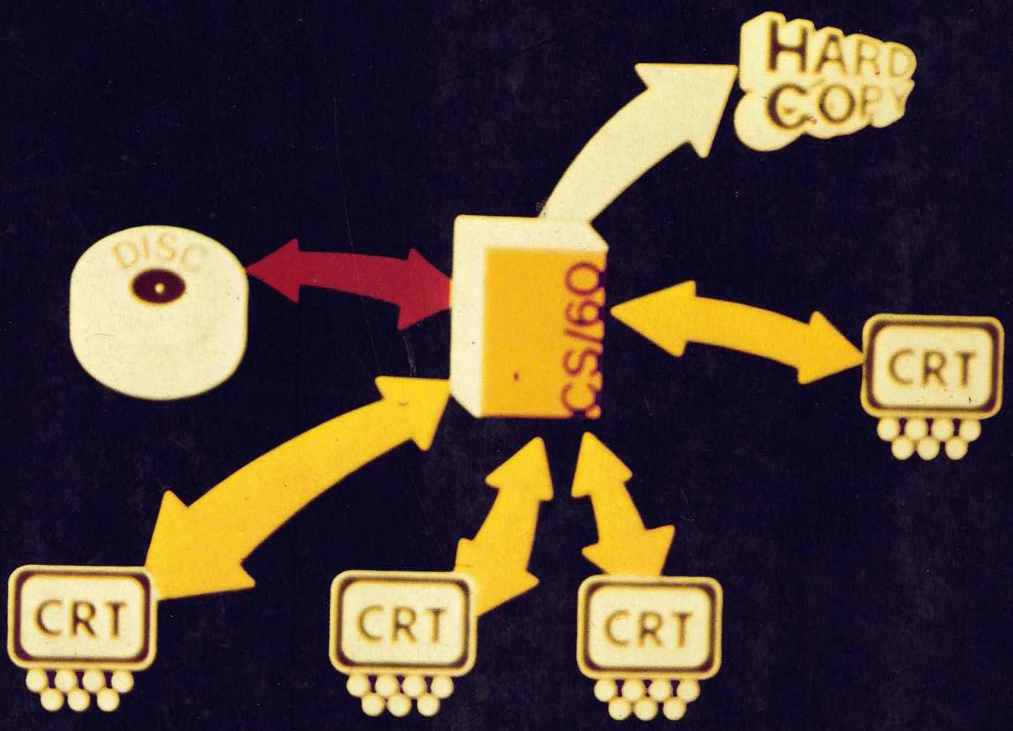
PROGRAMACION DE ORDENADORES

LENGUAJE COBOL



EDITORIAL tesys

TOMO 7

El presente tratado de **Programación de Ordenadores** está compuesto por un conjunto de temas que constituyen la estructura imprescindible para todo aquel que desee un amplio conocimiento del campo de la Programación.

Dichos temas vienen englobados en volúmenes diversos que a continuación se detallan; constituyéndose por separado como unidades monográficas susceptibles de un estudio particular, al margen del conocimiento conjunto y global de todos ellos, como compendio completo para el logro final de programar.

La agrupación por tomos se desarrolla de la siguiente forma:

Tomo 1. **Estructura Básica del Proceso de datos.**

Tomo 2. **Estructura Básica del Proceso de datos.**

Tomo 3. **Estructura Básica del Proceso de datos.**

Tomo 4. **Estructura Básica del Proceso de datos.**

Tomo 5. **Estructura Básica del Proceso de datos.**

Tomo 6. **Lenguaje Basic**

Tomo 7. **Lenguaje Cobol**

Tomo 8. **Lenguaje Cobol**

Tomo 9. **Leyes de Construcción de programas.**

Tomo 10. **Diccionario de Proceso de datos**

El tratamiento de los temas comprende tanto los aspectos teóricos como las aplicaciones al diseño de estructuras básicas de posible uso inmediato, reagrupándose en los siguientes conjuntos específicos que por sí mismos pasan a ser unidades independientes.

Son los siguientes:

Grupo I: Tomos 1, 2, 3, 4 y 5. **(Estructura Básica del Proceso de Datos).**

Grupo II: Tomo 6. **(Lenguaje Basic).**

Grupo III: Tomos 7 y 8. **(Lenguaje Cobol).**

Grupo IV: Tomo 9. **(Leyes de Construcción de Programas).**

PROGRAMACION DE ORDENADORES

LENGUAJE COBOL

PROGRAMACION DE ORDENADORES

LENGUAJE COBOL

AUTOR

JOSE LUIS MORALES PASCUAL
Licenciado en Informática

COLABORADORES

CARLOS ANTONIO MARTINEZ TREVIN
Ingeniero de Telecomunicación

ANDRES J. ERNESTO RUANA GIRONES
Ingeniero Industrial

BARCELONA (ESPAÑA)

© E.C.C. - Barcelona, 1981

Editado por:
EDITORIAL TESYS, Bruch 20, Barcelona-10. (España), 1981

Primera edición: Mayo, 1981

Depósito legal: B.: 25.869-1980 (VII)

I.S.B.N. 84-85838-00-9 (Obra completa)
I.S.B.N. 84-85838-07-6 (Tomo 7)

Impreso por:
GRAFOS, Arte sobre papel.
Paseo Carlos I, 157 - Barcelona (13).

Impreso en España - Printed in Spain

RECONOCIMIENTO

Este manual está basado en el USA Standard COBOL (X3.23-1968) desarrollado por el United States of America Standards Institute (USASI). De acuerdo con su petición, se reproduce a continuación el siguiente «reconocimiento» en su totalidad:

«Cualquier organización interesada en reproducir los informes y especificaciones COBOL, *total o parcialmente, utilizando ideas de este informe como* base para un manual o para cualquier otro propósito, es libre de hacerlo. Sin embargo, dichas organizaciones deben reproducir esta sección como parte de la introducción del documento. A aquellas que utilicen solamente un fragmento o un resumen se les pide mencionar la palabra «COBOL» como reconocimiento del origen pero no es preciso que reproduzcan esta sección entera.

»COBOL es un lenguaje de la industria y no es propiedad de ninguna compañía ni grupo de compañías ni de ninguna organización ni grupo de organizaciones.

»Ninguno de los colaboradores ni el comité COBOL garantizan, expresa o implícitamente, la exactitud y funcionamiento del sistema de programación y el lenguaje. Por otra parte, ninguno de los colaboradores ni el comité asumen ninguna responsabilidad en relación con los mismos.

»Se han establecido procedimientos para el mantenimiento del COBOL. Las consultas sobre los procedimientos para proponer cambios deben dirigirse al Executive Committes of the Conference on Data Systems Languages (CODASYL).

»Los autores y los propietarios literarios del material registrado utilizado:

> FLOW-MATIC (Marca de la Sperry Rand Corporation). Programación para el UNIVAC I y II, Data Automation Systems, registrados en 1958 y 1959 por Sperry Rand Corporation; IBM Commercial Translator Form n.° F 28-8013, registrado en 1959 por IBM; FACT, DSI 27A5260-2760, registrado en 1960 por la Minneapolis-Honeywell

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

CONTENIDO

Introducción 11

1. Elementos del COBOL 13

2. Identification Division y Environment Division 37

3. Data Division 61

4. Procedure Division 95

Soluciones de los ejercicios 147

Indice general 149

INTRODUCCION

El COBOL (abreviatura de COmmon Business Oriented Language, que significa Lenguaje Común Orientado a los Negocios) es un lenguaje de programación diseñado para resolver problemas de tipo comercial.

El COBOL fue desarrollado originalmente por CODASYL (Conference on Data Systems Languages) en el año 1959. A partir de entonces han aparecido diversas versiones, como por ejemplo, COBOL-60, COBOL-61, COBOL-68 (ANSI) o el actual COBOL-74.

Además, algunos aspectos del COBOL son distintos en ordenadores diferentes, por lo que para trabajar con una determinada máquina debe disponerse de un ejemplar del manual suministrado por el fabricante.

En este libro no se estudia una versión particular del COBOL, sino que se desarrollan las características más generales de este lenguaje de programación. Esto se debe a que las diferencias entre las diversas versiones se basan en el número de elementos optativos y en las características del sistema empleado, pero que en esencia todos son el mismo lenguaje.

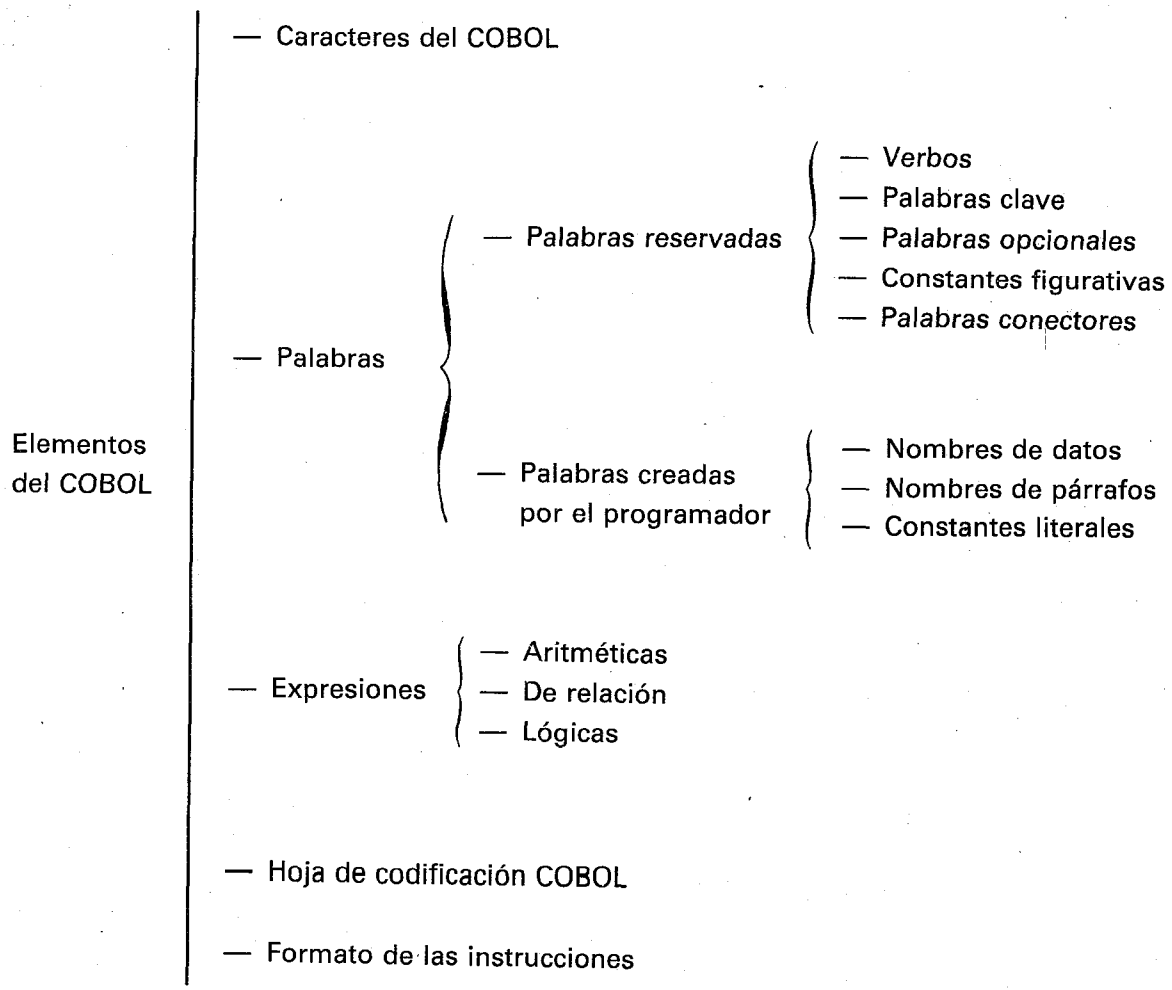
El COBOL se ha diseñado para producir programas fuente que sean comunes o que puedan utilizarse en la mayoría de los ordenadores.

Por último, hay que destacar, de entre sus características, que se trata de un lenguaje relativamente fácil de interpretar por el hombre, ya que en un programa COBOL predominan las palabras y frases (escritas en inglés) sobre las fórmulas o expresiones matemáticas.

1.

ELEMENTOS DEL COBOL

CUADRO RESUMEN DE LA LECCION



1. ELEMENTOS DEL COBOL

Los elementos que constituyen el lenguaje de programación COBOL deben escribirse de forma exacta y precisa, es decir, según un *formato* establecido. Estos elementos constan de:

— Palabras imprescindibles para el tratamiento adecuado de un programa COBOL.

— Palabras opcionales.

— Información que debe proporcionar el programador (usuario).

— Etc.

En los siguientes apartados se presentan estos elementos y sus *formatos* (reglas de presentación).

1.1. CARACTERES DEL COBOL

En un octeto (byte) pueden representarse hasta 256 caracteres distintos que corresponden a las 256 combinaciones posibles de los ocho bits del byte ($2^8 = 256$).

De estos 256 caracteres, en COBOL se utilizan generalmente 52 caracteres:

Caracteres	*Significado*
0, 1, 2, ..., 9	Cifras
A, B, C, ..., Z	Letras
b	Blanco o espacio (en la hoja de codificación se escribe como b)
.	Punto
,	Coma
;	Punto y coma
'	Apóstrofo
"	Comillas (el apóstrofo y las comillas se utilizan indistintamente)
=	Signo igual
<	Menor que
>	Mayor que
+	Signo más
–	Signo menos o guión
*	Asterisco
/	Barra inclinada
$	Signo monetario (Dólar)
(	Paréntesis izquierdo
)	Paréntesis derecho

1.2. PALABRAS

En la codificación de un programa COBOL se utilizan dos tipos de palabras:

— Palabras reservadas.

— Palabras creadas por el usuario.

1.2.1. Palabras reservadas

Las palabras reservadas sólo pueden escribirse como indica este texto, y siempre en inglés (y en mayúscula), ya que cada una de ellas tiene un determinado significado para el compilador COBOL.

Palabras reservadas	*Significado*
ACCEPT	aceptar
ACCESS	acceso
ACTUAL	actual
ADD	sumar
ADVANCING	avanzando
AFTER	después
ALL	todo
ALPHABETIC	alfabético
ALTER	modificar
ALTERNATE	alterno
AND	y
APPLY	aplicar
ARE	son
AREA	área
AREAS	áreas
ASCENDING	ascendente
ASCII	(código)
ASSIGN	asignar
AT	a, en
AUTHOR	autor
BEFORE	antes
BEGINNING	comenzando
BLANK	blanco (espacio)
BLOCK	bloque
BLOCK-COUNT	contador de bloques
BLOCK-LENGTH-CHEK	control de bloque
BY	por
CALL	llamar
CARD-PUNCH	perforadora de tarjetas
CARD-READER	lectora de tarjetas
CHARACTER	carácter
CHARACTERS	caracteres
CHANGED	cambiado
CLOSE	cerrar
COBOL	(lenguaje)

Palabras reservadas	*Significados*
COMMA	coma
COMP	computacional
COMP-1	computacional-1
COMP-2	computacional-2
COMP-3	computacional-3
COMP-4	computacional-4
COMPUTATIONAL	computacional
COMPUTATIONAL-1	computacional-1
COMPUTATIONAL-2	computacional-2
COMPUTATIONAL-3	computacional-3
COMPUTATIONAL-4	Computacional-4
COMPUTE	computar, calcular
CONFIGURATION	configuración
CONTAINS	contiene
COPY	copiar
CORR	correspondiendo
CORRESPONDING	correspondiendo
CURRENCY	corriente, curso
DATA	datos
DATE-COMPILED	fecha de compilación
DATE-WRITTEN	fecha de escritura
DECIMAL-POINT	punto decimal
DECLARATIVES	declarativas
DEPENDING	dependiendo
DESCENDING	descendiendo
DIRECT	directo
DISC	disco
DISPLAY	visualizar
DIVIDE	dividir
DIVISION	división
DOWN	abajo
EBCDIC	(código)
ELSE	si no, de otro modo
END	fin
ENDING	final

Palabras reservadas	*Significados*
ENTER	entrar
ENTRY	entrada
ENVIRONMENT	equipo
EQUAL	igual
EQUALS	iguales
ERROR	error
EVERY	cada
EXAMINE	examinar
EXCEEDS	excede
EXHIBIT	exhibir
EXIT	salida
EXTENDED	extendido
FD	descripción de fichero
FILE	fichero
FILE-CONTROL	control de ficheros
FILLER	relleno
FIRST	primero
FOR	para
FORM-OVERFLOW	área de desbordamiento
FROM	de, desde
GENERATE	generado
GIVING	dando
GO	ir, vaya
GREATER	mayor
HIGH-VALUE	valor más alto
HIGH-VALUES	valores más altos
I-O	entrada-salida (Input-Output)
I-O-CONTROL	control de entrada-salida
IDENTIFICATION	identificación
IF	si
IN	en
INDEX	índice
INDEXED	indexado
INDICES	índices
INITIATE	iniciado
INPUT	entrada
INPUT-OUTPUT	entrada-salida
INSERT	insertar
INSTALLATION	instalación
INTO	en, a
INVALID	inválido
IS	es
JUST	justificado
JUSTIFIED	justificado
KEY	clave
LABEL	etiqueta
LEADING	llevar
LEFT	izquierdo
LESS	menor, menos
LINE	línea
LINES	líneas
LINKAGE	acoplamiento
LOCK	bucle, cerrar
LOW-VALUE	valor más bajo
LOW-VALUES	valores más bajos
MAP	mapa
MASTER-INDEX	índice-maestro
MEMORY	memoria
MODE	modo
MODULES	módulos
MONITOR	monitor
MOVE	mover, transferir
MULTIPLE	múltiple
MULTIPLY	multiplicar

Palabras reservadas	*Significados*
NEGATIVE	negativo
NEXT	siguiente
NO	no
NOT	no
NOTE	anotación, comentario
NUMERIC	numérico
OBJECT-COMPUTER	ordenador-objeto
OCCURS	ocurre
OF	de
OFF	fuera de
OMITTED	omitido
ON	en, sobre, según
OPEN	abrir
OPTIONAL	opcional
OR	o
ORGANIZATION	organización
OTHERWISE	de lo contrario, en otro caso
OUTPUT	salida
PERFORM	bifurcar
PIC	formato, modelo (Picture)
PICTURE	formato, modelo (Pic)
POSITION	posición
POSITIVE	positivo
PRINTER	impresora
PROCEDURE	procedimiento
PROCEED	procede
PROCESSING	procesamiento, tratamiento
PROGRAM	programa
PROGRAM-ID	identificación de programa
QUOTE	comilla
QUOTES	comillas
RANDOM	aleatorio
READ	leer
READY	listo, preparado
RECORD	registro
RECORDING	registrando, registrar
RECORDS	registros
REDEFINES	redefinir
REEL	carrete
RELATIVE	relativo
RELEASE	permitir
REMAINDER	resto
REMARKS	comentarios, observaciones
RENAMES	renombre
REPLACING	reemplazar
RERUN	reiniciar
RESERVE	reservar
RESET	volver a colocar
RETURN	retornar
REVERSED	inverso
REWIND	rebobinar
REWRITE	reimprimir
RIGHT	derecho
ROUNDED	redondeado
RUN	empezar
SAME	mismo
SD	Sort Description
SEARCH	buscar, explorar
SECTION	sección
SECURITY	seguridad
SEEK	buscar
SELECT	elegir, seleccionar
SENTENCE	sentencia, oración
SEQUENTIAL	secuencial
SET	poner

Palabras reservadas	Significados
SIGN	signo
SIZE	tamaño
SORT	clasificación
SOURCE-COMPUTER	ordenador-fuente
SPACE	espacio
SPACES	espacios
SPECIAL-NAMES	nombres especiales
STANDARD	estándar, normal
STATUS	estado
STOP	parar
SUBTRACT	restar
SYMBOLIC	simbólico
SYNC	sincronizado
SYNCHRONIZED	sincronizado
SYSERR	error
SYSLST	listado
SYSSWCH	interruptor
SYSTIME	tiempo
TALLY	copia
TALLING	copiar
TAPE	cinta
TAPES	cintas
TERMINATE	terminado
THAN	que (comparación)
THEN	entonces
THROUGH	a través de
THRU	verdadero
TIME	tiempo, vez
TIMES	tiempos, veces
TO	a
TRACE	señal
TRACKS	señales
TRAILING	señalar
TRANSFORM	transformar
UNEQUAL	distinto
UNIT	unidad
UNTIL	hasta
UP	arriba
UPON	a través
USAGE	uso
USE	usar
USING	usando
VALUE	valor
VALUES	valores
VARYING	variando
VERIFY	verificar
WHEN	cuando
WITH	con
WORDS	palabras
WORKING-STORAGE	área de trabajo
WRITE	escribir
ZERO	cero
ZEROS	ceros
ZEROES	ceros

Las palabras reservadas pueden ser de distintos tipos:

— Verbos
— Palabras clave
— Palabras opcionales
— Constantes figurativas
— Palabras conectores

Verbos

Los *verbos* expresan acciones ejecutadas por el programa objeto o por el compilador COBOL. Por ejemplo: MOVE (mover), ADD (sumar), WRITE (escribir), etc.

Estas palabras representan una orden para el ordenador.

Palabras clave

Las *palabras clave* deben aparecer siempre en una determinada *cláusula* (palabra o conjunto de palabras que expresan un concepto determinado).

Las palabras clave se presentan en los formatos escritas en ***ESTE TIPO DE LETRA.*** Por ejemplo:

ADD A, B, C ***GIVING*** D

Palabras opcionales

Las *palabras opcionales* se utilizan para facilitar la lectura de un programa COBOL. Su presencia o ausencia no altera el significado o el tratamiento de la proposición (declaración) durante la compilación o ejecución del programa.

Las palabras opcionales se representan, en los formatos, escritas en *ESTE TIPO DE LETRA.* Por ejemplo:

DATA RECORD *IS* DATO-1

La palabra IS es palabra reservada opcional, mientras que DATA y RECORD son palabras clave. En este ejemplo, DATO-1 no es una palabra reservada.

Constantes figurativas

Las *constantes figurativas* son una serie de palabras reservadas que tienen un valor determinado (constante). Se utilizan para asignar un valor concreto, numérico o no numérico, a un elemento.

Las constantes figurativas son las siguientes:

a) Constantes que representan el valor numérico cero.

ZERO: genera un cero.
ZEROS: genera dos o más ceros.
ZEROES: genera dos o más ceros.

b) Constantes que representan espacios o blancos (b).

SPACE: genera un espacio.
SPACES: genera dos o más espacios.

c) Constantes que representan los valores más altos o más bajos en la secuencia de intercalación del sistema operativo.

HIGH-VALUE: genera un carácter FF hexadecimal.

HIGH-VALUES: genera dos o más caracteres FF hexadecimales.

LOW-VALUE: genera un carácter 00 hexadecimal.

LOW-VALUES: genera dos o más caracteres 00 hexadecimales.

d) Constantes que generan apóstrofos (').

QUOTE: genera un apóstrofo.
QUOTES: genera dos o más apóstrofos.

e) Constantes que generan literales.

ALL: genera una o más veces el literal que le sigue. Este literal debe ser no numérico o una constante figurativa que no sea ALL.

La constante ALL no puede utilizarse con las proposiciones DISPLAY, EXAMINE, STOP ni COPY.

Palabras conectores

Las *palabras conectores* se utilizan para asociar frases entre sí.

Existen algunas palabras conectores que son opcionales y otras de las que no se puede prescindir en una sentencia. Las palabras conectores son las siguientes:

a) IN (en) y OF (de): se utilizan para asociar nombres de campos.

b) OR, AND y NOT: se llaman conectores lógicos y se utilizan para formar condiciones compuestas.

c) , (coma): se utiliza para enlazar dos o más operandos o proposiciones consecutivas. Este carácter es opcional.

1.2.2. Palabras creadas por el programador

El programador debe facilitar palabras en el programa para expresar campos, datos o conceptos determinados. Las palabras reservadas no pueden ser utilizadas para esta función. Por ejemplo, ERROR es una palabra reservada pero ERROR-3, ERROR3, REGISTRO, FICHERO, etc., son palabras creadas por el programador para nombrar determinados conceptos.

Las palabras creadas por el usuario (o programador) pueden ser de tres tipos:

— Nombres de datos.
— Nombres de párrafos.
— Constantes literales.

Nombres de datos

El conjunto de uno o más caracteres constituye un *campo de datos*. Por ejemplo, un campo de datos puede estar formado por el número de cuenta o por la dirección de un empleado de una determinada empresa.

Un campo de datos contiene información que es procesada. Para hacer referencia a un campo de datos se utiliza un símbolo o una palabra denominada *nombre de datos.*

En cada uno de los procesos que realiza el ordenador se emplean cantidades variables cuyos valores asignados a ellas pueden cambiar o variar durante

la ejecución de un programa. En el lenguaje COBOL se hace referencia a estas variables por medio de nombres de datos.

Las reglas para crear un nombre de datos son las siguientes:

— Debe comprender de 1 a 30 caracteres.

— Los caracteres deben ser letras, dígitos y guiones. No debe contener espacios en blanco.

— Uno de los caracteres, por lo menos, debe ser una letra.

— Ningún nombre de datos puede empezar ni terminar por un guión (–).

Las palabras reservadas no pueden usarse como nombres de datos ni para funciones distintas de las que especifica cada una de ellas.

Ejemplo:

Los siguientes nombres de datos son válidos:

A2
2AB
EMPLEADO
NOMBRE
NOMBRE-3
AB754
NOMBRE-EMPLEADO

Los siguientes nombres de datos no son válidos:

-AB3
NOMBRE EMPLEADO
AB3-
37-35
A.B
15.5
$15.5

Nombre de párrafos

Un programa COBOL está constituido por *párrafos.* Un *párrafo* está formado, en COBOL, por un conjunto de *frases,* siendo una frase, a su vez, un conjunto de palabras. Cada frase constituye una instrucción para el ordenador.

Las frases deben ir separadas entre sí por los caracteres blanco (b), coma (,), punto (.) o punto y coma (;).

Cada párrafo debe tener un *nombre de párrafo* para poder hacer referencia a ese párrafo o para distinguirlo de los demás párrafos.

Al final de cada párrafo debe ir obligatoriamente un punto (.).

Los nombres de párrafo deben escribirse siguiendo las siguientes reglas:

— Pueden contener de 1 a 30 caracteres.

— Los caracteres utilizados son las cifras (del 0 al 9), las letras (de la A a la Z) y el guión (-).

— El guión (-) no puede ser el primero ni el último carácter.

— Debe preceder a un párrafo referenciado.

Constantes literales

Una *constante literal* es un conjunto de uno o más caracteres que no cambia de valor durante la ejecución de un programa.

Las constantes literales pueden ser de dos tipos:

— Literales numéricos.
— Literales no numéricos.

Literales numéricos

Los literales numéricos son series de caracteres con los que se pueden realizar operaciones aritméticas. Pueden ser de dos tipos:

— De punto fijo.
— De punto flotante.

Un literal numérico de *punto fijo* es una serie de hasta 20 caracteres formada por las cifras (0 a 9), el signo (+ ó −) y un punto decimal (·).

Las reglas son las siguientes:

— Un literal numérico de punto fijo debe contener de 1 a 18 dígitos más el signo y un punto decimal.

— Sólo puede contener un signo, y éste debe ser el primer carácter de la izquierda. Si no contiene signo se considera que es positivo.

— Sólo puede contener un punto decimal. Si no contiene punto decimal se considera que es entero.

— El punto decimal no puede ser el último carácter de la serie.

— Los signos y el punto decimal son opcionales.

Ejemplo:

Son válidos los siguientes literales numéricos de punto fijo.

```
15740
-324
+4167.05
```

No son válidos los literales:

```
15740.
-324.5A
  27B
```

Un literal numérico de *punto flotante* es una serie de hasta 22 caracteres formada por las cifras (0 a 9), el signo (+ ó −), el punto decimal (.) y el carácter E.

El formato, es decir, la forma de representar el literal numérico de punto flotante, es el siguiente:

[{±}] mantisa E [{±}] exponente

En este formato, los corchetes indican que los signos son opcionales (si no se ponen se consideran que la mantisa o el exponente son positivos); las llaves indican las dos opciones posibles (es decir, + ó −).

Otras reglas son:

— La mantisa debe contener de 1 a 16 dígitos y un punto decimal.

— El exponente debe contener uno o dos dígitos. No debe sobrepasar los límites de 0.72E76 (es decir, 0.72 * 10 ** 76) y de 5.4E−79.

Ejemplos:

```
+ 256E−2        − 0.56E12        3.45E+20
```

Literal no numérico

Los literales no numéricos (o alfanuméricos) se utilizan para imprimir datos de salida (cabeceras, etc.) y para procesar caracteres. No pueden ser utilizados para realizar operaciones aritméticas.

Las reglas para la codificación de un literal no numérico son:

— Debe escribirse entre apóstrofos o comillas.

— Debe contener de 1 a 132 caracteres.

— Entre las comillas pueden escribirse cualquier tipo de caracteres excepto otras comillas o apóstrofos.

— Pueden utilizarse, si se desea, palabras reservadas (entre comillas).

— Todos los espacios encerrados entre comillas forman parte del literal.

Ejemplos:

```
'FICHERO MAESTRO'
'356.74'
'A – B – C'
```

1.3. EXPRESIONES

Otro de los elementos del COBOL son las *expresiones*. Las expresiones pueden ser de tres tipos:

— Expresiones aritméticas.
— Expresiones de relación.
— Expresiones lógicas.

Expresiones aritméticas

Las *expresiones aritméticas* están formadas por la combinación de constantes y variables por medio de los operadores aritméticos.

Las constantes son literales numéricas; y las variables, nombres de datos.

Los operadores aritméticos son:

+ adición
− sustracción
* multiplicación
/ división
** potenciación

Para formar expresiones aritméticas en COBOL deben respetarse las siguientes reglas:

— Se utilizan paréntesis para agrupar operaciones al igual que en álgebra.

— Antes de abrir un paréntesis (y después de cerrarlo) debe dejarse un espacio en blanco.

— No pueden aparecer dos operadores aritméticos uno a continuación del otro.

— Los operadores aritméticos deben ir precedidos y seguidos de un espacio en blanco (excepto en el caso de que sea una constante con signo y vaya encerrada entre paréntesis).

Ejemplos:

```
A * B - C + D / X ** 2
35 * (A + B) / (C * D)
0.5 * (- 26)
- A + B
```

Expresiones de relación

Las *expresiones de relación* utilizan los operadores de relación:

Operador	*Significado*	*Palabras COBOL*
=	es igual a	IS EQUAL TO
<	es menor que	IS LESS THAN
>	es mayor que	IS GREATER THAN

Ejemplo:

```
A = B    A IS EQUAL TO B
A < B    A IS LESS THAN B
A > B    A IS GREATER THAN B
```

Expresiones lógicas

Una *expresión lógica* es una serie de constantes y variables relacionadas por medio de operadores lógicos.

Los operadores lógicos son:

NOT	no (negación lógica)
OR	o (suma lógica)
AND	y (producto lógico)

Las operaciones lógicas relacionan variables aritméticas con variables lógicas (Verdadero = True; Falso = False).

Ejemplo:

— Negación lógica (NOT):

Si A1 = T entonces NOT A1 = F
Si A1 = F entonces NOT A1 = T

— Suma lógica (OR):

A1 OR A2 → A3
Si A1 = T OR A2 = T entonces A3 = T

siendo A3 el resultado de la suma lógica (A1 OR A2).

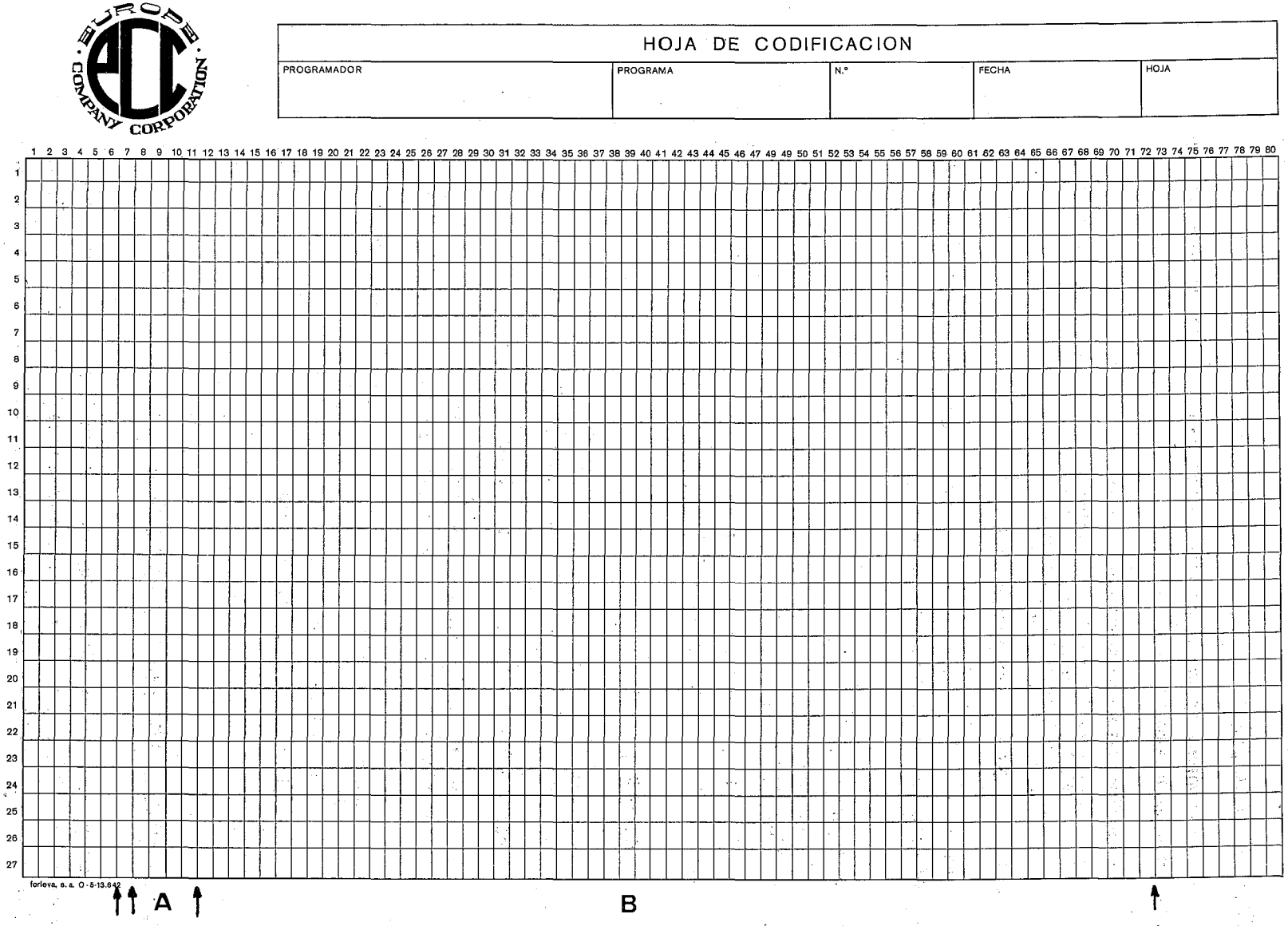
Si A1 = T OR A2 = F entonces A3 = T
Si A1 = F OR A2 = F entonces A3 = F

— Producto lógico (AND):

A1 AND A2 → A3
Si A1 = T AND A2 = T entonces A3 = T, en otro caso, A3 = F.

1.4. HOJA DE CODIFICACION COBOL

Las hojas de codificación se han diseñado de forma que sirvan de documento fuente para perforar fichas (de 80 columnas) de un programa COBOL. Cada línea de esta hoja representa una ficha. La hoja tiene 80 columnas que corresponden a las 80 columnas de una ficha.



Hoja de codificación COBOL

— Las columnas 1 a 6 se utilizan para numerar las líneas del programa.

— La columna 7 se emplea para poner un guión que significa la continuación de literales no numéricos.

— Las columnas 8 a 72 se emplean para escribir las frases o instrucciones del programa. Estas columnas están agrupadas en dos *márgenes*, el margen A (columnas 8-11) y el margen B (columnas 12-72).

— Las columnas 73-80 pueden utilizarse opcionalmente para escribir el nombre del programa.

En el margen A deben comenzar obligatoriamente:

— Los títulos de división.

— Los títulos de sección.

— Los títulos de párrafo.

— Las descripciones de ficheros (indicadores de nivel).

Todas las demás frases comienzan en el margen B.

1.4.1. Reglas de codificación:

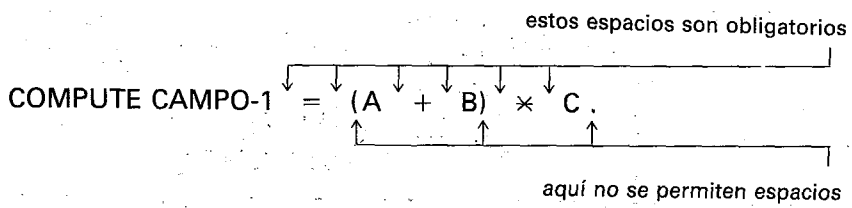
Cada elemento debe estar separado del siguiente por un espacio, como mínimo, excepto algunos símbolos. Por ejemplo, el paréntesis izquierdo no debe ir seguido de un espacio. Por otro lado, no pueden ir precedidos por un espacio los símbolos siguientes:

— El paréntesis derecho.

— La coma o el punto y coma que separan las frases entre sí.

— El punto final de una frase.

Ejemplo:



1.5. FORMATO DE LAS INSTRUCCIONES

Los formatos son ejemplos de la forma exacta que debe tener cada instrucción. Así pues, dentro del formato se utiliza una determinada anotación:

— Las palabras reservadas se escriben totalmente en mayúsculas.

— Las palabras reservadas obligatorias se transcriben en ***ESTE TIPO DE LETRA.*** Estas palabras sólo podrán omitirse si la parte de formato que las incluye es opcional.

— Las palabras reservadas opcionales se transcriben en *ESTE OTRO TIPO DE LETRA.* Estas palabras sólo se emplean para hacer la instrucción más legible.

— La información a crear por el programador se escribe totalmente en minúsculas.

— Las partes opcionales del formato se encierran dentro de corchetes. Estas partes pueden emplearse u omitirse según las exigencias del programa.

— Una elección entre partes opcionales se indica encerrándolas dentro de otros corchetes más grandes. Esto indica que puede emplearse una parte o ninguna.

— Una elección entre partes obligatorias se indica encerrándolas dentro de llaves. Esto indica que debe emplearse una de las partes.

— Las partes que puedan repetirse varias veces van seguidas por tres puntos. Cuando los puntos van después de una palabra, sólo se refieren a esta palabra. Cuando los puntos van después de corchetes o llaves, se refieren a toda la parte del formato encerrada.

Ejemplo de formato:

FILE-CONTROL . ***SELECT*** [***OPTIONAL***] nombre-de-fichero-1

ASSIGN *TO* [entero-1] nombre-de-dispositivo

[***RESERVE*** { entero-2 / ***NO*** } *ALTERNATE* { *AREA* / *AREAS* }]

[***ACCESS*** *MODE* ***IS*** { ***RANDOM*** / ***SEQUENTIAL*** }]

EJERCICIOS

1. Las columnas 8-11 de la hoja de codificación se emplean para:

   a) Numerar las líneas.
   b) Comenzar todas las frases del programa.
   c) Comenzar determinadas frases.

2. La constante figurativa HIGH-VALUE genera un carácter:

   a) 00 hexadecimal.
   b) 7D hexadecimal.
   c) FF hexadecimal.

3. Los conectores lógicos OR, AND y NOT se utilizan para:

   a) Asociar nombres de campos.
   b) Enlazar dos o más operandos.
   c) Formar condiciones compuestas.

4. Las palabras creadas por el programador pueden ser:

   a) Palabras clave.
   b) Palabras reservadas.
   c) Nombres de datos.

5. Los nombres de datos deben contener:

   a) De 1 a 30 caracteres.
   b) De 1 a 8 caracteres.
   c) De 1 a 20 caracteres.

6. El siguiente nombre de dato es válido:

   a) 257.
   b) NOMBRE CLIENTE.
   c) NOMBRE-CLIENTE.

7. Al final de cada párrafo debe ir obligatoriamente:

   a) Una coma.
   b) Un punto.
   c) Un punto y coma.

8. ¿Cuál es el válido de los literales numéricos de punto fijo siguientes?:

   a) 173002.
   b) 427BD
   c) −132.02

9. ¿Cuál es el válido de los literales numéricos de punto flotante siguientes?:

   a) A73E−2
   b) −73E+3
   c) −1005

10. ¿Cuál de los literales no numéricos siguientes es válido?:

    a) 'NOMBRE DEL EMPLEADO'
    b) FICHERO MAESTRO.
    c) NOMBRE-DEL-EMPLEADO.

11. La columna 7 de la hoja de codificación COBOL se emplea para:

    a) Numerar la línea.
    b) Poner un guión para la continuación.
    c) Comenzar a codificar el programa.

2.

IDENTIFICATION DIVISION Y ENVIRONMENT DIVISION

CUADRO RESUMEN DE LA LECCCION

DIVISIONES DE UN PROGRAMA COBOL (I).

- 2.1. IDENTIFICATION DIVISION
  - — Formato
  - — Ejemplo de codificación
- 2.2. ENVIRONMENT DIVISION
  - — CONFIGURACION SECTION
    - — SOURCE-COMPUTER
    - — OBJECT-COMPUTER
    - — SPECIAL-NAMES
  - — INPUT-OUTPUT SECTION
    - — FILE-CONTROL
      - — SELECT
      - — OPTIONAL
      - — ASSIGN
      - — MULTIPLE REEL
      - — RESERVE
      - — ACCES MODE
      - — ORGANIZATION
      - — ACTUAL KEY
      - — RELATVE KEY
      - — SYMBOLIC KEY
      - — RECORD KEY
      - — PRIORITY
    - — I-O-CONTROL
      - — APPLY
      - — RERUN
      - — SAME AREA
  - — Ejemplo de codificación

Los programas COBOL constan de una serie de *frases* formadas a su vez por palabras. Cada frase representa una instrucción para la máquina.

Estas instrucciones están dispuestas formando *divisiones, secciones* y *párrafos.* En general, una división consta de secciones, y una sección está formada por párrafos.

2. DIVISIONES DE UN PROGRAMA COBOL

Todos los programas COBOL están divididos en cuatro divisiones las cuales deben aparecer en el orden siguiente:

1) IDENTIFICATION DIVISION.
2) ENVIRONMENT DIVISION.
3) DATA DIVISION.
4) PROCEDURE DIVISION.

La IDENTIFICATION DIVISION contiene informaciones que identifican el programa. En general, sirve para informar a quienes lo leen.

La ENVIRONMENT DIVISION contiene información sobre el equipo que deberá emplearse para compilar y ejecutar el programa objeto. Esta división enlaza los dispositivos del ordenador con los ficheros de datos a procesar. Cada fichero queda asignado, por su nombre, a un dispositivo de entrada o de salida.

La DATA DIVISION describe los datos a procesar por el programa objeto. Describe los campos que constituyen cada uno de los ficheros indicados en la

ENVIRONMENT DIVISION, y además describe los campos que constituyen la memoria central o memoria de trabajo (tales como constantes y áreas de trabajo).

La PROCEDURE DIVISION indica las operaciones necesarias para procesar los datos. Indica también el orden en que deberán ejecutarse las operaciones.

2.1. IDENTIFICATION DIVISION

La *Identification Division (división de identificación)* debe contener información para identificar el programa fuente. En esta división es obligatorio codificar el título de la división (IDENTIFICATION DIVISION) y el párrafo PROGRAM-ID (*identificación del programa)* el cual debe contener el nombre del programa (que se elige arbitrariamente).

La *Identification Division* debe aparece en todos los programas fuente y delante del resto de las divisiones.

La forma en que debe codificarse esta división, es decir, el formato de esta división es el siguiente:

Formato:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. nombre-de-programa.
```

Los puntos que aparecen en este formato son obligatorios (después de cada párrafo o de un nombre de párrafo debe aparecer un punto). A continuación de cada punto debe dejarse como mínimo un espacio en blanco.

El *nombre-de-programa* sirve para identificar el programa en una ficha de control de trabajo cuando se ejecuta el programa objeto.

El *nombre-de-programa* consta de 1 a 30 caracteres alfabéticos o numéricos, siendo el primero de ellos alfabético (en algunos sistemas, el nombre del programa debe contener 8 caracteres como máximo y encerrados entre comillas).

En esta división pueden incluirse, opcionalmente, cláusulas tales como: autor, instalación, fecha de codificación, fecha de compilación y observaciones. Quedando el formato siguiente:

IDENTIFICATION DIVISION.

PROGRAM-ID. nombre-del-programa.

[***AUTHOR.*** [comentario] ...]

[***INSTALLATION.*** [comentario] ...]

[***DATE-WRITTEN.*** [comentario] ...]

[***DATE-COMPILED.*** [comentario] ...]

[***SECURITY.*** [comentario] ...]

[***REMARKS.*** [comentario] ...]

De todas las cláusulas que aparecen en este formato de la IDENTIFICATION DIVISION sólo PROGRAM-ID afecta al programa objeto.

A Nombre del centro de calculo en el que se procesa.
B Quien puede utilizar el programa?,

Ejemplo de codificación:

EUROPE COMPANY CORPORATION

HOJA DE CODIFICACION

PROGRAMADOR	PROGRAMA	N.º	FECHA	HOJA

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1  IDENTIFICATION DIVISION.
2  PROGRAM-ID. EJEMPLO.
3  AUTHOR. DPTO PROGRAMACION.
4  DATE-WRITTEN. 8-5-80.
5  DATE-COMPILED. 9-5-80.
6  SECURITY. MAIN MENU.
7  REMARKS.
8      ESTE PROGRAMA ES EL PROGRAMA PRINCIPAL DE LA
9      APLICACION DEL CONTROL Y MANTENIMIENTO DEL
10     FICHERO DE ALUMNOS CREADO POR EL PROGRAMA
11     "NALUM"
12     UTILIZA LOS SUBPROGRAMAS SIGUIENTES:
13
14     * EJEM-1 - CONSULTA DE REGISTROS
15     * EJEM-2 - MODIFICACION DE REGISTROS
16     * EJEM-3 - DESTRUCCION DE REGISTROS
17     * EJEM-4 - RECUPERACION DE REGISTROS.
18
19
20
21
22
23
24
25 *
26
27
```

forieva, s. a. O-5-13.642

2.2. ENVIRONMENT DIVISION

La *división de equipo* (ENVIRONMENT DIVISION) especifica los elementos del programa COBOL que dependen de las características físicas de los sistemas de proceso de datos empleados, tanto para la compilación del programa fuente, como para la ejecución del programa objeto.

En esta división, además, se asigna cada fichero a un dispositivo de entrada/salida y se especifican las condiciones especiales de E/S.

Esta división debe estar presente en todos los programas COBOL. Su formato es el siguiente:

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. ordenador-fuente.

OBJECT-COMPUTER. ordenador-objeto.

[***SPECIAL-NAMES.*** anotación.]

INPUT-OUTPUT SECTION.

FILE-CONTROL. [anotación.]

[***I-O-CONTROL.*** anotación.]

Los nombres de sección y de párrafo son obligatorios cuando se escriban los párrafos correspondientes. Estos nombres deben comenzar a escribirse en el margen A (columnas de la 8 a la 11). Los párrafos o anotaciones correspondientes deben comenzar a codificarse en el margen B (columnas 12 a 71).

Esta división puede tener dos secciones:

— CONFIGURATION SECTION (sección de configuración).
— INPUT-OUTPUT SECTION (sección de entrada-salida).

2.2.1. Configuration section

Esta sección especifica las características de los ordenadores fuente y objeto y especifica además nombres especiales o mnemotécnicos para los diferentes dispositivos del equipo.

El formato de esta sección es:

CONFIGURATION SECTION.

SOURCE-COMPUTER. ordenador-fuente.

OBJECT-COMPUTER. ordenador-objeto.

[***SPECIAL-NAMES.*** anotación]

Source-Computer

Este párrafo contiene el nombre del ordenador fuente, es decir, el ordenador que compila el programa fuente. Permite al programador especificar el nombre del ordenador para su identificación.

El nombre del ordenador depende del sistema utilizado y no afecta al programa objeto.

Ejemplo:

```
SOURCE-COMPUTER. UNIVAC-9030.
```

Object-Computer

Este párrafo especifica el ordenador que se emplea para ejecutar el programa compilado (programa objeto).

Generalmente, el ordenador utilizado para la compilación del programa es el mismo que lo ejecuta.

Ejemplo:

```
OBJECT-COMPUTER. UNIVAC-9030.
```

El nombre del ordenador, en este caso, tampoco afecta al programa objeto, sólo sirve para identificar el ordenador utilizado para la ejecución del programa.

En este párrafo, además, puede añadirse, opcionalmente, más información respecto al ordenador que ejecuta el programa. Esta información es referente al tamaño de la memoria central y a la parte de la MC que puede ser utilizada por el compilador o por el programa objeto en el caso de que algunas partes de la MC vayan a utilizarse para otros fines.

El formato del párrafo OBJECT-COMPUTER en este caso es:

```
OBJECT-COMPUTER. ordenador-objeto.[ MEMORY SIZE entero { CHARACTERS } ]
                                                        { WORDS      }
```

La cláusula MEMORY SIZE es opcional y define la MC como un número entero de caracteres (CHARACTERS) o palabras (WORDS).

Special-Names

El párrafo de *nombres-especiales* se usa para especificar nombres mnemotécnicos para los diferentes dispositivos del equipo y para las condiciones de dichos dispositivos.

El proveedor de un determinado sistema especifica un nombre de dispositivo como, por ejemplo, PRINTER a la impresora. El programador, mediante la cláusula SPECIAL-NAMES, puede dar otro nombre a este dispositivo, por ejemplo, IMPRESORA.

En el formato de este párrafo pueden darse varias opciones:

A)

```
SPECIAL-NAMES. nombre-de-dispositivo-1 IS nombre mnemotécnico-1
                [,nombre-de-dispositivo-2 IS nombre mnemotécnico-2...].
```

B)

SPECIAL-NAMES. nombre-de-dispositivo-1 [***IS*** nombre-mnemotécnico-1]
[, ***ON*** *STATUS* ***IS*** nombre-de-condición-1]
[, ***OFF*** *STATUS* ***IS*** nombre-de-condición-2]
[, nombre-de-dispositivo-2...].

Switch = bit

En este caso, si el dispositivo es un interruptor (por ejemplo, con el nombre SWITCH-1). El estado (STATUS) de este interruptor puede establecerse en ON o en OFF, es decir, *en línea* o *fuera de línea.*

El programador puede darles a estos estados, por ejemplo, los nombres de condición EN-LINEA y FUERA-DE-LINEA. La codificación de este párrafo será, en este caso:

```
SPECIAL-NAMES. SWITCH-1  ON STATUS IS EN-LINEA,
                         OFF STATUS IS FUERA-DE-LINEA.
```

C)

Otra de las opciones de este párrafo es la que utiliza la cláusula DECIMAL-POINT IS COMMA (el punto decimal es una coma):

```
SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
```

Esta cláusula invierte las funciones del punto decimal y la coma.

2.2.2. Input-Output Section

La *sección de entrada/salida* permite especificar los dispositivos de E/S de los ficheros utilizados por el programa fuente y proporciona la información sobre la transferencia de datos entre los dispositivos de E/S y el programa objeto.

Esta sección contiene dos párrafos:

— FILE-CONTROL (control de ficheros).
— I-O-CONTROL (control de E/S).

El formato es el siguiente:

INPUT-OUTPUT SECTION.

FILE-CONTROL. anotación.

[***I-O-CONTROL.*** anotación.]

La INPUT-OUTPUT SECTION se incluye en la ENVIRONMENT DIVISION cuando se emplean ficheros de entrada o de salida en los programas. Puede decirse, por lo tanto, que en la mayoría de los casos debe incluirse esta sección.

En el caso de codificarse esta sección, debe incluirse obligatoriamente el párrafo FILE-CONTROL, siendo opcional el párrafo I-O-CONTROL.

File-Control

El párrafo FILE-CONTROL realiza la función de asignar los ficheros a determinados dispositivos del equipo. Es decir, permite dar un nombre a cada fichero, identificar el dispositivo del equipo que contiene al fichero y especificar áreas alternativas de entrada/salida.

El formato de este párrafo es el siguiente:

Random = posicion física

8 **FILE-CONTROL. SELECT** [**OPTIONAL**] nombre-de-fichero-1

12 **ASSIGN** *TO* [entero-1] nombre-de-dispositivo-1 { TAPE / DISK / PRINTER }

[, nombre-de-dispositivo-2 ...]

[*FOR* **MULTIPLE REEL**]

12 [**RESERVE** { entero-2 / **NO** } *ALTERNATE* { *AREA* / *AREAS* }]

12 [**ACCESS** *MODE IS* { **RANDOM** / **SEQUENTIAL** / DYNAMIC }] → Relativo, indexado. → cualquier fichero indexado

12 [**ORGANIZATION** *IS* { **INDEXED** / **RELATIVE** / **SEQUENTIAL** }]

[{ **ACTUAL** *KEY IS* nombre-de-datos-1 / **RELATIVE** *KEY IS* nombre-de-datos-2 }] → Relativo
variable
Record Key Is → Indexado → Dynamic

[**SYMBOLIC** *KEY IS* nombre-de-dato-3]

[**RECORD** *KEY IS* nombre-de-dato-4]

[**PRIORITY** *IS* prioridad].

[**SELECT** ...].

FILE STATUS Is variable. / para la condicion de error

Alternate Key - es cuando 2 registros tienen el mismo nombre uno de ellos hay de llevar un apellidoque se pone en el Alternate key

Select.

Las cláusulas pueden estar separadas entre sí por medio de comas, puntos y comas o por espacios en blanco. Después de la última notación debe haber un punto. Después de este último punto puede haber otra cláusula SELECT si es necesario. Con esta cláusula deben aparecer cada uno de los ficheros mencionados en la división de datos (DATA DIVISION).

La información sobre cada fichero debe comenzar con la cláusula SELECT y terminar con un punto.

Optional.

La palabra OPTIONAL se utiliza cuando el nombre-de-fichero se refiere a un fichero secuencial de entrada, el cual puede estar o no presente siempre que se ejecute el programa objeto.

Assign.

La cláusula ASSIGN especifica el dispositivo de entrada o salida al que se asigna un fichero, utilizando los nombres de equipo especificados por el proveedor. Así los nombres especificados por el proveedor a los distintos dispositivos podrían ser, por ejemplo:

Dispositivos de E/S	*Nombre del proveedor*
Lectora de fichas	CARD-READER
Perforadora de fichas	CARD-PUNCH
Impresora	PRINTER
Discos	DISCK
Cintas	TAPE

El formato de la cláusula ASSIGN con *entero-1* se emplea cuando el fichero se encuentra en cintas magnéticas y se le va a asignar más de una unidad de cinta.

Multiple reel.

La cláusula FOR MULTIPLE REEL se usa para cuando se dé el caso de que hubieran más carretes de cinta de un fichero que unidades de cinta asignadas a dicho fichero.

Reserve.

La cláusula RESERVE permite al programador modificar las áreas de entrada y salida asignadas por el proveedor. Esta cláusula indica el número de áreas de E/S que se desean añadir. Si se usa la opción *entero-2* entonces se reserva un área adicional. El *entero-2* debe ser un 1.

Si se emplea la opción NO, entonces no se reservan áreas adicionales.

En el caso de que se omita la cláusula RESERVE, se asigna un área de E/S adicional en los dispositivos tales como: lectora de fichas, perforadora de fichas, impresora, cintas y discos cuya organización sea secuencial. En los discos con organización indexada o relativa no se asignan áreas adicionales de E/S cuando se omite la cláusula RESERVE.

Access mode.

El *acceso* de un fichero significa la forma en que se leen y se escriben los registros que constituyen el fichero.

Pueden haber dos métodos de acceso:

— Acceso secuencial.
— Acceso al azar.

La cláusula ACCESS MODE especifica el modo en que se leen o escriben los registros de un fichero. Si se omite esta cláusula se supone que el acceso es secuencial.

Acceso secuencial.

En un fichero de acceso secuencial, la lectura/escritura de los registros se efectúa en serie, secuencialmente.

El Acceso secuencial puede ser utilizado en cualquier tipo de organización de fichero. La cláusula correspondiente es:

```
ACCESS MODE IS SEQUENTIAL
```

Acceso al azar.

Para acceder a un registro con este método de acceso se utiliza una *clave* (KEY) para indicar el registro deseado.

El acceso al azar puede utilizarse en ficheros de organización relativa o indexada.

La cláusula es:

```
ACCESS MODE IS RANDOM
```

Organization.

La cláusula ORGANIZATION especifica la organización del fichero.

La organización de un fichero especifica el formato y la estructura lógica del fichero. Pueden haber tres clases de organización de ficheros:

— Secuencial.
— Relativa.
— Indexada.

En el caso de que se omita esta cláusula significa que la organización del fichero es secuencial.

Organización secuencial.

La estructura lógica de un fichero de organización secuencial se caracteriza porque los registros se encuentran en orden, uno tras otro, secuencialmente. El orden o la secuencia en que se leen los registros es la misma en la que se escriben. La cláusula para indicar que el fichero es secuencial es:

```
ORGANIZATION IS SEQUENTIAL
```

Organización relativa.

En un fichero de organización relativa la posición de cada registro está relacionado con respecto al primer registro. Es decir, el registro 1 ocupa la primera posición física del fichero, el registro 2 la segunda posición, y así sucesivamente. De este modo, los registros pueden leerse o escribirse directamente indicando el número de registro correspondiente. Este tipo de control de fichero se denomina *acceso al azar* (RANDOM ACCESS).

Así por ejemplo, en un fichero con organización relativa se puede acceder directamente al registro número 26 sin necesidad de tratar primero los 25 anteriores como ocurre en los ficheros de organización secuencial.

La cláusula correspondiente a esta organización es:

```
ORGANIZATION IS RELATIVE
```

Organización indexada.

Los ficheros con organización indexada contienen un área para los datos y un *índice* que permite acceder a los registros del área de datos. Cada registro del fichero contiene un campo llamado *clave*.

La *clave* es el control que emplea el método de acceso en la grabación del fichero y en su posterior lectura. El método de acceso utiliza una búsqueda del índice para localizar la dirección del registro que contiene la clave indicada.

A estos ficheros se les llama también de organización secuencial indexada, ya que el método de acceso requiere que los ficheros indexados se creen en secuencia de clave.

La cláusula que indica que el fichero es de organización secuencial indexada es:

```
ORGANIZATION IS INDEXED
```

Actual Key, Relative Key.

Las cláusulas ACTUAL KEY *(clave actual)* y RELATIVE KEY *(clave relativa)* especifican el nombre-de-dato que contiene el número relativo del registro al que se va a acceder.

Estas cláusulas se usan en organizaciones relativas de ficheros, para proporcionar la posición física de un registro con respecto al comienzo del fichero.

En una organización relativa, los registros del fichero se direccionan como registro número 1, 2, 3, etc. El nombre-de-dato debe definirse como un número entero.

Symbolic Key.

La cláusula SYMBOLIC KEY *(clave simbólica)* se emplea cuando la organización del fichero es indexada con el fin de proporcionar la identificación de registro para la recuperación aleatoria y el posicionamiento secuencial.

Record Key.

Esta cláusula *(clave de registro)* se utiliza con organizaciones indexadas para proporcionar el campo de identificación de registro.

Priority.

Esta cláusula se utiliza en sistemas que pueden procesar más de un programa en COBOL al mismo tiempo. Su significado lo especifica el proveedor.

Ejemplos:

```
SELECT MOVIMIENTOS ASSIGN TO CARD-READER.

SELECT FICHERO-MAESTRO ASSIGN TO 2 TAPES FOR MULTIPLE REEL.

SELECT INFORME ASSIGN TO PRINTER.

SELECT FICH-1 ASSIGN TO CARD-READER, DISC.

SELECT FICHERO ASSIGN TO SYS004 RESERVE NO ALTERNATE AREAS.
```

I-O-Control.

El párrafo I-O-CONTROL sirve para describir una serie de acciones no relacionadas entre sí.

Especifica:

— Las técnicas de E/S.
— El área de la MC compartida por varios ficheros.
— La situación de cada fichero en un carrete de cinta que contiene varios ficheros.
— Los intervalos en los que se van a establecer los puntos de reinicio.

El formato del párrafo I-O-CONTROL es:

I-O-CONTROL. [APPLY técnica de entrada-salida ***ON*** nombre-de-fichero-1
[***APPLY***...]]
[***RERUN*** [***ON*** nombre-de-dispositivo] ***EVERY*** entero-1
RECORDS ***OF*** nombre-de-fichero-2 [, nombre-de-fichero-3] ...]
[***SAME*** [***RECORD***] ***AREA*** ***FOR*** nombre-de-fichero-4, nombre-de-fichero-5...].

Las anotaciones de este párrafo pueden estar separadas entre sí por una coma o punto y coma.

Después de la última notación debe ponerse un punto.

Apply

Una de las funciones de esta cláusula consiste en definir un *nombre-de-condición* que se utilizará en una interrogación de FORM-OVERFLOW del informe impreso, dentro de la PROCEDURE DIVISION.

La interrogación de OVERFLOW determina si los formularios de una impresora han llegado al final de página.

Esta cláusula sólo se requiere para imprimir ficheros de salida cuya impresión sea discontinua por saltar de hoja en hoja.

Su formato es:

APPLY nombre-de-overflow *TO* ***FORM-OVERFLOW ON*** nombre-de-fichero.

Otra de las funciones consiste en especificar técnicas especiales de E/S para los ficheros.

La cláusula APPLY puede utilizarse cuando hay más de una técnica de entrada y salida disponible. Estas técnicas que se utilizarán deben ser definidas por el proveedor.

Rerun

Esta cláusula indica que se van a grabar registros de *punto de reinicio* en la unidad de cintas o discos especificada por el *nombre-de-dispositivo.*

El *punto de reinicio* es aquel en el que se desea poder reiniciar o relanzar la ejecución del programa, en el caso de que tenga que suspenderse la ejecución del programa a causa de un error de la máquina o del operador, por la llegada de un programa con mayor prioridad o por otros motivos. La información necesaria para reanudar el programa en ese punto está contenida en el registro de punto de reinicio. Estos registros se grabarán cuando se produzca *entero-1* RECORDS para nombre-de-fichero-2.

Una de las operaciones que puede efectuarse en un punto de reinicio consiste en vaciar el contenido de la memoria central a una cinta y, mediante un programa especial de reinicio, proporcionar la información suficiente para comenzar de nuevo a partir del punto del vaciado. Mediante las opciones de la cláusula RERUN puede escogerse el lugar donde se efectúa el vaciado de la memoria. Una de las técnicas más empleadas consiste en vaciar la memoria sobre el final de cada cinta de salida; otra técnica es la de utilizar una cinta para contener únicamente vaciados de memoria y producir un punto de reinicio al final de cada cinta de entrada.

Por último, la cláusula RERUN permite efectuar un vaciado de memoria al final de un cierto número de registros de cualquier fichero, al final de un cierto intervalo de tiempo o cuando se produzca alguna condición especial.

El *nombre-de-dispositivo* que aparece en el formato de la cláusula RERUN debe aparecer también en una proposición SELECT del párrafo FILE-CONTROL.

El dispositivo especificado por *nombre-de-dispositivo* recibe los registros de punto de reinicio que contienen la información sobre el estado del programa COBOL.

Nombre-de-fichero-2, nombre-de-fichero-3, etc., que aparecen en el formato de la cláusula RERUN, determinan cuándo deben grabarse los registros de reinicio.

El *nombre-de-dispositivo* puede aparecer en varias cláusulas RERUN si es necesario. Un *nombre-de-fichero* determinado sólo puede aparecer en una cláusula RERUN.

Si se especifican el acceso y la organización (en FILE-CONTROL) para un dispositivo, especificado en la cláusula RERUN por *nombre-de-dispositivo*, tanto el acceso como la organización deben ser secuenciales.

Si el dispositivo es una unidad de discos, el fichero correspondiente se destina a recibir registros de punto de reinicio.

Si el dispositivo es una unidad de cintas, el fichero correspondiente puede escribir registros de punto de reinicio y también puede recibir otra salida de programa.

Si se producen errores en estos ficheros se visualizan mensajes de error y continúa el proceso.

Same area

Esta cláusula especifica que dos o más ficheros comparten la misma zona o área de memoria durante el proceso.

Si se omite la palabra RECORD, el área que se va a compartir incluye todas las áreas de memoria asignadas a los ficheros. Por lo tanto, sólo puede estar abierto un fichero en un instante determinado.

Si se especifica la palabra RECORD, cualquier número de ficheros puede utilizar la misma área de memoria para procesar el registro actual.

EUROPE COMPANY CORPORATION

HOJA DE CODIFICACION

PROGRAMADOR	PROGRAMA	N.º	FECHA	HOJA

```
1       ENVIRONMENT DIVISION.
2       CONFIGURATION SECTION.
3       SOURCE-COMPUTER. CS-60.
4       OBJECT-COMPUTER. CS-60.
5     - SPECIAL-NAMES.
6                DECIMAL-POINT IS COMA
7                SW1 ON STATUS IS ENCENDIDO
8                   OFF STATUS IS APAGADO
9       INPUT-OUTPUT SECTION.
10      FILE-CONTROL.
11               SELECT FICHERO-ALUMNOS
12                 ASSIGN TO DISK
13               ORGANIZATION IS INDEXED.
14               ACCES IS DYNAMIC
15               FILE STATUS IS FALLO
16               RECORD KEY IS CLAVE
17               SELECT IMPRESORA
18                          ASSIGN TO PRINTER
19                          ORGANIZATION IS SEQUENTIAL
20                          ACCES IS SEQUENTIAL
21                          FILE STATUS IS F-I.
22
23
24
25
26
27
```

forleva, s. a. O-5-13.642

EJERCICIOS

1. En la IDENTIFICATION DIVISION es obligatorio codificar además del nombre de la división:

   a) AUTHOR.
   b) PROGRAM-ID.
   c) DATE-COMPILED.

2. El párrafo SOURCE-COMPUTER contiene:

   a) El nombre del ordenador objeto.
   b) El nombre del ordenador fuente.
   c) El nombre del programa fuente.

3. La INPUT-OUTPUT SECTION permite indicar

   a) Las características de los ordenadores fuente y objeto y nombres especiales.
   b) Nombres mnemotécnicos para los diferentes dispositivos del equipo.
   c) Los dispositivos de E/S de los ficheros utilizados por el programa fuente.

4. La cláusula ASSIGN:

   a) Especifica el dispositivo de E/S al que se asigna un fichero.
   b) Permite modificar las áreas de E/S asignadas por el proveedor.
   c) Permite asignar un área adicional de E/S en los discos y cintas magnéticas.

5. La cláusula ACTUAL KEY se utiliza:

   a) En organizaciones relativas.
   b) En organizaciones indexadas.
   c) En organizaciones secuenciales.

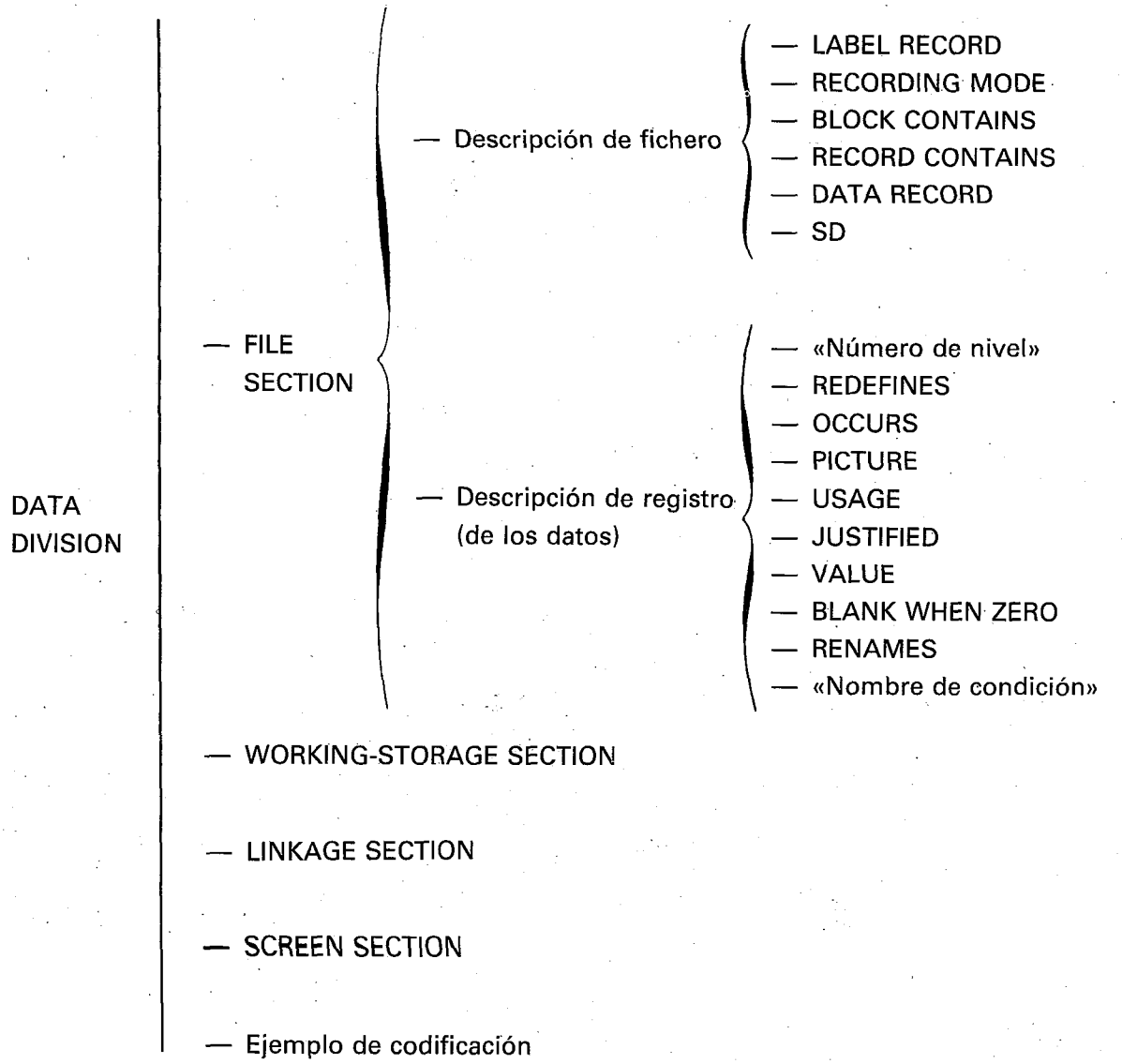
6. La cláusula que indica que se va a grabar registros de punto de reinicio es:

   a) APPLY
   b) RERUN
   c) SAME AREA

3.

DATA DIVISION

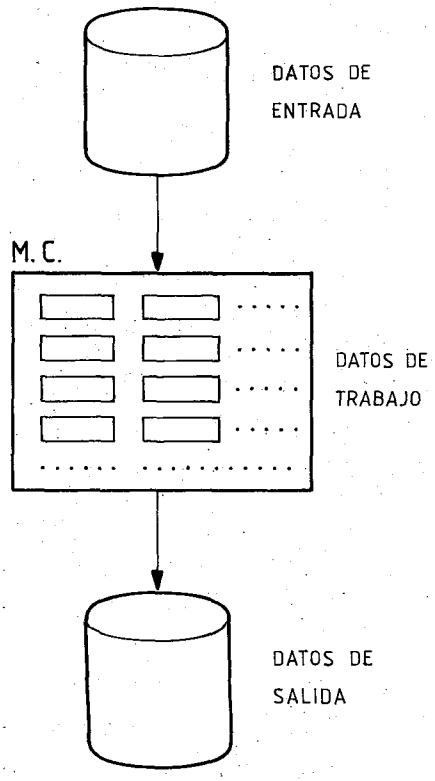
CUADRO RESUMEN DE LA LECCION



3. DATA DIVISION

La DATA DIVISION *(división de datos)* sirve para indicarle al ordenador los datos de entrada que se utilizan en el programa, así como el formato de los resultados que se desean obtener.

En esta división se describen los datos de entrada y los de salida.



En la DATA DIVISION se describen todos los datos que intervienen en el proceso

Todos los campos de datos referenciados en la PROCEDURE DIVISION. *(división de procedimiento)* deben describirse en la DATA DIVISION, excepto algunos campos, como son las constantes figurativas o los literales.

Las estructuras de los ficheros se describen mediante anotaciones de descripción de fichero. Los campos de datos y los registros se describen mediante anotaciones de descripción de registro o de campo elemental.

El formato general de la DATA DIVISION es:

```
DATA DIVISION.
[FILE SECTION.]

[WORKING-STORAGE SECTION.]

[LINKAGE SECTION.]

[SCREEN SECTION.]
```

El nombre DATA DIVISION debe aparecer en todos los programas COBOL.

Las secciones son opcionales, es decir, pueden omitirse siempre que no sean necesarias. En caso de que figuren estas secciones deben describirse en el orden indicado en el formato.

3.1. FILE SECTION:

La *sección de ficheros* (FILE SECTION) consta de anotaciones de *descripción de fichero* (File Description, o abreviado, FD) que describen la estructura de los archivos utilizados y dan nombre a los registros pertenecientes a cada uno de los ficheros. Esta sección consta además de anotaciones de *descripción de*

registro, que van inmediatamente a continuación de cada anotación FD y describen detalladamente cada uno de los formatos de registro empleados en el fichero.

Así pues, el formato de la FILE-SECTION consta de anotaciones de:

— Descripción de fichero.
— Descripción de registro.

3.1.1. Descripción de Fichero

Las anotaciones de descripción de fichero (FD) proporcionan información sobre la estructura física, las etiquetas y los nombres de registro de un determinado fichero.

El formato es:

```
FD nombre-de fichero

     LABEL { RECORD IS  }   { OMITTED  }
           { RECORDS ARE}   { STANDARD }

   [ RECORDING MODE IS { F } ]
                       { U }
                       { V }

   [ BLOCK CONTAINS [entero-1 TO] entero-2 { CHARACTERS } ]
                                           { RECORDS    }

   [ RECORD CONTAINS [entero-3 TO] entero-4 CHARACTERS ].

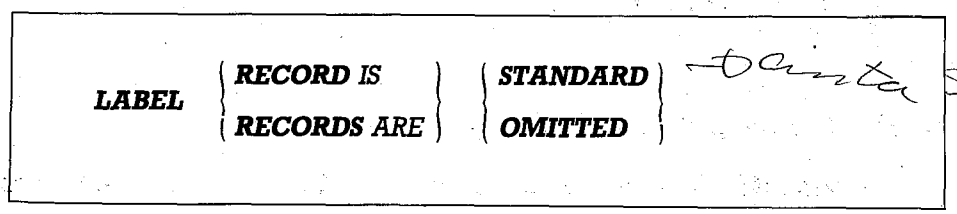
   [ DATA { RECORD IS  } nombre-de-dato-1 [nombre-de-dato-2] ...]. ]
          { RECORDS ARE}
```

En este formato, las cláusulas que siguen a *nombre-de-fichero* pueden aparecer en cualquier orden.

Cláusula Label Record

La cláusula LABEL RECORD *(registro de etiqueta)* permite al compilador relacionar la descripción de un registro de etiqueta con un fichero asociado.

El formato de esta cláusula es:



La cláusula STANDARD indica que existen etiquetas estándar para el fichero.

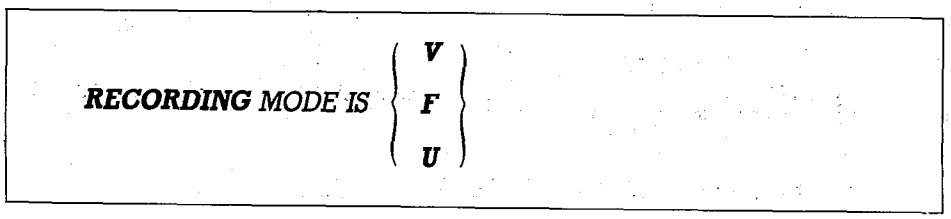
Las etiquetas estándar las proporciona el sistema de proceso de datos utilizado, aunque también pueden haber etiquetas estándar del usuario. La cláusula STANDARD indica que no se van a comprobar las etiquetas en los ficheros de entrada, ni se van a grabar en los ficheros de salida.

La cláusula OMITTED indica que en el fichero no existen etiquetas estándar. Estas etiquetas no estándar deben describirse y procesarse como registros de datos.

Cláusula Recording Mode

La cláusula RECORDING MODE *(modo de grabación)* especifica el formato de los registros que constituyen el fichero.

El formato es:



El modo de grabación V se refiere a ficheros con registros de *longitud variable.*

El modo de grabación de los datos F se utiliza cuando todos los registros del fichero tienen la misma longitud, es decir, cuando el formato de los registros es de *longitud fija.*

El modo de grabación U se refiere a que los registros del fichero no están bloqueados y que su longitud puede variar. Es decir, se utiliza este modo de grabación cuando el formato de los registros *no está definido.* Este modo de grabación no puede utilizarse para ficheros en disco.

Si no se especifica la cláusula RECORDING MODE, se supone que los distintos dispositivos tienen los siguientes formatos:

Dispositivo	*Formato*
Lectora de fichas	F
Perforadora de fichas	F
Impresora	F
Cintas	V
Discos con organización secuencial	V
Discos con organización relativa	F
Discos con organización indexada	F

Cláusula Block Contains

La cláusula BLOCK CONTAINS *(el bloque contiene) especifica el tamaño del bloque (registro físico).*

El formato es:

```
BLOCK CONTAINS [entero-1 TO] entero-2 { CHARACTERS }
                                      { RECORDS    }
```

Si se utiliza la opción CHARACTERS, esta cláusula indica el número de caracteres por bloque.

Si se utiliza la opción RECORDS y el modo de grabación es F, esta cláusula indica el número de registros por bloque.

Ahora bien, si el modo de grabación es V, esta cláusula especifica el número de registros de mayor tamaño por bloque.

Cláusula Record Contains

La cláusula RECORD CONTAINS *(el registro contiene)* indica el tamaño (número de caracteres) de los registros de datos.

El formato es:

```
RECORD CONTAINS [entero-1 TO] entero-2 CHARACTERS
```

Entero-2 debe ser mayor que *entero-1,* y ambos deben ser números enteros sin signo y distintos de cero.

Si se utilizan *entero-1* y *entero-2,* se refieren respectivamente a la longitud mínima y máxima de los registros.

Si sólo se utiliza *entero-2,* todos los registros del fichero deben ser del mismo tamaño. En este caso, *entero-2* indica el número de caracteres de los registros del fichero.

La cláusula RECORD CONTAINS es opcional, puesto que cada uno de los registros del fichero están definidos en la *descripción de registro.*

Cláusula Data Record

La cláusula DATA RECORD *(registro de datos)* indica los nombres de los registros de un fichero.

Esta cláusula es opcional y su formato es:

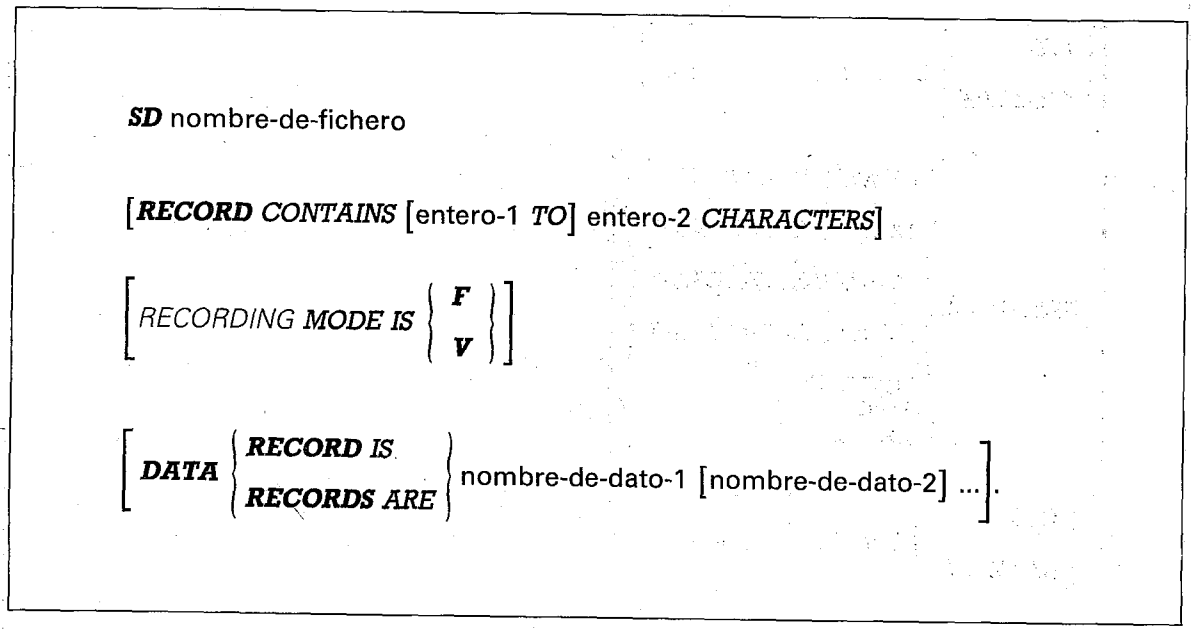
```
DATA { RECORD IS  } nombre-de-dato-1 [nombre-de-dato-2]...
     { RECORDS ARE }
```

Cada nombre-de-dato especificado debe aparecer en un número de nivel 01 que siga a una anotación FD.

Descripción de Fichero de clasificación (SD)

La cláusula SD (Sort Description o *descripción de clasificación*) identifica el comienzo de una descripción de un fichero de clasificación y proporciona el nombre del fichero.

Formato:



Se necesita una cláusula SD para cada fichero que se vaya a clasificar.

Los nombres de dato especificados deben aparecer con números de nivel 01 a continuación de una anotación SD.

Si se omite RECORDING MODE se supone que el modo de grabación es V.

3.1.2. Descripción de los datos

En la *descripción de datos de registro* se definen las características de un determinado campo de datos. Pueden darse tres tipos de formatos:

Formato 1:

número-de-nivel { ***FILLER*** / nombre-de-dato-1 }

[***REDEFINES*** nombre-de-dato-2]

[***OCCURS*** entero-2 *TIMES*]

[***OCCURS*** [entero-1 *TO*] entero-2 *TIMES* ***DEPENDING*** *ON* nombre-de-dato-1]

[{ ***PIC*** / ***PICTURE*** } *IS* serie-de-caracteres]

[[***USAGE*** *IS*] { ***COMPUTATIONAL*** / ***COMPUTATIONAL-1*** / ***COMPUTATIONAL-2*** / ***COMPUTATIONAL-3*** / ***DISPLAY*** / ***INDEX*** }]

[{ ***JUST*** / ***JUSTIFIED*** } ***RIGHT***]

[***VALUE*** *IS* literal]

[***BLANK*** *WHEN* ***ZERO***]

Formato 2:

66 nombre-de-dato-1 ***RENAMES*** nombre-de-dato-2 [***THRU*** nombre-de-datos-3]

Formato 3:

```
88 nombre-de-condición { VALUE IS  } literal-1 [THRU literal-2]
                       { VALUES ARE }

                       [literal-3 [THRU literal-4]]...
```

Cláusula «número-de-nivel»

El *número de nivel* se utiliza principalmente para indicar la jerarquía de los datos dentro de un registro.

El formato es:

```
número-de-nivel { FILLER          }
                { nombre-de-dato  }
```

Para cada anotación de descripción de datos es obligatorio especificar un número de nivel como primer elemento de dicha anotación.

Los números de nivel empiezan en 01 para registros y se incrementan progresivamente para los campos de los registros. El número de nivel más alto permitido es 49. Es decir, los números de nivel permitidos son: 01, 02, 03, 04, ..., 49.

El número de nivel 01 identifica la primera anotación de cada descripción de registro.

Existen otros números de nivel especiales. Estos son: 66, 77 y 88.

El número de nivel 66 se utiliza únicamente para la cláusula RENAMES.

El número de nivel 77 se utiliza en la WORKING-STORAGE SECTION para describir campos de datos y constantes no contiguas.

El número de nivel 88 se utiliza para anotaciones que definen nombres de condición asociados a una variable condicional.

La opción FILLER se utiliza para dar nombre a un campo elemental de un registro.

La palabra FILLER no puede utilizarse con un número de nivel 88.

Cláusula Redefines

La cláusula REDEFINES *(define de nuevo)* permite definir el mismo área de memoria central mediante diferentes descripciones de datos.

El formato es:

```
número-de-nivel nombre-de-dato-1 REDEFINES nombre-de-dato-2
```

Los números de nivel de nombre-de-dato-1 y nombre-de-dato-2 deben ser iguales y no pueden ser 66 ni 88.

La cláusula REDEFINES debe ir inmediatamente a continuación de nombre-de-dato-1.

Comienza a definirse de nuevo (redefinirse) a partir de nombre-de-dato-2 y continúa hasta encontrar un número de nivel menor o igual al de nombre-de-dato-2.

Cuando el número de nivel que se redefine no es 01, nombre-de-dato-1 debe especificar un área de memoria del mismo tamaño que nombre-de-dato-2.

El nombre-de-dato-2 no puede contener cláusulas OCCURS ni VALUE.

Todas las redefiniciones del mismo área de memoria deben utilizar el nombre-de-dato de la anotación que definió la primera vez el área.

La cláusula REDEFINES no puede aplicarse a anotaciones de nivel 01 de la FILE SECTION o LINKAGE SECTION, pero sí está permitida en la WORKING-STORAGE SECTION.

Cláusula Occurs

La cláusula OCCURS *(ocurre)* indica la repetición de un campo. Esta cláusula permite eliminar anotaciones diferentes para datos repetidos y proporciona

la información necesaria para la aplicación de índices o subíndices. El formato puede ser de dos tipos:

Formato 1:

OCCURS entero *TIMES* [**ASCENDING** / **DESCENDING**] *KEY IS* nombre-de-dato

[**INDEXED** *BY* nombre-de-índice]

La cláusula especifica el número de veces que se repite el campo. Por ejemplo:

OCCURS 50 TIMES

o bien solamente

OCCURS 50

indica que el campo a que hace referencia se repite 50 veces.

En este formato de la cláusula OCCURS, el valor de *entero* representa el número de repeticiones.

La cláusula OCCURS se utiliza para definir grupos de campos idénticos o tablas de datos, que almacenan datos relacionados entre sí.

Las cláusulas de descripción de datos, asociadas con un campo de datos cuya descripción incluye una cláusula OCCURS, se aplican a cada repetición del campo descrito.

La cláusula OCCURS no puede especificarse en una anotación de descripción de datos que tenga un número de nivel, 01, 66, 77 u 88.

Formato 2:

OCCURS [entero-1 **TO**] entero-2 *TIMES* **DEPENDING** *ON* nombre-de-dato-1

[**ASCENDING** / **DESCENDING**] *KEY IS* nombre-de-dato-2 [**INDEXED** *BY* nombre-de-índice]

Este formato indica que el campo tiene un valor variable de repeticiones. El valor de *entero-2* representa el número máximo de repeticiones y el valor de *entero-1* el número mínimo de repeticiones.

El valor de *nombre-de-dato* es la cuenta del número de repeticiones. Este valor debe ser un entero positivo y debe estar comprendido entre *entero-1* y *entero-2.*

La palabra clave DEPENDING sólo debe utilizarse cuando no se pueda determinar de otra forma el final de las repeticiones.

Definición de una tabla

A cada uno de los campos de datos de una tabla se le llama *elemento* de la tabla.

El elemento de una tabla debe ser el sujeto de una cláusula OCCURS en la descripción de datos. Esta cláusula indica el número de veces que aparece el elemento de la tabla.

Las tablas pueden ser de una, dos o tres dimensiones (es decir, tablas de uno, dos o tres índices).

Para definir una tabla de una dimensión, se escribe una cláusula OCCURS como parte de la descripción de datos del campo que debe repetirse.

Si se define una tabla de una dimensión dentro de cada repetición de un elemento de la tabla, se obtiene una tabla de dos dimensiones. Esto se consigue escribiendo una cláusula OCCURS como parte de la descripción de un campo de datos subordinado a otro campo (es decir, que tenga un número de nivel mayor) que también tenga una cláusula OCCURS. Así pueden definirse tablas de hasta tres dimensiones. Cada dimensión debe definirse mediante una cláusula OCCURS y con diferente nivel jerárquico.

Para hacer referencia a un elemento de la tabla es necesario especificar la repetición del elemento que se desea.

La referencia se efectúa poniendo a continuación del *nombre-de-dato* un conjunto de especificaciones relativas al número de repetición (subíndices o nombre-de-índice) encerrados entre paréntesis.

Los números de repetición se especifican mediante *subíndices* o mediante *índices.*

Los *subíndices* se utilizan para referenciar elementos individuales de una tabla de elementos homogéneos, a los que no se ha asignado nombres-de-dato individuales. Un valor de subíndice identifica los ítems elementales de la tabla.

Su formato es:

```
nombre-de-dato (subíndice-1 [subíndice-2 [subíndice-3] ] )
```

El subíndice más bajo debe ser 1 y el más alto debe ser el que representa el número especificado en la cláusula OCCURS.

Los *índices* se utilizan para hacer referencia a elementos individuales de una tabla de elementos homogéneos, a los que no se ha asignado nombre-de-datos individuales. Un nombre-de-índice contiene el número de repeticiones de un elemento de la tabla.

Su formato es:

```
nombre-de-dato (nombre-de-índice-1 [ {±} entero-1]
                [nombre-de-índice-2 [ {±} entero-2 ]]
                [nombre-de-índice-3 [ {±} entero-3 ]])
```

Los *nombres-de-índice* se definen utilizando la opción INDEXED BY en la cláusula OCCURS.

Los nombres-de-índice se inicializan y se modifican en el programa objeto mediante la proposición SET.

Para hacer referencia a campos individuales de una tabla de elementos homogéneos, se especifica el nombre del elemento de la tabla seguido de uno o varios nombres-de-índice relacionados entre paréntesis.

Un campo de datos de un fichero se describe mediante una cláusula USAGE IS INDEX.

Cláusula Picture

La cláusula PICTURE *(modelo)* describe las características generales de edición de un campo de datos elemental.

El formato es:

```
{ PIC     }
{ PICTURE } IS serie-de-caracteres
```

La *serie-de-caracteres* puede contener de 1 a 30 caracteres.

Con una cláusula PICTURE pueden definirse cinco tipos de datos:

— Alfabéticos.
— Numéricos.
— Alfanuméricos.
— Alfanuméricos editados.
— Numéricos editados.

Definición de un campo alfabético

La serie de caracteres de la cláusula PICTURE de un campo de datos alfabéticos sólo puede contener el carácter A. El contenido de esta serie de caracteres debe ser cualquier combinación de los 26 caracteres alfabéticos y el espacio en blanco.

Definición de un campo numérico

Los campos numéricos pueden ser de dos tipos: de punto fijo (o coma fija) y de punto flotante (o coma flotante).

Para definir un campo como numérico de *punto fijo* deben considerarse los siguientes aspectos:

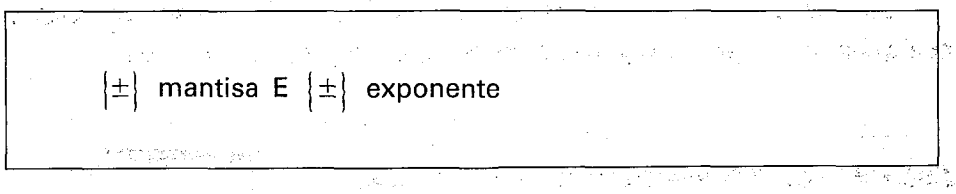
— La serie de caracteres de su PICTURE solo puede contener los símbolos 9, P, S, V y H.
— Su serie de caracteres debe contener por lo menos un 9.

— El número máximo de dígitos permitido es 18.
— El campo de datos puede incluir cualquier combinación de los dígitos 0 a 9 y un signo de operación.

Existen dos tipos de campos numéricos de *punto flotante:*

— DISPLAY
— De cálculo (COMP-1 o COMP-2).

La serie de caracteres de un *campo numérico DISPLAY* de punto flotante tiene el siguiente formato:



En la mantisa sólo pueden utilizarse ciertos símbolos. Estos son: el punto (.), el dígito 9 y la letra V.

El punto (.) se utiliza para representar un punto decimal real y la V para representar un punto decimal supuesto. En la mantisa debe aparecer uno u otro símbolo. La V no ocupa espacio en la memoria.

El dígito 9 puede repetirse en la mantisa hasta 16 veces.

La letra E indica el exponente del campo. Este carácter ocupa un octeto en la memoria.

Los signos empleados, tanto el que precede a la mantisa como el que precede al exponente, ocupan cada uno de ellos un octeto en la memoria.

El exponente se representa por dos símbolos 9.

Así pues, el valor de un campo numérico DISPLAY de punto flotante es igual al producto de la mantisa por la potencia de 10 representada por el exponente.

Para definir un campo como de *cálculo numérico* de punto flotante no se utiliza la cláusula PICTURE puesto que COMP-1 y COMP-2 especifican el formato de cálculo de punto flotante.

COMP-1 especifica un campo de punto flotante de simple precisión que ocupa 4 octetos.

COMP-2 especifica un campo de punto flotante de doble precisión que ocupa 8 octetos.

Los campos de punto flotante no se permiten cuando se requieren valores enteros.

Definición de un campo alfanumérico

La serie de caracteres de la cláusula PICTURE de un campo alfanumérico está formada por una o más letras X.

Los campos alfanuméricos se emplean para almacenar caracteres de cualquier tipo: letras, dígitos, caracteres especiales o espacios.

Definición de un campo alfanumérico editado

La serie de caracteres de la cláusula PICTURE de un campo alfanumérico editado está formada por combinaciones de los símbolos A, X, 9, B y 0. Esta serie de caracteres debe contener al menos:

— Una B y una X.
— Un 0 y una X.
— Un 0 y una A.
— Una A y una B.

Definición de un campo numérico editado

Su serie de caracteres está formada por combinaciones de los símbolos: B, P, V, Z, CR, DB, 9, coma (,), punto (.), *, +, −, 0 (cero) y $ (símbolo monetario).

El contenido de los campos numéricos editados debe constar de los dígitos 0 a 9 y los símbolos de edición indicados.

El número de posiciones de dígito no puede ser superior a 18.

Reglas generales de la cláusula Picture

1. Un número entre paréntesis a continuación de cualquiera de los símbolos A, X, 9, P, Z, *, B, 0, +, − y $ indica el número de repeticiones consecutivas de ese símbolo.

2. Los siguientes símbolos sólo pueden aparecer una vez en una PICTURE:

   S V CR DB E H

3. Un número entero entre paréntesis a continuación de uno de los símbolos A X 9 P Z * B 0 + − $ indica el número de repeticiones consecutivas de ese símbolo.
   Por ejemplo, X(12) indica que el símbolo X se repite 12 veces consecutivas.

El significado de estos símbolos se representa en la tabla siguiente:

Símbolo	*Representa*	*Reglas*
9	Un carácter numérico.	
S	Un signo de operación asociado a un campo de datos.	Sólo se permite una S en cada
V	Punto decimal supuesto en el campo de datos.	Sólo se permite una V en cada
P	Punto decimal supuesto fuera del campo de datos. Cada P representa una posición de carácter.	
A	Un carácter alfabético o espacio.	
X	Un carácter alfanumérico.	
Z	Supresión de ceros de cabecera (reemplazados por blancos o espacios)	
*	Protección de información numérica. Reemplaza ceros a la izquierda por asteriscos.	
(coma)	Inserción de una coma en la posición de carácter.	
(punto)	Inserción de un punto decimal real en la posición de carácter.	No puede ser el último carácter.
B	Inserción de un blanco o espacio en la posición de carácter.	

Símbolo	*Representa*	*Reglas*
CR	Inserción de los dos caracteres CR si el campo de datos es de valor negativo. Inserción de dos blancos si es positivo.	Debe ser el último símbolo, excepto P ó V.
DB	Inserción de los caracteres DB si el valor del campo es negativo. Si es positivo, inserción de dos blancos.	Debe ser el último símbolo, excepto P ó V.
$	Inserción del signo $ en la posición de carácter. Si hay más de uno, indica signo $ flotante.	
0 (cero)	Inserción de un cero en la posición de carácter.	
+ (más)	Inserción de un signo + en la posición de carácter si el valor del campo es positivo. En caso contrario, si es negativo se insertas un −. Si hay más de un signo + indica signo flotante.	
− (menos)	Inserción de − en la posición de carácter si el valor del campo es negativo. Si es positivo, inserta un blanco. Si hay más de un − indica signo flotante.	

La siguiente tabla presenta ejemplos de campos fuentes y campos receptores:

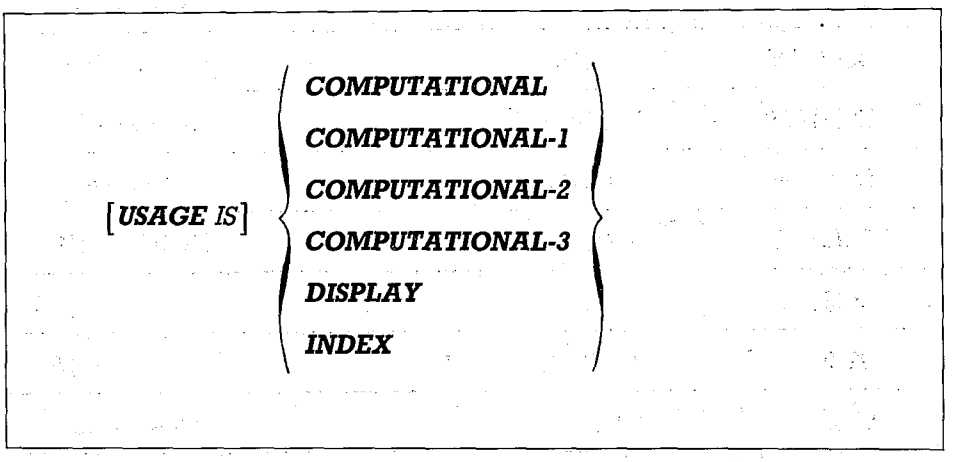
Campo fuente		*Campo receptor*	
Picture	*Datos a mover*	*Picture*	*Datos después de movidos*
9(3)	123	ZZ,ZZZ.99	123.00
9(5)V99	0045678	ZZ,ZZZ.99	456.78
9(4)V99	123456	$$,$$$.99	$1,234.56

Campo fuente		Campo receptor	
Picture	*Datos a mover*	*Picture*	*Datos después de movidos*
9(3)	012	$$,$$$.99	$12.00
S9(3)	+123	$$,$$$.99CR	$123.00
S9(3)V99	+12345	$$,$$$.99CR	$123.45
S9(3)V99	−00123	$$,$$$.99CR	$1.23 CR
S9(3)V99	−00123	$$,$$$.99DB	$1.23 DB
S9(3)	+012	$$,$$$.99DB	$12.00
S9(3)	+123	++,+++.00	+123.00
S9(5)	−67453	−−,−−−.99	−67,453.00
9999	0012	$✳✳✳✳.99	$✳✳12.00
9(3)	456	90909	40506
9(3)	456	9B9B9	4 5 6
A(4)	CDTR	ABAAB00A	C DT 00R
X(4)	D5F7	XBX0X0BX	D 50F0 7
999	345	9999	0345
9(6)	123456	9(4).99	3456.00
99V99	1234	9(5).99	00012.34
AAAA	CDEF	A(7)	CDEFbbb
A(7)	CDEFGHI	AA	CD
9(3)PP	123	9(5)	12300
VPPP9(3)	123	.9(6)	.000123
V9(3)	123	Z(3).99	bbb.12
V9(3)	123	9(3).999	000.123

Cláusula Usage

La cláusula USAGE *(uso)* indica el formato de un campo en la memoria central.

El formato de esta cláusula es:



La palabra COMPUTATIONAL de este formato puede sustituirse por COMP que tiene el mismo significado.

Esta cláusula puede utilizarse para campos elementales y para campos compuestos. El USAGE especificado para un campo compuesto se aplica a todos los campos que lo constituyen.

Las reglas de esta cláusula son las siguientes:

A)

La palabra USAGE o las palabras USAGE IS son opcionales. Así por ejemplo, las cláusulas siguientes son equivalentes:

```
USAGE IS COMP-3
USAGE COMP-3
COMP-3
```

Si se omite la cláusula USAGE, se supone que es DISPLAY.

B)

La opción USAGE IS DISPLAY indica que cada carácter del campo se almacena ocupando un octeto. Esta opción se utiliza para campos alfabéticos, alfa-

numéricos, alfanuméricos editados, numéricos editados, numéricos decimales DISPLAY y numéricos de punto flotante DISPLAY.

C)

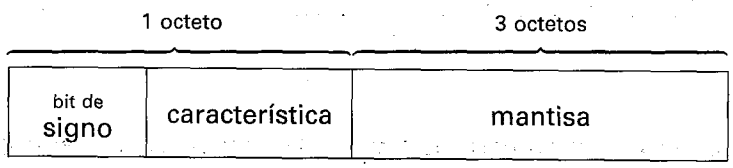
La opción COMPUTATIONAL especifica el formato binario. La cantidad de memoria asignada depende del número de dígitos de la PICTURE: si el número de dígitos 9 en la PICTURE es de 1 a 4 la memoria asignada será de 2 octetos. Si es de 5 a 9 la memoria asignada será 4 octetos. Y, por último, si es de 10 a 18 la memoria asignada será de 8 octetos.

D)

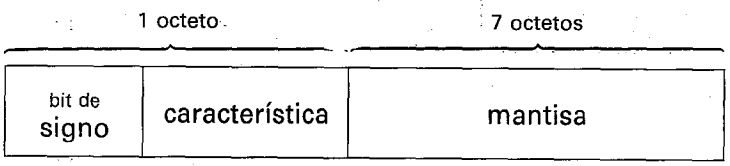
Las opciones COMPUTATIONAL-1 y COMPUTATIONAL-2 indican el formato de cálculo de punto flotante. COMPUTATIONAL-1 especifica un campo de punto flotante de simple precisión que ocupa 4 octetos Y COMPUTATIONAL-2 indica un campo de punto flotante de doble precisión que ocupa 8 octetos.

Estos formatos definen campos de datos que contienen valores demasiado grandes para representarlos en punto fijo.

En el formato de simple precisión, la mantisa aparece en los tres octetos de la derecha; la característica se encuentra en el octeto de la izquierda; y el signo de la mantisa se encuentra en el bit de más a la izquierda



En el formato de doble precisión, el signo de la mantisa y la característica ocupan las mismas posiciones que en el formato de simple precisión. En este caso, la mantisa se encuentra en los siete octetos restantes.



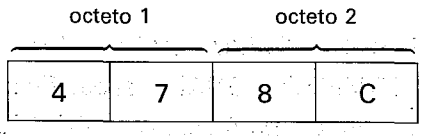
E)

Si el USAGE es COMPUTATIONAL-3, esta cláusula especifica el formato *decimal empaquetado.* En este formato, se almacenan dos dígitos decimales en

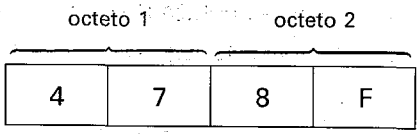
cada octeto, excepto en el octeto de la derecha en donde se almacena un dígito decimal y el signo del campo. Así pues, cada dígito decimal ocupa medio octeto.

Si no se especifica el signo para el campo, la posición del signo está ocupada por una serie de bits que se interpreta como positiva.

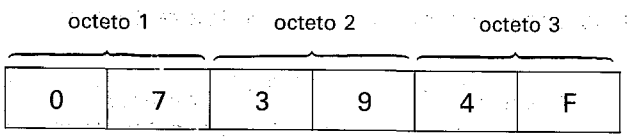
Si el número de dígitos del campo es impar, el primer dígito decimal del campo ocupará el medio octeto de más a la izquierda y el resto de dígitos a continuación del anterior. Por ejemplo, si el número es +478 se tiene:



Si no se especifica signo, es decir, si el número es 478, en la memoria central aparecerá:



Por otro lado, si el número de dígitos del campo es par, el primer medio octeto de la izquierda está ocupado por un cero. Por ejemplo, si el número es 7394, en la memoria central aparecerá:



F)

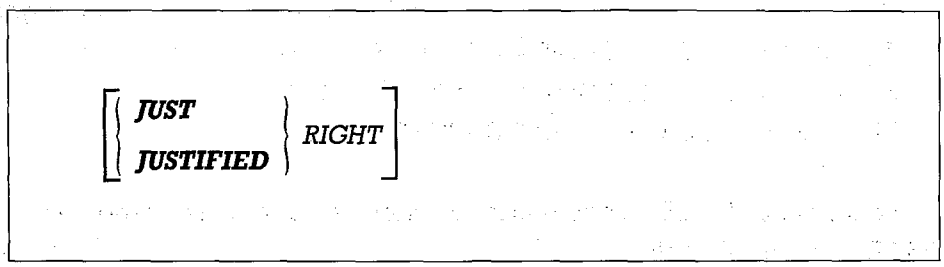
A un campo elemental descrito con la cláusula USAGE IS INDEX se le llama *campo índice* y contiene el valor que corresponde al número de repetición de un elemento de tabla.

La cláusula PICTURE no debe aparecer en este caso.

Cláusula Justified

La cláusula JUSTIFIED *(justificar)* indica el posicionamiento no standard de los datos dentro de un campo receptor.

Esta cláusula sólo puede especificarse a nivel de campo elemental. Su formato es:



La cláusula JUSTIFIED no se puede utilizar para datos numéricos, ya que éstos se alinean siempre por su punto decimal y, en este caso, dicho punto está presente. Por el contrario, si el punto decimal no está presente, se justifica a la derecha (RIGHT).

Si no se especifica la cláusula JUSTIFIED, los datos alfabéticos y alfanuméricos se justifican a la izquierda con relleno de espacios a la derecha.

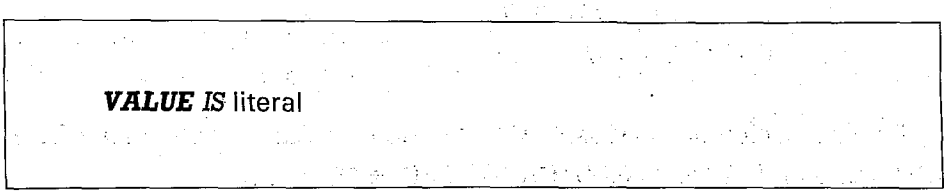
Cuando el campo receptor se describe con la cláusula JUSTIFIED y el campo emisor es mayor, los caracteres de la izquierda se truncan.

Cuando el campo receptor está justificado y el campo emisor es menor, los datos se alinean por la derecha con relleno de espacios a la izquierda.

Cláusula Value

La cláusula VALUE *(valor)* puede tener dos formatos diferentes:

Formato 1



Este formato se utiliza para definir el valor inicial de un campo de datos de la WORKING-STORAGE SECTION.

Mediante la cláusula VALUE el campo toma el valor especificado al comienzo del programa objeto. Esta cláusula puede utilizarse para especificar el valor inicial o constante de cualquier anotación de número-de-nivel 77.

Ejemplos:

```
77  CAMPO-1 USAGE COMP PIC 9 VALUE 0.
77  CAMPO-3 USAGE COMP PIC 999 VALUE 123.
77  CAMPO-3 USAGE COMP PIC 99V99 VALUE 25.50.
```

La cláusula VALUE puede utilizar constantes figurativas, como son ZERO SPACE, etc. Por ejemplo:

```
77  CAMPO-4 USAGE COMP PIC 99 VALUE ZEROS.
```

Formato 2:

```
{ VALUE IS  }
{ VALUE ARE } literal-1 [ THRU literal-2 ] ... [ literal-3 [ THRU literal-4 ]]
```

Este formato sólo se utiliza en una anotación de *nombre-de-condición*. Esta cláusula especifica el valor asociado con el nombre-de-condición.

Todas las anotaciones de nombre-de-condición tienen el número de nivel 88.

Ejemplo:

```
02  CONDICIONES PICTURE X.
    88  CONDICION-1 VALUE '4'.
    88  CONDICION-2 VALUE '6'.
    88  CONDICION-3 VALUE '8'.
```

Esta aplicación de la cláusula VALUE puede utilizarse tanto en la FILE SECTION como en la WORKING-STORAGE SECTION.

Cláusula Blank When Zero

Esta cláusula pone el valor de un campo receptor a blancos cuando el valor del campo emisor es cero.

El formato es:

```
BLANK WHEN ZERO.
```

Esta cláusula sólo puede utilizarse con un campo elemental numérico o numérico editado. Cuando se emplea con un campo numérico, se considera que es numérico editado.

Cláusula Renames

Esta cláusula permite agrupaciones alternativas de campos elementales. Su formato es:

```
66  nombre-de-dato-1 RENAMES nombre-de-dato-2 [THRU nombre-de-dato-3]
```

Cláusula Nombre-de-condición

Esta cláusula asigna un nombre a uno o varios valores. Su formato es:

```
88  nombre-de-condición { VALUE IS  } literal-1 [THRU literal-2] [literal-3 [THRU literal-4]] ...
                        { VALUES ARE }
```

Si, por ejemplo, se trata de un campo elemental puede escribirse:

```
02  nombre-de-dato PIC XXX.
    88  nombre-de-condición VALUE 'LMN'.
```

Si se trata de un campo de varios valores se puede escribir, por ejemplo:

```
03  nombre-de-dato.
    88  nombre-de-condición VALUE '15' THRU '30'.
```

3.2. WORKING-STORAGE SECTION

Esta sección de la DATA DIVISION se emplea para describir las áreas de memoria destinadas a contener los resultados parciales del proceso, constantes con nombre y otros datos que se almacenan temporalmente durante la ejecución del programa.

La WORKING-STORAGE SECTION («sección de áreas de trabajo») tiene el siguiente formato:

WORKING-STORAGE SECTION.

[77 descripción de campo independiente]

[01 descripción de registro]

[88 anotación de nombre-de-condición]

El *número de nivel 77* se asigna solamente a áreas de campos elementales (o independientes). Su formato es:

77 nombre-de-dato ***PICTURE IS*** serie-de-caracteres

Las anotaciones de número de nivel 77 deben encontrarse al comienzo de la WORKING-STORAGE SECTION.

El *número de nivel 01* se utiliza para describir áreas contiguas de datos que no forman parte de un fichero. Su formato es:

```
01  nombre-de-registro
    (cláusulas y campos subordinados)
```

Los campos subordinados son los de nivel 02, 03, etc.

3.3. LINKAGE SECTION

Esta sección de la DATA DIVISION se utiliza para describir datos que están disponibles en un programa.

Esta sección es necesaria en cualquier programa que contenga una proposición ENTRY con una opción USING o la proposición USING de la PROCEDURE DIVISION para un programa llamado.

La transferencia de control desde un programa que llama a uno llamado se efectúa por medio de una proposición CALL *(llamar)* con un nombre-de-entrada idéntico al de la proposición ENTRY del programa llamado.

El control se devuelve al programa que llama (o programa principal) ejecutándose una proposición EXIT PROGRAM en el programa llamado (o subprograma).

Los campos de datos del programa principal y a los que se hace referencia en el subprograma se describen en la FILE SECTION o en la WORKING-STORAGE SECTION del programa principal. Estos campos se definen también en la LINKAGE SECTION del subprograma.

El compilador no asigna memoria a los campos descritos en la LINKAGE SECTION, puesto que ya ocupan memoria en el programa principal, el cual proporciona sus direcciones al subprograma en la fase de ejecución.

3.4. SCREEN SECTION

La *Screen Section* (Sección de Pantalla) se desarrollará en el tomo 8 ya que su utilización pertenece a técnicas avanzadas de la programación en Cobol.

Esta sección sirve para mantener una comunicación constante entre el programa en ejecución y el operador.

▶ ***Ejemplo de codificación***

EUROPE COMPANY CORPORATION

HOJA DE CODIFICACION

PROGRAMADOR	PROGRAMA	N.º	FECHA	HOJA

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1       DATA DIVISION.
2       FILE SECTION.
3       FD  ALUMNOS
4           LABEL RECORD IS OMITTED
5           DATA RECORD IS DATOS ALUMNO.
6       01  DATOS ALUMNO.
7           02 PERSONALES.
8              03 A/O-INSCRIPCION PIC XX.
9              03 NUM-ALUMNO PIC X(9).
10             03 FILLER PIC X(30).
11          02 ESTUDIOS PIC X(14).
12      FD  ACTUALIZACION
13          LABEL RECORD IS OMITTED
14          DATA RECORD IS TRANSFERENCIA.
15      01  TRANSFERENCIA.
16          02 DENTRO-O-FUERA PIC XX.
17          02 NUM-FICHA PIC X(9).
18          02 FILLER PIC X(64).
19      WORKING-STORAGE SECTION.
20      77  NUM-CLAVE PIC X(9).
21      01  TABLA.
22          02 DATO-1 PIC XX.
23          02 DATO-2.
24             03 DATO-2-1 PIC 9(5).
25             03 DATO-2-2 PIC X(7).
26
27
```

forteva, s.a. O-5-13.642

EJERCICIOS

1. La descripción de los ficheros pertenece a la sección de la DATA DIVISION:

   a) FILE SECTION.
   b) WORKING-STORAGE SECTION.
   c) LINKAGE SECTION.

2. La cláusula RECORDING MODE:

   a) Permite al compilador relacionar la descripción de un registro de etiqueta con un fichero asociado.
   b) Especifica el formato de los registros que constituyen al fichero.
   c) Especifica el tamaño del registro físico.

3. La cláusula que especifica el número de los registros de datos es:

   a) BLOCK CONTAINS.
   b) RECORD CONTAINS.
   c) LABEL RECORD.

4. El número de nivel o indicador de nivel se utiliza para indicar la jerarquía:

   a) De los registros dentro de un fichero.
   b) De los registros dentro de un bloque.
   c) De los datos dentro de un registro.

5. El número de nivel 77 se utiliza para describir campos de datos y constantes no contiguas en la:

   a) FILE SECTION.
   b) WORKING-STORAGE SECTION.
   c) LINKAGE SECTION.

6. La cláusula REDEFINES:

   a) Permite definir el mismo área de la MC mediante diferentes descripciones de datos.
   b) Permite dar nombre a un campo elemental de un registro.
   c) Indica la repetición de un campo.

7. Si se omite la cláusula USAGE se supone que es:

   a) COMPUTATIONAL.
   b) DISPLAY.
   c) INDEX.

8. Los nombres de condición tienen el número de nivel:

   a) 66
   b) 77
   c) 88

4.

PROCEDURE DIVISION

CUADRO RESUMEN DE LA LECCION

DIVISIONES DE UN PROGRAMA COBOL (III).

Procedure division

- Proposición USING.
- Secciones.
- Párrafos.
- Sentencias.
- Proposiciones.
- Verbos
  - Aritméticos.
  - De bifurcación.
  - De transferencia.
  - De E/S.
  - De final.
  - Condicionales.
  - Directivos del compilador.
  - Comunicaciones entre programas.

4. PROCEDURE DIVISION

La PROCEDURE DIVISION *(división de procedimiento)* contiene las instrucciones o pasos necesarios para resolver un problema determinado. Su formato es:

```
PROCEDURE DIVISION [USING nombre-de-dato-1 [nombre-de-dato-2] ...]

                   [nombre-de-sección SECTION [número-de-prioridad].]

                   {nombre-de-párrafo. {sentencia}...}...
```

4.1. PROPOSICION USING

Si la proposición USING *(uso)* va inmediatamente a continuación de la cabecera PROCEDURE DIVISION, actúa como declaración de punto de entrada y sólo puede aparecer en un subprograma llamado. Su formato es:

```
USING nombre-de-dato-1 [nombre-de-dato-2] ...
```

Los nombres de dato referencian a campos descritos en el subprograma. Sus números de nivel sólo pueden ser 01 o 77 y deben estar definidos en la LINKAGE SECTION.

4.2. SECCIONES:

La PROCEDURE DIVISION debe estar dividida en secciones, con sus correspondientes números de prioridad cuando el programa está segmentado.

El formato de cada sección es:

```
[nombre-de-sección SECTION [número-de-prioridad].]

{nombre-de-párrafo. {sentencia} ... }...
```

A cada sección se le puede asignar un nombre de procedimiento.

Una sección puede dividirse en cualquier número de párrafos.

4.3. PARRAFOS

Un *párrafo* es un conjunto de una o más frases de procedimiento que está encabezado por un nombre para identificarlo y referenciarlo en las transferencias.

Su formato es:

```
{nombre-de-párrafo. {sentencia }... }...
```

Un párrafo puede constar de una o varias sentencias.

4.4. SENTENCIAS

Una *sentencia* consta de una o más proposiciones y termina con un punto. Su formato es:

```
proposición-1 [proposición-2 ...].
```

4.5. PROPOSICIONES

Una *proposición* consta de un verbo, más las palabras reservadas y las palabras facilitadas por el usuario que sean necesarias.

Las proposiciones pueden ser de tres tipos:

— proposiciones imperativas
— proposiciones condicionales
— proposiciones directivas del compilador.

Las proposiciones *imperativas* indican acciones incondicionales específicas que tiene que realizar el programa objeto.

Las preposiciones *condicionales* indican que se va a determinar el valor verdadero de una condición y que la acción del programa objeto depende de este valor verdadero.

Por ejemplo, son condicionales las proposiciones IF *(sí condicional)* o los verbos de entrada/salida que especifiquen la opción AT END *(al final...)*.

Las proposiciones *directivas del compilador* dirigen el compilador para que realice ciertas acciones en la fase de compilación.

El formato de las proposiciones es:

```
verbo serie-de-palabras.
```

4.6. VERBOS

Un *verbo* es una palabra reservada que se emplea en la PROCEDURE DIVISION para especificar la acción que va a realizar el ordenador o el compilador.

Los diversos tipos de verbos son:

— *Aritméticos:* ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE.

— *De bifurcación:* GO TO, PERFORM, EXIT, ALTER.

— *De transferencia:* MOVE, EXAMINE, SET.

— *De entrada/salida:* ACCEPT, OPEN, CLOSE, READ, WRITE, REWRITE, RETURN, DISPLAY, INSERT, RELEASE, SORT, SEEK.

— *De parada:* STOP.

— *Condicionales:* IF, SEARCH.

— *Directivos del compilador:* COPY, ENTER, NOTE, USE.

— *Comunicación entre programas:* CALL, ENTRY.

4.6.1. Verbos aritméticos

Los verbos aritméticos permiten realizar cálculos con los datos.

Hay cinco verbos aritméticos:

— ADD.
— SUBTRACT.
— MULTIPLY.
— DIVIDE.
— COMPUTE.

Los cuatro primeros verbos corresponden a las cuatro operaciones aritméticas básicas: ADD *(suma)*, SUBTRACT *(resta)*, MULTIPLY *(multiplicación)* y DIVIDE *(división)*.

El verbo COMPUTE *(calcular)* permite realizar cálculos aritméticos utilizando expresiones aritméticas.

Todos los campos de datos referenciados en proposiciones aritméticas deben representar campos numéricos elementales definidos previamente en la DATA DIVISION. Además, todos los literales utilizados en estas proposiciones también deben ser numéricos. El tamaño máximo de los operandos es de 18 dígitos decimales, excepto los campos de punto flotante.

Las descripciones de datos (PICTURE) de los operandos pueden ser diferentes. La alineación por el punto decimal se realiza automáticamente.

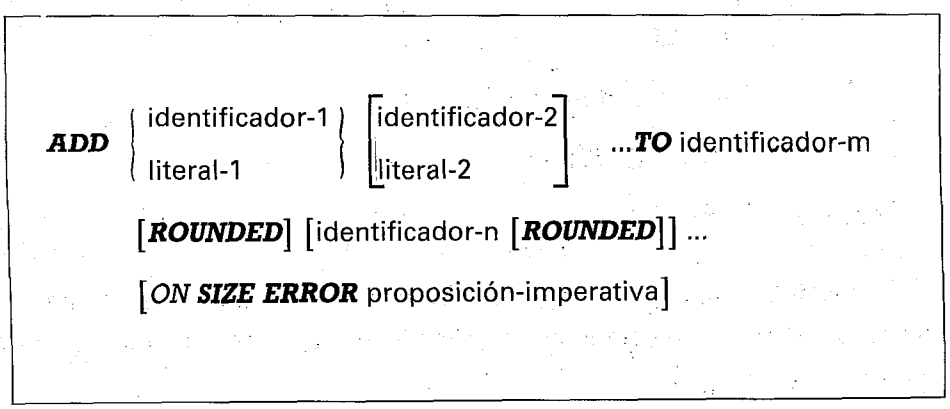
Si después de una alineación por el punto decimal, el número de posiciones de fracción del resultado de una operación decimal es mayor que el número de posiciones de fracciones del campo receptor del resultado, se produce una truncación. Con la opción ROUNDED *(redondeo)*, el valor absoluto del campo receptor se incrementa en uno si el dígito más significativo del exceso es igual o mayor que cinco.

Si después de la alineación por el punto decimal, el valor del resultado excede al valor máximo que puede contener el campo receptor, se produce una condición de *error de tamaño*. En este caso, la acción a realizar depende de si se ha especificado la opción ON SIZE ERROR. Si se especifica esta opción, el valor del campo receptor no se alterará y la proposición imperativa asociada con esta opción se ejecutará después de haber examinado el último campo receptor.

Add

La proposición ADD *(adición)* se emplea para sumar dos o más operandos numéricos y almacenar el resultado. Existen tres tipos de formatos:

Formato 1



Los identificadores deben ser campos numéricos elementales.

Todos los literales deben ser numéricos.

En este formato, se suman los valores de los operandos que preceden a la palabra TO. A continuación, el resultado obtenido se suma al valor actual del identificador-m, identificador-n, etc. El resultado de las sumas individuales se almacena en el identificador-m, identificador-n, etc., respectivamente.

Formato 2

```
ADD { identificador-1 } { identificador-2 } [identificador-3] ...
    { literal-1     }   { literal-2     }   [literal-3     ]
    GIVING identificador-n [ROUNDED]
    [ON SIZE ERROR proposición-imperativa]
```

Los identificadores situados a la derecha de la palabra GIVING pueden ser campos numéricos o campos numéricos editados.

Los valores de los operandos que preceden a la palabra GIVING se suman y el resultado se almacena como nuevo valor en el identificador-n.

Formato-3

```
ADD { CORRESPONDING } identificador-1 TO
    { CORR          }
    identificador-2 [ROUNDED]
    [ON SIZE ERROR proposición-imperativa]
```

Con este formato, los campos de datos de identificador-1 se suman a los campos de datos corresondientes a identificador-2, almacenándose el resultado en identificador-2.

Subtract

La proposición SUBTRACT *(sustracción)* se emplea para restar uno o varios campos de datos numéricos de otro u otros campos y almacenar los resultados en uno o más campos. Pueden haber tres tipos de formatos:

Formato 1

```
SUBTRACT { identificador-1 } [ identificador-2 ] ...
         { literal-1       } [ literal-2       ]

FROM identificador-m [ROUNDED] [identificador-n [ROUNDED]] ...

[ON SIZE ERROR proposición-imperativa]
```

Los identificadores y literales que preceden a la palabra FROM se suman y el resultado se resta de identificador-m, identificador-n, etc. El resultado de las respectivas restas se almacena como nuevo valor en identificador-m, identificador-n, etc.

Formato-2

```
SUBTRACT { identificador-1 } [ identificador-2 ] ...
         { literal-1       } [ literal-2       ]

FROM { identificador-m } GIVING identificador-n [ROUNDED]
     { literal-m       }

[ON SIZE ERROR proposición-imperativa]
```

El identificador-n puede contener símbolos de edición. Todos los demás identificadores deben ser campos numéricos elementales.

Con este formato, todos los identificadores literales que preceden a la palabra FROM se suman y el total se resta de literal-m ó de identificador-m. El resultado de la resta se almacena en identificador-n.

Formato 3

```
SUBTRACT { CORRESPONDING } identificador-1
         { CORR          }
    FROM identificador-2 [ROUNDED]
    [ON SIZE ERROR proposición-imperativa]
```

Con este formato, los campos de datos de identificador-1 se restan de los correspondientes campos de identificador-2 y el resultado se almacena en identificador-2.

Multiply

La proposición MULTIPLY *(multiplicación)* se utiliza para multiplicar campos numéricos y almacenar los resultados en un campo de datos.

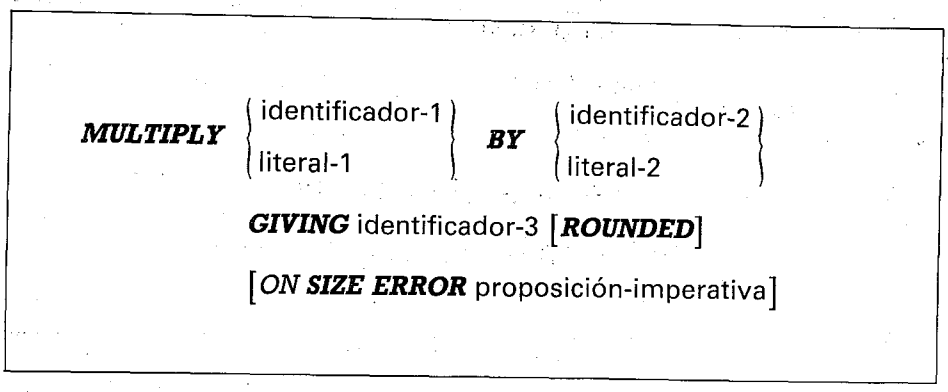
Pueden haber dos tipos de formatos:

Formato 1

```
MULTIPLY { identificador-1 } BY identificador-2 [ROUNDED]
         { literal-1       }
    [ON SIZE ERROR proposición-imperativa ]
```

El valor inicial de identificador-1, o literal-1, se multiplica por el valor inicial de identificador-2. El valor de identificador-2 se sustituye por el resultado obtenido.

Formato 2



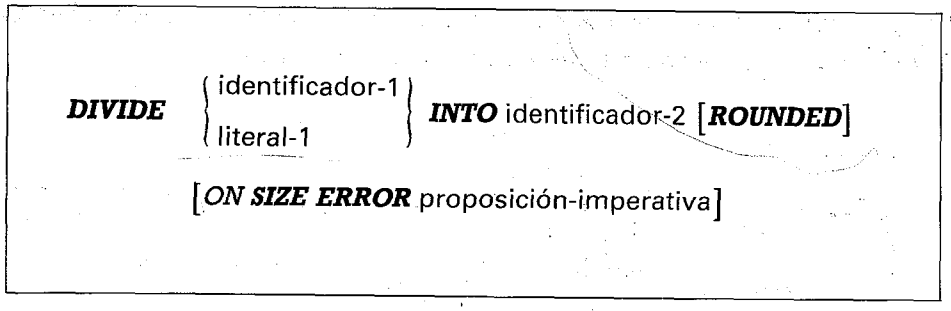
El valor inicial de identificador-1, o literal-1, se multiplica por el valor inicial de identificador-2 o literal-2 y el resultado se almacena en identificador-3.

Divide

La proposición DIVIDE *(división)* se utiliza para dividir un campo numérico por otro y almacenar el resultado en un campo de datos.

Pueden haber cinco formatos diferentes:

Formato 1



El campo de datos que precede a la palabra INTO *(entre)*, es decir identificador-1 o literal-1, constituye el dividendo y el identificador-2 el divisor. El resultado de la división sustituye al identificador-2.

La *proposición-imperativa* de la opción ON SIZE ERROR se ejecutará cuando se intente dividir por cero.

Formato 2

```
DIVIDE { identificador-1 } INTO { identificador-2 }
       { literal-1       }      { literal-2       }
    GIVING identificador-3 [ROUNDED]
    [ON SIZE ERROR proposición-imperativa]
```

En este caso, el resultado de la división se almacena en el identificador-3.

Formato 3

```
DIVIDE { identificador-1 } BY { identificador-2 }
       { literal-1       }    { literal-2       }
    GIVING identificador-3 [ROUNDED]
    [ON SIZE ERROR proposición-imperativa]
```

Los campos de datos, identificador-1 o literal-1, que preceden a la palabra BY *(por)* constituyen el divisor y el identificador-2 (o el literal-2) es el dividendo. El resultado de la división se almacena en el identificador-3.

Formato 4

```
DIVIDE { identificador-1 } INTO { identificador-2 }
       { literal-1       }      { literal-2       }
    GIVING identificador-3 [ROUNDED]
    REMAINDER identificador-4
    [ON SIZE ERROR proposición-imperativa]
```

El formato 4 se utiliza cuando se desea obtener el resto de una división en el identificador-4.

El resto de una división se define como el resultado de restar al dividendo el producto del cociente por el divisor.

Si se especifica la opción ROUNDED, el cociente se redondea una vez determinado el resto.

Formato 5

```
DIVIDE { identificador-1 } BY { identificador-2 }
       { literal-1       }    { literal-2       }
    GIVING identificador-3 [ROUNDED]
    REMAINDER identificador-4
    [ON SIZE ERROR proposición-imperativa]
```

Este formato tiene las mismas funciones que el formato 4 con la diferencia de que en este caso el divisor es el campo que precede a la palabra BY, es decir, identificador-1 o literal-1.

Compute

La proposición compute *(calcular)* asigna a un campo el valor de otro campo numérico, un literal o una expresión aritmética. Su formato es:

```
COMPUTE identificador-1 [ROUNDED] = { expresión-aritmética }
                                    { identificador-2      }
                                    { literal              }
    [ON SIZE ERROR proposición-imperativa]
```

Como en los demás verbos aritméticos, en este caso el literal debe ser numérico.

Los identificadores deben ser campos numéricos elementales excepto el identificador-1 que, en estè caso, puede ser un campo numérico editado.

Las opciones *identificador-2* y *literal* permiten poner el valor de *identificador-1* igual al de *identificador-2* o *literal.*

La opción *expresión-aritmética* permite utilizar cualquier combinación de identificadores, literales numéricos, operadores aritméticos y, si son necesarios, paréntesis. El resultado final de las operaciones en una expresión aritmética se sitúa en identificador-1.

Una proposición COMPUTE con varios operandos puede tener resultados intermedios. El compilador trata la proposición como una sucesión de operaciones y reserva áreas de memoria para los resultados intermedios. El compilador también determina el número de posiciones de entero y decimales que ha de reservar para los resultados intermedios.

La opción ON SIZE ERROR se aplica solamente a la división por cero y a los resultados finales.

En una proposición COMPUTE se pueden utilizar los siguientes operadores aritméticos:

operación	*operador*
suma	+
resta	−
multiplicación	*
división	/
exponenciación	**

Además, pueden utilizarse paréntesis para indicar el orden en que deben hacerse las operaciones.

4.6.2. Verbos de bifurcación

Las proposiciones, párrafos y secciones de la PROCEDURE DIVISION se ejecutan, generalmente, en el orden en que se presentan. Sin embargo, a menudo es necesario alterar esta secuencia de operación, saltar a un punto dife-

rente del programa para ejecutar una serie de posiciones y después volver a la proposición siguiente de la secuencia normal.

Los verbos de bifurcación de procedimiento que permiten modificar la secuencia de operaciones lógicas son:

— GO TO
— PERFORM
— EXIT
— ALTER

Go To

La proposición GO TO *(ir a)* transfiere el control desde una parte de la PROCEDURE DIVISION a otra.

Pueden distinguirse dos tipos de formatos de esta proposición:

Formato 1

```
GO TO nombre-de-procedimiento
```

El control del programa se transfiere al nombre-de-procedimiento.

El *nombre-de-procedimiento* es el nombre de un párrafo o sección de la PROCEDURE DIVISION.

Formato 2

```
GO TO nombre-de-procedimiento-1 [nombre-de-procedimiento-2]...,
      nombre-de-procedimiento-n DEPENDING ON identificador
```

El control del programa se transfiere a nombre-de-procedimiento-1, nombre-de-procedimiento-2, ..., nombre-de-procedimiento-n, dependiendo de que el valor del identificador sea 1, 2, ..., n.

El *indentificador* es el nombre de un campo numérico elemental cuyo valor es un número entero.

Si el valor del identificador es mayor que n o igual a cero, el control se transfiere a la sentencia que sigue a esta proposición.

Perform

La proposición PERFORM permite la salida temporal de la secuencia normal de ejecución para realizar uno o varios procedimientos un número determinado de veces y a continuación se devuelve el control automáticamente a la secuencia normal.

Pueden distinguirse cuatro formatos:

Formato 1

```
PERFORM nombre-de-procedimiento-1
        [THRU nombre-de-procedimiento-2]
```

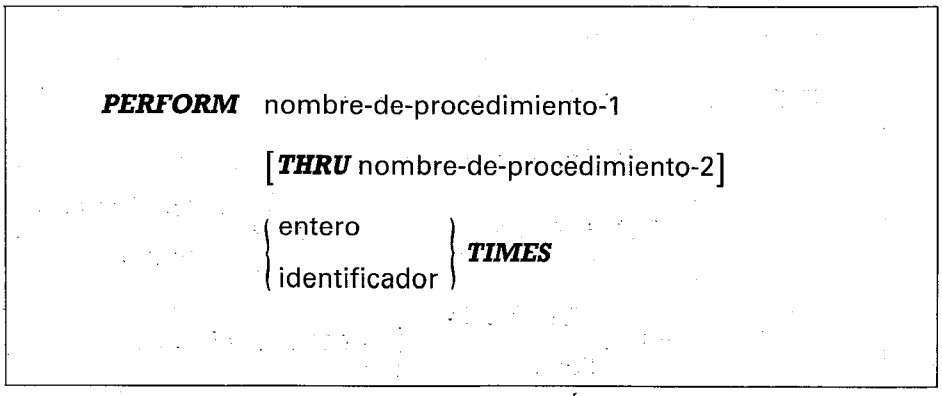
Cuando se ejecuta esta proposición, el control se transfiere a nombre-de-procedimiento-1.

Si no se ha especificado nombre-de-procedimiento-2, se devuelve el control de la ejecución a la última proposición de nombre-de-procedimiento-1.

Si se ha especificado nombre-de-procedimiento-2, se devuelve el control después de la ejecución de la última proposición de nombre-de-procedimiento-2.

Cuando se utiliza este formato, el procedimiento referenciado por PERFORM se ejecuta una sola vez y después se devuelve el control a la proposición que sigue a la PERFORM.

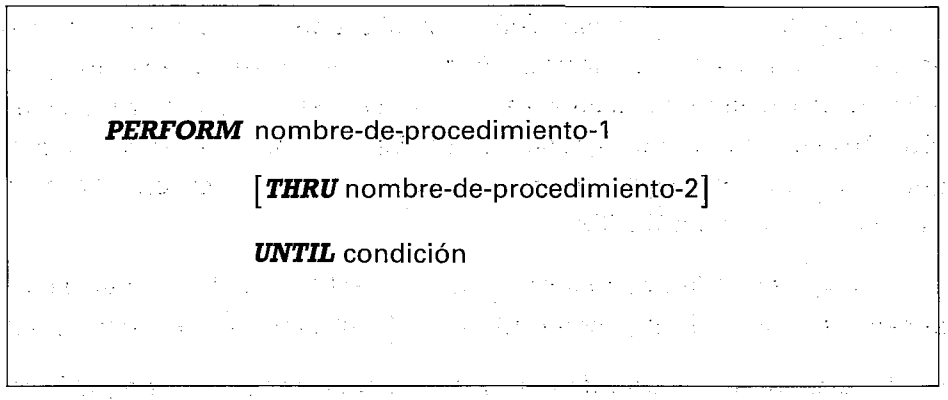
Formato 2



El *identificador* representa un campo numérico elemental descrito en la DATA DIVISION cuyo valor es entero y positivo.

Cuando se utiliza la opción TIMES, los procedimientos referenciados se ejecutan el número de veces especificado por *entero* o *identificador*. A continuación, se transfiere el control a la proposición siguiente a PERFOM.

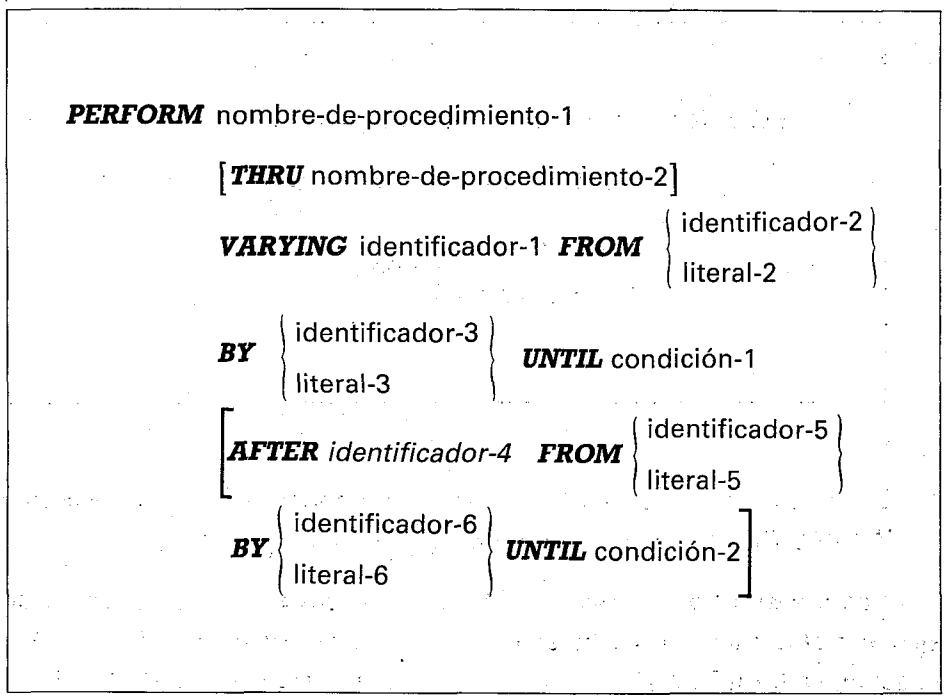
Formato 3



Cuando se utiliza la opción UNTIL, los procedimientos referenciados se ejecutan *hasta* que se cumpla la *condición* especificada.

Hay que tener en cuenta que si la *condición* se cumple al comienzo de la ejecución de la PERFORM, no se ejecuta el procedimiento especificado y el control se devuelve a la proposición siguiente a PERFORM.

Formato 4



La opción VARYNG se utiliza para variar el valor de uno o más identificadores durante la ejecución de una proposición PERFORM.

Cuando se varía *un identificador,* identificador-1 se pone a su valor inicial (que es identificador-2 o literal-2). Si condición-1 no se cumple, los procedimientos se ejecutan una sola vez y el valor de identificador-1 se incrementa o disminuye en el valor de identificador-3 o literal-3. Este proceso continúa hasta que condición-1 sea verdadera, en cuyo caso se devuelve el control a la proposición que sigue a PERFORM.

Cuando se varían *dos identificadores,* identificador-1 e identificador-4 se ponen a su valor inicial (identificador-2 e identificador-5 respectivamente).

Si la condición-1 es verdadera, se transfiere el control a la proposición que sigue a PERFORM. Si es falsa, se evalúa la condición-2. Si esta última es verdadera, identificador-4 se pone a su valor inicial (identificador-5) identificador-1 se modifica con el valor de identificador-3 y la condición-1 vuelve a evaluarse. Si, por el contrario, la condición-2 es falsa, la secuencia de procedimientos se ejecuta una vez, identificador-4 se modifica con el valor de identificador-6 y se vuelve a evaluar la condición-2, repitiéndose este ciclo hasta que la condición-2 sea verdadera.

El organigrama de la lógica de la PERFORM puede verse en la figura inferior. Donde In son los identificadores y donde NPn son los nombres de procedimiento.

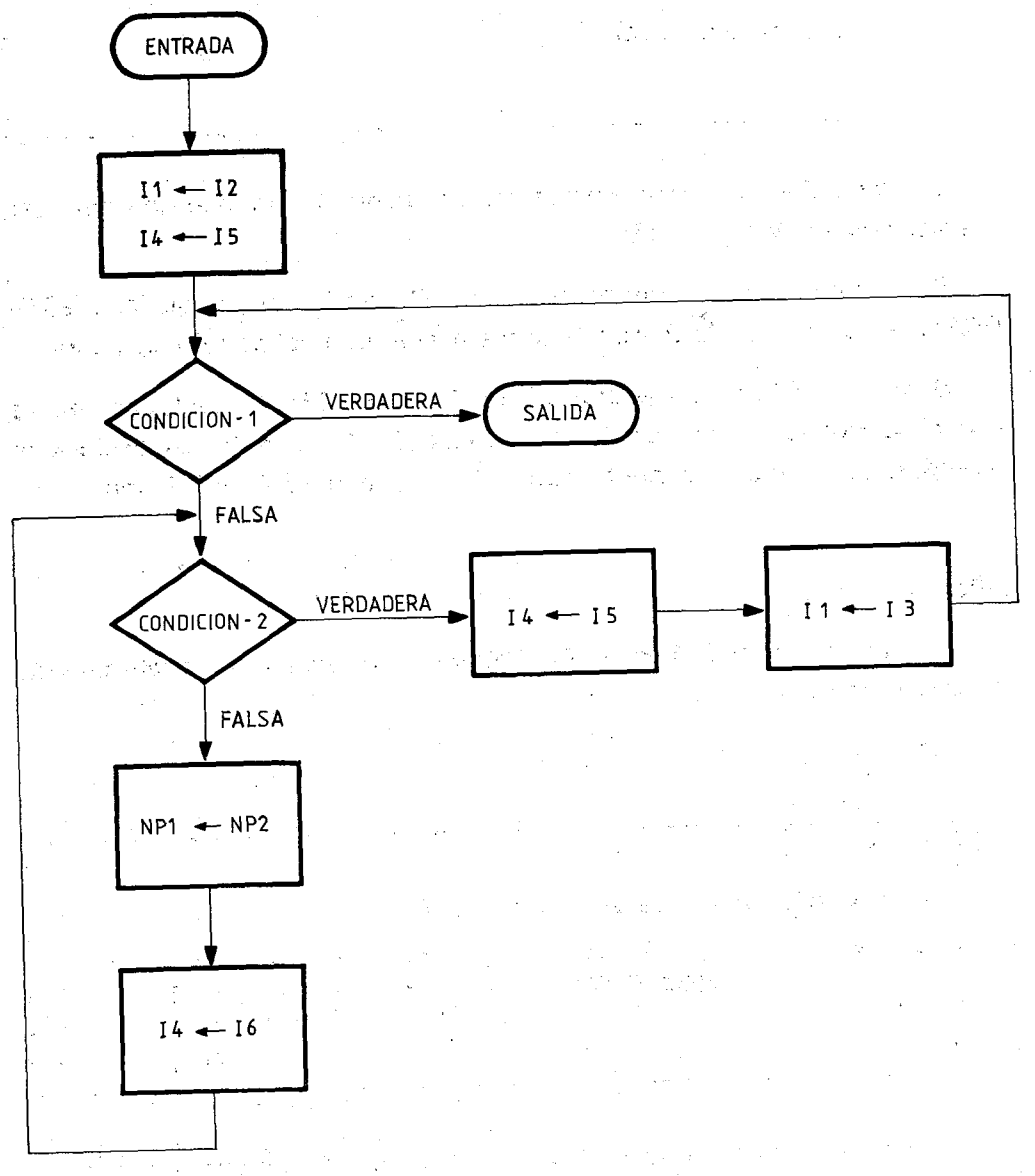


Diagrama de la proposición PERFORM

Exit

La proposición EXIT proporciona un punto de *salida* o de terminación común a una serie de procedimientos o indica el final de un programa llamado.

Su formato es:

```
EXIT [PROGRAM]
```

La proporción EXIT debe ir precedida por un nombre-de-párrafo y debe ser la única sentencia del párrafo.

Si el control llega a una proposición EXIT sin la palabra opcional PROGRAM, el control se transfiere a la primera sentencia del párrafo siguiente.

Si el control llega a una proposición EXIT PROGRAM mientras se opera bajo el control de una proposición de *llamada* (CALL), el control se devuelve al programa que llama en el punto siguiente a la proposición de llamada.

Alter

La proposición ALTER *(modificar)* cambia una secuencia predeterminada de operaciones.

Formato:

```
ALTER nombre-de-procedimiento-1

      [TO PROCEED TO] nombre-de-procedimiento-2
```

El nombre-de-procedimiento-1 es un nombre de párrafo que contiene una proposición GO TO sin la opción DEPENDING ON.

El nombre-de-procedimiento-2 es un nombre de párrafo o sección de la PROCEDURE DIVISION.

Durante la ejecución del programa objeto, la proposición ALTER modifica la proposición GO TO en cuanto al párrafo nombre-de-procedimiento-1, sustituyendo el objeto de la GO TO por nombre-de-procedimiento-2.

4.6.3. Verbos de transferencia de datos

En este grupo, pueden considerarse los siguientes verbos:

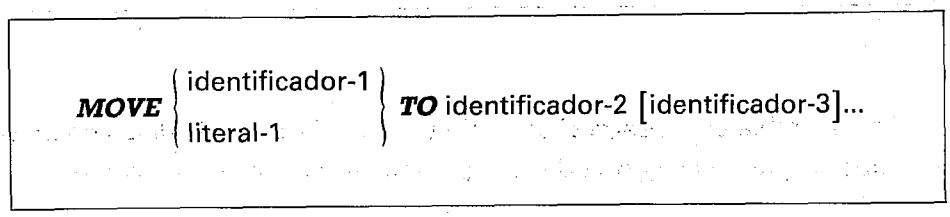
— MOVE
— EXAMINE
— SET

Move

La proposición MOVE *(mover)* transfiere datos a una o más áreas siguiendo las reglas de edición.

Pueden distinguirse dos formatos diferentes:

Formato 1



Si la transferencia es a más de un área, los datos designados por identificador-1 o literal-1 se transfieren en primer lugar a identificador-2, después a identificador-3, etc.

Las transferencias de datos en las que los campos emisor y receptor son elementales son transferencias elementales. Todos los campos elementales pueden pertenecer a una de las siguientes categorías:

— alfabética
— numérica
— alfanumérica
— numérica editada
— alfanumérica editada

Según la categoría del campo emisor y receptor puede construirse la siguiente tabla:

EMISOR	*RECEPTOR*				
	Alfabético	Numérico	Alfanumérico	Numérico editado	Alfanumérico editado
Alfabético	SI	NO	SI	NO	SI
Numérico	NO	SI	SI	SI	SI
Alfanumérico	SI	SI	SI	SI	SI
Numérico editado	NO	NO	SI	NO	SI
Alfanumérico editado	SI	NO	SI	NO	SI

En esta tabla se muestra si es posible la transferencia de datos de un campo a otro, según la categoría del campo emisor y del campo receptor.

Si el tamaño del campo emisor es mayor que el del receptor, los caracteres que sobran se truncan una vez lleno el campo receptor.

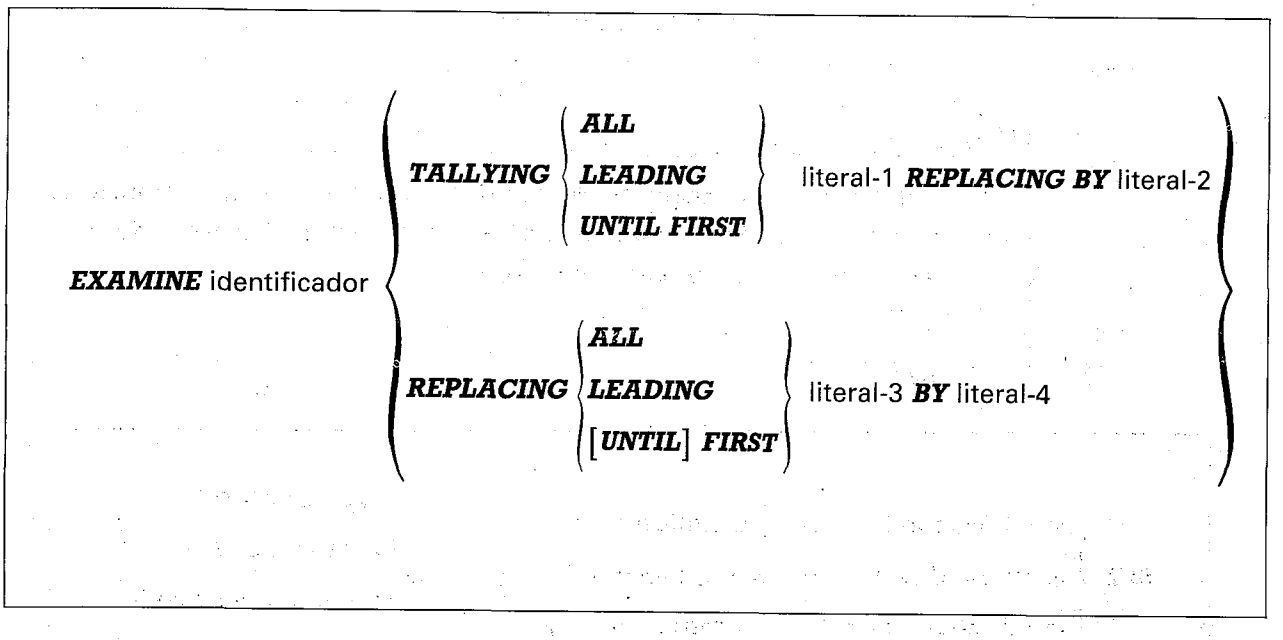
Durante la transferencia elemental se realiza cualquier conversión necesaria de datos (de una forma de representación a otra), así como cualquier edición especificada en el campo receptor.

Si la transferencia no es elemental, se trata como si fuera una transferencia entre campos alfanuméricos elementales pero, en este caso, no hay conversión de datos de una forma de representación interna a otra.

La constante figurativa SPACE (SPACES) pertenece a la categoría alfabética y la constante figurativa ZERO (ZEROS, ZEROES) pertenece a la categoría numérica. El resto de las constantes figurativas pertenecen a la categoría alfanumérica.

Examine

Esta proposición *reemplaza* o *cuenta* el número de repeticiones de un carácter dado en un campo de datos. Su formato es:



La descripción del identificador debe tener USAGE IS DISPLAY.

Cada literal debe constar de un sólo carácter perteneciente a una clase compatible con la clase del identificador.

La EXAMINE *(examen)* de identificador comienza con el primer carácter de la izquierda hasta alcanzar el último de la derecha. Cada uno de los caracteres se examina individualmente.

La opción TALLING crea una cuenta en un registro especial llamado TALLY. Con la opción ALL cuenta el número de repeticiones de literal-1; la opción LEADING cuenta el número de repeticiones de literal-1 antes de encontrar un carácter distinto; y la opción UNTIL FIRST cuenta el número de caracteres distintos a literal-1 encontrados antes de la primera repetición de literal-1.

Si se utiliza la opción REPLACING *(reemplazar)*, las reglas de sustitución son las siguientes:

— Con la opción ALL, literal-2 o literal-4 reemplazan todas las repeticiones de literal-1 o literal-3.
— Con la opción LEADING, las sustituciones por literal-2 o literal-4 terminan al encontrar un carácter distinto de literal-1 o literal-3.

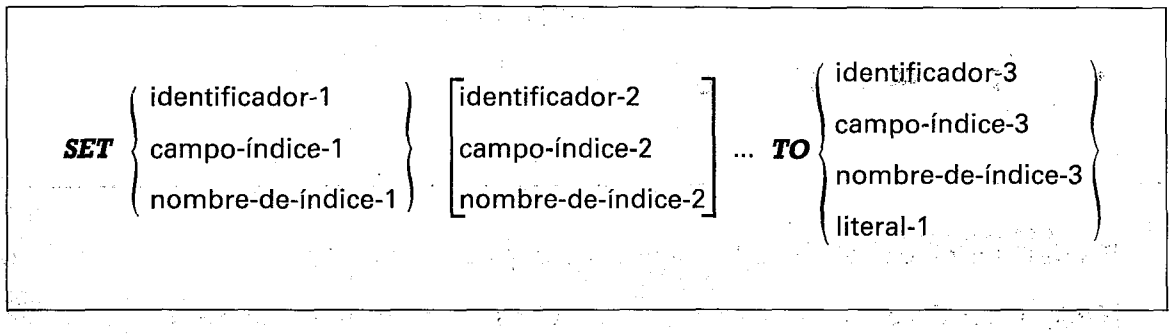
— Con la opción UNTIL FIRST, las sustituciones por literal-2 o literal-4 terminan al encontrar el primer literal-1 o literal-3.

— Con la opción FIRST, la primera repetición de literal-3 se reemplaza por literal-4.

Set

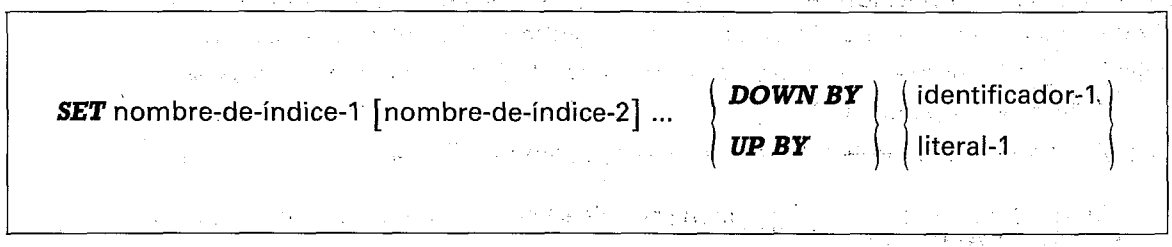
La proposición SET establece puntos de referencia para operaciones de manejo de tablas, fijando el contenido de los nombres-de-índice asociados con los elementos de la tabla. Sus formatos son:

Formato 1



Los nombres-de-índice se consideran relacionados con una tabla de datos y se definen mediante la cláusula INDEXED BY.

Formato 2



En este formato, el contenido de nombre-de-índice-1, nombre-de-índice-2, ..., se incrementa (UP BY) o disminuye (DOWN UP) en el valor correspondiente al número de repeticiones representado por el valor de identificador-1 o literal-1.

Las combinaciones permitidas de la proposición SET son:

A)

```
SET nombre-de-índice-1 TO { nombre-de-índice-2 }
                          { campo-índice       }
```

El valor del número de repetición de nombre-de-índice-2 (o campo-índice) se utiliza para calcular un nuevo valor de desplazamiento para el nombre-de-índice-1. Además el valor del número de repetición de nombre-de-índice-3 reemplaza al de nombre-de-índice-1.

B)

```
SET nombre-de-índice { DOWN BY } { identificador }
                     { TO      } { literal     }
                     { UP BY   }
```

El valor de identificador o literal se trata como un número de repetición y se utiliza para calcular un nuevo valor de desplazamiento para nombre-de-índice. El valor de identificador o de literal sustituye (TO), incrementa (UP BY) o disminuye (DOWN BY) el valor del número de repetición de nombre-de-índice.

C) y *D)*

```
SET campo-índice-1 TO { campo-índice-2 }
                      { nombre-índice  }

SET identificador TO nombre-de-índice
```

4.6.4. Verbos de entrada/salida

Los verbos de entrada/salida controlan y coordinan las transferencias de datos entre la memoria central y los soportes de información externos, como unidades de fichas, cintas o discos. Estos verbos permiten obtener registros para procesarlos y después devolverlos al soporte de información externo.

Los verbos de E/S son: ACCEPT, DISPLAY, OPEN, CLOSE, READ, WRITE, INSERT, REWRITE, SORT, RETURN y SEEK

Accept

La proposición ACCEPT *(aceptar)* permite leer pequeñas cantidades de datos desde un dispositivo de E/S o una posición de memoria.

Su formato es:

ACCEPT identificador [***FROM*** nombre-mnemotécnico]

esta proposición permite sustituir el contenido del campo identificador por la serie de datos disponibles en *nombre-mnemotécnico.*

La descripción de identificador determina el número de fichas aceptadas. El identificador debe describirse con USAGE IS DISPLAY.

El nombre-mnemotécnico debe estar asociado a un nombre de proveedor del párrafo SPECIAL-NAMES de la ENVIRONMENT DIVISION.

Por ejemplo, para indicar que se aceptan datos desde la consola:

```
ACCEPT IDENT FROM CONSOLE
```

Otros ejemplos de la proposición ACCEPT son:

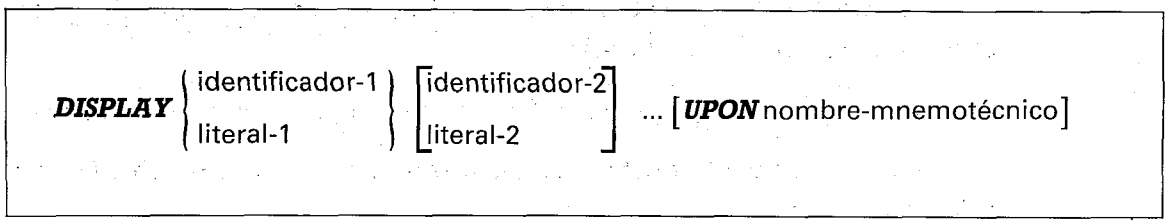
```
ACCEPT IDENT FROM DATE

ACCEPT IDENT FROM TIME
```

Esta proposición pone la fecha (DATE) y la hora (TIME) a disposición del programa

Display

La proposición DISPLAY escribe pequeñas cantidades de datos en un periférico adecuado o en la memoria central. Su formato es:



Si se omite la opción UPON, los datos se escriben en la consola del sistema (CONSOLE).

Por el contrario, si se especifica esta opción, el nombre-mnemotécnico debe estar asociado con un nombre del proveedor en el párrafo SPECIAL-NAMES de la ENVIRONMENT DIVISION.

Ejemplos:

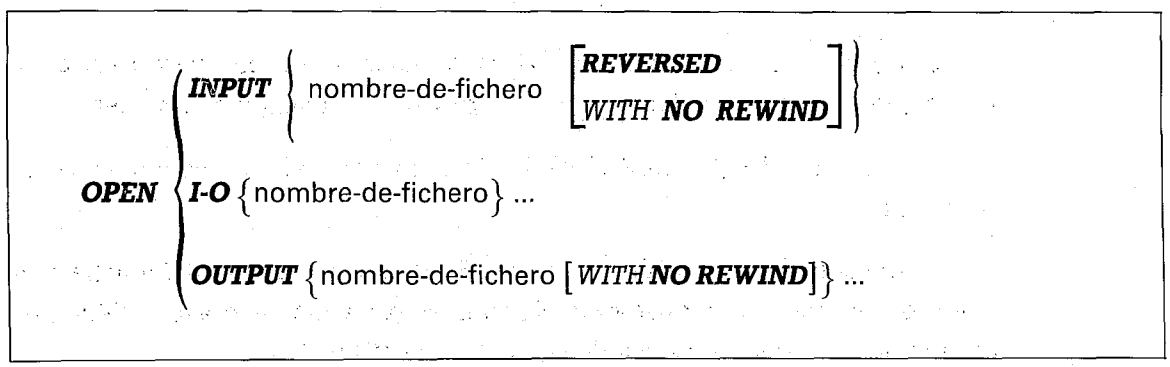
```
DISPLAY IDENTIFICADOR UPON CONSOLE

DISPLAY IDENTIFICADOR UPON LST
```

Estas proposiciones permiten al programa COBOL visualizar mensajes en la consola del sistema (CONSOLE) o en la impresora (LST).

Open

La proposición OPEN *(abrir)* inicia el proceso de ficheros de entrada/salida. Además inicia la comprobación o escritura de etiquetas y otras operaciones de E/S. Su formato es:



La proposición OPEN debe aplicarse a todos los ficheros.

Nombre-de-fichero es el nombre FD de la FILE SECTION de la DATA DIVISION.

Las opciones REVERSED y NO REWIND se aplican solamente al proceso secuencial de ficheros de un solo carrete.

La opción I-O sólo se utiliza en ficheros de memoria de gran capacidad.

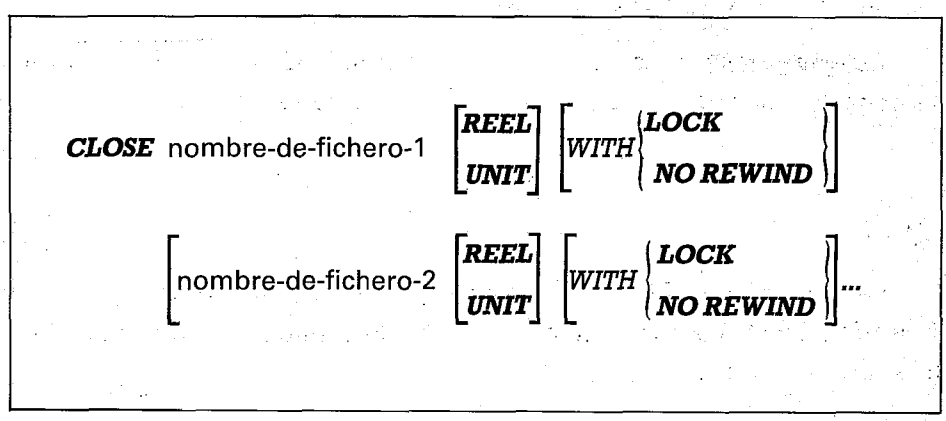
La proposición OPEN debe ejecutarse antes de que se haga cualquier otra referencia al fichero.

No se puede ejecutar una segunda proposición OPEN *(abrir)* antes de haber ejecutado una proposición CLOSE *(cerrar)* para un determinado fichero.

Close

La proposición CLOSE *(cerrar)* permite finalizar el proceso de uno o más ficheros de entrada/salida.

El formato es:



Las opciones REEL *(carrete)* o UNIT *(unidad)* se utilizan para realizar un cambio de carrete o unidad durante el proceso secuencial de un fichero.

Las opciones REEL, LOCK y NO REWIND sólo se aplican a ficheros de cinta magnética.

Si se especifica la opción LOCK para REEL, el carrete actual del fichero en cintas se rebobina y se desmonta. Cuando se especifica la opción LOCK para todo el fichero, éste no se puede volver a abrir (OPEN).

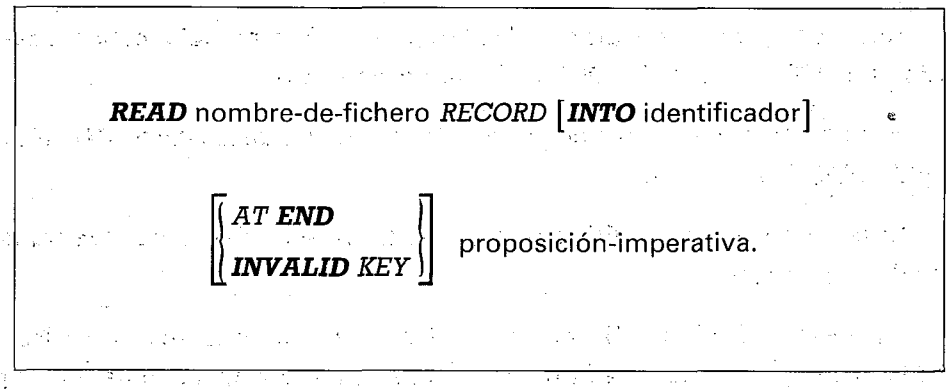
Cuando se especifica la opción NO REWIND, el carrete del fichero permanece en la posición en que se encuentra en el momento de la CLOSE.

Si no se especifica LOCK ni NO REWIND, se rebobinan todos los carretes del fichero.

Read

La función de la proposición READ *(leer)* consiste en obtener el siguiente registro de un fichero.

Su formato es:



La cláusula AT END o INVALID KEY se requiere en todas las organizaciones de ficheros excepto en la organización indexada, en la que es opcional.

En el proceso secuencial de ficheros, esta proposición permite además la ejecución de la *proposición-imperativa* al detectar el fin de fichero (AT END).

En el proceso al azar de ficheros, READ permite también la ejecución de la *proposición-imperativa* si el contenido de las claves (KEYS) no es válido (INVALID).

Antes de ejecutarse la primera proposición READ para un fichero, debe haberse ejecutado una proposición OPEN INPUT u OPEN I-O para ese fichero.

La opción INTO sólo se puede utilizar cuando el archivo de entrada contiene registros de un sólo tamaño.

No se puede acceder al área de registros antes de una READ o después de cumplirse una condición AT END.

Write

La proposición WRITE *(escribir)* permite escribir un registro en un fichero de salida. Tiene dos formatos diferentes:

Formato-1

```
WRITE nombre-de-registro [FROM identificador-1]
      [ { AFTER  } ADVANCING { identificador-2 LINES }  ]
      [ { BEFORE }           { entero LINES          }  ]
```

Antes de ejecutarse la primera WRITE para un fichero, debe haberse ejecutado una OPEN OUTPUT u OPEN I-O para ese fichero.

El *nombre-de-registro* es el nombre de un registro de la FILE SECTION de la DATA DIVISION.

Si se utiliza la opción FROM los datos se transfieren desde *identificador-1* a *nombre-de-registro.*

La opción ADVANCING permite controlar el posicionamiento vertical de cada registro en la hoja de la impresora. Si se omite esta opción para un fichero de impresora, ésta avanzará automáticamente una línea antes de imprimir.

Cuando se desea una sobreimpresión, el valor de *identificador-2* o de *entero* debe ponerse a cero.

Formato 2

```
WRITE nombre-de-registro [FROM identificador-1];
      [INVALID KEY proposición-imperativa]
```

La cláusula INVALID KEY se utiliza para el proceso de fichero de acceso directo y es obligatoria en todas las organizaciones de fichero, excepto en la

organización indexada, en la que es opcional. La ejecución de la INVALID KEY depende de la organización y modo de acceso del fichero.

Esta proposición envía un registro lógico a un fichero de salida.

En ambos formatos, una vez ejecutada la proposición WRITE, la información de nombre-de-registro ya no está disponible, pero sí lo está la información del identificador-1. No puede accederse al área de registros asociada con un fichero de salida antes de abrir ese fichero.

Insert

La proposición INSERT *(insertar)* se utiliza para insertar un registro en un fichero de acceso directo o indexado.

Formato:

```
INSERT nombre-de-registro [FROM identificador]
       [INVALID KEY proposición-imperativa]
```

Si se utiliza la opción FROM, se transfieren los datos desde *identificador* a *nombre-de-registro.* Una vez ejecutada la INSERT, ya no se puede disponer de la información de *nombre-de-registro,* pero sí de la de *identificador.*

La opción INVALID KEY se requiere en todas las organizaciones de fichero excepto en la organización indexada, en la que es opcional.

Rewrite

La proposición REWRITE permite sustituir un registro del fichero por un registro especificado. Su formato es:

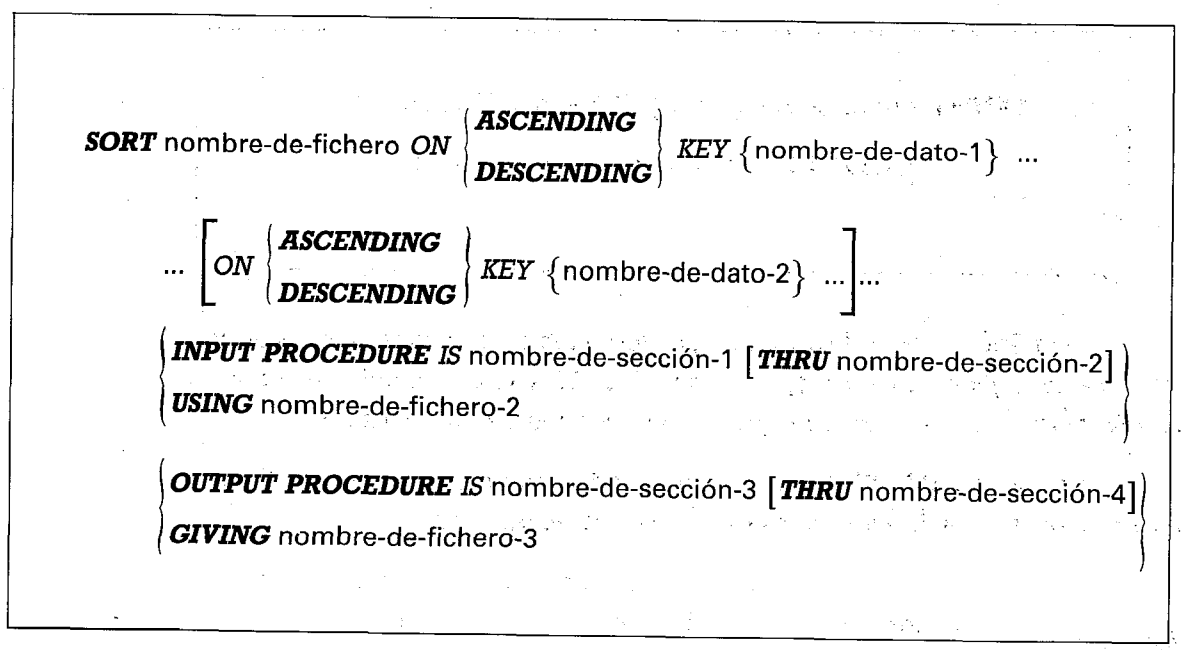
```
REWRITE nombre-de-registro [FROM identificador]
        [INVALID KEY proposición-imperativa]
```

Si se utiliza la opción FROM, se transfieren los datos de *identificador* a *nombre-de-registro*. Una vez ejecutada la REWRITE, sólo puede utilizarse la información de *identificador-1*.

La opción INVALID KEY se utiliza cuando se procesan ficheros de acceso directo.

Sort

La proposición SORT crea un fichero de clasificación mediante la ejecución de procedimientos de entrada, o la transferencia de registros desde otro fichero. Clasifica los registros y los pone a disposición de uno o más procedimientos de salida, o de un fichero de salida. Su formato es:



Nombre-de-fichero-1 debe estar descrito en la DATA DIVISION con un nivel SD (Sort Description).

Nombre-de-sección-1 y *nombre-de-sección-3* son nombres de un procedimiento de entrada y salida, respectivamente.

Nombre-de-fichero-2 y *nombre-de-fichero-3* deben describirse en una anotación FD de la DATA DIVISION.

Cuando se emplea la opción ASCENDING, la secuencia clasificada va del valor de clave (KEY) menor al mayor. La secuencia clasificada es la inversa cuando se utiliza la opción DESCENDING.

Cuando se indica INPUT PROCEDURE, el control pasa a nombre-de-sección-1 antes de poner en secuencia nombre-de-fichero-1. La proposición SORT realiza automáticamente la función de las proposiciones OPEN, READ, USE y CLOSE para nombre-de-fichero-2 (el cual debe ser de acceso secuencial).

Cuando se especifica OUTPUT PROCEDURE, el control pasa a nombre-de-sección-3 después de poner en secuencia nombre-de-fichero-1 mediante la SORT.

Si se utiliza la opción GIVING, todos los registros clasificados se transfieren a nombre-de-fichero-3 (que debe ser de acceso secuencial).

Return

Esta proposición obtiene los registros clasificados de la fase final de una operación de clasificación.

Formato:

RETURN nombre-de-fichero *RECORD* [***INTO*** identificador]

AT ***END*** proposición-imperativa

La proposición RETURN sólo puede utilizarse dentro de un procedimiento de salida asociado con una proposición SORT para nombre-de-fichero.

Cuando se ejecuta la proposición RETURN, el siguiente registro queda disponible para su proceso en las áreas de registros asociadas al fichero de clasificación.

Seek

Esta proposición inicia el acceso a un registro de una memoria de gran capacidad para su lectura o escritura.

Formato:

```
SEEK nombre-de-fichero RECORD
```

La proposición SEEK sólo puede utilizarse en ficheros en discos con organizaciones relativas o indexadas (excepto si la organización es indexada y el acceso aleatorio).

Cuando la organización es relativa, la proposición SEEK utiliza el valor de identificador de la cláusula ACTUAL KEY o RELATIVE KEY para obtener la posición del registro al que se va a acceder.

Si la organización es indexada, se utiliza el valor de identificador de la cláusula SYMBOLIC KEY.

4.6.5. Verbo de final (STOP)

La proposición STOP *(parar)* se utiliza para detener la ejecución del programa objeto temporal o permanentemente.

Formato:

```
STOP { literal }
     { RUN   }
```

El *literal* se comunica al operador a través de la consola del sistema y el programa continúa con la ejecución de la proposición que siga a STOP. El literal puede ser numérico o no numérico.

Por ejemplo:

```
STOP 'ERROR'.
```

produce la visualización del literal 'ERROR'.

Cuando se utiliza la opción RUN, el programa objeto se detiene permanentemente.

4.6.6. Verbos condicionales

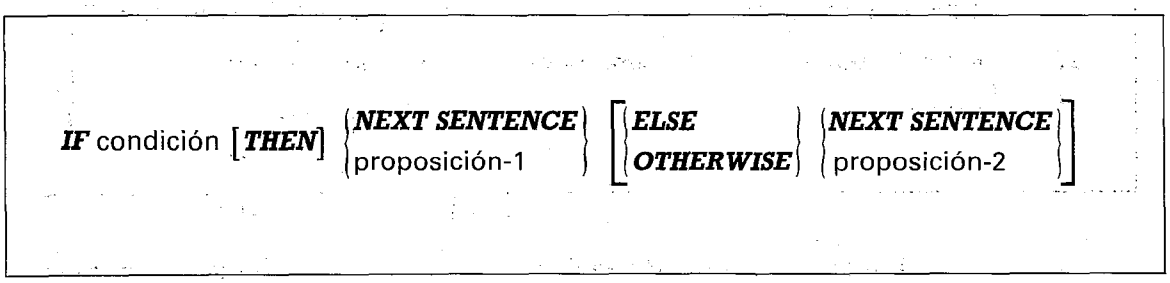
Las expresiones condicionales se emplean en situaciones en las que el resultado de una comparación determina el siguiente paso lógico que se va a ejecutar.

Los verbos condicionales son IF y SEARCH.

If

La proposición IF *(si condicional)* produce la evaluación de una condición. El paso siguiente a ejecutar dependerá de que el valor de la condición sea verdadero o falso.

Formato:



La proposición-1 y proposición-2 son proposiciones imperativas o condicionales.

Cuando se ejecuta una proposición IF pueden darse dos casos:

a) *Condición verdadera:* si la condición se cumple, se ejecutan las proposiciones que siguen inmediatamente a la condición *(proposición-1).* A continuación, pasa el control del programa a la sentencia siguiente.

b) *Condición falsa:* si la condición no se cumple, se ejecuta la *proposición-2.* Si se omite la opción ELSE se ejecutará la sentencia siguiente.

Proposición-1 y *proposición-2* pueden contener una proposición IF. Las proposiciones IF dentro de otras IF se consideran como combinaciones de IF y ELSE emparejadas, procediendo de izquierda a derecha.

La condición de una proposición IF hace que el programa objeto seleccione entre varias alternativas de control, dependiendo de que el valor de la comparación sea verdadero o falso.

Pueden darse cinco tipos de condiciones:

— condición de relación
— condición de clase
— condición de nombre-de-condición
— condición de estado del interruptor
— condición de signo

Condición de relación

Una condición de relación produce una comparación de dos operandos.

Formato:

```
IF { expresión-aritmética-1 }                    { expresión-aritmética-2 }
   { identificador-1        } operador-de-relación { identificador-2        }
   { literal-1              }                     { literal-2              }
```

El operador-de-relación indica el tipo de comparación que va a realizarse. Los operadores de relación son:

```
                                  { IS { [NOT] GREATER THAN }                    }
                                  {    { [NOT] >            }                    }
                                  { IS { [NOT] LESS THAN    }                    }
IF { expresión-aritmética-1 }     {    { [NOT] >            }                    } { expresión-aritmética-2 }
   { identificador-1        }     { IS { [NOT] EQUAL TO     }                    } { identificador-2        }
   { literal-1              }     {    { [NOT] =            }                    } { literal-2              }
                                  { EQUALS                                       }
                                  { UNEQUAL                                      }
                                  { EXCEEDS                                      }
```

Por ejemplo,

A < B AND NOT < C OR D

es equivalente a

A < B AND NOT A < C OR A < D

o bien,

A < B AND (NOT A < C) OR A < D

Los *operadores lógicos* utilizados son:

— OR
— AND
— NOT

A	B	IF A OR B	IF A AND B	IF NOT A
Falsa	Falsa	Falsa	Falsa	Verdadera
Falsa	Verdadera	Verdadera	Falsa	Verdadera
Verdadera	Falsa	Verdadera	Falsa	Falsa
Verdadera	Verdadera	Verdadera	Verdadera	Falsa

Condición de clase

La condición de clase determina si un operando es numérico o alfabético.

Formato:

IF identificador *IS* [***NOT***] { ***NUMERIC*** / ***ALPHABETIC*** }

Condición de nombre-de-condición

En este caso, se compara una variable condicional para determinar si su valor es igual a uno de los valores asociados a un nombre-de-condición.

Formato:

IF **[*NOT*]** nombre-de-condición

Condición de estado del interruptor

Determina el estado ON *(conectado)* u OFF *(desconectado)* de un interruptor (SWITCH).

Formato:

IF **[*NOT*]** nombre-de-condición

Condición de signo

Determina si el valor de un operando es menor, igual o mayor que cero.

Formato:

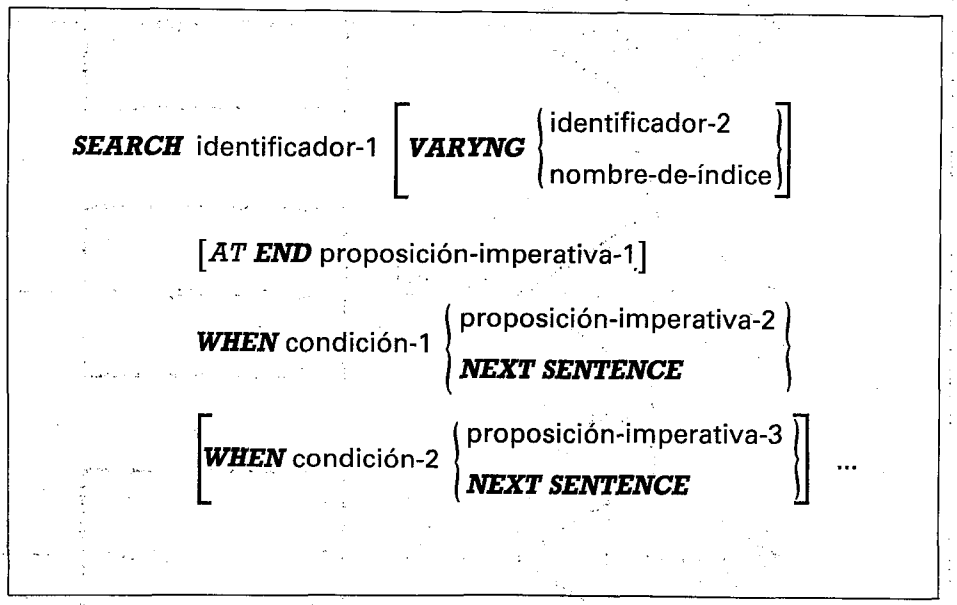
IF { expresión-aritmética / identificador } ***IS*** **[*NOT*]** { ***NEGATIVE*** / ***POSITIVE*** / ***ZERO*** }

Search

Esta proposición se emplea para buscar en una tabla un elemento que cumpla la condición especificada.

Pueden darse dos tipos de formatos:

Formato 1



La descripción de *identificador-1* debe contener una OCCURS y una cláusula INDEXED BY, pero no debe estar indexado.

Cuando se especifica *identificador-2,* debe describirse como USAGE IS INDEX o como campo numérico elemental sin posiciones decimales.

Utilizando este formato, se produce una operación de búsqueda en serie, comenzando con el contenido actual del índice. Si se especifica la opción VARYING nombre-de-índice y este nombre-de-índice aparece en la cláusula INDEXED BY de identificador-1, se utiliza dicho nombre-de-índice en la operación de búsqueda.

Además, el número de repetición representado por el nombre-de-índice se incrementa en la misma cantidad que el número de repetición representado por el nombre-de-índice asociado con identificador-1.

Si, por el contrario, se especifica VARYING identificador-2, se utiliza en la operación de búsqueda el primer nombre-de-índice de la cláusula INDEXED BY

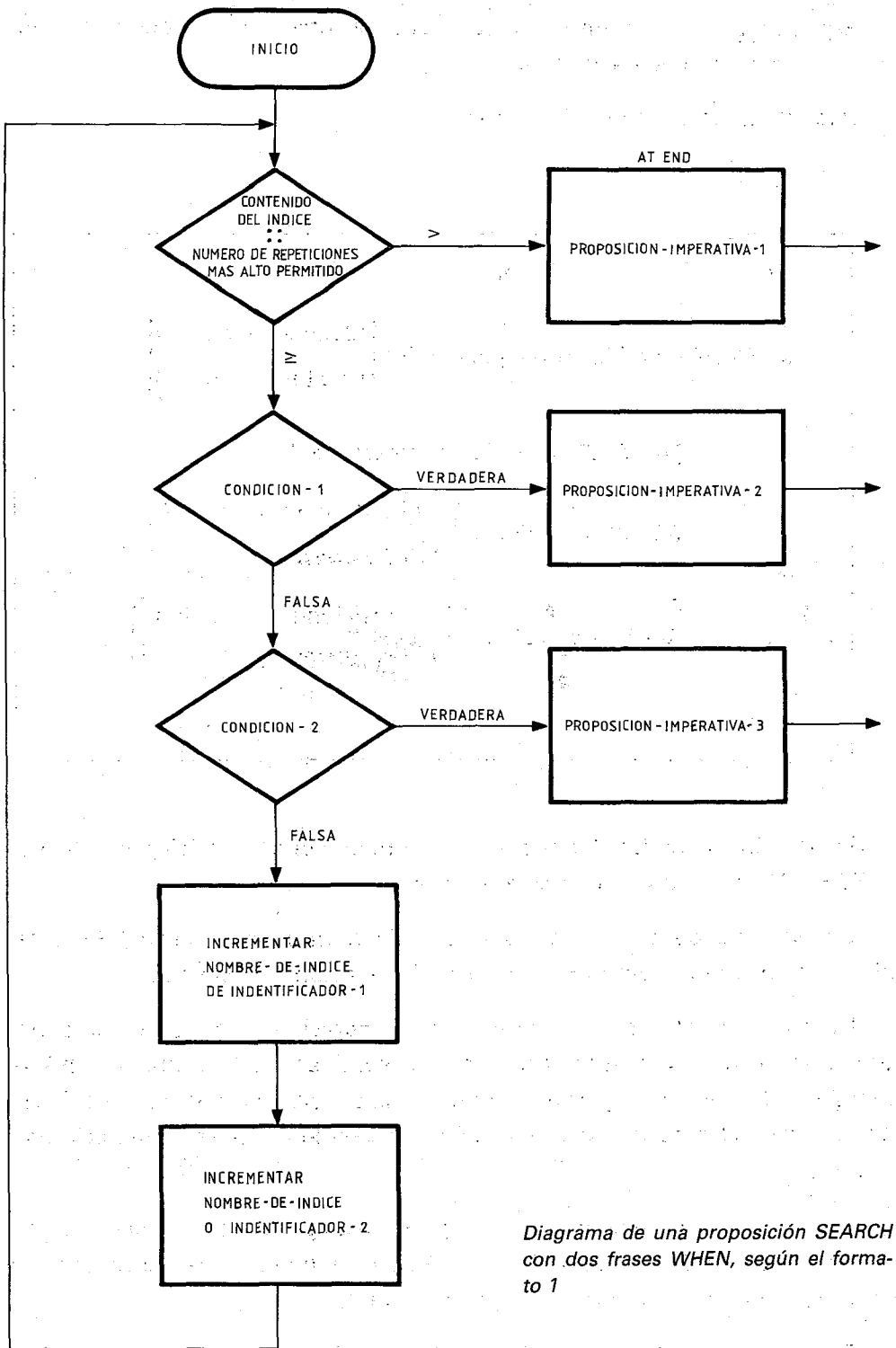


Diagrama de una proposición SEARCH con dos frases WHEN, según el formato 1

de identificador-1. Además, se incrementa identificador-2 en la misma cantidad que el número de repetición indicado por el nombre-de-índice asociado con identificador-1.

Formato 2

```
SEARCH ALL identificador-1 [AT END proposición imperativa-1]

        WHEN condición-1 { proposición-imperativa-2 }
                         { NEXT SENTENCE            }
```

Si se utiliza este formato, se produce una operación de búsqueda no en serie. Se ignora el valor inicial de nombre-de-índice para identificador-1 y se varía durante la operación de búsqueda utilizando una técnica de búsqueda binaria.

El nombre-de-índice utilizado para la operación de búsqueda es el primer (o único) nombre-de-índice que aparece en la cláusula INDEXED BY de identificador-1. Cualquier otro nombre-de-índice de identificador-1 permanece inalterable.

4.6.7. Verbos directivos del compilador

Son los verbos que dirigen al compilador para que efectúe una acción determinada. Estos verbos son: NOTE, COPY, ENTER y USE.

Note

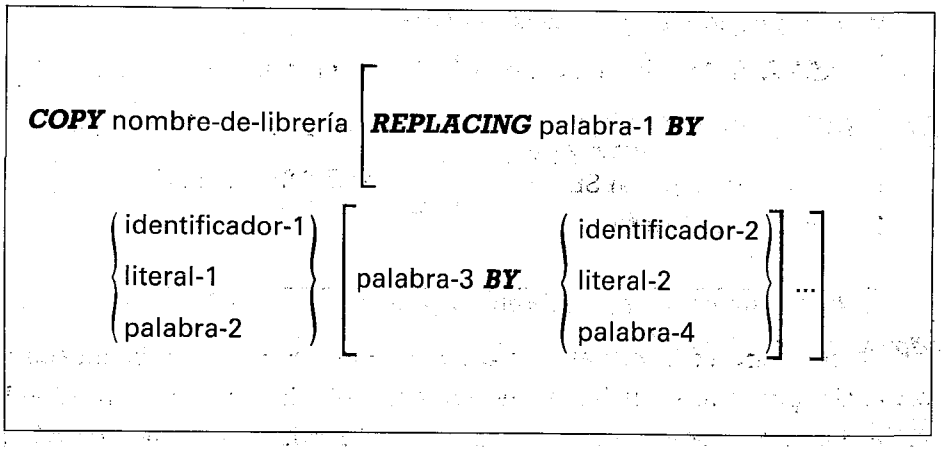
Esta proposición permite al programador escribir anotaciones o comentarios que aparecerán en el listado, pero que no se compilarán. Formato.

```
NOTE serie-de-caracteres
```

COPY

La proposición COPY *(copiar)* permite copiar textos de la librería COBOL en el programa fuente.

Formato



La proposición COPY y las proposiciones del texto de librería aparecen en el listado de salida

Cuando se utiliza la opción REPLACING, cada repetición de palabra-1, palabra-3, etc., en el texto de la librería que se está copiando, se *reemplaza* por la palabra, identificador o literal asociado con ella en la opción REPLACING.

Esta proposición puede aparecer en una sección o párrafo de la PROCEDURE DIVISION. Además, puede aparecer en cualquier párrafo de la ENVIRONMENT DIVISION o en cualquier anotación de indicador de nivel de la DATA DIVISION.

En la ENVIRONMENT DIVISION puede aparecer COPY en los siguientes párrafos:

SOURCE-COMPUTER. proposición COPY.
OBJECT-COMPUTER. proposición COPY.
SPECIAL-NAMES. proposición COPY.
FILE-CONTROL. proposición COPY.
I-O-CONTROL. proposición COPY.

En la FILE SECTION de la DATA DIVISION:

FD nombre-de-fichero proposición COPY.
SD nombre-de-fichero-de-clasificación proposición COPY.
01 nombre-de-dato proposición COPY.

En la WORKING-STORAGE SECTION de la DATA DIVISION:

01 nombre-de-dato proposición COPY.

En la PROCEDURE DIVISION:

{ nombre-de-sección SECTION. / nombre-de-párrafo } proposición COPY.

Todo lo que contenga la librería bajo el nombre-de-librería especificado se copia en el programa.

Enter

Esta proposición se utiliza para permitir, en tiempo de ejecución, las comunicaciones entre el programa principal y subprogramas compilados previamente.

Formato 1

ENTER LINKAGE.

CALL nombre-de-entrada [***USING*** { nombre-de-fichero / identificador / nombre-de-procedimiento / nombre-de-clasificación } ...]

ENTER COBOL

El control se transfiere desde un programa objeto a otro dentro de la misma unidad de trabajo.

Nombre-de-entrada indica el punto de entrada en el subprograma al que se llama (CALL). Este nombre puede ser un literal no numérico.

Nombre-de-fichero, nombre-de-procedimiento y *nombre-de-clasificación* sólo se utilizan si el subprograma llamado no está escrito en COBOL.

Si el subprograma está escrito en COBOL, el punto de entrada es el principio de la PROCEDURE DIVISION y el nombre-de-entrada debe ser el mismo de PROGRAM-ID del subprograma llamado.

Formato 2

```
ENTER LINKAGE
ENTRY nombre-de-entrada [USING  nombre-de-dato  ...].
ENTER COBOL.
```

En este caso, si el punto de entrada del subprograma está indicado por una proposición ENTRY, el nombre-de-entrada del formato 1 debe ser el mismo que el del formato 2.

Los nombres-de-dato son los nombres de los campos de nivel 01 ó 77 especificados en la LINKAGE SECTION de este subprograma.

Formato 3

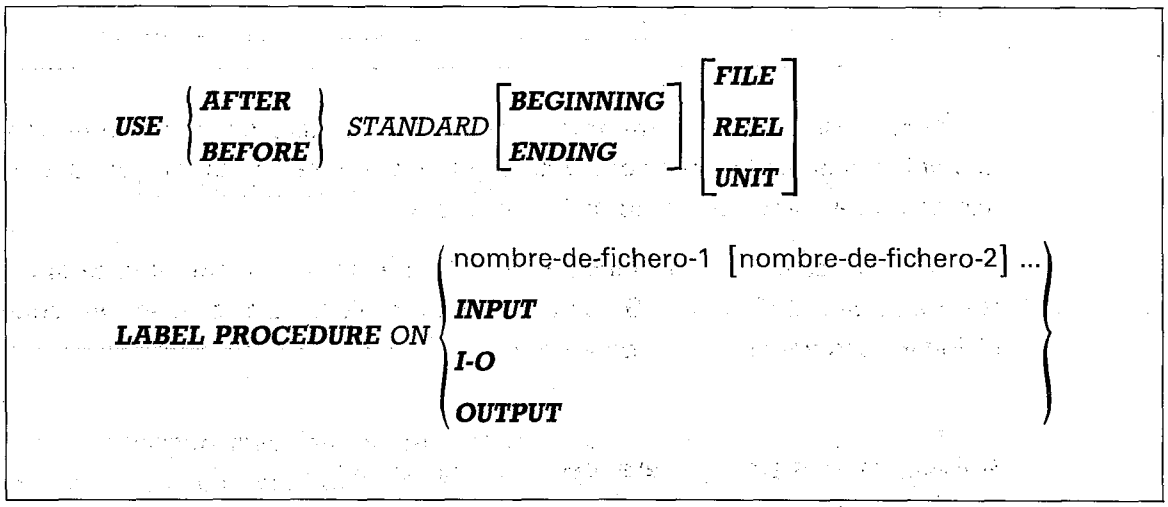
```
ENTER LINKAGE
{EXIT PROGRAM.}
{RETURN.      }
ENTER COBOL.
```

En el subprograma llamado, este formato devuelve el control al programa que llama. El control vuelve al programa principal para ejecutar la instrucción que sigue inmediatamente a la instrucción CALL.

Use

Esta proposición se emplea para especificar procedimientos especiales de manejo de errores y etiquetas de E/S, así como los procedimientos standard del sistema de E/S.

Formato 1



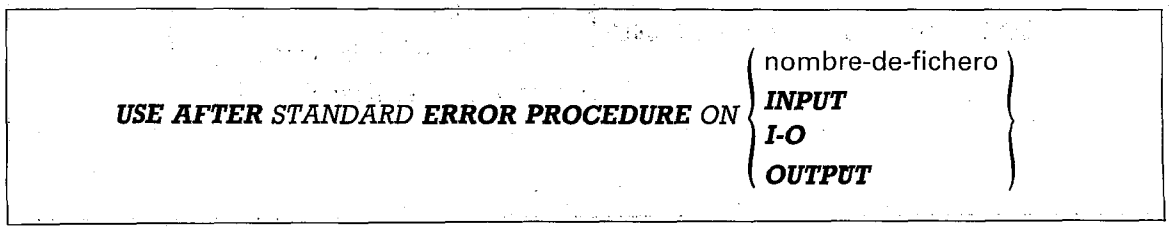
Si se utiliza la opción nombre-de-fichero, la anotación FD para nombre-de-fichero-1 debe contener la cláusula LABEL RECORDS ARE nombre-de-dato.

Si se omite la opción BEGINNING *(principio)* o ENDING *(final),* los procedimientos especificados se ejecutan para las etiquetas de principio y final. La opción ENDING no puede utilizarse en ficheros de acceso directo cuya organización no sea secuencial.

La opción REEL no puede utilizarse para memorias de gran capacidad.

La opción UNIT no puede utilizarse en ficheros de acceso al azar.

Formato 2



En este caso, el procedimiento USE se inicia al terminar los procedimientos standard de recuperación de errores de E/S.

Formato 3

***USE** FOR **FROM-OVERFLOW** ON* nombre-de-fichero

El control se transfiere al procedimiento USE al detectarse una condición de página-llena en la impresora. Esta condición se detecta durante las funciones de impresión y espaciado de la impresora.

Si se especifica el posicionamiento del impreso por medio de la cinta de control de carro (ADVANCING nombre-mnemotécnico), no se produce una condición de página-llena en la impresora.

4.6.8. Comunicaciones entre programas

Las comunicaciones entre programas se establecen mediante el verbo CALL. El punto de entrada en el programa llamado se especifica con el verbo ENTRY.

Call

La proposición CALL *(llamar)* se utiliza con el verbo ENTER, en el programa principal, para establecer una comunicación con el punto de entrada del subprograma. Su formato es:

CALL nombre-de-entrada [***USING*** { nombre-de-fichero / identificador / nombre-de-procedimiento / nombre-de-clasificación } ...]

Entry

Esta proposición se utiliza junto con ENTER en un programa llamado para establecer un punto de entrada.

Formato:

```
ENTRY nombre-de-entrada [USING nombre-de-dato...].
```

▶ Ejemplo de codificación

EUROPE COMPANY CORPORATION

HOJA DE CODIFICACION

PROGRAMADOR	PROGRAMA	N.º	FECHA	HOJA

```
1  PROCEDURE DIVISION.
2  COMIENZO.
3      OPEN INPUT FICHERO-ALUMNO
4                 FICHERO-FICHAS
5           UUTPUT FICHERO IMPRESION
6  PROCESO.
7      READ FICHERO-FICHAS AT END GO TO FINAL.
8      MOVE DATO-2 TO DATO SALIDA.
9      WRITE LINEA AFTER PAGE.
10 MENSAJE.
11     DISPLAY NUMERO-FICHA UPON CONSOLE.
12     GO TO PROCESO.
13 FINAL.
14     CLOSE FICHERO-ALUMNO
15           FICHERO-IMPRESION.
16     STOP RVM.
17 SVB-1.
18     READ FICHERO-ALUMNO INVALID KEY GO TO MENSAJE.
19     MOVE ALUMNO TO SALIDA.
20     PERFOM SUB-2.
21     ADD 1 TO CONTADOR.
22 SUB-2.
23     IF CODIGO = 7 COMPUTE NUM = COD * PREV.
24 SVB-2-1.
25     EXIT.
26
27
```

forieva, s.a. O-5-13.642

EJERCICIOS

1. Los nombres-de-dato de la cláusula USING referencian a campos descritos en el subprograma y sólo deben estar definidos en la:

   a) FILE SECTION.
   b) WORKING-STORAGE SECTION.
   c) LINKAGE SECTION.

2. Una proposición consta de:

   a) Una o más sentencias.
   b) Uno o más párrafos.
   c) Un verbo más una serie de palabras.

3. Las proposiciones imperativas:

   a) Indican acciones incondicionales que tiene que realizar el programa objeto.
   b) Indican que se va a determinar el valor verdadero de una condición y que la acción del programa objeto depende de esta condición.
   c) Dirigen al compilador para que efectúe ciertas acciones en la fase de compilación.

4. Los verbos de bifurcación son:

   a) MOVE, EXAMINE, SET.
   b) GO TO, PERFORM, EXIT, ALTER.
   c) IF, SEARCH.

5. Si se desea salir temporalmente de la secuencia normal de ejecución del programa para realizar uno o varios procedimientos, un número dado de veces, se emplea la proposición:

   a) GO TO.
   b) PERFORM.
   c) EXIT.

6. La proposición EXAMINE:

   a) Transfiere datos a una o más áreas.
   b) Establece puntos de referencia para manejo de tablas.
   c) Reemplaza el número de repeticiones de un carácter determinado en un campo.

SOLUCIONES DE LOS EJERCICIOS

1. Elementos del COBOL

1. *c);* 2. *c);* 3. *c);* 4. *c);* 5. *a);* 6. *c);* 7. *b);* 8. *c);* 9. *b);* 10. *a);* 11. *b).*

2. Identification y Environment Division.

1. *b);* 2. *b);* 3. *c);* 4. *a);* 5. *a);* 6. *b).*

3. Data Division

1. *a);* 2. *b);* 3. *a);* 4. *c);* 5. *b);* 6. *a);* 7. *b);* 8. *c).*

4. Procedure Division.

1. *c);* 2. *c);* 3. *a);* 4. *b);* 5. *b);* 6. *c).*

Indice general

Introducción 11

1. ELEMENTOS DEL COBOL

Cuadro resumen 15
1. Elementos del Cobol 16
1.1. Caracteres del Cobol 16
1.2. Palabras 17
1.2.1. Palabras reservadas 17
- Verbos 22
- Palabras clave 22
- Palabras opcionales 22
- Constantes figurativas 23
- Palabras conectores 24

1.2.2. Palabras creadas por el programador 24
- Nombres de datos 24
- Nombres de párrafos 25
- Constantes literales 26

1.3. Expresiones 28
- Expresiones aritméticas 28
- Expresiones de relación 29
- Expresiones lógicas 30

1.4. Hojas de codificación COBOL 30
1.4.1. Reglas de codificación 32
1.5. Formato de las instrucciones 33

Ejercicios 34

2. IDENTIFICATION DIVISION. ENVIRONMENT DIVISION

Cuadro resumen 39
2. Divisiones de un programa COBOL 40
2.1. Identification Division 41
2.2. Environment Division 44
2.2.1. Configuration section 45
- Source-Computer 45
- Object-Computer 45
- Special-Names 46

2.2.2. Input-Output Section 47
- File-Control 48
- I-O-Control 55

Ejercicios 59

3. DATA DIVISION

Cuadro resumen 63
3. Data division 64
3.1. File Section 65
3.1.1. Descripción de Fichero 66
- Cláusula Label Record 67
- Cláusula Recording Mode 67
- Cláusula Block Contains 68
- Cláusula Record Contains 69
- Cláusula Data Record 69
- Descripción de fichero de clasificación (SD) 70

3.1.2. Descripción de los datos 70
- Cláusula «número-de-nivel» 72
- Cláusula Redefines 73
- Cláusula Occurs 73
- Cláusula Picture 77
- Cláusula USAGE 83
- Cláusula Justified 86
- Cláusula Value 86
- Cláusula Blank When Zero 88
- Cláusula Renames 88
- Cláusula Nombre-de-condición 88

3.2. Working-Storage Section 89
3.3. Linkage Section 90
3.4. Screen Section 91

Ejercicios 93

4. PROCEDURE DIVISION

Cuadro Resumen 97
4. Procedure Division 98
4.1. Proposición Using 98
4.2. Secciones 99
4.3. Párrafos 99
4.4. Sentencias 100
4.5. Proposiciones 100
4.6. Verbos 101
4.6.1. Verbos Aritméticos 101
- Add 102
- Subtract 104
- Multiply 105
- Divide 106
- Compute 107

4.6.2. Verbos de bifurcación 109
- Go To 110
- Perform 111
- Exit 115
- Alter 115

4.6.3. Verbos de transferencia de datos 116
- Move 116
- Examine 118
- Set 119

4.6.4. Verbos de entrada/salida 121
- Accept 121
- Display 122
- Open 122
- Close 123
- Read 124
- Write 125
- Insert 126
- Rewrite 126

– Sort 127
– Return 128
– Seek 128
4.6.5. Verbo de final (Stop) 129
4.6.6. Verbos condicionales 130
– If 130
– Search 134
4.6.7. Verbos directivos del compilador 136
– Note 136
– Copy 137
– Enter 138
– Use 140
4.6.8. Comunicaciones entre programas 141
– Call 141
– Entry 142
Ejercicios 144
Soluciones de los ejercicios 147

Este volumen expone los fundamentos y desarrollo así como las utilidades del lenguaje de programación, COBOL.

El COBOL es uno de los lenguajes más utilizados en las técnicas del Proceso de Datos.

El dominio del COBOL proporciona al programador un extenso campo de actuación, dado que es un lenguaje de utilización universal.

Los tomos destinados al estudio del COBOL, dentro de la obra «PROGRAMACION DE ORDENADORES», proporcionan al lector un completo conocimiento del mismo.

El cuadro resumen y los ejercicios que se incluyen al principio y final de cada uno de los temas, son elementos didácticos que favorecen y ayudan en la correcta asimilación de los conceptos desarrollados.

El dibujo de la cubierta es cortesía de GENERAL DE COMPUTADORES, S.A.