Sentencia de terminación	**Programa principal**	**Subprograma**
GOBACK	Vuelve al programa de llamada. (Puede ser el sistema, lo que hace que la aplicación finalice.)	Vuelve al programa de llamada.

sentencia GO TO

La sentencia GO TO transfiere el control de una parte de PROCEDURE DIVISION a otra.

Los tipos de sentencias GO TO son:

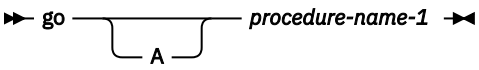
- Incondicional
- Condicional
- Alterado

VA A incondicional

La sentencia GO TO incondicional transfiere el control a la primera sentencia del párrafo o sección identificada por el nombre de procedimiento, a menos que la sentencia GO TO haya sido modificada por una sentencia ALTER.

Para obtener más información, consulte “Sentencia ALTER” en la página 299.

Formato 1: sentencia GO TO incondicional



procedure-name-1
Debe nombrar un procedimiento o una sección en la misma PROCEDURE DIVISION que la sentencia GO TO.

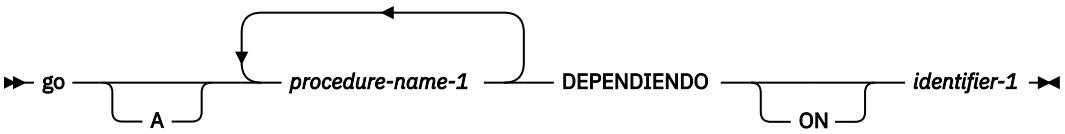
Cuando la sentencia GO TO incondicional no es la última sentencia de una secuencia de sentencias imperativas, las sentencias que siguen a GO TO no se ejecutan.

Cuando se hace referencia a un párrafo mediante una sentencia ALTER, el párrafo debe constar de un nombre de párrafo seguido de una sentencia GO TO incondicional o alterada.

GO condicional TO

La sentencia GO TO condicional transfiere el control a uno de una serie de procedimientos, en función del valor del elemento de datos al que hace referencia el *identifier-1*.

Formato 2: sentencia GO TO condicional



procedure-name-1
Debe ser un procedimiento o una sección en la misma PROCEDURE DIVISION que la sentencia GO TO. El número de nombres de procedimiento no debe superar 255.

identifier-1
Debe ser un elemento de datos elemental numérico que sea un entero. *identifier-1* no puede ser un campo de fecha con ventana.

Si es 1, el control se transfiere a la primera sentencia del procedimiento denominado por la primera aparición de *procedure-name-1*.

Si es 2, el control se transfiere a la primera sentencia del procedimiento denominado por la segunda aparición de *procedure-name-1*, y así sucesivamente.

Si el valor del identificador es distinto de un valor comprendido entre 1 y n (donde n es el número de nombres de procedimiento especificado en esta sentencia GO TO), no se produce ninguna transferencia de control. En su lugar, el control pasa a la siguiente sentencia en la secuencia normal de ejecución.

GO TO alterado

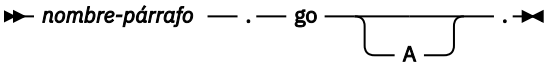
La sentencia GO TO alterada transfiere el control a la primera sentencia del párrafo especificado en la sentencia ALTER.

No puede especificar la sentencia GO TO alterada en los casos siguientes:

- Un programa que tiene el atributo RECURSIVE

Una sentencia ALTER que haga referencia al párrafo que contiene la sentencia GO TO modificada debe ejecutarse antes de que se ejecute la sentencia GO TO. De lo contrario, la sentencia GO TO actúa como una sentencia CONTINUE.

Formato 3: sentencia GO TO alterada

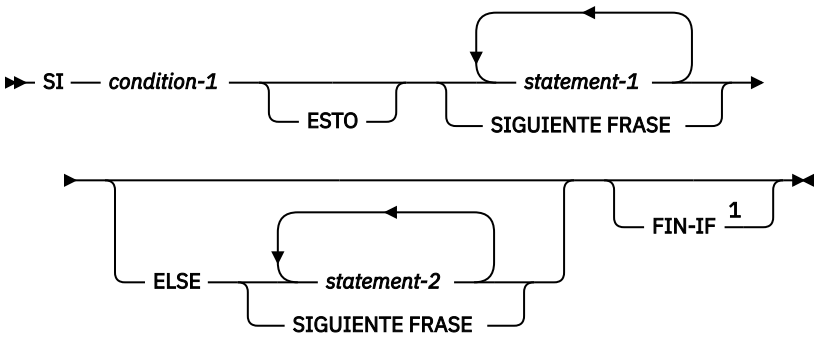


Cuando una sentencia ALTER hace referencia a un párrafo, el párrafo sólo puede constar del nombre de párrafo seguido de una sentencia GO TO incondicional o alterada.

sentencia IF

La sentencia IF evalúa una condición y proporciona acciones alternativas en el programa objeto, en función de la evaluación.

Formato



Notas:

[1] END-IF se puede especificar con *statement-2* o NEXT ORACIÓN.

condition-1
Puede ser cualquier condición simple o compleja, tal como se describe en "Expresiones condicionales" en la página 250.

statement-1, statement-2
Puede ser cualquiera de las siguientes opciones:

- Una declaración imperativa
- Una sentencia condicional
- Una sentencia imperativa seguida de una sentencia condicional

SIGUIENTE FRASE
La frase NEXT STATEMENT transfiere el control a una sentencia CONTINUE implícita inmediatamente después del siguiente *punto de separación*.

Cuando se especifica NEXT STATEMENT con END-IF, el control no pasa a la sentencia que sigue a END-IF. En su lugar, el control pasa a la sentencia después del periodo siguiente más cercano.

frase END-IF

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia IF. END-IF permite que una sentencia IF condicional se anide en otra sentencia condicional. Para obtener más información sobre terminadores de ámbito explícitos, consulte "Sentencias de ámbito delimitado" en la página 278.

El ámbito de una sentencia IF se puede terminar mediante cualquiera de las siguientes opciones:

- Una frase END-IF en el mismo nivel de anidamiento
- Un punto separador
- Si está anidado, mediante una frase ELSE asociada a una sentencia IF en un nivel superior de anidamiento

Transferencia de control

El tema describe las acciones que deben llevarse a cabo cuando las condiciones probadas son verdaderas o falsas.

Si la condición probada es verdadera, se lleva a cabo una de las acciones siguientes:

- Si se especifica *statement-1* , se ejecuta *statement-1* . Si *statement-1* contiene una ramificación de procedimiento o sentencia condicional, el control se transfiere de acuerdo con las reglas de dicha sentencia. Si *statement-1* no contiene una sentencia de ramificación de procedimiento, la frase ELSE,

si se especifica, se ignora y el control pasa a la siguiente sentencia ejecutable después del punto de separación o END-IF correspondiente.

- Si se especifica NEXT STATEMENT, el control pasa a una sentencia CONTINUE implícita inmediatamente después del siguiente punto de separación.

Si la condición probada es falsa, se lleva a cabo una de las acciones siguientes:

- Si se especifica ELSE *statement-2* , se ejecuta *statement-2* . Si *statement-2* contiene una sentencia condicional o de ramificación de procedimiento, se transfiere el control, de acuerdo con las reglas de dicha sentencia. Si *statement-2* no contiene una sentencia condicional o de ramificación de procedimiento, el control se pasa a la siguiente sentencia ejecutable después del punto de separación o END-IF correspondiente.
- Si se especifica ELSE NEXT STATEMENT, el control pasa a una CONTINUE STATEMENT implícita inmediatamente después del siguiente punto de separación.
- Si no se especifica ELSE *statement-2* ni ELSE NEXT STATEMENT, el control pasa a la siguiente sentencia ejecutable después del punto de separación o END-IF correspondiente.

Cuando se omite la frase ELSE, todas las sentencias que siguen a la condición y que preceden a la END-IF correspondiente o al punto de separación de la frase se consideran parte de *statement-1*.

Sentencias IF anidadas

Cuando una sentencia IF aparece como *statement-1* o *statement-2*, o como parte de *statement-1* o *statement-2*, esa sentencia IF está *anidada*.

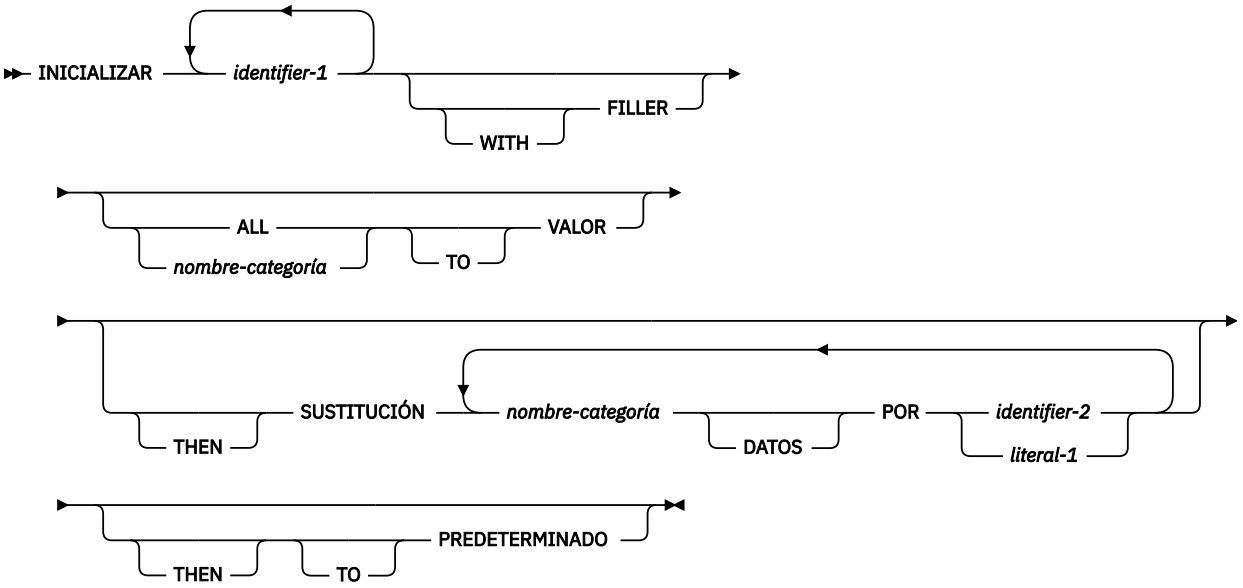
Cuando una sentencia IF aparece como *statement-1* o *statement-2*, o como parte de *statement-1* o *statement-2*, esa sentencia IF está *anidada*.

Las sentencias IF anidadas se consideran coincidentes con las combinaciones IF, ELSE y END-IF que proceden de izquierda a derecha. Por lo tanto, cualquier ELSE encontrado se compara con el IF precedente más cercano que todavía no se ha comparado con un ELSE o que no se ha terminado de forma implícita o explícita. Cualquier END-IF encontrado se compara con el IF precedente más cercano que no se ha terminado implícita o explícitamente.

Sentencia INITIALIZE

La sentencia INITIALIZE establece categorías seleccionadas de campos de datos en valores predeterminados. La sentencia INITIALIZE es funcionalmente equivalente a una o más sentencias MOVE.

Formato



Donde *nombre-categoría* es:

- ALFABÉTICO
- ALPHANUMERIC
- ALFANUMÉRICO EDITADO
- DBCS
- EGCS
- NACIONAL
- NACIONAL-EDITADO
- NUMÉRICO
- NUMÉRICO-EDITADO

identifier-1

Áreas de recepción.

identifier-1 debe hacer referencia a uno de los elementos siguientes:

- Un elemento de grupo alfanumérico
- Un elemento de grupo nacional
- Un elemento de datos elemental de una de las categorías siguientes:
  - Alfabético
  - Alfanumérico
  - Alfanumérico-editado
  - DBCS
  - Coma flotante externa
  - Coma flotante interna
  - Nacional
  - Nacional-editado
  - Numérico
  - Numérico-editado

- Un registro especial que es válido como operando receptor en una sentencia MOVE con *identifer-2* o *literal-1* como operando emisor.

identifier-1 hace referencia a un elemento elemental o a un elemento de grupo. El efecto de la ejecución de una sentencia INITIALIZE es como si se ejecutara una serie de sentencias MOVE implícitas, cada una de las cuales tiene un elemento de datos elemental como su operando receptor.

Cuando *identifier-1* hace referencia a un elemento de grupo nacional, *identifier-1* se procesa como un elemento de grupo.

identifier-2*, *literal-1
Áreas de envío.

Cuando *identifier-2* hace referencia a un elemento de grupo nacional, *identifier-2* se procesa como un elemento de datos elemental de categoría nacional.

identifier-2 debe hacer referencia a un elemento de datos elemental (o a un elemento de grupo nacional tratado como elemental) que sea válido como operando de envío en una sentencia MOVE con *identifier-1* como operando de recepción.

literal-1 debe ser un literal que sea válido como operando de envío en una sentencia MOVE con *identifier-1* como operando de recepción.

Se puede especificar un elemento con subíndice para *identifier-1*. Una tabla completa sólo se puede inicializar especificando *identifier-1* como un grupo que contiene la tabla completa.

Nota de uso: La entrada de descripción de datos para *identifier-1* puede contener la frase DEPENDING de la cláusula OCCURS. Sin embargo, no puede utilizar la sentencia INITIALIZE para inicializar un elemento ubicado de forma variable o un elemento de longitud variable.

La entrada de descripción de datos para *identifier-1* no debe contener una cláusula RENAMES.

Se pueden especificar registros especiales para *identifier-1* y *identifier-2* sólo si son campos de recepción válidos o campos de envío, respectivamente, para las sentencias MOVE implícitas.

Frase FILLER

Cuando se especifica la frase FILLER, se inicializarán los elementos de datos elementales de recepción que tienen una cláusula FILLER explícita o implícita.

Frase VALUE

Cuando se especifica la frase VALUE:

- Si se especifica ALL en la frase VALUE, es como si se hubieran especificado todas las categorías listadas en *nombre-categoría* .
- La misma categoría no se puede repetir en una frase VALUE.

Frase SUSTITUIR

Cuando se especifica la frase SUSTITUIR:

- *identifier-2* debe hacer referencia a un elemento de una categoría que sea válido como operando emisor en una sentencia MOVE a un elemento de la categoría correspondiente especificada en la frase SUSTITUIR.
- *literal-1* debe ser de una categoría que sea válida como operando de envío en una sentencia MOVE para un elemento de la categoría correspondiente especificada en la frase SUSTITUIR.
- Un literal de coma flotante, un elemento de datos de coma flotante interna de categoría o un elemento de datos de coma flotante externa de categoría se trata como si estuviera en la categoría NUMERIC.
- La misma categoría no se puede repetir en una frase SUSTITUIR.

Con la excepción de EGCS, la palabra clave que aparece después de la palabra SUSTITUIR corresponde a una categoría de datos que se muestra en la “Clases y categorías de datos” en la página 146.

EGCS en la frase SUSTITUIR es sinónimo de DBCS.

Reglas de sentencia INITIALIZE

El efecto de la ejecución de una sentencia INITIALIZE es como si se ejecutara una serie de sentencias MOVE implícitas, cada una de las cuales tiene un elemento de datos elemental como su operando receptor. Los operandos receptores de estas sentencias implícitas se definen en la regla 1 y los operandos emisores se definen en la regla 2.

1. El operando receptor en cada sentencia MOVE implícita se determina aplicando las reglas a, b y c en el orden en que aparecen a continuación. Tenga en cuenta que si una regla determinada no excluye un elemento de datos como receptor, puede excluirse como receptor cuando se aplica una regla posterior. Por ejemplo, si un elemento de datos no está excluido por la regla a, dicho elemento de datos puede seguir estando excluido por la regla b o la regla c.
   a. En primer lugar, los siguientes elementos de datos se excluyen como operandos de recepción:
      - Los identificadores que no son operandos de recepción válidos de una sentencia MOVE.
      - Elementos de datos elementales que tienen una cláusula FILLER explícita o implícita si no se especifica la frase FILLER.
      - Cualquier elemento de datos elemental subordinado a *identifier-1* cuya entrada de descripción de datos contenga una cláusula REDEFINES o RENAMES o esté subordinado a un elemento de datos cuya entrada de descripción de datos contenga una cláusula REDEFINES. Sin embargo, el propio *identifier-1* podría tener una cláusula REDEFINES o estar subordinado a un elemento de datos con una cláusula REDEFINES.
   b. En segundo lugar, un elemento de datos elemental es un posible elemento receptor en cualquiera de los siguientes casos:
      - *identifier-1*hace referencia de forma explícita a él.
      - Está contenido en el elemento de datos de grupo al que hace referencia *identifier-1*. Si el elemento de datos elementales es un elemento de tabla, cada aparición del elemento de datos elementales es un posible operando receptor.
   c. Finalmente, cada operando receptor posible es un operando receptor si se cumple al menos una de las condiciones siguientes:
      - Se especifica la frase VALUE, la categoría del elemento de datos elemental es una de las categorías especificadas o implícitas en la frase VALUE, y cualquiera de las condiciones siguientes es verdadera:
        – Se ha especificado una cláusula VALUE de formato de elemento de datos en la entrada de descripción de datos del elemento de datos elemental.
        – Se ha especificado una cláusula VALUE de formato de tabla en la entrada de descripción de datos del elemento elemental y esa cláusula VALUE especifica un valor para la aparición concreta del elemento de datos elemental.
      - Se especifica la frase SUSTITUIR y la categoría del elemento de datos elementales es una de las categorías especificadas en la frase SUSTITUIR.
      - Se especifica la frase DEFAULT.
      - No se ha especificado ni la frase SUSTITUIR ni la frase VALUE.
2. El operando emisor de cada sentencia MOVE implícita se determina de la siguiente manera:
   - Si el elemento de datos se califica como un operando receptor debido a la frase VALUE, el operando emisor se determina mediante el literal de la cláusula VALUE especificada en la entrada de descripción de datos del elemento de datos. Si el elemento de datos es un elemento de tabla, el literal de la cláusula VALUE que corresponde a la aparición que se inicializa determina el operando emisor. El operando emisor real es un literal que, cuando se mueve al operando receptor con una sentencia MOVE, produce el mismo resultado que el valor inicial del elemento de datos generado por la aplicación de la cláusula VALUE.

- Si el elemento de datos no se califica como un operando receptor debido a la frase VALUE, pero sí se califica debido a la frase SUSTITUIR, el operando emisor es el *literal-1* o *identifier-2* asociado con la categoría especificada en la frase SUSTITUIR.
- Si el elemento de datos no se ajusta a las dos reglas anteriores, el operando remitente utilizado depende de la categoría del operando receptor de la forma siguiente:
  - SPACE es el artículo de envío implícito para recibir artículos de categoría alfabética, alfanumérica, alfanumérica editada, DBCS, EGCS, nacional o nacional editada.
  - CERO es el artículo de envío implícito para recibir artículos de categoría numérica o numérica editada.

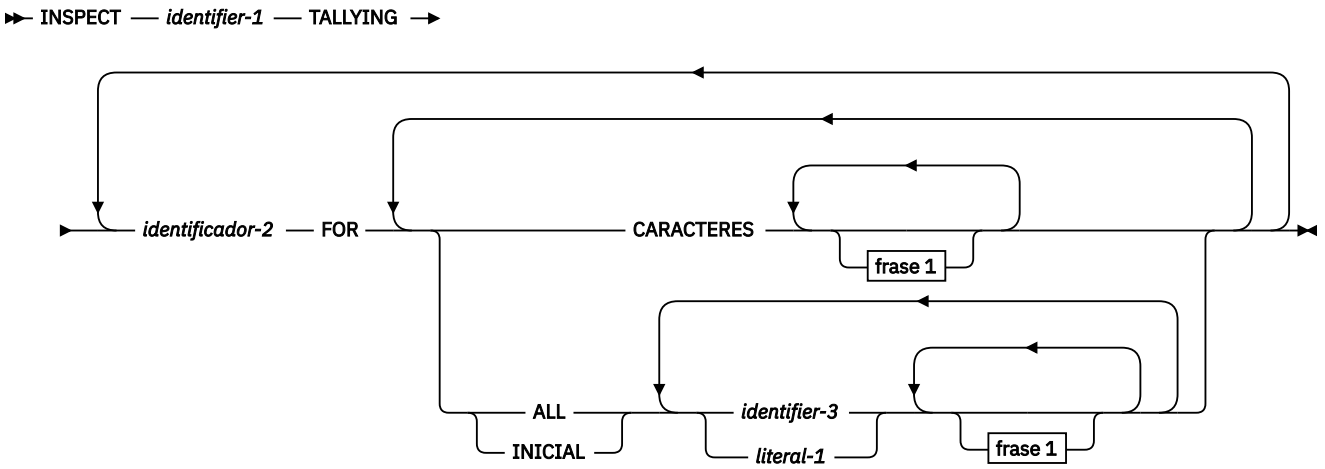
sentencia INSPECT

La sentencia INSPECT examina caracteres o grupos de caracteres en un elemento de datos.

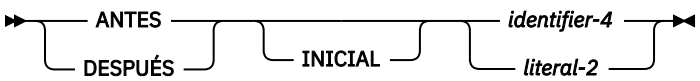
La sentencia INSPECT realiza las tareas siguientes:

- Cuenta las apariciones de un carácter específico (alfanumérico, DBCS o nacional) en un elemento de datos (formatos 1 y 3).
- Cuenta las apariciones de caracteres específicos y rellena todas o partes de un elemento de datos con caracteres especificados, como espacios o ceros (formatos 2 y 3).
- Convierte todas las apariciones de caracteres específicos en un elemento de datos en caracteres de sustitución proporcionados por el usuario (formato 4).

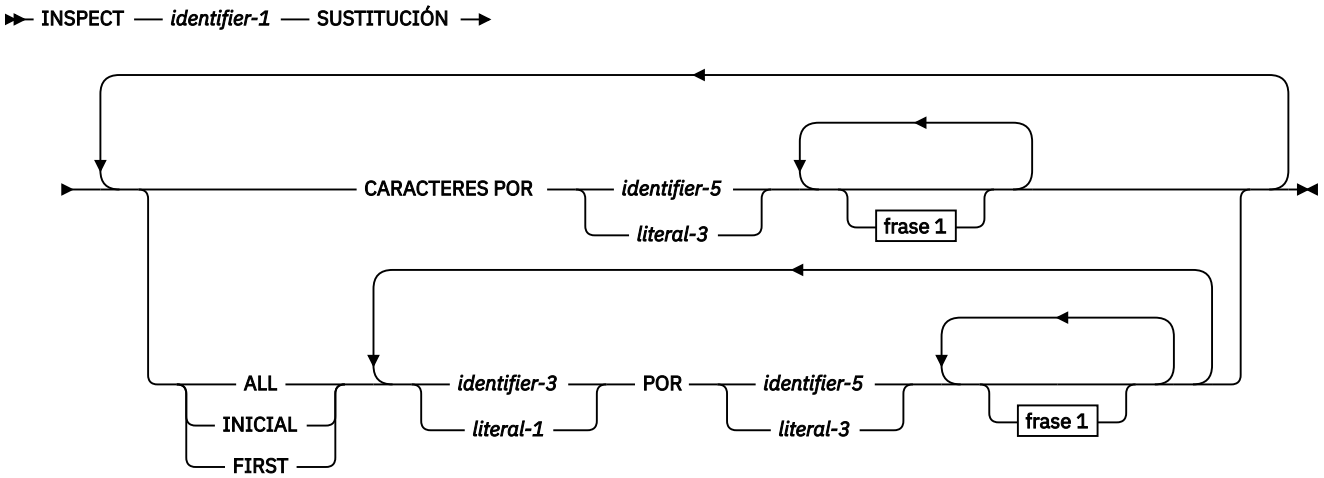
Formato 1: sentencia INSPECT con frase TALLYING



frase 1

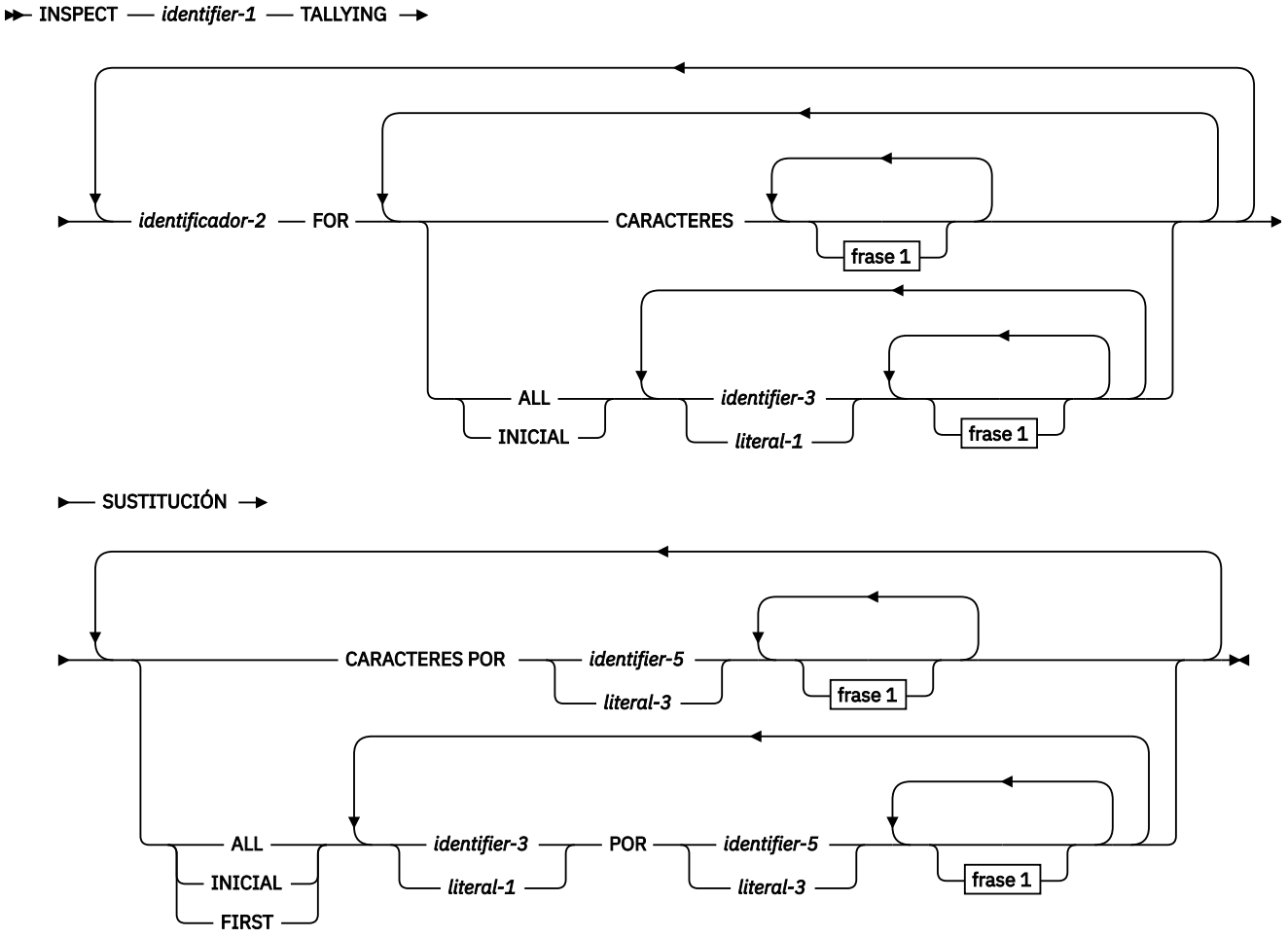


Formato 2: sentencia INSPECT con frase SUSTITUIR

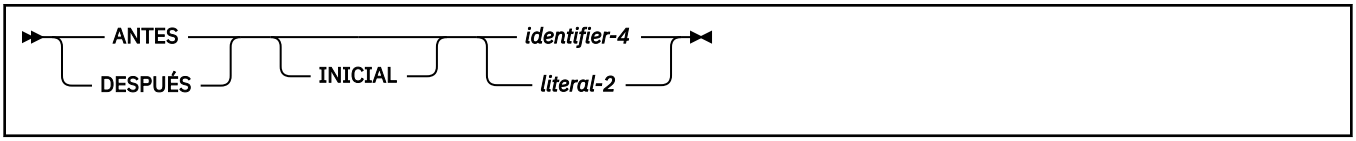


frase 1

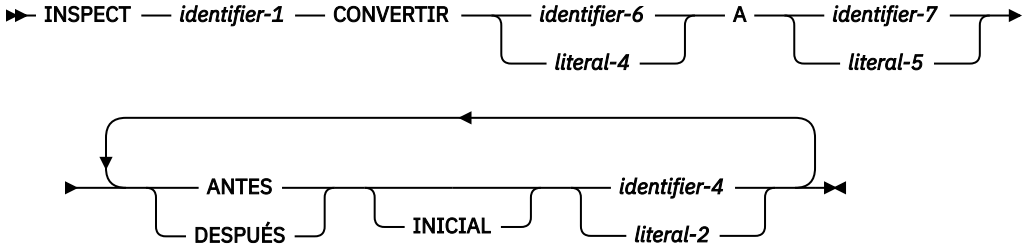
Formato 3: sentencia INSPECT con frases TALLYING y SUSTITUIR



frase 1



Formato 4: sentencia INSPECT con frase CONVERSION



identifier-1
Es el *elemento inspeccionado* y puede ser cualquiera de los elementos siguientes:

- Un elemento de grupo alfanumérico o un elemento de grupo nacional
- Elemento de datos elemental descrito explícita o implícitamente con el uso DISPLAY, DISPLAY-1o NATIONAL. El artículo puede tener cualquier categoría que sea válida para el uso seleccionado.

identifier-3 , identifier-4 , identifier-5 , identifier-6 , identifier-7
Debe hacer referencia a un elemento de datos elemental descrito explícita o implícitamente con el uso DISPLAY, DISPLAY-1o NATIONAL.

literal-1 , literal-2 , literal-3 , literal-4
Debe ser de categoría alfanumérica, DBCS o nacional.

Cuando *identifier-1* es de uso NACIONAL, los literales deben ser de categoría nacional.

Cuando *identifier-1* es de uso DISPLAY-1, los literales deben ser de la categoría DBCS.

Cuando *identifier-1* es de uso DISPLAY, los literales deben ser de categoría alfanumérica.

Cuando *identifier-1* es de uso, los literales DISPLAY-1 (DBCS) pueden ser la constante figurativa SPACE.

Cuando *identifier-1* es de uso DISPLAY o NATIONAL, los literales pueden ser cualquier constante figurativa que no empiece por la palabra ALL, tal como se especifica en "Constantes figurativas" en la página 15. La constante figurativa se trata como un literal alfanumérico de un carácter cuando *identifier-1* es de uso DISPLAY, y como un literal nacional de un carácter cuando *identifier-1* es de uso NATIONAL.

Todos los identificadores (excepto *identifier-2*) deben tener el mismo uso que *identifier-1*. Todos los literales deben tener una categoría alfanumérica, DBCS o nacional cuando *identifier-1* tiene el uso DISPLAY, DISPLAY-1o NATIONAL, respectivamente.

Ninguno de los identificadores de una sentencia INSPECT puede ser campos de fecha con ventana.

Frase TALLYING (formatos 1 y 3)

Esta frase cuenta las apariciones de un carácter específico o un carácter especial en un elemento de datos.

Cuando *identifier-1* es un elemento de datos DBCS, se cuentan los caracteres DBCS; cuando *identifier-1* es un elemento de datos de uso nacional, se cuentan los caracteres nacionales (unidades de codificación); de lo contrario, se cuentan los caracteres alfanuméricos (bytes).

identifier-2
Es el *campo de recuento*, y debe ser un elemento entero elemental definido sin el símbolo P en su serie de caracteres PICTURE.

identifier-2 no puede ser de categoría de coma flotante externa.

Debe inicializar *identifier-2* antes de que empiece la ejecución de la sentencia INSPECT.

Nota de uso: El campo de recuento puede ser un elemento de datos entero definido con el uso NATIONAL.

identifier-3* o *literal-1
Es el *campo de recuento* (el elemento cuyas apariciones se contabilizarán).

Caracteres
Cuando se especifica CHARACTERS y no se especifica la frase BEFORE ni AFTER, el campo de recuento (*identifier-2*) se incrementa en 1 para cada carácter (incluido el carácter de espacio) del elemento inspeccionado (*identifier-1*). Por lo tanto, la ejecución de una sentencia INSPECT con la frase TALLYING aumenta el valor en el campo de recuento por el número de posiciones de carácter en el elemento inspeccionado.

TODOS
Cuando se especifica ALL y no se especifica la frase BEFORE ni AFTER, el campo de recuento (*identifier-2*) se incrementa en 1 para cada aparición no solapada de la comparación de recuento (*identifier-3* o *literal-1*) en el elemento inspeccionado (*identifier-1*), empezando por la posición del carácter más a la izquierda y continuando por la derecha.

INICIAL
Cuando se especifica LÍDER y no se especifica la frase BEFORE ni AFTER, el campo de recuento (*identifier-2*) se incrementa en 1 para cada aparición contigua no solapada de la comparación de recuento en el elemento inspeccionado (*identifier-1*), siempre que la aparición más a la izquierda se produzca en el punto en que la comparación comenzó en el primer ciclo de comparación para el que la comparación de recuento puede participar.

FIRST (sólo formato 3)
Cuando se especifica FIRST y no se especifica la frase BEFORE ni AFTER, el campo de sustitución sustituye la aparición más a la izquierda del campo de asunto en el elemento inspeccionado (*identifier-1*).

Ejemplo de frase TALLYING

```
WORKING-STORAGE SECTION.
77 CNTR PIC 9(3) COMP.
77 CHARS PIC X(18).
PROCEDURE DIVISION.
...........
     MOVE 'In order to form a' To CHARS
     MOVE 0 To CNTR
     INSPECT CHARS TALLYING CNTR FOR ALL SPACES
     DISPLAY 'Number of spaces = ' CNTR
```

Salida de ejemplo:

```
Number of spaces = 004
```

Frase SUSTITUIR (formatos 2 y 3)

Esta frase rellena todos o partes de un elemento de datos con caracteres especificados, como espacios o ceros.

identifier-3* o *literal-1
Es el *campo de asunto*, que identifica los caracteres que se van a sustituir.

identifier-5* o *literal-3
Es el *campo de sustitución* (el elemento que sustituye al campo de asunto).

El campo de asunto y el campo de sustitución deben tener la misma longitud.

CARACTERES POR
Cuando se utiliza la frase CHARACTERS BY, el campo de sustitución debe tener una posición de carácter de longitud.

Cuando se especifica CHARACTERS BY y no se especifica la frase BEFORE ni AFTER, el campo de sustitución sustituye a cada carácter del elemento inspeccionado (*identifier-1*), empezando por la posición del carácter situado más a la izquierda y continuando por el situado más a la derecha.

TODOS
Cuando se especifica ALL y no se especifica la frase BEFORE ni AFTER, el campo de sustitución sustituye cada aparición no solapada del campo de asunto en el elemento inspeccionado (*identifier-1*), empezando por la posición del carácter situado más a la izquierda y continuando por el situado más a la derecha.

INICIAL
Cuando se especifica LÍDER y no se especifica la frase BEFORE ni AFTER, el campo de sustitución sustituye cada aparición contigua no solapada del campo de asunto en el elemento inspeccionado (*identifier-1*), siempre que la aparición más a la izquierda se produzca en el punto en que la comparación comenzó en el primer ciclo de comparación para el que este campo de sustitución puede participar.

PRIMERO
Cuando se especifica FIRST y no se especifica la frase BEFORE ni AFTER, el campo de sustitución sustituye la aparición más a la izquierda del campo de asunto en el elemento inspeccionado (*identifier-1*).

Cuando se especifican las frases TALLYING y SUSTITUIR (formato 3), la sentencia INSPECT se ejecuta como si se hubiera especificado una sentencia INSPECT TALLYING (formato 1), seguida inmediatamente por una sentencia INSPECT SUSTITUIR (formato 2).

Se aplican las siguientes reglas de sustitución:

- Cuando el campo de asunto es una constante figurativa, el campo de sustitución de un carácter sustituye a cada carácter del elemento inspeccionado que es equivalente a la constante figurativa.
- Cuando el campo de sustitución es una constante figurativa, el campo de sustitución sustituye cada aparición no solapada del campo de asunto en el artículo inspeccionado.
- Cuando los campos de asunto y sustitución son series de caracteres, la serie de caracteres especificada en el campo de sustitución sustituye cada aparición no solapada del campo de asunto en el elemento inspeccionado.
- Después de que se haya producido la sustitución en una posición de carácter determinada en el elemento inspeccionado, no se realiza ninguna sustitución adicional para dicha posición de carácter en esta ejecución de la sentencia INSPECT.

Ejemplo de frase SUSTITUIR

```
WORKING-STORAGE SECTION.
77 CNTR PIC 9(3) COMP.
77 CHARS PIC X(18).
PROCEDURE DIVISION.
  . . .
      MOVE 'more,perfect,union' To CHARS
      MOVE 0 To CNTR

      INSPECT CHARS TALLYING CNTR FOR ALL ','
        REPLACING ALL ',' BY SPACES

      DISPLAY 'Number of commas replaced = ' CNTR
      DISPLAY 'CHARS is now = ' CHARS
```

El resultado de ejemplo:

```
Number of commas replaced = 002
CHARS is now = more perfect union
```

Frases ANTES y DESPUÉS (todos los formatos)

Esta frase reduce el conjunto de elementos que se están contabilizando o sustituyendo.

No se puede especificar más de una frase BEFORE y una frase AFTER para ninguna frase ALL, BEFORE, CHARACTERS, FIRST o CONVERSION.

identifier-4* o *literal-2
Es el *delimitador*.

Los delimitadores no se cuentan ni se sustituyen.

INITIAL
La primera aparición de un elemento especificado.

Las frases ANTES y DESPUÉS cambian la forma en que se realiza el recuento y la sustitución:

- Cuando se especifica BEFORE, el recuento o la sustitución del elemento inspeccionado (*identifier-1*) empieza en la posición más a la izquierda del carácter y continúa hasta que se encuentra la primera aparición del delimitador. Si no hay ningún delimitador en el elemento inspeccionado, el recuento o la sustitución continúa hacia la posición de carácter más a la derecha.
- Cuando se especifica AFTER, el recuento o la sustitución del elemento inspeccionado (*identifier-1*) empieza por la primera posición de carácter a la derecha del delimitador y continúa hacia la posición de carácter más a la derecha en el elemento inspeccionado. Si no hay ningún delimitador en el elemento inspeccionado, no se realiza ningún recuento o sustitución.

Ejemplo de frases BEFORE y AFTER

```
WORKING-STORAGE SECTION.
77 CNTR1 PIC 9(3) COMP.
77 CNTR2 PIC 9(3) COMP.
77 CNTR3 PIC 9(3) COMP.
77 CHARS PIC X(50).

PROCEDURE DIVISION.

     MOVE '$some.confusing_text with.hyphens-periods.spaces'
         To CHARS
     MOVE 0 To CNTR1 CNTR2 CNTR3
     INSPECT CHARS
        TALLYING CNTR1 FOR CHARACTERS AFTER '$' BEFORE '_'
                 CNTR2 FOR ALL '.' AFTER  'h'
                 CNTR3 FOR ALL '.' BEFORE INITIAL 'a'
        REPLACING ALL '.' BY SPACES BEFORE 'h'
                  ALL '"' BY SPACES AFTER  's'
                  ALL '_' BY SPACES
     DISPLAY 'Number of characters after $ and before _  =' CNTR1
     DISPLAY 'Number of periods after h =' CNTR2
     DISPLAY 'Number of periods before initial a =' CNTR3
     DISPLAY 'CHARS is now = ' CHARS
```

El resultado de ejemplo:

Nota: Para cada frase BEFORE y AFTER, el escaneo continúa después de que se realice la frase anterior en lugar de empezar al principio de CHARS.

```
Number of characters after $ and before _  =014
Number of periods after h =002
Number of periods before initial a =000
CHARS is now = $some confusing text with.hyphens-periods.spaces
```

Frase CONVERSION (formato 4)

Esta frase convierte todas las apariciones de un carácter específico o serie de caracteres en un elemento de datos (*identifier-1*) en caracteres de sustitución proporcionados por el usuario.

identifier-6* o *literal-4
Especifica la serie de caracteres que se debe *sustituir*.

El mismo carácter no debe aparecer más de una vez en *literal-4* o *identifier-6*.

identifier-7* o *literal-5
Especifica la serie de caracteres de *sustitución* .

La serie de caracteres de sustitución (*identifier-7* o *literal-5*) debe tener el mismo tamaño que la serie de caracteres sustituida (*identifier-6* o *literal-4*).

Una sentencia INSPECT format-4 se interpreta y ejecuta como si se hubiera escrito una sentencia INSPECT format-2 con una serie de frases ALL (una para cada carácter de *literal-4*), especificar el mismo *identifier-1*. El efecto es como si se hiciera referencia a cada carácter individual de *literal-4* como *literal-1*, y el carácter único correspondiente de *literal-5* al que se hace referencia como *literal-3*. La correspondencia entre los caracteres de *literal-4* y los caracteres de *literal-5* es por posición ordinal dentro del elemento de datos.

Si *identifier-4*, *identifier-6*, o *identifier-7* ocupa la misma área de almacenamiento que *identifier-1*, el resultado de la ejecución de esta sentencia no está definido, incluso si están definidos por la misma entrada de descripción de datos.

La tabla siguiente describe el tratamiento de los elementos de datos que se pueden utilizar como operando en la sentencia INSPECT:

Tabla 48. ***Tratamiento del contenido de los elementos de datos***

Cuando se hace referencia mediante cualquier identificador excepto *identifier-2*, el contenido de cada elemento de la categoría ...	**Se trata ...**
Alfanumérico o alfabético	Como serie de caracteres alfanuméricos
DBCS	Como serie de caracteres DBCS
Nacional	Como serie de caracteres nacionales
Editado alfanumérico, editado numérico con uso DISPLAY, o numérico con uso DISPLAY (sin signo, decimal externo)	Como si se redefiniera como categoría alfanumérica, con la sentencia INSPECT haciendo referencia a una serie de caracteres alfanuméricos
Nacional-editado, numérico-editado con uso NACIONAL o numérico con uso NACIONAL (sin signo, decimal externo)	Como si se redefiniera como categoría nacional, con la sentencia INSPECT haciendo referencia a una serie de caracteres nacional
Numérico con uso DISPLAY (signo, decimal externo)	Como si se trasladara a un elemento decimal externo sin signo de uso DISPLAY con la misma longitud que el identificador y, a continuación, redefinido como alfanumérico de categoría, con la sentencia INSPECT haciendo referencia a una serie de caracteres alfanuméricos Si el signo es un carácter separado, el byte que contiene el signo no se examina y, por lo tanto, no se sustituye. Si el elemento al que se hace referencia es *identifier-1*, la serie resultante de cualquier acción de sustitución o conversión se copia de nuevo en *identifier-1*.

*Tabla 48. **Tratamiento del contenido de los elementos de datos** (continuación)*	
Cuando se hace referencia mediante cualquier identificador excepto *identifier-2*, el contenido de cada elemento de la categoría ...	**Se trata ...**
Numérico con uso NATIONAL (con signo, decimal externo)	Como si se hubiera movido a un elemento decimal externo sin signo de uso NATIONAL con la misma longitud que el identificador y, a continuación, se ha redefinido como de categoría nacional, con la sentencia INSPECT que hace referencia a una serie de caracteres nacional Si el signo es un carácter separado, el byte que contiene el signo no se examina y, por lo tanto, no se sustituye. Si el elemento al que se hace referencia es *identifier-1*, la serie resultante de cualquier acción de sustitución o conversión se copia de nuevo en *identifier-1*.
Coma flotante externa con uso DISPLAY	Como si se redefiniera como categoría alfanumérica, con la sentencia INSPECT haciendo referencia a una serie de caracteres alfanuméricos
Coma flotante externo con uso NATIONAL	Como si se redefiniera como categoría nacional, con la sentencia INSPECT que hace referencia a una serie de caracteres nacional

Ejemplo de frase CONVERSION

El ejemplo siguiente muestra el uso de las frases INSPECT CONVERSION con AFTER y BEFORE para examinar y sustituir caracteres en el elemento de datos DATA-4. Todos los caracteres que siguen a la primera instancia del carácter/pero que preceden a la primera instancia del carácter? (si hay alguno) se convierten de minúsculas a mayúsculas.

```
01  DATA-4           PIC X(11).
. . .
    INSPECT DATA-4
      CONVERTING
         "abcdefghijklmnopqrstuvwxyz" TO
         "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
      AFTER INITIAL "/"
      BEFORE INITIAL "?"
```

Tabla 49. Resultado de ejemplo CONVERSION	
DATA-4 antes de convertir	**DATA-4 después de convertir**
a/five/?seis	a/FIVE/?seis
r/Rexx/RRRr	r/REXX/RRRR
zfour? inspeccionar	zfour? inspeccionar

Flujo de datos

Excepto cuando se especifica la frase BEFORE o AFTER, la inspección empieza en la posición más a la izquierda del elemento inspeccionado (*identifier-1*) y continúa carácter por carácter a la posición más a la derecha.

Las comparaciones de las frases siguientes se comparan en el orden de izquierda a derecha en el que se especifican en la sentencia INSPECT:

- TALLYING (*literal-1* o *identifier-3*, ...)
- SUSTITUIR (*literal-3* o *identifier-5*, ...)

Si algún identificador tiene subíndice o se ha modificado la referencia, o es un identificador de función, el subíndice, el modificador de referencia o la función se evalúa sólo una vez como la primera operación en la ejecución de la sentencia INSPECT.

Para ver ejemplos de TALLYING y SUSTITUIR, consulte *Recuento y sustitución de elementos de datos (INSPECT)* en *COBOL for Linux en x86 Guía de programación*.

Ciclo de comparación

El ciclo de comparación consta de las acciones tal como se describe en este tema.

1. La primera comparación se compara con un número igual de posiciones de caracteres contiguos más a la izquierda en el elemento inspeccionado. El comparand coincide con los caracteres inspeccionados sólo si ambos son iguales, carácter por carácter.

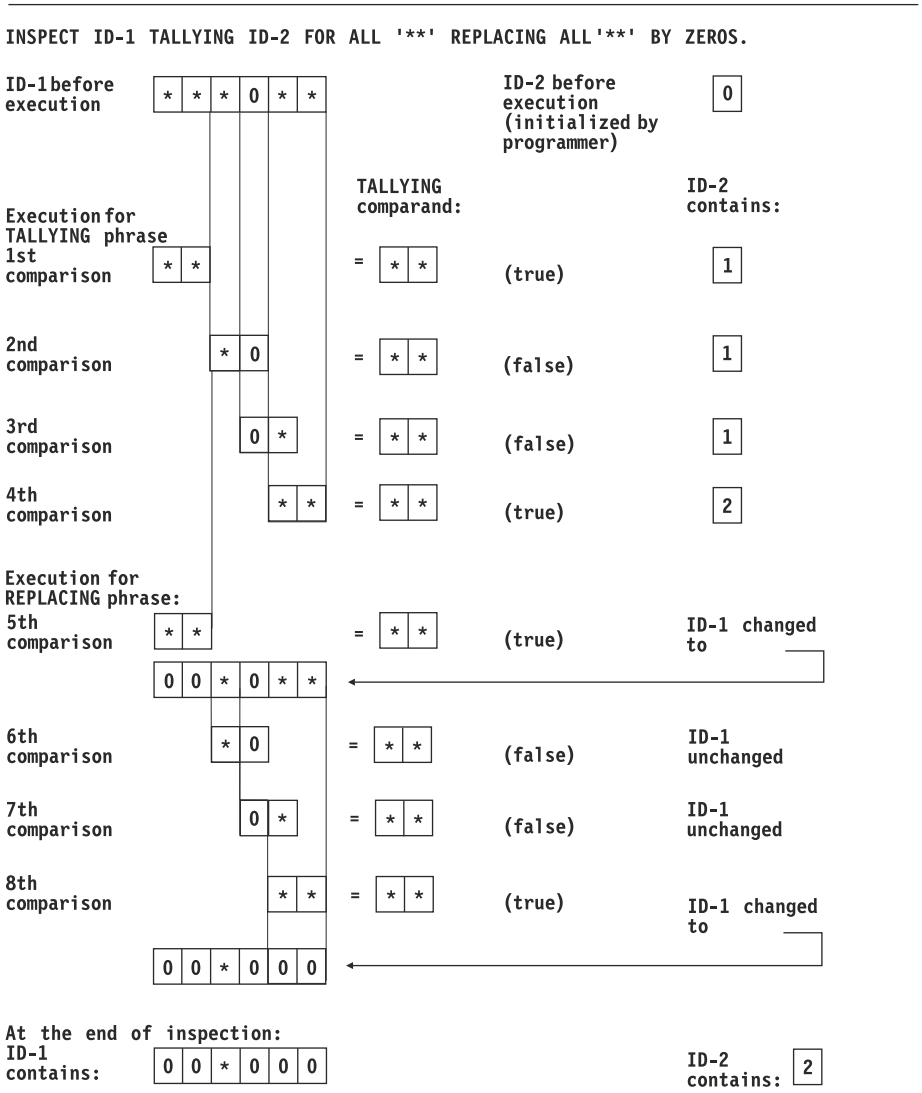
   Si se especifica la frase CHARACTERS, se utiliza una comparación implícita de un carácter. El carácter implícito siempre se considera que coincide con el carácter inspeccionado en el elemento inspeccionado.
2. Si no se produce ninguna coincidencia para la primera comparación y hay más comparaciones, la comparación se repite para cada comparación sucesiva y hasta que se encuentre una coincidencia o hasta que se haya actuado sobre todas las comparaciones.
3. En función de si se encuentra una coincidencia, se realizan estas acciones:
   - Si se encuentra una coincidencia, el recuento o la sustitución tiene lugar tal como se describe en las descripciones de frase TALLYING y SUSTITUIR.

     Si hay más posiciones de caracteres en el elemento inspeccionado, la primera posición de carácter después del carácter coincidente situado más a la derecha se considera ahora que está en la posición de carácter situado más a la izquierda. A continuación, se repite el proceso descrito en las acciones 1 y 2.
   - Si no se encuentra ninguna coincidencia y hay más posiciones de caracteres en el elemento inspeccionado, la primera posición de carácter que sigue al carácter inspeccionado más a la izquierda ahora se considera que está en la posición de carácter más a la izquierda. A continuación, se repite el proceso descrito en las acciones 1 y 2.
4. Las acciones del 1 al 3 se repiten hasta que la posición del carácter situado más a la derecha en el elemento inspeccionado ha coincidido o se ha considerado que está en la posición del carácter situado más a la izquierda.

Cuando se especifica la frase BEFORE o AFTER, se modifica el ciclo de comparación, tal como se describe en “Frases ANTES y DESPUÉS (todos los formatos)” en la página 338.

Ejemplo de la sentencia INSPECT

El tema muestra un ejemplo de los resultados de la sentencia INSPECT.

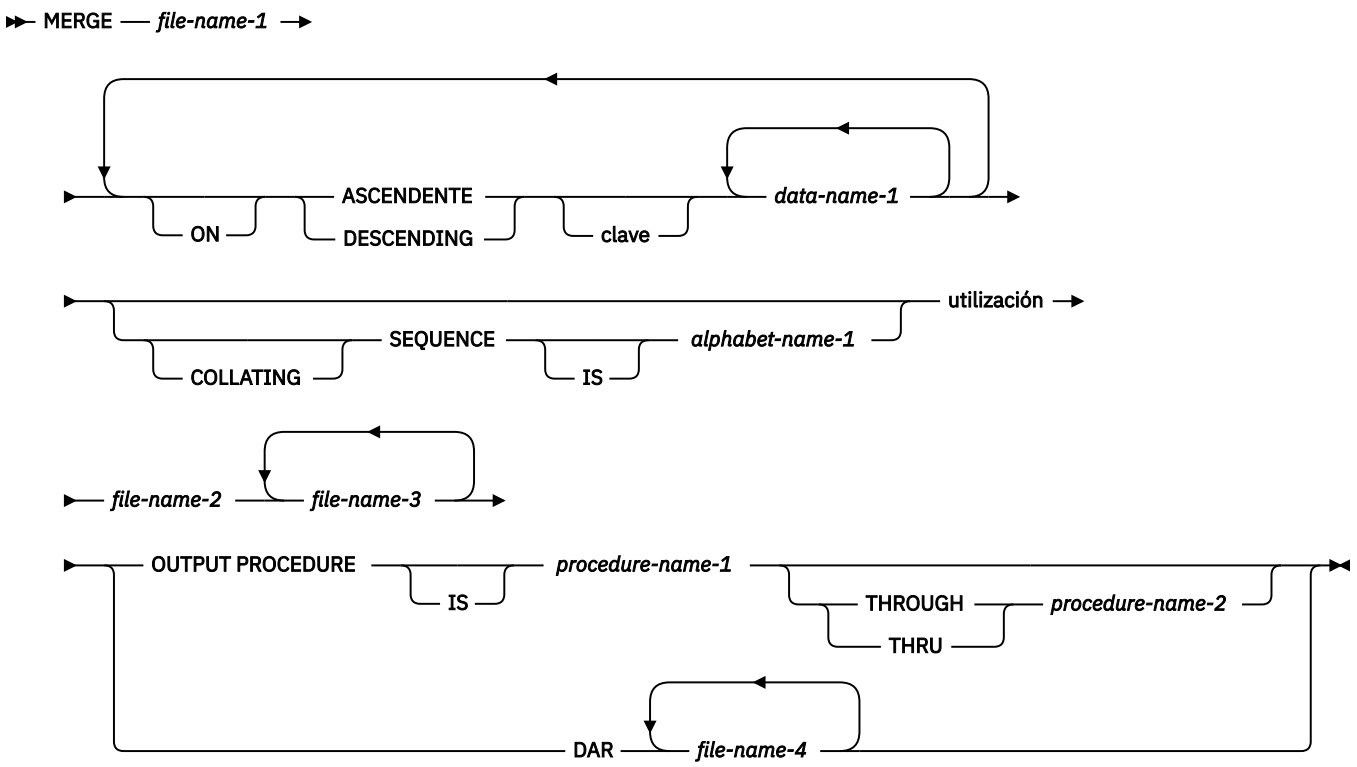


sentencia MERGE

La sentencia MERGE combina dos o más archivos secuenciados de forma idéntica (es decir, archivos que ya se han ordenado de acuerdo con un conjunto idéntico de claves ascendentes o descendentes) en una o más claves y hace que los registros estén disponibles en orden fusionado para un procedimiento de salida o archivo de salida.

Una sentencia MERGE puede aparecer en cualquier lugar de PROCEDURE DIVISION excepto en una sección declarativa.

Formato



file-name-1
El nombre proporcionado en la entrada SD que describe los registros que se van a fusionar.

No se puede repetir ningún nombre de archivo en la sentencia MERGE.

No se puede especificar ningún par de nombres de archivo en una sentencia MERGE en la misma cláusula SAME AREA, SAME SORT AREA o SAME SORT-MERGE AREA. Sin embargo, cualquier nombre de archivo de la sentencia MERGE puede especificarse en la misma cláusula SAME RECORD AREA.

Cuando se ejecuta la sentencia MERGE, todos los registros contenidos en *file-name-2*, *file-name-3*, ..., son aceptados por el programa de fusión y, a continuación, se fusionan de acuerdo con las claves especificadas.

Frase ASCENDING/DESCENDING KEY

Esta frase especifica que los registros deben procesarse en una secuencia ascendente o descendente (en función de la frase especificada), basándose en las claves de fusión especificadas.

data-name-1
Especifica un elemento de datos KEY en el que se basará la fusión. Cada nombre de datos de este tipo debe identificar un elemento de datos en un registro asociado con *file-name-1*. Los nombres de datos que siguen a la palabra KEY se listan de izquierda a derecha en la sentencia MERGE en orden de significación decreciente sin tener en cuenta cómo se dividen en frases KEY. El nombre de datos más a la izquierda es la clave principal, el siguiente nombre de datos es la siguiente clave más significativa, y así sucesivamente.

Se aplican las reglas siguientes:

- Un elemento de datos clave específico debe estar ubicado físicamente en la misma posición y tener el mismo formato de datos en cada archivo de entrada. Sin embargo, no es necesario que tenga el mismo nombre de datos.

- Si *file-name-1* tiene más de una descripción de registro, los elementos de datos KEY sólo se deben describir en una de las descripciones de registro.
- Si *file-name-1* contiene registros de longitud variable, todos los elementos de datos KEY deben estar contenidos en las primeras *n* posiciones de caracteres del registro, donde *n* es igual al tamaño mínimo de registros especificado para *file-name-1*.
- Los elementos de datos KEY no deben contener una cláusula OCCURS o estar subordinados a un elemento que contenga una cláusula OCCURS.
- Los elementos de datos KEY no pueden ser:
  - Ubicado de forma variable
  - Elementos de grupo que contienen elementos de datos de aparición variable
  - Campos de fecha con ventana
  - Numérico de categoría descrito con uso NATIONAL (tipo decimal nacional)
  - Categoría de coma flotante externa descrita con uso NACIONAL (coma flotante nacional)
  - Categoría DBCS
- Los elementos de datos KEY pueden calificarse.
- Los elementos de datos KEY pueden ser cualquiera de las siguientes categorías de datos:
  - Alfabético, alfanumérico, alfanumérico-editado
  - Numérico (excepto numérico con uso NACIONAL)
  - Numérico editado (con uso DISPLAY o NATIONAL)
  - Coma flotante interna o visualizar coma flotante
  - Nacional o editado a nivel nacional cuando la opción de compilador NCOLLSEQ (BINARY) está en vigor. La secuencia de clasificación binaria se aplica a las claves nacionales.

Si *file-name-4* hace referencia a un archivo indexado cuya clave de registro principal no es un *nombre-clave-registro*, la primera especificación de *data-name-1* debe estar asociada a una frase ASCENDING y el elemento de datos referenciado por ese *data-name-1* debe ocupar las mismas posiciones de caracteres en este registro que el elemento de datos asociado a la clave de registro principal para ese archivo.

Si *file-name-4* hace referencia a un archivo indexado cuya clave de registro principal es un *nombre-clave-registro*, debe especificar un *data-name-1* correspondiente para cada nombre de datos en la frase SOURCE del*nombre-clave-registro*. Cada *data-name-1* debe estar asociado con una frase ASCENDING, y cada *data-name-1* debe ocupar las mismas posiciones de caracteres que el nombre de datos correspondiente en la frase SOURCE.

Para obtener más información sobre *nombre-clave-registro*, consulte “Párrafo FILE-CONTROL” en la página 109.

La dirección de la operación de fusión depende de la especificación de las palabras clave ASCENDING o DESCENDING como se indica a continuación:

- Cuando se especifica ASCENDING, la secuencia va del valor de clave más bajo al valor de clave más alto.
- Cuando se especifica DESCENDING, la secuencia va del valor de clave más alto al valor de clave más bajo.

Si el elemento de datos KEY es alfabético, alfanumérico, alfanumérico editado o numérico editado, la secuencia de valores de clave depende de la secuencia de clasificación utilizada (consulte “Frase COLLATING SEQUENCE” en la página 345 más abajo).

Si el elemento de datos KEY se describe con el uso NATIONAL, la secuencia de los valores KEY se basa en los valores binarios de los caracteres nacionales.

Si la KEY es un elemento de coma flotante externo con el uso DISPLAY, la clave se trata como categoría alfanumérica. La secuencia en la que se fusionan los registros depende del orden de clasificación utilizado.

Si KEY es un elemento de coma flotante externo con el uso NATIONAL, la clave se trata como de categoría nacional.

Si la KEY es un elemento de coma flotante interno, la secuencia de valores de clave es de orden numérico.

Cuando no se especifica la frase COLLATING SEQUENCE, las comparaciones de claves se realizan de acuerdo con las reglas para la comparación de operandos en una condición de relación. Para obtener detalles, consulte "Condiciones generales de relación" en la página 255.

Cuando se especifica la frase COLLATING SEQUENCE, la secuencia de clasificación indicada se utiliza para elementos de datos clave de categorías alfabéticas, alfanuméricas, editadas alfanuméricas, de coma flotante externa y editadas numéricas. Para todos los demás elementos de datos clave, las comparaciones se realizan de acuerdo con las reglas para la comparación de operandos en una condición de relación.

Frase COLLATING SEQUENCE

Esta frase especifica el orden de clasificación que se utilizará en las comparaciones alfanuméricas para los elementos de datos KEY en esta operación de fusión.

La frase COLLATING SEQUENCE no tiene ningún efecto para las claves que no son alfabéticas o alfanuméricas.

La frase COLLATING SEQUENCE sólo es válida cuando está en vigor una página de códigos ASCII de un solo byte.

alphabet-name-1
Debe especificarse en la cláusula ALPHABET del párrafo SPECIAL-NAMES. Se puede especificar cualquiera de las frases de cláusula de nombre alfabético, con los resultados siguientes:

STANDARD-1
La secuencia de clasificación se basa en el orden de los valores hexadecimales del carácter.

STANDARD-2
La secuencia de clasificación se basa en el orden de los valores hexadecimales del carácter.

NATIVA
Se selecciona el orden de clasificación indicado por el entorno local de tiempo de ejecución.

EBCDIC
La secuencia de clasificación EBCDIC se utiliza para todas las comparaciones alfanuméricas. (El orden de clasificación EBCDIC se muestra en la "Secuencia de clasificación EBCDIC" en la página 555.)

literal
La secuencia de clasificación establecida por la especificación de literales en la cláusula ALPHABET-NAME se utiliza para todas las comparaciones alfanuméricas.

Cuando se omite la frase COLLATING SEQUENCE, la cláusula PROGRAM COLLATING SEQUENCE (si se especifica) en el párrafo OBJECT-COMPUTER identifica la secuencia de clasificación que se va a utilizar. Cuando se omiten la frase COLLATING SEQUENCE de la sentencia MERGE y la cláusula PROGRAM COLLATING SEQUENCE del párrafo OBJECT-COMPUTER, la opción de compilador COLLSEQ indica el orden de clasificación utilizado. Si se especifica COLLSEQ (EBCDIC), se utiliza la secuencia de clasificación EBCDIC. Si se especifica COLLSEQ (LOCALE), se utiliza el orden de clasificación indicado por el entorno local. Para obtener más información sobre los entornos locales, consulte Apéndice F, "Consideraciones sobre el entorno local", en la página 585.

frase USING

file-name-2 , file-name-3 , ...
Especifica los archivos de entrada.

Durante la operación MERGE, todos los registros de *file-name-2*, *file-name-3*, ... (es decir, los archivos de entrada) se transfieren a *file-name-1*. En el momento en que se ejecuta la sentencia MERGE, estos archivos no deben estar abiertos. Los archivos de entrada se abren, leen y cierran automáticamente. Si se especifican procedimientos DECLARATIVE para estos archivos para operaciones de entrada, los declarativos se activarán en busca de errores si se producen errores.

Todos los archivos de entrada deben especificar la modalidad de acceso secuencial o dinámico y deben describirse en las entradas FD de DATA DIVISION.

Si *file-name-1* contiene registros de longitud variable, el tamaño de los registros contenidos en los archivos de entrada (*file-name-2*, *file-name-3*, ...) no debe ser menor que el registro más pequeño ni mayor que el registro más grande descrito para *file-name-1*. Si *file-name-1* contiene registros de longitud fija, el tamaño de los registros contenidos en los archivos de entrada no debe ser mayor que el registro más grande descrito para *file-name-1*. Para obtener más información, consulte *Clasificación y fusión de archivos* en *COBOL for Linux en x86 Guía de programación*.

Frase CEDER

file-name-4 , ...
Especifica los archivos de salida.

Cuando se especifica la frase CEDER, todos los registros fusionados en *file-name-1* se transfieren automáticamente a los archivos de salida (*file-name-4*, ...).

Todos los archivos de salida deben especificar la modalidad de acceso secuencial o dinámico y deben describirse en las entradas FD de DATA DIVISION.

Si los archivos de salida (*file-name-4*, ...) contienen registros de longitud variable, el tamaño de los registros contenidos en *file-name-1* no debe ser menor que el registro más pequeño ni mayor que el registro más grande descrito para los archivos de salida. Si los archivos de salida contienen registros de longitud fija, el tamaño de los registros contenidos en *file-name-1* no debe ser mayor que el registro más grande descrito para los archivos de salida. Para obtener más información, consulte *Clasificación y fusión de archivos* en *COBOL for Linux en x86 Guía de programación*.

En el momento en que se ejecuta la sentencia MERGE, los archivos de salida (*file-name-4*, ...) no debe estar abierto. Los archivos de salida se abren automáticamente, se graban eny se cierran. Si se especifican procedimientos DECLARATIVE para estos archivos para operaciones de salida, los declarativos se manejarán en busca de errores si se producen errores.

Frase OUTPUT PROCEDURE

Esta frase especifica el nombre de un procedimiento que debe seleccionar o modificar registros de salida de la operación de fusión.

procedure-name-1
Especifica la primera (o única) sección o párrafo en el PROCEDIMIENTO DE SALIDA.

procedure-name-2
Identifica la última sección o párrafo del PROCEDIMIENTO DE SALIDA.

El OUTPUT PROCEDURE puede consistir en cualquier procedimiento necesario para seleccionar, modificar o copiar los registros que la sentencia RETURN pone a disposición de uno en uno en orden fusionado desde el archivo al que hace referencia *file-name-1*. El rango incluye todas las sentencias que se ejecutan como resultado de una transferencia de control por parte de las sentencias CALL, EXIT, GO TO, PERFORM y XML PARSE en el rango del procedimiento de salida. El rango también incluye todas las sentencias en procedimientos declarativos que se ejecutan como resultado de la ejecución de sentencias en el rango del procedimiento de salida. El rango del procedimiento de salida no debe provocar la ejecución de ninguna sentencia SORT MERGE, RELEASE o .

Si se especifica un procedimiento de salida, el control lo pasa después de que la sentencia MERGE haya secuenciado el archivo al que hace referencia *file-name-1* . El compilador inserta un mecanismo de retorno al final de la última sentencia en el procedimiento de salida y cuando el control pasa la última sentencia en el procedimiento de salida, el mecanismo de retorno proporciona la terminación de la fusión y luego pasa el control a la siguiente sentencia ejecutable después de la sentencia MERGE. Antes de entrar en el procedimiento de salida, el procedimiento de fusión alcanza un punto en el que puede seleccionar el siguiente registro en orden fusionado cuando se le solicite. Las sentencias RETURN en el procedimiento de salida son las peticiones para el siguiente registro.

La frase OUTPUT PROCEDURE es similar a una sentencia PERFORM básica. Por ejemplo, si nombra un procedimiento en un OUTPUT PROCEDURE, dicho procedimiento se ejecuta durante la operación de fusión como si se nombrara en una sentencia PERFORM. Al igual que con la sentencia PERFORM, la ejecución del procedimiento se termina después de que la última sentencia complete la ejecución. La última sentencia de un OUTPUT PROCEDURE puede ser la sentencia EXIT (consulte "sentencia EXIT" en la página 323).

Registros especiales MERGE

El tema describe registros especiales de la sentencia MERGE.

Registro especial SORT-CONTROL
Identifique el archivo de control de ordenación (a través del cual puede especificar opciones adicionales para la función de clasificación/fusión) con el registro especial SORT-CONTROL.

Si utiliza un archivo de control de clasificación para especificar sentencias de control, los valores especificados en el archivo de control de clasificación tienen prioridad sobre los de los otros registros especiales SORT.

Para obtener información, consulte "CONTROL DE SORT" en la página 24.

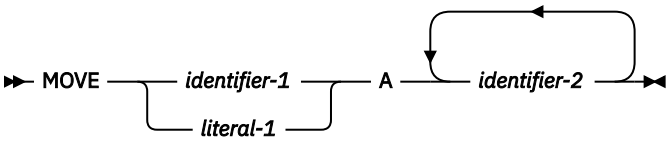
Registro especial SORT-MESSAGE
Para obtener información, consulte "ORDENAR-MENSAJE" en la página 25. El registro especial SORT-MESSAGE es equivalente a una palabra clave de sentencia de control de opción en el archivo de control de clasificación.

Registro especial SORT-RETURN
Para obtener información, consulte "ORDENAR-RETORNO" en la página 25.

sentencia MOVE

La sentencia MOVE transfiere datos de un área de almacenamiento a una o más áreas.

Formato 1: sentencia MOVE



Formato 2: sentencia MOVE con frase CORRESPONDIENTE

CORR es una abreviatura para, y es equivalente a, CORRESPONDIENTE.

identifier-1 , literal-1
El área de envío.

identifier-2
Las áreas receptoras. *identifier-2* no debe hacer referencia a una función intrínseca.

Cuando se especifica el formato 1:

- Cuando uno de *identifier-1* o *identifier-2* hace referencia a un elemento de grupo nacional y el otro operando hace referencia a un elemento de grupo alfanumérico, el grupo nacional se procesa como un elemento de grupo; en todos los demás casos, el elemento de grupo nacional se procesa como elemento de datos elementales de la categoría nacional.
- Los datos del área de envío se mueven al elemento de datos al que hace referencia cada *identifier-2* en el orden en el que se especifican los elementos de datos *identifier-2* en la sentencia MOVE. Consulte "Movimientos elementales" en la página 349 y "Movimientos de grupo" en la página 354 a continuación.

Cuando se especifica el formato 2:

- Ambos identificadores deben ser elementos de grupo.
- Un elemento de grupo nacional se procesa como un elemento de grupo (y no como un elemento de datos elemental de categoría nacional).
- Los elementos seleccionados en *identifier-1* se mueven a *identifier-2* de acuerdo con las reglas del "Frase CORRESPONDIENTE" en la página 279. Los resultados son los mismos que si se hiciera referencia a cada par de identificadores CORRESPONDIENTE en una sentencia MOVE independiente.

Los elementos de datos descritos con los siguientes tipos de uso no se pueden especificar en una sentencia MOVE:

- ÍNDICE
- PUNTERO
- PUNTERO DE FUNCIÓN
- PROCEDIMIENTO-PUNTERO

Un elemento de datos definido con un uso de INDEX, POINTER, FUNCTION-POINTER, o PROCEDURE-POINTER puede formar parte de un elemento de grupo alfanumérico al que se hace referencia en una sentencia MOVE CORRESPONDIENTE; sin embargo, no se produce ningún movimiento de datos desde esos elementos de datos.

La evaluación de la longitud del área de envío o recepción puede verse afectada por la frase DEPENDING ON de la cláusula OCCURS (consulte "cláusula OCCURS" en la página 184).

Si el campo de envío (*identifier-1*) es de referencia modificada o con subíndice, o es un identificador de funciónalfanumérico, numérico, entero o nacional , el modificador de referencia, subíndice, o la función se evalúa sólo una vez, inmediatamente antes de que los datos se muevan al primero de los operandos receptores.

Cualquier evaluación de longitud, subscripción o modificación de referencia asociada con un campo receptor (*identifier-2*) se evalúa inmediatamente antes de que los datos se muevan a ese campo receptor.

Por ejemplo, el resultado de la sentencia:

```
MOVE A(B) TO B, C(B).
```

es equivalente a:

```
MOVE A(B) TO TEMP.
MOVE TEMP TO B.
MOVE TEMP TO C(B).
```

donde TEMP se define como un elemento de resultado intermedio. El subíndice B ha cambiado de valor entre el momento en que tuvo lugar el primer movimiento y el momento en que se ejecutó el último movimiento a `C(B)` .

Para obtener más información sobre los resultados intermedios, consulte *Apéndice A. Resultados intermedios y precisión aritmética* en *COBOL for Linux en x86 Guía de programación*.

Después de la ejecución de una sentencia MOVE, los campos de envío contienen los mismos datos que antes de la ejecución.

Nota de uso: El solapamiento de operandos en una sentencia MOVE puede provocar resultados imprevisibles.

Movimientos elementales

Un movimiento elemental es aquel en el que el elemento de recepción es un elemento de datos elemental y el elemento de envío es un elemento de datos elemental o un literal.

Los operandos válidos pertenecen a una de las categorías siguientes:

- **Alfabético**: incluye elementos de datos de categoría alfabética y la constante figurativa SPACE
- **Alfanumérico**: incluye los elementos siguientes:
  - Elementos de datos de categoría alfanuméricos
  - Funciones alfanuméricas
  - Literales alfanuméricos
  - La constante figurativa ALL *literal-alfanumérico* y todas las demás constantes figurativas (excepto NULL) cuando se utilizan en un contexto que requiere un elemento de envío alfanumérico
- **Alfanumérico editado**: incluye elementos de datos de la categoría alfanuméricos editados
- **Booleano**: incluye elementos de datos booleanos y literales booleanos.
- **Fecha-Hora**: incluye los elementos de datos de fecha, hora e indicación de fecha y hora de la clase fecha-hora. Los elementos de datos de fecha y hora se definen como USAGE DISPLAY o PACKED-DECIMAL.
- **DBCS**: incluye elementos de datos de la categoría DBCS, literales DBCS y la constante figurativa ALL DBCS-literal.
- **coma flotante externa**: incluye elementos de datos de la categoría coma flotante externa (descrita con USAGE DISPLAY o USAGE NATIONAL) y literales de coma flotante.
- **Punto flotante interno**: incluye elementos de datos de punto flotante interno de categoría (definido como USAGE COMP-1 o USAGE COMP-2)
- **Nacional**: incluye los elementos siguientes:
  - Partidas del grupo nacional (tratadas como partidas elementales de la categoría nacional)
  - Elementos de datos de la categoría nacional
  - Literales nacionales
  - Funciones nacionales
  - Constantes figurativas ZERO, SPACE, QUOTE y ALL *literal-nacional* cuando se utilizan en un contexto que requiere un elemento de envío nacional
- **Nacional-editado**: incluye elementos de datos de la categoría nacional-editado
- **Numérico**: incluye los elementos siguientes:
  - Elementos de datos de categoría numéricos
  - Literales numéricos
  - La constante figurativa CERO (cuando CERO se mueve a un elemento numérico o editado numérica-editado).
- **Numérico editado**: incluye elementos de datos de la categoría numérica-editada.

Reglas de movimiento elementales

Cualquier conversión necesaria de datos de una forma de representación interna a otra tiene lugar durante el movimiento, junto con cualquier edición especificada en, o desedición implícita por, el elemento receptor. La página de códigos utilizada para la conversión a o desde caracteres alfanuméricos es la página de códigos aplicable al elemento de datos específico en tiempo de ejecución.

Las reglas siguientes describen la ejecución de movimientos elementales válidos. Cuando el campo de recepción es:

Alfabético:

- La alineación y cualquier relleno o truncamiento de espacio necesario se producen tal como se describe en “Reglas de alineación” en la página 151.
- Si el tamaño del elemento de envío es mayor que el tamaño del elemento de recepción, los caracteres en exceso de la derecha se truncan después de rellenar el elemento de recepción.

Alfanumérico o **alfanumérico editado**:

- Si el elemento de envío es un elemento entero decimal nacional, los datos de envío se convierten a DISPLAY de uso y se tratan como si se trasladaran a un elemento de datos temporal de categoría alfanumérica con el mismo número de posiciones de caracteres que el elemento de envío. El elemento de datos alfanuméricos resultante se trata como el elemento de envío.
- La alineación y cualquier relleno o truncamiento de espacio necesario tienen lugar, tal como se describe en “Reglas de alineación” en la página 151.
- Si el tamaño del elemento de envío es mayor que el tamaño del elemento de recepción, los caracteres en exceso de la derecha se truncan después de rellenar el elemento de recepción.
- Si el elemento de envío inicial tiene un signo operativo, se utiliza el valor sin signo. Si el signo operativo ocupa un carácter separado, ese carácter no se mueve y el tamaño del elemento emisor se considera un carácter menos que el tamaño real.
- Si el elemento de envío es booleano, los datos se mueven como si el elemento de envío se describiera como un elemento alfanumérico de longitud 1.
- Si el elemento de envío es de fecha y hora, el elemento de fecha y hora se trata como un elemento alfanumérico y se mueve al receptor siguiendo las reglas de un movimiento alfanumérico a alfanumérico. Si el elemento de fecha y hora de envío tiene un USAGE de PACKED-DECIMAL, primero se convierte a un USAGE de DISPLAY.

Booleano:

- Para un elemento de recepción booleano, sólo se mueve el primer byte del elemento de envío.
- Si el elemento de envío es alfanumérico, se mueve el primer carácter del elemento de envío. Los caracteres "0" y "1" son equivalentes a los valores booleanos B "0" y B "1", respectivamente.
- Si el elemento de envío es CERO, se trata como el literal booleano B "0".

DBCS:

- Si los elementos de envío y recepción no tienen el mismo tamaño, los datos de envío se truncan a la derecha o se rellenan con espacios DBCS a la derecha. Si el relleno necesario no está en un múltiplo coherente con caracteres de doble byte, se utilizan caracteres de un solo byte (por ejemplo, un elemento de datos DBCS movido a un elemento de grupo alfanumérico).

Fecha y hora:

- Si el elemento de envío es de fecha y hora, el formato del elemento de fecha y hora de envío se convierte primero al formato del receptor y, a continuación, se mueve. Si el elemento de envío es una indicación de fecha y hora, y el elemento de recepción es un elemento de fecha u hora, sólo la parte de fecha u hora del elemento de indicación de fecha y hora se mueve al elemento de recepción. Si el elemento de envío es un elemento de fecha u hora y el elemento de recepción es una indicación de fecha y hora, sólo se sustituye la parte de fecha u hora de la indicación de fecha y hora.

- Si el elemento de envío es numérico, cada uno de los especificadores de conversión numérica de elementos de recepción se sustituye por los dígitos del elemento de envío, empezando por el especificador de conversión situado más a la derecha y por el dígito situado más a la derecha de dicho especificador de conversión. Todos los especificadores de conversión alfanuméricos toman valores predeterminados.
- Si el elemento de envío es editado numérica-editado, el elemento editado numérica-editado se desedita. A continuación, el valor numérico resultante se mueve al elemento de fecha-hora.
- Si el elemento de envío es alfanumérico o alfanumérico editado, el elemento de fecha y hora de recepción se trata como un elemento alfanumérico, y el movimiento tiene lugar de acuerdo con las reglas de un movimiento alfanumérico a alfanumérico.

coma flotante externa:

- Para un elemento de envío de coma flotante, el valor de coma flotante se convierte al uso del elemento de coma flotante externo receptor (si es diferente de la representación del elemento de envío).
- Para otros elementos de envío, el valor numérico se trata como si ese valor se convirtiera a coma flotante interno y, a continuación, se convirtiera al uso del elemento de coma flotante externo receptor.

Punto flotante interno:

- Cuando la categoría del operando de envío no es de coma flotante interna, el valor numérico del elemento de envío se convierte al formato de coma flotante interno.

National o **national-editado**:

- Si la representación del elemento de envío no es caracteres nacionales, los datos de envío se convierten a caracteres nacionales y se tratan como si se trasladaran a un elemento de datos temporal de categoría nacional de una longitud que no provoque truncamiento o relleno. El elemento de datos nacional de categoría resultante se trata como el elemento de datos de envío.
- Si la representación del elemento emisor es de caracteres nacionales, los datos de envío se utilizan sin conversión.
- La alineación y cualquier relleno o truncamiento de espacio necesario tienen lugar tal como se describe en “Reglas de alineación” en la página 151. El programador es responsable de asegurarse de que varias unidades de codificación que forman un carácter gráfico no se dividen por truncamiento.
- Si el elemento emisor tiene un signo operativo, se utiliza el valor sin signo. Si el signo operativo ocupa un carácter separado, ese carácter no se mueve y el tamaño del elemento emisor se considera un carácter menos que el tamaño real.

Numérico o **numérico-editado**:

- Excepto cuando se sustituyen los ceros debido a los requisitos de edición, la alineación por coma decimal y cualquier relleno de cero necesario tienen lugar, tal como se describe en “Reglas de alineación” en la página 151.
- Si el elemento de recepción está firmado, el signo del elemento de envío se coloca en el elemento de recepción, con cualquier conversión de signo necesaria. Si el elemento emisor no está firmado, se genera un signo operativo positivo para el elemento receptor.
- Si el artículo de recepción no está firmado, no se genera ningún signo operativo para el artículo de recepción y se utiliza el valor absoluto del artículo de envío en el traslado.
- Cuando la categoría del artículo de envío es alfanumérica, alfanumérica, nacional o nacional, los datos se mueven como si el artículo de envío se describiera como un entero sin signo.
- Cuando el elemento emisor es de coma flotante, los datos se convierten primero a una representación decimal binaria o interna y, a continuación, se mueven.
- Cuando el elemento de recepción es numérico editado, la edición tiene lugar tal como se define en la serie de caracteres de imagen o en la cláusula BLANK WHEN ZERO asociada con el elemento de recepción.
- Cuando el elemento de envío es de edición numérica, el compilador anula la edición de los datos de envío para establecer el valor no editado del elemento de edición numérica (este valor puede estar

firmado). El valor numérico no editado se utiliza al mover al elemento de datos numérico o numérico de recepción.

Notas de uso:

1. Si el artículo de recepción es de categoría alfanumérica, editada alfanumérica, editada numérica, nacional o editada nacional y el campo de envío es numérico, se considera que cualquier posición de dígito descrita con el símbolo de imagen P en el artículo de envío tiene el valor cero. Cada P se cuenta en el tamaño del elemento de envío.
2. Si el elemento receptor es numérico y el campo emisor es un literal alfanumérico, un literal nacional o un literal ALL, todos los caracteres del literal deben ser numéricos.

Movimientos elementales válidos y no válidos

La tabla muestra movimientos elementales válidos y no válidos para cada categoría.

En la tabla:

- YES = Mover es válido.
- NO = Mover no es válido.
- Las cabeceras de columna indican categorías de artículo de recepción; las cabeceras de fila indican categorías de artículo de envío.

Tabla 50. ***movimientos elementales válidos y no válidos***

Categoría de artículo de envío	**Categoría de artículo de recepción**												
	Alfabético	**Alfanumérico**	**Alfanumérico editado**	**Booleano**	**fecha**	**hora**	**Indicación de fecha y hora-**	**Numérico**	**Numérico-editado**	**Coma flotante externa**	**Coma flotante interna**	**DBCS 1**	**Nacional, nacional-editado**
Alfabético y ESPACIO	Sí	Sí	Sí	No	No	No	No	No	No	No	No	No	Sí
Alfanumérico[2]	Sí	Sí	Sí	Sí[10]	Sí	Sí	Sí	Sí[3]	Sí[3]	Sí[8]	Sí[8]	No	Sí
Alfanumérico-editado	Sí	Sí	Sí	No	Sí	Sí	Sí	No	No	No	No	No	Sí
Booleano[11]	No	Sí	Sí	Sí	No	No	No	No	No	No	No	No	No
fecha	No	Sí	Sí	No	Sí	No	Sí	Sí	Sí	No	No	No	No
hora	No	Sí	Sí	No	No	Sí	Sí	Sí	Sí	No	No	No	No
Indicación de fecha y hora	No	Sí	Sí	No	Sí	Sí	Sí	Sí	Sí	No	No	No	No
Entero numérico[4]	No	Sí	Sí	No	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No	Sí
CERO[4]	No	Sí	Sí	Sí	No	No	No	Sí	Sí	Sí	Sí	No	Sí
No entero numérico[5]	No	No	No	No	No	No	No	Sí	Sí	Sí	Sí	No	No
Numérico-editado	No	Sí	Sí	No	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No	Sí
Coma flotante[6]	No	No	No	No	No	No	No	Sí	Sí	Sí	Sí	No	No
DBCS[7]	No	No	No	No	No	No	No	No	No	No	No	Sí	Sí
Nacional[9]	No	No	No	No	No	No	No	Sí	Sí	Sí	Sí	No	Sí
Nacional-editado	No	No	No	No	No	No	No	No	No	No	No	No	Sí

Tabla 50. ***movimientos elementales válidos y no válidos*** *(continuación)*

Categoría de artículo de envío	Categoría de artículo de recepción												
	Alfa-bético	**Alfanumérico**	**Alfanumérico editado**	**Booleano**	**fecha**	**hora**	**Indicación de fecha y hora-**	**Numérico**	**Numérico-editado**	**Coma flotante externa**	**Coma flotante interna**	**DBCS 1**	**Nacional, nacional-editado**

1. Incluye elementos de datos DBCS.
2. Incluye literales alfanuméricos.
3. Las constantes figurativas y los literales alfanuméricos sólo deben constar de caracteres numéricos y se tratarán como campos enteros numéricos.
4. Incluye literales numéricos enteros.
5. Incluye literales numéricos no enteros.
6. Incluye literales de coma flotante, elementos de datos de coma flotante externos (USAGE DISPLAY o USAGE NATIONAL) y elementos de datos de coma flotante internos (USAGE COMP-1 o USAGE COMP-2).
7. Incluye elementos de datos DBCS, literales DBCS y la constante figurativa SPACE.
8. Las constantes figurativas y los literales alfanuméricos sólo deben constar de caracteres numéricos y se tratarán como campos enteros numéricos. El literal ALL no se puede utilizar como elemento de envío.
9. Incluye elementos de datos nacionales, literales nacionales, funciones nacionales y constantes figurativas ZERO, SPACE, QUOTE y TODOS los literales nacionales.
10. Se mueve el primer carácter del elemento de envío, independientemente de su valor
11. Incluye literales booleanos

Movimientos que implican campos de fecha

Si el elemento de envío se especifica como un campo de fecha de último año, todos los campos de recepción también deben ser campos de fecha de último año con el mismo formato de fecha que el elemento de envío. Si se especifica un campo de fecha de último año como elemento de recepción, el elemento de envío debe ser un campo de fecha no de fecha o un campo de fecha de último año con el mismo formato de fecha que el elemento de recepción. En ambos casos, el movimiento se realiza como si todos los elementos no fueran fechas.

Tabla 51 en la página 353 describe el comportamiento de los movimientos que implican campos de fecha no de último año. Si el elemento de envío es un campo de fecha, el elemento de recepción debe ser un campo de fecha compatible. Si los elementos de envío y recepción son ambos campos de fecha, deben ser compatibles; es decir, deben tener el mismo formato de fecha, excepto para la parte del año, que puede estar en ventana o expandida.

Esta tabla utiliza los términos siguientes para describir los movimientos:

Normal
El movimiento se realiza sin ningún comportamiento sensible a la fecha, como si los elementos de envío y recepción no fueran fechas.

Expandido
El elemento de envío de campo de fecha con ventana se trata como si se convirtiera por primera vez en un formulario expandido, tal como se describe en "Semántica de campos de fecha con ventanas" en la página 172.

No válido
El movimiento no está permitido.

Tabla 51. ***Mueve los campos de fecha relacionados***

	Elemento de recepción no de fecha	Elemento de recepción de campo de fecha con ventana	Elemento de recepción de campo de fecha expandida
Elemento de envío sin fecha	Normal	Normal	Normal

*Tabla 51. **Mueve los campos de fecha relacionados** (continuación)*			
Artículo de envío de campo de fecha con ventana	No válido	Normal	Expandido
Artículo de envío de campo de fecha expandida	No válido	Normal[1]	Normal

1. Un movimiento de un campo de fecha expandido a un campo de fecha con ventana es, en efecto, un movimiento "con ventana", porque trunca el componente de siglo del campo de fecha expandido. Si el movimiento es alfanumérico, trata el campo de fecha de ventana de recepción como si su descripción de datos hubiera especificado JUSTIFIC RIGHT. Esto es cierto incluso si el campo de fecha de ventana de recepción es un elemento de grupo, para el que no se puede especificar la cláusula JUSTIFIC.

Movimientos que implican áreas de registro de archivo

La ejecución satisfactoria de una sentencia OPEN para un archivo determinado hace que el área de registro para ese archivo esté disponible. Puede mover datos a o desde las entradas de descripción de registro asociadas con un archivo sólo cuando el archivo está en estado abierto.

La ejecución de una sentencia CLOSE implícita o explícita elimina un archivo del estado abierto y hace que el área de registro no esté disponible.

Movimientos de grupo

Un movimiento de grupo puede ser cualquier movimiento en el que un elemento de grupo alfanumérico sea un elemento de envío o un elemento de recepción, o ambos.

Los movimientos de grupo son:

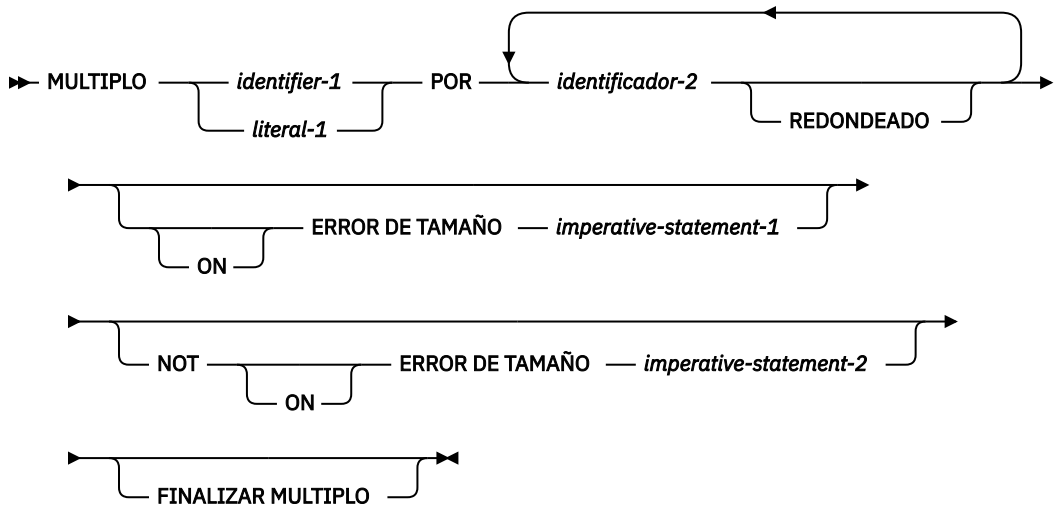
- Un movimiento a un elemento de grupo alfanumérico desde uno de los elementos siguientes:
  - cualquier elemento de datos elemental que sea válido como elemento de envío en la sentencia MOVE
  - un elemento de grupo nacional
  - un literal
  - una constante figurativa
- Un movimiento de un elemento de grupo alfanumérico a los elementos siguientes:
  - cualquier elemento de datos elemental que sea válido como elemento receptor en la sentencia MOVE
  - un elemento de grupo nacional
  - un elemento de grupo alfanumérico

Un movimiento de grupo se trata como si fuera un movimiento elemental alfanumérico a alfanumérico, excepto que no hay conversión de datos de una forma de representación interna a otra. En un movimiento de grupo, el área de recepción se rellena sin tener en cuenta los elementos elementales individuales contenidos en el área de envío o el área de recepción, excepto tal como se indica en la cláusula OCCURS. (Consulte “cláusula OCCURS” en la página 184.)

sentencia MULTIPLY

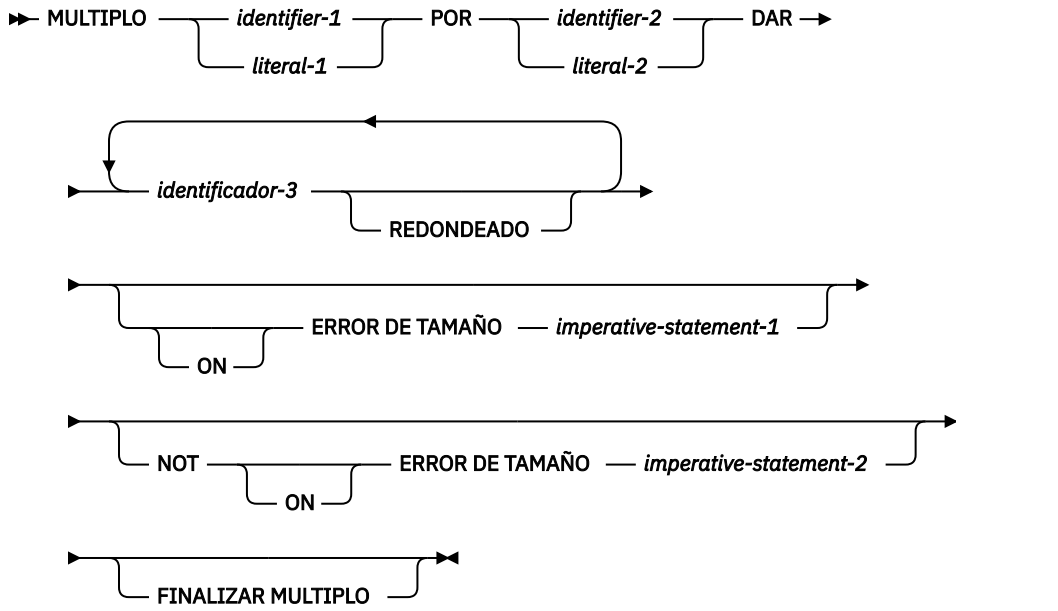
La sentencia MULTIPLY multiplica los elementos numéricos y establece los valores de los elementos de datos iguales a los resultados.

Formato 1: sentencia MULTIPLY



En el formato 1, el valor de *identifier-1* o *literal-1* se multiplica por el valor de *identificador-2*; A continuación, el producto se coloca en *identifier-2*. Para cada aparición sucesiva de *identifier-2*, la multiplicación tiene lugar en el orden de izquierda a derecha en el que se especifica *identifier-2* .

Formato 2: sentencia MULTIPLY con frase CEDER



En el formato 2, el valor de *identifier-1* o *literal-1* se multiplica por el valor de *identifier-2* o *literal-2*. A continuación, el producto se almacena en los elementos de datos a los que hace referencia *identifier-3*.

Para todos los formatos:

identifier-1* , *identifier-2
Debe nombrar un elemento numérico elemental. *identifier-1* y *identifier-2* no pueden ser campos de fecha.

literal-1* , *literal-2
Debe ser un literal numérico.

Para format-2:

identifier-3
Debe nombrar un elemento numérico elemental o numérico editado.

identifier-3, el identificador de frase CEDER, es el único identificador de la sentencia MULTIPLY que puede ser un campo de fecha.

Si *identifier-3* nombra un campo de fecha, Consulte "Almacenamiento de resultados aritméticos que implican campos de fecha" en la página 249 para obtener detalles sobre cómo se almacena el producto en *identifier-3*.

Los elementos de datos de coma flotante y literales se pueden utilizar en cualquier lugar donde se pueda especificar un elemento de datos numérico o literal.

Cuando la opción de compilador ARITH (COMPAT) está en vigor, el compuesto de operandos puede contener un máximo de 30 dígitos. Cuando la opción de compilador ARITH (EXTEND) está en vigor, el compuesto de operandos puede contener un máximo de 31 dígitos. Para obtener más información, consulte "Operandos de sentencia aritmética" en la página 282 y los detalles sobre los resultados intermedios aritméticos, *Apéndice A. Resultados intermedios y precisión aritmética* en *COBOL for Linux en x86 Guía de programación*.

Frase REDONDEADA

Para los formatos 1 y 2, consulte "Frase REDONDEADA" en la página 281.

Frases de ERROR DE TAMAÑO

Para los formatos 1 y 2, consulte "Frases de ERROR DE TAMAÑO" en la página 281.

frase END-MULTIPLY

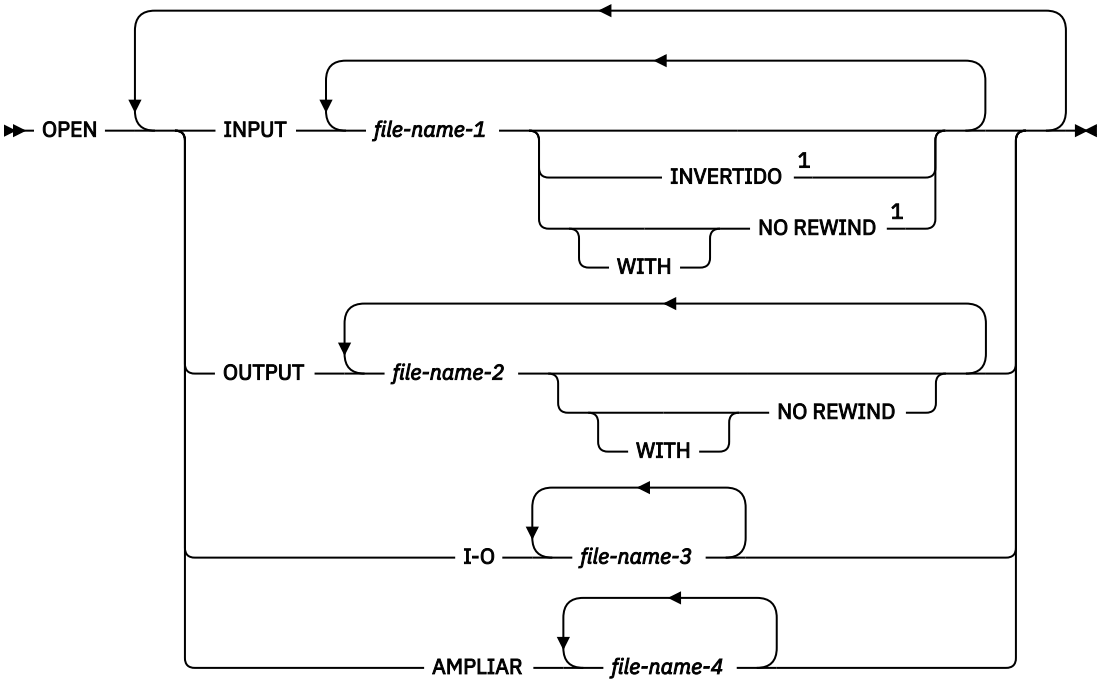
Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia MULTIPLY. END-MULTIPLY permite que una sentencia MULTIPLY condicional se anide en otra sentencia condicional. END-MULTIPLY también se puede utilizar con una sentencia MULTIPLY imperativa.

Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

sentencia OPEN

La sentencia OPEN inicia el proceso de archivos. También comprueba o escribe etiquetas, o ambas.

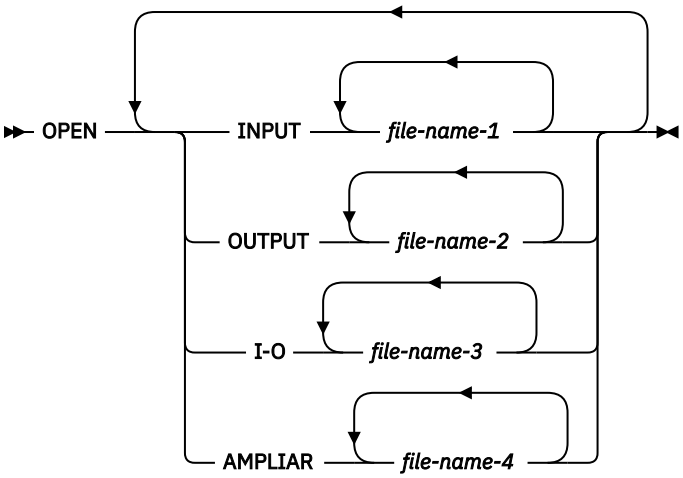
Formato 1: sentencia OPEN para archivos secuenciales



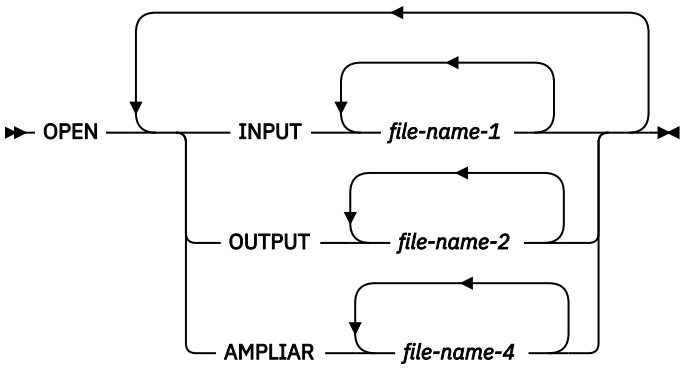
Notas:

[1] Las frases REVERSED y WITH NO REWIND se comprueban con la sintaxis, pero no tienen ningún efecto en la ejecución del programa.

Formato 2: sentencia OPEN para archivos indexados y relativos



Formato 3: sentencia OPEN para archivos secuenciales de línea



Las frases INPUT, OUTPUT, I-O y EXTEND especifican la modalidad que se utilizará para abrir el archivo. Se debe especificar al menos una de las frases INPUT, OUTPUT, I-O o EXTEND con la palabra clave OPEN. Las frases INPUT, OUTPUT, I-O y EXTEND pueden aparecer en cualquier orden.

Entrada
Permite operaciones de entrada.

Salida
Permite operaciones de salida. Esta frase se puede especificar cuando se crea el archivo.

No especifique OUTPUT para los archivos que contienen registros. El archivo se sustituirá por datos nuevos.

I-O
Permite operaciones de entrada y salida. La frase I-O sólo se puede especificar para archivos asignados a dispositivos de acceso directo.

La frase I-O no es válida para archivos secuenciales de línea.

Ampliar
Permite operaciones de salida que añaden o crean un archivo.

La frase EXTEND sólo está permitida para archivos de acceso secuencial si los nuevos datos se graban en secuencia ascendente. La frase EXTEND está permitida para los archivos que especifican la cláusula LINAGE.

file-name-1*, *file-name-2*, *file-name-3*, *file-name-4
Designe un archivo en el que operará la sentencia OPEN. Si se especifica más de un archivo, no es necesario que los archivos tengan la misma organización o modalidad de acceso. Cada nombre de archivo debe estar definido en una entrada FD en DATA DIVISION y no debe nombrar un archivo de clasificación o fusión. La entrada FD debe ser equivalente a la información proporcionada cuando se definió el archivo.

INVERTIDO
La frase REVERSED es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

NO REWIND
La frase NO REWIND es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Reglas generales

El tema muestra las reglas generales de la sentencia OPEN.

- Si un archivo abierto con la frase INPUT es un archivo opcional que no está disponible, la sentencia OPEN establece el indicador de posición de archivo para indicar que un archivo de entrada opcional no está disponible.

- La ejecución de una sentencia OPEN INPUT o OPEN I-O establece el indicador de posición de archivo:
  - Para archivos indexados, a los caracteres con la posición ordinal más baja en la secuencia de clasificación asociada con el archivo.
  - Para archivos secuenciales y relativos, a 1.
- Cuando se especifica la frase EXTEND, la sentencia OPEN coloca el archivo inmediatamente después del último registro grabado en el archivo. (El registro con el valor de clave de registro principal más alto para archivos indexados o el valor de clave relativa para archivos relativos se considera el último registro.) Las sentencias WRITE posteriores añaden registros como si el archivo se abriera OUTPUT. La frase EXTEND se puede especificar cuando se está creando un archivo; también se puede especificar para un archivo que contiene registros, o que ha contenido registros que se han suprimido. Para obtener más información, consulte la nota 1 en "Notas de la sentencia OPEN" en la página 359 y SELECT OPTIONAL en "cláusula SELECT" en la página 114.
- Cuando no se especifica la frase EXTEND, la sentencia OPEN coloca el archivo al principio.

Si se especifica más de un nombre de archivo en una sentencia OPEN, el resultado de ejecutar esta sentencia OPEN es el mismo que si se hubiera grabado una sentencia OPEN distinta para cada nombre de archivo en el mismo orden que el especificado en la sentencia OPEN. Estas sentencias OPEN separadas tendrían cada una la misma especificación de modalidad abierta, y la frase REWIND tal como se especifica en la sentencia OPEN. Si una sentencia OPEN implícita da como resultado la ejecución de un procedimiento declarativo que ejecuta una sentencia RESUME con la frase NEXT STATEMENT, el proceso se reanuda en la siguiente sentencia OPEN implícita, si la hay.

Etiquetar registros

El proceso de etiquetas no está soportado.

Se emite un mensaje de aviso si se encuentra alguno de los siguientes elementos de idioma:

- LABEL RECORDS IS nombre-datos
- USE ...DESPUÉS ...PROCEDIMIENTO DE ETIQUETA

Notas de la sentencia OPEN

El tema proporciona notas para la sentencia OPEN.

Las notas son:

1. La ejecución correcta de una sentencia OPEN determina la disponibilidad del archivo y hace que el archivo esté en modalidad abierta. Un archivo está *disponible* si está físicamente presente y reconocido por el sistema de control de entrada-salida. La tabla siguiente muestra los resultados de la apertura de archivos disponibles y no disponibles.

Tabla 52. ***Disponibilidad de un archivo***

Abierto como	**El archivo está disponible**	**El archivo no está disponible**
INPUT	Apertura normal	La apertura no se ha realizado correctamente. (estado de archivo 35)
INPUT (archivo opcional)	Apertura normal	Apertura normal; la primera lectura provoca la condición de finalización o la condición de clave no válida. (estado de archivo 05)
I-O	Apertura normal	La apertura no se ha realizado correctamente. (estado de archivo 35)
I-O (archivo opcional)	Apertura normal	Abrir hace que se cree el archivo. (estado de archivo 05)

*Tabla 52. **Disponibilidad de un archivo** (continuación)*		
Abierto como	**El archivo está disponible**	**El archivo no está disponible**
OUTPUT	Apertura normal; el archivo no contiene registros	Abrir hace que se cree el archivo.
AMPLIAR	Apertura normal	La apertura no se ha realizado correctamente. (estado de archivo 35)
EXTEND (archivo opcional)	Apertura normal	Abrir hace que se cree el archivo. (estado de archivo 05)

2. La ejecución satisfactoria de la sentencia OPEN coloca el archivo en estado abierto y hace que el área de registro asociada esté disponible para el programa.
3. La sentencia OPEN no obtiene ni libera el primer registro de datos.
4. Puede mover datos a o desde el área de registro sólo cuando el archivo está en estado abierto.
5. Una sentencia OPEN debe ejecutarse correctamente antes de la ejecución de cualquiera de las sentencias de entrada-salida permitidas, excepto una sentencia SORT o MERGE con la frase USING o GIVING. En la tabla siguiente, una ' X' indica que la sentencia especificada se puede utilizar con la modalidad de apertura proporcionada en la parte superior de la columna.

*Tabla 53. **Sentencias admisibles para archivos secuenciales***				
Sentencia	**Modalidad de apertura de entrada**	**Modalidad de apertura de salida**	**Modalidad de apertura I-O**	**Ampliar modalidad abierta**
READ	X		X	
WRITE		X		X
REWRITE			X	

En la tabla siguiente, una ' X' indica que la sentencia especificada, utilizada en la modalidad de acceso proporcionada para esa fila, se puede utilizar con la modalidad de apertura proporcionada en la parte superior de la columna.

*Tabla 54. **Sentencias admisibles para archivos indexados y relativos***					
Modalidad de acceso a archivo	**Sentencia**	**Modalidad de apertura de entrada**	**Modalidad de apertura de salida**	**Modalidad de apertura I-O**	**Ampliar modalidad abierta**
secuencial	READ	X		X	
	WRITE		X		X
	REWRITE			X	
	START	X		X	
	DELETE			X	

Tabla 54. ***Sentencias admisibles para archivos indexados y relativos*** *(continuación)*					
Modalidad de acceso a archivo	**Sentencia**	**Modalidad de apertura de entrada**	**Modalidad de apertura de salida**	**Modalidad de apertura I-O**	**Ampliar modalidad abierta**
Aleatorio	READ	X		X	
	WRITE		X	X	
	REWRITE			X	
	START				
	DELETE			X	
dinámico	READ	X		X	
	WRITE		X	X	
	REWRITE			X	
	START	X		X	
	DELETE			X	

En la tabla siguiente, una ' X' indica que la sentencia especificada se puede utilizar con la modalidad de apertura proporcionada en la parte superior de la columna.

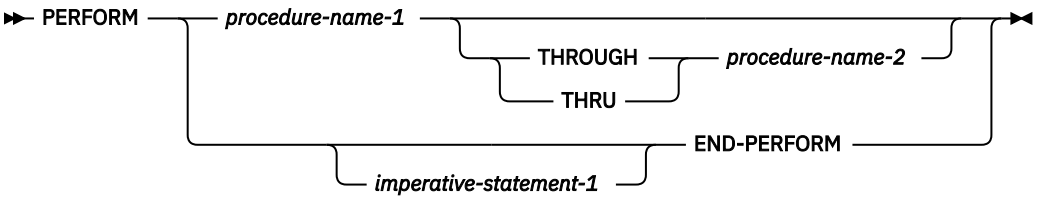
Tabla 55. ***Sentencias permisibles para archivos secuenciales de línea***				
Sentencia	**Modalidad de apertura de entrada**	**Modalidad de apertura de salida**	**Modalidad de apertura I-O**	**Ampliar modalidad abierta**
READ	X			
WRITE		X		X
REWRITE				

1. Se puede abrir un archivo para INPUT, OUTPUT, I-O o EXTEND (sólo archivos secuenciales y secuenciales de línea) en el mismo programa. Después de la primera ejecución de sentencia OPEN para un archivo determinado, cada ejecución de sentencia OPEN posterior debe ir precedida de una ejecución de sentencia de archivo CLOSE satisfactoria sin la frase LOCK.
2. Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, la clave de estado de archivo asociada se actualiza cuando se ejecuta la sentencia OPEN.
3. Si se ejecuta una sentencia OPEN para un archivo que ya está abierto, se ejecuta el procedimiento EXCEPTION/ERROR (si se especifica) para este archivo.

PERFORM, sentencia

La sentencia PERFORM transfiere el control explícitamente a uno o más procedimientos y devuelve implícitamente el control a la siguiente sentencia ejecutable después de que se haya completado la ejecución de los procedimientos especificados.

Formato 1: sentencia PERFORM básica



procedure-name-1* , *procedure-name-2
Debe nombrar una sección o párrafo en la división de procedimiento.

Cuando se especifican *procedure-name-1* y *procedure-name-2* , si cualquiera de los dos es un nombre de procedimiento en un procedimiento declarativo, ambos deben ser nombres de procedimiento en el mismo procedimiento declarativo.

Si se especifica *procedure-name-1* , *imperative-statement-1* y no se debe especificar la frase END-PERFORM.

Si se omite *procedure-name-1* , se deben especificar *imperative-statement-1* y la frase END-PERFORM.

imperative-statement-1
Las sentencias que se ejecutarán para un PERFORM en línea

Sentencias PERFORM en línea y fuera de línea

La sentencia PERFORM es una sentencia PERFORM en línea, cuando se omite *procedure-name-1* .

La sentencia PERFORM es una sentencia PERFORM fuera de línea, cuando se especifica *procedure-name-1* .

Un PERFORM en línea debe estar delimitado por la frase END-PERFORM.

Los formatos en línea y fuera de línea no se pueden combinar. Por ejemplo, si se especifica *procedure-name-1* , no se deben especificar las sentencias imperativas y la frase END-PERFORM.

Puede utilizar la sentencia EXIT PERFORM para salir de un PERFORM en línea sin utilizar una sentencia GO TO o un PERFORM ... Sentencia THROUGH. Para obtener más información, consulte “Formato 5 (ejecución en línea)” en la página 324.

FIN-PERFORM

Delimita el ámbito de la sentencia PERFORM en línea. La ejecución de un PERFORM en línea se completa después de que se haya ejecutado la última sentencia contenida en él.

Sentencia PERFORM básica

Los procedimientos a los que se hace referencia en la sentencia PERFORM básica se ejecutan una vez y, a continuación, el control pasa a la siguiente sentencia ejecutable después de la sentencia PERFORM.

Atención: Una sentencia PERFORM no debe hacer que se ejecute a sí misma. Esto constituye un PERFORM recursivo, que puede provocar resultados imprevisibles. Por lo tanto, no debe especificar sentencias PERFORM recursivas.

Una sentencia PERFORM en línea funciona de acuerdo con las mismas reglas generales que una sentencia PERFORM fuera de línea idéntica, excepto que las sentencias contenidas en el PERFORM en línea se ejecutan en lugar de las sentencias contenidas en el rango de *procedure-name-1* (a través de

procedure-name-2, si se especifica). A menos que esté calificada específicamente por la palabra *in-line* o la palabra *out-of-line*, todas las reglas que se aplican a la sentencia PERFORM out-of-line también se aplican a PERFORM in-line.

Siempre que se ejecuta una sentencia PERFORM fuera de línea, el control se transfiere a la primera sentencia del procedimiento denominada *procedure-name-1*. El control siempre se devuelve a la sentencia que sigue a la sentencia PERFORM. El punto desde el que se devuelve este control se determina de la siguiente manera:

- Si *procedure-name-1* es un nombre de párrafo y no se especifica *procedure-name-2* , la devolución se realiza después de la ejecución de la última sentencia del párrafo *procedure-name-1* .
- Si *procedure-name-1* es un nombre de sección y no se especifica *procedure-name-2* , la devolución se realiza después de la ejecución de la última sentencia del último párrafo en la sección *procedure-name-1* .
- Si se especifica *procedure-name-2* y es un nombre de párrafo, la devolución se realiza después de la ejecución de la última sentencia del párrafo *procedure-name-2* .
- Si se especifica *procedure-name-2* y es un nombre de sección, la devolución se realiza después de la ejecución de la última sentencia del último párrafo en la sección *procedure-name-2* .

La única relación necesaria entre *procedure-name-1* y *procedure-name-2* es que se ejecute una secuencia consecutiva de operaciones, empezando por el procedimiento denominado por *procedure-name-1* y finalizando con la ejecución del procedimiento denominado por *procedure-name-2*.

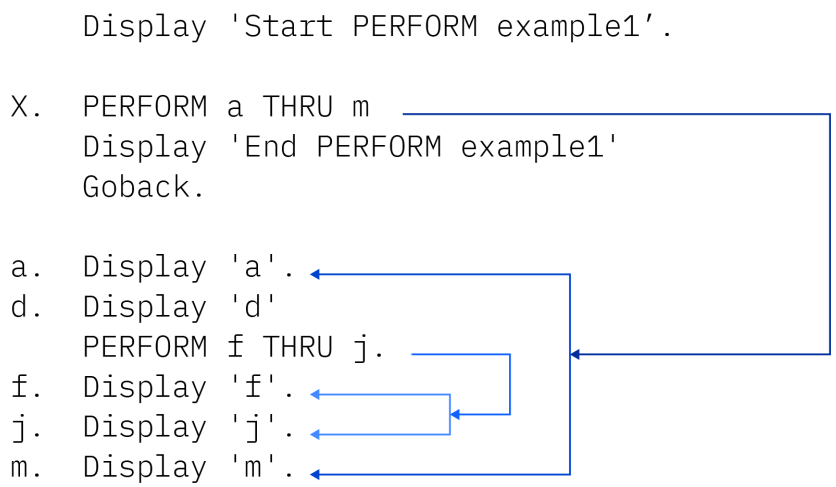
Las sentencias PERFORM pueden especificarse dentro del procedimiento realizado. Si hay dos o más vías de acceso lógicas al punto de retorno, *procedure-name-2* puede nombrar un párrafo que consta sólo de una sentencia EXIT; todas las vías de acceso al punto de retorno deben llevar a este párrafo.

Cuando los procedimientos realizados incluyen otra sentencia PERFORM, la secuencia de procedimientos asociados con la sentencia PERFORM incorporada debe estar totalmente incluida o totalmente excluida de los procedimientos realizados de la primera sentencia PERFORM. Es decir, una sentencia PERFORM activa cuyo punto de ejecución empieza dentro del rango de procedimientos realizados de otra sentencia PERFORM activa no debe permitir que el control pase a través del punto de salida de la otra sentencia PERFORM activa. Sin embargo, dos o más sentencias PERFORM activas pueden tener una salida común.

Cuando el control pasa a la secuencia de procedimientos por medios distintos de una sentencia PERFORM, el control pasa a través del punto de salida a la siguiente sentencia ejecutable, como si ninguna sentencia PERFORM hiciera referencia a estos procedimientos.

Las figuras siguientes ilustran secuencias válidas de ejecución para sentencias PERFORM.

Ejemplo 1

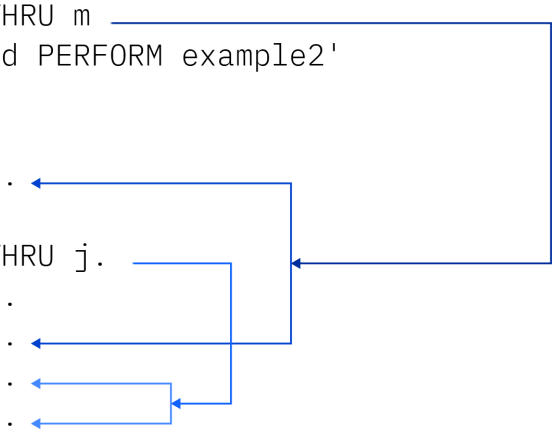


La salida del ejemplo 1 es similar a la siguiente:

```
OUTPUT:
Start PERFORM example1
a
d
f
j
f
j
m
End   PERFORM example1
```

Ejemplo 2

```
      Display 'Start PERFORM example2'.

  X.  PERFORM a THRU m
      Display 'End PERFORM example2'
      Goback.

  a.  Display 'a'.
  d.  Display 'd'
      PERFORM f THRU j.
  h.  Display 'h'.
  m.  Display 'm'.
  f.  Display 'f'.
  j.  Display 'j'.
```

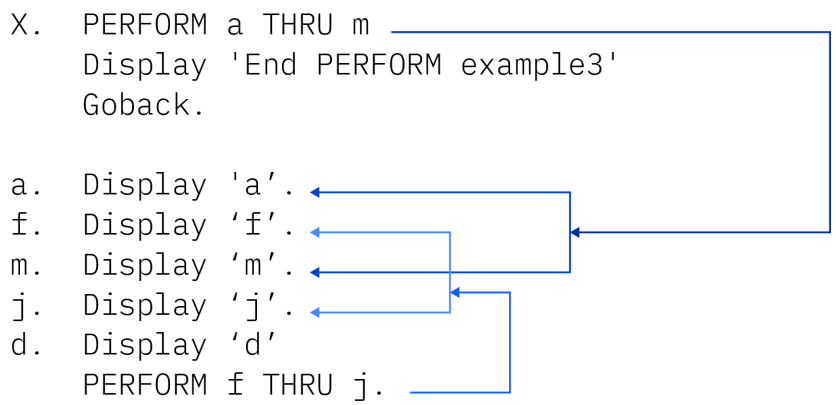


La salida del ejemplo 2 es similar a la siguiente:

```
OUTPUT:
Start PERFORM example2
a
d
f
j
h
m
End   PERFORM example2
```

Ejemplo 3

```
     Display 'Start PERFORM example3'.

X.   PERFORM a THRU m
     Display 'End PERFORM example3'
     Goback.

a.   Display 'a'.
f.   Display 'f'.
m.   Display 'm'.
j.   Display 'j'.
d.   Display 'd'
     PERFORM f THRU j.
```

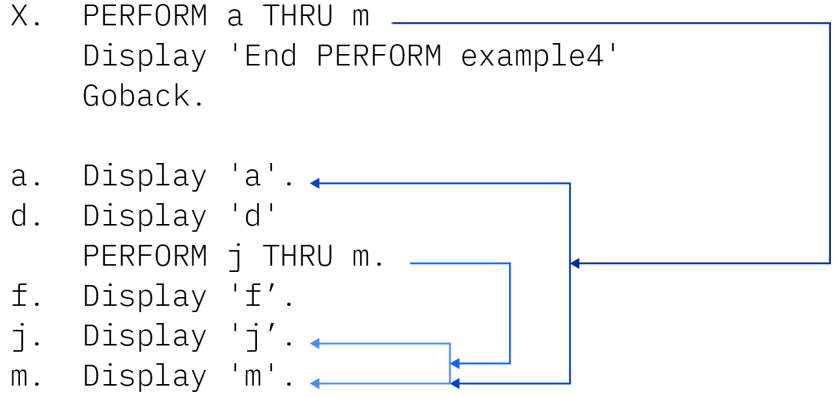


La salida del ejemplo 3 tiene este aspecto:

```
OUTPUT:
Start PERFORM example3
a
f
m
End   PERFORM example3
```

Ejemplo 4

```
     Display 'Start PERFORM example4'.

X.   PERFORM a THRU m
     Display 'End PERFORM example4'
     Goback.

a.   Display 'a'.
d.   Display 'd'
     PERFORM j THRU m.
f.   Display 'f'.
j.   Display 'j'.
m.   Display 'm'.
```



La salida del ejemplo 4 es similar a la siguiente:

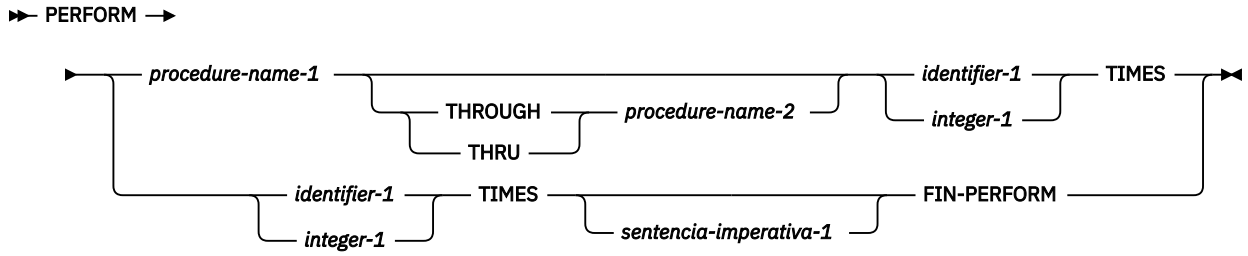
```
OUTPUT:
Start PERFORM example4
a
d
j
m
f
j
```

```
m
End   PERFORM example4
```

PERFORM con frase TIMES

Los procedimientos a los que se hace referencia en la frase TIMES de la sentencia PERFORM se ejecutan el número de veces especificado por el valor en *identifier-1* o *integer-1*, hasta un máximo de 999.999.999 veces. A continuación, el control pasa a la siguiente sentencia ejecutable después de la sentencia PERFORM.

Formato 2: sentencia PERFORM con frase TIMES



Si se especifica *procedure-name-1* , *imperative-statement-1* y no se debe especificar la frase END-PERFORM.

identifier-1
Debe nombrar un elemento entero. *identifier-1* no puede ser un campo de fecha con ventana.

Si *identifier-1* es cero o un número negativo en el momento en que se inicia la sentencia PERFORM, el control pasa a la sentencia que sigue a la sentencia PERFORM.

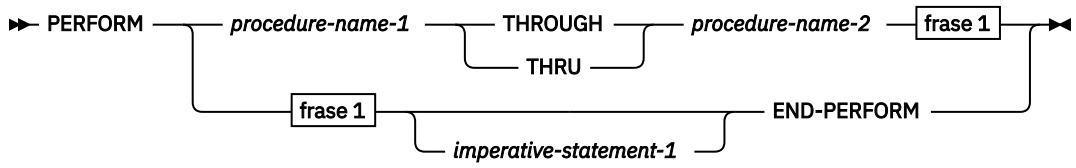
Una vez iniciada la sentencia PERFORM, cualquier cambio en el *identifier-1* no tiene ningún efecto en la variación del número de veces que se inician los procedimientos.

integer-1
Puede ser un entero con signo positivo.

PERFORM con frase UNTIL

En el formato de frase UNTIL, los procedimientos a los que se hace referencia se realizan *hasta* que la condición especificada por la frase UNTIL es verdadera. A continuación, el control se pasa a la siguiente sentencia ejecutable después de la sentencia PERFORM.

Formato 3: sentencia PERFORM con frase UNTIL



frase 1

Si se especifica *procedure-name-1* , *imperative-statement-1* y no se debe especificar la frase END-PERFORM.

condition-1

Puede ser cualquier condición descrita en "Expresiones condicionales" en la página 250. Si la condición es verdadera en el momento en que se inicia la sentencia PERFORM, los procedimientos especificados no se ejecutan.

Cualquier suscripción asociada con los operandos especificados en *condition-1* se evalúa cada vez que se prueba la condición.

Si se especifica o se asume la frase TEST BEFORE, la condición se prueba antes de que se ejecute cualquier sentencia (corresponde a DO WHILE).

Si se especifica la frase TEST AFTER, las sentencias que se van a realizar se ejecutan al menos una vez antes de que se pruebe la condición (corresponde a DO UNTIL).

En cualquier caso, si la condición es verdadera, el control se transfiere a la siguiente sentencia ejecutable después del final de la sentencia PERFORM. Si no se especifica la frase TEST BEFORE ni la frase TEST AFTER, se asume la frase TEST BEFORE.

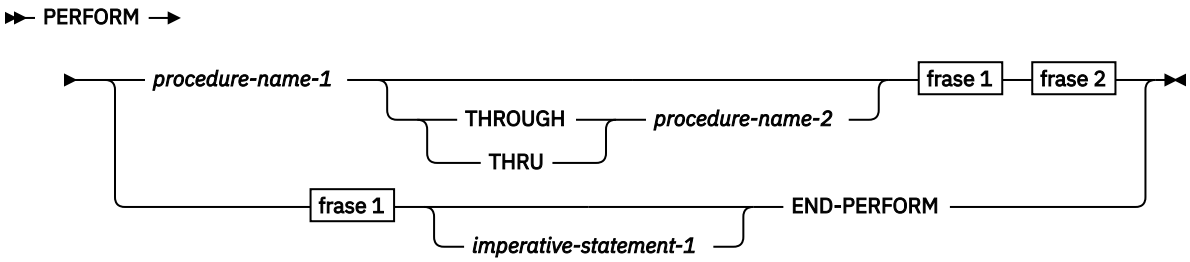
PERFORM con frase VARYING

La frase VARYING aumenta o disminuye el valor de uno o más identificadores o nombres de índice, de acuerdo con determinadas reglas.

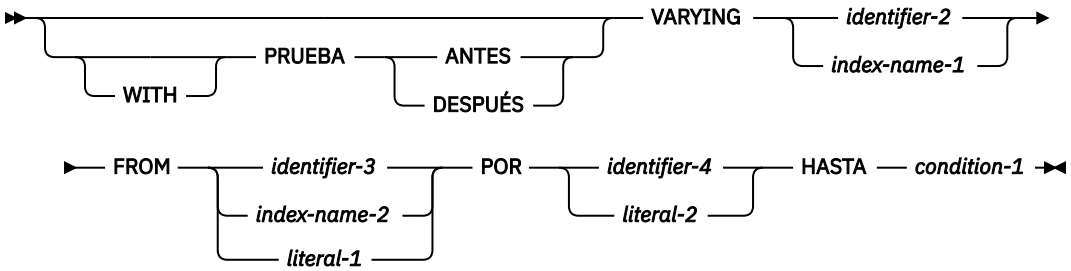
Para obtener más información, consulte el apartado "Reglas de frase variable" en la página 371.

La sentencia format-4 VARYING de frase PERFORM puede buscar en serie una tabla de siete dimensiones completa.

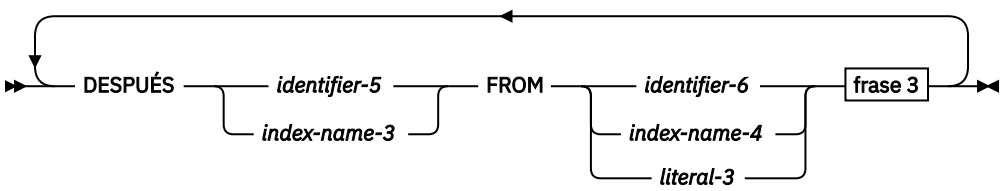
Formato 4: sentencia PERFORM con frase VARYING



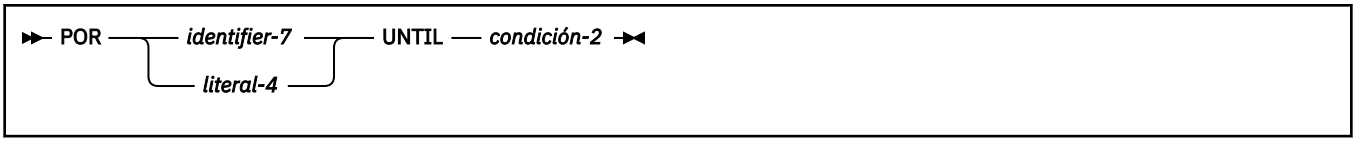
frase 1



frase 2



frase 3



Si se especifica *procedure-name-1* , *imperative-statement-1* y no se debe especificar la frase END-PERFORM. Si se omite *procedure-name-1* , no se debe especificar la frase AFTER.

identifier-2* a *identifier-7
Debe nombrar un elemento elemental numérico. Estos identificadores no pueden ser campos de fecha con ventana.

literal-1* a *literal-4
Debe representar un literal numérico.

condition-1*, *condition-2
Puede ser cualquier condición descrita en "Expresiones condicionales" en la página 250. Si la condición es verdadera en el momento en que se inicia la sentencia PERFORM, los procedimientos especificados no se ejecutan.

Después de que se cumplan las condiciones especificadas en la frase UNTIL, el control se pasa a la siguiente sentencia ejecutable después de la sentencia PERFORM.

Si alguno de los operandos especificados en *condition-1* o *condition-2* tiene subíndice, se modifica la referencia o es un identificador de función, el subíndice, modificador de referencia o función se evalúa cada vez que se prueba la condición.

Los elementos de datos de coma flotante y literales se pueden utilizar en cualquier lugar donde se pueda especificar un elemento de datos numérico o literal.

Cuando se indica TEST BEFORE, todas las condiciones especificadas se prueban antes de la primera ejecución, y las sentencias que se van a realizar se ejecutan, si es que se ejecutan, sólo cuando *todas* las pruebas especificadas fallan. Cuando se indica TEST AFTER, las sentencias que deben realizarse se ejecutan al menos una vez, antes de que se pruebe cualquier condición.

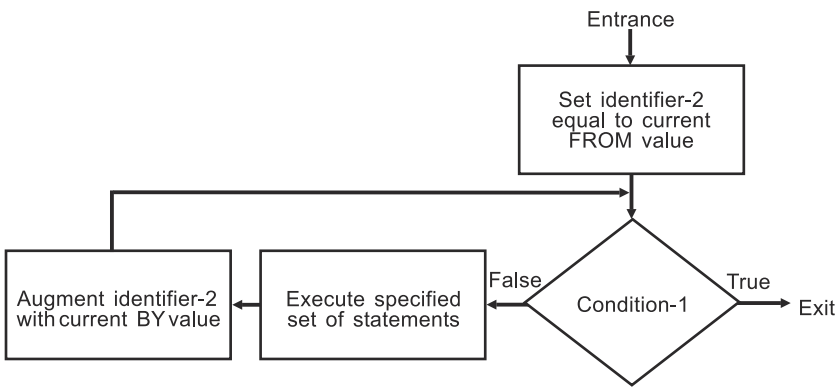
Si no se especifica la frase TEST BEFORE ni la frase TEST AFTER, se asume la frase TEST BEFORE.

Identificadores variables

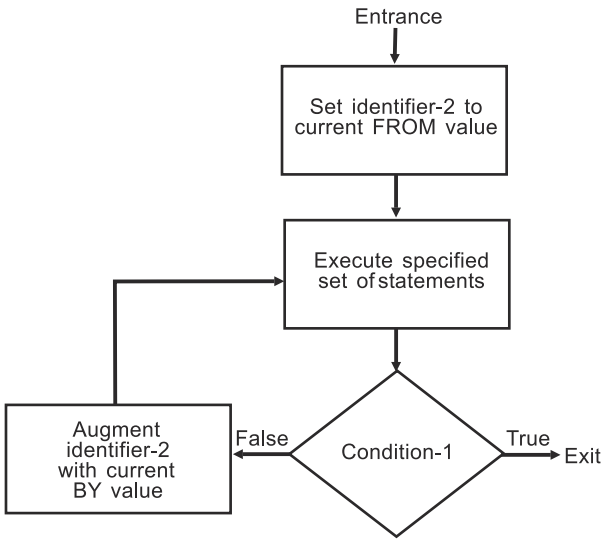
La forma en que se aumentan o disminuyen los operandos depende del número de variables especificadas. En la discusión, cada referencia a *identifier-n* hace referencia igualmente a *index-name-n* (excepto cuando *identifier-n* es el objeto de la frase BY).

Si *identifier-2* o *identifier-5* está subíndice, los subíndices se evalúan cada vez que se establece o aumenta el contenido del elemento de datos al que hace referencia el identificador. Si *identifier-3*, *identifier-4*, *identifier-6*, o *identifier-7* es un subíndice, los subíndices se evalúan cada vez que se utiliza el contenido del elemento de datos al que hace referencia el identificador en un valor o en una operación de aumento.

La figura siguiente ilustra la lógica de la sentencia PERFORM cuando se cambia un identificador con TEST BEFORE.



La figura siguiente ilustra la lógica de la sentencia PERFORM cuando se cambia un identificador con TEST AFTER.



Variación de dos identificadores

En el tema se listan los pasos para variar dos identificadores.

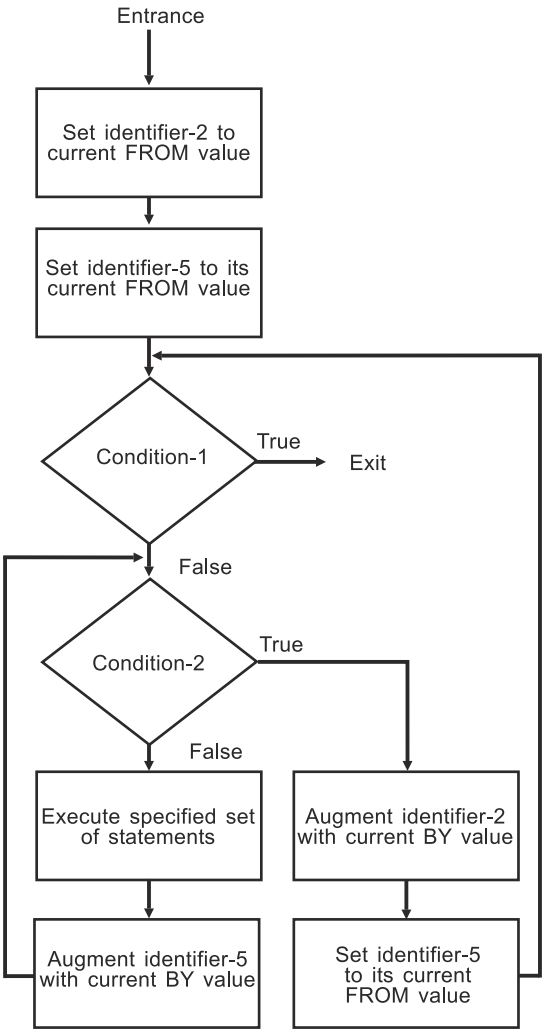
```
PERFORM PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2
      VARYING IDENTIFIER-2 FROM IDENTIFIER-3
            BY IDENTIFIER-4 UNTIL CONDITION-1
         AFTER IDENTIFIER-5 FROM IDENTIFIER-6
            BY IDENTIFIER-7 UNTIL CONDITION-2
```

1. *identifier-2* y *identifier-5* se establecen en sus valores iniciales, *identifier-3* y *identifier-6*, respectivamente.
2. *condition-1* se evalúa de la siguiente manera:
   a. Si es false, se ejecutan los pasos del 3 al 7.
   b. Si es true, el control pasa directamente a la sentencia que sigue a la sentencia PERFORM.
3. *condition-2* se evalúa de la siguiente manera:
   a. Si es false, se ejecutan los pasos del 4 al 6.
   b. Si es true, *identifier-2* se aumenta mediante *identifier-4*, *identifier-5* se establece en el valor actual de *identifier-6*, y el paso 2 se repite.
4. *procedure-name-1* y *procedure-name-2* se ejecutan una vez (si se especifica).
5. *identifier-5* se aumenta mediante *identifier-7*.
6. Los pasos del 3 al 5 se repiten hasta que *condition-2* es true.
7. Los pasos del 2 al 6 se repiten hasta que *condition-1* sea verdadero.

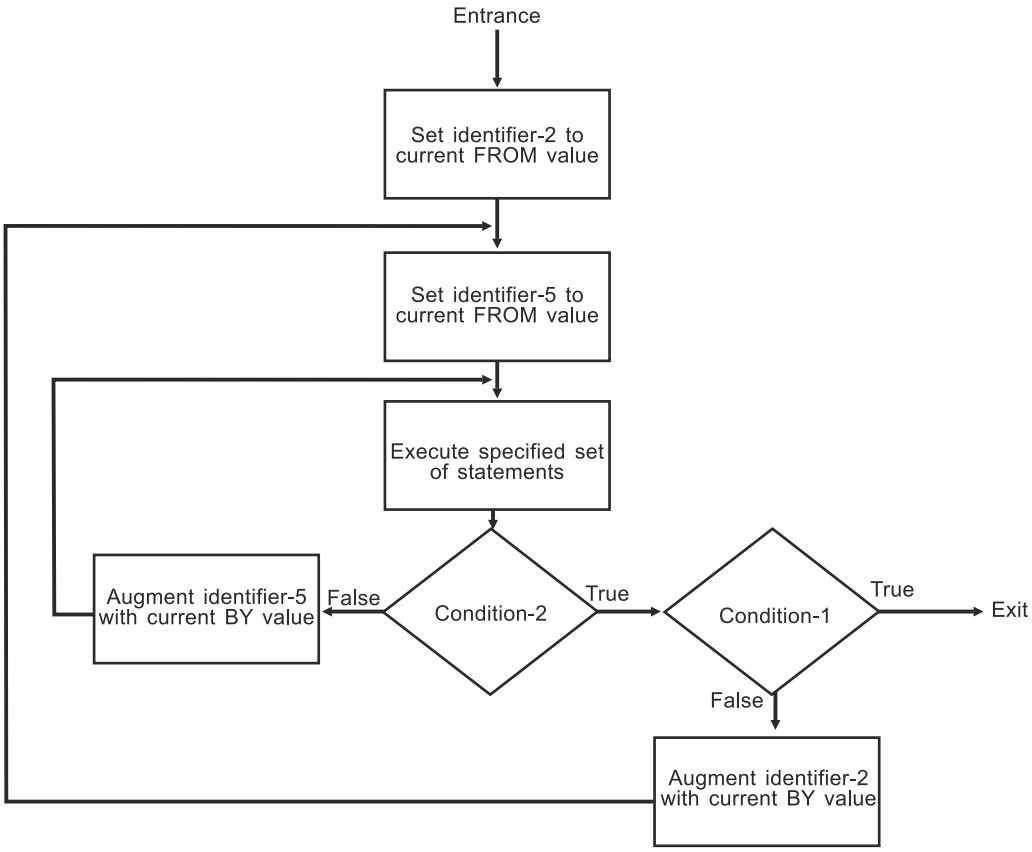
Al final de la ejecución de la sentencia PERFORM:

- *identifier-5* contiene el valor actual de *identifier-6*.
- *identifier-2* tiene un valor que excede el último valor utilizado por el valor de incremento o decremento (a menos que *condition-1* fuera verdadero al principio de la ejecución de la sentencia PERFORM, en cuyo caso, *identifier-2* contiene el valor actual de *identifier-3*).

La figura siguiente ilustra la lógica de la sentencia PERFORM cuando se varían dos identificadores con TEST BEFORE.



La figura siguiente ilustra la lógica de la sentencia PERFORM cuando se varían dos identificadores con TEST AFTER.



Variación de tres identificadores

El tema lista los pasos para variar tres identificadores.

```
PERFORM PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2
      VARYING IDENTIFIER-2 FROM IDENTIFIER-3
            BY IDENTIFIER-4 UNTIL CONDITION-1
          AFTER IDENTIFIER-5 FROM IDENTIFIER-6
            BY IDENTIFIER-7 UNTIL CONDITION-2
          AFTER IDENTIFIER-8 FROM IDENTIFIER-9
            BY IDENTIFIER-10 UNTIL CONDITION-3
```

Las acciones son las mismas que las de dos identificadores, excepto que *identifier-8* pasa por el ciclo completo cada vez que se aumenta *identifier-5* mediante *identifier-7*, que, a su vez, pasa por un ciclo completo cada vez que se cambia *identifier-2* .

Al final de la ejecución de la sentencia PERFORM:

- *identifier-5* y *identifier-8* contienen los valores actuales de *identifier-6* y *identifier-9*, respectivamente.
- *identifier-2* tiene un valor que excede su último valor utilizado por un valor de incremento/decremento (a menos que *condition-1* fuera verdadero al principio de la ejecución de la sentencia PERFORM, en cuyo caso *identifier-2* contiene el valor actual de *identifier-3*).

Variación de más de tres identificadores

Puede producir acciones de sentencia PERFORM análogas al ejemplo anterior con la adición de hasta cuatro frases AFTER.

Reglas de frase variable

Hay ciertas reglas que se aplican a esta frase, sin importar cuántas variables se especifiquen.

Las reglas son:

- En las frases VARYING o AFTER, cuando se especifica un nombre de índice:

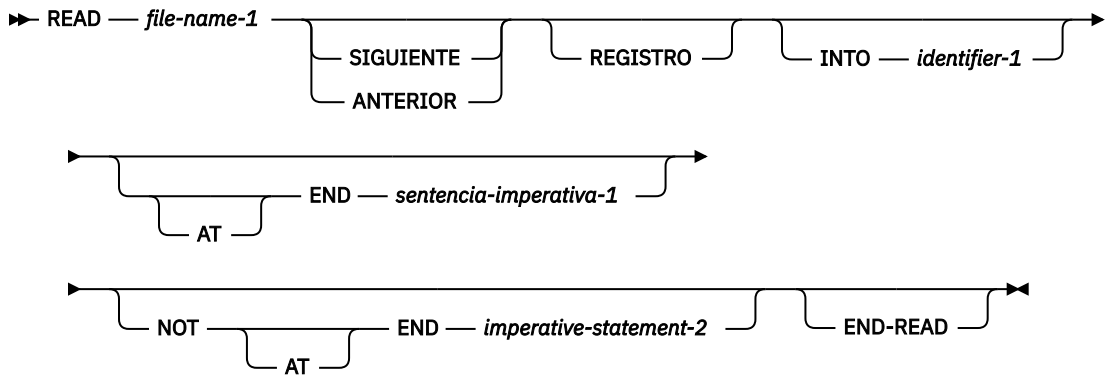
  - El nombre de índice se inicializa e incrementa o disminuye de acuerdo con las reglas en “frase INDEX” en la página 229. (Consulte también “sentencia SET” en la página 389.)
  - En la frase FROM asociada, un identificador debe describirse como un entero y tener un valor positivo; un literal debe ser un entero positivo.
  - En la frase BY asociada, un identificador debe describirse como un entero; un literal debe ser un entero distinto de cero.
- En la frase FROM, cuando se especifica un nombre de índice:
  - En la frase VARYING o AFTER asociada, un identificador debe describirse como un entero. Se inicializa tal como se describe en la sentencia SET.
  - En la frase BY asociada, un identificador debe describirse como un entero y tener un valor distinto de cero; un literal debe ser un entero distinto de cero.
- En la frase BY, los identificadores y literales deben tener valores distintos de cero.
- El cambio de los valores de identificadores o nombres de índice en las frases VARYING, FROM y BY durante la ejecución cambia el número de veces que se ejecutan los procedimientos.

sentencia READ

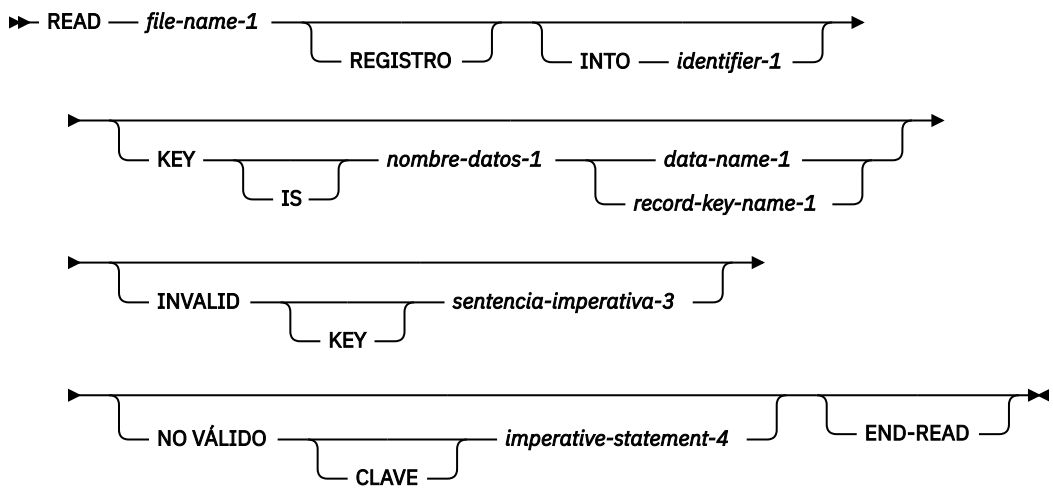
Para el acceso secuencial, la sentencia READ hace que el siguiente registro lógico de un archivo esté disponible para el programa objeto. Para el acceso aleatorio, la sentencia READ hace que un registro especificado de un archivo de acceso directo esté disponible para el programa objeto.

Cuando se ejecuta la sentencia READ, el archivo asociado debe estar abierto en modalidad INPUT o I-O.

Formato 1: sentencia READ para recuperación secuencial



Formato 2: sentencia READ para recuperación aleatoria



Restricción: *nombre-clave-registro* sólo está soportado para el sistema de archivos STL.

file-name-1
Debe definirse en una entrada DATA DIVISION FD.

SIGUIENTE REGISTRO
Lee el siguiente registro en la secuencia lógica de registros. NEXT es opcional cuando la modalidad de acceso es secuencial y no tiene ningún efecto en la ejecución de sentencias READ.

Debe especificar la frase NEXT o la frase PREVIOUS para recuperar registros secuencialmente de archivos en modalidad de acceso dinámico.

REGISTRO ANTERIOR
Lee el registro anterior en la secuencia lógica de registros. PREVIOUS se aplica a los archivos indexados y relativos con la modalidad de acceso DYNAMIC.

Para recuperar registros secuencialmente, debe especificar la frase NEXT o la frase PREVIOUS para los archivos en modalidad de acceso dinámico.

Si especifica READ ...PREVIOUS y no existe ningún registro lógico anterior, se produce la condición AT END y la sentencia READ no es satisfactoria.

Al especificar READ ...ANTERIOR, el valor del indicador de posición de archivo se utiliza para determinar qué registro debe estar disponible de acuerdo con las reglas siguientes:

- Si el indicador de posición de archivo indica que no se ha establecido ningún registro anterior válido, READ no es satisfactorio.
- Si el indicador de posición de archivo se coloca mediante la ejecución de una sentencia OPEN, se produce la condición AT END.
- Si el indicador de posición de archivo se establece mediante una sentencia START anterior, se selecciona el primer registro existente en el archivo cuyo número de registro relativo (si es un archivo relativo) o cuyo valor de clave (si es un archivo indexado) es menor o igual que el indicador de posición de archivo.
- Si el indicador de posición de archivo se establece mediante una sentencia READ anterior, se selecciona el primer registro existente en el archivo cuyo número de registro relativo (si es un archivo relativo) o cuyo valor de clave (si es un archivo indexado) es menor que el indicador de posición de archivo.

INTO *identifier-1*
identifier-1 es el campo de recepción.

identifier-1 debe ser un campo de recepción válido para la entrada de descripción de registro de envío seleccionada de acuerdo con las reglas de la sentencia MOVE.

Las áreas de registro asociadas con *file-name-1* y *identifier-1* no deben ser la misma área de almacenamiento.

Cuando sólo hay una descripción de registro asociada con *file-name-1* o todos los registros y el elemento de datos referenciado por *identifier-1* describen un elemento alfanumérico elemental o un elemento de grupo alfanumérico, el resultado de la ejecución de una sentencia READ con la frase INTO es equivalente a la aplicación de las siguientes normas en el orden especificado:

- La ejecución de la misma sentencia READ sin la frase INTO.
- El registro actual se mueve del área de registro al área especificada por *identifier-1* de acuerdo con las reglas para la sentencia MOVE sin la frase CORRESPONDIENTE. El tamaño del registro actual viene determinado por las reglas especificadas para la cláusula RECORD. Si la entrada de descripción de archivo contiene una cláusula RECORD IS VARIABLE, el movimiento implícito es un movimiento de grupo. La sentencia MOVE implícita no se produce si la ejecución de la sentencia READ no ha sido satisfactoria. Cualquier subíndice o modificación de referencia asociada con *identifier-1* se evalúa después de que se haya leído el registro e inmediatamente antes de que se mueva al elemento de datos. El registro está disponible tanto en el área de registro como en el elemento de datos al que hace referencia *identifier-1*.

  Si *identifier-1* es un campo de fecha, la sentencia MOVE implícita se realiza de acuerdo con el comportamiento descrito en “Movimientos que implican campos de fecha” en la página 353.

Cuando hay varias descripciones de registro asociadas con *file-name-1* y no todas describen un elemento de grupo alfanumérico o un elemento alfanumérico elemental, se aplican las reglas siguientes:

1. Si el archivo al que hace referencia *file-name-1* se describe como que contiene registros de longitud variable, tendrá lugar un movimiento de grupo.
2. Si el archivo al que hace referencia *file-name-1* se describe como que contiene registros de longitud fija, se realizará un movimiento de acuerdo con las reglas de una sentencia MOVE utilizando, como descripción de campo de envío, el registro que especifica el mayor número de posiciones de caracteres. Si existe más de un registro de este tipo, el registro de campo de envío seleccionado será el que aparezca primero bajo la descripción de *file-name-1*.

Frase KEY IS

La frase KEY IS sólo puede especificarse para archivos indexados.

data-name-1 o *record-key-name-1* debe especificarse en la cláusula RECORD KEY o en una cláusula ALTERNO RECORD KEY asociada con *file-name-1*. *data-name-1* o *record-key-name-1* se puede calificar.

frases AT END

Para el acceso secuencial, se puede omitir la frase AT END y un procedimiento EXCEPTION/ERROR aplicable.

Para obtener información sobre el proceso de condición de finalización, consulte Condición AT END.

Frases INVALID KEY

Se puede omitir la frase INVALID KEY y un procedimiento EXCEPTION/ERROR aplicable.

Para obtener información sobre el proceso de frase INVALID KEY, consulte “Condición de clave no válida” en la página 290.

Frase END-READ

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia READ. END-READ permite que una sentencia READ condicional se anide en otra sentencia condicional. END-READ también se puede utilizar con una sentencia READ imperativa. Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

Proceso de varios registros

Si hay más de una entrada de descripción de registro asociada con *file-name-1*, estos registros comparten automáticamente la misma área de almacenamiento; es decir, se redefinen implícitamente. Después de ejecutar una sentencia READ, sólo se sustituyen los elementos de datos dentro del rango del registro actual; los elementos de datos almacenados fuera de ese rango no están definidos. El ejemplo siguiente ilustra este concepto.

Si el rango del registro actual supera las entradas de descripción de registro para *file-name-1*, el registro se trunca a la derecha hasta el tamaño máximo.

En cualquiera de los casos anteriores, la sentencia READ es satisfactoria y el estado de I-O se establece en 04, lo que indica que se ha producido un conflicto de longitud de registro.

El ejemplo siguiente muestra dos áreas de registro de diferentes tamaños en un FD. Cuando se lee un registro más corto, el contenido del área de registro restante no está definido.

```
FD INPUT-FILE LABEL RECORD OMITTED.
01    RECORD-1 PICTURE X(30).
01    RECORD-2 PICTURE X(20).
```

Contenido del área de entrada cuando se ejecuta la sentencia READ:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234
```

Contenido del registro que se lee en (RECORD-2):

```
01234567890123456789
```

Contenido del área de entrada después de ejecutar la sentencia READ:

```
0123456789012345678?????????
```

El "?" son caracteres no definidos en el área de entrada.

Modalidad de acceso secuencial

El formato 1 debe utilizarse para todos los archivos en modalidad de acceso secuencial.

La ejecución de una sentencia READ format-1 recupera el siguiente registro lógico del archivo. La organización del archivo determina el siguiente registro al que se accede.

Archivos secuenciales

El NEXT RECORD es el siguiente registro de una secuencia lógica de registros. No es necesario especificar la frase NEXT; no tiene ningún efecto en la ejecución de la sentencia READ.

Si se especifica SELECT OPTIONAL en la entrada de control de archivos para este archivo, y el archivo no está disponible durante esta ejecución del programa objeto, la ejecución de la primera sentencia READ provoca una condición de finalización; sin embargo, puesto que no hay ningún archivo disponible, no se realiza el proceso de fin de archivo definido por el sistema.

condición AT END

Si el indicador de posición de archivo indica que no existe ningún registro lógico siguiente, o que no está disponible un archivo de entrada opcional, el proceso de condición al final se produce en un orden específico.

El orden es:

1. Un valor derivado del valor del indicador de posición de archivo se coloca en el estado I-O asociado con *file-name-1* para indicar la condición de finalización.
2. Si se especifica la frase AT END en la sentencia que provoca la condición, el control se transfiere a *imperative-statement-1* en la frase AT END. Cualquier procedimiento USE AFTER STANDARD EXCEPTION asociado con *file-name-1* no se ejecuta.
3. Si no se especifica la frase AT END y existe un procedimiento USE AFTER STANDARD EXCEPTION aplicable, se ejecuta el procedimiento. El retorno de ese procedimiento es a la siguiente sentencia ejecutable después del final de la sentencia READ.

   Se puede omitir tanto la frase AT END como un procedimiento EXCEPTION/ERROR aplicable.

   Cuando se produce la condición de finalización, la ejecución de la sentencia READ no es satisfactoria. El contenido del área de registro asociada no está definido y el indicador de posición de archivo se ha establecido para indicar que no se ha establecido ningún registro siguiente válido.

Si no se produce una condición de finalización durante la ejecución de una sentencia READ, la frase AT END se ignora, si se especifica, y se producen las acciones siguientes:

1. Se establece el indicador de posición de archivo y se actualiza el estado I-O asociado con *file-name-1* .
2. Si existe una condición de excepción que no es una condición de finalización, el control se transfiere al final de la sentencia READ después de la ejecución de cualquier procedimiento USE AFTER STANDARD EXCEPTION aplicable a *file-name-1*.

   Si no se especifica ningún procedimiento USE AFTER STANDARD EXCEPTION, el control se transfiere al final de la sentencia READ o a *imperative-statement-2*, si se especifica.
3. Si no existe ninguna condición de excepción, el registro se pone a disposición en el área de registro y se ejecuta cualquier movimiento implícito resultante de la presencia de una frase INTO. El control se transfiere al final de la sentencia READ o a *imperative-statement-2*, si se especifica. En este último caso, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en *imperative-statement-2*. Si se ejecuta una ramificación de procedimiento o una sentencia condicional que provoca una transferencia explícita del control, el control se transfiere de acuerdo con las reglas de dicha sentencia; de lo contrario, al finalizar la ejecución de *imperative-statement-2*, el control se transfiere al final de la sentencia READ.

Después de la ejecución no satisfactoria de una sentencia READ, el contenido del área de registro asociada no está definido y el indicador de posición de archivo se establece para indicar que no se ha establecido ningún registro siguiente válido. Los intentos de acceder o mover datos al área de registro después de una lectura no satisfactoria pueden dar como resultado una violación de segmentación.

Archivos indexados o relativos

El NEXT RECORD es el siguiente registro lógico de la secuencia de teclas.

El PREVIOUS RECORD es el registro lógico precedente en la secuencia de teclas.

Para archivos indexados, la secuencia de teclas es la secuencia de valores ascendentes de la clave de referencia actual. Para archivos relativos, la secuencia de teclas es la secuencia de valores ascendentes de números de registro relativos para los registros que existen en el archivo.

Antes de ejecutar la sentencia READ, el indicador de posición de archivo debe haberse establecido mediante una sentencia OPEN, START o READ satisfactoria. Cuando se ejecuta la sentencia READ, el registro indicado por el indicador de posición de archivo pasa a estar disponible si todavía es accesible a través de la vía de acceso indicada por el indicador de posición de archivo.

Si el registro ya no es accesible (porque se ha suprimido, por ejemplo), el indicador de posición de archivo se actualiza para que apunte al siguiente registro (o anterior) existente en el archivo, y ese registro está disponible.

Para archivos en modalidad de acceso secuencial, no es necesario especificar la frase NEXT.

Para los archivos en modalidad de acceso dinámico, se debe especificar la frase NEXT (o la frase PREVIOUS) para la recuperación secuencial de registros.

condición AT END

Esta condición existe cuando el indicador de posición de archivo indica que no existe ningún registro lógico siguiente (o no existe ningún registro anterior) o que no está disponible un archivo de entrada opcional. Consulte la descripción de PREVIOUS RECORD anterior.

Si no se produce una condición de clave de extremo ni una condición de clave no válida durante la ejecución de una sentencia READ, la frase AT END o INVALID KEY se ignora, si se especifica. Se producen las mismas acciones que cuando la condición de finalización no se produce con archivos secuenciales (consulte Condición AT END).

Archivos indexados a los que se accede secuencialmente

Cuando una CLAVE DE REGISTRO ALTERNATIVA con DUPLICATES es la clave de referencia, los registros de archivo con valores de clave duplicados se ponen a disposición en el orden en el que se colocaron en el archivo.

Archivos relativos a los que se accede secuencialmente

Si se especifica la cláusula RELATIVE KEY para este archivo, la ejecución de la sentencia READ actualiza el elemento de datos RELATIVE KEY para indicar el número de registro relativo del registro que está disponible.

Modalidad de acceso aleatorio

El formato 2 debe especificarse para archivos indexados y relativos en modalidad de acceso aleatorio, y también para archivos en modalidad de acceso dinámico cuando la recuperación de registros es aleatoria.

La ejecución de la sentencia READ depende de la organización del archivo, tal como se explica en las secciones siguientes.

Archivos indexados

La ejecución de una sentencia format-2 READ hace que el valor de la clave de referencia se compare con el valor del elemento de datos de clave correspondiente en los registros de archivo, hasta que se encuentre el primer registro que tenga un valor igual. El indicador de posición de archivo se coloca en este registro, que a continuación pasa a estar disponible. Si no se puede identificar ningún registro, existe una condición INVALID KEY y la ejecución de la sentencia READ no es satisfactoria. (Consulte “Condición de clave no válida” en la página 290 para obtener detalles de la condición de clave no válida.)

Si no se especifica la frase KEY, RECORD KEY se convierte en la clave de referencia para esta petición. Cuando se especifica el acceso dinámico, también se utiliza RECORD KEY principal como clave de referencia para las ejecuciones posteriores de sentencias READ secuenciales, hasta que se establezca una clave de referencia diferente.

Cuando se especifica la frase KEY, *data-name-1* se convierte en la clave de referencia para esta solicitud. Cuando se especifica el acceso dinámico, esta clave de referencia se utiliza para las ejecuciones posteriores de sentencias READ secuenciales, hasta que se establece una clave de referencia diferente.

Para un archivo indexado al que se accede a través de un determinado conector de archivo, si se especifica la frase KEY, *data-name-1* o *record-key-name-1* se convierte en la clave de referencia para esta recuperación. Cuando se especifica el acceso dinámico, esta clave de referencia también se utiliza para las ejecuciones posteriores de sentencias READ secuenciales para el archivo a través del conector de archivo, hasta que se establezca una clave de referencia diferente para el archivo a través de ese conector de archivo.

Archivos relativos

La ejecución de una sentencia READ format-2 establece el puntero de indicador de posición de archivo en el registro cuyo número de registro relativo está contenido en el elemento de datos RELATIVE KEY y hace que dicho registro esté disponible.

Si el archivo no contiene un registro de este tipo, la condición INVALID KEY existe y la ejecución de la sentencia READ no es satisfactoria. (Consulte "Condición de clave no válida" en la página 290 para obtener detalles de la condición de clave no válida).

La frase KEY no debe especificarse para archivos relativos.

Modo de acceso dinámico

Para archivos con organización indexada o relativa, se puede especificar la modalidad de acceso dinámico en la entrada de control de archivos. En el modo de acceso dinámico, se puede utilizar la recuperación de registros secuenciales o aleatorios, en función del formato utilizado.

El formato 1 con la frase NEXT debe especificarse para la recuperación secuencial. Se aplican todas las demás reglas para el acceso secuencial.

Notas de la sentencia READ

Este tema proporciona notas sobre la sentencia READ.

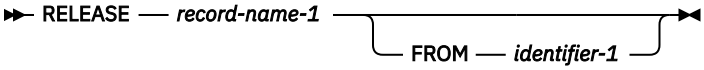
- Si se especifica la cláusula FILE-STATUS en la entrada de control de archivos, la clave de estado de archivo asociada se actualiza cuando se ejecuta la sentencia READ.
- Después de una ejecución incorrecta de la sentencia READ, el contenido del área de registro asociada y el valor del indicador de posición de archivo no están definidos. Los intentos de acceder o mover datos al área de registro después de una lectura no satisfactoria pueden dar como resultado una violación de segmentación.

sentencia RELEASE

La sentencia RELEASE transfiere registros de un área de entrada/salida a la fase inicial de una operación de clasificación.

La sentencia RELEASE sólo puede utilizarse dentro del rango de un INPUT PROCEDURE asociado a una sentencia SORT.

Formato: RELEASE



Dentro de un INPUT PROCEDURE, debe especificarse al menos una sentencia RELEASE.

Cuando se ejecuta la sentencia RELEASE, el contenido actual de *record-name-1* se coloca en el archivo de clasificación. Esto hace que el registro esté disponible para la fase inicial de la operación de clasificación.

record-name-1
Debe especificar el nombre de un registro lógico en una entrada de descripción de archivo de fusión de clasificación (SD). *record-name-1* se puede calificar.

Frase FROM
El resultado de la ejecución de la sentencia RELEASE con la frase FROM *identifier-1* es equivalente a la ejecución de las sentencias siguientes en el orden especificado.

```
MOVE identifier-1 to record-name-1.
RELEASE record-name-1.
```

El MOVE se realiza de acuerdo con las reglas para la sentencia MOVE sin la frase CORRESPONDIENTE.

identifier-1

identifier-1 debe hacer referencia a uno de los elementos siguientes:

- Una entrada en WORKING-STORAGE SECTION, LOCAL-STORAGE SECTION o LINKAGE SECTION
- Una descripción de registro para otro archivo abierto anteriormente
- Una función alfanumérica o nacional.

identifier-1 debe ser un elemento de envío válido con *record-name-1* como elemento de recepción de acuerdo con las reglas de la sentencia MOVE.

identifier-1 y *record-name-1* no deben hacer referencia a la misma área de almacenamiento.

Después de ejecutar la sentencia RELEASE, la información sigue estando disponible en *identifier-1*. (Consulte “Frases INTO y FROM” en la página 290 en "Instalaciones de proceso comunes".)

Si la sentencia RELEASE se ejecuta sin especificar la entrada SD para *file-name-1* en una cláusula SAME RECORD AREA, la información de *record-name-1* ya no estará disponible.

Si la entrada SD se especifica en una cláusula SAME RECORD AREA, *record-name-1* sigue estando disponible como registro de los otros archivos nombrados en dicha cláusula.

Cuando se especifica FROM *identifier-1* , la información sigue estando disponible en *identifier-1*.

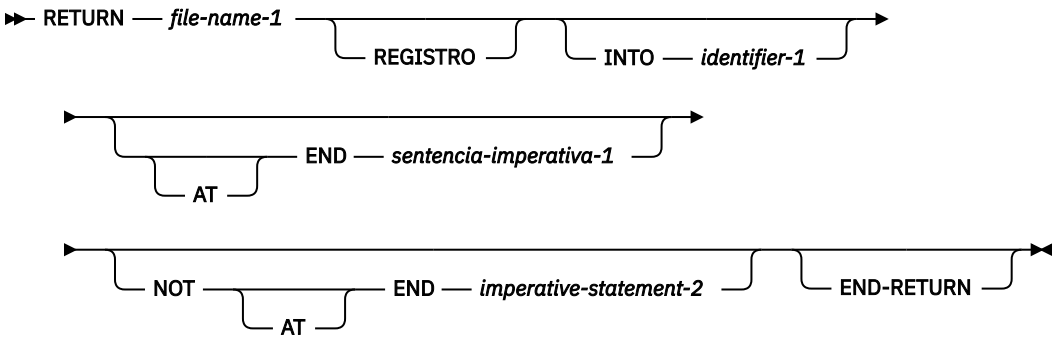
Cuando el control pasa desde INPUT PROCEDURE, el archivo de clasificación consta de todos los registros colocados en él mediante la ejecución de sentencias RELEASE.

RETURN, sentencia

La sentencia RETURN transfiere registros de la fase final de una operación de clasificación o fusión a un OUTPUT PROCEDURE.

La sentencia RETURN sólo puede utilizarse dentro del rango de un OUTPUT PROCEDURE asociado con una sentencia SORT o MERGE.

Formato: sentencia RETURN



Dentro de un OUTPUT PROCEDURE, debe especificarse al menos una sentencia RETURN.

Cuando se ejecuta la sentencia RETURN, el siguiente registro de *file-name-1* pasa a estar disponible para que lo procese OUTPUT PROCEDURE.

file-name-1

Debe describirse en una entrada DATA DIVISION SD.

Si hay más de una descripción de registro asociada con *file-name-1*, estos registros comparten automáticamente el mismo almacenamiento; es decir, el área se vuelve a definir implícitamente. Después de la ejecución de la sentencia RETURN, sólo están disponibles los contenidos del registro

actual. Si algún elemento de datos se encuentra más allá de la longitud del registro actual, su contenido no está definido.

Frase INTO

Cuando sólo hay una descripción de registro asociada a *file-name-1* o todos los registros y el elemento de datos al que hace referencia *identifier-1* describen un elemento alfanumérico elemental o un elemento de grupo alfanumérico, el resultado de la ejecución de una sentencia RETURN con la frase INTO es equivalente a la aplicación de las reglas siguientes en el orden especificado:

- La ejecución de la misma sentencia RETURN sin la frase INTO.
- El registro actual se mueve del área de registro al área especificada por *identifier-1* de acuerdo con las reglas para la sentencia MOVE sin la frase CORRESPONDIENTE. El tamaño del registro actual viene determinado por las reglas especificadas para la cláusula RECORD. Si la entrada de descripción de archivo contiene una cláusula RECORD IS VARIABLE, el movimiento implícito es un movimiento de grupo. La sentencia MOVE implícita no se produce si la ejecución de la sentencia RETURN no ha sido satisfactoria. Cualquier subíndice o modificación de referencia asociada con *identifier-1* se evalúa después de que se haya leído el registro e inmediatamente antes de que se mueva al elemento de datos. El registro está disponible tanto en el área de registro como en el elemento de datos al que hace referencia *identifier-1*.

Cuando hay varias descripciones de registro asociadas con *file-name-1* y no todas describen un elemento de grupo alfanumérico o un elemento alfanumérico elemental, se aplican las reglas siguientes:

1. Si el archivo al que hace referencia *file-name-1* contiene registros de longitud variable, se produce un movimiento de grupo.
2. Si el archivo al que hace referencia *file-name-1* contiene registros de longitud fija, se produce un movimiento de acuerdo con las reglas de una sentencia MOVE utilizando, como descripción de campo de envío, el registro que especifica el mayor número de posiciones de caracteres. Si existe más de un registro de este tipo, el registro de campo de envío seleccionado será el que aparezca primero bajo la descripción de *file-name-1*.

identifier-1 debe ser un campo de recepción válido para la entrada de descripción de registro de envío seleccionada de acuerdo con las reglas de la sentencia MOVE.

Las áreas de registro asociadas con *file-name-1* y *identifier-1* no deben ser la misma área de almacenamiento.

frases AT END

La sentencia imperativa especificada en la frase AT END se ejecuta después de que se hayan devuelto todos los registros de *file-name-1*. No se pueden ejecutar más sentencias RETURN como parte del procedimiento de salida actual.

Si no se produce una condición de finalización durante la ejecución de una sentencia RETURN, después de que el registro esté disponible y después de ejecutar cualquier movimiento implícito resultante de la presencia de una frase INTO, el control se transfiere a la sentencia imperativa especificada por la frase NOT AT END. Si se produce una condición de finalización, el control se transfiere al final de la sentencia RETURN.

Frase END-RETURN

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia RETURN. END-RETURN permite que una sentencia RETURN condicional se anide en otra sentencia condicional. END-RETURN también se puede utilizar con una sentencia RETURN imperativa.

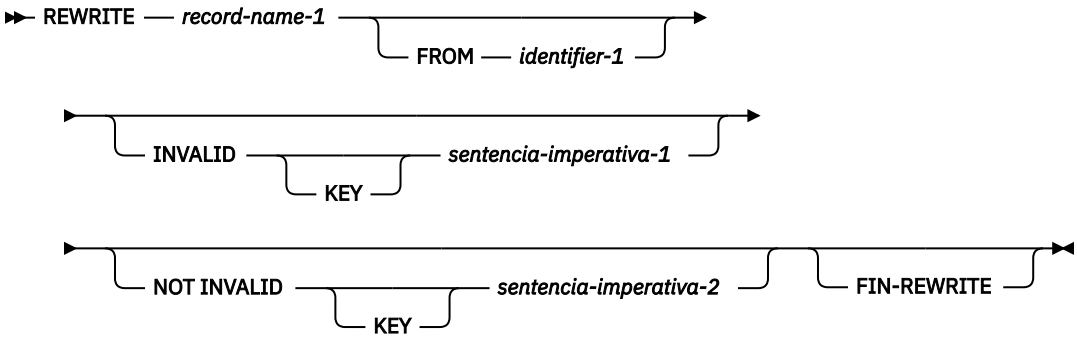
Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

Sentencia REWRITE

La sentencia REWRITE sustituye lógicamente un registro existente en un archivo de acceso directo. Cuando se ejecuta la sentencia REWRITE, el archivo de acceso directo asociado debe estar abierto en modalidad I-O.

La sentencia REWRITE no está soportada para archivos secuenciales de línea.

Formato: sentencia REWRITE



record-name-1
Debe ser el nombre de un registro lógico en una entrada DATA DIVISION FD. El nombre de registro puede calificarse.

Frase FROM
El resultado de la ejecución de la sentencia REWRITE con la frase FROM *identifier-1* es equivalente a la ejecución de las sentencias siguientes en el orden especificado.

```
MOVE identifier-1 TO record-name-1.
REWRITE record-name-1
```

El MOVE se realiza de acuerdo con las reglas para la sentencia MOVE sin la frase CORRESPONDIENTE.

identifier-1
identifier-1 puede hacer referencia a uno de los elementos siguientes:

- Una descripción de registro para otro archivo abierto anteriormente
- Una función alfanumérica o nacional
- Un elemento de datos definido en WORKING-STORAGE SECTION, LOCAL-STORAGE SECTION o LINKAGE SECTION

identifier-1 debe ser un elemento de envío válido con *record-name-1* como elemento de recepción de acuerdo con las reglas de la sentencia MOVE.

identifier-1 y *record-name-1* no deben hacer referencia a la misma área de almacenamiento.

Después de ejecutar la sentencia REWRITE, la información sigue estando disponible en *identifier-1* (“Frases INTO y FROM” en la página 290 en "Recursos de proceso comunes").

Frases INVALID KEY

Existe una condición INVALID KEY cuando:

- La modalidad de acceso es secuencial y el valor contenido en la CLAVE DE REGISTRO principal del registro que se va a sustituir no es igual al valor del elemento de datos CLAVE DE REGISTRO principal del último registro recuperado del archivo
- El valor contenido en la CLAVE DE REGISTRO primo no es igual al de ningún registro del archivo

- El valor de un elemento de datos ALTERNATE RECORD KEY para el que no se ha especificado DUPLICATES es igual al de un registro que ya está en el archivo

Para obtener detalles sobre el proceso de claves no válidas, consulte Condición de clave no válida.

frase END-REWRITE

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia REWRITE. END-REWRITE permite que una sentencia REWRITE condicional se anide en otra sentencia condicional. END-REWRITE también se puede utilizar con una sentencia REWRITE imperativa.

Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

Reutilización de un registro lógico

Después de la ejecución satisfactoria de una sentencia REWRITE, el registro lógico ya no está disponible en *record-name-1* a menos que el archivo asociado se denomine en una cláusula SAME RECORD AREA (en cuyo caso, el registro también está disponible como registro de los otros archivos nombrados en la cláusula SAME RECORD AREA).

El indicador de posición de archivo no se ve afectado por la ejecución de la sentencia REWRITE.

Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, la clave de estado de archivo asociada se actualiza cuando se ejecuta la sentencia REWRITE.

Archivos secuenciales

Para los archivos en modalidad de acceso secuencial, la última sentencia de entrada/salida anterior ejecutada para este archivo debe ser una sentencia READ ejecutada correctamente. Cuando se ejecuta la sentencia REWRITE, el registro recuperado por dicha sentencia READ se sustituye lógicamente.

El número de posiciones de caracteres en *record-name-1* debe ser igual al número de posiciones de caracteres en el registro que se está sustituyendo.

La frase INVALID KEY no debe especificarse para un archivo con organización secuencial. Se puede especificar un procedimiento EXCEPTION/ERROR.

Archivos indexados

El número de posiciones de caracteres en *record-name-1* puede ser diferente del número de posiciones de caracteres en el registro que se está sustituyendo.

Cuando la modalidad de acceso es secuencial, el registro que se va a sustituir se especifica mediante el valor contenido en la CLAVE DE REGISTRO PRINCIPAL. Cuando se ejecuta la sentencia REWRITE, este valor debe ser igual al valor del elemento de datos de clave de registro principal en el último registro leído de este archivo.

Se puede omitir la frase INVALID KEY y un procedimiento EXCEPTION/ERROR aplicable.

Cuando la modalidad de acceso es aleatoria o dinámica, el registro que se va a sustituir se especifica mediante el valor contenido en RECORD KEY principal.

Los valores de los elementos de datos de CLAVE DE REGISTRO ALTERNATIVA en el registro reescrito pueden diferir de los del registro que se está sustituyendo. El sistema garantiza que el acceso posterior al registro se pueda basar en cualquiera de las claves de registro.

Si existe una condición de clave no válida, la ejecución de la sentencia REWRITE no es satisfactoria, la operación de actualización no tiene lugar y los datos de *record-name-1* no se ven afectados. (Consulte Condición de clave no válida en "Instalaciones de proceso comunes".)

Archivos relativos

El número de posiciones de caracteres en *record-name-1* puede ser diferente del número de posiciones de caracteres en el registro que se está sustituyendo.

Para archivos relativos en modalidad de acceso secuencial, no debe especificarse la frase INVALID KEY. Se puede especificar un procedimiento EXCEPTION/ERROR.

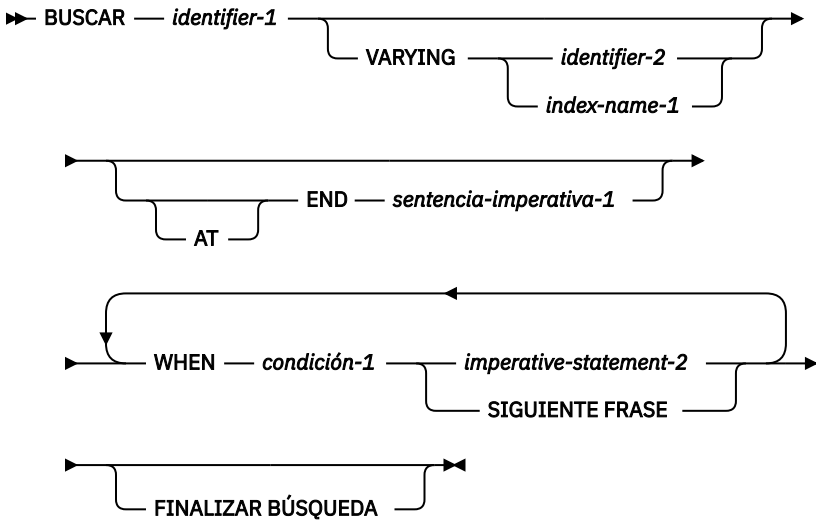
Para archivos relativos en modalidad de acceso aleatorio o dinámico, se puede especificar la frase INVALID KEY o un procedimiento EXCEPTION/ERROR aplicable. Ambos se pueden omitir.

Cuando la modalidad de acceso es aleatoria o dinámica, el registro que se va a sustituir se especifica en el elemento de datos RELATIVE KEY. Si el archivo no contiene el registro especificado, existe una condición de clave no válida y, si se especifica, se ejecuta la sentencia imperativa INVALID KEY. (Consulte Condición de clave no válida en "Instalaciones de proceso comunes".) La operación de actualización no tiene lugar y los datos del nombre de registro no se ven afectados.

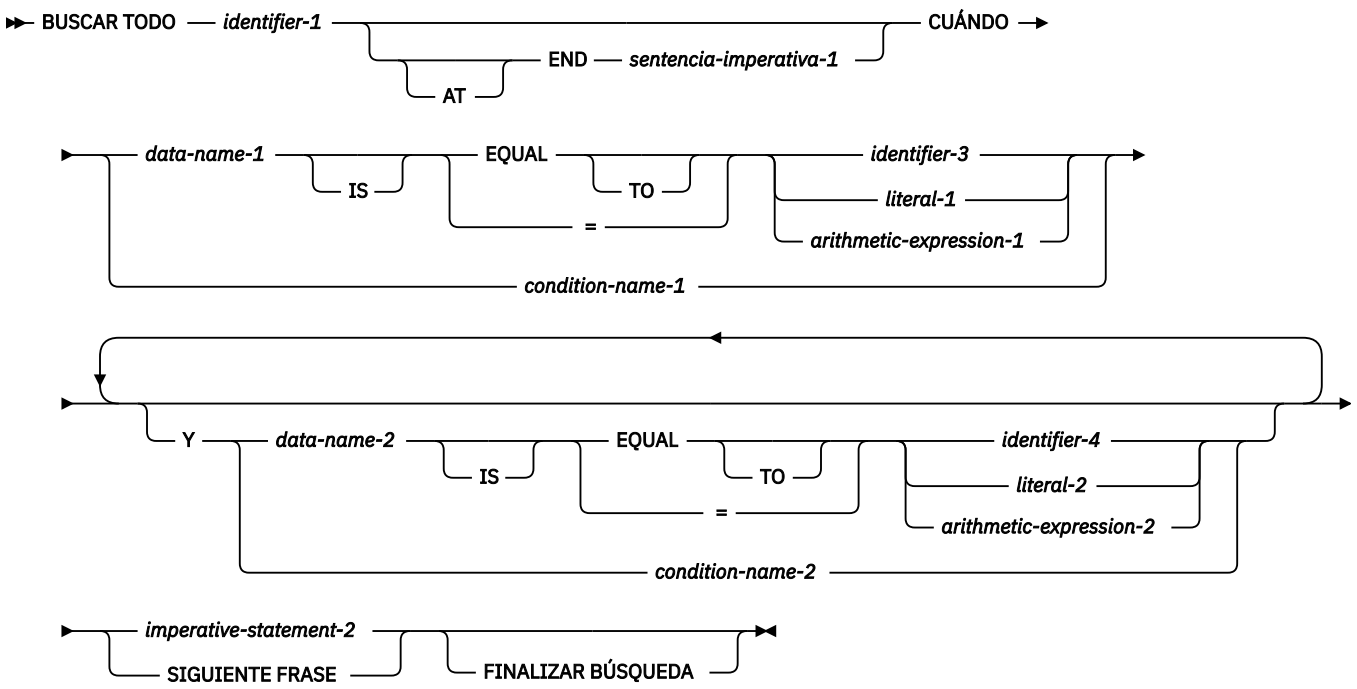
Sentencia SEARCH

La sentencia SEARCH busca en una tabla un elemento que satisfaga la condición especificada y ajusta el índice asociado para indicar ese elemento.

Formato 1: sentencia SEARCH para búsqueda en serie



Formato 2: sentencia SEARCH para búsqueda binaria



Utilice el formato 1 (búsqueda en serie) cuando la tabla en la que desea buscar no se haya ordenado. Utilice el formato 1 para buscar en una tabla ordenada cuando desee realizar una búsqueda en serie a través de la tabla o cuando desee controlar subíndices o índices.

Utilice el formato 2 (búsqueda binaria) cuando desee buscar de forma eficaz en todas las apariciones de una tabla. La tabla debe haberse ordenado previamente, y puede ordenar la tabla con el formato 2 sentencia SORT.

Frases AT END y WHEN

Después de ejecutar *imperative-statement-1* o *imperative-statement-2* , el control pasa al final de la sentencia SEARCH, a menos que *imperative-statement-1* o *imperative-statement-2* termine con una sentencia GO TO.

La función de la frase AT END es la misma para una búsqueda en serie y una búsqueda binaria.

SIGUIENTE FRASE

NEXT STATEMENT transfiere el control a la primera sentencia después del punto de separación más próximo.

Cuando se especifica NEXT STATEMENT con END-SEARCH, el control no pasa a la sentencia que sigue a END-SEARCH. En su lugar, el control pasa a la sentencia después del periodo siguiente más cercano.

Para la sentencia format-2 SEARCH ALL, no es necesario *imperative-statement-2* ni NEXT STATEMENT. Sin ellos, la sentencia SEARCH establece el índice en el valor de la tabla que ha coincidido con la condición.

La función de la frase NEXT ORACIÓN es la misma para una búsqueda en serie y una búsqueda binaria.

Frase END-SEARCH

Este terminador de ámbito explícito delimita el ámbito de la sentencia SEARCH. END-SEARCH permite que una sentencia SEARCH condicional se anide en otra sentencia condicional.

Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

La función de END-SEARCH es la misma para una búsqueda en serie y una búsqueda binaria.

Búsqueda en serie

El tema proporciona información sobre cómo utilizar la sentencia SEARCH para la búsqueda en serie.

***identifier-1* (búsqueda en serie)**

identifier-1 identifica la tabla que se debe buscar. *identifier-1* hace referencia a todas las apariciones dentro de esa tabla.

La entrada de descripción de datos para *identifier-1* debe contener una cláusula OCCURS.

La entrada de descripción de datos para *identifier-1* debe contener una cláusula OCCURS con la frase INDEXED BY, pero se puede buscar en una tabla utilizando un índice definido para una tabla diferente descrita adecuadamente.

identifier-1 puede hacer referencia a un elemento de datos que está subordinado a un elemento de datos que se describe con una cláusula OCCURS (es decir, *identifier-1* puede ser una tabla subordinada dentro de una tabla multidimensional). En este caso, las entradas de descripción de datos deben especificar una frase INDEXED BY para cada dimensión de la tabla.

identifier-1 no debe tener suscripciones ni modificarse mediante referencia.

AT END

La condición que existe cuando la operación de búsqueda termina sin satisfacer la condición especificada en ninguna de las frases WHEN asociadas.

Antes de ejecutar una búsqueda en serie, debe establecer el valor del primer (o único) índice asociado con *identifier-1* (el índice de búsqueda) para indicar la aparición inicial de la búsqueda.

Antes de utilizar una búsqueda en serie en una tabla multidimensional, también debe establecer el valor del índice para cada dimensión superordenada.

La sentencia SEARCH sólo modifica el valor en el índice de búsqueda y, si se especifica la frase VARYING, el valor en *index-name-1* o *identifier-2*. Por lo tanto, para buscar una tabla completa bidimensional a siete dimensiones, debe ejecutar una sentencia SEARCH para cada dimensión. En las frases WHEN, debe especificar los índices para todas las dimensiones. Antes de ejecutar cada sentencia SEARCH, debe inicializar los índices asociados con sentencias SET.

La sentencia SEARCH ejecuta una búsqueda en serie que empieza en el valor actual del índice de búsqueda.

Cuando empieza la búsqueda, si el valor del índice asociado con *identifier-1* no es mayor que el número de aparición más alto posible, se llevan a cabo las acciones siguientes:

- Las condiciones de la frase WHEN se evalúan en el orden en que se escriben.
- Si no se cumple ninguna de las condiciones, el índice para *identifier-1* se aumenta para que se corresponda con el siguiente elemento de tabla y el paso 1 se repite.
- Si tras la evaluación se cumple una de las condiciones WHEN, la búsqueda termina inmediatamente y se ejecuta la *imperative-statement-2* asociada con dicha condición. El índice apunta al elemento de tabla que ha satisfecho la condición. Si se especifica NEXT STATEMENT, el control pasa a la sentencia que sigue al punto más próximo.
- Si se alcanza el final de la tabla (es decir, el valor del índice incrementado es mayor que el número de aparición más alto posible) sin que se cumpla la condición WHEN, la búsqueda se termina.

Si, cuando empieza la búsqueda, el valor del nombre de índice asociado con *identifier-1* es mayor que el número de aparición más alto posible, la búsqueda termina inmediatamente.

Cuando la búsqueda termina, si se especifica la frase AT END, se ejecuta *imperative-statement-1* . Si se omite la frase AT END, el control pasa a la siguiente sentencia después de la sentencia SEARCH.

Ejemplo: búsqueda de serie multidimensional

El siguiente fragmento de código muestra una búsqueda de la dimensión interna (tabla C) en la tercera aparición dentro de la tabla superordenada (tabla R):

```
     . . .
     Working-storage section.
     1 G.
       2 R occurs 10 indexed by Rindex.
         3 C occurs 10 ascending key X indexed by Cindex.
           4 X pic 99.
     1 Arg pic 99 value 34.
     Procedure division.
     . . .
* To search within occurrence 3 of table R, set its index to 3
* To search table C beginning at occurrence 1, set its index to 1
     Set Rindex to 3
     Set Cindex to 1
* In the SEARCH statement, specify C without indexes
     Search C
* Specify indexes for both dimensions in the WHEN phrase
       when X(Rindex Cindex) = Arg
       display "Found " X(Rindex Cindex)
     End-search
     . . .
```

Frase VARYING

index-name-1
Se aplica una de las acciones siguientes:

- Si *index-name-1* es un índice para *identifier-1*, este índice se utiliza para la búsqueda. De lo contrario, se utiliza el primer (o único) nombre de índice.
- Si *index-name-1* es un índice para otro elemento de tabla, se utiliza el primer (o único) nombre-índice para *identifier-1* para la búsqueda; el número de aparición representado por *index-name-1* se incrementa en la misma cantidad que el nombre-índice de búsqueda y al mismo tiempo.

Cuando se omite la frase VARYING *index-name-1* , se utiliza el primer (o único) nombre-índice para *identifier-1* para la búsqueda.

Si se utiliza la indexación para buscar una tabla sin una frase INDEXED BY, los resultados correctos sólo se aseguran si tanto la tabla definida con el índice como la tabla definida sin el índice tienen elementos de tabla de la misma longitud y con el mismo número de apariciones.

Cuando el objeto de la frase VARYING es un nombre de índice para otro elemento de tabla, una sentencia SEARCH serie recorre dos elementos de tabla a la vez.

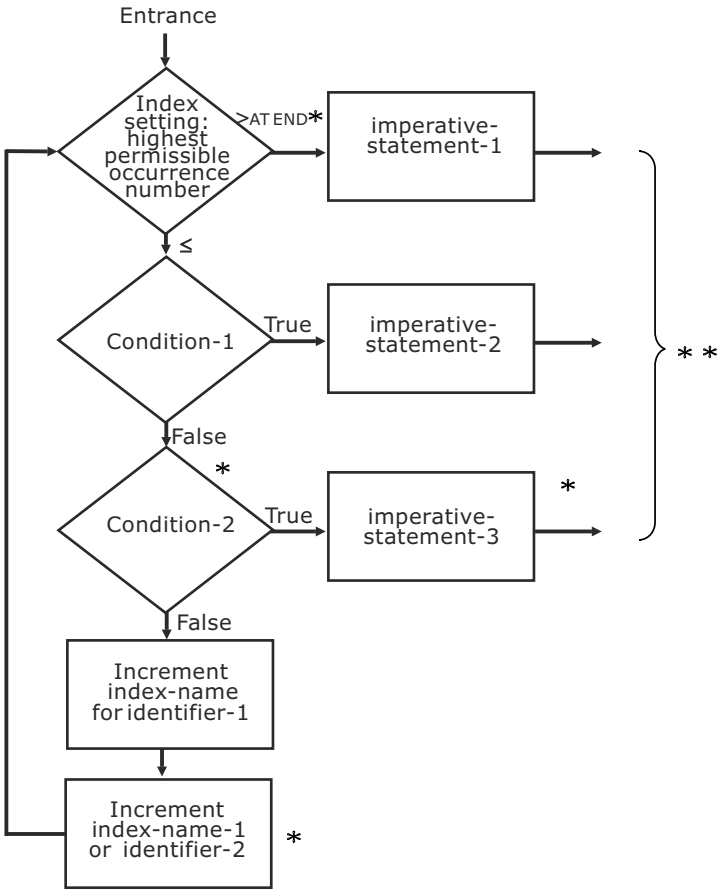
identifier-2
Debe ser un elemento de datos de índice o un elemento entero elemental. *identifier-2* no puede ser un campo de fecha con ventana. *identifier-2* no puede ser subíndice por el primer (o único) nombre de índice especificado para *identifier-1*. Durante la búsqueda, se aplica una de las acciones siguientes:

- Si *identifier-2* es un elemento de datos de índice, siempre que se aumente el índice de búsqueda, el elemento de datos de índice especificado se incrementará simultáneamente en la misma cantidad.
- Si *identifier-2* es un elemento de datos entero, siempre que se aumente el índice de búsqueda, el elemento de datos especificado se aumentará simultáneamente en 1.

Frase WHEN (búsqueda en serie)

condition-1
Puede ser cualquier condición descrita en “Expresiones condicionales” en la página 250.

La figura siguiente ilustra una operación format-1 SEARCH que contiene dos frases WHEN.



*These operations are included only when called for in the statement.
* *Control transfers to the next sentence, unless the imperative statement ends with a GO TO statement.

Búsqueda binaria

El tema proporciona información sobre cómo utilizar la sentencia SEARCH para la búsqueda binaria.

***identifier-1* (búsqueda binaria)**

identifier-1 identifica la tabla que se debe buscar. *identifier-1* hace referencia a todas las apariciones dentro de esa tabla.

La entrada de descripción de datos para *identifier-1* debe contener una cláusula OCCURS con las frases INDEXED BY y KEY IS.

identifier-1 puede hacer referencia a un elemento de datos que está subordinado a un elemento de datos que contiene una cláusula OCCURS (es decir, *identifier-1* puede ser una tabla subordinada dentro de una tabla multidimensional). En este caso, la entrada de descripción de datos debe especificar una frase INDEXED BY para cada dimensión de la tabla.

identifier-1 no debe tener suscripciones ni modificarse mediante referencia.

AT END

Condición que existe cuando la operación de búsqueda termina sin satisfacer las condiciones especificadas en la frase WHEN.

La sentencia SEARCH ALL ejecuta una búsqueda binaria. Las sentencias SET no necesitan inicializar el índice asociado con *identifier-1* (el índice de búsqueda). El índice de búsqueda se varía durante la operación de búsqueda de modo que su valor no sea en ningún momento menor que el valor del primer elemento de tabla, ni nunca mayor que el valor del último elemento de tabla. El índice utilizado es siempre el asociado con el primer nombre de índice especificado en la cláusula OCCURS.

Antes de utilizar una búsqueda binaria en una tabla multidimensional, debe ejecutar sentencias SET para establecer el valor del índice para cada dimensión superordenada.

La sentencia SEARCH sólo modifica el valor del índice de búsqueda. Por lo tanto, para buscar una tabla completa bidimensional a siete dimensiones, debe ejecutar una sentencia SEARCH para cada dimensión. En las frases WHEN, debe especificar los índices para todas las dimensiones.

Si la búsqueda finaliza sin que se cumpla la condición WHEN y se especifique la frase AT END, se ejecuta *imperative-statement-1* . Si se omite la frase AT END, el control pasa a la siguiente sentencia después de la sentencia SEARCH.

Los resultados de una operación SEARCH ALL sólo son predecibles cuando:

- Los datos de la tabla se ordenan en orden CLAVE ASCENDENTE o CLAVE DESCENDENTE
- El contenido de las claves ASCENDING o DESCENDING especificadas en la cláusula WHEN proporcionan una referencia de tabla exclusiva.

Frase WHEN (búsqueda binaria)

Si se especifica una condición de relación en la frase WHEN, la evaluación de la relación se basa en la USAGE del elemento de datos al que hace referencia *data-name-1*. El argumento de búsqueda se mueve a un elemento de datos temporal con el mismo USAGE que *data-name-1*, y este elemento de datos temporal se utiliza para las operaciones de comparación asociadas con SEARCH. Si data-name-1 es un elemento numérico, el elemento de datos temporal se firma o no se firma de forma coherente con la presencia o ausencia de un signo en la descripción de datos de data-name-1. Si el argumento de búsqueda está firmado y data-name-1 no está firmado, el signo se eliminará del argumento de búsqueda antes de que se realice la comparación.

Si la frase WHEN no se puede satisfacer para ningún valor del índice dentro de este rango, la búsqueda no es satisfactoria. El control se pasa a *imperative-statement-1* de la frase AT END, cuando se especifica, o a la siguiente sentencia después de la sentencia SEARCH. En cualquier caso, el valor final del índice no es predecible.

Si se puede satisfacer la frase WHEN, el control pasa a *imperative-statement-2*, si se especifica, o a la siguiente frase ejecutable si se especifica la frase NEXT ORACIÓN. El índice contiene el valor que indica la aparición que permitió que se cumplieran las condiciones WHEN.

Después de ejecutar *imperative-statement-2* , el control pasa al final de la sentencia SEARCH, a menos que *imperative-statement-2* termine con una sentencia GO TO.

condition-name-1 , condition-name-2
Cada nombre de condición especificado debe tener un único valor y cada uno debe estar asociado con un elemento de datos ASCENDING KEY o DESCENDING KEY para este elemento de tabla.

data-name-1 , data-name-2
Debe especificar un elemento de datos ASCENDING KEY o DESCENDING KEY en el elemento de tabla al que hace referencia *identifier-1* y debe tener un subíndice del primer nombre de índice asociado con *identifier-1*. Cada nombre de datos se puede calificar.

data-name-1 debe ser un operando válido para la comparación con *identifier-3*, *literal-1*, o *arithmetic-expression-1* de acuerdo con las reglas de comparación.

data-name-2 debe ser un operando válido para la comparación con *identifier-4*, *literal-2*, o *arithmetic-expression-2* de acuerdo con las reglas de comparación.

data-name-1 y *data-name-2* no pueden hacer referencia:

- Elementos de datos de coma flotante
- Elementos de grupo que contienen elementos de datos de aparición variable
- Campos de fecha con ventana

identifier-3 , identifier-4
No debe ser un elemento de datos ASCENDING KEY o DESCENDING KEY para *identifier-1* o un elemento que esté subíndice por el primer nombre de índice para *identifier-1*.

identifier-3 y *identifier-4* no pueden ser elementos de datos definidos con ninguno de los usos POINTER, FUNCTION-POINTER, o PROCEDURE-POINTER.

identifier-3 y *identifier-4* no pueden ser campos de fecha con ventana.

Si *identifier-3* o *literal-1* es de clase nacional, *data-name-1* debe ser de clase nacional.

Si *identifier-4* o *literal-2* es de clase nacional, *data-name-2* debe ser de clase nacional.

literal-1* , *literal-2
literal-1 o *literal-2* deben ser un operando válido para la comparación con *data-name-1* o *data-name-2*, respectivamente.

expresión-aritmética
Puede ser cualquiera de las expresiones definidas bajo "Expresiones aritméticas" en la página 246, con la restricción siguiente: cualquier identificador en *expresión-aritmética* no debe ser un elemento de datos ASCENDING KEY o DESCENDING KEY para *identifier-1* o un elemento al que el primer nombre de índice para *identifier-1*haya asignado un subíndice.

Cuando se especifica un elemento de datos ASCENDING KEY o DESCENDING KEY, de forma explícita o implícita, en la frase WHEN, también deben especificarse todos los nombres de datos ASCENDING KEY o DESCENDING KEY anteriores para *identifier-1* .

Consideraciones sobre sentencias de búsqueda

El tema enumera consideraciones sobre el uso de la sentencia SEARCH.

Los elementos de datos de índice no se pueden utilizar como subíndices, debido a las restricciones sobre la referencia directa a los mismos.

Para garantizar la ejecución correcta de una sentencia SEARCH para una tabla de longitud variable, asegúrese de que el objeto de la cláusula OCCURS DEPENDING ON (*data-name-1*) contiene un valor que especifica la longitud actual de la tabla.

El ámbito de una sentencia SEARCH puede terminar con cualquiera de los elementos siguientes:

- Una frase END-SEARCH en el mismo nivel de anidamiento
- Un punto separador
- Una frase ELSE o END-IF asociada con una sentencia IF anterior

sentencia SET

La sentencia SET se utiliza para realizar una operación tal como se describe en este tema.

Las operaciones son:

- Colocación de valores asociados con elementos de tabla en índices asociados con nombres de índice
- Incremento o disminución de un número de aparición
- Establecer el estado de un conmutador externo en ON u OFF
- Mover datos a nombres de condición para que las condiciones sean verdaderas
- Establecimiento de elementos de datos USAGE POINTER en una dirección de datos
- Establecimiento de elementos de datos USAGE PROCEDURE-POINTER en una dirección de entrada
- Establecimiento de elementos de datos USAGE FUNCTION-POINTER en una dirección de entrada
- Establecimiento y consulta de las categorías de entorno local del entorno local actual

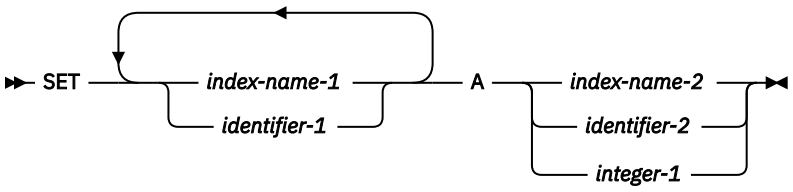
Los nombres de índice están relacionados con una tabla determinada mediante la frase INDEXED BY de la cláusula OCCURS; no están más definidos en el programa.

Cuando los campos de envío y recepción de una sentencia SET comparten parte de su almacenamiento (es decir, los operandos se solapan), el resultado de la ejecución de dicha sentencia SET no está definido.

Formato 1: SET para el manejo básico de tablas

Cuando se ejecuta este formato de la sentencia SET, el valor actual del campo receptor se sustituye por el valor del campo emisor (con conversión).

Formato 1: sentencia SET para el manejo básico de tablas



index-name-1
Campo de recepción.

Debe nombrar un índice especificado en la frase INDEXED BY de una cláusula OCCURS.

identifier-1
Campo de recepción.

Debe nombrar un elemento de datos de índice o un elemento entero numérico elemental. Un campo de recepción no puede ser un campo de fecha con ventana.

index-name-2
Campo de envío.

Debe nombrar un índice especificado en la frase INDEXED BY de una cláusula OCCURS. El valor del índice antes de que se ejecute la sentencia SET debe corresponder a un número de aparición de su tabla asociada.

identifier-2
Campo de envío.

Debe nombrar un elemento de datos de índice o un elemento entero numérico elemental. Un campo de envío no puede ser un campo de fecha con ventana.

integer-1
Campo de envío.

Debe ser un entero positivo.

La tabla siguiente muestra combinaciones válidas de campos de envío y recepción en una sentencia format-1 SET.

Tabla 56. ***Envío y recepción de campos para la sentencia format-1 SET***

Campo de envío	**nombre-índice campo de recepción**	**Campo de recepción de elemento de datos de índice**	**Campo de recepción de elemento de datos entero**
Nombre-índice *	Válido	** válido	Válido
Elemento de datos de índice *	** válido	** válido	No válido
Elemento de datos entero	Válido	No válido	No válido
Literal entero	Válido	No válido	No válido

Tabla 56. ***Envío y recepción de campos para la sentencia format-1 SET*** *(continuación)*			
Campo de envío	**nombre-índice campo de recepción**	**Campo de recepción de elemento de datos de índice**	**Campo de recepción de elemento de datos entero**
*Un nombre de índice hace referencia a un índice especificado en la frase INDEXED BY de una cláusula OCCURS. Un elemento de datos de índice se define con la cláusula USAGE IS INDEX. **No se realiza ninguna conversión.			

Los campos de recepción se actúan en el orden de izquierda a derecha en el que se especifican. Cualquier subíndice o indexación asociada con *identifier-1* se evalúa inmediatamente antes de que se actúe sobre ese campo de recepción.

El valor utilizado para el campo de envío es el valor al principio de la ejecución de la sentencia SET.

El valor de un índice después de la ejecución de una sentencia SEARCH o PERFORM puede no estar definido; por lo tanto, utilice una sentencia SET format-1 para reinicializar dichos índices antes de intentar otras operaciones de manejo de tablas.

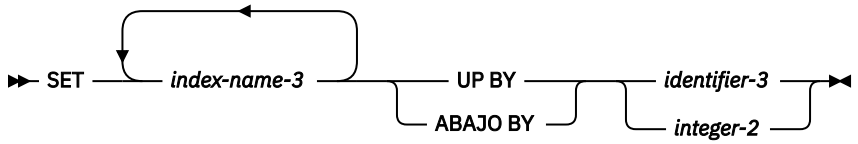
Si *index-name-2* es para una tabla que tiene un elemento subordinado que contiene una cláusula OCCURS DEPENDING ON, los valores no definidos se pueden recibir en *identifier-1*.

Para obtener más información sobre OCCURS DEPENDING ON complejo, consulte *SE PRODUCE COMPLEJO EN FUNCIÓN DE* en la publicación *COBOL for Linux en x86 Guía de programación*.

Formato 2: SET para ajustar índices

Cuando se ejecuta esta forma de la sentencia SET, el valor del índice de recepción aumenta (UP BY) o disminuye (DOWN BY) en un valor que corresponde al valor del campo de envío.

Formato 2: sentencia SET para ajustar índices



El *campo de recepción* es un índice especificado por *index-name-3*. El valor de índice antes y después de la ejecución de la sentencia SET debe corresponder a un número de aparición en una tabla asociada.

El *campo de envío* se puede especificar como *identifier-3*, que debe ser un elemento de datos entero elemental, o como *integer-2*, que debe ser un entero distinto de cero. *identifier-3* no puede ser un campo de fecha con ventana.

Cuando se ejecuta la sentencia SET format-2 , El contenido del campo de recepción se incrementa (UP BY) o disminuye (DOWN BY) en un valor que corresponde al número de apariciones representadas por el valor de *identifier-3* o *integer-2*. Los campos de recepción se actúan en el orden de izquierda a derecha en el que se especifican. El valor del campo de incremento o disminución al principio de la ejecución de la sentencia SET se utiliza para todos los campos de recepción.

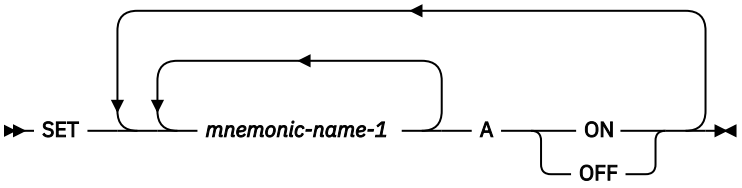
Si *index-name-3* es para una tabla que tiene un elemento subordinado que contiene una cláusula OCCURS DEPENDING ON, y si el objeto ODO se cambia antes de ejecutar una sentencia format-2 SET, entonces *index-name-3* no puede contener un valor que corresponda a un número de aparición de su tabla asociada.

Para obtener más información sobre OCCURS DEPENDING ON complejo, consulte *SE PRODUCE COMPLEJO EN FUNCIÓN DE* en la publicación *COBOL for Linux en x86 Guía de programación*.

Formato 3: SET para conmutadores externos

Cuando se ejecuta esta forma de la sentencia SET, el estado de cada conmutador externo asociado con el nombre mnemotécnico especificado se activa o desactiva.

Formato 3: sentencia SET para conmutadores externos

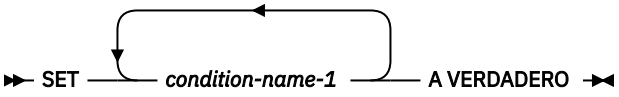


mnemonic-name-1
Debe estar asociado con un conmutador externo, cuyo estado se puede modificar.

Formato 4: SET para nombres de condición

Cuando se ejecuta esta forma de la sentencia SET, el valor asociado con un nombre-condición se coloca en su variable condicional de acuerdo con las reglas de la cláusula VALUE.

Formato 4: sentencia SET para nombres de condición



condition-name-1
Debe estar asociado con una variable condicional.

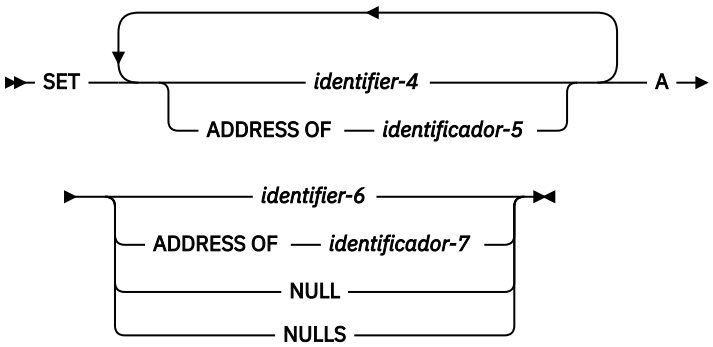
Si se especifica más de un literal en la cláusula VALUE de *condition-name-1*, su variable condicional asociada se establece igual al primer literal.

Si se especifican varios nombres de condición, los resultados son los mismos que si se hubiera escrito una sentencia SET separada para cada nombre de condición en el mismo orden en el que se han especificado en la sentencia SET.

Formato 5: SET para elementos de datos USAGE IS POINTER

Cuando se ejecuta este formato de la sentencia SET, el valor actual del campo receptor se sustituye por el valor de dirección contenido en el campo de envío.

Formato 5: sentencia SET para elementos de datos USAGE IS POINTER



identifier-4
Campos de recepción.

Debe definirse como USAGE IS POINTER .

ADDRESS OF *identifier-5*
Campos de recepción.

identifier-5 debe ser level-01 o level-77 elementos definidos en LINKAGE SECTION. Las direcciones de estos elementos se establecen en el valor del operando especificado en la frase TO.

identifier-5 no debe ser modificado por referencia.

identifier-6
Campo de envío.

Debe definirse como USAGE IS POINTER .

No puede contener una dirección dentro de la propia WORKING-STORAGE SECTION, FILE SECTION o LOCAL-STORAGE SECTION del programa.

ADDRESS OF *identifier-7*
Campo de envío.

identifier-7 debe nombrar un elemento de cualquier nivel excepto 66 o 88 en LINKAGE SECTION, WORKING-STORAGE SECTION o LOCAL-STORAGE SECTION. ADDRESS OF *identifier-7* contiene la dirección del identificador y no el contenido del identificador.

NULL, NULLS
Campo de envío.

Establece el campo de recepción para que contenga el valor de una dirección no válida.

La tabla siguiente muestra combinaciones válidas de campos de envío y recepción en una sentencia format-5 SET.

Tabla 57. ***Envío y recepción de campos para la sentencia SET format-5***

Campo de envío	**Campo de recepción USAGE IS POINTER**	**DIRECCIÓN DE campo de recepción**	**NULL/NULLS campo de recepción**
EL USO ES PUNTERO	Válido	Válido	No válido
DIRECCIÓN DE	Válido	Válido	No válido
NULL/NULLS	Válido	Válido	No válido

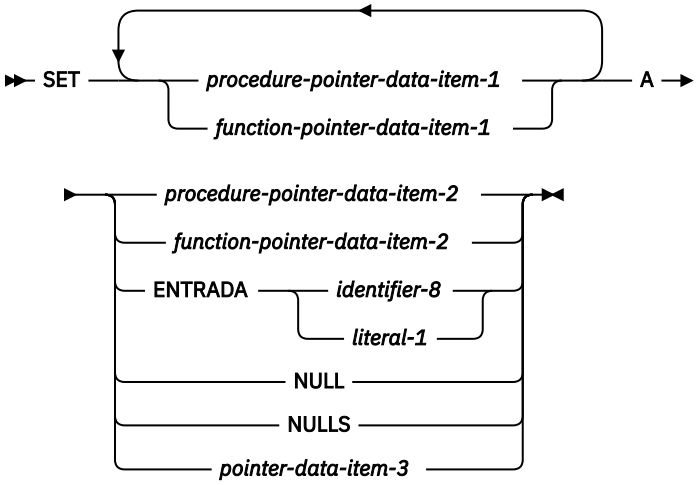
Formato 6: SET para elementos de datos de puntero de procedimiento y puntero de función

Cuando se ejecuta este formato de la sentencia SET, el valor actual del campo receptor se sustituye por el valor de dirección especificado por el campo emisor.

En tiempo de ejecución, los punteros de función y los punteros de procedimiento pueden hacer referencia a la dirección del punto de entrada primario de un programa COBOL, un punto de entrada alternativo en un programa COBOL o un punto de entrada en un programa no COBOL; o pueden ser NULL.

Los punteros de función COBOL se utilizan más fácilmente que los punteros de procedimiento para la interoperación con funciones C.

Formato 6: sentencia SET para punteros de procedimiento y punteros de función



procedure-pointer-data-item-1 , procedure-pointer-data-item-2
Debe describirse como USAGE IS PROCEDURE-POINTER. *procedure-pointer-data-item-1* es un campo de recepción; *procedure-pointer-data-item-2* es un campo de envío.

function-pointer-data-item-1 , function-pointer-data-item-2
Debe describirse como USAGE IS FUNCTION-POINTER. *function-pointer-data-item-1* es un campo receptor; *function-pointer-data-item-2* es un campo emisor.

identifier-8
Debe definirse como un elemento alfabético o alfanumérico de forma que el valor pueda ser un nombre de programa. Para obtener más información, consulte "Párrafo PROGRAM-ID" en la página 88. Para puntos de entrada en programas no COBOL, *identifier-8* puede contener los caracteres @, # y $.

literal-1
Debe ser alfanumérico y debe ajustarse a las reglas para la formación de nombres de programa. Para obtener detalles sobre las reglas de formación, consulte la discusión del nombre de programa en "Párrafo PROGRAM-ID" en la página 88.

identifier-8 o *literal-1* debe hacer referencia a uno de los siguientes tipos de puntos de entrada:

- El punto de entrada primario de un programa COBOL tal como lo define el párrafo PROGRAM-ID. El PROGRAM-ID debe hacer referencia al programa más externo de una unidad de compilación; no debe hacer referencia a un programa anidado.
- Punto de entrada alternativo de un programa COBOL tal como lo define una sentencia ENTRY de COBOL.
- Punto de entrada en un programa no COBOL.

El nombre de programa al que hace referencia SET ... La sentencia TO ENTRY puede verse afectada por la opción de compilador PGMNAME. Para obtener detalles, consulte *PGMNAME* en la publicación *COBOL for Linux en x86 Guía de programación*.

NULL, NULLS
Establece el campo de recepción para que contenga el valor de una dirección no válida.

pointer-data-item-3
Debe definirse con USAGE POINTER. Debe establecer *pointer-data-item-3* en un programa no COBOL para que apunte a un punto de entrada de programa válido.

Ejemplo de interoperatividad de COBOL/C

El ejemplo siguiente muestra una función CALL de COBOL a C que devuelve un puntero de función a un servicio, seguido de una CALL de COBOL al servicio:

```
IDENTIFICATION DIVISION.
PROGRAM-ID DEMO.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 FP USAGE FUNCTION-POINTER.
PROCEDURE DIVISION.
    CALL "c-function" RETURNING FP.
    CALL FP.
```

Formato 7: SET para elementos de datos USAGE OBJECT REFERENCE

Este formato no está soportado actualmente.

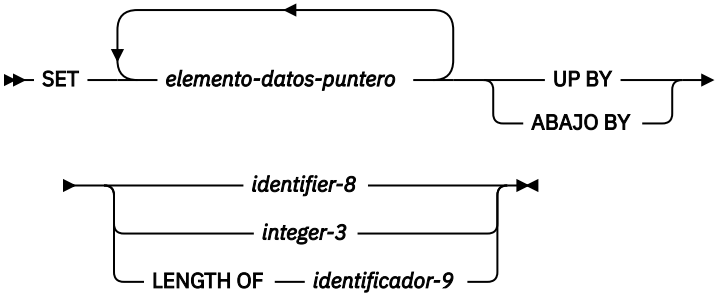
Formato 8: SET para longitud de elementos elementales de longitud dinámica

Este formato no está soportado actualmente.

Formato 9: SET para ajustar punteros

Cuando se ejecuta este formato de la sentencia SET, la dirección contenida en el elemento de datos de puntero aumenta (UP BY) o disminuye (DOWN BY) en un valor que corresponde al valor del campo de envío.

Formato 9: sentencia SET para ajustar punteros



elemento-datos-puntero
El campo de recepción debe ser un elemento de datos elemental con USAGE IS POINTER.

identifier-8
Este campo de envío debe ser un elemento de datos entero elemental.

identifier-8 no puede ser un elemento de datos de coma flotante.

integer-3
Este campo de envío debe ser un entero.

identifier-9
Para obtener más información sobre las reglas para *identifier-9*, consulte "LENGTH OF" en la página 21.

Formato 10: SET para categorías de entorno local

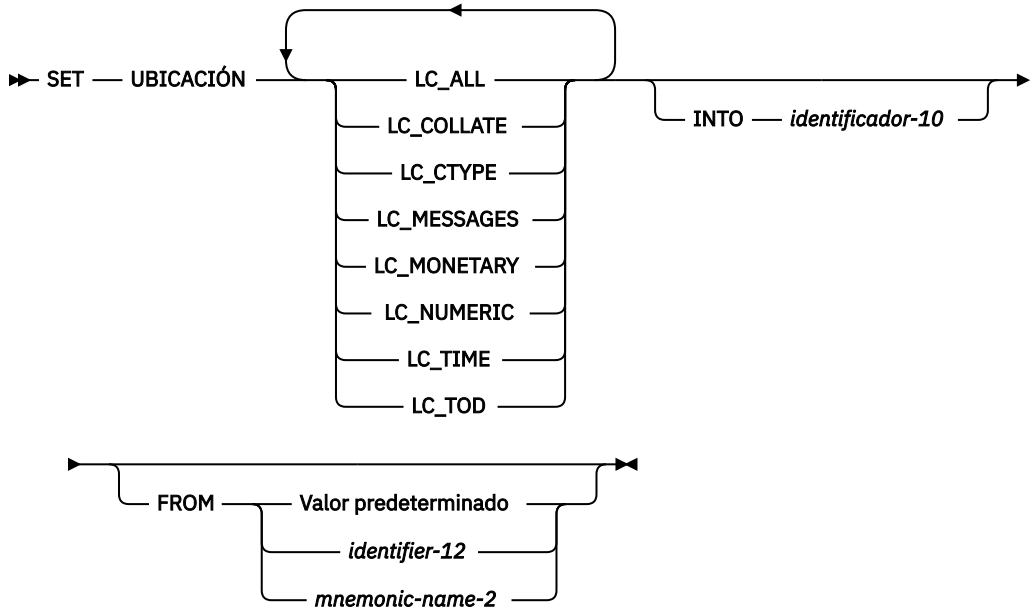
Este formato de la sentencia SET le permite establecer y consultar las categorías de entorno local del entorno local actual.

Un **entorno local** es un objeto del sistema que contiene información de idioma y cultural. Por ejemplo, un entorno local contiene el formato adecuado para una fecha u hora en una región determinada del mundo. La información de un entorno local se divide en **categorías de entorno local**. Por ejemplo, la categoría de entorno local LC_TIME contiene información sobre los formatos de fecha y hora. Para cada unidad de ejecución hay un entorno local DEFAULT, un entorno local actual y de cero a muchos entornos locales específicos. El entorno local actual se altera estableciendo algunas o todas sus categorías de entorno local en el valor DEFAULT o un entorno local específico. El nombre del entorno local específico en el que se ha establecido una categoría de entorno local (del entorno local actual) se puede colocar en un identificador. El contenido de una categoría de entorno local se puede cambiar estableciendo la categoría de entorno local desde:

- El valor predeterminado del sistema
- Un entorno local definido en un elemento de datos elemental alfanumérico
- El nombre mnemotécnico especificado en el párrafo SPECIAL-NAMES.

Cada categoría de entorno local especificada permanece en vigor mientras dure la unidad de ejecución o hasta que se procese otra sentencia SET que especifique la categoría.

Formato 10: sentencia SET para categorías de entorno local



LC_ALL
Categorías de entorno local LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, LC_TIME y LC_TOD, así como cualquier otra categoría incluida en el entorno local.

LC_COLLATE
La categoría de entorno local que define la secuencia de ordenación.

LC_CTYPE
La categoría de entorno local que define la clasificación de caracteres y el tipo de carácter.

MENSAJS_LC
La categoría de entorno local que define el formato de los mensajes informativos y de diagnóstico, y las respuestas interactivas.

MONETARIO_LC
La categoría de entorno local que define el formato monetario.

NÚMERO_LC
La categoría de entorno local que define el formato numérico.

HORA_LC
Elcategoría de entorno local que define el formato de fecha y hora.

LC_TOD
La categoría de entorno local que define las definiciones de diferencias de huso horario, los nombres de huso horario y los puntos de inicio y finalización del horario de verano.

identifier-10
El valor de *identifier-10* hace referencia a una categoría de entorno local. *identifier-10* debe ser un elemento de datos alfanumérico elemental. Si se especifica la frase INTO, la identificación del entorno local actual para la categoría especificada se almacena en el elemento de datos al que hace referencia el *identifier-10*. La frase INTO se procesa antes de la frase FROM, utilizando las reglas de la sentencia MOVE para un movimiento alfanumérico a alfanumérico.

Valor predeterminado
Establece la categoría de entorno local en el valor predeterminado actual. El entorno local predeterminado existe en el momento en que se activa una unidad de ejecución y sigue siendo el valor predeterminado para la duración de la unidad de ejecución. El entorno local predeterminado también se convierte en el entorno local actual en el momento en que se activa una unidad de ejecución y sigue siendo el entorno local actual hasta que se conmuta utilizando el formato 10 de la sentencia SET.

identifier-12
El valor de *identifier-10* hace referencia a una categoría de entorno local. *identifier-12* debe hacer referencia a un elemento de datos alfanuméricos elemental. Si el entorno local especificado en *identifier-12* no está disponible, se emite un mensaje de escape del sistema operativo. Si se especifica la frase FROM, el entorno local actual para la categoría especificada se establece en el contenido del elemento de datos al que hace referencia el *identifier-12*. La identificación del entorno local actual se almacena utilizando las reglas de la sentencia MOVE para un movimiento alfanumérico a alfanumérico.

mnemonic-name-2
Si el entorno local especificado en *mnemonic-name-2* no está disponible, se emite un mensaje de escape del sistema operativo. Si se especifica la frase FROM, el entorno local actual para la categoría especificada se establece en la categoría de entorno local identificada por *mnemonic-name-2*.

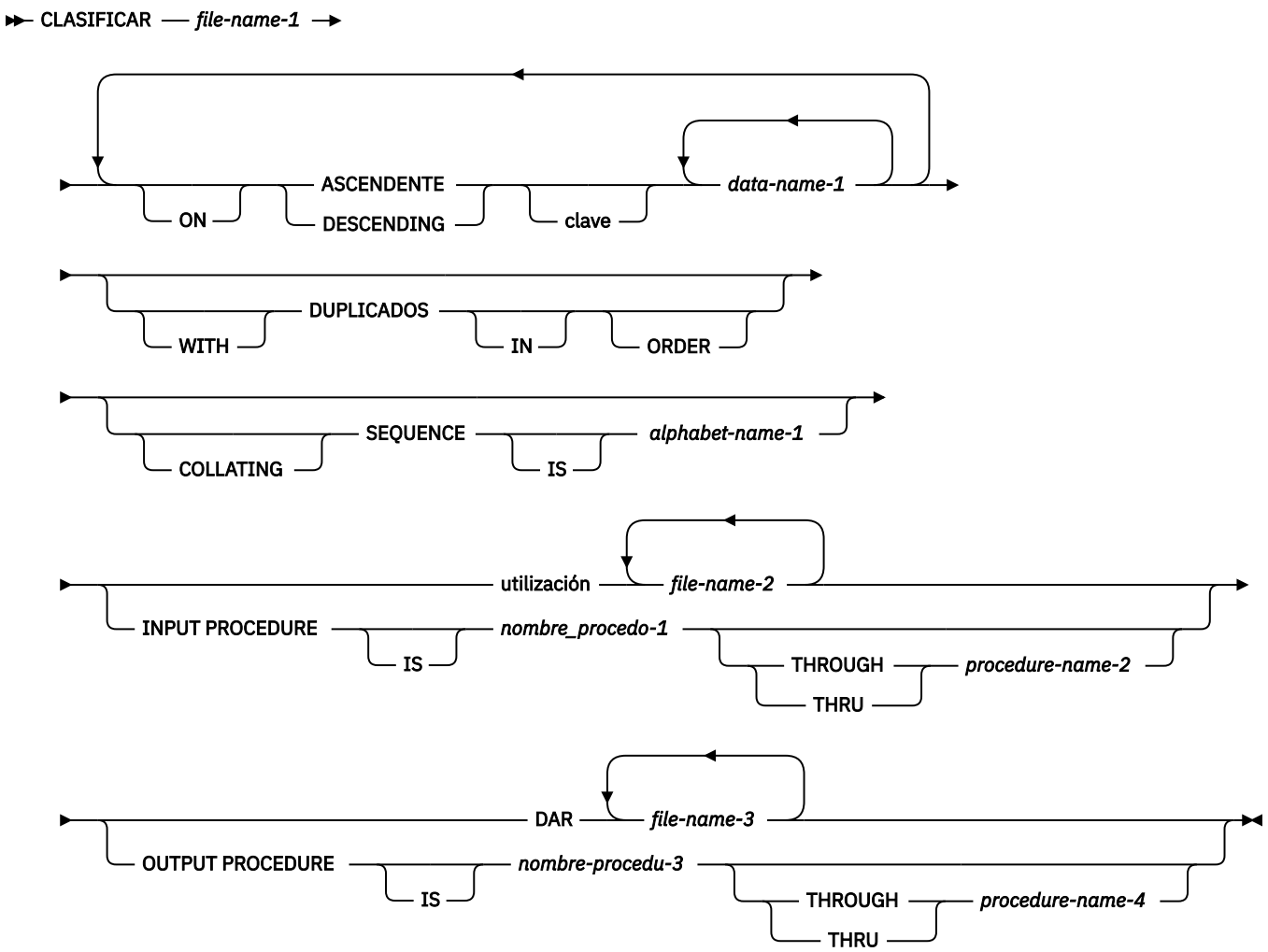
sentencia SORT

La sentencia SORT hace que un conjunto de registros o elementos de tabla se ordene en una secuencia especificada por el usuario.

Para ordenar archivos, la sentencia SORT acepta registros de uno o más archivos, los ordena según las claves especificadas y hace que los registros ordenados estén disponibles a través de un procedimiento de salida o en un archivo de salida.

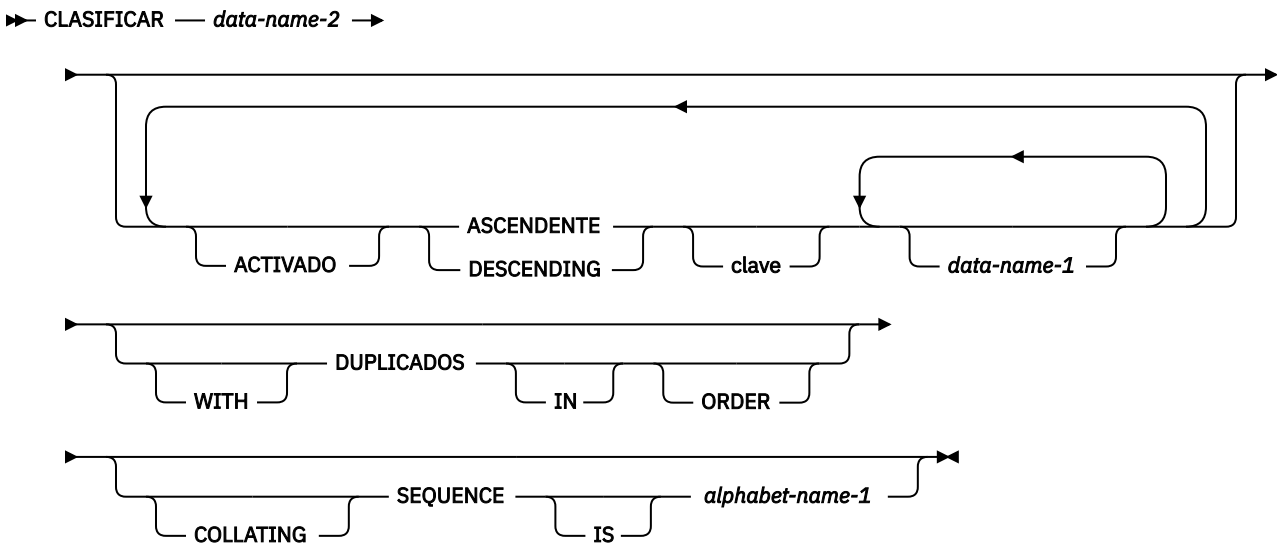
Para ordenar tablas, la sentencia SORT ordena los elementos de tabla según las claves de tabla especificadas.

Formato



Las sentencias SORT de formato 1 pueden aparecer en cualquier lugar de PROCEDURE DIVISION excepto en la parte declarativa. Consulte también “sentencia MERGE” en la página 342.

Formato 2: sentencia SORT de tabla



Las sentencias SORT de formato 2 pueden aparecer en cualquier lugar de PROCEDURE DIVISION.

file-name-1
El nombre proporcionado en la entrada SD que describe los registros que se van a ordenar.

No se puede especificar ningún par de nombres de archivo en una sentencia SORT en la misma cláusula SAME SORT AREA o en la cláusula SAME SORT-MERGE AREA. Nombres de archivo asociados con la cláusula CEDER (*file-name-3*, ...) no se puede especificar en la cláusula SAME AREA; sin embargo, se pueden asociar con la cláusula SAME RECORD AREA.

data-name-2
Especifica un nombre de datos de tabla que está sujeto a las reglas siguientes:

- *data-name-2* debe tener una cláusula OCCURS en la entrada de descripción de datos.
- *data-name-2* puede calificarse.
- *data-name-2* puede tener un subíndice. El subíndice más a la derecha o sólo el subíndice de la tabla debe omitirse o sustituirse por la palabra ALL.

El número de apariciones de elementos de tabla a los que hace referencia *data-name-2* lo determinan las reglas de la cláusula OCCURS. Los elementos de tabla ordenados se colocan en la misma tabla a la que hace referencia *data-name-2*.

Frases ASCENDING KEY y DESCENDING KEY (formato 1)

Esta frase especifica que los registros deben procesarse en secuencia ascendente o descendente (en función de la frase especificada), basándose en las teclas de ordenación especificadas.

data-name-1
Especifica un elemento de datos KEY en el que se basará la sentencia SORT. Cada nombre de datos de este tipo debe identificar un elemento de datos en un registro asociado con *file-name-1*. Los nombres de datos que siguen a la palabra KEY se listan de izquierda a derecha en la sentencia SORT en orden de significación decreciente sin tener en cuenta cómo se dividen en frases KEY. El nombre de datos más a la izquierda es la clave principal, el siguiente nombre de datos es la siguiente clave más significativa, y así sucesivamente. Se aplican las reglas siguientes:

- Un elemento de datos KEY específico debe estar ubicado físicamente en la misma posición y tener el mismo formato de datos en cada archivo de entrada. Sin embargo, no es necesario que tenga el mismo nombre de datos.

- Si *file-name-1* tiene más de una descripción de registro, los elementos de datos KEY sólo se deben describir en una de las descripciones de registro.
- Si *file-name-1* contiene registros de longitud variable, todos los elementos de datos KEY deben estar contenidos en las primeras *n* posiciones de caracteres del registro, donde *n* es igual al tamaño mínimo de registros especificado para *file-name-1*.
- Los elementos de datos KEY no deben contener una cláusula OCCURS o estar subordinados a un elemento que contenga una cláusula OCCURS.
- Los elementos de datos KEY no pueden ser:
  - Ubicado de forma variable
  - Elementos de grupo que contienen elementos de datos de aparición variable
  - Campos de fecha con ventana
  - Numérico de categoría descrito con uso NACIONAL (elemento decimal nacional)
  - Categoría de coma flotante externa descrita con uso NATIONAL (elemento de coma flotante nacional)
  - Categoría DBCS
- Los elementos de datos KEY pueden calificarse.
- Los elementos de datos KEY pueden pertenecer a cualquiera de las siguientes categorías de datos:
  - Alfabético, alfanumérico, alfanumérico-editado
  - Numérico (excepto numérico con uso NACIONAL)
  - Numérico editado (con uso DISPLAY o NATIONAL)
  - Coma flotante interna o visualizar coma flotante
  - Nacional o editado a nivel nacional cuando la opción de compilador NCOLLSEQ (BINARY) está en vigor. La secuencia de clasificación binaria se aplica a las claves nacionales.

Si *file-name-3* hace referencia a un archivo indexado cuya clave de registro principal no es un *nombre-clave-registro*, la primera especificación de *data-name-1* debe estar asociada a una frase ASCENDING y el elemento de datos referenciado por ese *data-name-1* debe ocupar las mismas posiciones de caracteres en este registro que el elemento de datos asociado a la clave de registro principal para ese archivo.

Si *file-name-3* hace referencia a un archivo indexado cuya clave de registro principal es un *nombre-clave-registro*, debe especificar un *data-name-1* correspondiente para cada nombre de datos en la frase SOURCE del *nombre-clave-registro*. Cada *data-name-1* debe estar asociado con una frase ASCENDING, y cada *data-name-1* debe ocupar las mismas posiciones de caracteres que el nombre de datos correspondiente en la frase SOURCE.

Para obtener más información sobre *nombre-clave-registro*, consulte "Párrafo FILE-CONTROL" en la página 109.

La dirección de la operación de clasificación depende de la especificación de las palabras clave ASCENDING o DESCENDING como se indica a continuación:

- Cuando se especifica ASCENDING, la secuencia va del valor de clave más bajo al valor de clave más alto.
- Cuando se especifica DESCENDING, la secuencia va del valor de clave más alto al más bajo.
- Si el elemento de datos KEY es alfabético, alfanumérico, alfanumérico editado o numérico editado, la secuencia de valores de clave depende de la secuencia de clasificación utilizada (consulte "Frase COLLATING SEQUENCE (ambos formatos)" en la página 403).
- Si el elemento de datos KEY se describe con el uso NATIONAL, la secuencia de los valores KEY se basa en los valores binarios de los caracteres nacionales.
- Si la KEY es un elemento de coma flotante de visualización, el compilador trata el elemento de datos como datos de tipo carácter del mismo tamaño que la clave, en lugar de como datos numéricos. La secuencia en la que se ordenan los registros depende del orden de clasificación utilizado.

- Si el elemento de datos KEY es de coma flotante interna, la secuencia de valores de clave estará en orden numérico.
- Cuando no se especifica la frase COLLATING SEQUENCE, las comparaciones de claves se realizan de acuerdo con las reglas para la comparación de operandos en una condición de relación. Consulte "Condiciones generales de relación" en la página 255.
- Cuando se especifica la frase COLLATING SEQUENCE, la secuencia de clasificación indicada se utiliza para elementos de datos clave de categorías alfabéticas, alfanuméricas, editadas alfanuméricas, de coma flotante externa y editadas numéricas. Para todos los demás elementos de datos clave, las comparaciones se realizan de acuerdo con las reglas para la comparación de operandos en una condición de relación.

Frases ASCENDING KEY y DESCENDING KEY (formato 2)

Esta frase especifica que los elementos de tabla deben procesarse en secuencia ascendente o descendente, basándose en la frase y las teclas de ordenación especificadas.

data-name-1
Especifica un nombre de datos KEY que está sujeto a las reglas siguientes:

- El elemento de datos identificado por un nombre de datos clave debe ser el mismo que el elemento de datos al que hace referencia *data-name-2*, o estar subordinado al mismo.
- Los elementos de datos KEY pueden calificarse.
- Los elementos de datos KEY pueden pertenecer a cualquiera de las siguientes categorías de datos:
  - Alfabético, alfanumérico, alfanumérico-editado
  - Numérico (excepto numérico con uso NACIONAL)
  - Numérico editado (con uso DISPLAY o NATIONAL)
  - Coma flotante interna o visualizar coma flotante
  - Nacional o nacional-editado
- Los elementos de datos KEY no pueden ser:
  - Ubicado de forma variable
  - Elementos de grupo que contienen elementos de datos de aparición variable
  - Número de categoría que se describe con el uso NATIONAL (elemento decimal nacional)
  - Categoría de coma flotante externa que se describe con el uso NATIONAL (elemento de coma flotante nacional)
  - Categoría DBCS
  - Objeto de clase
  - USAGE POINTER, USAGE PROCEDURE-POINTER o USAGE FUNCTION-POINTER
  - Subíndice
- Si el elemento de datos identificado por un nombre de datos KEY está subordinado a *data-name-2*, se aplican las reglas siguientes:
  - El elemento de datos no se puede describir con una cláusula OCCURS.
  - El elemento de datos no puede estar subordinado a una entrada que también está subordinada a *data-name-2* y que contiene una cláusula OCCURS.

La frase KEY sólo se puede omitir si la descripción de la tabla a la que hace referencia *data-name-2* contiene una frase KEY.

Las palabras ASCENDING y DESCENDING son transitivas entre todas las apariciones de *data-name-1* hasta que se encuentra otra palabra ASCENDING o DESCENDING.

Los elementos de datos a los que hace referencia *data-name-1* son elementos de datos clave, y estos elementos de datos determinan el orden en el que se almacenan los elementos de tabla ordenados. El

orden de significación de las claves es el orden en el que se especifican los elementos de datos en la sentencia SORT, sin tener en cuenta la asociación con frases ASCENDING o DESCENDING.

La sentencia SORT ordena la tabla a la que hace referencia *data-name-2* y presenta la tabla ordenada en *data-name-2*. El orden de clasificación viene determinado por las frases ASCENDING y DESCENDING (si se ha especificado), o por la frase KEY asociada con *data-name-2*.

La dirección de la operación de clasificación depende de la especificación de las palabras clave ASCENDING o DESCENDING:

- Cuando se especifica ASCENDING, la secuencia va del valor de clave más bajo al más alto.
- Cuando se especifica DESCENDING, la secuencia va del valor de clave más alto al más bajo.
- Si el elemento de datos KEY se describe con el uso NATIONAL, la secuencia de los valores KEY se basa en los valores binarios de los caracteres nacionales.
- Si el elemento de datos KEY es de coma flotante interna, la secuencia de valores de clave está en el orden numérico.
- Cuando no se especifica la frase COLLATING SEQUENCE, la secuencia EBCDIC se utiliza para elementos de datos clave de categorías alfabéticas, alfanuméricas, editadas alfanuméricas, de coma flotante externa y editadas numéricas. Para todos los demás elementos de datos clave, las comparaciones se realizan de acuerdo con las reglas para la comparación de operandos en una condición de relación.
- Cuando se especifica la frase COLLATING SEQUENCE, la secuencia de clasificación indicada se utiliza para elementos de datos clave de categorías alfabéticas, alfanuméricas, editadas alfanuméricas, de coma flotante externa y editadas numéricas. Para todos los demás elementos de datos clave, las comparaciones se realizan de acuerdo con las reglas para la comparación de operandos en una condición de relación.

Para determinar el orden relativo en el que se almacenan los elementos de tabla, el contenido de los elementos de datos clave correspondientes se comparan de acuerdo con las reglas para la comparacion de operandos en una condicion de relacion. La ordenación empieza con el elemento de datos clave más significativo con las reglas siguientes:

- Si el contenido de los elementos de datos clave correspondientes no es igual y la clave está asociada con la frase ASCENDING, el elemento de tabla que contiene el elemento de datos clave con el valor inferior tiene el número de aparición inferior.
- Si el contenido de los elementos de datos clave correspondientes no es igual y la clave está asociada con la frase DESCENDING, el elemento de tabla que contiene el elemento de datos clave con el valor más alto tiene el número de aparición más bajo.
- Si el contenido de los elementos de datos clave correspondientes es igual, la determinación se basa en el contenido del siguiente elemento de datos clave más significativo.

Si no se especifica la frase KEY, la secuencia viene determinada por la frase KEY de la entrada de descripción de datos de la tabla a la que hace referencia *data-name-2*.

Si se especifica la frase KEY, altera temporalmente cualquier frase KEY especificada en la entrada de descripción de datos de la tabla a la que hace referencia *data-name-2*.

Si se omite *data-name-1* , el elemento de datos al que hace referencia *data-name-2* es el elemento de datos clave.

Frase DUPLICATES (formato 1)

Si se especifica la frase DUPLICATES y el contenido de todos los elementos clave asociados a un registro es igual a los elementos clave correspondientes en uno o más registros, el orden de devolución de estos registros es el siguiente:

- El orden de los archivos de entrada asociados tal como se especifica en la sentencia SORT. Dentro de un archivo determinado, el orden es aquel en el que se accede a los registros desde ese archivo.
- El orden en el que un procedimiento de entrada libera estos registros, cuando se especifica un procedimiento de entrada.

Si no se especifica la frase DUPLICATES, el orden de estos registros no está definido.

Frase DUPLICATES (formato 2)

Cuando se cumplen las dos condiciones siguientes, el contenido de los elementos de tabla está en el orden relativo que es el mismo que el orden antes de la operación de ordenación:

- Se ha especificado la frase DUPLICATES.
- El contenido de todos los elementos de datos clave que están asociados con un elemento de tabla es igual al contenido de los elementos de datos clave correspondientes que están asociados con uno o más elementos de tabla.

Si no se especifica la frase DUPLICATES y existe la segunda condición, el orden relativo del contenido de estos elementos de tabla no está definido.

Frase COLLATING SEQUENCE (ambos formatos)

Esta frase especifica el orden de clasificación que se utilizará en las comparaciones alfanuméricas para los elementos de datos KEY en esta operación de clasificación.

La frase COLLATING SEQUENCE no tiene ningún efecto para las claves que no son alfabéticas o alfanuméricas.

La frase COLLATING SEQUENCE sólo es válida cuando está en vigor una página de códigos ASCII de un solo byte.

alphabet-name-1
Debe especificarse en la cláusula ALPHABET del párrafo SPECIAL-NAMES. *alphabet-name-1* se puede asociar con cualquiera de las frases de la cláusula ALPHABET, con los resultados siguientes:

STANDARD-1
La secuencia de clasificación se basa en el orden de los valores hexadecimales del carácter.

STANDARD-2
La secuencia de clasificación se basa en el orden de los valores hexadecimales del carácter.

NATIVA
Se selecciona el orden de clasificación indicado por el entorno local.

EBCDIC
La secuencia de clasificación EBCDIC se utiliza para todas las comparaciones alfanuméricas. (El orden de clasificación EBCDIC se muestra en la Apéndice B, “Secuencias de clasificación EBCDIC y ASCII”, en la página 555.)

literal
La secuencia de clasificación establecida por la especificación de literales en la cláusula de nombre de alfabeto se utiliza para todas las comparaciones alfanuméricas.

Cuando se omite la frase COLLATING SEQUENCE, la cláusula PROGRAM COLLATING SEQUENCE (si se especifica) en el párrafo OBJECT-COMPUTER especifica el orden de clasificación que debe utilizarse. Cuando se omiten la frase COLLATING SEQUENCE y la cláusula PROGRAM COLLATING SEQUENCE, se utiliza la secuencia de clasificación EBCDIC se utiliza la secuencia de clasificación indicada por la opción de compilador COLLSEQ.

frase USING

file-name-2 , ...
Los archivos de entrada.

Cuando se especifica la frase USING, todos los registros de *file-name-2*, ..., (es decir, los archivos de entrada) se transfieren automáticamente a *file-name-1*. En el momento en que se ejecuta la sentencia SORT, estos archivos no deben estar abiertos. El compilador abre, lee, pone los registros a disposición y cierra estos archivos automáticamente. Si se especifican procedimientos EXCEPTION/ERROR para estos archivos, el compilador establece el enlace necesario con estos procedimientos.

Todos los archivos de entrada deben describirse en entradas FD en DATA DIVISION.

Si se especifica la frase USING y si *file-name-1* contiene registros de longitud variable, el tamaño de los registros contenidos en los archivos de entrada (*file-name-2*, ...) no debe ser menor que el registro más pequeño ni mayor que el registro más grande descrito para *file-name-1*. Si *file-name-1* contiene registros de longitud fija, el tamaño de los registros contenidos en los archivos de entrada no debe ser mayor que el registro más grande descrito para *file-name-1*. Para obtener más información, consulte *Descripción de la entrada para ordenar o fusionar* en la publicación *COBOL for Linux en x86 Guía de programación*.

Frase INPUT PROCEDURE

Esta frase especifica el nombre de un procedimiento que debe seleccionar o modificar registros de entrada antes de que comience la operación de clasificación.

procedure-name-1
Especifica la primera (o única) sección o párrafo en el procedimiento de entrada.

procedure-name-2
Identifica la última sección o párrafo del procedimiento de entrada.

El procedimiento de entrada puede consistir en cualquier procedimiento necesario para seleccionar, modificar o copiar los registros que la sentencia RELEASE pone a disposición de uno en uno en el archivo al que hace referencia *file-name-1*. El rango incluye todas las sentencias que se ejecutan como resultado de una transferencia de control mediante sentencias CALL, EXIT, GO TO, PERFORM y XML PARSE en el rango del procedimiento de entrada, así como todas las sentencias en procedimientos declarativos que se ejecutan como resultado de la ejecución de sentencias en el rango del procedimiento de entrada. El rango del procedimiento de entrada no debe provocar la ejecución de ninguna sentencia MERGE, RETURN o format 1 SORT.

Si se especifica un procedimiento de entrada, el control se pasa al procedimiento de entrada antes de que la sentencia SORT secuencie el archivo al que hace referencia *file-name-1* . El compilador inserta un mecanismo de retorno al final de la última sentencia en el procedimiento de entrada. Cuando el control pasa la última sentencia en el procedimiento de entrada, los registros que se han liberado en el archivo al que hace referencia *file-name-1* se ordenan.

Frase CEDER

file-name-3 , ...
Los archivos de salida.

Cuando se especifica la frase CEDER, todos los registros ordenados en *file-name-1* se transfieren automáticamente a los archivos de salida (*file-name-3*, ...).

Todos los archivos de salida deben describirse en entradas FD en DATA DIVISION.

Si los archivos de salida (*file-name-3*, ...) contienen registros de longitud variable, el tamaño de los registros contenidos en *file-name-1* no debe ser menor que el registro más pequeño ni mayor que el registro más grande descrito para los archivos de salida. Si los archivos de salida contienen registros de longitud fija, el tamaño de los registros contenidos en *file-name-1* no debe ser mayor que el registro más grande descrito para los archivos de salida. Para obtener más información, consulte *Descripción de la salida de la ordenación o fusión* en la publicación *COBOL for Linux en x86 Guía de programación*.

En el momento en que se ejecuta la sentencia SORT, los archivos de salida (*file-name-3*, ...) no debe estar abierto. Para cada uno de los archivos de salida, la ejecución de la sentencia SORT hace que se realicen las acciones siguientes:

- Se inicia el proceso del archivo. La iniciación se realiza como si se hubiera ejecutado una sentencia OPEN con la frase OUTPUT.
- Los registros lógicos ordenados se devuelven y se graban en el archivo. Cada registro se escribe como si se hubiera ejecutado una sentencia WRITE sin ninguna frase opcional.

  Para un archivo relativo, el elemento de datos de clave relativa para el primer registro devuelto contiene el valor '1'; para el segundo registro devuelto, el valor '2'. Después de la ejecución de la sentencia SORT, el contenido del elemento de datos de clave relativa indica el último registro devuelto al archivo.
- El proceso del archivo ha terminado. La terminación se realiza como si se hubiera ejecutado una sentencia CLOSE sin frases opcionales.

Estas funciones implícitas se realizan de tal forma que se ejecuten los procedimientos USE AFTER EXCEPTION/ERROR asociados; sin embargo, la ejecución de dicho procedimiento USE no debe provocar la ejecución de ninguna sentencia que manipule el archivo referenciado por, o que acceda al área de registro asociada con, *file-name-3*. En el primer intento de grabar más allá de los límites definidos externamente del archivo, se ejecuta cualquier procedimiento USE AFTER STANDARD EXCEPTION/ERROR especificado para el archivo. Si se devuelve el control desde ese procedimiento USE o si no se especifica dicho procedimiento USE, se termina el proceso del archivo.

Frase OUTPUT PROCEDURE

Esta frase especifica el nombre de un procedimiento que debe seleccionar o modificar registros de salida de la operación de ordenación.

procedure-name-3
Especifica la primera (o única) sección o párrafo en el procedimiento de salida.

procedure-name-4
Identifica la última sección o párrafo del procedimiento de salida.

El procedimiento de salida puede consistir en cualquier procedimiento necesario para seleccionar, modificar o copiar los registros que la sentencia RETURN pone a disposición de uno en uno en orden de clasificación desde el archivo al que hace referencia *file-name-1*. El rango incluye todas las sentencias que se ejecutan como resultado de una transferencia de control por parte de las sentencias CALL, EXIT, GO TO, PERFORM y XML PARSE en el rango del procedimiento de salida. El rango también incluye todas las sentencias en procedimientos declarativos que se ejecutan como resultado de la ejecución de sentencias en el rango del procedimiento de salida. El rango del procedimiento de salida no debe provocar la ejecución de ninguna sentencia SORT MERGE, RELEASE o format 1 .

Si se especifica un procedimiento de salida, el control lo pasa después de que la sentencia SORT haya secuenciado el archivo al que hace referencia *file-name-1* . El compilador inserta un mecanismo de retorno al final de la última sentencia en el procedimiento de salida y cuando el control pasa la última sentencia en el procedimiento de salida, el mecanismo de retorno proporciona la terminación de la clasificación y luego pasa el control a la siguiente sentencia ejecutable después de la sentencia SORT. Antes de entrar en el procedimiento de salida, el procedimiento de clasificación alcanza un punto en el que puede seleccionar el siguiente registro en orden ordenado cuando se le solicite. Las sentencias RETURN en el procedimiento de salida son las peticiones para el siguiente registro.

Las frases INPUT PROCEDURE y OUTPUT PROCEDURE son similares a las de una sentencia PERFORM básica. Por ejemplo, si nombra un procedimiento en un procedimiento de salida, dicho procedimiento se ejecuta durante la operación de clasificación como si se nombrara en una sentencia PERFORM. Al igual que con la sentencia PERFORM, la ejecución del procedimiento se termina después de que la última sentencia complete la ejecución. La última sentencia de un procedimiento de entrada o salida puede ser la sentencia EXIT (consulte “sentencia EXIT” en la página 323).

Registros especiales SORT

Los registros especiales, SORT-CORE-SIZE, SORT-MESSAGE y SORT-MODE-SIZE, son equivalentes a las palabras clave de la sentencia de control de opción en el archivo de control de clasificación. Puede definir el archivo de control de ordenación con el registro especial SORT-CONTROL.

Nota de uso: Si utiliza un archivo de control de ordenación para especificar sentencias de control, los valores especificados en el archivo de control de ordenación tienen prioridad sobre los del registro especial.

Registro especial SORT-MESSAGE
Consulte “ORDENAR-MENSAJE” en la página 25.

Registro especial SORT-CORE-SIZE
Consulte “TAMAÑO-NÚCLEO-CLASIFICACIÓN” en la página 24.

Registro especial SORT-FILE-SIZE
Consulte “SORT-FILE-SIZE” en la página 24.

Registro especial SORT-MODE-SIZE
Consulte “TAMAÑO-MODALIDAD-CLASIFICACIÓN” en la página 25.

Registro especial SORT-CONTROL
Consulte “CONTROL DE SORT” en la página 24.

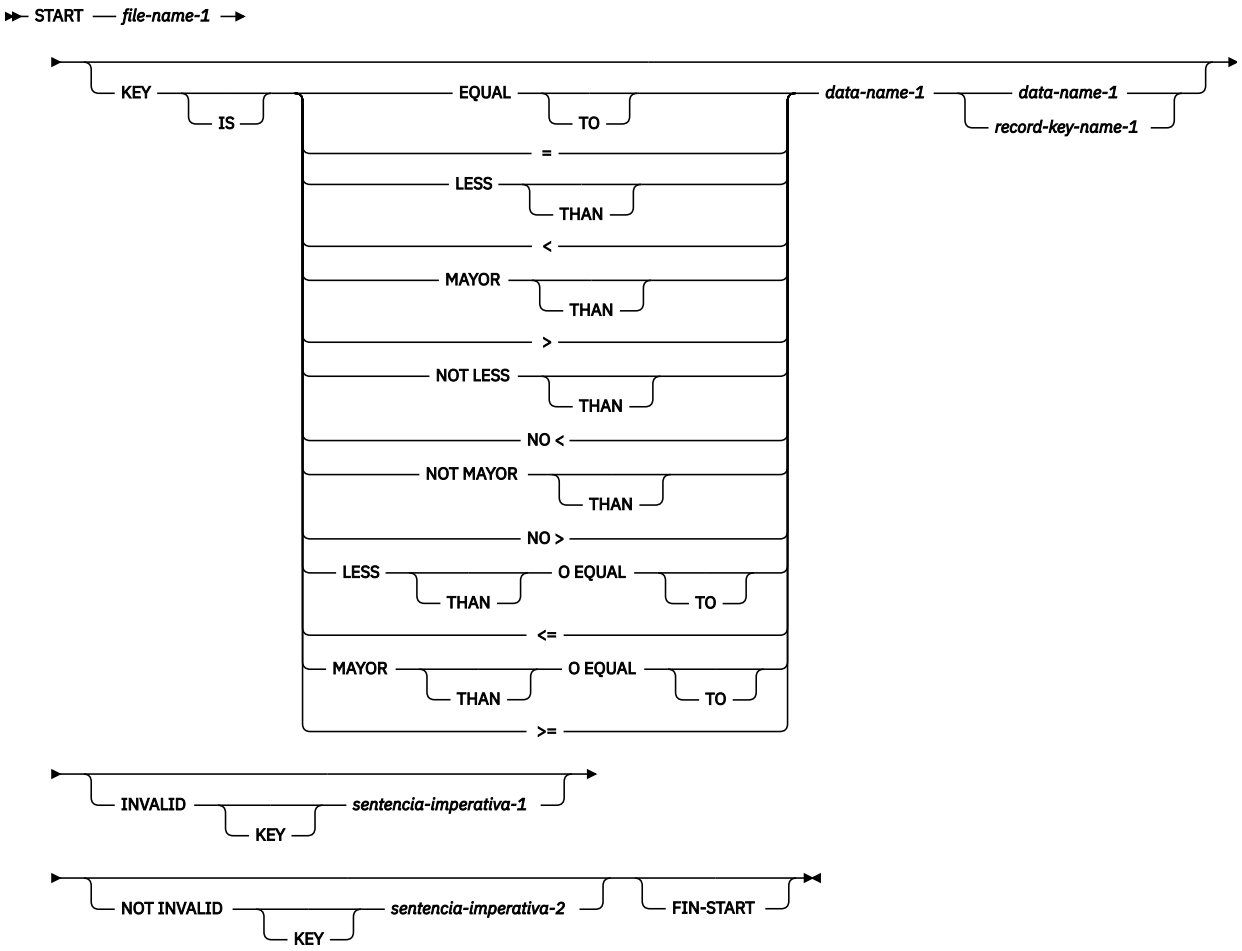
Registro especial SORT-RETURN
Consulte “ORDENAR-RETORNO” en la página 25.

sentencia START

La sentencia START proporciona un medio de posicionamiento dentro de un archivo indexado o relativo para la posterior recuperación secuencial de registros.

Cuando se ejecuta la sentencia START, el archivo indexado o relativo asociado debe estar abierto en modalidad INPUT o I-O.

Formato



Restricción: *nombre-clave-registro* sólo está soportado para el sistema de archivos STL.

file-name-1
Debe nombrar un archivo con acceso secuencial o dinámico. *file-name-1* debe estar definido en una entrada FD en DATA DIVISION y no debe nombrar un archivo de ordenación.

Frase KEY

record-key-name-1 debe especificarse con la frase SOURCE en la cláusula RECORD KEY o en la cláusula ALTERNATE RECORD KEY en la entrada de control de archivo para *file-name-1*. *data-name-1* o *record-key-name-1* se puede calificar. *data-name-1* no se puede suscribir.

Cuando se especifica la frase KEY, el indicador de posición de archivo se coloca en el registro lógico del archivo cuyo campo de clave satisface la comparación.

Cuando no se especifica la frase KEY, KEY IS EQUAL (a la clave de registro principal) está implícito.

Si especifica que la KEY sea 'menor que' o 'menor o igual que' el elemento de datos, el indicador de posición de archivo se sitúa en el último registro lógico que existe actualmente en el archivo que satisface la comparación.

Para un archivo indexado, si la clave que satisface la comparación tiene entradas duplicadas, el indicador de posición de archivo se coloca en la última de estas entradas.

data-name-1
Se puede calificar; no se puede suscribir.

Cuando se ejecuta la sentencia START, se realiza una comparación entre el valor actual en el nombre de datos de clave y el campo de clave correspondiente en el índice del archivo.

Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, la clave de estado de archivo asociada se actualiza cuando se ejecuta la sentencia START (consulte “Clave de estado de archivo” en la página 285).

Frases INVALID KEY

Si ningún registro del archivo satisface la comparación, existe una condición de clave no válida; la posición del indicador de posición de archivo no está definida y (si se especifica) se ejecuta la sentencia imperativa INVALID KEY. (Consulte “Frases INTO y FROM” en la página 290 en "Instalaciones de proceso comunes".)

Se puede omitir la frase INVALID KEY y un procedimiento EXCEPTION/ERROR aplicable.

Frase END-START

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia START. END-START permite que una sentencia START condicional se anide en otra sentencia condicional. END-START también se puede utilizar con una sentencia START imperativa.

Para obtener más información, consulte “Sentencias de ámbito delimitado” en la página 278.

Archivos indexados

Cuando se especifica la frase KEY, el elemento de datos clave utilizado para la comparación es *data-name-1* o *record-key-name-1*.

Cuando no se especifica la frase KEY, el elemento de datos clave utilizado para la comparación EQUAL TO es la clave de registro principal.

Cuando la ejecución de la sentencia START es satisfactoria, la CLAVE DE REGISTRO o la CLAVE DE REGISTRO ALTERNATIVA con la que está asociada *data-name-1* o *record-key-name-1* se convierte en la clave de referencia para las sentencias READ posteriores.

data-name-1* o *record-key-name-1
Puede ser cualquiera de los elementos siguientes:

- CLAVE DE REGISTRO PRINCIPAL.
- CUALQUIER CLAVE DE REGISTRO ALTERNATIVA.
- Elemento de datos dentro de una descripción de registro para un archivo cuya posición de carácter más a la izquierda corresponde a la posición de carácter más a la izquierda de dicha clave de registro; puede calificarse. El tamaño del elemento de datos debe ser menor o igual que la longitud de la clave de registro para el archivo.

Independientemente de su categoría, *data-name-1* o *record-key-name-1* se trata como un elemento alfanumérico a efectos de la operación de comparación.

El indicador de posición de archivo apunta al primer registro del archivo cuyo campo de clave satisface la comparación. Si los operandos de la comparación son de longitudes desiguales, la comparación continúa como si el campo más largo se truncara a la derecha a la longitud del campo más corto. Se aplican todas las demás reglas de comparación numéricas y alfanuméricas, excepto que la cláusula PROGRAM COLLATING SEQUENCE, si se especifica, no tiene ningún efecto.

Cuando la ejecución de la sentencia START es satisfactoria, RECORD KEY con el que se asocia *data-name-1* o *record-key-name-1* se convierte en la clave de referencia para las sentencias READ posteriores.

Cuando la ejecución de la sentencia START no es satisfactoria, la clave de referencia no está definida.

Archivos relativos

Cuando se especifica la frase KEY, *data-name-1* debe especificar RELATIVE KEY.

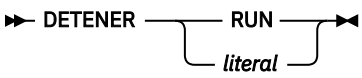
Tanto si se especifica la frase KEY como si no, el elemento de datos clave utilizado en la comparación es el elemento de datos RELATIVE KEY. Se aplican las reglas de comparación numérica.

El indicador de posición de archivo apunta al registro lógico del archivo cuya clave satisface la comparación especificada.

sentencia STOP

La sentencia STOP detiene la ejecución del programa objeto de forma permanente o temporal.

Formato



literal
Puede ser un literal numérico de punto fijo (con signo o sin signo), un literal alfanumérico o un literal booleano. Puede ser cualquier constante figurativa excepto ALL *literal*.

Cuando se especifica STOP *literal* , el literal se comunica al operador y se suspende la ejecución del programa objeto. La ejecución del programa sólo se reanuda después de la intervención del operador y continúa en la siguiente sentencia ejecutable en secuencia.

La sentencia *literal* STOP es útil para situaciones especiales cuando se necesita la intervención del operador durante la ejecución del programa; por ejemplo, cuando se debe montar una cinta o un disco especial o se debe especificar un código diario específico. Sin embargo, las sentencias ACCEPT y DISPLAY son las preferidas cuando se necesita la intervención del operador.

Cuando se especifica STOP RUN, la ejecución termina y el control se devuelve al sistema. Cuando STOP RUN no es la última o única sentencia de una secuencia de sentencias imperativas dentro de una frase, las sentencias que siguen a STOP RUN no se ejecutan.

La sentencia STOP RUN cierra *todos* los archivos definidos en cualquiera de los programas de la unidad de ejecución.

Para utilizar la sentencia STOP RUN en la llamada y los programas llamados, consulte la tabla siguiente.

Sentencia de terminación	Programa principal	Subprograma
DETENER EJECUCIÓN	Vuelve al programa de llamada. (Puede ser el sistema, lo que hace que la aplicación finalice.)	Vuelve directamente al programa que ha llamado al programa principal. (Puede ser el sistema, lo que hace que la aplicación finalice.)

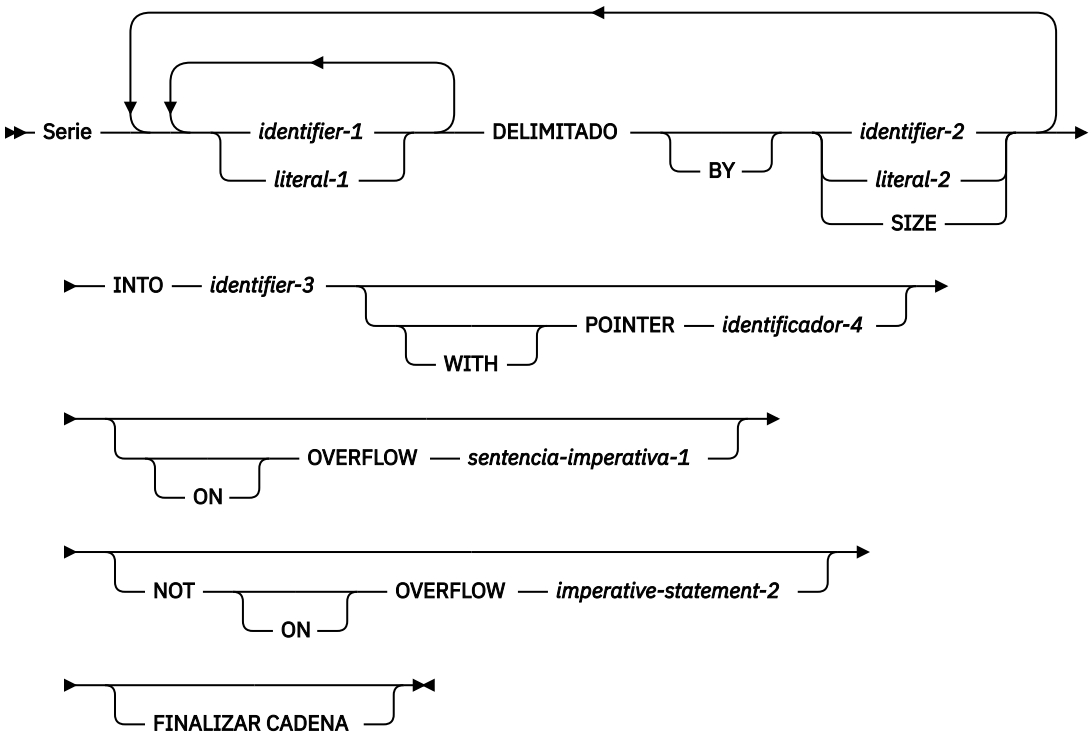
Nota: En CICS, si se invoca STOP RUN desde un programa que se ha iniciado mediante una sentencia EXEC CICS LINK, entonces STOP RUN se comporta como si se hubiera emitido GOBACK o EXEC CICS RETURN: STOP RUN devuelve el control al programa de enlace.

sentencia STRING

La sentencia STRING encadena el contenido parcial o completo de dos o más elementos de datos o literales en un único elemento de datos.

Se puede escribir una sentencia STRING en lugar de una serie de sentencias MOVE.

Formato



identifier-1*, *literal-1
Representa los *campos de envío*.

Frase DELIMITED BY
Establece los límites de la serie.

identifier-2*, *literal-2
Son delimitadores; es decir, caracteres que delimitan los datos que se van a transferir.

TAMAÑO
Transfiere el área de envío completa.

Frase INTO
Identifica el campo de recepción.

identifier-3
Representa el *campo de recepción*.

Frase POINTER
Apunta a una posición de carácter en el campo de recepción. El campo de puntero indica una posición de carácter alfanumérico relativo, una posición de carácter DBCS o una posición de carácter nacional cuando el campo de recepción es de uso DISPLAY, DISPLAY-1o NATIONAL, respectivamente.

identifier-4
Representa el *campo de puntero*. *identifier-4* debe ser lo suficientemente grande para contener un valor igual a la longitud del campo receptor más 1. Debe inicializar *identifier-4* en un valor distinto de cero antes de que comience la ejecución de la sentencia STRING.

Se aplican las reglas siguientes:

- Todos los identificadores excepto *identifier-4* deben hacer referencia a elementos de datos descritos explícita o implícitamente como de uso DISPLAY, DISPLAY-1o NATIONAL.
- *literal-1* o *literal-2* debe ser de categoría alfanumérica, DBCS o nacional y puede ser cualquier constante figurativa que no empiece por la palabra ALL (excepto NULL).

- Si *identifier-1* o *identifer-2* hace referencia a un elemento de datos de categoría numérica, cada elemento numérico debe describirse como un entero sin el símbolo 'P' en su serie de caracteres PICTURE.
- *identifier-3* no debe hacer referencia a un elemento de datos de categoría numérico editado, alfanumérico editado o nacional editado; un elemento de datos de coma flotante externo de uso DISPLAY, o un elemento de datos de coma flotante externo de uso NATIONAL.
- *identifier-3* no debe describirse con la cláusula JUSTIFIC.
- Si *identifier-3* es de uso DISPLAY, *identifier-1* y *identifier-2* deben ser de uso DISPLAY y todos los literales deben ser literales alfanuméricos. Se puede especificar cualquier constante figurativa excepto una que empiece con la palabra ALL. Cada constante figurativa representa un literal alfanumérico de 1 carácter.
- Si *identifier-3* es de uso DISPLAY-1, *identifier-1* y *identifier-2* deben ser de uso DISPLAY-1 y todos los literales deben ser literales DBCS. La única constante figurativa que se puede especificar es SPACE, que representa un literal DBCS de 1 carácter. No debe especificarse ALL *literal-DBCS* .
- Si *identifier-3* es de uso NACIONAL, *identifier-1* y *identifier-2* deben ser de uso NATIONAL y todos los literales deben ser literales nacionales. Se puede especificar cualquier constante figurativa excepto *carácter-simbólico* y una que empiece por la palabra ALL. Cada constante figurativa representa un literal nacional de 1 carácter.
- Si *identifier-1* o *identifier-2* hace referencia a un elemento de datos elemental de uso DISPLAY que se describe como de categoría numérica, numérica editada o alfanumérica editada, el elemento se trata como si se hubiera redefinido como categoría alfanumérica.
- Si *identifier-1* o *identifier-2* hace referencia a un elemento de datos elementales de uso NATIONAL que se describe como elemento de categoría numérico, editado numérica-editado o editado a nivel nacional, la partida se trata como si se redefiniera como categoría nacional.
- *identifier-4* no debe describirse con el símbolo P en su serie de caracteres PICTURE.
- Ninguno de los identificadores de una sentencia STRING puede ser un campo de fecha con ventana.

La evaluación de subíndices, modificación de referencia, longitudes de variable, ubicaciones de variable e identificadores de función se realiza sólo una vez, al principio de la ejecución de la sentencia STRING. Por lo tanto, si se utiliza *identifier-3* o *identifier-4* como subíndice, modificador de referencia o argumento de función en la sentencia STRING, o afecta a la longitud o ubicación de cualquiera de los identificadores de la sentencia STRING, los valores calculados para dichos subíndices, modificadores de referencia, longitudes de variable, ubicaciones de variable y funciones no se ven afectados por ningún resultado de la sentencia STRING.

Si *identifier-3* y *identifier-4* ocupan la misma área de almacenamiento, se producirán resultados no definidos, incluso si los identificadores están definidos por la misma entrada de descripción de datos.

Si *identifier-1* o *identifier-2* ocupa la misma área de almacenamiento que *identifier-3* o *identifier-4*, se producirán resultados no definidos, incluso si los identificadores están definidos por la misma entrada de descripción de datos.

Consulte “Flujo de datos” en la página 412 para obtener detalles sobre el proceso de sentencias STRING.

Frases ON OVERFLOW

imperative-statement-1
Ejecutado cuando el valor del puntero (explícito o implícito):

- Es menor que 1
- Excede un valor igual a la longitud del campo de recepción

Cuando se produce cualquiera de las condiciones anteriores, existe una condición de desbordamiento y no se transfieren más datos. A continuación, se termina la operación STRING, se ignora la frase NOT ON OVERFLOW, si se especifica, y el control se transfiere al final de la sentencia STRING o, si se especifica la frase ON OVERFLOW, a *imperative-statement-1*.

Si el control se transfiere a *imperative-statement-1*, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en *imperative-statement-1*. Si se ejecuta una ramificación de procedimiento o una sentencia condicional que provoca una transferencia explícita de control, el control se transfiere de acuerdo con las reglas de dicha sentencia; de lo contrario, al finalizar la ejecución de *imperative-statement-1*, el control se transfiere al final de la sentencia STRING.

Si en el momento de la ejecución de una sentencia STRING, no se encuentran las condiciones que causarían una condición de desbordamiento, después de la finalización de la transferencia de datos, la frase ON OVERFLOW, si se especifica, se ignora. A continuación, el control se transfiere al final de la sentencia STRING, o si se especifica la frase NOT ON OVERFLOW, a *imperative-statement-2*.

Si el control se transfiere a *imperative-statement-2*, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en *imperative-statement-2*. Si se ejecuta una ramificación de procedimiento o una sentencia condicional que provoca una transferencia explícita de control, el control se transfiere de acuerdo con las reglas de dicha sentencia. De lo contrario, al finalizar la ejecución de *imperative-statement-2*, el control se transfiere al final de la sentencia STRING.

frase END-STRING

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia STRING. END-STRING permite que una sentencia STRING condicional se anide en otra sentencia condicional. END-STRING también puede utilizarse con una sentencia STRING imperativa.

Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

Flujo de datos

Cuando se ejecuta la sentencia STRING, los caracteres se transfieren desde los campos de envío al campo de recepción. El orden en el que se procesan los campos de envío es el orden en el que se especifican.

Se aplican las reglas siguientes:

- Los caracteres de los campos de envío se transfieren a los campos de recepción de la siguiente manera:
  - Para los campos de envío nacionales, los datos se transfieren utilizando las reglas de la declaración MOVE para movimientos nacionales elementales, excepto que no se realiza ningún llenado de espacio.
  - Para campos de envío DBCS, los datos se transfieren utilizando las reglas de la sentencia MOVE para movimientos elementales de DBCS a DBCS, excepto que no se realiza ningún llenado de espacio.
  - De lo contrario, los datos se transfieren a los campos de recepción utilizando las reglas de la sentencia MOVE para movimientos alfanuméricos a alfanuméricos elementales, excepto que no tiene lugar ningún relleno de espacio (consulte "sentencia MOVE" en la página 347).
- Cuando se especifica DELIMITED BY *identifier-2* o *literal-2* , el contenido de cada elemento de envío se transfiere, carácter por carácter, empezando por la posición del carácter más a la izquierda y continuando hasta:
  - Se ha alcanzado un delimitador para este campo de envío (el propio delimitador no se transfiere).
  - Se ha transferido el carácter situado más a la derecha de este campo de envío.
- Cuando se especifica DELIMITED BY SIZE, cada campo de envío completo se transfiere al campo de recepción.
- Cuando se rellena el campo de recepción o cuando se han procesado todos los campos de envío, finaliza la operación.
- Cuando se especifica la frase POINTER, el usuario COBOL dispone de un campo de puntero explícito para controlar la colocación de datos en el campo receptor. El usuario debe establecer el valor inicial del puntero explícito, que no debe ser menor que 1 ni mayor que el recuento de posición de caracteres del campo receptor.

Nota de uso: El campo de puntero debe definirse como un campo lo suficientemente grande como para contener un valor igual a la longitud del campo receptor más 1; esto impide el desbordamiento aritmético cuando el sistema actualiza el puntero al final de la transferencia.

- Cuando no se especifica la frase POINTER, no hay ningún puntero disponible para el usuario. Sin embargo, el sistema utiliza un puntero implícito conceptual con un valor inicial de 1.
- Conceptualmente, cuando se ejecuta la sentencia STRING, el valor de puntero inicial (explícito o implícito) es la primera posición de carácter dentro del campo receptor al que se van a transferir los datos. A partir de esa posición, los datos se posicionan, carácter por carácter, de izquierda a derecha. Después de colocar cada carácter, el puntero explícito o implícito se incrementa en 1. El valor del campo de puntero sólo se cambia de esta manera. Al final del proceso, el valor del puntero siempre indica un valor igual a una posición de carácter más allá del último carácter transferido al campo receptor.

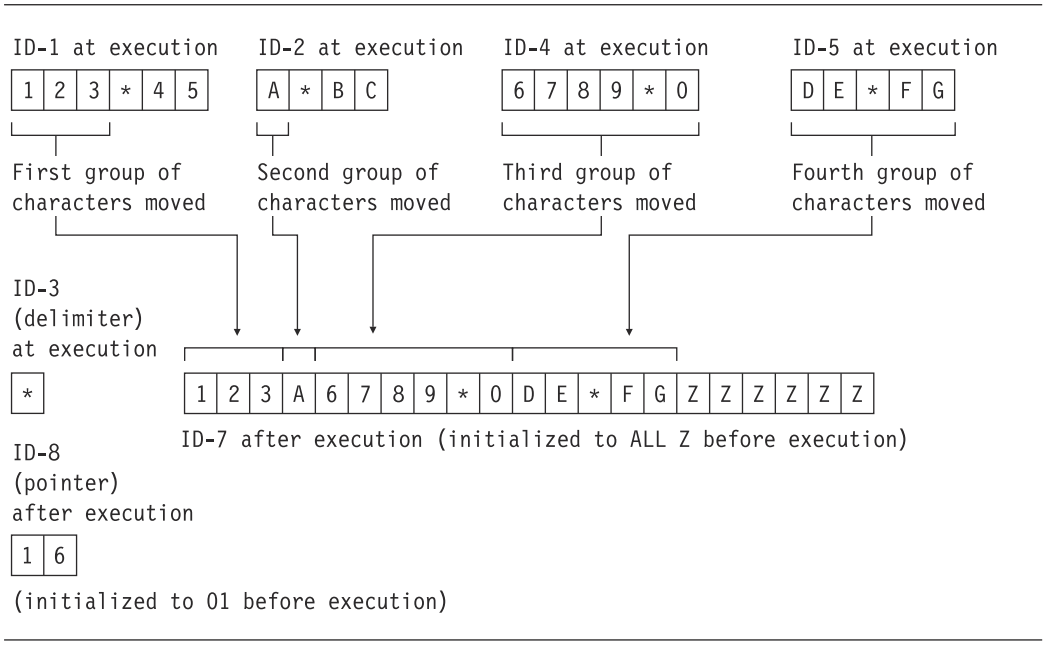
Una vez completada la ejecución de la sentencia STRING, sólo se cambia la parte del campo receptor a la que se transfirieron los datos. El resto del campo de recepción contiene los datos que estaban presentes antes de esta ejecución de la sentencia STRING.

Ejemplo de la sentencia STRING

Este tema lista un ejemplo para la sentencia STRING.

Cuando se ejecuta la siguiente sentencia STRING, los resultados obtenidos serán como los que se ilustran en la figura después de la sentencia.

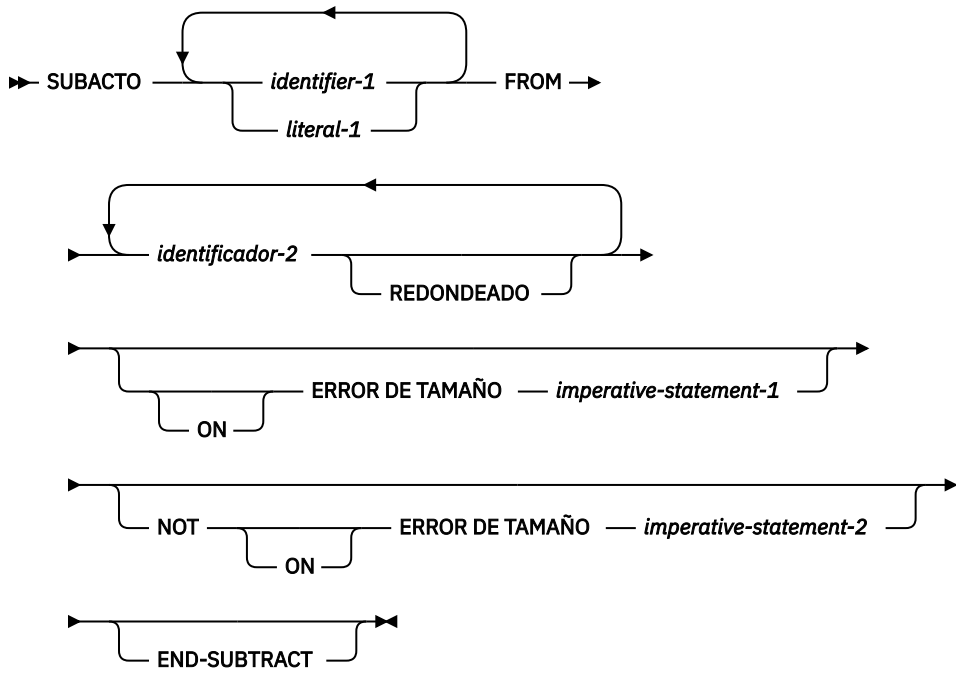
```
STRING ID-1 ID-2 DELIMITED BY ID-3
       ID-4 ID-5 DELIMITED BY SIZE
  INTO ID-7 WITH POINTER ID-8
END-STRING
```



SUBTRACT, sentencia

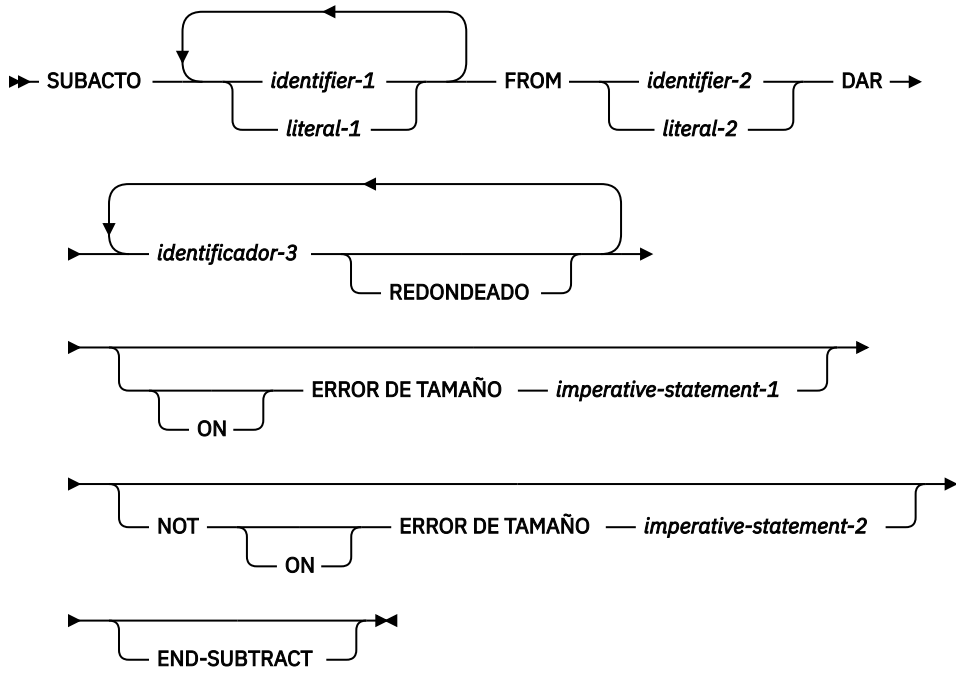
La sentencia SUBTRACT resta un elemento numérico, o la suma de dos o más elementos numéricos, de uno o más elementos numéricos, y almacena el resultado.

Formato 1: sentencia SUBTRACT



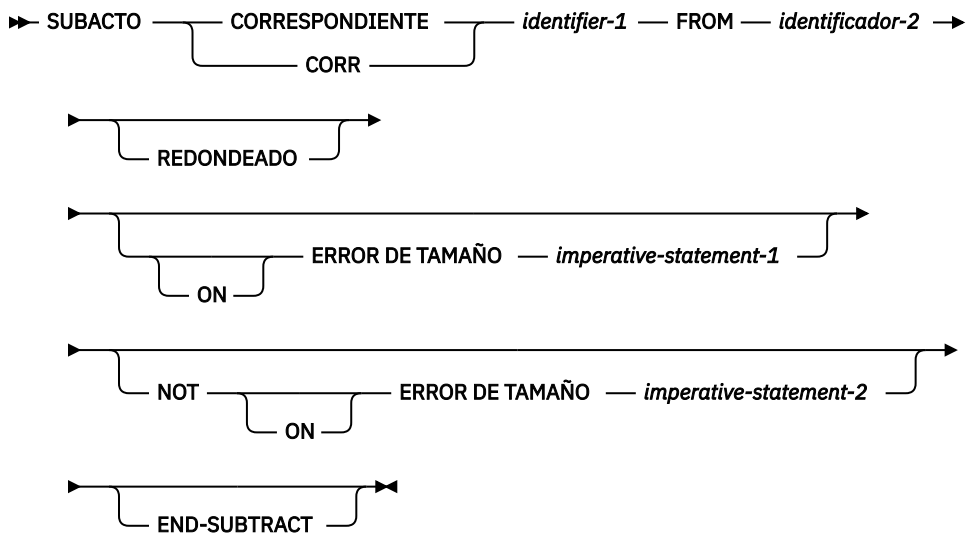
Todos los identificadores o literales que preceden a la palabra clave FROM se suman y su suma se resta y se almacena inmediatamente en *identifier-2*. Este proceso se repite para cada aparición sucesiva de *identifier-2*, en el orden de izquierda a derecha en el que se especifica *identifier-2* .

Formato 2: sentencia SUBTRACT con frase CEDER



Todos los identificadores o literales que preceden a la palabra clave FROM se suman y su suma se resta de *identifier-2* o *literal-2*. El resultado de la resta se almacena como el nuevo valor de cada elemento de datos al que hace referencia *identifier-3*.

Formato 3: sentencia SUBTRACT con frase CORRESPONDIENTE



Los elementos de datos elementales dentro de *identifier-1* se restan de, y los resultados se almacenan en, los elementos de datos elementales correspondientes dentro de *identifier-2*.

Cuando la opción de compilador ARITH (COMPAT) está en vigor, el compuesto de operandos puede contener un máximo de 30 dígitos. Cuando la opción de compilador ARITH (EXTEND) está en vigor, el compuesto de operandos puede contener un máximo de 31 dígitos. Para obtener más información sobre los resultados intermedios aritméticos, consulte *Apéndice A. Resultados intermedios y precisión aritmética* en *COBOL for Linux en x86 Guía de programación*.

Para todos los formatos:

identificador

En el formato 1, debe nombrar un elemento de datos numérico elemental.

En el formato 2, debe nombrar un elemento de datos numérico elemental, a menos que el identificador siga la palabra DANDO. Cada identificador que sigue a la palabra CEDER debe nombrar un elemento de datos elemental numérico o numérico editado.

En el formato 3, debe nombrar un elemento de grupo alfanumérico o un elemento de grupo nacional.

Las restricciones siguientes se aplican a los campos de fecha:

- En el formato 1, *identifier-1* puede especificar como máximo un campo de fecha. Si *identifier-1* especifica un campo de fecha, entonces cada instancia de *identifier-2* debe especificar un campo de fecha que sea compatible con el campo de fecha especificado por *identifier-1*. Si *identifier-1* no especifica un campo de fecha, *identifier-2* puede especificar uno o más campos de fecha, sin ninguna restricción sobre sus cláusulas DATE FORMAT.
- En el formato 2, *identifier-1* y *identifier-2* pueden especificar cada uno como máximo un campo de fecha. Si *identifier-1* especifica un campo de fecha, entonces FROM *identifier-2* debe ser un campo de fecha que sea compatible con el campo de fecha especificado por *identifier-1*. *identifier-3* puede especificar uno o más campos de fecha. Si *identifier-2* especifica un campo de fecha y *identifier-1* no lo especifica, entonces cada instancia de *identifier-3* debe especificar un campo de fecha que sea compatible con el campo de fecha especificado por *identifier-2*.
- En el formato 3, si un elemento dentro de *identifier-1* es un campo de fecha, el elemento correspondiente dentro de *identifier-2* debe ser un campo de fecha compatible.
- Un campo de fecha de último año sólo se permite en una sentencia SUBTRACT como *identifier-1* y cuando el resultado de la resta es una fecha no.

Hay dos pasos para determinar el resultado de una sentencia SUBTRACT que implica uno o más campos de fecha:

1. Resta: determine el resultado de la operación de resta, tal como se describe en "Resta que implica campos de fecha" en la página 249.
2. Almacenamiento: determina cómo se almacena el resultado en el campo de recepción. (En los formatos 1 y 3, el campo de recepción es *identifier-2*; en el formato 3, el campo de recepción es el DANDO *identifier-3*.) Para obtener detalles, consulte "Almacenamiento de resultados aritméticos que implican campos de fecha" en la página 249.

literal
Debe ser un literal numérico.

Los elementos de datos de coma flotante y literales se pueden utilizar en cualquier lugar donde se puedan especificar elementos de datos numéricos y literales.

Frase REDONDEADA

Para obtener información sobre la frase REDONDEADA y para consideraciones de operando, consulte "Frase REDONDEADA" en la página 281.

Frases de ERROR DE TAMAÑO

Para obtener información sobre las frases SIZE ERROR y para consideraciones de operando, consulte "Frases de ERROR DE TAMAÑO" en la página 281.

Frase CORRESPONDIENTE (formato 3)

Consulte "Frase CORRESPONDIENTE" en la página 279.

Frase END-SUBTRACT

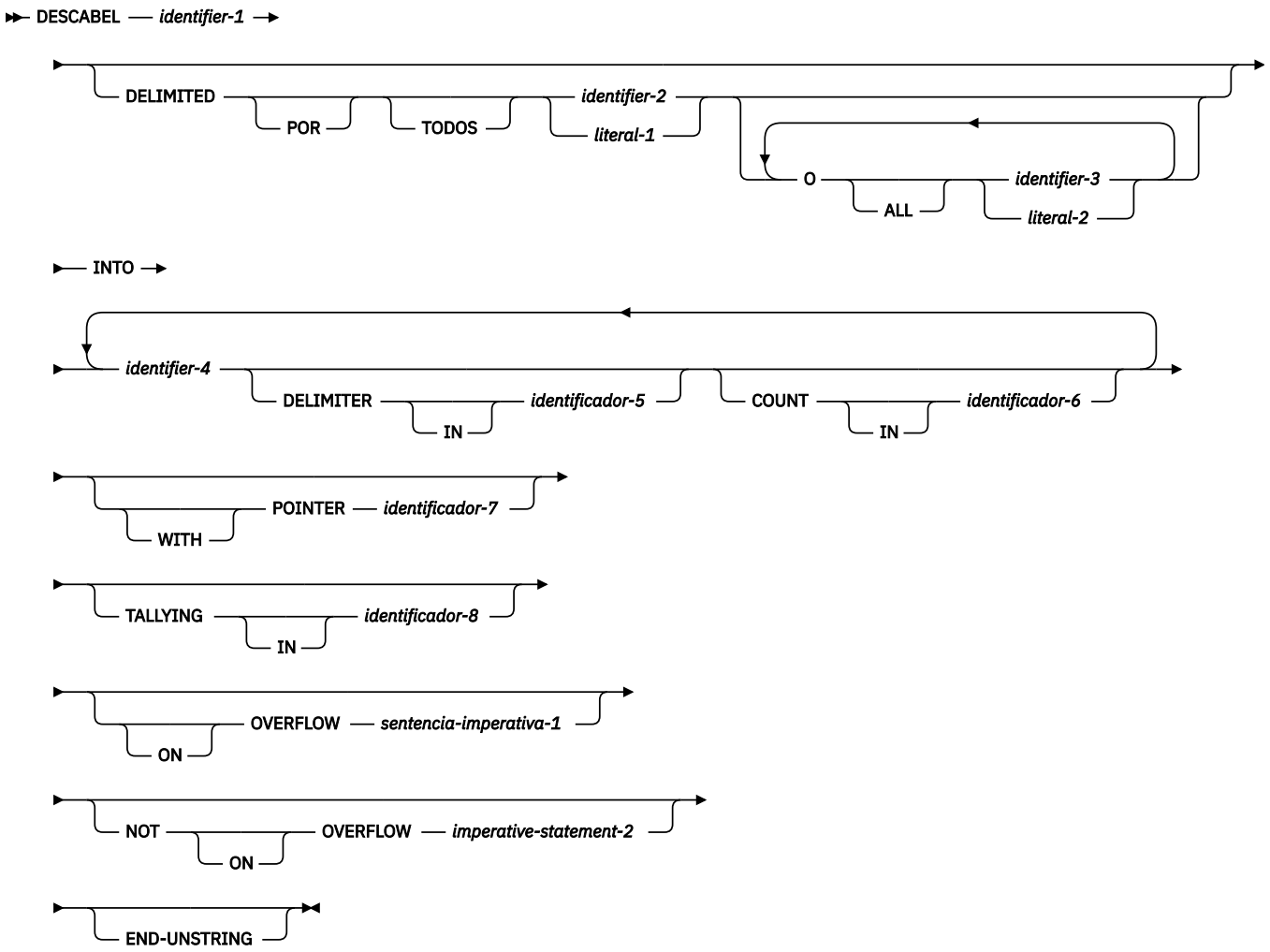
Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia SUBTRACT. END-SUBTRACT permite que una sentencia SUBTRACT condicional se anide en otra sentencia condicional. END-SUBTRACT también se puede utilizar con una sentencia SUBTRACT imperativa.

Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

Sentencia UNSTRING

La sentencia UNSTRING hace que los datos contiguos de un campo de envío se separen y se coloquen en varios campos de recepción.

Formato



identifier-1

Representa el *campo de envío*. Los datos se transfieren desde este campo a los campos de recepción de datos (*identifier-4*).

identifier-1 debe hacer referencia a un elemento de datos de categoría alfabética, alfanumérica, editada alfanumérica, DBCS, nacional o editada nacional.

identifier-2*, *literal-1*, *identifier-3*, *literal-2

Especifica uno o más delimitadores.

identifier-2 y *identifier-3* deben hacer referencia a elementos de datos de categoría alfabética, alfanumérica, editada alfanumérica, DBCS, nacional o editada nacional.

literal-1 o *literal-2* debe ser de categoría alfanumérica, DBCS o nacional y no debe ser una constante figurativa que empiece por la palabra ALL.

identifier-4

Especifica uno o más campos de recepción.

identifier-4 debe hacer referencia a un elemento de datos de categoría alfabética, alfanumérica, numérica, DBCS o nacional. Si el elemento de datos referenciado es de categoría numérica, su serie de caracteres de imagen no debe contener el símbolo de imagen P, y su uso debe ser DISPLAY o NATIONAL.

identifier-5
Especifica un campo para recibir el delimitador asociado con *identifier-4*.

Identifier-5 debe hacer referencia a un elemento de datos de categoría alfabética, alfanumérica, DBCS o nacional.

identifier-6
Especifica un campo que contiene el recuento de caracteres que se transfieren al *identifier-4*.

identifier-6 debe ser un elemento de datos entero definido sin el símbolo P en su serie de caracteres PICTURE.

identifier-7
Especifica un campo que debe contener una posición de carácter relativo durante el proceso UNSTRING.

identifier-7 debe ser un elemento de datos entero definido sin el símbolo P en la serie PICTURE.

identifier-7 debe describirse como un elemento de datos de tamaño suficiente para contener un valor igual a 1 más el número de posiciones de caracteres en el elemento de datos al que hace referencia *identifier-1*.

identifier-8
Especifica un campo que se incrementa en el número de campos delimitados procesados.

identifier-8 debe ser un elemento de datos entero definido sin el símbolo P en su serie de caracteres PICTURE.

Se aplican las reglas siguientes:

- Si *identifier-4* hace referencia a un elemento de datos de uso DISPLAY, *identifier-1*, *identifier-2*, *identifier-3*, y *identifier-5* también deben hacer referencia a elementos de datos de uso DISPLAY y todos los literales deben ser literales alfanuméricos. Se puede especificar cualquier constante figurativa excepto NULL o una que empiece por la palabra ALL. Cada constante figurativa representa un literal alfanumérico de 1 carácter.
- Si *identifier-4* hace referencia a un elemento de datos de uso DISPLAY-1, *identifier-1*, *identifier-2*, *identifier-3*, y *identifier-5* también deben hacer referencia a elementos de datos de uso DISPLAY-1 y todos los literales deben ser literales DBCS. La constante figurativa SPACE es la única constante figurativa que se puede especificar. Cada constante figurativa representa un literal DBCS de 1 carácter.
- Si *identifier-4* hace referencia a un elemento de datos de uso NATIONAL, *identifier-1*, *identifier-2*, *identifier-3*, y *identifier-5* también deben hacer referencia a elementos de datos de uso NATIONAL y todos los literales deben ser literales nacionales. Se puede especificar cualquier constante figurativa excepto NULL o una que empiece por la palabra ALL. Cada constante figurativa representa un literal nacional de 1 carácter.
- Ninguno de los identificadores de una sentencia UNSTRING puede ser campos de fecha con ventanas.

Los campos de recuento (*identifier-6*) y los campos de puntero (*identifier-7*) se incrementan por el número de posiciones de caracteres (alfanuméricas, DBCS o nacionales), no por el número de bytes.

Una sentencia UNSTRING puede tomar el lugar de una serie de sentencias MOVE, excepto que la evaluación o el cálculo de determinados elementos se realiza una sola vez, al principio de la ejecución de la sentencia UNSTRING. Para obtener más información, consulte "Valores al final de la ejecución de la sentencia UNSTRING" en la página 422.

Las reglas para mover son las mismas que para una sentencia MOVE para un elemento de envío elemental de la categoría de *identifier-1*, con el *identifier-4* adecuado como elemento receptor (consulte "sentencia MOVE" en la página 347). Por ejemplo, las reglas para mover un elemento DBCS se utilizan cuando *identifier-1* es un elemento DBCS.

Frase DELIMITED BY

Esta frase especifica delimitadores dentro de los datos que controlan la transferencia de datos.

Cada *identifier-2*, *identifier-3*, *literal-1*, o *literal-2* representa un delimitador.

Si la frase DELIMITED BY *no* se especifica, *no* se deben especificar las frases DELIMITER IN y COUNT IN.

all
Varias apariciones contiguas de cualquier delimitador se tratan como si solo hubiera una aparición; esta única aparición se mueve al campo de recepción de delimitador (*identifier-5*), si se especifica. Los caracteres delimitadores en el campo de envío se tratan como un elemento elemental del mismo uso y categoría que *identifier-1* y se mueven al campo de recepción de delimitador actual de acuerdo con las reglas de la sentencia MOVE.

Cuando *no* se especifica DELIMITED BY ALL y se encuentran dos o más apariciones contiguas de cualquier delimitador, el campo de recepción de datos actual (*identifier-4*) se rellena con espacios o ceros, según la descripción del campo de recepción de datos.

Delimitador con dos o más caracteres

Un delimitador que contiene dos o más caracteres se reconoce como delimitador sólo si los caracteres delimitadores están en los dos casos siguientes:

- Contiguo
- En la secuencia especificada en el campo de envío

Dos o más delimitadores

Cuando se especifican dos o más delimitadores, existe una condición OR y cada aparición no solapada de cualquiera de los delimitadores se reconoce en el campo de envío en la secuencia especificada.

Por ejemplo:

```
DELIMITED BY "AB" or "BC"
```

Una aparición de AB o BC en el campo de envío se considera un delimitador. Una aparición de ABC se considera una aparición de AB.

Frase INTO

Esta frase especifica los campos a los que se van a mover los datos.

identifier-4 representa los *campos de recepción de datos*.

DELIMITADOR EN
Esta frase especifica los campos donde se van a mover los delimitadores.

identifier-5 representa los *campos de recepción de delimitador*.

La frase DELIMITER IN *no* debe especificarse si la frase DELIMITED BY *no* se especifica.

CUENTA EN
Esta frase especifica el campo en el que se mantiene el recuento de posiciones de caracteres examinadas.

identifier-6 es el *campo de recuento de datos* para cada transferencia de datos. Cada campo contiene el recuento de posiciones de caracteres examinadas en el campo de envío, terminadas por los delimitadores o el final del campo de envío, para el movimiento a este campo de recepción. Los delimitadores no se incluyen en este recuento.

La frase COUNT IN *no* debe especificarse si la frase DELIMITED BY *no* se especifica.

Frase POINTER

Cuando se especifica la frase POINTER, el valor del campo de puntero, *identifier-7*, se comporta como si se incrementara en 1 para cada posición de carácter examinada en el campo de envío. Cuando se

completa la ejecución de la sentencia UNSTRING, el campo de puntero contiene un valor igual a su valor inicial más el número de posiciones de caracteres examinadas en el campo de envío.

Cuando se especifica esta frase, el usuario debe inicializar el campo de puntero antes de que comience la ejecución de la sentencia UNSTRING.

frase TALLYING IN

Cuando se especifica la frase TALLYING, el campo de recuento de área, *identifier-8*, contiene (al final de la ejecución de la sentencia UNSTRING) un valor igual al valor inicial más el número de áreas de recepción de datos sobre las que se ha actuado.

Cuando se especifica esta frase, el usuario debe inicializar el campo de recuento de área antes de que comience la ejecución de la sentencia UNSTRING.

Frases ON OVERFLOW

Existe una condición de desbordamiento cuando:

- El valor del puntero (explícito o implícito) es menor que 1.
- El valor de puntero (explícito o implícito) excede un valor igual a la longitud del campo de envío.
- Se ha actuado sobre todos los campos de recepción de datos y el campo de envío todavía contiene posiciones de caracteres no examinadas.

Cuando se produce una condición de desbordamiento

Una condición de desbordamiento da como resultado las acciones siguientes:

1. No se transfieren más datos.
2. La operación UNSTRING ha terminado.
3. La frase NOT ON OVERFLOW, si se especifica, se ignora.
4. El control se transfiere al final de la sentencia UNSTRING o, si se especifica la frase ON OVERFLOW, a *imperative-statement-1*.

imperative-statement-1
Sentencia o sentencias para tratar con una condición de desbordamiento.

Si el control se transfiere a *imperative-statement-1*, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en imperativo- *statement-1*. Si se ejecuta una ramificación de procedimiento o una sentencia condicional que provoca una transferencia explícita de control, el control se transfiere de acuerdo con las reglas de dicha sentencia. De lo contrario, al finalizar la ejecución de *imperative-statement-1*, el control se transfiere al final de la sentencia UNSTRING.

Cuando no se produce una condición de desbordamiento

Cuando, durante la ejecución de una sentencia UNSTRING, no se encuentran las condiciones que causarían una condición de desbordamiento, entonces:

1. La transferencia de datos se ha completado.
2. La frase ON OVERFLOW, si se especifica, se ignora.
3. El control se transfiere al final de la sentencia UNSTRING o, si se especifica la frase NOT ON OVERFLOW, a *imperative-statement-2*.

imperative-statement-2
Sentencia o sentencias para tratar con una condición de desbordamiento que no se produce.

Si el control se transfiere a *imperative-statement-2*, la ejecución continúa de acuerdo con las reglas para cada sentencia especificada en imperativo- *statement-2*. Si se ejecuta una ramificación de procedimiento o una sentencia condicional que provoca una transferencia explícita de control, el

control se transfiere de acuerdo con las reglas de dicha sentencia. De lo contrario, al finalizar la ejecución de *imperative-statement-2*, el control se transfiere al final de la sentencia UNSTRING.

frase END-UNSTRING

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia UNSTRING. END-UNSTRING permite que una sentencia UNSTRING condicional se anide en otra sentencia condicional. END-UNSTRING también puede utilizarse con una sentencia imperativa UNSTRING.

Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

Flujo de datos

El flujo de datos para la sentencia UNSTRING se basa en determinadas reglas.

Cuando se inicia la sentencia UNSTRING, los datos se transfieren desde el campo de envío al campo de recepción de datos actual, de acuerdo con las reglas siguientes:

Etapa 1: Examinar

1. Si se especifica la frase POINTER, se examina el campo, empezando por la posición de carácter relativo especificada por el valor del campo de puntero.

   Si la frase POINTER *no* se especifica, se examina la serie de caracteres del campo emisor, empezando por la posición de carácter más a la izquierda.
2. Si se especifica la frase DELIMITED BY, el examen continúa de izquierda a derecha, examinando las posiciones de carácter una por una hasta que se encuentre un delimitador. Si se alcanza el final del campo de envío antes de que se encuentre un delimitador, el examen finaliza con la última posición de carácter en el campo de envío. Si hay más campos de recepción, se selecciona el siguiente; de lo contrario, se produce una condición de desbordamiento.

   Si la frase DELIMITED BY *no* se especifica, el número de posiciones de caracteres examinadas es igual al tamaño del campo de recepción de datos actual, tal como se describe en la tabla siguiente. El tamaño depende del tratamiento de categoría del campo de recepción, tal como se muestra en la Tabla 48 en la página 339.

Tabla 58. ***Posiciones de caracteres examinadas cuando no se especifica DELIMITED BY***

Si el campo de recepción es ...	**El número de posiciones de carácter examinadas es ...**
Alfanumérico o alfabético	Igual al número de posiciones de caracteres alfanuméricos en el campo de recepción actual
DBCS	Igual al número de posiciones de caracteres DBCS en el campo receptor actual
Nacional	Igual al número de posiciones de caracteres nacionales en el campo de recepción actual
Numérico	Igual al número de posiciones de caracteres en la parte entera del campo de recepción actual
Se describe con la cláusula SIGN IS SEPARATE	1 menor que el tamaño del campo receptor actual
Se describe como un elemento de datos de longitud variable	Determinado por el tamaño del campo de recepción actual al principio de la operación UNSTRING

Etapa 2: Mover

3. Las posiciones de caracteres examinadas (excluyendo cualquier carácter delimitador) se tratan como un elemento de datos elemental de la misma categoría de datos que el campo de envío, excepto para los casos que se muestran en la tabla siguiente.

Categoría de *identifier-1* (sending-field)	**Categoría de elemento de datos elemental**
Alfanumérico-editado	Alfanumérico
Nacional-editado	Nacional

   Ese elemento de datos elemental se mueve al campo de recepción de datos actual de acuerdo con las reglas para la sentencia MOVE para las categorías de los campos de envío y recepción tal como se describe en "sentencia MOVE" en la página 347.
4. Si se especifica la frase DELIMITER IN, los caracteres delimitadores del campo de envío se tratan como un elemento alfanumérico elemental y se mueven al campo de recepción del delimitador actual, de acuerdo con las reglas de la sentencia MOVE. Si la condición de delimitador es el final del campo de envío, el campo de recepción de delimitador actual se rellena con espacios.
5. Si se especifica la frase COUNT IN, se mueve un valor igual al número de posiciones de caracteres examinadas (excluyendo cualquier delimitador) al campo de recuento de datos, de acuerdo con las reglas para un movimiento elemental.

   Etapa 3: Iteraciones sucesivas
6. Si se especifica la frase DELIMITED BY, el campo de envío se examina más a fondo, empezando por la primera posición de carácter a la derecha del delimitador.

   Si la frase DELIMITED BY *no* se especifica, el campo de envío se examina más a fondo, empezando por la primera posición de carácter a la derecha de la última posición de carácter examinada.
7. Para cada campo de recepción de datos satisfactorio, este proceso de examen y movimiento se repite hasta que se produce una de las siguientes condiciones:
   - Se han transferido todos los caracteres del campo de envío.
   - No hay más campos de recepción de datos sin rellenar.

Valores al final de la ejecución de la sentencia UNSTRING

Al principio de la ejecución de la sentencia UNSTRING, ciertas operaciones se realizan sólo una vez.

Las operaciones son:

- Cálculos de subíndices, modificaciones de referencia, longitudes variables, ubicaciones variables
- Evaluaciones de funciones

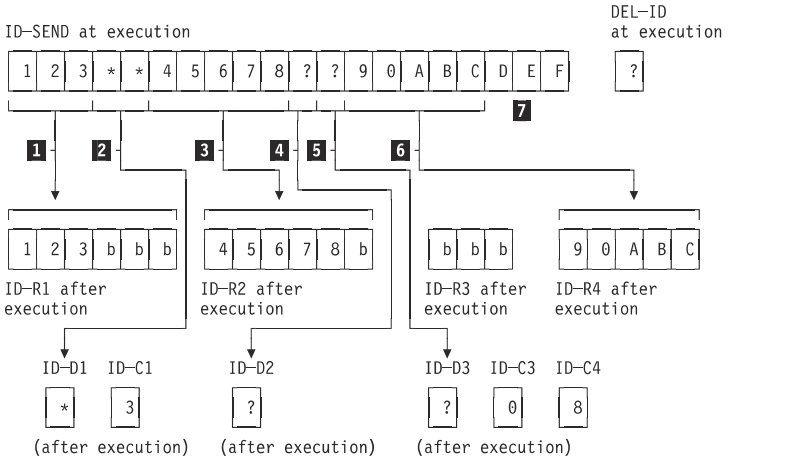
Por lo tanto, si *identifier-4*, *identifier-5*, *identifier-6*, *identifier-7*, o *identifier-8* se utiliza como subíndice, modificador de referencia o argumento de función en la sentencia UNSTRING, o afecta a la longitud o ubicación de cualquiera de los identificadores de la sentencia UNSTRING, estos valores se determinan al principio de la sentencia UNSTRING y *no* se ven afectados por ningún resultado de la sentencia UNSTRING.

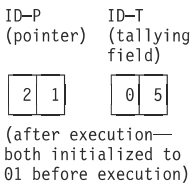
Ejemplo de la sentencia UNSTRING

Este tema lista un ejemplo para la sentencia UNSTRING.

La figura siguiente muestra los resultados de ejecución de un ejemplo de la sentencia UNSTRING.

```
UNSTRING ID-SEND DELIMITED BY DEL-ID OR ALL "*"
    INTO ID-R1 DELIMITER IN ID-D1 COUNT IN ID-C1      (All the data
         ID-R2 DELIMITER IN ID-D2                     receiving fields
         ID-R3 DELIMITER IN ID-D3 COUNT IN ID-C3      are defined as
         ID-R4 COUNT IN ID-C4                         alphanumeric)
    WITH POINTER ID-P
    TALLYING IN ID-T
    ON OVERFLOW GO TO OFLOW-EXIT.
```





The order of execution is:

1. 3 characters are placed in ID-R1.
2. Because ALL * is specified, all consecutive asterisks are processed, but only one asterisk is placed in ID-D1.
3. 5 characters are placed in ID-R2.
4. A ? is placed in ID-D2. The current receiving field is now ID-R3.
5. A ? is placed in ID-D3; ID-R3 is filled with spaces; no characters are transferred, so 0 is placed in ID-C3.
6. No delimiter is encountered before 5 characters fill ID-R4; 8 is placed in ID-C4, representing the number of characters examined since the last delimiter.
7. ID-P is updated to 21, the total length of the sending field + 1; ID-T is updated to 5, the number of fields acted upon + 1. Since there are no unexamined characters in the ID-SEND, the OVERFLOW EXIT is not taken.

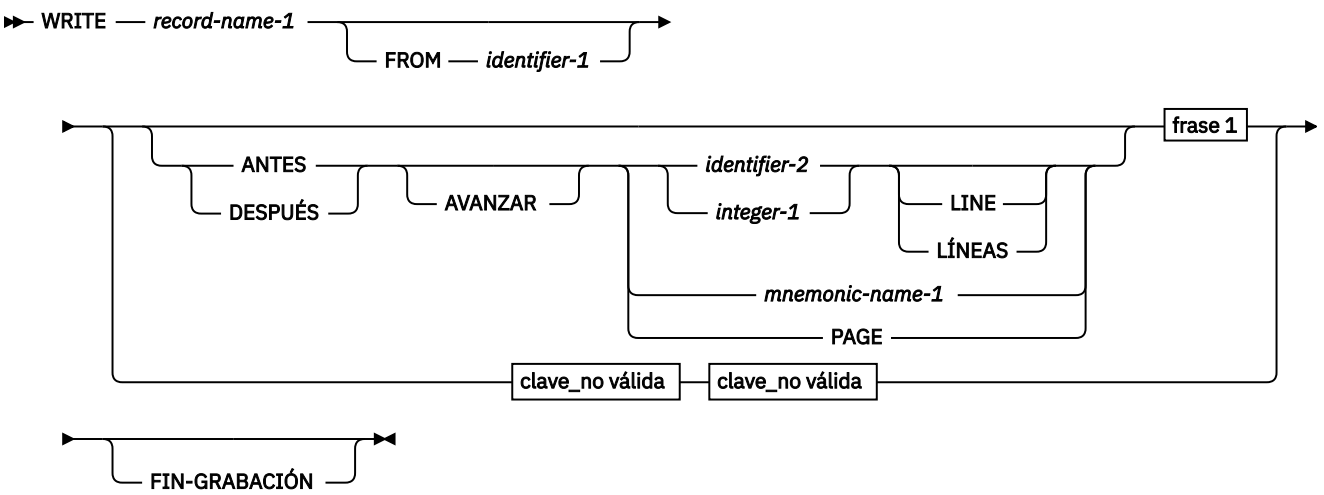
sentencia WRITE

La sentencia WRITE libera un registro lógico en un archivo de salida o de entrada/salida.

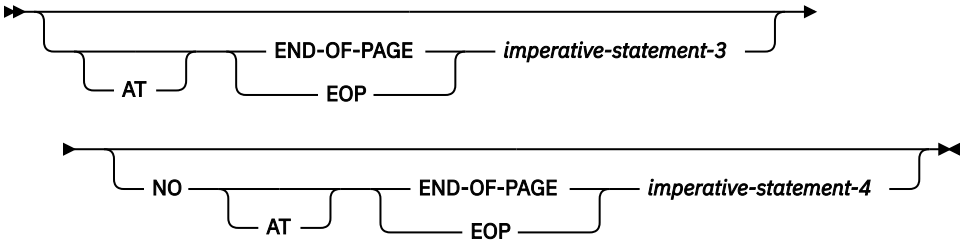
Cuando se ejecuta la sentencia WRITE:

- El archivo secuencial asociado debe estar abierto en modalidad OUTPUT o EXTEND.
- El archivo indexado o relativo asociado debe estar abierto en modalidad OUTPUT, I-O o EXTEND.

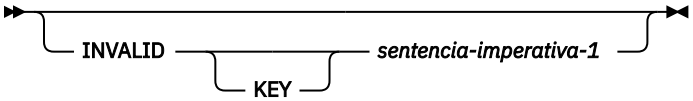
Formato 1: sentencia WRITE para archivos secuenciales



frase 1

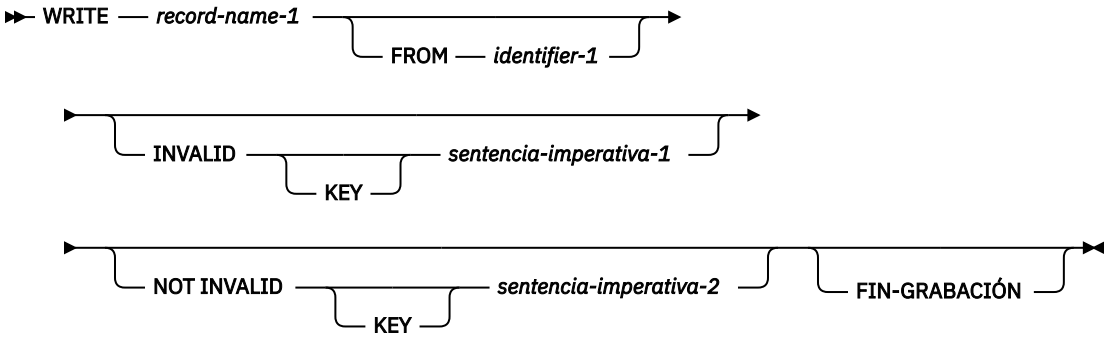


clave_no válida

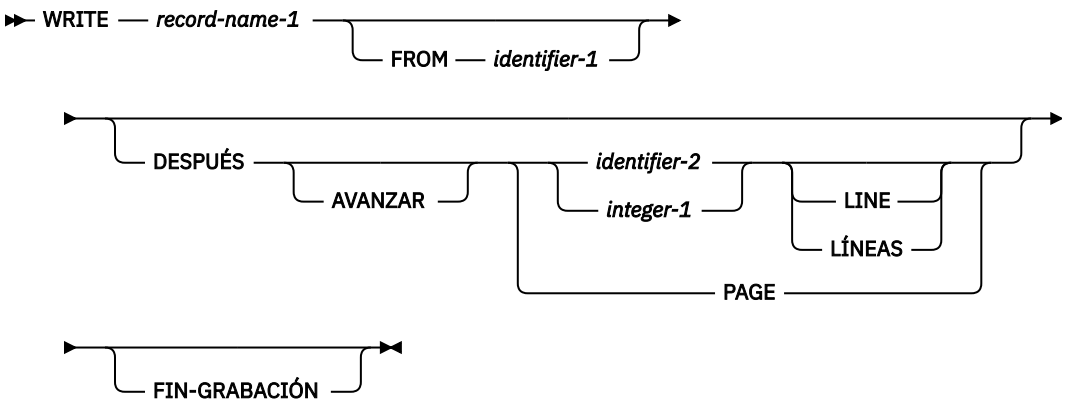


no_clave_no válida

Formato 2: sentencia WRITE para archivos indexados y relativos



Formato 3: sentencia WRITE para archivos secuenciales de línea



record-name-1
Debe definirse en una entrada DATA DIVISION FD. *record-name-1* se puede calificar. No debe estar asociado con un archivo de clasificación o fusión.

Para archivos relativos, el número de posiciones de caracteres en el registro que se está grabando puede ser diferente del número de posiciones de caracteres en el registro que se está sustituyendo.

Frase FROM
El resultado de la ejecución de la sentencia WRITE con la frase FROM *identifier-1* es equivalente a la ejecución de las sentencias siguientes en el orden especificado:

```
MOVE identifier-1 TO record-name-1.
WRITE record-name-1.
```

El MOVE se realiza de acuerdo con las reglas para una sentencia MOVE sin la frase CORRESPONDIENTE.

identifier-1
identifier-1 puede hacer referencia a cualquiera de los elementos siguientes:

- Un elemento de datos definido en WORKING-STORAGE SECTION, LOCAL-STORAGE SECTION o LINKAGE SECTION
- Una descripción de registro para otro archivo abierto anteriormente
- Una función alfanumérica
- Una función nacional

identifier-1 debe ser un elemento de envío válido para una sentencia MOVE con *record-name-1* como elemento de recepción.

identifier-1 y *record-name-1* no deben hacer referencia a la misma área de almacenamiento.

Después de ejecutar la sentencia WRITE, la información sigue estando disponible en *identifier-1*. (Consulte “Frases INTO y FROM” en la página 290 en "Instalaciones de proceso comunes".)

identifier-2
Debe ser un elemento de datos entero.

Frase AVANZANDO

La frase AVANZANDO controla la posición del registro de salida en la página.

Cuando se utiliza WRITE AVANZANDO con los nombres de entorno C01-C012 o S01-S05, una línea es avanzada.

Reglas de frase AVANZANDO

Cuando se especifica la frase AVANZANDO, se aplican las reglas siguientes:

1. Cuando se especifica BEFORE AVANZAR, la línea se imprime antes de que la página sea avanzada.
2. Cuando se especifica AFTER AVANZAR, la página se avanza antes de que se imprima la línea.
3. Cuando se especifica *identifier-2* , la página avanza el número de líneas igual al valor actual en *identifier-2*. *identifier-2* debe nombrar un elemento de datos entero elemental. *identifier-2* no puede nombrar un campo de fecha con ventana.
4. Cuando se especifica un entero, la página avanza el número de líneas igual al valor de entero.
5. El entero o el valor en *identifier-2* puede ser cero.
6. Cuando se especifica PAGE, el registro se imprime en la página lógica BEFORE o AFTER (en función de la frase utilizada), el dispositivo se coloca en la siguiente página lógica. Si PAGE no tiene significado para el dispositivo utilizado, se proporciona BEFORE o AFTER (dependiendo de la frase especificada) AVANZANDO 1 LÍNEA.

   Si la entrada FD contiene una cláusula LINAGE, el reposicionamiento se realiza en la primera línea imprimible de la página siguiente, tal como se especifica en dicha cláusula. Si se omite la cláusula LINAGE, el reposicionamiento se realiza en la línea 1 de la siguiente página siguiente.

Cuando se omite la frase AVANZANDO, se proporciona el avance automático de la línea, como si se hubiera especificado DESPUÉS DE AVANZAR 1 LÍNEA.

LINAGE-Reglas COUNTER

Si se especifica la cláusula LINAGE para este archivo, el registro especial LINAGE-COUNTER asociado se modifica durante la ejecución de la sentencia WRITE, de acuerdo con las reglas siguientes:

1. Si se especifica AVANZANDO PÁGINA, LINAGE-COUNTER se restablece en 1.
2. Si se especifica AVANZANDO *identifier-2* o *entero* , LINAGE-COUNTER se incrementa con el valor de *identifier-2* o *integer*.
3. Si se omite la frase AVANZANDO, LINAGE-COUNTER se incrementa en 1.
4. Cuando el dispositivo se vuelve a colocar en la primera línea disponible de una página nueva, LINAGE-COUNTER se restablece en 1.

frases END-OF-PAGE

Cuando se especifica END-OF-PAGE y se alcanza el final lógico de la página impresa durante la ejecución de la sentencia WRITE, se ejecuta la sentencia imperativa END-OF-PAGE. Cuando se especifica la frase END-OF-PAGE, la entrada FD para este archivo debe contener una cláusula LINAGE.

El final lógico de la página impresa se especifica en la cláusula LINAGE asociada.

Se alcanza una condición END-OF-PAGE cuando la ejecución de una sentencia WRITE END-OF-PAGE provoca la impresión o el espaciado dentro del área de pie de página de un cuerpo de página. Esto ocurre cuando la ejecución de una sentencia WRITE de este tipo hace que el valor del registro especial LINAGE-COUNTER sea igual o superior al valor especificado en la frase WITH CABECERA de la cláusula LINAGE. Se ejecuta la sentencia WRITE y, a continuación, se ejecuta la sentencia imperativa END-OF-PAGE.

Se alcanza una condición de desbordamiento de página automática siempre que la ejecución de una sentencia WRITE determinada (con o sin la frase END-OF-PAGE) no se pueda ejecutar completamente dentro del cuerpo de página actual. Esto ocurre cuando una sentencia WRITE, si se ejecuta, haría que el valor de LINAGE-COUNTER excediera el número de líneas para el cuerpo de página especificado en la cláusula LINAGE. En este caso, la línea se imprime ANTES o DESPUÉS (dependiendo de la opción especificada) el dispositivo se vuelve a colocar en la primera línea imprimible en la siguiente página

lógica, tal como se especifica en la cláusula LINAGE. Si se especifica la frase END-OF-PAGE, se ejecuta la sentencia imperativa END-OF-PAGE.

Si no se especifica la frase WITH CABECERA de la cláusula LINAGE, la condición de desbordamiento automático de página existe porque no se puede detectar ninguna condición de fin de página (como distinta de la condición de desbordamiento de página).

Si se especifica la frase WITH INICIAL, pero la ejecución de una sentencia WRITE determinada provocaría que LINAGE-COUNTER excediera tanto el valor de pie como el valor de cuerpo de página especificados en la cláusula LINAGE, entonces la condición de fin de página y la condición de desbordamiento automático de página se producen simultáneamente.

Las palabras clave END-OF-PAGE y EOP son equivalentes.

Puede especificar la frase AVANZANDO PÁGINA y la frase END-OF-PAGE en una sola sentencia WRITE.

Frases INVALID KEY

Una condición de clave no válida se debe a los casos siguientes:

- Para archivos secuenciales, se realiza un intento de grabar más allá del límite definido externamente del archivo.
- Para archivos indexados:
  - Se ha intentado grabar más allá del límite definido externamente del archivo.
  - Se especifica ACCESS SEQUENTIAL y se abre el archivo OUTPUT, y el valor de la clave de registro principal no es mayor que el del registro anterior.
  - El archivo se abre OUTPUT o I-O y el valor de la clave de registro principal es igual al de un registro ya existente.
- Para archivos relativos:
  - Se ha intentado grabar más allá del límite definido externamente del archivo.
  - Cuando la modalidad de acceso es aleatoria o dinámica y el elemento de datos RELATIVE KEY especifica un registro que ya existe en el archivo.
  - El número de dígitos significativos en el número de registro relativo es mayor que el tamaño del elemento de datos de clave relativa para el archivo.

Cuando se produce una condición de clave no válida:

- Si se especifica la frase INVALID KEY, se ejecuta *imperative-statement-1* . Para obtener detalles sobre el proceso de claves no válidas, consulte Condición de clave no válida.
- De lo contrario, la sentencia WRITE no se ejecuta correctamente y el contenido de *nombre-registro* no se ve afectado y se produce el siguiente caso:
  - Para archivos secuenciales, la clave de estado de archivo, si se especifica, se actualiza y existe una condición EXCEPTION/ERROR.

    Si se especifica un procedimiento EXCEPTION/ERROR explícito o implícito para el archivo, se ejecuta el procedimiento. Si no se especifica ningún procedimiento de este tipo, los resultados son imprevisibles.
  - Para los archivos relativos e indexados, la ejecución del programa continúa de acuerdo con las reglas descritas por Condición de clave no válida en "Instalaciones de proceso comunes".

    Las condiciones INVALID KEY que se aplican a un archivo relativo en modalidad OPEN OUTPUT también se aplican a una en modalidad OPEN EXTEND.
- Si se especifica la frase NOT INVALID KEY y existe una condición de clave válida al final de la ejecución de la sentencia WRITE, el control se pasa a *imperative-statement-4*.

Se puede omitir la frase INVALID KEY y un procedimiento EXCEPTION/ERROR aplicable.

frase END-WRITE

Este terminador de ámbito explícito sirve para delimitar el ámbito de la sentencia WRITE. END-WRITE permite que una sentencia WRITE condicional se anide en otra sentencia condicional. END-WRITE también se puede utilizar con una sentencia WRITE imperativa.

Para obtener más información, consulte "Sentencias de ámbito delimitado" en la página 278.

WRITE para archivos secuenciales

El tamaño máximo de registro para archivos secuenciales se establece en el momento en que se crea el archivo y no se puede cambiar posteriormente.

Después de ejecutar la sentencia WRITE, el registro lógico ya no está disponible en *record-name-1* a menos que:

- El archivo asociado se denomina en una cláusula SAME RECORD AREA (en cuyo caso, el registro también está disponible como registro de los otros archivos nombrados en la cláusula SAME RECORD AREA)
- La sentencia WRITE no es satisfactoria debido a una violación de límite.

En cualquiera de estos dos casos, el registro lógico sigue estando disponible en *record-name-1*.

El indicador de posición de archivo no se ve afectado por la ejecución de la sentencia WRITE.

El número de posiciones de caracteres necesarias para almacenar el registro en un archivo puede o no ser el mismo que el número de posiciones de caracteres definidas por la descripción lógica de dicho registro en el programa COBOL. (Consulte "Edición de cláusula PICTURE" en la página 206 y "cláusula USAGE" en la página 225.)

Si se especifica la cláusula FILE STATUS en la entrada de control de archivos, la clave de estado de archivo asociada se actualiza cuando se ejecuta la sentencia WRITE, tanto si la ejecución es satisfactoria como si no.

La sentencia WRITE sólo se puede ejecutar para un archivo secuencial abierto en la modalidad OUTPUT .

WRITE para archivos indexados

Antes de ejecutar la sentencia WRITE para archivos indexados, debe establecer la clave de registro principal (el elemento de datos RECORD KEY, tal como se define en la entrada de control de archivos) en el valor required . Tenga en cuenta que los valores de RECORD KEY deben ser exclusivos dentro de un archivo.

Si la cláusula ALTERNATE RECORD KEY también se especifica en la entrada de control de archivos, cada clave de registro alternativa debe ser exclusiva, a menos que se especifique la frase DUPLICATES. Si se especifica la frase DUPLICATES, es posible que los valores de clave de registro alternativos no sean exclusivos. En este caso, el sistema almacena los registros para que el acceso secuencial posterior a los registros permita la recuperación en el mismo orden en el que se almacenaron.

Cuando se especifica ACCESS IS SEQUENTIAL en la entrada de control de archivos, los registros deben liberarse en orden ascendente de valores RECORD KEY.

Cuando se especifica ACCESS IS RANDOM o ACCESS IS DYNAMIC en la entrada de control de archivos, los registros pueden liberarse en cualquier orden especificado por el programador.

WRITE para archivos relativos

Para los archivos OUTPUT de registro relativo, la sentencia WRITE provoca acciones tal como se describe en el tema.

- Si se especifica ACCESS IS SEQUENTIAL:

  El primer registro publicado tiene el número de registro relativo 1, el segundo registro publicado tiene el número de registro relativo 2, el tercer número 3, etc.

Si se especifica RELATIVE KEY en la entrada de control de archivos, el número de registro relativo del registro que se acaba de liberar se coloca en RELATIVE KEY durante la ejecución de la sentencia WRITE.

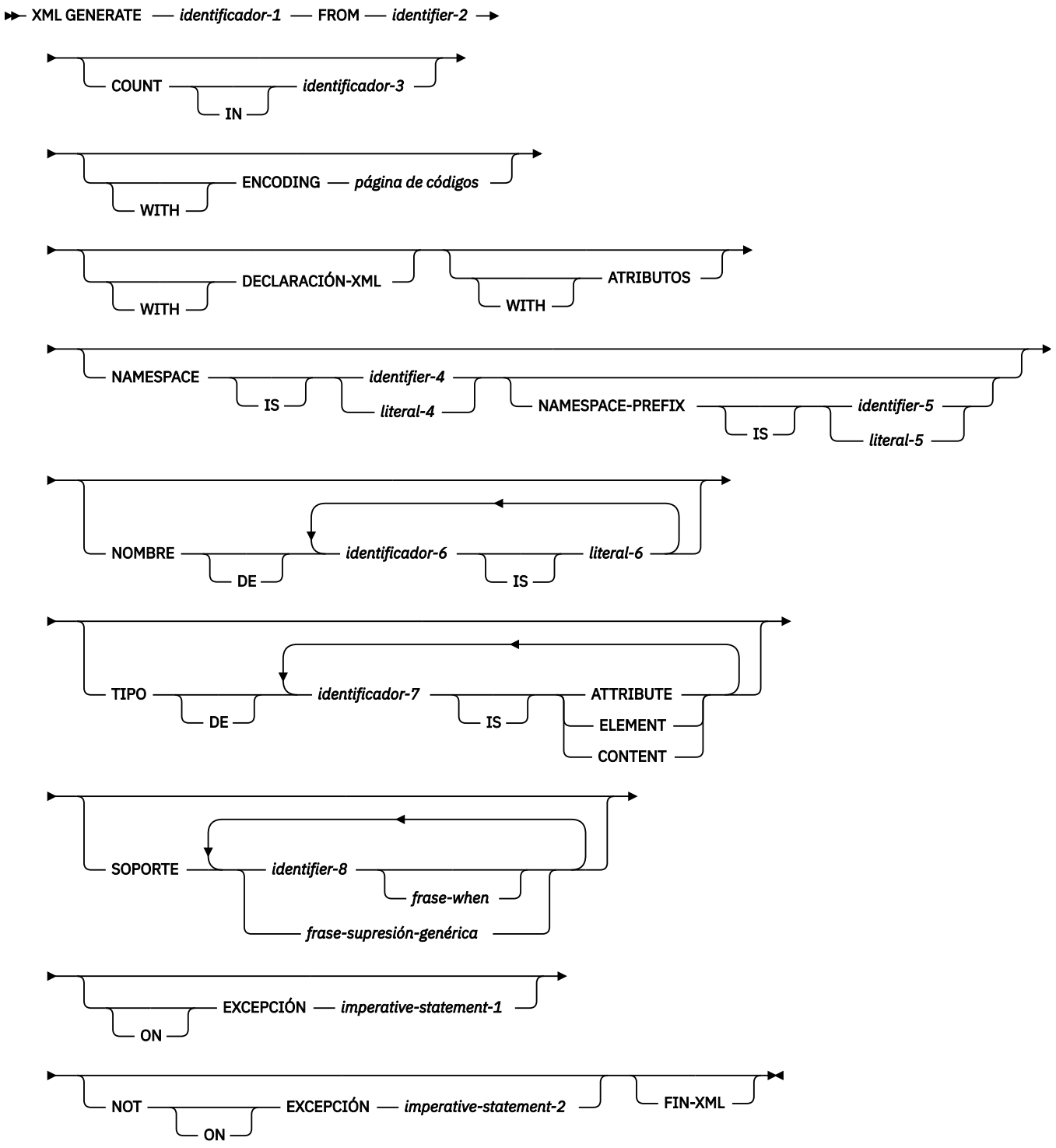
- Si se especifica ACCESS IS RANDOM o ACCESS IS DYNAMIC, RELATIVE KEY debe contener el número de registro relativo necesario para este registro antes de ejecutar la sentencia WRITE. Cuando se ejecuta la sentencia WRITE, este registro se coloca en la posición de número de registro relativo especificada en el archivo.

Para los archivos I-O, se debe especificar ACCESS IS RANDOM o ACCESS IS DYNAMIC; la sentencia WRITE inserta nuevos registros en el archivo. La RELATIVE KEY debe contener el número de registro relativo necesario para este registro antes de que se ejecute la sentencia WRITE. Cuando se ejecuta la sentencia WRITE, este registro se coloca en la posición de número de registro relativo especificada en el archivo.

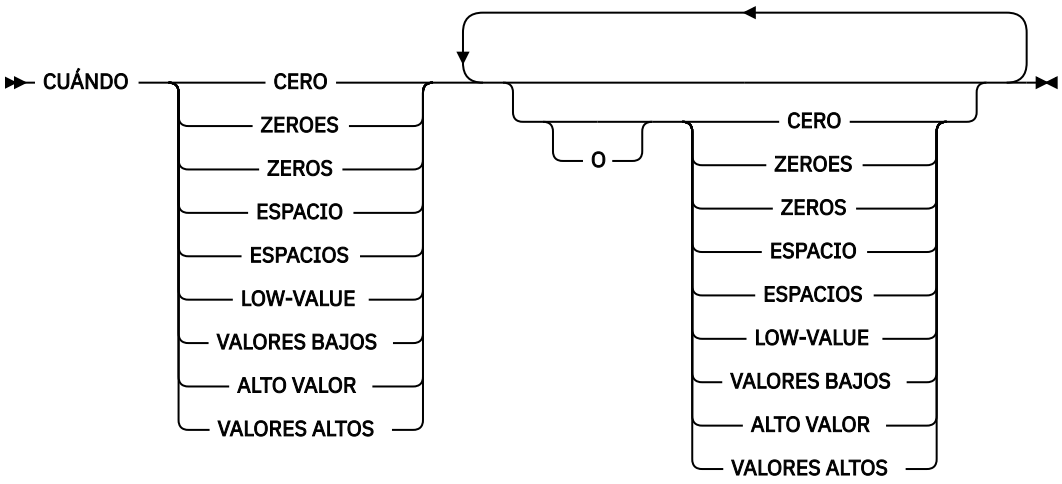
sentencia XML GENERATE

La sentencia XML GENERATE convierte los datos al formato XML.

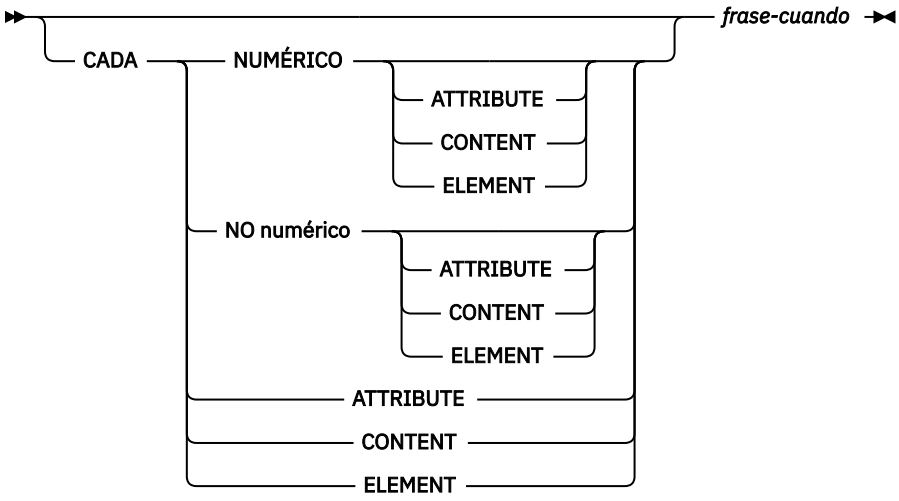
Formato


XML GENERATE
identificador-1
FROM
identifier-2
COUNT
IN
identificador-3
WITH
ENCODING
página de códigos
WITH
DECLARACIÓN-XML
WITH
ATRIBUTOS
NAMESPACE
IS
identifier-4
literal-4
NAMESPACE-PREFIX
IS
identifier-5
literal-5
NOMBRE
DE
identificador-6
IS
literal-6
TIPO
DE
identificador-7
IS
ATTRIBUTE
ELEMENT
CONTENT
SOPORTE
identifier-8
frase-when
frase-supresión-genérica
EXCEPCIÓN
ON
imperative-statement-1
NOT
ON
EXCEPCIÓN
imperative-statement-2
FIN-XML

Formato de *frase-when*



Formato *frase-supresión-genérica*



identifier-1

El área de recepción para un documento XML generado. *identifier-1* debe hacer referencia a uno de los elementos siguientes:

- Un elemento de datos elemental de categoría alfanumérica
- Un elemento de grupo alfanumérico
- Un elemento de datos elemental de categoría nacional
- Un elemento de grupo nacional

Cuando *identifier-1* hace referencia a un elemento de grupo nacional, *identifier-1* se procesa como un elemento de datos elemental de categoría nacional. Cuando *identifier-1* hace referencia a un elemento de grupo alfanumérico, *identifier-1* se trata como si fuera un elemento de datos elemental de categoría alfanumérica.

identifier-1 no debe describirse con la cláusula JUSTIFIC y no puede ser un identificador de función. *identifier-1* puede tener un subíndice o una referencia modificada.

identifier-1 no debe solaparse *identifier-2*, *identifier-3*, *página de códigos* (si es un identificador), *identifier-4*o *identifier-5*.

La salida XML generada se codifica tal como se describe en la documentación de la frase ENCODING.

El *identifier-1* debe hacer referencia a un elemento de datos de categoría nacional, o si *identifier-1* es alfanumérico de categoría, la codificación del documento debe ser Unicode UTF-8, si se cumple alguna de las siguientes afirmaciones:

- *identifier-4* o *identifier-5* hace referencia a un elemento de datos de categoría nacional.
- *literal-4*, *literal-5*o *literal-6* es de categoría nacional.
- *página de códigos* es un literal nacional o hace referencia a un elemento de datos de categoría nacional.
- El XML generado incluye datos de *identifier-2* para:
  - Cualquier elemento de datos de clase nacional o clase DBCS
  - Cualquier elemento de datos con un nombre multibyte (es decir, un elemento de datos cuyo nombre consta de multibyte caracteres)
  - Cualquier elemento de datos de clase alfanumérico que contenga multibyte caracteres

identifier-1 debe ser lo suficientemente grande para contener el documento XML generado. Normalmente, debe ser de 5 a 10 veces el tamaño de *identifier-2*, en función de la longitud del nombre de datos o nombres de datos dentro de *identifier-2*. Si *identifier-1* no es lo suficientemente grande, existe una condición de error al final de la sentencia XML GENERATE.

identifier-2

El elemento de datos elemental o de grupo que se va a convertir al formato XML.

Si *identifier-2* hace referencia a un elemento de grupo nacional, identifier-2 se procesa como un elemento de grupo. Cuando identifier-2 incluye un elemento de grupo nacional subordinado, dicho elemento subordinado se procesa como un elemento de grupo.

identifier-2 no puede ser un identificador de función ni ser una referencia modificada, pero puede ser un subíndice.

identifier-2 no debe solaparse *identifier-1* o *identifier-3*.

La entrada de descripción de datos para *identifier-2* no debe contener una cláusula RENAMES.

La sentencia XML GENERATE ignora los siguientes elementos de datos especificados por *identifier-2* :

- Elementos de datos elementales sin nombre subordinados o elementos de datos FILLER elementales
- Bytes de holgura insertados para elementos SYNCHRONIZED
- Cualquier elemento de datos subordinado a *identifier-2* que se describe con la cláusula REDEFINES o que está subordinado a dicho elemento de redefinición
- Cualquier elemento de datos subordinado a *identifier-2* que se describe con la cláusula RENAMES
- Cualquier elemento de datos de grupo cuyos elementos de datos subordinados se ignoran

Todos los elementos de datos especificados por *identifier-2* que no se ignoran de acuerdo con las reglas anteriores deben cumplir las condiciones siguientes:

- Cada elemento de datos elemental debe tener una clase alfabética, alfanumérica, numérica o nacional, o ser un elemento de datos de índice. (Es decir, no se puede describir ningún elemento de datos elemental con la frase USAGE POINTER, USAGE FUNCTION-POINTER, o USAGE PROCEDURE-POINTER .)
- Debe haber al menos un elemento de datos elemental de este tipo.
- Cada nombre de datos que no sea FILLER debe ser exclusivo dentro de cualquier elemento de datos de grupo superordenado inmediatamente.
- Los nombres de datos de multibyte , cuando se convierten a Unicode, deben ser legales como nombres en la especificación XML, versión 1.0. Para obtener detalles sobre la especificación XML, consulte *Especificación XML*.

- Los elementos de datos no deben especificar la cláusula DATE FORMAT, o la opción de compilador DATEPROC no debe estar en vigor.

Por ejemplo, considere la siguiente declaración de datos:

```
01 STRUCT.
  02 STAT PIC X(4).
  02 IN-AREA PIC X(100).
  02 OK-AREA REDEFINES IN-AREA.
    03 FLAGS PIC X.
    03 PIC X(3).
    03 COUNTER USAGE COMP-5 PIC S9(9).
    03 ASFNPTR REDEFINES COUNTER USAGE FUNCTION-POINTER.
    03 UNREFERENCED PIC X(92).
  02 NG-AREA1 REDEFINES IN-AREA.
    03 FLAGS PIC X.
    03 PIC X(3).
    03 PTR USAGE POINTER.
    03 ASNUM REDEFINES PTR USAGE COMP-5 PIC S9(9).
    03 PIC X(92).
  02 NG-AREA2 REDEFINES IN-AREA.
    03 FN-CODE PIC X.
    03 UNREFERENCED PIC X(3).
    03 QTYONHAND USAGE BINARY PIC 9(5).
    03 DESC USAGE NATIONAL PIC N(40).
    03 UNREFERENCED PIC X(12).
```

Los siguientes elementos de datos del ejemplo anterior se pueden especificar como *identifier-2*:

- STRUCT, de los cuales los elementos de datos subordinados STAT y IN-AREA se convertirían al formato XML. (OK-AREA, NG-AREA1y NG-AREA2 se ignoran porque especifican la cláusula REDEFINES.)
- OK-AREA, de los cuales se convertirían los elementos de datos subordinados FLAGS, COUNTERy UNREFERENCED . (El elemento cuya entrada de descripción de datos especifica 03 PIC X(3) se ignora porque es un elemento de datos FILLER elemental. ASFNPTR se ignora porque especifica la cláusula REDEFINES.)
- Cualquiera de los elementos de datos elementales que están subordinados a STRUCT excepto:
  - ASFNPTR o PTR (uso no permitido)
  - UNREFERENCED OF NG-AREA2 (nombres no exclusivos para elementos de datos que de otro modo serían elegibles)
  - Elementos de datos FILLER

Los siguientes elementos de datos no se pueden especificar como *identifier-2*:

- NG-AREA1, porque el elemento de datos subordinado PTR especifica USAGE POINTER pero no especifica la cláusula REDEFINES. (PTR se ignoraría si especificara la cláusula REDEFINES.)
- NG-AREA2, porque los elementos de datos elementales subordinados tienen el nombre no exclusivo UNREFERENCED.

Frase COUNT IN

Si se especifica la frase COUNT IN, *identifier-3* contiene (después de la ejecución de la sentencia XML GENERATE) el recuento de unidades de codificación de caracteres XMLgeneradas. Si *identifier-1* (el receptor) tiene la categoría nacional, el recuento está en unidades de codificación de caracteres UTF-16 . Para todas las demás codificaciones (incluyendo UTF-8), el recuento está en bytes.

identifier-3

El campo de recuento de datos. Debe ser un elemento de datos entero definido sin el símbolo P en su serie de imagen.

identifier-3 no debe solaparse *identifier-1*, *identifier-2*, *página de códigos* (si es un identificador), *identifier-4*o *identifier-5*.

Frase ENCODING

La frase ENCODING, si se especifica, determina la codificación del documento XML generado.

página de códigos

Debe ser un elemento de datos entero sin signo, un literal entero sin signo, un literal alfanumérico, un literal nacional o hacer referencia a un elemento de datos de categoría alfanumérica o nacional. Si *página de códigos* es un literal, no debe ser una constante figurativa.

Si *página de códigos* es de clase alfanumérica o nacional, debe identificar un nombre de página de códigos primario o alias que esté soportado por bibliotecas de conversión ICU (consulte *International Components for Unicode: Converter Explorer*). Si *página de códigos* es un entero, el entero debe ser un número CCSID válido.

Si *identifier-1* hace referencia a un elemento de datos de categoría nacional, *codepage* debe identificar UTF-16 en formato little-endian .

Si *identifier-1* hace referencia a un elemento de datos de categoría alfanumérica, *página de códigos* debe identificar UTF-8 o una página de códigos ASCII o EBCDIC de un solo byte:

- Si la opción de compilador CHAR (EBCDIC) no está en vigor o la entrada de descripción de datos para *identifier-1* contiene la frase NATIVE, *codepage* debe identificar UTF-8 o una página de códigos ASCII de un solo byte.
- Si CHAR (EBCDIC) está en vigor y la entrada de descripción de datos para *identifier-1* no contiene la frase NATIVE, *codepage* debe identificar una página de códigos EBCDIC de un solo byte.

Si *página de códigos* es un identificador, no debe solapar *identifier-1* o *identifier-3*.

Si se omite la frase ENCODING y *identifier-1* es de categoría nacional, la codificación del documento es Unicode UTF-16 en formato little-endian.

Si se omite la frase ENCODING y *identifier-1* es de categoría alfanumérica, entonces:

- Si CHAR (EBCDIC) no está en vigor o la entrada de descripción de datos para *identifier-1* contiene la frase NATIVE, el documento XML se codifica utilizando la página de códigos del entorno local de ejecución.
- Si la opción CHAR (EBCDIC) está en vigor y la entrada de descripción de datos para *identifier-1* no contiene la frase NATIVE, el documento XML se codifica utilizando la página de códigos de la variable de entorno EBCDIC_CODEPAGE. Si EBCDIC_CODEPAGE no está establecido, la codificación es la página de códigos EBCDIC predeterminada asociada con el entorno local de ejecución.

Frase XML-DECLARE

Si se especifica la frase XML-DECLARE, el documento XML generado empieza por una declaración XML que incluye la información de versión XML y una declaración de codificación.

Si *identifier-1* es de categoría nacional, la declaración de codificación tiene el valor UTF-16 (`encoding="UTF-16"`).

Si *identifier-1* es de categoría alfanumérica, la declaración de codificación se deriva de la frase ENCODING, si se especifica, o de la opción de compilador o la variable de entorno EBCDIC_CODEPAGE si no se especifica la frase ENCODING. Consulte la descripción de la frase ENCODING para obtener más detalles.

Para ver un ejemplo del efecto de codificar la frase XML-DECLARE, consulte *Generación de salida XML* en la publicación *COBOL for Linux en x86 Guía de programación*.

Si se omite la frase XML-DECLARE, el documento XML generado no incluye una declaración XML.

Frase ATTRIBUTES

Si se especifica la frase ATTRIBUTES, cada elemento elegible incluido en el documento XML generado se expresa como un atributo del elemento XML que corresponde al elemento de datos inmediatamente superior al elemento elegible, en lugar de como un elemento hijo del elemento XML. Para ser elegible, un elemento de datos debe ser elemental, debe tener un nombre distinto de FILLER y no debe especificar una cláusula OCCURS en su entrada de descripción de datos.

Si se especifica la frase TYPE para determinados identificadores, la frase TYPE tiene prioridad para dichos identificadores sobre la frase WITH ATTRIBUTES.

Para ver un ejemplo del efecto de la frase ATTRIBUTES, consulte *Generación de salida XML* en la publicación *COBOL for Linux en x86 Guía de programación*.

Frases NAMESPACE y NAMESPACE-PREFIX

Utilice la frase NAMESPACE para identificar un *namespace* para el documento XML generado. Si no se especifica la frase NAMESPACE, o si *identifier-4* tiene una longitud cero o contiene todos los espacios, los nombres de elemento de los documentos XML producidos por la sentencia XML GENERATE no están en ningún espacio de nombres.

Utilice la frase NAMESPACE-PREFIX para calificar el código de inicio y final de cada elemento del documento XML generado con un prefijo.

Si no se especifica la frase NAMESPACE-PREFIX, o si *identifier-5* tiene una longitud cero o contiene todos los espacios, el espacio de nombres especificado por la frase NAMESPACE especifica el espacio de nombres predeterminado para el documento. En este caso, el espacio de nombres declarado en el elemento raíz se aplica de forma predeterminada a cada nombre de elemento del documento, incluido el del elemento raíz. (Las declaraciones de espacio de nombres predeterminadas no se aplican directamente a los nombres de atributo.)

Si se especifica la frase NAMESPACE-PREFIX y *identifier-5* no tiene la longitud cero y no contiene todos los espacios, el código de inicio y final de cada elemento del documento generado se califica con el prefijo especificado. Por lo tanto, el prefijo debe ser, preferiblemente, corto. Cuando se ejecuta la sentencia XML GENERATE, el prefijo debe ser un nombre XML válido, pero sin dos puntos (`:`), tal como se define en *Espacios de nombres en XML 1.0*. El prefijo puede tener espacios finales, que se eliminan antes de su uso.

identifier-4*, *literal-4*; *identifier-5*, *literal-5

identifier-4, *literal-4*: el identificador de espacio de nombres, que debe ser un *URI (Uniform Resource Identifier)* válido tal como se define en *URI (Uniform Resource Identifier: Generic Syntax*.

identifier-5, *literal-5*: el prefijo de espacio de nombres, que sirve como alias para el identificador de espacio de nombres.

identifier-4 y *identifier-5* deben hacer referencia a elementos de datos de categoría alfanumérica o nacional.

identifier-4 y *identifier-5* no deben solaparse *identifier-1* o *identifier-3*.

literal-4 y *literal-5* deben ser de categoría alfanumérica o nacional, y no deben ser constantes figurativas.

Para obtener detalles completos sobre los espacios de nombres, consulte *Espacios de nombres en XML 1.0*.

Para ver ejemplos que muestran el uso de las frases NAMESPACE y NAMESPACE-PREFIX, consulte *Generación de salida XML* en la publicación *COBOL for Linux en x86 Guía de programación*.

Frase NAME

Le permite proporcionar nombres de elemento y atributo.

identifier-6 debe hacer referencia a *identifier-2* o a uno de sus elementos de datos subordinados. No puede ser un identificador de función y no puede ser una referencia modificada o subscripción. No debe especificar ningún elemento de datos que sea ignorado por la sentencia XML GENERATE. Para obtener más información sobre *identifier-2*, consulte la descripción de *identifier-2*. Si se especifica *identifier-6* más de una vez en la frase NAME, se utiliza la última especificación.

literal-6 debe ser un literal alfanumérico o nacional que contenga el atributo o nombre de elemento que se va a generar en el documento XML correspondiente a *identifier-6*. Debe ser un nombre local XML válido. Si *literal-6* es un literal nacional, *identifier-1* debe hacer referencia a un elemento de datos de categoría nacional o la frase de codificación debe especificar UTF-16.

Frase TYPE

Permite controlar la generación de atributos y elementos.

identifier-7 debe hacer referencia a un elemento de datos elemental que esté subordinado a *identifier-2*. No puede ser un identificador de función y no puede ser una referencia modificada o

subscripción. No debe especificar ningún elemento de datos que sea ignorado por la sentencia XML GENERATE. Para obtener más información sobre *identifier-2*, consulte la descripción de *identifier-2*. Si se especifica *identifier-7* más de una vez en la frase TYPE, se utiliza la última especificación.

- Si la sentencia XML GENERATE también incluye una frase WITH ATTRIBUTES, la frase TYPE tiene prioridad para *identifier-7*.
- Cuando se especifica ATTRIBUTE, *identifier-7* debe ser elegible para ser un atributo XML. *identifier-7* se expresa en el XML generado como un atributo del elemento XML inmediatamente superior a *identifier-7* en lugar de como un elemento hijo.
- Cuando se especifica ELEMENT, *identifier-7* se expresa en el XML generado como un elemento. El nombre de elemento XML se deriva de *identifier-7* y el contenido de carácter de elemento se deriva del contenido convertido de *identifier-7* tal como se describe en "Operación de XML GENERATE" en la página 438.
- Cuando se especifica CONTENT, *identifier-7* se expresa en el XML generado como contenido de caracteres de elemento del elemento XML que corresponde al elemento de datos inmediatamente superior a *identifier-7*. El valor del contenido del carácter de elemento se deriva del contenido convertido de *identifier-7* tal como se describe en "Operación de XML GENERATE" en la página 438. Cuando se especifica CONTENT para varios identificadores, todos correspondientes al mismo identificador de orden superior, se concatenan las varias contribuciones al contenido de caracteres del elemento.

Frase SUPPRESS

Permite identificar y suprimir incondicionalmente elementos que están subordinados a *identifier-2* y generar de forma selectiva la salida para la sentencia XML GENERATE. Si se especifica la frase SUPPRESS, *identifier-1* debe ser lo suficientemente grande para contener el documento XML generado antes de cualquier supresión.

Con la *frase-supresión-genérica*, los elementos elementales subordinados al *identifier-2* que, de otro modo, las operaciones XML GENERATE no ignoran se identifican genéricamente para una posible supresión. Pueden suprimirse elementos de clase numérica, si se especifica la palabra clave NUMERIC, o elementos que no son de clase numérica, si se especifica la palabra clave NONNUMERIC, o ambos. Si se especifica la palabra clave ATTRIBUTE, sólo se identifican los elementos que se expresarían en el documento XML generado como un atributo XML para la supresión potencial. Si se especifica la palabra clave ELEMENT, sólo se identifican los elementos que se expresarían en el documento XML generado como un elemento XML para la supresión potencial. Si se especifica la palabra clave CONTENT, sólo se identifican los elementos que se expresarían en el documento XML generado como contenido de caracteres de elemento del elemento XML correspondiente al elemento de datos que se superordena al elemento de datos CONTENT para su posible supresión.

Si se especifican varias *frase-supresión-genérica* , el efecto es acumulativo.

identifier-8 identifica explícitamente elementos para la supresión potencial. Si se especifica la frase WHEN, *identifier-8* debe hacer referencia a un elemento de datos elemental que esté subordinado a *identifier-2* y que las operaciones GENERATE XML no ignoren de otro modo. *identifier-8* no puede ser un identificador de función y no puede ser una referencia modificada o con subíndice.Si se omite la frase WHEN, *identifier-8* puede hacer referencia no solo a un elemento de datos elemental, sino también a un elemento de datos de grupo. Ese elemento de datos de grupo y todos los elementos de datos que están subordinados al elemento de grupo se suprimen. Si se especifica *identifier-8* más de una vez en la frase SUPPRESS, se utiliza la última especificación. La especificación de supresión explícita para *identifier-8* altera temporalmente la especificación de supresión implícita en cualquier *frase-supresión-genérica*, si *identifier-8* también es uno de los identificadores identificados genéricamente.

Si se especifica *identifier-8* , se le aplican las reglas siguientes:

- Si se especifica CERO, ZEROES o ZEROS en la frase WHEN, *identifier-8* no debe ser de USAGE DISPLAY-1.
- Si se especifica SPACE o SPACES en la frase WHEN, *identifier-8* debe ser USAGE DISPLAY, DISPLAY-1o NATIONAL. Si *identifier-8* es un elemento decimal con zona o nacional, debe ser un entero.

- Si se especifica LOW-VALUE, LOW-VALUES, HIGH-VALUE o HIGH-VALUES en la frase WHEN, *identifier-8* debe ser de USAGE DISPLAY o NATIONAL. Si *identifier-8* es un elemento decimal con zona o nacional, debe ser un entero.

Si se especifica la *frase-supresión-genérica* , los elementos de datos se seleccionan para la supresión potencial de acuerdo con las reglas siguientes:

- Si se especifica CERO, ZEROES o ZEROS en la frase WHEN, se seleccionan todos los elementos de datos excepto los definidos con USAGE DISPLAY-1 .
- Si se especifica SPACE o SPACES en la frase WHEN, se seleccionan los elementos de datos de USAGE DISPLAY, DISPLAY-1o NATIONAL. Para elementos decimales con zona o nacionales, sólo se seleccionan enteros.
- Si se especifica LOW-VALUE, LOW-VALUES, HIGH-VALUE o HIGH-VALUES en la frase WHEN, se seleccionan los elementos de datos de USAGE DISPLAY o NATIONAL. Para elementos decimales con zona o nacionales, sólo se seleccionan enteros.

La operación de comparación que determina si un elemento se suprimirá es una condición de relación tal como se muestra en la tabla de Comparaciones que implican constantes figurativas. Es decir, la comparación es una comparación numérica si el valor especificado es CERO, ZEROS o ZEROES, y el elemento es de clase numérica. Para todos los demás casos, la operación de comparación es una comparación alfanumérica, DBCS o nacional, en función de si el elemento es de uso DISPLAY, DISPLAY-1 o NATIONAL, respectivamente.

Cuando se especifica la frase SUPPRESS, se suprime un elemento de grupo subordinado a *identifier-2* en el documento XML generado si se suprimen todos los elementos elegibles subordinados al elemento de grupo o si, después de suprimir cualquier elemento subordinado, el XML correspondiente al elemento de grupo sería un elemento vacío sin atributos. El elemento raíz siempre se genera, incluso si se suprimen todos los elementos subordinados a *identifier-2* .

Frase ON EXCEPTION

Existe una condición de excepción cuando se produce un error durante la generación del documento XML, por ejemplo, si *identifier-1* no es lo suficientemente grande para contener el documento XML generado. En este caso, la generación de XML se detiene y el contenido del receptor, *identifier-1*, no está definido. Si se especifica la frase COUNT IN, *identifier-3* contiene el número de posiciones de caracteres que se han generado, que puede ir de 0 a la longitud de *identifier-1*.

Si se especifica la frase ON EXCEPTION, el control se transfiere a *imperative-statement-1*. Si no se especifica la frase ON EXCEPTION, la frase NOT ON EXCEPTION, si la hay, se ignora y el control se transfiere al final de la sentencia XML GENERATE. El registro especial XML-CODE contiene un código de excepción, como se detalla en *Manejo de excepciones XML GENERATE* en la publicación *COBOL for Linux en x86 Guía de programación*.

Frase NOT ON EXCEPTION

Si no se produce una condición de excepción durante la generación del documento XML, el control se pasa a *imperative-statement-2*, si se especifica, de lo contrario al final de la sentencia XML GENERATE. La frase ON EXCEPTION, si se especifica, se ignora. El registro especial XML-CODE contiene cero después de la ejecución de la sentencia XML GENERATE.

Frase END-XML

Este terminador de ámbito explícito delimita el ámbito de las sentencias XML GENERATE o XML PARSE. END-XML permite que una sentencia XML GENERATE o XML PARSE condicional (es decir, una sentencia XML GENERATE o XML PARSE que especifica la frase ON EXCEPTION o NOT ON EXCEPTION) se anide en otra sentencia condicional.

El ámbito de una sentencia XML GENERATE o XML PARSE condicional se puede terminar mediante:

- Una frase END-XML en el mismo nivel de anidamiento
- Un punto separador

END-XML también se puede utilizar con una sentencia XML GENERATE o XML PARSE que no especifique la frase ON EXCEPTION o NOT ON EXCEPTION.

Para obtener más información sobre terminadores de ámbito explícitos, consulte "Sentencias de ámbito delimitado" en la página 278.

Sentencias XML GENERATE o XML PARSE anidadas

Cuando una sentencia XML GENERATE o XML PARSE determinada aparece como *imperative-statement-1* o *imperative-statement-2*, o como parte de *imperative-statement-1* o *imperative-statement-2* de otra sentencia XML GENERATE o XML PARSE, que la sentencia XML GENERATE o XML PARSE dada es una sentencia XML GENERATE o XML PARSE *anidada* .

Las sentencias XML GENERATE o XML PARSE anidadas se consideran combinaciones XML GENERATE y END-XML coincidentes, o combinaciones XML PARSE y END-XML, que proceden de izquierda a derecha. Por lo tanto, cualquier frase END-XML que se encuentre se compara con la sentencia XML GENERATE o XML PARSE anterior más cercana que no se haya terminado implícita o explícitamente.

Operación de XML GENERATE

El contenido de cada elemento de datos elemental elegible dentro de *identifier-2* que no se ha suprimido de la generación XML de acuerdo con una frase SUPPRESS , se convierte al formato de caracteres.

Sólo se procesa la primera definición de cada área de almacenamiento. No se incluyen las redefiniciones de elementos de datos. Tampoco se incluyen los elementos de datos definidos de forma efectiva por la cláusula RENAMES.

Para obtener información sobre la conversión de formato de datos elementales, consulte "Conversión de formato de datos elementales" en la página 439 y "Recorte de datos XML generados" en la página 440.

Si se especifica la frase TYPE OF, el contenido convertido se procesa como contenido de carácter de elemento o valor de atributo, de acuerdo con las especificaciones de dicha frase. Si no se especifica la frase TYPE OF, de forma predeterminada el contenido convertido se inserta como contenido de caracteres de elemento o, si se especifica la frase WITH ATTRIBUTES y el elemento de datos es elegible para expresarse como un atributo, como el valor del atributo, en el documento XML generado.

Los nombres de elemento XML y los nombres de atributo se obtienen de la frase NAME si se especifica; de lo contrario, de forma predeterminada se derivan de los nombres de datos dentro de *identifier-2* tal como se describe en "Nombre de elemento XML y formación de nombre de atributo" en la página 440. Los nombres de los elementos de grupo que contienen los elementos elementales seleccionados se conservan como elementos padre. Si se especifica la frase NAMESPACE-PREFIX, se utiliza el valor de prefijo, menos los espacios finales, para calificar el código de inicio y final de cada elemento.

No se inserta ningún espacio en blanco adicional (líneas nuevas, sangría, etc.) para que el XML generado sea más legible. Se genera una declaración XML si se especifica la frase XML-DECLARE.

Si el área de recepción especificada por *identifier-1* no es lo suficientemente grande como para contener el documento XML resultante, existe una condición de error. Consulte la descripción de la frase ON EXCEPTION anterior para obtener más detalles.

Si *identifier-1* es más largo que el documento XML generado, sólo se cambia la parte del *identifier-1* en la que se genera XML. El resto de *identifier-1* contiene los datos que estaban presentes antes de esta ejecución de la sentencia XML GENERATE. Para evitar hacer referencia a esos datos, inicialice *identifier-1* en espacios antes de la sentencia XML GENERATE o especifique la frase COUNT IN.

Si se especifica la frase COUNT IN, *identifier-3* contiene (después de la ejecución de la sentencia XML GENERATE) el número total de posiciones de caracteres (UTF-16 unidades de codificación o bytes) que se han generado. Puede utilizar *identifier-3* como campo de longitud de modificación de referencia para hacer referencia a la parte de *identifier-1* que contiene el documento XML generado.

Después de la ejecución de la sentencia XML GENERATE, el registro especial XML-CODE contiene cero, lo que indica una finalización satisfactoria, o un código de excepción distinto de cero. Para obtener más detalles, consulte *Manejo de excepciones XML GENERATE* en la publicación *COBOL for Linux en x86 Guía de programación*.

La sentencia XML PARSE también utiliza el registro especial XML-CODE. Por lo tanto, si codifica una sentencia XML GENERATE en el procedimiento de proceso de una sentencia XML PARSE, guarde el valor de XML-CODE antes de que se ejecute la sentencia XML GENERATE y restaure el valor guardado después de que termine la sentencia XML GENERATE.

No se genera una marca de orden de bytes para documentos XML que tengan codificación Unicode.

Conversión de formato de datos elementales

Los elementos de datos elementales dentro de *identifier-2* se convierten en una secuencia de varios pasos, algunos de ellos son opcionales tal como se describe a continuación.

Conversión a formato de caracteres:

Los elementos de datos elementales se convierten al formato de caracteres en función del tipo de elemento de datos:

- Los elementos de datos de categoría alfabética, alfanumérica, alfanumérica editada, DBCS, coma flotante externa, nacional, nacional editada y numérica editada no se convierten.
- Los elementos de datos numéricos de punto fijo que no sean COMPUTATIONAL-5 (COMP-5) elementos de datos binarios o elementos de datos binarios compilados con la opción de compilador TRUNC (BIN) se convierten como si se hubieran movido a un elemento editado numérica-que tiene:
  - Tantas posiciones de enteros como tenga el elemento numérico, pero con al menos una posición de entero
  - Una coma decimal explícita, si el elemento numérico tiene al menos una posición decimal
  - El mismo número de posiciones decimales que tiene el elemento numérico
  - Un símbolo de imagen '-' inicial si el elemento de datos está firmado (tiene una S en su cláusula PICTURE)
- Los elementos de datos binarios COMPUTATIONAL-5 (COMP-5) o los elementos de datos binarios compilados con la opción de compilador TRUNC (BIN) se convierten del mismo modo que los otros elementos numéricos de punto fijo, excepto el número de posiciones enteras. El número de posiciones enteras se calcula en función del número de símbolos '9' en la serie de caracteres de imagen como se indica a continuación:
  - 5 menos el número de posiciones decimales, si el elemento de datos tiene de 1 a 4 '9' símbolos de imagen
  - 10 menos el número de posiciones decimales, si el elemento de datos tiene de 5 a 9 '9' símbolos de imagen
  - 20 menos el número de posiciones decimales, si el elemento de datos tiene de 10 a 18 '9' símbolos de imagen
- Los elementos de datos de coma flotante internos se convierten como si se hubieran movido a un elemento de datos como se indica a continuación:
  - Para COMP-1: un elemento de datos de coma flotante externo con PICTURE -9.9 (8)E+99
  - Para COMP-2: un elemento de datos de coma flotante externo con PICTURE -9.9 (17)E+99 (no permitido debido al número de posiciones de dígitos)
- Los elementos de datos de índice se convierten como si se hubieran declarado USAGE COMP-5 PICTURE S9(9).

Recorte:

Después de cualquier conversión al formato de caracteres, se eliminan los espacios iniciales y finales y los ceros iniciales, tal como se describe en “Recorte de datos XML generados” en la página 440.

Conversión a la codificación del documento:

Todos los valores se convierten según sea necesario a la codificación del documento, tal como se indica a continuación. Si *identifier-1* es:

- La categoría alfanumérica, y la frase NATIVE se ha especificado en la descripción de datos o CHAR (NATIVE) estaba en vigor, los valores se convierten a UTF-8 o a la página de códigos ASCII elegida.
- Categoría alfanumérica, y la frase NATIVE no se ha especificado en la descripción de datos, y CHAR (EBCDIC) estaba en vigor, los valores se convierten a la página de códigos EBCDIC elegida.
- Categoría nacional: Los valores no nacionales se convierten al formato nacional.

Conversión de caracteres especiales a referencias XML:

Las instancias restantes de los cinco caracteres & (ampersand), ' (apóstrofo), > (signo mayor que), < (signo menor que) y " (comillas) se convierten en las referencias XML equivalentes '&', ''', '>', '<' y '"', respectivamente.

Sustitución de caracteres Unicode fuera de rango:

Cualquier carácter Unicode restante que tenga un valor escalar Unicode mayor que x 'FFFF' se sustituye por una referencia de carácter XML. Por ejemplo, si el documento contiene un carácter con el valor escalar Unicode x '10813', en UTF-16LE, dicho valor se representa mediante el par suplente (NX'02D8', NX'13DC'), que se sustituye por la referencia '𐠓'. Para una codificación de documento de UTF-8, la secuencia de bytes que es equivalente a la referencia de caracteres '𐠓'is X'F090A093'.

Recorte de datos XML generados

El recorte se realiza en los valores de datos después de su conversión al formato de caracteres.

Para obtener más información sobre la conversión, consulte “Conversión de formato de datos elementales” en la página 439.

Para los valores convertidos a partir de valores numéricos con signo, el espacio inicial se elimina si el valor es positivo.

Para los valores convertidos a partir de elementos numéricos, se eliminan los ceros iniciales (después de cualquier signo menos inicial) hasta, pero sin incluir, el dígito inmediatamente antes de la coma decimal real o implícita. Se conservan los ceros finales después de una coma decimal. Por ejemplo:

- -012.340 pasa a ser -12.340.
- 0000.45 se convierte en 0.45.
- 0013 se convierte en 13.
- 0000 se convierte en 0.

Los valores de caracteres de elementos de datos de clase alfabéticos, alfanuméricos, DBCS y nacionales tienen espacios finales o iniciales eliminados, en función de si los elementos de datos correspondientes tienen justificación izquierda (valor predeterminado) o derecha, respectivamente. Es decir, los espacios finales se eliminan de los valores cuyos elementos de datos correspondientes no especifican la cláusula JUSTIFIC. Los espacios iniciales se eliminan de los valores cuyos elementos de datos especifican la cláusula JUSTIFIC. Si un valor de carácter consta únicamente de espacios, un espacio permanece como el valor después de finalizar el recorte.

Nombre de elemento XML y formación de nombre de atributo

En los documentos XML que se generan a partir de *identifier-2*, los nombres de elemento XML y nombres de atributo son obtenido de la frase NAME si se especifica; de lo contrario, se derivan de los nombres del elemento de datos especificado por *identifier-2* y de cualquier nombre de datos elegible que esté subordinado a *identifier-2*.

La formación del nombre de elemento XML y el nombre de atributo es la siguiente:

- Se conserva la ortografía exacta en mayúsculas y minúsculas de los nombres de datos de la entrada de descripción de datos. No se utilizan las ortografías de las referencias a elementos de datos (por ejemplo, en una cláusula OCCURS DEPENDING ON).
- Los nombres de datos que empiezan por un dígito llevan como prefijo un subrayado. Por ejemplo, el nombre de datos '3D' se convierte en etiqueta XML o atributo nombre '_3D'.

- Los nombres de datos que empiezan por los caracteres 'xml', en cualquier combinación de mayúsculas y minúsculas, llevan como prefijo un subrayado. Por ejemplo, el nombre de datos 'Xml' se convierte en código XML o nombre de atributo '_Xml'.

Los nombres de datos de Multibyte , cuando se convierten a Unicode, deben ser legales como nombres en la especificación XML, versión 1.0. Para obtener detalles sobre la especificación XML, consulte *Especificación XML*.

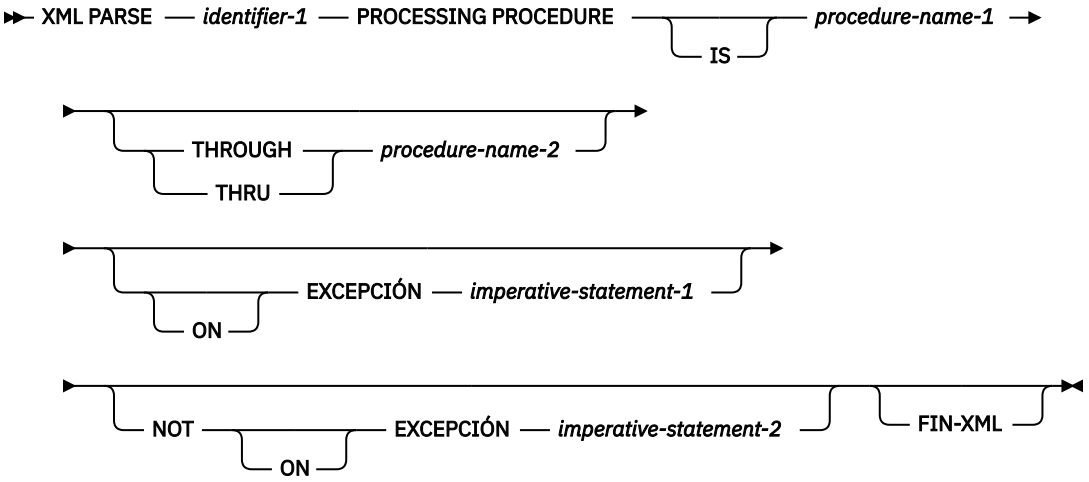
sentencia XML PARSE

La sentencia XML PARSE es la interfaz de lenguaje COBOL para el analizador XML de alta velocidad que forma parte del tiempo de ejecución COBOL.

La sentencia XML PARSE analiza un documento XML en sus partes individuales y pasa cada pieza, de una en una, a un procedimiento de proceso escrito por el usuario.

Las sentencias XML PARSE no deben especificarse en procedimientos declarativos.

Formato



identifier-1

identifier-1 debe ser un elemento de datos elemental de categoría nacional, un grupo nacional, un elemento de datos elemental de categoría alfanumérico o un elemento de grupo alfanumérico. *identifier-1* no puede ser un identificador de función. *identifier-1* contiene la corriente de caracteres del documento XML.

Si *identifier-1* es un elemento de grupo nacional, *identifier-1* se procesa como un elemento de datos elemental de categoría nacional.

Si *identifier-1* es de categoría nacional, su contenido debe codificarse utilizando Unicode UTF-16LE (CCSID 1200). *identifier-1* no debe contener ninguna entidad de caracteres que se represente utilizando varias unidades de codificación. Utilice una referencia de carácter para representar dichos caracteres, por ejemplo:

- "" o
- "𐠓"

La letra x debe estar en minúsculas.

CHAR (NATIVE) y alfanumérico identifier-1

Si *identifier-1* es alfanumérico y la opción de compilador CHAR (EBCDIC) no está en vigor, el contenido de *identifier-1* debe codificarse utilizando UTF-8 Unicode o una página de códigos ASCII de un solo byte soportada por las bibliotecas de conversión ICU (consulte *Componentes internacionales para Unicode: Converter Explorer*).

Para asegurarse de que un documento de un elemento de datos de este tipo se analice en UTF-8 en lugar de en ASCII:

- La página de códigos para el entorno local de tiempo de ejecución debe ser un entorno local UTF-8 , o
- El documento debe incluir una declaración de codificación XML que especifique UTF-8, o
- El documento debe empezar con una marca de orden de bytes UTF-8 .

Los documentos UTF-8 no deben contener ningún carácter con un valor escalar Unicode mayor que x 'FFFF'. Utilice una referencia de caracteres para dichos caracteres.

Si el documento XML de un elemento de datos de este tipo no especifica una declaración de codificación y no empieza con una marca de orden de bytes UTF-8 , se analiza con la página de códigos indicada por el entorno local de tiempo de ejecución actual.

CHAR (EBCDIC) y identifier-1

Si *identifier-1* es alfanumérico y la opción de compilador CHAR (EBCDIC) está en vigor, El contenido del *identifier-1* debe codificarse utilizando una página de códigos EBCDIC de un solo byte soportada por las bibliotecas de conversión ICU (consulte *International Components for Unicode: Converter Explorer*). Si *identifier-1* es un elemento elemental, la palabra clave NATIVE no debe especificarse en su entrada de descripción de datos.

Si el documento XML de un elemento de datos de este tipo no especifica una declaración de codificación, el documento XML se analiza con la página de códigos especificada por la variable de entorno EBCDIC_CODEPAGE, o si la variable de entorno EBCDIC_CODEPAGE no está establecida, la página de códigos EBCDIC predeterminada seleccionada para el entorno local de ejecución actual, tal como se describe en *Entornos locales y páginas de códigos soportadas* en *COBOL for Linux en x86 Guía de programación*.

Establecimiento y utilización de entornos locales y páginas de códigos de tiempo de ejecución

Para obtener más información sobre cómo establecer y utilizar entornos locales y páginas de códigos de tiempo de ejecución, consulte *Entornos locales y páginas de códigos soportadas* en la publicación *COBOL for Linux en x86 Guía de programación*. Las páginas de códigos ASCII y EBCDIC de un solo byte son aquellas para las que la columna etiquetada Grupo de idiomas (la columna situada más a la derecha) de la tabla Locales y páginas de códigos soportadas no especifica "Idiomas ideográficos".

Frase PROCESSING PROCEDURE

Especifica el nombre de un procedimiento para manejar los diversos sucesos que genera el analizador XML.

procedure-name-1, procedure-name-2
Debe nombrar una sección o párrafo en PROCEDURE DIVISION. *Procedure-name-1* y *procedure-name-2* no deben nombrar un nombre de procedimiento en una sección declarativa.

procedure-name-1
Especifica la primera (o única) sección o párrafo en el procedimiento de proceso.

procedure-name-2
Especifica la última sección o párrafo del procedimiento de proceso.

Para cada suceso XML, el analizador transfiere el control a la primera sentencia del procedimiento denominado *procedure-name-1*. El control siempre se devuelve desde el procedimiento de proceso al analizador XML. El punto desde el que se devuelve el control se determina de la siguiente manera:

- Si *procedure-name-1* es un nombre de párrafo y no se especifica *procedure-name-2* , la devolución se realiza después de la ejecución de la última sentencia del párrafo *procedure-name-1* .
- Si *procedure-name-1* es un nombre de sección y no se especifica *procedure-name-2* , la devolución se realiza después de la ejecución de la última sentencia del último párrafo en la sección *procedure-name-1* .
- Si se especifica *procedure-name-2* y es un nombre de párrafo, la devolución se realiza después de la ejecución de la última sentencia del párrafo *procedure-name-2* .

- Si se especifica *procedure-name-2* y es un nombre de sección, la devolución se realiza después de la ejecución de la última sentencia del último párrafo en la sección *procedure-name-2* .

La única relación necesaria entre *procedure-name-1* y *procedure-name-2* es que definen una secuencia consecutiva de operaciones a ejecutar, empezando por el procedimiento denominado por *procedure-name-1* y finalizando con la ejecución del procedimiento denominado por *procedure-name-2*.

Si hay dos o más vías de acceso lógicas al punto de retorno, *procedure-name-2* puede nombrar un párrafo que conste de sólo una sentencia EXIT; todas las vías de acceso al punto de retorno deben llevar a este párrafo.

El procedimiento de proceso consta de todas las sentencias en las que se manejan los sucesos XML. El rango del procedimiento de proceso incluye todas las sentencias ejecutadas por las sentencias CALL, EXIT, GO TO, GOBACK, MERGE, PERFORM y SORT que están en el rango del procedimiento de proceso, así como todas las sentencias de los procedimientos declarativos que se ejecutan como resultado de la ejecución de sentencias en el rango del procedimiento de proceso.

El rango del procedimiento de proceso no debe provocar la ejecución de ninguna sentencia GOBACK o EXIT PROGRAM, excepto para devolver el control de un método o programa al que se ha pasado el control mediante una sentencia CALL, respectivamente, que se ejecuta en el rango del procedimiento de proceso.

El rango del procedimiento de proceso no debe provocar la ejecución de una sentencia XML PARSE, a menos que la sentencia XML PARSE se ejecute en un método o programa externo al que se haya pasado el control mediante una sentencia CALL que se ejecuta en el rango del procedimiento de proceso.

Un programa que se ejecuta en varias hebras puede ejecutar la misma sentencia XML o distintas sentencias XML simultáneamente.

El procedimiento de proceso puede terminar la unidad de ejecución con una sentencia STOP RUN.

Para obtener más detalles sobre el procedimiento de proceso, consulte "Flujo de control" en la página 444.

EN EXCEPCIÓN

La frase ON EXCEPTION especifica sentencias imperativas que se ejecutan cuando la sentencia XML PARSE emite una condición de excepción.

Existe una condición de excepción cuando el analizador XML detecta un error al procesar el documento XML. El analizador primero señala una excepción XML pasando el control al procedimiento de proceso con el registro especial XML-EVENT que contiene 'EXCEPTION'. El analizador también proporciona un código de error numérico en el registro especial XML-CODE, tal como se detalla en en *Manejo de excepciones XML PARSE* en *COBOL for Linux en x86 Guía de programación*.

También existe una condición de excepción si el procedimiento de proceso establece XML-CODE en -1 antes de volver al analizador para cualquier suceso XML normal. En este caso, el analizador no señala un suceso EXCEPTION XML y el análisis termina.

Si se especifica la frase ON EXCEPTION, el analizador transfiere el control a *imperative-statement-1*. Si no se especifica la frase ON EXCEPTION, la frase NOT ON EXCEPTION, si la hay, se ignora y el control se transfiere al final de la sentencia XML PARSE.

El registro especial XML-CODE contiene el código de error numérico para la excepción XML o -1 después de la ejecución de la sentencia XML PARSE.

Si el procedimiento de proceso maneja el suceso de excepción XML y establece XML-CODE en cero antes de devolver el control al analizador, la condición de excepción ya no existe. Si no se producen otras excepciones no manejadas antes de la terminación del analizador, el control se transfiere a *imperative-statement-2* de la frase NOT ON EXCEPTION, si se especifica.

NO EN EXCEPCIÓN

La frase NOT ON EXCEPTION especifica sentencias imperativas que se ejecutan cuando no existe ninguna condición de excepción al finalizar el proceso XML PARSE.

Si no existe una condición de excepción al finalizar el proceso XML PARSE, el control se transfiere a *imperative-statement-2* de la frase NOT ON EXCEPTION, si se especifica. Si no se especifica la frase NOT ON EXCEPTION, el control se transfiere al final de la sentencia XML PARSE. La frase ON EXCEPTION, si se especifica, se ignora.

El registro especial XML-CODE contiene cero después de la ejecución de la sentencia XML PARSE.

Frase END-XML

Este terminador de ámbito explícito delimita el ámbito de las sentencias XML GENERATE o XML PARSE. END-XML permite que una sentencia XML GENERATE o XML PARSE condicional (es decir, una sentencia XML GENERATE o XML PARSE que especifica la frase ON EXCEPTION o NOT ON EXCEPTION) se anide en otra sentencia condicional.

El ámbito de una sentencia XML GENERATE o XML PARSE condicional se puede terminar mediante:

- Una frase END-XML en el mismo nivel de anidamiento
- Un punto separador

END-XML también se puede utilizar con una sentencia XML GENERATE o XML PARSE que no especifique la frase ON EXCEPTION o NOT ON EXCEPTION.

Para obtener más información sobre terminadores de ámbito explícitos, consulte "Sentencias de ámbito delimitado" en la página 278.

Sentencias XML GENERATE o XML PARSE anidadas

Cuando una sentencia XML GENERATE o XML PARSE determinada aparece como *imperative-statement-1* o *imperative-statement-2*, o como parte de *imperative-statement-1* o *imperative-statement-2* de otra sentencia XML GENERATE o XML PARSE, que la sentencia XML GENERATE o XML PARSE dada es una sentencia XML GENERATE o XML PARSE *anidada* .

Las sentencias XML GENERATE o XML PARSE anidadas se consideran coincidentes XML GENERATE y END-XML, o combinaciones XML PARSE y END-XML que proceden de izquierda a derecha. Por lo tanto, cualquier frase END-XML que se encuentre se compara con la sentencia XML GENERATE o XML PARSE anterior más cercana que no se haya terminado implícita o explícitamente.

Flujo de control

Cuando el analizador XML recibe el control de una sentencia XML PARSE, el analizador analiza el documento XML y transfiere el control en puntos específicos del proceso.

Los puntos son:

- El inicio del proceso de análisis
- Cuando se encuentra un fragmento de documento
- Cuando el analizador detecta un error al analizar el documento XML
- El final del proceso del documento XML

El control vuelve al analizador XML cuando se alcanza el final del procedimiento de proceso.

El intercambio de control entre el analizador y el procedimiento de proceso continúa hasta que:

- Se ha analizado todo el documento XML, finalizando con el suceso END-OF-DOCUMENT.
- El procedimiento de proceso termina el análisis deliberadamente estableciendo XML-CODE en -1 antes de volver al analizador.
- el analizador detecta una excepción (que no sea un conflicto de codificación) y el procedimiento de proceso no restablece el registro especial XML-CODE a cero antes de volver al analizador.
- el analizador detecta una excepción de conflicto de codificación y el procedimiento de proceso no restablece el registro especial XML-CODE a cero o al CCSID de la codificación del documento.

En cada caso, el procedimiento de proceso devuelve el control al analizador. A continuación, el analizador termina y devuelve el control a la sentencia XML PARSE con el registro especial XML-CODE que contiene

el valor más reciente establecido por el analizador o -1 (que puede haber sido establecido por el analizador o por el procedimiento de proceso).

Para cada suceso XML pasado al procedimiento de proceso, los registros especiales XML-CODE y XML-EVENT contienen información sobre el suceso concreto. El registro especial XML-EVENT se establece en el nombre de suceso, como por ejemplo 'START-OF-DOCUMENT'. Para la mayoría de los sucesos, el registro especial XML-TEXT o XML-NTEXT contiene texto de documento. Consulte “XML-EVENT” en la página 27 para obtener más detalles.

El contenido del registro especial XML-CODE se define durante y después de la ejecución de una sentencia XML PARSE. El contenido de todos los demás registros especiales XML no están definidos fuera del rango del procedimiento de proceso.

Para sucesos XML normales, el registro especial XML-CODE contiene cero cuando el procedimiento de proceso recibe el control. Para sucesos de excepción XML, XML-CODE contiene un código de excepción XML tal como se describe en *Excepciones XML PARSE que permiten la continuación* y *Excepciones XML PARSE que no permiten la continuación* en *COBOL for Linux en x86 Guía de programación*.

Para obtener más información sobre los registros especiales XML, consulte:

- “CÓDIGO XML” en la página 26
- “XML-EVENT” en la página 27
- “XML-NTEXT” en la página 29
- “XML-TEXTO” en la página 30

Para obtener una introducción a los registros especiales, consulte “Registros especiales” en la página 17

Para obtener más información sobre el suceso EXCEPTION y el proceso de excepción, consulte *Manejo de excepciones XML PARSE* en la publicación *COBOL for Linux en x86 Guía de programación*.

Parte 7. Funciones intrínsecas

Capítulo 20. Funciones intrínsecas

Una *función intrínseca* es una función que realiza una operación matemática, de caracteres o lógica. Puede utilizar funciones intrínsecas para hacer referencia a un elemento de datos cuyo valor se deriva automáticamente durante la ejecución.

Los problemas de proceso de datos a menudo requieren el uso de valores a los que no se puede acceder directamente en el almacenamiento de datos asociado con el programa objeto, pero en su lugar deben derivarse mediante la realización de operaciones en otros datos. Una *función intrínseca* es una función que realiza una operación matemática, de caracteres o lógica y, por lo tanto, le permite hacer referencia a un elemento de datos cuyo valor se deriva automáticamente durante la ejecución.

Las funciones intrínsecas pueden ser agrupadas en seis categorías, basadas en el tipo de servicio realizado:

- Matemático
- Estadísticas
- Fecha/hora
- Financiero
- Manejo de caracteres
- General

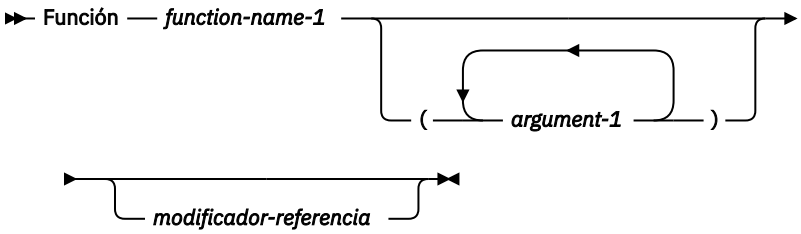
Puede hacer referencia a una función especificando su nombre, junto con los argumentos necesarios, en una sentencia PROCEDURE DIVISION.

Las funciones son elementos de datos elementales y devuelven caracteres alfanuméricos, caracteres nacionales, numéricoso valores enteros, enteros, booleanos o valores de fecha-hora. Las funciones no pueden servir como operandos de recepción.

Especificación de una función

En este tema se describe el formato general de un identificador de función.

Formato: Identificador de función



function-name-1
function-name-1 debe ser uno de los nombres de función intrínseca.

argument-1
argument-1 debe ser un identificador, un literal (que no sea una constante figurativa) o una expresión aritmética que satisfaga los requisitos de argumento para la función especificada.

argument-1 no puede ser un campo de fecha con ventana, excepto en la función intrínseca UNDATE.

modificador-referencia
Sólo se puede especificar para funciones de tipo alfanumérico o nacional.

Se puede especificar un identificador de función siempre que se permita un elemento de datos del tipo de la función. El argumento de una función puede ser cualquier función o una expresión que contenga

una función, incluida otra evaluación de la misma función, cuyo resultado cumpla los requisitos del argumento.

Dentro de una sentencia PROCEDURE DIVISION, cada identificador-función se evalúa al mismo tiempo que se evaluaría cualquier modificación de referencia o subscripción asociada con un identificador en esa misma posición.

Definición y evaluación de funciones

La clase y las características de una función, y el número y los tipos de argumentos que requiere, están determinados por su definición de función.

Estas características incluyen:

- Para las funciones de tipo alfanumérico y nacional, el tamaño del valor devuelto
- Para funciones de tipo numérico y entero, el signo del valor devuelto y si la función es entera
- El valor real devuelto por la función
- La longitud del valor devuelto para las funciones de tiempo de datos

Para algunas funciones, la clase y las características están determinadas por los argumentos de la función.

La evaluación de cualquier función intrínseca no se ve afectada por el contexto en el que aparece; en otras palabras, la evaluación de la función no se ve afectada por operaciones u operandos fuera de la función. Sin embargo, la evaluación de una función puede verse afectada por los atributos de sus argumentos.

Dentro de una sentencia PROCEDURE DIVISION, cada identificador-función se evalúa al mismo tiempo que se evaluaría cualquier modificación de referencia o subscripción asociada con un identificador en esa misma posición.

Tipos de funciones

El tema introduce tipos de funciones en COBOL.

COBOL tiene los siguientes tipos de funciones:

- Alfanumérico
- Booleano
- Fecha-hora
- DBCS
- Nacional
- Numérico
- Entero

Las funciones *alfanuméricas* son de clase y categoría alfanuméricas. El valor devuelto tiene un uso implícito de DISPLAY sin la frase NATIVE. El número de posiciones de caracteres en el valor devuelto viene determinado por la definición de la función.

Las funciones *booleanas* son de clase y categoría booleanas. El valor devuelto tiene un uso implícito de DISPLAY y es un valor booleano true (B "1") o un valor booleano false (B "0").

Las funciones *Fecha-hora* son de la clase fecha-hora y de la categoría fecha, hora o indicación de fecha y hora. El valor devuelto tiene un uso implícito de DISPLAY. El número de posiciones de caracteres en el valor devuelto viene determinado por la definición de la función.

Las funciones *DBCS* son de la clase y categoría DBCS. El valor devuelto tiene un uso implícito de DISPLAY-1, y el número de posiciones de caracteres en el valor devuelto lo determina la definición de función.

Las funciones *nacionales* son de clase y categoría nacionales. El valor devuelto tiene un uso implícito de NATIONAL y se representa en caracteres nacionales (UTF-16). El número de posiciones de caracteres en el valor devuelto viene determinado por la definición de la función.

Las funciones *numéricas* son de clase y de categoría numéricas. El valor devuelto siempre se considera que tiene un signo operativo y es un resultado intermedio numérico. Para obtener más información, consulte *Utilización de funciones intrínsecas numéricas* en la publicación *COBOL for Linux en x86 Guía de programación*.

Las funciones *Entero* son de clase y categoría numéricas. El valor devuelto siempre se considera que tiene un signo operativo y es un resultado intermedio entero. El número de posiciones de dígito en el valor devuelto está determinado por la definición de función. Para obtener más información, consulte *Utilización de funciones intrínsecas numéricas* en la publicación *COBOL for Linux en x86 Guía de programación*.

Reglas de uso

El tema describe las reglas de uso de distintos tipos de funciones.

Funciones alfanuméricas

Una función alfanumérica se puede especificar en cualquier lugar de los formatos generales en los que se permite un elemento de datos de clase y categoría alfanuméricos y en los que las reglas asociadas con los formatos generales no prohíben específicamente la referencia a las funciones, excepto como se indica a continuación.

Se puede utilizar una función alfanumérica como argumento para cualquier función que permita un argumento alfanumérico.

Se permite la modificación de referencia de una función alfanumérica. Si se especifica una modificación de referencia para una función, la evaluación de la modificación de referencia tiene lugar inmediatamente después de la evaluación de la función; es decir, el valor devuelto de la función se modifica por referencia.

No se puede utilizar una función alfanumérica:

- Como operando receptor de cualquier sentencia
- Cuando las normas asociadas a los formatos generales exijan que el elemento de datos al que se hace referencia tenga características particulares (tales como clase y categoría, uso, tamaño y valores permisibles) y la evaluación de la función de acuerdo con su definición y los argumentos particulares especificados no tendrían esas características.

Funciones booleanas

Se puede especificar una función booleana siempre que se permita un identificador booleano en los formatos generales, y siempre que las reglas asociadas con los formatos generales no prohíban específicamente la referencia a funciones. Sin embargo, no se puede especificar:

- Como operando receptor de cualquier sentencia
- Cuando las normas asociadas a los formatos generales exijan que el elemento de datos al que se hace referencia tenga características particulares (tales como clase y categoría, uso, tamaño y valores permisibles), y la evaluación de la función de acuerdo con su definición y los argumentos particulares especificados no tendrían estas características.

Se permite una función booleana como parte de una condición de relación. Si se especifica una función booleana en una condición de relación, la evaluación de la condición de relación tiene lugar inmediatamente después de la evaluación de la función.

Se puede hacer referencia a una función booleana como argumento para una función que permite un argumento booleano.

Funciones de fecha-hora

Se puede especificar una función de fecha y hora siempre que se permita un identificador de fecha y hora en los formatos generales, y siempre que las normas asociadas a los formatos generales no prohíban específicamente la referencia a las funciones. Sin embargo, no se puede especificar:

- Como operando receptor de cualquier sentencia
- Cuando las normas asociadas a los formatos generales exijan que el elemento de datos al que se hace referencia tenga características particulares (tales como clase y categoría, uso, tamaño y valores permisibles), y la evaluación de la función de acuerdo con su definición y los argumentos particulares especificados no tendrían estas características.

Se permite una función de fecha y hora como parte de una condición de relación. Si se especifica una función de fecha y hora en una condición de relación, la evaluación de la condición de relación tiene lugar inmediatamente después de la evaluación de la función.

Se puede hacer referencia a una función de fecha-hora como argumento para una función que permite un argumento de fecha-hora.

Funciones nacionales

Una función nacional se puede especificar en cualquier lugar de los formatos generales en los que se permite un elemento de datos de clase y categoría nacional y en los que las reglas asociadas con los formatos generales no prohíben específicamente la referencia a funciones, excepto como se indica a continuación.

Una función nacional puede utilizarse como argumento para cualquier función que permita un argumento nacional.

Se permite la modificación de referencia de una función nacional. Si se especifica una modificación de referencia para una función, la evaluación de la modificación de referencia tiene lugar inmediatamente después de la evaluación de la función; es decir, el valor devuelto de la función se modifica por referencia.

No se puede utilizar una función nacional:

- Como operando receptor de cualquier sentencia
- Cuando las normas asociadas a los formatos generales exijan que el elemento de datos al que se hace referencia tenga características particulares (tales como clase y categoría, uso, tamaño y valores permisibles) y la evaluación de la función de acuerdo con su definición y los argumentos particulares especificados no tendrían esas características.

Funciones numéricas

Sólo se puede utilizar una función numérica cuando se puede especificar una expresión aritmética.

Se puede hacer referencia a una función numérica como un argumento para una función que permite un argumento numérico.

No se puede utilizar una función numérica cuando se necesita un operando entero, incluso si la referencia en particular produciría un valor entero. Las funciones INTEGER o INTEGER-PART se pueden utilizar para forzar que el tipo de un argumento numérico sea un entero.

Funciones enteras

Sólo se puede utilizar una función entera cuando se puede especificar una expresión aritmética.

Se puede hacer referencia a una función entera como un argumento para una función que permite un argumento numérico.

Notas de uso:

- No se puede utilizar un identificador de función en la frase BY REFERENCE de una sentencia CALL (es decir, *identifier-2* de la sentencia CALL no debe ser un identificador de función).
- La sentencia COPY acepta identificadores de función de todos los tipos en la frase SUSTITUIR.
- La sentencia MOVE acepta identificadores de función alfanuméricos, numéricos, enteros y nacionales en el campo de envío (*identifier-1*).

argumentos

El valor devuelto por algunas funciones viene determinado por los argumentos especificados en el identificador de función cuando se evalúan las funciones. Algunas funciones no requieren argumentos; otras requieren un número fijo de argumentos, y todavía otras aceptan un número variable de argumentos.

Un argumento debe ser uno de los elementos siguientes:

- Un identificador de elemento de datos
- Una expresión aritmética
- Un identificador de función
- Un literal que no sea una constante figurativa
- Un registro especial
- Un nombre mnemotécnico
- Una palabra clave
- Un nombre-tipo

Consulte "Definiciones de función" en la página 456 para obtener las especificaciones de argumentos específicos de función.

Los tipos de argumentos son:

Alfabético
Elemento de datos elemental de la clase alfabética o literal alfanumérico que contiene sólo caracteres alfabéticos. El contenido del argumento se utiliza para determinar el valor de la función. La longitud del argumento se puede utilizar para determinar el valor de la función.

Alfanumérico
Elemento de datos de la clase alfabética o alfanumérica o un literal alfanumérico. El contenido del argumento se utiliza para determinar el valor de la función. La longitud del argumento se puede utilizar para determinar el valor de la función.

Booleano
Un elemento de datos de clase booleano, o un literal booleano.

Fecha-hora
Elemento de datos de la clase fecha-hora. El contenido del argumento se utiliza para determinar el valor de la función. La longitud del argumento se puede utilizar para determinar el valor de la función.

DBCS
Elemento de datos elemental de clase DBCS o literal DBCS. El contenido del argumento se utiliza para determinar el valor de la función. La longitud del argumento se puede utilizar para determinar el valor de la función. (Un literal o elemento de datos DBCS puede utilizarse como argumento sólo para la función NATIONAL-OF.)

Entero
Expresión aritmética que siempre da como resultado un valor entero. El valor de la expresión, incluido su signo, se utiliza para determinar el valor de la función.

Palabra clave
Debe especificarse una palabra clave de acuerdo con la definición de función.

Mnemotécnico-Nombre
Se especificará un nombre nemotécnico definido en el párrafo SPECIAL-NAMES. La característica asociada con el nombre mnemotécnico se puede utilizar para determinar el valor de la función.

Nacional
Un elemento de datos de clase nacional (categoría nacional, nacional editada o numérica editada). El contenido del argumento se utiliza para determinar el valor de la función. La longitud del argumento se puede utilizar para determinar el valor de la función.

Numérico
Una expresión aritmética. La expresión puede incluir literales numéricos y elementos de datos de categorías numéricas, de coma flotante interna y de coma flotante externa. Los elementos de datos numéricos pueden tener cualquier uso permitido para la categoría del elemento de datos (incluido NATIONAL). El valor de la expresión, incluido su signo, se utiliza para determinar el valor de la función.

Registro especial
Debe especificarse un registro especial de acuerdo con la definición de la función. La informacion asociada con el registro especial puede ser utilizada para determinar el valor de la funcion.

Declaración de tipo
Se especificará un nombre de tipo. El tamaño asociado con la declaración de tipo puede utilizarse para determinar el valor de la función.

Algunas funciones ponen restricciones en sus argumentos, como el rango aceptable de valores. Si los valores asignados como argumentos para una función no cumplen con las restricciones especificadas, el valor devuelto no está definido.

Si se utiliza una función anidada como argumento, la evaluación de sus argumentos no se ve afectada por los argumentos de la función externa.

Sólo los argumentos del mismo nivel de función interactúan entre sí. Esta interacción se produce en dos áreas:

- El cálculo de una expresión aritmética que aparece como argumento de función se ve afectado por otros argumentos para dicha función.
- La evaluación de la función tiene en cuenta los atributos de todos sus argumentos.

Cuando se evalúa una función, sus argumentos se evalúan individualmente en el orden especificado en la lista de argumentos, de izquierda a derecha. El argumento que se está evaluando puede ser un identificador de función o una expresión que incluya identificadores de función.

Si se especifica una expresión aritmética como argumento y si el primer operador de la expresión es un signo más unario o un signo menos unario, la expresión debe ir inmediatamente precedida de un paréntesis izquierdo.

Los literales de coma flotante se permiten siempre que se permita un argumento numérico y en expresiones aritméticas utilizadas en funciones que permiten un argumento numérico.

Los elementos internos de coma flotante y los elementos externos de coma flotante (ambos muestran la coma flotante y la coma flotante nacional) se pueden utilizar siempre que se permita un argumento numérico y en expresiones aritméticas como argumentos para una función que permite un argumento numérico.

Los elementos de coma flotante y los literales de coma flotante *no pueden* utilizarse cuando se requiere un argumento entero o cuando se requiere un argumento de clase alfanumérico o nacional (como en las funciones LOWER-CASE, REVERSE, UPPER-CASE, NUMVAL y NUMVAL-C).

Ejemplos

Vea ejemplos de uso de diferentes tipos de funciones intrínsecas.

La sentencia siguiente ilustra el uso de la función intrínseca UPPER-CASE para sustituir cada letra minúscula en un argumento alfanumérico por la letra mayúscula correspondiente.

```
MOVE FUNCTION UPPER-CASE('hello') TO DATA-NAME.
```

Esta sentencia mueve HELLO a DATA-NAME.

La sentencia siguiente ilustra el uso de la función intrínseca LOWER-CASE para sustituir cada letra mayúscula en un argumento nacional por la letra minúscula correspondiente.

```
MOVE FUNCTION LOWER-CASE(N'HELLO') TO N-DATA-NAME.
```

Esta sentencia mueve los caracteres nacionales hello a N-DATA-NAME.

La sentencia siguiente ilustra el uso de una función intrínseca numérica:

```
COMPUTE NUM-ITEM = FUNCTION SUM(A B C)
```

Esta sentencia utiliza la función numérica SUM para añadir los valores de A, B y C y coloca el resultado en NUM-ITEM.

TODOS los suscripciones

Cuando una función permite que un argumento se repita un número variable de veces, puede hacer referencia a una tabla especificando el nombre de datos y cualquier calificador que identifique la tabla. Esto puede ser seguido inmediatamente por un subíndice donde uno o más de los subíndices es la palabra ALL.

Sugerencia: La evaluación de un subíndice ALL debe dar como resultado al menos un argumento o el valor devuelto por la función no estará definido; sin embargo, la situación se puede diagnosticar en tiempo de ejecución especificando la opción de compilador SSRANGE y la opción de tiempo de ejecución CHECK.

Especificar ALL como subíndice es equivalente a especificar todos los elementos de tabla posibles utilizando cada subíndice válido en esa posición de subíndice.

Para un argumento de tabla especificado como Table-name(ALL), el orden de la especificación implícita de cada elemento de tabla como argumento es de izquierda a derecha, donde el primer argumento (o más a la izquierda) es Table-name(1) y ALL se ha sustituido por 1. El siguiente argumento es Table-name(2), donde el subíndice se ha incrementado en 1. Este proceso continúa, con el subíndice incrementado en 1 para producir un argumento implícito, hasta que el subíndice ALL se ha incrementado a través de su rango de valores.

Por ejemplo,

```
FUNCTION MAX(Table(ALL))
```

es equivalente a

```
FUNCTION MAX(Table(1) Table(2) Table(3) ... Table(n))
```

donde *n* es el número de elementos de la tabla.

Si hay varios subíndices ALL, Table-name(ALL, ALL, ALL), el primer argumento implícito es Table-name(1, 1, 1), donde cada ALL se ha sustituido por 1. El siguiente argumento es Table-name(1, 1, 2), donde el subíndice situado más a la derecha se ha incrementado en 1. El subíndice representado por el ALL situado más a la derecha se incrementa a través de su rango de valores para producir un argumento implícito para cada valor.

Una vez que un subíndice especificado como ALL se ha incrementado a través de su rango de valores, el siguiente subíndice a la izquierda que se especifica como ALL se incrementa en 1. Cada subíndice especificado como ALL a la derecha del subíndice recién incrementado se establece en 1 para producir un argumento implícito. Una vez más, el subíndice representado por el ALL situado más a la derecha se incrementa a través de su rango de valores para producir un argumento implícito para cada valor. Este proceso se repite hasta que cada subíndice especificado como ALL se ha incrementado a través de su rango de valores.

Por ejemplo,

```
FUNCTION MAX(Table(ALL, ALL))
```

es equivalente a

```
FUNCTION MAX(Table(1, 1) Table(1, 2) Table(1, 3) ... Table(1, n)
             Table(2, 1) Table(2, 2) Table(2, 3) ... Table(2, n)
             Table(3, 1) Table(3, 2) Table(3, 3) ... Table(3, n)
             ...
             Table(m, 1) Table(m, 2) Table(m, 3) ... Table(m, n))
```

donde *n* es el número de elementos de la dimensión de columna de `Table`, y *m* es el número de elementos de la dimensión de fila de `Table`.

TODOS los subíndices se pueden combinar con subíndices literales, de nombre de datos o de nombre de índice para hacer referencia a tablas multidimensionales.

Por ejemplo,

```
FUNCTION MAX(Table(ALL, 2))
```

es equivalente a

```
FUNCTION MAX(Table(1, 2)
             Table(2, 2)
             Table(3, 2)
             ...
             Table(m, 2))
```

donde *m* es el número de elementos de la dimensión de fila de `Table`.

Si se especifica un subíndice ALL para un argumento y el argumento se modifica por referencia, el modificador de referencia se aplica a cada uno de los elementos especificados implícitamente de la tabla.

Si se especifica un subíndice ALL para un operando modificado por referencia, el modificador de referencia se aplica a cada uno de los elementos especificados implícitamente de la tabla.

Si el subíndice ALL está asociado con una cláusula OCCURS DEPENDING ON, el rango de valores lo determina el objeto de la cláusula OCCURS DEPENDING ON.

Por ejemplo, dada una definición de registro de nómina como:

```
01 PAYROLL.
   02 PAYROLL-WEEK   PIC 99.
   02 PAYROLL-HOURS  PIC 999 OCCURS 1 TO 52
        DEPENDING ON PAYROLL-WEEK.
```

Las siguientes sentencias COMPUTE podrían utilizarse para identificar las horas totales del año hasta la fecha, las horas máximas trabajadas en cualquier semana y la semana específica correspondiente a las horas máximas:

```
COMPUTE YTD-HOURS = FUNCTION SUM (PAYROLL-HOURS(ALL))
COMPUTE MAX-HOURS = FUNCTION MAX (PAYROLL-HOURS(ALL))
COMPUTE MAX-WEEK  = FUNCTION ORD-MAX (PAYROLL-HOURS(ALL))
```

En estas invocaciones de función, el subíndice ALL se utiliza para hacer referencia a todos los elementos de la matriz `PAYROLL-HOURS` (en función del valor de tiempo de ejecución del campo `PAYROLL-WEEK`).

Definiciones de función

Esta sección proporciona una visión general del tipo de argumento, tipo de función y valor devuelto para cada una de las funciones intrínsecas.

Para obtener más información sobre las funciones intrínsecas, consulte Tabla 59 en la página 458.

Los tipos de argumento y los tipos de función se abrevian como se indica a continuación:

Abreviatura	Significado
A	Alfabético
B	Booleano
D	DBCS
DE	Fecha-hora
I	Entero
IX	Índice
K	Palabra clave
M	Nombre mnemotécnico
N	Numérico
O	Otro, tal como se especifica en la definición de función (puntero, puntero de función, o puntero de procedimiento)
P	Puntero
S	Registro especial
T	Tipo-nombre
U	Nacional
X	Alfanumérico

El comportamiento de las funciones marcadas como"DP" depende de si la opción de compilador DATEPROC o NODATEPROC está en vigor.

Si la opción de compilador DATEPROC está en vigor, las siguientes funciones intrínsecas devuelven campos de fecha:

Función	El valor devuelto tiene un valor DATE FORMAT implícito
FECHA-DE-ENTERO	YYYYXXXX
DATE-TO-AAAAMMDD	YYYYXXXX
DÍA-OF-INTEGER	YYYYXXX
DAY-TO-AAAADDD	YYYYXXX
AÑO-A-AAAA	AAAA
DATEVAL	Depende del formato especificado por DATEVAL
VENTANA	AAAA

Si la opción de compilador NODATEPROC está en vigor:

- Las siguientes funciones intrínsecas devuelven los mismos valores que cuando DATEPROC está en vigor, pero sus valores devueltos son no fechas:
  - DÍA-OF-INTEGER
  - DATE-TO-AAAAMMDD
  - DAY-TO-AAAADDD
  - AÑO-A-AAAA
- Las funciones intrínsecas DATEVAL y UNDATE no tienen ningún efecto y simplemente devuelven sus (primeros) argumentos sin modificar.

- La función intrínseca YEARWINDOW devuelve 0 incondicionalmente.

Cada función intrínseca se describe en detalle en los temas que siguen a la tabla siguiente.

Tabla 59. ***Tabla de funciones***

Nombre de función	**argumentos**	**Tipo de función**	**Valor devuelto**
ACOS	N1	N	Arcoseno de N1
AÑADIR DURACIÓN	DA1, K2, I3	DE	Un elemento de fecha-hora con la duración añadida
ANNUIDAD	N1, I2	N	Ratio de anualidad pagada para los períodos I2 en interés de N1 respecto a la inversión inicial de uno
ASIN	N1	N	Arcoseno de N1
ATAN	N1	N	Arcotangente de N1
char	I1	X	Carácter en la posición I1 del orden de clasificación del programa
SOCQ	N1	N	Coseno de N1
ACTUAL-FECHA	Ninguna	X	Fecha y hora actuales y diferencia con respecto a la hora media de Greenwich
CONVERTIR-FECHA-HORA	DA1 o X1 o I1, K2, X3 o K3 o S3, M4 o S4	DE	Elemento de fecha-hora convertido
DATE-OF-INTEGER[DP]	I1	I	Fecha estándar equivalente (AAAAMMDD) de fecha entera
DATE-TO-AAAAMMDD[DP]	I1, I2	I	Fecha estándar equivalente (AAAAMMDD) de I1 (fecha estándar con un año con ventana, AAMMDD), según el intervalo de 100 años cuyo año final se especifica mediante la suma de I2 y el año en tiempo de ejecución
DATEVAL[DP]	I1	I	Campo de fecha equivalente a I1
	X1	X	Campo de fecha equivalente a X1
DAY-OF-INTEGER[DP]	I1	I	Fecha juliana equivalente (AAAADDD) de fecha entera
DAY-TO-YYYYDDD[DP]	I1, I2	I	Equivalente de fecha juliana (YYYYDDD) de I1 (fecha juliana con un año con ventana, YYDDD), según el intervalo de 100 años cuyo año final se especifica mediante la suma de I2 y el año en tiempo de ejecución
VISUALIZAR-DE	U1 o U1, I2, o U1, X2o D2	X	Cada carácter de U1 se convierte en una representación de caracteres correspondiente utilizando una página de códigos identificada mediante I2, si se especifica, o la página de códigos especificada por el entorno local de ejecución si I2 no está especificado

Tabla 59. ***Tabla de funciones*** *(continuación)*

Nombre de función	**argumentos**	**Tipo de función**	**Valor devuelto**
EXTRAER-FECHA-HORA	DA1, X2o K2	I o X	Parte de un elemento de fecha, hora o indicación de fecha y hora extraído
FACTORIAL	I1	I	Factorial de I1
BUSCAR DURACIÓN	DA1, DA2o K3	I	La duración entera entre 2 elementos de fecha y hora
Entero	N1	I	El entero mayor no mayor que N1
INTEGER-OF-DATE	I1	I	Fecha entera equivalente a la fecha estándar (AAAAMMDD)
INTEGER-OF-DAY	I1	I	Fecha entera equivalente a la fecha juliana (AAAADDD)
ENTERO-PARTE	N1	I	Parte entera de N1
LENGTH	A1, B1, D1, DA1, IX1, N1, O1, P1, T1, X1o U1	I	Longitud del argumento en posiciones de caracteres nacionales o en posiciones de caracteres alfanuméricos o bytes, en función del tipo de argumento
log	N1	N	Logaritmo natural de N1
LOG10	N1	N	Logaritmo en base 10 de N1
LOWER-CASE	A1 o X1	X	Todas las letras del argumento establecidas en minúsculas
	U1	U	Todas las letras del argumento establecidas en minúsculas
MÁX	A1...	X	Valor del argumento máximo; tenga en cuenta que el tipo de función depende de los argumentos
	I1...	I	Valor del argumento máximo; tenga en cuenta que el tipo de función depende de los argumentos
	N1...	N	Valor del argumento máximo; tenga en cuenta que el tipo de función depende de los argumentos
	X1...	X	Valor del argumento máximo; tenga en cuenta que el tipo de función depende de los argumentos
	U1...	U	Valor del argumento máximo; tenga en cuenta que el tipo de función depende de los argumentos
MEAN	N1...	N	Media aritmética de argumentos
MEDIO	N1...	N	Mediana de argumentos
MIDRANGE	N1...	N	Media de argumentos mínimos y máximos

Tabla 59. ***Tabla de funciones*** *(continuación)*

Nombre de función	argumentos	Tipo de función	Valor devuelto
MIN	A1...	X	Valor del argumento mínimo; tenga en cuenta que el tipo de función depende de los argumentos
	I1...	I	Valor del argumento mínimo; tenga en cuenta que el tipo de función depende de los argumentos
	N1...	N	Valor del argumento mínimo; tenga en cuenta que el tipo de función depende de los argumentos
	X1...	X	Valor del argumento mínimo; tenga en cuenta que el tipo de función depende de los argumentos
	U1...	U	Valor del argumento mínimo; tenga en cuenta que el tipo de función depende de los argumentos
MOD	I1, I2	I	I1 módulo I2
NACIONAL-DE	A1, X1, o D1	U	Los caracteres del argumento convertidos a caracteres nacionales, utilizando la página de códigos identificada por I2, si se especifica, o una página de códigos especificada por el entorno local de tiempo de ejecución si I2 no está especificado
	A1, X1, o D1; I2	U	Los caracteres del argumento convertidos a caracteres nacionales, utilizando la página de códigos identificada por I2, si se especifica, o una página de códigos especificada por el entorno local de tiempo de ejecución si I2 no está especificado
	A1 o X1 o D1, U2	U	Los caracteres del argumento convertidos a caracteres nacionales, utilizando la página de códigos identificada por I2, si se especifica, o una página de códigos especificada por el entorno local de tiempo de ejecución si I2 no está especificado
NUMVAL	X1 o U1	N	Valor numérico de cadena numérica simple
NUMVAL-C	X1 o U1; X1, X2; U1, U2	N	Valor numérico de cadena numérica con comas opcionales y signo de moneda
VOR	A1 o X1	I	Posición ordinal del argumento en el orden de clasificación
ORD-MAX	A1..., N1..., X1..., o U1...	I	Posición ordinal del argumento máximo
SEÑOR-MIN	A1..., N1..., X1..., o U1...	I	Posición ordinal del argumento mínimo

*Tabla 59. **Tabla de funciones** (continuación)*

Nombre de función	argumentos	Tipo de función	Valor devuelto
VALOR ACTUAL	N1, N2...	N	Valor actual de una serie de importes finales de período futuros, N2, a una tasa de descuento de N1
ALEATORIO	I1, ninguno	N	Número aleatorio
RANGO	I1...	I	Valor del argumento máximo menos valor del argumento mínimo; tenga en cuenta que el tipo de función depende de los argumentos.
	N1...	N	Valor del argumento máximo menos valor del argumento mínimo; tenga en cuenta que el tipo de función depende de los argumentos.
REM	N1, N2	N	Resto de N1/N2
INVERTIR	A1 o X1	X	Orden inverso de los caracteres del argumento
	U1	U	Orden inverso de los caracteres del argumento
SIN	N1	N	Seno de N1
SQRT	N1	N	Raíz cuadrada de N1
DESVIACIÓN TÍPICA	N1...	N	Desviación estándar de argumentos
RESTAR DURACIÓN	DA1, K2, I3	DE	Un elemento de fecha y hora con la duración restada
SUM	I1...	I	Suma de argumentos; tenga en cuenta que el tipo de función depende de los argumentos.
	N1...	N	Suma de argumentos; tenga en cuenta que el tipo de función depende de los argumentos.
TAN	N1	N	Tangente de N1
TEST-FECHA-HORA	DA1 o X1 o I1, K2, X3 o K3 o S3, M4 o S4	B	Verdadero (B "1") si es un elemento de fecha-hora válido, de lo contrario es falso
TRIM	A1 o X1	X	Serie con espacios en blanco a la izquierda y a la derecha o caracteres especificados recortados
	A1, A2 o X1, X2		Serie con espacios en blanco a la izquierda y a la derecha o caracteres especificados recortados
	D2 o D1, D2	D	Serie con espacios en blanco a la izquierda y a la derecha o caracteres especificados recortados
	NL1 o NL1, NL2	NL	Serie con espacios en blanco a la izquierda y a la derecha o caracteres especificados recortados

*Tabla 59. **Tabla de funciones** (continuación)*

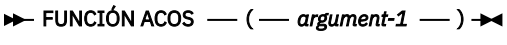
Nombre de función	argumentos	Tipo de función	Valor devuelto
TRIML	A1 o X1	X	Serie con espacios en blanco a la izquierda o caracteres especificados recortados
	A1, A2 o X1, X2		Serie con espacios en blanco a la izquierda o caracteres especificados recortados
	D1 o D1, D2	D	Serie con espacios en blanco a la izquierda o caracteres especificados recortados
	NL1 o NL1, NL2	NL	Serie con espacios en blanco a la izquierda o caracteres especificados recortados
TRIMR	A1 o X1	X	Serie con espacios en blanco a la derecha o caracteres especificados recortados
	A1, A2 o X1, X2		Serie con espacios en blanco a la derecha o caracteres especificados recortados
	D1 o D1, D2	D	Serie con espacios en blanco a la derecha o caracteres especificados recortados
	NL1 o NL1, NL2	NL	Serie con espacios en blanco a la derecha o caracteres especificados recortados
UNDATE[DP]	I1	I	No fecha equivalente al campo de fecha I1 o X1
	X1	X	No fecha equivalente al campo de fecha I1 o X1
UPPER-CASE	A1 o X1	X	Todas las letras del argumento establecidas en mayúsculas
	U1	U	Todas las letras del argumento establecidas en mayúsculas
UTF8STRING	A1, X1, D1o NL1	A	Cadena de caracteres UTF-8 de longitud variable
VARIANCE	N1...	N	Varianza de argumentos
WHEN-COMPILADO	Ninguna	X	Fecha y hora en que se compiló el programa
YEAR-TO-YYYY[DP]	I1, I2	I	Equivalente de año expandido (AAAA) de I1 (año con ventana, AA), según el intervalo de 100 años cuyo año final se especifica mediante la suma de I2 y el año en tiempo de ejecución
YEARWINDOW[DP]	Ninguna	I	Si la opción de compilador DATEPROC está en vigor, devuelve el año inicial (con el formato AAAA) de la ventana de siglo especificada por la opción de compilador YEARWINDOW; si NODATEPROC está en vigor, devuelve 0

ACOS

La función ACOS devuelve un valor numérico en radianes que se aproxima al arcocoseno del argumento especificado.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica. El valor de *argument-1* debe ser mayor o igual que -1 y menor o igual que +1.

El valor devuelto es la aproximación del arcocoseno del argumento y es mayor o igual que cero y menor o igual que Pi.

AÑADIR DURACIÓN

La función ADD-DURATION añade una duración a un elemento de fecha, hora o indicación de fecha y hora y devuelve el elemento modificado.

El tipo de función es fecha-hora.

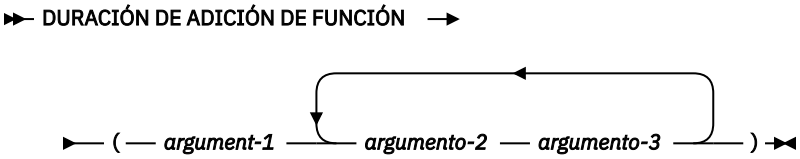
La longitud del valor devuelto depende de la longitud del elemento de fecha, hora o indicación de fecha y hora especificado en *argument-1*. El valor devuelto se truncará a la longitud de *argument-1*.

Si se añade una duración a un elemento de fecha, la fecha devuelta debe estar dentro de un rango determinado:

- Para las fechas de 4 dígitos, el rango debe ser de 0001/01/01 a 9999/12/31
- Para fechas de 2 dígitos, el rango debe ser de 0001/01/01 a 9999/12/31, pero el año se trunca a 2 dígitos
- Para un año de 3 dígitos (un siglo de 1 dígito y un año de 2 dígitos), el rango debe ser de 1900/01/01 a 2899/12/31 (el valor predeterminado). Este rango se puede cambiar especificando la opción de compilador DATETIME .

Si se añade una duración a un elemento de fecha de 2 dígitos, el rango es el mismo que para un año de 4 dígitos, pero el año del valor devuelto se trunca a 2 dígitos.

Formato



argument-1
Debe ser un elemento de datos de fecha, hora o indicación de fecha y hora.

argument-1 es un elemento de datos que contiene un valor al que se añade una duración. La duración se especifica en *argument-2* y *argument-3*.

argument-2
argument-2 es una palabra clave que representa una duración. Las palabras clave de duración válidas son:

- AÑOS
- MESES
- DÍAS
- HORAS
- MINUTOS

- SEGUNDOS
- MICROSEGUNDOS
- PICOSEGUNDOS

La palabra clave de duración debe ser coherente con *argument-1*. Por ejemplo, las palabras clave de duración más obedecen a las reglas siguientes:

1. YEARS, MONTHS y DAYS solo se pueden añadir a un elemento de fecha o indicación de fecha y hora.
2. HOURS, MINUTES, SECONDS y MICROSECONDS solo se pueden añadir a un elemento de hora o indicación de fecha y hora.
3. PICOSECONDS sólo se puede añadir a un elemento de indicación de fecha y hora.

argument-3
Debe ser una expresión aritmética entera. *argument-3* es el número de unidades de la duración, tal como se especifica en *argument-2*, que se van a añadir a *argument-1*.

argument-3 puede ser un entero negativo, pero la función sólo toma su valor absoluto. Si *argument-3* tiene más de 9 dígitos, se trunca.

argument-2 y *argument-3* se pueden repetir. No debe haber ningún *argument-2* duplicado en una función intrínseca.

Si se añade una duración a una fecha y el resultado no es válido, la fecha se ajusta. Por ejemplo, si se añade una duración de 1 mes a la fecha `March 31, 1997`, el resultado sería la fecha no válida `April 31, 1997`. Esta fecha se ajustará a la fecha válida `April 30, 1997`.

Ejemplos

Los ejemplos siguientes muestran cómo se puede utilizar la función intrínseca ADD-DURATION:

```
MOVE FUNCTION ADD-DURATION (date-3 MONTHS 1)
     TO  date-2.
MOVE FUNCTION ADD-DURATION (date-3 MONTHS int-1 * 2)
     TO  date-1.
MOVE FUNCTION ADD-DURATION (date-1  YEARS 1 MONTHS 5 DAYS 23)
     TO date-2.
```

Referencias relacionadas
“RESTAR DURACIÓN” en la página 492

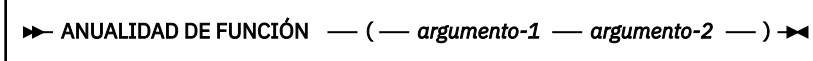
ANNUIDAD

La función ANNUITY devuelve un valor numérico que se aproxima a la proporción de una anualidad pagada al final de cada período, para un número determinado de períodos, a un tipo de interés determinado, a un valor inicial de uno.

El número de periodos se especifica mediante *argumento-2*; el tipo de interés se especifica mediante *argument-1*. Por ejemplo, si *argument-1* es cero y *argument-2* es cuatro, el valor devuelto es la aproximación de la razón 1/4.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica. El valor de *argument-1* debe ser mayor o igual que cero.

argument-2
Debe ser un entero positivo.

Cuando el valor de *argument-1* es cero, el valor devuelto por la función es la aproximación de:

1/ *argument-2*

Cuando el valor de *argument-1* no es cero, el valor de la función es la aproximación de:

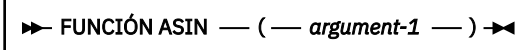
argument-1 /(1-(1 + *argument-1*) ** (- *argument-2*))

ASIN

La función ASIN devuelve un valor numérico en radianes que se aproxima al arcoseno del argumento especificado.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica. El valor de *argument-1* debe ser mayor o igual que -1 y menor o igual que +1.

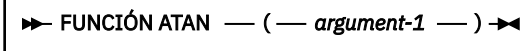
El valor devuelto es la aproximación del arcoseno del *argument-1* y es mayor o igual que -Pi/2 y menor o igual que +Pi/2.

ATAN

La función ATAN devuelve un valor numérico en radianes que se aproxima a la arcotangente del argumento especificado.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

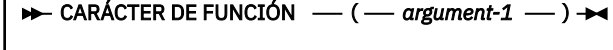
El valor devuelto es la aproximación de la arcotangente del *argument-1* y es mayor que -Pi/2 y menor que +Pi/2.

char

La función CHAR devuelve un valor alfanumérico de un carácter que es un carácter del orden de clasificación del programa que tiene la posición ordinal igual al valor del argumento especificado.

El tipo de función es alfanumérico.

Formato



argument-1
Debe ser un entero. El valor debe ser mayor que cero y menor o igual que el número de posiciones en la secuencia de clasificación asociada con elementos de datos alfanuméricos (un máximo de 256).

Si más de un carácter tiene la misma posición en la secuencia de clasificación del programa, el carácter devuelto como valor de función es el del primer literal especificado para dicha posición de carácter en la cláusula ALPHABET.

Si la secuencia de clasificación del programa actual no se ha especificado mediante una cláusula ALPHABET, la opción de compilador COLLSEQ indica el orden de clasificación utilizado. Por ejemplo, si se especifica COLLSEQ (EBCDIC) y no se especifica PROGRAM COLLATING SEQUENCE (o es NATIVE), se aplica la secuencia de clasificación EBCDIC. (Consulte “Expresiones condicionales” en la página 250.)

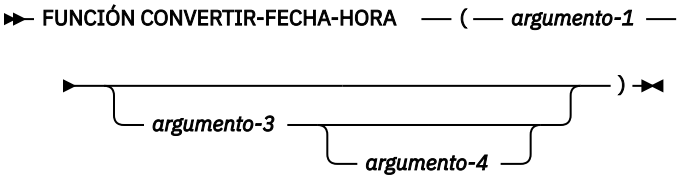
CONVERTIR-FECHA-HORA

La función CONVERT-DATE-TIME toma un elemento de clase alfanumérico, numérico o de fecha-hora y devuelve un elemento de fecha-hora.

El tipo de función es fecha-hora.

La longitud del valor devuelto depende de la longitud permitida para el formato del elemento de fecha, hora o indicación de fecha y hora especificado en el *argument-2* a través del *argument-4*.

Formato



argument-1
Puede ser:

- Un elemento de fecha, hora o indicación de fecha y hora
- Un elemento de clase alfanumérico
- Un literal no numérico
- Un elemento de entero numérico de clase

argument-2
Especifica la categoría del valor de retorno y debe ser una de las palabras clave siguientes:

- fecha
- hora
- INDICACIÓN de fecha y hora

Si *argument-1* es un elemento de fecha, hora o indicación de fecha y hora, CONVERT-DATE-TIME sólo puede convertir:

- Una fecha hasta una fecha o una indicación de fecha y hora
- Una hora a una hora o una indicación de fecha y hora
- Una indicación de fecha y hora a una fecha, una hora o una indicación de fecha y hora

Si *argument-2* es TIMESTAMP, *argument-3* sólo se puede especificar con el registro especial FORMAT OF, y *argument-4* no se pueden especificar.

Si *argument-1* es un elemento de fecha-hora, se realiza un movimiento de fecha-hora.

Si *argument-1* es un entero numérico, el elemento de fecha-hora devuelto se justificará a la derecha y se truncará, si es más largo de lo que permite el formato de fecha-hora especificado en *argument-3*.

Si *argument-1* es cualquier otra cosa, el elemento de fecha-hora devuelto se justificará a la izquierda y se truncará, si es más largo de lo que permite el formato de fecha-hora especificado en *argument-3*.

argument-3
Especifica el formato de un elemento de fecha u hora. Debe ser:

- Un literal no numérico de al menos 2 caracteres de longitud
- La palabra clave LOCALE
- El registro especial FORMAT OF

Para obtener una lista de literales válidos y las reglas que este argumento debe seguir, consulte la cláusula SPECIAL-NAMES FORMAT descrita en “cláusula FORMAT” en la página 104.

argument-3 debe representar una categoría a la que se hace referencia mediante *argument-2*.

Si *argument-3* es la palabra clave LOCALE, el formato de la fecha u hora se basa en un LOCALE. Si no se especifica *argument-4* , se utiliza el entorno local actual; de lo contrario, se utiliza el entorno local asociado con el nombre mnemotécnico o el registro especial LOCALE OF.

Si no se especifica *argument-3* , el formato del valor devuelto depende de la cláusula SPECIAL-NAMES FORMAT. Si no se ha definido ningún formato en el párrafo SPECIAL-NAMES, se utiliza el formato *ISO. Para TIMESTAMP, si no se especifica *argument-3* , se utiliza el formato predeterminado de @Y-%m-%d-%H%M%S.@Sm .

argument-4
Debe ser un nombre mnemotécnico asociado con un LOCALE, o el registro especial LOCALE OF.

argument-4 debe seguir estas reglas:

- Si se especifica *argument-4* y *argument-3* es un literal de formato basado en el entorno local, por ejemplo, contiene %p, entonces el literal de formato basado en el entorno local utilizaría el entorno local especificado en *argument-4* para determinar el valor real de los especificadores de conversión.
- Si *argument-3* es un literal de formato basado en entorno local (por ejemplo, contiene %p) y no se especifica *argument-4* , el literal de formato basado en entorno local utilizaría el entorno local actual para determinar el valor real de los especificadores de conversión.
- Si *argument-3* es un literal de formato basado en el entorno local (por ejemplo, contiene %p) y el registro especial LOCALE OF se utiliza para hacer referencia a un elemento que no es del entorno local, el literal de formato basado en el entorno local utilizaría el entorno local predeterminado para determinar el valor real de los especificadores de conversión.

Ejemplos

Los ejemplos siguientes muestran cómo se puede utilizar la función intrínseca CONVERT-DATE-TIME:

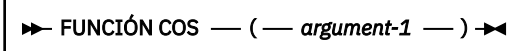
```
MOVE FUNCTION CONVERT-DATE-TIME ('95/05/30' DATE)
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE '%y/%m/%d')
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE '%y/%m/%d' my-locale)
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE LOCALE my-locale)
     TO  date-1.
```

SOCQ

La función COS devuelve un valor numérico que se aproxima al coseno del ángulo o arco especificado por el argumento en radianes.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

El valor devuelto es la aproximación del coseno del argumento y es mayor o igual que -1 y menor o igual que +1.

ACTUAL-FECHA

La función CURRENT-DATE devuelve un valor alfanumérico de 21 caracteres que representa la fecha de calendario, la hora del día y el diferencial de tiempo con respecto a la hora media de Greenwich proporcionada por el sistema en el que se evalúa la función.

El tipo de función es alfanumérico.

Formato

Leyendo de izquierda a derecha, las posiciones de 21 caracteres del valor devuelto son las siguientes:

Posiciones de caracteres	Contenido
1-4	Cuatro dígitos numéricos del año en el calendario gregoriano
5-6	Dos dígitos numéricos del mes del año, en el rango de 01 a 12
7-8	Dos dígitos numéricos del día del mes, en el rango de 01 a 31
9-10	Dos dígitos numéricos de las horas posteriores a la medianoche, en el rango de 00 a 23
11-12	Dos dígitos numéricos de los minutos después de la hora, en el rango de 00 a 59
13-14	Dos dígitos numéricos de los segundos después del minuto, en el rango de 00 a 59
15-16	Dos dígitos numéricos de las centésimas de segundo después del segundo, en el rango de 00 a 99. El valor 00 se devuelve si el sistema en el que se evalúa la función no tiene el recurso para proporcionar la parte fraccional de un segundo.
17	El carácter '-' o el carácter '+'. Se devuelve el carácter '-' si la hora local indicada en las posiciones de caracteres anteriores está detrás de la hora media de Greenwich. El carácter '+' se devuelve si la hora local indicada es igual o anterior a la hora media de Greenwich. El carácter '0' se devuelve si el sistema en el que se evalúa esta función no tiene el recurso para proporcionar el factor diferencial de hora local.
18-19	Si la posición 17 del carácter es '-', se devuelven dos dígitos numéricos en el rango de 00 a 12 que indican el número de horas que la hora notificada está por detrás de la hora media de Greenwich. Si la posición 17 del carácter es '+', se devuelven dos dígitos numéricos en el rango de 00 a 13 que indican el número de horas que la hora notificada está por delante de la hora media de Greenwich. Si la posición de carácter 17 es '0', se devuelve el valor 00.

Posiciones de caracteres	Contenido
20-21	Se devuelven dos dígitos numéricos en el rango de 00 a 59 que indican el número de minutos adicionales que la hora notificada está por delante o por detrás de la hora media de Greenwich, dependiendo de si la posición 17 del carácter es '+' o '-', respectivamente. Si la posición de carácter 17 es '0', se devuelve el valor 00.

Ejemplo

Dado que la indicación de fecha y hora del entorno operativo actual es "1995-02-15 05:14:27.812479168304" Eastern Standard Time, FUNCTION CURRENT-DATE devuelve un campo alfanumérico de 21 caracteres que se puede utilizar de la siguiente manera:

```
WORKING-STORAGE SECTION.
01  WS-CURRENT-DATE-FIELDS.
    05  WS-CURRENT-DATE.
        10  WS-CURRENT-YEAR    PIC  9(4).
        10  WS-CURRENT-MONTH   PIC  9(2).
        10  WS-CURRENT-DAY     PIC  9(2).
    05  WS-CURRENT-TIME.
        10  WS-CURRENT-HOUR    PIC  9(2).
        10  WS-CURRENT-MINUTE  PIC  9(2).
        10  WS-CURRENT-SECOND  PIC  9(2).
        10  WS-CURRENT-MS      PIC  9(2).
    05  WS-DIFF-FROM-GMT       PIC S9(4).
PROCEDURE DIVISION.
    MOVE FUNCTION CURRENT-DATE TO WS-CURRENT-DATE-FIELDS
```

WS-CURRENT-DATE-FIELDS contiene "1995021505142781-0500".

Para obtener más información, consulte *Ejemplos: funciones intrínsecas numéricas* en *COBOL for Linux en x86 Guía de programación*.

FECHA-DE-ENTERO

La función DATE-OF-INTEGER convierte una fecha del calendario gregoriano del formato de fecha entera al formato de fecha estándar (AAAAMMDD).

El tipo de función es entero.

El resultado de la función es un entero de ocho dígitos.

Si la opción de compilador DATEPROC está en vigor, el valor devuelto es un campo de fecha expandido con DATE FORMAT YYYYXXXX implícito.

Formato

argument-1
Un entero positivo que representa un número de días posteriores al 31 de diciembre de 1600, en el calendario gregoriano. El rango válido es de 1 a 3.067.671, que corresponde a fechas comprendidas entre el 1 de enero de 1601 y el 31 de diciembre de 9999.

El valor devuelto representa la fecha estándar ISO (International Standards Organization) equivalente al entero especificado como *argument-1*.

El valor devuelto es un entero con el formato AAAAMMDD donde AAAA representa un año en el calendario gregoriano; MM representa el mes de ese año; y DD representa el día de ese mes.

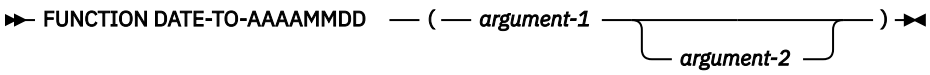
DATE-TO-AAAAMMDD

La función DATE-TO-YYYYMMDD convierte *argument-1* de una fecha con un año de dos dígitos (YYnnnn) a una fecha con un año de cuatro dígitos (YYYYnnnn). *argument-2*, cuando se añade al año en el momento de la ejecución, define el año final de un intervalo de 100 años, o ventana de siglo deslizante, en el que cae el año del *argument-1* .

El tipo de función es entero.

Si la opción de compilador DATEPROC está en vigor, el valor devuelto es un campo de fecha expandido con DATE FORMAT YYYYXXXX implícito.

Formato



argument-1
Debe ser cero o un entero positivo menor que 991232.

Nota: El tiempo de ejecución COBOL no verifica que el valor sea una fecha válida.

argument-2
Debe ser un entero. Si se omite *argument-2* , la función se evalúa suponiendo que se haya especificado el valor 50.

La suma del año en el momento de la ejecución y el valor de *argument-2* debe ser menor que 10.000 y mayor que 1.699.

Consulte los ejemplos siguientes con valores devueltos por de la función DATE-TO-YYYYMMDD:

Año actual	Valor *argument-1*	Valor *argument-2*	Valor devuelto
2002	851003	120	20851003
2002	851003	-20	18851003
2002	851003	10	19851003
1994	981002	-10	18981002

DATEVAL

La función DATEVAL convierte una no fecha en un campo de fecha, para un uso inequívoco con campos de fecha.

Si la opción de compilador DATEPROC está en vigor, el valor devuelto es un campo de fecha que contiene el valor de *argument-1* sin modificar. Para obtener información sobre cómo utilizar el campo de fecha resultante:

- En aritmética, consulte “Aritmética con campos de fecha” en la página 248
- En expresiones condicionales, consulte “Comparación de campos de fecha” en la página 264

Si la opción de compilador NODATEPROC está en vigor, la función DATEVAL no tiene ningún efecto y devuelve el valor de *argumento*-1 sin modificar.

El tipo de función depende del tipo de *argumento-1*:

Tipo de argumento	Tipo de función
Alfanumérico	Alfanumérico

Tipo de argumento	Tipo de función
Entero	Entero

Formato

argument-1
Debe ser uno de los siguientes:

- Elemento alfanumérico de clase con el mismo número de caracteres que el formato de fecha especificado por *argument-2*.
- Un entero. Esto se puede utilizar para especificar valores fuera del rango especificado por *argument-2*, incluidos los valores negativos.

El valor de *argument-1* representa una fecha del formato especificado por *argument-2*.

argument-2
Debe ser un literal alfanumérico que especifique un patrón de fecha, tal como se define en "cláusula DATE FORMAT" en la página 172. El patrón de fecha consta de AA o AAAA (que representa un año con ventana o un año expandido, respectivamente), opcionalmente precedido o seguido de una o más X (que representan otras partes de una fecha, como el mes y el día), tal como se muestra a continuación. Tenga en cuenta que los valores no distinguen entre mayúsculas y minúsculas; las letras X e Y del *argument-2* pueden ser cualquier combinación de mayúsculas y minúsculas.

Serie de patrón de fecha	Especifica que *argument-1* contiene
AA	Un año con ventana (dos dígitos)
AAAA	Un año expandido (cuatro dígitos)
X	Un único carácter; por ejemplo, un dígito que representa un semestre o trimestre (1–4)
XX	Dos caracteres; por ejemplo, dígitos que representan un mes (01–12)
XXX	Tres caracteres; por ejemplo, dígitos que representan un día del año (001–366)
XXXX	Cuatro caracteres; por ejemplo, dos dígitos que representan un mes (01–12) y dos dígitos que representan un día del mes (01–31)

DÍA-OF-INTEGER

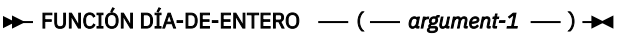
La función DAY-OF-INTEGER convierte una fecha del calendario gregoriano del formato de fecha entera al formato de fecha Juliana (AAAADDD).

El tipo de función es entero.

El resultado de la función es un entero de siete dígitos.

Si la opción de compilador DATEPROC está en vigor, el valor devuelto es un campo de fecha expandido con DATE FORMAT YYYYXXX implícito.

Formato



argument-1
Un entero positivo que representa un número de días posteriores al 31 de diciembre de 1600, en el calendario gregoriano. El rango válido es de 1 a 3.067.671, que corresponde a fechas comprendidas entre el 1 de enero de 1601 y el 31 de diciembre de 9999.

El valor devuelto representa el equivalente juliano del entero especificado como *argument-1*. El valor devuelto es un entero con el formato AAAADD, donde AAAA representa un año en el calendario gregoriano y DDD representa el día de ese año.

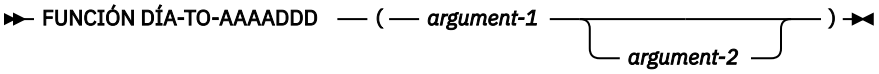
DAY-TO-AAAADDD

La función DAY-TO-YYYYDDD convierte *argument-1* de una fecha con un año de dos dígitos (YYnnn) a una fecha con un año de cuatro dígitos (YYYYnnn). *argument-2*, cuando se añade al año en el momento de la ejecución, define el año final de un intervalo de 100 años, o ventana de siglo deslizante, en el que cae el año del *argument-1* .

El tipo de función es entero.

Si la opción de compilador DATEPROC está en vigor, el valor devuelto es un campo de fecha expandido con DATE FORMAT YYYYXXX implícito.

Formato



argument-1
Debe ser cero o un entero positivo menor que 99367.

El tiempo de ejecución COBOL no verifica que el valor sea una fecha válida.

argument-2
Debe ser un entero. Si se omite *argument-2* , la función se evalúa suponiendo que se haya especificado el valor 50.

La suma del año en el momento de la ejecución y el valor de *argument-2* debe ser menor que 10.000 y mayor que 1.699.

A continuación se muestran algunos ejemplos de valores devueltos de la función DAY-TO-YYYYYYDDD:

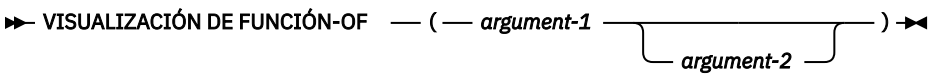
Año actual	**Valor *argument-1***	**Valor *argument-2***	**Valor devuelto**
2002	10004	-20	1910004
2002	10004	-120	1810004
2002	10004	20	2010004
2013	95005	-10	1995005

VISUALIZAR-DE

La función DISPLAY-OF devuelve una serie de caracteres alfanuméricos que consta del contenido del *argument-1* convertido a una representación de página de códigos específica.

El tipo de la función es alfanumérico.

Formato



argument-1
Debe ser de clase nacional (categorías nacionales, nacionales editadas y numéricas editadas descritas con el uso NATIONAL). *argument-1* identifica la serie de origen para la conversión.

argument-2
Debe ser un entero o de clase alfanumérica. *argument-2* identifica la página de códigos de salida para la conversión.

Si *argument-2* es de clase alfanumérica, debe identificar un nombre de página de códigos primario o alias soportado por las bibliotecas de conversión ICU (consulte *International Components for Unicode: Converter Explorer*).

Si *argument-2* es un entero, el entero debe ser un número CCSID válido.

Si se omite *argument-2* , la página de códigos de salida se determina a partir del entorno local de ejecución.

El valor devuelto es una serie de caracteres alfanuméricos que consta de los caracteres del *argument-1* convertidos a la representación de página de códigos de salida. Cuando un carácter de origen no se puede convertir a un carácter en la página de códigos de salida, el carácter de origen se sustituye por un carácter de sustitución. La tabla siguiente muestra caracteres de sustitución para algunas páginas de códigos de uso amplio:

Página de códigos de salida	**Carácter de sustitución**
SBCS ASCII PC Windows SBCS	X'7F'
SBCS ISO	X'1A'
SBCS EBCDIC	X'3F'
DBCS ASCII	X'FCFC "
EBCDIC DBCS (excepto para tailandés)	X'FEFE "
EBCDIC DBCS (tailandés)	X'41B8'
PC DBCS (japonés o chino)	X'FCFC "
PC DBCS (coreano)	X'BFFC '
EUC (coreano)	X'AFFE "
EUC (japonés)	X'747E'
UTF-8	Desde SBCS: X'1A' Desde MBCS: X'EFBFBD '
UTF-16	De SBCS: X'1A00' De MBCS: X'FDFF '

No se genera ninguna condición de excepción.

La longitud del valor devuelto depende del contenido del *argument-1* y de las características de la página de códigos de salida.

Notas de uso

- El uso de nombres de página de códigos proporciona coherencia con otro software de Linux , pero el código fuente no es portable a Enterprise COBOL for z/OS.
- El CCSID para UTF-8 es 1208.
- Si la página de códigos de salida incluye caracteres DBCS, el valor devuelto puede ser una serie SBCS y DBCS mixta.
- La función DISPLAY-OF, con *argument-2* especificado, se puede utilizar para generar datos de caracteres representados en una página de códigos que difiera de la página de códigos en vigor para datos ASCII o EBCDIC. Las operaciones COBOL posteriores en esos datos pueden implicar conversiones implícitas que presuponen que los datos se representan en la página de códigos ASCII o EBCDIC en vigor. Consulte *Conversión a o desde representación nacional (Unicode)* en la publicación *COBOL for Linux en x86 Guía de programación* para ver ejemplos y técnicas de programación para procesar datos representados utilizando más de una página de códigos dentro de un único programa.

Excepción: si la conversión falla, se produce un error de tiempo de ejecución grave.

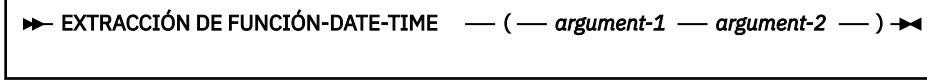
EXTRAER-FECHA-HORA

La función EXTRACT-DATE-TIME devuelve parte de un elemento de fecha, hora o indicación de fecha y hora.

El tipo de función es entero o alfanumérico. Si *argument-2* es una palabra clave (por ejemplo, MONTHS o DAYS), o consta sólo de especificadores numéricos, se devuelve un entero. De lo contrario, se devuelve un elemento de datos alfanumérico.

La longitud del resultado depende de los valores extraídos del elemento de fecha-hora.

Formato



argument-1
Debe ser un elemento de fecha, hora o indicación de fecha y hora.

argument-2
Especifica los valores que debe devolver la función EXTRACT-DATE-TIME.

Argument-2 es una palabra clave que representa una duración o un literal no numérico que contiene uno o más separadores y especificaciones de conversión.

Si el literal no numérico contiene sólo especificadores de conversión numérica, el valor devuelto por la función EXTRACT-DATE-TIME es un entero. Un literal no numérico que contiene separadores o especificadores de conversión alfanuméricos da como resultado un valor de retorno alfanumérico.

Las duraciones válidas y sus especificaciones de conversión equivalentes son:

- AÑOS ('@Y')
- MESES ('%m')
- DÍAS ('%d')
- HORAS ('%H')
- MINUTOS ('%M')
- SEGUNDOS ('%S')
- MICROSEGUNDOS ('@Sm')
- PICOSECONDS ('@Sp')

Para obtener una lista de otras especificaciones de conversión válidas, consulte la Tabla 1 en la descripción de la cláusula FORMAT del párrafo SPECIAL-NAMES.

La palabra clave de duración o el especificador de conversión utilizado en *argument-2* debe ser coherente con *argument-1*. Por ejemplo, las palabras clave de duración deben cumplir las reglas siguientes:

1. YEARS, MONTHS y DAYS solo se pueden extraer de un elemento de fecha o indicación de fecha y hora.
2. HOURS, MINUTES, SECONDS y MICROSECONDS solo se pueden extraer de un elemento de hora o indicación de fecha y hora.
3. Si *argument-1* es un elemento de datos basado en el entorno local y *argument-2* contiene especificadores de conversión basados en el entorno local (como %p), el especificador de conversión basado en el entorno local (%p, en este caso) utiliza el entorno local de *argument-1*.

   Si *argument-1 no* es un elemento de datos basado en el entorno local, el especificador de conversión basado en el entorno local (%p, en este caso) se trata como un especificador de conversión no basado en el entorno local y el% se sustituye por un @ (%p, en este caso, se convierte en @p y @p es el equivalente no basado en el entorno local de %p).
4. PICOSECONDS sólo se puede extraer de un elemento de indicación de fecha y hora.

Ejemplos

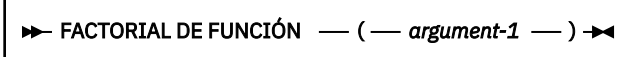
```
COMPUTE integer-1 = FUNCTION EXTRACT-DATE-TIME (date-3 MONTHS).
COMPUTE integer-1 = FUNCTION EXTRACT-DATE-TIME (date-3 '%m').
MOVE FUNCTION EXTRACT-DATE-TIME (date-2 '%m/%d') to alphanum-1.
```

FACTORIAL

La función FACTORIAL devuelve un entero que es el factorial del argumento especificado.

El tipo de función es entero.

Formato



argument-1
Si la opción de compilador ARITH (COMPAT) está en vigor, *argument-1* debe ser un entero mayor o igual que cero y menor o igual que 28. Si la opción de compilador ARITH (EXTEND) está en vigor, *argument-1* debe ser un entero mayor o igual que cero y menor o igual que 29.

Si el valor de *argument-1* es cero, se devuelve el valor 1; de lo contrario, se devuelve el factorial de *argument-1* .

BUSCAR DURACIÓN

La función FIND-DURATION devuelve un entero en forma de unidades completas de la duración especificada. Cualquier redondeo se realiza hacia abajo. El cálculo de duraciones incluye microsegundos.

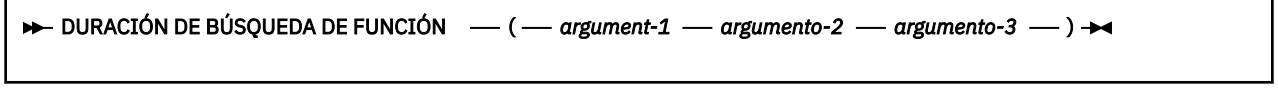
Utilice la función FIND-DURATION para calcular una duración entre:

- Dos fechas
- Una fecha y una indicación de fecha y hora
- Dos veces
- Una hora y una indicación de fecha y hora
- Dos indicaciones de fecha y hora

El tipo de función es entero.

El resultado de la función es un entero de nueve dígitos. Si el resultado de la función es mayor que 9 dígitos (999.999.999), se produce una comprobación de máquina.

Formato



argument-1, argument-2
Debe ser un elemento de fecha, hora o indicación de fecha y hora.

Argument-1 se resta del *argument-2*. El valor devuelto es el número de unidades completas de *argument-3*. Si *argument-1* es posterior a *argument-2*, el resultado es negativo. Si *argument-1* es anterior a *argument-2*, el resultado es positivo.

argument-3
Es una palabra clave que representa una duración. Las palabras clave de duración válidas son:

- AÑOS
- MESES
- DÍAS
- HORAS
- MINUTOS
- SEGUNDOS
- MICROSEGUNDOS
- PICOSEGUNDOS

Para determinar las palabras clave de duración válidas, se aplican las reglas siguientes:

1. Si *argument-1* o *argument-2* es un elemento de fecha, la duración especificada debe ser coherente con una fecha.
2. Si *argument-1* o *argument-2* es un elemento de tiempo, la duración especificada debe ser coherente con una hora.
3. Si el valor devuelto no es un entero, se trunca. Por ejemplo, la duración entre el 17 de marzo de 2020 y el 2 de mayo de 2020 es de 1,5 meses. Puesto que FIND-DURATION sólo devuelve un entero, el .5 se trunca y el valor real devuelto es 1.
4. Sólo puede solicitar la duración de PICOSECONDS cuando *argument-1* y *argument-2* son elementos de indicación de fecha y hora.

Ejemplos

Los ejemplos siguientes muestran cómo se puede utilizar la función intrínseca FIND-DURATION:

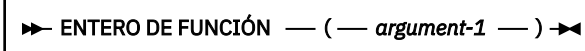
```
COMPUTE integer-1 = FUNCTION FIND-DURATION (date-3 date-4 MONTHS).
COMPUTE integer-1 = FUNCTION FIND-DURATION (timestamp-1 date-5 MONTHS).
```

Entero

La función INTEGER devuelve el valor entero más grande que es menor o igual que el argumento especificado.

El tipo de función es entero.

Formato



argument-1
Debe ser de clase numérica.

El valor devuelto es el mayor entero menor o igual que el valor de *argument-1*. Por ejemplo, FUNCTION INTEGER (2.5) devuelve un valor de 2 y FUNCTION INTEGER (-2.5) devuelve un valor de -3.

INTEGER-OF-DATE

La función INTEGER-OF-DATE convierte una fecha del calendario gregoriano del formato de fecha estándar (AAAAMMDD) al formato de fecha entero.

El tipo de función es entero.

El resultado de la función es un entero de siete dígitos con un rango de 1 a 3.067.671.

Formato

argument-1
Debe ser un entero con el formato AAAAMMDD, cuyo valor se obtiene a partir del cálculo (AAAA * 10.000) + (MM * 100) + DD, donde:

- AAAA representa el año en el calendario gregoriano. Debe ser un entero mayor que 1600, pero no mayor que 9999.
- MM representa un mes y debe ser un entero positivo menor que 13.
- DD representa un día y debe ser un entero positivo menor que 32, siempre que sea válido para la combinación de mes y año especificada.

El valor devuelto es un entero que es el número de días que la fecha representada por *argument-1* se ejecuta correctamente el 31 de diciembre de 1600 en el calendario gregoriano.

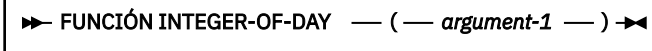
INTEGER-OF-DAY

La función INTEGER-OF-DAY convierte una fecha del calendario gregoriano del formato de fecha juliana (AAAADDD) al formato de fecha entera.

El tipo de función es entero.

El resultado de la función es un entero de siete dígitos.

Formato



argument-1
Debe ser un entero con el formato AAAADDD cuyo valor se obtiene del cálculo (AAAA * 1000) + DDD, donde:

- AAAA representa el año en el calendario gregoriano. Debe ser un entero mayor que 1600, pero no mayor que 9999.
- DDD representa el día del año. Debe ser un entero positivo menor que 367, siempre que sea válido para el año especificado.

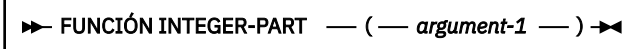
El valor devuelto es un entero que es el número de días que la fecha representada por *argument-1* se ejecuta correctamente el 31 de diciembre de 1600 en el calendario gregoriano.

ENTERO-PARTE

La función INTEGER-PART devuelve un entero que es la parte entera del argumento especificado.

El tipo de función es entero.

Formato



argument-1
Debe ser de clase numérica.

Si el valor de *argument-1* es cero, el valor devuelto es cero. Si el valor de *argument-1* es positivo, el valor devuelto es el mayor entero menor o igual que el valor de *argument-1*. Si el valor de *argument-1* es negativo, el valor devuelto es el menor entero mayor o igual que el valor de *argument-1*.

LONGITUD

La función LENGTH devuelve un entero igual a la longitud del argumento en posiciones de caracteres nacionales para argumentos de uso NATIONAL y en posiciones de caracteres alfanuméricos o bytes para todos los demás argumentos. Una posición de carácter alfanumérico y un byte son equivalentes.

El tipo de la función es entero.

Formato

argument-1
Puede ser:

- Un literal alfanumérico o un literal nacional
- Un elemento de datos de cualquier clase excepto DBCS
- Un elemento de datos descrito con el uso POINTER, PROCEDURE-POINTER, o FUNCTION-POINTER
- La DIRECCIÓN DEL registro especial
- El registro especial LENGTH OF
- El registro especial XML-NTEXT
- El registro especial XML-TEXT

El valor devuelto es un entero de 9 dígitos si se especifica la opción de compilador **ADDR(32)** o un entero de 18 dígitos si se especifica la opción de compilador **ADDR(64)** . El valor se determina de la forma siguiente:

- Si *argument-1* es un literal alfanumérico o un elemento de datos elemental de clase alfabético o alfanumérico, el valor devuelto es igual al número de posiciones de caracteres alfanuméricos en el argumento.

  Si *argument-1* es un literal alfanumérico terminado en nulo, el valor devuelto es igual al número de posiciones de caracteres alfanuméricos en el literal excluyendo el carácter nulo al final del literal.

  La longitud de un literal o elemento de datos alfanuméricos que contiene una mezcla de caracteres de un solo byte y de doble byte se cuenta como si cada byte fuera un carácter de un solo byte.

- Si *argument-1* es un elemento de grupo alfanumérico, el valor devuelto es igual a la longitud del *argument-1* en posiciones de caracteres alfanuméricos independientemente del contenido del grupo. Si algún elemento de datos subordinado a *argument-1* se describe con la frase DEPENDING de la cláusula OCCURS, la longitud del *argument-1* se determina utilizando el contenido del elemento de datos especificado en la frase DEPENDING. Esta evaluación se realiza de acuerdo con las reglas de la cláusula OCCURS para un elemento de datos de envío. Para obtener más información, consulte las discusiones de "cláusula OCCURS" en la página 184 y "cláusula USAGE" en la página 225.

  El valor devuelto incluye posiciones FILLER implícitas, si las hay.
- Si *argument-1* es un literal nacional o un elemento de datos elemental descrito con el uso NATIONAL, el valor devuelto es igual a la longitud del *argument-1* en posiciones de caracteres nacionales.

  Por ejemplo, si *argument-1* se define como PIC 9 (3) con uso NATIONAL, el valor devuelto es 3, aunque el tamaño de almacenamiento del argumento es de 6 bytes.
- Si *argument-1* es un elemento de grupo nacional, el valor devuelto es igual a la longitud de *argument-1* en posiciones de caracteres nacionales. Si algún elemento de datos subordinado a *argument-1* se describe con la frase DEPENDING de la cláusula OCCURS, la longitud del *argument-1* se determina utilizando el contenido del elemento de datos especificado en la frase DEPENDING. Esta evaluación se realiza de acuerdo con las reglas de la cláusula OCCURS para un elemento de datos de envío. Para obtener más información, consulte las discusiones de "cláusula OCCURS" en la página 184 y "cláusula USAGE" en la página 225.

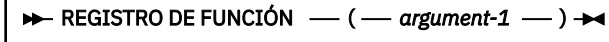
  El valor devuelto incluye posiciones FILLER implícitas, si las hay.
- De lo contrario, el valor devuelto es el número de bytes de almacenamiento ocupados por *argument-1*.

log

La función LOG devuelve un valor numérico que se aproxima al logaritmo en la base e (registro natural) del argumento especificado.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica. El valor de *argument-1* debe ser mayor que cero.

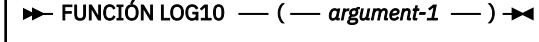
El valor devuelto es la aproximación del logaritmo a la base e del *argument-1*.

LOG10

La función LOG10 devuelve un valor numérico que se aproxima al logaritmo en la base 10 del argumento especificado.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica. El valor de *argument-1* debe ser mayor que cero.

El valor devuelto es la aproximación del logaritmo a la base 10 del *argument-1*.

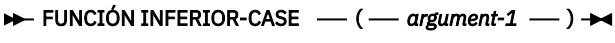
LOWER-CASE

La función LOWER-CASE devuelve una serie de caracteres que contiene los caracteres en el argumento con cada letra mayúscula sustituida por la letra minúscula correspondiente.

El tipo de función depende del tipo del argumento, como se indica a continuación:

Tipo de argumento	Tipo de función
Alfabético	Alfanumérico
Alfanumérico	Alfanumérico
Nacional	Nacional

Formato



argument-1
Debe ser de clase alfabética, alfanumérica, o nacional y debe tener al menos una posición de carácter de longitud.

Se devuelve la misma serie de caracteres que *argument-1* , excepto que cada letra mayúscula se sustituye por la letra minúscula correspondiente.

La conversión de caracteres de mayúsculas a minúsculas se basa en la especificación de atributos de caracteres en el entorno local de ejecución aplicable.

Para algunos entornos locales, la conversión de caracteres de mayúsculas a minúsculas puede dar como resultado una serie de caracteres con una longitud diferente a la longitud del *argument-1*. Esto puede ocurrir para todos los tipos de argumento posibles. Para los argumentos alfabéticos y alfanuméricos que constan únicamente de letras latinas en mayúsculas de la A a la Z, letras latinas en minúsculas de la a a la z y dígitos del 0 al 9, la longitud de la serie de caracteres devuelta es la misma que la longitud del *argument-1*.

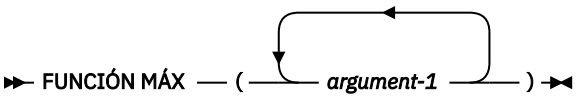
MÁX

La función MAX devuelve el contenido del argumento que contiene el valor máximo.

El tipo de función depende del tipo de argumento, tal como se indica a continuación:

Tipo de argumento	Tipo de función
Alfabético	Alfanumérico
Alfanumérico	Alfanumérico
Nacional	Nacional
Todos los argumentos enteros (incluye argumentos enteros de uso NATIONAL)	Entero
Numérico (algunos argumentos pueden ser enteros) (incluye argumentos numéricos de uso NATIONAL)	Numérico

Formato



argument-1
Debe ser alfabético de clase, alfanumérico, nacional o numérico.

Todos los argumentos deben ser de la misma clase, excepto que se permite una combinación de argumentos alfabéticos y alfanuméricos.

El valor devuelto es el contenido del *argument-1* que tiene el valor más alto. Las comparaciones utilizadas para determinar el valor más alto se realizan de acuerdo con las reglas para condiciones simples. Para obtener más información, consulte "Expresiones condicionales" en la página 250.

Si más de un *argument-1* tiene el mismo valor más alto, se devuelve el *argument-1* más a la izquierda que tiene ese valor.

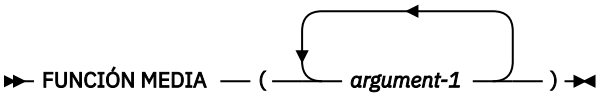
Si el tipo de la función es alfanumérico o nacional, el tamaño del valor devuelto es el tamaño del *argument-1*seleccionado.

MEAN

La función MEAN devuelve un valor numérico que se aproxima al promedio aritmético de sus argumentos.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

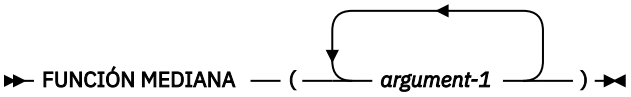
El valor devuelto es la media aritmética de la serie *argument-1* . El valor devuelto se define como la suma de la serie *argument-1* dividida por el número de apariciones referenciadas por *argument-1*.

MEDIO

La función MEDIAN devuelve el contenido del argumento cuyo valor es el valor medio de la lista formada organizando los argumentos en orden ordenado.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

El valor devuelto es el contenido del *argument-1* que tiene el valor medio en la lista formada por la organización de todos los valores *argument-1* en orden ordenado.

Si el número de apariciones referenciadas por *argument-1* es impar, el valor devuelto es tal que al menos la mitad de las apariciones a las que hace referencia el *argument-1* son mayores o iguales que el valor devuelto y al menos la mitad son menores o iguales. Si el número de apariciones referenciadas por *argument-1* es par, el valor devuelto es la media aritmética de los valores referenciados por las dos apariciones intermedias.

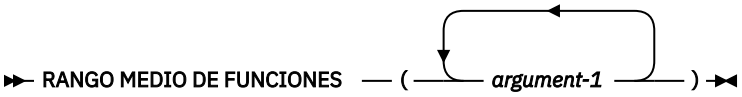
Las comparaciones utilizadas para organizar los valores de argumento en orden ordenado se realizan de acuerdo con las reglas para condiciones simples. Para obtener más información, consulte "Expresiones condicionales" en la página 250.

MIDRANGE

La función MIDRANGE devuelve un valor numérico que se aproxima al promedio aritmético de los valores del argumento mínimo y el argumento máximo.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

El valor devuelto es la media aritmética del valor del mayor *argument-1* y el valor del menor *argument-1*. Las comparaciones utilizadas para determinar los valores mayores y menores se realizan de acuerdo con las reglas para condiciones simples. Para obtener más información, consulte "Expresiones condicionales" en la página 250.

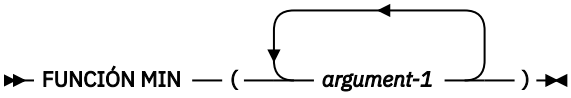
MIN

La función MIN devuelve el contenido del argumento que contiene el valor mínimo.

El tipo de función depende del tipo de argumento, tal como se indica a continuación:

Tipo de argumento	Tipo de función
Alfabético	Alfanumérico
Alfanumérico	Alfanumérico
Nacional	Nacional
Todos los argumentos enteros (incluye argumentos enteros de uso NATIONAL)	Entero
Numérico (algunos argumentos pueden ser enteros) (incluye argumentos numéricos de uso NATIONAL)	Numérico

Formato



argument-1
Debe ser alfabético de clase, alfanumérico, nacional o numérico.

Todos los argumentos deben ser de la misma clase, excepto que se permite una combinación de argumentos alfabéticos y alfanuméricos.

El valor devuelto es el contenido del *argument-1* que tiene el menor valor. Las comparaciones utilizadas para determinar el menor valor se realizan de acuerdo con las reglas para condiciones simples. Para obtener más información, consulte "Expresiones condicionales" en la página 250.

Si más de un *argument-1* tiene el mismo valor mínimo, se devuelve el *argument-1* situado más a la izquierda que tiene ese valor.

Si el tipo de la función es alfanumérico o nacional, el tamaño del valor devuelto es el tamaño del *argument-1*seleccionado.

MOD

La función MOD devuelve un valor entero que es *argument-1* módulo *argument-2*.

El tipo de función es entero.

El resultado de la función es un entero con tantos dígitos como el más corto de *argument-1* y *argument-2*.

Formato

argument-1
Debe ser un entero.

argument-2
Debe ser un entero. No debe ser cero.

El valor devuelto es *argument-1* módulo *argument-2*. El valor devuelto se define como:

argument-1 -(*argument-2* * FUNCTION INTEGER (*argument-1* / *argument-2*))

La siguiente tabla lista los resultados esperados de para algunos valores de *argument-1* y *argument-2*.

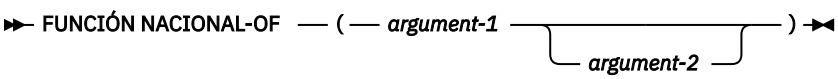
argument-1	***argument-2***	**Valor devuelto**
11	5	1
-11	5	4
11	-5	-4
-11	-5	-1

NACIONAL-DE

La función NATIONAL-OF devuelve una serie de caracteres nacionales que consta de la representación de caracteres nacionales de los caracteres en *argument-1*.

El tipo de la función es nacional.

Formato



argument-1
Debe ser de clase alfabética, alfanumérica, o DBCS. *argument-1* especifica la serie de origen para la conversión.

argument-2
Debe ser un entero o de clase alfanumérica. *argument-2* identifica la página de códigos fuente para la conversión.

Si *argument-2* es de clase alfanumérica, debe identificar un nombre de página de códigos primario o alias soportado por las bibliotecas de conversión ICU (consulte *International Components for Unicode: Converter Explorer*).

Si *argument-2* es un entero, el entero debe ser un número CCSID válido.

Si se omite *argument-2* , la página de códigos fuente se determina de la forma siguiente:

- Si *argument-1* es un elemento nativo (USAGE DISPLAY o USAGE DISPLAY-1 que contiene datos ASCII o ASCII DBCS, EUC o UTF-8), la página de códigos fuente se determina a partir del entorno local de tiempo de ejecución.
- Si *argument-1* es un elemento USAGE DISPLAY o USAGE DISPLAY-1 que contiene datos EBCDIC o EBCDIC DBCS, la página de códigos fuente se determina a partir de la variable de entorno EBCDIC_CODEPAGE, si se ha establecido. Si la variable de entorno EBCDIC_CODEPAGE no está establecida, la página de códigos fuente es la página de códigos predeterminada especificada en *Entornos locales y páginas de códigos soportadas* en *COBOL for Linux en x86 Guía de programación*.

El valor devuelto es una serie de caracteres nacionales que consta de los caracteres del *argument-1* convertidos a representación de caracteres nacionales. Cuando un carácter de origen no se puede convertir a un carácter nacional, el carácter de origen se convierte a un carácter de sustitución. El carácter de sustitución es:

- X'1A00' si se convierte un carácter de un solo byte
- X'FDFF' si se convierte un carácter de varios bytes

No se genera ninguna condición de excepción.

La longitud del valor devuelto depende del contenido del *argument-1* y de las características de la página de códigos fuente.

Notas de uso:

- El uso de nombres de página de códigos proporciona coherencia con otro software de Linux , pero el código fuente no es portable a Enterprise COBOL for z/OS.
- El CCSID para UTF-8 es 1208.
- El CCSID para UTF-16LE es 1200.

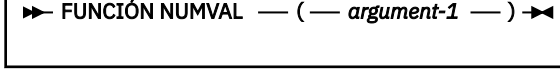
Excepción: si la conversión falla, se produce un error de tiempo de ejecución grave.

NUMVAL

La función NUMVAL devuelve el valor numérico representado por la serie de caracteres alfanuméricos o la serie de caracteres nacionales especificada como argumento. La función elimina los espacios iniciales o finales de la serie para producir un valor numérico.

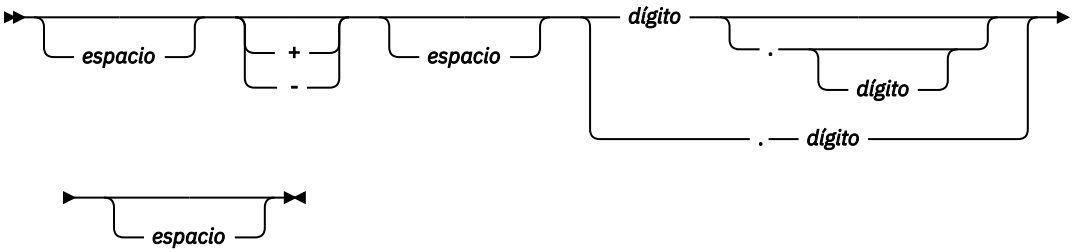
El tipo de función es numérico.

Formato

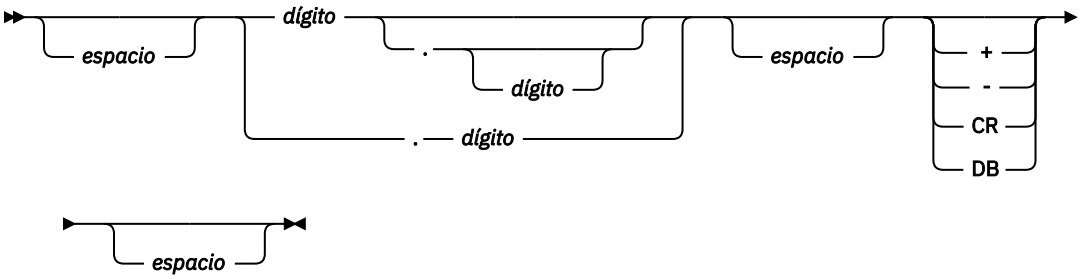


argument-1
Debe ser un literal alfanumérico, un literal nacional o un elemento de datos de clase nacional o alfanumérico de clase que contenga una serie de caracteres en cualquiera de los formatos siguientes:

Formato 1: argument-1



Formato 2: argument-1, formato monetario



espacio
Una serie de uno o más espacios.

dígito
Una serie de uno o más dígitos. Si la opción de compilador ARITH (COMPAT) está en vigor, el número total de dígitos no debe exceder de 18. Si la opción de compilador ARITH (EXTEND) está en vigor, el número total de dígitos no debe exceder de 31.

Si la cláusula DECIMAL-POINT IS COMA se especifica en el párrafo SPECIAL-NAMES, se debe utilizar una coma en *argument-1* en lugar de una coma decimal.

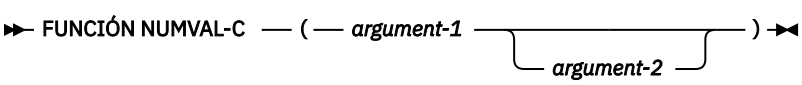
El valor devuelto es una aproximación de coma flotante del valor numérico representado por *argument-1*. La precisión del valor devuelto depende del valor de la opción de compilador ARITH. Para obtener más detalles, consulte *Conversión a números (NUMVAL, NUMVAL-C, NUMVAL-F)* en *COBOL for Linux en x86 Guía de programación*.

NUMVAL-C

La función NUMVAL-C devuelve el valor numérico representado por la serie de caracteres alfanuméricos o la serie de caracteres nacionales especificada como *argument-1*. La función elimina la serie de moneda, si la hay, y los separadores de agrupación (comas o puntos) para producir un valor numérico.

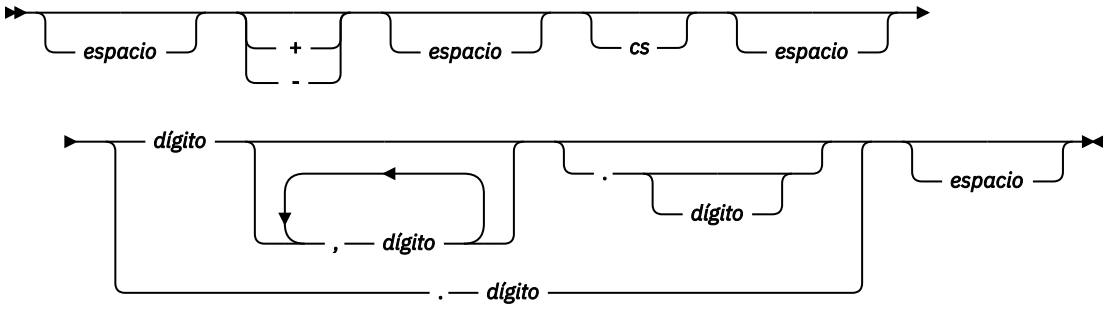
El tipo de función es numérico.

Formato

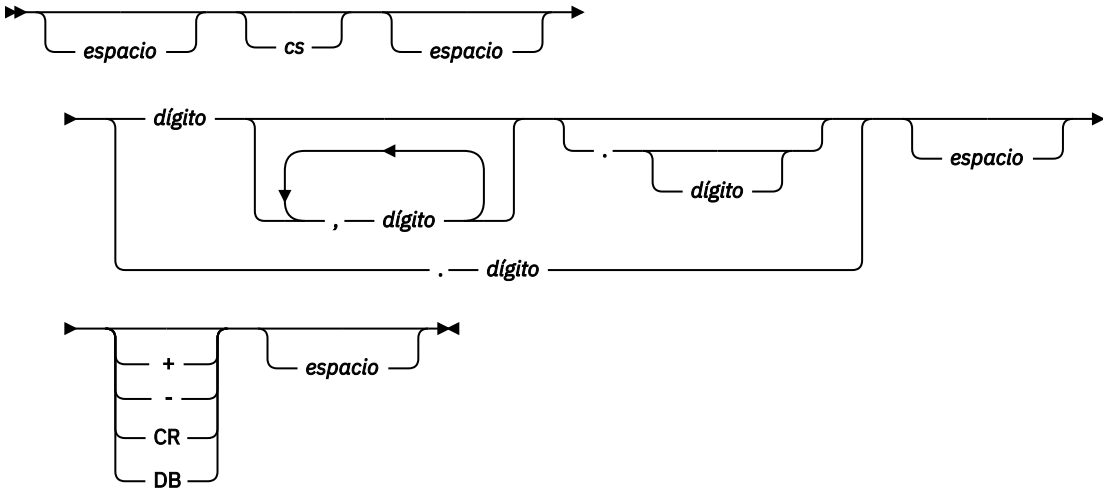


argument-1
Debe ser un literal alfanumérico, un literal nacional o un elemento de datos de clase alfanumérica o nacional de clase que contenga una serie de caracteres en cualquiera de los formatos siguientes:

Formato 1: argument-1



Formato 2: argument-1, formato monetario



espacio
Una serie de uno o más espacios.

cs
La serie de uno o más caracteres que forman el signo de moneda. Como máximo, puede aparecer una copia de los caracteres especificados por *cs* en el *argument-1*.

dígito
Una serie de uno o más dígitos. Si la opción de compilador ARITH (COMPAT) está en vigor, el número total de dígitos no debe exceder de 18. Si la opción de compilador ARITH (EXTEND) está en vigor, el número total de dígitos no debe exceder de 31.

Si la cláusula DECIMAL-POINT IS COMA se especifica en el párrafo SPECIAL-NAMES, las funciones de la coma y coma decimal en *argument-1* se invierten.

argument-2
Especifica el valor de serie de moneda.

Se aplican las reglas siguientes:

- *argument-2* debe especificarse si el programa contiene más de una cláusula CURRENCY SIGN.
- *argument-2*, si se especifica, debe ser de la misma clase que *argument-1*.

- *argument-2* no debe contener ninguno de los dígitos del 0 al 9, ningún espacio inicial o final, ni ninguno de los caracteres especiales '+', '-', '.' o ','.
- *argument-2* puede tener cualquier longitud válida para un elemento de datos elemental o de grupo de la clase de *argument-2*, incluido cero.
- La coincidencia de *argument-2* distingue entre mayúsculas y minúsculas. Por ejemplo, si especifica *argument-2* como 'CHF', no coincidirá con 'ChF', 'chf' o 'chF'.

Si no se especifica *argument-2* , el carácter utilizado para *cs* es el símbolo de moneda especificado para el programa.

El valor devuelto es una aproximación de coma flotante del valor numérico representado por *argument-1*. La precisión del valor devuelto depende del valor de la opción de compilador ARITH. Para obtener más detalles, consulte *Conversión a números (NUMVAL, NUMVAL-C, NUMVAL-F)* en *COBOL for Linux en x86 Guía de programación*.

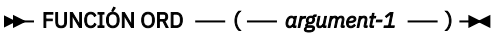
VOR

La función ORD devuelve un valor entero que es la posición ordinal de su argumento en el orden de clasificación del programa. La posición ordinal más baja es 1.

El tipo de función es entero.

El resultado de la función es un entero de tres dígitos.

Formato



argument-1
Debe tener un carácter de longitud y debe ser de clase alfabética o alfanumérica.

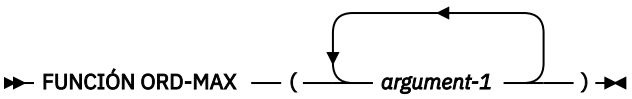
El valor devuelto es la posición ordinal de *argument-1* en el orden de clasificación del programa; oscila entre 1 y 256 en función del orden de clasificación.

ORD-MAX

La función ORD-MAX devuelve un valor que es la posición ordinal en la lista de argumentos del argumento que contiene el valor máximo.

El tipo de función es entero.

Formato



argument-1
Debe ser alfabético de clase, alfanumérico, nacional o numérico.

Todos los argumentos deben ser de la misma clase, excepto que se permite una combinación de argumentos alfabéticos y alfanuméricos.

El valor devuelto es el número ordinal que corresponde a la posición del *argument-1* que tiene el valor más alto en la serie *argument-1* .

Las comparaciones utilizadas para determinar el *argument-1* de mayor valor se realizan de acuerdo con las reglas para condiciones simples. Para obtener más información, consulte “Expresiones condicionales” en la página 250.

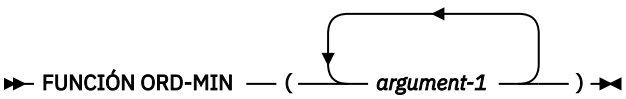
Si más de un *argument-1* tiene el mismo valor más alto, el número devuelto corresponde a la posición del *argument-1* más a la izquierda que tiene ese valor.

SEÑOR-MIN

La función ORD-MIN devuelve un valor que es la posición ordinal en la lista de argumentos del argumento que contiene el valor mínimo.

El tipo de función es entero.

Formato



argument-1
Debe ser alfabético de clase, alfanumérico, nacional o numérico.

Todos los argumentos deben ser de la misma clase, excepto que se permite una combinación de argumentos alfabéticos y alfanuméricos.

El valor devuelto es el número ordinal que corresponde a la posición del *argument-1* que menos valor tiene en la serie *argument-1* .

Las comparaciones utilizadas para determinar el *argument-1* menos valorado se realizan de acuerdo con las reglas para condiciones simples. Para obtener más información, consulte "Expresiones condicionales" en la página 250.

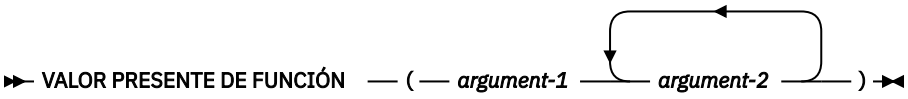
Si más de un *argument-1* tiene el mismo valor mínimo, el número devuelto corresponde a la posición del *argument-1* más a la izquierda que tiene ese valor.

VALOR ACTUAL

La función PRESENT-VALUE devuelve un valor que se aproxima al valor actual de una serie de importes finales de periodo futuros especificados por *argument-2* a una tasa de descuento especificada por *argument-1*.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica. Debe ser mayor que -1.

argument-2
Debe ser de clase numérica.

El valor devuelto es una aproximación de la suma de una serie de cálculos con cada término en el formato siguiente:

argument-2 /(1 + *argument-1*) ** *n*

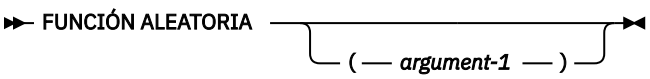
Hay un término para cada aparición del *argument-2*. El exponente *n* se incrementa de 1 en 1 para cada término de la serie.

ALEATORIO

La función RANDOM devuelve un valor numérico que es un número pseudoaleatorio de una distribución rectangular.

El tipo de función es numérico.

Formato



argument-1
Si se especifica *argument-1* , debe ser cero o un entero positivo. Sin embargo, sólo los valores en el rango de cero hasta e incluyendo 2.147.483.645 producen una secuencia distinta de números pseudoaleatorios.

Si una referencia posterior especifica *argument-1*, se inicia una nueva secuencia de números pseudoaleatorios.

Si la primera referencia a esta función en la unidad de ejecución no especifica *argument-1*, el valor de semilla utilizado será cero.

En cada caso, las referencias posteriores sin especificar *argument-1* devuelven el siguiente número en la secuencia actual.

El valor devuelto está exclusivamente entre cero y uno.

Para un valor de semilla determinado, la secuencia de números pseudoaleatorios es siempre la misma.

La función RANDOM se puede utilizar en programas con hebras. Para una semilla inicial, se devuelve una única secuencia de números pseudoaleatorios, independientemente de la hebra que se esté ejecutando cuando se invoca RANDOM.

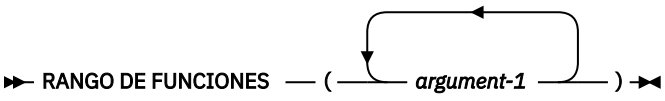
RANGO

La función RANGE devuelve un valor que es igual al valor del argumento máximo menos el valor del argumento mínimo.

El tipo de función depende de los tipos de argumento, como se indica a continuación:

Tipo de argumento	Tipo de función
Todos los argumentos enteros	Entero
Numérico (algunos argumentos pueden ser enteros)	Numérico

Formato



argument-1
Debe ser de clase numérica.

El valor devuelto es igual a *argument-1* con el mayor valor menos el *argument-1* con el menor valor. Las comparaciones utilizadas para determinar los valores mayores y menores se realizan de acuerdo con las

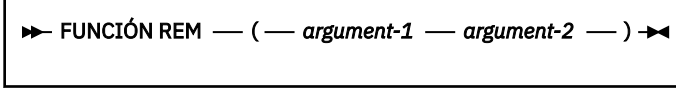
reglas para condiciones simples. Para obtener más información, consulte “Expresiones condicionales” en la página 250.

REM

La función REM devuelve un valor numérico que es el resto del *argument-1* dividido por *argument-2*.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

argument-2
Debe ser de clase numérica. No debe ser cero.

El valor devuelto es el resto del *argument-1* dividido por *argument-2*. Se define como la expresión:

argument-1 -(*argument-2* * FUNCTION INTEGER-PART (*argument-1* / *argument-2*))

INVERTIR

La función REVERSE devuelve una serie de caracteres de exactamente la misma longitud que el argumento, cuyos caracteres son exactamente los mismos que los especificados en el argumento, excepto que están en orden inverso. Para los argumentos de tipo nacional, las posiciones de caracteres se invierten.

El tipo de función depende del tipo del argumento, como se indica a continuación:

Tipo de argumento	Tipo de función
Alfabético	Alfanumérico
Alfanumérico	Alfanumérico
Nacional	Nacional

Formato

argument-1
Debe ser una clase alfabética, alfanumérica o nacional y debe tener al menos un carácter de longitud.

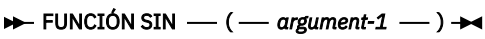
El valor devuelto es una serie de caracteres de la misma longitud que *argument-1*, con los caracteres de *argument-1* en orden inverso. Por ejemplo, si *argument-1* contiene ABC, el valor devuelto es CBA.

SIN

La función SIN devuelve un valor numérico que se aproxima al seno del ángulo o arco especificado por el argumento en radianes.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

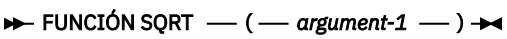
El valor devuelto es la aproximación del seno del *argument-1* y es mayor o igual que -1 y menor o igual que +1.

SQRT

La función SQRT devuelve un valor numérico que se aproxima a la raíz cuadrada del argumento especificado.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica. El valor del *argument-1* debe ser cero o positivo.

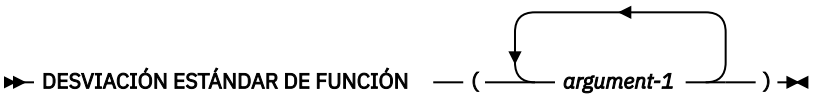
El valor devuelto es el valor absoluto de la aproximación de la raíz cuadrada del *argument-1*.

DESVIACIÓN TÍPICA

La función STANDARD-DESVIACIÓN devuelve un valor numérico que se aproxima a la desviación estándar de sus argumentos.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

El valor devuelto es la aproximación de la desviación estándar de la serie *argument-1* . El valor devuelto se calcula de la siguiente manera:

1. Se calcula y cuadra la diferencia entre cada *argument-1* y la media aritmética de la serie *argument-1* .
2. A continuación, se suman los valores obtenidos. Esta cantidad se divide por el número de valores de la serie *argument-1* .
3. A continuación, se calcula la raíz cuadrada del cociente obtenido. El valor devuelto es el valor absoluto de esta raíz cuadrada.

Si la serie *argument-1* consta de un solo valor, o si la serie *argument-1* consta de todos los elementos de datos de aparición de variables y el número total de apariciones para todos ellos es uno, el valor devuelto es cero.

RESTAR DURACIÓN

La función RESTTRACT-DURATION resta una duración de un elemento de fecha, hora o indicación de fecha y hora y devuelve el elemento modificado.

El tipo de función es fecha-hora.

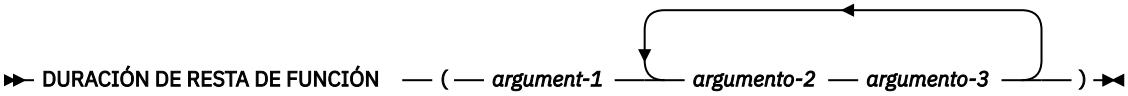
La longitud del valor de retorno depende de la longitud del elemento de fecha, hora o indicación de fecha y hora especificado en *argument-1*. El valor devuelto se truncará a la longitud de *argument-1*.

Si se resta una duración de un elemento de fecha, la fecha devuelta debe estar dentro de un rango determinado:

- Para las fechas de 4 dígitos, el rango debe ser de 0001/01/01 a 9999/12/31.
- Para fechas de 2 dígitos, el rango debe ser de 0001/01/01 a 9999/12/31, pero el año se trunca a 2 dígitos.
- Para un año de 3 dígitos (un siglo de 1 dígito y un año de 2 dígitos), el rango debe ser de 1900/01/01 a 2899/12/31 (el valor predeterminado). Este rango se puede cambiar especificando la opción de compilador DATETIME .

Si una duración se resta de un elemento de fecha de 2 dígitos, el rango es el mismo que para un año de 4 dígitos, pero el año del valor devuelto se trunca a 2 dígitos.

Formato



argument-1
Debe ser un elemento de fecha, hora o indicación de fecha y hora.

argument-1 es el valor del que se resta una duración. La duración se especifica en *argument-2* y *argument-3*.

argument-2
argument-2 es una palabra clave que representa una duración. Las duraciones válidas son:

- AÑOS
- MESES
- DÍAS
- HORAS
- MINUTOS
- SEGUNDOS
- MICROSEGUNDOS
- PICOSEGUNDOS

La palabra clave de duración o el especificador de conversión utilizados deben ser coherentes con el *argument-1*. Por ejemplo, las palabras clave de duración deben cumplir las reglas siguientes:

1. YEARS, MONTHS y DAYS solo se pueden restar de un elemento de fecha o indicación de fecha y hora.
2. HOURS, MINUTES, SECONDS y MICROSECONDS solo se pueden restar de un elemento de indicación de fecha y hora o de hora.
3. PICOSECONDS sólo se puede restar de un elemento de indicación de fecha y hora.

argument-3
Debe ser una expresión aritmética entera. *argument-3* es el número de unidades de la duración, tal como se especifica en *argument-2*, que se van a restar del *argument-1*.

argument-2 y *argument-3* se pueden repetir. No debe haber ningún *argument-2* duplicado en una función intrínseca.

argument-3 puede ser un entero negativo, pero la función sólo toma su valor absoluto. Si *argument-3* tiene más de 9 dígitos, se trunca.

Si una duración se resta de una fecha y el resultado no es válido, la fecha se ajusta. Por ejemplo, si se resta una duración de 1 mes de la fecha `May 31, 2020`, el resultado sería la fecha no válida `April 31, 2020`. Esta fecha se ajustará a la fecha válida `April 30, 2020`.

Ejemplos

Los ejemplos siguientes muestran cómo se puede utilizar la función intrínseca RESTTRACT-DURATION:

```
MOVE FUNCTION SUBTRACT-DURATION (date-1 MONTHS 1)
     TO date-2.
MOVE FUNCTION SUBTRACT-DURATION (date-2 MONTHS 1 + 2 * 3)
     TO date-1.
MOVE FUNCTION SUBTRACT-DURATION (date-3 MONTHS 5 DAYS 1000)
     TO date-1.
```

Referencias relacionadas
"AÑADIR DURACIÓN" en la página 463

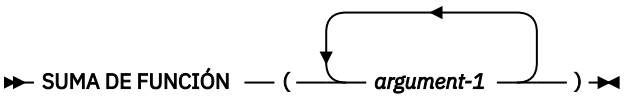
SUM

La función SUM devuelve un valor que es la suma de los argumentos.

El tipo de función depende de los tipos de argumento, como se indica a continuación:

Tipo de argumento	Tipo de función
Todos los argumentos enteros	Entero
Numérico (algunos argumentos pueden ser enteros)	Numérico

Formato



argument-1
Debe ser de clase numérica.

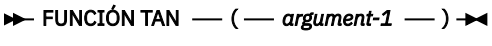
El valor devuelto es la suma de los argumentos. Si la serie *argument-1* son todos enteros, el valor devuelto es un entero. Si la serie *argument-1* no son todos enteros, se devuelve un valor numérico.

TAN

La función TAN devuelve un valor numérico que se aproxima a la tangente del ángulo o arco especificado por el argumento en radianes.

El tipo de función es numérico.

Formato



argument-1
Debe ser de clase numérica.

El valor devuelto es la aproximación de la tangente del *argument-1*.

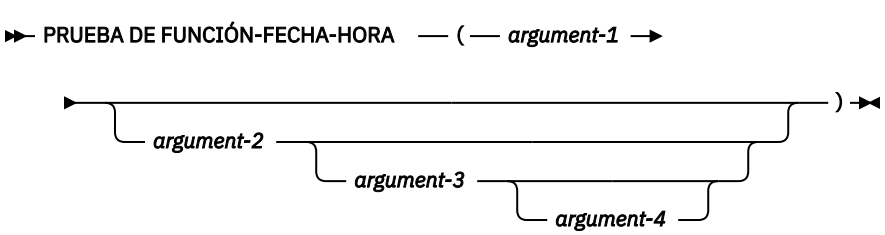
TEST-FECHA-HORA

La función TEST-DATE-TIME toma un elemento de fecha, hora, indicación de fecha y hora, alfanumérico, empaquetado numérico o con zona. La función determina si es una fecha, hora o indicación de fecha y hora válida. Devuelve true (B '1') si es un elemento válido o false (B '0') si no es un elemento válido.

El tipo de función es booleano.

La longitud del valor devuelto es de 1 byte.

Formato



argument-1
Un *argument-1* válido puede ser:

- Un elemento de fecha, hora o indicación de fecha y hora
- Un elemento de clase alfanumérico
- Un literal no numérico
- Un elemento de entero numérico de clase

Si *argument-1* es un elemento de fecha, hora o indicación de fecha y hora, *argument-2* a *argument-4* son opcionales. Si *argument-1* no es un elemento de fecha, hora o indicación de fecha y hora, Se debe especificar *argument-2* (*argument-3* y *argument-4* son opcionales).

Argument-1 se prueba para ver si es un elemento válido, en función de su tipo o del tipo de *argument-2* a *argument-4*.

argument-2
Un *argument-2* válido puede ser:

- fecha
- hora
- INDICACIÓN de fecha y hora

Si *argument-2* es TIMESTAMP, *argument-3* sólo se puede especificar con el registro especial FORMAT OF, y *argument-4* no se pueden especificar.

argument-3
Especifica el formato de un elemento de fecha u hora. Debe ser:

- Un literal no numérico de al menos 2 caracteres de longitud
- La palabra clave LOCALE
- El registro especial FORMAT OF

Si *argument-3* es la palabra clave LOCALE, el formato de la fecha u hora se basa en un LOCALE. Si no se especifica *argument-4* , se utiliza el entorno local actual; de lo contrario, se utiliza el entorno local asociado con el nombre mnemónico o el registro especial LOCALE OF.

Si no se especifica *argument-3* , el formato utilizado para la prueba es el definido en la cláusula SPECIAL-NAMES FORMAT. Para TIMESTAMP, si no se especifica *argument-3* , se utiliza el formato predeterminado de @Y-%m-%d-%H.%M.%S.@Sm .

argument-4

Debe ser un nombre mnemotécnico asociado con un LOCALE, o el registro especial LOCALE OF.

Argument-4 debe seguir estas reglas:

- Si se especifica *argument-4* y *argument-3* es un literal de formato basado en el entorno local, por ejemplo, contiene %p, entonces el literal de formato basado en el entorno local utilizaría el entorno local especificado en *argument-4* para determinar el valor real de los especificadores de conversión.
- Si *argument-3* es un literal de formato basado en el entorno local, por ejemplo, contiene %p y *argument-4 no se especifica* , el literal de formato basado en el entorno local utilizaría el entorno local actual para determinar el valor real de los especificadores de conversión.
- Si *argument-3* es un literal de formato basado en el entorno local, por ejemplo, contiene %p, y el registro especial LOCALE OF se utiliza para hacer referencia a un elemento que no es del entorno local, el literal de formato basado en el entorno local utiliza el entorno local predeterminado para determinar el valor real de los especificadores de conversión.

Ejemplos:

Los ejemplos siguientes muestran cómo se puede utilizar la función intrínseca TEST-DATE-TIME:

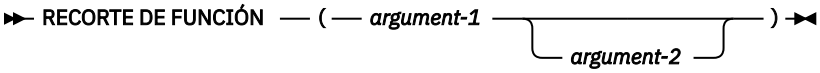
```
WORKING-STORAGE SECTION.
01 mydate1 PIC X(8) VALUE '07312013'.

PROCEDURE DIVISION.

IF FUNCTION TEST-DATE-TIME (mydate1 DATE '%d%m%Y') = B'1'
   DISPLAY 'Valid Date'
END-IF.
```

TRIM

La función TRIM devuelve la serie dada con los negros iniciales y finales que se han eliminado, o la serie dada con los caracteres especificados iniciales y finales eliminados.

El tipo de la función es alfanumérico, DBCS o nacional en función de la clase de su argumento.

Formato



argument-1

Debe ser un literal no numérico o un elemento de datos de clase alfabético, alfanumérico, DBCS o nacional. *Argument-1* identifica la serie de origen para el recorte.

argument-2

Si se especifica, debe ser un literal no numérico o un elemento de datos de la misma clase que *argument-1*. Especifica los caracteres que se deben recortar. Si no se especifica, el carácter de recorte toma el valor por omisión de blanco.

Si no se especifica *argument-2* , el valor devuelto es una serie de caracteres alfanuméricos, DBCS o nacional que consta de los caracteres del *argument-1* con los espacios en blanco iniciales y finales eliminados. El carácter en blanco es un carácter de espacio de 1 byte (" o X'20') cuando *argument-1* es de clase alfanumérica, o un espacio de doble byte (X'2020') cuando *argument-1* es de clase DBCS, o un espacio nacional (X'2000 'o X'0030') cuando *argument-1* es de clase nacional.

Si se especifica *argument-2* , todos los caracteres del *argument-2* se recortan de ambos extremos de la serie. El valor devuelto es una serie de caracteres alfanuméricos, DBCS o nacionales que consta de los caracteres del *argument-1* con los caracteres iniciales y finales especificados en el *argument-2* eliminado.

La longitud de la serie devuelta depende del contenido y de la clase de *argument-1*. Es la longitud de la serie devuelta en número de posiciones de caracteres. Si *argument-1* es un elemento de datos DBCS o nacional, la longitud está en posiciones de caracteres DBCS o nacionales.

Valores devueltos

El orden de los caracteres en el parámetro *argument-2* no afecta al resultado de la operación. Los caracteres son una lista de caracteres únicos. Por ejemplo, `FUNCTION TRIML (fld, "abc")` devuelve la subserie de fld que empieza por cualquier carácter que no sea 'a', 'b' o 'c'. Si fld contiene "caxyz", `FUNCTION TRIM (fld, "abc")` devuelve "xyz". Los caracteres pueden aparecer dos veces en el segundo parámetro sin errores. Por ejemplo, `FUNCTION TRIM (fld, "aba")` es válido. Esto significa lo mismo que `FUNCTION TRIM (fld, "ab")`.

Si se especifica el segundo parámetro de FUNCTION TRIM, TRIML o TRIMR, los espacios en blanco no se recortan a menos que aparezca un espacio en blanco como parte del *argument-2*. Las funciones TRIM, TRIML y TRIMR no son sensibles a series SBCS/DBCS mixtas. Tanto *argument-1* como *argument-2* se tratan como SBCS si su clase es alfanumérica.

Ejemplos:

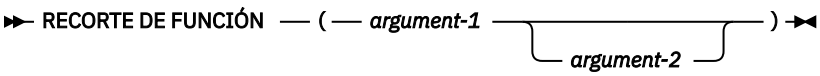
```
FUNCTION TRIM("xxxABxCxxx", "x")             // returns 'ABxC'
FUNCTION TRIMR(">>>>ABC<<<<<", "<>")         // returns '>>>>ABC'
MOVE "xyz" TO tc.
FUNCTION TRIML("xxyyzzyyzzABCxyzyxzxy", tc)  // returns 'ABCxyzyxzxy'
```

TRIML

La función TRIML devuelve la serie dada con los negros iniciales eliminados, o la serie dada con los caracteres iniciales especificados eliminados.

El tipo de la función es alfanumérico, DBCS o nacional en función de la clase de su argumento.

Formato



argument-1
Debe ser un literal no numérico o un elemento de datos de clase alfabético, alfanumérico, DBCS o nacional. *Argument-1* identifica la serie de origen para el recorte.

argument-2
Si se especifica, debe ser un literal no numérico o un elemento de datos de la misma clase que *argument-1*. Especifica los caracteres que se deben recortar. Si no se especifica, el carácter de recorte toma el valor por omisión de blanco.

Si no se especifica *argument-2* , el valor devuelto es una serie de caracteres alfanuméricos, DBCS o nacional que consta de los caracteres del *argument-1* con los blancos iniciales eliminados. El carácter en blanco es un carácter de espacio de 1 byte (' ' o X'40 ') cuando *argument-1* es de clase alfanumérica, o un espacio de doble byte (X'4040') cuando *argument-1* es de clase DBCS, o un espacio nacional (X'0020 'o X'3000') cuando *argument-1* es de clase nacional.

Si no se especifica *argument-2* , el valor devuelto es una serie de caracteres alfanuméricos, DBCS o nacional que consta de los caracteres del *argument-1* con los blancos iniciales eliminados. El

carácter en blanco es un carácter de espacio de 1 byte ('' o X'40 ') cuando *argument-1* es de clase alfanumérica, o un espacio de doble byte (X'4040') cuando *argument-1* es de clase DBCS, o un espacio nacional (X'0020 'o X'3000') cuando *argument-1* es de clase nacional.

Si se especifica *argument-2* , el valor devuelto es un valor alfanumérico, DBCS, o serie de caracteres nacionales que consta de los caracteres del *argument-1* con los caracteres iniciales especificados en el *argument-2* eliminado.

La longitud de la serie devuelta depende del contenido y de la clase de *argument-1*. Es la longitud de la serie devuelta en número de posiciones de caracteres. Si *argument-1* es un elemento de datos DBCS o nacional, la longitud está en posiciones de caracteres DBCS o nacionales.

Para obtener más información sobre los valores devueltos y ejemplos, consulte "TRIM" en la página 495.

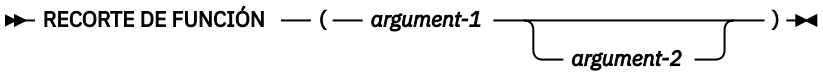
Referencias relacionadas
"TRIM" en la página 495

TRIMR

La función TRIMR devuelve la serie dada con los negros de cola eliminados, o la serie dada con los caracteres especificados de cola eliminados.

El tipo de la función es alfanumérico, DBCS o nacional en función de la clase de su argumento.

Formato



argument-1
Debe ser un literal no numérico o un elemento de datos de clase alfabético, alfanumérico, DBCS o nacional. *Argument-1* identifica la serie de origen para el recorte.

argument-2
Si se especifica, debe ser un literal no numérico o un elemento de datos de la misma clase que *argument-1*. Especifica los caracteres que se deben recortar. Si no se especifica, el carácter de recorte toma el valor por omisión de blanco.

Si no se especifica *argument-2* , el valor devuelto es una serie de caracteres alfanuméricos, DBCS o nacional que consta de los caracteres del *argument-1* con los blancos iniciales eliminados. El carácter en blanco es un carácter de espacio de 1 byte ('' o X'40 ') cuando *argument-1* es de clase alfanumérica, o un espacio de doble byte (X'4040') cuando *argument-1* es de clase DBCS, o un espacio nacional (X'0020 'o X'3000') cuando *argument-1* es de clase nacional.

Si se especifica *argument-2* , el valor devuelto es un valor alfanumérico, DBCS, o serie de caracteres nacionales que consta de los caracteres del *argument-1* con cualquier carácter final especificado en el *argument-2* eliminado.

La longitud de la serie devuelta depende del contenido y de la clase de *argument-1*. Es la longitud de la serie devuelta en número de posiciones de caracteres. Si *argument-1* es un elemento de datos DBCS o nacional, la longitud está en posiciones de caracteres DBCS o nacionales.

Para obtener más información sobre los valores devueltos y ejemplos, consulte "TRIM" en la página 495.

Referencias relacionadas
"TRIM" en la página 495

FECHA

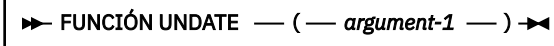
La función UNDATE convierte un campo de fecha en un campo no de fecha para un uso no ambiguo con no fechas.

Si la opción de compilador NODATEPROC está en vigor, la función UNDATE no tiene ningún efecto.

El tipo de función depende del tipo de *argumento-1*:

Tipo de argumento	Tipo de función
Alfanumérico	Alfanumérico
Entero	Entero

Formato



argument-1
Un campo de fecha.

El valor devuelto es un valor nondate que contiene el valor de *argument-1* sin modificar.

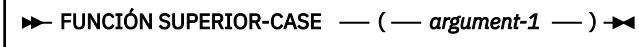
UPPER-CASE

La función UPPER-CASE devuelve una serie de caracteres que contiene los caracteres del argumento con cada letra minúscula sustituida por la letra mayúscula correspondiente.

El tipo de función depende del tipo del argumento, como se indica a continuación:

Tipo de argumento	Tipo de función
Alfabético	Alfanumérico
Alfanumérico	Alfanumérico
Nacional	Nacional

Formato



argument-1
Debe ser de clase alfabética, alfanumérica, o nacional y debe tener al menos una posición de carácter de longitud.

Se devuelve la misma serie de caracteres que *argument-1* , excepto que cada letra minúscula se sustituye por la letra mayúscula correspondiente.

La conversión de caracteres de minúsculas a mayúsculas se basa en la especificación de atributos de caracteres en el entorno local de ejecución aplicable.

Para algunos entornos locales, la conversión de caracteres de minúsculas a mayúsculas puede dar como resultado una serie de caracteres con una longitud diferente a la longitud del *argument-1*. Esto puede ocurrir para todos los tipos de argumentos posibles. Para los argumentos alfabéticos y alfanuméricos que constan únicamente de letras latinas en mayúsculas de la A a la Z, letras latinas en minúsculas de la a a la z y dígitos del 0 al 9, la longitud de la serie de caracteres devuelta es la misma que la longitud del *argument-1*.

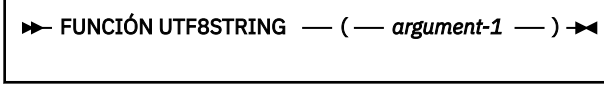
UTF8STRING

La función UTF8STRING convierte el argumento especificado en la serie UTF-8 correspondiente. La serie devuelta tiene una longitud variable. Se recomienda a los usuarios que permitan una longitud suficiente

para el argumento de recepción devuelto por esta función. La longitud máxima devuelva es el doble de la longitud del argumento original.

El tipo de función es alfanumérico.

Formato



argument-1

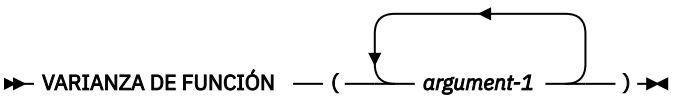
Debe ser alfabético, alfanumérico, DBCS o nacional, y debe tener al menos un carácter de longitud.

VARIANCE

La función VARIANCE devuelve un valor numérico que se aproxima a la varianza de sus argumentos.

El tipo de función es numérico.

Formato



argument-1

Debe ser de clase numérica.

El valor devuelto es la aproximación de la varianza de la serie *argument-1* .

El valor devuelto se define como el cuadrado de la desviación estándar de la serie *argument-1* . Este valor se calcula de la siguiente manera:

1. Se calcula y cuadra la diferencia entre cada valor de *argument-1* y la media aritmética de la serie *argument-1* .
2. A continuación, se suman los valores obtenidos. Esta cantidad se divide por el número de valores de la serie de argumentos.

Si la serie *argument-1* consta de un solo valor, o si la serie *argument-1* consta de todos los elementos de datos de aparición de variables y el número total de apariciones para todos ellos es uno, el valor devuelto es cero.

WHEN-COMPILADO

La función WHEN-COMPILE devuelve la fecha y hora en que se ha compilado el programa tal como lo ha proporcionado el sistema en el que se ha compilado el programa.

El tipo de función es alfanumérico.

Formato

Leyendo de izquierda a derecha, las posiciones de 21 caracteres del valor devuelto son las siguientes:

Carácter posiciones	Contenido
1-4	Cuatro dígitos numéricos del año en el calendario gregoriano
5-6	Dos dígitos numéricos del mes del año, en el rango de 01 a 12
7-8	Dos dígitos numéricos del día del mes, en el rango de 01 a 31
9-10	Dos dígitos numéricos de las horas posteriores a la medianoche, en el rango de 00 a 23
11-12	Dos dígitos numéricos de los minutos después de la hora, en el rango de 00 a 59
13-14	Dos dígitos numéricos de los segundos después del minuto, en el rango de 00 a 59
15-16	Dos dígitos numéricos de las centésimas de segundo después del segundo, en el rango de 00 a 99. El valor 00 se devuelve si el sistema en el que se evalúa la función no tiene el recurso para proporcionar la parte fraccional de un segundo.
17	El carácter '-' o el carácter '+'. Se devuelve el carácter '-' si la hora local indicada en las posiciones de caracteres anteriores está detrás de la hora media de Greenwich. El carácter '+' se devuelve si la hora local indicada es igual o anterior a la hora media de Greenwich. El carácter '0' se devuelve si el sistema en el que se evalúa esta función no tiene el recurso para proporcionar el factor diferencial de hora local.
18-19	Si la posición 17 del carácter es '-', se devuelven dos dígitos numéricos en el rango de 00 a 12 que indican el número de horas que la hora notificada está por detrás de la hora media de Greenwich. Si la posición 17 del carácter es '+', se devuelven dos dígitos numéricos en el rango de 00 a 13 que indican el número de horas que la hora notificada está por delante de la hora media de Greenwich. Si la posición de carácter 17 es '0', se devuelve el valor 00.
20-21	Se devuelven dos dígitos numéricos en el rango de 00 a 59 que indican el número de minutos adicionales que la hora notificada está por delante o por detrás de la hora media de Greenwich, dependiendo de si la posición 17 del carácter es '+' o '-', respectivamente. Si la posición de carácter 17 es '0', se devuelve el valor 00.

El valor devuelto es la fecha y hora de compilación de la unidad de origen que contiene esta función. Si un programa es un programa contenido, el valor devuelto es la fecha y hora de compilación asociada con el programa contenedor.

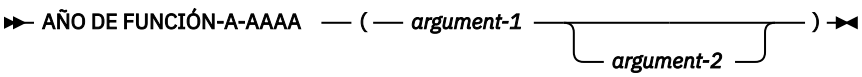
AÑO-A-AAAA

La función YEAR-TO-YYYY convierte *argument-1*, un año de dos dígitos, en un año de cuatro dígitos. *argument-2*, cuando se añade al año en el momento de la ejecución, define el año final de un intervalo de 100 años, o ventana de siglo deslizante, en el que cae el año del *argument-1* .

El tipo de función es entero.

Si la opción de compilador DATEPROC está en vigor, el valor devuelto es un campo de fecha expandido con DATE FORMAT YYYY implícito.

Formato



argument-1
Debe ser un entero no negativo que sea menor que 100.

argument-2
Debe ser un entero. Si se omite *argument-2* , la función se evalúa suponiendo que se haya especificado el valor 50.

La suma del año en el momento de la ejecución y el valor de *argument-2* debe ser menor que 10.000 y mayor que 1.699.

En la tabla siguiente se muestran ejemplos de valores de retorno de la función YEAR-TO-YYYY.

Año actual	**Valor *argument-1***	**Valor *argument-2***	**Valor devuelto**
1995	4	23	2004
1995	4	-15	1904
2008	98	23	1998
2008	98	-15	1898

VENTANA

Si la opción de compilador DATEPROC está en vigor, la función YEARWINDOW devuelve el año inicial de la ventana de siglo especificada por la opción de compilador YEARWINDOW.

El valor devuelto es un campo de fecha expandido con DATE FORMAT YYYY implícito.

Si la opción de compilador NODATEPROC está en vigor, la función YEARWINDOW devuelve 0.

El tipo de función es entero.

Formato

Parte 8. Sentencias de direccionamiento de compilador y directivas de compilador

Capítulo 21. Sentencias de direccionamiento de compilador

Una *sentencia de direccionamiento de compilador* es una sentencia que hace que el compilador realice una acción específica durante la compilación.

Puede utilizar sentencias de direccionamiento de compilador para los siguientes propósitos:

- Control de biblioteca de origen ampliado (sentencias BASIS, DELETE e INSERT)
- Manipulación de texto de origen (sentencias COPY y REPLACE)
- Manejo de excepciones (sentencia USE)
- Control de listados de compilador (sentencias *CONTROL, *CBL, EJECT, TITLE, SKIP1, SKIP2y SKIP3)
- Especificación de opciones de compilador (sentencias CBL y PROCESS)
- Especificación de procedimientos de manejo de excepciones COBOL (sentencias USE)

Las sentencias de direccionamiento de compilador siguientes no tienen ningún efecto: ENTER, READY o RESET TRACE, SERVICE LABEL y SERVICE RELOAD.

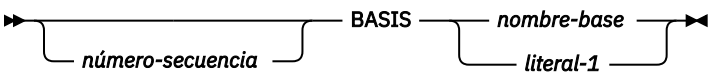
sentencia BASIS

La sentencia BASIS es una sentencia de biblioteca de texto de origen ampliado. Proporciona un programa COBOL completo como fuente para una compilación.

Un programa completo puede almacenarse como una entrada en una biblioteca definida por el usuario y puede utilizarse como fuente para una compilación. La entrada del compilador es una sentencia BASIS, seguida opcionalmente por cualquier número de sentencias INSERT y DELETE.

Nota: Una sentencia BASIS sólo puede ir seguida de sentencias INSERT o DELETE, y sólo puede ir precedida de sentencias PROCESS (CBL) o *CONTROL (*CBL).

Formato



número-secuencia
Puede aparecer opcionalmente en las columnas 1 a 6, seguidas de un espacio. El contenido de este campo se ignora.

BASIS
Puede aparecer en cualquier lugar de las columnas 1 a 72 en formato de origen fijo, o en las columnas 1 a 252 en formato de origen ampliado, seguido de *nombre-base*. No debe haber ningún otro texto en la sentencia.

nombre-base, literal-1
Es el nombre por el que el entorno del sistema conoce la entrada de biblioteca.

Para reglas de formación y reglas de proceso, consulte la descripción en *literal-1* y *text-name* de “sentencia COPY” en la página 508.

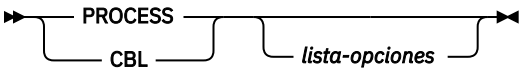
El archivo fuente permanece sin cambios después de la ejecución de la sentencia BASIS.

Nota de uso: Si se utilizan sentencias INSERT o DELETE para modificar el texto fuente COBOL proporcionado por una sentencia BASIS, el campo de secuencia del texto fuente COBOL debe contener números de secuencia numéricos en orden ascendente.

sentencia PROCESS (CBL)

Con la sentencia PROCESS (CBL), puede especificar opciones de compilador que se utilizarán en la compilación del programa. La sentencia PROCESS (CBL) se coloca antes de la cabecera IDENTIFICATION DIVISION de un programa más externo.

Formato



opciones-lista

Una serie de una o más opciones de compilador, cada una separada por una coma o un espacio.

Para obtener más información sobre las opciones de compilador, consulte *Opciones de compilador* en la publicación *COBOL for Linux en x86 Guía de programación*.

La sentencia PROCESS (CBL) puede ir precedida de un número de secuencia en las columnas 1 a 6. El primer carácter del número de secuencia debe ser numérico y PROCESS o CBL puede empezar en la columna 8 o después; si no se especifica un número de secuencia, PROCESS o CBL puede empezar en la columna 1 o después.

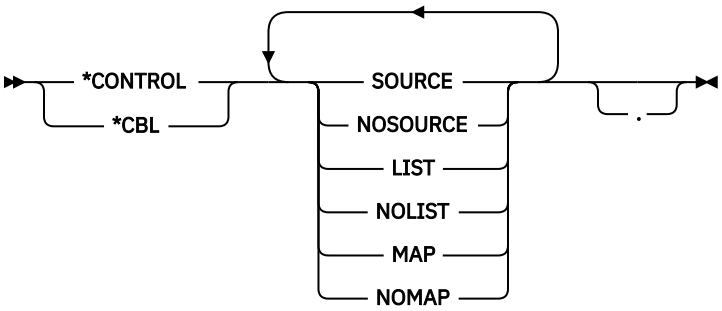
La sentencia PROCESS (CBL) debe finalizar antes o en la columna 72 en formato de origen fijo, o en la columna 252 en formato de origen ampliado, y las opciones no pueden continuar en varias sentencias PROCESS (CBL). Sin embargo, puede utilizar más de una sentencia PROCESS (CBL). Varias sentencias PROCESS (CBL) deben ir una detrás de la otra sin sentencias intermedias de ningún otro tipo.

La sentencia PROCESS (CBL) debe colocarse antes de cualquier línea de comentario u otra sentencia de direccionamiento de compilador.

Sentencia *CONTROL (*CBL)

Con la sentencia *CONTROL (o *CBL), puede visualizar o suprimir de forma selectiva el listado del código fuente y las correlaciones de almacenamiento en todo el texto fuente.

Formato



Para obtener una descripción completa de la salida generada por estas opciones, consulte *Obtención de listados* en la publicación *COBOL for Linux en x86 Guía de programación*.

Para obtener información sobre cómo especificar si los listados se codifican en UTF-8 o en la página de códigos especificada por el entorno local de tiempo de compilación, consulte *LSTFILE* en la publicación *COBOL for Linux en x86 Guía de programación*.

Las sentencias *CONTROL y *CBL son sinónimas. *CONTROL se acepta en cualquier lugar en el que se acepte *CBL.

Los caracteres *CONTROL o *CBL pueden empezar en cualquier columna que empiece por la columna 7, seguidos como mínimo de un espacio o coma y una o más palabras clave de opción. Las palabras clave de opción deben estar separadas por uno o más espacios o comas. Esta sentencia debe ser la única sentencia de la línea y no se permite la continuación. La sentencia se puede terminar con un punto.

Las sentencias *CONTROL y *CBL deben estar incorporadas en un fuente de programa. Por ejemplo, en el caso de aplicaciones por lotes, las sentencias *CONTROL y *CBL deben colocarse entre la sentencia PROCESS (CBL) y el final del programa (o el marcador END PROGRAM, si se especifica).

La línea fuente que contiene la sentencia *CONTROL (*CBL) no aparecerá en el listado fuente.

Si una opción se define durante la instalación como una opción fija, dicha opción fija tiene prioridad sobre todos los siguientes parámetros y sentencias:

- PARM (si está disponible)
- sentencia CBL
- Sentencia *CONTROL (*CBL)

Las opciones solicitadas se manejan de la siguiente manera:

1. Si una opción o su negación aparece más de una vez en una sentencia *CONTROL, se utiliza la última aparición de la palabra de opción.
2. Si se ha solicitado la opción correspondiente como parámetro para el compilador, una sentencia *CONTROL con la negación de la palabra de opción debe preceder a las partes del texto fuente para las que se va a inhibir la salida de listado. A continuación, la salida de listado se reanuda cuando se encuentra una sentencia *CONTROL con la palabra de opción afirmativa.
3. Si se ha solicitado la negación de la opción correspondiente como parámetro al compilador, *siempre* se inhibe dicho listado.
4. La sentencia *CONTROL sólo está en vigor dentro del programa fuente en el que se escribe, incluidos los programas contenidos. No permanece en vigor entre compilaciones por lotes de dos o más programas fuente COBOL.

Listado de código fuente

El tema lista sentencias que controlan el listado de las líneas de texto de origen de entrada.

La sentencia puede ser cualquiera de las siguientes:

```
*CONTROL SOURCE        [*CBL SOURCE]
*CONTROL NOSOURCE      [*CBL NOSOURCE]
```

Si se encuentra una sentencia *CONTROL NOSOURCE y se ha solicitado SOURCE como opción de compilación, la impresión del listado fuente se suprime a partir de este momento. Se emite un mensaje informativo (nivel I) que indica que se ha suprimido la impresión del origen.

Listado de códigos de objeto

El compilador siempre genera un listado de códigos de objeto. Las opciones LIST y NOLIST de la sentencia *CONTROL (o *CBL) se comprueban con la sintaxis pero no tienen ningún efecto en el listado de códigos de objeto.

Lista de correlaciones de almacenamiento

El tema lista las sentencias que controlan el listado de entradas de correlación de almacenamiento que se producen en DATA DIVISION.

La sentencia puede ser cualquiera de las siguientes:

```
*CONTROL MAP           [*CBL MAP]
*CONTROL NOMAP         [*CBL NOMAP]
```

Si se encuentra una sentencia *CONTROL NOMAP y se ha solicitado MAP como opción de compilación, el listado de entradas de correlación de almacenamiento se suprime a partir de este punto.

Por ejemplo, cualquiera de los siguientes conjuntos de sentencias genera un listado de correlaciones de almacenamiento en el que A y B no aparecerán:

```
*CONTROL NOMAP          *CBL NOMAP
    01  A                   01  A
    02  B                   02  B
*CONTROL MAP            *CBL MAP
```

sentencia COPY

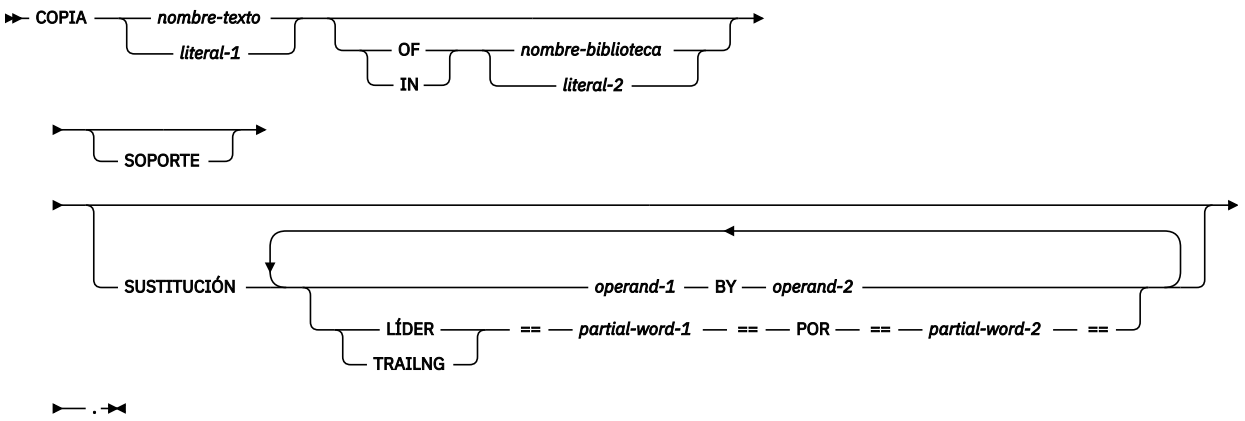
La sentencia COPY es una sentencia de biblioteca que coloca texto preescrito en una unidad de compilación COBOL.

Las entradas de código fuente preescritas pueden incluirse en una unidad de compilación durante la compilación. Por lo tanto, una instalación puede utilizar descripciones de archivo estándar, descripciones de registro o procedimientos sin recodificarlos. A continuación, estas entradas y procedimientos se pueden salvar en bibliotecas creadas por el usuario; a continuación, se pueden incluir en programas y definiciones de clase mediante la sentencia COPY.

La compilación del código fuente que contiene sentencias COPY es lógicamente equivalente a procesar todas las sentencias COPY antes de procesar el texto fuente resultante.

El efecto del proceso de una sentencia COPY es que el texto de biblioteca asociado con *nombre-texto* se copia en la unidad de compilación, sustituyendo lógicamente toda la sentencia COPY, empezando por la palabra COPY y terminando por el punto, ambos incluidos. Cuando no se especifica la frase SUSTITUIR, el texto de la biblioteca se copia sin cambios.

Formato



nombre-texto, nombre-biblioteca

nombre-texto identifica el texto de copia. *nombre-biblioteca* identifica dónde existe el texto de copia.

- Puede tener de 1 a 30 caracteres de longitud
- Puede contener los siguientes caracteres: letras mayúsculas latinas A-Z, letras minúsculas latinas a-z, dígitos 0-9, guión y subrayado

No es necesario que *text-name* ni *library-name* sean exclusivos dentro de un programa. Pueden ser idénticas a otras palabras definidas por el usuario en el programa.

No es necesario calificar *nombre-texto* . Si *nombre-texto* no está calificado, se presupone un nombre de biblioteca de SYSLIB.

literal-1 , literal-2
Deben ser literales alfanuméricos. *literal-1* identifica el texto de copia. *literal-2* identifica dónde existe el texto de copia.

El literal puede tener de 1 a 160 caracteres de longitud.

La exclusividad de *nombre-texto* y *nombre-biblioteca* se determina después de que se hayan aplicado las reglas de formación y conversión para un nombre dependiente del sistema.

Para obtener información sobre la correlación de caracteres en *nombre-texto*, *nombre-biblioteca*y literales, consulte *Sentencias de direccionamiento de compilador* en *COBOL for Linux en x86 Guía de programación*.

operand-1, operand-2
Puede ser pseudotexto, un identificador, un identificador de función, un literal o una palabra COBOL (excepto la palabra COPY). Para obtener detalles, consulte "Frase SUSTITUIR" en la página 510.

El texto de biblioteca puede constar de o incluir palabras, identificadores o literales que se pueden escribir en el texto de origen. Esto incluye palabras definidas por el usuario de varios bytes, literales de varios bytes y literales nacionales.

partial-word-1, partial-word-2
Puede ser una palabra parcial. Para obtener más información, consulte "Frase SUSTITUIR" en la página 510.

Cada sentencia COPY debe ir precedida de un espacio y finalizar con un punto separador.

Una sentencia COPY puede aparecer en el texto de origen en cualquier lugar donde aparezca una serie de caracteres o un separador.

Las sentencias COPY pueden estar anidadas y cualquier sentencia COPY de una cadena de sentencias COPY anidadas puede tener la frase SUSTITUIR, siempre que solo haya una sentencia COPY de este tipo en la cadena. Cuando se especifica la frase SUSTITUIR para una sentencia COPY que aparece en una cadena de sentencias COPY anidadas, la frase SUSTITUIR se aplica a todo el texto de biblioteca que se incluye en las sentencias COPY anidadas bajo la sentencia COPY que tiene la frase SUSTITUIR.

Una sentencia COPY anidada no puede provocar la recurrencia. Es decir, un miembro COPY sólo se puede nombrar una vez en un conjunto de sentencias COPY anidadas hasta que se alcance el final del archivo para ese miembro COPY. Por ejemplo, supongamos que el texto de origen contiene la sentencia: `COPY X.` y el texto de biblioteca X contiene la sentencia: `COPY Y.`.

En este caso, el texto de biblioteca contenido en Y no debe tener una sentencia `COPY X` o `COPY Y .`

Las líneas de depuración están permitidas dentro del texto de la biblioteca y el pseudo-texto. Las palabras de texto dentro de una línea de depuración participan en las reglas coincidentes como si la "D" no apareciera en el área de indicador. Se especifica una línea de depuración dentro del pseudotexto si la línea de depuración empieza en el texto de origen después del delimitador de pseudotexto de apertura pero antes del delimitador de pseudotexto de cierre coincidente.

Si se introducen líneas adicionales en el texto de origen como resultado de una sentencia COPY, cada palabra de texto introducida aparece en una línea de depuración si la sentencia COPY empieza en una línea de depuración o si la palabra de texto que se introduce aparece en una línea de depuración en el texto de la biblioteca. Cuando se introduce una palabra de texto especificada en la frase BY, aparece en una línea de depuración si la primera palabra de texto de biblioteca que se sustituye se especifica en una línea de depuración.

Cuando se especifica una sentencia COPY en una línea de depuración, el texto copiado se trata como si apareciera en una línea de depuración, excepto que las líneas de comentario del texto aparecen como líneas de comentario en el texto fuente resultante.

Si la palabra COPY aparece en una entrada de comentario, o en el lugar donde puede aparecer una entrada de comentario, se considera parte de la entrada de comentario.

Después de que se hayan procesado todas las sentencias COPY y REPLACE, se considerará que una línea de depuración tiene todas las características de una línea de comentario, si no se especifica la cláusula WITH DEBUGGING MODE en el párrafo SOURCE-COMPUTER.

Las líneas de comentario, los comentarios en línea o las líneas en blanco pueden aparecer en el texto de la biblioteca. Las líneas de comentario, comentarios en línea, o las líneas en blanco que aparecen en el texto de la biblioteca se copian en el texto de origen resultante sin modificar con la siguiente excepción: una línea de comentario, un comentario en línea, o una línea en blanco en el texto de la biblioteca no se copian si esa línea de comentario, comentario en línea, o línea en blanco aparece dentro de la secuencia de palabras de texto que coinciden con *operand-1* (consulte "Reglas de comparación y sustitución" en la página 512).

Las líneas que contienen sentencias *CONTROL (*CBL), EJECT, SKIP1, SKIP2, SKIP3o TITLE pueden aparecer en el texto de la biblioteca. Estas líneas se tratan como líneas de comentario durante el proceso de la sentencia COPY.

La corrección sintáctica de todo el texto fuente COBOL no se puede determinar hasta que todas las sentencias COPY y REPLACE se hayan procesado completamente, porque la corrección sintáctica del texto de biblioteca no se puede determinar de forma independiente.

El texto de biblioteca copiado de la biblioteca se coloca en la misma área del programa resultante que en la biblioteca. El texto de la biblioteca debe ajustarse a las reglas del formato 85 COBOL Estándar .

Nota: Los caracteres fuera de los definidos para palabras y separadores COBOL no deben aparecer en el texto de biblioteca o pseudo-texto excepto en líneas de comentario, comentarios en línea, entradas de comentario, literales alfanuméricos, literales DBCS o literales nacionales.

Frase SUPPRESS

La frase SUPPRESS especifica que el texto de la biblioteca no debe imprimirse en el listado fuente.

Frase SUSTITUIR

Cuando se especifica la frase SUSTITUIR, el texto de la biblioteca se copia y cada aparición coincidente de *operand-1* o *partial-word-1* dentro del texto de la biblioteca se sustituye por el *operand-2* o *partial-word-2*asociado.

En la siguiente discusión, cuando se especifica la palabra clave INICIAL o TRAILING de la frase SUSTITUIR, cada operando de la frase SUSTITUIR debe ser una palabra parcial. De lo contrario, cada *operando* puede constar de uno de los elementos siguientes:

- Pseudotexto
- Un identificador
- Un literal
- Una palabra COBOL (excepto la palabra COPY)
- Un identificador de función
- palabra parcial

pseudo-texto

Una secuencia de series de caracteres o separadores, o ambos, delimitados por, pero sin incluir, delimitadores de pseudotexto (==). Ambos caracteres de cada delimitador de pseudo-texto deben aparecer en una línea; sin embargo, las cadenas de caracteres dentro del pseudo-texto pueden continuar.

Las series de caracteres individuales dentro del pseudotexto pueden tener hasta 322 caracteres de longitud; pueden continuar sujetas a las reglas de continuación normales para el formato de código fuente.

Tenga en cuenta que una serie de caracteres debe estar delimitada por separadores. Para obtener más información, consulte Capítulo 1, “Caracteres”, en la página 3

pseudo-text-1 hace referencia al pseudotexto cuando se utiliza para *operand-1*, y *pseudo-text-2* hace referencia a pseudotexto cuando se utiliza para *operand-2*.

pseudo-text-1 puede constar únicamente de la coma de separador o el punto y coma de separador. *pseudo-text-2* puede ser nulo; puede constar únicamente de caracteres de espacio, líneas de comentario o comentarios en línea.

El pseudo-texto no debe contener la palabra COPY.

Cada palabra de texto de *pseudo-text-2* que se va a copiar en el programa se coloca en la misma área del programa resultante que el área en la que aparece en *pseudo-text-2*.

El pseudo-texto puede consistir o incluir palabras (excepto COPY), identificadores o literales que se pueden escribir en el texto de origen. Esto incluye multibyte palabras definidas por el usuario, literales DBCS y literales nacionales.

Las palabras definidas por el usuario de Multibyte deben estar totalmente formadas; es decir, no hay ninguna sustitución de palabra parcial para las palabras de multibyte .

Las palabras o literales que contienen multibyte caracteres no se pueden continuar entre líneas.

Utilice pseudotexto cuando sustituya una serie de caracteres PICTURE. Para evitar ambigüedades, toda la cláusula PICTURE, incluida la palabra clave PICTURE o PIC, debe especificarse en *pseudo-text-1*.

identificador
Puede definirse en cualquier sección de DATA DIVISION.

literal
Puede ser numérico, alfanumérico, DBCS o nacional.

palabra
Puede ser cualquier palabra COBOL individual (excepto COPY), incluidas las palabras definidas por el usuario de multibyte . Las palabras definidas por el usuario de Multibyte deben estar totalmente formadas. No puede sustituir parte de una palabra multibyte .

Puede incluir los caracteres COBOL no separadores (por ejemplo, + */$< > =) como parte de una palabra COBOL cuando se utiliza como operandos SUSTITUIR. Además, un guión o subrayado puede estar al principio de la palabra o un guión puede estar al final de la palabra.

identificador-función
Secuencia de series de caracteres y separadores que hace referencia de forma exclusiva al elemento de datos que resulta de la evaluación de una función. Para obtener más información, consulte el apartado “Identificador-función” en la página 68.

palabra-parcial
Una sola palabra de texto que está limitada por, pero sin incluir, los delimitadores de pseudo-texto (==). Ambos caracteres de cada delimitador de pseudotexto deben aparecer en una línea. Sin embargo, la palabra de texto dentro de una palabra parcial se puede continuar.

Las reglas siguientes se aplican a *partial-word-1* y *partial-word-2*:

- *partial-word-1* consta de una palabra de texto.
- *partial-word-2* consta de cero o una palabra de texto.
- *partial-word-1* y *partial-word-2* no pueden ser un literal alfanumérico, literal nacional, literal DBCS o palabra multibyte .

A efectos de coincidencia, cada identificador, literal, palabra o identificador de función se trata como pseudo-texto que contiene únicamente ese identificador, literal, palabra o identificador de función, respectivamente.

Reglas de comparación y sustitución

En este tema se presentan las reglas detalladas para la comparación y sustitución.

- Los operadores aritméticos y lógicos se consideran palabras de texto y sólo se pueden sustituir mediante un operando de pseudotexto.
- Los espacios en blanco iniciales y finales no se incluyen en el proceso de comparación de texto. Los blancos intercalados se utilizan en el proceso de comparación de texto para separar varias palabras de texto.
- Cuando *operand-1* es una constante figurativa, *operand-1* sólo coincide con la misma constante figurativa exacta. Por ejemplo, si se especifica `ALL "AB"` en *operand-1*, `"ABAB"` en el texto de la biblioteca no se considera una coincidencia; sólo `ALL "AB"` se considera una coincidencia.
- Cualquier coma separador, punto y coma o espacio que preceda a la palabra situada más a la izquierda en el texto de la biblioteca se copiará en el texto de origen. Empezando por la palabra de texto de la biblioteca más a la izquierda y el primer *operand-1* o *partial-word-1* especificado en la frase SUSTITUIR, el operando SUSTITUIR completo que precede a la palabra clave BY se compara con un número equivalente de palabras de texto de biblioteca contiguas.
- *operand-1* coincide con el texto de la biblioteca sólo si la secuencia ordenada de palabras de texto que forma *operand-1* es igual, carácter por carácter, a la secuencia ordenada de palabras de la biblioteca. Para los caracteres nacionales, la secuencia de caracteres nacionales debe ser igual, el carácter nacional para el carácter nacional, a la secuencia ordenada de palabras de biblioteca.
- Cuando se especifica la frase CABECERA, *partial-word-1* coincide con el texto de la biblioteca sólo si la secuencia contigua de caracteres que forma *partial-word-1* es igual, carácter para carácter, a un número igual de caracteres contiguos que empiezan por la posición de carácter más a la izquierda de una palabra de texto de biblioteca.

  Cuando se especifica la frase TRAILING, *partial-word-1* coincide con el texto de la biblioteca sólo si la secuencia contigua de caracteres que forma *partial-word-1* es igual, carácter para carácter, a un número igual de caracteres contiguos que terminan con la posición de carácter más a la derecha de una palabra de texto de biblioteca.
- A efectos de coincidencia, cada aparición de una coma de separador, un punto y coma de separador o una secuencia de uno o más espacios de separador se considera un único espacio.

  Sin embargo, cuando *operand-1* o *partial-word-1* consta únicamente de una coma de separador o un punto y coma de separador, *operand-1* o *partial-word-1* participa en la coincidencia como una palabra de texto. En este caso, se puede omitir el espacio que sigue al separador de coma o punto y coma.

  Cuando el texto de la biblioteca contiene una comilla de cierre que no va seguida inmediatamente de un espacio de separador, una coma de separador, un punto y coma de separador o un punto de separador, la comilla de cierre se considera una comilla de separador.
- Si no se produce ninguna coincidencia, la comparación se repite con cada *operand-1* o *partial-word-1* si se especifica, hasta que se encuentre una coincidencia o hasta que no existan más operandos SUSTITUIR.
- Siempre que se produce una coincidencia entre *operand-1* y el texto de la biblioteca, el *operand-2* correspondiente se copia en el texto de origen. Siempre que se produzca una coincidencia entre *partial-word-1* y la palabra de texto de biblioteca, Los caracteres coincidentes de esa palabra de texto de biblioteca se sustituyen por *partial-word-2* o se suprimen cuando *partial-word-2* consta de cero palabras de texto.
- Cuando se comparan todos los operandos y no se produce ninguna coincidencia, la palabra de texto de la biblioteca más a la izquierda se copia en el texto de origen.
- Una vez que una palabra de texto de biblioteca ha terminado de procesarse y se copia en el texto de origen sin cambios o se sustituye debido a una coincidencia, la siguiente palabra de texto de biblioteca no copiada sucesiva se considera entonces como la palabra de texto más a la izquierda, y el ciclo de comparación comienza de nuevo, a partir de la primera aparición de *operand-1* o *partial-word-1*. El proceso continúa hasta que se compara la palabra de texto de la biblioteca más a la derecha.

- La secuencia de palabras de texto en el texto de la biblioteca, *pseudo-text-1*, y *partial-word-1* está determinada por las reglas para el formato de referencia. Para obtener más información, consulte el apartado Capítulo 6, “Formato de referencia”, en la página 45.
- Cuando las palabras de texto se colocan en el texto de origen, sólo se introducen espacios adicionales entre las palabras de texto en las que ya existe un espacio, incluido el espacio asumido entre líneas de origen.
- Las reglas siguientes se aplican a líneas de comentario, comentarios en línea y líneas en blanco:
  - Las líneas de comentario, los comentarios en línea o las líneas en blanco en el texto de la biblioteca, *pseudo-text-1*o *partial-word-1* se ignoran a efectos de coincidencia.
  - Las líneas de comentario, los comentarios en línea o las líneas en blanco del texto de la biblioteca se copian en el texto fuente resultante sin modificar con la excepción siguiente: una línea de comentario, un comentario en línea o una línea en blanco del texto de la biblioteca no se copia si esa línea de comentario, comentario en línea, o aparece una línea en blanco en la secuencia de palabras de texto que coinciden con *pseudo-text-1* o *partial-word-1*.
  - Las líneas de comentario, los comentarios en línea o las líneas en blanco en *pseudo-text-2* o *partial-word-2* se copian en el programa resultante sin cambios siempre que *pseudo-text-2* o *partial-word-2* se coloque en el texto de origen como resultado de la sustitución de texto.
  - Si la palabra COPY aparece en una entrada de comentario, o en el lugar donde puede aparecer una entrada de comentario, se considera parte de la entrada de comentario.
- COPY SUSTITUIR no afecta a las sentencias EJECT, SKIP1, SKIP2, SKIP3o a las sentencias de dirección del compilador TITLE.
- Las líneas que contienen sentencias *CONTROL (*CBL), EJECT, SKIP1, SKIP2, SKIP3o TITLE pueden aparecer en el texto de la biblioteca. Estas líneas se copian en el texto de origen resultante sin modificar.
- Las reglas siguientes se aplican a las líneas de depuración:
  - Las líneas de depuración están permitidas en el texto de la biblioteca y en el pseudo-texto. Las palabras de texto dentro de una línea de depuración participan en las reglas de coincidencia como si la letra "D" no apareciera en el área de indicador. Se especifica una línea de depuración dentro del pseudotexto si la línea de depuración empieza en el texto de origen después del delimitador de pseudotexto de apertura pero antes del delimitador de pseudotexto de cierre coincidente.
  - Si se introducen más líneas en el texto de origen como resultado de una sentencia COPY, cada palabra de texto que se introduce aparece en una línea de depuración si la sentencia COPY empieza en una línea de depuración o si la palabra de texto que se introduce aparece en una línea de depuración en el texto de la biblioteca. Cuando se introduce una palabra de texto especificada en la frase BY, aparece en una línea de depuración si la primera palabra de texto de biblioteca que se está sustituyendo se especifica en una línea de depuración.
  - Cuando se especifica una sentencia COPY en una línea de depuración, el texto copiado se trata como si apareciera en una línea de depuración, excepto que las líneas de comentario del texto aparecen como líneas de comentario en el texto fuente resultante.
  - Después de que se procesen todas las sentencias COPY y REPLACE, se considerará que una línea de depuración tiene todas las características de una línea de comentario, si no se especifica la cláusula WITH DEBUGGING MODE en el párrafo SOURCE-COMPUTER.
- La corrección sintáctica de todo el texto fuente COBOL no se puede determinar hasta que todas las sentencias COPY y REPLACE se hayan procesado completamente, porque la corrección sintáctica del texto de biblioteca no se puede determinar de forma independiente.
- (Esta regla sólo se aplica a pseudotexto.) Si el texto de origen tiene apariciones de un operando ficticio `:TAG:` que está delimitado por dos puntos en el texto del programa, el compilador sustituye el operando ficticio por el texto necesario . El Ejemplo 3 muestra cómo se utiliza con el operando ficticio `:TAG:`. Los dos puntos sirven como separadores y hacen que TAG sea un operando autónomo.
- La sentencia COPY con la frase REPLACE se puede utilizar para sustituir partes de palabras, ya sea utilizando la frase `LEADING|TRAILING` *`partial-word-1`* `BY` *`partial-word-2`* o utilizando el método `:TAG:` de pseudotexto.

- Después de la sustitución, las palabras de texto se colocan en el texto de origen de acuerdo con las reglas de formato de 85 COBOL Estándar .

Ejemplos de comparación y sustitución

Este tema muestra ejemplos de comparación y sustitución.

Las secuencias de código (por ejemplo, descripciones de archivo y datos, rutinas de error y excepción) que son comunes a una serie de programas pueden salvarse en una biblioteca y, a continuación, utilizarse con la sentencia COPY. Si se establecen convenios de denominación para dicho código común, no es necesario especificar la frase SUSTITUIR. Si los nombres cambian de un programa a otro, se puede utilizar la frase SUSTITUIR para proporcionar nombres significativos para este programa.

Ejemplo 1

En este ejemplo, el texto de biblioteca PAYLIB consta de las siguientes entradas DATA DIVISION:

```
01  A.
  02  B    PIC S99.
  02  C    PIC S9(5)V99.
  02  D    PIC S9999 OCCURS 1 TO 52 TIMES
      DEPENDING ON B OF A.
```

Puede utilizar la sentencia COPY en DATA DIVISION de un programa como se indica a continuación:

```
COPY PAYLIB.
```

En este programa, se copia el texto de la biblioteca. El texto resultante se trata como si estuviera escrito de la siguiente manera:

```
01  A.
  02  B    PIC S99.
  02  C    PIC S9(5)V99.
  02  D    PIC S9999 OCCURS 1 TO 52 TIMES
      DEPENDING ON B OF A.
```

Ejemplo 2

Para cambiar algunos o todos los nombres dentro del texto de la biblioteca, puede utilizar la frase SUSTITUIR:

```
COPY PAYLIB REPLACING A BY PAYROLL
                      B BY PAY-CODE
                      C BY GROSS-PAY
                      D BY HOURS.
```

En este programa, se copia el texto de la biblioteca. El texto resultante se trata como si estuviera escrito de la siguiente manera:

```
01  PAYROLL.
  02  PAY-CODE    PIC S99.
  02  GROSS-PAY   PIC S9(5)V99.
  02  HOURS       PIC S9999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAY-CODE OF PAYROLL.
```

Los cambios que se muestran sólo se realizan para este programa. El texto permanece inalterado tal como aparece en la biblioteca.

Ejemplo 3

Si se siguen los siguientes convenios en el texto de la biblioteca, las partes de los nombres (por ejemplo, la parte de prefijo de los nombres de datos) se pueden cambiar con la frase SUSTITUIR.

En este ejemplo, el texto de biblioteca PAYLIB consta de las siguientes entradas DATA DIVISION:

```
01  :TAG:.
  02  :TAG:-WEEK          PIC S99.
  02  :TAG:-GROSS-PAY     PIC S9(5)V99.
  02  :TAG:-HOURS         PIC S999  OCCURS 1 TO 52 TIMES
      DEPENDING ON :TAG:-WEEK OF :TAG:.
```

Puede utilizar la sentencia COPY en DATA DIVISION de un programa como se indica a continuación:

```
COPY PAYLIB REPLACING ==:TAG:== BY ==Payroll==.
```

Nota de uso: En este ejemplo, el uso necesario de dos puntos o paréntesis como delimitadores en el texto de la biblioteca. Se recomiendan dos puntos para mayor claridad porque se pueden utilizar paréntesis para un subíndice, por ejemplo al hacer referencia a un elemento de tabla.

En este programa, se copia el texto de la biblioteca. El texto resultante se trata como si estuviera escrito de la siguiente manera:

```
01  PAYROLL.
  02  PAYROLL-WEEK        PIC S99.
  02  PAYROLL-GROSS-PAY   PIC S9(5)V99.
  02  PAYROLL-HOURS       PIC S999  OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

Los cambios que se muestran sólo se realizan para este programa. El texto permanece inalterado tal como aparece en la biblioteca.

Ejemplo 4

Este ejemplo muestra cómo sustituir de forma selectiva los números de nivel sin sustituir los números de la cláusula PICTURE:

```
COPY xxx REPLACING ==(01)== BY ==(01)==
                   == 01 == BY == 05 ==.
```

Ejemplo 5

Este ejemplo muestra el uso de la palabra clave LÍDER de la frase SUSTITUIR en la sentencia COPY. El texto de biblioteca PAYLIB consta de las siguientes entradas de DATA DIVISION:

```
01  DEPT.
  02  DEPT-WEEK       PIC S99.
  02  DEPT-GROSS-PAY  PIC S9(5)V99.
  02  DEPT-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON DEPT-WEEK OF DEPT.
```

Puede utilizar la sentencia COPY en DATA DIVISION de un programa como se indica a continuación:

```
COPY PAYLIB REPLACING LEADING == DEPT == BY == PAYROLL ==.
```

En este programa, se copia el texto de la biblioteca. El texto resultante se trata como si estuviera escrito de la siguiente manera:

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

Los cambios que se muestran sólo se realizan para este programa. El texto permanece inalterado tal como aparece en la biblioteca.

Ejemplo 6

Este ejemplo muestra el uso de la palabra clave TRAILING de la frase SUSTITUIR en la sentencia COPY. El texto de biblioteca PAYLIB consta de las siguientes entradas de DATA DIVISION:

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

Puede utilizar la sentencia COPY en DATA DIVISION de un programa como se indica a continuación:

```
COPY PAYLIB REPLACING TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

En este programa, se copia el texto de la biblioteca. El texto resultante se trata como si estuviera escrito de la siguiente manera:

```
01  PAYROLL.
  02  PAYROLL-WEEK     PIC S99.
  02  PAYROLL-NET-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

Los cambios que se muestran sólo se realizan para este programa. El texto permanece inalterado tal como aparece en la biblioteca.

Ejemplo 7

Este ejemplo muestra un escenario en el que se especifican dos tipos de sustitución de palabra parcial en una sola frase REPLACEMENT. El texto de biblioteca PAYLIB consta de las siguientes entradas de DATA DIVISION:

```
01  PAYROLL.
  02  PAYROLL-WEEK     PIC S99.
  02  :TAG:-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

Puede utilizar la sentencia COPY en DATA DIVISION de un programa como se indica a continuación:

```
COPY PAYLIB REPLACING == :TAG: == BY == PAYROLL ==
           TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

En este programa, se copia el texto de la biblioteca. El texto resultante se trata como si estuviera escrito de la siguiente manera:

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

Se especifican dos tipos de sustitución de palabra parcial en la misma operación REPLACEMENT, pero como de costumbre, se realiza una sustitución en una sola palabra de texto de biblioteca. Por lo tanto, aunque `:TAG:-GROSS-PAY` se considere una coincidencia con el primer operando de ambas operaciones de sustitución, después de que se realice la primera coincidencia y sustitución, no se realizará más sustitución en esa palabra.

Los cambios que se muestran sólo se realizan para este programa. El texto permanece inalterado tal como aparece en la biblioteca.

Ejemplo 8

Este ejemplo muestra el soporte para la frase SUSTITUIR en una cadena de sentencias COPY anidadas. En este ejemplo, es la sentencia COPY más externa la que tiene la frase SUSTITUIR, pero tenga en cuenta que la frase SUSTITUIR se puede especificar en cualquiera de las sentencias COPY anidadas de la cadena, siempre que se especifique sólo en una de ellas. El texto de biblioteca PAYLIB consta de las siguientes entradas de DATA DIVISION:

```
01  PAYROLL.
  02  PAYROLL-WEEK     PIC S99.
  02  :TAG:-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
      COPY PAYLIB2
```

El texto de biblioteca PAYLIB2 consta de las siguientes entradas de DATA DIVISION:

```
01  PAYROLL2.
  02  PAYROLL2-WEEK     PIC S99.
  02  :TAG:2-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL2-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

Puede utilizar la sentencia COPY en DATA DIVISION de un programa como se indica a continuación:

```
COPY PAYLIB REPLACING == :TAG: == BY == PAYROLL ==
            TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

En este programa, se copia el texto de la biblioteca. El texto resultante se trata como si estuviera escrito de la siguiente manera:

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-NET-PAY    PIC S9(5)V99.
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.

01  PAYROLL2.
  02  PAYROLL2-WEEK      PIC S99.
  02  PAYROLL2-NET-PAY   PIC S9(5)V99.
  02  PAYROLL2-HOURS     PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

La frase SUSTITUIR en la sentencia COPY más externa no sólo se aplica al texto de biblioteca en PAYLIB, sino también al texto en PAYLIB2.

Los cambios que se muestran sólo se realizan para este programa. El texto permanece inalterado tal como aparece en la biblioteca.

sentencia DELETE

La sentencia DELETE es una sentencia de biblioteca de origen ampliada. Elimina sentencias COBOL de un programa fuente incluido por una sentencia BASIS.

Formato

número-secuencia
Puede aparecer opcionalmente en las columnas 1 a 6, seguidas de un espacio. El contenido de este campo se ignora.

DELETE
Puede aparecer en cualquier lugar de las columnas 1 a 72 en formato de origen fijo, o en las columnas 1 a 252 en formato de origen ampliado. La palabra clave DELETE debe ir seguida de un espacio y del *campo-número-secuencia*. No debe haber ningún otro texto en la sentencia.

campo-número-secuencia
Cada número debe ser igual a un *número-secuencia* en el programa fuente BASIS. Este *número-secuencia* es el número de seis dígitos que el programador asigna en las columnas 1 a 6 del formulario de codificación COBOL. Los números a los que se hace referencia en el *campo-número-secuencia* de las sentencias INSERT o DELETE siempre deben especificarse en orden numérico ascendente.

El *campo-número-secuencia* debe ser una de las opcionessiguientes:

- Un único número
- Una serie de números únicos
- Un rango de números (indicado separando los dos números delimitadores del rango mediante un guión)
- Una serie de rangos de números
- Cualquier combinación de uno o más números únicos y uno o más rangos de números

Cada entrada del *campo-número-secuencia* debe estar separada de la entrada anterior por una coma seguida de un espacio. Por ejemplo:

```
000250 DELETE  000010-000050, 000400, 000450
```

Las sentencias de programa fuente pueden seguir a una sentencia DELETE. A continuación, estas sentencias de programa fuente se insertan en el programa fuente BASIS antes de la sentencia que sigue a la última sentencia suprimida (es decir, en el ejemplo anterior, antes de la siguiente sentencia que sigue a la sentencia suprimida 000450).

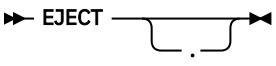
Si una sentencia DELETE empieza en la columna 12 o superior y un *campo-número-secuencia* válido no sigue a la palabra clave DELETE, el compilador presupone que esta sentencia DELETE es una sentencia DELETE de COBOL.

Nota de uso: Si se utilizan sentencias INSERT o DELETE para modificar el programa fuente COBOL proporcionado por una sentencia BASIS, el campo de secuencia del programa fuente COBOL debe contener números de secuencia numéricos en orden ascendente. El archivo de origen permanece sin cambios. Las sentencias INSERT o DELETE que hacen referencia a estos números de secuencia deben aparecer en orden ascendente.

sentencia EJECT

La sentencia EJECT especifica que la siguiente sentencia fuente se imprimirá en la parte superior de la página siguiente.

Formato



La sentencia EJECT debe ser la única sentencia de la línea. Se puede escribir en el Área A o en el Área B, y se puede terminar con un punto de separación.

La sentencia EJECT debe estar incorporada en un origen de programa. Por ejemplo, en el caso de aplicaciones por lotes, la sentencia EJECT debe colocarse entre la sentencia CBL (PROCESS) y el final del programa (o el marcador END PROGRAM, si se especifica).

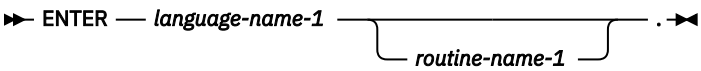
La sentencia EJECT no tiene ningún efecto en la compilación de la propia unidad de origen.

ENTER, sentencia

La sentencia ENTER está diseñada para facilitar el uso de más de un lenguaje fuente en el mismo programa fuente. Sin embargo, sólo se permite COBOL en el programa fuente.

La sentencia ENTER se comprueba con la sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Formato



language-name-1
Un nombre de sistema que no tiene ningún significado definido. Debe ser una palabra definida por el usuario con el formato correcto o la palabra "COBOL". Al menos un carácter debe ser alfabético.

routine-name-1
Debe seguir las reglas para la formación de una palabra definida por el usuario. Al menos un carácter debe ser alfabético.

sentencia INSERT

La sentencia INSERT es una sentencia de biblioteca que añade sentencias COBOL a un programa fuente incluido por una sentencia BASIS.

Formato

número-secuencia
Puede aparecer opcionalmente en las columnas 1 a 6, seguidas de un espacio. El contenido de este campo se ignora.

INSERT
Puede aparecer en cualquier lugar de las columnas 1 a 72 en formato de origen fijo, o en las columnas 1 a 252 en formato de origen ampliado, seguido de un espacio y el *campo de número de secuencia*. No debe haber ningún otro texto en la sentencia.

campo-número-secuencia
Un número que debe ser igual a un *número-secuencia* en el programa fuente BASIS. Este *número-secuencia* es un número de seis dígitos que el programador asigna en las columnas 1 a 6 de la línea de origen COBOL.

Los números a los que se hace referencia en el *campo-número-secuencia* de las sentencias INSERT o DELETE siempre deben especificarse en orden numérico ascendente.

El *campo-número-secuencia* debe ser un único número (por ejemplo, 000130). Al menos una sentencia de programa fuente nueva debe seguir a la sentencia INSERT para la inserción después del número de sentencia especificado por el *campo-número-secuencia*.

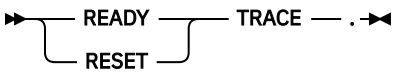
Las nuevas sentencias de programa fuente que siguen a la sentencia INSERT pueden incluir cualquier sintaxis COBOL.

Nota de uso: Si se utilizan sentencias INSERT o DELETE para modificar el programa fuente COBOL proporcionado por una sentencia BASIS, el campo de secuencia del programa fuente COBOL debe contener números de secuencia numéricos en orden ascendente. El archivo de origen permanece sin cambios. Las sentencias INSERT o DELETE que hacen referencia a estos números de secuencia deben aparecer en orden ascendente.

Sentencia READY o RESET TRACE

La sentencia READY o RESET TRACE se ha diseñado para rastrear la ejecución de procedimientos. La sentencia READY o RESET TRACE sólo puede aparecer en PROCEDURE DIVISION, pero no tiene ningún efecto en el programa.

Formato



Puede rastrear la ejecución de procedimientos utilizando la declarativa USE FOR DEBUGGING tal como se describe en *Ejemplo: USE FOR DEBUGGING* en *COBOL for Linux en x86 Guía de programación*.

sentencia REPLACE

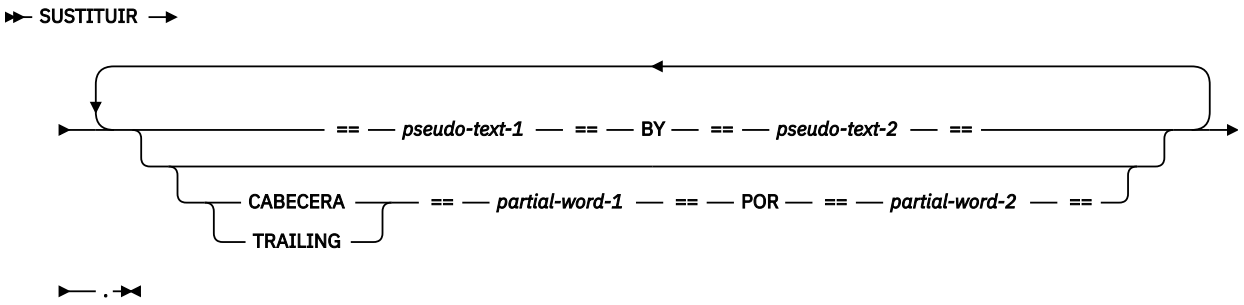
La sentencia REPLACE se utiliza para sustituir el texto de origen.

Una sentencia REPLACE puede aparecer en cualquier lugar del texto fuente en el que pueda aparecer una serie de caracteres. Debe ir precedido de un punto de separación, excepto cuando es la primera sentencia de un programa compilado por separado. Debe terminar con un punto separador.

La sentencia REPLACE proporciona un medio de aplicar un cambio a un grupo de compilación COBOL completo, o a parte de un grupo de compilación, sin tener que buscar y modificar manualmente todos los lugares que deben cambiarse. Es un método fácil de realizar sustituciones de serie simples. Es similar en acción a la frase SUSTITUIR de la sentencia COPY, excepto que actúa en todo el texto fuente, no sólo en el texto de las bibliotecas COPY.

Si la palabra REPLACE aparece en una entrada de comentario o en el lugar donde puede aparecer una entrada de comentario, se considera parte de la entrada de comentario.

Formato 1



Cada aparición coincidente de *pseudo-text-1* en el texto de origen se sustituye por el *pseudo-text-2*correspondiente.

Formato 2

Cualquier sustitución de texto actualmente en vigor se ha dejado de mantener con el formato format-2 de REPLACE. Si no se especifica el formato 2, una aparición específica de la sentencia REPLACE está en vigor desde el punto en el que se especifica hasta la siguiente aparición de una sentencia REPLACE o el final del programa compilado por separado.

pseudo-text-1

Debe contener una o más palabras de texto. Las cadenas de caracteres pueden continuar de acuerdo con las reglas de código fuente normales.

pseudo-text-1 puede constar únicamente de una coma de separador o un punto y coma de separador.

pseudo-text-2
Puede contener cero, una o más palabras de texto. Las series de caracteres se pueden continuar de acuerdo con las reglas de código fuente normales.

pseudo-text-1 y *pseudo-text-2* pueden contener cualquier palabra de texto (excepto la palabra COPY) que se pueda escribir en el texto de origen, incluidos los literales nacionales, los literales DBCS, y palabras definidas por el usuario de multibyte .

Los caracteres fuera de los que se permiten palabras y separadores COBOL no deben aparecer en el texto de biblioteca o pseudo-texto excepto en líneas de comentario, entradas de comentario, comentarios en línea, literales alfanuméricos, literales DBCS o literales nacionales.

Las palabras definidas por el usuario de Multibyte deben estar totalmente formadas; es decir, no hay ninguna sustitución de palabra parcial para las palabras de multibyte .

pseudo-text-1 y *pseudo-text-2* pueden contener caracteres de un solo byte y multibyte caracteres en líneas de comentario o entradas de comentario.

Las series de caracteres individuales dentro del pseudo-texto pueden tener hasta 322 caracteres de longitud, excepto que las series que contienen caracteres de multibyte no pueden continuar.

partial-word-1, partial-word-2
Una sola palabra de texto que está limitada por, pero sin incluir, los delimitadores de pseudo-texto (==). Ambos caracteres de cada delimitador de pseudotexto deben aparecer en una línea. Sin embargo, la palabra de texto dentro de una palabra parcial se puede continuar.

Las reglas siguientes se aplican a *partial-word-1* y *partial-word-2*:

- *partial-word-1* consta de una palabra de texto.
- *partial-word-2* consta de cero o una palabra de texto.
- *partial-word-1* y *partial-word-2* no pueden ser un literal alfanumérico, literal nacional, literal DBCS o palabra multibyte .

El compilador procesa sentencias REPLACE en texto fuente después del proceso de cualquier sentencia COPY. COPY debe procesarse primero para ensamblar el texto de origen completo. A continuación, se puede utilizar REPLACE para modificar el texto de origen, realizando una sustitución de serie simple. Las sentencias REPLACE no pueden contener ellas mismas sentencias COPY.

El texto que se produce como resultado del proceso de una sentencia REPLACE no debe contener una sentencia REPLACE.

Reglas de continuación para pseudotexto

Las cadenas de caracteres y separadores que componen el pseudo-texto pueden empezar en el Área A o en el Área B. Sin embargo, si un guión está en el área de indicador de una línea, y ese guión sigue al delimitador de apertura de pseudo-texto , el Área A de la línea debe estar en blanco, y las normas normales para la continuación de las líneas se aplican a la formación de palabras de texto. Consulte “Líneas de continuación” en la página 48.

Reglas de comparación

La operación de comparación que determina la sustitución de texto empieza por la palabra de texto de origen más a la izquierda que sigue a la sentencia REPLACE. y con la primera palabra de *pseudo-text-1* o *partial-word-1*.

- *pseudo-text-1* se compara con un número equivalente de palabras de texto de origen contiguas. *pseudo-text-1* sólo coincide con el texto de origen si la secuencia ordenada de palabras de texto que

forma *pseudo-text-1* es igual, carácter por carácter, a la secuencia ordenada de palabras de texto de origen. Para los caracteres nacionales, la secuencia de caracteres nacionales debe ser igual, el carácter nacional para el carácter nacional, a la secuencia ordenada de palabras de biblioteca.

- Cuando se especifica la frase CABECERA, *partial-word-1* coincide con el texto de origen sólo si la secuencia contigua de caracteres que forma *partial-word-1* es igual, carácter para carácter, a un número igual de caracteres contiguos que empiezan por la posición de carácter más a la izquierda de una palabra de texto fuente.

  Cuando se especifica la frase TRAILING, *partial-word-1* coincide con el texto de origen sólo si la secuencia contigua de caracteres que forma *partial-word-1* es igual, carácter para carácter, a un número igual de caracteres contiguos que terminan con la posición de carácter más a la derecha de una palabra de texto fuente.
- A efectos de coincidencia, cada aparición de una coma de separador, un punto y coma de separador o una secuencia de uno o más espacios de separador se considera un único espacio.

  Sin embargo, cuando *pseudo-text-1* o *partial-word-1* consta únicamente de una coma de separador o un punto y coma de separador, la coma o el punto y coma participa en la coincidencia como una palabra de texto. En este caso, se puede omitir el espacio que sigue al separador de coma o punto y coma.

  Cuando el texto de origen contiene una comilla de cierre que no va seguida inmediatamente de un espacio de separador, una coma de separador, un punto y coma de separador o un punto de separador, la comilla de cierre se considera una comilla de separador.
- Si no se produce ninguna coincidencia, la comparación se repite con cada aparición sucesiva de *pseudo-text-1* o *partial-word-1* si se especifica, hasta que se encuentre una coincidencia o hasta que no existan más operandos SUSTITUIR.
- Cuando se comparan todas las apariciones de *pseudo-text-1* o *partial-word-1* y no se produce ninguna coincidencia, la siguiente palabra de texto fuente sucesiva se considera entonces como la palabra de texto fuente más a la izquierda, y el ciclo de comparación comienza de nuevo, a partir de la primera aparición de *pseudo-text-1* o *partial-word-1*.
- Siempre que se produce una coincidencia entre *pseudo-text-1* y el texto de origen, el *pseudo-text-2* correspondiente sustituye el texto coincidente en el texto de origen. Siempre que se produzca una coincidencia entre *partial-word-1* y la palabra de texto de origen, los caracteres coincidentes de esa palabra de texto de origen se sustituyen por *partial-word-2* o se suprimen si *partial-word-2* consta de cero palabras de texto.La palabra de texto de origen que sigue inmediatamente a la palabra de texto situada más a la derecha que ha participado en la coincidencia se considera entonces como la palabra de texto de origen situada más a la izquierda. El ciclo de comparación se inicia de nuevo, empezando por la primera aparición de *pseudo-text-1* o *partial-word-1*.
- La operación de comparación continúa hasta que la palabra de texto situada más a la derecha en el texto de origen que está dentro del ámbito de la sentencia REPLACE ha participado en una coincidencia o se ha considerado como una palabra de texto de origen situada más a la izquierda y ha participado en un ciclo de comparación completo.

Reglas de sustitución

Este tema introduce reglas detalladas para la sustitución.

- La secuencia de palabras de texto en el texto de origen, *pseudo-text-1*, y *partial-word-1* está determinada por las reglas para el formato de referencia. Para obtener más información, consulte el apartado Capítulo 6, “Formato de referencia”, en la página 45.
- Las palabras de texto que se insertan en el texto de origen como resultado del proceso de una sentencia REPLACE se colocan en el texto de origen de acuerdo con las reglas para el formato de referencia. Al insertar palabras de texto de *pseudo-text-2* o *partial-word-2* en el texto de origen, sólo se introducen espacios adicionales entre palabras de texto en las que ya existe un espacio, incluido el espacio asumido entre líneas de origen.
- Si se introducen más líneas en el texto de origen como resultado del proceso de sentencias REPLACE, el área de indicador de las líneas introducidas contiene el mismo carácter que la línea en la que

empieza el texto que se va a sustituir, a menos que dicha línea contenga un guión, en cuyo caso la línea introducida contiene un espacio.

- Si cualquier literal dentro de *pseudo-text-2* o *partial-word-2* es de una longitud demasiado grande para acomodarlo en una sola línea sin continuación a otra línea en el programa resultante, y el literal no se coloca en una línea de depuración, se introducen más líneas de continuación que contienen el resto del literal. Si la sustitución requiere que el literal continuado continúe en una línea de depuración, el programa es erróneo.
- Cada palabra de *pseudo-text-2* o *partial-word-2* que se va a colocar en el programa resultante empieza en la misma área del programa resultante que aparece en *pseudo-text-2* o *partial-word-2*.
- Las reglas siguientes se aplican a líneas de comentario, comentarios en línea y líneas en blanco:
  - Las líneas de comentario, los comentarios en línea o las líneas en blanco en el texto de origen, *pseudo-text-1*o *partial-word-1* se ignoran a efectos de coincidencia.
  - Las líneas de comentario, los comentarios en línea o las líneas en blanco en el texto de origen se copian en el texto de origen resultante sin modificar con la excepción siguiente: una línea de comentario, un comentario en línea o una línea en blanco en el texto de origen no se copia si esa línea de comentario, comentario en línea, o aparece una línea en blanco en la secuencia de palabras de texto que coinciden con *pseudo-text-1* o *partial-word-1*.
  - Las líneas de comentario, los comentarios en línea o las líneas en blanco en *pseudo-text-2* o *partial-word-2* se copian en el programa resultante sin cambios siempre que *pseudo-text-2* o *partial-word-2* se coloque en el texto de origen como resultado de la sustitución de texto.
- Las líneas que contienen sentencias *CONTROL (*CBL), EJECT, SKIP1, SKIP2, SKIP3o TITLE pueden aparecer en el texto fuente. Estas líneas se copian en el texto de origen resultante sin modificar.
- Las reglas siguientes se aplican a las líneas de depuración:
  - Las líneas de depuración están permitidas en pseudotexto o en palabras parciales. Las palabras de texto dentro de una línea de depuración participan en las reglas de coincidencia como si la letra "D" no apareciera en el área de indicador.
  - Cuando se especifica una sentencia REPLACE en una línea de depuración, la sentencia se trata como si la letra "D" no apareciera en el área de indicador.
  - Después de procesar todas las sentencias COPY y REPLACE, se considera que una línea de depuración tiene todas las características de una línea de comentario si no se especifica la cláusula WITH DEBUGGING MODE en el párrafo SOURCE-COMPUTER.
- Excepto para las sentencias COPY y REPLACE, la corrección sintáctica del texto fuente no se puede determinar hasta que todas las sentencias COPY y REPLACE se procesen completamente.
- (Esta regla sólo se aplica al pseudotexto.) Si el texto de origen tiene apariciones de un operando ficticio `:TAG:` que está delimitado por dos puntos en el texto del programa, el compilador sustituye el operando ficticio por el texto necesario. Los dos puntos sirven como separadores y hacen que TAG sea un operando autónomo.

  Por ejemplo, puede utilizar la sentencia REPLACE en DATA DIVISION de un programa como se indica a continuación:

```
REPLACE ==:TAG:== BY ==Payroll==.

01  :TAG:.
  02  :TAG:-WEEK          PIC S99.
  02  :TAG:-GROSS-PAY     PIC S9(5)V99.
  02  :TAG:-HOURS         PIC S999  OCCURS 1 TO 52 TIMES
      DEPENDING ON :TAG:-WEEK OF :TAG:.
```

- La sentencia REPLACE se puede utilizar para sustituir partes de palabras, ya sea utilizando la frase `LEADING|TRAILING` *`partial-word-1`* `BY` *`partial-word-2`* o utilizando el método `:TAG:` de pseudotexto.

SERVICE LABEL, sentencia

La sentencia SERVICE LABEL es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Formato

La sentencia SERVICE LABEL sólo puede aparecer en PROCEDURE DIVISION, pero no en la sección de declaraciones.

SERVICE RELOAD, sentencia

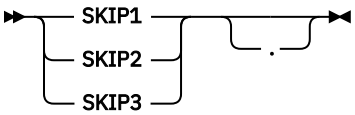
La sentencia SERVICE RELOAD es comprobación de sintaxis, pero no tiene ningún efecto en la ejecución del programa.

Formato

Sentencias SKIP

Las sentencias SKIP1, SKIP2y SKIP3 especifican líneas en blanco que el compilador debe añadir al imprimir el listado fuente. Las sentencias SKIP no tienen ningún efecto en la compilación del propio texto de origen.

Formato



SKIP1
Especifica una sola línea en blanco que se debe insertar en el listado fuente.

SKIP2
Especifica dos líneas en blanco que deben insertarse en el listado fuente.

SKIP3
Especifica tres líneas en blanco que deben insertarse en el listado fuente.

SKIP1, SKIP2o SKIP3 se puede escribir en cualquier lugar del Área A o del Área B y se puede terminar con un punto de separación. Debe ser la única sentencia de la línea.

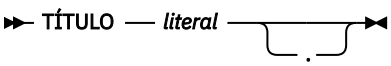
Las sentencias SKIP deben estar incluidas en un origen de programa. Por ejemplo, en el caso de aplicaciones por lotes, se debe colocar una sentencia SKIP1, SKIP2o SKIP3 entre la sentencia PROCESS (CBL) y el final del programa (o el marcador END PROGRAM, si se especifica)..

Sentencia TITLE

La sentencia TITLE especifica un título que se imprimirá en la parte superior de cada página del listado fuente producido durante la compilación.

Si no se encuentra ninguna sentencia TITLE, se genera un título que contiene la identificación del compilador y el nivel de release actual. El título se justifica a la izquierda en la línea de título.

Formato



literal

Debe ser un literal alfanumérico, un literal DBCS o un literal nacional y puede ir seguido de un punto separador.

No debe ser una constante figurativa.

Además del título predeterminado o elegido, el lado derecho de la línea de título contiene los elementos siguientes:

- Para programas, el nombre del programa del párrafo PROGRAM-ID para el programa más externo. (Este espacio está en blanco en las páginas que preceden al párrafo PROGRAM-ID para el programa más externo.)
- Para clases, el nombre de la clase del párrafo CLASS-ID.
- Número de página actual .
- Fecha y hora de compilación.

La sentencia TITLE:

- Fuerza una nueva página inmediatamente, si la opción de compilador SOURCE está en vigor
- No está impreso en sí mismo en el listado fuente
- No tiene ningún otro efecto en la compilación
- No tiene ningún efecto en la ejecución del programa
- No se puede continuar en otra línea
- Puede aparecer en cualquier lugar de cualquiera de las divisiones

Se genera una línea de título para cada página del listado generado por la opción LIST. Esta línea de título utiliza la última sentencia TITLE encontrada en las sentencias fuente o el valor predeterminado.

La palabra TITLE puede empezar en el Área A o en el Área B.

La sentencia TITLE debe estar incorporada en una clase o fuente de programa. Por ejemplo, en el caso de aplicaciones por lotes, la sentencia TITLE debe colocarse entre la sentencia PROCESS (CBL) y el final de la clase (o el marcador END PROGRAM, si se especifica).

Ninguna otra sentencia puede aparecer en la misma línea que la sentencia TITLE.

USE, sentencia

La sentencia USE define las condiciones bajo las cuales se ejecutarán los procedimientos que siguen a la sentencia.

Los formatos para la sentencia USE son:

- EXCEPTION/ERROR declarativo
- Declarativo DEBUGGING

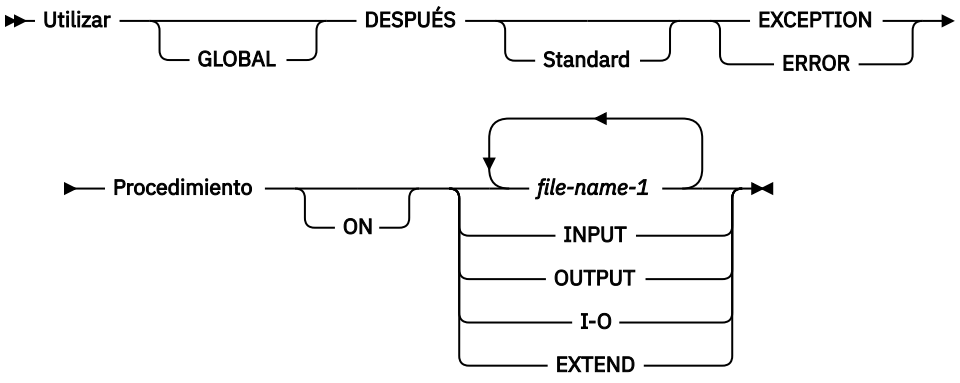
Para obtener información general sobre declarativos, consulte “Declaraciones” en la página 244.

EXCEPTION/ERROR declarativo

La declarativa EXCEPTION/ERROR especifica los procedimientos para la excepción de entrada/salida o el manejo de errores que se van a ejecutar además de los procedimientos del sistema estándar.

Las palabras EXCEPTION y ERROR son sinónimas y se pueden utilizar indistintamente.

Formato 1: sentencia USE para declaración EXCEPTION/ERROR



file-name-1
Válido para todos los archivos. Cuando se especifica esta opción, el procedimiento se ejecuta sólo para los archivos especificados. Ningún nombre de archivo puede hacer referencia a un archivo de clasificación o fusión. Para un archivo determinado, sólo puede especificarse un procedimiento EXCEPTION/ERROR; por lo tanto, la especificación de nombre de archivo no debe provocar solicitudes simultáneas para la ejecución de más de un procedimiento EXCEPTION/ERROR.

Una sentencia declarativa USE AFTER EXCEPTION/ERROR que especifica el nombre de un archivo tiene prioridad sobre una sentencia declarativa que especifica la modalidad de apertura del archivo.

INPUT
Válido para todos los archivos. Cuando se especifica esta opción, se ejecuta el procedimiento para todos los archivos abiertos en modalidad INPUT o en proceso de abrirse en modalidad INPUT que obtienen un error.

OUTPUT
Válido para todos los archivos. Cuando se especifica esta opción, el procedimiento se ejecuta para todos los archivos abiertos en modalidad OUTPUT o en proceso de abrirse en modalidad OUTPUT que obtienen un error.

I-O
Válido para todos los archivos de acceso directo. Cuando se especifica esta opción, se ejecuta el procedimiento para todos los archivos abiertos en modalidad I-O o en proceso de abrirse en modalidad I-O que obtengan un error.

AMPLIAR
Válido para todos los archivos. Cuando se especifica esta opción, el procedimiento se ejecuta para todos los archivos abiertos en modalidad EXTEND o en proceso de abrirse en modalidad EXTEND que obtengan un error.

Se ejecuta el procedimiento EXCEPTION/ERROR:

- Después de completar la rutina de error de entrada/salida definida por el sistema, o
- Tras el reconocimiento de una condición INVALID KEY o AT END cuando no se ha especificado una frase INVALID KEY o AT END en la sentencia de entrada/salida, o
- Tras el reconocimiento de una condición definida por IBMque hace que la clave de estado de archivo 1 se establezca en 9. (Consulte el apartado “Clave de estado de archivo” en la página 285.)

Después de la ejecución del procedimiento EXCEPTION/ERROR, el control se devuelve a la rutina de invocación en el sistema de control de entrada/salida. Si el valor de estado de entrada/salida no indica un error crítico de entrada/salida, el sistema de control de entrada/salida devuelve el control a la siguiente sentencia ejecutable después de la sentencia de entrada/salida cuya ejecución ha causado la excepción.

Se activa un procedimiento EXCEPTION/ERROR aplicable cuando se produce un error de entrada/salida durante la ejecución de una sentencia READ, WRITE, REWRITE, START, OPEN, CLOSE o DELETE. Para determinar qué condiciones son errores, consulte “Recursos de proceso comunes” en la página 284.

Un procedimiento declarativo no debe hacer referencia a un procedimiento no declarativo.

Una sentencia PERFORM en un procedimiento no declarativo puede hacer referencia a un procedimiento declarativo; de lo contrario, no se debe hacer referencia a un procedimiento declarativo desde un procedimiento no declarativo.

Puede incluir una sentencia que ejecute un procedimiento USE llamado anteriormente que todavía esté en control. Sin embargo, para evitar un bucle infinito, debe estar seguro de que hay una salida eventual en la parte inferior.

Los procedimientos EXCEPTION/ERROR se pueden utilizar para comprobar los valores de clave de estado de archivo siempre que se produzca un error de entrada/salida.

Reglas de prioridad para programas anidados

Se siguen reglas de prioridad especiales cuando los programas están contenidos en otros programas.

Al aplicar estas reglas, sólo se selecciona la primera declarativa calificadora para su ejecución. El orden de prioridad para seleccionar una declaración es:

1. Una declarativa específica de archivo (es decir, una declarativa con el formato USE AFTER ERROR ON *file-name-1*) dentro del programa que contiene la sentencia que ha causado la condición de calificación.
2. Una declarativa específica de modalidad (es decir, una declarativa con el formato USE AFTER ERROR ON INPUT) dentro del programa que contiene la sentencia que ha causado la condición de calificación.
3. Declarativo específico de archivo que especifica la frase GLOBAL y está dentro del programa que contiene directamente el programa que se ha examinado por última vez para una declarativa calificadora.
4. Declarativo específico de modalidad que especifica la frase GLOBAL y está dentro del programa que contiene directamente el programa que se examinó por última vez para una condición de calificación.

Los pasos 3 y 4 se repiten hasta que el último programa examinado sea el programa más externo, o hasta que se haya encontrado una declaración calificadora.

Declarativo DEBUGGING

Las secciones de depuración sólo están permitidas en el programa más externo; no son válidas en programas anidados. Las secciones de depuración nunca se desencadenan mediante procedimientos contenidos en programas anidados.

Las secciones de depuración no están permitidas en:

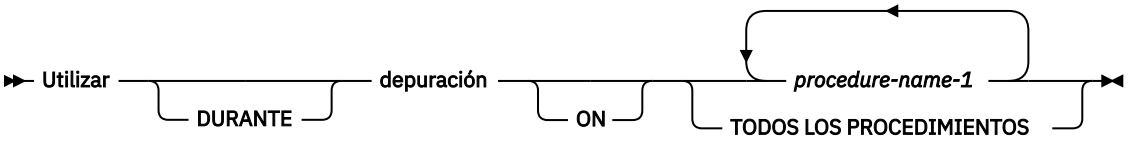
- Un programa definido con el atributo recursivo

La cláusula WITH DEBUGGING MODE del párrafo SOURCE-COMPUTER activa todas las secciones y líneas de depuración que se han compilado en el código de objeto. Consulte Apéndice C, “Depuración de lenguaje fuente”, en la página 563 para obtener detalles adicionales.

Cuando se suprime la modalidad de depuración no especificando la cláusula WITH DEBUGGING MODE, se inhiben todos los procedimientos declarativos USE FOR DEBUGGING y todas las líneas de depuración.

La ejecución automática de una sección de depuración no se debe a una sentencia que aparece en una sección de depuración.

Formato 2: sentencia USE para declarativa DEBUGGING



USO PARA DEPURACIÓN

Todas las sentencias de depuración deben escribirse juntas en una sección inmediatamente después de la cabecera DECLARATIVES.

Excepto para la propia sentencia USE FOR DEBUGGING, dentro del procedimiento de depuración no debe haber ninguna referencia a ningún procedimiento no declarativo.

procedure-name-1

No se debe definir en una sesión de depuración.

Tabla 60 en la página 528 muestra, para cada opción válida, los puntos durante la ejecución cuando se ejecutan los procedimientos USE FOR DEBUGGING.

Cualquier nombre de procedimiento dado sólo puede aparecer en una frase USE FOR DEBUGGING, y sólo una vez en esa frase. Todos los procedimientos deben aparecer en el programa más externo.

TODOS LOS PROCEDIMIENTOS

procedure-name-1 no debe especificarse en ninguna sentencia USE FOR DEBUGGING. La frase ALL PROCEDURES sólo puede especificarse una vez en un programa. Sólo los procedimientos contenidos en el programa más externo desencadenarán la ejecución de la sección de depuración.

Tabla 60. Ejecución de declarativos de depuración

Operando USE FOR DEBUGGING	**Tras la ejecución de lo siguiente, los procedimientos USE FOR DEBUGGING se ejecutan inmediatamente**
procedure-name-1	Antes de cada ejecución del procedimiento con nombre Después de la ejecución de una sentencia ALTER que hace referencia al procedimiento especificado
TODOS LOS PROCEDIMIENTOS	Antes de cada ejecución de cada procedimiento de no depuración en el programa más externo Después de la ejecución de cada sentencia ALTER en el programa más externo (excepto las sentencias ALTER en los procedimientos declarativos)

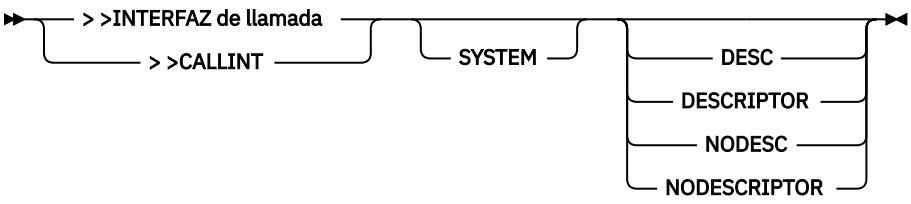
Capítulo 22. Directivas de compilador

Una *directiva de compilador* es una sentencia que hace que el compilador realice una acción específica durante la compilación.

INTERFAZ de llamada

La directiva CALLINTERFACE especifica el convenio de interfaz para las sentencias CALL . El convenio especificado permanece en vigor hasta que se encuentra otra directiva CALLINTERFACE en el origen.

Formato



SYSTEM
Especifica que el convenio de interfaz de llamada es el convenio de enlace de sistema estándar de la plataforma.

DESC, DESCRIPTOR
Indica que se pasa un descriptor de argumento para cada argumento en una sentencia CALL.

NODESC, NODESCRIPTOR
Indica que no se pasan descriptores de argumento para ningún argumento en una sentencia CALL. NODESC/NODESCRIPTOR es el valor predeterminado.

Especifique > >CALLINTERFACE sólo en la división de procedimiento.

Las posiciones de las sentencias CALL relativas a la directiva CALLINTERFACE se reconocen después de cualquier proceso de sentencias COPY y REPLACE. Por ejemplo, las sentencias CALL y las directivas CALLINTERFACE del texto COPY se procesan mediante las reglas especificadas para la directiva CALLINTERFACE.

Si especifica sintaxis en la directiva CALLINTERFACE que no está soportada, se ignora toda la directiva CALLINTERFACE.

Sintaxis y reglas generales

- Debe especificar > >CALLINTERFACE en una línea por sí mismo, en el Área B.
- No puede especificar > >CALLINTERFACE:
  - Dentro de una sentencia COPY o REPLACE
  - Entre las líneas de una serie de caracteres continuada
  - En medio de una sentencia COBOL
- La especificación > >CALLINTERFACE está limitada a la unidad de compilación actual de .
- La sentencia REPLACE y la frase REPLACE de la sentencia COPY no afectan a la directiva CALLINTERFACE.

Compilación condicional

La compilación condicional proporciona una forma de incluir u omitir líneas seleccionadas de código fuente en función de los valores de literales especificados por la directiva DEFINE. De esta forma, puede crear varias variantes del mismo programa sin necesidad de mantener rutas de origen separadas.

Las directivas de compilador que se utilizan para la compilación condicional son la directiva DEFINE, la directiva EVALUATE y la directiva IF. La directiva DEFINE se utiliza para definir variables de compilación a las que se hace referencia en las directivas EVALUATE e IF para seleccionar líneas de código fuente que deben incluirse u omitirse en un grupo de compilación.

Las directivas de compilación condicionales se procesan de acuerdo con las reglas siguientes:

- Dentro de un archivo fuente, si aparece una directiva de compilación condicional antes de una sentencia COPY o REPLACE, se procesa antes de que se procese la sentencia COPY o REPLACE. Esto significa que pueden utilizarse directivas de compilación condicionales para excluir las sentencias COPY y REPLACE de un programa.
- Las directivas de compilación condicionales no se ven afectadas por sustituciones realizadas como resultado de sentencias REPLACE o la frase REPLACE de sentencias COPY.
- Las directivas de compilación condicional pueden aparecer en los libros de copias.

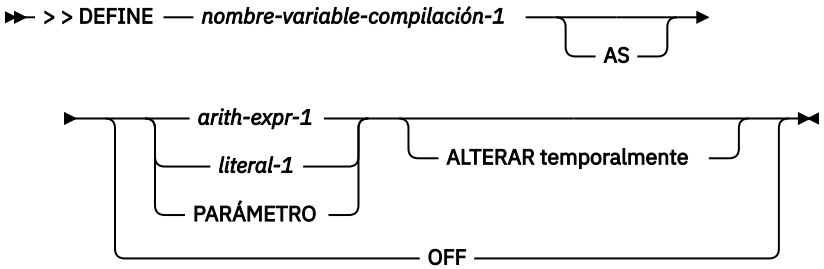
Nota: Las directivas de compilación condicionales pueden incluirse en un archivo que contiene la sentencia BASIS, pero en ese archivo, las directivas de compilación condicionales no controlan la inclusión o exclusión del origen de ese archivo. En su lugar, las directivas de compilación condicional se procesarán como cualquier otra línea fuente en el archivo BASIS que no sean sentencias INSERT o DELETE, y esas directivas se pasarán a través del origen que se está ensamblando para procesarse más adelante durante la fase de biblioteca.

Referencias relacionadas
"DEFECTO" en la página 530
"EVALUAR" en la página 531
"SI" en la página 534
Ejemplo: salida de compilación condicional (*COBOL for Linux en x86 Guía de programación*)

DEFECTO

La directiva DEFINE define o anula la definición de una variable de compilación. Las variables de compilación se pueden utilizar dentro de cualquiera de las directivas de compilación condicionales (DEFINE, EVALUATE e IF). La variable de compilación se trata como una referencia simbólica al valor literal que representa actualmente.

Formato



> > DEFECTO
Debe empezar en una línea nueva en el área A o B y debe especificarse por completo en esa línea.

compilation-variable-name-1
No debe ser igual que una palabra clave de directiva de compilador condicional y no debe ser uno de los nombres de variable de compilación predefinidos.

Si una directiva DEFINE no especifica la frase OFF o OVERRIDE, debe cumplirse una de las condiciones siguientes:

- *compilation-variable-name-1* no se ha declarado anteriormente dentro del mismo grupo de compilación.
- La directiva DEFINE anterior que hace referencia a *compilation-variable-name-1* debe haberse especificado con la frase OFF.
- La directiva DEFINE anterior que hace referencia a *compilation-variable-name-1* debe haber especificado el mismo valor.

literal-1
Debe ser uno de los elementos siguientes:

- Un literal alfanumérico, que se puede especificar como un literal alfanumérico normal ('abcd ') o como un literal hexadecimal (x'F1F2F3'). Los literales nacionales, los literales DBCS y los literales alfanuméricos terminados en nulo (literales Z) no están soportados.
- Un literal entero.
- Un literal booleano (solo se da soporte a B '0' y B '1').

arith-expr-1
Debe formarse de acuerdo con las reglas de expresión aritmética tal como se describe en "Expresiones aritméticas en tiempo de compilación" en la página 537.

Normas generales

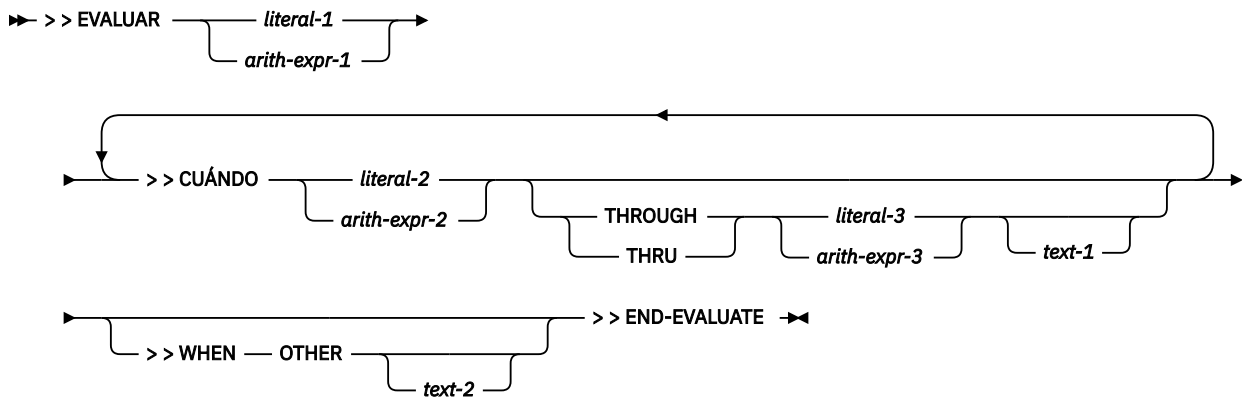
- Las directivas DEFINE que aparecen en el código que se omite como resultado de otras directivas de compilación condicional no se procesan.
- En el texto que sigue a una directiva DEFINE que define *compilation-variable-name-1* y no incluye la frase OFF, *compilation-variable-name-1* se puede utilizar en una directiva de compilación condicional siempre que se permita un literal de la categoría asociada con el nombre, incluyendo una condición definida y una condición booleana.
- En el texto que sigue a una directiva DEFINE que especifica *compilation-variable-name-1* con la frase OFF, *compilation-variable-name-1* sólo se puede utilizar en una condición definida, a menos que *compilation-variable-name-1* se redefina en una directiva DEFINE posterior sin la frase OFF.
- Si se especifica la frase OVERRIDE, *compilation-variable-name-1* se establece incondicionalmente para hacer referencia al valor del operando especificado.
- Si se especifica la frase AS PARAMETER, el valor al que hace referencia *compilation-variable-name-1* se obtiene de una opción DEFECTO para *compilation-variable-name-1*, si existe dicha opción. Si no hay ninguna opción DEFINE para *compilation-variable-name-1*, *compilation-variable-name-1* no está definido.
- Si se especifica *arith-expr-1* , *arith-expr-1* se evalúa de acuerdo con "Expresiones aritméticas en tiempo de compilación" en la página 537, y *compilation-variable-name-1* hace referencia al valor resultante.
- Si se especifica *literal-1* , *compilation-variable-name-1* hace referencia a *literal-1*.

Referencias relacionadas
"Condiciones definidas" en la página 536
"Variables de compilación predefinidas" en la página 538
DEFINE (*COBOL for Linux en x86 Guía de programación*)

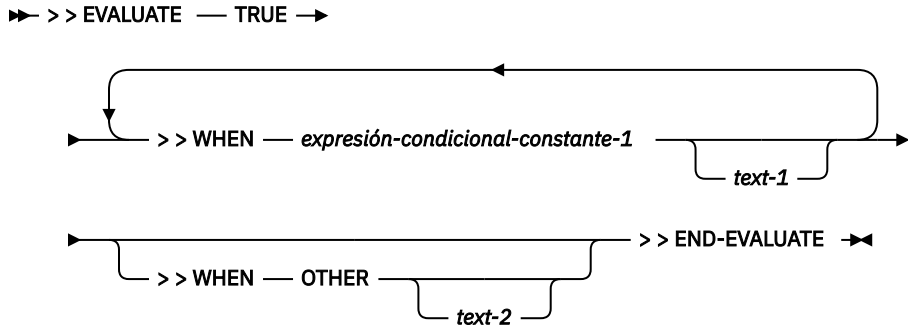
EVALUAR

La directiva EVALUATE proporciona un método de varias ramificaciones para elegir las líneas fuente que se van a incluir en un grupo de compilación.

Formato 1



Formato 2



A efectos descriptivos, en este tema:

- *operand-1* hace referencia a *literal-1* o *arith-expr-1* en formato 1, y a la palabra clave TRUE en el formato 2.
- *operand-2* hace referencia a *literal-2* o *arith-expr-2* en formato 1, y a *constant-conditional-expression-1* en el formato 2.
- *operand-3* hace referencia a *literal-3* o *arith-expr-3* en el formato 1.

Todos los formatos:

> > EVALUAR, > > WHEN, > > WHEN OTHER, > > END-EVALUATE
Debe empezar en una línea nueva en el área A o B y debe especificarse por completo en esa línea.

text-1, text-2
Debe empezar en una línea nueva y puede constar de varias líneas.

Puede ser cualquier tipo de líneas fuente, incluidas las directivas de compilador. *text-1* o *text-2* también puede incluir sentencias COPY.

Las frases de una directiva EVALUATE deben especificarse todas en el mismo texto de biblioteca o todas en el texto fuente. A efectos de esta regla, *text-1* y *text-2* no se consideran frases de la directiva EVALUATE. Una directiva EVALUATE anidada especificada en *text-1* o *text-2* se considera una nueva directiva EVALUATE.

Formato 1:

> > EVALUAR
Todos los operandos de una directiva EVALUATE deben ser de la misma categoría. Para esta regla, una expresión aritmética es de categoría numérica.

literal-1*, *arith-expr-1
Temas de selección.

literal-2*, *literal-3*, *arith-expr-2*, *arith-expr-3
Objetos de selección.

A TRAVÉS, A TRAVÉS
Las palabras THROUGH y THRU son equivalentes. Si se especifica la frase THROUGH, todos los sujetos de selección y los objetos de selección deben ser de categoría numérica.

arith-expr-1*, *arith-expr-2*, *arith-expr-3
Debe formarse de acuerdo con las reglas de expresión aritmética tal como se describe en "Expresiones aritméticas en tiempo de compilación" en la página 537.

Formato 2:

constant-conditional-expression-1
Debe formarse de acuerdo con las reglas de expresión condicional constante tal como se describe en "Expresiones condicionales constantes" en la página 536.

Normas generales

Todos los formatos:

- *text-1* y *text-2* no forman parte de la línea de directiva de compilador EVALUATE. *text-1* y *text-2* que están en la primera rama verdadera de la sentencia EVALUATE están sujetos a las reglas de coincidencia y sustitución de la sentencia COPY y la sentencia REPLACE.
- Si se alcanza la frase END-EVALUATE sin ninguna frase WHEN que se evalúe como TRUE, o sin encontrar una frase WHEN OTHER, todas las líneas de *text-1* asociadas con todas las frases WHEN se omiten del texto resultante.

Formato 1:

- El sujeto de selección se compara con los valores especificados en cada frase WHEN a su vez de la siguiente manera:
  - Si no se especifica la frase THROUGH, se devuelve un resultado TRUE si el asunto de selección es igual a *operand-2*.
  - Si se especifica la frase THROUGH, se devuelve un resultado TRUE si el asunto de selección es mayor o igual que *operand-2* y menor o igual que *operand-3*.
- Si una frase WHEN se evalúa como TRUE, todas las líneas de *text-1* asociadas con esa frase WHEN se incluyen en el texto resultante. Todas las líneas de *text-1* asociadas con otras frases WHEN de la directiva EVALUATE y todas las líneas de *text-2* asociadas con una frase WHEN OTHER se omiten del texto resultante.
- Si ninguna frase WHEN se evalúa como TRUE, todas las líneas de *text-2* asociadas con la frase WHEN OTHER, si se especifica, se incluyen en el texto resultante. Todas las líneas de *text-1* asociadas con las otras frases WHEN se omiten del texto resultante.
- Si *literal-1* es un literal alfanumérico, se utiliza una comparación de caracteres por caracteres para la igualdad basada en el valor binario de la codificación de cada carácter. Si los literales son de longitud desigual, no son iguales.

Formato 2:

- Para cada frase WHEN a su vez, la *constant-conditional-expression-1* se evalúa de acuerdo con las reglas de "Expresiones condicionales constantes" en la página 536.
- Si una frase WHEN se evalúa como TRUE, todas las líneas de *text-1* asociadas con esa frase WHEN se incluyen en el texto resultante. Todas las líneas de *text-1* asociadas con otras frases WHEN de la directiva EVALUATE y todas las líneas de *text-2* asociadas con una frase WHEN OTHER se omiten del texto resultante.

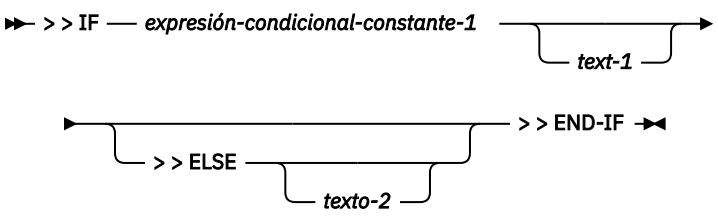
- Si ninguna frase WHEN se evalúa como TRUE, todas las líneas de *text-2* asociadas con la frase WHEN OTHER, si se especifica, se incluyen en el texto resultante. Todas las líneas de *text-1* asociadas con otras frases WHEN se omiten del texto resultante.

Referencias relacionadas
"sentencia COPY" en la página 508
"sentencia REPLACE" en la página 520

SI

La directiva IF proporciona una compilación condicional unidireccional o bidireccional.

Formato



> > SI, > > ELSE, > > END-IF
Debe empezar en una línea nueva en el área A o B y debe especificarse por completo en esa línea.

constant-conditional-expression-1
Debe formarse de acuerdo con las reglas de expresión condicional constante tal como se describe en "Expresiones condicionales constantes" en la página 536.

text-1, text-2

Debe empezar en una línea nueva y puede constar de varias líneas.

Puede ser cualquier tipo de líneas fuente, incluidas las directivas de compilador. *text-1* o *text-2* también puede incluir sentencias COPY.

Las frases de una directiva IF determinada deben especificarse todas en el mismo texto de biblioteca o todas en texto fuente. A efectos de esta regla, *text-1* y *text-2* no se consideran frases de la directiva IF. Una directiva IF anidada especificada en *text-1* o en *text-2* se considera una nueva directiva IF.

Reglas generales

- *text-1* y *text-2* no forman parte de la línea de directiva de compilador IF. El texto de la rama verdadera de la directiva IF (*text-1* o *text-2*) está sujeto a las reglas de coincidencia y sustitución de la sentencia COPY y la sentencia REPLACE.
- Si *constant-conditional-expression-1* se evalúa como TRUE, todas las líneas de *text-1* se incluyen en el texto resultante y todas las líneas de *text-2* se omiten del texto resultante.
- Si *constant-conditional-expression-1* se evalúa como FALSE, todas las líneas de *text-2* se incluyen en el texto resultante y todas las líneas de *text-1* se omiten del texto resultante.

Referencias relacionadas
"sentencia COPY" en la página 508
"sentencia REPLACE" en la página 520

Ejemplos de compilación condicional

Ejemplo 1: importar una variable de compilación booleana desde fuera del origen y probarla

Supongamos que DEFINE (DEBUG) está en vigor. En este caso, DEBUG hace referencia a una variable de compilación de categoría booleana con un valor de parámetro de B '1'.

```
>>DEFINE DEBUG AS PARAMETER
. . .
>>IF DEBUG IS DEFINED
     display “DEBUG: debugging mode is on”
>>END-IF
```

Ejemplo 2: importar un valor de variable numérica desde fuera del origen y probarlo

Supongamos que DEFINE (VAR1=10) está en vigor:

```
>>DEFINE VAR1 AS PARAMETER
. . .
>>DEFINE VAR2 AS VAR1 + 2
. . .
>>IF VAR2 < 12
     compute x = x + 1 *> this code should NOT be included
>>ELSE
     compute x = x – 1 *> this code should be included
>>END-IF
```

Ejemplo 3: utilizar la directiva EVALUATE de formato 1 con variables de compilación numéricas

```
>>DEFINE VAR1 AS 6
>>DEFINE VAR2 AS 1
. . .
>>EVALUATE VAR1
>>WHEN VAR2 + 2
       compute x = x + 1 *> this code should NOT be included
>>WHEN 4 THRU 8
       compute x = x – 1 *> this code should be included
>>WHEN OTHER
       compute x = x * 2 *> this code should NOT be included
>>END-EVALUATE
```

Ejemplo 4: utilizar la directiva EVALUATE de formato 2 con variables de compilación alfanuméricas

```
>>DEFINE VAR1 AS 'MOO'
. . .
>>EVALUATE TRUE
>>WHEN VAR2 IS DEFINED
       compute x = x + 1 *> this code should NOT be included
>>WHEN VAR1 IS EQUAL TO 'GOO' OR VAR1 IS EQUAL TO 'MOO'
       compute x = x – 1 *> this code should be included
>>END-EVALUATE
```

Ejemplo 5: utilizar OVERRIDE y OFF en la directiva DEFINE

```
>>DEFINE VAR AS 12
. . .
>>DEFINE VAR OFF
. . .
>>IF VAR IS DEFINED
     compute x = x + 1 *> this code should NOT be included
>>ELSE
     compute x = x - 1 *> this code should be included
>>END-IF
. . .
>>DEFINE VAR AS 16
. . .
>>DEFINE VAR AS VAR - 2 OVERRIDE
```

```
. . .
>>IF VAR IS EQUAL TO 16
     compute x = x + 1 *> this code should NOT be included
>>ELSE
     compute x = x - 1 *> this code should be included
>>END-IF
```

Ejemplo 6: uso general de literales booleanos y variables de compilación

```
>>DEFINE B1 B'1' *> B1 is category boolean
>>DEFINE B2 B'0' *> B2 is category boolean
. . .
>>IF B1 AND B2
     display “Both B1 and B2 are true” *> not included
>>ELSE
  >>IF B1
       display “Only B1 is true” *> included
  >>ELSE
    >>IF B2
         display “Only B2 is true” *> not included
    >>ELSE
         display “Neither B1 nor B2 is true” *> not included
    >>END-IF
  >>END-IF
>>END-IF
```

Expresiones condicionales constantes

Una expresión condicional constante es una expresión que se especifica en las directivas de compilación condicional y se evalúa durante el proceso de dichas directivas para determinar el texto que se incluye en el programa resultante.

Nota: En este tema, los "literales" también incluyen variables de compilación, lo que significa que puede utilizar variables de compilación en expresiones condicionales constantes.

Una expresión condicional constante será uno de los siguientes elementos:

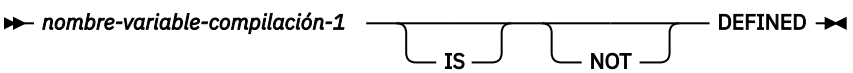
- Una condición de relación en la que ambos operandos son literales o expresiones aritméticas que sólo contienen términos literales. La condición se regirá por las normas relativas a las condiciones de relación, con las siguientes adiciones:
  - Los operandos serán de la misma categoría. Una expresión aritmética es de la categoría numérica.
  - Si se especifican literales y no son literales numéricos, el operador relacional será “IS EQUAL TO”, “IS NOT EQUAL TO”, “IS =”, “IS NOT =”o “IS <>”.
- Una condición definida.
- Una condición booleana.
- Una condición compleja formada combinando las formas anteriores de condiciones simples en condiciones complejas utilizando AND, OR y NOT. No se especificarán Condiciones de relación combinadas abreviadas .

Referencias relacionadas
“Condiciones de relación” en la página 255
“Condiciones definidas” en la página 536
“Condiciones de relación combinadas abreviadas” en la página 272

Condiciones definidas

Una expresión de condición definida comprueba si se ha definido una variable de compilación.

Formato



compilation-variable-name-1
No debe ser lo mismo que una palabra clave de directiva de compilador condicional.

ESTÁ DEFINIDO
Una condición definida que utiliza la sintaxis IS DEFINED se evalúa como TRUE si se ha definido *compilation-variable-name-1* .

Si una condición definida hace referencia a una variable de compilación que se ha definido mediante una opción de compilador DEFINE, pero antes de la condición definida en el programa no hay una directiva DEFINE correspondiente con la frase AS PARAMETER ni una directiva DEFINE sin la frase OFF para la variable de compilación, la condición definida para la variable de compilación se evalúa como FALSE.

NO ESTÁ DEFINIDO
Una condición definida que utiliza la sintaxis IS NOT DEFINED se evalúa como TRUE si el *compilation-variable-name-1* no está definido.

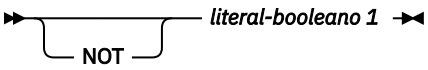
Una variable de compilación cuya definición más reciente es a través de una directiva DEFINE con la frase OFF se considera que no está definida.

Referencias relacionadas
"Variables de compilación predefinidas" en la página 538
DEFINE (*COBOL for Linux en x86 Guía de programación*)

Condiciones booleanas

Una condición booleana determina si un literal booleano es verdadero o falso.

Formato



boolean-literal-1
Se evalúa como true si es B '1' y se evalúa como false si es B '0'.

La condición NOT *boolean-literal-1* se evalúa en el valor de verdad inverso de *boolean-literal-1*.

Referencias relacionadas
DEFINE (*COBOL for Linux en x86 Guía de programación*)

Expresiones aritméticas en tiempo de compilación

Puede especificar una expresión aritmética de tiempo de compilación en las directivas DEFINE y EVALUATE y como parte de una expresión condicional constante, como las que se encuentran en las directivas IF o las frases WHEN de las directivas EVALUATE.

Nota: En este tema, los "literales" también incluyen variables de compilación, lo que significa que puede utilizar variables de compilación en expresiones aritméticas de tiempo de compilación.

Una expresión aritmética en tiempo de compilación sigue las reglas de expresión aritmética habituales, con las excepciones siguientes:

- No se especificará el operador de exponenciación.

- Todos los operandos serán literales numéricos enteros o expresiones aritméticas en las que todos los operandos son literales numéricos enteros.
- La expresión se especificará de tal manera que no se produzca una división por cero.
- Los resultados intermedios se calculan de acuerdo con las reglas descritas en *Datos de punto fijo y resultados intermedios* en *COBOL for Linux en x86 Guía de programación*. Para ello, los operandos enteros de expresiones aritméticas en tiempo de compilación se pueden considerar números de coma fija con 0 dígitos decimales. El valor de la opción ARITH (COMPAT | EXTEND) se tiene en cuenta al decidir cuántos dígitos de precisión se deben mantener para los resultados intermedios.

Referencias relacionadas
"Expresiones aritméticas" en la página 246
ARITH (*COBOL for Linux en x86 Guía de programación*)

Variables de compilación predefinidas

Existen variables de compilación que el compilador define automáticamente. Se puede hacer referencia a estas variables de compilación listadas en este tema en directivas de compilación condicionales siempre que se permita una variable de compilación.

Tabla 61. Variables de compilación predefinidas

Nombre de variable de compilación predefinida	**Descripción**	**Valor**
IGY-ADDR	Indica si se genera un programa de objeto de 32 bits o de 64 bits.	32 si la opción -q32 o ADDR (32) está en vigor; 64 si la opción -q64 o ADDR (64) está en vigor.
IGY-BINARIO	Indica la modalidad de endianness de los elementos de datos binarios (USAGE BINARY, USAGE COMP y USAGE COMP-4).	El valor de la opción BINARY utilizada para compilar el programa: • 0 representa BINARY (NATIVE) • 1 representa BINARY (BE) • 2 representa BINARY (LE)
IGY-CICS	Indica si se aceptan las sentencias CICS incorporadas.	B '1' si la opción de compilador CICS está en vigor; B '0' en caso contrario.
IGY-COMPILADOR-VRM	Indica la versión del compilador.	Un entero con el formato VVRRMM, donde: • VV representa el número de versión. • RR representa el número de release. • MM representa el número de modificación. Por ejemplo, la versión de compilador 1.2.0 tiene un valor IGY-COMPILER-VRM de 010200.
IGY-DYNAM	Indica si los programas invocados mediante la sentencia literal CALL se cargarán o se suprimirán dinámicamente en tiempo de ejecución.	B '1' si la opción de compilador DYNAM está en vigor; B '0' en caso contrario.

Tabla 61. Variables de compilación predefinidas (continuación)		
Nombre de variable de compilación predefinida	**Descripción**	**Valor**
IIGY-FLOAT	Indica la representación de los elementos de datos de coma flotante (USAGE COMP-1 y USAGE COMP-2).	El valor de la opción FLOAT utilizada para compilar el programa: • 0 representa FLOAT (NATIVE) • 1 representa FLOAT (BE) • 2 representa FLOAT (LE)
IGY-OPTIMIZAR	Indica el nivel de optimización.	El nivel de optimización que se ha utilizado para compilar el programa: 0, 1 o 2. 0, 1 o 2 representa NOOPTIMIZE, OPTIMIZE (STD) y OPTIMIZE (FULL) respectivamente.
IGY-SQL	Indica si el proceso de sentencias de SQL incorporado está habilitado.	B '1' si la opción de compilador SQL está en vigor; B '0' en caso contrario.
IGY-UTF16	Indica la modalidad de endianness de los elementos de datos nacionales (USAGE NATIONAL).	El valor de la opción UTF16 utilizada para compilar el programa: • 0 representa UTF16(NATIVE) • 1 representa UTF16(BE) • 2 representa UTF16(LE)

Referencias relacionadas
“DEFECTO” en la página 530
DEFINE (*COBOL for Linux en x86 Guía de programación*)

Apéndice A. Extensiones de IBM

Las extensiones de IBM son características, reglas de sintaxis o comportamientos definidos por IBM en lugar de por los estándares COBOL.

Los estándares COBOL se listan en Apéndice G, “Especificaciones de la industria”, en la página 587.

Tabla 62 en la página 541 lista las extensiones IBM con una breve descripción. El comportamiento estándar se muestra entre corchetes, [], cuando el comportamiento estándar no es obvio. Las extensiones se describen con más detalle en este documento, pero no se identifican más como extensiones.

Muchas extensiones de IBM se distinguen del lenguaje estándar por su sintaxis. Para otros, puede utilizar opciones de compilador para elegir entre el comportamiento estándar y de extensión. Generalmente, las opciones de compilador relacionadas se anotan en las reglas detalladas. Para obtener información sobre las opciones del compilador, consulte *Opciones de compilador* en *COBOL for Linux en x86 Guía de programación*.

Si un elemento se lista como una extensión, todas las reglas relacionadas también son extensiones. Por ejemplo, USAGE DISPLAY-1 para caracteres DBCS se lista como una extensión; sus muchos usos en sentencias y cláusulas también son extensiones, pero no se listan por separado.

Tabla 62. ***IBM***

Área de idioma	**Elementos de extensión**
Palabras COBOL	Palabras definidas por el usuario escritas en multibyte caracteres Nombres de sistema escritos en caracteres multibyte Las palabras definidas por el usuario pueden incluir un subrayado, pero no como primer carácter.
Soporte de caracteres nacionales (soporte Unicode)	Soporte para UTF-16 con USAGE NATIONAL Bonificación de UTF-8 con USAGE DISPLAY Uso NACIONAL para categorías de datos nacionales, editadas a nivel nacional, numéricas, editadas numéricas, decimales externos y coma flotante externa Cláusula GROUP-USAGE con la frase NATIONAL Literales nacionales (básico y hexadecimal) Valor de carácter nacional para constantes figurativas SPACE, ZERO, QUOTE, HIGH-VALUES, LOW-VALUES, ALL literal Funciones intrínsecas para la conversión de datos: • VISUALIZAR-DE • NACIONAL-DE Correlación de casos ampliada con las funciones UPPER-CASE y LOWER-CASE

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
Elementos implícitos	Registros especiales: • DIRECCIÓN DE • LONGITUD DE • CÓDIGO-RETORNO • MAYÚS-IN • MAYÚS-OUT • SORT-CONTROL • TAMAÑO-NÚCLEO-CLASIFICACIÓN • TAMAÑO DE ARCHIVO DE CLASIFICACIÓN • ORDENAR-MENSAJE • TAMAÑO-MODALIDAD-CLASIFICACIÓN • ORDENAR-DEVOLVER • RECUENTO • CUANDO-COMPILADO • CÓDIGO XML • XML-EVENT • XML-NTEXT • XML-TEXTO
Constantes figurativas	Selección de apóstrofo (') como valor de la constante figurativa QUOTE NULL y NULLS para punteros y referencias de objeto

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
Literales	Uso del apóstrofo (') como alternativa a las comillas (") en los delimitadores de apertura y cierre Caracteres de byte único y multibyte mezclados en literales alfanuméricos (literales mixtos) Notación hexadecimal para literales alfanuméricos, definida mediante delimitadores de apertura X" y X' Literales alfanuméricos terminados en nulo, definidos por delimitadores de apertura Z" y Z' Literales DBCS, definidos mediante delimitadores de apertura N", N', G"y G'. N" y N' se definen como DBCS cuando la opción de compilador NSYMBOL (DBCS) está en vigor. Literales alfanuméricos consecutivos (codificando dos literales alfanuméricos consecutivos terminando el primer literal en la columna 72 de una línea continua y comenzando el siguiente literal con una comilla en la línea de continuación) Literales nacionales N", N', NX", NX' para almacenar contenido literal como caracteres nacionales. N" y N' se definen como nacionales cuando la opción de compilador NSYMBOL (NATIONAL) está en vigor. Literales numéricos de punto fijo de 19 a 31 dígitos. [El 85 COBOL Estándar especifica un máximo de 18 dígitos.] Literales numéricos de coma flotante
Comentarios	Líneas de comentario antes de la cabecera IDENTIFICATION DIVISION Comentar líneas y entradas de comentario que contienen caracteres de varios bytes Comentarios en línea
Indexación y suscripción	Cómo hacer referencia a una tabla con un nombre de índice definido para una tabla diferente Especificación de un literal entero con signo positivo después del operador + o-en la suscripción relativa
Extensiones de idioma del milenio y campos de fecha.	Cláusula DATE FORMAT Campos de fecha con ventana, campos de fecha ampliados, campos de fecha de año a último, campos de fecha compatibles, formatos de fechay ventana de siglo Las siguientes funciones intrínsecas: • FECHA DE FECHA • FECHA • VENTANA • DATE-TO-AAAAMMDD • DAY-TO-AAAADDD • YEAR-TO-YYYY
Formato de referencia	Formato de origen ampliado

*Tabla 62. **IBM** (continuación)*

Área de idioma	Elementos de extensión
DIVISIÓN DE IDENTIFICACIÓN para programas	ID de abreviatura para IDENTIFICACIÓN Cláusula RECURSIVE Un punto de separador opcional después de las cabeceras de párrafo PROGRAM-ID, AUTHOR, INSTALLATION, DATE-WRITTEN y SECURITY. [El 85 COBOL Estándar requiere un punto después de cada una de estas cabeceras de párrafo.] Un punto de separador opcional a continuación del nombre de programa en el párrafo PROGRAM-ID. [85 COBOL Estándar requiere un punto a continuación del nombre de programa.] Un literal alfanumérico para el nombre de programa en el párrafo PROGRAM-ID; el subrayado puede ser el primer carácter; nombre de programa de hasta 160 caracteres de longitud. [El 85 COBOL Estándar requiere que el nombre de programa se especifique como una palabra definida por el usuario.]
Marcadores finales	Nombre de programa en un literal. [El 85 COBOL Estándar requiere que el nombre de programa se especifique como una palabra definida por el usuario.]
Párrafo SPECIAL-NAMES	El orden opcional de las cláusulas. [El 85 COBOL Estándar requiere que las cláusulas se codifiquen en el orden presentado en el diagrama de sintaxis.] Opcionalidad de un periodo posterior a la última cláusula cuando no se codifica ninguna cláusula. [El 85 COBOL Estándar requiere un punto, incluso cuando no hay cláusulas codificadas.] Varias cláusulas CURRENCY SIGN. [El 85 COBOL Estándar permite una cláusula CURRENCY SIGN única.] Frase WITH PICTURE SYMBOL en la cláusula CURRENCY SIGN Signos de moneda de varios caracteres y mayúsculas y minúsculas en la cláusula CURRENCY SIGN (cuando se especifica la frase WITH PICTURE SYMBOL). [El 85 COBOL Estándar sólo permite un carácter, y es tanto el signo de moneda como el símbolo de imagen de moneda. El signo de moneda estándar no debe ser: • El mismo carácter que cualquier símbolo de imagen estándar • Un dígito 0-9 • Uno de los caracteres especiales * +-,; () " =/ • Un espacio] Uso de caracteres alfabéticos en minúsculas como signo de moneda. [El 85 COBOL Estándar sólo permite caracteres en mayúsculas.] Desasignación de la cláusula SYMBOLIC CHARACTERS, cuando una página de códigos de varios bytes se indica mediante el valor de entorno local.

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
INPUT-OUTPUT SECTION, párrafo FILE-CONTROL	Opcionalidad de "FILE-CONTROL". Cuando se especifica INPUT-OUTPUT SECTION, no se especifica ningún párrafo de control de archivos y no hay archivos definidos en la unidad de compilación. [El 85 COBOL Estándar requiere que "FILE-CONTROL." se codificará si "INPUT-OUTPUT SECTION." está codificado.] Opcionalidad del archivo-control-párrafo cuando el "FILE CONTROL." y no hay archivos definidos en la unidad de compilación. [El 85 COBOL Estándar requiere que se codifique un párrafo de control de archivo si "INPUT-OUTPUT SECTION". está codificado.] Frase USING de la cláusula ASSIGN Cláusula PASSWORD El segundo *nombre-datos* en la cláusula FILE STATUS Opcionalidad de RECORD en la cláusula ALTERNATE RECORD KEY. [El 85 COBOL Estándar requiere la palabra RECORD.] La falta de un efecto de cláusula RESERVE en la ejecución Un elemento de datos de clave de registro numérico, editado numéricamente, editado alfanumérico, alfabético, de coma flotante interno, de coma flotante externo, nacional, de edición nacional o DBCS primario o alternativo. [El 85 COBOL Estándar requiere que la clave sea alfanumérica.] Clave de registro primaria o alternativa definida fuera del tamaño mínimo de registro para archivos indexados que contienen registros de longitud variable. [El 85 COBOL Estándar requiere que las claves de registro primarias y alternativas estén dentro del tamaño mínimo de registro.] Acceso secuencial a registros en orden descendente. [El 85 COBOL Estándar sólo proporciona un orden ascendente de acceso.] Un elemento de datos numéricos de uso DISPLAY o NATIONAL en la cláusula FILE STATUS. [El 85 COBOL Estándar requiere un elemento de datos de estado de archivo alfanumérico.] Cláusula ORGANIZATION IS LINE SEQUENTIAL y formato de control de archivo secuencial de línea Literal nacional en la cláusula PADDING CHARACTER
INPUT-OUTPUT SECTION, párrafo I-O-CONTROL	Cláusula APPLY WRITE-ONLY Especificar sólo un nombre de archivo en la cláusula SAME en los formatos secuencial, indexado y de fusión de clasificación de la entrada de control I-O. [85 COBOL Estándar requiere al menos dos nombres de archivo.] Opcionalidad de la palabra clave ON en la cláusula RERUN. [El 85 COBOL Estándar requiere que se codifique ON.] La entrada de control I-O de formato secuencial de línea La cláusula RERUN en la entrada de control I-O de fusión de clasificación

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
DIVISIÓN DE DATOS	SECCIÓN DE ALMACENAMIENTO LOCAL La cláusula GLOBAL en la LINKAGE SECTION Especificar números de nivel que son inferiores a otros números de nivel en el mismo nivel jerárquico en una entrada de descripción de datos. [El 85 COBOL Estándar requiere que a todos los elementos elementales o de grupo del mismo nivel de la jerarquía se les asignen números de nivel idénticos.] Categorías de datos de coma flotante interna, coma flotante externa, DBCS, nacional y editada a nivel nacional. Categoría de datos numérica con uso NACIONAL. Categoría de datos numérica-editada con uso NATIONAL.
SECCIÓN DE ARCHIVO	*nombre-datos* en la cláusula LABEL RECORDS, para especificar etiquetas de usuario Cláusula REGISTRO MODE Entrada de descripción de archivo de formato secuencial de línea
Entrada de descripción de archivo de clasificación/fusión	Las cláusulas siguientes: • BLOQUE CONTIENE • ETIQUETAR REGISTROS • VALOR DE • ENLACE • CÓDIGO-CONJUNTO • CON PIE • LÍNEAS EN
Cláusula VALUE OF	La falta de efecto de cláusula VALUE en la ejecución cuando se especifica bajo una SD
Cláusula DATA RECORDS	Opcionalidad de una entrada de descripción de registro 01 para un *nombre-datos*especificado. [El 85 COBOL Estándar requiere que se especifique un registro 01 con el mismo *nombre-datos* .]
Cláusula LINAGE	Especificación de LINAGE para archivos abiertos en modalidad EXTEND
Entrada de descripción de datos	cláusula DATE-FORMAT
Cláusula BLANK WHEN ZERO	Ortografía alternativa ZEROS y ZEROES para CERO
Cláusula GLOBAL	Especificación de GLOBAL en la SECCIÓN LINKAGE
Frase INDEXED BY	Nombres de índice no exclusivos no referenciados

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
Cláusula OCCURS	Omisión de "*integer-1* TO" para tablas de longitud variable SE PRODUCE COMPLEJO DEPENDIENDO DE. [El 85 COBOL Estándar requiere que una entrada que contenga OCCURS DEPENDING ON vaya seguida únicamente de entradas subordinadas y que ninguna entrada que contenga OCCURS DEPENDING ON esté subordinada a una entrada que contenga OCCURS DEPENDING ON.] Calificación implícita de una clave especificada sin calificadores cuando el nombre de clave no es exclusivo Referencia a una tabla mediante indexación cuando no se especifica ninguna frase INDEXED BY Claves de usos COMPUTATIONAL-1, COMPUTATIONAL-2, COMPUTATIONAL-3, COMPUTATIONAL-4y COMPUTATIONAL-5 en la frase ASCENDING/DESCENDING KEY Aceptación de nombres de índice no exclusivos a los que no se hace referencia
Cláusula PICTURE	Serie de caracteres de imagen que contiene de 31 a 50 caracteres. [El 85 COBOL Estándar permite un máximo de 30 caracteres.] Símbolos de imagen G y N Símbolo de imagen E y formato de imagen de coma flotante externa Codificar un carácter de inserción de coma final o un carácter de inserción de punto final inmediatamente seguido de una coma de separador o un punto y coma de separador en una cláusula PICTURE que no es la última cláusula de una entrada de descripción de datos. [El 85 COBOL Estándar requiere que una cláusula PICTURE que contenga una imagen que termine con una coma o un punto sea la última cláusula de la entrada y que vaya seguida inmediatamente de un punto separador.] Selección de un signo de moneda y un símbolo de moneda con la opción de compilador CURRENCY Símbolos de moneda sensibles a mayúsculas y minúsculas El máximo de 31 dígitos para elementos numéricos de usos DISPLAY y PACKED-DECIMAL y para elementos editados numéricos de USAGE DISPLAY El efecto de la opción de compilador TRUNC en el valor de los elementos de datos descritos con un uso de BINARY, COMPUTATIONAL o COMPUTATIONAL-4
Cláusula REDEFINES	Especificación de REDEFINES de un elemento de datos redefinido En un nivel subordinado, especificar un elemento de datos de redefinición que tenga un tamaño mayor que el tamaño del elemento de datos redefinido
Cláusula SYNCHRONIZED	Especificación de SYNCHRONIZED para una entrada de nivel 01

*Tabla 62. **IBM** (continuación)*

Área de idioma	Elementos de extensión
Cláusula USAGE	Las frases siguientes: • NATIVA • COMP-1 y COMPUTATIONAL-1 • COMP-2 y COMPUTATIONAL-2 • COMP-3 y COMPUTATIONAL-3 • COMP-4 y COMPUTATIONAL-4 • COMP-5 y COMPUTATIONAL-5 • DISPLAY-1 • NACIONAL • PUNTERO • PROCEDIMIENTO-PUNTERO • PUNTERO DE FUNCIÓN Uso de la cláusula SYNCHRONIZED para elementos de uso INDEX
Cláusula VALUE para entradas de nombre de condición	Una cláusula VALUE en el archivo y LINKAGE SECTION distinta de las entradas de nombre de condición Una cláusula VALUE para una entrada de nombre de condición en un grupo que tiene usos distintos de DISPLAY Especificación del rango de valores en una frase THROUGH utilizando una secuencia de clasificación definida por el entorno local VALUE IS NULL y VALUE IS NULLS
DIVISIÓN DE PROCEDIMIENTO	Omisión de un nombre de sección Omisión de un nombre de párrafo cuando se omite un nombre de sección Hacer referencia a elementos de datos en la LINKAGE SECTION sin una frase USING en la cabecera PROCEDURE DIVISION (cuando esos nombres de datos son el operando de una frase ADDRESS OF o ADDRESS OF registro especial) Las sentencias siguientes: • ENTRADA • GOBACK • XML PARSE • XML GENERATE
Cabecera PROCEDURE DIVISION	La frase BY VALUE La frase DEVOLVER Especificación de un elemento de datos en la frase USING cuando el elemento de datos tiene una cláusula REDEFINES en su descripción Especificación de varias instancias de un elemento de datos determinado en la frase USING

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
Procedimientos declarativos	Ejecución de una declaración activa
procedimientos	Especificar *priority-number* como un literal numérico con signo positivo. [El 85 COBOL Estándar requiere un entero sin signo.] Omitir la cabecera de sección después de las declarativas o cuando no hay declarativas. [El 85 COBOL Estándar requiere una cabecera de sección después de "DECLARATIVES." y a continuación de "END DECLARATIVES." sintaxis.] Omitir un *nombre-párrafo* inicial si no hay declarativos. [85 COBOL Estándar requiere un nombre de párrafo en las siguientes circunstancias: • Después de la sentencia USE si hay sentencias en el procedimiento declarativo • Seguir una cabecera de sección fuera de los procedimientos declarativos • Antes de cualquier declaración de procedimiento si no hay declaraciones y el 85 COBOL Estándar requiere que las sentencias procedimentales estén dentro de un párrafo.] Especificar párrafos que no están contenidos en una sección, incluso si algunos párrafos están contenidos. [El 85 COBOL Estándar requiere que los párrafos estén dentro de una sección excepto cuando no hay declarativos. 85 COBOL Estándar requiere que todos los párrafos estén en secciones o que ninguno lo esté.]
Expresiones condicionales	Condiciones de clase DBCS y KANJI Especificación de elementos de datos de uso COMPUTATIONAL-3 o uso PACKED-DECIMAL en una prueba de clase NUMERIC
Condición de relación	Encerrar un literal alfanumérico, DBCS o nacional entre paréntesis El formato de puntero de datos, el puntero de procedimiento, y el formato de puntero de función Comparación de un nombre de índice con una expresión aritmética Uso de paréntesis dentro de condiciones de relación combinadas abreviadas
Frase CORRESPONDIENTE	Especificación de un identificador que está subordinado a un elemento de relleno
Frase INVALID KEY	Omisión de la frase INVALID KEY y de un procedimiento EXCEPTION/ERROR aplicable. [El 85 COBOL Estándar requiere al menos uno de ellos.]
Sentencia ACCEPT	El operando *environment-name* de la frase FROM La frase DATE AAAAMMDD La frase DAY YYYYDDD
Sentencia ADD	Un compuesto de operandos de más de 18 dígitos

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
sentencia CALL	Los operandos de puntero de procedimiento y puntero de función para identificar el programa al que se va a llamar Las frases y parámetrossiguientes: • DIRECCIÓN DE • LONGITUD DE • OMITIDO • POR VALOR • DEVOLVIENDO Especificación del nombre de programa llamado en un elemento de datos alfabético o decimal con zona Especificación de un argumento definido como un elemento de grupo subordinado. [El 85 COBOL Estándar requiere que los argumentos sean un elemento de datos elemental o un elemento de grupo definido con el nivel 01.]
Sentencia CANCEL	Especificación del nombre del programa que se va a cancelar en un elemento de datos alfabético o decimal con zona Efecto de la opción de compilador PGMNAME en el nombre del programa que se va a cancelar
Sentencia CLOSE	frase WITH NO REWIND El formato secuencial de línea
Sentencia COMPUTE	El uso de la palabra EQUAL en lugar del signo igual (=)
Sentencia DISPLAY	El operando *environment-name* de la frase UPON Visualización de literales numéricos con signo y literales numéricos no enteros
Sentencia DIVIDE	Un compuesto de operandos de más de 18 dígitos
sentencia EXIT	Especificar la sentencia EXIT en una frase que tiene sentencias antes o después de la sentencia EXIT o en un párrafo que tiene otras frases. [El 85 COBOL Estándar requiere que la sentencia EXIT se especifique en una frase por sí misma y que la frase sea la única frase del párrafo.]
Sentencia EXIT PROGRAM	Especificar EXIT PROGRAM antes de la última sentencia en una secuencia de sentencias imperativas. [El 85 COBOL Estándar requiere que la sentencia EXIT PROGRAM se especifique como la última sentencia de una secuencia de sentencias imperativas.]
sentencia GO TO	Codificar el formato incondicional antes de la última sentencia en una secuencia de sentencias imperativas. [El 85 COBOL Estándar requiere que se codifique un GO TO incondicional: • Sólo en un párrafo de sentencia única si no se especifica ningún nombre-procedimiento • De lo contrario, como la última declaración de una frase.]
Sentencia IF	El uso de END-IF con la frase NEXT ORACIÓN. [El 85 COBOL Estándar no permite el uso de END-IF con NEXT ORACIÓN.]

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
Sentencia INITIALIZE	DBCS, EGCS, NATIONAL y NATIONAL-EDITED en la frase SUSTITUIR Inicialización de un elemento de datos que contiene la frase DEPENDING de la cláusula OCCURS
sentencia MERGE	Especificación de nombres de archivo en una cláusula SAME
Sentencia MULTIPLY	Un compuesto de operandos de más de 18 dígitos
Sentencia OPEN	El formato secuencial de línea Especificación de la frase EXTEND para los archivos que tienen una cláusula LINAGE
Sentencia PERFORM	Una sentencia PERFORM en línea vacía Una salida común para dos o más REALIZA activos
Sentencia READ	Frase PREVIOUS Omisión de la frase AT END y de un procedimiento declarativo aplicable Omisión de la frase INVALID KEY y de un procedimiento declarativo aplicable Leer en un elemento que no es un elemento de grupo alfanumérico ni un elemento alfanumérico elemental
Sentencia RETURN	Volver a un elemento que no sea un elemento de grupo alfanumérico ni un elemento alfanumérico elemental
Sentencia REWRITE	Omisión de la frase INVALID KEY y de un procedimiento declarativo aplicable Reescritura de un registro con un número de posiciones de caracteres diferente al número de posiciones de caracteres del registro que se está regrabando
Sentencia SEARCH	Especificación de END SEARCH con NEXT ORACIÓN Omisión de la frase NEXT ORACIÓN y de las sentencias imperativas en el formato de búsqueda binaria
Sentencia SET	El formato del puntero de datos El puntero de procedimiento y el formato de puntero de función
Sentencia SORT	Especificación de nombres de archivo CEDER en la cláusula SAME

*Tabla 62. **IBM** (continuación)*

Área de idioma	**Elementos de extensión**
Sentencia START	Omisión de la frase INVALID KEY y de un procedimiento de excepción aplicable Utilización de una clave de una categoría distinta de la alfanumérica Los siguientes operadores relacionales: • MENOR QUE • < • NO MAYOR QUE • NO > • MENOR O IGUAL QUE • <=
Sentencia STOP	Especificación de un literal de punto fijo no entero o un entero numérico con signo o un literal de punto fijo no entero Codificación de STOP como distinta de la última sentencia de una frase
Sentencia STRING	Modificación de referencia del elemento de datos especificado en la frase INTO
Sentencia SUBTRACT	Un compuesto de operandos de más de 18 dígitos
sentencia UNSTRING	Modificación de referencia del campo de envío
Sentencia WRITE	Frases INVALID KEY y NOT ON INVALID KEY El formato secuencial de línea Para un archivo relativo, escribir un número de posiciones de caracteres diferente al número de posiciones de caracteres del registro que se está sustituyendo Especificación de las frases AVANZANDO PAGE y END-OF-PAGE en una única sentencia WRITE Para un archivo relativo o indexado, omisión de la frase INVALID KEY y un procedimiento de excepción aplicable

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
Funciones intrínsecas	Efecto de las opciones de compilador DATEPROC e INTDATE en las funciones INTEGER-OF-DATE e INTEGER-OF-DAY Las funciones siguientes: • "AÑADIR DURACIÓN" en la página 463 • "CONVERTIR-FECHA-HORA" en la página 466 • "DATE-TO-AAAAMMDD" en la página 470 • "DATEVAL" en la página 470 • "DAY-TO-AAAADDD" en la página 472 • "VISUALIZAR-DE" en la página 472 • "EXTRAER-FECHA-HORA" en la página 474 • "BUSCAR DURACIÓN" en la página 475 • "NACIONAL-DE" en la página 483 • "RESTAR DURACIÓN" en la página 492 • "TEST-FECHA-HORA" en la página 494 • "TRIML" en la página 496 • "TRIMR" en la página 497 • "FECHA" en la página 497 • "AÑO-A-AAAA" en la página 500 • "VENTANA" en la página 501
Función LENGTH	Especificación de un puntero, el registro especial ADDRESS OF o el registro especial LENGTH OF como argumento para la función
Sentencias de direccionamiento de compilador	Las sentencias siguientes: • BASIS • PROCESO (CBL) • *CONTROL y *CBL • DELETE • EJECT • INSERT • READY o RESET TRACE • ETIQUETA DE SERVICIO • CARGA DE SERVICIO • SKIP1, SKIP2y SKIP3 • TÍTULO
Directivas de compilador	Directiva CALLINTERFACE

*Tabla 62. **IBM** (continuación)*	
Área de idioma	**Elementos de extensión**
Sentencia COPY	La opcionalidad de la sintaxis "OF *nombre-biblioteca*" para especificar un calificador de nombre-texto Literales para especificar *nombre-texto* y *nombre-biblioteca* Frase SUPPRESS Sentencias COPY anidadas Guión como el primer o último carácter de la forma *word* de los operandos SUSTITUCIÓN El uso de cualquier carácter (que no sea un separador COBOL) en el formato *word* de los operandos SUSTITUIR. [El 85 COBOL Estándar sólo acepta los caracteres utilizados en la formación de palabras definidas por el usuario.]

Apéndice B. Secuencias de clasificación EBCDIC y ASCII

Una secuencia de clasificación define el orden de los caracteres dentro de un juego de caracteres codificado o alfabeto COBOL para fines de clasificación, fusión y comparación de datos y para procesar archivos con una organización indexada.

En este apéndice se muestran las secuencias de clasificación ascendente para los conjuntos de caracteres EBCDIC (Extended Binary Coded Decimal Interchange Code) de un solo byte y ASCII (American National Standard Code for Information Interchange) de un solo byte. La secuencia de clasificación se define por el número ordinal de caracteres del juego de caracteres, relativo a 1.

Los símbolos y significados asociados que se muestran para la secuencia de clasificación EBCDIC son los definidos en la página de códigos EBCDIC definida con CCSID 1140. Los símbolos y significados pueden variar para otras páginas de códigos EBCDIC, pero el orden de clasificación no cambia.

Secuencia de clasificación EBCDIC

La secuencia de clasificación EBCDIC es el orden en el que se definen los caracteres en EBCDIC.

La tabla siguiente presenta el orden de clasificación para la página de códigos EBCDIC de un solo byte 1140.

Los puntos suspensivos (...) indican la omisión de un rango de números ordinales entre los números ordinales predecesor y sucesor.

Tabla 63. ***Orden de clasificación EBCDIC***

Número ordinal	Símbolo	Significado	Representación decimal	Representación hexadecimal
. . .				
65		Espacio	64	40
. . .				
75	¢	Signo de centavo	74	4A
76	.	Punto, coma decimal	75	4B
77	<	Signo menor que	76	4C
78	(	Paréntesis izquierdo	77	4D
79	+	Signo más	78	4E
80	|	Barra vertical, OR lógico	79	4F
81	&	Ampersand	80	50
. . .				
91	!	Signo de exclamación	90	5A
92	$	Signo de dólar	91	5B
93	*	Asterisco	92	5C
94	)	Paréntesis derecho	93	5D
95	;	Punto y coma	94	5E

Tabla 63. ***Orden de clasificación EBCDIC*** *(continuación)*

Número ordinal	Símbolo	Significado	Representación decimal	Representación hexadecimal
96	¬	NOT lógico	95	5F
97	-	Menos, guión	96	60
98	/	Barra inclinada	97	61
. . .				
108	,	Coma	107	6B
109	%	Signo de porcentaje	108	6C
110	_	Subrayado	109	6D
111	>	Signo mayor que	110	6E
112	?	Signo de interrogación	111	6F
. . .				
122	`	Acento grave	121	79
123	:	Dos puntos	122	7A
124	#	Signo de almohadilla, signo de almohadilla	123	7B
125	@	Signo de arroba	124	7C
126	'	Apóstrofo, signo primo	125	7D
127	=	Signo igual	126	7E
128	"	Comillas	127	7F
. . .				
130	a		129	81
131	b		130	82
132	c		131	83
133	d		132	84
134	e		133	85
135	f		134	86
136	g		135	87
137	h		136	88
138	i		137	89
. . .				
146	j		145	91
147	k		146	92
148	l		147	93
149	m		148	94
150	n		149	95

Tabla 63. ***Orden de clasificación EBCDIC*** *(continuación)*

Número ordinal	Símbolo	Significado	Representación decimal	Representación hexadecimal
151	o		150	96
152	p		151	97
153	q		152	98
154	r		153	99
. . .				
160	€	Signo de moneda euro	159	9F
. . .				
162	~	Tilde	161	A1
163	s		162	A2
164	t		163	A3
165	u		164	A4
166	v		165	A5
167	w		166	A6
168	x		167	A7
169	y		168	A8
170	z		169	A9
. . .				
177	^	Acento circunflejo	176	B0
. . .				
188	[	Corchete de apertura	187	BA
189	]	Corchete de cierre	188	BB
. . .				
193	{	Llave de apertura	192	C0
194	A		193	C1
195	B		194	C2
196	C		195	C3
197	D		196	C4
198	E		197	C5
199	F		198	C6
200	G		199	C7
201	H		200	C8
202	I		201	C9
. . .				
209	}	Llave de cierre	208	D0

Tabla 63. ***Orden de clasificación EBCDIC*** *(continuación)*

Número ordinal	Símbolo	Significado	Representación decimal	Representación hexadecimal
210	J		209	D1
211	K		210	D2
212	L		211	D3
213	M		212	D4
214	N		213	D5
215	O		214	D6
216	P		215	D7
217	Q		216	D8
218	R		217	D9
. . .				
225	\	Barra inclinada invertida	224	E0
. . .				
227	S		226	E2
228	T		227	E3
229	U		228	E4
230	V		229	E5
231	W		230	E6
232	X		231	E7
233	Y		232	E8
234	Z		233	E9
. . .				
241	0		240	F0
242	1		241	F1
243	2		242	F2
244	3		243	F3
245	4		244	F4
246	5		245	F5
247	6		246	F6
248	7		247	F7
249	8		248	F8
250	9		249	F9
. . .				

Página de códigos ASCII en inglés de EE.UU.

La secuencia de clasificación ASCII es el orden en el que se definen los caracteres en ASCII.

La tabla siguiente presenta la secuencia de clasificación para la página de códigos ASCII en inglés de EE.UU. La secuencia de clasificación es el orden en el que se definen los caracteres en ANSI INCITS 4, el código estándar nacional americano de 7 bits para el intercambio de información (ASCII de 7 bits) y en la versión de referencia internacional de *ISO/IEC 646, juego de caracteres codificados de 7 bits para el intercambio de información*.

Puntos suspensivos (.. .) indica la omisión de un rango de números ordinales entre los números ordinales predecesores y sucesores.

Tabla 64. ***Orden de clasificación ASCII***

Número ordinal	Símbolo	Significado	Representación decimal	Representación hexadecimal
1		Nulo	0	0
. . .				
33		Espacio	32	20
34	!	Signo de exclamación	33	21
35	"	Comillas	34	22
36	#	Signo de número	35	23
37	$	Signo de dólar	36	24
38	%	Signo de porcentaje	37	25
39	&	Ampersand	38	26
40	'	Apóstrofo, signo primo	39	27
41	(	Paréntesis de apertura	40	28
42	)	Paréntesis de cierre	41	29
43	*	Asterisco	42	2A
44	+	Signo más	43	2B
45	,	Coma	44	2C
46	-	Guión, menos	45	2D
47	.	Punto, coma decimal	46	2E
48	/	Barra inclinada, solidus	47	2F
49	0		48	30
50	1		49	31
51	2		50	32
52	3		51	33
53	4		52	34
54	5		53	35
55	6		54	36
56	7		55	37

*Tabla 64. **Orden de clasificación ASCII** (continuación)*				
Número ordinal	**Símbolo**	**Significado**	**Representación decimal**	**Representación hexadecimal**
57	8		56	38
58	9		57	39
59	:	Dos puntos	58	3A
60	;	Punto y coma	59	3B
61	<	Signo menor que	60	3C
62	=	Signo igual	61	3D
63	>	Signo mayor que	62	3E
64	?	Signo de interrogación	63	3F
65	@	Signo de arroba comercial	64	40
66	A		65	41
67	B		66	42
68	C		67	43
69	D		68	44
70	E		69	45
71	F		70	46
72	G		71	47
73	H		72	48
74	I		73	49
75	J		74	4A
76	K		75	4B
77	L		76	4C
78	M		77	4D
79	N		78	4E
80	O		79	4F
81	P		80	50
82	Q		81	51
83	R		82	52
84	S		83	53
85	T		84	54
86	U		85	55
87	V		86	56
88	W		87	57
89	X		88	58
90	Y		89	59

*Tabla 64. **Orden de clasificación ASCII** (continuación)*				
Número ordinal	**Símbolo**	**Significado**	**Representación decimal**	**Representación hexadecimal**
91	Z		90	5A
92	[	Corchete de apertura	91	5B
93	\	Barra inclinada invertida, solidus inverso	92	5C
94	]	Corchete de cierre	93	5D
95	^	Acento circunflejo	94	5E
96	_	Subrayado	95	5F
97	`	Acento grave	96	60
98	a		97	61
99	b		98	62
100	c		99	63
101	d		100	64
102	e		101	65
103	f		102	66
104	g		103	67
105	h		104	68
106	i		105	69
107	j		106	6A
108	k		107	6B
109	l		108	6C
110	m		109	6D
111	n		110	6E
112	o		111	6F
113	p		112	70
114	q		113	71
115	r		114	72
116	s		115	73
117	t		116	74
118	u		117	75
119	v		118	76
120	w		119	77
121	x		120	78
122	y		121	79
123	z		122	7A

*Tabla 64. **Orden de clasificación ASCII** (continuación)*

Número ordinal	Símbolo	Significado	Representación decimal	Representación hexadecimal
124	{	Llave de apertura	123	7B
125	|	Barra vertical	124	7C
126	}	Llave de cierre	125	7D
127	~	Tilde	126	7E

Apéndice C. Depuración de lenguaje fuente

Varios elementos de lenguaje COBOL implementan la característica de depuración.

Los elementos de lenguaje COBOL son:

- Depuración de líneas
- Secciones de depuración
- Registro especial DEBUG-ITEM
- Conmutador de tiempo de compilación (cláusula WITH DEBUGGING MODE)
- Conmutador de tiempo de objeto

Depuración de líneas

Una *línea de depuración* es una sentencia que se compila sólo cuando se activa el conmutador de tiempo de compilación. Las líneas de depuración le permiten, por ejemplo, comprobar el valor de un elemento de datos en determinados puntos de un procedimiento.

Para especificar una línea de depuración en el programa, codifique una D en la columna 7 (el área de indicador). Puede incluir líneas de depuración sucesivas, pero cada una debe tener una D en la columna 7. No puede dividir series de caracteres en dos líneas.

Todas las líneas de depuración deben escribirse para que el programa sea sintácticamente correcto, tanto si las líneas de depuración se compilan como si se tratan como comentarios.

Puede codificar líneas de depuración en cualquier lugar del programa después del párrafo OBJECT-COMPUTER.

Una línea de depuración que contiene sólo espacios en el área A y en el área B se trata como una línea en blanco.

Secciones de depuración

Las secciones de depuración sólo están permitidas en el programa más externo; no son válidas en programas anidados. Las secciones de depuración nunca se desencadenan mediante procedimientos contenidos en programas anidados.

Las secciones de depuración son procedimientos declarativos. Los procedimientos declarativos se describen en “USE, sentencia” en la página 525. Una sección de depuración se puede invocar, por ejemplo, mediante una sentencia PERFORM que provoca la ejecución repetida de un procedimiento. Cualquier sección declarativa de depuración *nombre-procedimiento* asociada se ejecuta una vez para cada repetición.

Una sección de depuración ejecuta *sólo* si se activan tanto el conmutador de tiempo de compilación como el conmutador de tiempo de objeto.

La característica de depuración reconoce cada aparición separada de una sentencia imperativa *dentro* de una sentencia imperativa como el principio de una sentencia separada.

No puede hacer referencia a un procedimiento definido en una sección de depuración desde una sentencia fuera de la sección de depuración.

Las referencias al registro especial DEBUG-ITEM sólo se pueden realizar desde un procedimiento declarativo de depuración.

Registro especial DEBUG-ITEM

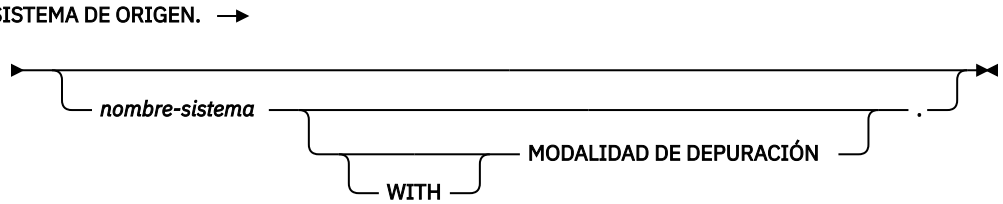
El registro especial DEBUG-ITEM proporciona información para un procedimiento declarativo de depuración sobre las condiciones que provocan la ejecución de la sección de depuración.

Para obtener detalles del registro especial DEBUG-ITEM, consulte “ELEMENTO DE DEPURACIÓN” en la página 19.

Activar conmutador de tiempo de compilación

El conmutador de tiempo de compilación activa las líneas y secciones de depuración. Para poner en vigor el conmutador de tiempo de compilación, especifique WITH DEBUGGING MODE en el párrafo SOURCE COMPUTER de la sección de configuración.

Formato



CON MODALIDAD DE DEPURACIÓN

Cuando se especifica WITH DEBUGGING MODE, se compilan todas las secciones de depuración y líneas de depuración.

Cuando se omite WITH DEBUGGING MODE, todas las secciones de depuración y líneas de depuración se tratan como comentarios.

Nota de uso: Si incluye una sentencia COPY como línea de depuración, la letra "D" debe aparecer en la primera línea de la sentencia COPY. El compilador trata el texto copiado como la línea o líneas de depuración. La sentencia COPY se ejecuta, independientemente de si se ha especificado WITH DEBUGGING MODE o no.

Activar conmutador de tiempo de objeto

El conmutador de tiempo de objeto se establece cuando se especifica la opción de tiempo de ejecución DEBUG o NODEBUG. (NODEBUG es el valor predeterminado proporcionado por IBM.)

Para obtener detalles sobre el formato, consulte *DEPURAR* en *COBOL for Linux en x86 Guía de programación*.

Los procedimientos declarativos USE FOR DEBUGGING se activan cuando DEBUG está en vigor y se inhiben cuando NODEBUG está en vigor.

Las líneas de depuración (líneas con "D" o "d" en la columna 7) no se ven afectadas por la opción DEBUG o NODEBUG; siempre están activas si se han compilado.

Cuando WITH DEBUGGING MODE *no* se especifica en el párrafo SOURCE-COMPUTER, el conmutador de tiempo de objeto no tiene ningún efecto en la ejecución del programa objeto.

No es necesario volver a compilar la unidad de origen para activar o desactivar el conmutador de tiempo de objeto.

Apéndice D. Palabras reservadas

Una *palabra reservada* es una serie de caracteres con un significado predefinido en una unidad de origen COBOL.

La tabla siguiente identifica las palabras que están reservadas en COBOL para Linux y las palabras que debe evitar porque pueden estar reservadas en un futuro release de COBOL para Linux.

- Las palabras marcadas como *X* en *Reserved* están reservadas para la función implementada en COBOL para Linux. Si se utiliza como nombres definidos por el usuario, estas palabras se marcan con un mensaje de nivel S.
- Las palabras marcadas como *X* bajo *Sólo estándar* son 85 COBOL Estándar palabras reservadas para la función no implementada en COBOL para Linux. El uso de estas palabras como nombres definidos por el usuario se marca con un mensaje de nivel S.
- Las palabras marcadas como *X* en *Palabras reservadas potenciales* son palabras que pueden reservarse en un futuro release de COBOL para Linux. IBM recomienda no utilizar estas palabras como nombres definidos por el usuario. El uso de estas palabras como nombres definidos por el usuario se marca con un mensaje de nivel I.

  Esta columna incluye las palabras reservadas en Estándar COBOL 2002.

Tabla 65. ***Palabras reservadas***

Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
+ Operador aritmético-suma o suma unaria	X		
-Operador aritmético-unario menos o resta	X		
* operador aritmético-multiplicación	X		
/operador aritmético-división	X		
** Operador aritmético-exponenciación	X		
> Operador relacional-mayor que	X		
< Operador relacional-menor que	X		
= Operador relacional-igual y operador de asignación en COMPUTE	X		
== Pseudo-delimitador de texto en las sentencias COPY y REPLACE	X		
> = Operador relacional-mayor o igual que	X		
< = Operador relacional-menor o igual que	X		
< > Operador relacional-no igual a		X	
* > Indicador de comentario	X		
> > Indicador de directiva de compilador		X	
ACCEPT	X		
Access	X		
ACTIVO-CLASE			X

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
ADD	X		
ADDRESS	X		
AVANZANDO	X		
AFTER	X		
Alineada			X
TODOS	X		
ALLOCATE	X		
ALFABETO	X		
Alfabético	X		
ALFABÉTICO-INFERIOR	X		
ALFABÉTICO-SUPERIOR	X		
Alfanumérico	X		
Alfanumérico editado	X		
TAMBIÉN	X		
ALTER	X		
Alternativo	X		
Y	X		
CUALQUIERA	X		
CUALQUIER caso			X
APPLY	X		
ARE	X		
AREA	X		
ÁREAS	X		
ascendente	X		
ASSIGN	X		
AT	X		
Autor	X		
B-Y			X
B-NO			X
B-O			X
B-XOR			X
BASADO			X
BASIS	X		

*Tabla 65. **Palabras reservadas** (continuación)*

Word	Reservado	Sólo estándar	Palabras reservadas potenciales
BEFORE	X		
PRINCIPIO	X		
BINARY	X		
BINARY-CHAR			X
BINARIO-DOBLE			X
BINARIO-LARGO			X
BINARIO-CORTO			X
BIT			X
BLANK	X		
BLOCK	X		
BOOLEAN	X		
Abajo	X		
BY	X		
CALL	X		
CANCEL	X		
CBL	X		
CD		X	
CF		X	
CH		X	
CHARACTER	X		
Caracteres	X		
clase			X
CLASS-ID			X
UNIDADES DE RELOJ		X	
CERRAR	X		
COBOL	X		
Código	X		
CÓDIGO-CONJUNTO	X		
Col			X
REVESTIMIENTO DE	X		
COLS			X
COLUMN		X	
Columnas			X

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
COM-REG	X		
Coma	X		
Común	X		
COMMUNICATION		X	
COMP	X		
COMP-1	X		
COMP-2	X		
COMP-3	X		
COMP-4	X		
COMP-5	X		
COMPUTATIONAL	X		
COMPUTATIONAL-1	X		
COMPUTATIONAL-2	X		
COMPUTATIONAL-3	X		
COMPUTATIONAL-4	X		
COMPUTATIONAL-5	X		
COMPUTE	X		
Condición			X
configuración	X		
Constante			X
CONTIENE	X		
CONTENT	X		
CONTINUE	X		
Control		X	
controles		X	
Está convirtiéndose	X		
COPIAR	X		
CORR	X		
CORRESPONDIENTE	X		
COUNT	X		
CRT			X
CURRENCY	X		
CURSOR			X

Tabla 65. ***Palabras reservadas*** *(continuación)*

Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
Datos	X		
PUNTERO DE DATOS			X
FECHA	X		
FECHA-COMPILADO	X		
DATE-GRABADO	X		
DÍA	X		
DÍA DE LA SEMANA	X		
DBCS	X		
DE		X	
CONTENIDO DE DEPURACIÓN	X		
ELEMENTO DE DEPURACIÓN	X		
LÍNEA DE DEPURACIÓN	X		
NOMBRE DE DEPURACIÓN	X		
DEBUG-SUB-1	X		
DEBUG-SUB-2	X		
DEBUG-SUB-3	X		
depurar	X		
COMA DECIMAL	X		
DECLARATIVES	X		
DEFAULT	X		
DELETE	X		
DELIMITADO	X		
DELIMITER	X		
DEPENDIENDO	X		
descendente	X		
destino		X	
DETAIL		X	
DISABLE		X	
MOSTRAR	X		
DISPLAY-1	X		
Dividir	X		
División	X		
Abajo	X		

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
Duplicados	X		
DINÁMICO	X		
AE			X
EGCS	X		
EGI		X	
EJECT	X		
ELSE	X		
EMI		X	
ENABLE		X	
END	X		
END-ACEPTAR			X
END-ACEPTAR	X		
END-ADD	X		
FIN-CALL	X		
END-COMPUTE	X		
FIN-DELETE	X		
FIN-VISUALIZAR			X
END-DIVIDIR	X		
END-EVALUATE	X		
END-EXEC	X		
END-IF	X		
END-INVOCAR			X
END-JSON			X
END-MULTIPLY	X		
END-OF-PAGE	X		
END-PERFORM	X		
END-READ	X		
FIN-RECEIVE		X	
FIN-RETORNO	X		
END-REWRITE	X		
FIN-BUSCAR	X		
END-START	X		
END-STRING	X		

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
END-SUBTRACT	X		
FIN-DESCABEZADO	X		
FIN-GRABACIÓN	X		
END-XML	X		
finalizar	X		
INTRO	X		
ENTRADA	X		
Entorno	X		
EO			X
EOP	X		
IGUAL	X		
ERROR	X		
ESI		X	
EVALUATE	X		
EVERY	X		
Excepción	X		
OBJETO DE EXCEPCIÓN			X
EXEC	X		
EXECUTE	X		
EXIT	X		
Ampliar	X		
EXTERNAL	X		
FACTORY	X		
FALSE	X		
FD	X		
ARCHIVO	X		
CONTROL DE ARCHIVOS	X		
filler	X		
Final		X	
PRIMERO	X		
FLOAT-AMPLIADO			X
FLOAT-LONG			X
FLOAT-CORTO			X

*Tabla 65. **Palabras reservadas** (continuación)*

Word	Reservado	Sólo estándar	Palabras reservadas potenciales
PIE	X		
FOR	X		
FORMAT			X
FREE	X		
DESDE	X		
FUNCTION	X		
ID-FUNCIÓN			X
FUNCTION-POINTER	X		
Generar	X		
Obtener			X
DANDO	X		
GLOBAL	X		
IR	X		
GOBACK	X		
MAYOR	X		
GROUP		X	
GRUPO-PÁGINA	X		
Cabecera		X	
ALTO VALOR	X		
VALORES ALTOS	X		
I-O	X		
I-O-CONTROL	X		
ID	X		
Identificación	X		
SI	X		
IN	X		
INDEX	X		
INDEXED	X		
INDICAR		X	
Hereda	X		
INITIAL	X		
Inicializar	X		
INICIAR		X	

Tabla 65. ***Palabras reservadas*** *(continuación)*

Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
INPUT	X		
ENTRADA-SALIDA	X		
INSERT	X		
INSPECT	X		
Instalación	X		
INTERFACE			X
INTERFACE-ID			X
INTO	X		
No válido	X		
Invocar			X
IS	X		
CÓDIGO JSON			X
JUST	X		
JUSTIFIED	X		
Kanji	X		
CLAVE	X		
LABEL	X		
Última		X	
LÍDER	X		
LEFT	X		
LONGITUD	X		
Menos	X		
LIMIT		X	
LIMITS		X	
LINAGE	X		
LINAGE-COUNTER	X		
Línea	X		
LINE-CONTADOR		X	
LINES	X		
LINKAGE	X		
ALMACENAMIENTO LOCAL	X		
LOCALE	X		
BLOQUEAR	X		

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
LOW-VALUE	X		
VALORES BAJOS	X		
MEMORY	X		
MERGE	X		
MESSAGE		X	
METHOD			X
METHOD-ID			X
menos			X
MODE	X		
Módulos	X		
MÁS-ETIQUETAS	X		
MOVE	X		
MÚLTIPLE	X		
Multiplicar	X		
NATIONAL	X		
NACIONAL-EDITADO	X		
Nativo	X		
Negativo	X		
ANIDADO			X
NEXT	X		
NO	X		
NO	X		
NULL	X		
NULOS	X		
NUMBER		X	
NUMERIC	X		
Numérico editado	X		
OBJECT			X
OBJETO-SISTEMA	X		
REFERENCIA-OBJETO			X
OCCURS	X		
OF	X		
DESACTIVADO	X		

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
OMITIDO	X		
ACT.	X		
OPEN	X		
Opcional	X		
Opciones			X
OR	X		
ORDER	X		
Organización	X		
OTHER	X		
OUTPUT	X		
OVERFLOW	X		
OVERRIDE	X		
DECIMAL EMPAQUETADO	X		
Relleno	X		
página	X		
PAGE-COUNTER		X	
PASSWORD	X		
PERFORM	X		
PF		X	
PH		X	
PIC	X		
PICTURE	X		
más		X	
POINTER	X		
Posición	X		
Positivo	X		
Presente			X
Imprimir		X	
PROCEDIMIENTO	X		
PROCEDURE-POINTER	X		
Procedimientos	X		
Continuar	X		
Proceso	X		

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
PROGRAM	X		
PROGRAM-ID	X		
PUNTERO DE PROGRAMA			X
propiedad			X
Prototipo			X
PURGE		X	
QUEUE		X	
QUOTE	X		
Comillas	X		
RAISE			X
emitir			X
Random	X		
RD		X	
LECTURA	X		
LISTO	X		
RECEIVE		X	
RECORD	X		
RECORDING	X		
Registros	X		
recursive	X		
REDEFINE	X		
RELA	X		
REFERENCIA	X		
Referencias	X		
Relativo	X		
RELEASE	X		
RELOAD	X		
Resto	X		
ELIMINACIÓN	X		
RENAMES	X		
SUSTITUIR	X		
sustituir	X		
INFORME		X	

*Tabla 65. **Palabras reservadas** (continuación)*

Word	Reservado	Sólo estándar	Palabras reservadas potenciales
Informes		X	
Informes		X	
REPOSITORY	X		
RERUN	X		
RESERVE	X		
RESET	X		
RESUME			X
RETRY			X
RETURN	X		
CÓDIGO-RETORNO	X		
DEVOLVIENDO	X		
INVERTIDO	X		
REWIND	X		
REWRITE	X		
FR		X	
RH		X	
RIGHT	X		
REDONDEADO	X		
RUN	X		
SAME	X		
SCREEN		X	
SD	X		
SEARCH	X		
SECTION	X		
SECURITY	X		
Segmento		X	
LÍMITE DE SEGMENTO	X		
Seleccionar	X		
SELF	X		
SEND		X	
FRASE	X		
SEPARADO	X		
SEQUENCE	X		

*Tabla 65. **Palabras reservadas** (continuación)*			
Word	**Reservado**	**Sólo estándar**	**Palabras reservadas potenciales**
Secuencial	X		
SERVICE	X		
SET	X		
compartir			X
SHIFT-IN	X		
SHIFT-OUT	X		
signo	X		
BLOQUE	X		
SKIP1	X		
SKIP2	X		
SKIP3	X		
SORT	X		
CONTROL DE SORT	X		
TAMAÑO-NÚCLEO-CLASIFICACIÓN	X		
SORT-FILE-SIZE	X		
CLASIFICAR-FUSIONAR	X		
ORDENAR-MENSAJE	X		
TAMAÑO-MODALIDAD-CLASIFICACIÓN	X		
ORDENAR-RETORNO	X		
ORIGEN		X	
ORIGEN	X		
ORDENADOR DE ORIGEN	X		
Orígenes			X
SPACE	X		
espacios	X		
NOMBRES ESPECIALES	X		
SQL	X		
SQLIMS	X		
STANDARD	X		
STANDARD-1	X		
STANDARD-2	X		
INICIO	X		
ESTATUS	X		

*Tabla 65. **Palabras reservadas** (continuación)*

Word	Reservado	Sólo estándar	Palabras reservadas potenciales
DETENER	X		
STRING	X		
SUB-QUEUE-1		X	
SUB-QUEUE-2		X	
SUB-QUEUE-3		X	
Restar	X		
SUM		X	
SUPER	X		
Suprimir	X		
Simbólico	X		
SYNC	X		
Sincronizado	X		
SYSTEM-PREDETERMINADO			X
TABLE		X	
TALLY	X		
TALLYING	X		
TAPE	X		
TERMINAL		X	
TERMINATE		X	
TEST	X		
TEXT		X	
THAN	X		
THEN	X		
a la	X		
A través	X		
TIME	X		
Veces	X		
Título	X		
A	X		
TOP	X		
TRACE	X		
FINAL	X		
TRUE	X		

*Tabla 65. **Palabras reservadas** (continuación)*

Word	Reservado	Sólo estándar	Palabras reservadas potenciales
TYPE	X		
TYPEDEF		X	
UNIT	X		
Universal			X
UNLOCK		X	
UnString	X		
UNTIL	X		
ARRIBA	X		
ACTIVADO	X		
USO	X		
USE	X		
USER-PREDETERMINADO			X
USING	X		
VAL-ESTADO			X
Válido			X
Validar			X
VALIDAR-ESTADO			X
VALUE	X		
VALUES	X		
VARYING	X		
WHEN	X		
WHEN-COMPILED	X		
WITH	X		
WORDS	X		
TRABAJO-ALMACENAMIENTO	X		
WRITE	X		
SÓLO ESCRITURA	X		
XML	X		
CÓDIGO XML	X		
XML-EVENT	X		
XML-NTEXT	X		
XML-ESQUEMA	X		
XML-TEXTO	X		

*Tabla 65. **Palabras reservadas** (continuación)*

Word	Reservado	Sólo estándar	Palabras reservadas potenciales
ZERO	X		
ZEROES	X		
ZEROS	X		

Apéndice E. Palabras sensibles al contexto

Una palabra sensible al contexto es una palabra COBOL que está reservada sólo en los formatos generales en los que se especifica. Si se utiliza una palabra sensible al contexto donde se permite la palabra sensible al contexto en el formato general, la palabra se trata como una palabra clave; de lo contrario, se trata como una palabra definida por el usuario.

Tabla 66. Palabras sensibles al contexto

Palabra sensible al contexto	**Construcción de lenguaje o contexto**
CICLO	sentencia EXIT
INICIALIZADO	sentencia ALLOCATE
NOMBRE	sentencia XML GENERATE
PÁRRAFO	sentencia EXIT
RECURSIVO	Párrafo PROGRAM-ID
AAAADDD	Sentencia ACCEPT
AAAAMMDD	Sentencia ACCEPT

Referencias relacionadas
“Palabras definidas por el usuario” en la página 13

Apéndice D, “Palabras reservadas”, en la página 565

Apéndice F. Consideraciones sobre el entorno local

Un *entorno local* es una colección de convenios específicos de idioma y específicos de cultura para el proceso de información. Un entorno local hace que la información sobre idioma y cultura esté disponible en tiempo de ejecución para que el mismo programa pueda visualizar o procesar datos de forma diferente para distintos países o culturas.

Para obtener información adicional sobre los entornos locales y los detalles de configuración de entornos locales para idiomas y entornos culturales específicos, consulte *Establecimiento del entorno local* en la publicación *COBOL for Linux en x86 Guía de programación*.

Tiempo de compilación frente a entorno local de tiempo de ejecución

En general, el tiempo de ejecución COBOL determina el entorno local en vigor cuando se activa la aplicación COBOL. Sin embargo, el proceso siguiente se basa en el entorno local en vigor en el momento de la compilación:

- Evaluación de nombres definidos por el usuario y valores literales

  El compilador utiliza la página de códigos indicada por el entorno local en vigor durante la compilación para evaluar el programa fuente. Esto incluye la evaluación de nombres definidos por el usuario y valores literales.

  Cuando se asocia un valor literal con un elemento de datos de forma que se necesita la conversión de página de códigos, se utiliza la página de códigos en vigor en tiempo de ejecución para el elemento de datos. Para un elemento de clase alfanumérico con codificación ASCII nativa, se utiliza la página de códigos indicada por el entorno local en vigor en tiempo de ejecución. Para un elemento de clase alfanumérica con codificación EBCDIC, se utiliza la página de códigos EBCDIC en vigor en tiempo de ejecución.
- Evaluación de secuencias de clasificación

  Cuando la opción de compilador COLLSEQ (LOCALE) o la opción de compilador NCOLLSEQ (LOCALE) está en vigor, el entorno local de tiempo de compilación determina los valores siguientes:

  – El rango de caracteres especificado por literales en una frase THRU para los siguientes elementos de lenguaje:

    - Una cláusula VALUE de nombre-condición
    - Una sentencia EVALUATE
    - Una cláusula ALPHABET en el párrafo SPECIAL-NAMES
    - Una cláusula CLASS en el párrafo SPECIAL-NAMES

  – Las posiciones ordinales de los caracteres especificados en una cláusula SIMBÓLICO CHARACTERS

En las secciones siguientes se describen los efectos de los entornos locales en el proceso de tiempo de ejecución COBOL.

Páginas de códigos

El entorno local de tiempo de ejecución se utiliza para determinar:

- La página de códigos para la codificación de elementos de datos alfanuméricos y DBCS que tienen codificación nativa (no EBCDIC). La página de códigos EBCDIC en vigor se utiliza cuando se especifica la opción de compilador CHAR (EBCDIC), no se establece la variable de entorno EBCDIC_CODEPAGE y el elemento de datos se describe sin la frase NATIVE.
- Correlaciones de casos para las funciones intrínsecas UPPER-CASE y LOWER-CASE
- La página de códigos de salida para la función intrínseca DISPLAY-OF cuando se omite un argumento de página de códigos. La página de códigos EBCDIC en vigor se utiliza cuando se especifica la opción de

compilador CHAR (EBCDIC), no se establece la variable de entorno EBCDIC_CODEPAGE y el elemento de datos se describe sin la frase NATIVE.

- La página de códigos fuente para la función intrínseca NATIONAL-OF cuando se omite un argumento de página de códigos. La página de códigos EBCDIC en vigor se utiliza cuando se especifica la opción de compilador CHAR (EBCDIC), no se establece la variable de entorno EBCDIC_CODEPAGE y el elemento de datos se describe sin la frase NATIVE.
- La página de códigos para la conversión de datos para aceptarlos en elementos de datos de uso NACIONAL con la sentencia ACCEPT
- La página de códigos para la conversión de datos para la visualización de elementos de datos de uso NACIONAL con la sentencia DISPLAY

Dos páginas de códigos de tiempo de ejecución pueden estar en vigor: una para datos nativos (datos no EBCDIC) y otra para datos de host (datos EBCDIC). Para obtener información sobre cómo se determinan estas páginas de códigos de tiempo de ejecución, consulte *Especificación de la página de códigos para datos de caracteres* en la publicación *COBOL for Linux en x86 Guía de programación*.

Secuencias de clasificación

En general, COBOL para Linux utiliza el orden de clasificación definido por el entorno local de tiempo de ejecución cuando la opción de compilador COLLSEQ (LOCALE) o la opción de compilador NCOLLSEQ (LOCALE) está en vigor. COLLSEQ (LOCALE) afecta a elementos de datos alfanuméricos y DBCS; NCOLLSEQ (LOCALE) afecta a elementos descritos con USAGE NATIONAL.

Cuando se especifica COLLSEQ (BINARY) o NCOLLSEQ (BINARY), se utiliza una secuencia de clasificación binaria.

En algunos casos, las reglas de idioma especifican el uso de una secuencia de clasificación no de entorno local independientemente del valor de la opción de compilador COLLSEQ o NCOLLSEQ. Por ejemplo:

- Siempre se utiliza una secuencia de clasificación binaria para las claves de archivo indexado.
- Para las comparaciones alfanuméricas se utiliza una secuencia de clasificación especificada en la cláusula PROGRAM COLLATING SEQUENCE del párrafo OBJECT-COMPUTER.
- Se utiliza una secuencia de clasificación especificada en la cláusula PROGRAM COLLATING SEQUENCE del párrafo OBJECT-COMPUTER para sentencias SORT o MERGE con claves alfanuméricas, a menos que se especifique la frase COLLATING SEQUENCE en la sentencia SORT o MERGE.
- Se utiliza una secuencia de clasificación especificada en la frase COLLATING SEQUENCE de una sentencia SORT o MERGE para las claves alfabéticas y las claves alfanuméricas.

Entornos locales soportados

COBOL para Linux da soporte a entornos locales para determinadas combinaciones de idioma, país y página de códigos. Las variables de entorno del sistema identifican el entorno local de tiempo de ejecución en vigor para categorías de entorno local específicas. Para obtener detalles, consulte *Utilización de variables de entorno para especificar un entorno local* en *COBOL for Linux en x86 Guía de programación*.

Apéndice G. Especificaciones de la industria

COBOL para Linux da soporte a diversos estándares del sector.

- Estándares ISO COBOL
  - ISO 1989:1985, *Lenguajes de programación-COBOL*

    ISO 1989:1985 es idéntico a ANSI INCITS 23-1985 (R2001), *Programming Languages-COBOL*
  - ISO/IEC 1989/AMD1:1992, *Lenguajes de programación-COBOL: módulo de función intrínseca*

    ISO/IEC 1989/AMD1:1992 es idéntico a ANSI INCITS 23a-1989 (R2001), *Programming Languages-Intrinsic Function Module for COBOL*
  - ISO/IEC 1989/AMD2:1994, *Lenguajes de programación-Enmienda de corrección y aclaración para COBOL*

    ISO/IEC 1989/AMD2:1994 es idéntico a ANSI INCITS 23b-1993 (R2001), *Lenguaje de programación-Enmienda de corrección para COBOL*

    Todos los módulos necesarios están soportados en el nivel más alto definido por el estándar.

    Se admiten los siguientes módulos opcionales del estándar:

    - Funciones intrínsecas (1 ITR 0, 1)
    - Depuración (1 DEB 0, 2)

    Los siguientes módulos opcionales del estándar no están soportados:

    - Escritor de informes
    - Comunicaciones
    - Depuración (2 DEB 0, 2)
    - Segmentación (2 SEG 0, 2)
  - ISO/IEC 1989:2002, *Information technology-Programming languages-COBOL* (soporte parcial)

    ISO/IEC 1989:2002 es idéntico a ANSI INCITS 1989-2002 (R2013), *Information technology-Programming languages COBOL*
  - ISO/IEC 1989:2014, *Information technology-Lenguajes de programación, sus entornos e interfaces de software del sistema-Lenguaje de programación COBOL* (soporte parcial)

    ISO/IEC 1989:2014 es idéntico a ANSI INCITS 1989-2014, *Tecnologías de la información-Lenguajes de programación, sus entornos e interfaces de software del sistema-Lenguaje de programación COBOL*
- Estándares ANSI COBOL
  - ANSI INCITS 23-1985 (R2001), *Idiomas de programación-COBOL*
  - ANSI INCITS 23a-1989 (R2001), *Programming Languages-Intrinsic Function Module for COBOL*
  - ANSI INCITS 23b-1993 (R2001), *Enmienda de corrección de lenguaje de programación para COBOL*

    Todos los módulos necesarios están soportados en el nivel más alto definido por el estándar.

    Se admiten los siguientes módulos opcionales del estándar:

    - Funciones intrínsecas (1 ITR 0, 1)
    - Depuración (1 DEB 0, 2)

    Los siguientes módulos opcionales del estándar no están soportados:

    - Comunicaciones
    - Depuración (2 DEB 0, 2)

- Segmentación (2 SEG 0, 2)

- Versión de referencia internacional de *ISO/IEC 646, juego de caracteres codificados de 7 bits para intercambio de información*
- El juego de caracteres codificado de 7 bits definido en *American National Standard X3.4-1977, Code for Information Interchange*
- *SPIRIT (Service Provider's Requirements for Information Technology), Parte 6—Perfil de lenguaje COBOL*, publicado por Network Management Forum.
- *MIA (Multivendor Integration Architecture), requisitos técnicos*, especificado por Nippon Telegraph and Telephone Corp (NTT)

COBOL para Linux tiene la restricción siguiente relacionada con los estándares COBOL:

- Cuando se produce una división por cero en una expresión aritmética y no se especifica una frase ON SIZE ERROR, el proceso termina de forma anómala.

Consulte "Valores de opción para conformidad con 85 COBOL Standard" en la publicación *COBOL for Linux en x86 Guía de programación* para obtener la especificación de las opciones de compilador y las opciones de tiempo de ejecución de necesarias para dar soporte a los estándares anteriores.

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en EE.UU.

Es posible que IBM no ofrezca los productos, servicios o funciones que se tratan en esta publicación en otros países. Consulte al representante local de IBM para obtener información sobre los productos y servicios disponibles actualmente en su área. Las referencias a programas, productos o servicios de IBM no pretenden establecer ni implicar que sólo puedan utilizarse dichos productos, programas o servicios de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

Es posible que IBM tenga patentes o solicitudes de patente pendientes que traten el tema descrito en este documento. El hecho de proporcionar este documento no concede ninguna licencia sobre estas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director de licencias
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE. UU.

Para consultas sobre licencias en las que se solicite información sobre juegos de caracteres de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a la dirección siguiente:

Licencia de propiedad intelectual
Ley de Propiedad Intelectual y Legal
IBM Japón, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde estas disposiciones sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos países no permiten la renuncia a garantías explícitas o implícitas en ciertas transacciones, por lo que la declaración anterior puede no aplicarse en su caso.

Esta información puede incluir imprecisiones técnicas o errores tipográficos. Periódicamente se realizan cambios en la información aquí contenida; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede reservarse el derecho de realizar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento sin previo aviso.

Las referencias contenidas en esta información a sitios web que no son deIBM se proporcionan únicamente para su comodidad y no constituyen en modo alguno un aval de dichos sitios web. Los materiales de dichos sitios web no forman parte de los materiales para este producto IBM y el uso de dichos sitios web es a cuenta y riesgo del usuario.

IBM puede utilizar o distribuir la información que usted le suministre del modo que IBM considere conveniente sin incurrir por ello en ninguna obligación para con usted.

Los titulares de licencias de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de la información que se ha intercambiado, deben ponerse en contacto con:

IBM Director de licencias IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE. UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones correspondientes, incluyendo, en algunos casos, el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia disponible para el mismo son proporcionados por IBM bajo los términos del Acuerdo de cliente de IBM , IBM Acuerdo internacional de licencia de programa o cualquier acuerdo equivalente entre las partes.

Todos los datos de rendimiento contenidos en el presente documento se han obtenido en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Algunas de las medidas podrían proceder de sistemas en proceso de desarrollo y no se garantiza que dichas medidas sean las mismas en sistemas disponibles para uso general. Además, es posible que algunas de las medidas se hayan estimado a través de una extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben consultar los datos que corresponden a su entorno específico.

La información relativa a productos que no son de IBM se obtuvo de los proveedores de esos productos, sus anuncios publicados u otras fuentes de disponibilidad pública. IBM no ha probado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad o cualquier otro aspecto relacionado con los productos que no son de IBM. Las preguntas relacionadas con productos que no son de IBM deberán dirigirse a los proveedores de estos productos.

Las declaraciones relativas a la dirección o intenciones futuras de IBM pueden cambiar o ser retiradas sin aviso, y representan sólo propósitos y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con los nombres y direcciones utilizados por una empresa real es mera coincidencia.

LICENCIA DE DERECHOS DE AUTOR:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustra técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin realizar ningún pago a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Estos ejemplos no se han probado a fondo en todas las condiciones. Por lo tanto, IBMno puede garantizar ni dar por supuesta la fiabilidad, la capacidad de servicio o el funcionamiento de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin garantía de ningún tipo. IBM no será responsable de ningún daño resultante del uso de los programas de ejemplo.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado, debe incluir un aviso de copyright como el siguiente:

© (nombre de su empresa) (año). Partes de este código derivan de IBM Corp. Programas de ejemplo. © Copyright IBM Corp. .

CONSIDERACIONES SOBRE LA POLÍTICA DE PRIVACIDAD:

IBM , incluido el software como soluciones de servicio, ("Ofertas de software") pueden utilizar cookies u otras tecnologías para recopilar información de uso del producto, para ayudar a mejorar la experiencia del usuario final o para adaptar las interacciones con el usuario final, o para otros fines. En muchos casos, las ofertas de software no recopilan información de identificación personal. Algunas de nuestras ofertas de software pueden ayudarle a recopilar información de identificación personal. Si esta oferta de software utiliza cookies para recopilar información de identificación personal, la información específica sobre el uso de cookies de esta oferta se establece a continuación.

Esta Oferta de Software no utiliza cookies u otras tecnologías para recopilar información de identificación personal.

Si las configuraciones desplegadas para esta oferta de software le proporcionan como cliente la capacidad de recopilar información de identificación personal de los usuarios finales a través de cookies y otras tecnologías, debe buscar su propio asesoramiento legal sobre cualquier legislación aplicable a dicha recopilación de datos, incluidos los requisitos de aviso y consentimiento.

Para obtener más información sobre el uso de diversas tecnologías, incluidas las cookies, para estos fines, consulte la Política de privacidad de IBMen http://www.ibm.com/privacy y la Declaración de privacidad en línea de IBMen http://www.ibm.com/privacy/details en la sección titulada "Cookies, Web Beacons y otras tecnologías," y "Declaración de privacidad de productos de software y software como servicio de IBM" en http://www.ibm.com/software/info/product-privacy.

Información de interfaz de programación

Este manual de consulta del lenguaje documenta las interfaces de programación previstas que permiten al cliente escribir programas para obtener los servicios de COBOL para Linux.

Marcas comerciales

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp. registradas en muchas jurisdicciones en todo el mundo.

Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Hay disponible una lista actual de marcas registradas de IBM en la web en "Copyright and trademark information" en www.ibm.com/legal/copytrade.shtml.

Intel es una marca registrada de Intel Corporation o sus filiales en Estados Unidos y otros países.

Java™ y todas las marcas y logotipos basados en Java son marcas comerciales o marcas registradas de Oracle y de sus filiales.

Linux es una marca registrada de Linus Torvalds en Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NTy el logotipo de Windows son marcas registradas de Microsoft Corporation en Estados Unidos, otros países, o ambos.

UNIX es una marca registrada de The Open Group en Estados Unidos y otros países.

Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas.

Glosario

Los términos de este glosario se definen de acuerdo con su significado en COBOL. Estos términos pueden tener o no el mismo significado en otros idiomas.

glossary.html

Este glosario incluye términos y definiciones de las publicaciones siguientes:

- *ANSI INCITS 23-1985, Lenguajes de programación-COBOL*, modificado por *ANSI INCITS 23a-1989, Idiomas de programación-COBOL-Módulo de función intrínseca para COBOL*, y *ANSI INCITS 23b-1993, Idiomas de programación-Enmienda de corrección para COBOL*
- *ISO 1989:1985, Lenguajes de programación-COBOL*, modificado por *ISO/IEC 1989/AMD1:1992, Lenguajes de programación-COBOL: Módulo de función intrínseca*
- *ANSI X3.172-2002, American National Standard Dictionary for Information Systems*
- *INCITS/ISO/IEC 1989-2002, Tecnologías de la información-Lenguajes de programación-COBOL*
- *INCITS/ISO/IEC 1989:2014, Tecnologías de la información-Lenguajes de programación, sus entornos e interfaces de software del sistema-Lenguaje de programación COBOL*

Las definiciones de American National Standard van precedidas de un asterisco (*).

A

*** condición de relación combinada abreviada**
La condición combinada que resulta de la omisión explícita de un sujeto común o un sujeto común y operador relacional común en una secuencia consecutiva de condiciones de relación.

terminación anómala
Terminación anómala de un programa.

*** modalidad de acceso**
La forma en la que se van a operar los registros dentro de un archivo.

*** coma decimal real**
La representación física, utilizando los caracteres de coma decimal punto (.) o coma (,), de la posición de coma decimal en un elemento de datos.

codificación de documento real
Para un documento XML, una de las siguientes categorías de codificación que el analizador XML determina examinando los primeros bytes del documento:

- ASCII
- EBCDIC
- UTF-8
- UTF-16, ya sea big-endian o little-endian
- Otra codificación no soportada
- Sin codificación reconocible

Linux sistema de archivos nativo
Cualquiera de los sistemas de archivos locales o de red que soportan directamente archivos continuos codificados o binarios.

Los sistemas de archivos nativos Linux dan soporte a archivos secuenciales de línea directamente y se utilizan como almacén de archivos para todos los demás tipos de archivos COBOL.

*** nombre-alfabeto**
Palabra definida por el usuario, en el párrafo `SPECIAL-NAMES` de `ENVIRONMENT DIVISION`, que asigna un nombre a un juego de caracteres específico o a una secuencia de clasificación o a ambos.

*** carácter alfabético**
Una letra o un carácter de espacio.

elemento de datos alfabéticos
Elemento de datos que se describe con una serie de caracteres PICTURE que contiene sólo el símbolo A. Un elemento de datos alfabético tiene USAGE DISPLAY.

*** carácter alfanumérico**
Cualquier carácter del juego de caracteres de un solo byte del sistema.

posición de carácter alfanumérico
Véase *posición de carácter*.

elemento de datos alfanuméricos
Referencia general a un elemento de datos que se describe implícita o explícitamente como USAGE DISPLAY, y que tiene una categoría alfanumérica, alfanumérica editada o numérica editada.

elemento de datos editado alfanumérico
Elemento de datos que se describe mediante una serie de caracteres PICTURE que contiene al menos una instancia del símbolo A o X y al menos uno de los símbolos de inserción simples B, 0o /. Un elemento de datos editado alfanumérico tiene USAGE DISPLAY.

*** función alfanumérica**
Función cuyo valor se compone de una serie de uno o más caracteres del conjunto de caracteres alfanuméricos del sistema.

elemento de grupo alfanumérico
Elemento de grupo que se define sin una cláusula GROUP-USAGE NATIONAL . Para operaciones como INSPECT, STRINGy UNSTRING, un elemento de grupo alfanumérico se procesa como si todo su contenido se describiera como USAGE DISPLAY independientemente del contenido real del grupo. Para las operaciones que requieren el proceso de los elementos elementales dentro de un grupo, como MOVE CORRESPONDING, ADD CORRESPONDINGo INITIALIZE, un elemento de grupo alfanumérico se procesa utilizando la semántica de grupo.

LITERAL ALFANUMÉRICO
Un literal que tiene un delimitador de apertura del conjunto siguiente: ', ", X', X", Z'o Z". La serie de caracteres puede incluir cualquier carácter en el juego de caracteres del sistema.

*** clave de registro alternativa**
Clave, distinta de la clave de registro principal, cuyo contenido identifica un registro dentro de un archivo indexado.

ANSI (American National Standards Institute)
Organización que consta de productores, consumidores y grupos de interés general y establece los procedimientos mediante los cuales las organizaciones acreditadas crean y mantienen estándares voluntarios de la industria en los Estados Unidos.

argumento
(1) Un identificador, un literal, una expresión aritmética o un identificador de función que especifica un valor que debe utilizarse en la evaluación de una función. (2) Un operando de la frase USING de una sentencia CALL , utilizada para pasar valores a un programa llamado.

*** operación aritmética**
El proceso causado por la ejecución de una sentencia aritmética, o la evaluación de una expresión aritmética, que da como resultado una solución matemáticamente correcta a los argumentos presentados.

*** operador aritmético**
Un único carácter, o una combinación fija de dos caracteres que pertenece al siguiente conjunto:

Carácter	Significado
+	Suma
-	Resta
*	Multiplicación

Carácter	Significado
/	División
**	Exponenciación

*** sentencia aritmética**
Sentencia que hace que se ejecute una operación aritmética. Las sentencias aritméticas son ADD, COMPUTE, DIVIDE, MULTIPLYy SUBTRACT.

matriz
Agregado que consta de objetos de datos, a cada uno de los cuales se puede hacer referencia de forma exclusiva mediante subscripción. Una matriz es aproximadamente análoga a una tabla COBOL.

*** clave ascendente**
Clave sobre los valores de los que se ordenan los datos, empezando por el valor más bajo de la clave hasta el valor más alto de la clave, de acuerdo con las reglas para comparar elementos de datos.

ASCII
American National Standard Code for Information Interchange. El código estándar utiliza un juego de caracteres codificado que se basa en caracteres codificados de 7 bits (8 bits incluyendo comprobación de paridad). El estándar se utiliza para el intercambio de información entre sistemas de procesamiento de datos, sistemas de comunicación de datos y equipo asociado. El conjunto ASCII consta de caracteres de control y caracteres gráficos.

IBM ha definido una extensión para ASCII (caracteres 128-255).

Página de códigos multibyte basada en ASCII
Una página de códigos UTF-8, EUC o ASCII DBCS. Cada página de códigos multibyte basada en ASCII incluye caracteres de un solo byte y de varios bytes. La codificación de los caracteres de un solo byte es la codificación ASCII.

DBCS ASCII
Véase *ASCII de doble byte*.

nombre-asignación
Nombre que identifica la organización de un archivo COBOL y el nombre por el que el sistema lo conoce.

*** punto decimal asumido**
Posición de coma decimal que no implica la existencia de un carácter real en un elemento de datos. La coma decimal asumida tiene un significado lógico pero no tiene representación física.

AT END condición
Una condición que se produce durante la ejecución de una sentencia READ, RETURNo SEARCH bajo determinadas condiciones:

- Una sentencia READ se ejecuta en un archivo al que se accede secuencialmente cuando no existe ningún siguiente registro lógico en el archivo, o cuando el número de dígitos significativos en el número de registro relativo es mayor que el tamaño del elemento de datos de clave relativa, o cuando no está disponible un archivo de entrada opcional.
- Una sentencia RETURN se ejecuta cuando no existe ningún registro lógico siguiente para el archivo de clasificación o fusión asociado.
- Una sentencia SEARCH se ejecuta cuando la operación de búsqueda termina sin satisfacer la condición especificada en ninguna de las frases WHEN asociadas.

B

juego de caracteres básico
El conjunto básico de caracteres utilizados en la escritura de palabras, series de caracteres y separadores del idioma. El juego de caracteres básico se implementa en caracteresde un solo byte. El juego de caracteres ampliado incluye los caracteres DBCS, UTF-8o EUC , que se pueden utilizar en comentarios, literales y palabras definidas por el usuario.

Sinónimo de *juego de caracteres COBOL* en 85 COBOL Estándar.

big-endian
El formato predeterminado que el sistema principal y la estación de trabajo de Linux utilizan para almacenar datos binarios y caracteres UTF-16 . En este formato, el byte menos significativo de un elemento de datos binarios está en la dirección más alta y el byte menos significativo de un carácter UTF-16 está en la dirección más alta. Compare con *little-endian*.

elemento binario
Elemento de datos numérico que se representa en notación binaria (en el sistema de numeración base 2). El equivalente decimal consta de los dígitos decimales del 0 al 9, más un signo operativo. El bit situado más a la izquierda del elemento es el signo operativo.

búsqueda binaria
Una búsqueda de dicotomización en la que, en cada paso de la búsqueda, el conjunto de elementos de datos se divide por dos; se realiza una acción adecuada en el caso de un número impar.

*** bloque**
Unidad física de datos que normalmente se compone de uno o más registros lógicos. Para archivos de almacenamiento masivo, un bloque puede contener una parte de un registro lógico. El tamaño de un bloque no tiene relación directa con el tamaño del archivo en el que está contenido el bloque o con el tamaño de los registros lógicos que están contenidos en el bloque o que solapan el bloque. Sinónimo de *registro físico*.

condición booleana
Una condición booleana determina si un literal booleano es verdadero o falso. Una condición booleana sólo se puede utilizar en una expresión condicional constante.

literal booleano
Puede ser B '1', que indica un valor verdadero, o B '0', que indica un valor falso. Los literales booleanos sólo se pueden utilizar en expresiones condicionales constantes.

punto de interrupción
Lugar de un programa informático, normalmente especificado por una instrucción, donde la intervención externa o un programa supervisor puede interrumpir el programa a medida que se ejecuta.

almacenamiento intermedio
Parte del almacenamiento que se utiliza para contener datos de entrada o salida temporalmente.

Función incorporada
Véase *función intrínseca*.

byte
Serie que consta de un determinado número de bits, normalmente ocho, tratados como una unidad y que representa un carácter o una función de control.

marca de orden de bytes (BOM)
Carácter Unicode que se puede utilizar al principio del texto UTF-16 o UTF-32 para indicar el orden de bytes del texto posterior; el orden de bytes puede ser big-endian o little-endian.

código de bytes
Código independiente de la máquina generado por el compilador Java y ejecutado por el intérprete Java . (Oracle)

C

programa llamado
Programa que es el objeto de una sentencia CALL . En tiempo de ejecución, el programa llamado y el programa de llamada se combinan para producir una *unidad de ejecución*.

*** programa de llamada**
Programa que ejecuta un CALL en otro programa.

estructura de caso
Lógica de proceso de programa en la que se prueba una serie de condiciones para elegir entre varias acciones resultantes.

CCSID
Consulte *identificador de juego de caracteres codificados*.

Ventana de siglo
Intervalo de 100 años dentro del cual cualquier año de dos dígitos es exclusivo. Hay varios tipos de ventana de siglo disponibles para los programadores de COBOL:

- Para los campos de fecha con ventana, utilice la opción de compilador YEARWINDOW .
- Para las funciones intrínsecas de ventana DATE-TO-YYYYMMDD, DAY-TO-YYYYDDDy YEAR-TO-YYYY, especifique la ventana de siglo con *argument-2*.

*** carácter**
Unidad indivisible básica del idioma.

unidad de codificación de caracteres
Unidad de datos que corresponde a un elemento de código de un juego de caracteres codificado. Se utilizan una o más unidades de codificación de caracteres para representar un carácter en un juego de caracteres codificado. También se conoce como *unidad de codificación*.

Para USAGE NATIONAL, una unidad de codificación de caracteres corresponde a un punto de código de 2 bytes de UTF-16.

Para USAGE DISPLAY, una unidad de codificación de caracteres corresponde a un byte.

Para USAGE DISPLAY-1, una unidad de codificación de caracteres corresponde a un elemento de código de 2 bytes en el juego de caracteres DBCS.

posición de carácter
Cantidad de almacenamiento físico o espacio de presentación necesario para contener o presentar un carácter. El término se aplica a cualquier clase de carácter. Para clases específicas de caracteres, se aplican los términos siguientes:

- *Posición de carácter alfanumérico*, para los caracteres representados en USAGE DISPLAY
- *Posición de carácter DBCS*, para caracteres DBCS representados en USAGE DISPLAY-1
- *Posición de carácter nacional*, para los caracteres representados en USAGE NATIONAL; sinónimo de *unidad de codificación de caracteres* para UTF-16

juego de caracteres
Colección de elementos que se utilizan para representar información textual, pero para los que no se presupone ninguna representación codificada. Véase también *juego de caracteres codificado*.

serie de caracteres
Secuencia de caracteres contiguos que forman una palabra COBOL, un literal, una serie de caracteres PICTURE o una entrada de comentario. Una serie de caracteres debe estar delimitada por separadores.

punto de comprobación
Punto en el que se puede registrar información sobre el estado de un trabajo y el sistema para que el paso de trabajo se pueda reiniciar más adelante.

*** clase**
Entidad que define el comportamiento común y la implementación para cero, uno o más objetos. Los objetos que comparten la misma implementación se consideran objetos de la misma clase. Las clases se pueden definir jerárquicamente, lo que permite que una clase herede de otra.

*** condición de clase**
La proposición (para la que se puede determinar un valor de verdad) de que el contenido de un elemento es totalmente alfabético, es totalmente numérico, es totalmente DBCS, es totalmente Kanji, o consiste exclusivamente en los caracteres que se listan en la definición de un nombre de clase.

*** nombre-clase (de datos)**
Una palabra definida por el usuario que está definida en el párrafo SPECIAL-NAMES del ENVIRONMENT DIVISION; esta palabra asigna un nombre a la proposición (para la que se puede definir un valor de verdad) de que el contenido de un elemento de datos consta exclusivamente de los caracteres que se listan en la definición del nombre de clase.

*** cláusula**
Conjunto ordenado de series de caracteres COBOL consecutivas cuya finalidad es especificar un atributo de una entrada.

Conjunto de caracteres COBOL
El conjunto de caracteres utilizados en la escritura de sintaxis COBOL. El juego de caracteres COBOL completo consta de estos caracteres:

Carácter	Significado
0,1, . . . ,9	Dígito
A, B,... , Z	Letra mayúscula
a, b,... , z	Letra minúscula
	Espacio
+	Signo más
-	Signo menos (guión)
*	Asterisco
/	Inclinado (barra inclinada)
=	Signo igual
$	Símbolo de moneda
,	Coma
;	Punto y coma
.	Punto (coma decimal, punto completo)
"	comilla
'	Apóstrofo
(	Paréntesis izquierdo
)	Paréntesis derecho
>	Mayor que
<	Menor que
:	Dos puntos
_	Subrayado

*** Palabra COBOL**
Véase *palabra*.

Página de códigos
Asignación de caracteres gráficos y significados de función de control a todos los elementos de código. Por ejemplo, una página de códigos podría asignar caracteres y significados a 256 puntos de código para código de 8 bits, y otra página de códigos podría asignar caracteres y significados a 128 puntos de código para código de 7 bits. Por ejemplo, una de las páginas de códigos de IBM para inglés en la estación de trabajo es IBM-1252 y en el host es IBM-1047.

elemento de código
Patrón de bits exclusivo que se define en un juego de caracteres codificado (página de códigos). Los símbolos gráficos y los caracteres de control se asignan a los elementos de código.

juego de caracteres codificado
Conjunto de reglas inequívocas que establecen un juego de caracteres y la relación entre los caracteres del conjunto y su representación codificada. Ejemplos de juegos de caracteres codificados son los juegos de caracteres representados por páginas de códigos ASCII o EBCDIC o por el esquema de codificación UTF-16 para Unicode.

identificador de juego de caracteres codificados (CCSID)
Un número definido por IBMen el rango de 1 a 65.535 que identifica una página de códigos específica.

*** orden de clasificación**
Secuencia en la que los caracteres que son aceptables para un sistema se ordenan para fines de clasificación, fusión, comparación y para procesar archivos indexados secuencialmente.

*** columna**
Posición de byte dentro de una línea de impresión o dentro de una línea de formato de referencia. Las columnas se numeran desde 1, por 1, empezando por la posición más a la izquierda de la línea y extendiéndose hasta la posición más a la derecha de la línea. Una columna contiene un carácter de un solo byte.

*** condición combinada**
Condición que es el resultado de conectar dos o más condiciones con el operador lógico AND o OR . Véase también *condición* y *condición combinada negada*.

*** entrada-comentario**
Una entrada en IDENTIFICATION DIVISION que se utiliza para la documentación y no tiene ningún efecto en la ejecución.

línea de comentario
Una línea de programa de origen representada por un asterisco (*) en el área de indicador de la línea o por un asterisco seguido por un signo mayor que (*>) como la primera serie de caracteres en el área de texto del programa (Área A más Área B), y cualquier carácter del juego de caracteres del sistema que sigue a en el Área A y el Área B de esa línea. Una línea de comentario sólo sirve para la documentación. Una forma especial de línea de comentario representada por una barra inclinada (/) en el área de indicador de la línea y cualquier carácter del juego de caracteres del sistema en el Área A y el Área B de esa línea provoca la expulsión de la página antes de que se imprima el comentario.

*** programa común**
Un programa que, a pesar de estar directamente contenido dentro de otro programa, puede ser llamado desde cualquier programa directa o indirectamente contenido en ese otro programa.

campo de fecha compatible
El significado del término *compatible*, cuando se aplica a campos de fecha, depende de la división COBOL en la que se produce el uso:

- DATA DIVISION: dos campos de fecha son compatibles si tienen USAGE idénticos y cumplen al menos una de las condiciones siguientes:
  - Tienen el mismo formato de fecha.
  - Ambos son campos de fecha con ventana, donde uno solo consta de un año con ventana, DATE FORMAT YY.
  - Ambos son campos de fecha expandidos, donde uno solo consta de un año expandido, DATE FORMAT YYYY.
  - Uno tiene DATE FORMAT YYXXXXy el otro tiene YYXX.
  - Uno tiene DATE FORMAT YYYYXXXXy el otro tiene YYYYXX.

  Un campo de fecha con ventana puede estar subordinado a un elemento de datos que es un grupo de fechas expandido. Los dos campos de fecha son compatibles si el campo de fecha subordinado tiene USAGE DISPLAY, empieza dos bytes después del inicio del campo de fecha expandida de grupo y los dos campos cumplen al menos una de las condiciones siguientes:
  - El campo de fecha subordinado tiene un patrón DATE FORMAT con el mismo número de X que el patrón DATE FORMAT del campo de fecha de grupo.
  - El campo de fecha subordinada tiene DATE FORMAT YY.
  - El campo de fecha de grupo tiene DATE FORMAT YYYYXXXX y el campo de fecha subordinado tiene DATE FORMAT YYXX.
- PROCEDURE DIVISION: Dos campos de fecha son compatibles si tienen el mismo formato de fecha excepto para la parte del año, que se puede incluir en ventanas o expandirse. Por ejemplo, un campo de fecha con ventana con DATE FORMAT YYXXX es compatible con:

- Otro campo de fecha con ventana con DATE FORMAT YYXXX
- Un campo de fecha expandido con DATE FORMAT YYYYXXX

*** compilar**
(1) Para traducir un programa expresado en un lenguaje de alto nivel a un programa expresado en un lenguaje intermedio, lenguaje ensamblador, o un lenguaje de computadora. (2) Preparar un programa de lenguaje de máquina a partir de un programa de computadora escrito en otro lenguaje de programación haciendo uso de la estructura lógica general del programa, o generando más de una instrucción de computadora para cada declaración simbólica, o ambas, así como realizando la función de un ensamblador.

variable de compilación
Un nombre simbólico para un valor literal determinado o el valor de una expresión aritmética de tiempo de compilación tal como especifica la directiva DEFINE o la opción de compilador DEFINE .

*** tiempo de compilación**
La hora a la que el código fuente COBOL se convierte, mediante un compilador COBOL, en un programa de objeto COBOL.

expresión aritmética en tiempo de compilación
Subconjunto de expresiones aritméticas que se especifican en las directivas DEFINE y EVALUATE o en una expresión condicional constante. La diferencia entre las expresiones aritméticas en tiempo de compilación y las expresiones aritméticas regulares es que en una expresión aritmética en tiempo de compilación:

- No se especificará el operador de exponenciación.
- Todos los operandos serán literales numéricos enteros o expresiones aritméticas en las que todos los operandos son literales numéricos enteros.
- La expresión se especificará de tal manera que no se produzca una división por cero.

compilador
Programa que convierte el código fuente escrito en un lenguaje de nivel superior en un código de objeto de lenguaje de máquina.

sentencia de direccionamiento de compilador
Sentencia que hace que el compilador realice una acción específica durante la compilación. Las sentencias de direccionamiento de compilador estándar son COPY, REPLACEy USE.

Directiva de compilador
Directiva que hace que el compilador realice una acción específica durante la compilación. COBOL para Linux da soporte a la directiva de compilador CALLINTERFACE , así como a las directivas de compilador de compilación condicional (DEFINE, EVALUATE e IF).

*** condición compleja**
Condición en la que uno o más operadores lógicos actúan sobre una o más condiciones. Véase también *condición*, *condición simple negada*y *condición combinada negada*.

ODO complejo
Ciertas formas de la cláusula OCCURS DEPENDING ON :

- Elemento o grupo de ubicación variable: un elemento de datos descrito por una cláusula OCCURS con la opción DEPENDING ON va seguido de un elemento o grupo de datos no subordinado. El grupo puede ser un grupo alfanumérico o un grupo nacional.
- Tabla de ubicación variable: un elemento de datos descrito por una cláusula OCCURS con la opción DEPENDING ON va seguido de un elemento de datos no subordinado descrito por una cláusula OCCURS .
- Tabla con elementos de longitud variable: un elemento de datos descrito por una cláusula OCCURS contiene un elemento de datos subordinado descrito por una cláusula OCCURS con la opción DEPENDING ON .
- Nombre de índice para una tabla con elementos de longitud variable.
- Elemento de una tabla con elementos de longitud variable.

Componente
(1) Una agrupación funcional de archivos relacionados. (2) En programación orientada a objetos, un objeto o programa reutilizable que realiza una función específica y está diseñado para trabajar con otros componentes y aplicaciones. JavaBeans es la arquitectura de Oraclepara crear componentes.

*** nombre-sistema**
Nombre del sistema que identifica el sistema en el que se va a compilar o ejecutar el programa.

condición (excepción)
Cualquier alteración del flujo normal programado de una aplicación. Las condiciones pueden ser detectadas por el hardware o el sistema operativo y dar como resultado una interrupción. También se pueden detectar mediante código generado específico del idioma o código de biblioteca de idioma.

condición (expresión)
Estado de los datos en tiempo de ejecución para los que se puede determinar un valor de verdad. Donde se utiliza en esta información en o en referencia a "condición" (*condition-1*, *condition-2*,.). de un formato general, el término se refiere a una expresión condicional que consiste en una condición simple opcionalmente entre paréntesis o una condición combinada (que consiste en la combinación sintácticamente correcta de condiciones simples, operadores lógicos y paréntesis) para los que se puede determinar un valor de verdad. Véase también *condición simple*, *condición compleja*, *condición simple negada*, *condición combinada*, y *condición combinada negada*.

*** expresión condicional**
Una condición simple o una condición compleja especificada en una sentencia EVALUATE, IF, PERFORMo SEARCH . Véase también *condición simple* y *condición compleja*.

*** frase condicional**
Frase que especifica la acción que se debe realizar al determinar el valor de verdad de una condición que resulta de la ejecución de una sentencia condicional.

*** sentencia condicional**
Sentencia que especifica que el valor de verdad de una condición debe determinarse y que la acción subsiguiente del programa objeto depende de este valor de verdad.

*** variable condicional**
Un elemento de datos uno o más valores de los cuales tiene asignado un nombre de condición.

*** nombre-condición**
Palabra definida por el usuario que asigna un nombre a un subconjunto de valores que una variable condicional puede asumir; o una palabra definida por el usuario asignada a un estado de un conmutador o dispositivo definido por el implementador.

*** condición de nombre-condición**
La proposición (para la cual se puede determinar un valor de verdad) de que el valor de una variable condicional es un miembro del conjunto de valores atribuidos a un nombre-condición asociado con la variable condicional.

*** `CONFIGURATION SECTION`**
Una sección de `ENVIRONMENT DIVISION` que describe las especificaciones generales de los programas de origen y objeto.

CONSOLE
Un nombre de entorno COBOL asociado a la consola del operador.

expresión condicional constante
Subconjunto de expresiones condicionales que se pueden utilizar en `IF` directivas o `WHEN` frases de las directivas `EVALUATE` .

Una expresión condicional constante será uno de los siguientes elementos:

- Condición de relación en la que ambos operandos son literales o expresiones aritméticas que sólo contienen términos literales. La condición se regirá por las normas relativas a las condiciones de relación, con las siguientes adiciones:
  - Los operandos serán de la misma categoría. Una expresión aritmética es de la categoría numérica.

– Si se especifican literales y no son literales numéricos, el operador relacional será “IS EQUAL TO”, “IS NOT EQUAL TO”, “IS =”, “IS NOT =”o “IS <>”.

Véase también *condición de relación*.

- Una condición definida. Véase también *condición definida*.
- Una condición booleana. Véase también *condición booleana*.
- Una condición compleja formada combinando las formas anteriores de condiciones simples en condiciones complejas utilizando AND, OR y NOT. No se especificarán las condiciones de relación combinadas abreviadas. Véase también *condición compleja*.

programa contenido
Un programa COBOL que está anidado dentro de otro programa COBOL.

*** elementos contiguos**
Elementos que se describen mediante entradas consecutivas en DATA DIVISIONy que tienen una relación jerárquica definida entre sí.

libro de copias
Archivo o miembro de biblioteca que contiene una secuencia de código que se incluye en el programa fuente durante la compilación utilizando la sentencia COPY . El archivo puede ser creado por el usuario, proporcionado por COBOL, o proporcionado por otro producto. Sinónimo de *archivo de copia*.

*** contador**
Elemento de datos utilizado para almacenar números o representaciones de números de una manera que permite que estos números se incrementen o disminuyan por el valor de otro número, o que se cambien o restablezcan a cero o a un valor positivo o negativo arbitrario.

Listado de referencias cruzadas
Parte del listado del compilador que contiene información sobre dónde se definen, hacen referencia y modifican los archivos, los campos y los indicadores en un programa.

valor de signo de moneda
Serie de caracteres que identifica las unidades monetarias almacenadas en un elemento editado numérica-. Los ejemplos típicos son $, USD y EUR. Un valor de signo de moneda se puede definir mediante la opción de compilador CURRENCY o la cláusula CURRENCY SIGN en el párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION. Si no se especifica la cláusula CURRENCY SIGN y la opción de compilador NOCURRENCY está en vigor, se utiliza el signo de dólar ($) como valor de signo de moneda predeterminado. Véase también *símbolo de moneda*.

Símbolo de moneda
Carácter utilizado en una cláusula PICTURE para indicar la posición de un valor de signo de moneda en un elemento editado numérica-editado. Un símbolo de moneda se puede definir mediante la opción de compilador CURRENCY o la cláusula CURRENCY SIGN en el párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION. Si no se especifica la cláusula CURRENCY SIGN y la opción de compilador NOCURRENCY está en vigor, se utiliza el signo de dólar ($) como valor de signo de moneda predeterminado y símbolo de moneda. Se pueden definir varios símbolos de moneda y valores de signo de moneda. Véase también *valor de signo de moneda*.

*** registro actual**
En el proceso de archivos, el registro que está disponible en el área de registro asociada a un archivo.

*** puntero de volumen actual**
Entidad conceptual que apunta al volumen actual de un archivo secuencial.

D

*** cláusula de datos**
Cláusula, que aparece en una entrada de descripción de datos en el DATA DIVISION de un programa COBOL, que proporciona información que describe un atributo determinado de un elemento de datos.

*** entrada de descripción de datos**
Una entrada en el DATA DIVISION de un programa COBOL que se compone de un número de nivel seguido de un nombre de datos, si es necesario, y seguido de un conjunto de cláusulas de datos, según sea necesario.

DATA DIVISION
La división de un programa COBOL que describe los datos que debe procesar el programa: los archivos que se van a utilizar y los registros contenidos en ellos; los registros WORKING-STORAGE internos que serán necesarios; los datos que estarán disponibles en más de un programa en la unidad de ejecución COBOL.

*** elemento de datos**
Unidad de datos (excluyendo literales) definida por un programa COBOL o por las reglas para la evaluación de funciones.

*** nombre-datos**
Palabra definida por el usuario que nombra un elemento de datos descrito en una entrada de descripción de datos. Cuando se utiliza en los formatos generales, el nombre de datos representa una palabra que no debe ser modificada por referencia, subclasificada o calificada a menos que lo permitan específicamente las reglas para el formato.

campo de fecha
Cualquiera de los elementos siguientes:

- Elemento de datos cuya entrada de descripción de datos incluye una cláusula DATE FORMAT .
- Un valor devuelto por una de las siguientes funciones intrínsecas:

```
DATE-OF-INTEGER
DATE-TO-YYYYMMDD
DATEVAL
DAY-OF-INTEGER
DAY-TO-YYYYDDD
YEAR-TO-YYYY
YEARWINDOW
```

- Los elementos de datos conceptuales DATE, DATE YYYYMMDD, DAY y DAY YYYYDDD de la sentencia ACCEPT .
- El resultado de determinadas operaciones aritméticas. Para obtener detalles, consulte Aritmética con campos de fecha (*COBOL for Linux en x86 Consulta de lenguaje*).

El término *campo de fecha* hace referencia tanto al *campo de fecha expandida* como al *campo de fecha con ventana*. Véase también *no fecha*.

formato de fecha
El patrón de fecha de un campo de fecha, especificado de cualquiera de las formas siguientes:

- Explícitamente, mediante la cláusula DATE FORMAT o la función intrínseca DATEVAL argument-2
- Implícitamente, por sentencias y funciones intrínsecas que devuelven campos de fecha. Para obtener detalles, consulte Campo de fecha (*COBOL for Linux en x86 Consulta de lenguaje*).

Sistema de archivosDb2
El sistema de archivos Db2 admite archivos secuenciales, indexados y relativos. Proporciona una interoperación mejorada con CICS, lo que permite a los programas COBOL por lotes acceder a los archivos CICS ESDS, KSDS y RRDS almacenados en Db2.

DBCS
Véase *juego de caracteres de doble byte (DBCS)*.

Carácter DBCS
Cualquier carácter definido en el juego de caracteres de doble byte de IBM.

Posición de carácter DBCS
Véase *posición de carácter*.

elemento de datos DBCS
Un elemento de datos que se describe mediante una serie de caracteres PICTURE que contiene al menos un símbolo Go, cuando la opción de compilador NSYMBOL(DBCS) está en vigor, al menos un símbolo N. Un elemento de datos DBCS tiene USAGE DISPLAY-1.

*** línea de depuración**
Cualquier línea con una D en el área de indicador de la línea.

*** sección de depuración**
Una sección que contiene una sentencia `USE FOR DEBUGGING` .

*** frase declarativa**
Una sentencia de direccionamiento de compilador que consta de una única sentencia USE terminada por el punto separador.

*** declarativas**
Un conjunto de una o más secciones de propósito especial, escritas al principio de `PROCEDURE DIVISION`, la primera de las cuales va precedida de la palabra clave `DECLARATIVE` y la última de las cuales va seguida de las palabras clave `END DECLARATIVES`. Una declarativa se compone de una cabecera de sección, seguida de una frase de dirección del compilador USE , seguida de un conjunto de cero, uno o más párrafos asociados.

*** deseditar**
Eliminación lógica de todos los caracteres de edición de un elemento de datos editado numérico para determinar el valor numérico no editado del elemento.

condición definida
Condición de tiempo de compilación que comprueba si se ha definido una variable de compilación. Las condiciones definidas se especifican en `IF` directivas o `WHEN` frases de las directivas `EVALUATE` .

*** sentencia de ámbito delimitado**
Cualquier sentencia que incluya su terminador de ámbito explícito.

*** delimitador**
Carácter o secuencia de caracteres contiguos que identifican el final de una serie de caracteres y separan dicha serie de caracteres de la siguiente serie de caracteres. Un delimitador no forma parte de la serie de caracteres que delimita.

*** clave descendente**
Clave sobre cuyos valores se ordenan los datos empezando por el valor más alto de clave hasta el valor más bajo de clave, de acuerdo con las reglas para comparar elementos de datos.

dígito
Cualquiera de los números del 0 al 9. En COBOL, el término no se utiliza para hacer referencia a ningún otro símbolo.

*** posición de dígito**
La cantidad de almacenamiento físico necesaria para almacenar un único dígito. Esta cantidad puede variar en función del uso especificado en la entrada de descripción de datos que define el elemento de datos.

*** acceso directo**
El recurso para obtener datos de dispositivos de almacenamiento o para introducir datos en un dispositivo de almacenamiento de tal manera que el proceso dependa solamente de la ubicación de esos datos y no de una referencia a los datos a los que se ha accedido previamente.

visualizar elemento de datos de coma flotante
Elemento de datos que se describe implícita o explícitamente como `USAGE DISPLAY` y que tiene una serie de caracteres `PICTURE` que describe un elemento de datos de coma flotante externo.

*** división**
Colección de cero, una o más secciones o párrafos, denominada cuerpo de división, que se forman y combinan de acuerdo con un conjunto específico de reglas. Cada división consta de la cabecera de división y el cuerpo de división relacionado. Hay cuatro divisiones en un programa COBOL: Identificación, Entorno, Datos y Procedimiento.

*** cabecera de división**
Combinación de palabras seguida de un punto de separación que indica el principio de una división. Las cabeceras de división son:

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
```

```
DATA DIVISION.
PROCEDURE DIVISION.
```

realizar construcción
En la programación estructurada, se utiliza una sentencia DO para agrupar una serie de sentencias en un procedimiento. En COBOL, una sentencia PERFORM en línea funciona de la misma forma.

hacer hasta
En la programación estructurada, se ejecutará un bucle do-until al menos una vez, y hasta que se cumpla una condición determinada. En COBOL, una frase TEST AFTER utilizada con la sentencia PERFORM funciona de la misma forma.

hacer mientras
En la programación estructurada, se ejecutará un bucle do-while si, y mientras, se cumple una condición determinada. En COBOL, una frase TEST BEFORE utilizada con la sentencia PERFORM funciona de la misma forma.

declaración de tipo de documento
Elemento XML que contiene o apunta a declaraciones de marcación que proporcionan una gramática para una clase de documentos. Esta gramática se conoce como definición de tipo de documento o DTD.

definición de tipo de documento (DTD)
Gramática de una clase de documentos XML. Véase *declaración de tipo de documento*.

ASCII de doble byte
Un juego de caracteres IBM que incluye caracteres DBCS y ASCII de un solo byte. (También conocido como DBCS ASCII.)

EBCDIC de doble byte
Un juego de caracteres IBM que incluye caracteres DBCS y EBCDIC de un solo byte. (También conocido como EBCDIC DBCS.)

juego de caracteres de doble byte (DBCS)
Conjunto de caracteres en el que cada carácter está representado por 2 bytes. Los idiomas como el japonés, el chino y el coreano, que contienen más símbolos de los que pueden representarse mediante 256 elementos de código, requieren juegos de caracteres de doble byte. Debido a que cada carácter requiere 2 bytes, la entrada, visualización e impresión de caracteres DBCS requiere hardware y software de soporte que sea compatible con DBCS.

DWARF
DWARF ha sido desarrollado por UNIX International Programming Languages Special Interest Group (SIG). Está diseñado para satisfacer las necesidades simbólicas de depuración a nivel de fuente de diferentes lenguajes de forma unificada proporcionando información de depuración independiente del lenguaje. Un archivo DWARF contiene datos de depuración organizados en distintos elementos. Para obtener más información, consulte *Información del programa DWARF* en la publicación *Referencia de biblioteca de extensiones DWARF/ELF*.

*** acceso dinámico**
Modalidad de acceso en la que se pueden obtener o colocar registros lógicos específicos en un archivo de almacenamiento masivo de forma no secuencial y se pueden obtener de un archivo de forma secuencial durante el ámbito de la misma sentencia OPEN .

Llamada dinámica
Una sentencia CALL *literal* en un programa que se ha compilado con la opciónde DYNAM , o una sentencia CALL *identificador* en un programa.

E

*** EBCDIC (Código de intercambio decimal ampliado codificado en binario)**
Juego de caracteres codificado basado en caracteres codificados de 8 bits.

Carácter EBCDIC
Cualquiera de los símbolos incluidos en el conjunto EBCDIC (Extended Binary-Coded-Decimal Interchange Code).

EBCDIC DBCS
Véase *EBCDIC de doble byte*.

elemento de datos editado
Elemento de datos que se ha modificado suprimiendo ceros o insertando caracteres de edición o ambos.

*** carácter de edición**
Un único carácter o una combinación fija de dos caracteres que pertenece al siguiente conjunto:

Carácter	Significado
	Espacio
0	ZERO
+	Signo más
-	Menos
PC	Crédito
BD	Débito
Z	Supresión de cero
*	Comprobar protección
$	Símbolo de moneda
,	Coma (coma decimal)
.	Punto (coma decimal)
/	Inclinado (barra inclinada)

elemento (elemento de texto)
Unidad lógica de una serie de texto, como la descripción de un único elemento de datos o verbo, precedida de un código exclusivo que identifica el tipo de elemento.

*** elemento elemental**
Elemento de datos que se describe como no subdividido lógicamente.

Sistema de archivos SFS deCICS
Véase *sistema de archivos SFS*.

unidad de codificación
Véase *unidad de codificación de caracteres*.

*** fin de PROCEDURE DIVISION**
La posición física de un programa fuente COBOL después de la cual no aparecen más procedimientos.

*** finalizar marcador de programa**
Combinación de palabras, seguida de un punto de separación, que indica el final de un programa fuente COBOL. El marcador de programa final es:

```
END PROGRAM program-name.
```

*** entrada**
Cualquier conjunto descriptivo de cláusulas consecutivas terminadas por un punto de separación y escritas en IDENTIFICATION DIVISION, ENVIRONMENT DIVISIONo DATA DIVISION de un programa COBOL.

*** cláusula de entorno**
Cláusula que aparece como parte de una entrada ENVIRONMENT DIVISION .

ENVIRONMENT DIVISION
Una de las cuatro partes de componente principales de un programa COBOL. El ENVIRONMENT DIVISION describe los sistemas en los que se compila el programa fuente y aquellos en los que

se ejecuta el programa objeto. Proporciona un enlace entre el concepto lógico de archivos y sus registros, y los aspectos físicos de los dispositivos en los que se almacenan los archivos.

nombre-entorno
Nombre, especificado por IBM, que identifica las unidades lógicas del sistema, los caracteres de control de perforación de impresora y tarjeta, los códigos de informe, los conmutadores de programa o todos ellos. Cuando un nombre de entorno está asociado con un nombre mnemotécnico en ENVIRONMENT DIVISION, el nombre mnemotécnico se puede sustituir en cualquier formato en el que dicha sustitución sea válida.

entorno, variable
Cualquiera de una serie de variables que definen algún aspecto del entorno informático y son accesibles a los programas que operan en dicho entorno. Las variables de entorno pueden afectar al comportamiento de los programas que son sensibles al entorno en el que operan.

tiempo de ejecución
Véase *tiempo de ejecución*.

entorno de ejecución
Véase *entorno de ejecución*.

campo de fecha expandida
Campo de fecha que contiene un año expandido (cuatro dígitos). Véase también *campo de fecha* y *año expandido*.

año expandido
Campo de fecha que consta sólo de un año de cuatro dígitos. Su valor incluye el siglo: por ejemplo, 1998. Compárese con *año de ventana*.

*** terminador de ámbito explícito**
Palabra reservada que termina el ámbito de una sentencia PROCEDURE DIVISION determinada.

exponente
Número que indica la potencia a la que se va a elevar otro número (la base). Los exponentes positivos denotan multiplicación; los exponentes negativos denotan división; y los exponentes fraccionados denotan una raíz de una cantidad. En COBOL, una expresión exponencial se indica con el símbolo ** seguido del exponente.

*** expresión**
Una expresión aritmética o condicional.

*** modalidad de ampliación**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase EXTEND especificada para dicho archivo, y antes de la ejecución de una sentencia CLOSE , sin la frase REEL o UNIT para dicho archivo.

Lenguaje de códigos ampliable
Véase *XML*.

ampliaciones
Sintaxis y semántica COBOL soportadas por los compiladores de IBM además de las descritas en 85 COBOL Estándar.

página de códigos externa
Para documentos XML ASCII o UTF-8 , la página de códigos indicada por el entorno local de tiempo de ejecución actual. Para documentos XML EBCDIC, o bien:

- La página de códigos especificada en la variable de entorno EBCDIC_CODEPAGE
- La página de códigos EBCDIC predeterminada seleccionada para el entorno local de tiempo de ejecución actual si la variable de entorno EBCDIC_CODEPAGE no está establecida

*** datos externos**
Los datos que se describen en un programa como elementos de datos externos y conectores de archivos externos.

*** elemento de datos externo**
Elemento de datos que se describe como parte de un registro externo en uno o más programas de una unidad de ejecución y al que se puede hacer referencia desde cualquier programa en el que se describe.

*** registro de datos externo**
Registro lógico que se describe en uno o más programas de una unidad de ejecución y cuyos elementos de datos constitutivos pueden referenciarse desde cualquier programa en el que se describen.

elemento de datos decimales externos
Véase *elemento de datos decimal con zona* y *elemento de datos decimal nacional*.

*** conector de archivo externo**
Conector de archivo al que pueden acceder uno o varios programas de objeto de la unidad de ejecución.

elemento de datos de coma flotante externo
Véase *elemento de datos de coma flotante de visualización* y *elemento de datos de coma flotante nacional*.

Programa externo
El programa más externo. Un programa que no está anidado.

*** conmutador externo**
Dispositivo de hardware o software, definido y nombrado por el implementador, que se utiliza para indicar que existe uno de los dos estados alternativos.

F

*** constante figurativa**
Valor generado por el compilador al que se hace referencia mediante el uso de determinadas palabras reservadas.

*** archivo**
Colección de registros lógicos.

*** condición de conflicto de atributo de archivo**
Se ha realizado un intento no satisfactorio de ejecutar una operación de entrada-salida en un archivo y los atributos de archivo, tal como se ha especificado para ese archivo en el programa, no coinciden con los atributos fijos para ese archivo.

*** cláusula file**
Cláusula que aparece como parte de cualquiera de las siguientes entradas de `DATA DIVISION`: entrada de descripción de archivo (entrada`FD`) y entrada de descripción de archivo de fusión de clasificación (entrada`SD`).

*** conector de archivo**
Área de almacenamiento que contiene información sobre un archivo y se utiliza como enlace entre un nombre de archivo y un archivo físico y entre un nombre de archivo y su área de registro asociada.

*** entrada de control de archivo**
Una cláusula `SELECT` y todas sus cláusulas subordinadas que declaran los atributos físicos relevantes de un archivo.

`FILE-CONTROL` párrafo
Un párrafo en el `ENVIRONMENT DIVISION` en el que se declaran los archivos de datos para una unidad de origen determinada.

*** entrada de descripción de archivo**
Una entrada en `FILE SECTION` del `DATA DIVISION` que se compone del indicador de nivel `FD`, seguido de un nombre de archivo y, a continuación, seguido de un conjunto de cláusulas de archivo según sea necesario.

*** nombre-archivo**
Palabra definida por el usuario que nombra un conector de archivo descrito en una entrada de descripción de archivo o una entrada de descripción de archivo de fusión de clasificación dentro del `FILE SECTION` de `DATA DIVISION`.

*** organización de archivos**
Estructura de archivo lógico permanente establecida en el momento en que se crea un archivo.

indicador de posición de archivo
Entidad conceptual que contiene el valor de la clave actual dentro de la clave de referencia para un archivo indexado, o el número de registro del registro actual para un archivo secuencial, o el número de registro relativo del registro actual para un archivo relativo, o indica que no existe ningún registro lógico siguiente, o que no está disponible un archivo de entrada opcional, o que ya existe la condición `AT END`, o que no se ha establecido ningún registro siguiente válido.

*** `FILE SECTION`**
La sección de `DATA DIVISION` que contiene entradas de descripción de archivo y entradas de descripción de archivo de fusión de clasificación junto con sus descripciones de registro asociadas.

Sistema de archivos
Colección de archivos que se ajustan a un conjunto específico de protocolos de registro de datos y de descripción de archivos, y a un conjunto de programas que gestionan estos archivos.

*** atributos de archivo fijo**
Información sobre un archivo que se establece cuando se crea un archivo y que no se puede cambiar posteriormente durante la existencia del archivo. Estos atributos incluyen la organización del archivo (secuencial, relativo o indexado), la clave de registro principal, las claves de registro alternativas, el conjunto de códigos, el tamaño mínimo y máximo de registro, el tipo de registro (fijo o variable), la secuencia de clasificación de las claves para archivos indexados, el factor de bloqueo, el carácter de relleno y el delimitador de registro.

*** registro de longitud fija**
Un registro asociado a un archivo cuya descripción de archivo o entrada de descripción de fusión de clasificación requiere que todos los registros contengan el mismo número de bytes.

elemento de punto fijo
Elemento de datos numérico definido con una cláusula `PICTURE` que especifica la ubicación de un signo opcional, el número de dígitos que contiene y la ubicación de una coma decimal opcional. El formato puede ser binario, decimal empaquetado o decimal externo.

indicadores de comentario flotante (* >)
Un indicador de comentario flotante indica una línea de comentario si es la primera serie de caracteres en el área de texto de programa (Área A más Área B), o indica un comentario en línea si está detrás de una o más series de caracteres en el área de texto de programa.

floating point
Formato para representar números en el que un número real se representa mediante un par de números distintos. En una representación de coma flotante, el número real es el producto de la parte de punto fijo (el primer numeral) y un valor obtenido elevando la base de coma flotante implícita a una potencia indicada por el exponente (el segundo numeral). Por ejemplo, una representación de coma flotante del número 0.0001234 es 0.1234 -3, donde 0.1234 es la mantisa y -3 es el exponente.

elemento de datos de coma flotante
Elemento de datos numérico que contiene una fracción y un exponente. Su valor se obtiene multiplicando la fracción por la base del elemento de datos numérico elevado a la potencia que el exponente especifica.

*** formato**
Organización específica de un conjunto de datos.

*** función**
Elemento de datos temporal cuyo valor se determina en el momento en que se hace referencia a la función durante la ejecución de una sentencia.

*** identificador-función**
Combinación sintácticamente correcta de series de caracteres y separadores que hacen referencia a una función. El elemento de datos representado por una función se identifica de forma exclusiva mediante un nombre de función con sus argumentos, si los hay. Un identificador de función puede incluir un modificador de referencia. Un identificador de función que hace referencia a una función alfanumérica se puede especificar en cualquier lugar de los formatos generales en los que se puede especificar un identificador, sujeto a determinadas restricciones. Se puede hacer referencia a un

identificador de función que hace referencia a una función entera o numérica en cualquier lugar de los formatos generales en los que se puede especificar una expresión aritmética.

nombre-función
Palabra que nombra el mecanismo cuya invocación, junto con los argumentos necesarios, determina el valor de una función.

elemento de datos de puntero de función
Elemento de datos en el que se puede almacenar un puntero a un punto de entrada. Un elemento de datos definido con la cláusula `USAGE IS FUNCTION-POINTER` contiene la dirección de un punto de entrada de función. Normalmente se utiliza para comunicarse con programas C y Java .

G

Recogida de basura
La liberación automática por parte del sistema de tiempo de ejecución Java de la memoria para objetos a los que ya no se hace referencia.

GDG
Véase *grupo de datos de generación (GDG)*.

GDS
Véase *conjunto de datos de generación (GDS)*.

grupo de datos de generación (GDG)
Colección de archivos relacionados cronológicamente; cada uno de estos archivos se denomina un *conjunto de datos de generación (GDS)* o generación.

conjunto de datos de generación (GDS)
Uno de los archivos de un *grupo de datos de generación (GDG)*; cada archivo de este tipo está relacionado cronológicamente con los otros archivos del grupo.

*** nombre global**
Nombre que se declara en un solo programa pero al que se puede hacer referencia desde el programa y desde cualquier programa contenido en el programa. Los nombres de condición, los nombres de datos, los nombres de archivo, los nombres de registro, los nombres de informe y algunos registros especiales pueden ser nombres globales.

elemento de grupo
(1) Un elemento de datos que se compone de elementos de datos subordinados. Véase *elemento de grupo alfanumérico* y *elemento de grupo nacional*. (2) Cuando no se califique explícitamente o por contexto como grupo nacional o grupo alfanumérico, el término se refiere a los grupos en general.

separador de agrupación
Carácter utilizado para separar unidades de dígitos en números para facilitar la lectura. El valor predeterminado es la coma de carácter.

H

etiqueta de cabecera
(1) Una etiqueta de que precede a los registros de datos en una unidad de soporte de grabación. (2) Sinónimo de *etiqueta de inicio de archivo*.

*** fin de orden superior**
El carácter situado más a la izquierda de una serie de caracteres.

elemento de datos alfanuméricos de host
(De documentos XML) Elemento de datos alfanuméricos de categoría cuya entrada de descripción de datos no contiene la frase `NATIVE` y que se ha compilado con la opción `CHAR(EBCDIC)` en vigor. La codificación del elemento de datos es la página de códigos EBCDIC en vigor. Esta página de códigos se determina a partir de la variable de entorno EBCDIC_CODEPAGE, si se establece, de lo contrario a partir de la página de códigos predeterminada asociada con el entorno local de ejecución.

I

IBM Extensión COBOL
Sintaxis y semántica COBOL soportadas por los compiladores de IBM además de las descritas en 85 COBOL Estándar.

ICU
Consulte *International Components for Unicode (ICU)*.

IDENTIFICATION DIVISION
Una de las cuatro partes de componente principales de un programa COBOL. `IDENTIFICATION DIVISION` identifica el programa, la clase. `IDENTIFICATION DIVISION` puede incluir la siguiente documentación: nombre de autor, instalación o fecha.

*** identificador**
Combinación sintácticamente correcta de series de caracteres y separadores que nombra un elemento de datos. Al hacer referencia a un elemento de datos que no es una función, un identificador consta de un nombre de datos, junto con sus calificadores, subíndices y modificador de referencia, según sea necesario para la exclusividad de la referencia. Cuando se hace referencia a un elemento de datos que es una función, se utiliza un identificador de función.

*** sentencia imperativa**
Sentencia que empieza con un verbo imperativo y especifica una acción incondicional que se debe realizar o es una sentencia condicional delimitada por su terminador de ámbito explícito (sentencia de ámbito delimitado). Una sentencia imperativa puede consistir en una secuencia de sentencias imperativas.

*** terminador de ámbito implícito**
Un punto de separación que termina el ámbito de cualquier sentencia no terminada anterior, o una frase de una sentencia que por su aparición indica el final del ámbito de cualquier sentencia contenida en la frase anterior.

*** índice**
Un área de almacenamiento de computadora o registro, cuyo contenido representa la identificación de un elemento particular en una tabla.

*** elemento de datos de índice**
Elemento de datos en el que los valores asociados a un nombre de índice se pueden almacenar en un formato especificado por el implementador.

nombre-datos-indexado
Identificador compuesto de un nombre de datos, seguido de uno o más nombres de índice entre paréntesis.

*** archivo indexado**
Un archivo con una organización indexada.

*** organización indexada**
Estructura de archivo lógico permanente en la que cada registro se identifica mediante el valor de una o más claves dentro de ese registro.

indexación
Sinónimo de *subscripción* utilizando nombres de índice.

*** nombre-índice**
Palabra definida por el usuario que nombra un índice asociado a una tabla específica.

*** programa inicial**
Programa que se coloca en un estado inicial cada vez que se llama al programa en una unidad de ejecución.

*** estado inicial**
El estado de un programa cuando se llama por primera vez en una unidad de ejecución.

inline
En un programa, instrucciones que se ejecutan secuencialmente, sin ramificar a rutinas, subrutinas u otros programas.

*** archivo de entrada**
Un archivo que se abre en la modalidad de entrada.

*** modalidad de entrada**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase INPUT especificada, para dicho archivo y antes de la ejecución de una sentencia CLOSE , sin la frase REEL o UNIT para dicho archivo.

*** archivo de entrada-salida**
Un archivo que se abre en modalidad I-O .

*** INPUT-OUTPUT SECTION**
La sección de ENVIRONMENT DIVISION que nombra los archivos y el soporte externo que necesita un programa objeto y que proporciona información necesaria para la transmisión y el manejo de datos en tiempo de ejecución.

*** sentencia de entrada-salida**
Sentencia que hace que los archivos se procesen realizando operaciones en registros individuales o en el archivo como una unidad. Las sentencias de entrada-salida son ACCEPT (con la frase de identificador), CLOSE, DELETE, DISPLAY, OPEN, READ, REWRITE, SET (con la frase TO ON o TO OFF), STARTy WRITE.

*** procedimiento de entrada**
Un conjunto de sentencias, al que se proporciona control durante la ejecución de una sentencia SORT , con el fin de controlar la liberación de los registros especificados que se van a ordenar.

*** entero**
(1) Un literal numérico que no incluye ninguna posición de dígito a la derecha de la coma decimal. (2) Un elemento de datos numérico definido en DATA DIVISION que no incluye ninguna posición de dígito a la derecha de la coma decimal. (3) Una función numérica cuya definición establece que todos los dígitos a la derecha de la coma decimal son cero en el valor devuelto para cualquier posible evaluación de la función.

función de entero
Función cuya categoría es numérica y cuya definición no incluye ninguna posición de dígito a la derecha de la coma decimal.

comunicación entre lenguajes (ILC)
La capacidad de las rutinas escritas en diferentes lenguajes de programación para comunicarse. El soporte de ILC le permite crear fácilmente aplicaciones a partir de rutinas de componentes escritas en diversos idiomas.

resultado intermedio
Campo intermedio que contiene los resultados de una sucesión de operaciones aritméticas.

*** datos internos**
Los datos que se describen en un programa y excluyen todos los elementos de datos externos y conectores de archivos externos. Los elementos descritos en LINKAGE SECTION de un programa se tratan como datos internos.

*** elemento de datos interno**
Elemento de datos que se describe en un programa en una unidad de ejecución. Un elemento de datos interno puede tener un nombre global.

elemento de datos decimal interno
Elemento de datos que se describe como USAGE PACKED-DECIMAL o USAGE COMP-3y que tiene una serie de caracteres PICTURE que define el elemento como numérico (una combinación válida de símbolos 9, S, Po V). Sinónimo de *elemento de datos decimal empaquetado*.

*** conector de archivo interno**
Un conector de archivo al que sólo puede acceder un programa objeto de la unidad de ejecución.

elemento de datos de coma flotante interno
Elemento de datos que se describe como USAGE COMP-1 o USAGE COMP-2. COMP-1 define un elemento de datos de coma flotante de precisión simple. COMP-2 define un elemento de datos de coma flotante de precisión doble. No hay ninguna cláusula PICTURE asociada con un elemento de datos de coma flotante interno.

Componentes internacionales para Unicode (ICU)
Un proyecto de desarrollo de código abierto patrocinado, soportado y utilizado por IBM. Las bibliotecas ICU proporcionan servicios Unicode robustos y con todas las características en una amplia variedad de plataformas, incluyendo AIX y Linux.

*** estructura de datos intraregistro**
Colección completa de grupos y elementos de datos elementales de un registro lógico que define un subconjunto contiguo de las entradas de descripción de datos. Estas entradas de descripción de datos incluyen todas las entradas cuyo número de nivel es mayor que el número de nivel de la primera entrada de descripción de datos que describe la estructura de datos dentro del registro.

función intrínseca
Una función predefinida, como una función aritmética de uso común, llamada por una referencia de función incorporada.

*** condición de clave no válida**
Una condición, en tiempo de ejecución, se produce cuando se determina que un valor específico de la clave asociada con un archivo indexado o relativo no es válido.

*** I-O-CONTROL**
El nombre de un párrafo ENVIRONMENT DIVISION en el que se especifican los requisitos de programa de objeto para volver a ejecutar puntos, el uso compartido de las mismas áreas por varios archivos de datos y el almacenamiento de varios archivos en un único dispositivo de entrada-salida.

*** I-O-CONTROL entrada**
Una entrada en el párrafo I-O-CONTROL de ENVIRONMENT DIVISION; esta entrada contiene cláusulas que proporcionan la información necesaria para la transmisión y el manejo de datos en archivos con nombre durante la ejecución de un programa.

*** Modalidad I-O**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase I-O especificada, para dicho archivo y antes de la ejecución de una sentencia CLOSE sin la fase REEL o UNIT para dicho archivo.

*** Estado de I-O**
Entidad conceptual que contiene el valor de dos caracteres que indica el estado resultante de una operación de entrada-salida. Este valor se pone a disposición del programa mediante el uso de la cláusula FILE STATUS en la entrada de control de archivos para el archivo.

estructura de iteración
Lógica de proceso de programa en la que se repite una serie de sentencias mientras una condición es verdadera o hasta que una condición es verdadera.

J

J2EE
Consulte *Java 2 Platform, Enterprise Edition (J2EE)*.

Java 2 Platform, Enterprise Edition (J2EE)
Entorno para el desarrollo y el despliegue de aplicaciones de empresa, definido por Oracle. La plataforma J2EE consta de un conjunto de servicios, interfaces de programación de aplicaciones (API) y protocolos que proporcionan la funcionalidad para desarrollar aplicaciones basadas en web de varios niveles. (Oracle)

Java Native Interface (JNI)
Interfaz de programación que permite que el código Java que se ejecuta dentro de una máquina virtual Java (JVM) interopere con aplicaciones y bibliotecas escritas en otros lenguajes de programación.

Máquina virtualJava (JVM)
Implementación de software de una unidad de proceso central que ejecuta programas Java compilados.

JSON
JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos.

JVM
Consulte *Máquina virtualJava (JVM)*.

K

K
Al hacer referencia a la capacidad de almacenamiento, dos a la décima potencia; 1024 en notación decimal.

*** clave**
Elemento de datos que identifica la ubicación de un registro o un conjunto de elementos de datos que sirven para identificar el orden de los datos.

*** clave de referencia**
La clave, ya sea primaria o alternativa, que se está utilizando actualmente para acceder a los registros dentro de un archivo indexado.

*** palabra clave**
Palabra sensible al contexto o palabra reservada cuya presencia es necesaria cuando el formato en el que aparece la palabra se utiliza en una unidad de origen.

kilobyte (KB)
Un kilobyte equivale a 1024 bytes.

L

*** nombre-idioma**
Nombre de sistema que especifica un lenguaje de programación determinado.

último estado utilizado
Un estado en el que se encuentra un programa si sus valores internos siguen siendo los mismos que cuando se salió del programa (los valores no se restablecen a sus valores iniciales).

*** letra**
Un carácter que pertenece a uno de los dos conjuntos siguientes:

1. Letras mayúsculas: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
2. Letras minúsculas: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

*** indicador de nivel**
Dos caracteres alfabéticos que identifican un tipo específico de archivo o una posición en una jerarquía. Los indicadores de nivel en `DATA DIVISION` son: `CD`, `FD`y `SD`.

*** número-nivel**
Palabra definida por el usuario (expresada como un número de dos dígitos) que indica la posición jerárquica de un elemento de datos o las propiedades especiales de una entrada de descripción de datos. Los números de nivel en el rango de 1 a 49 indican la posición de un elemento de datos en la estructura jerárquica de un registro lógico. Los números de nivel en el rango de 1 a 9 se pueden escribir como un solo dígito o como un cero seguido de un dígito significativo. Los números de nivel 66, 77 y 88 identifican propiedades especiales de una entrada de descripción de datos.

*** nombre-biblioteca**
Palabra definida por el usuario que nombra una biblioteca COBOL que el compilador va a utilizar para compilar un programa fuente determinado.

*** texto de biblioteca**
Una secuencia de palabras de texto, líneas de comentario, el espacio de separador o el delimitador de pseudotexto de separador en una biblioteca COBOL.

Fecha de Lilian
El número de días desde el inicio del calendario gregoriano. El primer día es el viernes 15 de octubre de 1582. El formato de fecha Lilian es nombrado en honor a Luigi Lilio, el creador del calendario gregoriano.

*** contador de linaje**
Un registro especial cuyo valor apunta a la posición actual dentro del cuerpo de la página.

link
(1) La combinación de la conexión de enlace (el medio de transmisión) y dos estaciones de enlace, una en cada extremo de la conexión de enlace. Un enlace se puede compartir entre varios enlaces en una configuración de multipunto o red en anillo. (2) Interconectar elementos de datos o partes de uno o varios programas de sistema; por ejemplo, enlazar programas de objeto mediante un editor de enlaces para generar una biblioteca compartida.

LINKAGE SECTION
La sección del DATA DIVISION del programa llamado que describe los elementos de datos disponibles en el programa de llamada. Tanto el programa de llamada como el programa llamado pueden hacer referencia a estos elementos de datos.

literal
Serie de caracteres cuyo valor se especifica mediante el conjunto ordenado de caracteres que comprende la serie o mediante el uso de una constante figurativa.

little-endian
El formato predeterminado que utilizan los procesadores Intel para almacenar datos binarios y caracteres UTF-16 . En este formato, el byte más significativo de un elemento de datos binarios está en la dirección más alta y el byte más significativo de un carácter UTF-16 está en la dirección más alta. Compárese con *big-endian*.

Entorno local
Conjunto de atributos para un entorno de ejecución de programa que indica consideraciones culturalmente sensibles, como la página de códigos de caracteres, la secuencia de clasificación, el formato de fecha y hora, la representación de valores monetarios, la representación de valores numéricos o el idioma.

*** LOCAL-STORAGE SECTION**
La sección del DATA DIVISION que define el almacenamiento que se asigna y libera por invocación, en función del valor asignado en las cláusulas VALUE .

*** operador lógico**
Una de las palabras reservadas AND, OR o NOT. En la formación de una condición, ya sea AND, o OR, o ambos se pueden utilizar como conectivos lógicos. NO se puede utilizar para la negación lógica.

*** registro lógico**
El elemento de datos más inclusivo. El número de nivel de un registro es 01. Un registro puede ser un elemento elemental o un grupo de elementos. Sinónimo de *registro*.

*** fin de orden inferior**
El carácter situado más a la derecha de una serie de caracteres.

Sistema de archivos LSQ
El sistema de archivos LSQ sólo admite archivos LINE SEQUENTIAL.

m

Programa principal
En una jerarquía de programas y subrutinas, el primer programa que recibe el control cuando los programas se ejecutan dentro de un proceso.

makefile
Archivo de texto que contiene una lista de los archivos de la aplicación. El programa de utilidad make utiliza este archivo para actualizar los archivos de destino con los últimos cambios.

*** almacenamiento masivo**
Medio de almacenamiento en el que los datos se pueden organizar y mantener de forma secuencial y no secuencial.

*** dispositivo de almacenamiento masivo**
Dispositivo que tiene una gran capacidad de almacenamiento, como un disco magnético.

*** archivo de almacenamiento masivo**
Colección de registros que se almacena en un medio de almacenamiento masivo.

MBCS
Véase *juego de caracteres multibyte (MBCS)*.

*** megabyte (MB)**
Un megabyte equivale a 1.048.576 bytes.

*** archivo de fusión**
Colección de registros que una sentencia MERGE debe fusionar. El archivo de fusión se crea y sólo puede ser utilizado por la función de fusión.

*** nombre-mnemotécnico**
Palabra definida por el usuario que está asociada en ENVIRONMENT DIVISION con un nombre de implementación especificado.

archivo de definición de módulo
Archivo que describe los segmentos de código dentro de un módulo de carga.

carácter multibyte
Cualquier carácter que esté representado en 2 o más bytes en un juego de caracteres de varios bytes. Por ejemplo, un carácter DBCS o cualquier carácter UTF-8 que esté representado en dos o más bytes. Los caracteres UTF-16 no son caracteres multibyte porque UTF-16 no es un juego de caracteres multibyte.

juego de caracteres multibyte (MBCS)
Juego de caracteres codificado que se compone de caracteres representados en un número variable de bytes. Los ejemplos son: EUC (Extended Unix Code), UTF-8y juegos de caracteres compuestos de una mezcla de caracteres EBCDIC o ASCII de un solo byte y de doble byte.

multitarea
Modalidad de operación que proporciona la ejecución simultánea o intercalada de dos o más tareas.

multihebra
Operación simultánea de más de una vía de acceso de ejecución dentro de un sistema. Sinónimo de *multiproceso*.

N

nombre
Palabra (compuesta de no más de 30 caracteres) que define un operando COBOL.

espacio de nombres
Véase *espacio de nombres XML*.

carácter nacional
(1) Un carácter UTF-16 en un elemento de datos USAGE NATIONAL o literal nacional. (2) Cualquier carácter representado en UTF-16.

datos de caracteres nacionales
Referencia general a los datos representados en UTF-16.

posición de carácter nacional
Véase *posición de carácter*.

datos nacionales
Véase *datos de caracteres nacionales*.

elemento de datos nacional
Un elemento de datos de categoría nacional, editado a nivel nacional o editado a nivel numérico de USAGE NATIONAL.

elemento de datos decimales nacionales
Elemento de datos decimal externo que se describe implícita o explícitamente como USAGE NATIONAL y que contiene una combinación válida de PICTURE símbolos 9, S, Py V.

elemento de datos editado a nivel nacional
Elemento de datos que se describe mediante una serie de caracteres PICTURE que contiene al menos una instancia del símbolo N y al menos uno de los símbolos de inserción simples B, 0o /. Un elemento de datos editado a nivel nacional tiene USAGE NATIONAL.

elemento de datos de coma flotante nacional
Elemento de datos de coma flotante externo que se describe implícita o explícitamente como USAGE NATIONAL y que tiene una serie de caracteres PICTURE que describe un elemento de datos de coma flotante.

elemento de grupo nacional
Elemento de grupo que se describe explícita o implícitamente con una cláusula GROUP-USAGE NATIONAL . Un elemento de grupo nacional se procesa como si se hubiera definido como un elemento de datos elemental de categoría nacional para operaciones como INSPECT, STRINGy UNSTRING. Este proceso garantiza el relleno y truncamiento correctos de los caracteres nacionales, en contraste con la definición de elementos de datos de USAGE NATIONAL dentro de un elemento de grupo alfanumérico. Para las operaciones que requieren el proceso de los elementos elementales dentro de un grupo, como MOVE CORRESPONDING, ADD CORRESPONDINGy INITIALIZE, un grupo nacional se procesa utilizando la semántica de grupo.

elemento de datos alfanuméricos nativos
(De documentos XML) Elemento de datos alfanuméricos de categoría que se describe con la frase NATIVE o que se ha compilado con la opción CHAR(NATIVE) en vigor. La codificación del elemento de datos es la página de códigos ASCII o UTF-8 del entorno local de ejecución en vigor.

*** juego de caracteres nativo**
El juego de caracteres definido por el implementador asociado con el sistema especificado en el párrafo OBJECT-COMPUTER .

*** secuencia de clasificación nativa**
El orden de clasificación definido por el implementador asociado con el sistema especificado en el párrafo OBJECT-COMPUTER .

*** condición combinada negada**
El operador lógico NOT seguido inmediatamente por una condición combinada entre paréntesis. Véase también *condición* y *condición combinada*.

*** condición simple negada**
El operador lógico NOT seguido inmediatamente por una condición simple. Véase también *condición* y *condición simple*.

programa anidado
Programa que está directamente contenido en otro programa.

*** siguiente frase ejecutable**
Se completa la siguiente frase a la que se transferirá el control después de la ejecución de la sentencia actual.

*** siguiente sentencia ejecutable**
La siguiente sentencia a la que se transferirá el control después de que se haya completado la ejecución de la sentencia actual.

*** registro siguiente**
El registro que sigue lógicamente al registro actual de un archivo.

*** elementos no contiguos**
Elementos de datos elementales en WORKING-STORAGE SECTION y LINKAGE SECTION que no tienen ninguna relación jerárquica con otros elementos de datos.

no-fecha
Cualquiera de los elementos siguientes:

- Un elemento de datos cuya entrada de descripción de fecha no incluye la cláusula DATE FORMAT
- Un literal
- Un campo de fecha que se ha convertido utilizando la función UNDATE
- Un campo de fecha de modificación de referencia
- El resultado de determinadas operaciones aritméticas que pueden incluir operandos de campo de fecha; por ejemplo, la diferencia entre dos campos de fecha compatibles

NULL
Constante figurativa que se utiliza para asignar, a elementos de datos de puntero, el valor de una dirección que no es válida. NULLS se puede utilizar siempre que se pueda utilizar NULL .

*** carácter numérico**
Carácter que pertenece al siguiente conjunto de dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

elemento de datos numéricos
(1) Un elemento de datos cuya descripción restringe su contenido a un valor representado por caracteres elegidos entre los dígitos del 0 al 9. Si está firmado, el elemento también puede contener un signo +,-u otra representación de un signo operativo. (2) Un elemento de datos de categoría numérica, de coma flotante interna o de coma flotante externa. Un elemento de datos numérico puede tener USAGE DISPLAY, NATIONAL, PACKED-DECIMAL, BINARY, COMP, COMP-1, COMP-2, COMP-3, COMP-4o COMP-5.

elemento de datos editado numérica-editado
Elemento de datos que contiene datos numéricos en un formato adecuado para su uso en la salida impresa. El elemento de datos puede constar de dígitos decimales externos de 0 a 9, el separador decimal, comas, el signo de moneda, caracteres de control de signo y otros caracteres de edición. Un elemento editado numérica-se puede representar en USAGE DISPLAY o USAGE NATIONAL.

*** función numérica**
Una función cuya clase y categoría son numéricas pero que para alguna posible evaluación no satisface los requisitos de las funciones enteras.

*** literal numérico**
Literal compuesto de uno o más caracteres numéricos que pueden contener una coma decimal o un signo algebraico, o ambos. La coma decimal no debe ser el carácter situado más a la derecha. El signo algebraico, si está presente, debe ser el carácter más a la izquierda.

O

código de objeto
Salida de un compilador o ensamblador que es en sí mismo código de máquina ejecutable o es adecuado para el proceso para producir código de máquina ejecutable.

*** OBJECT-COMPUTER**
El nombre de un párrafo ENVIRONMENT DIVISION en el que se describe el entorno del sistema, donde se ejecuta el programa objeto.

*** entrada de sistema de objetos**
Una entrada en el párrafo OBJECT-COMPUTER de ENVIRONMENT DIVISION; esta entrada contiene cláusulas que describen el entorno del sistema en el que se va a ejecutar el programa objeto.

*** objeto de entrada**
Conjunto de operandos y palabras reservadas, dentro de una entrada DATA DIVISION de un programa COBOL, que sigue inmediatamente al asunto de la entrada.

programa objeto
Conjunto o grupo de instrucciones de lenguaje de máquina ejecutables y otro material diseñado para interactuar con los datos para proporcionar soluciones de problemas. En este contexto, un programa objeto es generalmente el resultado de lenguaje de máquina de la operación de un compilador COBOL en una definición de programa fuente . Cuando no hay peligro de ambigüedad, se puede utilizar la palabra *programa* en lugar de *programa objeto*.

*** hora de objeto**
La hora a la que se ejecuta un programa objeto. Sinónimo de *tiempo de ejecución*.

*** elemento obsoleto**
Un elemento de lenguaje COBOL en 85 COBOL Estándar que se ha suprimido de Estándar COBOL 2002.

ODBC
Véase *Open Database Connectivity (ODBC)*.

Objeto ODO
En el ejemplo siguiente, X es el objeto de la cláusula OCCURS DEPENDING ON (objeto ODO).

```
WORKING-STORAGE SECTION.
01  TABLE-1.
    05  X                   PIC S9.
    05  Y OCCURS 3 TIMES
          DEPENDING ON X    PIC X.
```

El valor del objeto ODO determina cuántos de los sujetos ODO aparecen en la tabla.

Asunto de ODO
En el ejemplo anterior, Y es el sujeto de la cláusula OCCURS DEPENDING ON (sujeto ODO). El número de sujetos ODO de Y que aparecen en la tabla depende del valor de X.

Conectividad de base de datos abierta (ODBC)
Especificación para una interfaz de programación de aplicaciones (API) que proporciona acceso a datos en una variedad de bases de datos y sistemas de archivos.

*** modalidad abierta**
El estado de un archivo después de la ejecución de una sentencia OPEN para dicho archivo y antes de la ejecución de una sentencia CLOSE sin la frase REEL o UNIT para dicho archivo. La modalidad de apertura concreta se especifica en la sentencia OPEN como INPUT, OUTPUT, I-Oo EXTEND.

*** operando**
(1) La definición general del operando es "el componente sobre el que se opera." (2) A los efectos del presente documento, cualquier palabra (o palabras) en minúscula que aparezca en una declaración o formato de entrada puede considerarse un operando y, como tal, es una referencia implícita a los datos indicados por el operando.

Operación
Servicio que se puede solicitar de un objeto.

*** signo operativo**
Signo algebraico asociado con un elemento de datos numérico o un literal numérico, para indicar si su valor es positivo o negativo.

archivo opcional
Un archivo que se declara como no necesariamente disponible cada vez que se ejecuta el programa objeto.

*** palabra opcional**
Palabra reservada que se incluye en un formato específico sólo para mejorar la legibilidad del idioma. Su presencia es opcional para el usuario cuando el formato en el que aparece la palabra se utiliza en una unidad de origen.

*** archivo de salida**
Archivo que se abre en modalidad de salida o en modalidad de ampliación.

*** modalidad de salida**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase OUTPUT o EXTEND especificada, para dicho archivo y antes de la ejecución de una sentencia CLOSE sin la frase REEL o UNIT para dicho archivo.

*** procedimiento de salida**
Conjunto de sentencias a las que se otorga control durante la ejecución de una sentencia SORT después de que se haya completado la función de clasificación, o durante la ejecución de una sentencia MERGE después de que la función de fusión alcance un punto en el que pueda seleccionar el siguiente registro en orden fusionado cuando se le solicite.

condición de desbordamiento
Condición que se produce cuando una parte del resultado de una operación excede la capacidad de la unidad de almacenamiento prevista.

P

elemento de datos decimal empaquetado
Véase *elemento de datos decimal interno*.

carácter de relleno
Carácter alfanumérico o nacional que se utiliza para rellenar las posiciones de caracteres no utilizadas en un registro físico.

página
División vertical de los datos de salida que representa una separación física de los datos. La separación se basa en requisitos lógicos internos o características externas del medio de salida o ambos.

*** cuerpo de página**
Parte de la página lógica en la que se pueden escribir o espaciar líneas o ambas.

*** párrafo**
En PROCEDURE DIVISION, un nombre de párrafo seguido de un punto de separación y de cero, una o más frases. En IDENTIFICATION DIVISION y ENVIRONMENT DIVISION, una cabecera de párrafo seguida de cero, una o más entradas.

*** cabecera de párrafo**
Palabra reservada, seguida del punto separador, que indica el principio de un párrafo en IDENTIFICATION DIVISION y ENVIRONMENT DIVISION. Las cabeceras de párrafo permitidas en IDENTIFICATION DIVISION son:

```
PROGRAM-ID. (Program IDENTIFICATION
   DIVISION)
AUTHOR.
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
```

Las cabeceras de párrafo permitidas en ENVIRONMENT DIVISION son:

```
SOURCE-COMPUTER.
OBJECT-COMPUTER.
SPECIAL-NAMES.
REPOSITORY. (Program
   CONFIGURATION SECTION)
FILE-CONTROL.
I-O-CONTROL.
```

*** nombre-párrafo**
Palabra definida por el usuario que identifica e inicia un párrafo en PROCEDURE DIVISION.

Parámetro
Datos pasados entre un programa de llamada y un programa llamado.

*** frase**
Conjunto ordenado de una o más series de caracteres COBOL consecutivas que forman una parte de una sentencia de procedimiento COBOL o de una cláusula COBOL.

*** registro físico**
Véase *bloque*.

elemento de datos de puntero
Elemento de datos en el que se pueden almacenar los valores de dirección. Los elementos de datos se definen explícitamente como punteros con la cláusula USAGE IS POINTER . Los registros especiales de ADDRESS OF se definen implícitamente como elementos de datos de puntero. Los elementos de datos de puntero pueden compararse por igualdad o moverse a otros elementos de datos de puntero.

port
(1) Modificar un programa informático para que pueda ejecutarse en una plataforma diferente. (2) En el conjunto de protocolos de Internet, un conector lógico específico entre el Protocolo de Control de Transmisión (TCP) o el Protocolo de Datagramas de Usuario (UDP) y un protocolo o aplicación de nivel superior. Un puerto se identifica mediante un número de puerto.

Portabilidad
La capacidad de transferir un programa de aplicación de una plataforma de aplicación a otra con relativamente pocos cambios en el programa fuente.

*** clave de registro principal**
Clave cuyo contenido identifica de forma exclusiva un registro dentro de un archivo indexado.

*** número-prioridad**
Palabra definida por el usuario que clasifica secciones en PROCEDURE DIVISION para fines de segmentación. Los números de segmento sólo pueden contener los caracteres del 0 al 9. Un número de segmento puede expresarse como uno o dos dígitos.

*** procedimiento**
Un párrafo o grupo de párrafos lógicamente sucesivos, o una sección o grupo de secciones lógicamente sucesivos, dentro de PROCEDURE DIVISION.

*** sentencia de ramificación de procedimiento**
Sentencia que provoca la transferencia explícita del control a una sentencia distinta de la siguiente sentencia ejecutable en la secuencia en la que se escriben las sentencias en el código fuente. Las sentencias de ramificación de procedimiento son: ALTER, CALL, EXIT, EXIT PROGRAM, GO TO, MERGE (con la frase OUTPUT PROCEDURE), PERFORM y SORT (con la frase INPUT PROCEDURE o OUTPUT PROCEDURE), XML PARSE.

PROCEDURE DIVISION
La división COBOL que contiene instrucciones para resolver un problema.

integración de procedimientos
Una de las funciones del optimizador COBOL es simplificar las llamadas a procedimientos realizados o programas contenidos.

La integración de procedimientos PERFORM es el proceso por el que una sentencia PERFORM se sustituye por sus procedimientos realizados. La integración de procedimiento de programa contenido es el proceso en el que se sustituye una llamada a un programa contenido por el código de programa.

*** nombre-procedimiento**
Palabra definida por el usuario que se utiliza para nombrar un párrafo o sección en PROCEDURE DIVISION. Consta de un nombre de párrafo (que puede calificarse) o un nombre de sección.

Puntero de procedimiento
Elemento de datos en el que se puede almacenar un puntero a un punto de entrada. Un elemento de datos definido con la cláusula USAGE IS PROCEDURE-POINTER contiene la dirección de un punto de entrada de procedimiento.

elemento de datos de puntero de procedimiento
Elemento de datos en el que se puede almacenar un puntero a un punto de entrada. Un elemento de datos definido con la cláusula USAGE IS PROCEDURE-POINTER contiene la dirección de un punto de entrada de procedimiento. Normalmente se utiliza para comunicarse con programas COBOL .

proceso
El curso de los sucesos que se producen durante la ejecución de todo o parte de un programa. Varios procesos se pueden ejecutar simultáneamente y los programas que se ejecutan dentro de un proceso pueden compartir recursos.

programa
(1) Una secuencia de instrucciones adecuadas para ser procesadas por un ordenador. El proceso puede incluir el uso de un compilador para preparar el programa para su ejecución, así como un entorno de ejecución para ejecutarlo. (2) Conjunto lógico de uno o más módulos interrelacionados. Se pueden ejecutar varias copias del mismo programa en distintos procesos.

*** entrada de identificación de programa**
En el párrafo PROGRAM-ID de IDENTIFICATION DIVISION, entrada que contiene cláusulas que especifican el nombre de programa y asignan atributos de programa seleccionados al programa.

nombre-programa
En el IDENTIFICATION DIVISION y el marcador de programa final, una palabra definida por el usuario o un literal alfanumérico que identifica un programa fuente COBOL.

Proyecto
Conjunto completo de datos y acciones necesarios para crear un destino, como por ejemplo una biblioteca de enlaces dinámicos (DLL) u otro ejecutable (EXE).

*** pseudo-texto**
Una secuencia de palabras de texto, líneas de comentario, o el espacio de separador en un programa fuente o biblioteca COBOL limitada por, pero sin incluir, delimitadores de pseudotexto.

*** delimitador de pseudotexto**
Dos caracteres de signo igual contiguos (==) utilizados para delimitar el pseudo-texto.

*** carácter de puntuación**
Un carácter que pertenece al conjunto siguiente:

Carácter	Significado
,	Coma
;	Punto y coma
:	Dos puntos
.	Periodo (parada completa)
"	comilla
(	Paréntesis izquierdo
)	Paréntesis derecho
	Espacio
=	Signo igual

Q

QSAM (Método de acceso secuencial en cola)
Versión ampliada del método de acceso secuencial básico (BSAM). Cuando se utiliza este método, se forma una cola de bloques de datos de entrada que están a la espera de proceso o de bloques de datos de salida que se han procesado y que están a la espera de transferencia al almacenamiento auxiliar o a un dispositivo de salida.

Sistema de archivos QSAM
El sistema de archivos QSAM (método de acceso secuencial en cola) da soporte a registros fijos, variables y expandidos, y le permite acceder directamente a un archivo QSAM que ha transferido (utilizando z/OS FTP) de z/OS a AIX o Linux con las opciones `binary and quote site rdw`. Un archivo QSAM da soporte a todos los tipos de datos COBOL del registro.

*** nombre-datos calificado**
Un identificador que se compone de un nombre de datos seguido de uno o más conjuntos de cualquiera de las conexiones `OF` y `IN` seguido de un calificador de nombre de datos.

*** calificador**
(1) Un nombre de datos o un nombre asociado a un indicador de nivel que se utiliza en una referencia junto con otro nombre de datos (que es el nombre de un elemento que está subordinado al calificador) o junto con un nombre de condición. (2) Un nombre de sección que se utiliza en una referencia junto con un nombre de párrafo especificado en dicha sección. (3) Un nombre de biblioteca que se utiliza en una referencia junto con un nombre de texto asociado con esa biblioteca.

R

*** acceso aleatorio**
Modalidad de acceso en la que el valor especificado por el programa de un elemento de datos clave identifica el registro lógico obtenido, suprimido o colocado en un archivo relativo o indexado.

*** registro**
Véase *registro lógico*.

*** área de registro**
Área de almacenamiento asignada con el fin de procesar el registro descrito en una entrada de descripción de registro en el FILE SECTION del DATA DIVISION. En FILE SECTION, el número actual de posiciones de caracteres en el área de registro viene determinado por la cláusula RECORD explícita o implícita.

*** descripción de registro**
Véase *entrada de descripción de registro*.

*** entrada de descripción de registro**
Conjunto total de entradas de descripción de datos asociadas a un registro determinado. Sinónimo de *descripción de registro*.

Clave de registro
Clave cuyo contenido identifica un registro dentro de un archivo indexado.

nombre-clave-registro
Palabra definida por el usuario que nombra una clave asociada a un archivo indexado.

*** nombre-registro**
Palabra definida por el usuario que nombra un registro descrito en una entrada de descripción de registro en el DATA DIVISION de un programa COBOL.

*** número de registro**
El número ordinal de un registro en el archivo cuya organización es secuencial.

modalidad de grabación
El formato de los registros lógicos en un archivo. La modalidad de registro puede ser F (longitud fija), V (longitud variable), S (distribuida) o U (no definida).

recursividad
Programa que se llama a sí mismo o que es llamado directa o indirectamente por uno de sus programas llamados.

con capacidad recursiva
Un programa tiene capacidad recursiva (se puede llamar recursivamente) si el atributo RECURSIVE está en la sentencia PROGRAM-ID .

carrete
Parte discreta de un medio de almacenamiento, cuyas dimensiones son determinadas por cada implementador que contiene parte de un archivo, todo un archivo o cualquier número de archivos. Sinónimo de *unidad* y *volumen*.

reentrante
El atributo de un programa o rutina que permite que más de un usuario comparta una sola copia de un objeto de programa módulo de carga.

*** formato de referencia**
Formato que proporciona un método estándar para describir programas fuente COBOL.

Modificación de referencia
Método para definir un nuevo elemento de datos alfanumérico de categoría, DBCS de categoría o nacional de categoría especificando el carácter más a la izquierda y la longitud relativa a la posición de carácter más a la izquierda de un elemento de datos USAGE DISPLAY, DISPLAY-1o NATIONAL .

*** modificador-referencia**
Combinación sintácticamente correcta de series de caracteres y separadores que define un elemento de datos exclusivo. Incluye un separador de paréntesis izquierdo delimitador, la posición de carácter más a la izquierda, un separador de dos puntos, opcionalmente una longitud y un separador de paréntesis derecho delimitador.

*** relación**
Véase *operador relacional* o *condición de relación*.

*** carácter de relación**
Un carácter que pertenece al conjunto siguiente:

Carácter	Significado
>	Mayor que
<	Menor que
=	Igual a

*** condición de relación**
La proposición (para la que se puede determinar un valor de verdad) de que el valor de una expresión aritmética, elemento de datos, literal alfanumérico o nombre de índice tiene una relación específica con el valor de otra expresión aritmética, elemento de datos, literal alfanumérico o nombre de índice. Véase también *operador relacional*.

*** operador relacional**
Una palabra reservada, un carácter de relación, un grupo de palabras reservadas consecutivas o un grupo de palabras reservadas consecutivas y caracteres de relación utilizados en la construcción de una condición de relación. Los operadores permitidos y sus significados son:

Carácter	Significado
`IS GREATER THAN`	Mayor que
`IS >`	Mayor que
`IS NOT GREATER THAN`	No superior a
`IS NOT >`	No superior a
`IS LESS THAN`	Menor que
`IS <`	Menor que
`IS NOT LESS THAN`	No inferior a
`IS NOT <`	No inferior a
`IS EQUAL TO`	Igual a
`IS =`	Igual a
`IS NOT EQUAL TO`	No igual a
`IS NOT =`	No igual a
`IS GREATER THAN OR EQUAL TO`	Mayor que o igual a
`IS >=`	Mayor que o igual a
`IS LESS THAN OR EQUAL TO`	Menor que o igual a
`IS <=`	Menor que o igual a

*** archivo relativo**
Un archivo con una organización relativa.

*** clave relativa**
Clave cuyo contenido identifica un registro lógico en un archivo relativo.

*** organización relativa**
Estructura de archivo lógico permanente en la que cada registro se identifica de forma exclusiva mediante un valor entero mayor que cero, que especifica la posición ordinal lógica del registro en el archivo.

*** número de registro relativo**
El número ordinal de un registro en un archivo cuya organización es relativa. Este número se trata como un literal numérico que es un entero.

*** palabra reservada**
Palabra COBOL que se especifica en la lista de palabras que se pueden utilizar en un programa fuente COBOL, pero que no debe aparecer en el programa como palabra o nombre de sistema definido por el usuario.

*** recurso**
Recurso o servicio, controlado por el sistema operativo, que puede utilizar un programa en ejecución.

*** identificador resultante**
Elemento de datos definido por el usuario que debe contener el resultado de una operación aritmética.

rutina
Conjunto de sentencias de un programa COBOL que hace que el sistema realice una operación o serie de operaciones relacionadas.

*** nombre-rutina**
Palabra definida por el usuario que identifica un procedimiento escrito en un lenguaje distinto de COBOL.

Sistema de archivos RSD
El sistema de archivos delimitado secuencial de registros es un sistema de archivos de estación de trabajo que soporta archivos secuenciales. Un archivo RSD da soporte a todos los tipos de datos COBOL en registros de longitud fija o variable, puede ser editado por la mayoría de editores de archivos y puede ser leído por programas escritos en otros lenguajes. Este sistema sólo soporta archivos secuenciales.

*** tiempo de ejecución**
La hora a la que se ejecuta un programa objeto. Sinónimo de *hora de objeto*.

entorno de ejecución
Entorno en el que se ejecuta un programa COBOL.

*** unidad de ejecución**
Programa de objeto autónomo, o varios programas de objeto, que interactúan mediante sentencias COBOL `CALL` y funcionan en tiempo de ejecución como una entidad.

S

SBCS
Véase *juego de caracteres de un solo byte (SBCS)*.

terminador de ámbito
Palabra reservada de COBOL que marca el final de determinadas sentencias de `PROCEDURE DIVISION` statements.It puede ser explícita (`END-ADD`, por ejemplo) o implícita (punto separador).

*** sección**
Un conjunto de cero, uno o más párrafos o entidades, denominado un cuerpo de sección, el primero de los cuales va precedido de una cabecera de sección. Cada sección consta de la cabecera de sección y el cuerpo de sección relacionado.

*** cabecera de sección**
Combinación de palabras seguida de un punto de separación que indica el principio de una sección en cualquiera de estas divisiones: `ENVIRONMENT`, `DATA`o `PROCEDURE`. En `ENVIRONMENT DIVISION` y `DATA DIVISION`, una cabecera de sección se compone de palabras reservadas seguidas de un punto separador. Las cabeceras de sección permitidas en `ENVIRONMENT DIVISION` son:

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

Las cabeceras de sección permitidas en `DATA DIVISION` son:

```
FILE SECTION.
WORKING-STORAGE SECTION.
LOCAL-STORAGE SECTION.
LINKAGE SECTION.
```

En PROCEDURE DIVISION, una cabecera de sección se compone de un nombre de sección, seguido de la palabra reservada SECTION, seguida de un punto de separación.

*** nombre-sección**
Una palabra definida por el usuario que nombra una sección en PROCEDURE DIVISION.

segmentación

Hace referencia al módulo de segmentación estándar COBOL 85. Esta característica ha quedado obsoleta y se ha eliminado de las versiones posteriores de COBOL Standard y no está soportada en COBOL para Linux.

estructura de selección
Lógica de proceso de programa en la que se ejecuta una u otra serie de sentencias, en función de si una condición es verdadera o falsa.

*** frase**
Secuencia de una o más sentencias, la última de las cuales termina con un punto de separación.

*** programa compilado por separado**
Programa que, junto con sus programas contenidos, se compila por separado de todos los demás programas.

*** separador**
Un carácter o dos caracteres contiguos utilizados para delimitar series de caracteres.

*** coma de separador**
Una coma (,) seguida de un espacio utilizado para delimitar series de caracteres.

*** punto separador**
Un punto (.) seguido de un espacio utilizado para delimitar series de caracteres.

*** punto y coma de separador**
Un punto y coma (;) seguido de un espacio utilizado para delimitar las series de caracteres.

estructura de secuencia
Lógica de proceso de programa en la que se ejecuta una serie de sentencias en orden secuencial.

*** acceso secuencial**
Modalidad de acceso en la que los registros lógicos se obtienen de un archivo o se colocan en él en una secuencia de registro lógico de predecesor a sucesor consecutiva determinada por el orden de los registros del archivo.

*** archivo secuencial**
Un archivo con organización secuencial.

*** organización secuencial**
Estructura de archivo lógico permanente en la que un registro se identifica mediante una relación predecesor-sucesor establecida cuando el registro se coloca en el archivo.

búsqueda en serie
Búsqueda en la que se examinan consecutivamente los miembros de un conjunto, empezando por el primer miembro y terminando por el último.

Sistema de archivos SFS
El sistema de archivos CICS Structured File Server es un sistema de archivos orientado a registros que da soporte al acceso secuencial, relativo y a archivos indexados por clave.

biblioteca compartida
Biblioteca creada por el enlazador que contiene al menos una subrutina que pueden utilizar varios procesos. Los programas y subrutinas están enlazados como de costumbre, pero el código común a diferentes subrutinas se combina en un archivo de biblioteca que se puede cargar en tiempo de ejecución y compartir con muchos programas. Una clave para identificar el archivo de biblioteca compartida está en la cabecera de cada subrutina.

*** condición de signo**
La proposición (para la que se puede determinar un valor de verdad) de que el valor algebraico de un elemento de datos o una expresión aritmética es menor que, mayor que o igual a cero.

firma
El nombre de una operación y sus parámetros.

*** condición simple**
Cualquier condición individual elegida de este conjunto:

- Condición de relación
- Condición de clase
- Condición de nombre de condición
- Condición de estado de conmutador
- Condición de firma

Véase también *condición* y *condición simple negada*.

juego de caracteres de un solo byte (SBCS)
Conjunto de caracteres en el que cada carácter se representa mediante un solo byte. Consulte también *ASCII* y *EBCDIC (Extended Binary-Coded Decimal Interchange Code)*.

bytes de holgura (dentro de registros)
Bytes insertados por el compilador entre elementos de datos para garantizar la alineación correcta de algunos elementos de datos elementales. Los bytes de holgura no contienen datos significativos. La cláusula SYNCHRONIZED indica al compilador que inserte bytes de holgura cuando sean necesarios para una alineación adecuada.

bytes de holgura (entre registros)
Bytes insertados por el programador entre registros lógicos bloqueados de un archivo, para garantizar la alineación correcta de algunos elementos de datos elementales. En algunos casos, los bytes de holgura entre registros mejoran el rendimiento de los registros procesados en un almacenamiento intermedio.

*** ordenar archivo**
Colección de registros que se ordenarán mediante una sentencia SORT . El archivo de ordenación se crea y sólo puede ser utilizado por la función de ordenación.

*** entrada de descripción de archivo de fusión de clasificación**
Una entrada en FILE SECTION del DATA DIVISION que se compone del indicador de nivel SD, seguido de un nombre de archivo y, a continuación, seguido de un conjunto de cláusulas de archivo según sea necesario.

*** SOURCE-COMPUTER**
El nombre de un párrafo ENVIRONMENT DIVISION en el que se describe el entorno del sistema, donde se compila el programa fuente.

*** entrada de sistema de origen**
Una entrada en el párrafo SOURCE-COMPUTER de ENVIRONMENT DIVISION; esta entrada contiene cláusulas que describen el entorno del sistema en el que se va a compilar el programa fuente.

*** elemento de origen**
Identificador designado por una cláusula SOURCE que proporciona el valor de un elemento imprimible.

programa fuente
Aunque un programa fuente puede estar representado por otras formas y símbolos, en este documento el término siempre hace referencia a un conjunto sintácticamente correcto de sentencias COBOL. Un programa fuente COBOL comienza con la sentencia IDENTIFICATION DIVISION o COPY y termina con el marcador de programa final, si se especifica, o con la ausencia de líneas de programa fuente adicionales.

unidad de origen
Unidad de código fuente COBOL que se puede compilar por separado: un programa. También se conoce como *unidad de compilación*.

Carácter especial
Un carácter que pertenece al siguiente conjunto:

Carácter	Significado
+	Signo más
-	Signo menos (guión)
*	Asterisco
/	Inclinado (barra inclinada)
=	Signo igual
$	Símbolo de moneda
,	Coma
;	Punto y coma
.	Punto (coma decimal, punto completo)
"	comilla
'	Apóstrofo
(	Paréntesis izquierdo
)	Paréntesis derecho
>	Mayor que
<	Menor que
:	Dos puntos
_	Subrayado

SPECIAL-NAMES
El nombre de un párrafo de ENVIRONMENT DIVISION en el que los nombres de entorno están relacionados con nombres mnemotécnicos especificados por el usuario.

*** entrada de nombres especiales**
Una entrada en el párrafo SPECIAL-NAMES del ENVIRONMENT DIVISION; esta entrada proporciona medios para especificar el signo de moneda; elegir el separador decimal; especificar caracteres simbólicos; relacionar nombres de implementador con nombres mnemónicos especificados por el usuario; relacionar nombres de alfabeto con conjuntos de caracteres o secuencias de clasificación; y relacionar nombres de clase con conjuntos de caracteres.

*** registros especiales**
Determinadas áreas de almacenamiento generadas por el compilador cuyo uso principal es almacenar información producida junto con el uso de una característica COBOL específica.

*** sentencia**
Combinación sintácticamente válida de palabras, literales y separadores, empezando por un verbo, escrito en un programa fuente COBOL.

Sistema de archivos STL
El sistema de archivos de lenguaje estándar es el sistema de archivos de estación de trabajo nativa para COBOL. Este sistema soporta archivos secuenciales, relativos e indexados.

programación estructurada
Una técnica para organizar y codificar un programa de computadora en el cual el programa comprende una jerarquía de segmentos, cada segmento tiene un único punto de entrada y un único punto de salida. El control se pasa hacia abajo a través de la estructura sin ramas incondicionales a niveles más altos de la jerarquía.

*** asunto de entrada**
Un operando o palabra reservada que aparece inmediatamente después del indicador de nivel o el número de nivel en una entrada de DATA DIVISION .

*** subprograma**
Véase *programa llamado*.

*** subíndice**
Número de aparición representado por un entero, un nombre de datos seguido opcionalmente por un entero con el operador + o-, o un nombre de índice seguido opcionalmente por un entero con el operador + o-, que identifica un elemento determinado de una tabla. Un subíndice puede ser la palabra ALL cuando se utiliza el identificador de subíndice como argumento de función para una función que permite un número variable de argumentos.

*** nombre-datos-subíndice**
Identificador compuesto de un nombre de datos seguido de uno o más subíndices entre paréntesis.

Carácter de sustitución
Carácter que se utiliza en una conversión de una página de códigos fuente a una página de códigos de destino para representar un carácter que no está definido en la página de códigos de destino.

par sustituto
En el formato UTF-16 de Unicode, un par de unidades de codificación que juntas representan un único carácter gráfico Unicode. La primera unidad del par se denomina *sustituto alto* y la segunda *sustituto bajo*. El valor de código de un sustituto alto está en el rango de X'D800'a X'DBFF'. El valor de código de un sustituto bajo está en el rango de X'DC00'a X'DFFF'. Los pares suplentes proporcionan más caracteres que los 65.536 caracteres que caben en el juego de caracteres codificado Unicode de 16 bits.

condición de estado de conmutador
La proposición (para la cual se puede determinar un valor de verdad) de que un conmutador UPSI, capaz de establecerse en un estado de encendido o apagado, se ha establecido en un estado específico.

*** carácter-simbólico**
Palabra definida por el usuario que especifica una constante figurativa definida por el usuario.

Sintaxis
(1) La relación entre caracteres o grupos de caracteres, independientemente de su significado o de la forma en que se interpreten y utilicen. (2) La estructura de las expresiones en un lenguaje. (3) Las normas que rigen la estructura de una lengua. (4) La relación entre símbolos. (5) Normas para la construcción de una declaración.

SYSADATA
Archivo de información de compilación adicional que se genera si la opción de compilador ADATA está en vigor.

SYSIN
El archivo o archivos de entrada del compilador primario.

syslib
El archivo o archivos de entrada del compilador secundario, que se procesan si la opción de compilador LIB está en vigor.

SYSPRINT
El archivo de listado del compilador.

*** nombre-sistema**
Palabra COBOL que se utiliza para comunicarse con el entorno operativo.

T

*** tabla**
Un conjunto de elementos de datos lógicamente consecutivos que se definen en DATA DIVISION mediante la cláusula OCCURS .

*** elemento de tabla**
Elemento de datos que pertenece al conjunto de elementos repetidos que forman una tabla.

*** nombre-texto**
Palabra definida por el usuario que identifica el texto de la biblioteca.

*** palabra de texto**
Un carácter o una secuencia de caracteres contiguos entre el margen A y el margen R en una biblioteca COBOL, programa fuente o pseudotexto que sea cualquiera de los caracteres siguientes:

- Un separador, excepto para el espacio; un delimitador de pseudotexto; y los delimitadores de apertura y cierre para literales alfanuméricos. Los caracteres de paréntesis derecho y paréntesis izquierdo, independientemente del contexto dentro de la biblioteca, programa fuente o pseudotexto, siempre se consideran palabras de texto.
- Un literal que incluye, en el caso de literales alfanuméricos, la comilla de apertura y la comilla de cierre que enlazan el literal.
- Cualquier otra secuencia de caracteres COBOL contiguos excepto las líneas de comentario y la palabra C0PY limitada por separadores que no son ni un separador ni un literal.

hebra
Secuencia de instrucciones de sistema (iniciadas por una aplicación dentro de un proceso) que está en control de un proceso.

señal
En el editor COBOL, unidad de significado de un programa. Una señal puede contener datos, una palabra clave de idioma, un identificador u otra parte de la sintaxis de lenguaje.

diseño descendente
Diseño de un programa informático que utiliza una estructura jerárquica en la que se realizan funciones relacionadas en cada nivel de la estructura.

Desarrollo descendente
Véase *programación estructurada*.

etiqueta-remolque
(1) Una etiqueta de que sigue a los registros de datos en una unidad de soporte de grabación. (2) Sinónimo de *etiqueta de fin de archivo*.

resolución de problemas
Para detectar, localizar y eliminar problemas en el uso de software de computadora.

*** valor de verdad**
La representación del resultado de la evaluación de una condición en términos de uno de dos valores: verdadero o falso.

U

*** operador unario**
Un signo más (+) o un signo menos (-) que precede a una variable o un paréntesis izquierdo en una expresión aritmética y que tiene el efecto de multiplicar la expresión por + 1 o -1, respectivamente.

Unicode
Estándar de codificación de caracteres universal que da soporte al intercambio, proceso y visualización de texto escrito en cualquiera de los idiomas del mundo moderno. Existen varios esquemas de codificación para representar Unicode, incluidos UTF-8, UTF-16y UTF-32. COBOL para Linux da soporte a Unicode utilizando UTF-16 en formato little-endian como representación para el tipo de datos nacional.

Identificador uniforme de recursos (URI)
Secuencia de caracteres que nombra de forma exclusiva un recurso; en COBOL para Linux, el identificador de un espacio de nombres. La sintaxis de URI la define el documento *Identificador universal de recursos (URI): Sintaxis genérica*.

unit
Un módulo de acceso directo, cuyas dimensiones están determinadas por IBM.

*** ejecución no satisfactoria**
El intento de ejecución de una sentencia que no da como resultado la ejecución de todas las operaciones especificadas por dicha sentencia. La ejecución no satisfactoria de una sentencia no afecta a los datos a los que hace referencia dicha sentencia, pero puede afectar a los indicadores de estado.

Conmutador UPSI
Conmutador de programa que realiza las funciones de un conmutador de hardware. Se proporcionan ocho: UPSI-0 a UPSI-7.

URI
Véase *Uniform Resource Identifier (URI)*.

*** palabra definida por el usuario**
Palabra COBOL que debe proporcionar el usuario para satisfacer el formato de una cláusula o sentencia.

V

*** variable**
Elemento de datos cuyo valor se puede cambiar mediante la ejecución del programa objeto. Una variable utilizada en una expresión aritmética debe ser un elemento elemental numérico.

elemento de longitud variable
Elemento de grupo que contiene una tabla descrita con la frase `DEPENDING` de la cláusula `OCCURS` .

*** registro de longitud variable**
Un registro asociado a un archivo cuya descripción de archivo o entrada de descripción de fusión de clasificación permite que los registros contengan un número variable de posiciones de caracteres.

*** elemento de datos de aparición variable**
Un elemento de datos de aparición de variable es un elemento de tabla que se repite un número variable de veces. Un elemento de este tipo debe contener una cláusula `OCCURS DEPENDING ON` en su entrada de descripción de datos o estar subordinado a dicho elemento.

*** grupo de ubicación variable**
Elemento de grupo que sigue, y no subordinado, a una tabla de longitud variable en el mismo registro. El elemento de grupo puede ser un grupo alfanumérico o un grupo nacional.

*** elemento de ubicación variable**
Un elemento de datos a continuación, y no subordinado, de una tabla de longitud variable en el mismo registro.

*** verbo**
Palabra que expresa una acción que debe realizar un compilador COBOL o un programa de objetos.

volumen
Un módulo de almacenamiento externo. Para los dispositivos de cinta es un carrete; para los dispositivos de acceso directo es una unidad.

Sistema de archivos VSAM
Sistema de archivos que da soporte a organizaciones COBOL secuenciales, relativas e indexadas.

VSAM
Término genérico para el *sistema de archivos STL* o el *sistema de archivos SFS*.

W

servicio web
Una aplicación modular que realiza tareas específicas y es accesible a través de protocolos abiertos como HTTP y SOAP.

espacio en blanco
Caracteres que introducen espacio en un documento. Son las siguientes:

- Espacio
- Tabulación horizontal
- Retorno de carro
- Salto de línea
- Línea siguiente

tal como se indica en el estándar Unicode.

campo de fecha de ventana
Campo de fecha que contiene un año con ventana (dos dígitos). Véase también *campo de fecha* y *año de ventana*.

año de ventana
Campo de fecha que consta sólo de un año de dos dígitos. Este año de dos dígitos se puede interpretar utilizando una ventana de siglo. Por ejemplo, 10 podría interpretarse como 2010. Véase también *ventana de siglo*. Compárese con *año expandido*.

*** palabra**
Serie de caracteres de no más de 30 caracteres que forma una palabra definida por el usuario, un nombre de sistema, una palabra reservada o un nombre de función.

*** WORKING-STORAGE SECTION**
La sección del DATA DIVISION que describe los elementos de datos de WORKING-STORAGE , compuestos de elementos no contiguos o registros de WORKING-STORAGE o de ambos.

estación de trabajo
Término genérico para sistemas, incluidos sistemas personales, terminales 3270, estaciones de trabajo inteligentes y terminales UNIX . A menudo, una estación de trabajo está conectada a un sistema principal o a una red.

envoltura
Objeto que proporciona una interfaz entre el código orientado a objetos y el código orientado a procedimientos. El uso de derivadores permite que otros sistemas reutilicen y accedan a los programas.

X

X
El símbolo de una cláusula PICTURE que puede contener cualquier carácter del juego de caracteres del sistema.

XML
Extensible Markup Language. Un metalenguaje estándar para definir lenguajes de marcación que se ha derivado de y es un subconjunto de SGML. XML omite las partes más complejas y menos utilizadas de SGML y hace que sea mucho más fácil escribir aplicaciones para manejar tipos de documentos, crear y gestionar información estructurada, y transmitir y compartir información estructurada entre diversos sistemas informáticos. El uso de XML no requiere las aplicaciones robustas y el proceso necesario para SGML. XML se desarrolla bajo los auspicios de World Wide Web Consortium (W3C).

Datos XML
Datos que se organizan en una estructura jerárquica con elementos XML. Las definiciones de datos se definen en las declaraciones de tipo de elemento XML.

declaración XML
Texto XML que especifica las características del documento XML como, por ejemplo, la versión de XML que se está utilizando y la codificación del documento.

Documento XML
Un objeto de datos que está bien formado tal como lo define la especificación XML W3C .

Espacio de nombres XML
Mecanismo, definido por las especificaciones de espacio de nombres XML W3C , que limita el ámbito de una colección de nombres de elemento y nombres de atributo. Un espacio de nombres XML elegido de forma exclusiva garantiza la identidad exclusiva de un nombre de elemento o nombre de atributo en varios documentos XML o varios contextos dentro de un documento XML.

S

expansión de campo de año
Expansión explícita de campos de fecha que contienen años de dos dígitos para contener años de cuatro dígitos en archivos y bases de datos y, a continuación, uso de estos campos en formato expandido en programas. Este es el único método para asegurar un proceso de fecha fiable para las aplicaciones que han utilizado años de dos dígitos.

Z

elemento de datos decimales con zona
Elemento de datos decimal externo que se describe implícita o explícitamente como USAGE DISPLAY y que contiene una combinación válida de PICTURE símbolos 9, S, Py V. El contenido de un elemento de datos decimal con zona se representa en caracteres del 0 al 9, opcionalmente con un signo. Si la serie PICTURE especifica un signo y se especifica la cláusula SIGN IS SEPARATE , el signo se representa como caracteres + o-. Si no se especifica SIGN IS SEPARATE , el signo es un dígito hexadecimal que se superpone a los 4 primeros bits de la posición del signo (inicial o final).

#

entrada-descripción-nivel-77
Entrada de descripción de datos que describe un elemento de datos no contiguo que tiene el número de nivel 77.

85 COBOL Estándar
El lenguaje COBOL definido por los estándares siguientes:

- *ANSI INCITS 23-1985, lenguajes de programación-COBOL*, modificado por *ANSI INCITS 23a-1989, Lenguajes de programación-COBOL-Módulo de función intrínseca para COBOL*
- *ISO 1989:1985, Lenguajes de programación-COBOL*, modificado por *ISO/IEC 1989/AMD1:1992, Lenguajes de programación-COBOL: Módulo de función intrínseca*

Estándar COBOL 2002
El lenguaje COBOL definido por el siguiente estándar:

- *INCITS/ISO/IEC 1989-2002, Tecnologías de la información-Lenguajes de programación-COBOL*

Estándar COBOL 2014
El lenguaje COBOL definido por el siguiente estándar:

- *INCITS/ISO/IEC 1989:2014, Tecnologías de la información-Lenguajes de programación, sus entornos e interfaces de software del sistema-Lenguaje de programación COBOL*

Lista de recursos

Publicaciones COBOL for Linux

Novedades en COBOL for Linux en x86 1.2, SC28-3453-00

Guía de instalación, GC28-3116-01

Consulta de lenguaje, SC28-3117-01

Guía de programación, SC28-3118-01

Guía de migración, SC28-3454-00

Apoyo

Si tiene un problema al utilizar COBOL for Linux, Visite el sitio web IBM Support , que proporciona información de soporte actualizada.

Publicaciones relacionadas

DB2 para Linux, UNIX, y Windows

Puede encontrar las publicaciones siguientes en la Documentación deIBM Documentation:

- *Consulta de mandatos*
- *Conceptos de administración de bases de datos y referencia de configuración*
- *Referencia de SQL para Db2 Versión 11.1 para Linux, UNIXy Windows*

TXSeries para Multiplatforms

- *Documentación deIBM TXSeries for Multiplatforms*

IBM CICS TX

- *IBM Documentación de CICS TX*

Representación de Unicode y caracteres

- *Unicode*
- *Componentes internacionales para Unicode: Converter Explorer*
- *Arquitectura de representación de datos de caracteres: Referencia y registro*

XML

- *XML (Extensible Markup Language)*
- *Espacios de nombres en XML 1.0*
- *Espacios de nombres en XML 1.1*
- *Especificación XML*

Índice

Caracteres Especiales

Números

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Número de Programa: 5737-L11

SC28-3117-01