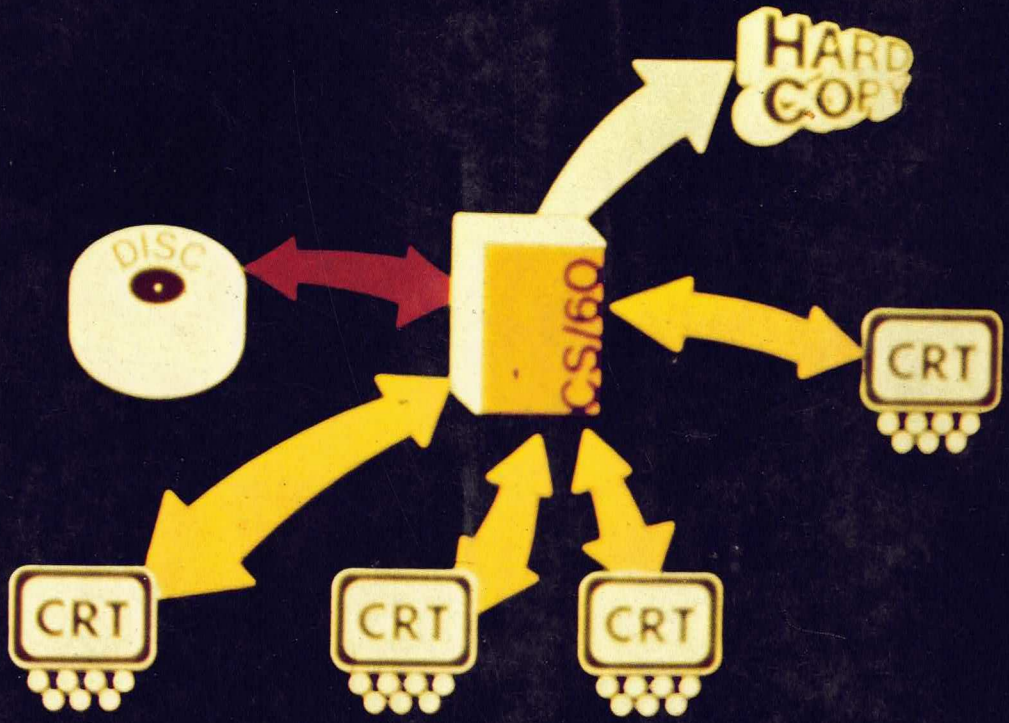
PROGRAMACION DE ORDENADORES

LENGUAJE COBOL



EDITORIAL tesys

TOMO 8

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: Septiembre 1981

Depósito Legal: B. 25.869-1981 (VIII)

ISBN: 84-85838-00-9 (Obra completa)
ISBN: 84-85838-08-4 (Tomo 8)

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

Impreso en España - Printed in Spain

CONTENIDO

1. Manejo de tablas 9

2. Máscaras 51

3. Tratamiento de ficheros 85

4. Técnicas de optimización 141

Soluciones de los ejercicios 171

Indice general 173

1.

MANEJO DE TABLAS

CUADRO RESUMEN DE LA LECCION

Definición de una tabla

- Manejo de Tablas
  - — Subíndices
  - — Indices
- Referencia a un elemento
- Redefinición de tablas
- Definición de índices fuera de la tabla
- Modificación de índices
- Exploración de una tabla

1. MANEJO DE TABLAS

Una tabla es un conjunto de datos homogéneos agrupados bajo un mismo nombre, que permite un fácil y rápido acceso a cualquiera de ellos, independientemente de su posición dentro de la tabla.

Los datos, llamados *elementos,* están dispuestos de manera continua y en un orden lógico y significativo.

Ejemplo:

SEMANA
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo

En este ejemplo, la tabla cuyo nombre es SEMANA agrupa a siete elementos (los siete días de una semana del calendario).

1.1. DEFINICION DE UNA TABLA

Una tabla se define mediante la inclusión de la cláusula OCCURS en la definición de un nombre-de-campo dentro de la DATA DIVISION.

La utilización de esta cláusula elimina la necesidad de codificar por separado varios campos con las mismas características.

Su formato es:

OCCURS n TIMES

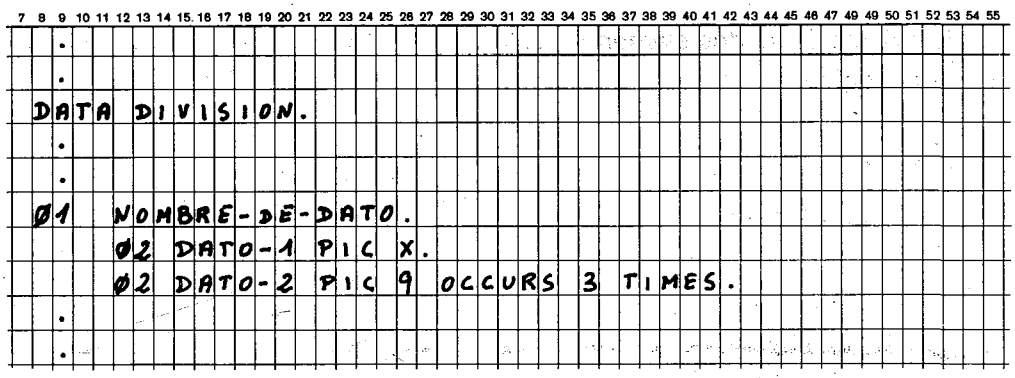
siendo:

OCCURS	= palabra reservada de uso obligatorio
n	= cantidad de elementos que son idénticos
TIMES	= palabra reservada opcional

Esta cláusula especifica la cantidad de elementos que se agruparán bajo un mismo nombre.

Una tabla definida de esta manera, se llama *tabla uni-dimensional* (tabla con una dimensión) o también *Vector.*

Ejemplo-1

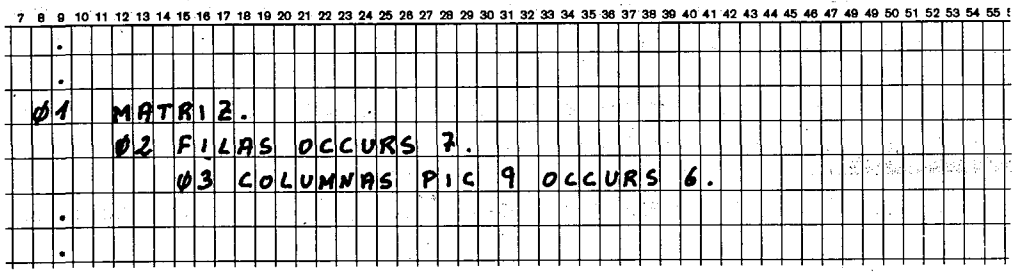


En esta codificación, el campo NOMBRE-DE-DATO se subdivide en cuatro campos:

— Un campo con el nombre DATO-1.
— Tres campos bajo el nombre DATO-2.

Si se define una tabla (mediante la cláusula OCCURS) sobre un elemento perteneciente a una tabla, se obtiene una *tabla bidimensional* (tabla con dos dimensiones) o *Matriz.*

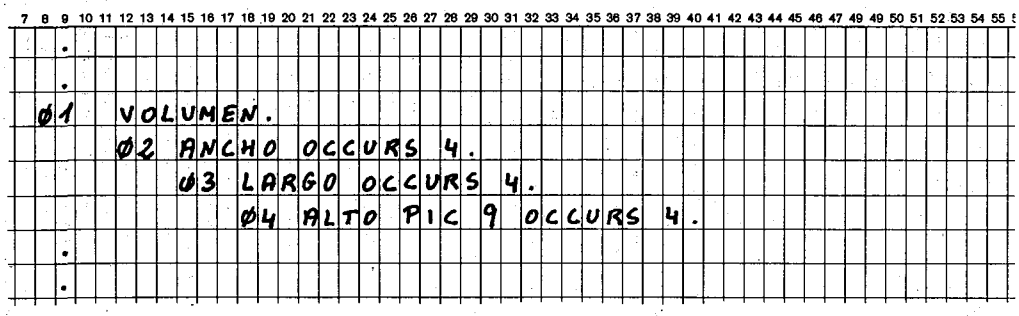
Ejemplo-2



En este ejemplo, se define un campo con el nombre MATRIZ subdividido en siete campos bajo el nombre FILAS. A su vez, cada elemento de la tabla FILAS se subdivide en 6 campos bajo el nombre COLUMNAS. Por tanto, el campo MATRIZ se subdivide en un total de 42 elementos.

Las *tablas tridimensionales* (tablas de tres dimensiones) se logran definiendo una tabla sobre un elemento perteneciente a una tabla que a su vez sea un elemento de una tabla.

Ejemplo-3



Con este ejemplo, se define un campo con el nombre VOLUMEN subdividido en cuatro campos bajo el nombre ANCHO. Cada elemento ANCHO se subdivide a su vez en cuatro elementos bajo el nombre LARGO. Y cada elemento de la tabla LARGO se subdivide en cuatro elementos bajo el nombre ALTO. Por tanto, el campo VOLUMEN se subdivide en un total de 64 elementos.

Una tabla puede tener un máximo de tres dimensiones.

La única limitación en la cantidad de elementos de una tabla la establece la capacidad de la memoria interna del ordenador empleado.

1.2. REFERENCIA A UN ELEMENTO

Para hacer referencia a un elemento de una tabla, se especifica la posición que ocupa dentro de la tabla.

Esta referencia se hace poniendo entre paréntesis y a continuación del nombre del elemento, la posición que ocupa dentro de la tabla.

La posición de un elemento dentro de una tabla se puede indicar mediante:

— Subíndices.
— Indices.

1.2.1. Subíndices

Los subíndices son constantes numéricas que indican la posición de un elemento dentro de una tabla.

Ejemplo-1:

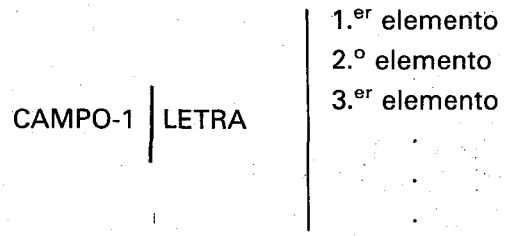
```
DATA DIVISION.

01   CAMPO-1.
     02 LETRA PIC A OCCURS 26.

PROCEDURE DIVISION.

     MOVE "A" TO LETRA (1).
     MOVE "B" TO LETRA (2).
     MOVE "C" TO LETRA (3).
```

En la DATA DIVISION se crea un campo con el nombre CAMPO-1 que se divide en veintiséis campos bajo el nombre LETRA, con la siguiente estructura:



En este caso, LETRA es el nombre de cada uno de los veintiséis elementos.

Para hacer referencia al primer elemento LETRA se codifica:

LETRA (1)

Para hacer referencia al segundo elemento LETRA se codifica:

LETRA (2)

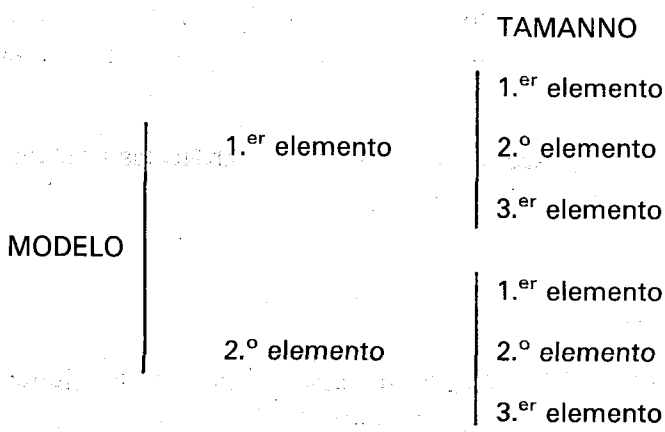
y así sucesivamente para cada uno de los elementos.

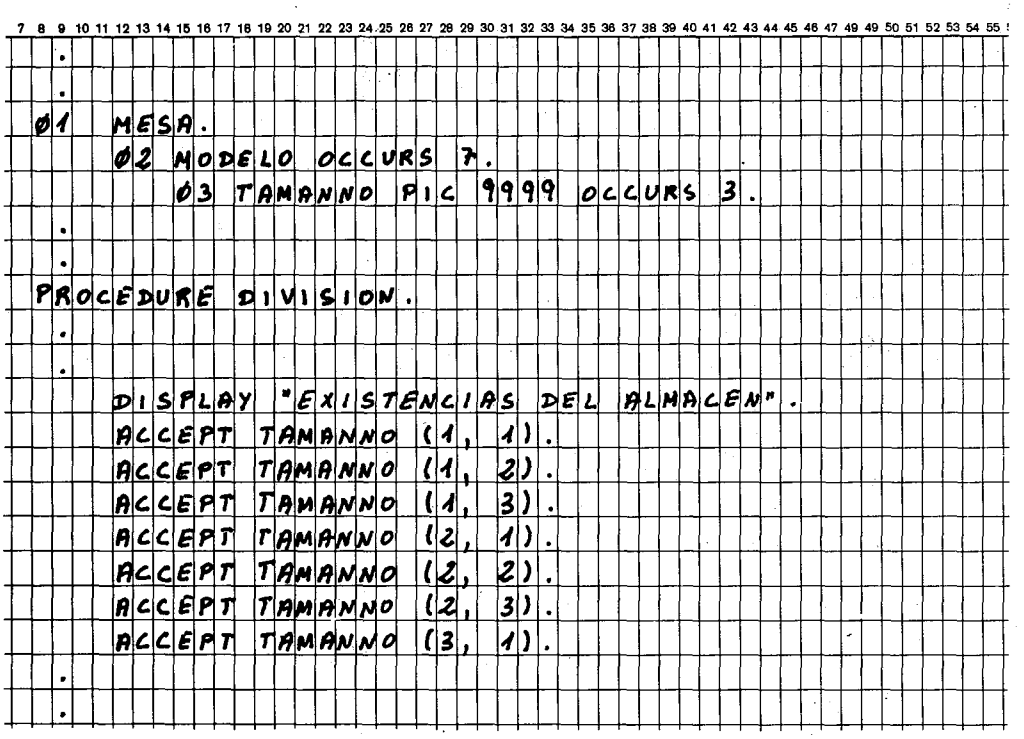
En este programa, se asigna a cada elemento LETRA un carácter alfabético, resultando que:

LETRA (1) = A
LETRA (2) = B
LETRA (3) = C

Ejemplo-2:

En él, se define MODELO con siete elementos. Cada uno de estos elementos se divide a su vez en tres elementos. Por lo tanto, MODELO posee veintiún elementos con la siguiente estructura:





```
01   MESA.
     02 MODELO OCCURS 7.
        03 TAMANNO PIC 9999 OCCURS 3.
.
.
PROCEDURE DIVISION.
.
.
     DISPLAY "EXISTENCIAS DEL ALMACEN".
     ACCEPT TAMANNO (1, 1).
     ACCEPT TAMANNO (1, 2).
     ACCEPT TAMANNO (1, 3).
     ACCEPT TAMANNO (2, 1).
     ACCEPT TAMANNO (2, 2).
     ACCEPT TAMANNO (2, 3).
     ACCEPT TAMANNO (3, 1).
.
.
```

Para hacer referencia a un elemento de esta tabla bidimensional, se utilizan dos subíndices. El primero de ellos indica el elemento MODELO y el segundo indica el elemento TAMANNO correspondientes.

Si:

```
01 MESA
  02 MODELO
    03 TAMANNO
```

es la definición de uno de los artículos que trata el programa, cada mesa se distingue de las demás por el modelo al que pertenece y por su tamaño dentro de un modelo concreto.

Este programa almacena en la matriz TAMANNO las existencias de mesas del almacén según el modelo y el tamaño.

Por tanto,

TAMANNO (3, 2)

hace referencia a las mesas que posee el almacén del segundo tamaño del tercer modelo.

Reglas de los subíndices:

— El valor de un subíndice ha de ser un número entero sin signo comprendido entre 1 y la cantidad de elementos que posea la dimensión subindicada.

— Los subíndices se especifican entre paréntesis. El paréntesis izquierdo debe ir precedido al menos por un espacio en blanco y no debe ir seguido de ningún espacio en blanco. El paréntesis derecho no debe ir precedido por ningún espacio en blanco.

— Si la referencia a un elemento de una tabla requiere dos o tres subíndices, deben separarse por comas y al menos un espacio en blanco.

Ejemplo-3

```
01   P-1.
     02 P-2 OCCURS 4.
        03 P-3 PIC 9 OCCURS 2.

PROCEDURE DIVISION.

     ADD P-3(4, 2) TO TOTAL.
     MOVE P-3 (1) TO PARCIAL.
     ADD P-3 (4, 3) TO INCREMENTO.
     ADD P-3 (-1, 2) TO CONCEPTO.
     MOVE P-3 (3,1) TO CUENTA.
     ADD P-3 (3, 1, 7) TO GLOBAL.
```

Esta codificación es incorrecta porque:

— ADD P-3(4, 2) TO TOTAL

Entre P-3 y (4, 2) debe haber por lo menos un espacio en blanco.

— MOVE P-3 (1) TO PARCIAL

Falta especificar un subíndice.

— ADD P-3 (4, 3) TO INCREMENTO

El segundo subíndice sobrepasa el valor máximo de la dimensión que subindica en la matriz P-2.

— ADD P-3 (−1, 2) TO CONCEPTO

El primer subíndice no es un número entero sin signo.

— MOVE P-3 (3, 1) TO CUENTA

Después de la coma (que separa el primer del segundo subíndice) y el segundo subíndice falta por lo menos un espacio en blanco.

— ADD P-3 (3, 1, 7) TO GLOBAL

Sobra un subíndice.

1.2.2. Indices

Los índices son nombres-de-campo de tipo numérico en los que se almacena la posición del elemento de una tabla al que se hace referencia.

Los índices no deben estar definidos dentro de la WORKING-STORAGE SECTION ya que es el propio compilador el que los define. A la definición de tabla mediante la cláusula OCCURS, se le añadirá la cláusula INDEXED BY. Esta cláusula especifica cuál será el índice perteneciente a una determinada dimensión de una tabla.

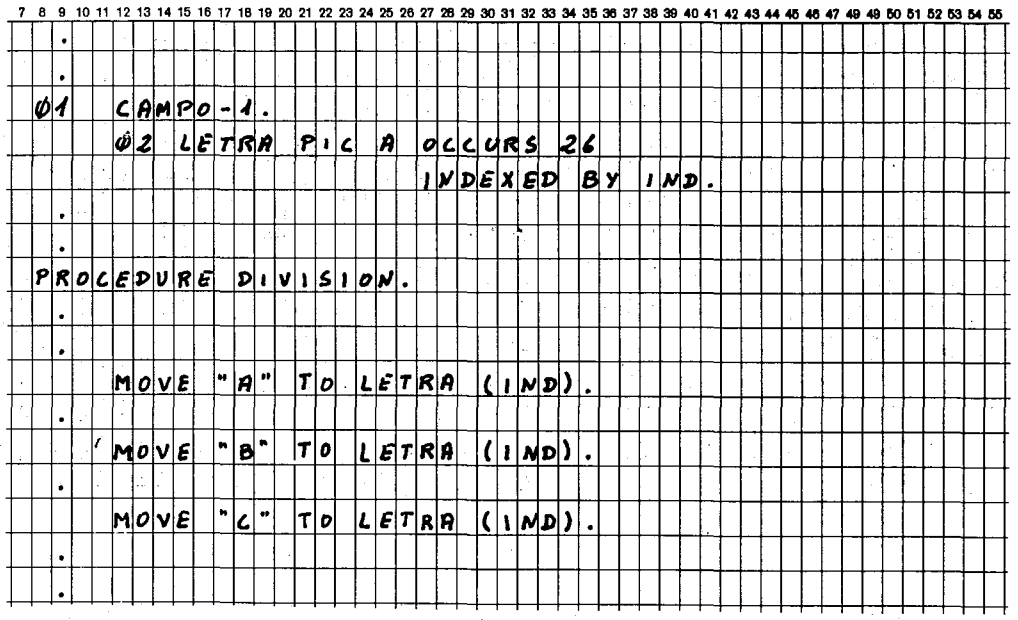
Ejemplo-1

```
01  NOMBRE-DE-DATO.
    02 DATO-1 PIC X.
    02 DATO-2 PIC 9 OCCURS 294
                    INDEXED BY IND.
```

En la definición de tabla del campo DATO-2 se incluye la cláusula INDEXED BY IND, que indica que la indexación se realizará mediante el valor almacenado

en el campo IND, campo que definirá el compilador y se utilizará para la indexación de los elementos de la tabla.

Ejemplo-2



Este ejemplo logra la misma finalidad que el ejemplo 1 del apartado «Subíndices».

Si al ejecutarse la instrucción:

MOVE "A" TO LETRA (IND)

el valor de IND es uno, se pone el carácter A en el primer elemento LETRA.

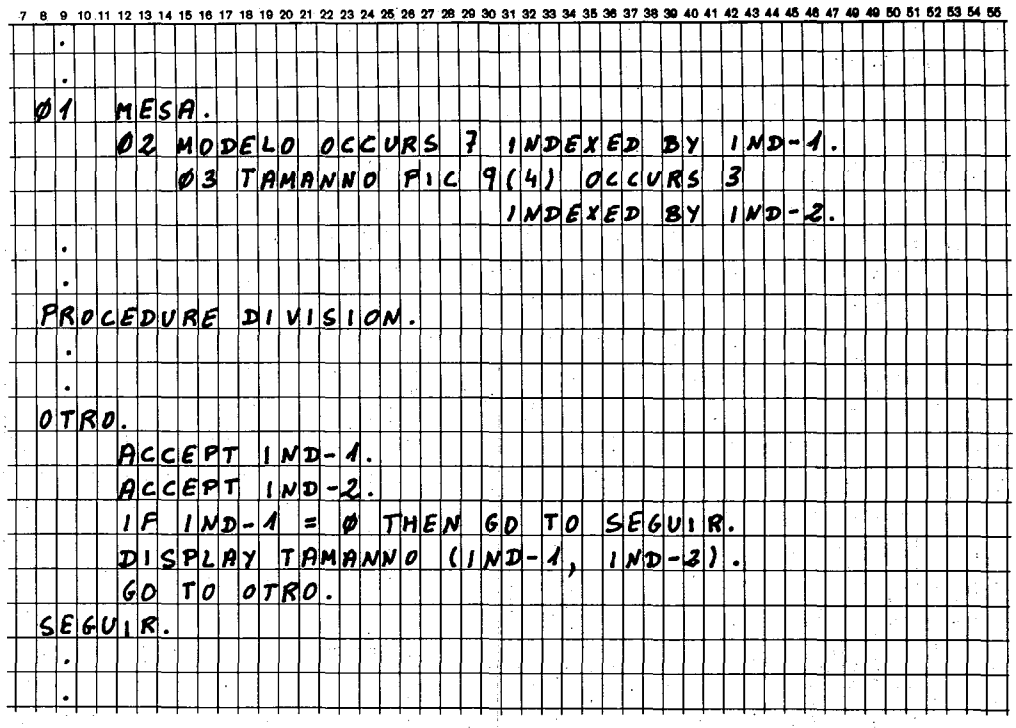
La instrucción

MOVE "B" TO LETRA (IND)

pone el carácter B en el segundo elemento LETRA si el valor de IND es dos. Y así sucesivamente.

Ejemplo-3

Al igual que en el ejemplo 2 del apartado «Subíndices», en este ejemplo se contempla una tabla para la clasificación de las mesas según el modelo y su tamaño.



```
01  MESA.
    02 MODELO OCCURS 7 INDEXED BY IND-1.
       03 TAMANNO PIC 9(4) OCCURS 3
                   INDEXED BY IND-2.
.
.
PROCEDURE DIVISION.
.
.
OTRO.
    ACCEPT IND-1.
    ACCEPT IND-2.
    IF IND-1 = 0 THEN GO TO SEGUIR.
    DISPLAY TAMANNO (IND-1, IND-2).
    GO TO OTRO.
SEGUIR.
.
.
```

En este ejemplo, el modelo de mesa es indexado por IND-1 y el tamaño por IND-2.

El funcionamiento de las instrucciones contenidas en el párrafo OTRO es el siguiente:

Las instrucciones

```
ACCEPT IND-1
ACCEPT IND-2
```

solicitan al operador cuál es el modelo (IND-1) y el tamaño (IND-2) de la mesa.

Con la instrucción

```
IF IND-1 = 0 THEN GO TO SEGUIR
```

cuando el modelo de la mesa sea el cero, se dan por finalizadas las consultas de las existencias del almacén y se continúa en secuencia a partir del párrafo SEGUIR.

La instrucción

```
DISPLAY TAMANNO (IND-1, IND-2)
```

displaya en la consola del operador el número almacenado en el elemento TAMANNO correspondiente al modelo indicado por el valor de IND-1 y el tamaño indicado por el valor de IND-2.

La indexación de un elemento de una tabla puede hacerse de dos maneras:

— Indexación directa.
— Indexación relativa.

Todos los ejemplos vistos hasta ahora sobre indexación, son del tipo indexación directa.

En la indexación relativa, al índice de la tabla se le suma o resta un número entero distinto de cero.

Ejemplo-4

```
01  CAMPO-1.
    02 DATO PIC 9(5) OCCURS 58
                    INDEXED BY IND-1.

PROCEDURE DIVISION.

    ADD DATO (IND-1) TO DATO (IND-1 + 10).
```

Se define una tabla de una sola dimensión con el nombre DATO que posee 58 elementos indexados por el campo IND-1.

Si el valor del campo IND-1 es 17, con la instrucción:

ADD DATO (IND-1) TO DATO (IND-1 + 10)

se suma el valor del elemento DATO (17) (con indexación directa) al elemento DATO (27) (con indexación relativa IND-1 + 10).

Las reglas de los índices son:

— Cada índice no debe estar definido dentro de la DATA DIVISION

— A la cláusula OCCURS se le añadirá INDEXED BY y a continuación el nombre del campo que se utilizará para la indexación.

— Los índices deben ir codificados entre paréntesis y separados por comas (si hay más de uno) en la misma manera que los subíndices.

— Un índice no puede ser un elemento de una tabla.

— El valor de un índice en indexación directa debe estar comprendido entre 1 y el valor de la dimensión que indexa dentro de la tabla.

— En indexación relativa, el valor resultado de combinar el índice y el entero que le acompaña (sumado o restado) debe mantenerse entre 1 y el valor de la dimensión indexada.

Ejemplos-5 y 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 49 50 51 52 53 54 55
                .
                .
    01    LINEA-1.
          02 CAM PIC 9 OCCURS 4
                       INDEXED BY IND (1).
    01    LINEA-2.
          02 CAMP PIC 99 OCCURS 7
                       INDEXED BY IND (2).
                .
                .
```

```
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 49 50 51 52 53 54 55
                .
                .
    01    ARMARIO.
          02 TIPO PIC X(20) OCCURS 74.
                .
                .
PROCEDURE DIVISION.
                .
                .
          DISPLAY TIPO (IND-4).
                .
                .
```

En el primer ejemplo la codificación es incorrecta ya que el índice de la tabla CAM y el índice de la tabla CAMP son elementos de una tabla.

En el otro ejemplo falta la cláusula INDEXED BY en la definición de la tabla TIPO, debiendo quedar así:

```
01 ARMARIO
   02 TIPO PIC X(2Ø) OCCURS 74 INDEXED BY IND-4
```

1.3. REDEFINICION DE TABLAS

La principal utilidad del uso de descripciones de campos redefinidos sobre la descripción de una tabla, consiste en la obtención del dato asociado, de una manera lógica, con su subíndice.

Ejemplo-1

```
01  MES-1.
    02  FILLER PIC X(10)  VALUE "ENERO".
    02  FILLER PIC X(10)  VALUE "FEBRERO".
    02  FILLER PIC X(10)  VALUE "MARZO".
    02  FILLER PIC X(10)  VALUE "ABRIL".
    02  FILLER PIC X(10)  VALUE "MAYO".
    02  FILLER PIC X(10)  VALUE "JUNIO".
    02  FILLER PIC X(10)  VALUE "JULIO".
    02  FILLER PIC X(10)  VALUE "AGOSTO".
    02  FILLER PIC X(10)  VALUE "SEPTIEMBRE".
    02  FILLER PIC X(10)  VALUE "OCTUBRE".
    02  FILLER PIC X(10)  VALUE "NOVIEMBRE".
    02  FILLER PIC X(10)  VALUE "DICIEMBRE".
01  MES-2 REDEFINES MES-1.
    02 MES PIC X(10)  OCCURS 12
                     INDEXED BY NUM.

PROCEDURE DIVISION.

    DISPLAY "QUE MES?".
    ACCEPT NUM.
    DISPLAY MES (NUM).
```

En este programa se define un campo con el nombre MES-1 que se subdivide en doce campos. Cada uno de ellos contiene el nombre de un mes y están codificados de una forma lógica y racional. El campo MES-2 redefine al campo MES-1 mediante la cláusula REDEFINES. Este campo se subdivide en doce campos iguales bajo el nombre MES mediante la cláusula OCCURS 12 que se indexan mediante NUM.

Como los campos en que se subdividen los campos MES-1 y MES-2 son de las mismas características, el dato almacenado en el primer elemento MES es Enero, el dato almacenado en el segundo elemento es Febrero y así sucesivamente con los doce elementos de MES.

Al ejecutarse la instrucción:

```
DISPLAY "QUE MES?"
```

el operador sabrá que debe introducir el número del mes que desea conocer.

Con la instrucción:

```
ACCEPT NUM
```

el operador introduce el número del mes y el ordenador lo almacena en el campo NUM.

Con la instrucción:

```
DISPLAY MES (NUM)
```

el operador obtendrá el mes correspondiente al número introducido de acuerdo a la siguiente tabla:

1 – Enero	5 – Mayo	9 – Septiembre
2 – Febrero	6 – Junio	10 – Octubre
3 – Marzo	7 – Julio	11 – Noviembre
4 – Abril	8 – Agosto	12 – Diciembre

Por ejemplo: si introdujo el número 5, el campo NUM tomará el valor cinco y con la instrucción

```
DISPLAY MES (NUM)
```

obtendrá el quinto elemento MES, como MES-2 está redefinido sobre los nombres de cada uno de los meses, displayará el quinto campo en el que se subdivide el campo MES-1. En este caso obtendrá el dato Mayo.

Ejemplo-2

En este ejemplo se define el campo PROD-1 subdividido en 15 campos, los cuales contienen la descripción de cada uno de los productos que trata este programa. Hay cinco productos:

Jamón
Ternera
Cerdo
Pollo
Conejo

y de cada producto existen tres calidades.

El campo PROD-2 redefine el campo PROD-1, por lo tanto, comparten la misma zona de memoria. El campo PROD-2 se subdivide en cinco campos bajo

```
01  PROD-1.
    02 FILLER PIC X(19) VALUE "JAMON DE JABUGO".
    02 FILLER PIC X(19) VALUE "JAMON SUPERIOR".
    02 FILLER PIC X(19) VALUE "JAMON DE CONSUMO".
    02 FILLER PIC X(19) VALUE "TERNERA (SOLOMILLO)".
    02 FILLER PIC X(19) VALUE "TERNERA (COSTILLAS)".
    02 FILLER PIC X(19) VALUE "TERNERA (MAGRA)".
    02 FILLER PIC X(19) VALUE "CERDO (SOLOMILLO)".
    02 FILLER PIC X(19) VALUE "CERDO (COSTILLAS)".
    02 FILLER PIC X(19) VALUE "CERDO (MAGRA)".
    02 FILLER PIC X(19) VALUE "POLLO (GRANJA)".
    02 FILLER PIC X(19) VALUE "POLLO (CRIADERO)".
    02 FILLER PIC X(19) VALUE "GALLINA".
    02 FILLER PIC X(19) VALUE "CONEJO EXTRA".
    02 FILLER PIC X(19) VALUE "CONEJO SUPERIOR".
    02 FILLER PIC X(19) VALUE "CONEJO DE CONSUMO".
01  PROD-2 REDEFINES PROD-1.
    02 PRODUCTO OCCURS 5 INDEXED BY PROD.
       03 CALIDAD PIC X(19) OCCURS 3
                            INDEXED BY CALI.
01  EXISTENCIAS.
    02 PRODUC OCCURS 5 INDEXED BY PROD.
       03 CALID PIC 9(6)V99 OCCURS 3
                            INDEXED BY CALI.
01  CANTIDAD PIC 9(6)V99.
```

```
PROCEDURE DIVISION.
.
.
OTRO.
    DISPLAY "CANTIDAD(6.2), PRODUCTO(1-5)"
-           ", CALIDAD(1-3)".
    PERFORM OTRO-CEROS
            VARYING PROD FROM 1
                    UNTIL PROD > 5
            AFTER CALI FROM 1
                  UNTIL CALI > 3.
OTRO-1.
    ACCEPT CANTIDAD.
    ACCEPT PROD.
    ACCEPT CALI.
    IF CANTIDAD = 0 THEN GO TO OTRO-FIN.
    ADD CANTIDAD TO CALID (PROD, CALI).
    GO TO OTRO-1.
OTRO-FIN.
    PERFORM OTRO-SALIDA
            VARYING PROD FROM 1
                    UNTIL PROD > 5
            AFTER CALI FROM 1
                  UNTIL CALI > 3.
SIGUIENTE.
```

```
.
.
OTRO-CEROS.
    MOVE ZEROES TO CALID (PROD, CALI).
OTRO-SALIDA.
    DISPLAY CALIDAD (PROD, CALI).
    DISPLAY CALID (PROD, CALI).
FIN-DEL-PROGRAMA.
    STOP-RUN.
```

el nombre PRODUCTO indexados por PROD, y cada uno de ellos se subdivide a su vez en tres campos de nombre CALIDAD indexados por CALI.

Otro campo es CANTIDAD, que contendrá los kilos pedidos de uno de los quince productos.

Y finalmente, el campo EXISTENCIAS, subdividido en cinco campos bajo el nombre PRODUC que a su vez se subdividen en tres campos.

Cada uno de los quince elementos de la tabla PRODUC almacena los kilos vendidos por producto.

Todas las tablas utilizan PROD para indexar la primera dimensión (tipo de producto) y CALI para indexar la segunda dimensión (calidad del tipo de producto). Esto facilita la búsqueda del elemento correspondiente a un producto determinado en la tabla de existencias y en la tabla de descripción de productos (en PROD-1 mediante la tabla PROD-2).

El funcionamiento de las instrucciones pertenecientes a los párrafos OTRO, OTRO-1, OTRO-FIN, OTRO-CEROS y OTRO-SALIDA es el siguiente:

La instrucción:

```
DISPLAY "CANTIDAD (6.2), PRODUCTO (1-5), CALIDAD (1-3)"
```

indica al operador el orden en el que debe introducir los datos y el valor que pueden tener.

(6-2) = Cuatro enteros y dos decimales.
(1-5) = Un entero comprendido entre uno y cinco.
(1-3) = Un entero comprendido entre uno y tres.

La instrucción

```
PERFORM OTRO-CEROS VARYNG PROD FROM 1 UNTIL PROD > 5
                                AFTER CALI FROM 1 UNTIL CALI > 3
```

ejecuta las instrucciones contenidas en el párrafo OTRO-CEROS variando PROD desde uno hasta cinco y CALI desde uno hasta tres.

Con:

```
MOVE ZEROES TO CALID (PROD, CALI)
```

se pone a ceros el elemento CALID de la tabla PRODUC que indique el valor de los índices PROD y CALI.

Las instrucciones:

```
ACCEPT CANTIDAD
ACCEPT PROD
ACCEPT CALI
```

piden al operador los kilos vendidos de un producto definido por el tipo de producto y la calidad de dicho producto.

Mediante la instrucción:

```
IF CANTIDAD = 0 THEN GO TO OTRO-FIN
```

se comprueba que el valor asignado al campo CANTIDAD sea distinto de cero, porque cuando el operador quiere dar por finalizada la introducción de los kilos vendidos por producto, lo indicará (en este caso) diciendo que la cantidad vendida es cero y el control del procesador seguirá en secuencia con las instrucciones del párrafo OTRO-FIN.

La instrucción:

```
ADD CANTIDAD TO CALI (PROD, CALI)
```

suma al elemento definido por los índices PROD y CALI de la tabla PRODUC el valor almacenado en el campo CANTIDAD.

Con la instrucción:

```
GO TO OTRO-1
```

el control del procesador vuelve a la instrucción

```
ACCEPT CANTIDAD
```

La instrucción:

```
PERFORM OTRO-SALIDA VARYNG PROD FROM 1 UNTIL PROD > 5
                                AFTER CALI FROM 1 UNTIL CALI > 3
```

ejecuta las instrucciones contenidas en el párrafo OTRO-SALIDA variando PROD desde uno hasta cinco y CALI desde uno hasta tres.

Las instrucciones:

```
DISPLAY CALIDAD (PROD, CALI)
DISPLAY CALID (PROD, CALI)
```

displayan por la consola del operador los elementos CALIDAD Y CALID indicados por el valor de los índices PROD y CALI.

El funcionamiento del programa es el siguiente: Cuando el control del procesador ejecuta la instrucción:

```
DISPLAY "CANTIDAD (6.2), PRODUCTO (1-5), CALIDAD (1-3)"
```

en la pantalla del operador aparece:

```
CANTIDAD (6.2), PRODUCTO (1-5), CALIDAD (1-3)
```

Cuando se ejecuta la instrucción:

```
PERFORM OTRO-CEROS VARYNG PROD FROM 1 UNTIL PROD > 5
                  AFTER CALI FROM 1 UNTIL CALI > 3
```

se transfiere el control del procesador a la instrucción del párrafo OTRO-CEROS (quince veces) variando PROD desde uno hasta cinco y CALI desde uno hasta tres de la siguiente manera:

	PROD.	*CALI.*
1.ª	1	1
2.ª	1	2
3.ª	1	3
4.ª	2	1
5.ª	2	2
6.ª	2	3
7.ª	3	1
8.ª	3	2
9.ª	3	3
10.ª	4	1
11.ª	4	2
12.ª	4	3
13.ª	5	1
14.ª	5	2
15.ª	5	3

la instrucción del párrafo OTRO-CEROS

```
MOVE ZEROES TO CALID (PROD, CALI)
```

pondrá a ceros los elementos CALID a medida que sea ejecutada mediante la instrucción

```
PERFORM VARYNG PROD FROM 1 UNTIL PROD > 5
               AFTER CALI FROM 1 UNTIL CALI > 3
```

Las instrucciones del párrafo OTRO-1

```
ACCEPT CANTIDAD
ACCEPT PROD
ACCEPT CALI
```

almacenan en el campo CANTIDAD los kilos que indique el operador y en los índices PROD y CALI las posiciones que ocupa dentro de las tablas de los productos PROD-2 y PRODUC el producto al que hace referencia.

Si el valor introducido en el campo CANTIDAD es distinto de cero, el procesador ignorará la bifurcación codificada en la instrucción:

```
IF CANTIDAD = 0 THEN GO TO OTRO-FIN
```

y continuará en secuencia.

La siguiente instrucción que se ejecutará si el valor CANTIDAD es distinto de cero:

```
ADD CANTIDAD TO CALID (PROD, CALI)
```

suma al valor ya existente en el elemento de la tabla PRODUC, indexado por el valor de los índices PROD y CALI, el valor almacenado en CANTIDAD.

A continuación se ejecutará la instrucción:

```
GO TO OTRO-1
```

que devuelve el control del procesador a la primera instrucción del bucle destinado al procesamiento de los kilos vendidos por cada producto.

La salida de este bucle se logra cuando el valor introducido en CANTIDAD es cero. Dicha salida es hacia el párrafo OTRO-FIN.

En este párrafo, la primera instrucción:

```
PERFORM OTRO-SALIDA VARYNG PROD FROM 1 UNTIL PROD > 5
                                AFTER CALI FROM 1 UNTIL CALI > 3
```

transfiere el control del procesador quince veces al párrafo OTRO-SALIDA variando PROD desde uno hasta cinco y CALI desde uno hasta tres.

Las instrucciones del párrafo OTRO-SALIDA:

```
DISPLAY PRO-2 (PROD, CALI)
DISPLAY PRODUC (PROD, CALI)
```

imprimen en la consola del operador la descripción de cada uno de los quince productos y el total de kilos vendidos por cada uno de ellos.

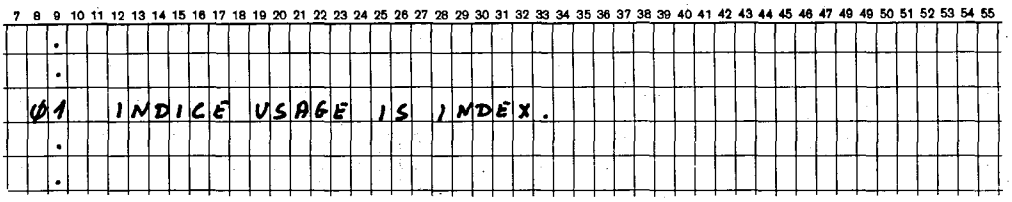
Una vez que estas instrucciones han sido ejecutadas quince veces, el procesador pasa a ejecutar las instrucciones del párrafo SIGUIENTE.

1.4. DEFINICION DE INDICES FUERA DE LA TABLA

Esta cláusula acompañará a la definición de un campo dentro de la WORKING-STORAGE SECTION siempre que dicho campo esté destinado a ser utilizado como índice de cualquier tabla en el programa.

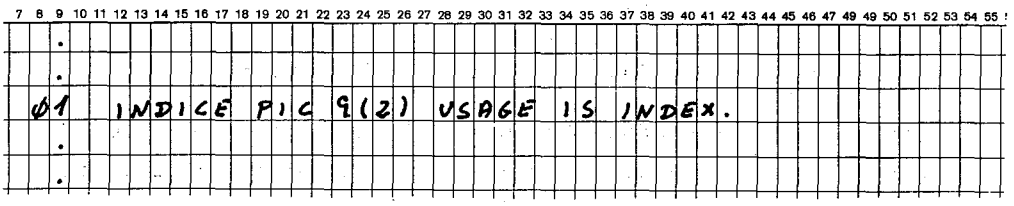
Esta cláusula es exclusiva, es decir, debe ser la única cláusula que contenga la descripción de dicho campo.

Ejemplo-1



En este ejemplo, el campo de nombre INDICE será utilizado únicamente para la indexación de una tabla.

Ejemplo-2



En este caso, la codificación del campo con el nombre INDICE es incorrecta ya que la cláusula USAGE IS INDEX debe ser la única cláusula en la descripción de un campo.

Ejemplo-3

En este ejemplo, se ha definido el campo INDICE-1 para ser utilizado en indexación y la tabla TABLA de cien elementos indexada por INDICE-2.

Dentro de la PROCEDURE DIVISION, y mediante las instrucciones

```
ACCEPT INDICE-1
ACCEPT INDICE-2
```

el operador almacena en INDICE-1 y en INDICE-2 dos valores (que deben estar comprendidos entre uno y cien).

```
01  INDICE-1 USAGE IS INDEX.
01  TAB.
    02 TABLA PIC 9 OCCURS 100 TIMES
                  INDEXED BY INDICE-2.

PROCEDURE DIVISION.

    ACCEPT INDICE-1.
    ACCEPT INDICE-2.
    ADD TABLA (INDICE-1) TO
        TABLA (INDICE-2).
```

Con la siguiente instrucción:

ADD TABLA (INDICE-1) TO TABLA (INDICE-2)

el ordenador suma al elemento de TABLA indexado por el valor almacenado en INDICE-1 el elemento de TABLA indexado por el valor almacenado en INDICE-2.

Ejemplo-4

Esta codificación es incorrecta ya que en la instrucción:

MOVE TABLA-1 [illegible] TABLA-2 (IND-1)

el índice utilizado para indexar la tabla TABLA-2 es IND-1 y dicho índice sólo puede indexar a la tabla TABLA-1.

La codificación correcta de esta instrucción sería la siguiente.

```
    MOVE TABLA-1 (IND-1) TO
         TABLA-2 (IND-2).
```

1.5. MODIFICACION DE INDICES

La instrucción SET modifica el valor de un índice perteneciente a una tabla o de un campo definido mediante la cláusula USAGE IS INDEX.

Ejemplo-1

```
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 49 50 51 52 53 54 55
      .
      .
01    TAB.
      02 TABLA PIC X OCCURS 27
                    INDEXED BY IND-TABLA.
      .
      .
PROCEDURE DIVISION.
      .
      .
      SET IND-TABLA TO 5.
      DISPLAY TABLA (IND-TABLA).
      .
      .
```

En este ejemplo, la instrucción:

SET IND-TABLA TO 5

asigna el valor cinco al índice IND-TABLA perteneciente a la tabla TABLA. Y con la instrucción:

DISPLAY TABLA (IND-TABLA)

se displaya el contenido del quinto elemento TABLA.

Ejemplo-2

En este otro ejemplo, la instrucción

MOVE 2 TO IND-TAB-1

es incorrecta ya que IND-TAB-1 es un campo destinado a la indexación de la tabla TAB-1. La codificación correcta sería la mostrada en la otra figura.

La instrucción SET es la única instrucción que puede mover o un dato nu-

```
01  TAB.
    02 TAB-1 PIC 99 OCCURS 18
                    INDEXED BY IND-TAB-1.

PROCEDURE DIVISION.

    MOVE 2 TO IND-TAB-1.
```

Codificación incorrecta

```
PROCEDURE DIVISION.

    SET IND-TAB-1 TO 2.
```

Codificación correcta

mérico almacenado en un campo o una constante numérica, hacia un campo destinado a indexación.

Si el índice que trata esta instrucción ha sido definido mediante la cláusula USAGE IS INDEX, el dato numérico a mover hacia dicho índice, debe estar almacenado en un campo. No puede ser una constante numérica.

Ejemplo-3

En el caso presentado en la primera figura de la página siguiente, la codificación es incorrecta ya que no se puede almacenar la constante numérica 19 en el campo IND-5. Para que IND-5 tome el valor 19 debe codificarse según la otra figura.

```
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 49 50 51 52 53 54 55
    .
    .
01  IND-5 USAGE IS INDEX.
    .
    .
PROCEDURE DIVISION.
    .
    .
        SET IND-5 TO 19.
    .
    .
```

Codificación incorrecta

```
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 49 50 51 52 53 54 55
    .
    .
01  IND-5 USAGE IS INDEX.
01  CAMPO PIC 9(4).
    .
    .
PROCEDURE DIVISION.
    .
    .
        MOVE 19 TO CAMPO.
        SET IND-5 TO CAMPO.
    .
    .
```

Codificación correcta

Ejemplo-4

```
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 49 50 51 52 53 54 55
    .
    .
01  TABLA-14.
        02 TAB-14 PIC 9(7) OCCURS 20
                  INDEXED BY IND-14.
    .
    .
PROCEDURE DIVISION.
    .
    .
        SET IND-14 TO 2.
    .
    .
```

En este otro caso, la codificación sí es correcta porque el índice IND-14 no ha sido definido mediante la cláusula USAGE IS INDEX.

Si la instrucción SET se utiliza para incrementar o decrementar un índice, dicho índice no debe haber sido definido mediante la cláusula USAGE IS INDEX.

Ejemplo-5

```
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 49 50 51 52 53 54 55
      .
      .
  01  IND-11 USAGE IS INDEX.
  01  IND-12 USAGE IS INDEX.
  01  CAM-1 PIC 9(6).
  01  CAM-2 PIC 9(5).
      .
      .
PROCEDURE DIVISION.
      .
      .
          SET IND-11 UP BY CAM-1.
          SET IND-12 DOWN BY CAM-2.
      .
      .
```

Este ejemplo está mal codificado, ya que para incrementar el valor del índice IND-11 y decrementar el valor de IND-12 con los valores almacenados en los campos CAM-1 y CAM-2 respectivamente, se debía haber codificado según la figura siguiente. Donde se incrementa CAM-1 con el valor almacenado en

```
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 49 50 51 52 53 54 55
      .
      .
PROCEDURE DIVISION.
      .
      .
          ADD IND-11 TO CAM-1.
          SET IND-11 TO CAM-1.
          SUBTRACT CAM-2 FROM IND-12
                                GIVING CAM-2.
          SET IND-12 TO CAM-2.
      .
      .
```

IND-11 y se asigna dicho valor incrementado a IND-11. A continuación se almacena en CAM-2 el resultado de restar al valor de IND-12 el valor de CAM-2 y se asigna dicho valor decrementado a IND-12.

Ejemplo-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 49 50 51 52 53 54 55
    .
    .
01  C-2 PIC 9(8).
01  C-3 PIC 9(5).
01  TABLA-8.
    02 TAB-8 OCCURS 23
                INDEXED BY IND-8.
       03 TAB-8-23 PIC X(20)
                   OCCURS 10
                   INDEXED BY IND-8-23.
    .
    .
PROCEDURE DIVISION.
    .
    .
    SET IND-8 UP BY C-2.
    SET IND-8-23 DOWN BY C-3.
    .
    .
```

Con los valores almacenados en los campos CAM-1 y CAM-2 respectivamente, esta codificación es correcta ya que IND-8 e IND-8-23 pueden ser incrementado y decrementado respectivamente por C-2 y C-3 porque son índices que no han sido definidos mediante la cláusula USAGE IS INDEX.

1.6. EXPLORACION DE UNA TABLA

Existen dos instrucciones para realizar la exploración de los elementos de una tabla:

— La instrucción PERFORM.
— La instrucción SEARCH.

Si se utiliza la instrucción PERFORM, es el programador el que debe codificar la rutina de búsqueda.

Ejemplo-1:

Detectar qué elemento de los trece últimos elementos de la tabla unidimensional TAB-2 almacena el valor dos millones. Una de las posibles codificaciones es la siguiente:

```
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
      .
      .
01    TAB.
        02 TAB-1 PIC 9(7) OCCURS 20
                          INDEXED BY IND-1.
01      SWITCH-1 PIC 9 USAGE IS COMP.
      .
      .
PROCEDURE DIVISION.
      .
      .
      .
DOS-MILLONES.
        MOVE ZERO TO SWITCH-1.
        PERFORM COMPROBAR
                VARYING IND-1 FROM 7
                                BY 1
                                UNTIL IND-1 > 20.
        IF SWITCH-1 NOT EQUAL TO ZERO THEN
                                GO TO SI-EXISTE.
NO-EXISTE.
      .
      .
SI-EXISTE.
      .
      .
      .
COMPROBAR.
        IF TAB-1 (IND-1) EQUAL TO 2000000
```

```
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
                          THEN GO TO ENCONTRADO.
OTRO-PARRAFO-1.
      .
      .
ENCONTRADO.
        MOVE 1 TO SWITCH-1.
ENCONTRADO-EXIT.
        EXIT.
OTRO-PARRAFO-2.
      .
      .
```

El funcionamiento de este programa es el siguiente:

— El campo SWITCH-9 tendrá un cero después de la rutina de búsqueda del valor dos millones entre los elementos de la tabla TAB-1 si ningún elemento almacena el valor dos millones y tendrá un uno en caso contrario.
— En el párrafo DOS-MILLONES están las instrucciones destinadas a la exploración de la tabla TAB-1.
— En el párrafo COMPROBAR se verifica si un determinado elemento almacena el valor dos millones.
— Los párrafos ENCONTRADO y ENCONTRADO-EXIT ponen a uno el campo SWITCH-9.

Los pasos seguidos por el progama serán:

— Poner a cero el SWITCH-9 antes de iniciar la exploración mediante la instrucción:

```
MOVE ZERO TO SWITCH-9
```

— Ejecutar el párrafo COMPROBAR trece veces variando IND-1 desde siete hasta veinte mediante:

```
PERFORM COMPROBAR  VARYING IND-1
                          FROM 7 BY 1
                          UNTIL IND-1 > 20
```

— Comprobar si el elemento TAB-1 indexado por IND-1 almacena el dato dos millones mediante la instrucción:

```
IF TAB-1  (IND-1) EQUAL TO 2 000 000
          THEN GO TO ENCONTRADO
```

Si no lo almacena, el control del procesador regresa a la instrucción PERFORM, incrementa en uno el valor del índice y transfiere de nuevo el control a la instrucción del párrafo COMPROBAR. Este proceso se repetirá hasta que:

— Se explore el último elemento TAB-1 y el resultado de la exploración sea negativo.
— Se encuentre un elemento que posea el valor ya mencionado.

En el primer caso, cuando la instrucción PERFORM haya sido completada, se ejecutará la instrucción:

```
IF SWITCH-9  NOT EQUAL TO ZERO
             THEN GO TO SI-EXISTE
```

pero como el valor de SWITCH-9 no habrá sido modificado, SWITCH-9 seguirá almacenando el valor cero y por tanto, el siguiente paso es ejecutar las instrucciones del párrafo NO-EXISTE.

En el segundo caso (se encuentra un elemento que tenga almacenado el valor dos millones), la instrucción:

```
IF TAB-1 (IND-1) EQUAL TO 2 000 000
         THEN GO TO ENCONTRADO
```

bifurcará hacia el párrafo ENCONTRADO.

En el párrafo ENCONTRADO, la instrucción:

```
MOVE 1 TO SWITCH-9
```

pone un uno en SWITCH-9. El procesador continúa en secuencia y ejecuta la instrucción:

```
EXIT
```

del párrafo ENCONTRADO-EXIT. Mediante esta instrucción, se da por finalizada la exploración controlada por la instrucción PERFORM devolviendo el control del procesador a la instrucción IF. Esta instrucción bifurcará al párrafo SI-EXISTE porque SWITCH-9 está a uno.

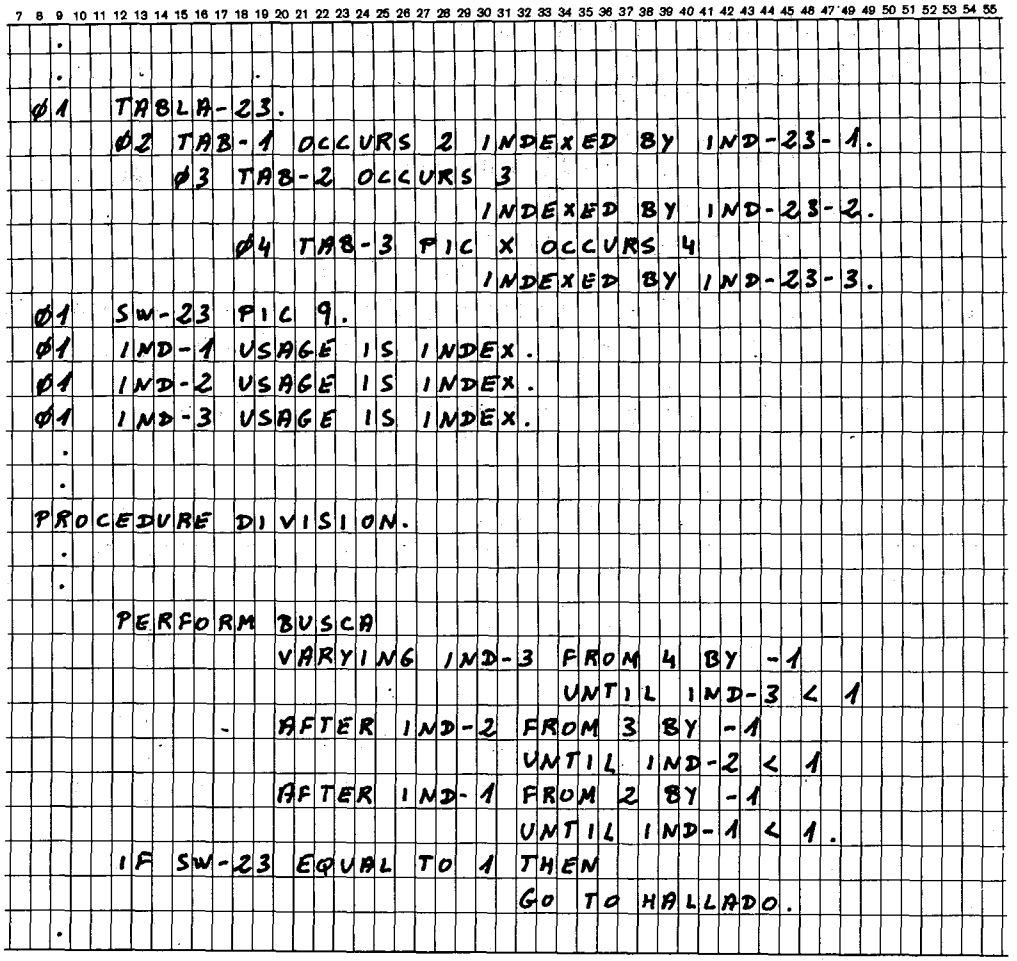
Ejemplo-2

Buscar en orden descendente (desde el último elemento hasta el primer elemento) dentro de la tabla tridimensional TAB-3 qué elemento almacena el carácter A o el carácter B ver página siguiente.

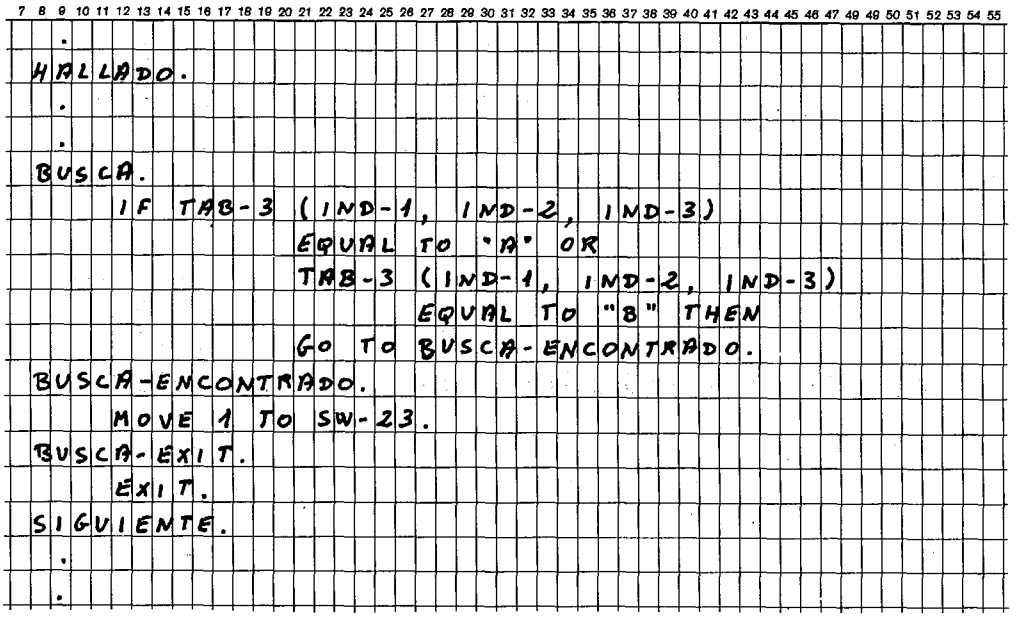
La instrucción:

```
PERFORM BUSCA VARYING IND-3 FROM 4 BY -1 UNTIL IND-3 < 1
              AFTER IND-2 FROM 3 BY -1 UNTIL IND-2 < 1
              AFTER IND-1 FROM 2 BY -1 UNTIL IND-1 < 1
```

varía el valor de los índices IND-1, IND-2 e IND-3 desde su máximo valor para la tabla TAB-3 hasta su valor mínimo y ejecuta las instrucciones del párrafo BUSCA.



```
01   TABLA-23.
     02 TAB-1 OCCURS 2 INDEXED BY IND-23-1.
        03 TAB-2 OCCURS 3
                 INDEXED BY IND-23-2.
           04 TAB-3 PIC X OCCURS 4
                    INDEXED BY IND-23-3.
01   SW-23 PIC 9.
01   IND-1 USAGE IS INDEX.
01   IND-2 USAGE IS INDEX.
01   IND-3 USAGE IS INDEX.
.
.
PROCEDURE DIVISION.
.
.
     PERFORM BUSCA
             VARYING IND-3 FROM 4 BY -1
                     UNTIL IND-3 < 1
             AFTER IND-2 FROM 3 BY -1
                     UNTIL IND-2 < 1
             AFTER IND-1 FROM 2 BY -1
                     UNTIL IND-1 < 1.
     IF SW-23 EQUAL TO 1 THEN
                     GO TO HALLADO.
.
```



```
HALLADO.
.
.
BUSCA.
     IF TAB-3 (IND-1, IND-2, IND-3)
              EQUAL TO "A" OR
              TAB-3 (IND-1, IND-2, IND-3)
                    EQUAL TO "B" THEN
              GO TO BUSCA-ENCONTRADO.
BUSCA-ENCONTRADO.
     MOVE 1 TO SW-23.
BUSCA-EXIT.
     EXIT.
SIGUIENTE.
.
.
```

En el párrafo BUSCA, la instrucción:

```
IF TAB-3 (IND-1, IND-2, IND-3)
         EQUAL TO "A" OR
         TAB-3 (IND-1, IND-2, IND-3)
         EQUAL TO "B" THEN
         GO TO BUSCA-ENCONTRADO
```

verifica si el elemento TAB-3 indexado por los valores que va asignando la instrucción PERFORM a los índices almacena el carácter A o el carácter B.

En este ejemplo, los índices tomarían los siguientes valores.

IND-1	*IND-2*	*IND-3*
2	3	4
1	3	4
2	2	4
1	2	4
2	1	4
1	1	4
2	3	3
1	3	3
2	2	3
1	2	3
2	1	3
1	1	3
2	3	2
.	.	.
.	.	.
.	.	.
1	1	1

Otro método para explorar tablas es utilizar la instrucción SEARCH.

Ejemplo-3

Explorar la tabla unidimensional TAB-14 compuesta por veinticuatro elementos para comprobar qué elementos posee el carácter $.

```
01   TABLA-14.
     02 TAB-14 PIC X OCCURS 24
                INDEXED BY IND-14.
.
.
PROCEDURE DIVISION.
.
.
     SEARCH ALL TAB-14
          WHEN TAB-14 (IND-14) = "$"
               GO TO ENCONTRADO
          AT END GO TO ERROR-TAB-14.
.
.
ENCONTRADO.
.
.
ERROR-TAB-14.
.
.
```

Al ejecutar la instrucción:

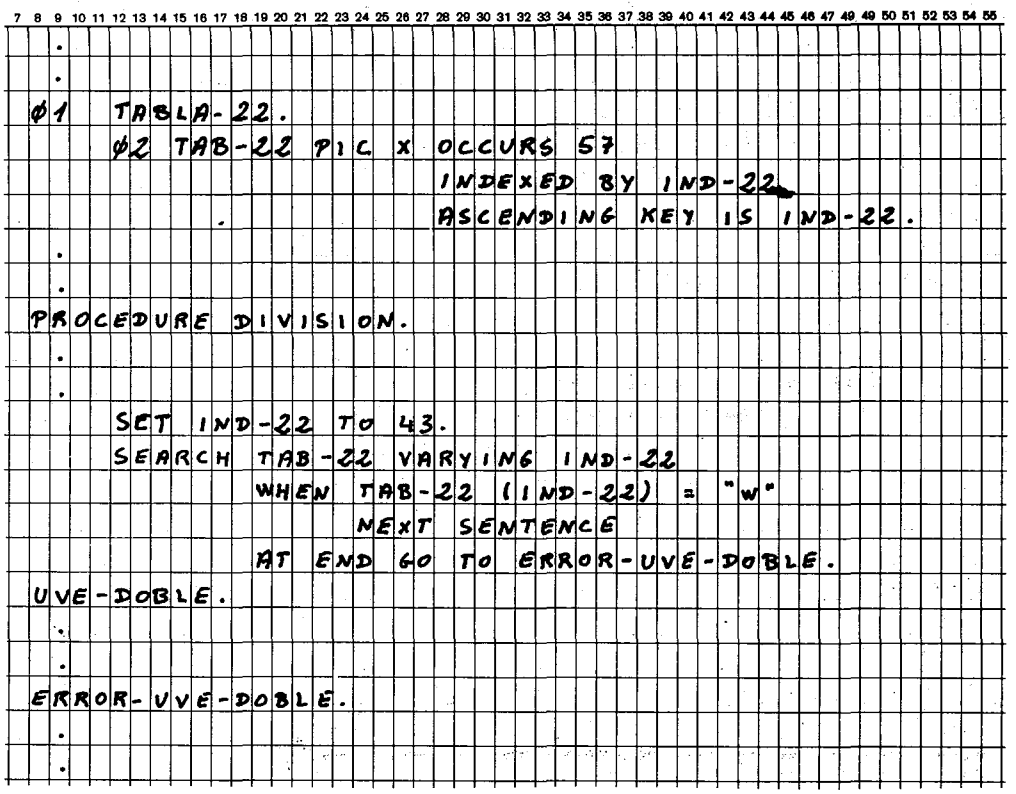
```
SEARCH ALL TAB-14
       WHEN TAB-14 (IND-14) = "$" GO TO ENCONTRADO
       AT END GO TO ERROR-TAB-14.
```

se explorarán todos y cada uno de los elementos de TAB-14. Si hay alguno que almacene el carácter $ el control del procesador es bifurcado hacia el párrafo ENCONTRADO. SI no hay ningún elemento con el carácter $, se transfiere el control del procesador al párrafo ERROR-TAB-14.

Ejemplo-4

Detectar qué elemento de los quince últimos elementos de la tabla TAB-22 almacena el carácter N.

La codificación sería la de la siguiente:



La instrucción:

SET IND-22 TO 43

almacena la posición del quinceavo elemento de TAB-22 empezando por el final en IND-22.

La instrucción:

```
SEARCH  TAB-22 VARYING IND-22
            WHEN TAB-22 (IND-22) = "N" NEXT SENTENCE
            AT END GO TO ERROR-UVE-DOBLE
```

explora TAB-22 buscando un elemento que almacene el carácter W a partir del elemento que ocupa la posición cuarenta y tres incrementando IND-22 hasta el último elemento de TAB-22.

La cláusula ASCENDING KEY en la definición de TAB-22 indica que la exploración es desde el elemento indexado por IND-22 hasta el final de la tabla.

Si encuentra un elemento que cumpla esta condición, bifurca a la siguiente instrucción que está en el párrafo UVE-DOBLE. Si no encuentra un elemento que cumpla dicha condición, bifurca hacia el párrafo ERROR-UVE-DOBLE.

Ejemplo-5

Buscar en orden descendente a partir del elemento indicado por el valor de IND-18-1 qué elemento dentro de la tabla TAB-18-1 almacena los caracteres ABCD o los caracteres DCBA.

```
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
      .
      .
01    TABLA-18.
        02 TAB-18-1 PIC X(4)
                    OCCURS 8 TIMES
                    INDEXED BY IND-TAB-18-1
                    DESCENDING KEY IS IND-18-1.
01    IND-18-1 USAGE IS INDEX.
      .
      .
PROCEDURE DIVISION.
      .
      .
          SEARCH TAB-18-1 VARYING IND-18-1
                 WHEN TAB-18-1 (IND-18-1)
                      EQUAL TO "ABCD"
                              GO TO A-B-C-D
                 WHEN TAB-18-1 (IND-18-1)
                      EQUAL TO "DCBA"
                              GO TO D-C-B-A.
NO-ENCONTRADO.
      .
      .
A-B-C-D.
      .
      .
D-C-B-A.
```

En este ejemplo, la cláusula

```
DESCENDING KEY IS IND-18-1
```

especifica que se utilizará el índice IND-18-1 para explorar la tabla TAB-18-1 en orden descendente.

La instrucción:

```
SEARCH TAB-18-1    VARYING IND-18-1
                   WHEN TAB-18-1 (IND-18-1)
                       EQUAL TO "ABCD"
                       GO TO A-B-C-D
                   WHEN TAB-18-1 (IND-18-1)
                       EQUAL TO "DCBA"
                       GO TO A-B-C-D
```

explora TAB-18-1 a partir del elemento indexado por el valor almacenado en IND-18-1 hasta el primer elemento de la tabla (en orden descendente). Si algún elemento explorado [TAB-18-1 (IND-18-1)] almacena los caracteres ABCD, se bifurcará el control del procesador al párrafo A-B-C-D y si algún elemento almacena D-C-B-A, también se bifurcará hacia A-B-C-D.

Ejemplo-6

Cambiar el carácter N por el carácter X en todos los elementos de la tabla NOMBRE-2 que almacenen el carácter N (vea la codificación en la página siguiente).

En primer lugar, se asigna el valor uno a los índices que se utilizarán para la exploración de la tabla mediante:

```
SET IND-N-1, IND-N-2 TO 1
```

la instrucción que remplaza en cada elemento que almacene el carácter N por el carácter X es:

```
SEARCH NOMBRE-2  VARYING IND-N-1, IND-N-2
                 WHEN NOMBRE-2 (IND-N-1, IND-N-2)
                 EQUAL TO "N"
                 MOVE "X" TO NOMBRE-2 (IND-N-1, IND-N-2)
```

```
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 49 50 51 52 53 54 55
      .
      .
  01  TABLA-NOMBRE.
      02 NOMBRE-1 OCCURS 14 TIMES
                  INDEXED BY IND-N-1
                  ASCENDING KEY IS IND-N-1.
         03 NOMBRE-2 PIC X
                     OCCURS 20 TIMES
                     INDEXED BY IND-N-2
                     ASCENDING KEY IS IND-N-2.
      .
      .
PROCEDURE DIVISION.
      .
      .
      SET IND-N-1, IND-N-2 TO 1.
      SEARCH NOMBRE-2 VARYING IND-N-1, IND-N-2
             WHEN NOMBRE-2 (IND-N-1, IND-N-2)
                  EQUAL TO "N"
                     MOVE "X" TO NOMBRE-2
                     (IND-N-1, IND-N-2).
      .
      .
```

EJERCICIOS

1. Una tabla es un conjunto de datos .....

   .....

2. Para definir una tabla se emplea la cláusula .....

3. La posición de un elemento de una tabla se puede indicar mediante

   a) La instrucción SET
   b) El nombre de la tabla
   c) Indices o subíndices

4. ¿Cuál es la referencia correcta a la tabla?

```
01 MESA
    02 MODELO
        03 TAMAÑO
```

   a) TAMAÑO (1, 1)
   b) MESA (1, 1)
   c) MODELO (1, 1)

5. El valor de un subíndice ha de estar comprendido entre

   a) Uno y la cantidad de elementos que posea la dimensión subindicada
   b) Cero y la cantidad de elementos de la tabla
   c) Uno y la cantidad de elementos de la tabla

6. La cláusula REDEFINES

   a) Redefine una tabla sobre otra
   b) Define una tabla
   c) Redefine los valores de una tabla

7. Los índices asociados a una tabla deben estar definidos

   a) En la Working-Storage Section
   b) Junto con la definición de la dimensión de la tabla
   c) Junto con la cláusula Redefines.

2.
MASCARAS

CUADRO RESUMEN DE LA LECCION

Máscaras	— Definición de máscaras — Posición de cada campo — Campos con constantes — Campos con variables — Transferencia de datos hacia la pantalla — Transferencia de datos desde el teclado — Borrado de la pantalla — Campos con display intermitentes — Campos alineados por la derecha — Sustitución de ceros por blancos — Campos de seguridad — Campos obligatorios — Llamada de atención — Cambio automático de campo — Borrado de una línea — Scrolling

Las *máscaras* sirven para mantener una comunicación constante y flexible entre un programa en ejecución y el operador que trabaja desde un terminal (*método interactivo*), cuando el elemento de salida de dicho terminal es una pantalla (DISPLAY) y el elemento de entrada es un teclado (KEYBOARD).

2.1. DEFINICION DE MASCARAS

Una máscara es el conjunto de elementos que deben aparecer en la pantalla del operador durante la ejecución de un programa.

Cada elemento se define mediante números de nivel dentro de la SCREEN SECTION (perteneciente a la DATA DIVISION) del mismo modo que se hace en la Working-Storage Section al definir los campos de memoria.

Dentro de la Screen Section, se pueden definir tantas máscaras como sea necesario.

Ejemplo-1

En este ejemplo, el programa EJEMPLO-1 utilizará dos máscaras referenciadas mediante los nombres MASCARA-UNO y MASCARA-DOS.

El nombre de cada máscara llevará el número de nivel 01 y los elementos de los que se componen, irán jerarquizados con números de nivel del 02 al 49.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EJEMPLO-1.
.
.
ENVIRONMENT DIVISION.
.
.
DATA DIVISION.
FILE SECTION.
.
.
WORKING-STORAGE SECTION.
.
.
SCREEN SECTION.
01  MASCARA-UNO.
      .
      .
01  MASCARA-DOS.
      .
      .
PROCEDURE DIVISION.
.
.
```

Ejemplo-2

```
.
.
SCREEN SECTION.
.
.
01  MASCARA-CINCO.
    02 DATOS-DEL-ALUMNO.
       03 ..........
          .
          .
    02 DATOS-DEL-TUTOR.
       03 ..........
          .
          .
.
.
```

En este ejemplo, se define una máscara con el nombre MASCARA-CINCO (números de nivel 01) que se subdivide en (número de nivel 02): DATOS-DEL-ALUMNO y DATOS-DEL-TUTOR. Y cada uno de ellos se subdivide a su vez en varios elementos (números de nivel 03).

Cada elemento en la descripción de una máscara corresponde a un campo en la pantalla.

Dichos campos sirven para:

— Mensajes explicativos sobre el tipo de datos solicitado por el programa.
— Asignación de valores a las variables de un programa.

No es necesario que cada elemento de una máscara tenga nombre, a menos que sea necesario hacer referencia a un elemento de la misma.

2.2. POSICION DE LOS CAMPOS

La posición en la pantalla de cada campo, se define mediante la línea y la columna (dentro de esa línea) en la que debe aparecer el primer carácter del campo.

La línea se indica mediante la cláusula:

$$\text{LINE} \left\{ \begin{array}{l} \text{X} \\ \text{PLUS Y} \end{array} \right\}$$

siendo:

LINE y PLUS = Palabras reservadas

X e Y = Constantes numéricas enteras sin signo.

Con:

```
LINE 14
```

se indica que el campo que define el elemento que tiene esta cláusula está situado en la línea número 14 de la pantalla.

Con:

```
LINE PLUS 3
```

se indica que el campo está situado tres líneas más abajo a partir del último campo definido con anterioridad.

La columna se indica mediante la cláusula:

$$\text{COL} \left\{ \begin{array}{l} \text{X} \\ \text{PLUS Y} \end{array} \right\}$$

siendo

COL y PLUS = Palabras reservadas
X e Y = Constantes numéricas enteras sin signo

Con:

COL 35

se indica que el campo está situado en la columna 35.

Con:

COL PLUS 19

se indica que el campo está situado 19 columnas más a la derecha del último campo definido.

Si se omite la cláusula COL, en la descripción de un elemento de una máscara, el compilador asume

COL 1

Si se omite la cláusula LINE, el compilador asume el mismo valor de la última cláusula LINE especificada.

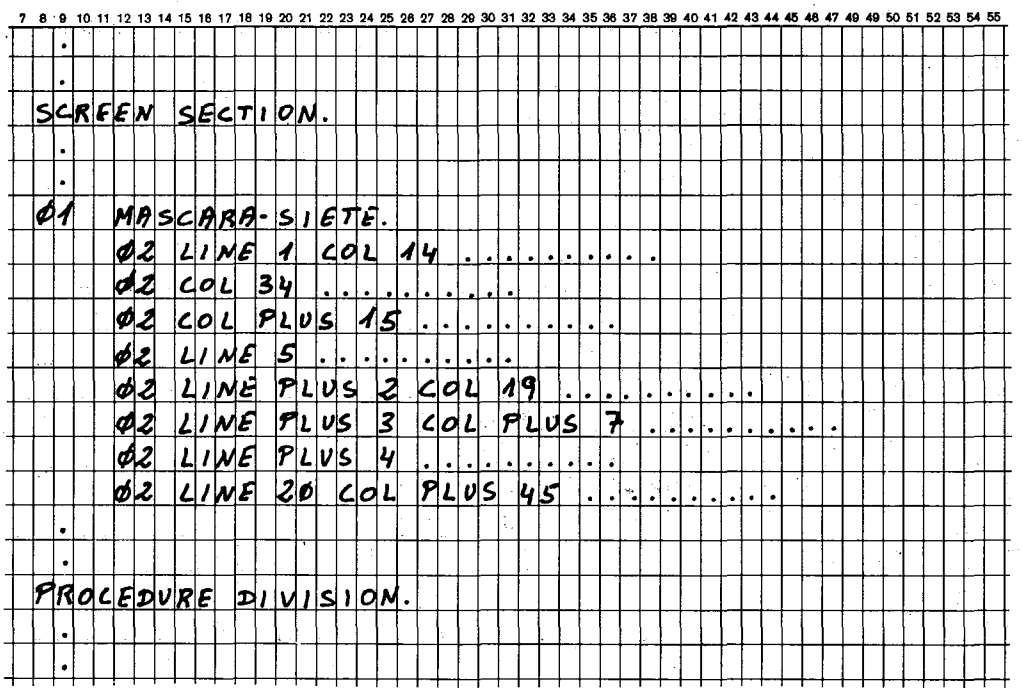
Ejemplo

En este ejemplo, (vea la codificación en la página siguiente), la máscara MASCARA-SIETE estructura la pantalla con ocho campos (la descripción de la máscara posee ocho elementos) dispuestos de la siguiente manera:

— 02 LINE 1 COL 14
En la línea 1, columna 14

— 02 COL 34
En la línea 1 (ya que al estar omitida la cláusula LINE, se toma el valor de la última línea especificada), columna 34.

— 02 COL PLUS 15
En la línea 1, columna 49 (puesto que la cláusula COL PLUS 15 sitúa el campo quince columnas a la derecha del último campo definido).



— 02 LINE 5
En la línea 5, columna 1 (al no estar incluida la cláusula COL, se asume el valor uno).

— 02 LINE PLUS 2 COL 19
En la línea 7 (puesto que con LINE PLUS 2 el campo se sitúa dos filas más abajo del último campo definido), columna 19.

— 02 LINE PLUS 3 COL PLUS 7
En la línea 10, columna 26.

— 02 LINE PLUS 4
En la línea catorce, columna uno.

— 02 LINE 20 COL PLUS 45
En la línea veinte, columna 46.

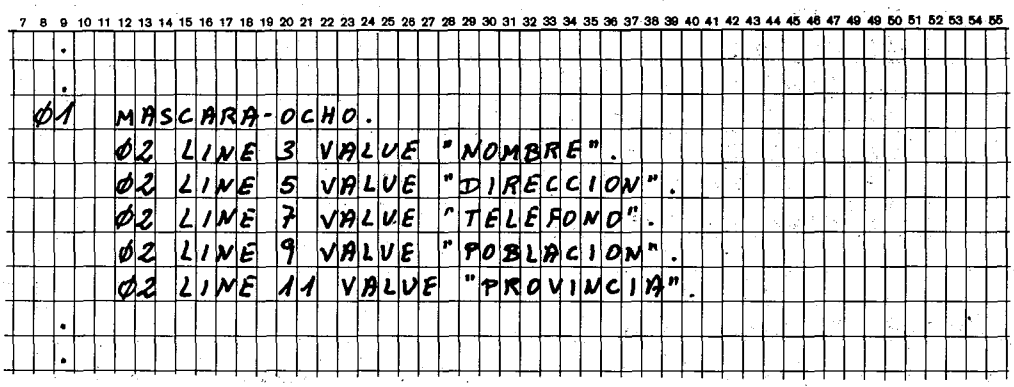
2.3. CAMPOS CON CONSTANTES

Los campos con constantes sirven para informar al operador del tipo de datos solicitados por el programa.

Para asignar una constante a un campo de una máscara, se utiliza la cláusula.

VALUE {IS} «constante»

Ejemplo



En este ejemplo se define la máscara MASCARA-OCHO con cinco campos que contienen constantes definidas mediante la cláusula VALUE.

— 02 LINE 3 VALUE "NOMBRE".
En la primera columna de la tercera línea de la pantalla aparecerá la constante: NOMBRE

— 02 LINE 5 VALUE "DIRECCION".
En la primera columna de la quinta línea de la pantalla aparecerá la constante: DIRECCION

— 02 LINE 7 VALUE "TELEFONO".
En la primera columna de la séptima línea aparecerá: TELEFONO

— 02 LINE 9 VALUE "POBLACION"
En la primera columna de la novena línea aparecerá: POBLACION

— 02 LINE 11 VALUE "PROVINCIA".
En la primera columna de la línea once aparecerá: PROVINCIA

2.4. CAMPOS CON VARIABLES

Los campos con variables sirven para:

— Mostrar en la pantalla del operador los valores almacenados en las va-

riables especificadas mediante la cláusula:

FROM variable

— Asignar valores, mediante el teclado, a las variables especificadas por la cláusula:

TO variable

— Modificar los valores almacenados en las variables especificadas mediante la cláusula:

USING variable

Cada una de estas cláusulas debe ir acompañada por la cláusula PICTURE. La definición de cada variable como campo de la máscara, debe coincidir con su definición en la Working-Storage Section.

Ejemplo-1

```
WORKING-STORAGE SECTION.
01  DATOS.
    02 NOM PIC X(30).
    02 DIR PIC X(20).
    02 TEL PIC 9(7).
    02 POB PIC X(15).
    02 PRO PIC X(15).
.
.
SCREEN SECTION.
01  MASCARA-DATOS.
    02 LINE 3  VALUE "NOMBRE".
    02 COL 8   PIC X(30) TO NOM.
    02 LINE 5  VALUE "DIRECCION".
    02 COL 11  PIC X(20) TO DIR.
    02 LINE 7  VALUE "TELEFONO".
    02 COL 10  PIC 9(7) TO TEL.
    02 LINE 9  VALUE "POBLACION".
    02 COL 11  PIC X(15) TO POB.
    02 LINE 11 VALUE "PROVINCIA".
    02 COL 11  PIC X(15) TO PRO.
.
.
```

En este ejemplo, la máscara MASCARA-DATOS posee diez campos, cinco de los cuales tienen las constantes ya explicadas en el ejemplo -4 y las cinco restantes son:

— 02 COL 8 PIC X(30) TO NOM.
Campo con treinta caracteres alfanuméricos destinados a almacenar en la variable NOM (definida en la Working-Storage Section) el dato tecleado por el operador. (En este, tal como explica el campo anterior, un nombre).

— 02 COL 11 PIC X(20) TO DIR.
Campo de veinte caracteres alfanuméricos destinado a almacenar en la variable DIR una dirección.

— 02 COL 10 PIC 9(7) TO TEL.
Campo de siete caracteres numéricos destinado a almacenar un número de teléfono en la variable TEL.

— 02 COL 11 PIC X(15) TO POB.
Campo destinado a almacenar el dato que teclee el operador en la variable POB.

— 02 COL 11 PIC X(15) TO PRO.
Almacena el dato introducido por el operador en la variable PRO.

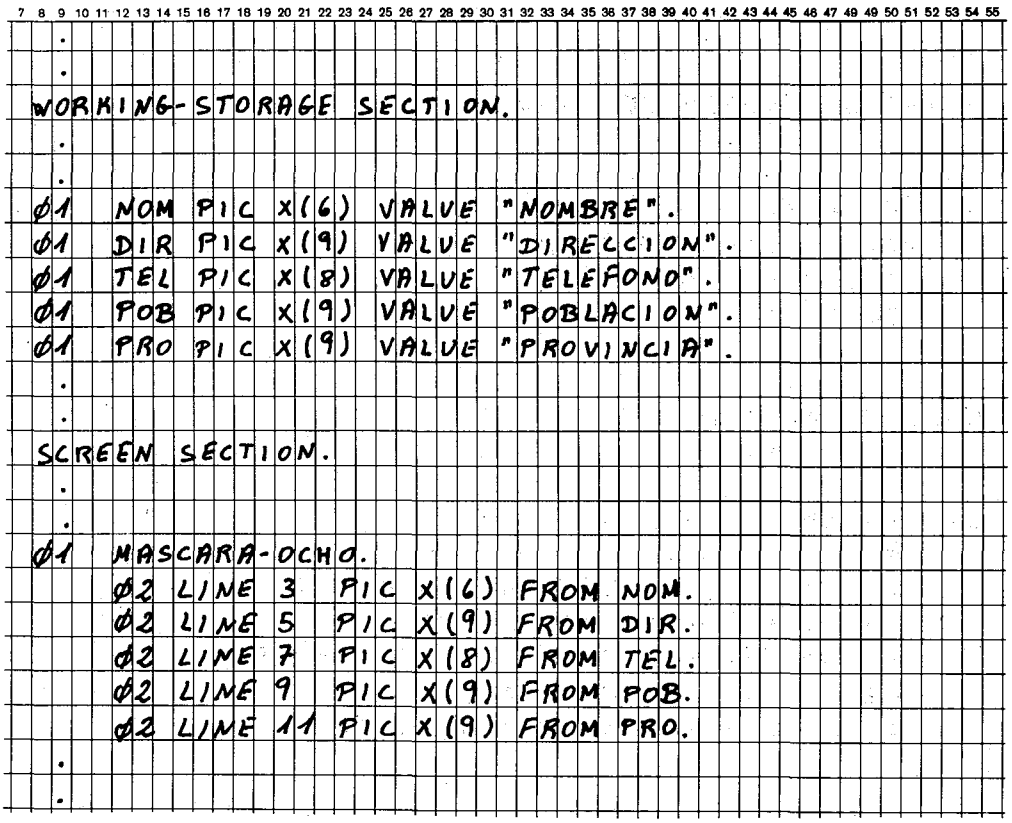
Ejemplo-2

En este ejemplo, la MASCARA-OCHO posee cinco campos, al igual que el ejemplo del apartado 2.3, definidos por:

— 02 LINES PIC X(6) FROM NOM.
En la columna uno de la tercera línea de la pantalla, aparecerá el contenido de la variable NOM.

— 02 LINE 5 PIC X(9) FROM DIR.
En la columna uno de la quinta línea aparecerá el contenido de la variable DIR.

— 02 LINE 7 PIC X(8) FROM TEL.
Aparecerá el contenido de la variable TEL.



```
WORKING-STORAGE SECTION.

01  NOM PIC X(6) VALUE "NOMBRE".
01  DIR PIC X(9) VALUE "DIRECCION".
01  TEL PIC X(8) VALUE "TELEFONO".
01  POB PIC X(9) VALUE "POBLACION".
01  PRO PIC X(9) VALUE "PROVINCIA".

SCREEN SECTION.

01  MASCARA-OCHO.
    02 LINE 3  PIC X(6) FROM NOM.
    02 LINE 5  PIC X(9) FROM DIR.
    02 LINE 7  PIC X(8) FROM TEL.
    02 LINE 9  PIC X(9) FROM POB.
    02 LINE 11 PIC X(9) FROM PRO.
```

— 02 LINE 9 PIC X(9) FROM POB.
Mostrará en pantalla el contenido de la variable POB.

— 02 LINE 11 PIC X(9) FROM PRO.
Mostrará el contenido de la variable PRO.

Ejemplo-3

En este ejemplo, la máscara mostrará por pantalla:

— Línea 1, columna 1: La constante CODIGO.

— Línea 1, columna 10: El valor almacenado en la variable CODIGO permitiendo su modificación.

— Línea 2, columna 1: La constante IMPORTE.

— Línea 2, columna 10: El valor de la variable IMPORTE permitiendo su modificación.

```
WORKING-STORAGE SECTION.
.
.
01  CLIENTE.
    02 CODIGO PIC 9(5).
    02 NOMBRE PIC X(25).
    02 RUTA   PIC 99.
.
.
SCREEN SECTION.
.
.
01  MASCARA-CLIENTE.
    02 LINE 1 VALUE "CODIGO".
    02 COL 10 PIC 9(5) USING CODIGO.
    02 LINE 2 VALUE "NOMBRE".
    02 COL 10 PIC X(25) USING NOMBRE.
    02 COL 30 VALUE "RUTA".
    02 COL 36 PIC 99 USING RUTA.
.
.
```

— Línea 2, columna 30: La constante RUTA.

— Línea 2, columna 36: El valor de la variable RUTA permitiendo su modificación.

Por tanto, la cláusula USING puede considerarse como una mezcla de las cláusulas FROM y TO.

2.5. TRANSFERENCIA DE DATOS HACIA LA PANTALLA

Para transferir a la pantalla los datos que definen los elementos de una máscara, se utiliza la instrucción:

DISPLAY nombre-de-máscara

Los pasos seguidos por esta instrucción con los elementos que forman la máscara son:

— Cláusulas FROM y USING: Se transfiere el valor almacenado en las variables indicadas por estas cláusulas a los elementos correspondientes en la máscara.

— Cláusula TO: Se transfieren guiones a los elementos correspondientes en la máscara.

— Se displayan en la pantalla los datos de los elementos de la máscara.

Ejemplo

```
WORKING-STORAGE SECTION.
01  N-M   PIC 9(4).
01  P4-K  PIC X(6).
01  NOM   PIC X(20).
SCREEN SECTION.
01  MASCARA.
    02 LINE 2 COL 10 VALUE
       "DATOS DE UN ALUMNO".
    02 LINE 3 COL 10 VALUE
       "----------".
    02 LINE 7 VALUE "N. MATRICULA".
    02 COL 14 PIC 9(4) TO N-M.
    02 LINE 9 PIC X(6) FROM P4-K.
    02 LINE 11 VALUE "NOMBRE".
    02 COL 8  PIC X(20) USING NOM.
PROCEDURE DIVISION.
```

```
    DISPLAY MASCARA.
```

Al ejecutarse la instrucción:

DISPLAY MASCARA

ocurre lo siguiente:

— Se transfieren cuatro guiones al elemento definido por

02 COL 14 PIC 9(4) TO N-M

puesto que su longitud es de cuatro caracteres y posee la cláusula TO.

— Se transfiere el valor de la variable P4-K al elemento definido por:

02 LINE 9 PIC X(6) FROM P4-K.

— Se transfiere el valor almacenado en NOM al elemento definido por:

02 COL 8 PIC X(20) USING NOM.

— Se displayan por la pantalla los datos de cada campo definidos en MASCARA

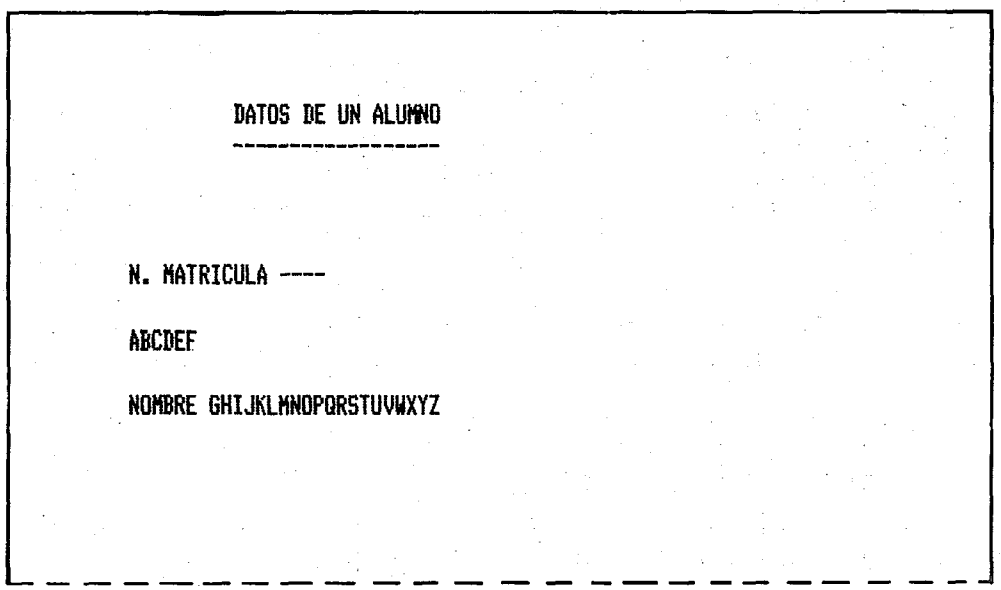
Si los valores almacenados en las variables son:

N.M 2 3 4 5

P4-K ..... A B C D E F

NOM ..... G H I J K L M N O P Q R S T U V W X Y Z

se obtiene lo mostrado en la figura.



2.6. TRANSFERENCIA DE DATOS DESDE EL TECLADO

Una vez la máscara está displayada en la pantalla, al ejecutarse la instrucción.

ACCEPT nombre-de-máscara

el operador puede asignar valores a las variables especificadas (mediante las cláusulas TO y USING) en los elementos de la máscara.

Ejemplo:

Si la Procedure Division del ejemplo anterior fuera como el que muestra la codificación siguiente, después de ejecutarse la instrucción

DISPLAY MASCARA

se ejecutará la instrucción

ACCEPT MASCARA

```
PROCEDURE DIVISION.

    DISPLAY MASCARA.
    ACCEPT MASCARA.
```

Ahora es cuando el operador tecleará el dato a almacenar en la variable N-M y podrá modificar (si lo cree conveniente) el dato almacenado en la variable NOM.

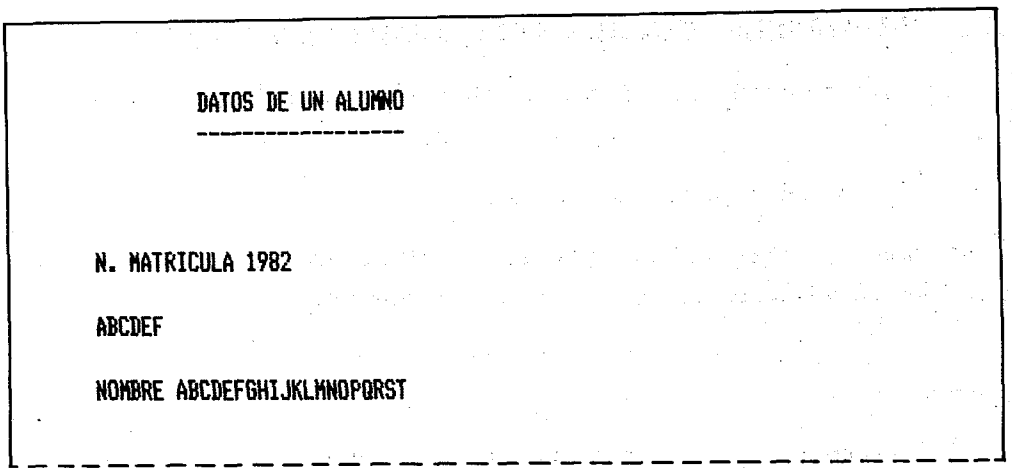
Si para N-M teclea

1982

y modifica las variables NOM con

ABCDEFGHIJKLMNOPQRST

en la pantalla se obtendrá el listado siguiente y los pasos seguidos por el procesador serán:



— Asignar a la variable N-M el dato tecleado en el campo definido por

02 COL 14 PIC 9(4) TO N-M.

— Asignar a la variable NOM el dato tecleado en el campo definido por

02 COL 8 PIC X(20) USING NOM.

2.7. BORRADO DE LA PANTALLA

Para evitar la superposición (en la pantalla del operador) de los elementos de una máscara precedente con la nueva máscara a displayar, es conveniente que el primer elemento de todas las máscaras borre (o limpie) la pantalla. Esto se logra con la cláusula

BLANK SCREEN

Ejemplo (ver codificación y listados en página siguiente).

Al ejecutarse la instrucción

DISPLAY MASCARA-UNO

se obtendrá el listado 1.

Y al ejecutar la instrucción

DISPLAY MASCARA-DOS

se obtendrá el listado 2, mezclándose las máscaras MASCARA-UNO y MASCARA-DOS.

```
SCREEN SECTION.
.
.
01  MASCARA-UNO.
    02 LINE 4 COL 4 VALUE "12345".
    02 LINE 6       VALUE "12345".
01  MASCARA-DOS.
    02 LINE 3       VALUE "12345".
    02 LINE 4       VALUE "12345".
    02 LINE 6 COL 4 VALUE "12345".
.
.
PROCEDURE DIVISION.
.
.
    DISPLAY MASCARA-UNO.
    DISPLAY MASCARA-DOS.
.
.
```

Codificación

```
    12345

12345
```

Listado 1

Listado 2

```
12345345
12345

12312345
```

Para evitar esto, se puede codificar el programa de la forma siguiente, con lo que al ejecutar la instrucción

DISPLAY MASCARA-DOS

se obtendrá el listado 3.

```
SCREEN SECTION.
.
.
01  MASCARA-UNO.
    02 BLANK SCREEN.
    02 LINE 4 COL 4 VALUE "12345".
    02 LINE 6       VALUE "12345".
01  MASCARA-DOS.
    02 BLANK SCREEN.
    02 LINE 3       VALUE "12345".
    02 LINE 4       VALUE "12345".
    02 LINE 6 COL 4 VALUE "12345".
.
.
PROCEDURE DIVISION.
.
.
    DISPLAY MASCARA-UNO.
    DISPLAY MASCARA-DOS.
.
.
```

Codificación

Listado 3

```
12345
12345

   12345
```

2.8. CAMPOS CON DISPLAY INTERMITENTE

Cuando se desea que el contenido de un campo aparezca en la pantalla de manera intermitente, a la descripción del elemento en la máscara se le añade la cláusula

BLINK

Esta cláusula no produce ningún efecto en elementos con la cláusula TO.

Ejemplo

```
SCREEN SECTION.
01  MASCARA.
    02 BLANK SCREEN.
    02 LINE 8                  BLINK VALUE "CUIDADO".
    02 COL PLUS 3              BLINK PIC 99 FROM UNO.
    02 LINE 11 COL 7 BLINK PIC 99 TO DOS.
```

En este programa, los campos correspondientes a la variable UNO y a la constante CUIDADO aparecerán y desaparecerán de manera intermitente, mientras que el campo de la variable DOS no parpadeará en la pantalla, lográndose el mismo resultado con la codificación siguiente.

```
SCREEN SECTION.
01  MASCARA.
    02 BLANK SCREEN.
    02 LINE 8                  BLINK VALUE "CUIDADO".
    02 COL PLUS 3              BLINK PIC 99 FROM UNO.
    02 LINE 11 COL 7           PIC 99 TO DOS.
```

2.9. CAMPOS ALINEADOS POR LA DERECHA

Para lograr la alineación por la derecha en un campo de una máscara, se incluye la cláusula

JUSTIFIED Sirve para alinear por la derecha.

en la descripción del elemento de la máscara.

Ejemplo

```
SCREEN SECTION.
01  MASCARA.
    02 BLANK SCREEN.
    02 LINE 4 PIC X(20) JUSTIFIED FROM NOM.
```

Si el dato almacenado en la variable NOM son los caracteres JORGE, se obtendrá la imagen del listado 1.

```
          JORGE
```

Listado 1

Si no se hubiera incluido la cláusula JUSTIFIED, se obtendría la imagen del listado 2.

```
SCREEN SECTION.
01  MASCARA.
    02 BLANK SCREEN.
    02 LINE 4 PIC X(20) FROM NOM.
```

```
JORGE
```

Listado 2

2.10. SUSTITUCION DE CEROS POR BLANCOS

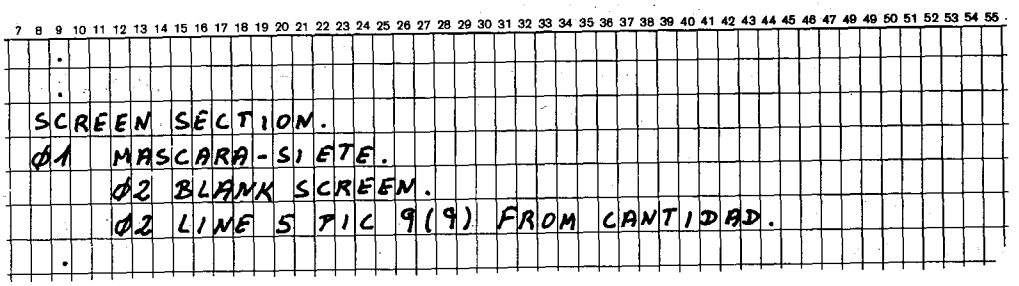
Si se desea que en un campo numérico, los ceros no significativos de la izquierda no aparezcan, se incluirá la cláusula

BLANK WHEN ZERO

en la definición del campo dentro de la máscara.

Con esta cláusula, los ceros no significativos son sustituidos por el carácter blanco.

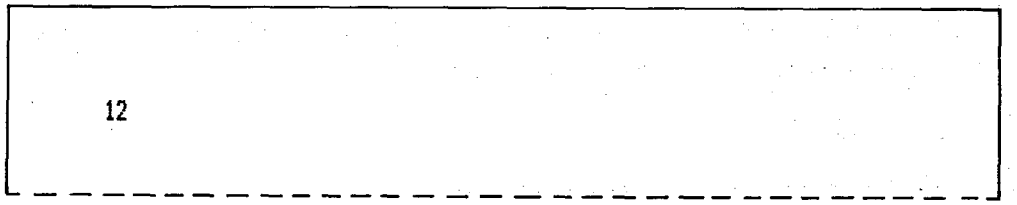
Ejemplo




```
000000012
```

Si el valor almacenado en CANTIDAD es 12, al displayar MASCARA-SIETE por pantalla, se obtendrá la imagen del listado anterior. Pero si la codificación fuera la que se muestra a continuación, los ceros no significativos quedarían suprimidos en la pantalla.

```
SCREEN SECTION.
01  MASCARA-SIETE.
    02 BLANK SCREEN.
    02 LINE 5 PIC 9(9) FROM CANTIDAD
                BLANK WHEN ZERO.
```



2.11. CAMPOS DE SEGURIDAD

Hay casos en los que nadie debe poder leer en la pantalla los caracteres que el operador introduce en un campo (por ejemplo: un código personal, una clave de seguridad,... etc.)

Cuando un campo deba poseer esta característica, se incluirá en su definición la cláusula

SECURE

Con esta cláusula:

— Se transmiten los caracteres que teclea el operador a la variable asignada al campo.

— Se transmiten asteriscos en sustitución de los caracteres que teclea el operador a la pantalla.

Ejemplo

Si el operador teclea 3728H/19 en la pantalla aparecerán ocho asteriscos tal como muestra el listado. (Vea la codificación y el listado en la página siguiente.)

```
SCREEN SECTION.
01  M-CODIGO.
    02 BLANK SCREEN.
    02 LINE 10 COL 10 VALUE
       "CUAL ES SU CODIGO?".
    02 COL PLUS 2 SECURE
       PIC X(8) TO CODIGO.
```

```
CUAL ES SU CODIGO? ********
```

2.12. CAMPOS OBLIGATORIOS

Si se quiere que el operador introduzca obligatoriamente un dato en un campo de entrada, en la definición del campo se incluirá la cláusula

REQUIRED [FULL]

Con la cláusula REQUIRED, el operador deberá introducir al menos un carácter y con la cláusula REQUIRED FULL el operador deberá llenar el campo (introducir todos los caracteres que admita).

Ejemplo-1

Al ejecutarse la instrucción

ACCEPT M-DATOS

el operador deberá introducir al menos un carácter en el campo TANT e introducir siete caracteres en el campo CIEN.

```
SCREEN SECTION.
01   M-DATOS.
     02 BLANK SCREEN.
     02 LINE 1 REQUIRED
        PIC X(7) TO TANT.
     02 LINE PLUS 5
        REQUIRED FULL
        PIC X(8) TO CIEN.

PROCEDURE DIVISION.

     DISPLAY M-DATOS.
     ACCEPT M-DATOS.
```

2.13. LLAMADA DE ATENCION

Cuando se considere oportuno llamar la atención del operador sobre un campo determinado de la máscara en pantalla, en la definición del campo se incluirá la cláusula:

BELL

Con esta cláusula, se activa el avisador acústico. En el ejemplo, cuando se displaye la constante PUEDE EMPEZAR, el operador oirá el avisador acústico de su terminal.

```
SCREEN SECTION.

01   M-AVISO.
     02 BLANK SCREEN.
     02 LINE 2 BELL VALUE
        "PUEDE EMPEZAR".
```

2.14. CAMBIO AUTOMATICO DE CAMPO

Una vez que el operador ha tecleado un dato, debe pulsar la tecla RETURN de su teclado para indicar al ordenador que da por finalizada la introducción de los caracteres del dato.

En procesos que exigen una gran rapidez del operador y que, además, necesitan la introducción de muchos datos, el hecho de tener que pulsar la tecla RETURN hace que la introducción de datos sea más lenta.

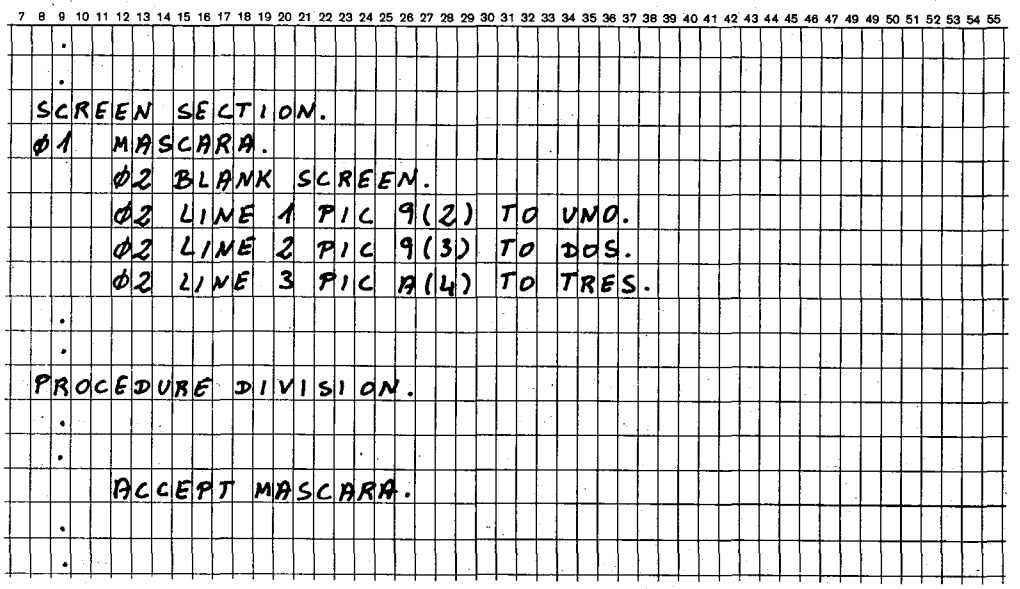
Para lograr más rapidez, puede codificarse la cláusula

AUTO

en la definición de los campos.

Con esta cláusula, cuando el operador haya completado un campo (introducidos todos los caracteres que admite), el ordenador da por finalizado el dato.

Ejemplo



```
SCREEN SECTION.
01   MASCARA.
     02 BLANK SCREEN.
     02 LINE 1 PIC 9(2) TO UNO.
     02 LINE 2 PIC 9(3) TO DOS.
     02 LINE 3 PIC A(4) TO TRES.
.
.
PROCEDURE DIVISION.
.
.
     ACCEPT MASCARA.
.
.
```

Las teclas pulsadas por el operador al ejecutarse la instrucción

ACCEPT MASCARA

para asignar a UNO el valor 14, a DOS el valor 258 y a TRES los caracteres ABCD son:

1, 4, Return, 2, 5, 8, Return, A, B, C, D, Return

Pero si el programa se codifica como el que se muestra a continuación, sólo deberá pulsar las teclas

1, 4, 2, 5, 8, A, B, C, D

Sin pulsar la tecla RETURN para indicar el final de un dato, ya que las cláusulas AUTO pasan el control del procesador al siguiente campo cuando un campo ha sido completado.

```
SCREEN SECTION.
01  MASCARA.
    02 BLANK SCREEN.
    02 LINE 1 AUTO PIC 9(2) TO UNO.
    02 LINE 2 AUTO PIC 9(3) TO DOS.
    02 LINE 3 AUTO PIC A(4) TO TRES.
```

2.15. BORRADO DE UNA LINEA

Cuando una máscara debe superponerse en una parte de la máscara que está visible en pantalla, los elementos a superponer deben borrar los posibles campos que tiene la máscara en pantalla, con el objeto de no entremezclar los campos anteriores con los campos nuevos.

Esto se logra codificando en cada elemento de la máscara a superponer la cláusula

BLANK LINE

Esta cláusula borra la línea en la que se superpone el elemento de la nueva máscara a partir de la columna en la que empieza el nuevo campo.

Ejemplo

```
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
.
.
01   MASCARA-UNO.
     02 BLANK SCREEN.
     02 LINE 5 VALUE "TEXTO DE LA".
     02 LINE 6 COL 12 VALUE "MASCARA-UNO".
01   MASCARA-DOS.
     02 LINE 6 COL 20 BLANK LINE VALUE "DOS".
.
.
PROCEDURE DIVISION.
.
. .
     DISPLAY MASCARA-UNO.
.
.
     DISPLAY MASCARA-DOS.
.
.
```

Después de ejecutarse la instrucción

DISPLAY MASCARA-UNO

La pantalla tendrá el aspecto siguiente ▶

```
TEXTO DE LA
          MASCARA-UNO
```

Después de ejecutarse la instrucción

DISPLAY MASCARA-DOS

La pantalla tendrá este aspecto ▶

```
TEXTO DE LA
          MASCARA-DOS
```

2.16. SCROLLING

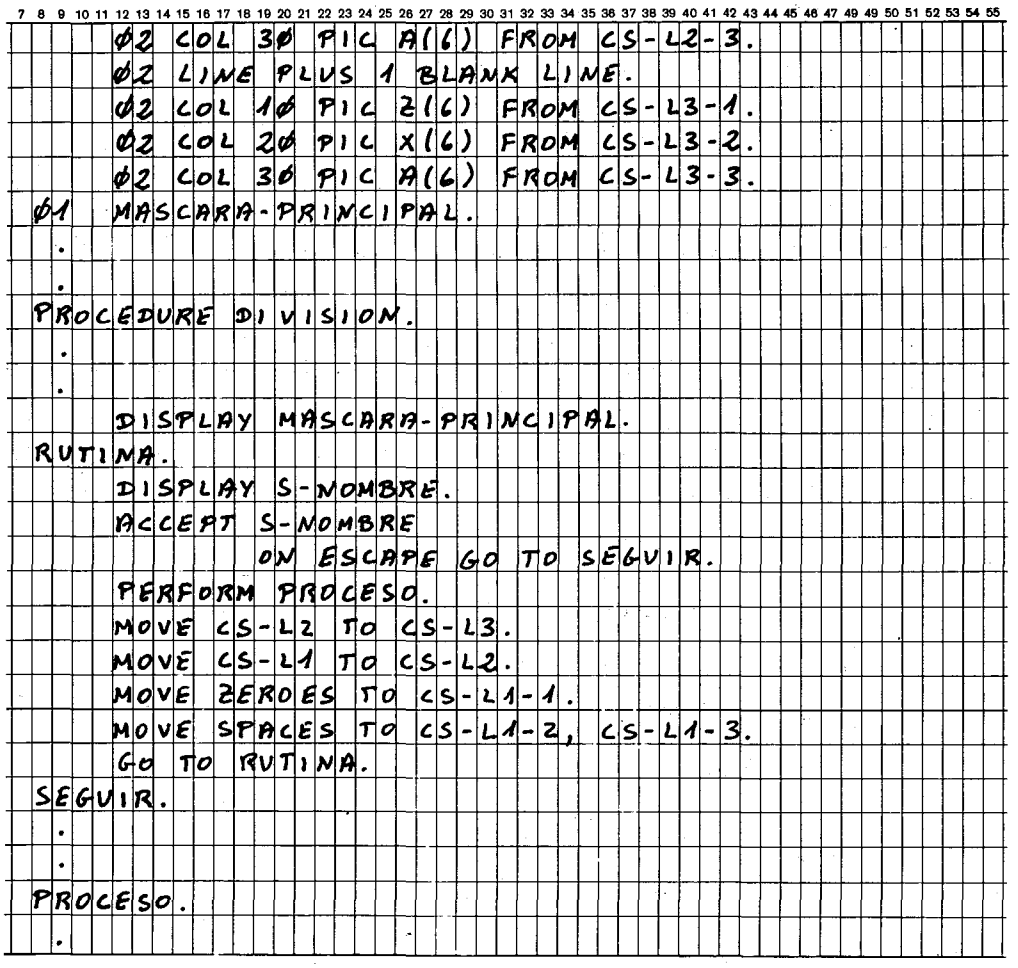
En la técnica de Máscaras, este término SCROLLING (la traducción de la palabra inglesa *Scroll* es *enrollar un pergamino*), hace referencia a la acción de mover todos los datos mostrados en pantalla una línea hacia abajo para permitir la entrada de más datos por la parte superior de la pantalla. De esta manera, se logra el efecto visual de que los nuevos datos empujan hacia abajo a los datos anteriores.

Ejemplo:

El programa siguiente realiza el Scrolling de tres campos (CS-L1-1, CS-L1-2 y CS-L1-3) a lo largo de las líneas 15, 16 y 17 de la pantalla.

```
WORKING-STORAGE SECTION.
01  CAMPO-SCROLL.
    02 CS-L1.
       03 CS-L1-1 PIC 9(6) VALUE ZEROES.
       03 CS-L1-2 PIC X(6) VALUE SPACES.
       03 CS-L1-3 PIC A(6) VALUE SPACES.
    02 CS-L2.
       03 CS-L2-1 PIC 9(6) VALUE ZEROES.
       03 CS-L2-2 PIC X(6) VALUE SPACES.
       03 CS-L2-3 PIC A(6) VALUE SPACES.
    02 CS-L3.
       03 CS-L3-1 PIC 9(6) VALUE ZEROES.
       03 CS-L3-2 PIC X(6) VALUE SPACES.
       03 CS-L3-3 PIC A(6) VALUE SPACES.

SCREEN SECTION.
01  S-NOMBRE.
    02 LINE 15 BLANK LINE.
    02 COL 10 PIC Z(6) USING CS-L1-1.
    02 COL 20 PIC X(6) USING CS-L1-2.
    02 COL 30 PIC A(6) USING CS-L1-3.
    02 LINE 16 BLANK LINE.
    02 COL 10 PIC Z(6) FROM CS-L2-1.
    02 COL 20 PIC X(6) FROM CS-L2-2.
```



```
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
           02 COL 30 PIC A(6) FROM CS-L2-3.
           02 LINE PLUS 1 BLANK LINE.
           02 COL 10 PIC Z(6) FROM CS-L3-1.
           02 COL 20 PIC X(6) FROM CS-L3-2.
           02 COL 30 PIC A(6) FROM CS-L3-3.
01   MASCARA-PRINCIPAL.
  .
  .
PROCEDURE DIVISION.
  .
  .
           DISPLAY MASCARA-PRINCIPAL.
RUTINA.
           DISPLAY S-NOMBRE.
           ACCEPT S-NOMBRE
                  ON ESCAPE GO TO SEGUIR.
           PERFORM PROCESO.
           MOVE CS-L2 TO CS-L3.
           MOVE CS-L1 TO CS-L2.
           MOVE ZEROES TO CS-L1-1.
           MOVE SPACES TO CS-L1-2, CS-L1-3.
           GO TO RUTINA.
SEGUIR.
  .
  .
PROCESO.
  .
```

En la Working-Storage Section se define CAMPO-SCROLL el cual contiene todos los campos necesarios para realizar el Scrolling a lo largo de tres líneas. Dentro de CAMPO-SCROLL hay tres divisiones (CS-L1, CS-L2 y CS-L3).

— CS-L1 contiene la definición de los campos CS-L1-1, CS-L1-2 y CS-L1-3.
— CS-L2 contiene la definición de los campos CS-L2-1, CS-L2-2 y CS-L2-3.
— CS-L3 contiene la definición de los campos CS-L3-1, CS-L3-2 y CS-L3-3.

Todos los campos están inicializados a ceros si son numéricos y a blancos en caso contrario.

En la Screen Section están definidas la MASCARA-PRINCIPAL y la máscara S-NOMBRE que es la encargada de visualizar el Scrolling dentro de la MASCARA-PRINCIPAL en las líneas 15, 16 y 17.

Los pasos seguidos en la Procedure Division son:

```
DISPLAY MASCARA-PRINCIPAL
```

En la pantalla aparece el contenido de MASCARA-PRINCIPAL.

```
ACCEPT S-NOMBRE ON ESCAPE GO TO SEGUIR
```

El operador introduce los datos que solicita el programa.

```
PERFORM PROCESO
```

Se ejecuta el párrafo PROCESO donde están codificadas las instrucciones relativas a los datos introducidos. Una vez finalizado este párrafo, se ejecutan las instrucciones que realizan el Scrolling de los datos introducidos en la línea 15

```
MOVE CS-L2 TO CS-L3
```

Se transfieren los datos displayados en la fila 16 a la fila 17.

```
MOVE CS-L1 TO CS-L2
```

Se transfieren los datos introducidos en la fila 15 a la fila 16.

```
MOVE ZEROS TO CS-L1-1
MOVE SPACES TO CS-L1-2, CS-L1-3
```

Se limpian los campos de la línea 15.

```
GO TO RUTINA
```

Se inicia de nuevo el proceso. Ahora, los datos introducidos con anterioridad aparecen en la línea 16 y los nuevos datos serán introducidos en la línea 15.

Una vez finalizado el párrafo RUTINA, cuando el proceso se realice de nuevo, los primeros datos pasarán de la línea 16 a la línea 17, los datos anteriores pasarán de la línea 15 a la línea 16, quedando la línea 15 limpia para la recepción de más datos. De esta manera se logra el efecto visual de que los nuevos datos van empujando hacia abajo los datos introducidos con anterioridad.

Cuando el operador quiera dar por finalizada la introducción de datos, pulsará la tecla ESCAPE de su teclado con lo que la instrucción

```
ACCEPT S-NOMBRE ON ESCAPE GO TO SEGUIR
```

bifurcará hacia el párrafo SEGUIR, abandonándose la rutina de Scrolling.

EJERCICIOS

1. Definir un campo en la tercera columna de la quinta línea de la pantalla.

2. ¿Qué cláusula convierte un campo en campo obligatorio?

   a) SECURE
   b) FULL
   c) REQUIRED

3. Las máscaras sirven para mantener una comunicación entre:

   a) El programa y el operador
   b) El programa y el ordenador
   c) El operador y el ordenador

4. ¿En qué División y en qué Sección se definen las máscaras?
   a) ENVIRONMENT DIVISION, SCREEN SECTION
   b) DATA DIVISION, SCREEN SECTION
   c) DATA DIVISION, WORKING-STORAGE SECTION

5. ¿Qué cláusula define un campo en la línea siete, columna cuarenta con la constante NOMBRE?

   a) LINE 7 COL 40 VALUE «NOMBRE»
   b) LINE 7 COL 40 VALUE NS NOMBRE
   c) LINE 40 COL 7 VALUE IS «NOMBRE»

6. Para modificar el valor de una variable mediante una máscara se utiliza la cláusula

   a) TO
   b) FROM
   c) USING

7. Con la cláusula SECURE, en pantalla

   a) Los caracteres no aparecen en la pantalla
   b) Los caracteres son sustituidos por asteriscos
   c) Se logra un display internamente

8. Para eliminar el uso de la tecla RETURN, se emplea la cláusula

   a) BELL
   b) BLINK
   c) AUTO

3.
TRATAMIENTO DE FICHEROS

CUADRO RESUMEN DE LA LECCION

- **Tratamiento de ficheros**
  - Definición de un fichero
    - Soporte físico y nombre
    - Etiqueta
    - Longitud de los bloques
    - Longitud de los registros
    - Descripción de los registros
    - Detección de errores
    - Buffer compartido
  - Organización de ficheros
  - Acceso a los ficheros
    - Acceso secuencial
      - Apertura
      - Lectura
      - Grabación
      - Cierre
      - Modificación
      - Ampliación
      - Desbloqueo
    - Acceso directo
      - Apertura
      - Lectura
      - Grabación
      - Cierre
      - Modificación
      - Ampliación
      - Desbloqueo
    - Acceso indexado
  - Eliminación de registros
  - Tratamiento de errores

En este capítulo se explican las instrucciones necesarias y las técnicas adecuadas para el tratamiento de los ficheros de acuerdo al tipo al que pertenecen.

3.1. DEFINICION DE UN FICHERO

Para definir un fichero dentro de un programa, son necesarios los siguientes datos:

- El soporte físico de información en el que está almacenado.
- El nombre del fichero.
- Si el fichero posee etiqueta.
- Tipo de organización.
- Qué tipo de acceso se realizará con sus registros.
- El número de registros que contiene cada bloque.
- Si los registros son de longitud fija o variable.
- La longitud, en caracteres, de cada registro.
- El nombre que tendrán los registros del fichero dentro del programa.
- La descripción de los campos del registro.
- Si el acceso a los registros del fichero no es secuencial, el nombre que tendrá la KEY del fichero dentro del programa.
- El nombre del campo de memoria dentro del programa destinado a almacenar el código de cualquier error que surja a lo largo del tratamiento del fichero.
- Si el área de memoria asignada para la entrada o salida de registros de un fichero dentro del programa es compartida por otro fichero.

3.1.1. Soporte físico y nombre

El soporte físico de información en el que está el fichero y el nombre del fichero se especifican mediante la cláusula

SELEC *nombre-fichero* ASSIGN TO *dispositivo, nombre hardware*

siendo:

Nombre-fichero = puede ser cualquier nombre que siga las reglas del COBOL.

Dispositivo = es el tipo de soporte físico:

DISK	Disco
PRINTER	Impresora
DISPLAY	Pantalla
KEYBOARD	Teclado

Nombre-hardware = Es el nombre con el que el ordenador identifica el soporte físico de información.

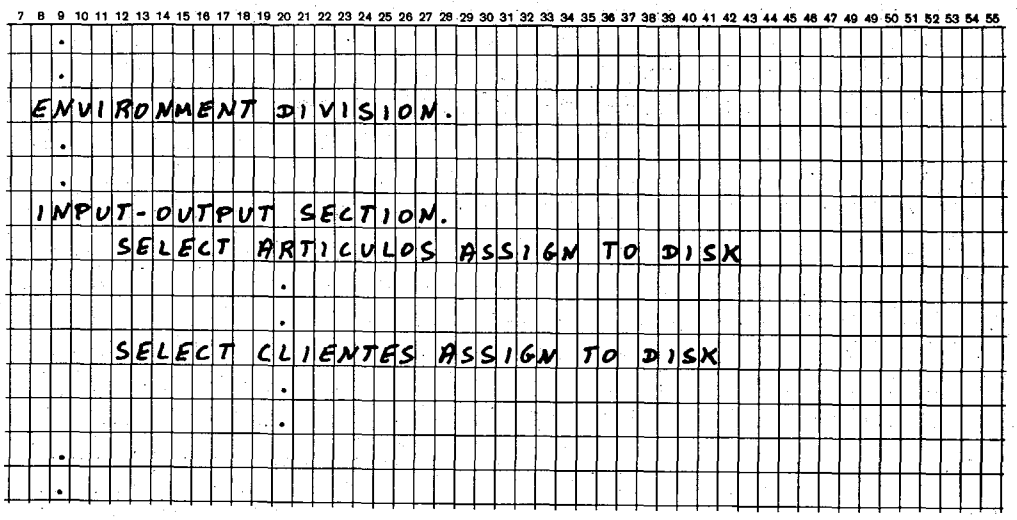
La codificación de este campo no es obligatoria, pero su inclusión aumenta la velocidad del ordenador ya que se indica en qué disco (por ejemplo) se encuentra el fichero, con lo que se evita que el ordenador busque el fichero en todos los discos hasta que lo encuentre.

Estos nombres también varían según el modelo y el fabricante del ordenador. A título de ejemplo se especifican algunos nombres de hardware para soportes físicos de información de DATA GENERAL:

nombre hardware	*soporte físico*
DZ x	Diskpack
MT x	Cinta magnética
$ TTO x	Display
$ TTI x	Keyboard
$ LPT x	Impresora
DP x	Discos y diskettes

siendo x el número de cada unidad.

Ejemplo:



— Codificación de un fichero con el nombre ARTICULOS y de un fichero con el nombre CLIENTES almacenados en la unidad de discos magnéticos.

3.1.2. Etiqueta

Para indicar si un fichero posee etiqueta de identificación se utiliza la cláusula

LABEL RECORD IS { STANDARD / OMITED }

Ejemplo:

En el ejemplo superior de la página siguiente se muestra la codificación correspondiente al fichero ARTICULOS con etiqueta y al fichero CLIENTES sin etiqueta.

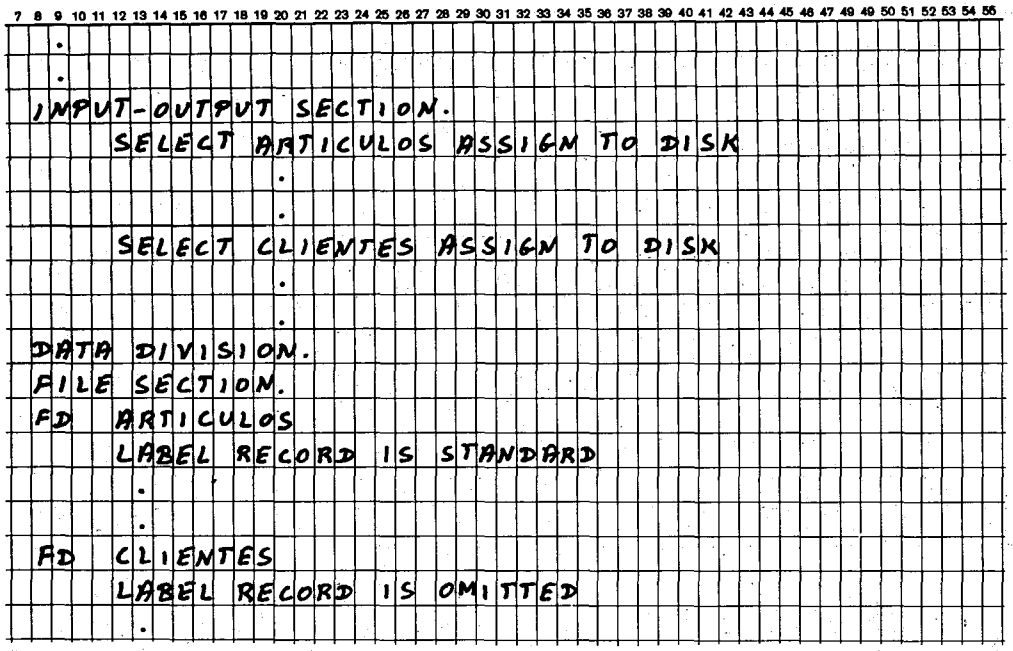
3.1.3. Longitud de los bloques

Para indicar la cantidad de registros que contiene cada bloque se utiliza la cláusula

BLOCK CONTAINS n RECORD[S]

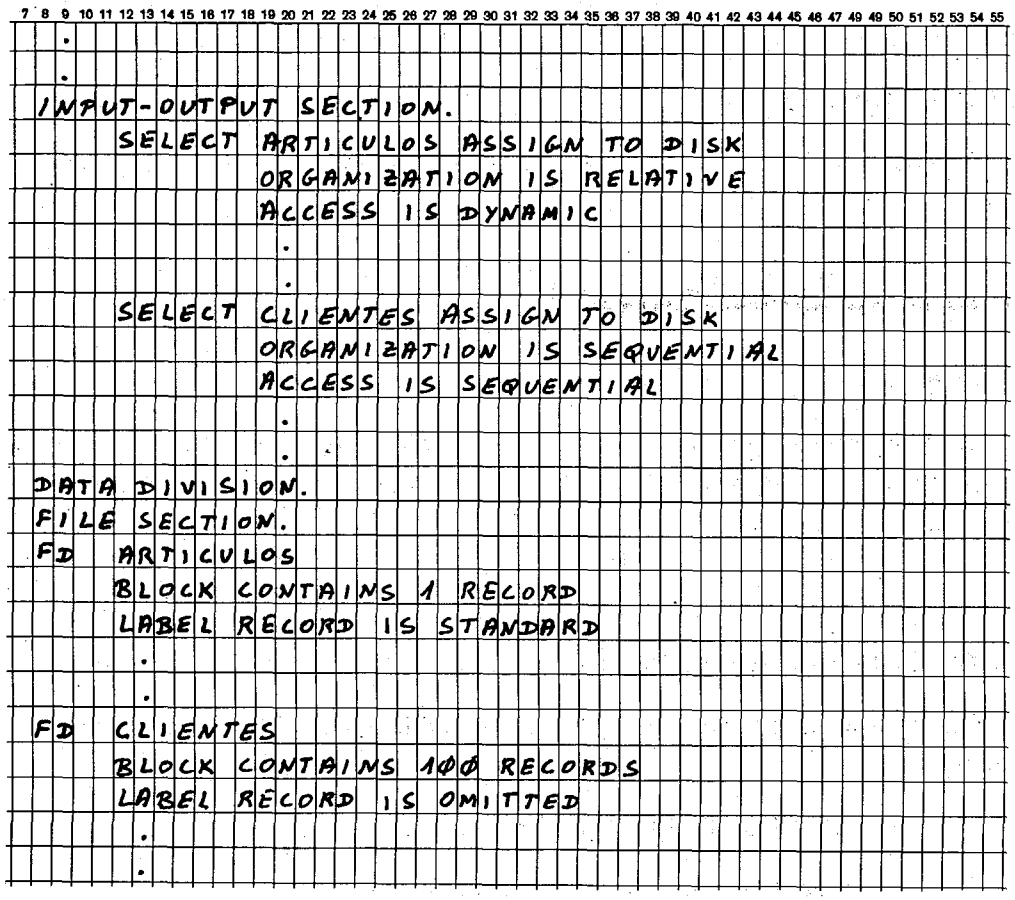
En donde n es un número entero y positivo.

Ejemplo de etiqueta



```
      .
      .
INPUT-OUTPUT SECTION.
      SELECT ARTICULOS ASSIGN TO DISK
                  .
                  .
      SELECT CLIENTES ASSIGN TO DISK
                  .
                  .
DATA DIVISION.
FILE SECTION.
FD    ARTICULOS
      LABEL RECORD IS STANDARD
        .
        .
FD    CLIENTES
      LABEL RECORD IS OMITTED
        .
```

Ejemplo de longitud



```
      .
      .
INPUT-OUTPUT SECTION.
      SELECT ARTICULOS ASSIGN TO DISK
                  ORGANIZATION IS RELATIVE
                  ACCESS IS DYNAMIC
                  .
                  .
      SELECT CLIENTES ASSIGN TO DISK
                  ORGANIZATION IS SEQUENTIAL
                  ACCESS IS SEQUENTIAL
                  .
                  .
DATA DIVISION.
FILE SECTION.
FD    ARTICULOS
      BLOCK CONTAINS 1 RECORD
      LABEL RECORD IS STANDARD
        .
        .
FD    CLIENTES
      BLOCK CONTAINS 100 RECORDS
      LABEL RECORD IS OMITTED
        .
        .
```

Ejemplo:

El ejemplo inferior de la página 91 corresponde al fichero ARTICULOS, no bloqueado, y al fichero CLIENTES, con cien registros por bloque.

3.1.4. Longitud de los registros

La longitud de todos los registros de un fichero puede ser:

— fija
— variable

y se indica mediante la cláusula

RECORDING MODE IS { FIXED / VARIABLE }

Si la longitud es FIXED (fija), todos los registros poseen la misma longitud y si es VARIABLE la longitud de los registros es distinta de unos a otros.

Los registros pueden ser de longitud variable solamente en ficheros organizados secuencialmente y almacenados en disco magnético.

La longitud de cada registro se mide en caracteres.

Si los registros son de longitud fija, se utiliza la cláusula

RECORD CONTAINS n CHARACTERS

siendo:

n = longitud en caracteres del registro.

Si los registros son de longitud variable, se utiliza la cláusula

RECORD CONTAINS n.º 1 TO n.º 2 CHARACTERS

siendo:

n.º 1 = longitud del registro más pequeño.
n.º 2 = longitud del registro más grande.

La longitud máxima de los registros depende de la unidad a la que está asignado el fichero:

— 132 bytes por registro en ficheros asignados a: impresoras, teclados o pantallas.

```
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 49 50 51 52 53 54 55
                .
                .
           SELECT ARTICULOS ASSIGN TO DISK
                  ORGANIZATION IS RELATIVE
                  ACCESS IS DYNAMIC
                    .
                    .
           SELECT CLIENTES ASSIGN TO DISK
                  ORGANIZATION IS SEQUENTIAL
                  ACCESS IS SEQUENTIAL
                    .
                    .
           SELECT ALBARANES ASSIGN TO DISK
                  ORGANIZATION IS SEQUENTIAL
                  ACCESS IS SEQUENTIAL
                    .
                    .
             .
             .
       FD    ARTICULOS
             BLOCK CONTAINS 1 RECORD
             RECORD CONTAINS 45 CHARACTERS
             RECORDING MODE IS FIXED
             LABEL RECORD IS STANDARD
               .
               .
       FD    CLIENTES
```

```
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 49 50 51 52 53 54 55
             BLOCK CONTAINS 100 RECORDS
             RECORD CONTAINS 180 CHARACTERS
             RECORDING MODE IS FIXED
             LABEL RECORD IS OMITTED
               .
               .
       FD    ALBARANES
             BLOCK CONTAINS 50 RECORDS
             RECORD CONTAINS 20 TO 500 CHARACTERS
             RECORDING MODE IS VARIABLE
             LABEL RECORD IS OMITTED
               .
               .
```

— 80 bytes por registro en ficheros asignados a lectoras o perforadoras de fichas.

— 4096 bytes por registro en ficheros asignados a: cintas magnéticas, discos,... etc.

Ejemplo:

En la página anterior se muestra la codificación correspondiente a los ficheros ARTICULOS y CLIENTES y un nuevo fichero, de nombre ALBARANES, sin etiqueta, de 50 registros por bloque también en la unidad de discos

— ARTICULOS: longitud fija de 45 caracteres por registro.

— CLIENTES: longitud fija de 180 caracteres por registro.

— ALBARANES: longitud variable entre 20 y 500 caracteres por registro.

```
FD  ARTICULOS
    LABEL RECORD IS STANDARD
    BLOCK CONTAINS 1 RECORD
    RECORDING MODE IS FIXED
    RECORD CONTAINS 45 CHARACTERS
    DATA RECORD IS REG-ART.
01  REG-ART.
      .
      .
FD  CLIENTES
    LABEL RECORD IS OMITTED
    BLOCK CONTAINS 100 RECORDS
    RECORD CONTAINS 180 CHARACTERS
    RECORDING MODE IS FIXED
    DATA RECORDS ARE REG-CLI-1
                     REG-CLI-2
                     REG-CLI-3.
01  REG-CLI-1.
      .
      .
01  REG-CLI-2.
      .
      .
01  REG-CLI-3.
      .
```

3.1.5. Descripción de los registros

Para describir los registros de un fichero en un programa hay que conocer:

— Nombre de cada registro.

— Descripción de los campos de cada registro.

El nombre de cada uno de los registros del fichero se indica en la cláusula.

DATA RECORD IS nombre-1, nombre-2, ...

La descripción de los campos de cada registro se realiza mediante los números de nivel o de jerarquía.

Ejemplo:

Fichero ARTICULOS: un sólo tipo de registro, con el nombre

REG-ART

Fichero CLIENTES: tres tipos de registros, con los nombres

REG-CLI-1, REG-CLI-2 y REG-CLI-3.

3.1.6. Detección de errores

Cuando aparece un error en el tratamiento de un fichero, el ordenador genera un código numérico que indica el tipo de error que se ha producido.

Un programa en el que no esté prevista la detección y tratamiento de estos errores, será cancelado por el ordenador cuando ocurra cualquier anomalía. Para prevenir esto, se describe un campo en memoria con el formato siguiente:

01 nombre-campo PIC XX.

y se indica al ordenador que almacene en dicho campo el código del error del fichero mediante la cláusula:

FILE STATUS IS nombre-campo

Ejemplo:

En la siguiente codificación se define el campo

ERR-FICH-ART

destinado a almacenar el código de un posible error del fichero ARTICULOS y el campo

ERR-FICH-CLI

destinado a almacenar el código de un posible error del fichero CLIENTES.

```
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 49 50 51 52 53 54 55
                .
                .
           SELECT ARTICULOS ASSIGN TO DISK
                  ORGANIZATION IS ..........
                  ACCESS IS ..........
                  FILE STATUS IS ERROR-FICH-ART
                    .
                    .
           SELECT CLIENTES ASSIGN TO DISK
                  ORGANIZATION IS ..........
                  ACCESS IS ..........
                  FILE STATUS IS ERROR-FICH-CLI
                    .
                    .
DATA DIVISION.
    .
    .
WORKING-STORAGE SECTION.
01   ERROR-FICH-ART PIC XX.
01   ERROR-FICH-CLI PIC XX.
    .
    .
```

Si durante la ejecución del programa aparece un error en el fichero ARTICULOS, el campo ERR-FICH-ART tendrá almacenado el código del error y si el error es en el fichero CLIENTES, el campo ERR-FICH-CLI contendrá el código del error correspondiente a este fichero.

3.1.7. Buffer compartido

Cuando varios ficheros de un programa comparten la misma zona de memoria del ordenador para realizar las lecturas y/o grabaciones de los registros, se indica mediante la cláusula

SAME RECORD AREA FOR fichero-1, fichero-2,...

Ejemplo:

Los ficheros ALBARANES y RECIBOS de la codificación siguiente comparten el mismo Buffer.

```
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 49 50 51 52 53 54 55
      .
      .
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
      SELECT ALBARANES ..........
                       .
                       .
      SELECT RECIBOS ..........
                       .
                       .
  I-O-CONTROL.
      SAME RECORD AREA FOR ALBARANES
                           RECIBOS.
      .
      .
```

Cuando varios ficheros comparten el mismo Buffer, solamente uno de ellos puede estar activo, debiendo permanecer inactivos los demás.

Esta técnica debe utilizarse con cuidado ya que puede originar errores en el tratamiento de los ficheros que comparten el mismo Buffer.

Sólo se recomienda que varios ficheros compartan el mismo Buffer, cuando el ordenador no posea la suficiente memoria para poder soportar todos los ficheros que necesita un programa.

3.2. ORGANIZACION

La organización de los registros de un fichero pueden ser de tres tipos:

— Secuencial
— Relativa
— Indexada

La cláusula que indica el tipo de organización es la cláusula

ORGANIZATION IS { SEQUENTIAL / RELATIVE / INDEXED }

Ejemplo:

A continuación se muestra la codificación correspondiente al fichero ARTICULOS organizado de forma relativa y al fichero CLIENTES organizado secuencialmente.

Si la organización es secuencial, los registros están físicamente uno a continuación del otro.

Si la organización es relativa, existe una clave (KEY) numérica que indica la posición que ocupa el registro dentro del fichero.

Y si la organización es indexada, existe una clave alfanumérica para referenciar a cada registro.

```
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 49 50 51 52 53 54 55
           .
           .
       INPUT-OUTPUT SECTION.
           SELECT ARTICULOS ASSIGN TO DISK
                  ORGANIZATION IS RELATIVE
                  .
                  .
           SELECT CLIENTES ASSIGN TO DISK
                  ORGANIZATION IS SEQUENTIAL
                  .
                  .
```

Si un fichero está organizado secuencialmente, sólo se puede acceder a sus registros de manera secuencial (Fig. 3.1.).

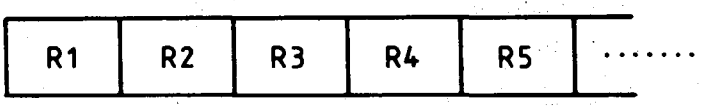


Fig. 3.1

Si un fichero está organizado de manera relativa, se puede acceder a sus registros, bien indicando en la KEY la posición que ocupa el registro deseado (acceso directo), bien de manera secuencial (Fig. 3.2).

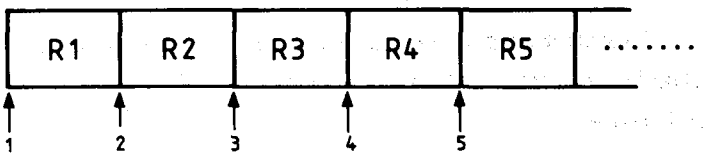


Fig. 3.2

Y si un fichero está organizado de manera indexada, el ordenador crea una tabla en la que asocia la clave alfanumérica correspondiente a cada registro con su posición dentro del fichero, con lo que el acceso puede ser o indexado o secuencial (Fig. 3.3).

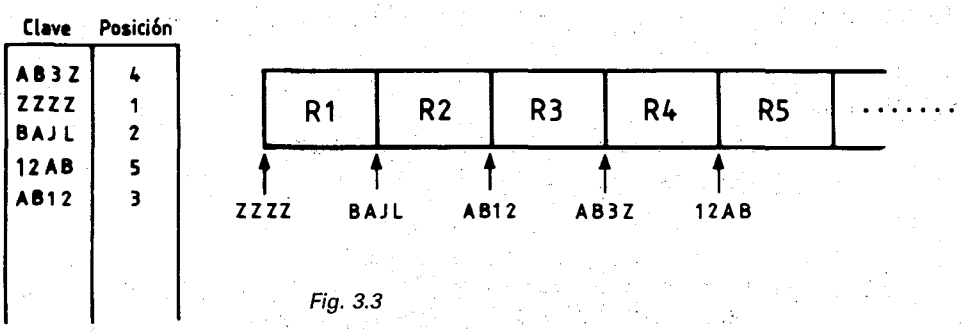


Fig. 3.3

Cuando un fichero no esté organizado de manera secuencial, pero vaya a ser tratado en acceso secuencial, antes de ejecutarse la primera instrucción de lectura debe ejecutarse la instrucción

START fichero KEY = campo-1

siendo:

Fichero = Nombre del fichero
Campo = El nombre de un campo de memoria

Con esta instrucción se inicializa el campo destinado a KEY con el valor almacenado en campo-1 y el ordenador buscará el siguiente registro (en orden secuencial) para cada instrucción de lectura.

3.3. ACCESO

El tipo de acceso en el que se procesarán los registros de un fichero se indica mediante la cláusula

ACCES MODE IS { SEQUENTIAL / RANDOM / DYNAMIC }

A un fichero organizado secuencialmente, sólo se puede acceder a sus registros en forma secuencial.

A un fichero organizado de manera relativa o indexada, se puede acceder de cualquier manera.

La codificación siguiente corresponde al fichero ARTICULOS, con acceso dinámico y al fichero CLIENTES, con acceso secuencial.

```
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 49 50 51 52 53 54 55
                .
                .
           SELECT  ARTICULOS ASSIGN TO DISK
                   ORGANIZATION IS RELATIVE
                   ACCESS IS DYNAMIC
                     .
                     .
           SELECT  CLIENTES ASSIGN TO DISK
                   ORGANIZATION IS SEQUENTIAL
                   ACCESS IS SEQUENTIAL
                     .
```

3.3.1. Acceso secuencial

Para tratar de manera secuencial los registros de un fichero, debe incluirse la cláusula

ACCESS IS SEQUENTIAL

El fichero puede estar almacenado en cualquier tipo de soporte físico de información.

Todo fichero, independientemente de su organización, puede ser activado para lectura de sus registros en modo secuencial.

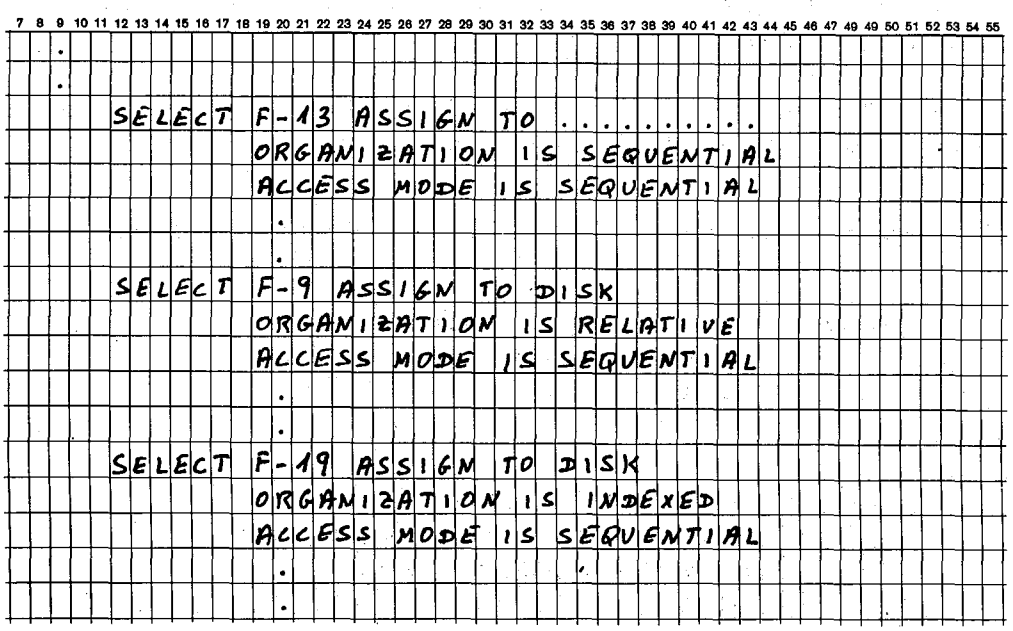
Pero si el fichero no existe, al ser creado en acceso secuencial, también debe ser organizado de manera secuencial mediante la cláusula

ORGANIZATION IS SEQUENTIAL

con lo que en tratamientos posteriores se utilizarán siempre las cláusulas

ACCESS IS SEQUENTIAL
ORGANIZATION IS SEQUENTIAL

Ejemplo:



Las conclusiones que se obtienen de la anterior codificación son:

El fichero F-13 o bien fue creado de manera secuencial, con lo que su tratamiento debe ser secuencial, o bien será creado por el programa con una organización secuencial.

El fichero F-9 será activado sólo para lectura ya que fue creado con organización relativa y en este programa es tratado secuencialmente.

El fichero F-19 también será activado sólo para lectura ya que está organizado de manera indexada y será tratado de manera secuencial.

Apertura de un fichero secuencial

Cuando un fichero es tratado de manera secuencial, sólo puede estar activo o bien para lectura o bien para grabación. Para activar un fichero secuencial se utiliza la instrucción

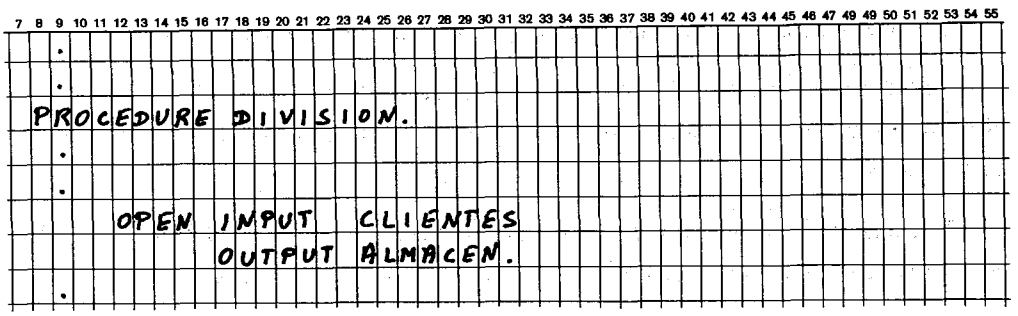
OPEN [EXCLUSIVE] { INPUT / OUTPUT } fichero-1, { INPUT / OUTPUT } fichero-2,...

Ejemplo-1:

```
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 49 50 51 52 53 54 55
           .
           .
PROCEDURE DIVISION.                                                                                                                          .
           .
           .
               OPEN INPUT CLIENTES.
               OPEN OUTPUT ALMACEN.
           .
```

En este ejemplo se muestra la codificación correcta de la apertura del fichero CLIENTES para la lectura de sus registros y la apertura del fichero ALMACEN para la grabación de registros.

Esta codificación es equivalente a la de la siguiente.



Cuando se activa un fichero como INPUT, el ordenador busca el fichero y comprueba todos los parámetros codificados en las cláusulas que los definen, generando un código de error en el caso de que detecte cualquier anormalidad. Si un fichero se activa en modo OUTPUT, pueden ocurrir dos cosas:

— Si el fichero existe, el ordenador comprueba los parámetros que lo definen (igual que en modo INPUT).

— Si el fichero no existe, el ordenador crea el fichero de acuerdo a los datos codificados en las cláusulas que lo definen.

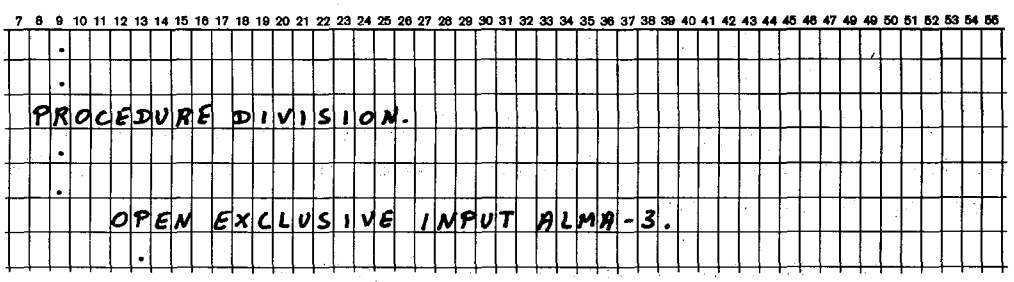
La cláusula EXCLUSIVE indica que el fichero sólo puede ser activado por un programa, no permitiendo que ningún otro usuario pueda tener dicho fichero activo en su programa mientras esté activo en el programa que incluye la cláusula EXCLUSIVE.

Ejemplo-2:

En la codificación siguiente, a partir del instante en el que se ejecute la instrucción

OPEN EXCLUSIVE INPUT ALMA-3

ningún otro usuario podrá activar el fichero ALMA-3.



Lectura secuencial de registros

La orden de lectura de un registro en acceso secuencial siempre proporciona el segundo registro al último registro tratado.

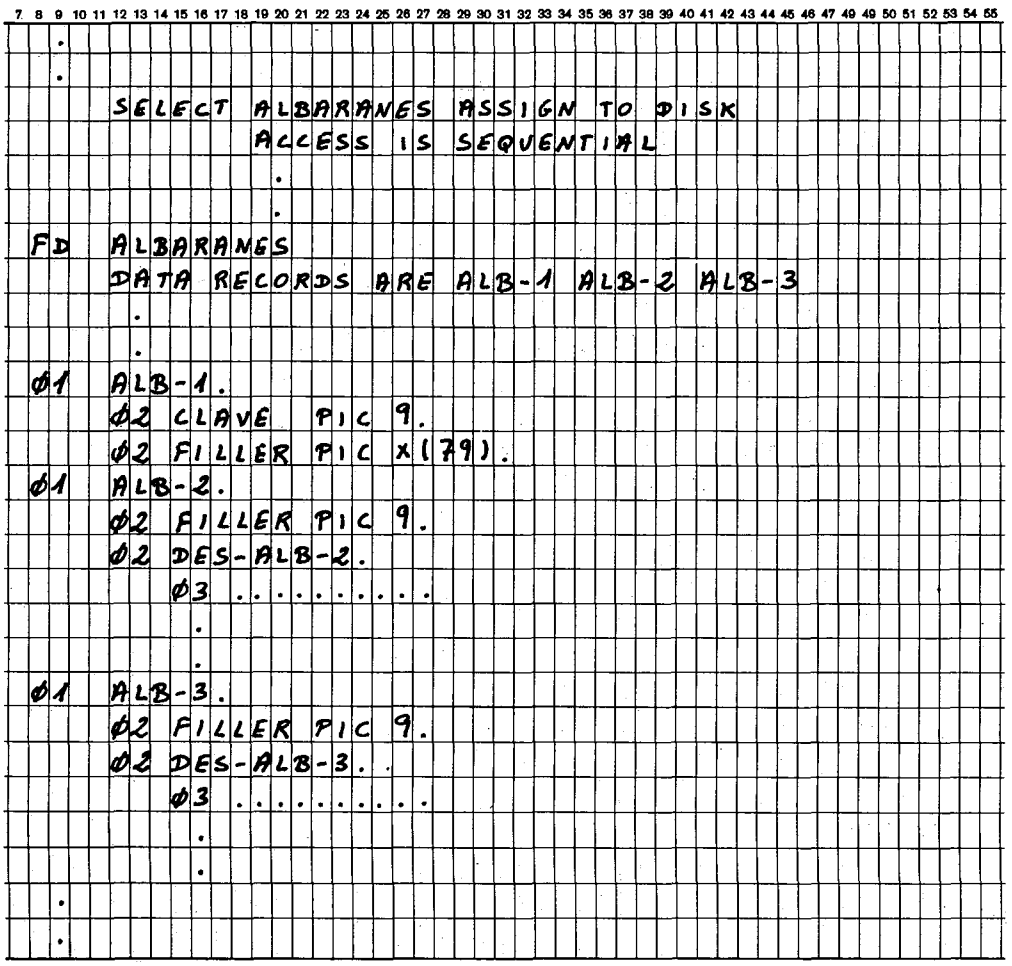
En el caso de la primera lectura, se obtendrá el primer registro del fichero, a la siguiente lectura se obtendrá el siguiente registro, y así sucesivamente para todas las lecturas que se efectúen en el fichero.

Para leer los registros de un fichero, el fichero debe estar activo en modo INPUT.

La instrucción para efectuar la lectura es:

READ nombre-fichero [INTO nombre-campo] [AT END instrucción]

Ejemplo-1



Se tiene un fichero de nombre ALBARANES con un número indeterminado de registros. Por lo tanto, se incluye la cláusula

AT END GO TO FIN-ALB

ya que una vez leído el último registro, la siguiente orden de lectura daría un error al detectarse el final del fichero. Con esta cláusula cuando se intente leer más allá del final del fichero el control del procesador bifurcará al párrafo FIN-ALB.

Existen dos tipos de registros definidos mediante ALB-2 y ALB-3. El primer carácter de cada registro es un dígito que indica el tipo de albarán que contiene el registro, por lo que se ha definido un tercer registro (ALB-1) que contempla sólo el primer carácter bajo el nombre CLAVE.

```
PROCEDURE DIVISION.
.
.
    OPEN INPUT ALBARANES.
      .
      .
BUCLE.
    READ ALBARANES AT END GO TO FIN-ALB.
    IF CLAVE ≠ 2 OR
       CLAVE ≠ 3 PERFORM ALB-ERROR.
    IF CLAVE = 2 PERFORM ALB-TIPO-2.
    IF CLAVE = 3 PERFORM ALB-TIPO-3.
    GO TO BUCLE.
.
.
ALB-TIPO-2.
    MOVE DES-ALB-2 TO ..........
      .
      .
ALB-TIPO-3.
    MOVE DES-ALB-3 TO ..........
      .
      .
ALB-ERROR.
      .
      .
FIM-ALB.
```

Una vez leído el registro la instrucción

```
IF CLAVE # 1 OR
    CLAVE # 2 PERFORM ALB-ERROR
```

bifurca hacia el párrafo ALB-ERROR en el caso de que la clave del registro leído no sea ni la 2 ni la 3, ya que en el programa se considera que un registro con una clave distinta a éstas es un albarán erróneo.

La instrucción

```
IF CLAVE = 2 PERFORM ALB-TIPO-2
```

bifurca hacia el párrafo ALB-TIPO-2 donde se procesan los registros con clave igual a 2 tomándose los campos de la definición de ALB-2 y la instrucción

```
IF CLAVE = 3 PERFORM ALB-TIPO-3
```

bifurca hacia el párrafo ALB-TIPO-3 donde se procesan los registros con clave igual a 3, que corresponden con la definición de ALB-3.

Ejemplo-2

La codificación de la página siguiente corresponde al tratamiento del mismo fichero utilizando la cláusula INTO nombre-campo.

La instrucción

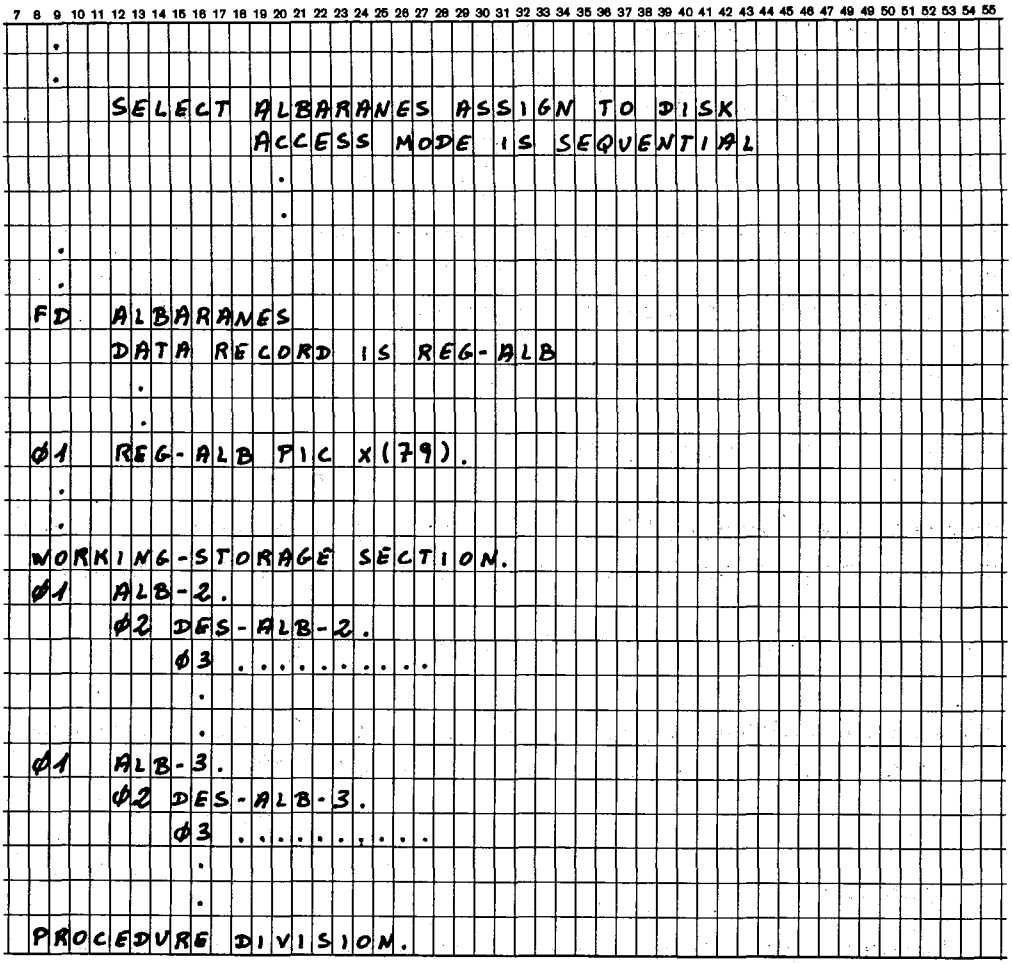
```
READ REG-ALB INTO ALB-2
```

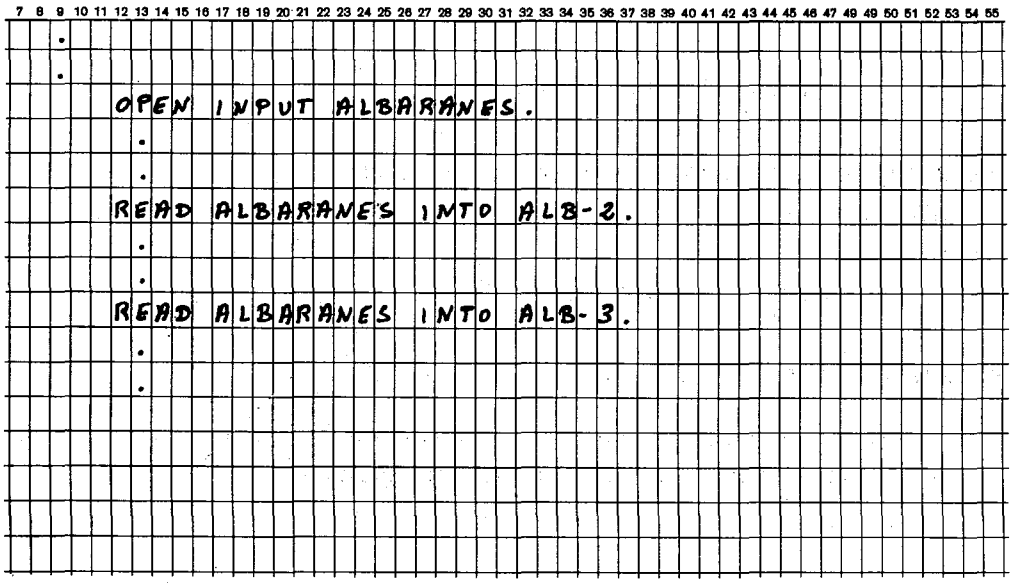
lee un registro del fichero ALBARANES y lo almacena en el campo ALB-2 y la instrucción

```
READ REG-ALB INTO ALB-3
```

obtiene un registro y lo almacena en el campo ALB-3.

Esta técnica puede utilizarse solamente cuando, habiendo más de un tipo de registro, se conoce el orden en el que están grabados.


SELECT ALBARANES ASSIGN TO DISK
ACCESS MODE IS SEQUENTIAL
FD ALBARANES
DATA RECORD IS REG-ALB
01 REG-ALB PIC X(79).
WORKING-STORAGE SECTION.
01 ALB-2.
02 DES-ALB-2.
03 .....
01 ALB-3.
02 DES-ALB-3.
03 .....
PROCEDURE DIVISION.


OPEN INPUT ALBARANES.
READ ALBARANES INTO ALB-2.
READ ALBARANES INTO ALB-3.

Grabación secuencial de registros

La instrucción para grabar un registro es

```
WRITE nombre-registro [FROM nombre-campo]
```

Los registros que se vayan grabando en un fichero, irán colocados uno a continuación del otro.

Para grabar registros en un fichero, debe estar activo en modo OUTPUT.

Ejemplo-1:

En la página siguiente se muestra la codificación correspondiente a la grabación de dos tipos de registros (ALB-2 y ALB-3) en el fichero ALBARANES.

Después de la activación del fichero para grabación mediante la instrucción

```
OPEN OUTPUT ALBARANES.
```

Las instrucciones

```
MOVE 2 TO CLAVE-2
MOVE 3 TO CLAVE-3
```

asignan los valores correspondientes a las claves de cada tipo de registro para que en la posterior lectura de este fichero (ver ejemplo-1 del apartado 3.2.2) cada registro pueda ser identificado y tratado correctamente en función del tipo al que pertenece.

La instrucción

```
WRITE ALB-2
```

graba un registro en el fichero ALBARANES a continuación del último registro grabado de acuerdo a los campos que describen el registro de nombre ALB-2. Anteriormente a la orden de grabación, el programador debe haber codificado las instrucciones correspondientes a la asignación de datos a los campos de DES-ALB-2.

La instrucción

```
WRITE ALB-3
```

graba un registro de acuerdo a los datos almacenados en las descripciones de ALB-3.

```
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 49 50 51 52 53 54 55
      .
      .
          SELECT ALBARANES ASSIGN TO DISK
                 ACCESS IS SEQUENTIAL
                    .
                    .
      .
      .
FD    ALBARANES
      DATA RECORDS ARE ALB-2 ALB-3
         .
         .
01    ALB-2.
      02 CLAVE-2 PIC 9.
      02 DES-ALB-2.
         03 . . . . . .
            .
            .
01    ALB-3.
      02 CLAVE-3 PIC 9.
      02 DES-ALB-3.
         03 . . . . . .
            .
            .
      .
      .
PROCEDURE DIVISION.
```

```
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 49 50 51 52 53 54 55
      .
      .
          OPEN OUTPUT ALBARANES.
          MOVE 2 TO CLAVE-2.
          MOVE 3 TO CLAVE-3.
            .
            .
          WRITE ALB-2.
            .
            .
          WRITE ALB-3.
            .
            .
```

```
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 49 50 51 52 53 54 55
                                                                                                                                                  
                                                                                                                                                  
                                                                                                                                                  
```

Ejemplo-2

En este caso se ha codificado el método a seguir al usar la cláusula FROM nombre-campo (ver página 110).

En la FD se define el nombre del registro y su longitud y en la Working-Storage Section se definen cada uno de los diferentes tipos de registros (ALB-2 y ALB-3).

La instrucción

```
WRITE REG-ALB FROM ALB-2
```

graba un registro en el fichero ALBARANES de acuerdo a los datos almacenados en ALB-2 y la instrucción

```
WRITE REG-ALB FROM ALB-3
```

graba un registro con los datos almacenados bajo el nombre ALB-3.

Cierre de un fichero

Cuando esté finalizado el tratamiento de los registros de un fichero, el fichero debe ser desactivado (o cerrado) mediante la instrucción

```
CLOSE  fichero-1   [WITH LOCK]   [WITH {NO} REWIND],
       fichero-2   [WITH LOCK]   [WITH {NO} REWIND],...
```

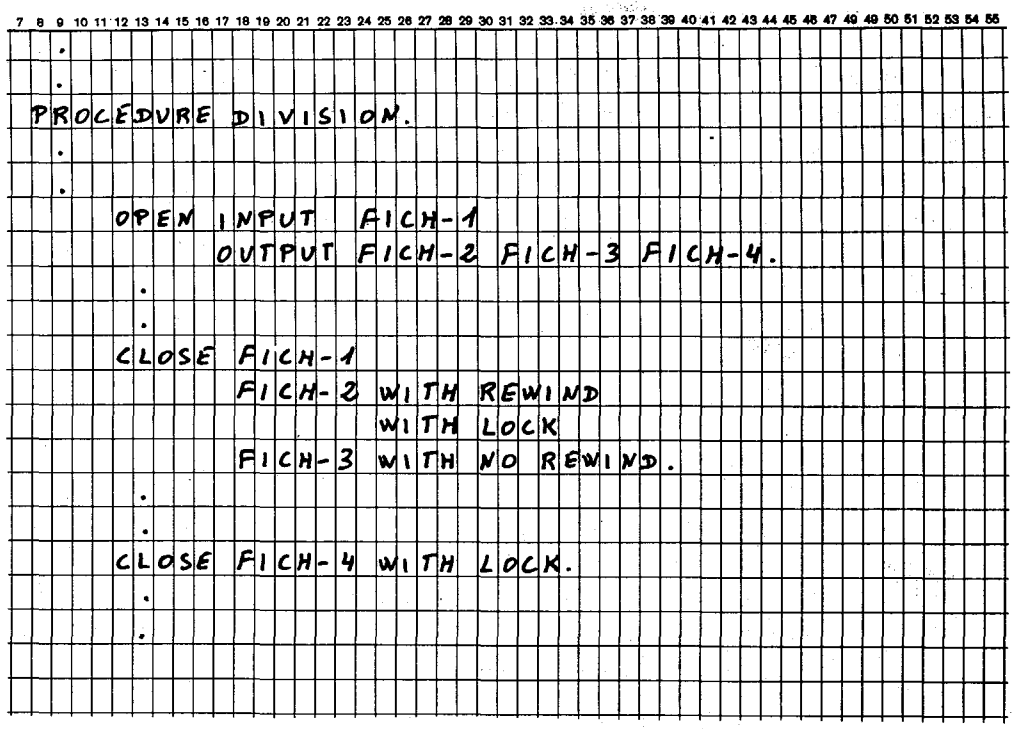
Si se utiliza la cláusula WITH LOCK, el fichero no podrá ser activado de nuevo por el programa.

Las cláusulas WITH REWIND y WITH NO REWIND se utilizan únicamente para ficheros en cinta magnética. Con la primera de ellas la cinta será rebobinada a la vez que el fichero es cerrado y con la segunda, el fichero es desactivado pero la cinta no es rebobinada.

Ejemplo:

En esta codificación (ver página siguiente) se activa el fichero FICH-1 para lectura y los ficheros FICH-2, FICH-3 y FICH-4 para grabación.

Los ficheros FICH-2 y FICH-3 están en cinta magnética.



Una vez finalizado el tratamiento de los registros de los ficheros FICH-1, FICH-2 y FICH-3, son desactivados mediante la instrucción

```
CLOSE  FICH-1
       FICH-2 WITH REWIND WITH LOCK
       FICH-3 WITH NO REWIND
```

quedando como se detalla:

FICH-1: puede volver a ser activado dentro del programa.

FICH-2: no puede ser activado de nuevo dentro del programa y la cinta es rebobinada.

FICH-3: puede ser activado de nuevo y la cinta no es rebobinada.

Después de ejecutada la instrucción

```
CLOSE FICH-4 WITH LOCK
```

el fichero FICH-4 queda desactivado, no pudiendo ser activado de nuevo dentro del programa.

Modificación de un fichero secuencial

La técnica a seguir para modificar los registros de un fichero secuencial es la creación de un nuevo fichero. Los registros que no se han de modificar se pasan del fichero *viejo* (fichero a modificar) al fichero *nuevo* (fichero con las modificaciones) y los registros a modificar son leídos en el fichero viejo, modificados dentro del programa y grabados en el fichero nuevo con las modificaciones efectuadas.

Ejemplo:

Destruir en el fichero ALB-2 los registros cuya clave sea el número 3, modificar los registros cuya clave sea el número 4 cambiéndola por el número 7 y dejar intactos los registros que posean CLAVE distinta a 3 o a 4.

El tratamiento seguido es el que se muestra en la codificación presentada en las páginas siguientes.

Se crea un nuevo fichero con el nombre ALB-2-1 que será el fichero con las modificaciones efectuadas en los registros del fichero ALB-2.

En el párrafo ABRIR-FICHEROS se activa el fichero ALB-2 con uso exclusivo, con el fin de que ningún otro operador utilice los datos del fichero viejo y se activa el fichero ALB-2-1 para grabación.

Como este fichero no existe, el ordenador lo creará según las cláusulas que lo definen (con etiqueta o sin etiqueta,... etc.)

En el párrafo PROCESO, la instrucción

```
READ ALB-2 AT END GO TO FIN-PROCESO
```

lee un registro del fichero viejo, bifurcando al párrafo CUATRO cuando hayan sido tratados todos sus registros.

La instrucción

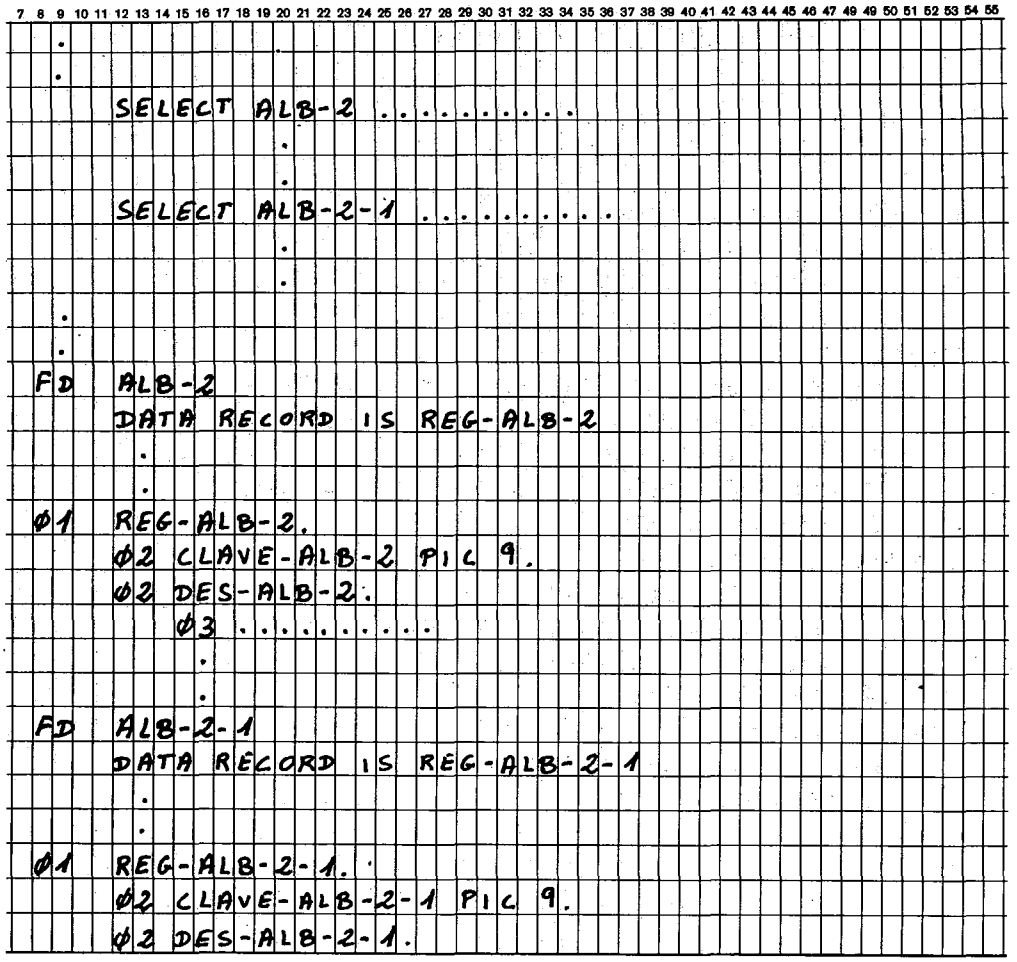
```
IF CLAVE-ALB-2 = 3 THEN GO TO PROCESO
```

bifurca de nuevo a la instrucción de lectura si la clave del registro es 3 (para obtener un nuevo registro), con lo cual este registro no será grabado en el fichero nuevo.

Si la clave del registro leído no es 3 pero sí es 4, la instrucción

```
IF CLAVE = 4 THEN GO TO CAMBIO-CLAVE
```

bifurca hacia el párrafo CAMBIO-CLAVE.



Si la clave no es ni 3 ni 4, el ordenador sigue en secuencia con lo que ejecuta la instrucción

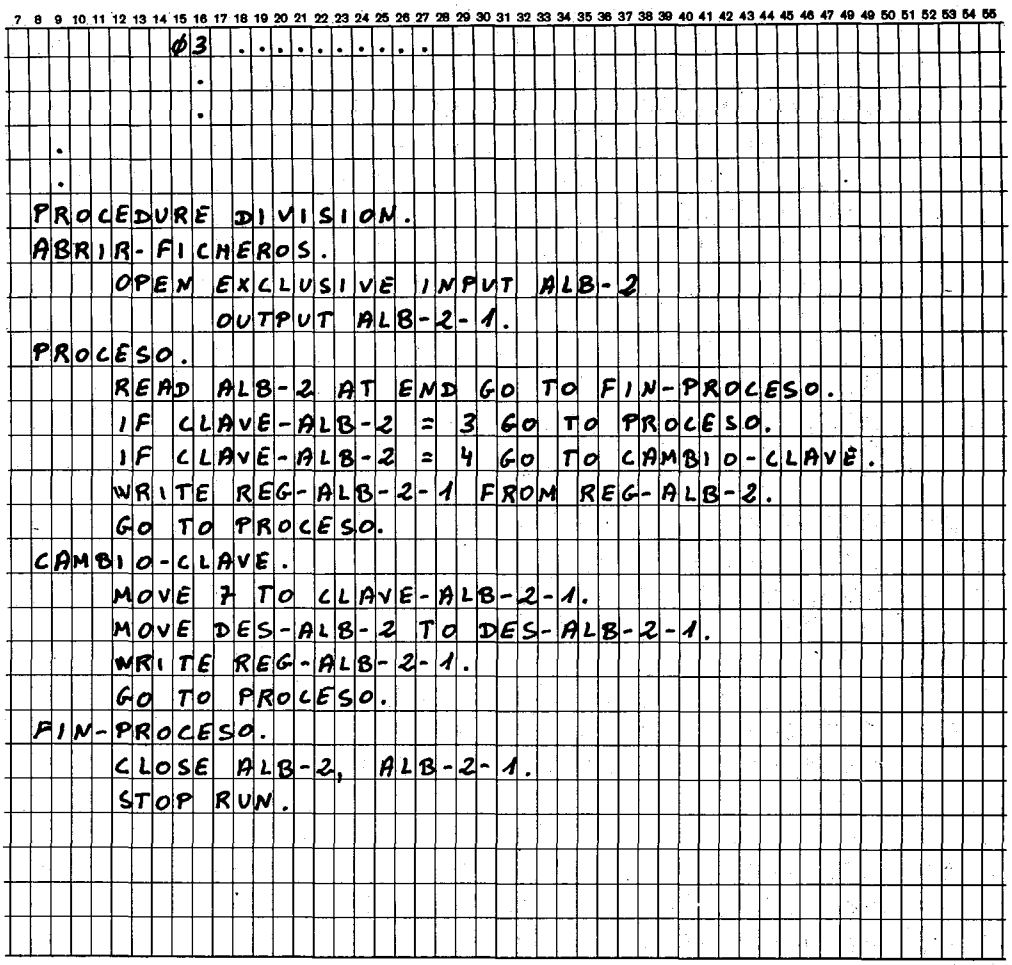
WRITE REG-ALB-2-1 FROM REG-ALB-2

y graba en el fichero ALB-2-1 el registro leído en ALB-2.

La instrucción

GO TO PROCESO

del párrafo PROCESO, bifurca hacia la instrucción de lectura para obtener otro registro del fichero ALB-2 y tratarlo según lo ya indicado.



```
      03 .....
       .
       .
  .
  .
PROCEDURE DIVISION.
ABRIR-FICHEROS.
        OPEN EXCLUSIVE INPUT ALB-2
                    OUTPUT ALB-2-1.
PROCESO.
        READ ALB-2 AT END GO TO FIN-PROCESO.
        IF CLAVE-ALB-2 = 3 GO TO PROCESO.
        IF CLAVE-ALB-2 = 4 GO TO CAMBIO-CLAVE.
        WRITE REG-ALB-2-1 FROM REG-ALB-2.
        GO TO PROCESO.
CAMBIO-CLAVE.
        MOVE 7 TO CLAVE-ALB-2-1.
        MOVE DES-ALB-2 TO DES-ALB-2-1.
        WRITE REG-ALB-2-1.
        GO TO PROCESO.
FIN-PROCESO.
        CLOSE ALB-2, ALB-2-1.
        STOP RUN.
```

Si el registro leído tiene la clave 4, el ordenador bifurca al párrafo CAMBIO-CLAVE donde la instrucción

MOVE 7 TO CLAVE-ALB-2-1

pone el número 7 en la clave del registro a grabar.

La instrucción

MOVE DES-ALB-2 TO DES-ALB-2-1

transfiere los demás campos del registro leído hacia el buffer del registro a grabar.

La instrucción

WRITE REG-ALB-2-1

graba en el fichero nuevo (ALB-2-1) el contenido de REG-ALB-2-1, quedando la clave cambiada de 4 a 7 y los demás campos intactos respecto al registro leído.

La instrucción

GO TO PROCESO

del párrafo CAMBIO-CLAVE bifurca hacia la instrucción de lectura una vez el registro ha sido grabado con las modificaciones señaladas.

Cuando han sido tratados todos los registros del fichero ALB-2, el ordenador bifurca al párrafo FIN-PROCESO donde desactiva los dos ficheros mediante la instrucción

CLOSE ALB-2, ALB-1

De esta manera se obtiene un nuevo fichero, ALB-2-1, que contiene las modificaciones realizadas respecto al fichero ALB-2.

Ampliación de un fichero secuencial

Cuando se deseen añadir más registros a un fichero secuencial, debe activarse con la instrucción

OPEN EXTEND nombre-fichero

Con la cláusula EXTEND, el fichero es activado para escritura a partir del último registro.

Ejemplo:

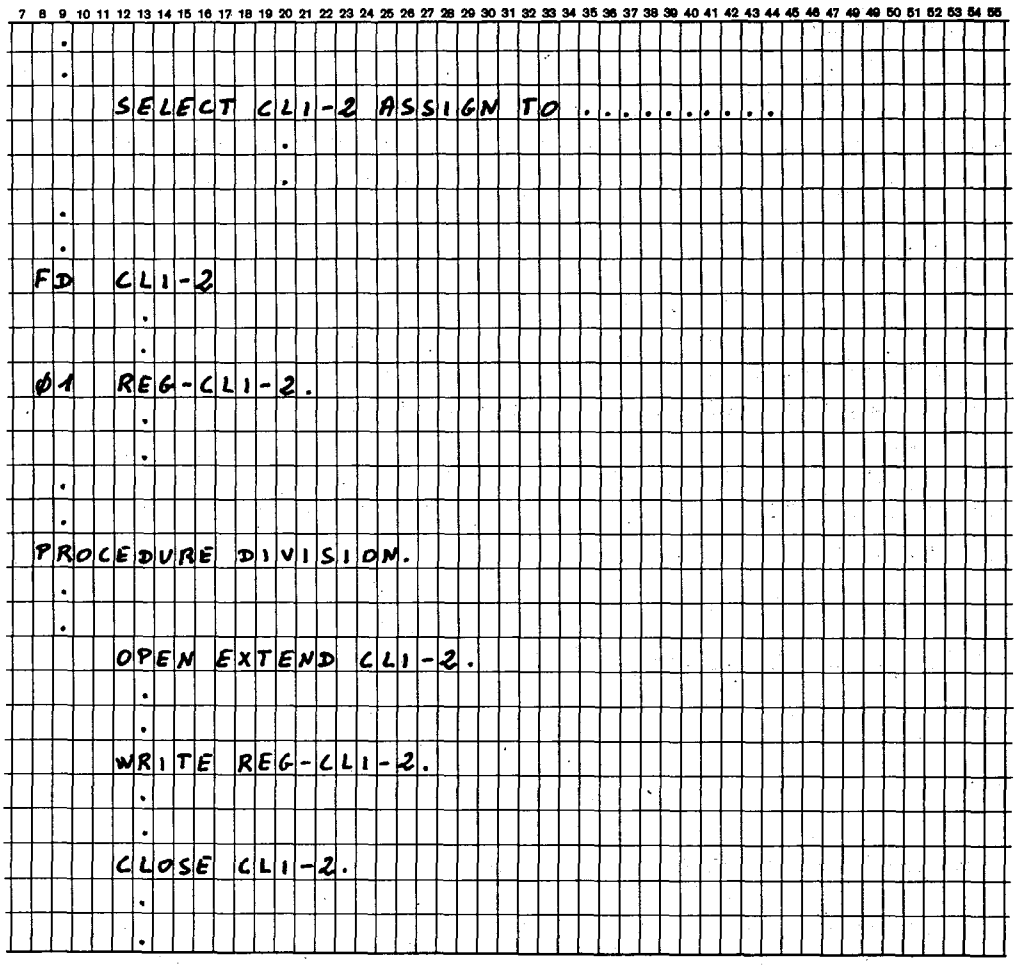
Se deben añadir más registros al fichero secuencial CLI-2. En la codificación siguiente se ejecuta la instrucción

OPEN EXTEND CLI-2

con lo que el fichero es preparado para su ampliación y con la instrucción

WRITE REG-CLI-2

se graban los nuevos registros.



Una vez finalizada la ampliación del fichero, es desactivado con la instrucción

CLOSE CLI-2.

Desbloqueo

Para desbloquear los registros bloqueados de un fichero en acceso secuencial, se utiliza la instrucción

UNLOCK fichero

Si los registros fueron bloqueados en su lectura (READ ... LOCK ...) todos los registros que quedaron bloqueados son desbloqueados.

Si fue bloqueado el fichero (OPEN EXCLUSIVE...) todos los registros del fichero quedan desbloqueados.

El modelo siguiente es un ejemplo del uso de esta instrucción.

Una vez los registros han sido desbloqueados, ya pueden ser tratados por otro programa al mismo tiempos que son tratados por el programa que los bloqueó en un principio.

```
PROCEDURE DIVISION.
.
.
    OPEN EXCLUSIVE INPUT FICHERO-1.
    OPEN INPUT FICHERO-2.
    .
    .
    READ FICHERO-2 LOCK ..........
    .
    .
    UNLOCK FICHERO-1.
    .
    .
    UNLOCK FICHERO-2.
    .
```

3.3.2. Acceso directo

Para tratar en acceso directo los registros de un fichero, deben cumplirse las siguientes condiciones:

— El fichero debe estar organizado aleatoriamente.

Esto se indica con la cláusula

ORGANIZATION IS RELATIVE

— El fichero no debe estar bloqueado.

— La clave de búsqueda de los registros (KEY) debe tener un nombre. Dicho nombre se asigna con la cláusula

RELATIVE KEY IS nombre-de-clave

— La descripción de la KEY debe estar situada como primer campo dentro de la definición de los campos de cada registro.

— La KEY debe poseer formato numérico.

— El soporte físico de información debe ser o disco magnético o tambor magnético.

— El acceso se indica con la cláusula ACCESS MODE IS RANDOM.

Ejemplo:

La codificación correcta de la modalidad de acceso directo al fichero CREDITOS es el siguiente, donde la clave (CLAVE-CRE) es el primer campo de la definición de los campos del registro.

En este caso, el número máximo de registros que tendrá este fichero es de noventa y nueve ya que con la cláusula PIC 99 éste es el número más grande que puede asignarse al campo CLAVE-CRE.

La clave (KEY) de un fichero debe almacenar siempre cualquier valor entero positivo comprendido entre 1 y el n.° máximo de registros del fichero.

```
SELECT CREDITOS ASSIGN TO DISK
       ACCESS MODE IS RANDOM
       ORGANIZATION IS RELATIVE
       RELATIVE KEY IS CLAVE-CRE.
.
.
FD  CREDITOS
    DATA RECLORD IS REG-CRE
    .
    .
01  REG-CRE.
    02  CLAVE-CRE PIC 99.
    02  DES-REG-CRE.
        03  ..........
        .
```

Apertura de un fichero directo

Cuando un fichero es tratado en acceso directo, está activo tanto para lectura como para grabación.

Para activar un fichero en acceso directo se utiliza la instrucción

OPEN [EXCLUSIVE] I-0 fichero-1, fichero-2,...

Ejemplo-1:

El modelo siguiente corresponde a la codificación de la apertura de los ficheros FICH-1 y FICH-2 en acceso directo.

```
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 49 50 51 52 53 54 55
      .
      .
          SELECT  FICH-1  . . . . . .
                  ACCESS  IS  RELATIVE
                  .
                  .
          SELECT  FICH-2  . . . . . .
                  ACCESS  IS  RELATIVE
                  .
                  .
      .
      .
  PROCEDURE DIVISION.
      .
      .
          OPEN I-O FICH-1,  FICH-2.
              .
```

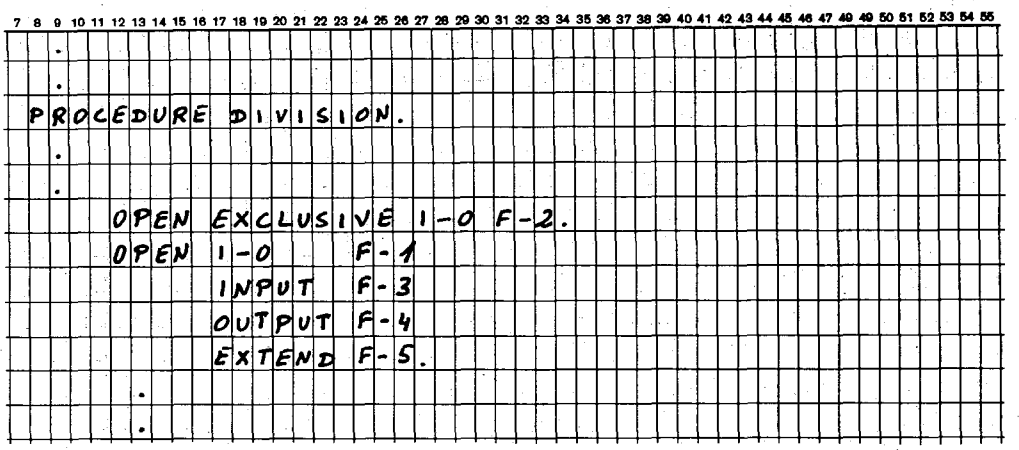
Ejemplo-2:

En la página derecha muestra la codificación de los ficheros F-1 (en acceso directo), F-2 (en acceso directo y uso exclusivo), F-3 (en acceso secuencial para lectura), F-4 (en acceso secuencial para grabación) y F-5 (en acceso secuencial para ampliación).

```
           SELECT F-1 ..........
                  ACCESS IS RELATIVE
                  .
                  .
           SELECT F-2 ..........
                  ACCESS IS RELATIVE
                  .
                  .
           SELECT F-3 ..........
                  ACCESS IS SEQUENTIAL
                  .
                  .
           SELECT F-4 ..........
                  ACCESS IS SEQUENTIAL
                  .
                  .
           SELECT F-5 ..........
                  ACCESS IS SEQUENTIAL
                  .
                  .
.
.
PROCEDURE DIVISION.
.
.
```

```
           OPEN I-O F-1.
           OPEN EXCLUSIVE I-O F-2.
           OPEN INPUT F-3.
           OPEN OUTPUT F-4.
           OPEN EXTEND F-5.
             .
             .
.
.
.
.
```

La anterior codificación es equivalente a esta:



Cuando un fichero es activado en modo directo, el ordenador ejecuta los mismos pasos que realiza cuando el fichero es activado en modo secuencial:

Lectura directa de registros

Para leer registros en acceso directo, y antes de la instrucción de lectura, ha de asignarse al campo definido como KEY, el número del registro que se desea leer.

La instrucción de lectura posee el formato

READ nombre-fiche [INTO] nombre-campo [LOCK] INVALID KEY instrucción

Ejemplo:

En el programa siguiente se define el fichero ART-3.

Dentro de la Procedure Division, la instrucción

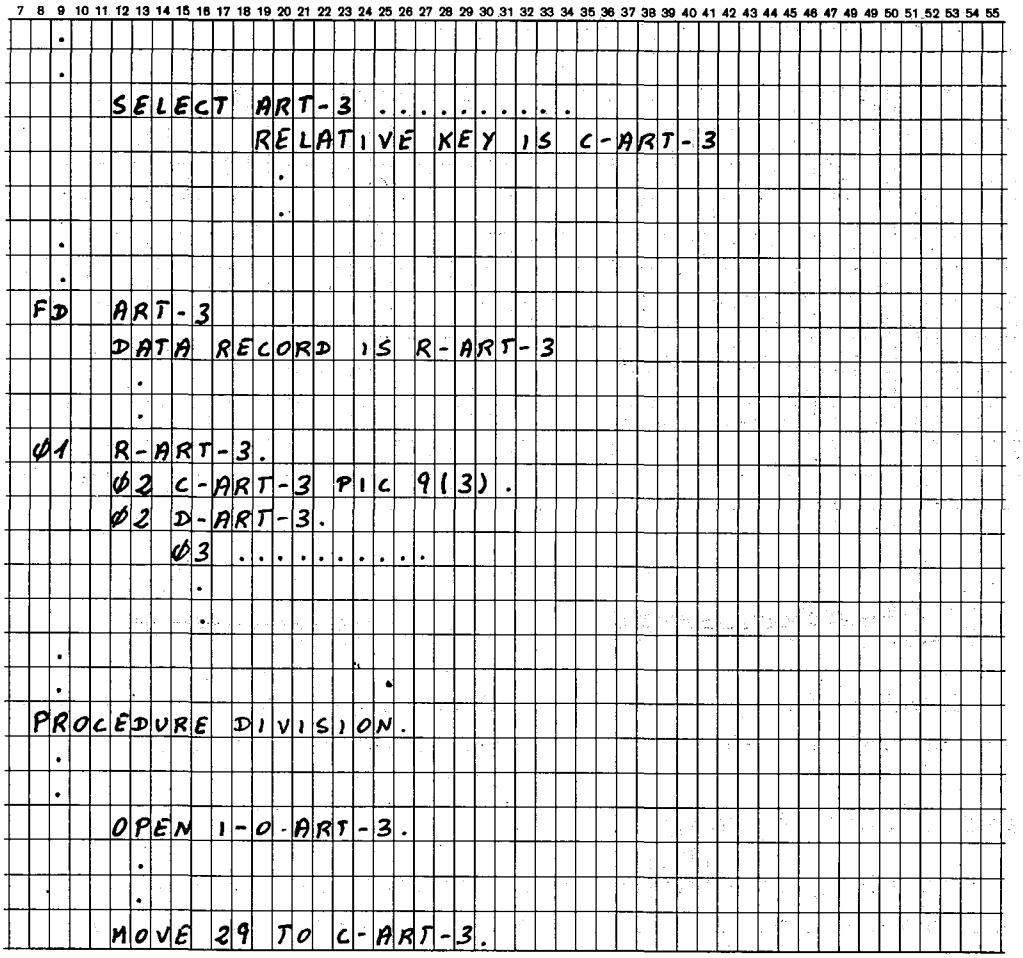
MOVE 29 TO C-ART-3

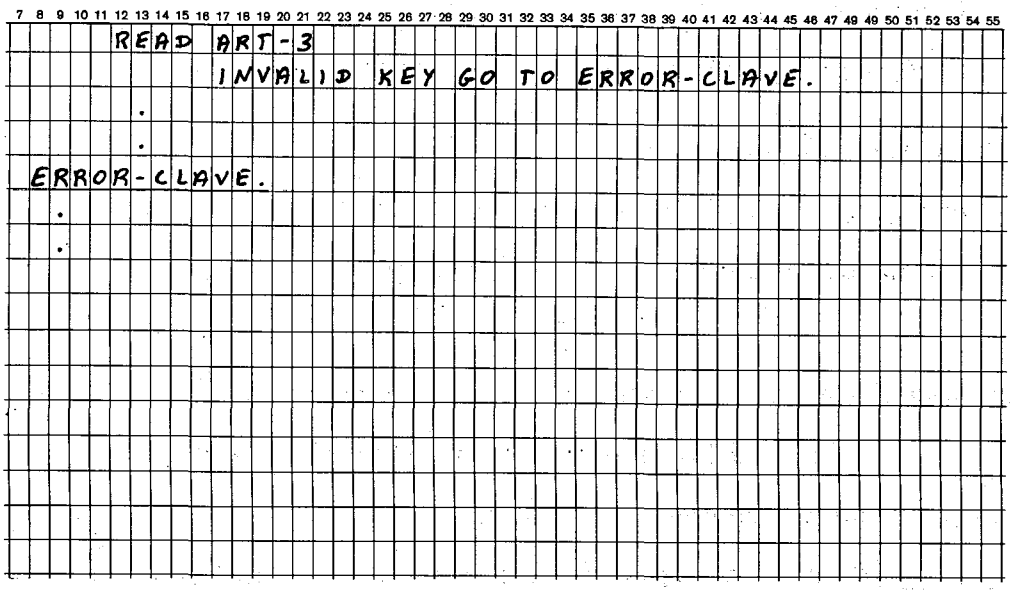
almacena el número veintinueve en la clave del fichero con lo que la instrucción

READ ART-3 INVALID KEY GO TO ERROR-CLAVE

leerá el registro veintinueve del fichero ART-3.

En caso de que dicho registro no exista, bifurcará al párrafo ERROR-CLAVE.


SELECT ART-3 .....
RELATIVE KEY IS C-ART-3
FD ART-3
DATA RECORD IS R-ART-3
01 R-ART-3.
02 C-ART-3 PIC 9(3).
02 D-ART-3.
03 .....
PROCEDURE DIVISION.
OPEN I-O ART-3.
MOVE 29 TO C-ART-3.


READ ART-3
INVALID KEY GO TO ERROR-CLAVE.
ERROR-CLAVE.

```
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 49 50 51 52 53 54 55
                .
                .
           SELECT CLIENTES .....
                  RELATIVE KEY IS C-1
                        .
                        .
                .
                .
       FD   CLIENTES
            DATA RECORD IS REG-CLIENTES
                .
                .
       01   REG-CLIENTES.
            02 C-1 PIC.999.
            02 DES-REG-CLIENTES.
               03 ..........
                     .
                     .
                .
                .
       WORKING-STORAGE SECTION.
                .
                .
       01   N-CLIENTE PIC 999.
                .
                .
       PROCEDURE DIVISION.
```

```
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 49 50 51 52 53 54 55
                .
                .
           MOVE N-CLIENTE TO C-1.
           WRITE REG-CLIENTES
                 INVALID KEY GO TO ERROR.
                .
                .
       ERROR.
                .
                .
```

Grabación directa de registros

Al igual que en la lectura en acceso directo primero hay que asignar un valor a la clave y a continuación grabar el registro con la instrucción

WRITE registro [FROM campo] INVALID KEY instrucción

Ejemplo:

En el programa de la página anterior se graban los registros cuya clave está almacenada en N-CLIENTE ya que antes de la instrucción de grabación, la instrucción

MOVE N-CLIENTE TO C-1

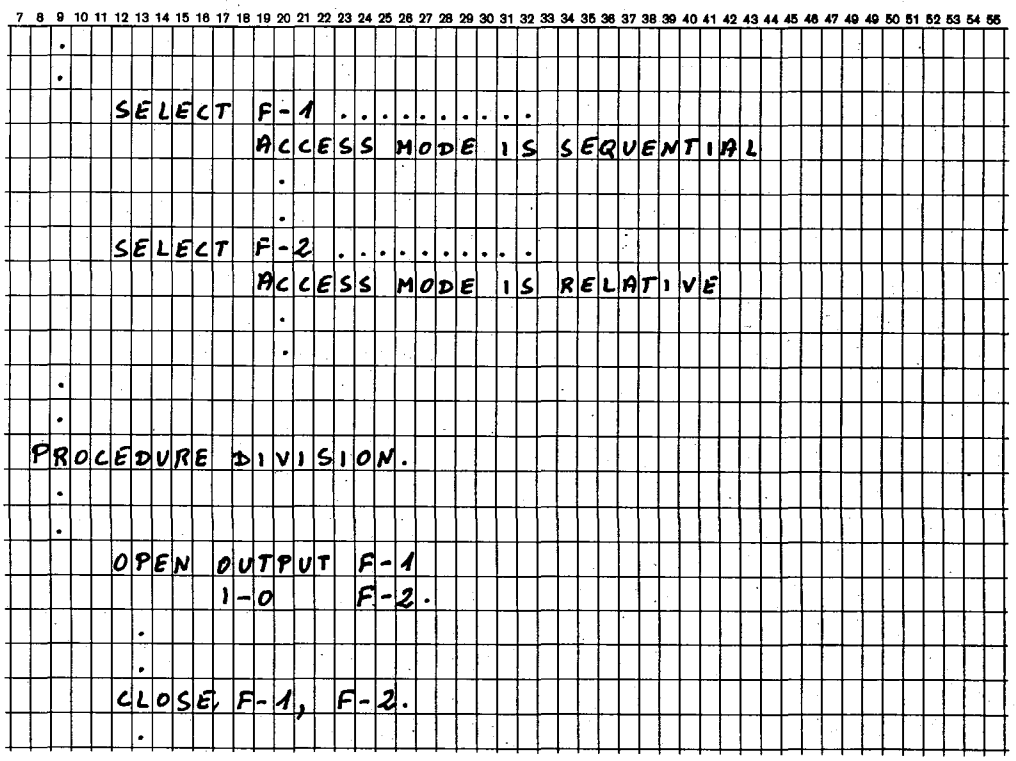
almacena en C-1 el contenido de N-CLIENTES.

Si la clave es incorrecta, la cláusula INVALID KEY GO TO ERROR bifurca al párrafo ERROR.

Cierre de un fichero directo

El procedimiento de desactivación de un fichero en acceso directo es el mismo que con los ficheros de acceso secuencial.

Ejemplo:



Modificación de un fichero directo

Para modificar registros en acceso directo, se utiliza la instrucción.

REWRITE registro [FROM campo] INVALID KEY instrucción.

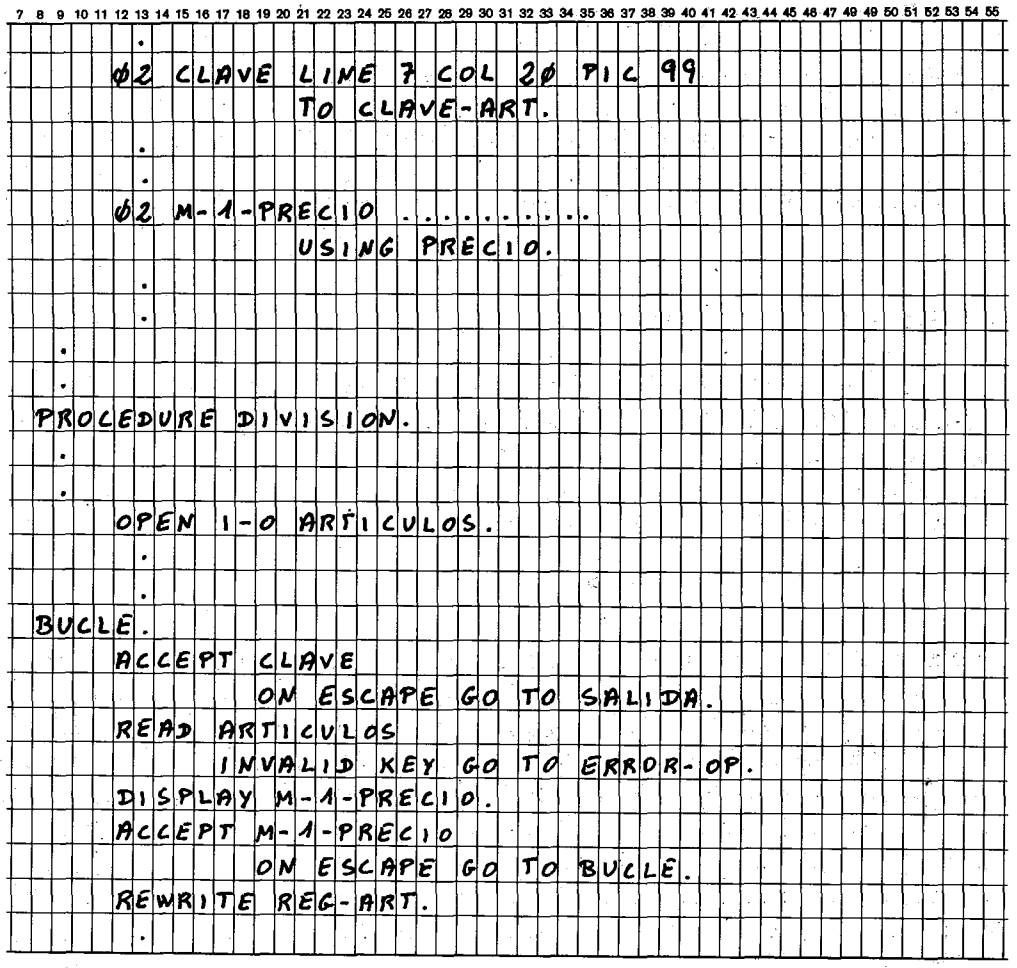
La palabra inglesa REWRITE significa *volver a grabar.*

Ejemplo:

Modificar el campo PRECIO del registro cuya clave indique el operador del fichero ARTICULOS. El operador debe ver el contenido de PRECIO antes de efectuar la modificación.

El nombre de la clave es CLAVE-ART.

```
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 49 50 51 52 53 54 55
           .
           .
                SELECT ARTICULOS ..........
                       RELATIVE KEY IS CLAVE-ART
                                .
                                .
           .
           .
       FD  ARTICULOS
           DATA RECORD IS REG-ART
             .
             .
       01  REG-ART.
           02 CLAVE-ART PIC 99.
           02 DES-ART.
              03 ..........
                 .
                 .
              03 PRECIO PIC 9(5).99.
                 .
                 .
           .
           .
       SCREEN SECTION.
       01  M-1.
           02 ..........
             .
```



```
           02 CLAVE LINE 7 COL 20 PIC 99
                        TO CLAVE-ART.
           02 M-1-PRECIO ..........
                        USING PRECIO.
PROCEDURE DIVISION.
           OPEN I-O ARTICULOS.
BUCLE.
           ACCEPT CLAVE
                        ON ESCAPE GO TO SALIDA.
           READ ARTICULOS
                        INVALID KEY GO TO ERROR-OP.
           DISPLAY M-1-PRECIO.
           ACCEPT M-1-PRECIO
                        ON ESCAPE GO TO BUCLE.
           REWRITE REG-ART.
           GO TO BUCLE.
ERROR-OP.
SALIDA.
```

En la Screen Section, y dentro de la máscara M-1, se define el campo CLAVE asignado a la variable CLAVE-ART y el campo M-1-PRECIO asignado a la variable PRECIO.

En la Procedure Division y después de la apertura del fichero ARTICULOS, la instrucción

```
ACCEPT CLAVE ON ESCAPE GO TO SALIDA
```

pide al operador la clave del registro que desea modificar, y almacena el dato numérico introducido en CLAVE-ART que es el campo que se utiliza como clave.

Si el operador desea finalizar las modificaciones de registros, pulsará la tecla ESCAPE con lo que el ordenador bifurcará hacia el párrafo SALIDA.

La siguiente instrucción

```
READ ARTICULOS
INVALID KEY GO TO ERROR-OP
```

lee el registro que indica el valor almacenado en CLAVE-ART. Si la clave introducida por el operador es incorrecta, el ordenador bifurca al párrafo ERROR-OP.

Una vez leído el registro, la instrucción

```
DISPLAY M-1-PRECIO
```

displaya el valor almacenado en PRECIO. Y la instrucción

```
ACCEPT M-1-PRECIO ON ESCAPE GO TO BUCLE
```

almacena en PRECIO el nuevo valor introducido por el operador. Pero si el operador pulsa la tecla ESCAPE, esta instrucción bifurcará hacia el principio del párrafo BUCLE con lo que el ordenador pedirá al operador la clave de otro registro, dejando sin modificar el último registro leído.

Una vez el operador ha modificado el contenido de PRECIO, la instrucción

```
REWRITE REG-ART
```

graba el registro con las modificaciones efectuadas.

Desbloqueo

La técnica a seguir para desbloquear los registros de un fichero en acceso directo, es la misma que la empleada en el acceso secuencial (apartado 3.2.7).

3.3.3. Acceso indexado

Para acceder en forma indexada a los registros de un fichero, debe utilizarse la cláusula

ACCESS IS DYNAMIC

El fichero debe estar organizado en forma indexada. Esto se indica con la cláusula

ORGANIZATION IS INDEXED

El nombre de la KEY se especifica con la cláusula

RECORD KEY IS nombre-campo

La descripción del campo utilizado como KEY debe poseer formato alfanumérico y debe estar situada como primer campo dentro de la definición de los campos del registro.

El soporte físico de información debe ser o disco magnético o tambor magnético.

Ejemplo:

Esta es la codificación correspondiente al fichero TABLA en acceso indexado.

```
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
                .
                .
                    SELECT TABLA ASSIGN TO DISK
                           ACCESS IS DYNAMIC
                           ORGANIZATION IS INDEXED
                           RECORD KEY IS TABLA-KEY.
                .
                .
           FD   TABLA
                DATA RECORD IS TABLA-REG
                  .
                  .
           01   TABLA-REG.
                02 TABLA-KEY PIC X(5).
                02 ..........
                  .
```

El tratamiento y las instrucciones de un fichero en acceso indexado son las mismas que en acceso directo.

La siguiente codificación corresponde al tratamiento del fichero definido en este ejemplo.

```
OPEN I-O TABLA.
.
.
MOVE "32B<%" TO TABLA-KEY.
.
.
READ TABLA
     INVALID KEY GO TO ERROR-1.
.
.
WRITE TABLA-REG
      INVALID KEY GO TO ERROR-2.
.
.
REWRITE TABLA-REG.
.
.
CLOSE TABLA.
.
.
```

3.4. ELIMINACION DE REGISTROS

Para destruir o eliminar registros de un fichero, se utiliza la instrucción

DELETE fichero RECORD [INVALID KEY instrucción]

Las normas a seguir son:

— El fichero no debe estar organizado de manera secuencial.

— Esta instrucción puede utilizarse con cualquier tipo de acceso, pero si el acceso es secuencial, no se codificará la cláusula INVALID KEY instrucción.

— Si el acceso es secuencial, será destruido el último registro leído.

— Si el acceso es directo o indexado, se destruirá el registro correspondiente a la clave almacenada en el campo definido como KEY del fichero.

— El registro eliminado puede ser reemplazado por otro registro.

Ejemplo:

En la figura 3.35 está el método de eliminación de registros de acuerdo al tipo de organización y acceso con que se definen los ficheros F-1, F-2, F-3, F-4 y F-5.

```
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
                                                                                                                                                  
                                                                                                                                                  
                                                                                                                                                  
```

```
OPEN INPUT F-1
           F-4
       I-O F-2
           F-3
           F-5.
MOVE "ABC" TO KEY-F-3.
MOVE 1 TO KEY-F-1
          KEY-F-4.
MOVE 17 TO KEY-F-2
          KEY-F-5.
START F-1 KEY = KEY-F-1.
START F-4 KEY = KEY-F-4.
DELETE F-2 RECORD
       INVALID KEY GO TO ERR.
DELETE F-5 RECORD
       INVALID KEY GO TO ERR.
DELETE F-3 RECORD
       INVALID KEY GO TO ERR.
READ F-1.
DELETE F-1.
READ F-4.
DELETE F-4.
  .
  .
ERR.
  .
  .
```

3.5. TRATAMIENTO DE ERRORES

Para poder tratar cualquier error que surja a lo largo del tratamiento de los registros de un fichero, se han de seguir estas normas:

— Para cada fichero al que se quiera tratar sus errores, debe destinarse un campo de la memoria en el que el ordenador almacenará el código del posible error.

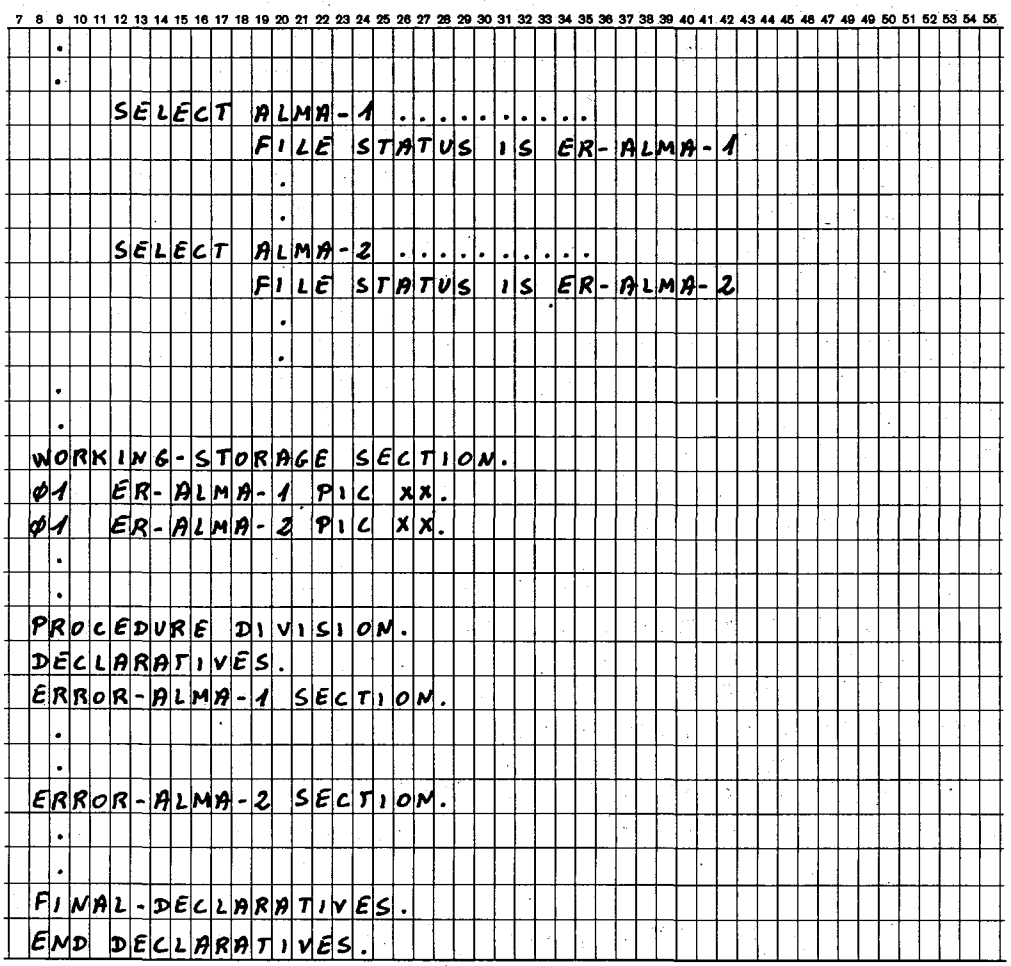
— La Procedure Division se dividirá en secciones. Cada sección contendrá las instrucciones que tratan el error de un fichero. Por tanto, en un programa con dos ficheros la Procedure Division estará dividida en tres secciones, dos de ellas con las instrucciones y la tercera, que tendrá las instrucciones del programa.

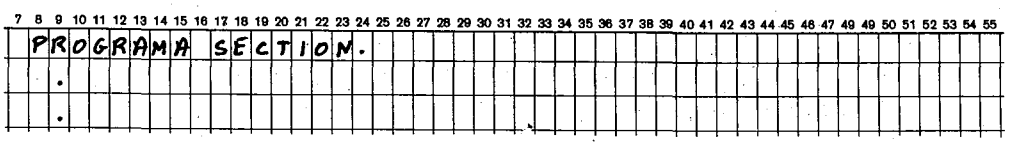
— Las secciones destinadas al tratamiento de errores, deben ser las primeras secciones de la Procedure Division.

— La sección con las instrucciones que resuelven el problema al que se destina el programa, debe estar a continuación de la última sección destinada al tratamiento de errores.

— El conjunto de secciones destinadas a tratar los errores en los ficheros, irá encabezado por la cláusula

DECLARATIVES





y al final de dicho conjunto, la cláusula

END DECLARATIVES

Esta cláusula debe ir encabezada por un nombre de párrafo.

En el modelo presentado en la página anterior se puede ver la estructura de todo lo expuesto.

La primera instrucción dentro de cada sección destinada al tratamiento de errores debe ser

USE AFTER ERROR PROCEDURE ON nombre-fichero

Esta instrucción indica que la sección en la que está codificada, tratará los posibles errores del fichero especificado en nombre-fichero.

Ejemplo:

```
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
      .
      .
PROCEDURE DIVISION.
DECLARATIVES.
ERROR-ALMA-1 SECTION.
          USE AFTER ERROR PROCEDURE ON ALMA-1.
UNO-ALMA-1.
                .
                .
DOS-ALMA-1.
                .
                .
ERROR-ALMA-2 SECTION.
          USE AFTER ERROR PROCEDURE ON ALMA-2.
UNO-ALMA-2.
                .
                .
DOS-ALMA-2.
                .
                .
FIN-ERRORES.
END DECLARATIVES.
PROGRAMA SECTION.
UNO-PROG.
                .
                .
DOS-PROG.
```

En este programa se indica que la sección ERROR-ALMA-1 tratará los errores del fichero ALMA-1 y la sección ERROR-ALMA-2 los errores del fichero ALMA-2 correspondientes al ejemplo del apartado 3.4.

Dentro de cada sección, el primer párrafo debe averiguar qué error ha ocurrido, consultando el código almacenado en el campo destinado a almacenar el código de los errores del fichero, mediante instrucciones IF.

```
.
.
PROCEDURE DIVISION.
DECLARATIVES.
ERROR-ALMA-1 SECTION.
        USE AFTER ERROR PROCEDURE ON ALMA-1.
UNO-ALMA-1.
        IF ER-ALMA-1 = "34" PERFORM DOS-ALMA-1.
        IF ER-ALMA-1 = "91" GO TO TRES-ALMA-1.
        IF ER-ALMA-1 = "30" STOP RUN.
        IF ER-ALMA-1 = "99" DISPLAY MASC-IMP-OCUP.
          .
          .
DOS-ALMA-1.
          .
          .
TRES-ALMA-1.
          .
          .
```

Antes de codificar estas instrucciones, debe consultarse en el manual del fabricante qué códigos están asociados a cada tipo de error. A título de ejemplo se utilizarán estos:

Código	*Error*
10	Fin de fichero
21	KEY inválida
22	El registro ya existe
23	El registro no existe
24	OVERFLOW en tabla de KEYs

Código	*Error*
30	Error de hardware
34	Disco completo
91	Error en la apertura
92	El tratamiento no concuerda con la apertura del fichero
94	Fichero bloqueado
96	El fichero no está activo
97	Registro bloqueado
98	El fichero no puede ser creado
99	Impresora ocupada

Si cualquier instrucción dentro del conjunto de secciones destinadas a tratar los errores de los ficheros, bifurca hacia la cláusula END DECLARATIVES, el control del procesador regresa a la siguiente instrucción a la que generó el error.

```
ERROR-ALMA-1 SECTION.
    USE AFTER ERROR PROCEDURE ON ALMA-1.
UNO-ALMA-1.
    IF ER-ALMA-1 = "97" GO TO DOS-ALMA-1.
    .
    .
DOS-ALMA-1.
    .
    .
    GO TO FIN-ERRORES.
TRES-ALMA-1.
    .
    .
FIN-ERRORES.
END DECLARATIVES.
PROGRAMA SECTION.
    .
    .
    READ ALMA-1.
    MOVE ..........
    .
```

En el programa anterior se supone que ha ocurrido un error con el código 97 en la instrucción

```
READ ALMA-1
```

El control del procesador bifurca hacia la sección ERROR-ALMA-1 donde la instrucción

```
IF ER-ALMA-1 = «97» GO TO DOS-ALMA-1
```

bifurca hacia el párrafo UNO-ALMA-1.

Una vez ejecutadas las instrucciones de este párrafo, la instrucción

```
GO TO FIN-ERRORES
```

transfiere el control del procesador a la cláusula END DECLARATIVES con lo que a continuación se ejecutará la instrucción

```
MOVE .....
```

Finalmente, las instrucciones de entrada y de salida de los ficheros que tengan una sección para el tratamiento de errores, no pueden llevar las cláusulas

```
INVALID KEY instrucción
AT END instrucción
```

ya que los pasos a seguir estarán definidos en la sección correspondiente.

EJERCICIOS

1. Para ampliar un fichero secuencial, la apertura del mismo debe ser en modo:

   a) No puede ser ampliado
   b) OUTPUT
   c) EXTEND

2. Para modificar un fichero secuencial

   a) Se utiliza la instrucción REWRITE
   b) Se utiliza la instrucción WRITE
   c) Se ha de crear otro fichero

3. En el acceso indexado el ordenador accede a los registros

   a) Por medio de una tabla de claves
   b) Por su posición física
   c) Al igual que en acceso directo

4. ¿Cuáles son las maneras en las que se puede organizar los registros de un fichero?

5. En acceso secuencial sobre ficheros no organizados secuencialmente, antes de ejecutarse la primera instrucción de lectura

   a) Este tratamiento no se puede realizar
   b) Debe inicializarse el campo KEY
   c) No es necesario hacer nada

6. Las secciones que tratan los errores de los ficheros van delimitadas por las cláusulas .....

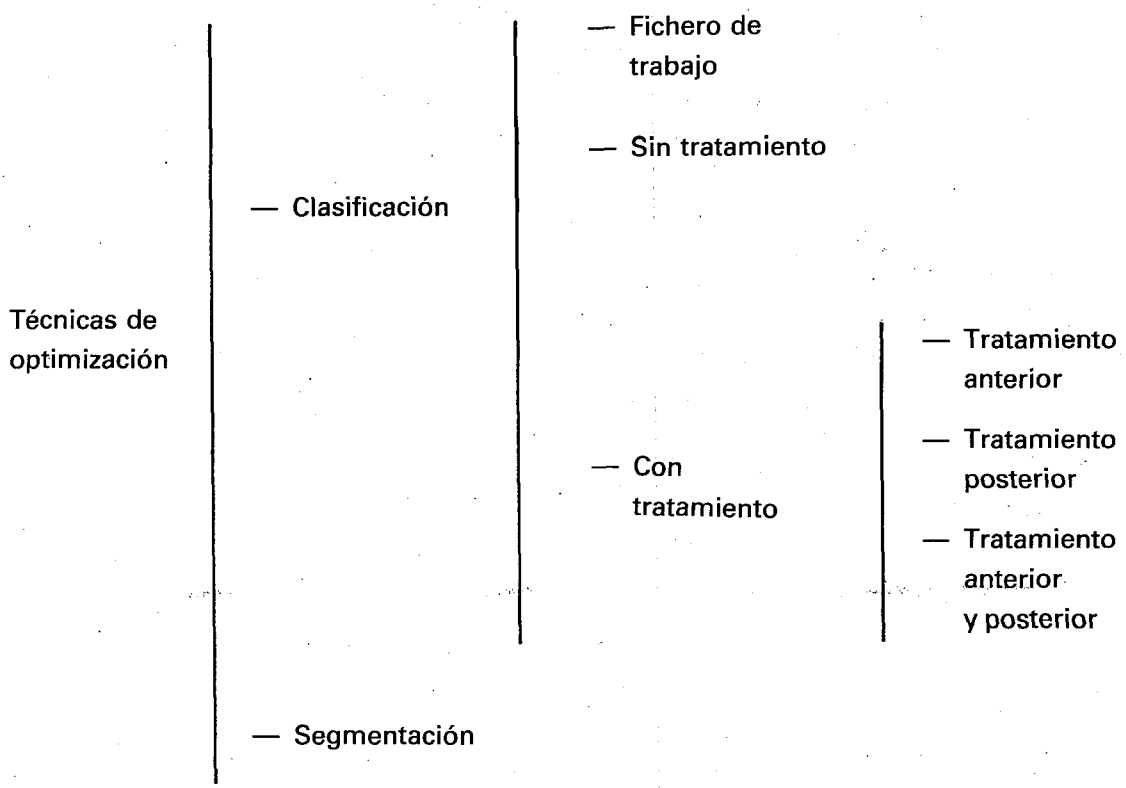
7. ¿Qué es bloquear un fichero? .....

8. ¿Qué diferencias hay entre el cierre de un fichero directo, un fichero secuencial y un fichero indexado?

4.

TECNICAS DE OPTIMIZACION

CUADRO RESUMEN DE LA LECCION



4.1. CLASIFICACION

Los procedimientos de clasificación se realizan sobre los registros de un fichero.

Para clasificar los registros de un fichero son necesarios tres ficheros:

— Un fichero de entrada
— Un fichero de trabajo
— Un fichero de salida

El fichero de entrada tiene almacenados los registros a clasificar.

El fichero de trabajo es un fichero que necesita el ordenador para efectuar la clasificación.

El fichero de salida es el fichero donde estarán almacenados los registros ya clasificados.

4.1.1. Fichero de trabajo

El fichero de trabajo debe utilizar como soporte físico o un disco magnético o un tambor magnético.

En su definición (dentro de la Environment Division), sólo debe utilizarse la cláusula SELECT.

En su descripción (dentro de la File Section), se utilizará SD en lugar de FD. Tampoco se pueden usar cláusulas excepto la DATA RECORD...

En la descripción de los campos hay que tener en cuenta que:

— Sólo es necesario describir los campos que se utilizarán para la clasificación de cada registro.

— La longitud total del registro debe ser la misma que la de los registros a clasificar.

Ejemplo:

Se desea clasificar el fichero ALUMNOS por el campo NOMBRE (ver codificación en la página siguiente).

El fichero de salida, ALUM-CLA, es el fichero que tendrá almacenados, en forma secuencial, los registros de ALUMNOS clasificados según el nombre.

El fichero de trabajo, FICH-TRA, está definido de acuerdo a lo explicado, con el campo NOMBRE-FICH-TRA, concordando en posición y longitud con los campos NOMBRE y NOMBRE-CLA para poder clasificar por el nombre los registros del fichero ALUMNOS.

4.1.2. Clasificación sin tratamiento

Si no se ha de realizar ningún tratamiento al fichero de entrada antes de efectuar la clasificación y tampoco se han de tratar los registros del fichero de salida a medida que van siendo clasificados, se utilizará la macro-instrucción

```
SORT fichero-trabajo { ASCENDING  } campo-1, campo-2
                     { DESCENDING }
...[ { DESCENDING } campo-3, campo-4, ... ] USING fichero-entrada
   { ASCENDING  }
GIVING fichero-salida
```

Con esta macro-instrucción, el contenido de los registros del fichero de salida será exactamente el mismo que el de los registros del fichero de entrada, pero clasificados.

Los campos por los cuales se desee clasificar los registros, deben estar descritos en la descripción del registro del fichero de trabajo. Por tanto, campo-1, campo-2, ... son campos del registro del fichero de clasificación.

```
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 49 50 51 52 53 54 55
      .
      .
          SELECT ALUMNOS ..........
                 .
                 .
          SELECT FICH-TRA ASSIGN TO DISK.
          SELECT ALUM-CLA ..........
                 ACCESS IS SEQUENTIAL
                 ORGANIZATION IS SEQUENTIAL
                 .
                 .
                 .
DATA DIVISION.
FILE SECTION.
FD   ALUMNOS
     DATA RECORD IS REG-ALUM
          .
          .
01   REG-ALUM.
     02 N-MATRICULA PIC 9(4).
     02 NOMBRE      PIC X(30).
     02 GRUPO       PIC 99.
     02 DIRECCION   PIC X(25).
     02 POBLACION   PIC X(15).
     02 PROVINCIA   PIC X(15).
     02 TELEFONO    PIC 9(7).
FD   ALUM-CLA
     DATA RECORD IS REG-ALUM-CLA
```

```
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 49 50 51 52 53 54 55
          .
          .
01   REG-ALUM-CLA.
     02 N-MATRICULA-CLA PIC 9(4).
     02 NOMBRE-CLA      PIC X(30).
     02 GRUPO-CLA       PIC 99.
     02 DIRECCION-CLA   PIC X(25).
     02 POBLACION-CLA   PIC X(15).
     02 PROVINCIA-CLA   PIC X(15).
     02 TELEFONO-CLA    PIC 9(7).
SD   FICH-TRA
     DATA RECORD IS REG-FICH-TRA.
01   REG-FICH-TRA.
     02 FILLER          PIC 9(4).
     02 NOMBRE-FICH-TRA PIC X(30).
     02 FILLER          PIC X(64).
WORKING-STORAGE SECTION.
     .
```

La cláusula ASCENDING indica que la clasificación es en orden ascendente (de menor a mayor) según campo-1, campo-2, ... y la cláusula DESCENDING en orden descendente (de mayor a menor) según campo-3, campo-4, ...

En el instante de ejecutarse la macroinstrucción SORT, los ficheros deben de estar desactivados, ya que esta macroinstrucción incluye todas las instrucciones necesarias para activar los ficheros, realizar la clasificación y desactivar los ficheros. Por tanto, una vez ejecutada esta macroinstrucción, los ficheros estarán desactivados.

Los pasos seguidos por la SORT son:

2 – activar para lectura el fichero de entrada
6 – activar para grabación el fichero de salida
1 – activar para lectura y grabación el fichero de trabajo
7 – copiar el fichero de trabajo en el fichero de salida
3 – copiar el fichero de entrada en el fichero de trabajo
5 – reorganizar el fichero de trabajo según el contenido de los campos especificados
8 – desactivar dos ficheros de salida
4 – desactivar el fichero de entrada
9 – desactivar el fichero de trabajo

Cuando la clasificación se realiza sin tratamiento, tanto el fichero de entrada como el fichero de salida deben estar organizados de manera secuencial y por tanto definidos con acceso secuencial.

Ejemplo-1:

Clasificar por el nombre en orden ascendente el mismo fichero ALUMNOS descrito en el párrafo anterior.

```
PROCEDURE DIVISION.
.
.
    SORT FICH-TRA
         ASCENDING NOMBRE-FICH-TRA
         USING ALUMNOS
         GIVING ALUM-CLA.
.
.
```

En la macro-instrucción SORT se indica que el fichero de trabajo es FICH-TRA, el fichero a clasificar es ALUMNOS y el fichero de salida es ALUM-CLA. La clasificación será en orden ascendente por el campo NOMBRE-FICH-TRA.

Cuando, en varios registros, el contenido del campo utilizado para su clasificación es el mismo, se indica otro campo adicional para clasificar, entre ellos, los registros mencionados.

Ejemplo-2

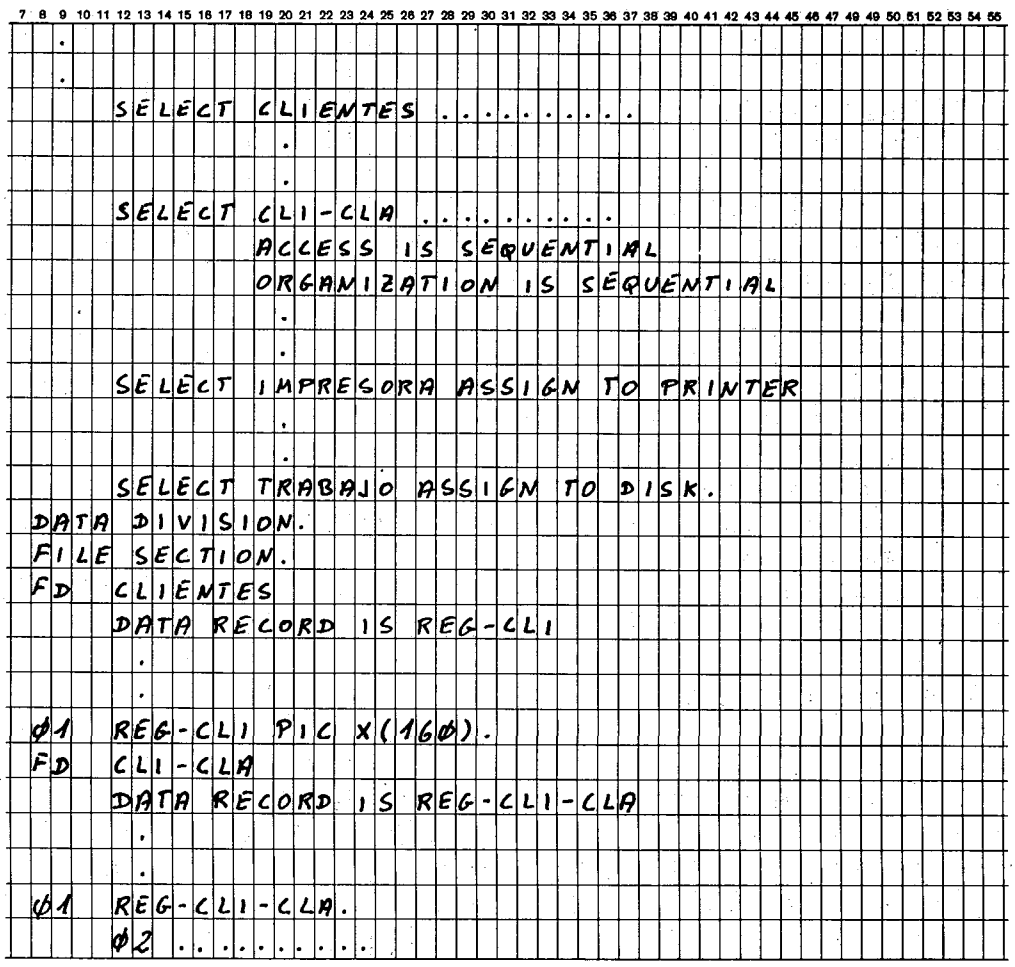
En este ejemplo se clasifica el mismo fichero ALUMNOS por nombres, en el caso de que haya dos (o más) alumnos con el mismo nombre, se clasifican a su vez por el grupo (de menor a mayor) y si resultara que hubiera dos o más

```
01  REG-ALUM.
    .
    .
01  REG-ALUM-CLA.
    .
    .
01  REG-FICH-TRA.
    02 N-MATRICULA-TRA  PIC 9(4).
    02 NOMBRE-TRA       PIC X(30).
    02 GRUPO-TRA        PIC 99.
    02 FILLER           PIC X(62).
    .
    .
PROCEDURE DIVISION.
    .
    .
    SORT FICH-TRA ASCENDING NOMBRE-TRA
                             GRUPO-TRA
                  DESCENDING N-MATRICULA-TRA
         USING ALUMNOS  GIVING ALUM-CLA.
    .
```

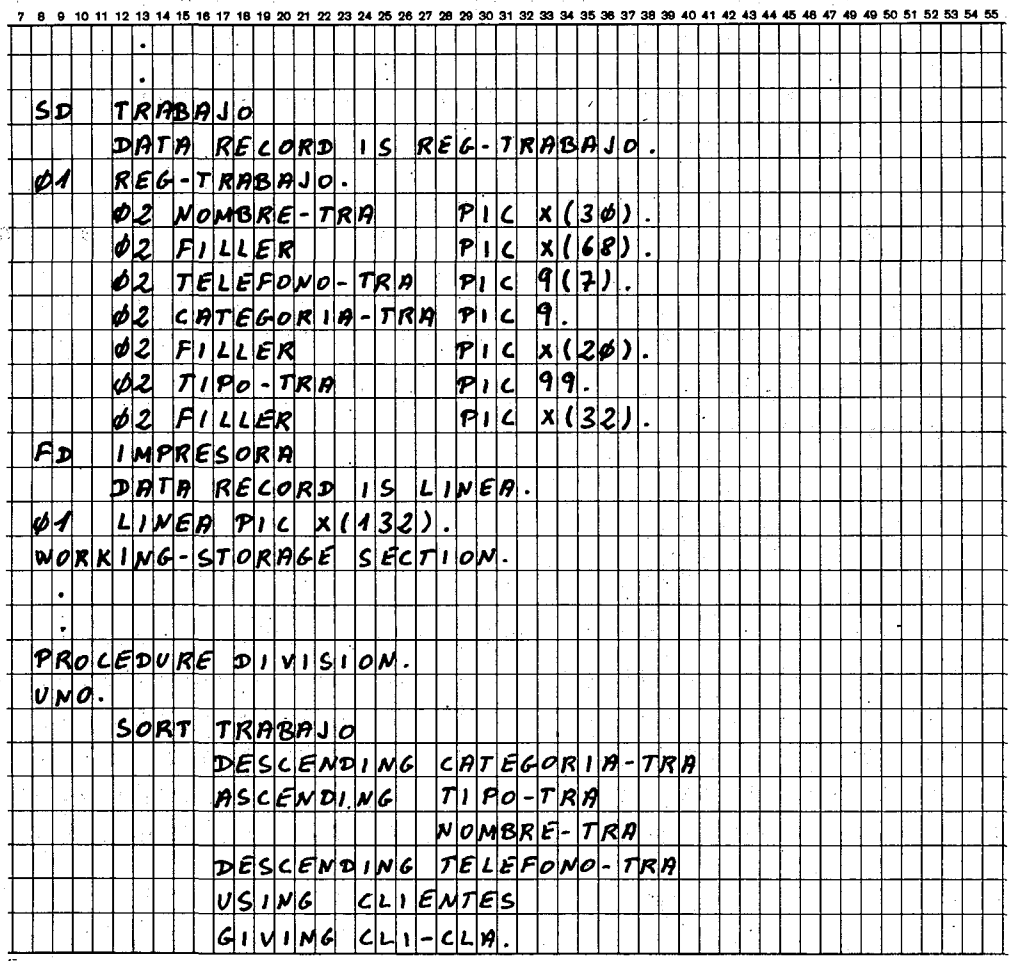
alumnos con el mismo nombre y que también estudiaran en el mismo grupo, son clasificados por el número de la matrícula de mayor a menor (en orden descendente).

Ejemplo-3

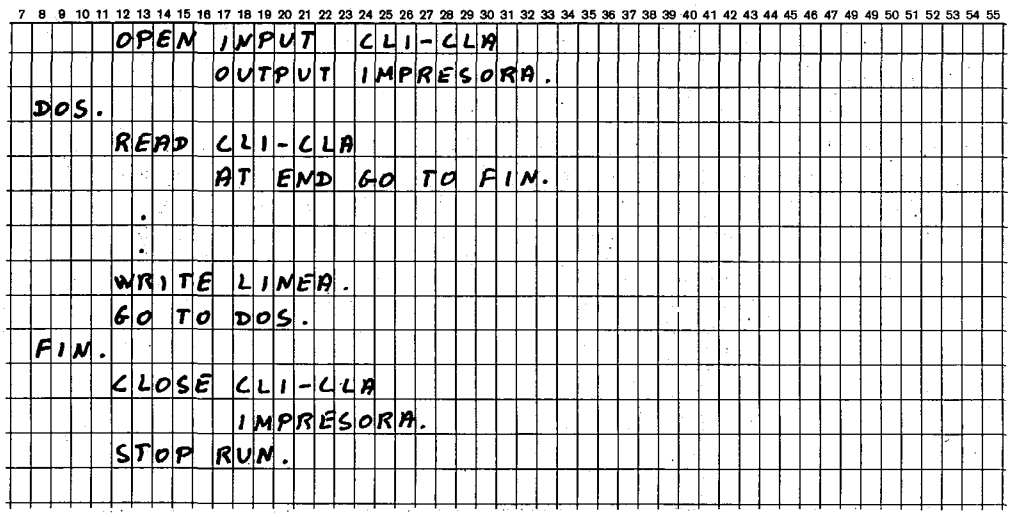
Dado un fichero de clientes, obtener su listado clasificándolos por categoría de mayor a menor y los que sean de la misma categoría clasificarlos a su vez por el tipo de establecimiento en orden ascendente. Si hay más de un cliente dentro de la misma categoría que sea también el mismo tipo de establecimiento, clasificarlos por nombre en orden ascendente. Y si también hubiera más de uno con estas características comunes, clasificarlos en orden descendente por el número de teléfono.



(Continuación del Ejemplo 3)



```
SD   TRABAJO
     DATA RECORD IS REG-TRABAJO.
01   REG-TRABAJO.
     02 NOMBRE-TRA        PIC X(30).
     02 FILLER            PIC X(68).
     02 TELEFONO-TRA      PIC 9(7).
     02 CATEGORIA-TRA     PIC 9.
     02 FILLER            PIC X(20).
     02 TIPO-TRA          PIC 99.
     02 FILLER            PIC X(32).
FD   IMPRESORA
     DATA RECORD IS LINEA.
01   LINEA PIC X(132).
WORKING-STORAGE SECTION.
.
.
PROCEDURE DIVISION.
UNO.
     SORT TRABAJO
          DESCENDING CATEGORIA-TRA
          ASCENDING  TIPO-TRA
                     NOMBRE-TRA
          DESCENDING TELEFONO-TRA
          USING CLIENTES
          GIVING CLI-CLA.
```



```
     OPEN INPUT  CLI-CLA
          OUTPUT IMPRESORA.
DOS.
     READ CLI-CLA
          AT END GO TO FIN.
     .
     .
     WRITE LINEA.
     GO TO DOS.
FIN.
     CLOSE CLI-CLA
           IMPRESORA.
     STOP RUN.
```

4.1.3. Clasificación con tratamiento

El tratamiento de registros durante la clasificación, se puede realizar de dos maneras:

— Sobre los registros que son copiados del fichero de entrada hacia el fichero de trabajo.

— Sobre los registros que son copiados del fichero de trabajo hacia el fichero de salida.

En el primer caso, el tratamiento es anterior al proceso de clasificación y en el segundo caso el tratamiento es posterior al proceso de clasificación (Fig. 4.1).

En un proceso de clasificación se pueden emplear las siguientes combinaciones de tratamientos:

— Tratamiento anterior
— Tratamiento posterior
— Tratamientos anterior y posterior.

Cuando se realiza tratamiento sobre el proceso de clasificación, las instrucciones de cada tratamiento deben ir en una sección de la Procedure Division y las instrucciones del programa en otra sección. Por tanto, si se realiza un tratamiento (anterior o posterior), en la Procedure Division habrá una sección para las instrucciones del programa y otra para las instrucciones del tratamiento. Y si se realiza tratamiento anterior y posterior, en la Procedure Division habrá la

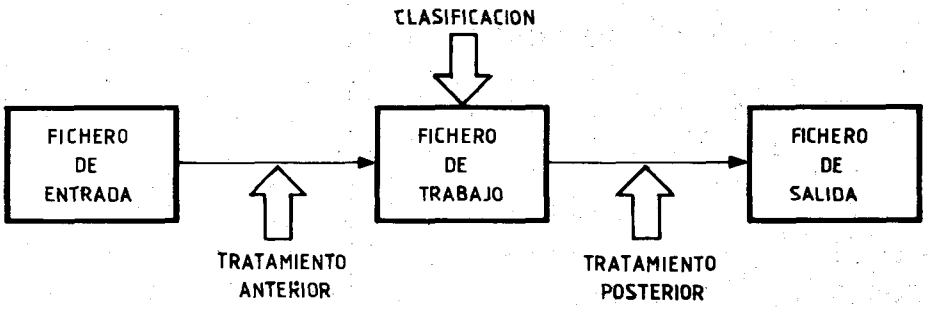
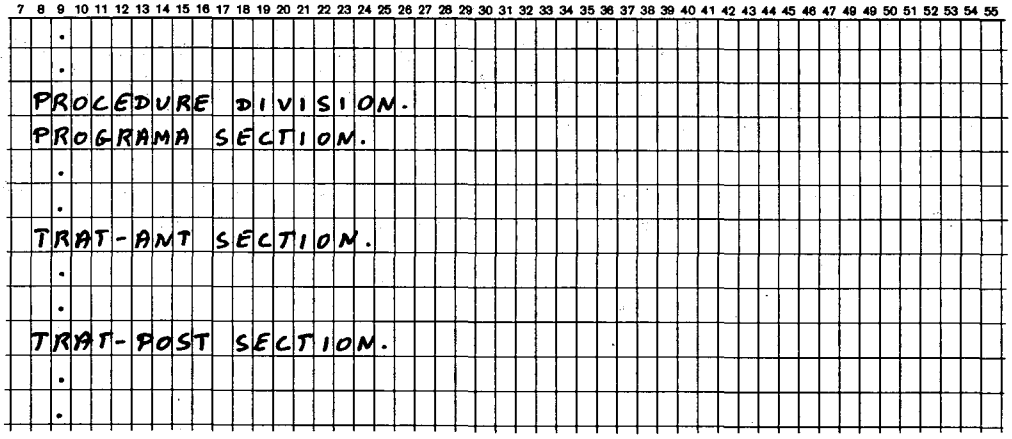


Fig. 4.1

sección con las instrucciones del programa, la sección con las instrucciones del tratamiento anterior y la sección con las instrucciones del tratamiento posterior.

Las secciones con instrucciones de tratamiento deben ser las últimas secciones dentro de la Procedure Division.



Tratamiento anterior

En este tipo de tratamiento, se sustituye la cláusula USING ... de la macroinstrucción SORT por la cláusula

INPUT PROCEDURE *nombre-sección*

Por tanto, el programador deberá codificar las instrucciones correspondientes para

— Activar el fichero de entrada
— Leer los registros del fichero de entrada
— Efectuar el tratamiento deseado
— Grabar los registros en el fichero de trabajo
— Desactivar el fichero de entrada

en la sección utilizada para el tratamiento anterior.

Todas las instrucciones son las utilizadas normalmente, excepto la de gra-

bación en el fichero de trabajo. Esta instrucción (que sustituye a la instrucción WRITE ...) es

RELEASE registro-trabajo [FROM campo]

Puesto que es el programador el que codifica las instrucciones para el acceso al fichero de entrada, dicho fichero puede estar organizado en cualquiera de las tres modalidades y el acceso será el que le corresponda de acuerdo al tratamiento a efectuar.

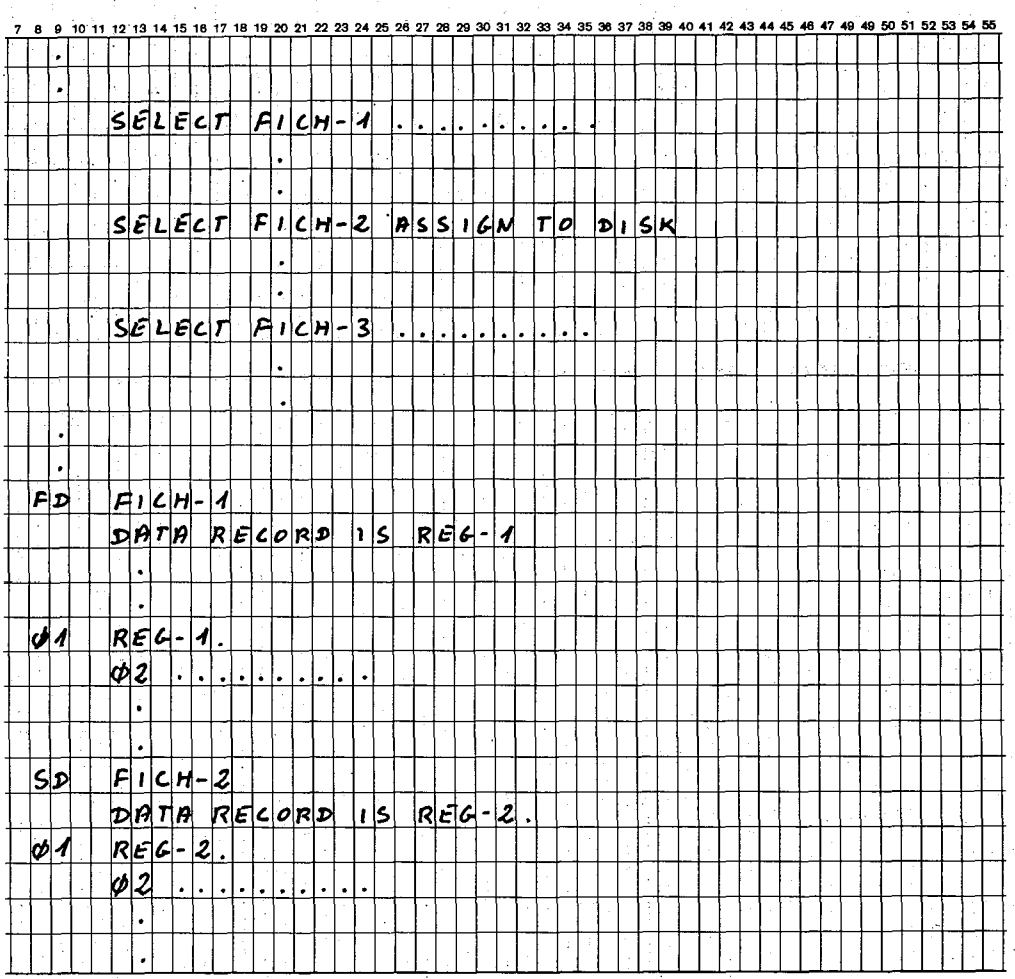
Con este tipo de tratamiento, cuando se ejecuta la macroinstrucción SORT, el ordenador bifurca hacia la sección indicada por la cláusula INPUT PROCEDURE ... ejecutando las instrucciones que contenga.

Una vez el ordenador alcanza el final de la sección, bifurca de nuevo hacia la SORT, reordena el fichero de trabajo y crea el fichero de salida.

▶

Ejemplo de codificación de un fichero con tratamiento anterior.

Codificación correspondiente a la clasificación del fichero FICH-1 con tratamiento anterior a su clasificación. El fichero de trabajo es FICH-2 y el fichero de salida es FICH-3.



```
FD   FICH-3
     DATA RECORD IS REG-3
       .
       .
01   REG-3.
     02 . . . . . .
       .
       .
  .
  .
PROCEDURE DIVISION.
UNO SECTION.
  .
  .
     SORT FICH-2
          ASCENDING . . . . . .
          DESCENDING . . . . . .
          INPUT PROCEDURE DOS
          GIVING FICH-3.
       .
       .
DOS SECTION.
ABRIR-FICH-1.
     OPEN INPUT FICH-1.
       .
       .
LEER-FICH-1.
```

```
       .
       .
     READ FICH-1
          INVALID KEY GO TO FIN-DOS.
TRATAMIENTO-DOS.
       .
       .
GRABACION-DOS.
     RELEASE REG-2.
     GO TO LEER-FICH-1.
FIN-DOS.
     CLOSE FICH-1.
```

Tratamiento posterior

Para lograr este tipo de tratamiento, se sustituye la cláusula GIVING ... por la cláusula

OUTPUT PROCEDURE nombre-sección

En la sección que realiza el tratamiento posterior a la clasificación, el programador debe codificar las instrucciones para:

— Activar el fichero de salida para grabación
— Leer los registros del fichero de trabajo
— Efectuar el tratamiento deseado
— Grabar los registros en el fichero de salida
— Desactivar el fichero de salida

La única instrucción que varía de las conocidas, es la de lectura de los registros del fichero de trabajo.

RETURN fichero-trabajo [INTO campo] AT END instrucción

(que sustituye a la instrucción READ ...)

Como el programador codifica las instrucciones para crear el fichero de salida, puede utilizar la organización y el acceso que considere oportuno.

Con este tratamiento, cuando el ordenador ejecuta la SORT, realiza todos los pasos hasta reorganizar el fichero de trabajo. Una vez hecho esto, bifurca hacia la sección indicada por la cláusula OUTPUT PROCEDURE..., ejecuta sus instrucciones y cuando alcanza el final de la sección, bifurca hacia la instrucción siguiente a la macro SORT.

▶

Codificación correspondiente al fichero FICH-1 del ejemplo anterior, pero con tratamiento posterior en lugar de tratamiento anterior.

```
PROCEDURE DIVISION.
UNO SECTION.
    .
    .
        SORT FICH-2
                ASCENDING ..........
                DESCENDING ..........
                USING FICH-1
                OUTPUT PROCEDURE TRES.
            .
            .
TRES SECTION.
ABRIR-FICH-3.
        OPEN OUTPUT FICH-3.
            .
            .
LEER-FICH-2.
        RETURN FICH-2
                AT END GO TO FIN-TRES.
TRATAMIENTO-TRES.
            .
            .
GRABACION-TRES.
            .
            .
        WRITE REG-3.
        GO TO LEER-FICH-2.
FIN-TRES.
        CLOSE FICH-3.
```

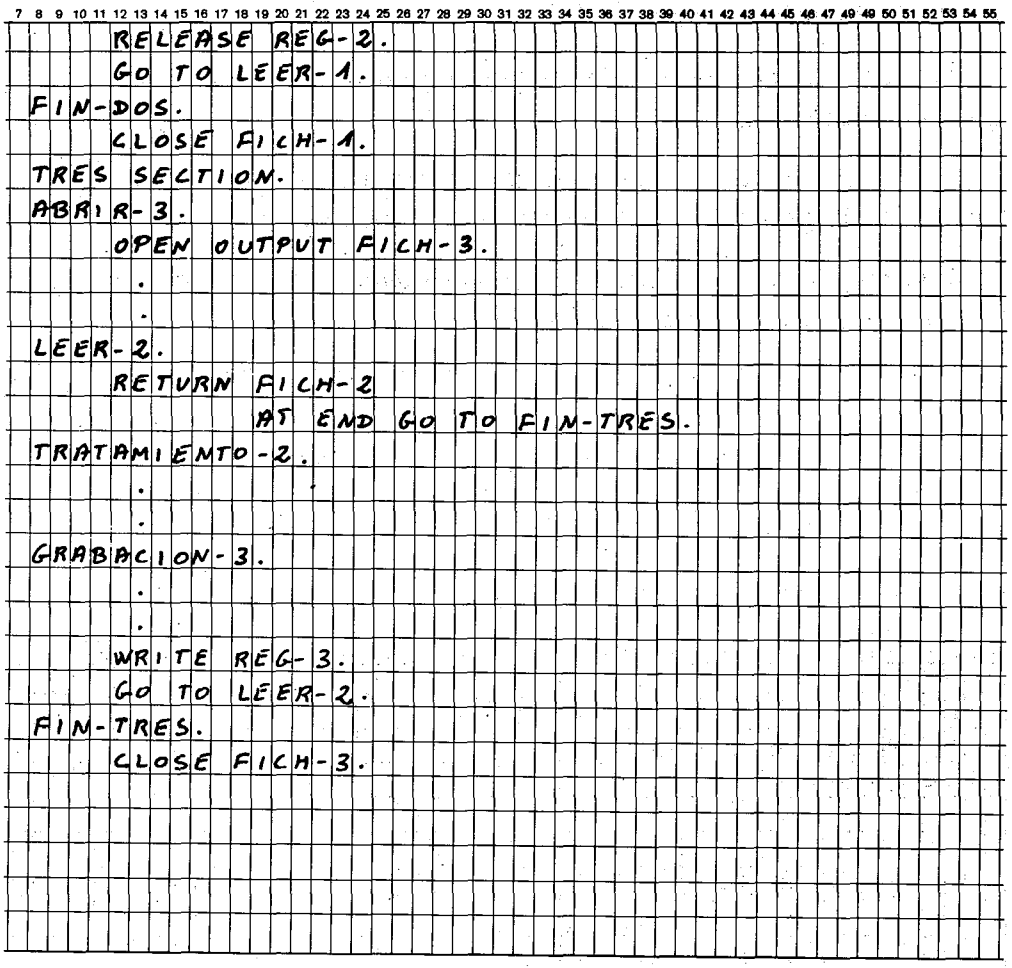
Tratamiento anterior y posterior

Este tipo de tratamiento es la combinación de los dos tipos de tratamiento (anterior y posterior).

El programador sólo utiliza la macroinstrucción SORT para realizar la clasificación (o reorganización) dentro del fichero de trabajo.

Un ejemplo de la combinación de los dos tratamientos con el fichero FICH-1 de los ejemplos anteriores, es el siguiente:

```
PROCEDURE DIVISION.
UNO SECTION.
    .
    .
        SORT FICH-2
             ASCENDING ..........
             DESCENDING ..........
             INPUT PROCEDURE DOS
             OUTPUT PROCEDURE TRES.
        .
        .
DOS SECTION.
ABRIR-1.
        OPEN INPUT FICH-1.
        .
        .
LEER-1.
        .
        .
        READ FICH-1
             INVALID KEY GO TO FIN-DOS.
TRATAMIENTO-1.
        .
        .
GRABACION-2.
```



```
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 49 50 51 52 53 54 55
                RELEASE REG-2.
                GO TO LEER-1.
FIN-DOS.
                CLOSE FICH-1.
TRES SECTION.
ABRIR-3.
                OPEN OUTPUT FICH-3.
                  .
                  .
LEER-2.
                RETURN FICH-2
                                AT END GO TO FIN-TRES.
TRATAMIENTO-2.
                  .
                  .
GRABACION-3.
                  .
                  .
                WRITE REG-3.
                GO TO LEER-2.
FIN-TRES.
                CLOSE FICH-3.
```

4.2. SEGMENTACION

Cuando la extensión de un programa sobrepasa la capacidad de la memoria interna del ordenador, debe procurarse reducir al máximo la cantidad de variables utilizadas. Si aún con esto, el programa sigue siendo demasiado extenso, se puede *trocear* la Procedure Division mediante la segmentación.

Esta técnica consiste en hacer grupos de instrucciones estrechamente relacionadas entre sí y que cada grupo pertenezca a una sección dentro de la Procedure Division.

Una vez se ha *segmentado* la Procedure Division (cada segmento o grupo es una sección), el programador debe clasificar las secciones de acuerdo al n.º de veces que las instrucciones que contienen son ejecutadas a lo largo de la ejecución del programa y también en su relación con las demás secciones.

Una vez esté decidida la importancia de cada sección, se indicará añadiendo a la cláusula SECTION un número

nombre-sección SECTION número

El compilador dividirá toda la Procedure Division en grupos de instrucciones de acuerdo a lo indicado por las definiciones de secciones. La técnica seguida por el ordenador en tiempo de ejecución es parecida a la que seguiría utilizando varios subprogramas ya que cada sección sería un subproceso dentro del proceso que realiza el programa.

En la memoria interna estarán almacenadas las secciones que tengan el mismo número de importancia y las que tengan distinto número el ordenador las mantendrá (a la espera de ser pasadas a la M.C.) en un fichero del sistema.

Las secciones que tengan como número de importancia el cero, estarán siempre en la memoria interna.

La segmentación permite que las rutinas menos empleadas en el programa estén fuera de la memoria interna y que sólo estén mientras se las necesite.

El programador no necesita codificar llamadas de ningún tipo para llevar los segmentos a la memoria principal, basta una referencia en el segmento que se esté ejecutando a cualquier instrucción de otro segmento para que el mismo ordenador genere la llamada.

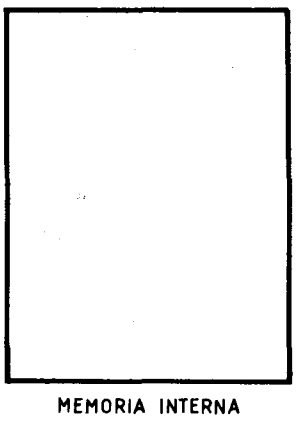
Ejemplo-1:

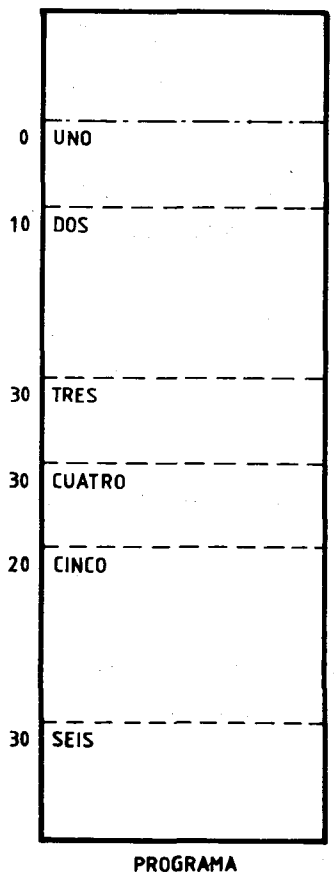
Se tiene un programa cuya longitud excede la capacidad de la memoria interna (Fig. 4.2).

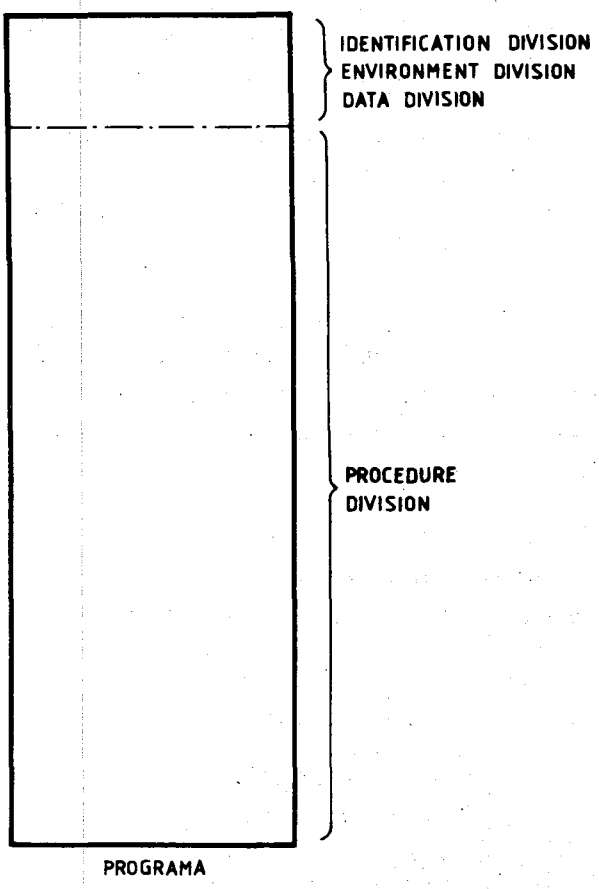
El primer paso es seccionar la Procedure Division con un segmento con importancia cero que estará siempre en la memoria y que dirigirá los pasos del programa.

El siguiente paso es asignar números de importancia a cada sección, dando un número igual a las secciones que deban estar juntas en la memoria interna y con números de importancia distintos las secciones que puedan estar *fuera* (Fig. 4.3).

Fig. 4.2 ▶


MEMORIA INTERNA

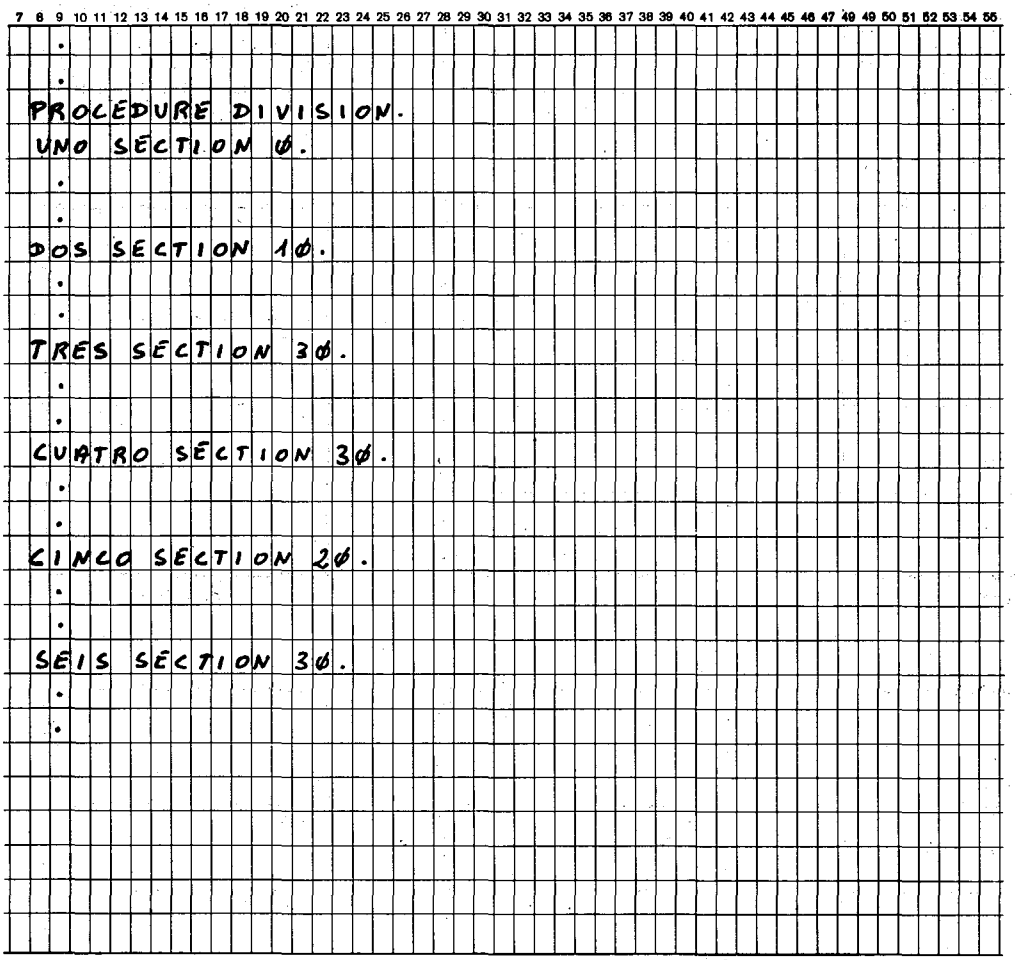

0
UNO
10
DOS
30
TRES
30
CUATRO
20
CINCO
30
SEIS
PROGRAMA


IDENTIFICATION DIVISION
ENVIRONMENT DIVISION
DATA DIVISION
PROCEDURE
DIVISION
PROGRAMA

◀ Fig. 4.3

La codificación de la figura 4.12 corresponde al esquema de la figura 4.3.

Un ejemplo gráfico de la ejecución de programa de la figura 4.3 es el de la figura 4.4.



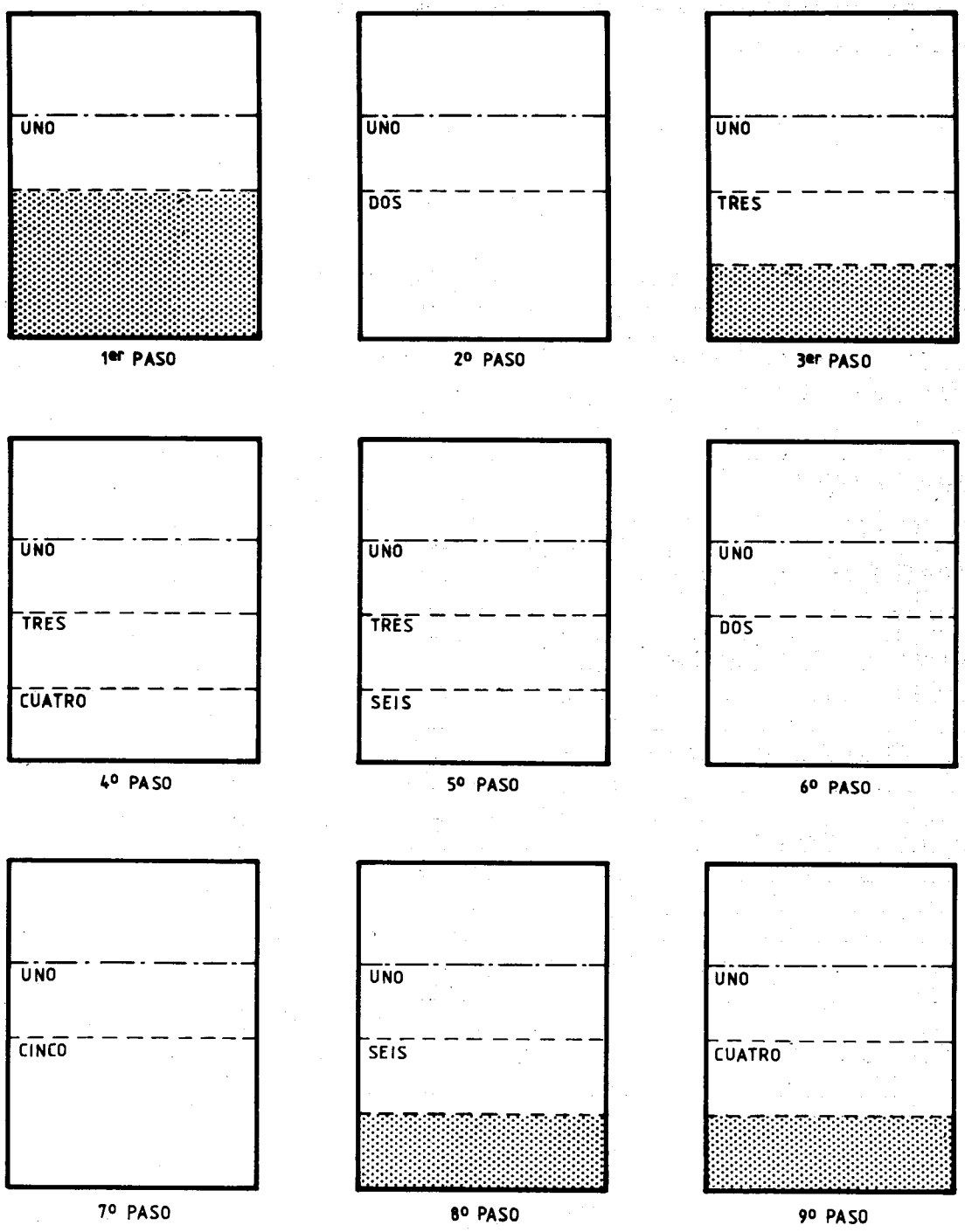

UNO
1er PASO
UNO
DOS
2º PASO
UNO
TRES
3er PASO
UNO
TRES
CUATRO
4º PASO
UNO
TRES
SEIS
5º PASO
UNO
DOS
6º PASO
UNO
CINCO
7º PASO
UNO
SEIS
8º PASO
UNO
CUATRO
9º PASO

Fig. 4.4

Ejemplo-2:

Los cometidos de las secciones descritas en la codificación que se presenta en estas dos páginas son los siguientes:

UNO: Sección que trata los errores del fichero utilizado en la sección CINCO.

DOS: Esta sección trata los errores del fichero utilizado en la sección OCHO.

TRES: Trata los errores del fichero utilizado en la sección SEIS.

```
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 49 50 51 52 53 54 55
      .
      .
PROCEDURE DIVISION.
DECLARATIVES.
UNO SECTION 50.
      .
      .
DOS SECTION 10.
      .
      .
          GO TO FIN-ERRORES.
TRES SECTION 70.
      .
      .
FIN-ERRORES.
END DECLARATIVES.
CUATRO SECTION 0.
      .
      .
          SORT ..........
                    INPUT PROCEDURE CINCO
                    OUTPUT PROCEDURE SEIS.
               .
               .
          PERFORM SIETE-1 THRU SIETE-99.
               .
               .
```

CUATRO: Contiene las instrucciones que es necesario que estén siempre en la memoria interna.

CINCO: Tratamiento anterior correspondiente al SORT de la sección CUATRO.

SEIS: Tratamiento posterior de la macro SORT.

SIETE: Rutina del programa que se utilizará pocas veces, por lo cual estas instrucciones no se codifican dentro de la sección CUATRO.

OCHO: Otra rutina que se emplea como subrutina de la sección SIETE.

```
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 49 50 51 52 53 54 55
CINCO SECTION 50.
    .
    .
SEIS SECTION 70.
    .
    .
SIETE SECTION 10.
SIETE-1.
          .
          .
     IF .......... GO TO OCHO-1.
          .
          .
SIETE-99.
          .
          .
OCHO SECTION 10.
OCHO-1.
          .
          .
OCHO-99.
     EXIT.
```

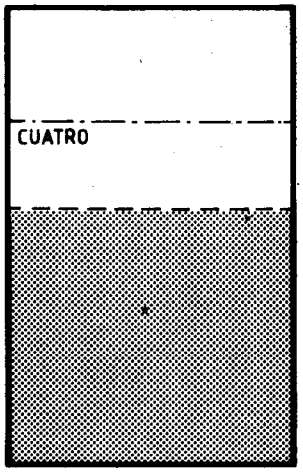


Fig. 4.5 ▶

La figura 4.5 es el primer paso efectuado por el ordenador durante la ejecución del programa es:

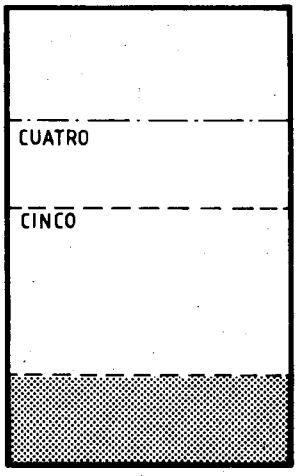


Fig. 4.6 ▶

Al ejecutarse el tratamiento anterior de la macro SORT la estructura de la memoria interna es el de la figura 4.6.

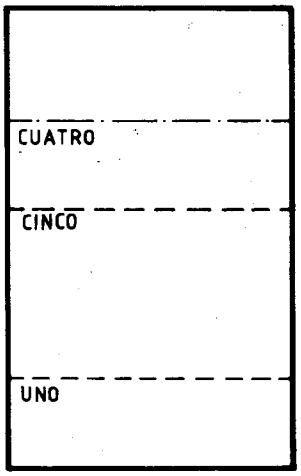


Fig. 4.7 ▶

Si durante este paso aparece algún error en el fichero de entrada la estructura será la de la figura 4.7.

En caso contrario, a continuación se realizará el tratamiento posterior (Fig. 4.8). Si hay algún error en el fichero de salida se cargará la sección TRES (Fig. 4.9).

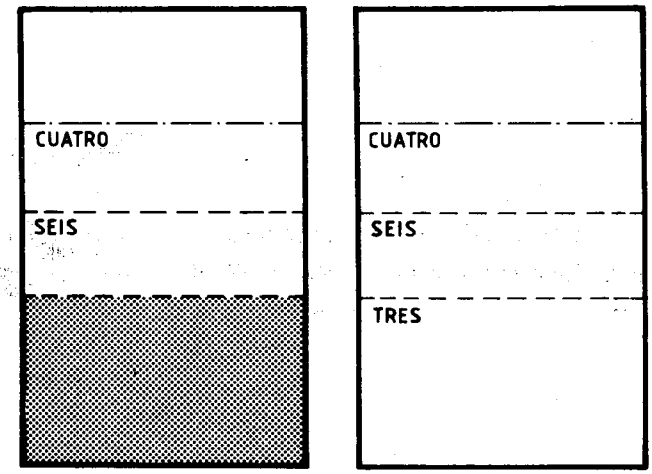


Fig. 4.8 y 4.9 ▶

La figura 4.10 corresponde a la ejecución de la rutina de la sección SIETE.

Si en esta fase, se cumple la condición

IF ..... GO TO OCHO-1

El mismo ordenador carga la sección OCHO entera para poder ejecutar las instrucciones del párrafo OCHO-1. (Fig. 4.11).

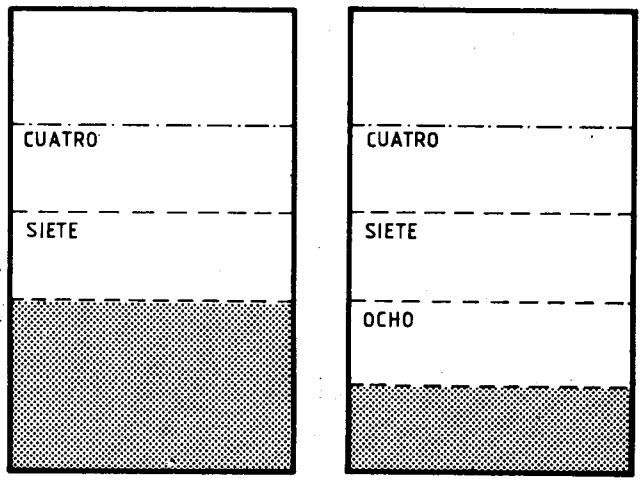


Fig. 4.10 y 4.11 ▶

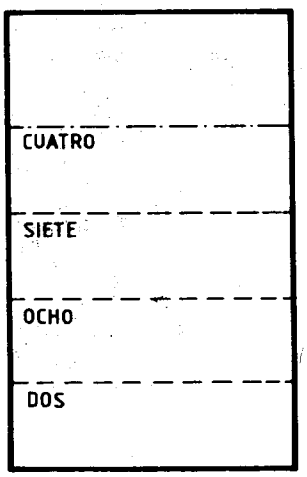


Fig. 4.12 ▶

Si es necesaria la sección DOS, la memoria interna tendrá la disposición de la figura 4.12.

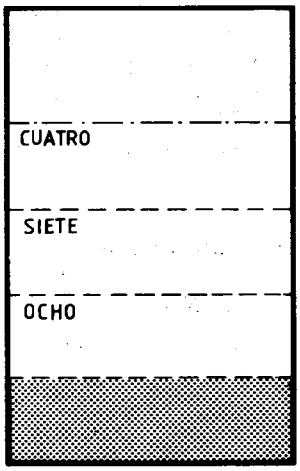


Fig. 4.13 ▶

Al ejecutarse

GO TO FIN-ERRORES

el control vuelve a la siguiente instrucción de la sección OCHO que originó el error (Fig. 4.13).

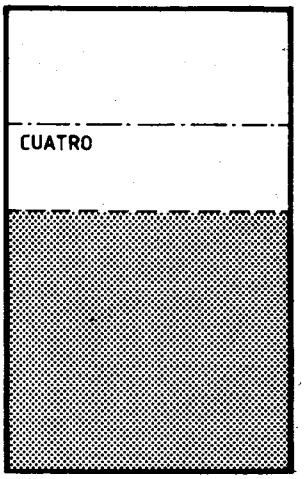


Fig. 4.14 ▶

Al ejecutarse la

EXIT

el control vuelve a la sección CUATRO (Fig. 4.14).

EJERCICIOS

1. Los procedimientos de clasificación se realizan sobre:

   a) Los registros de un fichero
   b) Los datos de un fichero
   c) Los campos de un registro

2. ¿Se necesita siempre un fichero de trabajo para realizar una clasificación?

   a) No
   b) Depende del tipo de ordenador
   c) Sí

3. ¿Cuántas cláusulas (dentro de la Environment Division) son necesarias para definir un fichero de trabajo?

   a) Depende del tipo de clasificación a efectuar
   b) Sólo la cláusula SELECT
   c) El fichero de trabajo no necesita ser definido

4. En la clasificación con tratamiento anterior el fichero de salida debe estar organizado

   a) De manera secuencial
   b) De manera aleatoria o indexada
   c) Según sean los datos almacenados

5. La instrucción para leer los registros del fichero de trabajo es la .....

6. ¿Cuántas secciones puede tener un programa con segmentación?

   a) Por lo menos dos secciones
   b) Tres o más secciones
   c) Depende de los números de importancia

7. Para que dos secciones no estén almacenadas al mismo tiempo en la memoria interna, sus números de importancia han de ser:

   a) Distintos
   b) Iguales
   c) Cero

8. ¿Qué número de importancia tendrán las secciones que están siempre en la memoria interna durante la ejecución del programa?

SOLUCIONES DE LOS EJERCICIOS

1. Manejo de tablas

1. *b)*; 2. OCCURS; 3. *c)*; 4. *a)*; 5. *a)*; 6. *c)*; 7. *b)*.

2. Máscaras

1. LINE 5 COL 3; 2. *c)*; 3. *a)*; 4. *b)*; 5. *a)*; 6. *c)*; 7. *b)*; 8. *c)*.

3. Tratamiento de ficheros

1. *c)*; 2. *c)*; 3. *a)*; 4. Secuencial, relativa, indexada; 5. *b)*; 6. DECLARATIVES. END DECLARATIVES; 7. No permitir su tratamiento desde otro programa; 8. Ninguna.

4. Técnicas de optimación

1. *a)*; 2. *c)*; 3. *b)*; 4. *a)*; 5. RETURN; 6. *b)*; 7. *a)*; 8. CERO.

Indice general

Contenido 7

1. MANEJO DE TABLAS

Cuadro resumen de la lección 11
1.1. Definición de una tabla 12
1.2. Referencia a un elemento 15
1.2.1. Subíndices 15
1.2.2. Indices 19
1.3. Redefinición de tablas 24
1.4. Definición de índices fuera de la tabla 31
1.5. Modificación de índices 34
1.6. Exploración de una tabla 38

Ejercicios 48

2. MASCARAS

Cuadro resumen de la lección 53
2.1. Definición de máscaras 54
2.2. Posición de los campos 56
2.3. Campos con constantes 58
2.4. Campos con variables 59
2.5. Transferencia de datos hacia la pantalla 63
2.6. Transferencia de datos desde el teclado 66
2.7. Borrado de la pantalla 67

2.8. Campos con display intermitente 70
2.9. Campos alineados por la derecha 71
2.10. Sustitución de ceros por blancos 72
2.11. Campos de seguridad 73
2.12. Campos obligatorios 74
2.13. Llamada de atención 75
2.14. Cambio automático de campo 76
2.15. Borrado de una línea 77
2.16. Scrolling 79

Ejercicios 82

3. TRATAMIENTO DE FICHEROS

Cuadro resumen de la lección 87
3.1. Definición de un fichero 88
3.1.1. Soporte físico y nombre 89
3.1.2. Etiqueta 90
3.1.3. Longitud de los bloques 90
3.1.4. Longitud de los registros 92
3.1.5. Descripción de los registros 95
3.1.6. Detección de errores 95
3.1.7. Buffer compartido 97
3.2. Organización 98
3.3. Acceso 100
3.3.1. Acceso secuencial 101
— Apertura de un fichero secuencial 102
— Lectura secuencial de registros 104
— Grabación secuencial de registros 108
— Cierre de un fichero 111
— Modificación de un fichero secuencial 113
— Ampliación de un fichero secuencial 116
— Desbloqueo 117
3.3.2. Acceso directo 118
— Apertura de un fichero directo 120
— Lectura directa de registros 122
— Grabación directa de registros 125
— Cierre de un fichero directo 125
— Modificación de un fichero directo 126
— Desbloqueo 128

3.3.3. Acceso indexado 129
3.4. Eliminación de registros 130
3.5. Tratamiento de errores 132

Ejercicios 138

4. TECNICAS DE OPTIMIZACION

Cuadro resumen de la lección 143
4.1. Clasificación 144
4.1.1. Fichero de trabajo 144
4.1.2. Clasificación sin tratamiento 145
4.1.3. Clasificación con tratamiento 151
— Tratamiento anterior 152
— Tratamiento posterior 156
— Tratamiento anterior y posterior 158
4.2. Segmentación 159

Ejercicios 169

Soluciones a los ejercicios 171

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.