R.M. cobol

MAS DE 60 PROGRAMAS

versiones 2.ØX

F. Charte Ojeda

A. Ligero Torres

COBOL

EL LIBRO DEL PROGRAMADOR

Edita: *D. Francisco Charte Ojeda y D. Armando Ligero Torres*
Imprime: *Gráficas Jaén, S.C.L.*
Autor: *D. Francisco Charte Ojeda y D. Armando Ligero Torres*
Depósito Legal: *J. 381 - 1987 - I.S.B.N.: 84 - 398 - 8843 - 0*

Autores:

D. Francisco Charte Ojeda
D. Armando Ligero Torres

Prohibida la reproducción total o parcial de esta obra,
por cualquier medio sin autorización de los propietarios.

I. B. M. es marca registrada de I. B. M.
MS-DOS es marca registrada de MICROSOFT.
UNIX es marca registrada de AT&T Corp.
XENIX es marca registrada de MICROSOFT.
RM-COBOL es marca registrada de Ryan McFarland.
PROLOGUE es marca registrada de BULL.

Este libro está pensado y escrito de una forma eminentemente práctica, dejando de lado la teoría todo lo posible. Se comienza desde cero en la metodología y programación en Cobol, sin embargo se suponen unos conceptos mínimos de informática básica sin los cuales será dificil entender palabras como 'registro' o 'fichero', esenciales para el entendimiento y uso de los mismos. Ante posibles dudas en alguno de estos conceptos remítase el lector a la bibliografía necesaria para entenderlo. A lo largo del libro y en cada momento se proponen programas, es más que aconsejable que estos se vayan haciendo y lo que es más importante, que se entiendan en su totalidad. Nunca pase a la siguiente instrucción si le ha quedado alguna duda de la que está aprendiendo, ya que seguramente llegará un momento en que se confunda. La finalidad de este libro es que cuando termine de usarlo , y no sólo de leerlo, sepa sacarle el mayor provecho posible a este lenguaje, estaré contento si lo he conseguido.

Los autores.

INDICE

1.- EL COBOL.
1.1. Un poco de historia 11
1.2. Orientación del Cobol 11

2.- ESTRUCTURA DEL COBOL.
2.1. Divisiones 11
2.2. Secciones 12
2.3. Párrafos 13
2.4. Márgenes 13
2.5. Puntos 14
2.6. Palabras en Cobol 14
2.7. Signos de puntuación 15
2.8. Constantes y variables 15
2.9. Constantes figurativas 15

3.- IDENTIFICATION DIVISION.
3.1. La cláusula PROGRAM-ID 17
3.2. La cláusula INSTALLATION 17
3.3. La cláusula DATE-WRITTEN 17
3.4. La cláusula DATE-COMPILED 17
3.5. La cláusula SECURITY 17

4.- ENVIRONMENT DIVISION.
4.1. La cláusula SOURCE-COMPUTER 18
4.2. La cláusula OBJECT-COMPUTER 19
4.3. La cláusula SPECIAL-NAMES 19
4.4. La INPUT-OUTPUT SECTION 19

5.- DATA DIVISION.
5.1. La FILE SECTION 20
5.2. La WORKING-STORAGE SECTION 20

6.- UTILIZACION DEL COBOL.
6.1. La compilación 27

7.- INSTRUCCIONES BASICAS.
7.1. La instrucción DISPLAY, primeros pasos 29
7.2. La instrucción MOVE, primeros pasos 32
7.3. La instrucción ACCEPT, primeros pasos 33
7.4. La instrucción GO TO, primeros pasos 36
7.5. La instrucción IF, primeros pasos 37
7.6. La instrucción COMPUTE, primeros pasos 39

8.- INSTRUCCIONES COBOL.
8.1. La instrucción ACCEPT, utilización avanzada 42
8.2. La instrucción DISPLAY, utilización avanzada51
8.3. Instrucciones aritméticas

8.3.1. Sumando con la instrucción ADD 52
8.3.2. Restando con la instr. SUBTRACT 56
8.3.3. Multip. con la instr. MULTIPLY 57
8.3.4. Dividiendo con la instr. DIVIDE 58
8.3.5. La instr. COMPUTE. 58
8.4. Instrucciones de transferencias de control
8.4.1. La instrucción GO TO 61
8.4.2. La instrucción ALTER 63
8.4.3. La instrucción STOP 66
8.5. La instrucción IF, utilización avanzada 68
8.6. Instrucciones de manipulación de datos
8.6.1. La instr. MOVE. 78
8.6.2. La instr. INSPECT 79
8.7. La instrucción PERFORM, rutinas y bucles 85
8.8. Manejando tablas con la instrucción SET 96
8.9. Modulación de programas
8.9.1. La instr. CALL 101
8.9.2. La instr. CANCEL 106

9.- UTILIZACION DE FICHEROS.
9.1. La INPUT-OUTPUT SECTION 106
9.2. La I-O-CONTROL 109
9.3. La FILE SECTION 113

10.- INSTRUCCIONES RELATIVAS A FICHEROS.
10.1. Abriendo ficheros con la instrucción OPEN 116
10.2. Escribiendo con la instrucción WRITE 119
10.3. Leyendo con la instrucción READ 123
10.4. Cerrando ficheros con la instrucción CLOSE 128
10.5. Mas sobre WRITE 135
10.6. Mas sobre READ 139
10.7. Borrando con la instrucción DELETE 140
10.8. Modificando con la instrucción REWRITE 158
10.9. La instrucción START 162
10.10 La instrucción UNLOCK 163

11.- La cláusula COPY. 163

12.- Las DECLARATIVES. 177

13.- PROGRAMAS.
13.1. De minúsculas a mayúsculas 182
13.2. Editando pantallas 184
13.3. Almacenando pantallas 214
13.4. Las tres en raya 230

14. APLICACION DE GESTION DE SUPERMERCADOS. 250

INDICE

15.- ERRORES EN FICHEROS. 368

16.- ERRORES DE COMPILACION 371

17.- INDICE ALFABETICO 385

EL COBOL

1.¿QUE ES EL COBOL?

1.1. UN POCO DE HISTORIA.

El COBOL (COMMON BUSSINES ORIENTED LANGUAJE), es un lenguaje de programación que apareció como tal en el año 1.960 en el comité ejecutivo de la Conferencia sobre Lenguajes de Sistemas. Desde aquella fecha este mismo comité se encargó de sucesivas revisiones de mejoras y ampliaciones hechas al lenguaje, siendo el más utilizado de la historia de la informática. En sus inicios el Cobol fué concebido para trabajar en máquinas grandes, en el año 1.968 el Instituto Americano de Estandarización promovió la creación de un Cobol capaz de incluirse en equipos más pequeños, a este se le denominó Cobol Ansi.

1.2. ORIENTACION DEL COBOL.

El Cobol es un lenguaje orientado a los procesos de grandes masas de datos, es un lenguaje de los llamados comerciales, su potencia radica en la posibilidad de manejar grandes ficheros de datos con suma facilidad y rapidez. El Cobol no es el único lenguaje orientado hacia este campo, pero si es el más transportable de ellos ya que el mismo programa puede correr bajo los sistemas MS-DOS, UNIX, XENIX, etc..., y en los equipos más dispares con mínimas o ausentes modificaciones. A lo largo de este libro se tratarán las especificaciones del RM-COBOL, este es el más extendido en la mayoría de los pequeños y medianos ordenadores.

2.- ESTRUCTURA DEL COBOL.

El Cobol es un lenguaje estructurado en el cual se pueden diferenciar varias partes, cada una de las cuales tiene una finalidad distinta.

2.1. DIVISIONES.

Todo programa Cobol está compuesto de cuatro divisiones. A continuación vamos a ver el nombre que reciben cada una de ellas y su finalidad. Estas divisiones Cobol han de aparecer obligatoriamente y en el mismo orden en que ha continuación se presentan.

IDENTIFICATION DIVISION..: Esta es la primera división que ha de aparecer en todo programa Cobol, la finalidad de esta división es la identificación del programa, en ella especificaremos el nombre del programa, el del

autor, la fecha en que fué escrito y compilado y algunas cosas más. Algunas de estas especificaciones son opcionales y otras obligatorias.

ESTRUCTURA DEL COBOL

ENVIRONMENT DIVISION..: Esta segunda división se utiliza para describir el entorno del trabajo del Cobol, en ella espedificaremos lo ordenadores en los cuales fué escrito y se ejecutará el programa, así como la relación de ficheros que utilizará el programa con los periféricos correspondientes.

DATA DIVISION..: En esta tercera división tendremos que declarar todos los campos y variables que vayamos a utilizar en nuestro programa, así como los registros de los ficheros que utilicemos.

PROCEDURE DIVISION..: En la cuarta y última división se encuentra el programa propiamente dicho, en ella daremos las ordenes pertinentes al programa, aquí utilizaremos todos los campos declarados en la **DATA DIVISION** así como los ficheros declarados en la **ENVIRONMENT DIVISION.**

2.2. SECCIONES.

Cada una de las divisiones vistas anteriormente puede estar dividida a su vez en secciones, por ejemplo la **ENVIRONMENT DIVISION** ha de tener dos secciones obligatoriamente, la **DATA DIVISION** puede tener una, dos o tres secciones y la **PROCEDURE DIVISION** también puede estar dividida. A continuación vamos a ver las posibles secciones que podemos tener en cada división.

CONFIGURATION SECTION..: Esta es la primera de las secciones, obligatoria, en la **ENVIRONMENT DIVISION**, en ella especificaremos el ordenador en el que fué escrito el programa, el ordenador donde se ejecutará dicho programa; así mismo también determinaremos una serie de símbolos para su posterior utilización.

INPUT-OUTPUT SECTION..; Esta segunda sección de la **ENVIRONMENT** es opcional, en ella relacionaremos los ficheros que vayamos a utilizar con sus correspondecias externas, si no vamos a utilizar ficheros se puede omitir.

FILE SECTION..: Esta es la primera sección que ha de aparecer en la **DATA DIVISION**, su utilización es opcional; en ella definiremos los registros de cada uno de los ficheros que vayamos a utilizar y que con anterioridad se habrán declarado en la **INPUT-OUTPUT SECTION**.

WORKING-STORAGE SECTION..: Es la segunda de las secciones que puede aparecer en la **DATA DIVISION**, su utilización también es opcional; su misión es acoger la definición de todos los campos o variables que vayamos a necesitar a lo largo del programa, la utilización de cualquier variable no declarada tendrá como resultado un error.

LINKAGE SECTION..: Tercera y última sección que puede aparecer en la **DATA DIVISION**, su utilización es opcional al igual que las dos secciones anteriores; se hace precisa cuando se desean conservar valores que han de pasar de un programa a otro.

2.3. PARRAFOS.

Dentro de cada una de las secciones vistas anteriormente puede haber una serie de párrafos, estos pueden ser obligatorios u opcionales, en la **PROCEDURE DIVISION** los nombres de dichos párrafos los estableceremos nosotros mismos, nos servirán para desviar la ejecución del programa a un cierto punto entre otras cosas.

2.5. MARGENES.

A la hora de escribir un programa hay que tener en cuenta la división en varios márgenes de que hace uso el Cobol. Las 80 columnas de que se hace uso se dividen de la siguiente forma:

Columnas 1 a 6..: Estas columnas las usa el compilador de Cobol para ir numerando cada una de las líneas del programa, al escribirlo las líneas no se han de numerar, de ello se ocupa el compilador.

Columna 7..: Esta columna se utiliza para la inclusión de un guión como continuación de un literal que por su extensión no coge en una sola línea, al poner el guión el compilador asumirá la línea como continuación de la anterior. Si en esta misma columna se coloca un asterisco '*', la línea en curso se tomará como un comentario y el compilador la ignorará.

Columnas 8 a 72..: En estas columnas se escribirán las declaraciones e instrucciones del programa, se divide a su vez en dos márgenes que se denominan..: margen 'A', vá desde la columna 8 a la 11, en este margen han de comenzar todos los nombres de divisiones, secciones y párrafos. Margen 'B', vá desde la columna 12 a la 72, en ella se incluyen todas las instrucciones y declaraciones del programa.

Columnas 73 a 80..: Estas columnas son ignoradas por el compilador, cualquier instrucción puesta en este margen será omitida en el programa.

2.5. PUNTOS.

El punto constituye en el Cobol el final de una línea de programa, han de terminar con punto todas las divisiones, secciones y párrafos, así como cada línea de programa. Si a lo largo de una línea física no se encuentra un punto la siguiente línea será tomada como continuación de la anterior. La línea anterior a cualquier división, sección o párrafo ha de terminar con punto a excepción de la línea anterior a la **IDENTIFICATION DIVISION.**

2.6. PALABRAS EN COBOL.

El Cobol dispone de una serie de palabras reservadas que iremos viendo a lo largo de esta libro, además de estas, en un programa usaremos nombres creados por nosotros mismos para párrafos y variables, estos nombres no podrán contener nunca a una palabra reservada. Todas las palabras Cobol, tanto reservadas como creadas por nosotros, se han de escribir en mayúsculas, ya que de lo contrario el compilador no las reconocería. El nombre de un párrafo o variable no debe exceder de 32 caracteres, estos pueden ser.: letras (A a Z), dígitos numéricos (0 a 9) y guiones; cualquier otro signo incluido diferente de los aquí descritos tendrá como resultado que el compilador no reconozca el campo o párrafo que lo contiene. Cuando se definen nombres de variables estos han de tener obligatoriamente una letra al menos, cuando el nombre vá a ser usado por un párrafo no es obligatorio que sea así.

A continuación tenemos unos ejemplos de nombres de variables y párrafos, no todos son correctos, a su lado tienes la causa.

NOMBRE	CORRECTO
PROVINCIA-POBLACION	CORRECTO
NOMBRE-APELLIDO-UNO-APELLIDO-DOS-DIRECCI	MAS DE 32 CARACTERES
DOLARE$	CARACTER '$' INVALIDO
-TELEFONO	CARACTER '-' INVALIDO
IMPORTE	CORRECTO
HORA MINUTOS	CARACTER ' ' INVALIDO
letras	MINUSCULAS INVALIDAS

2.7. SIGNOS DE PUNTUACION.

Además del punto para indicar final de una línea de programa, al escrribir un programa Cobol se pueden usar lo signos de coma ',' y punto y coma ';' para hacerlo más claro, en ningún caso es obligatorio a excepción del punto. Cuando se utilicen se ha de tener en cuenta que delante de uno de estos tres signo no puede haber un espacio en blanco y que detrás de dicho signo se hace obligatoria la presencia del espacio.

2.8. CONSTANTES Y VARIABLES.

En Cobol pueden existir dos tipos de constantes y variables, numéricas y literales. Las constantes y variables numéricas estarán compuestas de un máximo de 18 dígitos, las literales o alfanuméricas no tienen una longitud preestablecida. Las constantes numéricas se diferencian de las literales en que estas últimas han de aparecer siempre entrecomilladas, en las variables la única diferencia está al declararlas en la **DATA DIVISION.**

2.9. CONSTANTES FIGURATIVAS.

Una constante figurativa es una palabra reservada Cobol que puede usarse en cualquier parte de la **WORKING-STORAGE SECTION** o **PROCEDURE DIVISION** para dar valor a un campo. En RM-COBOL podemos usar las siguientes.

ZERO, ZEROS, ZEROES..: Se utilizan para dar valor a un campo numérico, el efecto es que tal campo toma tantos ceros como longitud tenga.

SPACE, SPACES..: Se utilizan para dar valor a campos no numéricos, el efecto es que tal campo toma tantos espacios como longitud tenga.

HIGH-VALUE, HIGH-VALUES..: Se utilizan para dar valor tanto a campos numéricos como alfanuméricos, el efecto

es que tal campo toma el mayor valor posible en su definición.

LOW-VALUE, LOW-VALUES..: Se utilizan para dar valor tanto a campos numéricos como alfanuméricos, el efecto es que el campo toma el menor valor posible en su definición.

QUOTE, QUOTES..: Se utilizan para dar valor a campos no numéricos, el efecto es que el campo toma uno o varios caracteres de comillas.

ALL "literal"..: Se utiliza para dar valor a campos no numéricos, el efecto es que el campo se llena del literal especificado.

Una vez vistas las generalidades del Cobol vamos a comenzar a ver cada de sus divisiones, secciones, párrafos e instrucciones. A los largo de esta parte se empleará la siguiente notación:

[xxxx] - Los elementos contenidos entre corchetes son opcionales.

MOVE - Las palabras subrayadas en negrita son palabras reservadas Cobol de aparición obligatoria a no ser que estén entre corchetes.

También en esta parte y a medida que vayamos viendo cada una de las instrucciones Cobol se irán proponiendo ejercicios y programas, lo conveniente es que se hagan para comprobar lo que se vá aprendiendo. Nunca te conformes con los programas que aquí se proponen, desde el primer momento intenta hacer los tuyos propios o mejorar los que aquí tengas.

3.- IDENTIFICATION DIVISION.

Como se dijo antes la **IDENTIFICATION DIVISION** es la parte de un programa Cobol donde se identifica el mismo, su síntaxis es la siguiente.

```
123456789012
      A  B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. nombre de programa.
      [AUTHOR. nombre del autor.]
      [INSTALLATION. lugar de instalación.]
```

IDENTIFICATION DIVISION

[DATE-WRITTEN. fecha de realización.]
[DATE-COMPILED. fecha de compilación.]
[SECURITY. comentarios.]

Podemos apreciar que además del nombre de la división el único párrafo obligatorio es donde se dá el nombre del programa, este nombre se utilizará cuando ocurra algún error en la ejecución para comunicarnos en que programa fué. Todos los demás párrafos son opcionales y su inclusión no afecta para nada al programa. A continuación vamos a ver uno a uno todos los párrafos que componen esta división.

3.1. **PROGRAM-ID..**: Este párrafo se utiliza como se ha dicho antes para dar nombre al programa Cobol, este nombre ha de empezar obligatoriamente por una letra y puede tener cualquier longitud, sin embargo el compilador sólo tendrá en cuenta los seis primeros caracteres, este nombre no ha de ser obligatoriamente el mismo que el archivo con el programa fuente que esté en el disco, sólo se utiliza en el caso de producirse un error.

3.2. **INSTALLATION..**: Este párrafo se utiliza para comunicar en que lugar fué realizado el programa, es opcional.

3.3. **DATE-WRITTEN..**: En este párrafo podremos especificar la fecha de ralización del programa, es opcional.

3.4. **DATE-COMPILED..**: La fecha en que se compiló el programa es la información acogida por este párrafo que al igual que los dos anteriores es opcional.

3.5. **SECURITY..**: Este párrafo se utiliza para comunicar una información acerca de la seguridad del programa o ficheros que utiliza.

Además de estos párrafos el programador puede incluir todos los comentarios que crea convenientes haciendo uso del '*'. A continuación tenemos un ejemplo de **IDENTIFICATION DIVISION**, en ella se identifica un programa llamado 'CONTA', este programa fué escrito por José Pérez en C.E.I. el día 26 de Octubre de 1986, fué compilado 5 dias despues.

```
123456789012
      A  B
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CONTA.
AUTHOR. JOSE-PEREZ.
INSTALLATION. CENTRO-DE-ESTUDIOS-INFORMATICOS.
DATE-WRITTEN. 26-OCTUBRE-1986.
DATE-COMPILED. 31-OCTUBRE-1986.
SECURITY. CONSULTAR-MANUAL-REFERENCIA.
```

Esto mismo se podía haber hecho de la siguiente forma teniendo el mismo efecto en el programa.

```
123456789012
      A   B

      IDENTIFICATION DIVISION.
      PROGRAM-ID. CONTA.
```

4.- ENVIRONMENT DIVISION.

La siguiente división de un programa Cobol es la **ENVIRONMENT DIVISION**, en ella como vimos antes especificabamos los ordenadores donde se había escrito y ejecutado el programa, además de esto se relacionaban los ficheros a utilizar con sus correspondencias externas. El formato de esta división es el que sigue.

```
123456789012
      A   B

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SOURCE-COMPUTER. ordenador-fuente.
      OBJECT-COMPUTER. ordenador-objeto.
      [SPECIAL-NAMES.
          CURRENCY SIGN IS literal
          DECIMAL-POINT IS COMMA.]
      [INPUT-OUTPUT SECTION.]
```

En esta primera parte sólo vamos a ver la **CONFIGURATION SECTION**, la **INPUT-OUTPUT SECTION** la veremos cuando empecemos a tratar con ficheros, mientras tanto la podemos omitir en nuestros programas. Vamos a ver a continuación cada uno de los párrafos anteriores.

4.1. **SOURCE-COMPUTER.**.: Este párrafo nos sirve para comunicar el ordenador donde se escribió el programa, denominado ordenador fuente, es obligatorio en todo programa.

4.2. **OBJECT-COMPUTER..:** En este párrafo especificaremos el ordenador donde se vá a ejecutar el programa, también denominado ordenador objeto, al igual que el anterior es obligatoría su aparición.

4.3. **SPECIAL-NAMES..:** En este párrafo podemos modificar ciertos símbolos de utilización habitual, por ejemplo la inclusión del párrafo 'DECIMAL-POINT IS COMMA' tiene como efecto que en la aparición de decimales se use la coma en vez del punto de notación anglosajona. Este párrafo es opcional.

4.4. **INPUT-OUTPUT SECTION..:** En esta sección se declararán los ficheros que se vayan a utilizar a lo largo del programa y se relacionarán estos con el periférico que los vaya a soportar. Esta parte se estudiará más a fondo cuando empecemos a tratar con ficheros, es una sección opcional que vamos a omitir en nuestros primeros programas.

A continuación vamos a hacer la **ENVIRONMENT DIVISION** de un programa que se ha hecho y ejecutado en un PC y que utiliza la coma en notación decimal.

```
123456789012
      A   B

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.
       SPECIAL-NAMES.
           DECIMAL-POINT IS COMMA.
```

5.- **DATA DIVISION.**

La siguiente división que tenemos es la **DATA DIVISION**, en ella tendremos que declarar tanto los registros de cada uno de los ficheros que vayamos a utilizar, como todas las variables y campos que vayamos a utilizar a lo largo de la ejecución del programa. El formato de esta división es el siguiente.

```
123456789012
      A   B

       DATA DIVISION.
       [FILE SECTION.]
       [WORKING-STORAGE SECTION.]
       [LINKAGE SECTION.]
```

DATA DIVISION

Las tres secciones de esta división son opcionales, en esta primera parte vamos a ver a fondo la **WORKING-STORAGE SECTION**, dejando para capítulos posteriores las otras dos secciones. A continuación vamos a ver el cometido de cada una de ellas.

5.1. **FILE SECTION**..: En esta sección, que estudiaremos más a fondo posteriormente, habremos de declarar los registros de cada uno de los ficheros, estos ficheros deben de haberse declarado con anterioridad en la **INPUT-OUTPUT SECTION**. Estas dos partes las veremos cuando tratemos con ficheros.

5.2. **WORKING-STORAGE SECTION**..: En esta sección se han de declarar los campos y variables que vayamos a usar a lo largo del programa, para ello disponemos de las ordenes que tenemos en el siguiente formato.

No nivel nombre-campo [FILLER] [PICTURE]
[VALUE]
[JUSTIFIED]
[REDEFINES]
[USAGE]
[BLANK WHEN ZERO]
[OCCURS]
[SYNCHRONIZED]
[RENAMES]

Números de nivel..: Cada vez que se declara un campo hay que anteponerle un número de nivel, este nos vá a servir para comunicarle al compilador que tipo de campos es. Cuando el campo que se vá a declarar es un campo elemental, que no se vá a dividir en otros varios, se utiliza el nivel 77, cuando el campo se vá a subdividir en otros menores se utilizan los números de nivel 01 a 49, siendo obligatoria la aparición del nivel 01 una vez en el campo que vá a ser subdividido, a continuación se puede usar un número de nivel mayor para las subdivisiones; si se desea hacer una división de un campo ya dividido se usará un número de nivel mayor que el tenga el campo. Además de estos números de nivel podemos utilizar dos más algo diferentes de los anteriores, uno es el nivel 88, en el se declaran posibles valores de un campo que está declarado en los niveles 01 a 49, si en el transcurso del programa este

campo toma uno de los valores declarados se cumplirán una serie de condiciones que en caso contrario serían falsas. Otro es el nivel 66, por medio de él se puede renombrar a una zona de memoria, esta zona puede ser parte de la definición de un campo con divisiones.

Los números de nivel 01 y 77 se han situar siempre al comienzo del margen A, como si de un nombre de párrafo se tratara, los demás niveles se pueden situar en cualquier columna del margen A o B a excepción de la 8.

A continuación del número de nivel habremos de poner el nombre que vá a recibir el campo, este nombre tendrá que cumplir las condiciones expuestas en el punto 2.6.

A continuación vamos a ver cada una de las palabras Cobol expuestas en el anterior formato.

FILLER..: Cuando un campo no se vá a usar a lo largo de la **PROCEDURE**, pero se hace necesaria su declaración, no es preciso darle un nombre, se puede declarar bajo la palabra **FILLER**, al hacerlo el campo se declarará igual que otro más pero no se podrá hacer referencia a él a lo largo del programa.

PICTURE - PIC..: Una vez especificado el número de nivel y el nombre del campo, hay que declarar su tipo y longitud, para ello nos serviremos de esta palabra Cobol y de los siguientes símbolos declarativos.

9.- Se utiliza para campos numéricos.

A.- Se utiliza para campos alfabéticos.

X.- Se utiliza para campos alfanuméricos.

S.- Representa la aparición del signo. Debe ser el primero de los símbolos en el formato.

V.- Representa el punto decimal.

P.- Representa la aparición de un cero, el uso de este símbolo no altera la longitud del campo pero forma parte de él.

Para representar la longitud del campo se puede

repetir el símbolo o ponerr a continuación y entre paréntesis la longitud. Los símbolos V y S no pueden aparecer más de una vez en un mismo campo.

Para que veas la utilización de los números de nivel y la declaración de campos a continuación tienes un ejemplo. Es la ficha de datos de una persona, en ella los campos nombre y dirección están subdivididos en otros más pequeños.

```
123456789012
      A   B

      WORKING-STORAGE SECTION.
      01  DATOS.
          03 NOMBRE.
             05 NOM              PIC X(20).
             05 APE              PIC X(20).
          03 DIRECCION.
             05 CALLE            PIC X(30).
             05 NUMERO           PIC 9(03).
          03 TELEFONO            PIC 9(07).
      77  CONTROL                PIC 9(01).
```

El campo DATOS está dividido en NOMBRE, DIRECCION y TELEFONO. El NOMBRE y la DIRECCION a su vez están divididos en subcampos elementales, es en estos donde especificamos el tipo y longitud, el TELEFONO es un campo elemental, y por último y fuera del grupo DATOS tenemos un campo llamado CONTROL a nivel 77, es un campo elemental sin ser división de otro.

Haz la definición de un campo llamado DATOS, este tendrá las divisiones LOCALIDAD, que estará dividido en POBLACION y PROVINCIA, y FECHA que estará dividido en DIA, MES y AÑO.

Los formatos para PICTURE vistos anteriormente son los usados para declarar los campos y operar con ellos, pero no son los más idóneos para imprimir su contenido, para este fin podremos declarar otros campos, llamados campos de edición, con los siguiente símbolos.

$.- Representa la aparición del signo dolar delante de cantidades numéricas. Este símbolo se puede cambiar por medio de la cláusula CURRENCY SIGN en el párrafo SPECIAL-NAMES.

..- El punto representa la aparición del punto decimal, si se desea utilizar la coma habrá que incluir la cláusula **DECIMAL-POINT IS COMMA** en el párrafo **SPECIAL-NAMES**.

Z.- Representa la aparición de un espacio en blanco si el valor del dígito que ocupa dentro del campo es cero.

,.- La coma representa la aparición de la coma de los miles. Si se ha incluido en el párrafo **SPECIAL-NAMES** la cláusula **DECIMAL-POINT IS COMMA**, esta coma cambiará por un punto.

*.- La aparición del asterisco al principio del formato tendrá como resultado que los ceros no significativos de la cantidad sean cambiados por este signo.

0.- Representa la aparición del caracter cero.

B.- Representa la aparición de un espacio.

- y +.- Representan la aparición del signo correspondiente.

Con los campos declarados para edición no se puede hacer ningún tipo de operación a excepción de imprimirlos.

VALUE..: Esta palabra Cobol puede acompañar a la definición de un campo para dotar a este de un valor inicial. A continuación de **VALUE** pondremos el valor que deseamos darle, para esto podremos usar constantes literales o numéricas y las constantes figurativas vistas en el punto 2.9. La utilización de esta palabra Cobol es opcional.

Ejemplo:

```
123456789012
      A   B

       77  LINEA1              PIC X(80) VALUE SPACES.
       77  LINEA2              PIC X(80) VALUE ALL "-".
```

En este ejemplo tenemos un campo llamado LINEA1 que tiene 80 espacios y otro llamado LINEA2 que tiene 80 guiones.

JUSTIFIED RIGHT - JUST RIGHT..: Cuando se dota de un valor a un campo, este normalmente se ajusta a la izquierda, si queremos que tal ajuste se haga a la derecha podremos usar esta calusula que es opcional. .

Ejemplo:

```
123456789012
      A   B

       77  NOMBRE          PIC X(10) VALUE "PEPE".
```

Si imprimiésemos este campo obtendríamos lo siguiente.

PEPE******

Si lo declaramos de la siguiente forma:

```
       77  NOMBRE          PIC X(10) VALUE "PEPE" JUST RIGHT
```

Obtendremos:

******PEPE

REDEFINES..: Esta cláusula se utiliza para dar un nombre y formato distinto a un mismo area de memoria. La palabra **REDEFINES** ha de ir a continuación del nombre del campo, y tras ella el nombre del campo del que se desea redefinir. Los dos campos, redefinidor y redefinido, han de estar al mismo número de nivel y han de estar uno tras otro en la declaración de campos.

Ejemplo:

```
123456789012
      A   B

       01  FECHA.
           03  DIA                    PIC 99.
           03  MES                    PIC 99.
           03  ANO                    PIC 99.
       01  FECHA1 REDEFINES FECHA     PIC X(06).
```

En este ejemplo el campo FECHA y FECHA1 utilizan el mismo area de memoria pero con nombres y formatos diferentes, de tal forma que si nosotros damos un valor a FECHA automáticamente pasará a FECHA1 y viceversa.

USAGE..: Con esta cláusula podemos manipular la forma en que se vá a almacenar en la memoria un campo numérico, tenemos las siguientes, posibilidades.

COMPUTATIONAL - COMP..: Almacena el valor en binario puro, tiene la ventaja de ser más rápido en la realización de operaciones pero no se puede imprimir.

COMP1 - COMP3 - COMP6..: Son diferentes grados de compactación numérica, ningún campo declarado con esta opción puede imprimirse.

DISPLAY..: Ocupa un byte por cada dígito guardándolo en representación ASCII. Este modo es el tomado por defecto.

INDEX..: Se utiliza para declarar un campo que vá a actuar como índice de una tabla, no hay que definir su tipo ni longitud.

BLANK WHEN ZERO..: Este cláusula se utiliza en la declaración de campos numéricos, su efecto es el llenado del campo con espacios cuando su valor sea cero.

OCCURS..: Mediante esta cláusula se pueden declarar tablas. Una tabla es un conjunto de elementos con un mismo tipo y longitud, se denominan por el mismo nombre y se diferencian por un subíndice. A continuación vamos a ver los distintos formatos posibles.

OCCURS nn TIMES..: Donde nn es el número de elementos que ha de tener la tabla. Este formato es el más simple, creará una tabla con el nombre expuesto y el número de elementos indicado. La cláusula **OCCURS** no se puede usar a nivel 77 ni 01.

Ejemplo:

```
123456789012
      A   B

      01  DATOS.
          03  NOMBRES    PIC X(10) OCCURS 50 TIMES.
```

En el ejemplo anterior creamos una tabla llamada NOMBRES con 50 elementos, a cada uno de estos nos podremos referir poniendo a continuación y entre paréntesis el número de elemento. La palabra **TIMES** es opcional.

OCCURS nn1 TO nn2 DEPENDING ON campo..: En este segundo formato se crea una tabla con un número de elementos variable que puede ir desde *nn1* como mínimo hasta nn2 como máximo dependiendo siempre del valor que tenga la variable **campo**, esta deber ser numérica y ha de estar declarada antes que la tabla.

Ejemplo:

```
123456789012
      A   B

      77  NUMERO                PIC 99.
      01  DATOS.
          03  NOMBRES           PIC X(10) OCCURS
              10 TO 100 DEPENDING ON NUMERO.
```

En el ejemplo anterior hemos creado una tabla llamada NOMBRES que vá a tener 10 elementos como mínimo y 100 como máximo dependiendo del valor que tenga la variable NUMERO.

OCCURS nn1 TO nn2 DEPENDING ON campo INDEXED BY var..: En este tercer y último formato hay una parte nueva **'INDEXED BY var'**, el efecto de esta partícula es el asignarle un índice a la tabla, esta variable no se ha de declarar como las demás variables, ya que al declararse como índice de una tabla el ordenador la toma en formato interno. Esta variable no se puede tratar como las demás, dispone de una serie de ordenes reservadas que veremos en la **PROCEDURE DIVISION.**

SYNCHRONIZED - SYNC LEFT - RIGHT..: Esta cláusula se utiliza generalmente para asegurar la alineación de los campos clave en los ficheros, tanto directos como indexados.

Ejemplo:

```
123456789012
      A   B

      77  NOM           PIC X(10) SYNC LEFT.
```

En el anterior ejemplo se declara un campo elemental con 10 posiciones alfanuméricas ajustando a la izquierda. Por defecto los campos se ajustan a la

izquierda, si se quiere hacer a la derecha habrá que sustituir 'LEFT' por 'RIGHT'.

RENAMES..: Esta palabra reservada Cobol nos permite redefinir un area de memoria sin cambiar su tipo ni longitud, el area a renombrar pueden ser uno o varios campos subdivididos, nunca campos a nivel 01 ó 77. El campo que vá a acoger a la cláusula **RENAMES** ha de estar a nivel 66 y al final del grupo al que vá a pertenecer.

Ejemplo:

```
123456789012
      A   B

      01  DATOS.
          03  NOMBRE          PIC X(15).
          03  APELLI          PIC X(20).
          03  DIRECC          PIC X(30).
          66  DAT RENAMES NOMBRE THRU APELLI.
```

En este ejemplo se tiene un grupo llamado DATOS con los campos NOMBRE, APELLI y DIRECC, al final del grupo y a nivel 66 renombramos con el nombre de DAT los campos comprendidos entre NOMBRE y APELLI, que en este caso son ellos mismos solamente.

UTILIZACION DEL COBOL

6.- UTILIZACION DEL COBOL.

Hasta ahora hemos vistos todas las declaraciones que necesita un programa Cobol, a partir de aquí vamos a entrar en la **PROCEDURE DIVISION**, en ella utilizaremos todo lo visto ahora por medio de las instrucciones que iremos viendo. A lo largo de esta parte irán apareciendo programas ejemplo de cada una de las instrucciones explicadas, para hacerlos tienes que usar un editor de textos y una vez escrito compilarlo, el compilador es un programa que has de tener en el disco y que se llama **RM**, a continuación vamos a ver como se compila.

6.1. LA COMPILACION.

RM nomprog [O] [L] [T] [X] [K] [P] [F=nn]

En este ejemplo llamamos al programa compilador, trás él

indicamos el nombre del programa que deseamos compilar, y a continuación una serie de parámetros que son opcionales y que vamos a ver a continuación.

[O[=disco]]..: Con esta opción le indicamos al programa compilador que genere un fichero objeto, por defecto el compilador siempre lo crea por lo cual no es necesario especificárselo. Si deseamos que el fichero objeto no se deposite en el disco por defecto sino en otra unidad, trás la opción 'O' pondremos el signo '=' y el nombre del disco seguido de ':'.

[L[=disco]]..: Con esta opción indicamos al compilador que al mismo tiempo que compila genere un listado de errores, si indicamos solamente 'L' tal listado se depositará en la unidad por defecto, si deseamos que se haga en otra unidad hay que especificárselo igual que en la opción 'O'. El listado de errores recibe el mismo nombre que el programa fuente y tiene la extensión 'LST'.

[T]..: Con esta opción le indicamos al compilador que simultáneamente a la compilación liste por pantalla el resultado de esta.

[X]..: Con esta opción indicamos al compilador que al final del listado de compilación o de errores si lo hay genere un listado con las referencias de todas las variables utilizadas.

[K]..: La utilización de esta opción tiene como resultado la no aparición en pantalla de los habituales mensajes del compilador.

[P]..: La inclusión de esta opción en la compilación tiene como efecto que simultáneamente a la compilación se genere un listado de errores por la impresora conectada al sistema.

[F=nn]..: Esta opción se utiliza al compilar cuando el ejecutor de Cobol no es de la misma versión que el compilador, la versión del ejecutor se puede ver tecleando 'RUN" y return. Si el ejecutor es de la versión 2.0B o 2.0C1 no hace falta comunicarle nada al compilador, pero si el ejecutor es de la versión 1.5F y el compilador de la versión 2.0C1 o 2.0B hay que especificar al compilar 'F=11'.

También a lo largo de la **PROCEDURE DIVISION** iremos utilizando una serie de párrafos creados por nosotros mismos, inmediatamente despues del nombre de la división tiene que haber un párrafo, a estos párrafos se les llama habitualmente etiquetas, siempre que creemos una etiqueta hemos de darle un nombre más o menos lógico y siempre ajustándonos a las reglas expuestas anteriormente.

7.- INSTRUCCIONES BASICAS.

Vamos a comenzar a ver cada una de las instrucciones que vamos a utilizar en esta división, de principio vamos a ver unas instrucciones de inicio con las cuales podremos comenzar a hacer pruebas.

7.1. LA INSTRUCCION DISPLAY.

DISPLAY..: Con esta palabra reservada Cobol podremos hacer impresiones por pantalla, el formato es el que sigue.

DISPLAY arg..: El argumento a imprimir por **DISPLAY** puede ser una constante o una variable, pero nunca una expresión. Si se imprime una constante alfabética y esta sobrepasa la columna 72 tendremos que hacer uso del guión como continuación de línea tal y como se explicó en el capítulo correspondiente.

Para ver funcionar esta instrucción vamos a hacer un pequeño programa que imprima una frase por pantalla, para ello coge tu editor de textos, abre un archivo con el nombre 'PRUEBA1' y teclea lo siguiente:

```
123456789012
      A  B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA1.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
```

```
        PROCEDURE DIVISION.

        COMIENZO.

            DISPLAY "HOLA BUENAS TARDES".
```

Una vez que lo hallas escrito grábalo y compílalo de la siguiente forma:

RM PRUEBA1 T

Esto hará que se compile el programa y al mismo tiempo se vaya listando por la pantalla el resultado de la compilación, si todo ha ido bien al final el compilador te debe de dar el siguiente mensaje.

```
READ ONLY BYTE SIZE =          >0056

READ/WRITE BYTE SIZE =         >0006

OVERLAY SEGMENT BYTE SIZE = >0000

TOTAL BYTE SIZE =              >005C

   0 ERRORS

   0 WARNINGS
Compilation complete:     0 Errors,     0 Warnings.
```

Si por el contrario te ha devuelto algún error revisa el programa con el que aquí tienes, corrige el error que tengas y vuelve a compilarlo hasta que devuelva el anterior mensaje. Una vez lo tengamos bien lo vamos a ejecutar, para ello tecleamos lo siguiente:

RUN PRUEBA1 K

Ponemos la opción 'K' para que se supriman los mensajes habituales del ejecutor, cuando lo ejecutes en la última línea ha de aparecer el saludo. Prueba a ejecutar el programa *sin* ponerle la opción 'K', ¿que ocurre?.

Además de constantes literales como hemos hecho en este nuestro primer programa, se puede imprimir cualquier variable declarada en la **DATA DIVISION**, para verlo vamos a teclear el siguiente programa, coge tu editor y ábrelo con el nombre de 'PRUEBA2.CBL'.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA2.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NOMBRE      PIC X(20) VALUE "HOLA ME LLAMO PC".

       PROCEDURE DIVISION.

       COMIENZO.

           DISPLAY NOMBRE.
```

Compílalo igual que antes y ejecútalo, verás aparecer en la última línea de la pantalla el saludo que le asignamos en la **WORKING-STORAGE** a la variable 'NOMBRE'.

Las dos impresiones que hemos hecho han aparecido en la última línea de la pantalla, y si no especificamos la opción 'K' al ejecutar no se vé el mensaje, para solucionar esto vamos a ver dos palabras Cobol que pueden acompañar a **DISPLAY** para posicionarse en cualquier punto de la pantalla.

LINE..: Por medio de ella podemos especificar la línea en la que se vá a imprimir, las líneas están numeradas de 1 a 25 comenzando por el margen superior de la pantalla. El número de línea se puede sustituir por una variable.

POSITION..: Por medio de ella podemos especificar la columna en la que se vá a imprimir, las columnas están numeradas de 1 a 80 comenzando por el margen izquierdo de la pantalla. Al igual que la línea el número de columna se puede especificar por medio de una variable.

Estas dos opciones de la instrucción **DISPLAY** han de ir tras el argumento que se quiere imprimir. Para verlo abre en tu editor un archivo con el nombre 'PRUEBA3.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA3.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  SALUDO          PIC X(21) VALUE "HOLA SOY TU ORDENADOR".
       77  DESPEDIDA       PIC X(14) VALUE "HASTA OTRA VEZ".
       77  NUMERO          PIC 9(04) VALUE 1250.

       PROCEDURE DIVISION.

       COMIENZO.

           DISPLAY SALUDO LINE 8 POSITION 10.
           DISPLAY DESPEDIDA LINE 10 POSITION 10.
           DISPLAY NUMERO LINE 12 POSITION 10.
```

Compilálo y ejecútalo, no hace falta que pongas la opción 'K' al ejecutar, verás como cada una de las impresiones se hace en la línea especificada por nosotros.

7.2. LA INSTRUCCION MOVE.

Vamos a ver a continuación otra instrucción básica que utilizaremos muy a menudo en nuestros programas.

MOVE..: Con esta instrucción podremos mover un determinado valor a una variable, su formato es el siguiente.

MOVE valor TO var..: Donde 'valor' puede ser una constante u otra variable, y 'var' la variable que vá a acoger al valor. El valor nunca podrá ser una expresión, si que puede ser cualquiera de las constantes figurativas vistas anteriormente.

Para ver el funcionamiento de esta instrucción vamos a teclear el siguiente programa, abre con tu editor un programa con el nombre 'PRUEBA4.CBL' y teclea:

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA4.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  LETRAS          PIC  X(15) VALUE SPACES.
       77  NUMEROS         PIC  9(08) VALUE ZEROS.

       PROCEDURE DIVISION.

       COMIENZO.

           MOVE "CADENA LITERAL" TO LETRAS.
           DISPLAY LETRAS LINE 6 POSITION 20.
           MOVE 125000 TO NUMEROS.
           DISPLAY NUMEROS LINE 7 POSITION 20.

           DISPLAY "USO DE LA CONSTANTE FIGURATIVA 'HIGH-VALUES'"
                   LINE 9 POSITION 30.
           MOVE HIGH-VALUES TO LETRAS.
           DISPLAY LETRAS LINE 10 POSITION 20.
           MOVE HIGH-VALUES TO NUMEROS.
           DISPLAY NUMEROS LINE 11 POSITION 20.

           DISPLAY "USO DE LA CONSTANTE FIGURATIVA 'LOW-VALUES'"
                  LINE 13 POSITION 30.
           MOVE LOW-VALUE TO LETRAS.
           DISPLAY LETRAS LINE 14 POSITION 20.
           MOVE LOW-VALUE TO NUMEROS.
           DISPLAY NUMEROS LINE 15 POSITION 20.
```

Tras teclearlo compílalo y ejecútalo, podrás ver el valor que se ha movido en cada momento a cada una de las variables.

7.3. LA INSTRUCCION ACCEPT.

ACCEPT..: Por medio de esta instrucción podremos pedir datos por teclado, para ello haremos uso del siguiente formato.

ACCEPT var..: Donde 'var' es la variable que vá a acoger al valor y que ha de estar declarada con anterioridad. Con esta instrucción podemos usar las opciones **LINE** y **POSITION** vistas en la instrucción **DISPLAY** teniendo el mismo efecto. Al pedir un campo el programa se detendrá y esperará hasta que le facilitemos el dato, para terminar de dárselo hemos de pulsar la tecla return, si el dato que deseamos darle excede de la longitud de la variable que lo vá a acoger automáticamente cuando se llene dicha variable el programa continuará sin necesidad de pulsar la tecla return.

Para ver esta instrucción en acción abre un archivo en tu editor con el nombre 'PRUEBA5.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA5.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NOMBRE                  PIC X(15).

       PROCEDURE DIVISION.
       COMIENZO.

           DISPLAY "¿COMO TE LLAMAS?..:" LINE 12 POSITION 15.
           ACCEPT NOMBRE LINE 12 POSITION 36.
           DISPLAY "HOLA" LINE 14 POSITION 30.
           DISPLAY NOMBRE LINE 14 POSITION 35.
```

Compílalo como de costumbre y ejecútalo, verás como el ordenador solita tu nombre, si tu nombre no coge en los 15 caracteres declarados cuando llegues a estos automáticamente el programa seguirá, si por el contrario es más corto al terminar de escribirlo tendrás que pulsar la tecla return, una vez hecho esto el ordenador te saludará.

Vamos a hacer otro programa, en él vamos a tener una ficha con los datos de una persona, esos datos se van a pedir por teclado, una vez pedidos los distintos datos hemos de imprimir el

nombre y los apellidos en una línea y la dirección y población en otra, para lo cual utilizaremos el nivel 66 y la cláusula **RENAMES** en la DATA DIVISION.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA6.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  DATOS.
           03  NOMBRE              PIC X(15).
           03  APELLIDOS           PIC X(15).
           03  DIRECCION           PIC X(20).
           03  POBLACION           PIC X(20).
           03  PROVINCIA           PIC X(20).
           66  DAT1 RENAMES NOMBRE THRU APELLIDOS.
           66  DAT2 RENAMES DIRECCION THRU POBLACION.

       PROCEDURE DIVISION.
       COMIENZO.

           DISPLAY "NOMBRE....:" LINE  8 POSITION 20.
           DISPLAY "APELLIDOS.:" LINE 10 POSITION 20.
           DISPLAY "DIRECCION.:" LINE 12 POSITION 20.
           DISPLAY "POBLACION.:" LINE 14 POSITION 20.
           DISPLAY "PROVINCIA.:" LINE 16 POSITION 20.

      * PETICION DE LOS CAMPOS.

           ACCEPT NOMBRE LINE  8 POSITION 31.
           ACCEPT APELLIDOS LINE 10 POSITION 31.
           ACCEPT DIRECCION LINE 12 POSITION 31.
           ACCEPT POBLACION LINE 14 POSITION 31.
           ACCEPT PROVINCIA LINE 16 POSITION 31.

      * IMPRESION DEL NOMBRE COMPLETO Y LA DIRECCION CON LA POBLACION.

           DISPLAY DAT1 LINE 18 POSITION 15.
           DISPLAY DAT2 LINE 20 POSITION 15.
```

Aquí además de lo que hemos visto hemos utilizado el asterisco para incluir una serie de comentarios, estos nos aclaran las distintas partes de las que se compone el programa.

7.4. LA INSTRUCCION GO TO.

GO TO..: Esta instrucción nos servirá para desplazarnos de una parte a otra del programa, el formato es el que sigue.

GO TO nometq..: Donde 'nometq' es el nombre de la etiqueta a donde queremos desplazarnos y que ha de cumplir las reglas expuestas en el capítulo correspondiente.

Utilizando esta instruccion vamos a hacer un programa que pida nuestro nombre y que lo imprima indefinidamente, abre en tu editor un archivo con el nombre 'PRUEBA7.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA7.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NOMBRE                  PIC X(15).

       PROCEDURE DIVISION.
       COMIENZO.

           DISPLAY "NOMBRE..:" LINE 6 POSITION 20.
           ACCEPT NOMBRE LINE 6 POSITION 30.

       IMPRIME.

           DISPLAY NOMBRE.
           GO TO IMPRIME.
```

Cuando lo compiles y ejecutes podrás ver como trás darle tu

nombre lo imprimirá indefinidamente, para detenerlo tendrás que pulsar la combinación de teclas Ctrl-Break.

7.5. LA INSTRUCCION IF.

IF..: Por medio de esta instrucción el programa es capaz de tomar decisiones controladas por nosotros, el formato es el siguiente.

IF cond sent..: Donde 'cond' es la condición que se ha de cumplir y 'sent' es la sentencia o sentencias que se ejecutarán si esa condición es cierta, si es falsa el programa seguirá su curso normal.

Para ver en funcionamiento esta nueva instrucción vamos a hacer un programa al que dándole nuestra edad nos diga si somos mayores o menores de edad, además vamos a hacer que cuando la edad sea 0 termine el programa. Abre en tu editor un archivo con el nombre 'PRUEBA8.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA8.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  EDAD                    PIC 99.

      PROCEDURE DIVISION.
      COMIENZO.
          DISPLAY "DAME TU EDAD" LINE 12 POSITION 20.
          ACCEPT EDAD LINE 12 POSITION 33.
          IF EDAD = 0 GO TO SALIR.
          IF EDAD < 18 DISPLAY "ERES MENOR DE EDAD" LINE 14
                POSITION 20, GO TO COMIENZO.
          DISPLAY "ERES MAYOR DE EDAD" LINE 14 POSITION 20.
          GO TO COMIENZO.

      SALIR.
```

Una vez lo hallas tecleado el programa compílalo y ejecútalo, verás como te solicita una edad y según esta te dirá si eres mayor o menor de edad, tras esto te seguirá solicitando edades hasta que como edad le des cero, entonces saltará a la etiqueta 'SALIR' que al no haber nada interrumpe el programa.

Vamos a hacer otro programa, en él vamos a tener una tabla con los nombres, direcciones y teléfonos de nuestros amigos, además de estos tres campos en la tabla vamos a tener un cuarto que se vá a llamar 'ESTADO', en este pondremos una 'A' si el registro está dado de alta o una 'B' si está dado de baja, de esta forma en cualquier momento podremos saber si el número de registro que nos piden ya existe o no, si existe el proceso a seguir sería imprimirlo por pantalla y preguntar si se quiere dar de baja, si es así ponemos una 'B' en el campo 'ESTADO' correspondiente y queda dado de baja, si por el contrario el número de registro que nos solicitan no está dado de alta tendremos que pedir los tres campos correspondientes a ese registro y poner una 'A' en el campo 'ESTADO' correspondiente. Despues de esta pequeña explicación de la estructura que ha de seguir el programa vamos a ponerlo en práctica, abre en tu editor un archivo con el nombre 'PRUEBA9.CBL' y teclea lo que sigue.

```
123456789012
      A   B

      IDENTIFICATION DIVISION.
      PROGRAM-ID. PRUEBA9.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SOURCE-COMPUTER. PC.
      OBJECT-COMPUTER. PC.

      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01  TABLAS.
          03  REGISTRO OCCURS 100.
              05  NOMBRE                  PIC  X(30).
              05  DIRECC                  PIC  X(30).
              05  TELEFO                  PIC  9(13).
              05  ESTADO                  PIC  X.
      77  CLAVE                           PIC  99.
      77  BORRA                           PIC  X(40) VALUE SPACES.
      77  SINO                            PIC  X.
      PROCEDURE DIVISION.
      COMIENZO.
```

```
        DISPLAY "CODIGO.....: [  ] 0=SALIR" LINE  6 POSITION 20.
        DISPLAY "NOMBRE.....: [                                  ]"
                LINE  8 POSITION 20.
        DISPLAY "DIRECCION..: [                                  ]"
                LINE  9 POSITION 20.
        DISPLAY "TELEFONO...: [              ]" LINE 10 POSITION 20.
        DISPLAY BORRA LINE 14 POSITION 10.

    PIDECLAVE.

        ACCEPT CLAVE LINE 6 POSITION 33 NO BEEP.
        IF CLAVE = 0 GO TO SALIR.
        IF ESTADO(CLAVE) = "A" GO TO EXISTE.
        ACCEPT NOMBRE(CLAVE) LINE  8 POSITION 33.
        ACCEPT DIRECC(CLAVE) LINE  9 POSITION 33.
        ACCEPT TELEFO(CLAVE) LINE 10 POSITION 33.
        MOVE "A" TO ESTADO(CLAVE).
        GO TO COMIENZO.

    EXISTE.

        DISPLAY NOMBRE(CLAVE) LINE  8 POSITION 33.
        DISPLAY DIRECC(CLAVE) LINE  9 POSITION 33.
        DISPLAY TELEFO(CLAVE) LINE 10 POSITION 33.
        DISPLAY "LO DOY DE BAJA?..:" LINE 14 POSITION 10.
        ACCEPT SINO LINE 14 POSITION 28 NO BEEP.
        IF SINO = "S" MOVE "B" TO ESTADO(CLAVE).
        GO TO COMIENZO.

    SALIR.
```

Una vez lo hallas tecleado compílalo y ejecútalo, te pedirá una clave, esta ha de ser un número de dos dígitos como máximo, al mismo tiempo te avisa de que si quieres salir debes teclear la clave 0; si la clave tecleada no existe te pedirá los campos y lo memorizará en la tabla, si por el contrario ya existía lo imprimirá y nos dirá si queremos darlo de baja, a lo que nosotros podremos contestar 'S' o 'N'.

7.6. LA INSTRUCCION COMPUTE.

COMPUTE..: Con esta palabra reservada Cobol podremos hacer cálculos en nuestros programas, el formato es el siguiente.

COMPUTE var = oper..: Donde 'var' es la variable que vá a acoger el valor y 'oper' la operación a realizar, en ella podremos usar tanto constantes como variables además de los siguientes signos.

+.- Mediante este signo indicamos que la operación a realizar es la suma.

-.- Mediante este signo indicamos que la operación a relizar es la resta.

*.- Mediante este signo indicamos que la operación a relizar es la multiplicación.

/.- Mediante este signo indicamos que la operación a relizar es la división.

**.- Mediante la aparición de dos asteriscos sin espacio se indica la operación de potenciación.

Vamos a probar esta nueva orden Cobol, para ello vamos a hacer un programa que nos pida diez nombres y los meta en una tabla y despues los ordena y nos los devuelva. Abre en tu editor un archivo con el nombre 'PRUEBA10.CBL' y teclea lo siguiente.

```
123456789012
       A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA10.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  VECES                      PIC  9(02).
       77  NUMERO                     PIC  9(02).
       77  NUMAS1                     PIC  9(02).
       77  BORRA                      PIC  X(15) VALUE SPACES.
       77  AUXIL                      PIC  X(15).
       01  TABLA.
           03  NOMBRE                 PIC  X(15) OCCURS 10 TIMES.

       PROCEDURE DIVISION.
       COMIENZO.

      * PONEMOS A CERO LAS VARIABLES NUMERO Y VECES.

           MOVE 0 TO NUMERO, VECES.
```

```
       PIDENOMBRES.

      * SE SUMA UNO A NUMERO QUE VA A SER EL QUE VAYA CONTANDO
      * EL NUMERO DE NOMBRES QUE LLEVAMOS PEDIDOS.

           COMPUTE NUMERO = NUMERO + 1.
           DISPLAY "DAME EL NOMBRE " LINE 12 POSITION 20.
           DISPLAY NUMERO LINE 12 POSITION 35.
           DISPLAY BORRA LINE 13 POSITION 20.

      * PEDIMOS EL NOMBRE Y PREGUNTAMOS SI YA HEMOS PEDIDO DIEZ
      * EN CASO DE NO SER ASI VOLVEMOS A SEGUIR PIDIENDO.

           ACCEPT NOMBRE(NUMERO) LINE 13 POSITION 20.
           IF NUMERO < 10 GO TO PIDENOMBRES.

       ORDENACION1.

      * PRINCIPIO DE LA ORDENACION
      * LA VARIABLE VECES CONTARA EL BUCLE EXTERIOR QUE SE HABRA
      * DE EJECUTAR DIEZ VECES, ESTE BUCLE TERMINA EN FINORDENACION

           COMPUTE VECES = VECES + 1.
           MOVE 0 TO NUMERO.

       ORDENACION2.

      * SUMAMOS 1 A NUMERO QUE VA A IR COMPARANDO LOS NOMBRES
      * ESTE BUCLE SE EJECUTARA 9 VECES PARA COMPARAR LOS DIEZ
      * NOMBRES, NUMAS1 LLEVA EL NOMBRE QUE SE HABRA DE COMPARAR
      * CON EL NOMBRE DE INDICE NUMERO.

           COMPUTE NUMERO = NUMERO + 1.
           COMPUTE NUMAS1 = NUMERO + 1.

      * SI EL NOMBRE DE INDICE NUMERO ES MAYOR QUE EL NOMBRE DE
      * INDICE NUMAS1 QUE ES NUMERO MAS 1, INTERCAMBIAMOS SUS
      * PUESTOS PARA ORDENARLOS, PARA ELLO NOS AYUDAMOS DE LA
      * VARIABLE AUXILIAR AUXIL.

           IF NOMBRE(NUMERO) > NOMBRE(NUMAS1)
                    MOVE NOMBRE(NUMERO) TO AUXIL,
                    MOVE NOMBRE(NUMAS1) TO NOMBRE(NUMERO),
                    MOVE AUXIL TO NOMBRE(NUMAS1).
      * EL BUCLE INTERIOR SE EJECUTARA 9 VECES.

           IF NUMERO < 9 GO TO ORDENACION2.
```

```
       FINORDENACION.

      * EL BUCLE EXTERIOR SE EJECUTAR 10 VECES.

           IF VECES < 10 GO TO ORDENACION1.
           MOVE 0 TO NUMERO.
           MOVE 8 TO VECES.

       IMPRIME.

      * UNA VEZ ORDENADOS SE IMPRIMEN LOS DIEZ NOMBRES, SE
      * UTILIZA LA VARIABLE VECES COMO LINEA PARA IMPRIMIRLOS
      * TODOS EN LA MISMA COLUMNA UNA LINEA BAJO LA OTRA.

           COMPUTE NUMERO = NUMERO + 1.
           COMPUTE VECES = VECES + 1.
           DISPLAY NOMBRE(NUMERO) LINE VECES POSITION 40.
           IF NUMERO < 10 GO TO IMPRIME.
```

Ejecútalo tras compilarlo, te pedirá diez nombres y te los devolverá ordenados alfabéticamente, el programa lleva incluidos una serie de comentarios para que te sea más fácil entenderlo.

8.- INSTRUCCIONES COBOL.

Hasta aquí hemos visto las instrucciones más básicas que nos hacían falta para poder comenzar a trabajar con el Cobol, a partir de aquí vamos a ir viendo cada una de las instrucciones Cobol por grupos y vamos a ir profundizando en su finalidad.

8.1. LA INSTRUCCION ACCEPT UTILIZACION AVANZADA.

ACCEPT..: La instrucción **ACCEPT** la hemos estado utilizando anteriormente, pero no en su totalidad, además de las opciones de especificar la línea y columna tiene una serie de opciones que vamos a ver a continuación.

HIGH..: La añadidura de esta opción tiene como causa que el dato que se entra por teclado aparezca en pantalla en video alto o brillante. Es el modo tomado por defecto.

LOW..: La inclusión de esta opción hace el dato a entrar por teclado aparezca en pantalla en video bajo.

REVERSE..: Con esta opción podremos hacer que el dato a entrar por teclado aparezca en video inverso.

BLINK..: Si lo que queremos es que el dato entrante aparezca parpadeante podremos usar esta opción.

SIZE..: Con esta opción podremos controlar el número de caracteres que queremos entren en el campo, por defecto se acepta el campo entero, si queremos entrar sólo parte de él incluiremos esta opción y tras ella el número de caracteres que queremos aceptar, este no podrá ser mayor que la longitud declarada de la variable.

NO BEEP..: Cada vez que se pide un dato por teclado habrás notado que se produce un pitido, a veces esto puede ser molesto si son muchos los campos a pedir, para evitarlo podremos incluir esta opción.

OFF..: La inclusión de esta opción en la petición tendrá como resultado que el campo que se pide no se refleje en pantalla.

TAB..: Cada vez que metemos un campo cuando llegamos al tope de su longitud el programa salta automáticamente y sigue su curso, si queremos que se haga obligatoria la pulsación de la tecla return para pasar al siguiente campo podremos incluir esta opción.

ERASE..: Al mismo tiempo que pedimos el campo podemos hacer que se borre la pantalla, si queremos borrar la pantalla completa incluiremos esta opción, si queremos borrar hasta el final de la línea donde estamos pidiendo el campo incluiremos esta misma opción y tras ella la palabra Cobol **EOL**, si queremos borrar hasta el final de la línea donde se pide y todas las líneas inferiores a ella incluiremos la palabra **EOS** tras la opción **ERASE**.

PROMPT..: Con esta opción podremos hacer que en pantalla aparezca la longitud del campo que vamos a pedir llena de guiones, si queremos sustituir este guión por otro signo pondremos a continuación de la palabra Cobol dicho signo entrecomillado. La utilización de esta opción es incompatible con la utilización de la opción **UPDATE**.

UPDATE..: La inclusión de esta opción en el formato de petición tiene como resultado que si se pulsa return sin haber escrito la entrada se tome el valor anterior que

tenia el campo, normalmente se usa en conjunto con la opción **ECHO**. La utilización de esta opción es incompatible con la utilización de la opción **PROMPT**.

ECHO..: Esta opción en la petición de un campo tendrá como resultado que en pantalla aparezca el último valor que tuvo esta variable, se usa a menudo en conjunción con la opción **UPDATE**.

CONVERT..: Habrás observado que al pedir un campo numérico por teclado se pueden teclear caracteres no numéricos, esto en RM-COBOL no se puede solucionar pero la inclusión de la opción **CONVERT** hará que despues de haber tecleado la entrada se explore el campo y se excluyan los caracteres no numéricos quedándose solamente los dígitos numéricos.

FROM..DATE - DAY - TIME..: Con este formato podremos hacer un petición de datos pero no por teclado sino al sistema, la inclusión de **DATE** no devolverá la fecha completa, **DAY** el día del mes y **TIME** la hora que es.

ON EXCEPTION campo sent..: Con esta opción podremos indicar al Cobol que haga una cierta acción si a lo largo de la petición del campo se pulsa una tecla especifica, esta tecla estará declarada a nivel 88 como parte de la variable **campo** y se le habrá dado como valor el código ASCII de la tecla que se quiere cause la excepción. No es aconsejable usar esta opción junto con la opción **CONVERT** ya que al hacerlo suele dar problemas y la pulsación de cualquier tecla no alfanumérica dá como resultado la excepción que debería de producir una sola tecla.

Para ver en funcionamiento cada una de las opciones posibles de **ACCEPT** vamos a hacer varios programas ejemplo, en el primero vamos a probar los atributos de pantalla, es decir borrar, pedir un campo en video alto y bajo, en video inverso o parpadeante. Abre en tu editor un archivo con el nombre 'PRUEBA11.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA11.
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. PC.
OBJECT-COMPUTER. PC.

DATA DIVISION.
WORKING-STORAGE SECTION.

77  NOMBRE                     PIC X(15).

PROCEDURE DIVISION.
COMIENZO.

* BORRA LA PANTALLA E IMPRIME UN MENSAJE.

    DISPLAY "HE BORRADO LA PANTALLA" LINE 1 POSITION 30 ERASE.
    DISPLAY "DAME TU NOMBRE" LINE 4 POSITION 10.

* PIDE EL NOMBRE EN VIDEO ALTO.

    ACCEPT NOMBRE LINE 4 POSITION 25 HIGH.
    DISPLAY "ESTO ES VIDEO ALTO" LINE 4 POSITION 45 HIGH.

* PIDE EL NOMBRE EN VIDEO BAJO.

    DISPLAY "DAME TU NOMBRE" LINE 6 POSITION 10.
    ACCEPT NOMBRE LINE 6 POSITION 25 LOW.
    DISPLAY "ESTO ES VIDEO BAJO" LINE 6 POSITION 45 LOW.

* PIDE EL NOMBRE EN VIDEO INVERSO.

    DISPLAY "DAME TU NOMBRE" LINE 8 POSITION 10.
    ACCEPT NOMBRE LINE 8 POSITION 25 REVERSE.
    DISPLAY "ESTO ES VIDEO INVERSO" LINE 8 POSITION 45 REVERSE.

* PIDE EL NOMBRE EN VIDEO PARPADEANTE

    DISPLAY "DAME TU NOMBRE" LINE 10 POSITION 10.
    ACCEPT NOMBRE LINE 10 POSITION 25 BLINK.
    DISPLAY "ESTO ES VIDEO PARPADEANTE" LINE 10 POSITION 45 BLINK.

* IMPRIME UN MENSAJE CON DISTINTOS ATRIBUTOS.

    DISPLAY "PULSA RETURN " LINE 15 POSITION 10 REVERSE.
    DISPLAY "PARA ABANDONAR " LINE 15 POSITION 23 REVERSE HIGH.
    DISPLAY "ESTE PROGRAMA" LINE 15 POSITION 38 REVERSE BLINK.
    ACCEPT NOMBRE.
```

Despues de compilárlo ejecútalo y podrás comprobar los distintos atributos que puedes utilizar, como habrás notado en el programa estos atributos funcionan exactamente igual con la orden **DISPLAY**.

Vamos a hacer otro programa donde vamos a utilizar las opciones **SIZE, NO BEEP, OFF Y TAB**; abre en tu editor un archivo con el nombre de 'PRUEBA12.CBL' y teclea lo siguiente.

```
123456789012
       A   B

        IDENTIFICATION DIVISION.
        PROGRAM-ID. PRUEBA12.

        ENVIRONMENT DIVISION.
        CONFIGURATION SECTION.
        SOURCE-COMPUTER. PC.
        OBJECT-COMPUTER. PC.

        DATA DIVISION.
        WORKING-STORAGE SECTION.
      * EL CAMPO NOMBRE TIENE 20 CARACTERES DE LONGITUD.

        77  NOMBRE                    PIC  X(20).

        PROCEDURE DIVISION.
        COMIENZO.

      * VAMOS A ACEPTAR EL CAMPO NOMBRE EN PARTE, SOLO 10 CARACTERES.

            DISPLAY "DAME UN NOMBRE" LINE 12 POSITION 10.
            DISPLAY "UTILIZACION DE SIZE" LINE 12 POSITION 50 BLINK.
            ACCEPT NOMBRE LINE 12 POSITION 25 SIZE 10.

      * IMPRIMIMOS EL NOMBRE.

            DISPLAY NOMBRE LINE 13 POSITION 50 REVERSE.

      * VAMOS A PEDIR EL NOMBRE EVITANDO EL PITIDO HABITUAL.

            DISPLAY "DAME UN NOMBRE" LINE 15 POSITION 10.
            DISPLAY "UTILIZACION DE NO BEEP" LINE 15 POSITION 50 BLINK.
            ACCEPT NOMBRE LINE 15 POSITION 25 NO BEEP.

      * IMPRIMIMOS EL NOMBRE.
```

```
           DISPLAY NOMBRE LINE 16 POSITION 50 REVERSE.

       * PEDIMOS EL NOMBRE SIN QUE SE REFLEJE POR LA PANTALLA.

           DISPLAY "DAME UN NOMBRE" LINE 18 POSITION 10.
           DISPLAY "UTILIZACION DE OFF" LINE 18 POSITION 50 BLINK.
           ACCEPT NOMBRE LINE 18 POSITION 25 OFF.

       * IMPRIMIMOS EL NOMBRE.

           DISPLAY NOMBRE LINE 19 POSITION 50 REVERSE.

       * PEDIMOS EL NOMBRE TENIENDO QUE PULSAR RETURN AL FINAL.

           DISPLAY "ESCRIBE TODO LO QUE PUEDAS" LINE 21 POSITION 10.
           DISPLAY "UTILIZACION DE TAB" LINE 21 POSITION 50 BLINK.
           ACCEPT NOMBRE LINE 22 POSITION 25 TAB NO BEEP.
```

Tras compilárlo ejecútalo y podrás ver el resultado de cada una de las opciones anteriores, en el programa ván incluidos unos comentarios aclarativos.

Vamos a hacer a continuación otro programa, en él vamos a ver el resultado de la utilización de las opciones **PROMPT**, **ECHO** y **UPDATE**; abre en tu editor un archivo con el nombre de 'PRUEBA13.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA13.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NOMBRE                    PIC  X(20).

       PROCEDURE DIVISION.
       COMIENZO.

      * UTILIZACION DE LA OPCION PROMPT.
```

```
           DISPLAY "OPCION PROMPT" LINE 4 POSITION 50 BLINK.
           DISPLAY "DAME UN NOMBRE" LINE 4 POSITION 5.
           ACCEPT NOMBRE LINE 4 POSITION 20 PROMPT.
           DISPLAY NOMBRE LINE 5 POSITION 50 REVERSE.

      * UTILIZACION DE LA OPCION ECHO.

           DISPLAY "OPCION ECHO" LINE 7 POSITION 50 BLINK.
           DISPLAY "DAME UN NOMBRE" LINE 7 POSITION 5.
           ACCEPT NOMBRE LINE 7 POSITION 20 ECHO.
           DISPLAY NOMBRE LINE 8 POSITION 50 REVERSE.

      * UTILIZACION DE LA OPCION UPDATE.

           DISPLAY "OPCION UPDATE" LINE 10 POSITION 50 BLINK.
           DISPLAY "DAME UN NOMBRE" LINE 10 POSITION 5.
           ACCEPT NOMBRE LINE 10 POSITION 20 UPDATE.
           DISPLAY NOMBRE LINE 11 POSITION 50 REVERSE.

      * UTILIZACION CONJUNTA DE LAS OPCIONES ECHO Y UPDATE.

           DISPLAY "UTILIZACION DE ECHO Y UPDATE JUNTOS"
                    LINE 13 POSITION 25 BLINK.
           DISPLAY "SI PULSAS RETURN DE PRINCIPIO SE TOMARA EL VALOR QU
      -            "E TENGA" LINE 14 POSITION 10 REVERSE.
           DISPLAY "DAME UN NOMBRE" LINE 16 POSITION 4.
           ACCEPT NOMBRE LINE 16 POSITION 20 ECHO UPDATE.
           DISPLAY NOMBRE LINE 17 POSITION 50 REVERSE.
```

Además de las opciones que se ven en este programa también tienes un ejemplo de utilización del guión como continuación de un literal que no coge en una línea, recuerda que la última columna donde se puede escribir algo es en la 72. Prueba repetidas veces con la última parte del programa donde se utilizan conjuntamente **ECHO** y **UPDATE** con estas dos opciones juntas si tecleas parte del campo la otra parte se coge de su contenido anterior y si tecleas return de principio se coge su contenido entero.

Vamos a hacer a continuación otro programa donde vamos a utilizar las opciones **CONVERT** y **FROM**, abre en tu editor un archivo con el nombre de 'PRUEBA14.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA14.
```

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NUMERO                  PIC 9(10).
       01  FECHA.
           03  ANO                 PIC 9(02).
           03  MES                 PIC 9(02).
           03  DIA                 PIC 9(02).

       PROCEDURE DIVISION.
       COMIENZO.

      * VAMOS A COGER LA FECHA DEL SISTEMA E IMPRIMIRLA POR PANTALLA.

           ACCEPT FECHA FROM DATE.
           DISPLAY DIA LINE 12 POSITION 20 ERASE.
           DISPLAY "DEL" LINE 12 POSITION 23.
           DISPLAY MES LINE 12 POSITION 27.
           DISPLAY "DE" LINE 12 POSITION 30.
           DISPLAY ANO LINE 12 POSITION 33.

      * VAMOS A ACEPTAR UN NUMERO CON CONVERT Y A IMPRIMIRLO.

           DISPLAY "DAME UN NUMERO PERO INCLUYE LETRAS"
                   LINE 16 POSITION 20.
           ACCEPT NUMERO LINE 18 POSITION 20 CONVERT NO BEEP.
           DISPLAY NUMERO LINE 18 POSITION 20 ERASE EOL.
```

Cuando lo ejecutes podrás comprobar que te dá la fecha del sistema, la variable que vá a acoger la fecha hay que declarar como lo hemos hecho porque ese es el formato en que dá la fecha el sistema, despues te solicitará un número, dáselo pero incluyendo letras, cuando termines podrás comprobar que imprime el número habiendo excluido los caracteres no numéricos.

Por último vamos hacer otro programa para ver la utilización de la opción **ON EXCEPTION**, para ello abre en tu editor un archivo con el nombre 'PRUEBA15.CBL' y teclea lo que sigue.

```
123456789012
      A   B
```

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA15.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  DATOS.
           03  NOMBRE                  PIC X(15).
           03  DIRECC                  PIC X(20).
           03  POBLAC                  PIC X(20).

      * SE DECLARA LA VARIABLE CON EL CODIGO DE LA TECLA QUE VA A
      * PRODUCIR LA EXCEPCION, ESTA VA A SER LA TECLA ESCAPE.

       01  ARRIBA                      PIC 9(02) COMP.
           88  ESCAPE VALUE 27.

       PROCEDURE DIVISION.
       COMIENZO.

           DISPLAY "NOMBRE....:" LINE 12 POSITION 20 ERASE.
           DISPLAY "DIRECCION.:" LINE 14 POSITION 20.
           DISPLAY "POBLACION.:" LINE 16 POSITION 20.

       PETICIONNOMBRE.

      * SE PIDE EL CAMPO NOMBRE SIN EXCEPCION YA QUE ES EL PRIMERO.

           ACCEPT NOMBRE LINE 12 POSITION 31 PROMPT NO BEEP.

       PETICIONDIRECCION.

      * SE PIDE LA DIRECCION CON EXCEPCION CAUSADA POR LA TECLA ESC.

           ACCEPT DIRECC LINE 14 POSITION 31 PROMPT NO BEEP

                 ON EXCEPTION ARRIBA GO TO PETICIONNOMBRE.

       PETICIONPOBLACION.

      * SE PIDE LA POBLACION IGUAL QUE LA DIRECCION.
```

```
ACCEPT POBLAC LINE 16 POSITION 31 PROMPT NO BEEP
      ON EXCEPTION ARRIBA GO TO PETICIONDIRECCION.
```

Cuando lo ejecutes te pedirá un nombre, una dirección y una población, dale el nombre y la dirección y cuando te pida la población pulsa la tecla escape, púlsala otra vez, ¿ves lo que ocurre?.

Hasta aquí hemos visto todas las posibilidades de la instrucción **ACCEPT**, además de los programas que aquí tienes intenta hacer los tuyos, puedes pensar por ejemplo en hacerte una agenda donde tengas los nombres, direcciones y teléfonos de tus amigos, para ello puedes usar tablas tal y como hicimos en uno de los programas.

8.2. LA INSTRUCCION DISPLAY UTILIZACIÓN AVANZADA.

Todas las opciones de atributos que hemos visto en la instrucción **ACCEPT** funcionan exactamente igual con la instrucción **DISPLAY**, además de ellas hay algunas más, vamos a verlas todas a continuación.

LINE..: Actúa igual que en la instrucción **ACCEPT**.

POSITION..: Actúa igual que en la instrucción **ACCEPT**.

HIGH..: Actúa igual que en la instrucción **ACCEPT**.

LOW..: Actúa igual que en la instrucción **ACCEPT**.

REVERSE..: Actúa igual que en la instrucción **ACCEPT**.

BLINK..: Actúa igual que en la instrucción **ACCEPT**.

SIZE..: Con esta opción podremos imprimir un número determinado de caracteres de un literal o constante figurativa, para ello a continuación de **SIZE** pondremos ese número.

BEEP..: Cuando se hace una impresión no se produce ningún pitido al igual que ocurre con **ACCEPT**, si queremos que se oiga añadiremos a la impresión esta opción.

ERASE - EOL - EOS..: Actúa como en la instrucción **ACCEPT**.

Para ver los atributos utilizados con la instrucción **DISPLAY** vamos a teclear el siguiente programa, abre en tu editor un archivo con el nombre de 'PRUEBA16.CBL' y teclea.

```
123456789012
      A  B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA16.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  TECLA                    PIC X.

       PROCEDURE DIVISION.
       COMIENZO.

           DISPLAY "ACABO DE BORRAR LA PANTALLA" LINE 1 POSITION 20
                                                 ERASE.
           DISPLAY "ESTO ES VIDEO NORMAL" LINE 4 POSITION 10.
           DISPLAY "ESTO ES VIDEO ALTO" LINE 4 POSITION 40 HIGH.
           DISPLAY "ESTO ES VIDEO BAJO" LINE 6 POSITION 10 LOW.
           DISPLAY " ESTO ES VIDEO INVERSO " LINE 6 POSITION 40 REVERSE.
           DISPLAY "ESTO ES VIDEO PARPADEANTE" LINE 8 POSITION 10 BLINK.
           DISPLAY "ESTO ES VIDEO PARPADEANTE E INVERSO"
                   LINE 12 POSITION 25 REVERSE BLINK.
           DISPLAY " PULSA UNA TECLA " LINE 20 POSITION 30 REVERSE.
           ACCEPT TECLA OFF NO BEEP.
```

8.3. INSTRUCCIONES DE CALCULO.

Vamos a pasar a otro tipo de instrucciones, conocemos una instrucción que nos sirve para hacer distintas operaciones que es **COMPUTE**, además de esta el Cobol tiene una serie de instrucciones aritméticas cada una especializada en una operación determinada y que vamos a ver a continuación.

8.3.1. SUMANDO CON LA INSTRUCCION ADD.

ADD..: Con esta instrucción vamos a poder hacer sumas, para ello haremos uso de los siguientes formatos.

ADD valor TO var..: Donde 'valor' podrá ser una constante o variable numérica y 'var' será la variable a la que se vá a sumar; en este formato, que es el más simple, se sumará 'valor' al contenido actual de 'var' y el resultado quedará en esta última.

ADD valor, valor, valor GIVING var2..: En este segundo formato se sumarán los valores y el resultado se almacenará en 'var2'.

ROUNDED..: Cuando el resultado de la suma se almacena en un variable que tiene un formato insuficiente para acoger a tal resultado, el número se mutila, para evitarlo se utiliza esta opción, su cometido es redondear el resultado al número más próximo dentro de la declaración de la variable si se produce un desbordamiento.

ON SIZE ERROR instr..: Esta opción al igual que la anterior se utiliza para el caso en que se desborde la capacidad de la variable que vá a acoger el resultado de la suma, su cometido es ejecutar la instrucción que a continuación se indique en caso de desbordamiento.

ADD CORR grp1 TO grp2..: Este formato de la instrucción ADD es muy útil, con él podemos sumar grupos de variables, para ello en cada grupo habrá una serie de campos que tendrán el mismo nombre, al ejecutar esta instrucción se cogerán los campos de 'grp1' y se sumarán con los que tengan el mismo nombre en 'grp2'. Los nombres de campos duplicados siempre tienen que ser divisiones de otros, no pueden estar a niveles 01, 77 ni 88; a la hora de referirnos a un campo que tiene duplicación lo haremos de la forma 'var OF grp', donde 'var' es el campo duplicado y 'grp' el nombre del grupo al que pertenece.

Una vez vistos todos los formatos que puede tener la instrucción ADD vamos a hacer un programa que nos ilustre lo visto, para ello abre en tu editor un archivo con el nombre de 'PRUEBA17.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA17.
```

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NUMERO                      PIC 9(05) VALUE 0.
       77  SUMA                        PIC 9(10) VALUE 0.
       77  SUMA1                       PIC 9(05) VALUE 0.
       77  TECLA                       PIC X.

       01  INGRESOS1.
           03  SUELDO                  PIC 9(06).
           03  EXTRAS                  PIC 9(05).
           03  DEVENGOS                PIC 9(05).

       01  INGRESOS2.
           03  SUELDO                  PIC 9(06).
           03  EXTRAS                  PIC 9(05).
           03  DEVENGOS                PIC 9(05).

       PROCEDURE DIVISION.
       COMIENZO.

    * VA A IR PIDIENDO Y SUMANDO NUMEROS HASTA QUE EL NUMERO DADO SEA 0.

           DISPLAY " SUMA DE NUMEROS " LINE 1 POSITION 35 REVERSE ERASE.
           DISPLAY "VE DANDOME NUMEROS, YO LOS IRE SUMANDO"
                   LINE 4 POSITION 25.
           DISPLAY "PARA TERMINAR DAME EL 0" LINE 5 POSITION 25.

      PIDENUMERO.

           DISPLAY "NUMERO..:" LINE 7 POSITION 10.
           DISPLAY SPACES SIZE 6 LINE 7 POSITION 19.
           ACCEPT NUMERO LINE 7 POSITION 19 NO BEEP.
           IF NUMERO = 0 GO TO FINSUMA.
           ADD NUMERO TO SUMA.
           GO TO PIDENUMERO.

      FINSUMA.

           DISPLAY "HA SUMADO " LINE 7 POSITION 10.
           DISPLAY SUMA LINE 7 POSITION 20.
```

```
* VAMOS A IR SUMANDO CANTIDADES GRANDES A UN CAMPO Y REDONDEANDO.

    DISPLAY "DAME CANTIDADES GRANDES PARA SUMAR REDONDEANDO"
            LINE 9 POSITION 20.
    MOVE 0 TO SUMA, NUMERO.

 PIDENUMERO1.

    DISPLAY "NUMERO..:" LINE 11 POSITION 10.
    DISPLAY SPACES SIZE 6 LINE 11 POSITION 19.
    ACCEPT NUMERO LINE 11 POSITION 19 NO BEEP.
    IF NUMERO = 0 GO TO FINPIDENUMERO1.
    ADD NUMERO TO SUMA1 ROUNDED.
    DISPLAY SUMA1 LINE 11 POSITION 40.
    GO TO PIDENUMERO1.

 FINPIDENUMERO1.

* AHORA VAMOS A IR SUMANDO NUMEROS HASTA QUE SE PRODUZCA
* EL DESBORDAMIENTO DEL CAMPO SUMA1.

    DISPLAY "VE DANDOME NUMEROS PARA SUMAR, CUANDO EL CAMPO SE DE
-   "SBORDE TE AVISARE" LINE 13 POSITION 10.
    MOVE 0 TO SUMA1.

 PIDENUMERO2.

    DISPLAY "NUMERO..:" LINE 15 POSITION 10.
    DISPLAY SPACES SIZE 6 LINE 15 POSITION 19.
    ACCEPT NUMERO LINE 15 POSITION 19 NO BEEP.
    ADD NUMERO TO SUMA1 ON SIZE ERROR GO TO FINPIDENUMERO2.
    DISPLAY SUMA1 LINE 15 POSITION 40.
    GO TO PIDENUMERO2.

 FINPIDENUMERO2.

    DISPLAY "EL CAMPO SE HA DESBORDADO" LINE 15 POSITION 47.
    DISPLAY SUMA1 LINE 15 POSITION 19.
    DISPLAY " PULSA UNA TECLA PARA CONTINUAR "
            LINE 19 POSITION 30 REVERSE BLINK.
    ACCEPT TECLA OFF NO BEEP.

* PETICION DE LOS CAMPOS DE LOS GRUPOS INGRESOS1 E INGRESOS2
* Y SUMA DE LOS DOS GRUPOS CON USO DE CORRESPONDING - CORR.

    DISPLAY "TE VOY A PEDIR LAS CANTIDADES CORRESPONDIENTES A LOS
-   " GRUPOS INGRESOS1 E      INGRESOS2, CUANDO LOS TENGA LOS SUMA
-   "RE."
```

```
            LINE 1 POSITION 5 ERASE.
        DISPLAY " INGRESOS1 " LINE 4 POSITION 15 REVERSE.
        DISPLAY " INGRESOS2 " LINE 4 POSITION 40 REVERSE.
        DISPLAY "SUELDO..:" LINE 6 POSITION 15.
        DISPLAY "SUELDO..:" LINE 6 POSITION 40.
        DISPLAY "EXTRAS..:" LINE 7 POSITION 15.
        DISPLAY "EXTRAS..:" LINE 7 POSITION 40.
        DISPLAY "DEVENGOS:" LINE 8 POSITION 15.
        DISPLAY "DEVENGOS:" LINE 8 POSITION 40.
        ACCEPT SUELDO OF INGRESOS1 LINE 6 POSITION 24 NO BEEP.
        ACCEPT EXTRAS OF INGRESOS1 LINE 7 POSITION 24 NO BEEP.
        ACCEPT DEVENGOS OF INGRESOS1 LINE 8 POSITION 24 NO BEEP.
        ACCEPT SUELDO OF INGRESOS2 LINE 6 POSITION 49 NO BEEP.
        ACCEPT EXTRAS OF INGRESOS2 LINE 7 POSITION 49 NO BEEP.
        ACCEPT DEVENGOS OF INGRESOS2 LINE 8 POSITION 49 NO BEEP.

    * SUMA DE LOS DOS GRUPOS.

        ADD CORR INGRESOS1 TO INGRESOS2.

    * IMPRESION DE LOS RESULTADOS.

        DISPLAY " SUMA DE INGRESOS " LINE 11 POSITION 30 REVERSE.
        DISPLAY "SUELDO..:" LINE 13 POSITION 30.
        DISPLAY "EXTRAS..:" LINE 14 POSITION 30.
        DISPLAY "DEVENGOS:" LINE 15 POSITION 30.
        DISPLAY SUELDO OF INGRESOS2 LINE 13 POSITION 39.
        DISPLAY EXTRAS OF INGRESOS2 LINE 14 POSITION 39.
        DISPLAY DEVENGOS OF INGRESOS2 LINE 15 POSITION 39.
        DISPLAY " PULSA UNA TECLA " LINE 19 POSITION 30
                REVERSE BLINK.
        ACCEPT TECLA OFF NO BEEP.
        DISPLAY " " ERASE.
```

El programa igual que los anteriores vá acompañado de una serie de comentarios aclarativos, tras compilárlo ejecútalo y verás en funcionamiento los dintintos formatos de la suma.

8.3.2. RESTANDO CON LA INSTRUCCION SUBTRACT.

SUBTRACT..: Esta instrucción Cobol nos vá a servir para hacer restas, para ello podremos hacer uso de los formatos que a continuación tenemos.

SUBTRACT valor FROM var..: El 'valor' se restará del contenido actual de 'var' y el resultado quedará en esta última; 'valor' puede ser una constante o una variable.

SUBTRACT valor1, valor2,.. FROM var..: En este segundo formato se sumarán los valores a la izquierda de **FROM** y el resultado se restará del contenido de 'var' quedando el resultado en esta última.

SUBTRACT valor1 FROM valor2 GIVING var..: En este formato se restará 'valor1' de 'valor2' y el resultado se almacenará en 'var'.

ROUNDED..: La inclusión de esta opción en el formato de la resta tendrá como resultado el redondeo de la cantidad en caso de desbordamiento del campo que vá a acoger el resultado de la resta.

ON SIZE ERROR instr..: Con esta opción en el formato de la resta en caso de desbordamiento del campo que vá a acoger el resultado de la resta se ejecutará 'instr'.

SUBTRACT CORR grp1 FROM grp2..: En este último formato se restarán los campos que tengan el mismo nombre del grupo 'grp1' con los correspondientes del grupo 'grp2', quedando el resultado en este último.

Para probar los formatos de esta instrucción puedes coger el programa 'PRUEBA17.CBL' y hacer los cambios pertinentes o hacer tú un programa que utilice esta instrucción.

8.3.3. MULTIPLICANDO CON LA INSTRUCCION MULTIPLY.

MULTIPLY..: Esta instrucción nos vá a servir para hacer multiplicaciones, para ello nos podremos servir de los siguientes formatos.

MULTIPLY valor BY var..: En este primer formato se multiplicará 'valor' por el contenido de 'var' quedando el resultado en esta última variable.

MULTIPLY valor1 BY valor2 GIVING var..: El resultado de la utilización de la opción **GIVING** tiene el mismo resultado que en las dos instrucciones anteriores, se multiplicará 'valor1' por 'valor2' y el resultado se almacenará en 'var'.

ROUNDED..: Esta opción funciona de igual forma que en las anteriores instrucciones.

ON SIZE ERROR instr..: Esta opción funciona de igual forma que en las anteriores instrucciones.

Estos son los formatos de la multiplicación, con ellos y lo visto en las anteriores instrucciones no te será difícil hacer un programa que las utilice, por ejemplo puedes hacer un programa al que dándole horas y minutos te devuelva el resultado en segundos, ¿lo haces?.

8.3.4. DIVIDIENDO CON LA INSTRUCCION DIVIDE.

DIVIDE..: Con esta instrucción Cobol podremos efectuar divisiones, para ello haremos uso de los formatos que a continuación tenemos.

DIVIDE valor BY valor GIVING var..: El 'valor' se dividirá por el contenido de 'var' y el resultado se almacenará en esta última variable.

DIVIDE valor INTO var..: La operación se hará en sentido inverso a la anterior, 'var' se dividirá entre 'valor', quedando el resultado en la variable.

DIVIDE valor1 INTO valor2 GIVING var..: Este formato funciona de forma similar a como lo hacía en las anteriores instrucciones.

ROUNDED..: Esta opción funciona de igual forma que en las anteriores instrucciones.

ON SIZE ERROR instr..: Esta opción funciona de igual forma que en las anteriores instrucciones.

DIVIDE valor1 INTO valor2 GIVING var1 REMAINDER var2..: En este último formato se dividirá 'valor1' entre 'valor2' y el resultado quedará en 'var1', el resto, si lo hay, quedará en 'var2'.

8.3.5. LA INSTRUCCION COMPUTE UTILIZACION AVANZADA.

COMPUTE..: Esta instrucción ya la conocemos y la hemos utilizado, además de lo que conocemos sobre ella tenemos los siguientes formatos.

COMPUTE ROUNDED = expr..: La opción ROUNDED en esta instrucción funciona de igual forma que en las anteriores instrucciones.

COMPUTE = expr ON SIZE ERROR instr..: Esta opción funciona de igual forma que lo hace en las anteriores instrucciones.

Aquí terminamos con las instrucciones aritméticas, para ver algunas de ellas en funcionamiento vamos a hacer un nuevo programa, este es una simulación de un generador de facturas de una empresa, en el mismo programa ván incluidos una serie de comentarios aclarativos así que abre en tu editor un archivo con el nombre de 'PRUEBA18.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA18.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  CLIENTE                 PIC X(20).
       77  PRECIO                  PIC 9(03).
       77  NUMKG                   PIC 9(04).
       77  IMPORTE1                PIC 9(08).
       77  TRANSPORTE              PIC 9(04).
       77  IMPORTE2                PIC 9(08).
       77  DESCUENTO               PIC 9(07).
       77  TPAGAR                  PIC 9(08).
       77  TECLA                   PIC X.
       77  IMPRESION               PIC ZZ,ZZZ,ZZZ.

       PROCEDURE DIVISION.
       COMENZAR.

      * SE SACA LA PRIMERA MASCARA.

           DISPLAY " VENTA DE GRAVA " LINE 1 POSITION 35 REVERSE ERASE.
           DISPLAY "FACTURA" LINE 3 POSITION 36.
           DISPLAY "CLIENTE..:" LINE 6 POSITION 20.
           DISPLAY "RETURN-FIN" LINE 6 POSITION 60.
           DISPLAY "PRECIO K.:" LINE 7 POSITION 20.
           DISPLAY "N. KILOS.:" LINE 8 POSITION 20.
           DISPLAY "TRANSPOR.:" LINE 9 POSITION 20.
```

INSTRUCCIONES COBOL

```
* SE PIDEN LOS CAMPOS.

      ACCEPT CLIENTE LINE 6 POSITION 30 NO BEEP.
      IF CLIENTE = SPACES DISPLAY " " ERASE, GO TO FIN.
      ACCEPT PRECIO LINE 7 POSITION 30 NO BEEP.
      ACCEPT NUMKG LINE 8 POSITION 30 NO BEEP.
      ACCEPT TRANSPORTE LINE 9 POSITION 30 NO BEEP.

* SE HACEN LAS OPERACIONES PARA HALLAR IMPORTES PARCIALES Y TOTALES.

      MULTIPLY PRECIO BY NUMKG GIVING IMPORTE1.
      ADD TRANSPORTE, IMPORTE1 GIVING IMPORTE2.
      COMPUTE DESCUENTO = IMPORTE2 / 10.
      SUBTRACT DESCUENTO FROM IMPORTE2 GIVING TPAGAR.

* SE HACE LA IMPRESION DE LA FACTURA.

      DISPLAY "F A C T U R A" LINE 2 POSITION 32 ERASE.
      DISPLAY "-------------" LINE 3 POSITION 32.
      DISPLAY "CLIENTE SENOR " LINE 6 POSITION 20.
      DISPLAY "PRECIO KILO..:" LINE 8 POSITION 20.
      DISPLAY "NUMERO KILOS.:" LINE 9 POSITION 20.
      DISPLAY "PRECIO.......:" LINE 10 POSITION 20.
      DISPLAY "TRANSPORTE...:" LINE 11 POSITION 20.
      DISPLAY "IMPORTE......:" LINE 13 POSITION 20.
      DISPLAY "DESCUENTO....:" LINE 14 POSITION 20.
      DISPLAY "TOTAL A PAGAR:" LINE 16 POSITION 20.
      DISPLAY " G R A C I A S   P O R   S U   V I S I T A "
              LINE 20 POSITION 25 REVERSE.
      DISPLAY CLIENTE LINE 6 POSITION 34.
      MOVE PRECIO TO IMPRESION.
      DISPLAY IMPRESION LINE 8 POSITION 34.
      MOVE NUMKG TO IMPRESION.
      DISPLAY IMPRESION LINE 9 POSITION 34.
      MOVE IMPORTE1 TO IMPRESION.
      DISPLAY IMPRESION LINE 10 POSITION 34.
      MOVE TRANSPORTE TO IMPRESION.
      DISPLAY IMPRESION LINE 11 POSITION 34.
      DISPLAY "-----------" LINE 12 POSITION 34.
      MOVE IMPORTE2 TO IMPRESION.
      DISPLAY IMPRESION LINE 13 POSITION 34.
      MOVE DESCUENTO TO IMPRESION.
      DISPLAY IMPRESION LINE 14 POSITION 34.
      DISPLAY "-----------" LINE 15 POSITION 34.
      MOVE TPAGAR TO IMPRESION.
      DISPLAY IMPRESION LINE 16 POSITION 34.
```

```
* ESPERA LA PULSACION DE UNA TECLA.

    ACCEPT TECLA OFF NO BEEP.

* SE VUELVE AL PRINCIPIO.

    GO TO COMENZAR.
FIN.
```

8.4. INSTRUCCIONES DE TRANSFERENCIAS DE CONTROL.

Seguidamente vamos a ver las instrucciones que nos ván a servir para alterar el flujo normal del programa, bien desviándola a otra parte o deteniéndola.

8.4.1. DE UN LADO A OTRO CON LA INSTRUCCION GO TO.

GO TO..: Esta instrucción como ya hemos visto y utilizado en veces anteriores nos sirve para bifurcar a una parte del programa, pero siempre a la que hallamos indicado, si queremos que se bifurque a un cierto lugar del programa según el valor de una variable tenemos que hacer una serie de preguntas, para evitarlo GO TO dispone del siguiente formato.

> **GO TO et1, et2, et3, ... DEPENDING ON var..:** Donde 'etn' es la etiqueta a donde queremos bifurcar y 'var' es la variable según la que se quiere bifurcar; esta variable ha de ser numérica; si 'var' vale 1 se irá a la primera etiqueta, si vale 2 a la segunda y así sucesivamente, si 'var' vale 0 o su valor sobrepasa el número de etiquetas especificado el programa seguirá su curso normal.

Para ver esta instrucción en funcionamiento vamos a hacer un programa donde se nos dé opción a varias partes tecleando un número, según este número nos iremos a la parte elegida. Abre en tu editor un archivo con nombre de 'PRUEBA19.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA19.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
```

```
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  OPCION                      PIC 9.

       PROCEDURE DIVISION.
       COMIENZO.

      * IMPRESION EN PANTALLA DE LAS OPCIONES DEL MENU.

           DISPLAY " M E N U   P R I N C I P A L "
                   LINE 4 POSITION 25 REVERSE ERASE.
           DISPLAY "1.- MANTENIMIENTOS." LINE 8 POSITION 30.
           DISPLAY "2.- LISTADOS." LINE 9 POSITION 30.
           DISPLAY "3.- PRESUPUESTOS." LINE 10 POSITION 30.
           DISPLAY "4.- OBTENCION DE INFORMES." LINE 11 POSITION 30.
           DISPLAY "5.- PROCESOS ESPECIALES." LINE 12 POSITION 30.
           DISPLAY "6.-" LINE 13 POSITION 30.
           DISPLAY "7.-" LINE 14 POSITION 30.
           DISPLAY "ELIJE OPCION..[ ]" LINE 17 POSITION 10.

      * PETICION DE LA OPCION.

       PIDEOPCION.

           ACCEPT OPCION LINE 17 POSITION 25 NO BEEP.

      * SEGUN LA OPCION PULSADA SE VA A UNA ETIQUETA U OTRA.

           GO TO MANTENIMIENTOS, LISTADOS, PRESUPUESTOS,
                 OBTENCION-DE-INFORMES, PROCESOS-ESPECIALES
                 DEPENDING ON OPCION.

      * SI SE LLEGA AQUI QUIERE DECIR QUE LA OPCION ERA INCORRECTA.

           GO TO PIDEOPCION.

       MANTENIMIENTOS.

           DISPLAY "ESTAS EN MANTENIMIENTOS" LINE 20 POSITION 30.
           GO TO FIN.

       LISTADOS.

           DISPLAY "ESTAS EN LISTADOS" LINE 20 POSITION 30.
           GO TO FIN.
```

```
       PRESUPUESTOS.

           DISPLAY "ESTAS EN PRESUPUESTOS" LINE 20 POSITION 30.
           GO TO FIN.

       OBTENCION-DE-INFORMES.

           DISPLAY "ESTAS EN OBTENCION DE INFORMES" LINE 20 POSITION 30.
           GO TO FIN.

       PROCESOS-ESPECIALES.

           DISPLAY "ESTAS EN PROCESOS ESPECIALES" LINE 20 POSITION 30.

       FIN.
```

Tras compilarlo ejecútalo y podrás ver como según la opción que elijas se bifurca a una u otra de las etiquetas especificadas, intenta hacer exactamente el mismo programa pero sin utilizar este nuevo formato de GO TO.

8.4.2. LA INSTRUCCION ALTER.

ALTER..: Esta instrucción nos vá a servir para cambiar la referencia a una etiqueta, para ello utilizaremos el siguiente formato.

> **ALTER et1 TO PROCEED TO et2..:** Donde 'et1' y 'et2' son dos etiquetas, lo que se hará es cambiar la referencia de 'et1' por 'et2', de tal forma que cuando se haga una bifurcación a la etiqueta 'et1' el programa se irá a 'et2'. Esto nos puede ser bastante útil piensa en una parte de programa que se tenga que repetir desde distintas partes del programa, una solución sería poner esa parte al final y llamarla cuando haga falta, pero ¿como sabrá luego donde ha de volver si no sabe de donde ha sido llamado?, aquí entraría en acción esta instrucción.

Para que veas el funcionamiento y la utilidad que pueda tener esta instrucción abre en tu editor un archivo con el nombre de 'PRUEBA20.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA20.
```

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  TECLA                    PIC X.
       77  OPCION                   PIC 9.

       PROCEDURE DIVISION.
       COMIENZO.

      * IMPRIME EN PANTALLA LAS OPCIONES DE UN MENU.

           DISPLAY " M E N U " LINE 4 POSITION 37 ERASE REVERSE.
           DISPLAY "1.- OPCION INSERVIBLE" LINE 7 POSITION 30.
           DISPLAY "2.- IGUAL QUE LA 1" LINE 8 POSITION 30.
           DISPLAY "3.- FUERA DE SERVICIO" LINE 9 POSITION 30.
           DISPLAY "4.- HE SALIDO" LINE 10 POSITION 30.
           DISPLAY "ELIJE LA QUE QUIERAS.[ ]" LINE 15 POSITION 10.

      * PIDE LA OPCION.

       PEDIROPCION.

           ACCEPT OPCION LINE 15 POSITION 32 NO BEEP.

      * SE BIFURCA SEGUN LA OPCION PULSADA.

           GO TO ET1, ET2, ET3, ET4 DEPENDING ON OPCION.

      * SI SE LLEGA AQUI LA OPCION ES INCORRECTA.

           GO TO PEDIROPCION.

       ET1.

           DISPLAY "ESTA OPCION TE DIJE QUE ERA INSERVIBLE"
                   LINE 20 POSITION 30.

      * SE CAMBIA LA REFERENCIA DE VUELVE POR ET11.

           ALTER VUELVE TO PROCEED TO ET11.
           GO TO PULSATECLA.

       ET11.
```

```
           DISPLAY "SIGO EN INSERVIBLE, ME VOY" LINE 22 POSITION 30.
           GO TO FIN.

       ET2.

           DISPLAY "ESTA OPCION ERA IGUAL DE INSERVIBLE"
                   LINE 20 POSITION 30.

      * SE CAMBIA LA REFERENCIA DE VUELVE POR ET21.

           ALTER VUELVE TO PROCEED TO ET21.
           GO TO PULSATECLA.

       ET21.

           DISPLAY "SIGO TAN INSERVIBLE COMO ANTES, ME VOY"
                LINE 22 POSITION 30.
           GO TO FIN.

       ET3.

           DISPLAY "ESTA OPCION ESTA AVERIADA"
                   LINE 20 POSITION 30.

      * SE CAMBIA LA REFERENCIA DE VUELVE POR ET31.

           ALTER VUELVE TO PROCEED TO ET31.
           GO TO PULSATECLA.

       ET31.

           DISPLAY "SIGO FUERA DE SERVICIO"
                   LINE 22 POSITION 30.
           GO TO FIN.

       ET4.

           DISPLAY "ESTA OPCION ESTA DE VACACIONES"
                   LINE 20 POSITION 30.

      * SE CAMBIA LA REFERENCIA DE VUELVE POR ET41.

           ALTER VUELVE TO PROCEED TO ET41.
           GO TO PULSATECLA.
```

```
ET41.

    DISPLAY "SIGO DE VACACIONES"
            LINE 22 POSITION 30.
    GO TO FIN.

PULSATECLA.

    ACCEPT TECLA OFF NO BEEP.

VUELVE.

    GO TO D.

D.

FIN.
```

En este ejemplo se nos dá a elegir entre las opciones de un menú, cada una de estas opciones imprime un texto diferente y despues espera la pulsación de una tecla, en lugar de repetir el acepte de la tecla en cada una de las opciones se hace uso del párrafo 'PULSATECLA', este espera la pulsación de una tecla y despues hace un 'GO TO D', pero antes de venir a este párrafo se habrá cambiado la referencia de esta etiqueta por la correspondiente. En este caso el programa es sólo una demostración, ya que hubiera sido más corto aceptar la tecla en cada una de las opciones.

Es importante que trás la etiqueta a la que se vá a cambiar la referencia no halla otra instrucción más que **GO TO**. Una vez que se ha cambiado la referencia a una etiqueta esta sigue cambiada hasta que otra instrucción **ALTER** la vuelva a modificar.

8.4.3. LA INSTRUCCION STOP.

STOP..: Hasta ahora cada vez que hemos querido terminar la ejecución de un programa nos hemos ido a una etiqueta vacía, el ejecutor Cobol al no encontrar nada detiene la ejecución del programa, pero hay una instrucción especial para provocar paradas temporales o fijas en un programa, esta es **STOP**, para usarla tenemos los siguientes formatos.

STOP RUN..: Provoca el fin del programa, la ejecución de esta instrucción dá como resultado el mensaje 'COBOL STOP RUN AT' dejando el programa donde estuviera.

STOP "literal"..: Este formato causa la salida por pantalla del 'literal' y nos preguntará si queremos continuar la ejecución del programa, nos puede servir para causar una parada temporal en el programa.

Para ver en funcionamiento los dos formatos de esta instrucción abre en tu editor un archivo con el nombre de 'PRUEBA21.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA21.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA Y HACE USO DE STOP "literal".

           DISPLAY " " ERASE.
           STOP "QUIERES CONTINUAR".

      * PROVOCA UN STOP RUN.

           DISPLAY "ESTO LO VERAS POR PANTALLA" LINE 15 POSITION 20.
           STOP RUN.

       OTRA.

           DISPLAY "PERO ESTO NO LLEGARAS A VERLO" LINE 16 POSITION 20.
```

Cuando lo ejecutes veras aparecer en la última línea de la pantalla el mensaje 'CONTINUE (Y/N)', el mensaje puesto como literal te costará trabajo verlo ya que desaparece con mucha rapidez; si a la pregunta que se nos formula respondemos 'N' el programa se detendrá en ese punto, si respondemos 'Y' el programa continuará y veremos en pantalla la impresión del primer mensaje, el segundo no saldrá ya que antes se ha hecho un **STOP RUN** y esto detendrá el programa. Pruébalo.

8.5. LA INSTRUCCION IF.

IF..: Esta instrucción ya la conocemos, nos sirve para tomar decisiones y actuar en consecuencia, para ello tenemos los siguientes formatos.

IF cond sent..: Este formato es el que ya conocemos, en él cuando la condición 'cond' sea verdadera entonces se ejecutará la sentencia 'sent'.

IF cond sent1 ELSE sent2..: En este segundo formato cuando 'cond' sea verdadero se ejecutará 'sent1' en caso contrario se ejecutará 'sent2'.

Además en cualquiera de estos formatos podemos usar los operadores lógicos siguientes para concatenar varias condiciones.

AND..: Al unir dos condiciones con este operador para que la respuesta a la pregunta global sea verdadera lo han de ser las dos condiciones expuestas.

OR..: Al unir dos condiciones con este operador para que la respuesta a la pregunta global sea verdadera bastará con que lo sea una de las dos condiciones expuestas.

NOT..: La anteposición de este operador a una condición tiene como resultado la inversión de la respuesta a la condición, es decir si era verdadera dará como respuesta falsa y viceversa.

Además de estos operadores lógicos podemos utilizar los siguientes operadores relacionales.

<..: El signo de menor preguntará si el valor expuesto a la izquierda del signo es menor que el expuesto a la derecha de este.

>..: El signo mayor preguntará si el valor expuesto a la izquierda del signo es mayor que el expuesto a la derecha de este.

=..: El signo igual preguntará por la igualdad del valor a ambos lados del signo.

Para ver la utilización de distintos formatos de decisión vamos a hacer un programa que haga utilización de ellos; el programa es el conocido juego donde el ordenador te dice que pienses un animal y que vayas respondiendo a sus preguntas, según

estas el ordenador irá deduciendo que animal es, si al final falla te hará una serie de preguntas sobre ese animal para identificarlo y diferenciarlo de los demás, para ello tendrá que tomar bastantes decisiones y además haremos uso de muchas de las instrucciones vistas hasta ahora. El listado es bastante extenso pero vale la pena tecleario. ¿Estas dispuesto?, pues abre en tu editor un archivo con el nombre de 'PRUEBA22.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA22.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

      * TABLA QUE VA A ACOGER LOS DATOS DE LOS ANIMALES.

        01  DATOS-ANIMALES.
            03  HABITATS              PIC X(20) OCCURS  5 TIMES.
            03  ESPECIES                        OCCURS 10 TIMES.
                05  NOM-ESP           PIC X(20).
                05  HAB-ESP           PIC 9     OCCURS  5 TIMES.
            03  TAMANOS               PIC X(15) OCCURS 10 TIMES.
            03  SONIDOS               PIC X(15) OCCURS 50 TIMES.
            03  ANIMALES                        OCCURS 1000 TIMES.
                05  HAB-ANI           PIC 9.
                05  ESP-ANI           PIC 99.
                05  TAM-ANI           PIC 99.
                05  SON-ANI           PIC 99.
                05  NOM-ANI           PIC X(20).

      * CONTADORES CON EL NUMERO DE ESPECIES, HABITATS, TAMAROS
      * SONIDOS Y ANIMALES QUE SE CONOCEN HASTA EL MOMENTO.

        77  NUM-HAB                   PIC 9.
        77  NUM-ESP                   PIC 99.
        77  NUM-TAM                   PIC 99.
        77  NUM-SON                   PIC 99.
        77  NUM-ANI                   PIC 9(04).
```

```
      * VARIABLES AUXILIARES PARA CONTADORES.

       77  CUENTA                          PIC 9(04).
       77  CUENTA1                         PIC 9(04).

      * VARIABLE QUE LLEVARA LA LINEA DE IMPRESION.

       77  LINEA                           PIC 99 VALUE ZERO.

      * VARIABLE PARA RECOGER LAS RESPUESTAS.

       77  RESPUESTA                       PIC X.

      * VARIABLES AUXILIARES PARA CONTROL DE RESPUESTAS.

       77  ANIMAL                          PIC X(20).
       77  HABITAT                         PIC 9.
       77  ESPECIE                         PIC 99.
       77  TAMANO                          PIC 99.
       77  SONIDO                          PIC 99.
       77  ESPECIE1                        PIC X(20).
       77  TAMANO1                         PIC X(15).
       77  SONIDO1                         PIC X(15).
       77  ENCONTRADO                      PIC 9.

       PROCEDURE DIVISION.
       COMIENZO.

      * SE MUEVE A LAS TABLAS LOS ANIMALES QUE SE CONOCEN DE BASE.

           MOVE "TERRESTRE" TO HABITATS(1),
           MOVE "ACUATICO"  TO HABITATS(2),
           MOVE 2 TO NUM-HAB,
           MOVE "MAMIFERO"  TO NOM-ESP(1),
           MOVE "AVE" TO NOM-ESP(2),
           MOVE "REPTIL" TO NOM-ESP(3),
           MOVE 1 TO HAB-ESP(1, 1),
           MOVE 2 TO HAB-ESP(1, 2),
           MOVE 1 TO HAB-ESP(2, 1),
           MOVE 2 TO HAB-ESP(2, 2),
           MOVE 1 TO HAB-ESP(3, 1),
           MOVE 2 TO HAB-ESP(3, 2),
           MOVE 3 TO NUM-ESP,
           MOVE "MUY GRANDE" TO TAMANOS(1),
           MOVE "GRANDE"     TO TAMANOS(2),
           MOVE "MEDIANO"    TO TAMANOS(3),
           MOVE "PEQUENO"    TO TAMANOS(4),
```

```
           MOVE "MUY PEQUENO" TO TAMANOS(5),
           MOVE 5 TO NUM-TAM,
           MOVE "LADRA" TO SONIDOS(1),
           MOVE "MAULLA" TO SONIDOS(2),
           MOVE "PIA" TO SONIDOS(3),
           MOVE "SILBA" TO SONIDOS(4),
           MOVE 4 TO NUM-SON,
           MOVE 1 TO HAB-ANI(1),
           MOVE 1 TO ESP-ANI(1),
           MOVE 4 TO TAM-ANI(1),
           MOVE 1 TO SON-ANI(1),
           MOVE "PERRO" TO NOM-ANI(1),
           MOVE 1 TO HAB-ANI(2),
           MOVE 1 TO ESP-ANI(2),
           MOVE 4 TO TAM-ANI(2),
           MOVE 2 TO SON-ANI(2),
           MOVE "GATO" TO NOM-ANI(2),
           MOVE 1 TO HAB-ANI(3),
           MOVE 2 TO ESP-ANI(3),
           MOVE 4 TO TAM-ANI(3),
           MOVE 3 TO SON-ANI(3),
           MOVE "CANARIO" TO NOM-ANI(3),
           MOVE 2 TO HAB-ANI(4),
           MOVE 1 TO ESP-ANI(4),
           MOVE 3 TO TAM-ANI(4),
           MOVE 4 TO SON-ANI(4),
           MOVE "DELFIN" TO NOM-ANI(4),
           MOVE 1 TO HAB-ANI(5),
           MOVE 3 TO ESP-ANI(5),
           MOVE 4 TO TAM-ANI(5),
           MOVE 4 TO SON-ANI(5),
           MOVE "SALAMANDRA" TO NOM-ANI(5),
           MOVE 5 TO NUM-ANI.

       * COMIENZA EL JUEGO AVISANDO CON UN MENSAJE.

        COMIENZOJUEGO.

           MOVE 7 TO LINEA.

           DISPLAY "PIENSA EN UN ANIMAL" LINE 2 POSITION 30 ERASE.
           DISPLAY "YO INTENTARE ADIVINARLO" LINE 3 POSITION 30.
           DISPLAY "VE RESPONDIENDO A MIS PREGUNTAS" LINE 4 POSITION 30.

        VOLVER.

           DISPLAY "RESPONDE SIEMPRE S o N ¿VALE?" LINE 5 POSITION 30.
```

```
* PONE A CERO LAS VARIABLES DE CONTROL DE RESPUESTAS.

    MOVE 0 TO HABITAT, ESPECIE, TAMANO, SONIDO.

 PREGUNTAHABITAT.

    MOVE 0 TO CUENTA.

* PREGUNTA EL HABITAT DE ESE ANIMAL.

 PREGUNTA1.

    ADD 1 TO CUENTA, ADD 1 TO LINEA.
    IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.
    DISPLAY "ES" LINE LINEA POSITION 5.
    DISPLAY HABITATS(CUENTA) LINE LINEA POSITION 8.
    DISPLAY "?" LINE LINEA POSITION 29.

 PREGUNTA12.

* SI ES EL QUE SE HA PUESTO PASA A PEDIR LA ESPECIE
* EN CASO CONTRARIO SIGUE PREGUNTANDO POR EL HABITAT
* SI NO ES NINGUNO DE LOS CONOCIDOS SE VA A LA ETIQUETA
* NOLOSE DONDE SE DARA POR VENCIDO Y PREGUNTARA LOS
* DATOS NECESARIOS.

    ACCEPT RESPUESTA LINE LINEA POSITION 31 NO BEEP.
    IF RESPUESTA = "S" OR RESPUESTA = "s" MOVE CUENTA TO HABITAT,
       GO TO PREGUNTAESPECIE.
    IF RESPUESTA NOT = "N" AND RESPUESTA NOT = "n"
       GO TO PREGUNTA12.
    IF CUENTA = NUM-HAB GO TO NOLOSE ELSE GO TO PREGUNTA1.

 PREGUNTAESPECIE.

    MOVE 0 TO CUENTA.

* VA SUMANDOLE UNO A LA VARIABLE DE CONTROL DE LINEA
* SI LLEGA A 23 SE BORRA LA PANTALLA Y SE PONE LA VARIABLE
* A 1 PARA SEGUIR.

 PREGUNTA2.

    ADD 1 TO CUENTA, ADD 1 TO LINEA.
    IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.
    MOVE 0 TO CUENTA1, ENCONTRADO.
```

```
BUSCAESPECIE.

* BUSCA UNA ESPECIE QUE SE CORRESPONDA CON EL HABITAT
* QUE TIENE EL ANIMAL.

    ADD 1 TO CUENTA1.
    IF HAB-ESP(CUENTA, CUENTA1) = HABITAT
       MOVE 5 TO CUENTA1, MOVE 1 TO ENCONTRADO.
    IF CUENTA1 < 5 GO TO BUSCAESPECIE.
    IF ENCONTRADO NOT = 0 GO TO PREGUNTA21.
    IF CUENTA = NUM-ESP GO TO NOLOSE.

* NOS PREGUNTA LA ESPECIE USANDO EL MISMO SISTEMA QUE PARA
* PEDIRNOS EL HABITAT.

PREGUNTA21.

    DISPLAY "ES" LINE LINEA POSITION 5.
    DISPLAY NOM-ESP(CUENTA) LINE LINEA POSITION 8.
    DISPLAY "?" LINE LINEA POSITION 29.

PREGUNTA22.

    ACCEPT RESPUESTA LINE LINEA POSITION 31 NO BEEP.
    IF RESPUESTA = "S" OR RESPUESTA = "s" MOVE CUENTA TO ESPECIE,
       GO TO PREGUNTATAMANO.
    IF RESPUESTA NOT = "N" AND RESPUESTA NOT = "n"
       GO TO PREGUNTA22.
    IF CUENTA = NUM-ESP GO TO NOLOSE ELSE GO TO PREGUNTA2.

PREGUNTATAMANO.

    MOVE 0 TO CUENTA.

PREGUNTA3.

* PREGUNTA EL TAMAÑO Y SIGUE EL MISMO SISTEMA.

    ADD 1 TO CUENTA, ADD 1 TO LINEA.
    IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.
    DISPLAY "ES" LINE LINEA POSITION 5.
    DISPLAY TAMANOS(CUENTA) LINE LINEA POSITION 8.
    DISPLAY "?" LINE LINEA POSITION 29.

PREGUNTA31.

    ACCEPT RESPUESTA LINE LINEA POSITION 31 NO BEEP.
```

```
        IF RESPUESTA = "S" OR RESPUESTA = "s" MOVE CUENTA TO TAMANO,
           GO TO PREGUNTASONIDO.
        IF RESPUESTA NOT = "N" AND RESPUESTA NOT = "n"
           GO TO PREGUNTA31.
        IF CUENTA = NUM-TAM GO TO NOLOSE ELSE GO TO PREGUNTA3.

    PREGUNTASONIDO.

        MOVE 0 TO CUENTA.

    PREGUNTA4.

    * PREGUNTA EL SONIDO IGUAL QUE LOS PUNTOS ANTERIORES.

        ADD 1 TO CUENTA, ADD 1 TO LINEA.
        IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.
        DISPLAY SONIDOS(CUENTA) LINE LINEA POSITION 8.
        DISPLAY "?" LINE LINEA POSITION 29.

    PREGUNTA41.

        ACCEPT RESPUESTA LINE LINEA POSITION 31 NO BEEP.
        IF RESPUESTA = "S" OR RESPUESTA = "s" MOVE CUENTA TO SONIDO,
           GO TO PREGUNTAANIMAL.
        IF RESPUESTA NOT = "N" AND RESPUESTA NOT = "n"
           GO TO PREGUNTA41.
        IF CUENTA = NUM-SON GO TO NOLOSE ELSE GO TO PREGUNTA4.

    PREGUNTAANIMAL.

        MOVE 0 TO CUENTA.

    PREGUNTA5.

    * UNA VEZ TIENE TODOS LOS DATOS ANTERIORES BUSCARA
    * EN LA TABLA DE ANIMALES LOS QUE CUMPLAN DICHAS CONDICIONES
    *  Y NOS PREGUNTARA SI ES ESE EL QUE HABIAMOS PENSADO.

        ADD 1 TO CUENTA.
        IF HAB-ANI(CUENTA) = HABITAT AND
           ESP-ANI(CUENTA) = ESPECIE AND
           TAM-ANI(CUENTA) = TAMANO AND
           SON-ANI(CUENTA) = SONIDO GO TO PREGUNTA51.
        IF CUENTA = NUM-ANI GO TO NOLOSE ELSE GO TO PREGUNTA5.

    PREGUNTA51.

        ADD 1 TO LINEA.
```

```
           IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.
           DISPLAY "ES" LINE LINEA POSITION 5.
           DISPLAY NOM-ANI(CUENTA) LINE LINEA POSITION 8.
           DISPLAY " ESE ANIMAL?" LINE LINEA POSITION 29.

       PREGUNTA52.

      * SI ES ESE EL ANIMAL QUE BUSCABAMOS SE VA A LA EQTIQUETA
      * ENCONTRADP, EN CASO CONTRARIO SIGUE BUSCANDO ENTRE LOS
      * ANIMALES QUE CONOCE, SI NO LO ENCUENTRA SE VA A LA ETIQUETA
      * NOLOSE PARA PREGUNTAR EL ANIMAL QUE ERA.

           ACCEPT RESPUESTA LINE LINEA POSITION 41 NO BEEP.
           IF RESPUESTA = "S" OR RESPUESTA = "s" GO TO ACERTADO.
           IF RESPUESTA NOT = "N" AND RESPUESTA NOT = "n"

              GO TO PREGUNTA52.
           IF CUENTA = NUM-ANI GO TO NOLOSE ELSE GO TO PREGUNTA5.

       ACERTADO.

      * SI HA ACERTADO EL ORDENADOR SE AUTOELOGIA Y SE VA A LA
      * ETIQUETA JUGAROTRAVEZ.

           ADD 1 TO LINEA.
           DISPLAY "A QUE SOY INTELIGENTE?" LINE LINEA POSITION 30.
           GO TO JUGAROTRAVEZ.

       NOLOSE.

      * A ESTA ETIQUETA LLEGA CUANDO NO HA PODIDO DESCUBRIR EL
      * ANIMAL QUE ERA.

           ADD 1 TO LINEA.
           IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.

           DISPLAY "ME DOY POR VENCIDO" LINE LINEA POSITION 30.

           ADD 1 TO LINEA.

      * NOS PREGUNTA EL NOMBRE QUE RECIBE ESE ANIMAL.

           DISPLAY "QUE ANIMAL ERA?" LINE LINEA POSITION 10.
           ACCEPT ANIMAL LINE LINEA POSITION 41 NO BEEP.

           ADD 1 TO LINEA.
           IF LINEA = 23 DISPLAY " " ERASE MOVE 1 TO LINEA.
```

```
* SI LA VARIABLE HABITAT ES IGUAL A CERO QUIERE DECIR QUE
* NO SABEMOS EL HABITAT QUE TIENE ESE ANIMAL, POR TANTO
* SE LO PREGUNTAMOS. EL MISMO SISTEMA SE SIGUE PARA PREGUNTAR
* LOS DEMAS CAMPOS.

    IF HABITAT = 0 DISPLAY "QUE HABITAT TIENE ESTE ANIMAL?"
       LINE LINEA POSITION 10,
       ADD 1 TO NUM-HAB,
       MOVE NUM-HAB TO HABITAT,
       ACCEPT HABITATS(NUM-HAB) LINE LINEA POSITION 41 NO BEEP.

    ADD 1 TO LINEA.
    IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.

    IF ESPECIE NOT = 0 GO TO PREGTAM.
    DISPLAY "DIME EN QUE ESPECIE ESTA?"
            LINE LINEA POSITION 10,
            ACCEPT ESPECIE1 LINE LINEA POSITION 41 NO BEEP,
            MOVE 0 TO CUENTA.

BUSCA1.

        ADD 1 TO CUENTA.

        IF NOM-ESP(CUENTA) = ESPECIE1
           MOVE CUENTA TO ESPECIE,
           MOVE HABITAT TO HAB-ESP(ESPECIE, 2),
           MOVE NUM-ESP TO CUENTA.
        IF CUENTA NOT = NUM-ESP GO TO BUSCA1.
        IF ESPECIE = 0
           ADD 1 TO NUM-ESP,
           MOVE NUM-ESP TO ESPECIE,
           MOVE ESPECIE1 TO NOM-ESP(NUM-ESP),
           MOVE HABITAT TO HAB-ESP(NUM-ESP, 1).

PREGTAM.

    ADD 1 TO LINEA.
    IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.

    IF TAMANO NOT = 0 GO TO PREGSON.
    DISPLAY "EN QUE TAMAÑO LO ENCUADRO?"
            LINE LINEA POSITION 10,
            ACCEPT TAMANO1 LINE LINEA POSITION 41 NO BEEP.
            MOVE 0 TO CUENTA.

BUSCA2.
```

```
           ADD 1 TO CUENTA.
           IF TAMANOS(CUENTA) = TAMANO1
              MOVE CUENTA TO TAMANO,
              MOVE NUM-TAM TO CUENTA.
           IF CUENTA NOT = NUM-TAM GO TO BUSCA2.
           IF TAMANO = 0
              ADD 1 TO NUM-TAM,
              MOVE NUM-TAM TO TAMANO,
              MOVE TAMANO1 TO TAMANOS(NUM-ESP).

       PREGSON.

           ADD 1 TO LINEA.
           IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.

           IF SONIDO = 0 DISPLAY "QUE SONIDO EMITE ESTE ANIMAL?"
              LINE LINEA POSITION 10,
              ADD 1 TO NUM-SON,
              MOVE NUM-SON TO SONIDO,
              ACCEPT SONIDOS(NUM-SON) LINE LINEA POSITION 41 NO BEEP.

           ADD 1 TO LINEA.
           IF LINEA = 23 DISPLAY " " ERASE, MOVE 1 TO LINEA.

      * UNA VEZ HA PEDIDO TODOS LOS DATOS SE DISCULPA DICIENDO
      * QUE PARA LA PROXIMA YA LO SABE.

           DISPLAY "DE ACUERDO, LA PROXIMA LO RECORDARE"
                   LINE LINEA POSITION 30.

      * GUARDA TODOS LOS DATOS PEDIDOS EN LA TABLA DE ANIMALES.

           ADD 1 TO NUM-ANI,
               MOVE HABITAT TO HAB-ANI(NUM-ANI),
               MOVE ESPECIE TO ESP-ANI(NUM-ANI),
               MOVE TAMANO TO TAM-ANI(NUM-ANI),
               MOVE SONIDO TO SON-ANI(NUM-ANI),
               MOVE ANIMAL TO NOM-ANI(NUM-ANI).

       JUGAROTRAVEZ.

      * EN ESTA ETIQUETA NOS PREGUNTA SI QUEREMOS SEGUIR JUGANDO.

           ADD 1 TO LINEA.
           DISPLAY "JUEGAS OTRA VEZ?" LINE LINEA POSITION 20.

       PREGUNTAJ.
```

```
           ACCEPT RESPUESTA LINE LINEA POSITION 37 NO BEEP.
           IF RESPUESTA = "N" OR RESPUESTA = "n" GO TO SALIR.
           IF RESPUESTA = "S" OR RESPUESTA = "s"
              DISPLAY " " ERASE, MOVE 7 TO LINEA, GO TO VOLVER.

       SALIR.

           DISPLAY "**** ADIOS ****" LINE 12 POSITION 34 ERASE.
           STOP RUN.
```

Tras compilárlo lo podrás ejecutar, al comenzar el ordenador conoce cinco animales, te dirá que pienses un animal y te irá preguntando una serie de cosas a las que tendrás que responder sin mentir como es natural, si tras el interrogatorio el ordenador no logra deducir el animal que has pensado se dará por vencido y te preguntará que animal era, te preguntará lo que no sepa sobre él y lo almacenará en la tabla, la próxima lo sabrá. El único inconveniente de este programa es que cuando se detiene pierde todo lo que había aprendido, cuando comencemos a tratar con ficheros podremos mejorarlo.

8.6. INSTRUCCIONES DE MANIPULACION DE DATOS.

Con las instrucciones que tenemos a continuación podremos manipular datos, podremos pasarlos de un identificador a otro y alterarlos.

8.6.1. LA INSTRUCCION MOVE.

MOVE...: Esta instrucción ya la conocemos, nos sirve para depositar un valor en una variable, al depositar un valor en una variable lo que tuviera anteriormente se desplaza; para usar esta instrucción tenemos los siguientes formatos.

MOVE valor TO var..: Donde 'valor' es el contenido que se desea depositar en 'var'. Este formato es el que hemos ido usando hasta ahora.

MOVE CORR grp1 TO grp2..: En este segundo formato se mueve el grupo 'grp1' al grupo 'grp2' correspondiendo con el nombre de cada uno de sus campos. Este formato funciona igual que el **ADD CORR**.

Esta instrucción ya la hemos usado bastante y de distintas formas, por lo cual no vamos a hacer ningún programa ejemplo, esta vez hazlo tú.

8.6.2. LA INSTRUCCION INSPECT.

INSPECT..: Esta instrucción Cobol nos vá a servir para inspeccionar campos, vamos a poder hacer dos cosas con ella, contar el número de veces que aparece un determinado caracter en una cadena y reemplazar un caracter por otro en una cadena. Vamos a ver primero los formatos con los cuales podemos contar la aparición de un caracter en una cadena.

INSPECT palabra TALLYING cont FOR ALL carac..: Donde 'palabra' es la cadena que vamos a inspeccionar, **TALLYING** es el formato con el cual le decimos a la instrucción que vamos a contar, 'cont' es la variable numérica donde vamos a almacenar el número de veces que ha aparecido el caracter buscado, el resultado de la búsqueda no se mueve a esta variable sino que se suma a su contenido anterior; **FOR ALL** es el modo de búsqueda, con él le decimos que cuente todas las apariciones del caracter buscado y por último 'carac' es el caracter que se busca en la cadena.

Para ilustrar este primer formato vamos a hacer un programa que nos pida una palabra, nos pida un caracter y nos diga la frecuencia de ese caracter en la palabra. Abre en tu editor un archivo con el nombre de 'PRUEBA23.CBL' y teclea lo que sigue.

```
123456789012
      A  B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA23.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  PALABRA                    PIC X(20).
       77  LETRA                      PIC X.
       77  CONTADOR                   PIC 99.

      PROCEDURE DIVISION.
      COMIENZO.

          DISPLAY "DAME LA PALABRA" LINE 3 POSITION 30 ERASE.
```

```
* PIDE LA PALABRA DONDE SE VA A BUSCAR.

    ACCEPT PALABRA LINE 5 POSITION 30 PROMPT NO BEEP.
    DISPLAY "QUE CARACTER BUSCAS" LINE 7 POSITION 30.

* PIDE EL CARACTER A BUSCAR.

    ACCEPT LETRA LINE 9 POSITION 30 NO BEEP.

* BUSCA LA APARICION DEL CARACTER Y LO ALMACENA EN CONTADOR
* PRIMERO PONEMOS CONTADOR A CERO.

    MOVE 0 TO CONTADOR.
    INSPECT PALABRA TALLYING CONTADOR FOR ALL LETRA.

* SE IMPRIMER EL NUMERO DE VECES QUE APARECIO.

    DISPLAY "APARECIO" LINE 11 POSITION 30.
    DISPLAY CONTADOR LINE 11 POSITION 39.
    DISPLAY "VECES" LINE 11 POSITION 42.

* DA OPCION A VOLVER O TERMINAR.

    DISPLAY "QUIERES SEGUIR S/N" LINE 15 POSITION 30.
    ACCEPT LETRA LINE 17 POSITION 30 NO BEEP.
    IF LETRA = "S" OR LETRA = "s" GO TO COMIENZO.
```

El programa cuando lo ejecutes te pedirá una palabra, a continuación te preguntará que caracter es el que buscas en la palabra, con estos dos datos inspecciona la cadena buscando el caracter, cada vez que encuentra un caracter igual al buscado le suma uno al contador que hemos utilizado, por eso lo ponemos a cero antes de comenzar; al final imprimirá el número de veces que apareció el caracter buscado y nos dará opción a continuar o abandonar el programa.

INSPECT palabra TALLYING cont FOR LEADING car..: En este segundo formato lo único que cambia es el modo de búsqueda que ahora es **FOR LEADING**, en este modo se contarán todas las apariciones del caracter que se busca hasta que encontremos uno diferente, en ese momento se detendrá la búsqueda; si el primer caracter de la palabra ya es diferente del que se busca nos devolverá un cero como resultado.

Para probar este segundo formato coge el programa

anterior y donde ponías **FOR ALL** pón **FOR LEADING**, despues prueba con las mismas palabras con las que probaste antes y verás la diferencia.

INSPECT palabra TALLYING cont FOR CHARACTERS..: En este tercer formato se ha omitido el caracter que se busca y se ha cambiado el modo por **FOR CHARACTERS**; en él no se cuenta un caracter especifico, sino todos los caracteres que tenga la cadena, incluidos los espacios en blanco, utilizándolo tal y como aquí lo tenemos siempre nos dará el mismo resultado, este será la longitud declarada del campo en la **DATA DIVISION**, para verlo coge el programa anterior y prueba con este nuevo formato.

Hasta ahora hemos visto como se hacían distintas búsquedas de caracteres en cadenas, pero siempre se comienza a contar desde el principio del campo comenzando por la izquierda y se termina por la derecha, no podemos empezar a partir de un determinado caracter por ejemplo, para ello tras los formatos de búsqueda vistos anteriormente podemo añadir una de las siguientes opciones.

BEFORE INITIAL car..: Donde 'car' es el caracter hasta donde se desea llegar, este formato buscará el caracter que se halla especificado hasta que aparezca el caracter 'car'.

AFTER INITIAL car..: Donde 'car' es el caracter a partir del cual queremos comenzar a buscar, este formato buscará el caracter 'car', y a partir de él comenzará a contar las apariciones del caracter buscado.

Para ver en funcionamiento esta dos últimas opciones vamos a hacer un programa al que dándole una palabra, una letra a buscar y una letra a partir de la que buscar nos diga cuantas veces aparece el caracter buscado antes del caracter especificado y cuantas veces aparece despues de este. Abre en tu editor un archivo con el nombre 'PRUEBA24.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA24.
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. PC.
OBJECT-COMPUTER. PC.

DATA DIVISION.
WORKING-STORAGE SECTION.

77  PALABRA                    PIC X(20).
77  LETRA1                     PIC X.
77  LETRA2                     PIC X.
77  CONTADOR                   PIC 99.

PROCEDURE DIVISION.
COMIENZO.

    DISPLAY "DAME LA PALABRA" LINE 3 POSITION 30 ERASE.

* SE PIDE LA PALABRA DONDE SE VA A BUSCAR.

    ACCEPT PALABRA LINE 5 POSITION 30 PROMPT NO BEEP.
    DISPLAY "QUE CARACTER BUSCAS" LINE 7 POSITION 30.

* SE PIDE EL CARACTER A BUSCAR.

    ACCEPT LETRA1 LINE 9 POSITION 30 NO BEEP.
    DISPLAY "PULSA EL CARACTER A PARTIR DEL CUAL CONTARE"
            LINE 11 POSITION 30.

* SE PIDE EL CARACTER QUE VA A SERVIR COMO BARRERA PARA CONTAR
* LAS DOS PARTES DE LA CADENA.

    ACCEPT LETRA2 LINE 13 POSITION 30 NO BEEP.

* SE BUSCA EL CARACTER ANTES DE LA APARICION DEL CARACTER BARRERA.

    MOVE 0 TO CONTADOR.
    INSPECT PALABRA TALLYING CONTADOR FOR ALL LETRA1
            BEFORE INITIAL LETRA2.
    DISPLAY "LA LETRA" LINE 15 POSITION 30.
    DISPLAY LETRA1 LINE 15 POSITION 39.
    DISPLAY "APARECE" LINE 15 POSITION 41.
    DISPLAY CONTADOR LINE 15 POSITION 49.
    DISPLAY "ANTES DE LA LETRA" LINE 15 POSITION 52.
    DISPLAY LETRA2 LINE 15 POSITION 70.
```

* SE BUSCA EL CARACTER DESPUES DE LA APARICION DEL CARACTER BARRERA.

```
MOVE 0 TO CONTADOR.
INSPECT PALABRA TALLYING CONTADOR FOR ALL LETRA1
        AFTER INITIAL LETRA2.
DISPLAY "LA LETRA" LINE 17 POSITION 30.
DISPLAY LETRA1 LINE 17 POSITION 39.
DISPLAY "APARECE" LINE 17 POSITION 41.
DISPLAY CONTADOR LINE 17 POSITION 49.
DISPLAY "DESPUES DE LA LETRA" LINE 17 POSITION 52.
DISPLAY LETRA2 LINE 17 POSITION 72.
```

* SE DA OPCION A SEGUIR EN EL PROGRAMA.

```
DISPLAY "QUIERES CONTINUAR S/N" LINE 20 POSITION 30.
ACCEPT LETRA1 LINE 22 POSITION 30 NO BEEP.
IF LETRA1 = "S" OR LETRA1 = "s" GO TO COMIENZO.
```

Como siempre tras compilarlo podrás ejecutárlo y ver el resultado, en este programa se combina el uso de **FOR ALL** con **AFTER INITIAL** y **BEFORE INITIAL**, prueba a combinar las opciones **CHARACTERS** y **LEADING** con las dos últimas opciones vistas.

Al comienzo de esta instrucción decíamos que nos iba a servir para contar y reemplazar, hasta ahora hemos visto todos los formatos posibles para contar las apariciones de un determinado caracter en una cadena, para reemplazar un caracter en una cadena tenemos los siguientes formatos.

INSPECT palabra REPLACING ALL car1 BY car2..: En este primer formato se cambian todas las apariciones del caracter 'car1' en 'palabra' por el caracter 'car2'.

Vamos a hacer un pequeño programa para ver actuar este formato, abre en tu editor un archivo con el nombre de 'PRUEBA24.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA25.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
```

```
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  PALABRA                 PIC  X(20).
       77  LETRA1                  PIC  X.
       77  LETRA2                  PIC  X.

       PROCEDURE DIVISION.
       COMIENZO.

           DISPLAY "DAME LA PALABRA" LINE 3 POSITION 30 ERASE.

      * SE PIDE LA PALABRA DONDE SE VA A REEMPLAZAR.

           ACCEPT PALABRA LINE 5 POSITION 30 PROMPT NO BEEP.
           DISPLAY "CARACTER A SUSTITUIR" LINE 7 POSITION 30.

      * SE PIDE EL CARACTER QUE SE VA A SUSTITUIR.

           ACCEPT LETRA1 LINE 9 POSITION 30 NO BEEP.
           DISPLAY "CARACTER POR EL QUE SUSTITUIR" LINE 11 POSITION 30.

      * SE PIDE EL CARACTER QUE SE HA DE PONER EN LUGAR DEL EXISTENTE.

           ACCEPT LETRA2 LINE 13 POSITION 30 NO BEEP.

      * SE INSPECCIONA LA PALABRA SUSTITUYENDO TODAS LAS APARICIONES
      * DE LETRA1 POR LETRA2.

           INSPECT PALABRA REPLACING ALL LETRA1 BY LETRA2.
           DISPLAY "LA PALABRA QUEDA ASI" LINE 15 POSITION 30.
           DISPLAY PALABRA LINE 17 POSITION 30.

      * SE DA OPCION A SEGUIR EN EL PROGRAMA O SALIR.

           DISPLAY "QUIERES SEGUIR S/N" LINE 20 POSITION 30.
           ACCEPT LETRA1 LINE 22 POSITION 30 NO BEEP.
           IF LETRA1 = "S" OR LETRA1 = "s" GO TO COMIENZO.
```

Al ejecutarlo prueba con diferentes palabras, por ejemplo pon una palabra toda en mayúsculas y despues le dices que un determinado caracter lo ponga en minúsculas.

INSPECT palabra REPLACING LEADING car1 BY car2..: En este segundo formato se sustituyen todas las apariciones del caracter 'car1' por 'car2' hasta que se encuentre un caracter diferente de 'car1', si la palabra comienza por un caracter que no es 'car1' no se sustituirá nada. Para probar este formato coge el programa anterior y modifícalo, prueba y vé la diferencia que hay.

INSPECT palabra REPLACING FIRST car1 BY car2..: Este tercer formato cambiará la primera aparición del caracter 'car1' por el caracter 'car2'.

INSPECT palabra REPLACING CHARACTERS BY car..: En este último formato se reemplazarán todos los caracteres de 'palabra' por el caracter 'car'.

Al igual que en los formatos donde contábamos el número de apariciones de un caracter, a la hora de reemplazar se comienza por la izquierda hasta el final del campo, para controlarlo podemos usar las dos opciones siguientes.

BEFORE INITIAL car..: Se reemplazarán los caracteres encontrados antes de la aparición del caracter 'car'.

AFTER INITIAL car..: Se reemplazarán los caracteres encontrados despues de la aparición del caracter 'car'.

Estas son todas las posibles opciones y formatos de la instrucción **INSPECT**, basándote en los programas que hemos hecho a lo largo de esta instrucción haz tú otros con los que probar todos los formatos que has visto.

8.7. LA INSTRUCCION PERFORM.

PERFORM...: Esta instrucción nos vá a servir entre otras cosas para hacer más modulares nuestros programas, con ella podremos dividir un programa en varios procedimientos y despues irlos llamando a medida que nos hagan falta; también nos vá a servir para ejecutar un número determinado de veces un acto, o lo que es lo mismo, ejecutar un bucle, tenemos los siguientes formatos.

PERFORM etq...: Donde 'etq' es un nombre de etiqueta por el que identificamos un trozo de programa, el efecto de la ejecución de esta instrucción es que el programa se desviará hasta el punto indicado y ejecutará todas las instrucciones que halla a continuación hasta encontrar

otra nombre de párrafo, cuando esto ocurra el control del programa volverá al mismo sitio donde estaba antes de bifurcar.

Para que veamos como funciona esta instrucción vamos a hacer un programa, en él tendremos un menú que nos dará opción a hacer una de las cuatro operaciones aritméticas básicas, según la que elijamos haremos una suma, una resta, una multiplicación o una división, pero siempre hay una parte que se repite, la petición de los dos números que van a entrar en la operación, en lugar de pedirlos en cada una de las operaciones la petición la vamos a poner en un procedimiento aparte que llamaremos cuando nos haga falta. Abre en tu editor un archivo con el nombre de 'PRUEBA26.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA26.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  OPCION                  PIC 9.
       77  NUMERO1                 PIC 9(04).
       77  NUMERO2                 PIC 9(04).
       77  RESULTADO               PIC 9(08).
       77  TECLA                   PIC X.

       PROCEDURE DIVISION.
       COMIENZO.

      * IMPRESION DEL MENU DE OPCIONES.

           DISPLAY " M E N U " LINE 2 POSITION 34 REVERSE ERASE.
           DISPLAY "1.- SUMAR" LINE 5 POSITION 30.
           DISPLAY "2.- RESTAR" LINE 7 POSITION 30.
           DISPLAY "3.- MULTIPLICAR" LINE 9 POSITION 30.
           DISPLAY "4.- DIVIDIR" LINE 11 POSITION 30.
```

```
           DISPLAY "ELIJE OPCION.[ ]" LINE 13 POSITION 10.
       * PETICION DE LA OPCION.
        PEDIROPCION.
           ACCEPT OPCION LINE 13 POSITION 24 NO BEEP.
       * SEGUN LA OPCION PULSADA SE BIFURCA A UNO DE LOS 4 PUNTOS.
           GO TO SUMA, RESTA, MULTIPLICA, DIVIDA DEPENDING ON OPCION.
       * SI SE LLEGA AQUI ES QUE LA OPCION NO ES VALIDA.
           GO TO PEDIROPCION.
       * PROCEDIMIENTO PARA SUMAR
        SUMA.
       * PRIMERO SE LLAMA A LA RUTINA PARA PEDIR DOS NUMEROS.
           PERFORM PEDIRNUMEROS.
       * UNA VEZ QUE SE TIENE SE HACE LA OPERACION Y SE IMPRIME.
           ADD NUMERO1, NUMERO2 GIVING RESULTADO.
           DISPLAY NUMERO1 LINE 20 POSITION 15.
           DISPLAY "+" LINE 20 POSITION 19.
           DISPLAY NUMERO2 LINE 20 POSITION 20.
           DISPLAY "=" LINE 20 POSITION 24.
           DISPLAY RESULTADO LINE 20 POSITION 25.
       * UNA VEZ SE HA IMPRESO EL RESULTADO SE VA A LA ETIQUETA SEGUIR.
           GO TO SEGUIR.
       * PROCEDIMIENTO PARA RESTAR
        RESTA.
       * PRIMERO SE LLAMA A LA RUTINA PARA PEDIR DOS NUMEROS.
           PERFORM PEDIRNUMEROS.
       * UNA VEZ QUE SE TIENE SE HACE LA OPERACION Y SE IMPRIME.
```

```
           SUBTRACT NUMERO1 FROM NUMERO2 GIVING RESULTADO.
           DISPLAY NUMERO1 LINE 20 POSITION 15.
           DISPLAY "-" LINE 20 POSITION 19.
           DISPLAY NUMERO2 LINE 20 POSITION 20.
           DISPLAY "=" LINE 20 POSITION 24.
           DISPLAY RESULTADO LINE 20 POSITION 25.

       * UNA VEZ SE HA IMPRESO EL RESULTADO SE VA A LA ETIQUETA SEGUIR.

           GO TO SEGUIR.

       * PROCEDIMIENTO PARA MULTIPLICAR.

        MULTIPLICA.

       * PRIMERO SE LLAMA A LA RUTINA PARA PEDIR DOS NUMEROS.

           PERFORM PEDIRNUMEROS.

       * UNA VEZ QUE SE TIENE SE HACE LA OPERACION Y SE IMPRIME.

           MULTIPLY NUMERO1 BY NUMERO2 GIVING RESULTADO.
           DISPLAY NUMERO1 LINE 20 POSITION 15.
           DISPLAY "*" LINE 20 POSITION 19.
           DISPLAY NUMERO2 LINE 20 POSITION 20.
           DISPLAY "=" LINE 20 POSITION 24.
           DISPLAY RESULTADO LINE 20 POSITION 25.

       * UNA VEZ SE HA IMPRESO EL RESULTADO SE VA A LA ETIQUETA SEGUIR.

           GO TO SEGUIR.

       * PROCEDIMIENTO PARA DIVIDIR.

        DIVIDA.

       * PRIMERO SE LLAMA A LA RUTINA PARA PEDIR DOS NUMEROS.

           PERFORM PEDIRNUMEROS.

       * UNA VEZ QUE SE TIENE SE HACE LA OPERACION Y SE IMPRIME.

           DIVIDE NUMERO1 BY NUMERO2 GIVING RESULTADO.
           DISPLAY NUMERO1 LINE 20 POSITION 15.
           DISPLAY "/" LINE 20 POSITION 19.
           DISPLAY NUMERO2 LINE 20 POSITION 20.
           DISPLAY "=" LINE 20 POSITION 24.
           DISPLAY RESULTADO LINE 20 POSITION 25.
```

```
* UNA VEZ SE HA IMPRESO EL RESULTADO SE VA A LA ETIQUETA SEGUIR.

    GO TO SEGUIR.

 PEDIRNUMEROS.

* RUTINA PARA PEDIR LOS DOS NUMEROS.

    DISPLAY "DAME UN NUMERO" LINE 16 POSITION 20.
    ACCEPT NUMERO1 LINE 16 POSITION 40 NO BEEP.
    DISPLAY "DAME OTRO NUMERO" LINE 18 POSITION 20.
    ACCEPT NUMERO2 LINE 18 POSITION 40 NO BEEP.

 SEGUIR.

* PREGUNTA SI QUIERE SEGUIR EN EL PROGRAMA.

    DISPLAY "DESEAS SEGUIR S/N" LINE 23 POSITION 20.
    ACCEPT TECLA LINE 23 POSITION 40 NO BEEP.
    IF TECLA = "S" OR TECLA = "s" GO TO COMIENZO.
```

Cuando lo compiles ejecútalo, se te dará a elegir entre las cuatro operaciones de suma, resta, multiplicación o división, según nuestra elección se irá a una de las cuatro etiquetas, desde todas estas etiquetas se llama siempre a la rutina de pedir dos números, esta tras pedirlos vuelve al mismo lugar donde estaba y hace la operación imprimiéndola, tras lo cual se nos preguntará si queremos seguir. Cuando te pida los dos números dale letras en los dos números verás lo que ocurre, para solucionarlo puedes añadirle a los dos **ACCEPT** la opción **CONVERT** que tras pedirte los números eliminará los caracteres no numéricos.

PERFORM etq1 THRU etq2..: En este segundo formato se transferirá el control a la etiqueta 'etq1' y se ejecutará a partir de ella hasta llegar a la etiqueta 'etq2' donde normalmente tendremos la palabra Cobol **EXIT** que indicará al programa que debe volver al punto donde estaba antes de llegar a 'etq1'.

PERFORM etq nn TIMES..: En este tercer formato se transferirá el control del programa a la etiqueta 'etq' tantas veces como indique 'nn' que deberá ser una constante o variable numérica y entera.

Para probar estos últimos formatos vamos a hacer un programa, este ha de pedirnos cinco nombres y meterlos en una tabla, una vez nos los ha pedido los debe imprimir. Abre en tu editor un archivo con el nombre de 'PRUEBA27.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA27.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  NOMBRES.
           03  NOMBRE                  PIC X(20) OCCURS 5 TIMES.
       77  CONTADOR                    PIC 9.
       77  LINEA                       PIC 99.

       PROCEDURE DIVISION.
       COMIENZO.

           DISPLAY " " ERASE.
           MOVE 0 TO CONTADOR.

      * SE VA 5 VECES A LA RUTINA DONDE SE PIDE CADA UNO DE LOS NOMBRES.

           PERFORM PEDIR THRU PEDIRFIN 5 TIMES.

      * UNA VEZ LOS HA PEDIDO PONE A CERO EL CONTADOR Y SE VA 5 VECES
      * A LA RUTINA DE IMPRESION DE LOS NOMBRES.

           DISPLAY " " ERASE.
           MOVE 0 TO CONTADOR.
           PERFORM IMPRIMIR 5 TIMES.

      * UNA VEZ LOS HA IMPRESO TERMINA.

           STOP RUN.

      * RUTINA PEDIR NOMBRES.
```

```
       PEDIR.
      * LE SUMA UNO AL CONTADOR.
           ADD 1 TO CONTADOR.
      * IMPRIME EL TEXTO DE PETICION DE UN NOMBRE.
           DISPLAY "DAME EL NOMBRE" LINE 10 POSITION 30.
           DISPLAY CONTADOR LINE 10 POSITION 45.
      * BORRA EL NOMBRE QUE HUBIERA ANTES.
           DISPLAY SPACES SIZE 20 LINE 10 POSITION 47.
      * NOS PIDE EL NOMBRE.
           ACCEPT NOMBRE(CONTADOR) LINE 10 POSITION 47 PROMPT NO BEEP.
       PEDIRFIN.
      * VUELVE.
           EXIT.
      * RUTINA IMPRIMIR CAMPOS.
       IMPRIMIR.
      * LE SUMA UNO A CONTADOR.
           ADD 1 TO CONTADOR.
      * CALCULA LA LINEA DONDE IMPRIMIRA EL NOMBRE.
           COMPUTE LINEA = CONTADOR * 2 + 5.
      * IMPRIME EL NOMBRE Y VUELVE.
           DISPLAY NOMBRE(CONTADOR) LINE LINEA POSITION 30.
```

PERFORM etq UNTIL cond..: En este cuarto formato se transferirá el control a la etiqueta 'etq' mientras la condición 'cond' no se cumpla, cuando llegue a cumplirse se dejará de bifurcar y el programa seguirá su flujo normal.

Para ver en funcionamiento este cuarto formato vamos a hacer un programa que nos pida números continuamente hasta que el número que demos sea cero, en ese momento que nos imprima la suma de todos los números que le hemos ido dando. Abre en tu editor un archivo con el nombre de 'PRUEBA28.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA28.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NUMERO                  PIC 9(04) VALUE 0.
       77  RESULTADO               PIC 9(10) VALUE 0.
       77  IMPRESULTADO            PIC Z(10).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA Y MUEVE UN 1 A NUMERO.

           DISPLAY " " ERASE.
           DISPLAY "VE DANDOME NUMEROS" LINE 2 POSITION 30.
           DISPLAY "PARA TERMINAR DAME EL 0" LINE 4 POSITION 30.
           MOVE 1 TO NUMERO.

      * TRANSFIERE EL CONTROL A LA ETIQUETA PEDIR MIENTRAS QUE NUMERO
      * NO SEA IGUAL A CERO.

           PERFORM PEDIR UNTIL NUMERO = 0.

      * CUANDO LLEGUE AQUI LE HABREMOS DADO CERO, IMPRIMIMOS
      * EL RESULTADO.

           DISPLAY "SUMAN" LINE 15 POSITION 30.
           MOVE RESULTADO TO IMPRESULTADO.
           DISPLAY IMPRESULTADO LINE 15 POSITION 36.
```

```
      * TERMINA.

          STOP RUN.

      * RUTINA PEDIR NUMEROS E IRLOS SUMANDO.

        PEDIR.

            ACCEPT NUMERO LINE 10 POSITION 30 NO BEEP CONVERT PROMPT.
            ADD NUMERO TO RESULTADO.
```

PERFORM etq VARYING var FROM ini BY inc UNTIL cond..: En este quinto formato se transferirá el control del programa a la etiqueta 'etq', al mismo tiempo se irá variando el valor de 'var' comenzando por el valor 'ini' y se irá incrementando en cada transferencia en 'inc' unidades, todo se irá ejecutando hasta que se cumpla la condición 'cond'.

Para probar este formato vamos a hacer el mismo programa de pedir cinco nombres e imprimirlos para ver el trabajo que nos podemos ahorrar. Abre en tu editor un archivo con el nombre de 'PRUEBA29.CBL' y teclea lo que sigue.

```
123456789012
       A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA29.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  CONTADOR                PIC 99.
       77  LINEA                   PIC 99.
       01  NOMBRES.
           03  NOMBRE              PIC X(20) OCCURS 5 TIMES.

       PROCEDURE DIVISION.
       COMIENZO.
```

```
      * BORRA LA PANTALLA.

           DISPLAY " " ERASE.

      * LLAMA A LA RUTINA PEDIR VARIANDO LA VARIABLE CONTADOR DESDE
      * UNO INCREMENTANDO 1 HASTA QUE SEA MAYOR QUE 5.

           PERFORM PEDIR VARYING CONTADOR FROM 1 BY 1
                         UNTIL CONTADOR > 5.

      * UNA VEZ LOS HA PEDIDO LOS IMPRIME.

           DISPLAY " " ERASE.
           PERFORM IMPRIMIR VARYING CONTADOR FROM 1 BY 1
                            UNTIL CONTADOR > 5.

      * CUANDO LOS HA IMPRESO TERMINA.

           STOP RUN.

      * RUTINA PEDIR NOMBRES.

       PEDIR.

           DISPLAY "DAME EL NOMBRE" LINE 15 POSITION 30.
           DISPLAY CONTADOR LINE 15 POSITION 45.
           ACCEPT NOMBRE(CONTADOR) LINE 15 POSITION 48 NO BEEP PROMPT.

      * RUTINA IMPRIMIR NOMBRES.

       IMPRIMIR.

           COMPUTE LINEA = CONTADOR * 2 + 5.
           DISPLAY NOMBRE(CONTADOR) LINE LINEA POSITION 30.
```

Cuando lo ejecutes verás que hace exactamente lo mismo que el que hicimos antes, pero nosotros no tenemos que ir actualizando variables ni contando nada.

PERFORM etq VARYING var1 FROM ini1 BY inc1 UNTIL cond1 AFTER var2 FROM ini2 BY inc2 UNTIL cond2 AFTER ...: Este último formato funciona de una forma parecida a la anterior, la diferencia es que actualiza varias variables al mismo tiempo, con el podemo ejecutar bucles anidados, en este ejemplo 'var2' comenzará con el valor 'ini2' incrementándose en 'inc2' unidades, cuando

'cond2' se cumpla entonces la variable 'var1' se incrementará en 'inc1' y se comprobará la condición 'cond1' si esta no se ha cumplido se comenzará otra vez el bucle con 'var2' y así estará continuamente.

Para ver este último formato de **PERFORM** vamos a hacer un programa que simule un cronómetro, en él tendremos las centésimas de segundo, cada vez que estas lleguen a 99 se incrementarán en 1 los segundos y se empezará otra vez desde cero las centésimas, cuando los segundos llegue a 59 habrá que aumentar en 1 los minutos y comenzar de nuevo con los segundos a cero; si analizas esto un poco de darás cuenta de que nos hacen falta tres bucles anidados. Abre en tu editor un archivo con el nombre de 'PRUEBA30.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA30.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  MINUTOS                 PIC 99.
       77  SEGUNDOS                PIC 99.
       77  CENTESIMAS              PIC 99.

       PROCEDURE DIVISION.
      . COMIENZO.

      * SE BORRA LA PANTALLA Y SE IMPRIME UNA CABECERA.

           DISPLAY "MINUTOS" LINE 4 POSITION 15 ERASE.
           DISPLAY "SEGUNDOS" LINE 4 POSITION 30.
           DISPLAY "CENTESIMAS" LINE 4 POSITION 45.

      * SE EJECUTA EL BUCLE.
```

```
        PERFORM RELOJ VARYING MINUTOS FROM 0 BY 1 UNTIL MINUTOS > 59
                      AFTER SEGUNDOS FROM 0 BY 1 UNTIL SEGUNDOS > 59
                      AFTER CENTESIMAS FROM 0 BY 1
                      UNTIL CENTESIMAS > 98.

    * CUANDO TERMINE SE DETIENE.

        STOP RUN.

     RELOJ.

    * IMPRESION DE LAS VARIABLES QUE FORMARAN EL CRONOMETRO.

        DISPLAY MINUTOS LINE 5 POSITION 15.
        DISPLAY SEGUNDOS LINE 5 POSITION 30.
        DISPLAY CENTESIMAS LINE 5 POSITION 45.
```

La velocidad del reloj depende mucho del ordenador en el que se ejecute, pero lo importante es que dará una idea bastante buena de como funciona este último formato de la instrucción **PERFORM**.

8.8. MANEJANDO TABLAS CON LA INSTRUCCION SET.

SET..: Cuando estuvimos viendo el manejo de tablas vimos una de las opciones donde al declarar la tabla se declaraba también el índice de esa tabla, además también vimos en la cláusula **USAGE** un modo que era **INDEX**; en los dos casos no había que declarar tipo ni longitud de la variable ya que de eso se ocupaba el ordenador, pero para manejar el valor de estas variables tenemos una instrucción especial que es **SET**, ya que a estos campos no se le pueden asignar valores mediante la instrucción **MOVE** como a los campos normales. Tenemos los siguientes formatos.

SET ind TO val..: Donde 'ind' es el índice al que queremos dar valor y 'val' es el valor que le queremos dar, si 'ind' es el índice declarado de una tabla el valor 'val' puede ser una constante o variable numérica cualquiera pero si 'ind' es una variable declarada como **USAGE INDEX** 'val' ha de ser necesariamente la variable que actúa como índice de la tabla.

SET ind UP BY val..: Donde 'ind' será el índice declarado de la tabla, este formato aumenta el índice en 'val' unidades, si se sobrepasa el límite de la tabla el programa se detendrá dándonos un error de índice fuera de rango.

SET int DOWN BY val..: Donde 'ind' será el índice declarado de la tabla, este formato disminuye el índice en 'val' unidades, si se llega a un número menor que uno el programa se detendrá dándonos un error de índice fuera de rango.

Una vez hemos visto los distintos formatos que tiene esta instrucción vamos a hacer un programa que nos ilustre con su utilización. El programa vá a ser una especie de agenda donde vamos a tener como máximo a 100 personas con sus respectivas direcciones y teléfonos. Abre en tu editor un archivo con el nombre de 'PRUEBA31.CBL' y teclea lo que sigue.

```
123456789012
       A   B

        IDENTIFICATION DIVISION.
        PROGRAM-ID. PRUEBA31.

        ENVIRONMENT DIVISION.
        CONFIGURATION SECTION.
        SOURCE-COMPUTER. PC.
        OBJECT-COMPUTER. PC.

        DATA DIVISION.
        WORKING-STORAGE SECTION.

      * TABLA DONDE SE VAN A ALMACENAR LOS DATOS, EL INDICE SE
      * LLAMARA INDICE.

        01  TABLAS.
            03  TABLA OCCURS 100 INDEXED BY INDICE.
                05  NOMBRE          PIC X(25).
                05  DIRECCION       PIC X(25).
                05  TELEFONO        PIC 9(07).

      * VARIABLES PARA PEDIR OPCION Y CODIGO.

        77  OPCION                  PIC X.
        77  CODIGO                  PIC 99.

      * MARGENES MAXIMO Y MINIMO DE LA TABLA DE DATOS.

        77  MINIMO                  USAGE IS INDEX.
        77  MAXIMO                  USAGE IS INDEX.
```

```
       PROCEDURE DIVISION.
       COMIENZO.

      * SE PONE EL MARGEN MINIMO EN CERO Y EL MAXIMO EN 99.
      * PRIMERO METEMOS EL VALOR EN INDICE Y DE HAY LO PASAMOS
      * A LAS VARIABLES MAXIMO Y MINIMO.

           SET INDICE TO 0.
           SET MINIMO TO INDICE.
           SET INDICE TO 99.
           SET MAXIMO TO INDICE.

      * IMPRESION DE LA PANTALLA CON LOS DISTINTOS CAMPOS.

           DISPLAY " F I C H E R O   D E   A M I G O S "
                   LINE 2 POSITION 25 ERASE REVERSE.
           DISPLAY "CODIGO.....:" LINE 5 POSITION 20.
           DISPLAY "NOMBRE.....:" LINE 7 POSITION 20.
           DISPLAY "DIRECCION...:" LINE 8 POSITION 20.
           DISPLAY "TELEFONO....:" LINE 9 POSITION 20.
           DISPLAY "A=ALTAS/B=BAJAS/D=ANTERIOR/S=SIGUIENTE..[ ]"
                   LINE 14 POSITION 10.

       PEDIRCODIGO.

      * PIDE EL CODIGO.

           ACCEPT CODIGO LINE 5 POSITION 33 NO BEEP.

      * COMPRUEBA SI EL CODIGO ES NUMERICO SINO VUELVE A PEDIRLO.

           IF CODIGO NOT NUMERIC GO TO PEDIRCODIGO.

      * SI EL CODIGO ES CERO SE VA A LA ETIQUETA SALIR.

           IF CODIGO = 0 GO TO SALIR.

      * EN CASO CONTRARIO GUARDA EL CODIGO PEDIDO EN INDICE.

           SET INDICE TO CODIGO.

      * SI EL NOMBRE QUE HEMOS PEDIDO NO ES IGUAL A ESPACIOS
      * QUIERE DECIR QUE EXISTE, SI ES ASI NOS VAMOS A EXISTE.

           IF NOMBRE(INDICE) NOT = SPACES GO TO EXISTE.

       NOEXISTE.
```

```
      * EN CASO CONTRARIO PEDIMOS LOS DEMAS CAMPOS.

           ACCEPT NOMBRE(INDICE) LINE 7 POSITION 33
                  PROMPT NO BEEP.
           ACCEPT DIRECCION(INDICE) LINE 8 POSITION 33
                  PROMPT NO BEEP.
           ACCEPT TELEFONO(INDICE) LINE 9 POSITION 33
                  PROMPT NO BEEP CONVERT.
           GO TO PEDIROPCION.

       EXISTE.

      * SI EXISTIA SE IMPRIMEN LOS CAMPOS Y SE LLEGA A PEDIROPCION.

           DISPLAY NOMBRE(INDICE) LINE 7 POSITION 33.
           DISPLAY DIRECCION(INDICE) LINE 8 POSITION 33.
           DISPLAY TELEFONO(INDICE) LINE 9 POSITION 33.

       PEDIROPCION.

      * PIDE LA OPCION Y SEGUN ESTA SE VA A UNA DE LAS
      * CUATRO ETIQUETAS.

           ACCEPT OPCION LINE 14 POSITION 51 NO BEEP.
           IF OPCION = "A" OR OPCION = "a" GO TO ALTAS.
           IF OPCION = "B" OR OPCION = "b" GO TO BAJAS.
           IF OPCION = "D" OR OPCION = "d" GO TO ANTERIOR.
           IF OPCION = "S" OR OPCION = "s" GO TO SIGUIENTE.

      * SI SE LLEGA AQUI QUIERE DECIR QUE LA OPCION ES INVALIDA
      * ASI QUE SE VUELVE A PEDIROPCION.

           GO TO PEDIROPCION.

       ALTAS.

      * EN LA RUTINA DE ALTAS SIMPLEMENTE SE LLAMA A LA RUTINA
      * DE BORRAR Y DESPUES VUELVE A PEDIRCODIGO.

           PERFORM BORRAR.
           GO TO PEDIRCODIGO.

       BAJAS.

      * SI SE QUERIA DAR DE BAJA SE METEN ESPACIOS EN EL NOMBRE
      * Y SE LLAMA A LA RUTINA DE BORRAR Y SE VUELVE A PEDIRCODIGO.
```

```
           MOVE SPACES TO NOMBRE(INDICE).
           PERFORM BORRAR.
           GO TO PEDIRCODIGO.

       ANTERIOR.

      * ESTA RUTINA IMPRIME EL REGISTRO ANTERIOR.
      * SUBE EN UNO EL INDICE.

           SET INDICE DOWN BY 1.

      * COMPRUEBA SI EL INDICE YA HA LLEGADO A SER EL MINIMO
      * DE SER ASI SE VA A LA ETIQUETA NOENCONTRADO.

           IF INDICE = MINIMO GO TO NOENCONTRADO.

      * SI EL NOMBRE QUE SE HA ENCONTRADO ES IGUAL A ESPACIOS
      * SE SIGUE BUSCANDO.

           IF NOMBRE(INDICE) = SPACES GO TO ANTERIOR.

      * SI SE ENCUENTRA SE LLAMA A LA RUTINA EXISTE QUE
      * IMPRIMIRA LOS CAMPOS.

           PERFORM EXISTE.

      * IMPRIME EL CODIGO

           SET CODIGO TO INDICE.
           DISPLAY CODIGO LINE 5 POSITION 33.

      * VUELVE A PEDIROPCION.

           GO TO PEDIROPCION.

       SIGUIENTE.

      * ESTA RUTINA IMPRIME EL SIGUIENTE REGISTRO.
      * AUMENTA EN UNO EL INDICE.

           SET INDICE UP BY 1.

      * COMPRUEBA SI HA LLEGADO AL MAXIMO, DE SER ASI SE VA
      * A NOENCONTRADO.

           IF INDICE = MAXIMO GO TO NOENCONTRADO.
```

```
* SI EL NOMBRE ENCONTRADO ES IGUAL A ESPACIOS SE SIGUE BUSCANDO.

    IF NOMBRE(INDICE) = SPACES GO TO SIGUIENTE.

* SI SE ENCUENTRA SE LLAMA A LA ETIQUETA EXISTE PARA IMPRIMIR
* LOS CAMPOS.

    PERFORM EXISTE.

* SE IMPRIME EL CODIGO.

    SET CODIGO TO INDICE.
    DISPLAY CODIGO LINE 5 POSITION 33.

* SE VUELVE A PEDIROPCION.

    GO TO PEDIROPCION.

 BORRAR.

* RUTINA DE BORRADO DE LOS CAMPOS DE LA PANTALLA.

    DISPLAY SPACES LINE 5 POSITION 33 SIZE 2.
    DISPLAY SPACES LINE 7 POSITION 33 SIZE 25.
    DISPLAY SPACES LINE 8 POSITION 33 SIZE 25.
    DISPLAY SPACES LINE 9 POSITION 33 SIZE 9.

 NOENCONTRADO.

* ESTA RUTINA SE UTILIZA CUANDO SE QUERIA IMPRIMIR EL
* ANTERIOR O SIGUIENTE Y NO EXISTIA.
* LLAMA A LA RUTINA BORRAR PARA LIMPIAR LA PANTALLA.

    PERFORM BORRAR.

* NO IMPRIME EL MENSAJE DE NO HAY MAS.

    DISPLAY "NO HAY MAS" LINE 5 POSITION 50 REVERSE.

* LLAMA A LA ETIQUETA ESPERA 500 VECES

    PERFORM ESPERA 500 TIMES.

* BORRA EL MENSAJE Y VUELVE A PEDIRCODIGO.

    DISPLAY SPACES SIZE 10 LINE 5 POSITION 50.
    GO TO PEDIRCODIGO.
```

```
       SALIR.

      * SI HEMOS PULSADO EL CODIGO CERO SE LLEGARA A ESTA ETIQUETA.
      * SE BORRA LA PANTALLA Y SE DETIENE EL PROGRAMA.

           DISPLAY " " ERASE.
           STOP RUN.

       ESPERA.

      * ESTA ETIQUETA SE UTILIZA PARA HACER UNA ESPERA, NO HACE NADA.

           MOVE OPCION TO OPCION.
```

8.9. MODULACION DE PROGRAMAS.

A veces se nos hace necesario desde un programa utilizar algo que puede hacer otro, esto el Cobol lo puede hacer de tal forma que cuando lo llamamos se queda ya en la memoria en espera de ser llamado, con lo cual la repidez es bastante grande. Con las instrucciones que seguidamente tenemos podremos tanto llamar a un programa como desmontarlo de la memoria.

8.9.1 LA INSTRUCCION CALL.

CALL..: Con esta instrucción podremos llamar desde un programa a otro programa, para ello haremos uso del siguiente formato.

> **CALL "nomprog" [USING var, var,..]..:** Donde 'nomprog' es el nombre que el programa que deseamos llamar recibe en el disco, si queremos pasar valores del programa actual al programa que va a ser llamado tendremos que especificar cada una de las variables a continuación de la cláusula **USING**.
>
> **ON OVERFLOW instr..:** Cada vez que se hace una llamada a un programa con la instrucción **CALL** este se queda en la memoria para un posterior uso, en un momento dado puede ser que la memoria se desborde, con la inclusión de esta opción le podremos indicar que haga una determinada cosa en caso de que ocurra.

Si no se desean pasar valores de un programa a otro bastará con especificar el nombre del programa al que queremos llamar, al final de este programa habremos de poner como instrucción única en la etiqueta la palabra Cobol **EXIT PROGRAM**, esto hará que el

control de ejecución vuelva al programa que llamó y siga por el mismo punto.

Para que veas como funciona esta instrucción vamos a hacer dos pequeños programas, uno imprimirá un saludo y llamará al otro programa, este esperará a que pulses una tecla y despues volverá al primero. Abre en tu editor un archivo con el nombre de 'PRUEBA32.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA32.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.

       PROCEDURE DIVISION.
       COMIENZO.

      * IMPRIME EL SALUDO.

           DISPLAY "HOLA, QUE TAL" LINE 5 POSITION 35 ERASE.

      * LLAMA AL SEGUNDO PROGRAMA.

           CALL "PRUEBA33".

      * CUANDO VUELVA IMPRIMIRA UN MENSAJE.

           DISPLAY "YA VOLVI" LINE 10 POSITION 30.
           STOP RUN.
```

Grábalo y abre otro archivo con el nombre de 'PRUEBA33.CBL' para teclear lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA33.
```

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  TECLA                    PIC X.

       PROCEDURE DIVISION.
       COMIENZO.

      * ESPERA LA PULSACION DE UNA TECLA.

           DISPLAY "PULSA UNA TECLA" LINE 6 POSITION 30.
           ACCEPT TECLA OFF NO BEEP.

      * DESPUES VUELVE AL PRIMER PROGRAMA.

       VOLVER.

           EXIT PROGRAM.
```

Cuando lo ejecutes verás como se imprime el saludo del primer programa, despues te pide que pulses una tecla estando en el segundo programa y despues vuelve otra vez al primero imprimiendo el texto indicado.

Cuando se quieren pasar valores de un programa a otro ademas de usar la cláusula **USING** en el programa que vá a llamar, al que vá a recibir hay que añadirle una división más en la **DATA**, esta es la **LINKAGE SECTION**, ¿te acuerdas de ella?, funciona exactamente igual que la **WORKING-STORAGE** sólo que en ella se declaran los campos que van a pasar de un programa a otro, estos se han de declarar al mismo nivel y con la misma longitud y tipo con que estaban declarados en el primer programa; si además estas variables las vamos a usar a lo largo de la **PROCEDURE** tendremos que añadirle a esta el formato **USING** var, var.. para especificar todas las variables que ván a pasar de un programa a otro.

Para ver el pase de variables de un programa a otro vamos a hacer otros dos pequeños programas, el primero nos ha de pedir nuestro nombre, una vez pedido llamará al segundo quien usando este nombre nos saludará. Abre en tu editor un archivo con el nombre de 'PRUEBA34.CBL' y teclea lo que a continuación tienes.

```
123456789012
      A   B
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA34.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  NOMBRE                  PIC X(20).

       PROCEDURE DIVISION.
       COMIENZO.

      * PIDE EL NOMBRE.

           DISPLAY "DAME TU NOMBRE" LINE 2 POSITION 30.
           ACCEPT NOMBRE LINE 2 POSITION 45 NO BEEP PROMPT.

      * LLAMA AL PROGRAMA PRUEBA35.CBL CON EL NOMBRE.

           CALL "PRUEBA35" USING NOMBRE.

      * CUANDO VUELVE IMPRIMIMOS UN MENSAJE.

           DISPLAY "HE VUELTO" LINE 15 POSITION 30.
```

Grábalo y abre otro archivo con el nombre 'PRUEBA35.CBL' para teclear lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA35.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.
```

```
       DATA DIVISION.
       LINKAGE SECTION.

       77  NOMBRE                  PIC X(20).

       PROCEDURE DIVISION USING NOMBRE.
       COMIENZO.

      * SE IMPRIME EL SALUDO USANDO EL NOMBRE.

           DISPLAY "HOLA" LINE 10 POSITION 30.
           DISPLAY NOMBRE LINE 10 POSITION 36.

      * VUELVE AL PROGRAMA ANTERIOR.

       VOLVER.

           EXIT PROGRAM.
```

Cuando los compiles los dos ejecuta 'PRUEBA34', este te pedirá un nombre y llamará a 'PRUEBA35' que te saludará, despues volverá al primero e imprimirá un texto para que veamos que ha vuelto.

8.9.2 LA INSTRUCCION CANCEL.

CANCEL..: Esta instrucción nos servirá para liberar areas de memoria que han sido ocupadas por programas llamados con la instrucción **CALL**. Para ello tenemos el siguiente formato.

CANCEL iden..: Donde iden puede ser un literal o una variable con el nombre de programa a excluir de la memoria. Esta instrucción se puede utilizar cuando al llamar un programa se llene la memoria.

9.- UTILIZACION DE FICHEROS.

A lo largo de lo que llevamos visto del Cobol hemos ido dejando una serie de partes sin ver como la **INPUT-OUTPUT SECTION**, la **FILE SECTION** y algunas cosas más que vamos a ver ahora.

9.1. LA INPUT-OUTPUT SECTION.

A la hora de utilizar ficheros en nuestros programas lo

primero que tendremos que hacer será declarar esos ficheros, para ello tenemos la INPUT-OUTPUT SECTION que tiene el siguiente formato.

```
123456789012
      A  B

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT nomlog ASSIGN TO tipo, "nomext";
          [ORGANIZATION IS org];
          [ACCESS MODE IS modo];
          [RELATIVE KEY IS cod];
          [RECORD KEY IS clv];
          [ALTERNATE RECORD KEY IS clv1]; [WITH DUPLICATES]
           FILE STATUS IS campo.
      [I-O-CONTROL.]
          [SAME RECORD AREAS FOR fic1, fic2..,]
```

Este sería el formato general de la INPUT-OUTPUT SECTION, vamos a pasar ahora a ver cada una de las partes anteriores, para que nos sirven cada una.

FILE-CONTROL..: Nombre de párrafo obligatorio siempre que exista la INPUT-OUTPUT SECTION, le indica al programa que lo que viene a continuación es el control de ficheros.

SELECT nomlog..: Con esta palabra Cobol le indicamos al programa que empezamos la definición de un fichero, 'nomlog' es el nombre que le vamos a dar al fichero dentro de nuestro programa.

ASSIGN TO tipo, "nomext"..: Con esta palabra le indicamos como vamos a utilizar el fichero, 'tipo' puede ser INPUT si el fichero vá a ser de lectura secuencial, OUTPUT si el fichero vá a ser de escritura secuencial, RANDOM nos dá opción a abrir el fichero como necesitemos, en modo directo, secuencial o indexado, y por último PRINT asigna ese fichero para salida por impresora; '"nomext"' es el nombre que vá a tener el fichero en el disco, si el fichero se ha asignado para impresora el nombre tendrá que ser '"PRINTER"'.

ORGANIZATION IS org..: Con esta cláusula indicamos al programa la organización de los registros del fichero que estamos declarando, esta puede ser SEQUENTIAL si la

organización vá a ser secuencial, **RELATIVE** si el fichero vá a ser directo o **INDEXED** si el fichero vá a ser indexado. Cuando el fichero vá a ser secuencial se puede omitir la cláusula **ORGANIZATION IS**.

ACCESS MODE IS modo..: Con esta palabra Cobol le indicamos al programa como a acceder a ese fichero, puede ser **SEQUENTIAL** si vamos a acceder secuencialmente, **RANDOM** si vamos a acceder aleatoriamente o **DYNAMIC** cuando no queremos especificar el modo en concreto con el que vamos a acceder, al utilizar este último modo a la hora de trabajar con el fichero podremos acceder a él como nos convenga. Cuando el fichero es secuencial este párrafo se puede omitir.

RELATIVE KEY IS cod..: Este párrafo solamente se utiliza cuando el fichero que estamos declarando vá a ser de organización directa, con él especificaremos que campo nos vá a servir de código para el fichero directo, 'cod' que será ese código deberá estar declarado en la **WORKING-STORAGE** como un campo numérico sin signo, a la hora de leer o escribir en el fichero en este campo deberemos especificar el número de registro.

RECORD KEY IS clv..: Este párrafo sólo se utiliza cuando el fichero que estamos declarando vá a ser de organización indexada, con él le indicamos que 'clv' vá a ser la clave que se vá a utilizar para escribir y leer en el fichero, 'clv' tendrá que estar declarado como un campo alfanumérico y como parte del registro de este fichero en la **FILE SECTION**.

ALTERNATE RECORD KEY IS clv1..: Al igual que el anterior este párrafo sólo se utiliza cuando el fichero vá a ser indexado, nos vá a servir para declarar otras claves alternativas para poder acceder al fichero, por ejemplo en un fichero de clientes nos puede interesar acceder por su nombre y por su D.N.I., uno de estos campos sería la clave principal y otro una clave alternativa. 'clv1' ha de estar declarado como campo alfanumérico y como parte del registro del fichero, además no podrá ser parte de la clave principal.

WITH DUPLICATES..: Este párrafo sólo aparece en la definición de un fichero cuando aparece el anterior, se utiliza para avisarle al programa de que de una determinada clave secundaria puede haber duplicaciones, por ejemplo si tenemos como clave secundaria el nombre

de cliente este se podrá duplicar, ya que puede haber dos personas con el mismo nombre, en este caso al definir la clave secundaria añadiremos este párrafo. Si no se ha avisado de que puede haber duplicaciones de claves y durante la ejecución del programa se intenta duplicar una clave se nos devolverá un error.

FILE STATUS IS campo..: Este párrafo, que es opcional en toda declaración de ficheros, nos vá a servir para indicarle al programa en que campo debe depositar el código de error sufrido en un fichero en caso de haberlo, 'campo' deberá estar declarado en la **WORKING** como un campo alfanumérico de dos dígitos.

9.2. LA I-O-CONTROL.

Cada vez que se declara y utiliza un fichero el programa reserva un area de memoria para ese fichero, pero si no vamos a utilizar al mismo tiempo dos ficheros determinados podemos indicarle al programa que esos dos o más ficheros utilicen el mismo area de memoria, para ello tenemos el párrafo **I-O-CONTROL** que es opcional y que tiene el siguiente formato.

SAME RECORD AREAS FOR fic1, fic2....: Con este párrafo en la **I-O-CONTROL** le indicamos al programa que ficheros deben de utilizar el mismo area de memoria, para ello especificaremos como 'fic1' y 'fic2' los nombres lógicos de los ficheros que vayan a utilizar ese area de memoria. Se pueden especificar más de una cláusula **SAME RECORD** para agrupar varios ficheros en distintas areas de memoria, al utilizar esto hemos de tener en cuenta que no podremos utilizar al mismo tiempo dos ficheros que utilicen el mismo area de memoria.

Para ver la utilización de todo lo visto vamos a hacer varios ejemplos practicos. En el primero vamos a hacer la definición de un fichero que se vá a llamar 'CLIENTES' y en el disco se vá a llamar 'FGECLI.DAT', este fichero vá a ser secuencial y vá tener como **FILE STATUS** al campo 'ERRORES'.

```
123456789012
      A   B

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
```

```
SELECT CLIENTES;
       ASSIGN TO RANDOM, "FGECLI.DAT";
       FILE STATUS IS ERRORES.
```

Como puedes ver especificamos solamente el nombre lógico y físico que vá a tener el fichero y su **FILE STATUS**, al ser un fichero secuencial todos los demás párrafos no son obligatorios. En la cláusula **ASSIGN** hemos puesto 'RANDOM' cuando el fichero es secuencial, esto nos dá opción a utilizar el fichero como de lectura o escritura según nuestras necesidades.

Vamos a declarar ahora un fichero con el mismo nombre lógico y físico, el mismo **FILE STATUS** y poniéndole como código el campo 'NUM-CLI'.

```
123456789012
      A   B

      INPUT-OUTPUT SECTION.
      FILE-CONTROL.

          SELECT CLIENTES;
                 ASSIGN TO RANDOM "FGECLI.DAT";
                 ORGANIZATION IS RELATIVE;
                 ACCESS MODE IS DYNAMIC;
                 RELATIVE KEY IS NUM-CLI;
                 FILE STATUS IS ERRORES.
```

Puedes compararlo con la anterior declaración y verás que le hemos añadido el párrafo **ORGANIZATION IS RELATIVE** para indicarle al programa que el fichero es directo, en modo de acceso hemos puesto **DYNAMIC** con lo cual podremos abrir este fichero como nos convenga en cada momento, y despues hemos especificado como campo clave a 'NUM-CLI', este campo estará en la **WORKING-STORAGE** declarado como campo numérico sin signo, por último declaramos el **FILE STATUS** igual que en la anterior definición.

Vamos ahora a definir un fichero indexado con los mismos nombres y poniendo como clave el campo 'NOM-CLI'.

```
123456789012
      A   B

      INPUT-OUTPUT SECTION.
      FILE-CONTROL.

          SELECT CLIENTES;
                 ASSIGN TO RANDOM "FGECLI.DAT";
```

```
                ORGANIZATION IS INDEXED;
                ACCESS MODE IS DYNAMIC;
                RECORD KEY IS NOM-CLI;
                FILE STATUS IS ERRORES.
```

Hemos cambiado la organización y hemos puesto **INDEXED** para indicar que el fichero será indexado, la clave será 'NOM-CLI' que deberá estar declarado como parte del registro del fichero.

Vamos a coger la definición del anterior fichero indexado y vamos a indicarle que además de tener 'NOM-CLI' como clave vamos a tener una clave secundaria que vá a ser 'POB-CLI' y que esta vá a tener duplicaciones.

```
123456789012
      A   B

      INPUT-OUTPUT SECTION.
      FILE-CONTROL.

          SELECT CLIENTES;
                ASSIGN TO RANDOM "FGECLI.DAT";
                ORGANIZATION IS INDEXED;
                ACCESS MODE IS DYNAMIC;
                RECORD KEY IS NOM-CLI;
                ALTERNATE RECORD KEY IS POB-CLI;
                FILE STATUS IS ERRORES.
```

Por último vamos a hacer una declaración completa de varios ficheros y sus campos en la **WORKING-STORAGE**, vamos a tener un fichero secuencial de nombre 'PARAMETROS' y que vá a tener en el disco el nombre de 'FGEPAR.DAT', también vamos a tener un fichero secuencial que se llama 'PANTALLA' y en el disco 'FGEPAN.DAT', estos dos ficheros van a utilizar el mismo area de memoria, ya que nunca se van a utilizar los dos al mismo tiempo; vamos a tener además un fichero directo llamado 'CLIENTES' y 'FGECLI.DAT' en el disco, este fichero vá a tener como código el campo 'COD-CLI', por último tendremos un fichero indexado de nombre 'HISTORICO' que el disco tendrá el nombre de 'FGEHIS.DAT', este fichero tendrá como clave un campo llamado 'CLV-HIS'; estos dós últimos ficheros también utilizarán el mismo area de memoria. Todos los ficheros declarados utilizarán el mismo status que se llamará 'ERRORES'. Vamos a poner manos a la obra y a declarar lo expuesto.

```
123456789012
      A   B

      IDENTIFICATION DIVISION.
      PROGRAM-ID. CONTA.
```

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT PARAMETROS;
                  ASSIGN TO RANDOM, "FGEPAR.DAT";
                  FILE STATUS IS ERRORES.

           SELECT PANTALLA;
                  ASSIGN TO RANDOM, "FGEPAN.DAT";
                  FILE STATUS IS ERRORES.

           SELECT CLIENTES;
                  ASSIGN TO RANDOM, "FGECLI.DAT";
                  ORGANIZATION IS RELATIVE;
                  ACCESS MODE IS DYNAMIC;
                  RELATIVE KEY IS COD-CLI;
                  FILE STATUS IS ERRORES.

           SELECT HISTORICO;
                  ASSIGN TO RANDOM, "FGEHIS.DAT";
                  ORGANIZATION IS INDEXED;
                  ACCESS MODE IS DYNAMIC;
                  RECORD KEY IS CLV-HIS;
                  FILE STATUS IS ERRORES.

       I-O-CONTROL.

           SAME RECORD AREAS FOR PARAMEROS, PANTALLA.
           SAME RECORD AREAS FOR CLIENTES, HISTORICO.

       DATA DIVISION.
       FILE SECTION.

       ......

       WORKING-STORAGE SECTION.

       77  ERRORES                    PIC X(02).
       77  COD-CLI                    PIC 9(03).

       PROCEDURE DIVISION.

       .....
```

Con lo que hemos visto anteriormente te será fácil comprender la definición que hemos hecho, si te queda alguna duda vuelve atrás y consulta lo que no entiendas.

9.3. LA FILE SECTION.

El siguiente paso tras declarar el fichero sería declarar su registro, para ello tenemos la **FILE SECTION** que tiene el siguiente formato.

```
123456789012
      A  B

       DATA DIVISION.
       FILE SECTION.

       FD  nomfic;
           [BLOCK CONTAINS n1 TO n2 RECORDS;]
           [RECORD CONTAINS n1 TO n2 CHARACTERS;]
            LABEL RECORD IS mod;
           [DATA RECORD IS nomreg;]
           [CODE-SET IS nomcam.]
       01  nomreg.

           ...
```

Este sería el formato general de la **FILE SECTION**, a continuación vamos a ver uno a uno los párrafos anteriores.

FD nomfic..: Donde 'nomfic' es el nombre lógico del fichero del cual vamos a definir el registro, este fichero tuvo que ser declarado con anterioridad en la **INPUT-OUTPUT SECTION**; con este párrafo le indicamos al programa que empezamos la definición del registro de ese fichero.

BLOCK CONTAINS n1 TO n2 RECORDS..: Este párrafo sólo se usa cuando la unidad de almacenamiento de los ficheros vá a ser una unidad de cinta magnética, en ese caso con este párrafo le indicamos cuandos registros de fichero o lógicos formarán un registro físico en la cinta.

RECORD CONTAINS n1 TO n2 CHARACTERS..: Con este párrafo se declara la longitud que vá a tener el registro del fichero que estamos definiendo, 'n1' y 'n2' serán dos cantidades entre las cuales estará la longitud, lo

normal es declarar una longitud fija. La longitud que aquí se especifica se comprobará al compilar con la longitud del registro, si no coinciden se nos devolverá un aviso al final de la compilación.

LABEL RECORD IS mod..: Con este párrafo le indicamos al programa que tipo de registro es el del fichero a utilizar, los tipos posibles son **STANDARD** cuando el registro se comprueba una vez escrito, este modo se usa siempre que el fichero este soportado en disco, y **OMITTED** cuando el registro no se ha de comprobar, se usa en el caso de que el fichero esté soportado en impresora. Esta cláusula es de aparición obligatoria en la definición del registro.

DATA RECORD IS nomreg..: Con este párrafo le indicaremos al programa que nombre recibe el grupo que vá a formar el registro, este nombre debe ser el que se pone a nivel 01.

CODE-SET IS nomcam..: Esta cláusula es inactiva en RM-COBOL, su especificación no tiene ningún efecto, su cometido sería especificar en que fuente de caracteres se ha de escribir el fichero en curso.

A partir de este punto habría que especificar a nivel 01 el nombre de registro y a continuación todo el registro del fichero.

Con lo visto anteriormente vamos a hacer la definición de un fichero directo llamado 'CLIENTES' y que en el disco tiene el nombre de 'FGECLI.DAT', este fichero tendrá los campos 'NOM-CLI', 'DIR-CLI', 'POB-CLI' y 'TLF-CLI', el código de este fichero será 'COD-CLI'.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. CLIENTES.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
```

```
           SELECT CLIENTES;
                  ASSIGN TO RANDOM, "FGECLI.DAT";
                  ORGANIZATION IS RELATIVE;
                  ACCESS MODE IS DYNAMIC;
                  RELATIVE KEY IS COD-CLI;
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  CLIENTES;
           RECORD CONTAINS 85 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS DATOS.

       01  DATOS.
           03  NOM-CLI                 PIC  X(25).
           03  DIR-CLI                 PIC  X(25).
           03  POB-CLI                 PIC  X(25).
           03  TLF-CLI                 PIC  9(10).

      WORKING-STORAGE SECTION.

      77  ERRORES                      PIC  X(02).
      77  COD-CLI                      PIC  9(04).
```

En este caso al ser el fichero directo la clave se ha de declarar en la **WORKING-STORAGE**, a continuación vamos a declarar otro fichero con los mismos nombres pero la clave vá a ser el campo 'NOM-CLI'.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. CLIENTES.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT CLIENTES;
                  ASSIGN TO RANDOM, "FGECLI.DAT";
                  ORGANIZATION IS INDEXED;
```

```
                ACCESS MODE IS DYNAMIC;
                RECORD KEY IS NOM-CLI;
                FILE STATUS IS ERRORES.

        DATA DIVISION.
        FILE SECTION.

        FD  CLIENTES;
            RECORD CONTAINS 85 CHARACTERS;
            LABEL RECORD IS STANDARD;
            DATA RECORD IS DATOS.

        01  DATOS.
            03  NOM-CLI                PIC  X(25).
            03  DIR-CLI                PIC  X(25).
            03  POB-CLI                PIC  X(25).
            03  TLF-CLI                PIC  9(10).

        WORKING-STORAGE SECTION.

        77  ERRORES                    PIC  X(02).
```

Como ves es exactamente igual, la diferencia está en que la clave se declara como parte del fichero.

10.- INSTRUCCIONES RELATIVAS A FICHEROS.

Una vez vista la declaración de los ficheros y sus registros vamos a comenzar a ver las instrucciones que nos ván a servir para tratar con ese fichero; con todo fichero se pueden hacer cuatro operaciones básicas, abrir ese fichero, escribir y leer en ese fichero y por último cerrarlo.

10.1. ABRIENDO FICHEROS CON LA INSTRUCCION OPEN.

OPEN..: Esta instrucción nos vá a servir para abrir un cierto fichero, este habrá sido declarado con anterioridad así como su registro; tiene el siguiente formato.

OPEN mod nomfic WITH LOCK, WITH NO REWIND..: Donde 'mod' es el modo en el que se vá a abrir el fichero, podrá ser: **OUTPUT** si el fichero vá a ser secuencial de escritura, la apertura de un fichero en este modo

implica su creación; **INPUT** si el fichero vá a ser secuencial de lectura, este modo de apertura implica que el fichero ya exista, de lo contrario se nos devolverá un error; **I-O** si el fichero vá a ser de acceso aleatorio, bién directo o indexado, la apertura de un fichero en este modo implica que ese fichero ya exista, de lo contrario se nos devolverá un error; **EXTEND** se utiliza cuando se le quieren añadir datos a un fichero secuencial, hemos visto que si lo abrimos en modo de escritura, **OUTPUT**, el fichero se vuelve a crear, por lo cual para ampliar ese fichero se utiliza este modo; 'nomfic' será el nombre lógico del fichero que deseamos abrir, el modo de apertura del fichero tendrá que estar de acuerdo con el modo en que se declaró el fichero. El párrafo **WITH LOCK**, que es opcional, se utiliza cuando se trabaja en multipuesto para bloquear ese fichero mientras se está utilizando para que otra estación no pueda acceder a él hasta que se halla terminado. El párrafo **WITH NO REWIND**, que al igual que el anterior es opcional, se utiliza en el caso de las cintas magnéticas para que esta no se rebobine al abrir el fichero, también se puede utilizar en el caso de las impresoras para que no efectúen un salto de página al abrir el fichero que las vá a utilizar.

Para ver la utilización de algún formato de los vistos vamos a hacer un programa donde se declaren dos ficheros, los dos secuenciales, uno se llamará 'PARAMETROS' y tendrá en el disco el nombre de 'FGEPAR.DAT', este fichero lo abriremos en modo escritura, con lo cual al abrirlo se creará; el segundo se llamará 'PANTALLA' y tendrá el nombre físico de 'FGEPAN.DAT', lo abriremos en modo de lectura secuencial, con lo cual al no existir en el disco, se nos devolverá un error; el primer fichero tendrá los siguientes campos en su registro: 'APU-PAR' con tres dígitos numéricos, 'ASI-PAR' con dos dígitos numéricos, 'LIN-PAR' con dos dígitos numéricos y 'CUE-PAR' con diez caracteres alfanuméricos; el segundo tendrá una tabla de 24 elementos con 80 caracteres alfanuméricos cada uno.

Abre en tu editor un archivo con el nombre de 'PRUEBA36.CBL' y teclea lo que sigue.

```
123456789012
      A   B
```

INSTRUCCIONES RELATIVAS A FICHEROS

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA36.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT PARAMETROS;
                  ASSIGN TO RANDOM, "FGEPAR.DAT";
                  FILE STATUS IS ERRORES.

           SELECT PANTALLA;
                  ASSIGN TO RANDOM, "FGEPAN.DAT";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  PARAMETROS;
           RECORD CONTAINS 17 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-PAR.

       01  FIC-PAR.
           03  APU-PAR                    PIC 9(03).
           03  ASI-PAR                    PIC 9(02).
           03  LIN-PAR                    PIC 9(02).
           03  CUE-PAR                    PIC X(10).

       FD  PANTALLA;
           RECORD CONTAINS 1920 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-PAN.

       01  FIC-PAN.
           03  FIC-PAN                    PIC X(80) OCCURS 24 TIMES.

       WORKING-STORAGE SECTION.

       77  ERRORES                        PIC X(02).

       PROCEDURE DIVISION.
       COMIENZO.
```

```
      * SE IMPRIME UN MENSAJE Y SE ABRE EL ARCHIVO PARAMETROS EN
      * MODO DE ESCRITURA SECUENCIAL.

           DISPLAY "VOY A ABRIR EL FICHERO DE PARAMETROS"
                   LINE 6 POSITION 25 REVERSE ERASE.

           OPEN OUTPUT PARAMETROS.

      * SE IMPRIME UN SEGUNDO MENSAJE Y SE ABRE EL ARCHIVO PANTALLA
      * EN MODO LECTURA SECUENCIAL, AL NO EXISTIR DARA UN ERROR.

           DISPLAY "VOY A ABRIR EL FICHERO DE PANTALLA"
                   LINE 10 POSITION 25 REVERSE.

           OPEN INPUT PANTALLA.
```

Al ejecutar el programa añade tras el nombre un espacio y una 'K', con esto podrás ver el error que dará el programa, esto se debe a que el fichero que intenta abrir no existe y por lo tanto no puede, lo mismo hubiera ocurrido si el fichero hubiera sido directo o extendido; el archivo de parámetros si das un directorio podrás ver que está en el directorio aunque antes no estuviera.

10.2. ESCRIBIENDO CON LA INSTRUCCION WRITE.

WRITE..: Esta instrucción nos vá a servir para escribir en un fichero, tanto si este es secuencial como directo o de impresora, a continuación vamos a ver el primer formato que es el de escritua en un fichero secuencial.

WRITE nomreg..: Donde 'nomreg' es el nombre que recibe el nombre del registro del fichero donde se desea escribir, antes de ejecutar esta instrucción el fichero ha de estar abierto y a los campos del registro se habrá movido el contenido que se desea escribir.

Vamos a hacer una prueba con esta instrucción, vamos a hacer un programa que nos pida cinco nombres y los escriba en un fichero que se llamará 'AMIGOS' y 'AMIGOS.DAT' en el disco, su registro sólo tendrá un campo llamado 'NOMBRE' con 30 caracteres alfanuméricos. Abre en tu editor un archivo con el nombre de 'PRUEBA37.CBL' y teclea lo que sigue.

```
123456789012
      A   B
```

INSTRUCCIONES RELATIVAS A FICHEROS

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA37.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  AMIGOS;
           RECORD CONTAINS 30 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AMI.

       01  FIC-AMI                          PIC X(30).

       WORKING-STORAGE SECTION.

       77  ERRORES                          PIC X(02).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA Y ABRE EL FICHERO.

           DISPLAY " " ERASE.
           OPEN OUTPUT AMIGOS.

      * VA CINCO VECES A LA RUTINA DONDE SE PIDE Y ESCRIBE UN NOMBRE.

           PERFORM PEDIR 5 TIMES.

      * TERMINA EL PROGRAMA.
           STOP RUN.

      * RUTINA DE PEDIR Y ESCRIBIR NOMBRES.

       PEDIR.
```

```
ACCEPT FIC-AMI LINE 12 POSITION 30 PROMPT NO BEEP.
WRITE FIC-AMI.
```

Despues de compilarlo y ejecutarlo mira lo que hay en el fichero 'AMIGOS.DAT', verás los cinco nombres que hallas tecleado en él, este fichero se podrá despues leer con lo cual hemos guardado unos datos pero que tras desconectar el ordenador los podremos volver a recuperar, con lo cual tenemos una ventaja sobre las tablas. El programa tiene un pequeño problema, que no es de el programa sino de el RM-COBOL, y es que no le gustan los nombres que contienen eñes y vocales acentuadas, prueba a introducir por ejemplo 'MUROZ' y verás como el programa se detiene devolviéndote un error.

WRITE ***nomreg*** **FROM** ***nomcam***...: En este segundo formato de escritura se trata de ahorrar un trabajo a la hora de pasar datos a los campos denominados como registro, con él estos datos no habrá que pasarlos, sino que el registro 'nomreg' se escribirá con el contenido que tenga '*nomcam*'.

Para ver la utilización de este formato vamos a hacer un programa en el cual se utilice el fichero 'AMIGOS' del ejemplo anterior, pero esta vez el registro será una línea de 100 caracteres; aparte de esto en la **WORKING-STORAGE** vamos a declarar un grupo con 'NOMBRE', 'APELLIDOS', 'DIRECCION' y 'POBLACION', de tal forma que entre ellos sumen 100 caracteres, la finalidad del programa es que con estos campos se pidan los datos y despues se escriban en el fichero. Abre en tu editor un archivo con el nombre de 'PRUEBA38.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA38.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
```

```
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  AMIGOS;
           RECORD CONTAINS 100 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC AMI.

       01  FIC-AMI                              PIC X(100).

       WORKING-STORAGE SECTION.

       77  ERRORES                              PIC X(02).
       01  DATOS.
           03  NOMBRE                           PIC X(25).
           03  APELLIDOS                        PIC X(25).
           03  DIRECCION                        PIC X(25).
           03  POBLACION                        PIC X(25).

       PROCEDURE DIVISION.
       COMIENZO.

      * SE BORRA LA PANTALLA Y SE ABRE EL ARCHIVO.

           DISPLAY " " ERASE.
           OPEN OUTPUT AMIGOS.

      * SE IMPRIME LA MASCARA DE PETICION DE CAMPOS.

           DISPLAY "NOMBRE.....:" LINE  6 POSITION 20.
           DISPLAY "APELLIDOS...:" LINE  8 POSITION 20.
           DISPLAY "DIRECCION...:" LINE 10 POSITION 20.
           DISPLAY "POBLACION...:" LINE 12 POSITION 20.

      * PETICION DE LOS DISTINTOS CAMPOS.

       PETICION.

           ACCEPT NOMBRE LINE 6 POSITION 33 PROMPT NO BEEP.
```

INSTRUCCIONES RELATIVAS A FICHEROS

```
* SI EL NOMBRE ES IGUAL A ESPACIOS SE TERMINA.

      IF NOMBRE = SPACES GO TO FIN.

* EN CASO CONTRARIO SE PIDEN LOS DEMAS CAMPOS.

      ACCEPT APELLIDOS LINE 8 POSITION 33 PROMPT NO BEEP.
      ACCEPT DIRECCION LINE 10 POSITION 33 PROMPT NO BEEP.
      ACCEPT POBLACION LINE 12 POSITION 33 PROMPT NO BEEP.

* SE GRABAN LOS CAMPOS PEDIDOS.

      WRITE FIC-AMI FROM DATOS.

* SE VUELVE A EMPEZAR.

      GO TO PETICION.

 FIN.

* FIN DEL PROGRAMA.

      STOP RUN.
```

Cuando lo ejecutes podrás ver como te vá pidiendo cada uno de los campos que hemos declarado en la **WORKING-STORAGE**, una vez pedidos escribe estos en el fichero sin tener que pasarlos uno a uno a los campos del registro, cosa que en este caso no se podría efectuar ya que el registro sólo tiene un campo. El programa te irá pidiendo registros hasta que como nombre le des espacios, en ese momento terminará, entonces mira el archivo 'AMIGOS.DAT', verás todo lo que has ido tecleando guardado en él.

10.3. LEYENDO CON LA INSTRUCCION READ.

READ..: Esta instrucción nos vá a servir para leer datos de un fichero, para ello el fichero ha de estar abierto antes de efectuar la operación de lectura, tenemos el siguiente formato.

READ nomfic..: Donde 'nomfic' es el nombre lógico del fichero del que se desea leer, al efectuar esta operación los datos que halla en el disco pasarán a los campos declarados como registro del fichero, si el fichero donde se está leyendo es secuencial el puntero de lectura avanzará al leer, si al efectuar la operación

de lectura no hay datos disponibles para tal fin el programa se detendrá dándonos un error.

Vamos a hacer un programa que coja el fichero 'AMIGOS' que escribimos antes y que lo lea imprimiendo su registro por pantalla, cada vez que lea e imprima nos preguntará si queremos leer otro registro con lo cual podremos controlar el número de registros a leer. Abre en tu editor un archivo con el nombre de 'PRUEBA39.CBL' y teclea lo que tienes a continuación.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA39.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  AMIGOS;
           RECORD CONTAINS 100 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AMI.

       01  FIC-AMI                      PIC X(100).

       WORKING-STORAGE SECTION.

       77  ERRORES                      PIC X(02).
       77  RESPUESTA                    PIC X(01).

       PROCEDURE DIVISION.
       COMIENZO.
```

```
* BORRA LA PANTALLA Y ABRE EL FICHERO PARA LECTURA.

    DISPLAY " " ERASE.
    OPEN INPUT AMIGOS.

* SE PREGUNTA SI SE LEE UN REGISTRO.

    DISPLAY "LEO UN REGISTRO?" LINE 6 POSITION 20.

 LEERSINO.

    ACCEPT RESPUESTA LINE 6 POSITION 36 NO BEEP.

    IF RESPUESTA = "N" OR RESPUESTA = "n" GO TO FIN.
    IF RESPUESTA = "S" OR RESPUESTA = "s" GO TO LEER.

* SI SE LLEGA AQUI ES QUE LA RESPUESTA NO FUE VALIDA.

    GO TO LEERSINO.

 LEER.

* LEE EL REGISTRO.

    READ AMIGOS.

* LO IMPRIME.

    DISPLAY FIC-AMI LINE 10 POSITION 1.

* VUELVE A PREGUNTAR SI SIGUE LEYENDO.

    GO TO LEERSINO.

 FIN.

* FIN DEL PROGRAMA.
    STOP RUN.
```

Cuando lo ejecutes verás como ván apareciendo en pantalla los registros que grabaste con el programa anterior, intenta leer más registros de los que grabaste, verás como el programa se detiene dándote un error.

READ nomfic AT END sent..: Este formato nos vá a servir

para leer de un fichero secuencial previniendo el error que nos dá el programa al llegar al fin de fichero, cuando este fin llegue se ejecutará 'sent' que puede ser cualquiera de las instrucciones vistas hasta ahora, de esta forma podremos leer todos los registros que tenemos en un fichero.

READ nomfic INTO nomcam..: Este formato es el complementario de **WRITE nomreg FROM**, en este último se escribía el registro con los datos de otros campos, en este formato de lectura los datos que se leen además de depositarse en los campos declarados como registro también lo harán en 'nomcam'.

Vamos a hacer un programa utilizando estos dos formatos últimos, cogeremos el fichero 'AMIGOS.DAT' que grabamos con anterioridad y lo vamos a leer dividiéndolo en los cuatro campos originales, los imprimiremos, esperaremos la pulsación de una tecla y seguiremos leyendo hasta el final del fichero. Abre en tu editor un archivo con el nombre 'PRUEBA40.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA40.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.

       FILE SECTION.

       FD  AMIGOS;
```

```
           RECORD CONTAINS 100 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AMI.

       01  FIC-AMI                          PIC X(100).

       WORKING-STORAGE SECTION.

       77  ERRORES                          PIC X(02).
       77  TECLA                            PIC X(01).
       01  DATOS.
           03  NOMBRE                       PIC X(25).
           03  APELLIDOS                    PIC X(25).
           03  DIRECCION                    PIC X(25).
           03  POBLACION                    PIC X(25).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA Y ABRE EL FICHERO.

           DISPLAY " " ERASE.
           OPEN INPUT AMIGOS.

      * IMPRIME LA MASCAR PARA IMPRESION DE LOS CAMPOS.

           DISPLAY "NOMBRE....:" LINE  6 POSITION 20.
           DISPLAY "APELLIDOS.:" LINE  8 POSITION 20.
           DISPLAY "DIRECCION.:" LINE 10 POSITION 20.
           DISPLAY "POBLACION.:" LINE 12 POSITION 20.

      * LECTURA DE UN REGISTRO.

       LEER.

           READ AMIGOS INTO DATOS AT END GO TO FIN.

      * IMPRESION DE LOS CAMPOS LEIDOS.

           DISPLAY NOMBRE LINE  6 POSITION 31.
           DISPLAY APELLIDOS LINE  8 POSITION 31.
           DISPLAY DIRECCION LINE 10 POSITION 31.
           DISPLAY POBLACION LINE 12 POSITION 31.

      * AVISO DE PULSE UNA TECLA.

           DISPLAY " PULSE UNA TECLA PARA SEGUIR "
                   LINE 15 POSITION 25 REVERSE.
```

```
       * ESPERA LA TECLA.

            ACCEPT TECLA OFF NO BEEP.

       * BORRA  EL MENSAJE Y VUELVE A LEER.

            DISPLAY SPACES SIZE 30 LINE 15 POSITION 25.
            GO TO LEER.

        FIN.

       * AQUI SE LLEGARA CUANDO EL FICHERO LLEGUE AL FINAL.

            DISPLAY "SE HA LLEGADO AL FIN DE FICHERO"
                    LINE 15 POSITION 25 BLINK.
            STOP RUN.
```

Ejecútalo tras compilarlo y podrás ver como todos los datos que metiste en el fichero aparecen en pantalla ordenados, el programa seguirá leyendo registros hasta que llegue al final, en ese momento imprimirá un mensaje en pantalla para avisarnos y se detendrá.

10.4. CERRANDO FICHEROS CON LA INSTRUCCION CLOSE.

CLOSE..: Esta instrucción nos vá a servir para cerrar un fichero, para ello el fichero tendrá que estar abierto, tenemos el siguiente formato.

CLOSE nomfic WITH LOCK, WITH NO REWIND..: Donde 'nomfic' es el nombre del fichero que se desea cerrar, al cerrar un fichero no se ha de especificar el modo, este sólo hace falta al abrirlo; **WITH LOCK** tiene la misma finalidad que en **OPEN**, su utilización es opcional; **WITH NO REWIND** también tiene la misma finalidad que en **OPEN**, y también es opcional, se utiliza más a menudo al cerrar el fichero destinado a impresora, ya que esta siempre salta una página al cerrar, si se quiere evitar se puede añadir esta cláusula a **CLOSE**.

Hemos vistos los formatos más elementales de cada una de las instrucciones de trato con ficheros, pero con ellas tenemos suficiente para almacenar una determinada información en un fichero y recuperarla cuando nos haga falta; basándonos en esto vamos a hacer un programa-agenda usando tablas, en ella

guardaremos los nombres, direcciones y teléfonos de nuestros amigos, dando opción de grabar los registros que tenemos en memoria o recuperar lo que se hubiera grabado en otro momento, para ello vamos a tener una tabla con los campos mencionados y con 100 elementos, tendremos un fichero con los campos 'NOMBRE' de 30 caracteres, 'DIRECCION' de 25 caracteres y 'TELEFONO' de 7 caracteres; cada vez que comencemos el programa se nos dará opción a cargar los registros grabados en otro momento o a entrar dentro del mantenimiento, al terminar dando como código el cero se nos dará opción a grabar lo hecho o a abandonarlo. Abre en tu editor un archivo con el nombre de 'PRUEBA41.CBL' y teclea lo que sigue.

```
123456789012
       A   B.

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA41.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  AMIGOS;
           RECORD CONTAINS 62 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AMI.

       01  FIC-AMI.
           03  NOM-AMI                    PIC X(30).
           03  DIC-AMI                    PIC X(25).
           03  TEL-AMI                    PIC X(07).

       WORKING-STORAGE SECTION.

       77  ERRORES                        PIC X(02).
       77  CODIGO                         PIC 9(03).
       77  OPCION                         PIC X(01).
```

```
       01  FICHAS.
           03  FICHA   OCCURS 100 TIMES.
               05  NOMBRE                  PIC X(30).
               05  DIRECCION               PIC X(25).
               05  TELEFONO                PIC X(07).

       PROCEDURE DIVISION.
       COMIENZO.

      * IMPRIME EL MENU QUE NOS DA OPCION A LEER

           DISPLAY " M E N U " LINE 2 POSITION 34 ERASE REVERSE.
           DISPLAY "1.- LEER EL FICHERO" LINE 8 POSITION 30.
           DISPLAY "2.- MANTENIMIENTO" LINE 10 POSITION 30.
           DISPLAY "3.- SALIR" LINE 12 POSITION 30.
           DISPLAY "ELIJE OPCION.[ ]" LINE 17 POSITION 10.

      * PETICION DE LA OPCION.

       PEDIROPCION1.

           ACCEPT OPCION LINE 17 POSITION 24 NO BEEP.

      * SEGUN LA OPCION PULSADA SE VA UN LUGAR U OTRO DEL PROGRAMA.

           IF OPCION = "1" GO TO LECTURA.
           IF OPCION = "2" GO TO MANTENIMIENTO.
           IF OPCION = "3" GO TO SALIR.

      * SI SE LLEGA AQUI ES QUE LA OPCION ES INVALIDA

           GO TO PEDIROPCION1.

       LECTURA.

      * BORRA LA PANTALLA Y PONE UN MENSAJE EN PANTALLA.

           DISPLAY " L E Y E N D O " LINE 12 POSITION 30
                                    REVERSE ERASE.

      * ABRE EL FICHERO EN MODO DE LECTURA SECUENCIAL.

           OPEN INPUT AMIGOS.

      * LEE LOS 100 REGISTROS Y LOS ALMACENA EN LA TABLA, PARA ELLO
      * SE HACE UN PERFORM VARIANDO EL CODIGO DE 1 EN 1 HASTA 100.
```

```
        PERFORM LEERFICHERO VARYING CODIGO FROM 1 BY 1
                                    UNTIL CODIGO > 100.

    * TRAS LEER CIERRA EL FICHERO Y VUELVE AL PRINCIPIO.

        CLOSE AMIGOS.
        GO TO COMIENZO.

     LEERFICHERO.

    * LEE UN REGISTRO Y LO ALMACENA EN LA TABLA.

        READ AMIGOS.
        MOVE NOM-AMI TO NOMBRE(CODIGO).
        MOVE DIC-AMI TO DIRECCION(CODIGO).
        MOVE TEL-AMI TO TELEFONO(CODIGO).

     MANTENIMIENTO.

    * EN ESTA PARTE SE DARA OPCION A DAR DE ALTA, BAJA O CONSULTAR
    * UN CIERTO AMIGO. PRIMERO SE IMPRIME LA MASCARA.

        DISPLAY " M A N T E N I M I E N T O " LINE 2 POSITION 25
                REVERSE ERASE.
        DISPLAY "CODIGO....:" LINE  4 POSITION 30 LOW.
        DISPLAY "NOMBRE....:" LINE  7 POSITION 30 LOW.
        DISPLAY "DIRECCION.:" LINE  9 POSITION 30 LOW.
        DISPLAY "TELEFONO..:" LINE 11 POSITION 30 LOW.
        DISPLAY "A=ALTA/B=BAJA/S=ABANDONAR..[ ]" LINE 16 POSITION 10.

    * SE PIDE EL CODIGO.

     PEDIRCODIGO.

        ACCEPT @CODIGO LINE 4 POSITION 41 NO BEEP.

    * SI EL CODIGO ES CERO SE VA A LA ETIQUETA FINALIZAR.

        IF CODIGO = 0 GO TO FINALIZAR.

    * SI EL DATO TECLEADO NO ES NUMERICO SE VUELVE A PEDIR.

        IF CODIGO NOT NUMERIC GO TO PEDIRCODIGO.

    * SI ES VALIDO COMPRUEBA SI YA EXISTIA, DE SER ASI SE VA A EXISTE.

        IF NOMBRE(CODIGO) NOT = SPACES GO TO EXISTE.
```

```
    * EN CASO CONTRARIO PIDE LOS DEMAS CAMPOS.

        ACCEPT NOMBRE(CODIGO) LINE 7 POSITION 41 NO BEEP PROMPT.
        ACCEPT DIRECCION(CODIGO) LINE 9 POSITION 41 NO BEEP PROMPT.
        ACCEPT TELEFONO(CODIGO) LINE 11 POSITION 41 NO BEEP PROMPT.

    * TRAS PEDIR LOS CAMPOS SE VA A PEDIROPCION2.

        GO TO PEDIROPCION2.

    * AQUI SE LLEGARA SI EL CODIGO SOLICITADO YA EXISTIA.

     EXISTE.

    * IMPRIME LOS DISTINTOS CAMPOS.

        DISPLAY NOMBRE(CODIGO) LINE 7 POSITION 41.
        DISPLAY DIRECCION(CODIGO) LINE 9 POSITION 41.
        DISPLAY TELEFONO(CODIGO) LINE 11 POSITION 41.

     PEDIROPCION2.

    * PIDE OPCION Y ACTUA EN CONSECUENCIA.

        ACCEPT OPCION LINE 16 POSITION 38 NO BEEP.
        IF OPCION = "A" OR OPCION = "a" GO TO ALTAS.
        IF OPCION = "B" OR OPCION = "b" GO TO BAJAS.
        IF OPCION = "S" OR OPCION = "s" GO TO ABANDONAR.

    * SI SE LLEGA AQUI ES QUE LA OPCION ERA INVALIDA.
        GO TO PEDIROPCION2.

     ALTAS.

    * SE BORRA LA PANTALLA Y SE VUELVE A PEDIR OTRO CODIGO.

        PERFORM BORRA.
        GO TO PEDIRCODIGO.

     BAJAS.

    * MUEVE ESPACIOS AL NOMBRE, CON LO CUAL CUANDO LA PROXIMA VEZ
    * SE PREGUNTE POR EL SERA COMO SI ESTUVIERA DADO DE BAJA, DESPUES
    * SE BORRA LA PANTALLA Y SE VUELVE A PEDIR OTRO CODIGO.

        MOVE SPACES TO NOMBRE(CODIGO).
        PERFORM BORRA.
```

```
           GO TO PEDIRCODIGO.

       ABANDONAR.

      * BORRA LA PANTALLA Y VUELVE A PEDIR OTRO CODIGO.

           PERFORM BORRA.
           GO TO PEDIRCODIGO.

       BORRA.

      * RUTINA PARA BORRAR LOS CAMPOS DE LA PANTALLA.

           DISPLAY SPACES SIZE  2 LINE  4 POSITION 41.
           DISPLAY SPACES SIZE 30 LINE  7 POSITION 41.
           DISPLAY SPACES SIZE 25 LINE  9 POSITION 41.
           DISPLAY SPACES SIZE  7 LINE 11 POSITION 41.

       FINALIZAR.

      * AQUI SE LLEGARA CUANDO COMO CODIGO SE DE CERO
      * PRIMERO SE IMPRIME UN MENU DANDONOS OPCION A GRABAR
      * LO REALIZADO, A SALIR DEL PROGRAMA SIN GRABAR O A
      * VOLVER AL MANTENIMIENTO.

           DISPLAY " M E N U " LINE 2 POSITION 34 REVERSE ERASE.
           DISPLAY "1.- GRABAR" LINE 6 POSITION 30.
           DISPLAY "2.- SALIR DEL PROGRAMA" LINE 8 POSITION 30.
           DISPLAY "3.- VOLVER AL MANTENIMIENTO" LINE 10 POSITION 30.
           DISPLAY "ELIJE OPCION.[ ]" LINE 15 POSITION 10.

      * SE PIDE OPCION Y SEGUN ESTA SE ACTUA.

       PEDIROPCION3.

           ACCEPT OPCION LINE 15 POSITION 24 NO BEEP.
           IF OPCION = "1" GO TO GRABAR.
           IF OPCION = "2" GO TO SALIR.
           IF OPCION = "3" GO TO MANTENIMIENTO.

      * SI SE LLEGA AQUI ES QUE LA OPCION ES INVALIDA

           GO TO PEDIROPCION3.

       GRABAR.

      * SE PONE UN MENSAJE EN PANTALLA.
```

```
        DISPLAY " G R A B A N D O " LINE 12 POSITION 25
                REVERSE ERASE.

    * SE ABRE EL FICHERO EN MODO DE ESCRITURA SECUENCIAL.

        OPEN OUTPUT AMIGOS.

    * MEDIANTE UN PERFORM SE LLAMA A LA RUTINA QUE PASA LOS DATOS
    * AL FICHERO Y LOS ESCRIBE.

        PERFORM ESCRIBE VARYING CODIGO FROM 1 BY 1
                                UNTIL CODIGO > 100.

    * TRAS ESCRIBIR SE CIERRA EL FICHERO Y VUELVE AL MENU FINALIZAR
    * POR SI SE QUIERE VOLVER AL MANTENIMIENTO, O SE QUIERE SALIR.

        CLOSE AMIGOS.
        GO TO FINALIZAR.

     ESCRIBE.

    * PASA LOS DATOS DE UN ELEMENTO DE LA TABLA AL REGISTRO DEL
    * FICHERO Y SE ESCRIBE ESTE.

        MOVE NOMBRE(CODIGO) TO NOM-AMI.
        MOVE DIRECCION(CODIGO) TO DIC-AMI.
        MOVE TELEFONO(CODIGO) TO TEL-AMI.
        WRITE FIC-AMI.

     SALIR.

    * A ESTA ETIQUETA SE LLEGA CUANDO SE QUIERE ABANDONAR EL PROGRAMA.

        DISPLAY " A D I O S " LINE 12 POSITION 35 REVERSE ERASE.
        STOP RUN.
```

Cuando lo ejecutes aparecerá un primer menú donde se nos dará opción a cargar lo hecho otras veces, a entrar en el mantenimiento o a salir del programa; para cargar el fichero hecho otras veces se tendrá que haber grabado al menos una vez, si se pulsa la opción cargar el programa abrirá el fichero de 'AMIGOS' y leerá los 100 registros que componen el fichero, tras ello volverá al menú inicial donde podremos elegir entre entrar al mantenimiento o la salida; una vez en el mantenimiento se nos pedirá un código de amigo, si este existe lo imprimirá y nos preguntará que queremos hacer con ese registro, si no existe nos irá pidiendo campos y al final nos pedirá opción; para saber si un registro existía ya o no se mira si el nombre de ese registro es igual a espacios, si es

así es que está vacío y por lo tanto no existe, si no es igual a espacios es que tiene algo, para dar de alta lo único que hace es dejar lo pedido en la tabla, cuando se trata de dar de baja lo que hace es meter espacios en el nombre, con lo cual la próxima vez que se pregunte por él estará dado de baja; cuando querramos dejar el mantenimiento tendremos que pulsar return al pedirnos el código, con ello nos aparecerá otro menú donde se nos dará opción a grabar lo realizado, salir del programa o volver al mantenimiento, si eligimos la opción grabar el programa abrirá el fichero en modo escritura y grabará los 100 elementos que componen la tabla, tras ello volverá al menú anterior, con lo cual podemos grabar y seguir en el mantenimiento si lo deseamos. Con esta breve explicación y los comentarios insertos en el programa te será fácil comprender este programa y mejorarlo porque no.

Hasta ahora hemos ido tratando con fichero secuenciales, vamos ahora a ver como se trata un fichero de impresora, lo primero que tendremos que hacer al declarar su registros es cambiar **LABEL RECORD IS STANDARD** por **LABEL RECORD IS OMITTED**, ya que en la escritura en impresora no existe comprobación de registros; en la impresora no podremos realizar operaciones de lectura como es natural, es un periférico de sólo salida, con lo cual sólo podremos escribir en él, para ello utilizaremos la instrucción **WRITE** ya vista a la cual podremos añadirle el siguiente formato.

10.5. MAS SOBRE WRITE.

WRITE nomreg AFTER ADVANCING nn LINES..: En este formato se escribe 'nomreg' en la impresora pero antes se saltan 'nn' lineas.

WRITE *nomreg* AFTER ADVANCING PAGE..: Este formato escribe 'nomreg' en la impresora pero antes salta una página.

WRITE nomreg BEFORE ADVANCING nn LINES..: Este formato funciona igual que el primero, pero salta 'nn' líneas despues de la impresión.

WRITE *nomreg* BEFORE ADVANCING PAGE..: Este formato funciona igual que el segundo, pero salta la página despues de haber efectuado la impresión.

WRITE nomreg FROM nomcam [op]..: En este formato se escribe en impresora el registro 'nomreg' con el

contenido de 'nomcam', este formato ya lo vimos en los ficheros secuenciales, '[op]' puede ser cualquiera de las cuatro opciones anteriores.

Para ver la utilización en impresora vamos a hacer un programa que lea el archivo que se crea con el programa anterior e imprima su contenido. Abre en tu editor un archivo con el nombre de 'PRUEBA42.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA42.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  FILE STATUS IS ERRORES.

           SELECT IMP;
                  ASSIGN TO PRINT, "PRINTER";
                  FILE STATUS IS ERRORES.

      DATA DIVISION.
      FILE SECTION.

      FD  AMIGOS;
          RECORD CONTAINS 62 CHARACTERS;
          LABEL RECORD IS STANDARD;
          DATA RECORD IS FIC-AMI.

      01  FIC-AMI.
          03  NOM-AMI                    PIC X(30).
          03  DIC-AMI                    PIC X(25).
          03  TEL-AMI                    PIC X(07).

      FD  IMP;
          RECORD CONTAINS 80 CHARACTERS;
```

```
       LABEL RECORD IS OMITTED;
       DATA RECORD IS LIN-IMP.

   01  LIN-IMP                          PIC X(80).

   WORKING-STORAGE SECTION.

   77  ERRORES                          PIC X(02).
   77  CODIGO                           PIC 9(03).
   77  TECLA                            PIC X(01).

   01  CABECERA1.
       03  FILLER                       PIC X(05) VALUE SPACES.
       03  FILLER                       PIC X(06) VALUE "NOMBRE".
       03  FILLER                       PIC X(29) VALUE SPACES.
       03  FILLER                       PIC X(09)
                                        VALUE "DIRECCION".
       03  FILLER                       PIC X(21) VALUE SPACES.
       03  FILLER                       PIC X(08)
                                        VALUE "TELEFONO".

   01  CABECERA2                        PIC X(80) VALUE ALL "-".

   01  CUERPO.
       03  FILLER                       PIC X(05) VALUE SPACES.
       03  NOM-IMP                      PIC X(30).
       03  FILLER                       PIC X(05) VALUE SPACES.
       03  DIC-IMP                      PIC X(25).
       03  FILLER                       PIC X(05) VALUE SPACES.
       03  TEL-IMP                      PIC X(07).

   PROCEDURE DIVISION.
   COMIENZO.

  * ESPERA LA PULSACION DE UNA TECLA.

       DISPLAY "PULSA UNA TECLA PARA COMENZAR" LINE 12
               POSITION 25 REVERSE ERASE.
       ACCEPT TECLA OFF NO BEEP.

  * BORRA LA PANTALLA Y ABRE LOS FICHEROS.

       DISPLAY " " ERASE.
       OPEN INPUT AMIGOS, OUTPUT IMP.

  * IMPRIME LAS CABECERAS 1 Y 2.
```

```
           WRITE LIN-IMP FROM CABECERA1.
           WRITE LIN-IMP FROM CABECERA2.

       * LLAMA 100 VECES A LA RUTINA QUE LEE UN REGISTRO Y LO IMPRIME
       * EN CASO DE NO ESTAR VACIO. VA VARIANDO LA VARIABLE CODIGO
       * PARA IR IMPRIMIENDO EN PANTALLA EL REGISTRO QUE SE IMPRIME.

           PERFORM LEE VARYING CODIGO FROM 1 BY 1 UNTIL CODIGO > 100.

       * UNA VEZ HA LEIDO E IMPRESO  CIERRA LOS FICHEROS Y TERMINA.

           CLOSE AMIGOS,  IMP WITH NO REWIND.

       * UTILIZA LA OPCION 'WITH NO REWIND' PARA QUE LA IMPRESORA NO
       * SALTE UNA PAGINA AL CERRAR.

           STOP RUN.

        LEE.

       * ESTA RUTINA LEE UN REGISTRO, SI EL NOMBRE NO ES IGUAL A
       * ESPACIOS LO PASA AL REGISTRO DE IMPRESION Y LO IMPRIME.
       * ANTES DE CADA IMPRESION DEJA DOS LINEA EN BLANCO.

           READ AMIGOS.
           IF NOM-AMI NOT = SPACES MOVE NOM-AMI TO NOM-IMP,
                                   MOVE DIC-AMI TO DIC-IMP,
                                   MOVE TEL-AMI TO TEL-IMP,
                                   WRITE LIN-IMP FROM CUERPO
                                   AFTER ADVANCING 2 LINES,
                                   DISPLAY "ESCRIBIENDO REGISTRO"
                                   LINE 12 POSITION 30,
                                   DISPLAY CODIGO LINE 12 POSITION 51.
```

Al ejecutarlo nos pedirá que pulsemos una tecla, esto nos dará tiempo a preparar la impresora, una vez pulsada la tecla el programa comenzará a buscar los registros que no esten vacios y los imprimirá, antes de esto se imprimen dos cabeceras que se han preparado en la **WORKING-STORAGE**, la 'CABECERA1' tiene los distintos campos que se ván a listar, y la 'CABECERA2' tiene una línea de guiones; las dos cabeceras se han declarado bajo la cláusula **FILLER**, esto se hace así porque a lo largo de la **PROCEDURE** no se ván a manipular los datos contenidos en este area de memoria, cuando nos hace falta imprimir la cabecera hacemos uso de su nombre de grupo; al imprimir los datos hacemos uso de un grupo que hemos llamado 'CUERPO', en él están contenidos los distintos campos que componen el registro y además entre ellos hemos dejado una serie de espacios para separar, estos espacios

también se declaran bajo la cláusula **FILLER**. En el listado se hace uso de la opción **AFTER ADVANCING** dentro de **WRITE**, con ello conseguimos que antes de imprimir un nueva línea de datos se dejen dos líneas en blanco, con lo cual el listado gana en claridad; al cerrar el fichero utilizamos la opción **WITH NO REWIND**, con esto lo que hacemos es que al cerrar la impresora no haga un salto de página, prueba el programa quitándole esta opción.

10.6. MAS SOBRE LA INSTRUCCION READ.

Vamos a dejar de momento los ficheros secuenciales de lado y vamos a ver algo sobre los ficheros directos, en ellos utilizaremos las mismas instrucciones que hemos vistos pero con otros formatos que vamos a ver a continuación.

READ nomfic INVALID KEY sent..: Con este formato se le dice al programa que lea un registro del fichero 'nomfic', antes de esto tendremos que haber puesto el código a leer en el campo declarado como clave; en caso de que el registro que intentamos leer no exista se ejecutará 'sent'. Este formato admite la opción **INTO**.

READ nomfic WITH NO LOCK INVALID KEY sent..: Este formato funciona igual que el anterior, pero con el párrafo **WITH NO LOCK** avisamos al programa de que no bloquee el registro que se está leyendo, cuando se trabaja en multipuesto el Cobol por defecto bloquea los registros que se leen en prevención de que otra estación pueda leerlos mientras en él se están haciendo modificaciones, sin embargo si el registro se lee sin pretensiones de modificación se le puede añadir esta opción, con la cual conseguiremos que varias estaciones de trabajo accedan al mismo tiempo al mismo registro.

WRITE nomreg INVALID KEY sent..: Con este formato indicamos al programa que escriba el registro indicado, para ello antes tendremos que haber depositado el código de escritura en el campo denominado como clave, si el registro que se intenta escribir ya esta escrito se ejecutará 'sent'. Este formato admite la opción **FROM**.

10.7. BORRANDO CON LA INSTRUCCION DELETE.

Además de las instrucciones anteriores, existentes también en los fichero secuenciales, los directos disponen de las siguientes.

INSTRUCCIONES RELATIVAS A FICHEROS

DELETE ..: Con esta instrucción podremos indicarle al programa que borre un cierto registro, para ello en el campo declarado como clave habremos puesto el código de registro a borrar. Tenemos el siguiente formato.

> **DELETE nomfic INVALID KEY sent**..: Donde 'nomfic' es el nombre del fichero del que se desea borrar, si el registro que se desea borrar no existe se ejecutará 'sent'.

10.8. MODIFICANDO CON LA INSTRUCCION REWRITE.

REWRITE..: Esta instrucción nos vá a servir para reescribir un registro ya existente, normalmente si intentamos escribir un registro que ya existe con la instrucción **WRITE** el programa se detendrá devolviéndonos un error a no ser que hallamos especificado la opción **INVALID KEY**, esta instrucción viene a solucionar este problema. Tenemos el siguiente formato.

> **REWRITE nomreg INVALID KEY sent**..: Donde 'nomreg' es el nombre del registro que se desea reescribir, antes de efectuar esta operación en el campo declarado como clave del fichero se habrá depositado el código del registro que se desea reescribir; si el registro a reescribir no existía se ejecutará 'sent'. Esta formato admite la opción **FROM**.

En todos los formatos de las instrucciones vistas la utilización de **INVALID KEY** es opcional, pero si se produjera un error al efectuar la operación indicada y no se hubiera especificado esta opción el programa se detendrá devolviéndonos un error.

Para ver en funcionamiento todas las instrucciones vistas vamos a hacer el programa de mantenimiento de amigos pero sin tablas, trabajando directamente sobre el fichero, en un fichero directo se puede leer y escribir al mismo tiempo, con lo cual no tendremos que abrir el fichero en un formato distinto para cada vez, lo que si nos hará falta es un programa que cree el fichero antes, ya que la apertura de un fichero en modo directo implica que esta exista ya en el disco, de no ser así se nos devolvería un error; por lo tanto necesitamos un programa que cree el fichero cada vez que lo necesitemos y otro que sea el mantenimiento propiamente dicho, para ello vamos a hacer tres programas, el primero hará la función de menú y nos dará paso a uno de los dos siguientes que serán el de creación o el de mantenimiento.

INSTRUCCIONES RELATIVAS A FICHEROS

Vamos a comenzar haciendo el program-menú, abre en tu editor un archivo con el nombre de 'PRUEBA43.CBL' y teclea lo que sigue.

```
123456789012
       A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA43.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       77  OPCION                          PIC 9(01).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA E IMPRIME EL MENU.

           DISPLAY " M E N U " LINE 2 POSITION 34 REVERSE ERASE.
           DISPLAY "1.- CREACION DE FICHERO" LINE 8 POSITION 30.
           DISPLAY "2.- MANTENIMIENTO DEL FICHERO" LINE 10
                                                  POSITION 30.
           DISPLAY "3.- SALIDA DEL PROGRAMA" LINE 12 POSITION 30.
           DISPLAY "ELIJE OPCION.[ ]" LINE 18 POSITION 10.

      * PETICION DE LA OPCION.

       PEDIROPCION.

           ACCEPT OPCION LINE 18 POSITION 24 NO BEEP.

      * SEGUN LA OPCION TECLEADA SE VA A UNA ETIQUETA U OTRA.

           GO TO CREACION, MANTENIMIENTO, SALIDA DEPENDING ON OPCION.

      * SI SE LLEGA AQUI ES QUE LA OPCION ERA INVALIDA.

           GO TO PEDIROPCION.

       CREACION.
```

```
      * SI LA OPCION ES CREAR SE LLAMA AL PROGRAMA PRUEBA44 QUE.
      * CREARA EL FICHERO, Y AL VOLVER DEL PROGRAMA SE VA AL PRINCIPIO.

           CALL "PRUEBA44".
           GO TO COMIENZO.

       MANTENIMIENTO.

      * SI LA OPCION ES MANTENIMIENTO LLAMA AL PROGRAMA PRUEBA45.
      * CUANDO SE VUELVA DE ESTE PROGRAMA SE IRA AL PRINCIPIO.

           CALL "PRUEBA45".
           GO TO COMIENZO.

       SALIDA.

      * SI LA OPCION ES SALIR SE BORRA LA PANTALLA Y SE TERMINA.

           DISPLAY " " ERASE.
           STOP RUN.
```

Este será el programa que actúe como menú, vamos a hacer ahora el programa que cree el fichero, abre en tu editor un archivo con el nombre de 'PRUEBA44.CBL' y teclea lo siguiente.

```
123456789012
       A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA44.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  ORGANIZATION IS RELATIVE;
                  ACCESS MODE IS DYNAMIC;
                  RELATIVE KEY IS CODIGO;
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.
```

```
       FD  AMIGOS;
           RECORD CONTAINS 62 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AMI.

       01  FIC-AMI.
           03  NOM-AMI                          PIC X(30).
           03  DIC-AMI                          PIC X(25).
           03  TEL-AMI                          PIC X(07).

       WORKING-STORAGE SECTION.

       77  ERRORES                              PIC X(02).
       77  CODIGO                               PIC 9(03).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA Y PONE EL MENSAJE DE CREANDO FICHERO.

           DISPLAY " C R E A N D O   F I C H E R O "
                   LINE 12 POSITION 25 REVERSE ERASE.

      * ABRE EL FICHERO EN MODO 'OUTPUT' CON LO CUAL SE CREARA

           OPEN OUTPUT AMIGOS.

      * CIERRA EL FICHERO Y SALE AL PROGRAMA ANTERIOR.

           CLOSE AMIGOS.

       SALIR.

           EXIT PROGRAM.
```

Por último vamos a hacer el programa que haga el mantenimiento del fichero, este programa se vá a llamar 'PRUEBA45.CBL'.

```
123456789012
       A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA45.

       ENVIRONMENT DIVISION.
```

INSTRUCCIONES RELATIVAS A FICHEROS

```
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AMIGOS;
                  ASSIGN TO RANDOM, "AMIGOS.DAT";
                  ORGANIZATION IS RELATIVE;
                  ACCESS MODE IS DYNAMIC;
                  RELATIVE KEY IS CODIGO;
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  AMIGOS;
           RECORD CONTAINS 62 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AMI.

       01  FIC-AMI.
           03  NOM-AMI                          PIC X(30).
           03  DIC-AMI                          PIC X(25).
           03  TEL-AMI                          PIC X(07).

       WORKING-STORAGE SECTION.

       77  ERRORES                              PIC X(02).
       77  CODIGO                               PIC 9(03).
       77  OPCION                               PIC X(01).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA Y ABRE EL FICHERO EN MODO DIRECTO.

           DISPLAY " " ERASE.
           OPEN I-O AMIGOS.

      * IMPRIME LA MASCARA DE PETICION DE DATOS.

           DISPLAY " M A N T E N I M I E N T O "
                   LINE 2 POSITION 25 REVERSE ERASE.
           DISPLAY "CODIGO....:[   ]"
                   LINE 5 POSITION 30.
           DISPLAY "NOMBRE....:[                              ]"
```

```
            LINE 8 POSITION 30.
      DISPLAY "DIRECCION.: [                              ]"
            LINE 10 POSITION 30.
      DISPLAY "TELEFONO..: [        ]"
            LINE 12 POSITION 30.
      DISPLAY "A=ALTAS/B=BAJAS/S=ABANDONAR. [ ]"
            LINE 18 POSITION 10.

    * PIDE EL CODIGO.

     PEDIRCODIGO.

         ACCEPT CODIGO LINE 5 POSITION 42 NO BEEP.

    * SI EL CODIGO ES CERO SE VA A SALIR.

         IF CODIGO = 0 GO TO SALIR.

    * SI EL CODIGO NO ES NUMERICO SE VUELVE A PEDIR.

         IF CODIGO NOT NUMERIC GO TO PEDIRCODIGO.

    * SI EL CODIGO ES VALIDO SE LEE PARA COMPROBAR SI EXISTE
    * SI AL LEER NO DA ERROR ES QUE LO HA LEIDO BIEN Y POR LO TANTO
    * EXISTE, POR EL CONTRARIO SI DA ERROR ES QUE EL REGISTRO NO
    * EXISTE Y SE PUEDEN PEDIR LOS CAMPOS.

         READ AMIGOS INVALID KEY GO TO NOEXISTE.

    * SI EXISTE IMPRIME LOS CAMPOS.

         DISPLAY NOM-AMI LINE 8 POSITION 42.
         DISPLAY DIC-AMI LINE 10 POSITION 42.
         DISPLAY TEL-AMI LINE 12 POSITION 42.

    * SE VA A PEDIR OPCION.

         GO TO PEDIROPCION.

     NOEXISTE.

    * AQUI SE LLEGARA CUANDO AL LEER DE UN ERROR LO QUE QUIERE DECIR
    * QUE EL REGISTRO NO EXISTE, SE PIDEN LOS DEMAS CAMPOS.

         ACCEPT NOM-AMI LINE 8 POSITION 42 NO BEEP.
         ACCEPT DIC-AMI LINE 10 POSITION 42 NO BEEP.
         ACCEPT TEL-AMI LINE 12 POSITION 42 NO BEEP.
```

INSTRUCCIONES RELATIVAS A FICHEROS

```
    * SE PIDE OPCION DE QUE HACER CON EL REGISTRO.

     PEDIROPCION.

         ACCEPT OPCION LINE 18 POSITION 39 NO BEEP.

    * SEGUN LA OPCION TECLEADA SE VA A UNA PARTE U OTRA.

         IF OPCION = "A" OR OPCION = "a" GO TO ALTAS.
         IF OPCION = "B" OR OPCION = "b" GO TO BAJAS.
         IF OPCION = "S" OR OPCION = "s" GO TO ABANDONAR.

    * SI SE LLEGA AQUI ES QUE LA OPCION ES INVALIDA.

         GO TO PEDIROPCION.

     ALTAS.

    * SE ESCRIBE EL REGISTRO, SI DA ERROR QUIERE DECIR QUE YA EXISTIA
    * POR LO TANTO SE REESCRIBE PARA GUARDAR LAS POSIBLE MODIFICACIONES
    * QUE SE HALLAN HECHO EN EL REGISTRO.

         WRITE FIC-AMI INVALID KEY REWRITE FIC-AMI.

    * EN REWRITE NO SE ESPECIFICA 'INVALID KEY' YA QUE SI AL ESCRIBIR
    * DA UN ERROR QUIERE DECIR QUE YA EXISTE, AL REESCRIBIRLO NO DEBE
    * DE DAR NINGUN ERROR.

    * SE BORRAN LOS DATOS DE LA PANTALLA.

         PERFORM BORRA.

    * SE VUELVE A PEDIR OTRO REGISTRO.

         GO TO PEDIRCODIGO.

     BAJAS.

    * SE BORRA EL REGISTRO, SI DA ERROR QUIERE DECIR QUE NO EXISTIA
    * EN ESE CASO NO SE HACE NADA Y SE SIGUE

         DELETE AMIGOS INVALID KEY MOVE OPCION TO OPCION.

    * TRAS BORRAR EL REGISTRO EN CASO DE EXISTIR SE BORRA LA PANTALLA
    * Y SE VUELVE A POR OTRO REGISTRO.

         PERFORM BORRA.
         GO TO PEDIRCODIGO.
```

```
    ABANDONAR.

  * PARA ABANDONAR SE BORRA LA PANTALLA Y SE VUELVE A POR OTRO CODIGO

      PERFORM BORRA.
      GO TO PEDIRCODIGO.

    BORRA.

  * RUTINA PARA BORRAR LOS DISTINTOS CAMPOS DE LA PANTALLA.

      DISPLAY SPACES SIZE  3 LINE  5 POSITION 42.
      DISPLAY SPACES SIZE 30 LINE  8 POSITION 42.
      DISPLAY SPACES SIZE 25 LINE 10 POSITION 42.
      DISPLAY SPACES SIZE  7 LINE 12 POSITION 42.

    SALIR.

  * AQUI SE LLEGA CUANDO COMO CODIGO SE DA CERO. SE CIERRAN LOS
  * FICHEROS Y SE SALE DEL PROGRAMA.

      CLOSE AMIGOS.

    SALIR1.

      EXIT PROGRAM.
```

Una vez terminado este último programa ejecuta 'PRUEBA43', en pantalla aparecerá un menú dándonos opción a crear el fichero, a entrar en el mantenimiento o a abandonarlo, el fichero habrá que crearlo al menos una vez, cuando entremos en el mantenimiento tendremos una pantalla con los distintos campos donde se nos pedirá el código y se actuará en consecuencia. En cada uno de los programas van incluidos una serie de comentarios aclarativos.

Hemos visto la utilización de los ficheros directos, los ficheros indexados se tratan exactamente igual y con las mismas instrucciones, la diferencia está en que el campo clave se ha de declarar como parte del registro y que tiene que ser alfanumérico, las instrucciones de lectura, escritura, borrado y reescritura son las misma que las vistas anteriormente.

Para ver la utilización de los ficheros indexados vamos a hacer un programa para nuestro ayuntamiento, en él lo que se nos pide es tener archivados los coches de la ciudad, la clave será la matrícula del coche, además de esta tendremos los campos

'PROPIETARIO', donde tendremos el nombre y apellidos de este, 'MODELO' donde tendremos la marca y modelo del coche, 'FECHA DE MATRICULACION', donde almacenaremos tal fecha. El proceso será parecido al de anteriores programas, primero pediremos la clave, con esta averiguaremos si ese coche ya estaba dado de alta o no, si estaba dado de alta imprimiremos los campos que componen el registro, si no existía pediremos todos los campos menos la fecha que la habremos pedido al entrar en el programa y la pasaremos de un programa a otro. Tras esta breve descripción del funcionamiento del programa vamos a pasar a la acción, tendremos que crear un primer programa que será de presentación y menú, un segundo que será de creación del fichero y un tercero que será el mantenimiento propiamente dicho. Abre en tu editor un archivo con el nombre 'PRUEBA47.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA47.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  FECHA.
           03  DIA                     PIC 9(02).
           03  MES                     PIC 9(02).
           03  ANO                     PIC 9(02).

       77  OPCION                      PIC 9(01).
       77  LINEA                       PIC 9(02).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA E IMPRIME LA PRESENTACION.

           DISPLAY " " ERASE.
      * AYUDANDOSE DE PERFORM LLAMA VARIAS VECES A DISTINTAS RUTINAS
      * QUE IMPRIMIRAN UNA SERIE DE ESPACIOS UNA VECES EN REVERSE
      * Y OTRAS EN NORMAL PARA FORMA UN CUADRO EN LA PANTALLA.
```

```
           PERFORM LIN1 VARYING LINEA FROM 5 BY 1 UNTIL LINEA > 19.
           PERFORM LIN2 VARYING LINEA FROM 7 BY 1 UNTIL LINEA > 17.
           PERFORM LIN3 VARYING LINEA FROM 9 BY 1 UNTIL LINEA > 15.

           DISPLAY "MANTENIMIENTO" LINE 10 POSITION 33 REVERSE.
           DISPLAY "DE" LINE 12 POSITION 38 REVERSE.
           DISPLAY "AUTOMOVILES" LINE 14 POSITION 33 REVERSE.

           DISPLAY "FECHA.[  /  /  ]" LINE 17 POSITION 20.

      * PETICION DE LA FECHA.

           ACCEPT DIA LINE 17 POSITION 27 NO BEEP.
           ACCEPT MES LINE 17 POSITION 30 NO BEEP.
           ACCEPT ANO LINE 17 POSITION 33 NO BEEP.

       * IMPRESION DEL MENU

       MENU.

           DISPLAY " M E N U " LINE 2 POSITION 34 REVERSE ERASE.
           DISPLAY "1.- CREACION DEL FICHERO"
                   LINE 8 POSITION 30.
           DISPLAY "2.- MANTENIMIENTO DEL FICHERO"
                   LINE 10 POSITION 30.
           DISPLAY "3.- SALIR DEL PROGRAMA"
                   LINE 12 POSITION 30.
           DISPLAY "ELIJE OPCION.[ ]" LINE 16 POSITION 10.

      * PETICION DE LA OPCION.

       PEDIROPCION.

           ACCEPT OPCION LINE 18 POSITION 24 NO BEEP.

      * SEGUN LA OPCION PULSADA SE VA A UNA PARTE U OTRA.

           GO TO CREACION, MANTENIMIENTO, SALIDA DEPENDING ON OPCION.

      * SI SE LLEGA AQUI ES QUE LA OPCION ERA INVALIDA.

           GO TO PEDIROPCION.

       CREACION.

      * PARA CREAR EL FICHERO SE LLAMA A PRUEBA48 Y SE VUELVE AL MENU.
      * PASA LA FECHA AL LLAMRA AL PROGRAMA PARA NO PERDERLA.
```

```
            CALL "PRUEBA48" USING FECHA.
            GO TO MENU.

        MANTENIMIENTO.

      * SE LLAMA A PRUEBA49 Y CUANDO VUELVE SE VA AL MENU.

            CALL "PRUEBA49" USING FECHA.
            GO TO MENU.

        SALIDA.

      * SE BORRA LA PANTALLA Y SE TERMINA.

            DISPLAY " " ERASE.
            STOP RUN.

      * RUTINAS DE IMPRESION DE LINEAS PARA LA PRESENTACION.

        LIN1.

            DISPLAY SPACES SIZE 60 LINE LINEA POSITION 10 REVERSE.

        LIN2.

            DISPLAY SPACES SIZE 40 LINE LINEA POSITION 20.

        LIN3.

            DISPLAY SPACES SIZE 20 LINE LINEA POSITION 30 REVERSE.
```

Este programa nos servirá de presentación y de menú, lo primero que hace es hacer un cuadro con espacios y poner dentro de él el título del programa, tras ello pide la fecha y pasa al menú, en él se nos dá opción a crear el fichero, entrar en el mantenimiento o salir del programa; al llamar a uno de los dos programas posibles, creación o mantenimiento, lo hace pasando el valor del grupo 'FECHA', esta misma fecha la utilizaremos al dar de alta un coche para que no tengamos que teclearla cada vez, en el programa de creación la fecha no nos es de ninguna utilidad, pero la pasamos para no perder su valor; tanto en el programa de creación como en el de mantenimiento deberá existir en la **DATA DIVISION** la **LINKAGE SECTION** para poder pasar las variables de un programa a otro, además en el programa donde se vaya a hacer uso de esta variable habrá que poner el formato **PROCEDURE DIVISION USING** para poder hacerlo.

Vamos a teclear seguidamente el programa de creación, abre en tu editor un archivo con el nombre de 'PRUEBA48.CBL' y teclea lo que sigue.

```
123456789012
      A   B

      IDENTIFICATION DIVISION.
      PROGRAM-ID. PRUEBA48.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SOURCE-COMPUTER. PC.
      OBJECT-COMPUTER. PC.

      INPUT-OUTPUT SECTION.
      FILE-CONTROL.

          SELECT AUTOS;
                 ASSIGN TO RANDOM, "AUTOS.DAT";
                 ORGANIZATION IS INDEXED;
                 ACCESS MODE IS DYNAMIC;
                 RECORD KEY IS MAT-AUT;
                 FILE STATUS IS ERRORES.

      DATA DIVISION.
      FILE SECTION.

      FD  AUTOS;
          RECORD CONTAINS 75 CHARACTERS;
          LABEL RECORD IS STANDARD;
          DATA RECORD IS FIC-AUT.

      01  FIC-AUT.
          03  MAT-AUT                 PIC X(09).
          03  PRO-AUT                 PIC X(40).
          03  MOD-AUT                 PIC X(20).
          03  FEC-AUT.
              05  DIA-AUT             PIC 9(02).
              05  MES-AUT             PIC 9(02).
              05  ANO-AUT             PIC 9(02).

      WORKING-STORAGE SECTION.

      77  ERRORES                     PIC X(02).
```

```
       LINKAGE SECTION.

       01  FECHA.
           03  DIA                          PIC 9(02).
           03  MES                          PIC 9(02).
           03  ANO                          PIC 9(02).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA E IMPRIME EL MENSAJE DE CREANDO FICHERO.

           DISPLAY " C R E A N D O   F I C H E R O "
                   LINE 12 POSITION 25 REVERSE ERASE.

      * SE ABRE EL FICHERO EN MODO OUTPUT PARA CREARLO.

           OPEN OUTPUT AUTOS.

      * SE CIERRA Y SE VUELVE AL PROGRAMA ANTERIOR.

           CLOSE AUTOS.

       SALIR.

           EXIT PROGRAM.
```

El funcionamiento de este programa es bastante simple, se declara el fichero que hemos llamado 'AUTOS' como indexado, se declara su registro y seguidamente se abre en modo OUTPUT, como sabes este modo de apertura siempre implica la creación del fichero, tras esto cerramos el fichero y volvemos al programa menú; cuando lo ejecutes para crear el fichero notarás como es más lento que el programa de creación que hicimos para el fichero directo de amigos, esto se debe a que los ficheros indexados se basan en la utilización de dos ficheros directos que se habrán creado al ejecutar el programa, tras ejecutarlo pide un directorio del disco y verás como además de 'AUTOS.DAT' también aparece un fichero llamado 'AUTOS.INX', este es el fichero índice, en él el programa guarda todas las claves ordenadas para despues poder hacer un acceso rápido.

Vamos ahora a hacer el tercer y último programa, en él trataremos el mantenimiento de nuestro fichero de automóviles, abre en tu editor un archivo con el nombre de 'PRUEBA49.CBL' y teclea lo que sigue.

INSTRUCCIONES RELATIVAS A FICHEROS

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA49.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AUTOS;
                  ASSIGN TO RANDOM, "AUTOS.DAT";
                  ORGANIZATION IS INDEXED;
                  ACCESS MODE IS DYNAMIC;
                  RECORD KEY IS MAT-AUT;
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  AUTOS;
           RECORD CONTAINS 75 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AUT.

       01  FIC-AUT.
           03  MAT-AUT                    PIC X(09).
           03  PRO-AUT                    PIC X(40).
           03  MOD-AUT                    PIC X(20).
           03  FEC-AUT.
               05  DIA-AUT                PIC 9(02).
               05  MES-AUT                PIC 9(02).
               05  ANO-AUT                PIC 9(02).

       WORKING-STORAGE SECTION.

       77  ERRORES                        PIC X(02).
       77  OPCION                         PIC X(01).
       77  NUMCAM                         PIC 9(01).

       LINKAGE SECTION.
```

```
       01  FECHA.
           03  DIA                      PIC 9(02).
           03  MES                      PIC 9(02).
           03  ANO                      PIC 9(02).

       PROCEDURE DIVISION USING FECHA.
       COMIENZO.

      * SE ABRE EL FICHERO EN MODO DIRECTO.

           OPEN I-O AUTOS.

      * SE BORRA LA PANTALLA Y SE IMPRIME LA MASCARA DE PETICION.

           DISPLAY " M A N T E N I M I E N T O "
                   LINE 2 POSITION 30 REVERSE ERASE.
           DISPLAY "    MATRICULA...: [         ]"
                   LINE 8 POSITION 20.
           DISPLAY "1.- PROPIETARIO.: [
                   "      ]" LINE 10 POSITION 20.
           DISPLAY "2.- MODELO......: [                    ]"
                   LINE 12 POSITION 20.
           DISPLAY "3.- FECHA MATRIC: [  /  /  ]"
                   LINE 14 POSITION 20.
           DISPLAY "A=ALTAS/B=BAJAS/M=MODIFICAR/S=SALIR. [ ]"
                   LINE 18 POSITION 10.

      * PETICION DE LA MATRICULA.

       PEDIRMATRICULA.

           ACCEPT MAT-AUT LINE 8 POSITION 38 NO BEEP.

      * SI LA MATRICULA ES IGUAL A ESPACIOS ES QUE SE QUIERE TERMINAR.

           IF MAT-AUT = SPACES GO TO SALIR.

      * LEE LA MATRICULA PARA VER SI EXISTE

           READ AUTOS INVALID KEY GO TO NOEXISTE.

      * SI DA ERROR ES QUE NO EXISTE, SE VA A PEDIR LOS DEMAS CAMPOS
      * EN CASO CONTRARIO IMPRIME LOS CAMPOS.

           DISPLAY PRO-AUT LINE 10 POSITION 38.
           DISPLAY MOD-AUT LINE 12 POSITION 38.
           DISPLAY DIA-AUT LINE 14 POSITION 38.
```

```
           DISPLAY MES-AUT LINE 14 POSITION 41.
           DISPLAY ANO-AUT LINE 14 POSITION 44.

       * Y SE VA A PEDIR OPCION.

           GO TO PEDIROPCION.

        NOEXISTE.

       * SI AL LEER DA ERROR SE LLEGA AQUI DONDE SE PIDEN LOS DEMAS
       * CAMPOS Y SE TOMA LA FECHA QUE SE DIO AL PRINCIPIO.

           ACCEPT PRO-AUT LINE 10 POSITION 38 NO BEEP.
           ACCEPT MOD-AUT LINE 12 POSITION 38 NO BEEP.
           MOVE DIA TO DIA-AUT.
           DISPLAY DIA-AUT LINE 14 POSITION 38.
           MOVE MES TO MES-AUT.
           DISPLAY MES-AUT LINE 14 POSITION 41.
           MOVE ANO TO ANO-AUT.
           DISPLAY ANO-AUT LINE 14 POSITION 44.

        PEDIROPCION.

       * SE PEDIRA OPCION Y SE ACTUARA EN CONSECUENCIA DE LA RESPUESTA.

           ACCEPT OPCION LINE 18 POSITION 47 NO BEEP.
           IF OPCION = "A" OR OPCION = "a" GO TO ALTAS.
           IF OPCION = "B" OR OPCION = "b" GO TO BAJAS.
           IF OPCION = "M" OR OPCION = "m" GO TO MODIFICAR.
           IF OPCION = "S" OR OPCION = "s" GO TO SALIDA.

       * SI SE LLEGA AQUI QUIERE DECIR QUE LA OPCION ES INVALIDA.

           GO TO PEDIROPCION.

        ALTAS.

       * SE GRABA EL REGISTRO.

           WRITE FIC-AUT INVALID KEY REWRITE FIC-AUT.

       * SE BORRA LA PANTALLA Y SE VUELVE A PEDIR MATRICULA.

           PERFORM BORRA.
           GO TO PEDIRMATRICULA.

        BAJAS.
```

```
        * SE BORRA EL REGISTRO.

            DELETE AUTOS INVALID KEY MOVE OPCION TO OPCION.

        * BORRA LA PANTALLA Y VUELVE A PEDIR MATRICULA.

            PERFORM BORRA.
            GO TO PEDIRMATRICULA.

         MODIFICAR.

        * SE PIDE EL NUMERO DEL CAMPO QUE SE DESEA MODIFICAR.

            DISPLAY "CAMPO A MODIFICAR.[ ] 0=SALIR"
                    LINE 16 POSITION 10.
            ACCEPT NUMCAM LINE 16 POSITION 29 NO BEEP.

        * SI EL CAMPO ES CERO SE VUELVE A PEDIR OPCION.

            IF NUMCAM = 0 DISPLAY SPACES SIZE 30
               LINE 16 POSITION 10, GO TO PEDIROPCION.

        * SEGUN EL NUMERO DE CAMPO SE VA A UNA ETIQUETA U OTRA.

            GO TO PEDIR1, PEDIR2, PEDIR3 DEPENDING ON NUMCAM.

        * SI SE LLEGA AQUI QUIERE DECIR QUE EL NUMERO ES INVALIDO.

            GO TO MODIFICAR.

         PEDIR1.

        * PIDE EL PRIMER CAMPO QUE ES EL PROPIETARIO.

            ACCEPT PRO-AUT LINE 10 POSITION 38 NO BEEP.
            GO TO MODIFICAR.

         PEDIR2.

        * PIDE EL SEGUNDO CAMPO QUE ES EL MODELO.

            ACCEPT MOD-AUT LINE 12 POSITION 38 NO BEEP.
            GO TO MODIFICAR.

         PEDIR3.

        * PIDE EL TERCER CAMPO QUE ES LA FECHA.
```

```
            ACCEPT DIA-AUT LINE 14 POSITION 38 NO BEEP.
            ACCEPT MES-AUT LINE 14 POSITION 41 NO BEEP.
            ACCEPT ANO-AUT LINE 14 POSITION 44 NO BEEP.
            GO TO MODIFICAR.

        SALIDA.
      * SE BORRA LA PANTALLA Y SE VUELVE A PEDIR MATRICULA.

            PERFORM BORRA.
            GO TO PEDIRMATRICULA.

        SALIR.

      * AQUI SE LLEGA AL ABANDONAR EL PROGRAMA, SE CIERRA ANTES
      * EL FICHERO Y DESPUES SE SALE.

            CLOSE AUTOS.

        SALIR1.

            EXIT PROGRAM.

        BORRA.

      * RUTINA PARA BORRAR LOS CAMPOS DE LA PANTALLA.

            DISPLAY SPACES SIZE  9 LINE  8 POSITION 38.
            DISPLAY SPACES SIZE 40 LINE 10 POSITION 38.
            DISPLAY SPACES SIZE 20 LINE 12 POSITION 38.
            DISPLAY "  /  /  " LINE 14 POSITION 38.
```

Como puedes apreciar en el programa la utilización de los ficheros indexados es muy similar a la de los directos, en este programa se ha incluido una parte que hasta ahora no habíamos hecho, que son las modificaciones, aquí se hacían imprescindibles ya que el programa pone por su cuenta al dar un coche de alta la fecha de matriculación, si ese coche se había matriculado otro día habría que cambiarla; para las modificaciones lo que se ha hecho es numerar los campos y cuando se teclea la opción *modificar* se pide el *número* del campo que se desea *modificar*, si este número es cero se vuelve a pedir opción, si es válido se pide el campo correspondiente y despues se vuelve a pedir número a *modificar*, con lo cual podemos modificar varios campos de una misma vez, cuando pulsemos como número de campo el cero el programa volverá a pedirnos opción; tras modificar un registo siempre hay que volver a escribirlo ya que de lo contrario las modificaciones se perderían.

INSTRUCCIONES RELATIVAS A FICHEROS

Vamos a ver algunas instrucciones más sobre los ficheros directos e indexados.

10.9. LA INSTRUCCION START.

START..: Esta instrucción nos vá a servir para posicionarnos en un cierto lugar del fichero y leer en secuencial a partir de ese punto, tenemos el siguiente formato.

START nomfic KEY IS cond nomclv INVALID KEY sent..: Donde 'nomfic' es el nombre del fichero donde nos deseamos posicionar, 'cond' puede ser '=' o **EQUAL TO**, '>' o **GREATER THAN**, **'NOT <'** o **NOT LESS THAN**; estos seis términos se agrupan de dos en dos tal y como hemos visto, son igual, mayor y no menor respectivamente; 'nomclv' debe ser la variable declarada como clave, en ella antes habremos puesto la clave a partir de donde queremos posicionarnos, en los ficheros indexados 'nomclv' puede ser una clave alternativa, lo que se hará es buscar por esa clave en vez de por la principal. El efecto de la ejecución de esta instrucción es el posicionamiento del puntero de lectura en esa clave si existe o en la siguiente; si como 'cond' se ha especificado '=' y la clave no se encuentra se ejecutará 'sent' si es que hemos usado **INVALID KEY**, de no ser así el programa se detendrá devolviéndonos un error; si se ha puesto '>' como 'cond' se buscará la siguiente clave a la especificada, si a partir de la clave especificada hasta el final del fichero no se encuentra ninguan clave entrará en acción 'sent'; por útlimo si como 'cond' se especificò **'NOT <'** el puntero de lectura se posicionará en la misma clave que hemos indicado, si no la encuentra se posicionará en la siguiente que halla, si no encuentra ninguna se ejecutará 'sent'; esta última opción de 'cond' es la más utilizada.

Vamos a hacer un programa utilizando esta última instrucción vista, en él se nos dará una clave de matrícula, con ella haremos un listado desde esa misma matrícula o la siguiente hasta el final del fichero, para leer en secuencial en un fichero aleatorio usaremos el siguiente formato de **READ**.

READ nomfic NEXT RECORD AT END sent..: Como ves al formato normal de lectura le añadimos el párrafo **NEXT RECORD** para indicar que se vá a leer un registro en secuencial pero en un fichero no secuencial, con el párrafo **AT END** controlaremos cuando se llega al final de fichero al igual que en los secuenciales.

Una vez visto esto vamos a comenzar, abre en tu editor un archivo con el nombre de 'PRUEBA50.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA50.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT AUTOS;
                  ASSIGN TO RANDOM, "AUTOS.DAT";
                  ORGANIZATION IS INDEXED;
                  ACCESS MODE IS DYNAMIC;
                  RECORD KEY IS MAT-AUT;
                  FILE STATUS IS ERRORES.

           SELECT IMP;
                  ASSIGN TO PRINT, "PRINTER";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  AUTOS;
           RECORD CONTAINS 75 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-AUT.

       01  FIC-AUT.
           03  MAT-AUT                 PIC X(09).
           03  PRO-AUT                 PIC X(40).
           03  MOD-AUT                 PIC X(20).
           03  FEC-AUT.
               05  DIA-AUT             PIC 9(02).
               05  MES-AUT             PIC 9(02).
               05  ANO-AUT             PIC 9(02).
```

```
FD  IMP;
    RECORD CONTAINS 80 CHARACTERS;
    LABEL RECORD IS OMITTED;
    DATA RECORD IS FIC-IMP.

01  FIC-IMP                        PIC X(80).

WORKING-STORAGE SECTION.

77  ERRORES                        PIC X(02).
77  TECLA                          PIC X.
01  CABECERA1.
    03  FILLER                     PIC X(09) VALUE "MATRICULA".
    03  FILLER                     PIC X(01) VALUE SPACES.
    03  FILLER                     PIC X(40) VALUE "PROPIETARIO".
    03  FILLER                     PIC X(01) VALUE SPACES.
    03  FILLER                     PIC X(20) VALUE "MODELO".
    03  FILLER                     PIC X(01) VALUE SPACES.
    03  FILLER                     PIC X(08) VALUE "FECHA M.".

01  CABECERA2                      PIC X(80) VALUE ALL "-".
01  CUERPO.
    03  MAT-OUT                    PIC X(09).
    03  FILLER                     PIC X(01) VALUE SPACES.
    03  PRO-OUT                    PIC X(40).
    03  FILLER                     PIC X(01) VALUE SPACES.
    03  MOD-OUT                    PIC X(20).
    03  FILLER                     PIC X(01) VALUE SPACES.
    03  DIA-OUT                    PIC ZZ.
    03  FILLER                     PIC X(01) VALUE "/".
    03  MES-OUT                    PIC ZZ.
    03  FILLER                     PIC X(01) VALUE "/".
    03  ANO-OUT                    PIC ZZ.

PROCEDURE DIVISION.
COMIENZO.

* SE BORRA LA PANTALLA Y SE ABRE EL FICHERO.

    DISPLAY " " ERASE.
    OPEN I-O AUTOS.

* PIDE LA MATRICULA A PARTIR DE LA CUAL SE HARA EL LISTADO.

    DISPLAY "MATRICULA...: [         ]"
            LINE 10 POSITION 20.
    ACCEPT MAT-AUT LINE 10 POSITION 34 NO BEEP.
```

INSTRUCCIONES RELATIVAS A FICHEROS

```
      * SE POSICIONA EL PUNTERO DE LECTURA EN LA CLAVE INDICADA,
      * SI NO EXISTE NINGUNA CLAVE A PARTIR DE ELLA SE VA A NOEXISTE.

           START AUTOS KEY IS NOT < MAT-AUT
                                 INVALID KEY GO TO NOEXISTE.

      * SI EXISTE SE AVISA DE QUE SE CONECTE LA IMPRESORA Y SE ESPERA
      * LA PULSACION DE UNA TECLA.

           DISPLAY " CONECTE LA IMPRESORA " LINE 16 POSITION 30
                   REVERSE BLINK.
           ACCEPT TECLA OFF NO BEEP.

      * SE ABRE EL FICHERO DE IMPRESORA Y SE IMPRIMEN LAS CABECERAS

           OPEN OUTPUT IMP.
           WRITE FIC-IMP FROM CABECERA1.
           WRITE FIC-IMP FROM CABECERA2.

      * LEE EN SECUENCIAL Y VA LISTANDO HASTA QUE SE LLEGUE AL FINAL
      * DEL FICHERO, EN ESE MOMENTO SE VA A FINALIZAR.

       LEER.

           READ AUTOS NEXT RECORD AT END GO TO FINALIZAR.

      * SE MUEVEN LOS CAMPOS DEL REGISTRO A LOS DEL GRUPO QUE VA A
      * FORMAR EL CUERPO DEL LISTADO.

           MOVE MAT-AUT TO MAT-OUT.
           MOVE PRO-AUT TO PRO-OUT.
           MOVE MOD-AUT TO MOD-OUT.
           MOVE DIA-AUT TO DIA-OUT.
           MOVE MES-AUT TO MES-OUT.
           MOVE ANO-AUT TO ANO-OUT.

      * SE ESCRIBE EL REGISTO EN IMPRESORA AVANZANDO 2 LINEAS.

           WRITE FIC-IMP FROM CUERPO AFTER ADVANCING 2 LINES.

      * SE VUELVE A SEGUIR LEYENDO.

           GO TO LEER.

       NOEXISTE.

      * A ESTA ETIQUETA LLEGARA CUANDO LE DEMOS UNA MATRICULA A PARTIR
      * DE LA CUAL NO HAY NINGUNA.
```

```
            DISPLAY "NO HAY NINGUN AUTOMOVIL A PARTIR DE ESE"
                    LINE 12 POSITION 25.
            DISPLAY " PULSE UNA TECLA " LINE 14 POSITION 30 REVERSE.
            ACCEPT TECLA.

        FINALIZAR.

      * AQUI LLEGARA CUANDO SE HA TERMINADO EL LISTADO O CUANDO NO
      * SE HA ENCONTRADO NINGUNA CLAVE A PARTIR DE LA MATRICULA
      * SOLICITADA.

      * CIERRA LOS FICHEROS, BORRA LA PANTALLA Y TERMINA.

            CLOSE AUTOS, IMP WITH NO REWIND.
            DISPLAY " " ERASE.
            STOP RUN.
```

Al ejecutarlo podrás comprobar como aunque no le des la matrícula completa buscará a partir de lo que le hallas dado, al listar lo harán por el orden alfabético que siguen las matrículas y no por el orden en que se fueron introduciendo. El programa se puede mejorar, por ejemplo puedes hacer que se liste de una cierta clave hasta otra, y no hasta el final como hace este programa.

10.10. LA INSTRUCCION UNLOCK.

UNLOCK..: Esta instrucción se utiliza solamente cuando se está trabajando en multipuesto, normalmente cuando se lee un registro en una estación, este registro se queda bloqueado, esta instrucción se utiliza para desbloquearlo. Tiene el siguiente formato.

> **UNLOCK nomfic RECORD..:** Donde 'nomfic' es el nombre del fichero del que se tiene bloqueado el registro, la ejecución de esta instrucción tendrá como efecto el desbloqueo de tal registro para que otras estaciones puedan acceder a él.

Con esta última hemos visto ya todas las instrucciones referentes a ficheros y practicamente todas las instrucciones Cobol. Vamos a ver a continuación una palabra Cobol que nos puede ahorrar largos ratos de escritura y despues veremos una sección que se puede incluir en la **PROCEDURE DIVISION.**

12.- LA CLAUSULA COPY.

COPY..: Con esta instrucción se puede introducir en un programa Cobol al compilar una librería que se encuentra en el disco. Una librería es una parte de programa que se vá a repetir en diversos programas y que con ella se ha formado un archivo aparte, de tal forma que cada vez que se necesita se llama con COPY sin tener que volver teclearlo. Tenemos el siguiente formato.

> COPY "nomlib"..: Donde 'nomlib' será el nombre de la librería que se quiere incluir en el programa, la ejecución de esta instrucción tendrá como resultado la apertura del archivo que contiene tal librería, esta se compilará como parte del programa y al terminar se cerrará. Si en la compilación se intenta leer una librería que no existe el compilador nos devolverá un error.

Para ver en acción esta instrucción y darnos cuenta del trabajo que nos puede ahorrar vamos a teclear los programas que tenemos a continuación. De lo que se trata es de simular una empresa de venta de ordenadores, primero tendremos un menú que nos dará opción a crear el fichero, entrar en el mantenimiento del fichero, hacer un listado del fichero o salir del programa. Los tres programas que van a utilizar el fichero tendrán una cosa en común, y es que en todos ellos tendremos que declarar el fichero donde vayamos a hacer el mantenimiento y la declaración de su registro; esa parte es la que vamos a hacer una librería y la llamaremos cuando nos haga falta. El fichero tendrá los campos: 'MARCA', 'MODELO', 'DISTRIBUIDOR', 'PVP' y 'MEMORIA'; la clave será la unión de los campos 'MARCA' y 'MODELO', ya que la 'MARCA' por si sólo no nos sirve, puede ser que una marca tenga varios modelos.

Lo primero que vamos a hacer es la creación de las librerías que nos van a hacer falta, y vamos a comenzar creando la librería que será la declaración de fichero; abre en tu editor un archivo con el nombre de 'ORDSEL.LIB' y teclea lo que sigue. Fíjate en las columnas, ya que se tienen que guardar igual que si estuvieras escribiendo el programa.

```
123456789012
      A  B
```

```
SELECT ORDENADOR;
       ASSIGN TO RANDOM, "FICORD.DAT";
       ORGANIZATION IS INDEXED;
       ACCESS MODE IS DYNAMIC;
       RECORD KEY IS MMO-ORD;
       FILE STATUS IS ERRORES.
```

Esta será la declaración del fichero, en la INPUT-OUTPUT en vez de poner la declaración del fichero en cada programa pondremos COPY "ORDSEL.LIB"., con lo cual nos ahorraremos el escribir tres veces la misma declaración. Vamos a hacer ahora lo mismo con la declaracion del registro. Abre en tu editor un archivo con el nombre de 'ORDFDR.LIB' y teclea lo que tienes a continuación.

```
123456789012
      A   B

      FD  ORDENADOR;
          RECORD CONTAINS 76 CHARACTERS;
          LABEL RECORD IS STANDARD;
          DATA RECORD IS FIC-ORD.

      01  FIC-ORD.
          03  MMO-ORD.
              05  MAR-ORD              PIC  X(15).
              05  MOD-ORD              PIC  X(10).
          03  DIS-ORD                  PIC  X(30).
          03  PVP-ORD                  PIC  9(06).
          03  MEM-ORD                  PIC  X(15).
```

Ya tenemos las dos librerias preparadas, vamos ahora a hacer los programas, primero haremos el programa que nos servirá de presentación y menú. Abre en tu editor un archivo con el nombre de 'PRUEBA51.CBL ' y teclea lo que sigue.

```
123456789012
      A   B

      IDENTIFICATION DIVISION.
      PROGRAM-ID. PRUEBA51.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
```

```
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  FECHA.
           03  DIA                          PIC 9(02).
           03  MES                          PIC 9(02).
           03  ANO                          PIC 9(02).

       77  OPCION                           PIC 9(01).
       77  LINEA                            PIC 9(02).
       77  ASTERISCOS                       PIC X(80) VALUE ALL "*".

       PROCEDURE DIVISION.
       COMIENZO.

      * SE HACE LA PRESENTACION DEL PROGRAMA

           DISPLAY ASTERISCOS ERASE LINE 1 POSITION 1.
           PERFORM LATERAL VARYING LINEA FROM 2 BY 1 UNTIL LINEA > 20.
           DISPLAY ASTERISCOS LINE 21 POSITION 1.

           DISPLAY " VENTA DE ORDENADORES " LINE  3 POSITION 29 REVERSE.
           DISPLAY "FECHA DE CREACION....:" LINE  7 POSITION 20 LOW.
           DISPLAY "FECHA DE INSTALACION.:" LINE  9 POSITION 20 LOW.
           DISPLAY "DISTRIBUIDOR.........:" LINE 11 POSITION 20 LOW.
           DISPLAY "FECHA ACTUAL.........:  /  /  "
                   LINE 15 POSITION 20 LOW.
           DISPLAY "18/11/1986" LINE  7 POSITION 42.
           DISPLAY "20/11/1986" LINE  9 POSITION 42.
           DISPLAY "CENTRO DE ESTUDIOS INFORMATICOS"
                   LINE 11 POSITION 42.


      * PETICION DE LA FECHA ACTUAL

           ACCEPT DIA LINE 15 POSITION 42 NO BEEP.
           ACCEPT MES LINE 15 POSITION 45 NO BEEP.
           ACCEPT ANO LINE 15 POSITION 48 NO BEEP.

       MENU.

      * IMPRIME EL MENU Y PIDE OPCION.


           DISPLAY " " ERASE.
```

```
           DISPLAY "1.- CREACION DEL FICHERO" LINE 7 POSITION 20.
           DISPLAY "2.- MANTENIMIENTO DEL FICHERO"
                   LINE 9 POSITION 20.
           DISPLAY "3.- LISTADO DEL FICHERO" LINE 11 POSITION 20.
           DISPLAY "4.- SALIR DEL PROGRAMA" LINE 13 POSITION 20.
           DISPLAY "ELIJE OPCION.[ ]" LINE 18 POSITION 10.

       PEDIROPCION.

           ACCEPT OPCION LINE 18 POSITION 24 NO BEEP.

      * SEGUN LA OPCION PULSADA SE VA A UNA PARTE U OTRA DEL PROGRAMA.

           GO TO CREACION, MANTENIMIENTO, LISTADO, SALIDA
                DEPENDING ON OPCION.

      * SI SE LLEGA AQUI ES QUE LA OPCION ES INVALIDA.

           GO TO PEDIROPCION.

       CREACION.
      * SE LLAMA AL PROGRAMA DE CREACION Y SE VUELVE AL MENU.

           CALL "PRUEBA52" USING FECHA.
           GO TO MENU.

       MANTENIMIENTO.

      * SE LLAMA AL PROGRAMA DE MANTENIMIENTO Y SE VUELVE AL MENU.

           CALL "PRUEBA53" USING FECHA.
           GO TO MENU.

       LISTADO.

      * SE LLAMA AL PROGRAMA DE LISTADO Y SE VUELVE AL MENU.

           CALL "PRUEBA54" USING FECHA.
           GO TO MENU.

       SALIDA.

      * BORRA LA PANTALLA Y TERMINA.

           DISPLAY " " ERASE.
           STOP RUN.
```

```
       LATERAL.

      * ESTA RUTINA IMPRIMIRA UN ASTERISCO A CADA LADO DE LA PANTALLA.

          DISPLAY "*" LINE LINEA POSITION 1.
          DISPLAY "*" LINE LINEA POSITION 80.
```

Este será el primer programa, lo primero que hace es imprimir una pantalla de presentación donde nos dice el nombre del programa, la fecha en que se creó e instaló y el distribuidor del programa, nos pregunta la fecha y tras ello se pasa al menú. Como ves pasamos la fecha a cada uno de los programas que llamamos, la fecha la vamos a utilizar en el listado. Vamos a hacer ahora el programa que creará el fichero, abre en tu editor un archivo con el nombre de 'PRUEBA52.CBL' y teclea lo siguiente.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA52.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

          COPY "ORDSEL.LIB".

       DATA DIVISION.
       FILE SECTION.

       COPY "ORDFDR.LIB".

       WORKING-STORAGE SECTION.

       77  ERRORES                    PIC X(02).

       LINKAGE SECTION.

       01  FECHA.
           03  DIA                    PIC  9(02).
           03  MES                    PIC  9(02).
           03  ANO                    PIC  9(02).
```

```
       PROCEDURE DIVISION.
       COMIENZO.

      * SE BORRA LA PANTALLA Y SE IMPRIME UN MENSAJE DE CREANDO.

           DISPLAY " C R E A N D O    F I C H E R O "
                   LINE 12 POSITION 25 REVERSE ERASE.

      * SE ABRE EL FICHERO EN MODO OUTPUT PARA CREARLO.

           OPEN OUTPUT ORDENADOR.

      * SE CIERRA Y SE VUELVE AL PROGRAMA MENU.

           CLOSE ORDENADOR.

       SALIR.

           EXIT PROGRAM.
```

Este será el programa de creación de fichero, donde estaría la definición del fichero y del registro hemos puesto el COPY de dos librerías, lo cual tendrá como efecto que al compilar estas librerías se abran y se incluyan en la compilación. A continuación vamos a hacer el tercer programa que será el de mantenimiento del fichero, abre en tu editor un archivo con el nombre de 'PRUEBA53.CBL' y teclea lo que tienes a continuación.

```
123456789012
       A   B
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA53.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           COPY "ORDSEL.LIB".

       DATA DIVISION.
       FILE SECTION.
```

LA CLAUSULA COPY

```
       COPY "ORDFDR.LIB".

       WORKING-STORAGE SECTION.

       77  ERRORES                          PIC X(02).
       77  OPCION                           PIC X(01).
       77  NUMCAM                           PIC 9(01).

       LINKAGE SECTION.

       01  FECHA.
           03  DIA                    PIC  9(02).
           03  MES                    PIC  9(02).
           03  ANO                    PIC  9(02).

       PROCEDURE DIVISION.
       COMIENZO.

      * SE BORRA LA PANTALLA Y SE IMPRIME LA MASCARA DE PANTALLA.

           DISPLAY " M A N T E N I M I E N T O "
                   LINE 2 POSITION 30 REVERSE ERASE.
           DISPLAY "MARCA.........: [                  ]"
                   LINE 6 POSITION 30 LOW.
           DISPLAY "MODELO........: [            ]"
                   LINE 8 POSITION 30 LOW.
           DISPLAY "DISTRIBUIDOR.: [                                    ]"
                   LINE 10 POSITION 30 LOW.
           DISPLAY "P.V.P.........: [       ]"
                   LINE 12 POSITION 30 LOW.
           DISPLAY "MEMORIA.......: [                  ]"
                   LINE 14 POSITION 30 LOW.
           DISPLAY "A=ALTAS/B BAJAS/M MODIFICAR/S=SALIR. [ ]"
                   LINE 18 POSITION 10.

      * SE ABRE EL FICHERO.

           OPEN I O ORDENADOR.

       PEDIRMARCA.

      * PIDE LA MARCA Y EL MODELO, JUNTOS FORMARAN LA CLAVE.

           ACCEPT MAR-ORD LINE  6 POSITION 45 NO BEEP.
           IF MAR-ORD = SPACES GO TO SALIR.
           ACCEPT MOD-ORD LINE  8 POSITION 45 NO BEEP.

      * CON LA CLAVE LEERA DEL FICHERO, SI DA ERROR ES QUE NO EXISTE.
```

```
           READ ORDENADOR INVALID KEY GO TO NOEXISTE.

       * SI NO DA ERROR ES QUE EXISTE, SE IMPRIMEN LOS CAMPOS.

           DISPLAY DIS-ORD LINE 10 POSITION 45.
           DISPLAY PVP-ORD LINE 12 POSITION 45.
           DISPLAY MEM-ORD LINE 14 POSITION 45.

       * TRAS IMPRIMIR SE VA A PEDIR OPCION.

           GO TO PEDIROPCION.

        NOEXISTE.

       * A ESTA ESTIQUETA SE LLEGA CUANDO EL REGISTRO NO EXISTE, SE
       * PIDEN LOS DEMAS CAMPOS.

           ACCEPT DIS-ORD LINE 10 POSITION 45 NO BEEP.
           ACCEPT PVP-ORD LINE 12 POSITION 45 NO BEEP.
           ACCEPT MEM-ORD LINE 14 POSITION 45 NO BEEP.

        PEDIROPCION.

       .* UNA VEZ HA PEDIDO O IMPRESO LOS CAMPOS PIDE OPCION.

           ACCEPT OPCION LINE 18 POSITION 47 NO BEEP.

       * SEGUN LA OPCION SE VA A UNA PARTE U OTRA DEL PROGRAMA.

           IF OPCION = "A" OR OPCION = "a" GO TO ALTAS.
           IF OPCION = "B" OR OPCION = "b" GO TO ALTAS.
           IF OPCION = "M" OR OPCION = "m" GO TO MODIFICAR.
           IF OPCION = "S" OR OPCION = "s" GO TO ABANDONAR.

       * SI SE LLEGA AQUI ES QUE LA OPCION ERA INVALIDA.

           GO TO PEDIROPCION.

        ALTAS.

       * SE GRABA EL REGISTRO, SI DA ERROR SE REESCRIBE PARA GRABAR
       * POSIBLES MODIFICACIONES.

           WRITE FIC-ORD INVALID KEY REWRITE FIC-ORD.

       * TRAS ESCRIBIR BORRA LA PANTALLA Y VUELVE A PEDIRMARCA.
```

```
           PERFORM BORRAR.
           GO TO PEDIRMARCA.

       BAJAS.

      * SE BORRA EL REGISTRO,  SI DA ERROR ES QUE NO EXISTIA Y SE DEJA

           DELETE ORDENADOR INVALID KEY MOVE OPCION TO OPCION.

      * SE BORRA LA PANTALLA Y SE VUELVE A PEDIR MARCA.

           PERFORM BORRAR.
           GO TO PEDIRMARCA.

       MODIFICAR.

      * PRIMERO SE PIDE EL NUMERO DE CAMPO A MODIFICAR.

           DISPLAY "CAMPO A MODIFICAR.[ ]" LINE 20 POSITION 10.
           ACCEPT NUMCAM LINE 20 POSITION 29 NO BEEP.

      * SI EL NUMERO ES CERO SE VUELVE A PEDIR OPCION.

           IF NUMCAM = 0 DISPLAY SPACES SIZE 40 LINE 20 POSITION 10,
           GO TO PEDIROPCION.

      * DEPENDIENDO DEL CAMPO PULSADO SE VA UN U OTRO.

           GO TO PEDIR1, PEDIR2, PEDIR3 DEPENDING ON NUMCAM.

      * SI SE LLEGA AQUI ES QUE EL NUMERO DE CAMPO ES INVALIDO.

           GO TO MODIFICAR.

       PEDIR1.

      * MODIFICACION DEL DISTRIBUIDOR.

           ACCEPT DIS-ORD LINE 10 POSITION 45 NO BEEP.

      * TRAS PEDIR SE VUELVE A MODIFICAR.

           GO TO MODIFICAR.

       PEDIR2.

      * MODIFICACION DEL PRECIO VENTA PUBLICO
```

```
           ACCEPT PVP-ORD LINE 12 POSITION 45 NO BEEP.
       * TRAS PEDIR SE VUELVE A MODIFICAR.
           GO TO MODIFICAR.
        PEDIR3.
       * MODIFICACION DE LA MEMORIA.
           ACCEPT MEM-ORD LINE 14 POSITION 45 NO BEEP.
       * TRAS PEDIR SE VUELVE A MODIFICAR.
           GO TO MODIFICAR.
        ABANDONAR.
       * PARA ABANDONAR SE BORRA LA PANTALLA Y SE VUELVE A PEDIR OPCION.
           PERFORM BORRAR.
           GO TO PEDIRMARCA.
        BORRAR.
       * RUTINA PARA BORRAR LOS CAMPOS DE LA PANTALLA.
           DISPLAY SPACES SIZE 15 LINE  6 POSITION 45.
           DISPLAY SPACES SIZE 10 LINE  8 POSITION 45.
           DISPLAY SPACES SIZE 30 LINE 10 POSITION 45.
           DISPLAY SPACES SIZE  6 LINE 12 POSITION 45.
           DISPLAY SPACES SIZE 15 LINE 14 POSITION 45.
        SALIR.
       * SE CIERRAN LOS FICHEROS Y SE VUELVE AL PROGRAMA ANTERIOR.
           CLOSE ORDENADOR.
        SALIR1.
           EXIT PROGRAM.
```

Por último vamos a hacer el programa que hará el listado de ordenadores, para ello pediremos desde que ordenador y modelo hasta que ordenador y modelo se quiere listar, con esto nos posicionaremos en el fichero e iremos listando hasta llegar al

ordenador y modelo dados como fin. Abre en tu editor un archivo con el nombre de 'PRUEBA53.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA54.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           COPY "ORDSEL.LIB".

           SELECT IMP;
                  ASSIGN TO PRINT, "PRINTER";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       COPY "ORDFDR.LIB".

       FD  IMP;
           RECORD CONTAINS 80 CHARACTERS;
           LABEL RECORD IS OMITTED;
           DATA RECORD IS FIC-IMP.

       01  FIC-IMP                        PIC X(80).

       WORKING-STORAGE SECTION.

       77  ERRORES                        PIC X(02).
       77  TECLA                          PIC X(01).
       01  CLV-FIN.
           03  MAR-FIN                    PIC X(15).
           03  MOD-FIN                    PIC X(10).

       01  CABECERA1.
           03  FILLER                     PIC X(15) VALUE "MARCA".
           03  FILLER                     PIC X(01) VALUE SPACES.
```

```
           03  FILLER                  PIC X(10) VALUE "MODELO".
           03  FILLER                  PIC X(01) VALUE SPACES.
           03  FILLER                  PIC X(30) VALUE "DISTRIBUIDOR".
           03  FILLER                  PIC X(01) VALUE SPACES.
           03  FILLER                  PIC X(06) VALUE "P.V.P.".
           03  FILLER                  PIC X(01) VALUE SPACES.
           03  FILLER                  PIC X(15) VALUE "MEMORIA".

       01  CABECERA2                   PIC X(80) VALUE ALL "-".

       01  CUERPO.
           03  MAR-OUT                 PIC X(15).
           03  FILLER                  PIC X(01) VALUE SPACES.
           03  MOD-OUT                 PIC X(10).
           03  FILLER                  PIC X(01) VALUE SPACES.
           03  DIS-OUT                 PIC X(30).
           03  FILLER                  PIC X(01) VALUE SPACES.
           03  PVP-OUT                 PIC Z(06).
           03  FILLER                  PIC X(01) VALUE SPACES.
           03  MEM-OUT                 PIC X(15).

       LINKAGE SECTION.

       01  FECHA.
           03  DIA                     PIC 9(02).
           03  MES                     PIC 9(02).
           03  ANO                     PIC 9(02).

       PROCEDURE DIVISION.
       COMIENZO.

      * BORRA LA PANTALLA Y ABRE EL FICHERO DE ORDENADORES.

           DISPLAY " " ERASE.
           OPEN I-O ORDENADOR.

      * PREGUNTA LA MARCA Y MODELO DE FIN Y LA MARCA Y MODELO DE INICIO.

           DISPLAY " I N I C I O " LINE 4 POSITION 10 REVERSE.
           DISPLAY "MARCA....: [                ]"
                   LINE 6 POSITION 10.
           DISPLAY "MODELO...: [          ]"
                   LINE 8 POSITION 10.
           DISPLAY " F I N " LINE 14 POSITION 10 REVERSE.
           DISPLAY "MARCA....: [                ]"
                   LINE 16 POSITION 10.
           DISPLAY "MODELO...: [          ]"
                   LINE 18 POSITION 10.
```

```
        ACCEPT MAR-ORD LINE 6 POSITION 21 NO BEEP.
        ACCEPT MOD-ORD LINE 8 POSITION 21 NO BEEP.

    * BUSCA SI A PARTIR DE ESA MARCA Y MODELO EXISTE ALGUN ORDENADOR.
    * SI DA ERROR ES QUE NO EXISTE NINGUNO, SE VA A LA ETIQUETA NOHAY.

        START ORDENADOR KEY IS NOT < MMO-ORD
              INVALID KEY GO TO NOHAY.

    * SI NO DA ERROR SE PIDE LA MARCA Y MODELO DE FIN.

        ACCEPT MAR-FIN LINE 16 POSITION 21 NO BEEP.
        ACCEPT MOD-FIN LINE 18 POSITION 21 NO BEEP.

    * AVISA PARA QUE PULSE UNA TECLA.

        DISPLAY " PULSA UNA TECLA " LINE 11 POSITION 32 REVERSE.
        ACCEPT TECLA NO BEEP.

    * ABRE LA IMPRESORA.

        OPEN OUTPUT IMP.

    * SE IMPRIMEN LAS CABECERAS.

        WRITE FIC-IMP FROM CABECERA1.
        WRITE FIC-IMP FROM CABECERA2.

    * LEE EN SECUENCAL HASTA QUE LLEGUE AL FINAL DEL FICHERO O
    * SE LLEGUE A LA CLAVE FINAL.


     LEER.

        READ ORDENADOR NEXT RECORD AT END GO TO FINFICHERO.
        IF MMO-ORD > CLV-FIN GO TO FINLISTADO.

    * SI NO SE LLEGA AL FINAL DE FICHERO NI DE LISTADO SE IMPRIME.

        MOVE MAR-ORD TO MAR-OUT.
        MOVE MOD-ORD TO MOD-OUT.
        MOVE DIS-ORD TO DIS-OUT.
        MOVE PVP-ORD TO PVP-OUT.
        MOVE MEM-ORD TO MEM-OUT.

        WRITE FIC-IMP FROM CUERPO AFTER ADVANCING 2 LINES.

    * TRAS IMPRIMIR SE VUELVE A SEGUIR LEYENDO.
```

```
        GO TO LEER.

     FINFICHERO.

    * SI SE LLEGA A FIN DE FICHERO SIN LLEGAR A LA CLAVE FINAL SE
    * INDICA EN PANTALLA.

        DISPLAY " NO EXISTE LA CLAVE FINAL " LINE 22 POSITION 30.

     FINLISTADO.

    * AL LLEGAR AL FIN DEL LISTADO SE CIERRA EL FICHERO Y SE VUELVE
    * AL PROGRAMA ANTERIOR.

        CLOSE ORDENADOR, IMP WITH NO REWIND.

     SALIR.

        EXIT PROGRAM.

     NOHAY.

    * AQUI LLEGA CUANDO NO SE HA ENCONTRADO NINGUNA CLAVE A PARTIR DE
    * LA INDICADA.

        DISPLAY " NO EXISTE NINGUNA CLAVE A PARTIR DE ESTA "
                LINE 11 POSITION 25 REVERSE.

    * LLAMA A LA ETIQUETA ESPERA 500 VECES Y SE VA A FINLISTADO.
          PERFORM ESPERA 500 TIMES.
          GO TO FINLISTADO.

      ESPERA.

      * ESTA ETIQUETA SOLO SIRVE PARA PERDER TIEMPO.

          MOVE TECLA TO TECLA.
```

Este es el último, tras compilar los cuatro ejecuta 'PRUEBA51', primero aparecerá una pantalla con algunos datos sobre el programa y preguntará la fecha, tras esto dará entrada al menú, lo primero que debemos hacer es crear el archivo, tras esto podremos entrar en el mantenimiento y comenzar a dar de alta ordenadores, cuando querramos salir bastará que como marca pulsemos return. Cuando saquemos un listado nos pedirá la clave de inicio y fin para el listado, si en principio pulsas return listará desde el principio.

12.- LAS DECLARATIVES.

Por último vamos a ver una sección que la **PROCEDURE DIVISION** puede tener, esta sección nos vá a servir para controlar algunos errores, hasta ahora hemos visto que si en la ejecución de un programa no teníamos antes creado el fichero el programa se detenía devolviéndonos un error, igual ocurre si duplicamos una clave o alguna cosa parecida, para controlar todos estos errores tenemos esta sección que tiene el siguiente formato.

```
123456789012
      A  B

      PROCEDURE DIVISION.
      DECLARATIVES.

      nombre SECTION.

          USE AFTER STANDARD ERROR PROCEDURE ON modo.

      nomparrafo.

          .......
          .......

      nombre SECTION.

          USE AFTER STANDARD ERROR PROCEDURE ON modo.

      nomparrafo.

          .......
          .......

      END DECLARATIVES.

      nombre SECTION.
```

Este es el formato general, vamos a ver cada uno de los párrafos.

DECLARATIVES..: Con este párrafo le indicamos al programa que comenzamos las **DECLARATIVES**, ha de ser el primer párrafo tras **PROCEDURE DIVISION.**

nombre SECTION..: Donde 'nombre' es cualquier nombre puesto por nosotros, cada vez que se inicien las declaraciones de un nuevo fichero se ha de comenzar con una sección de esta forma.

USE AFTER STANDARD ERROR PROCEDURE ON modo..: Con este párrafo le indicamos al programa para que fichero o ficheros esta hecho el siguiente párrafo, 'modo' puede ser el nombre lógico de un fichero, en este caso cualquier error que se produzca en el fichero causará la transferencia del programa a esta sección de las **DECLARATIVES**; 'modo' también puede ser una forma de apertura de fichero como **INPUT**, **OUTPUT**, I-O o **EXTEND**, el resultado es la transferencia de control del programa a esta sección en cuanto se produzca un error en salida, entrada, entrada salida o extendido respectivamente.

nomparrafo..: Debe ser un nombre de párrafo creado por nosotros, a partir de él podremos especificar lo que hay que hacer en caso de haberse producido un error, podemos preguntar por el campo declarado como **FILE STATUS**, averiguar el error que se produce y actuar en consecuencia. A la terminación de esta sección se puede declarar otra sección para otro fichero o modo.

END DECLARATIVES..: Con este párrafo le indicamos al programa que se han terminado las **DECLARATIVES**, a continuación de él para comenzar el programa propiamente dicho tendremos que poner un nombre de sección.

Con las **DECLARATIVES** y los códigos de error que pueden surgir en los ficheros se pueden controlar muchos errores, los códigos de error los podrás encontrar en un apéndice al final del libro.

Vamos a hacer un programa en el cual usemos las **DECLARATIVES**, en él abriremos un fichero en modo I-O sin haberlo creado antes, esto nos devolverá un error y se irá a las declaratives, el error que se produce cuando se quiere abrir un fichero que no existe es el 94. Abre en tu editor un archivo con el nombre de 'PRUEBA55.CBL' y teclea lo que sigue.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA55.
```

LAS DECLARATIVES

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT FICHERO;
                  ASSIGN TO RANDOM, "FICHERO.DAT";
                  ORGANIZATION IS INDEXED;
                  ACCESS MODE IS DYNAMIC;
                  RECORD KEY IS CLV-FIC;
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

       FD  FICHERO;
           RECORD CONTAINS 80 CHARACTERS;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-FIC.

       01  FIC-FIC.
           03  CLV-FIC                    PIC X(30).
           03  CAM-FIC                    PIC X(50).

       WORKING-STORAGE SECTION.

       77  ERRORES                        PIC X(02).
       77  TECLA                          PIC X(01).

       PROCEDURE DIVISION.
       DECLARATIVES.

       I-O-FICHERO SECTION.

      * SE DECLARA PARA QUE FICHERO ES ESTA SECCION.

           USE AFTER STANDARD ERROR PROCEDURE ON FICHERO.

       I-O-PROCESO.

      * SE ESPECIFICA QUE HAY QUE HACER EN CASO DE ERROR.
      * PREGUNTAMOS SI EL CAMPO ERRORES TIENE UN 94, DE SER
      * ASI EL ERROR ES QUE EL FICHERO NO ECISTE, EN ESTE CASO NOS
      * VAMOS A LA ETIQUETA I-O-CREACION.
```

```
            IF ERRORES = 94 GO TO I-O-CREACION.

       * SI NO ERA ESE EL ERROR IMPRIMIMOS EN PANTALLA EL ERROR QUE
       * SE HALLA PRODUCIDO.

            DISPLAY "SE HA PRODUCIDO UN ERROR" LINE 10 POSITION 20.
            DISPLAY ERRORES LINE 10 POSITION 50.

       * SE VA A UNA ETIQUETA QUE ES EL FINAL DE LAS DECLARATIVES.

            GO TO FIN.

        I-O-CREACION.

       * A ESTA ETIQUETA SE LLEGARA EN CASO DE QUE EL FICHERO NO
       * EXISTA EN EL DISCO, SE ABRE EN MODO OUTPUT PARA CREARLO
       * SE CIERRA Y SE VUELVE A ABRIR EN MODO I-O, YA QUE CUANDO
       * SE VUELVA AL PROGRAMA SE LLEGARA A LA SIGUIENTE INSTRUCCION
       * DE DONDE SE PRODUJO EL ERROR QUE FUE AL INTENTAR ABRIR EL
       * FICHERO.

            OPEN OUTPUT FICHERO.
            CLOSE FICHERO.
            OPEN I-O FICHERO.

        FIN.

       * SE LLEGA A ESTA ETIQUETA QUE ES EL FIN DE LAS DECLARATIVES.
       * SE IMPRIME UN MENSAJE EN PANTALLA.

            DISPLAY " SALGO DE LAS DECLARATIVES " LINE 5 POSITION 20
                    REVERSE.

       * FIN DE LAS DECLARATIVES.

        END DECLARATIVES.

       * COMIENZO DEL PROGRAMA CON UN NOMBRE DE SECCION.

        PROGRAMA SECTION.
        COMIENZO.

       * AQUI COMENZARA EL PROGRAMA CUANDO SE EJECUTE, LAS DECLARATIVES
       * LAS SALTARA.

       * SE BORRA LA PANTALLA Y SE IMPRIME UN MENSAJE.
```

```
        DISPLAY " PULSA UNA TECLA PARA ABRIR EL FICHERO "
                LINE 12 POSITION 25 REVERSE ERASE.

    * ESPERA LA PULSACION DE UNA TECLA.

        ACCEPT TECLA OFF NO BEEP.

    * ABRE EL FICHERO EN MODO I-O, AL NO PODER SE IRA A LAS DECLARATIVES

        OPEN I-O FICHERO.

    * CUANDO VUELVA DE LAS DECLARATIVES SE LLEGARA AQUI

        DISPLAY " HE VUELTO DE LAS DECLARATIVES "
                LINE 14 POSITION 25 REVERSE.

    * ESPERA LA PULSACION DE UNA TECLA.

        ACCEPT TECLA OFF NO BEEP.

    * SE CIERRA EL FICHERO.

        CLOSE FICHERO.

    * SE VUELVE A CERRAR, AL CERRAR UN FICHERO QUE NO ESTA ABIERTO
    * NOS DARA UN ERROR, SE IRA A LAS DECLARATIVES, PERO ESTE ERROR
    * NO ESTA CONTROLADO EN LAS DECLARATIVES CON LO CUAL NOS IMPRIMIRA
    * EL CODIGO DE ERROR POR PANTALLA.

        CLOSE FICHERO.

    * CUANDO VUELVA TERMINARA.

        STOP RUN.
```

Al ejecutarlo podrás ver los distintos mensajes que aparecen por pantalla, estos te indicarán por donde se está ejecutando en cada momento; igual que se ha controlado el error de que el fichero nó está creado se puede controlar cualquier otro.

13.-- PROGRAMAS.

Hasta aquí hemos visto todas las instrucciones, párrafos y formatos del Cobol, a partir de aquí se irán proponiendo, desarrollando y explicando una serie de programas y pequeñas utilidades; dentro de cada uno de los programas se iran haciendo comentarios del funcionamiento del programa, ante cualquier duda del funcionamiento de una instrucción remito al lector al capítulo correspondiente de dicha instrucción.

13.1. DE MINUSCULAS A MAYUCULAS.

El primer programa que vamos a hacer nos vá a servir para inspeccionar una cadena cambiando todas las letras minúsculas por mayúsculas, esto nos puede ser bastante útil, sobre todo cuando se nos pida una clave de un fichero indexado, si una vez le damos la clave en minúsculas y otra vez en mayúsculas no lo encontrará, este programa nos vá a servir para solucionar este problema. El programa se llama 'PRUEBA56.CBL' puedes abrir en tu editor un archivo y tecleario, lo tienes a continuación.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA56.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  LMINUSCULAS                 PIC X(26) VALUE
           "abcdefghijklmnopqrstuvwxyz".

      * EN ESTA TABLA TENDREMOS UNA LETRA MINUSCULA POR ELEMENTO.

       01  MINUSCULAS REDEFINES LMINUSCULAS.
           03  MINUSCULA               PIC X(01) OCCURS 26 TIMES.

       01  LMAYUSCULAS                 PIC X(26) VALUE
           "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
```

```
      * EN ESTA TABLA TENDREMOS UNA LETRA MAYUSCULA POR ELEMENTO
      * CORRESPONDIENDOSE EN LUGAR CON LAS MUNUSCULAS.

       01  MAYUSCULAS REDEFINES LMAYUSCULAS.
           03  MAYUSCULA                PIC X(01) OCCURS 26 TIMES.

       77  PALABRA                      PIC X(50).
       77  INDICE                       PIC 9(02).
       77  TECLA                        PIC X(01).

       PROCEDURE DIVISION.
       COMIENZO.


      * SE PIDE LA PALABRA.

           ACCEPT PALABRA LINE 10 POSITION 20 ERASE NO BEEP PROMPT.

      * SE LLAMA A LA RUTINA BUSCA VARIANDO LA VARIABLE INDICE DESDE
      * 1 HASTA 26 QUE SON LAS LETRAS QUE HAY.

           PERFORM BUSCA VARYING INDICE FROM 1 BY 1 UNTIL INDICE > 26.

      * CUANDO VUELVE IMPRIME LA PALABRA.


           DISPLAY PALABRA LINE 10 POSITION 20.


      * ESPERA LA PULSACION DE UNA TECLA.

           ACCEPT TECLA OFF NO BEEP.

      * VUELVE AL PRINCIPIO

           GO TO COMIENZO.

       BUSCA.

      * ESTA RUTINA BUSCARA EN LA VARIABLE PALABRA LA LETRA MINUSCULA
      * ELEMENTO INDICE CAMBIANDOLA POR LA LETRA MAYUSCULA ELEMENTO
      * INDICE, COMO INDICE IRA INCREMENTANDOSE DESDE 1 HASTA 26 SE
      * BUSCARAN LA 26 LETRAS MINUSCULAS POSIBLES Y SE CAMBIARAN POR
      * MAYUSCULAS.

           INSPECT PALABRA REPLACING ALL MINUSCULA(INDICE) BY
                   MAYUSCULA(INDICE).
```

Cuando lo ejecutes te pedirá una cadena, mezcla en ella letras mayúsculas y minúsculas y verás como te devuelve toda la cadena en mayúsculas. Comprendiendo bien el programa no te será difícil invertirlo, es decir que te convierta las mayúsculas en minúsculas. A la hora de utilizarlo para varios programas de mantenimiento de ficheros cada vez que se necesite se puede llamar con **CALL**, con esta misma instrucción se le puede pasar la cadena y al volver la tendrías toda en mayúsculas.

13.2. EDITANDO PANTALLAS.

Hasta ahora cada vez que hemos necesitado una mascara de petición de datos o algo parecido la hemos hecho a base de **DISPLAY**, ¿no crees que sería mucho mas cómodo tener las pantallas en un fichero y leerlas cuando nos hicieran falta?, nos ahorraríamos mucho trabajo. Para poder hacerlo necesitaríamos meter las pantallas en un fichero donde poder leerlas despues, para ello nos hace falta un editor de pantalla. Un editor de pantalla es un programa que nos permite movernos por la pantalla, escribir y realizar algunas operaciones con la pantalla como borrar e insertar líneas, desplazar la pantalla, etc.., el Cobol no es quiza el lenguaje más indicado para hacer un editor de pantalla, pero lo vamos a hacer, quiza no sea gran cosa comparado con otros pero a nosotros nos será útil y además vamos a ver a fondo la utilización de instrucciones como **PERFORM** y **ACCEPT**.

Vamos a ver a continuación las opciones que se han insertado en este editor de pantalla y a explicar un poco el funcionamiento del programa.

Cuando ejecutemos el editor lo primero que aparecerá en pantalla es el cursor en la esquina superior izquierda, en la última línea aparecerá un texto indicando que pulsemos la combinación control Q para obtener ayuda, y a continuación hay dos números que son la línea y columna donde nos encontramos en ese momento. A partir de ese momento podremos empezar a trabajar con el editor, para ello podemos usar las teclas que se especifican a continuación. Para detectar si lo que se ha pulsado es una tecla normal que se deba imprimir o es una tecla especial que tiene un cometido hemos usado la opción **ON EXCEPTION** de la instrucción **ACCEPT**, para ello hemos declarado a nivel 88 todos los códigos de las teclas que causarán excepción.

^Q..: Al pulsar esta combinación en pantalla aparecerá una especie de ventana en video inverso donde se nos dice las teclas que podemos usar para trabajar, no te preocupes si bajo la ventana tenías algo, cuando hallas

leido la ayuda pulsa una tecla y recuperarás la pantalla anterior, este mismo método se ha utilizado para todos los mensajes por pantalla, en lo que consiste es en tener la pantalla almacenada en una variable bidimensional como si fuera una red, con ello podemos recuperar en cualquier momento parte o toda la pantalla, ya que esta variable o red es el reflejo de la pantalla.

TECLAS DEL CURSOR..: Con las teclas de control del cursor nos podremos mover a cualquier zona de la pantalla dentro de sus límites, para que produzcan la excepción se declararon a nivel 88 el código de cada uno de sus códigos.

END..: Con esta tecla podremos invertir el estado del dibujo continuo, es decir, si está apagado se encienda y viceversa. El dibujo continuo cuando está conectado consiste en que por donde nos movamos con las teclas del cursor se vá imprimiendo el último caracter que hubieramos pulsado.

INS..: Con ella podemos activar y desactivar el modo de insercción, cuando está conectado al escribir algo todo lo que hay a la derecha se desplaza una columna.

DEL..: Con esta tecla podremos borrar el caracter que tengamos en ese momento bajo el cursor, tras borrarlo toda la línea a la derecha de la posición donde se ha borrado se desplazará una columna a la izquierda para ocupar ese vacío.

^N..: Con la pulsación de esta combinación nos podremos desplazar hasta el final de la línea actual, pero no hasta la columna 80, sino hasta el final de todo el texto que halla en esa línea, para hacerlo se vá buscando desde el final hasta el principio un caracter que no sea el espacio, si se encuentra ese es el final de la línea.

^Z..: Con esta combinación podremos irnos al principio de la línea actual, actúa a la inversa que la anterior opción.

^Y..: Con la pulsación de esta opción podremos desplazarnos una palabra a la izquierda.

^O..: Con esta combinación podremos desplazarnos hacia la derecha palabra a palabra.

ESC..: La pulsación de esta tecla nos dá paso a otra ventana donde tenemos las siguientes opciones.

ESC 0..: Nos dá paso a un segundo menú con unas opciones que veremos despues.

ESC 1..: La opción 1 del menú nos dá opción a hacer un scroll rotacional de la pantalla hacia arriba, este scroll consiste en que la primera línea pase a ser la última, la segunda a la primera y así sucesivamente.

ESC 2..: La segunda opción nos dá paso a hacer un scroll rotacional de la pantalla hacia abajo, funciona de forma inversa a la anterior opción, la última línea pasa a ser la primera, la penúltima a la última y así sucesivamente.

ESC 3..: Con esta se nos dá opción a hacer un scroll rotacional de la pantalla hacia la derecha, este consiste en que la columna 80 pase a ser la 1, la 79 pasa a ser la 80 y así sucesivamente.

ESC 4..: Esta opción es la inversa de la anterior, con ella podemos hacer un scroll rotacional de la pantalla hacia la izquierda, la columna 1 pasa a ser la 80, la 2 pasa a ser la 1 y así sucesivamente.

ESC 5..: Con esta opción podremos borrar la línea actual por completo, al borrar una línea las líneas inferiores se desplazan una línea arriba para ocupar el espacio libre.

ESC 6..: Con esta opción podremos insertar una línea por encima de la actual, al hacerlo todas las líneas inferiores se desplazan una posición abajo y la última línea se pierde.

ESC 7..: Con esta opción podremos cargar una pantalla del disco, para ello se nos pedirá el nombre y despues se leerá e imprimirá pasando a ser la pantalla de trabajo.

ESC 8..: Con esta opción podremos salvar la pantalla, para ello se nos preguntará por el nombre que le queremos dar a la pantalla, con este se creará un archivo en el disco y se creará la pantalla, tras ello el programa seguirá.

ESC 9..: Esta última opción nos facilita la salida del menú sin elegir ninguna de las opciones anteriores.

Estas son las opciones del primer menú por la tecla escape, vamos a ver las opciones del segundo menú, este aparece al pulsar 0 en el primero.

ESC 0 0..: Con esta opción podremos programar las teclas funcionales con un determinado caracter, al tiempo que se van programando van apareciendo en la última línea de la pantalla con su contenido, una vez efectuada la programación cada vez que pulsemos una de las teclas de función obtendremos el caracter que pusimos en ella.

ESC 0 1..: Con esta opción podremos borrar la pantalla, una vez hecho esto no podremos recuperarla.

ESC 0 2..: Esta opción nos permite guardar la pantalla actual, para ello utilizamos otra variable, esta pantalla despues se puede recuperar mediante otra opción, se suele utilizar cuando tenemos una pantalla hecha y vamos a hacerle unas modificaciones pero no estamos seguros de que vaya a quedar bien, en ese caso almacenamos la pantalla en memoria, hacemos las modificaciones y si no queda bien podremos recuperar la pantalla anterior.

ESC 0 3..: Esta opción no permitirá repetir un caracter en una línea, para ello nos pedirá el caracter a repetir y el número de veces que este se ha de repetir.

ESC 0 4..: Con esta opción podremos buscar un caraceter y sustituirlo por otro, para ello nos pedirá el caracter a buscar, el caracter por el que se ha de sustituir y desde que línea hasta que línea deseamos sustituir. Esta opción nos puede ahorrar trabajo, si por ejemplo empezamos una pantalla y queremos llenar desde la línea 7 hasta la 13 de asteriscos, en vez de teclearlos le podemos decir que busque desde la línea 7 hasta la 13 todos los espacios en blanco y los sustituya por asteriscos.

ESC 0 5..: Con esta opción podremos recuperar una pantalla que almacenamos en memoria, al hacerlo la pantalla almacenada en la varible auxiliar pasará a la red de la pantalla.

ESC 0 6..: Esta opción nos permite abandonar este menú sin elegir ninguna de las opciones anteriores.

ESC 0 7..: Esta última opción nos permitirá abandonar el editor.

Como ves el editor es bastante completo, a continuación tienes el listado, es bastante extenso pero vale la pena, en él van incluidos todos los comentarios necesarios para su mejor comprensión. Abre en tu editor un archivo con el nombre de 'PRUEBA57.CBL' y comienza a teclear.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA57.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

  * SE DECLARA EL FICHERO QUE NOS VA A SERVIR PARA GRABAR O CARGAR
  * LAS PANTALLAS QUE HAGAMOS, COMO NOMBRE FISICO EN EL DISCO PONEMOS
  * UNA VARIABLE, DE ESTA FORMA PODREMOS DARLE EL NOMBRE QUE QUERRAMOS
  * A LA PANTALLA. EL FICHERO SE DECLARA EN MODO DIRECTO Y NO
  * SECUENCIAL COMO CABRIA ESPERAR, LO HACEMOS ASI PORQUE EN UN
  * FICHERO SECUENCIAL NO PODRIAMOS ALMACENAR CARACTERES ESPAÑOLES
  * COMO LA 'Ñ' LAS VOCALES ACENTUADAS NI TAMPOCO SIGNOS GRAFICOS,
  * CON LO CUAL LA EDICION DE PANTALLAS SE VERIA LIMITADA.

       SELECT PANTA;
              ASSIGN TO RANDOM, NOM-PAN;
              ORGANIZATION IS RELATIVE;
              ACCESS MODE IS DYNAMIC;
              RELATIVE KEY IS CLV-PAN;
              FILE STATUS IS ERRORES.
```

```
      DATA DIVISION.
      FILE SECTION.

       FD  PANTA;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-PAN.

* EL REGISTRO DE PANTALLA SE DECLARA COMO UNA TABLA DE 80 CARACTERES
* POR 24 LINEAS,  ESTA ES LA ZONA DE PANTALLA QUE VAMOS A UTILIZAR.

 01  FIC-PAN.
     03  LIN-PAN                  PIC X(80) OCCURS 24 TIMES.

 WORKING-STORAGE SECTION.

* EN ESTA VARIABLE DECLARAMOS TODAS LAS TECLAS QUE CAUSARAN
* EXCEPCION A LA HORA DE PEDIR UN CARACTER.

 01  EXCEP                        PIC 9(03) COMP.

* LA TECLA ESCAPE TIENE EL CODIGO 27,  ESTA TECLA NOS LLEVARA A
* UN MENU DE OPCIONES.

     88  ESCAP VALUE IS 027.

* 16 ES EL CODIGO DEL CURSOR A LA IZQUIERDA.

     88  IZQUI VALUE IS 016.

* 18 ES EL CODIGO DEL CURSOR A LA DERECHA.

     88  DEREC VALUE IS 018.

* 13 ES EL CODIGO DE LA TECLA CURSOR ARRIBA,  ESTE TAMBIEN ES
* EL CODIGO DE LA TECLA RETURN,  PERO ESTO NO INTERFIERE PARA
* NADA EN EL FUNCIONAMIENTO DEL PROGRAMA.

     88  ARRIB VALUE IS 013.

* 21 ES EL CODIGO DEL CURSOR ABAJO.

     88  ABAJO VALUE IS 021.

* 17 ES EL CODIGO DE LA COMBINACION 'CTRL-Q',  AL PULSARLA EN
* PANTALLA APARECERA UN PEQUEÑA VENTANA DONDE SE NOS DIRA QUE
* TECLAS PODEMOS UTILIZAR A LA HORA DE EDITAR UNA PANTALLA.

     88  AYUDA VALUE IS 017.
```

```
      * 25 ES EL CODIGO DE LA COMBINACION 'CTRL Y', CON ELLA PODREMOS
      * DESPLAZARNOS HACIA LA IZQUIERDA PALABRA A PALABRA.

           88  PALIZ VALUE IS 025.

      * 15 ES EL CODIGO DE LA COMBINACION 'CTRL O', CON ELLA NOS
      * PODREMOS DESPLAZAR HACIA LA DERECHA PALABRA A PALABRA.

           88  PALDE VALUE IS 015.

      * 14 ES EL CODIGO DE LA COMBINACION 'CTRL-N', CON ELLA NOS
      * PODREMOS DESPLAZAR HASTA EL FINAL DE LA LINEA.

           88  FINLI VALUE IS 014.

      * 26 ES EL CODIGO DE LA COMBINACION 'CTRL-Z', CON ELLA NOS
      * PODREMOS DESPLAZAR HASTA EL PRINCIPIO DE LA LINEA.

           88  PRILI VALUE IS 026.

      * 23 ES EL CODIGO DE LA TECLA 'INS', POR MEDIO DE ELLA PODREMOS
      * ACTIVAR O DESACTIVAR EL MODO DE INSERCCION.

           88  INSER VALUE IS 023.

      * 24 ES EL CODIGO DE LA TECLA 'DEL', CON ELLA PODREMOS BORRAR
      * EL CARACTER QUE HALLA BAJO EL CURSOR.

           88  DELET VALUE IS 024.

      * 20 ES EL CODIGO DE LA TECLA 'END', POR MEDIO DE ELLA PODREMOS
      * ACTIVAR Y DESACTIVAR EL DIBUJO CONTINUO.

           88  CONTI VALUE IS 020.

      * 0 A 9 SON LOS CODIGOS DE LAS TECLAS F1 A F10, EN ELLAS PODREMOS
      * ALMACENAR UNOS CARACTERES Y RECUPERARLOS CON LA PULSACION DE LA
      * TECLA DONDE SE ALMACENO.

           88  F01   VALUE IS 000.
           88  F02   VALUE IS 001.
           88  F03   VALUE IS 002.
           88  F04   VALUE IS 003.
           88  F05   VALUE IS 004.
           88  F06   VALUE IS 005.
           88  F07   VALUE IS 006.
           88  F08   VALUE IS 007.
```

```
           88  F09    VALUE IS 008.
           88  F10    VALUE IS 009.

      * EN LA TABLA GRAFICOS SE VAN A ALMACENAR LOS CARACTERES GRAFICOS
      * QUE SE DEBEN IMPRIMIR EN CADA MOMENTO CON LA PULSACION DE LAS
      * TECLAS DE FUNCION.

       01  GRAFICOS.
           03  GRAFICO                    PIC X(01) OCCURS 10 TIMES.

      * LA TABLA PANTALLA NOS VA A SERVIR PARA CONVERTIR LA MISMA
      * EN UNA ESPECIE DE RED, ESTA RED SERA CARACTER, AL QUE DEBERMOS
      * HACER REFERNCIIA PONIENDO A CONTINUACION LA LINEA Y COLUMNA
      * DE LA PANTALLA A LA QUE NOS REFERIMOS. SE DECLARA CON UNA LINEA
      * Y UNA COLUMNA MAS DE LA QUE VAMOS A USAR, ESTO NOS VA A SERVIR
      * PARA CUANDO DESPLACEMOS LA PANTALLA TENER UNA VARIABLE AUXILIAR.

       01  PANTALLA.
           03  LINEAS                     OCCURS 25 TIMES.
               05  COLUMNAS               OCCURS 81 TIMES.
                   07  CARACTER           PIC X(01).

      * EN LA VARIABLE PANTALLA1 PODREMOS ALMACENAR EN CUALQUIER
      * MOMENTO LA PANTALLA Y DESPUES RECUPERARLA, UTILIZANDOLA BIEN
      * SE PUEDEN TENER DOS PANTALLAS AL MISMO TIEMPO EN MEMORIA.

       01  PANTALLA1.
           03  LINEAS1                    PIC X(80) OCCURS 24 TIMES.

      * LA VARIABLE LINEA IRA CONTANDO LA LINEA ACTUAL DEL CURSOR, NOS
      * SERVIRA PARA REFERIRNOS A UN ELEMENTO DE LA RED DE PANTALLA Y
      * PARA SITUARNOS ENTRE OTRAS COSAS.

       77  LINEA                          PIC 9(02) VALUE 1.

      * LAS TRES VARIABLES SIGUIENTES SE UTILIZAN COMO AUXILIARES A
      * LA HORA DE HACER DESPLAZAMIENTOS U OPERACIONES CON LA PANTALLA.

       77  LINEA1                         PIC 9(02).
       77  LINEA2                         PIC 9(02).
       77  LINEA3                         PIC 9(02).

      * LA VARIABLE COLUMNA VA A IR ALMACENANDO LA COLUMNA ACTUAL
      * DEL CURSOR, SU FINALIDAD SERA LA MISMA QUE LA DE LA LINEA.

       77  COLUMNA                        PIC 9(02) VALUE 1.

      * EN LA VARIABLE TECLA SE IRAN PIDIENDO LOS CARACTERES 1 A 1.
```

```
 77  TECLA                          PIC X(01).

* LA VARIABLE TECLA1 ALMACENARA EL ULTIMO CARACTER PULSADO PARA
* PODER UTILIZAR EL DIBUJO CONTINUO.

 77  TECLA1                        PIC X(01).

* LA VARIABLE OPCION-ESCAPE NOS SERVIRA PARA PEDIR OPCION EN LOS
* MENUS A LOS QUE DA ACCESO LA PULSACION DE LA TECLA ESC.

 77  OPCION-ESCAPE                 PIC 9(01).

* ERRORES ES EL FILE STATUS DEL FICHERO DE PANTALLA.

 77  ERRORES                       PIC X(02).

* NOM-PAN ES LA VARIABLE DONDE SE ALMACENARA EL NOMBRE DEL FICHERO
* DONDE SE DESEA GRABAR O DE DONDE SE DESEA CARGAR.

 77  NOM-PAN                       PIC X(12).

* CLV-PAN ES LA CLAVE DEL FICHERO DIRECTO, SE PONE A 1 Y NO SE
* TOCA A LO LARGO DE LA PROCEDURE, YA QUE SIEMPRE SE GRABARA EL
* REGISTRO 1.

 77  CLV-PAN                       PIC 9(01) VALUE 1.

* ESTAS DOS VARIABLES NOS VAN A SERVIR PARA LAS OPCIONES DE REPETIR
* BUSCAR Y SUSTITUIR UN CARACTER.

 77  CARAC                         PIC X(01).
 77  CARAC1                        PIC X(01).

* NUM NOS VA A SERVIR PARA PREGUNTAR CUANTAS VECES SE HA DE REPETIR
* UN CARACTER.

 77  NUM                           PIC 9(02).

* LA VARIABLE SW1 SERA UN INDICADOR QUE NOS DIRA SI ESTA ACTIVADO
* O DESACTIVADO EL MODO INSERCCION.

 77  SW1                           PIC 9(01) VALUE 0.

* LA VARIABLE SW2 SERA UN INDICADOR QUE NOS DIRA SI ESTA ACTIVADO
* O DESACTIVADO EL MODO CONTINUO.

 77  SW2                           PIC 9(01) VALUE 0.
```

```
      PROCEDURE DIVISION.
      DECLARATIVES.

    * PRINCIPIO DE LAS DECLARATIVES PARA EL FICHERO DE PANTALLA.

      I-O-ERROR SECTION.

          USE AFTER STANDARD ERROR PROCEDURE ON PANTA.

      I-O-CONTROL1.

    * SE PREGUNTA SI EL ERROR QUE SE HA PRODUCIDO ES EL 94, DE SER
    * ASI LO MAS POSIBLE ES QUE SE HALLA INTENTADO LEER UN FICHERO
    * DE PANTALLA QUE NO EXISTE, SI NO ES SE VA A I-O-CONTROL2 EN
    * CASO CONTRARIO SE NOS AVISA DE QUE EL FICHERO SOLICITADO NO
    * EXISTE.

          IF ERRORES NOT = "94" GO TO I-O-CONTROL2.
          DISPLAY " ESE FICHERO NO EXISTE " LINE 10
                  POSITION 20 REVERSE.

      I-O-PIDE.

    * SE NOS VUELVE A PEDIR EL NOMBREE DE LA PANTALLA Y SE VUELVE A
    * ABRIR, SI NO EXISTE SE VOLVERA A LLAMAR A LAS DECLARATIVES Y
    * NOS LO VOLVERA A PREGUNTAR, SI LA APERTURA HA SIDO SATISFACTORIA
    * SE VA A LA ETIQUETA I-O-CONTROLFIN.

          ACCEPT NOM-PAN LINE 12 POSITION 38 REVERSE NO BEEP.
          IF NOM-PAN = SPACES GO TO I-O-PIDE.
          OPEN INPUT PANTA.
          GO TO I-O-CONTROLFIN.

      I-O-CONTROL2.

    * SI EL ERROR NO ES EL 94 SE NOS AVISA POR PANTALLA DE QUE ERROR
    * ES EL QUE SE PRODUJO Y EL PROGRAMA SE DETIENE.

          DISPLAY " SE HA PRODUCIDO UN ERROR " LINE 10
                  POSITION 20 REVERSE.
          DISPLAY ERRORES LINE 10 POSITION 46 REVERSE.
          STOP RUN.

      I-O-CONTROLFIN.

    * CUANDO EL FICHERO HA SIDO ABIERTO SE VUELVE A LA ETIQUETA LEER
    * DONDE SE CONTINUARA CON LA OPERACION DE CARGA DE UNA PANTALLA.
```

```
* LA INCLUSION DE ESTA TRANSFERENCIA EN LAS DECLARATIVES DARA
* COMO RESULTADO UN 'WARNING' O AVISO AL FINAL DE LA COMPILACION
* PERO NO HAY QUEE TENERMO EN CUENTA YA QUE FUNCIONARA PERFECTAMENTE

    GO TO LEE.

 END DECLARATIVES.

* FIN DE LAS DECLARATIVES E INICIO DEL PROGRAMA PROPIAMENTE DICHO.

 PROGRAMAR SECTION.
 INICIO.

* IMPRIME EN LA ULTIMA LINEA DE LA PANTALLA UN MENSAJE INDICANDO
* QUE PARA OBTENER AYUDA SE HA DE PULSAR LA COMBINACION DE TECLAS
* CTRL-Q

    DISPLAY "^Q AYUDA" LINE 25 POSITION 60.

 COMIENZO.

* SE IMPRIME EN LA LINEA 25 LA LINEA Y COLUMNA DONDE ESTA EL
* CURSOR ACTUALMENTE.


    DISPLAY LINEA LINE 25 POSITION 70.
    DISPLAY COLUMNA LINE 25 POSITION 75.

* SE ESPERA LA PULSACION DE LA TECLA, SE USA EL FORMATO DE
* ON EXCEPTION PARA QUE EN CASO DE PULSAR ALGUNA DE LAS TECLAS
* DECLARADAS A NIVEL 88 SE VAYA A LA ETIQUETA CONTROLA.

    ACCEPT TECLA LINE LINEA POSITION COLUMNA NO BEEP
           OFF ON EXCEPTION EXCEP GO TO CONTROLA.


* SI NO ERA NINGUNA DE LAS TECLAS ESPECIALES MUEVE LA VARIABLE
* TECLA A TECLA1 PARA PREPARARLA EN CASO DE QUE SE USE EL DIBUJO
* CONTINUO.

    MOVE TECLA TO TECLA1.

* PREGUNTA SI SW1 ES IGUAL A 1, SI LA RESPUESTA ES AFIRMATIVA
* QUIERE DECIR QUE ESTA ACTIVADO EL MODO DE INSERCCION, EN ESE
* CASO SE VA A LA ETIQUETA INSERTANDO.

    IF SW1 = 1 GO TO INSERTANDO.
```

```
* SI NO SE ESTA INSERTANDO SE MUEVE LA TECLA PULSADA A LA POSICION
* DE LA RED INDICADA POR LINEA Y COLUMNA Y SE IMPRIME. SE LE SUMA
* 1 A LA COLUMNA PARA QUE CADA VEZ QUE SE PULSE UN CARACTER SE VAYA
* DESPLAZANDO HACIA LA DERECHA, SE PREGUNTA SI LA COLUMNA ES MAYOR
* DE 80 SE MUEVE 1 A LA COLUMNA Y SE LE SUMA UNA A LA LINEA, TRAS
* ELLO SE PREGUNTA SI LA LINEA ES MAYOR DE 24, DE SER ASI SE MUEVE
* 24 A LA LINEA.

    DISPLAY TECLA LINE LINEA POSITION COLUMNA,
            MOVE TECLA TO CARACTER(LINEA, COLUMNA),
            ADD 1 TO COLUMNA, IF COLUMNA > 80
            MOVE 1 TO COLUMNA, ADD 1 TO LINEA,
            IF LINEA > 24 MOVE 24 TO LINEA.

* TRAS HACER TODO EL PROCESO ANTERIOR SE VUELVE A COMIENZO
* PARA ESPERAR LA PULSACION DE OTRA TECLA.

    GO TO COMIENZO.

* A ESTA ETIQUETA SE LLEVA EN EL CASO DE QUE ESTE ACTIVADO EL
* MODO DE INSERCCION.

 INSERTANDO.

* SE LLAMA A LA RUTINA DESPLAZA VARIANDO LA VARIABLE LINEA1 DESDE
* 80 HASTA LA COLUMNA ACTUAL, ESTO SE HACE PARA ANTES DE IMPRIMIR
* LA TECLA PULSADA DESPLAZAR EL TEXTO UNA CASILLA A LA DERECHA, SE
* EMPIEZA POR LA COLUMNA 80 DESPLAZANDOLA A LA 81, LUEGO LA 79 SE
* DESPLAZA A LA 80 Y ASI SUCESIVAMENTE.

    PERFORM DESPLAZA VARYING LINEA1 FROM 80 BY -1
            UNTIL LINEA1 < COLUMNA.

* TRAS DESPLAZAR EL TEXTO SE MUEVE EL CARACTER PULSADO A LA POSICION
* CORRESPONDIENTE.

    MOVE TECLA TO CARACTER(LINEA, COLUMNA).

* SE IMPRIME LA LINEA DONDE SE HA HECHO LA INSERCCION YA CON LOS
* CAMBIOS HECHOS.

    DISPLAY LINEAS(LINEA) LINE LINEA POSITION 1.

* SE LE SUMA UNO A LA COLUMNA Y SE CONTROLA SI PASA DE 80, DE SER
* ASI SE LE SUMA UNO A LA LINEA Y LA COLUMNA SE PONE A 1, TRAS ELLO
* CONTROLA SI LA LINEA PASA DE 24, DE SER ASI SE PONE A 24.
```

```
           ADD 1 TO COLUMNA,  IF COLUMNA > 80 MOVE 1 TO COLUMNA,
                 ADD 1 TO LINEA,  IF LINEA > 24 MOVE 24 TO LINEA.

       * TRAS TODA LA OPERACION SE VUELVE A ESPERAR OTRA TECLA.

           GO TO COMIENZO.

       * ESTA RUTINA ES LLAMADA PARA DESPLAZAR EL TEXTO DE LA LINEA UNA
       * POSICION A LA DERECHA.

        DESPLAZA.

           COMPUTE LINEA2 = LINEA1 + 1.
           MOVE CARACTER(LINEA, LINEA1) TO CARACTER(LINEA, LINEA2).


       * A ESTA ETIQUETA SE LLEGA EN EL CASO DE QUE LA TECLA PULSADA
       * FUERA UNA DE LAS TECLAS DECLARADAS A NIVEL 88, SE CONTROLA
       * QUE TECLA HA SIDO.

        CONTROLA.

       * SI LA TECLA PULSADA HA SIDO ESCAPE SE VA A LA ETIQUETA ESCAPE.

           IF ESCAP GO TO ESCAPE.

       * SI LO QUE SE HA PULSADO ES LA TECLA DE MOVIMIENTO DE CURSOR A
       * A LA IZQUIERDA SE LE RESTA 1 A  COLUMNA, SE CONTROLA SI ES MENOR
       * DE 1,  DE SER ASI SE PONE A 1.

           IF IZQUI COMPUTE COLUMNA = COLUMNA - 1,
              IF COLUMNA < 1 MOVE 1 TO COLUMNA.

       * SI LO QUE SE HA PULSADO ES LA TECLA DE MOVIMIENTO DE CURSOR A
       * A LA DERECHA SE LE SUMA UNO A LA COLUMNA,  TRAS ELLO SE CONTROLA
       * SI LA COLUMNA PASA DE 80,  DE SER ASI SE MUEVE 80 A LA COLUMNA.

           IF DEREC COMPUTE COLUMNA = COLUMNA + 1,
              IF COLUMNA > 80 MOVE 80 TO COLUMNA.

       * SI SE HA PULSADO LA TECLA DE MOVIMIENTO DE CURSOR ARRIBA SE LE
       * RESTA 1 A LA LINEA Y SE CONTROLA QUE ESTA NO SEA MENOR DE 1.

           IF ARRIB COMPUTE LINEA = LINEA - 1,
              IF LINEA < 1 MOVE 1 TO LINEA.

       * SI LA TECLA PULSADA HA SIDO MOVIMIENTI DEL CURSOR ABAJO SE LE
       * SUMA 1 A LA LINEA Y SE CONTROLA QUE ESTA NO PASE DE 24.
```

```
           IF ABAJO COMPUTE LINEA = LINEA + 1,
              IF LINEA > 24 MOVE 24 TO LINEA.

       * SI LA TECLA PULSADA HA SIDO ALGUNA DE LAS CUATRO DE CONTROL DEL
       * CURSOR SE VA A LA ETIQUETA CONSINO, DONDE SE CONTROLARA SI SE
       * ESTABA TRABAJANDO EN DIBUJO CONTINUO.

           IF IZQUI OR DEREC OR ARRIB OR ABAJO GO TO CONTSINO.

       * SI LA TECLA PULSADA HA SIDO CONTROL Q, A LA QUE LLAMAMOS AYUDA
       * SE VA A LA ETIQUETA AYUDAMENU.

           IF AYUDA GO TO AYUDAMENU.

       * SI LA TECLA PULSADA HA SIDO LA DE PALABRA A LA IZQUIERDA SE VA
       * A LA ETIQUETA PALABRAIZQUIERDA.

           IF PALIZ GO TO PALABRAIZQUIERDA.

       * SI LA TECLA PULSADA HA SIDO LA DE PALABRA A LA DERECHA SE VA
       * A LA ETIQUETA PALABRADERECHA.

           IF PALDE GO TO PALABRADERECHA.

       * SI LA TECLA PULSADA HA SIDO LA DE IRSE AL FINAL DE LA LINEA SE
       * VA A LA ETIQUETA FINLINEA.

           IF FINLI GO TO FINLINEA.

       * SI LA ETIQUETA PULSADA HA SIDO LA DE IRSE AL PRINCIPIO DE LINE/
       * SE VA A LA ETIQUETA PRILINEA.

           IF PRILI GO TO PRILINEA.

       * SI LA TECLA PULSADA HA SIDO LA ACTIVACION/DESACTIVACION DEL
       * DIBUJO CONTINUO SE VA A LA ETIQUETA CONTINUO.

           IF CONTI GO TO CONTINUO.

       * SI LA TECLA PULSADA HA SIDO DEL SE VA A LA ETIQUETA DELETECAR.

           IF DELET GO TO DELETECAR.

       * SI LA TECLA PULSADA HA SIDO INS SE VA A LA ETIQUETA INSERTAR.

           IF INSER GO TO INSERTAR.

       * SI HA SIDO ALGUA DE LAS DIEZ TECLAS DE FUNCION SE MUEVE A LA
```

```
* VARIABLE LINEA1 EL NUMERO DE LA TECLA PULSADA Y SE VA A LA
* ETIQUETA TECLASF.

     IF F01 MOVE 1 TO LINEA1,  GO TO TECLASF.
     IF F02 MOVE 2 TO LINEA1,  GO TO TECLASF.
     IF F03 MOVE 3 TO LINEA1,  GO TO TECLASF.
     IF F04 MOVE 4 TO LINEA1,  GO TO TECLASF.
     IF F05 MOVE 5 TO LINEA1,  GO TO TECLASF.
     IF F06 MOVE 6 TO LINEA1,  GO TO TECLASF.
     IF F07 MOVE 7 TO LINEA1,  GO TO TECLASF.
     IF F08 MOVE 8 TO LINEA1,  GO TO TECLASF.
     IF F09 MOVE 9 TO LINEA1,  GO TO TECLASF.
     IF F10 MOVE 10 TO LINEA1,  GO TO TECLASF.

* SI NO HA SIFO NINGUNA DE ESTAS TECLAS SE VA AL PRINCIPIO,  ESTA
* INSTRUCCION EN TEORIA NO HARIA FALTA YA QUE OTRA TECLA NO
* DECLARADA NO PODRIA PRODUCIR UNA EXCEPCION,  PERO POR SI ACASO
* LA PONEMOS.

     GO TO COMIENZO.


 IMP.

* LLAMANDO A ESTA RUTINA Y PONIENDO EN LA VARIABLE LINEA EL
* NUMERO DE LINEA QUE SE DESEA REPONER ESTA SE IMPRIMIRA EN
* PANTALLA,  SE UTILIZA DESPUES DE LA APARICION DE LAS VENTANAS
* DE AYUDA Y OPCIONES PARA VOLVER A LA PANTALLA COMO ESTABA.

     DISPLAY LINEAS(LINEA) LINE LINEA POSITION 1 SIZE 80.

 AYUDAMENU.

* AQUI SE LLEGA CUANDO SE PULSA CONTROL Q,  IMPRIME EN PANTALLA
* UNA VENTANA CON UN TEXTO EXPLICATIVO DE LAS TECLAS QUE SE DEBEN
* DE USAR.


     DISPLAY " ------------------------------------------------
-            ""
              LINE 4 POSITION 15 REVERSE.
     DISPLAY " : UTILIZA LAS FLECHAS DEL CURSOR PARA MOVERTE.    :
-            ""
              LINE 5 POSITION 15 REVERSE.
     DISPLAY " : LA TECLA END ACTIVA/DESACTIVA DIBUJO CONTINUO. :
-            ""
              LINE 6 POSITION 15 REVERSE.
```

```
           DISPLAY " :  INS ACTIVA/DESACTIVA MODO DE INSERCCION.        :
      -          ""
                 LINE 7 POSITION 15 REVERSE.
           DISPLAY " :  DEL BORRA EL CARACTER BAJO EL CURSOR.           :
      -          ""
                 LINE 8 POSITION 15 REVERSE.
           DISPLAY " :  ^N IR FINAL LINEA        ^Z IR PRINCIPIO PAL.   :
      -          ""
                 LINE 9 POSITION 15 REVERSE.
           DISPLAY " :  ^Y PALABRA IZQUIERDA     ^O PALABRA DERECHA     :
      -          ""
                 LINE 10 POSITION 15 REVERSE.
           DISPLAY " :  ESC - OPCIONES POR SECUENCIA DE ESCAPE.         :
      -          ""
                 LINE 11 POSITION 15 REVERSE.
           DISPLAY " :                                                  :
      -          ""
                 LINE 12 POSITION 15 REVERSE.
           DISPLAY " ----------------------------------------------------
      -          ""
                 LINE 13 POSITION 15 REVERSE.

       * SE ESPERA A LA PULSACION DE UNA TECLA.

           ACCEPT TECLA OFF NO BEEP.

       * GUARDA LA VARIABLE LINEA EN LA AUXILIAR LINEA1.

           MOVE LINEA TO LINEA1.

       * LLAMA A LA RUTINA DE REPONER LINEAS VARIANDO LA VARIABLE LINEA
       * DESDE 4 HASTA 13 QUE SON LAS LINEAS QUE OCUPA LA VENTANA DE AYUDA.

           PERFORM IMP VARYING LINEA FROM 4 BY 1 UNTIL LINEA > 13.

       * DEVUELVE EL VALOR A LA VARIABLE LINEA.

           MOVE LINEA1 TO LINEA.

       * VUELVE A ESPERAR OTRA TECLA.

           GO TO COMIENZO.

        PALABRAIZQUIERDA.

       * ESTA RUTINA EFECTUA EL MOVIMIENTO DE UNA PALABRA A LA IZQUIERDA.
```

```
* RESTA 1 DE LA COLUMNA.

     SUBTRACT 1 FROM COLUMNA.

* SI YA SE ESTA AL PRINCIPIO DE LA PANTALLA SE VUELVE A COMIENZO.

     IF COLUMNA < 1 MOVE 1 TO COLUMNA GO TO COMIENZO.

* EN CASO CONTRARIO SE PREGUNTA SI LA POSICION EN LA QUE SE ESTA
* ES UN ESPACIO, DE SER ASI SE PARA Y SE VA A COMIENZO.

     IF CARACTER(LINEA, COLUMNA) = SPACES GO TO COMIENZO.

* SI NO ES UN ESPACIO SE VUELVE A PALABRAIZQUIERDA, CON LO CUAL
* SE LE RESTARA OTRO A COLUMNA Y SE SEGUIRA HASTA QUE SE ENCUENTR
* UN ESPACIO QUE ES EL SEPARADOR ENTRE DOS PALABRAS O SE LLEGUE
* AL PRINCIPIO DE LA LINEA.

     GO TO PALABRAIZQUIERDA.

 PALABRADERECHA.

* ESTA RUTINA HACE EL MOVIMIENTO A DERECHA PALABRA A PALABRA.
* FUNCIONA IGUAL QUE LA ANTERIOR PERO EN VEZ DE RESTAR SUMAMOS.

     ADD 1 TO COLUMNA.

     IF COLUMNA > 80 MOVE 80 TO COLUMNA, GO TO COMIENZO.
     IF CARACTER(LINEA, COLUMNA) = SPACES GO TO COMIENZO.
     GO TO PALABRADERECHA.

 FINLINEA.

* ESTA RUTINA ES LLAMADA CUANDO SE QUIERE IR AL FINAL DE LA LINEA.
* PRIMERO SE MUEVE 80 A LA COLUMNA, CON LO CUAL ESTAMOS AL FINAL DE
* LA LINEA.

     MOVE 80 TO COLUMNA.

 FINLINEA1.

* SI EL CARACTER DONDE SE ESTA ES UN ESPACIO Y TODAVIA NO SE HA
* LLEGADO AL PRINCIPIO DE PANTALLA SE LE RESTA UNO A LA COLUMNA
* Y SE SIGUE, DE ESTA FORMA SI HAY ALGO ESCRITO EN LA LINEA SE PONDRA
* AL FINAL DE TODO EL TEXTO.

     IF CARACTER(LINEA, COLUMNA) = SPACE AND COLUMNA > 1,
        SUBTRACT 1 FROM COLUMNA, GO TO FINLINEA1.
```

```
    * CUANDO HAYA LLEGADO AL FINAL SE VUELVE A COMIENZO.

        GO TO COMIENZO.

     PRILINEA.

    * ESTA RUTINA NOS LLEVA AL PRINCIPIO DE LINEA.
    * FUNCIONA IGUAL QUE LA ANTERIOR, PERO EN VEZ DE COMENZAR EN LA 80
    * E IR DISMINUYENDO SE COMIENZA EN LA 1 Y SE VA AUMENTANDO.

        MOVE 1 TO COLUMNA.

     PRILINEA1.

        IF CARACTER(LINEA, COLUMNA) = SPACE AND COLUMNA < 80,
           ADD 1 TO COLUMNA, GO TO PRILINEA1.
        GO TO COMIENZO.

     ESCAPE.

   * LA TECLA ESCAPE DA ENTRADA A UN PRIMER MENU QUE SE IMPRIME COMO
   * SI FUERA UNA VENTANA.


        DISPLAY "                                                    "
                LINE 6 POSITION 30 REVERSE.
        DISPLAY " :          SECUENCIAS DE ESCAPE                : "
                LINE 7 POSITION 30 REVERSE.
        DISPLAY " :  0.- SEGUNDO MENU DE SECUENCIAS              : "
                LINE 8 POSITION 30 REVERSE.
        DISPLAY " :  1.- SCROLL ARRIBA DE LA PANTALLA.           : "
                LINE 9 POSITION 30 REVERSE.
        DISPLAY " :  2.- SCROLL ABAJO DE LA PANTALLA.            : "
                LINE 10 POSITION 30 REVERSE.
        DISPLAY " :  3.- SCROLL DERECHA DE LA PANTALLA.          : "
                LINE 11 POSITION 30 REVERSE.
        DISPLAY " :  4.- SCROLL IZQUIERDA DE LA PANTALLA:        "
                LINE 12 POSITION 30 REVERSE.
        DISPLAY " :  5.- BORRAR LINEA COMPLETA.                  : "
                LINE 13 POSITION 30 REVERSE.
        DISPLAY " :  6.- INSERTAR UNA LINEA.                     : "
                LINE 14 POSITION 30 REVERSE.
        DISPLAY " :  7.- CARGAR UNA PANTALLA.                    : "
                LINE 15 POSITION 30 REVERSE.
        DISPLAY " :  8.- GRABAR LA PANTALLA.                     : "
                LINE 16 POSITION 30 REVERSE.
        DISPLAY " :  9.- ABANDONAR SECUENCIA DE ESCAPE.          : "
```

```
            LINE 17 POSITION 30 REVERSE.
        DISPLAY " :                                          : "
            LINE 18 POSITION 30 REVERSE.
        DISPLAY " :                ELIJE OPCION.[ ]          : "
            LINE 19 POSITION 30 REVERSE.
        DISPLAY " :                                          : "
            LINE 20 POSITION 30 REVERSE.
        DISPLAY " ------------------------------------------ "
            LINE 21 POSITION 30 REVERSE.

    PIDEOPCION1.

    * TRAS IMPRIMIR LA VENTANA SE PIDE OPCION.

        ACCEPT OPCION-ESCAPE LINE 19 POSITION 59 REVERSE NO BEEP.

    * SI LA OPCION TECLEADA NO ES NUMERICA SE VUELVE A PEDIR.

        IF OPCION-ESCAPE IS NOT NUMERIC GO TO PIDEOPCION1.

    * SE CONTROLA QUE LA OPCION ESTE ENTRE 0 Y 9.

        IF OPCION-ESCAPE < 0 OR OPCION-ESCAPE > 9 GO TO PIDEOPCION1.

    * SI LA OPCION FUE 0 SE VA A LA ETIQUET SEGUNDOMENU.

        IF OPCION-ESCAPE = 0 GO TO SEGUNDOMENU.

    * TRAS HABER TECLEADO LA OPCION LA OPCION SE REHACRE LA PANTALLA
    * CON LA RUTINA DE REPONER LINEAS.

        MOVE LINEA TO LINEA1.
        PERFORM IMP VARYING LINEA FROM 6 BY 1 UNTIL LINEA > 21.
        MOVE LINEA1 TO LINEA.


    * SEGUN LA OPCION PULSADA SE VA A UNA ETIQUETA U OTRA.

        GO TO SCROLLARRIBA, SCROLLABAJO, SCROLLDERECHA,
              SCROLLIZQUIERDA, BORRARLINEA, INSERTARLINEA,
              CARGAR, GRABAR, COMIENZO DEPENDING ON OPCION-ESCAPE.

     SEGUNDOMENU.


    * A ESTE SEGUNDO MENU SE LLEGA CON LA OPCION 0 DEL MENU ANTERIOR,
    * AQUI TENEMOS VARIAS OPCIONES MAS.
```

```
           DISPLAY "  ------------------------------------------------ "
                   LINE 9 POSITION 25 REVERSE .
           DISPLAY " :    SECUENCIAS DE ESCAPE MENU 2                 : "
                   LINE 10 POSITION 25 REVERSE .
           DISPLAY " :  0.  PROGRAMAR TECLAS F. CON GRAF.             : "
                   LINE 11 POSITION 25 REVERSE .
           DISPLAY " :  1.  BORRAR PANTALLA.                          : "
                   LINE 12 POSITION 25 REVERSE .
           DISPLAY " :  2.  ALMACENAR PANTALLA EN MEMORIA.            : "
                   LINE 13 POSITION 25 REVERSE .
           DISPLAY " :  3.  REPETICION DE UN CARACTER.                : "
                   LINE 14 POSITION 25 REVERSE .
           DISPLAY " :  4.  BUSCA/SUSTITUCION DE UN CARAC.            : "
                   LINE 15 POSITION 25 REVERSE .
           DISPLAY " :  5.  RECUPERAR PANTALLA DE MEMORIA.            : "
                   LINE 16 POSITION 25 REVERSE .
           DISPLAY " :  6.  ABANDONAR SECUENCIA DE ESCAPE.            : "
                   LINE 17 POSITION 25 REVERSE .
           DISPLAY " :  7.  SALIR DEL EDITOR.                         : "
                   LINE 18 POSITION 25 REVERSE .
           DISPLAY " :  8.                                            : "
                   LINE 19 POSITION 25 REVERSE .
           DISPLAY " :  9.                                            : "
                   LINE 20 POSITION 25 REVERSE .
           DISPLAY " :                                                : "
                   LINE 21 POSITION 25 REVERSE .
           DISPLAY " :                 ELIJE OPCION.[ ]               : "
                   LINE 22 POSITION 25 REVERSE .
           DISPLAY " :                                                : "
                   LINE 23 POSITION 25 REVERSE .
           DISPLAY " ------------------------------------------------ "
                   LINE 24 POSITION 25 REVERSE .

       PIDEOPCION2.

      * SE PIDE OPCION.

           ACCEPT OPCION-ESCAPE LINE 22 POSITION 54 REVERSE NO BEEP.

      * SE CONTROLA QUE LA OPCION PULSADA SEA NUMERICA Y ESTE ENTRE
      * 0 Y 7.

           IF OPCION-ESCAPE IS NOT NUMERIC GO TO PIDEOPCION2.
           IF OPCION-ESCAPE < 0 OR OPCION-ESCAPE > 7 GO TO PIDEOPCION2.

      * TRAS ESTO SE REHACE LA PANTALLA LLAMANDO A LA RUTINA DE REPONER
      * LINEAS.
```

```
       MOVE LINEA TO LINEA1.
       PERFORM IMP VARYING LINEA FROM 6 BY 1 UNTIL LINEA > 24.
       MOVE LINEA1 TO LINEA.

  * SEGUN LA OPCION PULSADA SE VA A UNA ETIQUETA U OTRA.

       GO TO BORRAPANTALLA, ALMACENA, REPETIR, BUSCAR-SUST
             RECUPERAR, COMIENZO, SALIR  DEPENDING ON OPCION-ESCAPE.


   PROGRAMARTECLAS.

  * ESTA ES LA OPCION 0 DEL SEGUNDO MENU,  POR MEDIO DE ELLA PODEMOS
  * PROGRAMAR LAS TECLAS FUNCIONALES CON UN CONTENIDO DE UN CARACTER.

  * PONEMOS EN PANTALLA UN TEXTO EXPLICATIVO DE LOS PASOS QUE SE DEBEN
  * SEGUIR.

       DISPLAY " A MEDIDA QUE VAYAS ESCUCHANDO "
               LINE 9 POSITION 25 REVERSE.
       DISPLAY " LOS PITITDOS  VE PULSANDO  EL "
               LINE 10 POSITION 25 REVERSE.
       DISPLAY " CARACTER QUE DESEAS GUARDAR.  "
               LINE 11 POSITION 25 REVERSE.

  * SE LLAMA A LA RUTINA PROGRAMA VARIANDO LA VARIABLE LINEA1 DESDE
  * 1 HASTA 10,  CON LO CUAL SE PEDIRAN LOS CONTENIDOS PARA LAS 10
  * TECLAS FUNCIONALES.

       PERFORM PROGRAMA VARYING LINEA1 FROM 1 BY 1
               UNTIL LINEA1 > 10.

  * TRAS HABER PROGRAMA LAS TECLAS SE REHACE LA PANTALLA.

       MOVE LINEA TO LINEA1.
       PERFORM IMP VARYING LINEA FROM 9 BY 1 UNTIL LINEA > 11.
       MOVE LINEA1 TO LINEA.

  * SE VUELVE A LA ETIQUETA COMIENZO.

       GO TO COMIENZO.

   PROGRAMA.

  * ESTA RUTINA ES LLAMADA PARA PROGRAMAR TECLAS FUNCIONALES.

  * PRIMERO SE ACEPTA EL CARACTER GRAFICO CORRESPONDIENTE A LA
  * VARIABLE LINEA1 QUE NOS FACILITAN.
```

```
      ACCEPT GRAFICO(LINEA1) OFF LINE 11 POSITION 58.

* CALCULAMOS EN QUE POSICION DE LA LINEA 25 HEMOS DE IMPRIMIR
* LA TECLA FUNCIONAL.

      COMPUTE LINEA2 = LINEA1 * 5.
      COMPUTE LINEA3 = LINEA2 - 3.

* CON ESTO SE IMPRIME LA TECLA FUNCIONAL NUMERO LINEA1 Y SU
* COTENIDO.

      DISPLAY "F" LINE 25 POSITION LINEA3 REVERSE.
      COMPUTE LINEA3 = LINEA3 + 1.
      DISPLAY LINEA1 LINE 25 POSITION LINEA3 REVERSE.
      DISPLAY GRAFICO(LINEA1) LINE 25 POSITION LINEA2.


 SCROLLARRIBA.

* ESTA RUTINA ES LLAMADA MEDIANTE LAS SECUENCIAS DE ESCAPE,
* LO QUE HACE ES DESPLAZAR LA PANTALLA UNA LINEA ARRIBA, LA PRIMERA
* LINEA PASA A SER LA ULTIMA.

      MOVE LINEAS(1) TO LINEAS(25).
      PERFORM ARRIBA VARYING LINEA1 FROM 2 BY 1 UNTIL LINEA1 > 25.
      MOVE LINEA TO LINEA1.
      PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
      MOVE LINEA1 TO LINEA.
      GO TO COMIENZO.


 ARRIBA.

      COMPUTE LINEA2 = LINEA1 - 1.
      MOVE LINEAS(LINEA1) TO LINEAS(LINEA2).

 SCROLLABAJO.

* ESTA RUTINA HACE LO CONTRARIO DE LA ANTERIOR, DESPLAZA LA
* PANTALLA UNA LINEA ABAJO, LA ULTIMA LINEA PASA A SER LA PRIMERA.

      PERFORM ABAJO VARYING LINEA1 FROM 24 BY -1 UNTIL LINEA1 < 1.
      MOVE LINEAS(25) TO LINEAS(1).
      MOVE LINEA TO LINEA1.
      PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
      MOVE LINEA1 TO LINEA.
      GO TO COMIENZO.
```

```
ABAJO.

    COMPUTE LINEA2 = LINEA1 + 1.
    MOVE LINEAS(LINEA1) TO LINEAS(LINEA2).

SCROLLDERECHA.

* ESTA RUTINA DESPLAZA LA PANTALLA UNA COLUMNA A LA DERECHA, LOS
* CARACTERES DE LA COLUMNA 80 PASAN A LA COLUMNA 1.

    DISPLAY " ESPERA " LINE 12 POSITION 30 REVERSE BLINK.
    PERFORM DERECHA VARYING LINEA1 FROM 80 BY -1
            UNTIL LINEA1 < 1 AFTER LINEA2
            FROM 1 BY 1 UNTIL LINEA2 > 24.
    PERFORM DERFIN VARYING LINEA1 FROM 1 BY 1 UNTIL LINEA1 > 24.
    MOVE LINEA TO LINEA1.
    PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
    MOVE LINEA1 TO LINEA.
    GO TO COMIENZO.

DERECHA.

    COMPUTE LINEA3 = LINEA1 + 1.
    MOVE CARACTER(LINEA2, LINEA1) TO CARACTER(LINEA2, LINEA3).

DERFIN.

    MOVE CARACTER(LINEA1, 81) TO CARACTER(LINEA1, 1).

SCROLLIZQUIERDA.

* ESTA RUTINA ES LA OPUESTA DE LA ANTERIOR, DESPLAZA LA PANTALLA
* UNA POSICION A LA IZQUIERDA.

    DISPLAY " ESPERA " LINE 12 POSITION 13 REVERSE BLINK.
    PERFORM IZQUIERDA VARYING LINEA1 FROM 1 BY 1
            UNTIL LINEA1 > 24.
    PERFORM IZQFIN VARYING LINEA1 FROM 1 BY 1
            UNTIL LINEA1 > 80 AFTER LINEA2 FROM 1
            BY 1 UNTIL LINEA2 > 24.
    MOVE LINEA TO LINEA1.
    PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
    MOVE LINEA1 TO LINEA.
    GO TO COMIENZO.

IZQUIERDA.
```

```
        MOVE CARACTER(LINEA1, 1) TO CARACTER(LINEA1, 81).

    IZQFIN.

        COMPUTE LINEA3 = LINEA1 + 1.
        MOVE CARACTER(LINEA2, LINEA3) TO CARACTER(LINEA2, LINEA1).

    BORRARLINEA.

    * ESTA RUTNA SE UTILIZA PARA BORRAR UNA LINEA COMPLETA, AL HACERLO
    * SUBE TODAS LAS LINEAS INFERIORES UNA LINEA.

        PERFORM BORRAR VARYING LINEA1 FROM LINEA BY 1
                UNTIL LINEA1 > 24.
        MOVE SPACES TO LINEAS(24).
        MOVE LINEA TO LINEA1.
        PERFORM IMP VARYING LINEA FROM LINEA1 BY 1
                UNTIL LINEA > 24.
        MOVE LINEA1 TO LINEA.
        GO TO COMIENZO.

    BORRAR.

        COMPUTE LINEA2 = LINEA1 + 1.
        MOVE LINEAS(LINEA2) TO LINEAS(LINEA1).

    INSERTARLINEA.

    * PARA INSERTAR UNA LINEA SE BAJAN TODAS LAS LINEAS A PARTIR DE LA
    * ACTUAL UNA LINEA, LA ULTIMA LINEA DE LA PANTALLA DESAPARECE.

        PERFORM INSERTA VARYING LINEA1 FROM 24 BY -1
                UNTIL LINEA1 < LINEA.
        MOVE SPACES TO LINEAS(LINEA).
        MOVE LINEA TO LINEA1.
        PERFORM IMP VARYING LINEA FROM LINEA1 BY 1 UNTIL LINEA > 24.
        MOVE LINEA1 TO LINEA.
        GO TO COMIENZO.

    INSERTA.

        COMPUTE LINEA2 = LINEA1 + 1.
        MOVE LINEAS(LINEA1) TO LINEAS(LINEA2).

    CARGAR.
```

```
* PARA CARGAR UNA PANTALLA PRIMERO PEDIMOS EL NOMBRE DE ESA PANTALLA.

     DISPLAY " DAME EL NOMBRE..:" LINE 12 POSITION 20 REVERSE.
     ACCEPT NOM-PAN LINE 12 POSITION 38 REVERSE NO BEEP.

* SI EL NOMBRE ES IGUAL A ESPACIOS SE VUELVE A EMPEZAR.

     IF NOM-PAN = SPACES GO TO GRABAR.

* SI EL NOMBRE ES VALIDO SE PONE EL CARTEL DE CARGANDO Y SE ABRE
* EL FICHERO SOLICITADO.

     DISPLAY " CARGANDO... " LINE 12 POSITION 25 REVERSE BLINK.
     OPEN INPUT PANTA.

 LEE.

* SE LEE EL REGISTRO DE LA PANTALLA.

     READ PANTA.

* SE LLAMA A LA RUTINA QUE PASARA EL REGISTRO A LA VARIABLE DE
* PANTALLA.

     PERFORM CARGA VARYING LINEA1 FROM 1 BY 1 UNTIL LINEA1 > 24.

* SE CIERRA EL FICHERO DE PANTALLA.

     CLOSE PANTA.

* SE REHACE LA PANTALLA.

     MOVE LINEA TO LINEA1.
     PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
     MOVE LINEA1 TO LINEA.

* SE VUELVE A LA ETIQUETA COMIENZO.

     GO TO COMIENZO.

 CARGA.

     MOVE LIN-PAN(LINEA1) TO LINEAS(LINEA1).

 GRABAR.

* PARA GRABAR UNA PANTALLA PRIMERO SE PIDE EL NOMBRE.
```

```
      DISPLAY " DAME EL NOMBRE..:" LINE 12 POSITION 20 REVERSE.
      ACCEPT NOM-PAN LINE 12 POSITION 38 REVERSE NO BEEP.

* SE COMPRUEBA QUE EL NOMBRE NO SEA IGUAL A ESPACIOS.

      IF NOM-PAN = SPACES GO TO GRABAR.

* SI ES VALIDO SE PONE EL CARTEL DE GRABANDO.

      DISPLAY " GRABANDO... " LINE 12 POSITION 25 REVERSE BLINK.

* SE ABRE EL FICHERO DE PANTALLA EN FORMA DE ESCRITURA.

      OPEN OUTPUT PANTA.

* SE LLAMA A LA RUTINA QUE PASARA EL REGISTRO DE LA PANTALLA A
* EL DE EL FICHERO.

      PERFORM GRABA VARYING LINEA1 FROM 1 BY 1 UNTIL LINEA1 > 24.

* SE ESCRIBE EL REGISTRO EN EL FICHERO.

      WRITE FIC-PAN.

* SE CIERRA EL FICHERO.

      CLOSE PANTA.

* SE REHACE LA PANTALLA Y SE VUELVE A LA ETIQUETA COMIENZO.

      MOVE LINEA TO LINEA1.
      PERFORM IMP VARYING LINEA FROM 12 BY 1 UNTIL LINEA > 12.
      MOVE LINEA1 TO LINEA.
      GO TO COMIENZO.

 GRABA.

     MOVE LINEAS(LINEA1) TO LIN-PAN(LINEA1).

 TECLASF.

* CUANDO SE PULSA UNA DE LAS TECLAS FUNCIONALES EN LINEA1 SE
* NOS MANDA EL NUMERO DE TECLA QUE SE HA PULSADO, CON ESTE
* MOVEMOS EL CONTENIDO DE LA TECLA FUNCIONAL A LA POSICION ACTUAL
* DEL CURSOR.

      MOVE GRAFICO(LINEA1) TO CARACTER(LINEA, COLUMNA), TECLA1.
```

```
* SE IMPRIME EL CARACTER.

     DISPLAY GRAFICO(LINEA1) LINE LINEA POSITION COLUMNA.

* SE SUMA UNO A LA COLUMNA Y SE CONTROLA QUE NO PASE DE 80, DESPUES
* SE VUELVE A LA ETIQUETA COMIENZO.

     ADD 1 TO COLUMNA.
     IF COLUMNA > 80 MOVE 80 TO COLUMNA.
     GO TO COMIENZO.

 BORRAPANTALLA.

* PARA BORRAR LA PANTALLA SE LLAMA A LA RUTINA BORRALI VARIANDO
* EL VALOR DE LINEA1 DESDE 1 HASTA 24.

     PERFORM BORRALI VARYING LINEA1 FROM 1 BY 1 UNTIL LINEA1 > 24.

* TRAS ELLO SE REHACE LA PANTALLA Y SE VUELVE A LA ETIQUETA COMIENZO.

     MOVE LINEA TO LINEA1.
     PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
     MOVE LINEA1 TO LINEA.
     GO TO COMIENZO.

 BORRALI.

* EN LA RUTINA SE MUEVE ESPACIOS A CADA UNA DE LAS LINEAS DE LA
* PANTALLA.

     MOVE SPACES TO LINEAS(LINEA1).

 ALMACENA.

* PARA ALMACENAR LA PANTALLA EN MEMORIA SE PASA LA VARIABLE QUE
* NOS SIRVE DE RED A LA VARIABLE QUE DECLARAMOS PARA ALMACENAR
* UNA PANTALLA.

     PERFORM ALMACENALI VARYING LINEA1 FROM 1 BY 1
             UNTIL LINEA1 > 24.
     GO TO COMIENZO.

 ALMACENALI.

     MOVE LINEAS(LINEA1) TO LINEAS1(LINEA1).

 REPETIR.
```

```
* PARA REPETIR UN CARACTER PRIMERO SE SOLICITA QUE SE PULSE EL
* CARACTER A REPETIR.

    DISPLAY " PULSE CARACTER " LINE 10 POSITION 20 REVERSE.
    ACCEPT CARAC NO BEEP LINE 10 POSITION 36 REVERSE.

* DESPUES SE PIDE EL NUMERO DE VECES QUE SE HA DE REPETIR EL
* CARACTER.

    DISPLAY " NUMERO DE VECES " LINE 12 POSITION 20 REVERSE.
    ACCEPT NUM LINE 12 POSITION 37 REVERSE NO BEEP.

* SE CALCULA EN LINEA2 LA POSICION ACTUAL MAS EL NUMERO DE VECES
* QUE SE HA DEE REPETIR EL CARACTER Y SE CONTROLA QUE NO PASE DE
* LA COLUMNA 80.

    COMPUTE LINEA2 = COLUMNA + NUM.
    IF LINEA2 > 80 MOVE 80 TO LINEA2.

* UNA VEZ HECHO EL CALCULO SE LLAMA A LA RUTINA REPITE VARIANDO
* LINEA1 DESDE LA POSICION ACTUAL HASTA LINEA2.

    PERFORM REPITE VARYING LINEA1 FROM COLUMNA BY 1
            UNTIL LINEA1 > LINEA2.

* TRAS ELLO SE REHACE LA PANTALLA Y SE VUELVE A LA ETIQUETA COMIENZO.

    MOVE LINEA TO LINEA1.
    PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
    MOVE LINEA1 TO LINEA.
    GO TO COMIENZO.

 REPITE.

* ESTA RUTINA MUEVE EL CARACTER PULSADO A LA POSICION DE PANTALLA
* INDICADA POR LINEA1.

    MOVE CARAC TO CARACTER(LINEA, LINEA1).

 BUSCAR-SUST.

* PARA LA OPCION DE BUSCA Y SUSTITUCION PRIMERO PEDIMOS EL
* CARACTER A BUSCAR.


    DISPLAY " PULSE CARACTER A BUSCAR " LINE 12
            POSITION 20 REVERSE.
    ACCEPT CARAC LINE 12 POSITION 45 REVERSE NO BEEP.
```

```
* DESPUES EL CARACTER POR EL QUE SE HA DE SUSTITUIR.

     DISPLAY " PULSE CARACTER POR EL QUE SUSTITUIR "
             LINE 14 POSITION 20 REVERSE.
     ACCEPT CARAC1 LINE 14 POSITION 57 REVERSE NO BEEP.

* A CONTINUACION SE PIDE A PARTIR DE QUE LINEA TENEMOS QUE
* EMPEZAR A SUSTITUIR.

     DISPLAY " A PARTIR DE QUE LINEA SUSTITUYO "
             LINE 16 POSITION 20 REVERSE.
     ACCEPT LINEA2 LINE 16 POSITION 53 REVERSE NO BEEP.

* POR ULTIMO PEDIMOS HASTA QUE LINEA SE HA DE SUSTITUIR.

     DISPLAY " HASTA QUE LINEA SUSTITUYO "
             LINE 18 POSITION 20 REVERSE.
     ACCEPT LINEA3 LINE 18 POSITION 47 REVERSE NO BEEP.

* CON LOS DATOS ANTERIORES SE LLAMA A LA RUTINA SUSTITUYE
* VARIANDO LINEA1 DESDE LINEA2 QUE ERA LA LINEA DE INICIO
* HASTA LINEA3 QUE ERA LA LINEA DE FIN.

     PERFORM SUSTITUYE VARYING LINEA1 FROM LINEA2 BY 1
             UNTIL LINEA1 > LINEA3.

* TRAS ELLO SE REHACE LA PANTALLA Y SE VUELVE A LA ETIQUETA COMIENZO.

     MOVE LINEA TO LINEA1.
     PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
     MOVE LINEA1 TO LINEA.
     GO TO COMIENZO.

 SUSTITUYE.

* EN LA RUTINA SUSTITUYE SE INSPECCIONA LA LINEA INDICADA POR LINEA1
* SUSTITUYENDO EL CARACETER CARAC POR CARAC1.

     INSPECT LINEAS(LINEA1) REPLACING ALL CARAC BY CARAC1.

 RECUPERAR.

* PARA RECUPERAR LA PANTALLA ALMACENADA EN MEMORIA SE PASA LA
* VARIABLE QUE DECLARAMOS A<PARA ALMACENARLA A LA VARIABLE QUE
* ES LA RED DE PANTALLA.
```

```
        PERFORM RECUPERALI VARYING LINEA1 FROM 1 BY 1
                UNTIL LINEA1 > 24.
        MOVE LINEA TO LINEA1.
        PERFORM IMP VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.
        MOVE LINEA1 TO LINEA, GO TO COMIENZO.

    RECUPERALI.

        MOVE LINEAS1(LINEA1) TO LINEAS(LINEA1).

    DELETECAR.

    * PARA BORRAR UN CARACTER SE MUEVEN TODOS LOS CARACTERES A
    * CONTINUACION DEL QUE SE DESEA BORRAR UNA POSICION A LA IZQUIERDA.

        MOVE SPACE TO CARACTER(LINEA, 81).
        PERFORM DELET VARYING LINEA1 FROM COLUMNA BY 1
                UNTIL LINEA1 > 80.

    * TRAS ELLO SE IMPRIME LA LINEA Y SE VUELVE A LA ETIQUETA COMIENZO.

        DISPLAY LINEAS(LINEA) LINE LINEA POSITION 1.
        GO TO COMIENZO.

    DELET.

    * ESTA RUTINA ES LLAMADA POR LA OPCION ANTERIOR PARA DESPLAZAR
    * UN LUGAR A LA IZQUIERDA LA LINEA.

        COMPUTE LINEA2 = LINEA1 + 1.
        MOVE CARACTER(LINEA, LINEA2) TO CARACTER(LINEA, LINEA1).

    INSERTAR.

    * PARA SABER SI EL MODO DE INSERCCION ESTA ACTIVADO O DESACTIVADO
    * TENEMOS A SW1, SI ESTE ES CERO LO PONEMOS A 1 CON LO CUAL HEMOS
    * CONECTADO EL MODO DE INSERCCION, SI ESTA A 1 LO PONEMOS A CERO
    * CON LO CUAL LO DESCONECTAMOS.

        IF SW1 = 0 MOVE 1 TO SW1, DISPLAY "INS"
           LINE 25 POSITION 55, GO TO COMIENZO ELSE MOVE 0 TO SW1,
           DISPLAY "   " LINE 25 POSITION 55, GO TO COMIENZO.

    CONTINUO.

    * LA CONEXION DESCONEXION DEL DIBUJO CONTINUO FUNCIONA DE FORMA
    * ANALOGA A LA DE INSERCCION PERO UTILIZANDO EL SW2.
```

```
        IF SW2 = 0 MOVE 1 TO SW2, DISPLAY "CONT"
           LINE 25 POSITION 58, GO TO COMIENZO ELSE MOVE 0 TO SW2,
           DISPLAY "     " LINE 25 POSITION 58, GO TO COMIENZO.


     CONTSINO.

    * A ESTA RUTINA SE LLAMA TRAS HABER PULSADO UNA DE LAS TECLAS DE
    * MOVIMIENTO DE CURSOR, LO QUE HACE ES COMPROBAR SI ESTA ACTIAVDO
    * EL DIBUJO CONTINUO, DE SER ASI MUEVE EL CARACETER ALMACENADO
    * EN TECLA1 A LA RED Y LO IMPRIME POR LA PANTALLA, TRAS ELLO VUELVE
    * A LA ETIQUETA COMIENZO.

        COMPUTE LINEA1 = COLUMNA - 1.
        IF LINEA1 < 1 MOVE 1 TO LINEA1.

        IF SW2 = 1 MOVE TECLA1 TO CARACTER(LINEA, LINEA1),
           DISPLAY TECLA1 LINE LINEA POSITION LINEA1.
        GO TO COMIENZO.

     SALIR.

    * SI SE ELIJE LA OPCION SALIR DEL EDITOR LO QUE SE HACE ES BORRAR
    * LA PANTALLA Y PARAR EL PROGRAMA.


        DISPLAY " " ERASE.
        STOP RUN.
```

Cuando lo compiles al final de dará un 'WARNING' o aviso, esto es debido a que hemos puesto una instrucción GO TO en las declaratives que hace referencia a una etiqueta que está fuera de ellas, esto en teoría no se puede hacer pero funciona correctamente y como cabría esperar. Cuando lo ejecutes podrás editar una pantalla y almacenarla en el disco, cuando la quieras recuperar en cualquier programa sólo tienes que declarar un fichero de pantalla con su nombre y leerla.

13.3. ALMACENANDO PANTALLAS.

Cuando se está haciendo una aplicación lo normal es que se tenga un número considerable de máscaras o pantallas, estas no ocupan mucho espacio en el disco pero cada una de ellas ocupa una entrada de directorio, un disco tiene un número de entradas de directorio fijas, ¿como podríamos meter todas las pantallas en una sola entrada de directorio?, la solución sería meter todas las

pantallas en un fichero como si cada pantalla fuera un registro y a la hora de sacarlas leerlas por medio de su nombre, para poder referirnos por el nombre tendría que ser un fichero indexado, en él tendríamos como clave el nombre y como registro la pantalla. Esta sería una posible solución, vamos a hacer un programa que nos coja pantallas y nos las meta en un fichero indexado con el nombre que nosotros especifiquemos, además de esto vamos a darles las siguientes opciones: que nos dé por pantalla en un momento dado los nombres de las pantallas que en ese momento tenemos en el fichero, que nos permita borrar una pantalla, que nos permita renombrar una pantalla, que nos permita sacar esa pantalla por impresora y por último que nos permita sacar esa pantalla a un fichero normal de una sola pantalla.

Con lo visto anteriormente vamos a poner manos a la obra, abre en tu editor un archivo con el nombre de 'PRUEBA58.CBL' y teclea lo que tienes a continuación, como siempre en el programa van incluidos todos los comentarios que se han creido convenientes.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA58.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

      * SE DECLARA EL FICHERO DE PANTALLAS COMO INDEXADO Y ASIGNA A
      * A LA VARIABLE NOM-FIC PARA PODER MANEJAR VARIOS FICHEROS DE
      * DE PANTALLA.

           SELECT FICHERO;
                  ASSIGN TO RANDOM, NOM-FIC;
                  ORGANIZATION IS INDEXED;
                  ACCESS MODE IS DYNAMIC;
                  RECORD KEY IS NOM-PAN-FIC;
                  FILE STATUS IS ERRORES.

      * SE ASIGNA UN FICHERO DE PANTALLA PARA PODER LEER LAS PANTALLAS
      * QUE SE HAN DE METER EN EL FICHERO INDEXADO.
```

```
       SELECT PANTALLA;
              ASSIGN TO RANDOM, NOM-PAN;
              ORGANIZATION IS RELATIVE;
              ACCESS MODE IS DYNAMIC;
              RELATIVE KEY IS CLV-PAN;
              FILE STATUS IS ERRORES.

 * SE ASIGNA UN FICHERO DE IMPRESORA PARA LA OPCION DE LISTADO
 * DE UNA PANTALLA.

       SELECT IMP;
              ASSIGN TO PRINT, "PRINTER";
              FILE STATUS IS ERRORES.

  DATA DIVISION.
  FILE SECTION.

 * SE DECLARA EL REGISTRO DEL FICHERO DE IMPRESORA, TENDRA
 * 15 CARACTERES PARA EL NOMBRE QUE HARA LAS VECES DE CLAVER
 * Y UNA TABLA DE 24 ELEMENTOS DE 80 CARACTERES PARA EL ALMACENAMIENTO
 * DE LA PANTALLA.

  FD  FICHERO;
      LABEL RECORD IS STANDARD;
      DATA RECORD IS FIC-FIC.

  01  FIC-FIC.
      03  NOM-PAN-FIC                  PIC X(15).
      03  DAT-FIC.
          05  LIN-FIC                      PIC X(80) OCCURS 24.

 * SE DECLARA EL FICHERO DE PANTALLA COMO UNA TABLA DE 24
 * ELEMENTOS CON 80 CARACTERES PARA LEER UNA PANTALLA CREADA
 * CON EL EDITOR QUE TENEMOS HECHO.

  FD  PANTALLA;
      LABEL RECORD IS STANDARD;
      DATA RECORD IS FIC-PAN.

  01  FIC-PAN.
      03  DAT-PAN                      PIC X(80) OCCURS 24 TIMES.

 * DECLARAMOS EL REGISTRO DE IMPRESORA COMO UNA LINEA DE 80
 * CARACTERES PARA IMPRIMIR CUALQUIER PANTALLA.

  FD  IMP;
      LABEL RECORD IS OMITTED;
      DATA RECORD IS LIN-IMP.
```

```
       01  LIN-IMP                          PIC X(80).

       WORKING-STORAGE SECTION.

      * ESTA VARIABLE NOS SERVIRA PARA DETECTAR LOS POSIBLES
      * ERRORES EN FICHEROS.

       77  ERRORES                          PIC X(02).

      * ESTA VARIABLE SERA LA CLAVE DE LECTURA EN EL FICHERO DE
      * PANTALLAS, LE DAMOS EL VALOR 1 Y NO LO MODIFICAMOS A LO
      * LARGO DEL PROGRAMA AL IGUAL QUE EN EL EDITOR.

       77  CLV-PAN                          PIC 9(01) VALUE 1.

      * CON ESTA VARIABLE PODREMOS LEER LA PANTALLA QUE DESEEMOS

       77  NOM-PAN                          PIC X(12).

      * ESTA VARIABLE SERA EL NOMBRE DEL FICHERO DE PANTALLAS EN
      * EL QUE SE VA A TRABAJAR.

       77  NOM-FIC                          PIC X(12).

      * ESTAS VARIABLES NOS SERVIRAN PARA PEDIR OPCION, RESPUESTAS
      * Y LLEVAR EL CONTROL DE ALGUNAS IMPRESIONES.

       77  OPCION                           PIC 9(01).
       77  SW1                              PIC 9(01).
       77  SINO                             PIC X(01).
       77  LINEA                            PIC 9(02).
       77  COLUMNA                          PIC 9(02).

       PROCEDURE DIVISION.
       DECLARATIVES.

      * ESTA SECCION DE LAS DECLARATIVES SE VA A USAR PARA CONTROLAR
      * LOS POSIBLES ERRORES EN EL FICHERO DE PANTALLAS.

       I-O-FICHERO SECTION.
           USE AFTER STANDARD ERROR PROCEDURE ON FICHERO.

       I-O-CREACION.

      * EL ERROR QUE SE PUEDE PRODUCIR ES QUE AL ABRIR EL FICHERO
      * ESTE NO EXISTE, ENTONCES SE ABRE EN MODO OUTPUT PARA CREARLO
      * SE CIERRA Y SE VUELVE A ABRIR EN MODO DIRECTO.
```

```
        OPEN OUTPUT FICHERO, CLOSE FICHERO, OPEN I-O FICHERO.

      * ESTA SECCION SE UTILIZA PARA PREVENIR LOS ERRORES EN LA
      * PANTALLA QUE SE DESEA TRATAR.


       I-O-PANTALLA SECTION.
           USE AFTER STANDARD ERROR PROCEDURE ON PANTALLA.

       I-O-MENSAJE.

      * EL ERROR QUE SE PUEDE PRODUCIR ES QUE SE DE UN NOMBRE DE
      * PANTALLA QUE NO EXISTE, EN ESE CASO SE AVISA Y SE MUEVE UN 1
      * A SW1, ESTO SE HACE PARA QUE CUANDO EL PROGRAMA VUELVA A SU
      * FLUJO NORMAL DETECTE QUE SE PRODUJO UN ERROR Y VUELVA A PEDIR
      * EL NOMBRE DE PANTALLA.

           DISPLAY " PANTALLA INEXISTENTE " LINE 12 POSITION 20
                   REVERSE BLINK BEEP.
           MOVE 1 TO SW1.

       END DECLARATIVES.
       PROGRAMA SECTION.
       COMIENZO.

      * AL EJECUTAR EL PROGRAMA LO PRIMERO QUE SE HACE ES PEDIR EL
      * NOMBRE DEL FICHERO DE PANTALLA DONDE VAN A RECAHER TODAS LAS
      * OPERACIONES A LO LARGO DEL PROGRAMA.

           DISPLAY " " ERASE.
           DISPLAY "NOMBRE DEL FICHERO DE TRABAJO..:"
                   LINE 2 POSITION 10 REVERSE.

      * SE PIDE EL NOMBRE Y SE ABRE EL FICHERO EN MODO DIRECTO, SI NO
      * EXISTE SE IRA A LAS DECLARATIVES Y LO CREARA.

           ACCEPT NOM-FIC LINE 2 POSITION 42 NO BEEP.
           OPEN I-O FICHERO.

       MENU.

      * SE IMPRIME EL MENU DE OPCIONES.


           DISPLAY "1.- ALTA DE UNA PANTALLA" LINE  8 POSITION 20 ERASE.
           DISPLAY "2.- BAJA DE UNA PANTALLA" LINE 10 POSITION 20.
           DISPLAY "3.- RENOMBRAR UNA PANTALLA" LINE 12 POSITION 20.
```

```
        DISPLAY "4.- CATALOGO DE PANTALLAS" LINE 14 POSITION 20.
        DISPLAY "5.- LISTADO DE UNA PANTALLA" LINE 16 POSITION 20.
        DISPLAY "6.- PASAR UNA PANTALLA A UN FICHERO" LINE 18
              POSITION 20.
        DISPLAY "ELIJE.[ ] 0-SALIR" LINE 22 POSITION 10.

     PIDEOPCION.

    * SE PIDE OPCION.

        ACCEPT OPCION LINE 22 POSITION 17 NO BEEP.

    * SI COMO OPCION SE A DADO 0 SE VA A SALIR.

        IF OPCION = 0 GO TO SALIR.

    * SI NO HA SIFO CERO SE VA A UNA DE LAS ETIQUETAS DEPENDIENDO
    * DE LA OPCION PULSADA.

        GO TO ALTAS, BAJAS, RENOMBRAR, CATALOGO, LISTADO, RECUPERAR
             DEPENDING ON OPCION.

    * SI SE LLEGA AQUI ES QUE LA OPCION ES INVALIDA.

        GO TO PIDEOPCION.

    * AQUI SE LLEGARA CUANDO LA OPCION PULSADA HALLA SIDO ALTAS.

     ALTAS.

    * SE IMPRIME EN LA LINEA 1 UNA CABECERA CON EL MENSAJE ALTAS
    * Y EL NOMBRE DEL FICHERO EN QUE SE ESTA TRABAJANDO.

        DISPLAY SPACES SIZE 80 LINE 1 POSITION 1 ERASE REVERSE.
        DISPLAY "ALTAS" LINE 1 POSITION 10 REVERSE.
        DISPLAY "FICHERO.:" LINE 1 POSITION 40 REVERSE.
        DISPLAY NOM-FIC LINE 1 POSITION 49 REVERSE.

     PIDEPANTALLA.

    * PIDE EL NOMBRE DE LA PANTALLA QUE SE DESEA DAR DE ALTA.

        DISPLAY "PANTALLA..:" LINE 6 POSITION 25.
        ACCEPT NOM-PAN LINE 6 POSITION 36 NO BEEP.

    * SI EL NOMBRE ES IGUAL A ESPACIOS VUELVE A MENU.
```

```
           IF NOM-PAN = SPACES GO TO MENU.

       * PIDE EL NOMBRE QUE SE QUIERE TENGA LA PANTALLA

           DISPLAY "NOMBRE....:" LINE 8 POSITION 25.
           ACCEPT NOM-PAN-FIC LINE 8 POSITION 36 NO BEEP.

       * ABRE LA PANTALLA INDICADA EN MODO DIRECTO, SI NO EXISTE DARA
       * ERROR Y SE IRA A LAS DECLARATIVES DONDE NOS DIRA QUE LA
       * PANTALLA NO EXISTE.

           OPEN I-O PANTALLA.

       * SI SE HA PRODUCIDO UN ERROR A LA VUELTA DE LAS DECLARATIVES
       * SW1 TENDRA UN 1, EN ESE CASO SE LE MUEVE UN CERO Y SE VUELVE
       * A PEDIR EL NOMBRE DE PANTALLA, EN CASO CONTRARIO SE SIGUE.

           IF SW1 = 1 MOVE 0 TO SW1, GO TO PIDEPANTALLA.

       * SE LEE EL REGISTRO DEL FICHERO ESPECIFICADO.

           READ PANTALLA.

       * SE IMPRIME EN PANTALLA.

           DISPLAY FIC-PAN LINE 2 POSITION 1.

       * TRAS IMPRIMIR SE VA A INTENTAR GRABAR ESA PANTALLA EN EL
       * FICHERO INDEXADO.

           GO TO NO-ESTA.

        EXISTE.

       * SI AL GRABAR LA PANTALLA DA ERROR QUIERE DECIR QUE ESA PANTALLA
       * YA EXISTIA, EN ESE CASO SE AVISA DE QUE EXISTE Y SE DA OPCION A
       * SOBREGRABAR O A NO GRABAR.

           DISPLAY "ESTA PANTALLA EXISTE, SOBREGRABO?"
                   LINE 20 POSITION 10 REVERSE.

        PIDESINO1.

           ACCEPT SINO LINE 20 POSITION 43 REVERSE NO BEEP.

       * SI NO QUEREMOS GRABAR SE VA A CIERRA1.

           IF SINO = "N" OR SINO = "n" GO TO CIERRA1.
```

```
* SI LA VARIABLE SINO NO ES IGUAL A N NI A S ES QUE LA OPCION
* SE HA TECLEADO MAL, SE VUELVE A PEDIR.

    IF SINO NOT = "S" AND SINO NOT = "n" GO TO PIDESINO1.

* SI SE LLEGA A ESTE PUNTO ES QUE LA OPCION TECLEADA A SIDO S,
* EN ESE CASO SE REESCRIBE LA PANTALLA.

    REWRITE FIC-FIC.

 CIERRA1.

* AL TERMINAR SE CIERRA LA PANTALLA Y SE VUELVE A ALTAS A PEDIR
* OTRA PANTALLA.

    CLOSE PANTALLA.
    GO TO ALTAS.

 NO-ESTA.

* TRAS IMPRIMIR LA PANTALLA SE VIENE AQUI DONDE NOS PREGUNTA SI
* ESTA CORRECTO.

    DISPLAY "CORRECTO?" LINE 20 POSITION 10 REVERSE.

 PIDESINO2.

    ACCEPT SINO LINE 20 POSITION 19 REVERSE NO BEEP.

* SI DECIMOS QUE NO SE VA A CIERRA2.

    IF SINO = "N" OR SINO = "n" GO TO CIERRA2.

* SI NO ES IGUAL A N PREGUNTAMOS SI ES DESIGUAL A S, EN ESE CASO
* LA OPCION SE TECLEADO MAL, SE VUELVE A PEDIR.

    IF SINO NOT = "s" AND SINO NOT = "S" GO TO PIDESINO2.

* EN CASO DE QUE LA RESPUESTA SEA S, SE MUEVE LA PANTALLA
* AL REGISTRO DEL FICHERO INDEXADO.

    MOVE FIC PAN TO DAT-FIC.

* SE ESCRIBE EL REGISTRO, SI ES CLAVE INVALIDA ES QUE YA EXISTE,
* EN ESE CASO SE VA A LA ETIQUETA EXISTE.

    WRITE FIC FIC INVALID KEY GO TO EXISTE.
```

```
 CIERRA2.

* DESPUES DE GRABAR SE CIERRA EL FICHERO DE PANTALLA Y SE VUELVE
* A LA ETIQUETA ALTAS.

     CLOSE PANTALLA.
     GO TO ALTAS.


* AQUI SE LLEGARA CUANDO LA OPCION ELEGIDA SEA LA DE DAR DE BAJA.

 BAJAS.

* SE IMPRIME UN MENSAJE CON BAJAS Y EL FICHERO DE TRABAJO.

     DISPLAY SPACES SIZE 80 LINE 1 POSITION 1 ERASE REVERSE.
     DISPLAY "BAJAS" LINE 1 POSITION 10 REVERSE.
     DISPLAY "FICHERO.:" LINE 1 POSITION 40 REVERSE.
     DISPLAY NOM-FIC LINE 1 POSITION 49 REVERSE.

 PIDENOMBREBAJA.

* SE PIDE EL NOMBRE DE LA PANTALLA QUE SE QUIERE DAR DE BAJA.

     DISPLAY "NOMBRE...:" LINE 6 POSITION 20.
     ACCEPT NOM-PAN-FIC LINE 6 POSITION 31 NO BEEP.

* SI EL NOMBRE ES IGUAL A ESPACIOS SE VUELVE A LA ETIQUETA MENU.

     IF NOM-PAN-FIC = SPACES GO TO MENU.

* EN CASO CONTRARIO LEE LA PANTALLA, SI DA CLAVE INVALIDA, SI LA
* PANTALLA NO EXISTE NO SE PUEDE DAR DE BAJA, SE VA A LA ETIQUETA
* NOEXISTEBAJA PARA COMUNICARLO.

     READ FICHERO INVALID KEY GO TO NOEXISTEBAJA.

* SI NO HA HABIDO PROBLEMA AL LEERLO SE VA A LA ETIQUETA DARDEBAJA.

     GO TO DARDEBAJA.

 NOEXISTEBAJA.

* SI LA PANTALLA ESPECIFICADA NO EXISTE SE AVISA Y SE VUELVE A
* PEDIR EL NOMBRE DE LA PANTALLA.


     DISPLAY " ESA PANTALLA NO EXISTE " LINE 18 POSITION 25
```

```
                REVERSE BLINK BEEP.
        GO TO PIDENOMBREBAJA.

 DARDEBAJA.

* ANTES DE DAR DE BAJA SE IMPRIME LA PANTALLA.

     DISPLAY DAT-FIC LINE 2 POSITION 1.

 DOYDEBAJASN.

* SE PREGUNTA SI ESTA CONFORME CON DARLO DE BAJA'

     DISPLAY "CORRECTO?" LINE 18 POSITION 10 REVERSE.
     ACCEPT SINO LINE 18 POSITION 19 REVERSE NO BEEP.

* SI LA RESPUESTA ES NO NO SE DA DE BAJA Y SE VUELVE A LA
* ETIQUETA BAJAS.

     IF SINO = "N" OR SINO = "n" GO TO BAJAS.

* SI LA RESPUESTA NO ES IGUAL A S NI N QUIERE DECIR QUE ES
* INVALIDA, POR LO TANTO SE VUELVE A PEDIR.

     IF SINO NOT = "S" AND SINO NOT = "s" GO TO DOYDEBAJASN.

* SI SE LLEGA AQUI QUIERE DECIR QUE SE HA TECLEADO S. SE BORRA
* LA PANTALLA Y SE VUELVE A LA ETIQUETA BAJAS.

     DELETE FICHERO.
     GO TO BAJAS.

* A ESTA ETIQUETA SE LLEGARA CUANDO LA OPCION TECLEADA HALLA
* SIDO LA DE RENOMBRAR UNA PANTALLA.


 RENOMBRAR.

* SE IMPRIME UN MENSAJE CON RENOMBRAR EL NOMBRE DEL ARCHIVO
* EN EL QUE SE ESTA TRABAJANDO.

     DISPLAY SPACES SIZE 80 LINE 1 POSITION 1 ERASE REVERSE.
     DISPLAY "RENOMBRAR" LINE 1 POSITION 10 REVERSE.
     DISPLAY "FICHERO.:" LINE 1 POSITION 40 REVERSE.
     DISPLAY NOM-FIC LINE 1 POSITION 49 REVERSE.

 PIDENOMBREACTUAL.
```

```
* PIDE EL NOMBRE QUE TIENE ACTUALMENTE LA PANTALLA EN EL FICHERO.

     DISPLAY "NOMBRE AGTUAL..:" LINE 6 POSITION 10.
     ACCEPT NOM-PAN-FIC LINE 6 POSITION 26 NO BEEP.

* SI EL NOMBRE ES IGUAL A ESPACIOS SE VUELVE AL MENU.

     IF NOM-PAN-FIC = SPACES GO TO MENU.

* EN CASO CONTRARIO SE HACE UN START PARA BUSCAR LA PANTALLA,
* SI DA CLAVE INVALIDA ES QUE LA PANTALLA NO EXISTE, POR LO
* TANTO NO SE PUEDE RENOMBRAR, SE VA A LA ETIQUETA NOEXISTE1.

     START FICHERO KEY IS = NOM-PAN-FIC INVALID KEY
           GO TO NOEXISTE1.

* SI NO HAY PROBLEMA SE VA A LA ETIQUETA PIDENOMBRENUEVO.

     GO TO PIDENOMBRENUEVO.

 NOEXISTE1.

* SI LA PANTALLA NO EXISTE SE IMPRIME UN MENSAJE AVISANDOLO Y
* SE VUELVE A PEDIR EL NOMBRE DE LA PANTALLA.

     DISPLAY " PANTALLA INEXISTENTE " LINE 18 POSITION 25
             REVERSE BLINK BEEP.
     GO TO PIDENOMBREACTUAL.

 PIDENOMBRENUEVO.

* SI LA PANTALLA EXISTE SE LEE Y SE PASA EL NOMBRE A NOM-PAN.

     READ FICHERO.
     MOVE NOM-PAN-FIC TO NOM-PAN.

* SE PIDE EL NOMBRE QUE SE LE DESEA PONER.

     DISPLAY "NOMBRE NUEVO...:" LINE 8 POSITION 10.
     ACCEPT NOM-PAN-FIC LINE 8 POSITION 26 NO BEEP.

* SE BUSCA SI EL NOMBRE QUE SE LE DESEA PONER EXISTE, SI DA
* CLAVE INVALIDA ES QUE NO EXISTE, EN ESE CASO SE VA A LA
* ETIQUETA RENOMBRA..
```

```
        START FICHERO KEY IS = NOM-PAN-FIC INVALID KEY
              GO TO RENOMBRA.

    * SI NO HA DADO ERROR QUIERE DECIR QUE LA PANTALLA EXISTE, POR
    * LO TANTO NO PODEMOS VOLVER A OCUPARLA, SE AVISA Y SE VUELVE
    * A PEDIR EL NOMBRE QUE SE LE QUIERE PONER.

        DISPLAY " ESA PANTALLA YA EXISTE " LINE 18 POSITION 25
                REVERSE BLINK BEEP.
        GO TO PIDENOMBRENUEVO.

     RENOMBRA.

    * SI EL NOMBRE NO ESTA OCUPADO SE ESCRIBE CON EL NUEVO NOMBRE.

        WRITE FIC-FIC.

    * MUEVE EL NOMBRE DE NOM-PAN QUE ERA EL ANTERIOR A NOM-PAN-FIC
    * PARA BORRARLO Y DESPUES VOLVER A LA ETIQUETA RENOMBRAR.

        MOVE NOM-PAN TO NOM-PAN-FIC.
        DELETE FICHERO.
        GO TO RENOMBRAR.

    * AQUI SE LLEGARA CUANDO LA OPCION ELEGIDA SEA LA DE CATALOGO.


     CATALOGO.

    * SE IMPRIME UN MENSAJE CON CATALOGO Y EL NOMBRE DEL FICHERO EN
    * QUE SE ESTA TRABAJANDO.

        DISPLAY SPACES SIZE 80 LINE 1 POSITION 1 ERASE REVERSE.
        DISPLAY "CATALOGO" LINE 1 POSITION 10 REVERSE.
        DISPLAY "FICHERO.:" LINE 1 POSITION 40 REVERSE.
        DISPLAY NOM-FIC LINE 1 POSITION 49 REVERSE.

    * MUEVE 1 A LINEA Y COLUMNA, ESTAS DOS VARIABLES NOS VAN A SERVIR
    * PARA IR IMPRIMIENDO LOS NOMBRES DE LAS PANTALLAS.

        MOVE 1 TO LINEA, COLUMNA.

    * MUEVE ESPACIOS AL NOMBRE DE LA PANTALLA Y HACE UN START, CON
    * LO CUAL EL PUNTERO DE LECTURA SE SITUA AL PRINCIPIO DEL FICHERO.

    * SI DA CLAVE INVALIDA QUIERE DECIR QUE EL FICHERO ESTA VACIO, EN
    * ESE CASO SE VA A LA ETIQUETA NOHAY.
```

```
      MOVE SPACES TO NOM-PAN-FIC.
      START FICHERO KEY IS NOT < NOM-PAN-FIC INVALID KEY
            GO TO NOHAY.

 LEEIMPRIME.

* SI HAY PANTALLAS SE LE SUMA UNO A LINEA

      ADD 1 TO LINEA.

* SI LA LINEA ES MAYOR DE 20 SE LE SUMA 15 A LA COLUMNA Y SE
* MUEVE 2 A LA LINEA, CON LO CUAL SE EMPIEZA A IMPRIMIR EN LA
* SIGUIENTE COLUMNA DE PANTALLA.

      IF LINEA > 20 ADD 15 TO COLUMNA, MOVE 2 TO LINEA.

* SI LA COLUMNA ES MAYOR DE 75 QUIERE DECIR QUE SE HA OCUPADO
* TODA LA PANTALLA, EN ESE CASO SE VA A LA ETIQUETA ESPERA.

      IF COLUMNA > 75 GO TO ESPERA.

* SE LEE EN FORMA SECUENCIAL DEL FICHERO, CUANDO SE LLEGUE
* AL FINAL DEL FICHERO SE VA A LA ETIQUETA FINCATALOGO.

      READ FICHERO NEXT RECORD AT END GO TO FINCATALOGO.

* IMPRIME EL NOMBRE DE LA PANTALLA EN LA LINEA Y COLUMNA QUE
* VIENE DADA POR LAS VARIABLES DEL MISMO NOMBRE.

      DISPLAY NOM-PAN-FIC LINE LINEA POSITION COLUMNA.

* TRAS IMPRIMIR  VUELVE A SEGUIR LEYENDO.

      GO TO LEEIMPRIME.

 FINCATALOGO.

* CUANDO TERMINE DE LEER TODOS LOS TITULOS NOS AVISA DE QUE PULSEMOS
* UNA TECLA Y DESPUES VUELVE A LA ETIQUETA MENU.

      DISPLAY " PULSA UNA TECLA " LINE 24 POSITION 30 REVERSE.
      ACCEPT OPCION OFF NO BEEP.
      GO TO MENU.

 NOHAY.

* SI EL FICHERO ESTA VACIO SE AVISA Y SE ESPERA A QUE SE PULSE
* UNA TECLA, TRAS ELLO SE VUELVE A LA ETIQUETA MENU.
```

```
       DISPLAY " EL FICHERO ESTA VACIO " LINE 24 POSITION 30
            REVERSE.
       ACCEPT OPCION OFF NO BEEP.
       GO TO MENU.

    ESPERA.

  * A ESTA ETIQUETA LLEGARA CUANDO LA PANTALLA SE HALLA LENADO
  * ESPERA  A QUE PULSEMOS UNA TECLA TRAS LO CUAL MUEVE UN 1 A
  * LA LINEA Y COLUMNA Y VUELVE A SEGUIR LEYENDO.


       DISPLAY " PULSA UNA TECLA " LINE 24 POSITION 30 REVERSE.
       ACCEPT OPCION OFF NO BEEP.
       MOVE 1 TO LINEA, COLUMNA.
       GO TO LEEIMPRIME.

  * AQUI SE LLEGARA CUANDO LA OPCION TECLEADA HALLA SIDO LA
  * DE LISTADO.


    LISTADO.

  * IMPRIME UN MENSAJE CON LISTADO Y EL NOMBRE DEL FICHERO DE TRABAJO.

       DISPLAY SPACES SIZE 80 LINE 1 POSITION 1 ERASE REVERSE.
       DISPLAY "LISTADO" LINE 1 POSITION 10 REVERSE.
       DISPLAY "FICHERO.:" LINE 1 POSITION 40 REVERSE.
       DISPLAY NOM-FIC LINE 1 POSITION 49 REVERSE.

    PIDENOMBRELISTAR.

  * PIDE EL NOMBRE DE LA PANTALLA A LISTAR.

       DISPLAY "NOMBRE...:" LINE 6 POSITION 20.
       ACCEPT NOM-PAN-FIC LINE 6 POSITION 30 NO BEEP.

  * SI EL NOMBRE ES IGUAL A ESPACIOS SE VA A LA ETIQUETA MENU.

       IF NOM-PAN-FIC = SPACES GO TO MENU.

  * EN CASO CONTRARIO LEE LA PANTALLA INDICADA, SI DA CLAVE INVALIDA
  * ES QUE NO EXISTE, EN ESE CASO SE VA A LA ETIQUETA NOEXISTELISTAR.

       READ FICHERO INVALID KEY GO TO NOEXISTELISTAR.

  * SI EXISTE NOS AVISA DE QUE CONECTEMOS LA IMPRESORA Y ESPERA
  * LA PULSACION DE UNA TECLA.
```

```
        DISPLAY " CONECTA LA IMPRESORA " LINE 15 POSITION 30
                REVERSE BLINK.
        ACCEPT OPCION OFF.

    * ABRE EL FICHERO DE IMPRESORA.

        OPEN OUTPUT IMP.

    * LLAMA A LA RUTINA IMPRIME VARIANDO LINEA DESDE 1 HASTA 24
    * PARA IMPRIMIR LAS 24 LINEAS DE QUE CONSTA LA PANTALLA.

        PERFORM IMPRIME VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 24.

    * TRAS ELLO CIERRA EL FICHERO DE IMPRESORA CON LA OPCION
    * WITH NO REWIND PARA QUE NO SALTE PAGINA.

        CLOSE IMP WITH NO REWIND.

    * Y VUELVE A LA ETIQUETA LISTADO.

        GO TO LISTADO.

     NOEXISTELISTAR.

    * SI LA PANTALLA NO EXISTE SE AVISA Y SE VUELVE A PEDIR EL NOMBRE.

        DISPLAY " PANTALLA INEXISTENTE " LINE 18 POSITION 30
                REVERSE BLINK BEEP.
        GO TO PIDENOMBRELISTAR.

     IMPRIME.

    * ESTA RUTINA COGE UN ELEMENTO DE LA TABLA QUE CONSTITUYE EL FICHERO
    * LO PASA A LA LINEA DE IMPRESORA Y LO IMPRIME.


        MOVE LIN-FIC(LINEA) TO LIN-IMP.
        WRITE LIN-IMP.

    * A ESTE PUNTO SE LLEGARA CUANDO SE HALLA ELEGIDO LA OPCION DE
    * PASAR UNA PANTALLA A UN FICHERO.

     RECUPERAR.

    * SE IMPRIME UN MENSAJE CON RECUPERAR Y EL NOMBRE DEL FICHERO
    * EN EL QUE SE ESTA TRABAJANDO.
```

```
        DISPLAY SPACES SIZE 80 LINE 1 POSITION 1 ERASE REVERSE.
        DISPLAY "RECUPERAR PANTALLA" LINE 1 POSITION 10 REVERSE.
        DISPLAY "FICHERO.:" LINE 1 POSITION 40 REVERSE.
        DISPLAY NOM-FIC LINE 1 POSITION 49 REVERSE.

    PIDENOMBRERECUPERAR.

    * SE PIDE EL NOMBRE DE LA PANTALLA A RECUPERAR.

        DISPLAY "NOMBRE DE LA PANTALLA..:" LINE 6 POSITION 10.
        ACCEPT NOM-PAN-FIC LINE 6 POSITION 34 NO BEEP.

    * SI EL NOMBRE ES IGUAL A ESPACIOS SE VA A LA ETIQUETA MENU.

        IF NOM-PAN-FIC = SPACES GO TO MENU.

    * EN CASO CONTRARIO LEE LA PANTALLA, SI DA CLAVE INVALIDA ES QUE
    * LA PANTALLA NO EXISTE, EN ESE CASO SE VA A LA ETIQUETA
    * NOEXISTERECUPERAR.

        READ FICHERO INVALID KEY GO TO NOEXISTERECUPERAR.

    * SI LA PANTALLA EXISTE SE PREGUNTA EL NOMBRE DEL FICHERO EN EL
    * QUE SE DESEA PONER LA PANTALLA.

        DISPLAY "NOMBRE DEL FICHERO.....:" LINE 8 POSITION 10.
        ACCEPT NOM-PAN LINE 8 POSITION 34 NO BEEP.

    * CREA EL FICHERO EN EL CUAL SE VA A PONER LA PANTALLA

        OPEN OUTPUT PANTALLA, CLOSE PANTALLA, OPEN I-O PANTALLA.

    * MUEVE EL REGISTRO DEL FICHERO INDEXADO AL REGISTRO DE PANTALLA..

        MOVE DAT-FIC TO FIC-PAN.

    * GRABA LA PANTALLA, CIERRA EL FICHERO Y VUELVE A RECUPERAR.

        WRITE FIC-PAN.
        CLOSE PANTALLA.
        GO TO RECUPERAR.

    NOEXISTERECUPERAR.

    * SI LA PANTALLA NO EXISTE SE AVISA Y SE VUELVE A PEDIR NOMBRE.

        DISPLAY " PANTALLA INEXISTENTE " LINE 15 POSITION 30
```

```
                REVERSE BLINK BEEP.
          GO TO PIDENOMBRERECUPERAR.

      * AQUI SE LLEGARA CUANDO AL PEDIR OPCIONN SE PULSE 0.

       SALIR.

      * SE CIERRA EL FICHERO DE PANTALLAS, SE BORRA LA PANTALLA Y SE
      * DETIENE EL PROGRAMA.

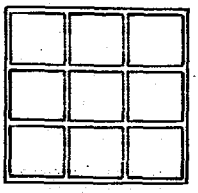
              CLOSE FICHERO.
              DISPLAY " " ERASE.
              STOP RUN.
```

A la hora de utilizar esta utilidad tendrías que meter primero todas las pantallas de la aplicación en el fichero y a todos los programas incluirles una rutina donde se abra el fichero y se lea la pantalla correspondiente.

13.4. LAS TRES EN RAYA.

Como dijimos al principio de este libro el lenguaje Cobol está enfocado a los procesos de gestión de ficheros y de grandes masas de datos, los juegos en este lenguaje no tendrían ningún fondo, entre otras cosas porque no tiene posibilidades de gráficos, de mover figuras ni nada por el estilo, sin embargo para un juego de inteligencia no nos hace falta nada de esto. Según esto vamos a intentar hacer un juego bastante conocido que es el tres en raya, se ha escogido este por su facilidad ya que un juego más complicado como damas o ajedrez exigiría un extensísimo programa, este que a continuación se expone no pretende ser más que una demostración de lo que se puede hacer además de programas de gestión. El programa es susceptible de mejoras y está abierto a modificaciones, particularmente puedo decir que tal y como está ya es bastante difícil ganarle.

Lo primero que nos vá a hacer falta es el tablero donde se van a jugar las tres en raya, para ello nos vamos a servir del editor de pantallas que tenemos, sitúate en la línea 7 columna 32 y a partir de hay haz el siguiente dibujo.



Para jugar como fichas vamos a utilizar asteriscos, cuando este aparezca en video bajo representará una jugada nuestra, cuando aparezca en video inverso representará una jugada del ordenador. En cada momento para controlar la situación del juego el tablero se ha divido en tres lineas, tres columnas y dos diagonales tal y como sigue.

	C1	C2	C3
L1	D2		D1
L2		D12	
L3	D1		D2

Cuando se quiere saber la situación de un grupo de fichas se explora la tabla dividida como hemos expuesto, para ello nos servimos de ocho contadores, uno para cada división, al inspeccionar las posiciones vamos sumando a estos contadores 1 cada vez que encontremos una ficha, por ejemplo la posición línea 1 columna 1 afecta a los contadores de la primera línea, la primera columna y la segunda diagonal.

Seguidamente tienes el listado del programa, abre en tu editor un archivo con el nombre de 'PRUEBA59.CBL' y teclealo, cuando lo termines y lo compiles dedica un rato a intentar ganar al ordenador.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA59.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
```

```
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

      * SE DECLARA EL FICHERO QUE FORMA EL TABLERO DE JUEGO.

           SELECT TABLERO;
                  ASSIGN TO RANDOM, "TABLERO";
                  ORGANIZATION IS RELATIVE;
                  ACCESS MODE IS DYNAMIC;
                  RELATIVE KEY IS CLV-TAB;
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

      * SE DECLARA EL REGISTRO  QUE FORMA LA PANTALLA.

       FD  TABLERO;
           LABEL RECORD IS STANDARD;
           DATA RECORD IS FIC-TAB.

       01  FIC-TAB.
           03 FIC-LIN                          PIC X(80) OCCURS 24 TIMES.

       WORKING-STORAGE SECTION.

      * LA VARIABLE ERRORES SERA DONDE SE DEPOSITE EL CODIGO DE ERROR
      * EN CASO DE PRODUCIRSE Y CLV-TAB ES EL CODIGO DEL FICHERO DE
      * PANTALLA QUE SIEMPRE SERA 1.

       77  ERRORES                            PIC X(02).
       77  CLV-TAB                            PIC 9(01) VALUE 1.

      * TENEMOS UN TABLA BIDIMENSIONAL DE 3 POR 3 PARA IR ALMACENANDO
      * LAS JUGADAS.

       01  JUGADAS.
           03  JUGADAC                        OCCURS 3 TIMES.
               05  JUGADAL                    OCCURS 3 TIMES.
                   07  JUGADA                 PIC 9(01).

      * EN LAS VARIABLES LINEA Y COLUMNA SE LLEVARA LA POSICION
      * DE LA ULTIMA JUGADA.
```

```
       77  LINEA                          PIC 9(01).
       77  COLUMNA                        PIC 9(01).

      * LA VARIABLE SINO SE UTILIZA PARA LAS PREGUNTAS.

       77  SINO                           PIC X(01).

      * LAS SIGUIENTES VARIABLES SERVIRAN COMO CONTADORES A LA HORA
      * DE INSPECCIONAR EL ESTADO DEL TABLERO.

       77  LINEA1                         PIC 9(02).
       77  LINEA2                         PIC 9(02).
       77  LINEA3                         PIC 9(01).
       77  COLUMNA1                       PIC 9(01).
       77  COLUMNA2                       PIC 9(01).
       77  COLUMNA3                       PIC 9(01).
       77  DIAGONAL1                      PIC 9(01).
       77  DIAGONAL2                      PIC 9(01).

      * EL SW1 SE PONDRA A 1 CUANDO HALLA GANADO EL ORDENADOR, TAMBIEN
      * SE UTILIZA PARA AVERIGUAR SI SE HA LLEGADO A TABLAS.

       77  SW1                            PIC 9(01).

       PROCEDURE DIVISION.
       COMIENZO.

      * SE BORRA LA PANTALLA.

           DISPLAY " " ERASE.

      * SE ABRE EL FICHERO QUE CONTIENE EL TABLERO.

           OPEN I-O TABLERO.

      * SE LEE Y SE VUELVE A CERRAR.

           READ TABLERO.
           CLOSE TABLERO.

      * SE IMPRIME EL TABLERO EN PANTALLA.

           DISPLAY FIC-TAB LINE 1 POSITION 1.

      * SE LLAMA A LA RUTINA LIMPIATABLA QUE LO QUE HACE ES PONER A CERO
      * TODAS LAS CASILLAS PARA COMENZAR A JUGAR.

           PERFORM LIMPIATABLA.
```

```
QUIENJUEGA.

* NOS PREGUNTA SI COMENZAMOS NOSOTROS.

    DISPLAY "JUEGAS TU?" LINE 15 POSITION 25.
    ACCEPT SINO LINE 15 POSITION 35 NO BEEP.

* SI LE DECIMOS QUE NO SE VA A LA ETIQUETA JUEGOYO.

    IF SINO = "N" OR SINO = "n" GO TO JUEGOYO.

* SI LA RESPUESTA NO ES S NI N VOLVEMOS A PREGUNTAR.

    IF SINO NOT = "S" AND SINO NOT = "s" GO TO QUIENJUEGA.

* TRAS LA PREGUNTA BORRAMOS EL MENSAJE.

    DISPLAY SPACES SIZE 20 LINE 15 POSITION 25.

JUEGASTU.

* PONEMOS A CERO TODOS LOS CONTADORES.

    MOVE 0 TO LINEA1, LINEA2, LINEA3, COLUMNA1,
              COLUMNA2, COLUMNA3, DIAGONAL1, DIAGONAL2.

* SE NOS PREGUNTA QUE A QUE FICHA MOVEMOS.

    DISPLAY "A QUE FICHA MUEVES?" LINE 15 POSITION 25.
    DISPLAY "COLUMNA..:" LINE 18 POSITION 25.

PIDECOLUMNA.

* NOS PIDE LA COLUMNA Y CONTROLA QUE ESTA NO SEA MAYOR DE 3
* NI MENOR DE 1 QUE SON LOS LIMITES.

    ACCEPT COLUMNA LINE 18 POSITION 35 NO BEEP.

    IF COLUMNA < 1 OR COLUMNA > 3 GO TO PIDECOLUMNA.
    DISPLAY "LINEA....:" LINE 20 POSITION 25.

PIDELINEA.

* SE PIDE LA LINEA Y SE CONTROLAN LOS LIMITES.

    ACCEPT LINEA LINE 20 POSITION 35 NO BEEP.
    IF LINEA < 1 OR LINEA > 3 GO TO PIDELINEA.
```

```
     OCUPADA.

    * SE IMPRIMEN UNA SERIE DE ESPACIOS PARA BORRAR UN MENSAJE DE
    * JUGADA INVALIDA QUE POSIBLEMENTE HUBIERA IMPRESO.

         DISPLAY SPACES SIZE 20 LINE 22 POSITION 30.

    * SI LA JUGADA QUE NOSOTROS HEMOS INDICADO TIENE UN CERO QUIERE
    * DECIR QUE ESTA LIBRE, POR LO TANTO LA JUGADA ES VALIDA.

         IF JUGADA(COLUMNA, LINEA) = 0 GO TO VALIDA.

    * SI SE LLEGA AQUI QUIERE DECIR QUE LA JUGADA ES INVALIDA PORQUE
    * YA ESTA OCUPADA, SE AVISA Y SE VUELVE A LA ETIQUETA JUEGASTU
    * PARA VOLVER A PEDIR LA JUGADA.

         DISPLAY " JUGADA INVALIDA " LINE 22 POSITION 30
                 REVERSE BLINK.
         GO TO JUEGASTU.

     VALIDA.

    * SI LA JUGADA ES VALIDA LLEGA AQUI, IMPRIME UN ASTERISCO EN
    * VIDEO BAJO EN LA JUGADA INDICADA.

         COMPUTE LINEA1 = LINEA * 2 + 6.
         COMPUTE LINEA2 = COLUMNA * 4 + 30.
         DISPLAY "*" LINE LINEA1 POSITION LINEA2 LOW.

    * MUEVE UN 1 A LA JUGADA INDICADA PARA SABER QUE LA FICHA QUE
    * OCUPA ESA POSICION ES NUESTRA, LA FICHA DEL ORDENADOR SE
    * REPRESENTARA POR UN 2.

         MOVE 1 TO JUGADA(COLUMNA, LINEA).

     EXPLORATABLA.


    * MUEVE 0 A TODOS LOS CONTADORES.

         MOVE 0 TO LINEA1, LINEA2, LINEA3, COLUMNA1, COLUMNA2,
                   COLUMNA3, DIAGONAL1, DIAGONAL2.

    * SE INSPECCIONA EL TABLERO DIVIDIDO EN LINEAS, COLUMNAS Y
    * DIAGONALES MIRANDO NUESTRAS JUGADAS PARA VER SI HEMOS HECHO
```

```
      * TRES EN RAYA.

           IF JUGADA(1, 1) = 1 ADD 1 TO LINEA1, ADD 1 TO COLUMNA1,
                               ADD 1 TO DIAGONAL2.

           IF JUGADA(2, 1) = 1 ADD 1 TO LINEA1, ADD 1 TO COLUMNA2.

           IF JUGADA(3, 1) = 1 ADD 1 TO LINEA1, ADD 1 TO COLUMNA3,
                               ADD 1 TO DIAGONAL1.

           IF JUGADA(1, 2) = 1 ADD 1 TO LINEA2, ADD 1 TO COLUMNA1.

           IF JUGADA(2, 2) = 1 ADD 1 TO LINEA2, ADD 1 TO COLUMNA2,
                               ADD 1 TO DIAGONAL1, ADD 1 TO DIAGONAL2.

           IF JUGADA(3, 2) = 1 ADD 1 TO LINEA2, ADD 1 TO COLUMNA3.

           IF JUGADA(1, 3) = 1 ADD 1 TO LINEA3, ADD 1 TO COLUMNA1,
                               ADD 1 TO DIAGONAL1.

           IF JUGADA(2, 3) = 1 ADD 1 TO LINEA3, ADD 1 TO COLUMNA2.

           IF JUGADA(3, 3) = 1 ADD 1 TO LINEA3, ADD 1 TO COLUMNA3,
                               ADD 1 TO DIAGONAL2.

      * SI ALGUNO DE LOS CONTADORES TIENE 3 QUIERE DECIR QUE LE HEMOS
      * VENCIDO, EN ESE CASO SE VA A LA ETIQUETA MEGANASTE.

           IF LINEA1 = 3 OR LINEA2 = 3 OR LINEA3 = 3 OR
              COLUMNA1 = 3 OR COLUMNA2 = 3 OR COLUMNA3 = 3 OR
              DIAGONAL1 = 3 OR DIAGONAL2 = 3 GO TO MEGANASTE.

       JUEGOYO.


      * SI NO HEMOS TRES EN RAYA EL ORDENADOR PREPARA SU JUGADA.

      * VUELVE A PONER A CERO TODOS LOS CONTADORES.

           MOVE 0 TO LINEA1, LINEA2, LINEA3, COLUMNA1, COLUMNA2,
                     COLUMNA3, DIAGONAL1, DIAGONAL2.

      * INSPECCIONA LA TABLA BUSCANDO SUS JUGADAS.

           IF JUGADA(1, 1) = 2 ADD 1 TO LINEA1, ADD 1 TO COLUMNA1,
                               ADD 1 TO DIAGONAL2.
```

```
        IF JUGADA(2, 1) = 2 ADD 1 TO LINEA1, ADD 1 TO COLUMNA2.

        IF JUGADA(3, 1) = 2 ADD 1 TO LINEA1, ADD 1 TO COLUMNA3,
                            ADD 1 TO DIAGONAL1.

        IF JUGADA(1, 2) = 2 ADD 1 TO LINEA2, ADD 1 TO COLUMNA1.

        IF JUGADA(2, 2) = 2 ADD 1 TO LINEA2, ADD 1 TO COLUMNA2,
                            ADD 1 TO DIAGONAL1, ADD 1 TO DIAGONAL2.

        IF JUGADA(3, 2) = 2 ADD 1 TO LINEA2, ADD 1 TO COLUMNA3.

        IF JUGADA(1, 3) = 2 ADD 1 TO LINEA3, ADD 1 TO COLUMNA1,
                            ADD 1 TO DIAGONAL1.

        IF JUGADA(2, 3) = 2 ADD 1 TO LINEA3, ADD 1 TO COLUMNA2.

        IF JUGADA(3, 3) = 2 ADD 1 TO LINEA3, ADD 1 TO COLUMNA3,
                            ADD 1 TO DIAGONAL2.

    PLANEARJUGADA.

*   SI EN ALGUNA DE LAS LINEAS, COLUMNAS O DIAGONALES TIENE DOS
*   FICHAS Y TIENE OPCION A PONER UNA TERCERA NOS HABRA VENCIDO.

*   HA CONTINUACION VA PREGUNTANDO SI LOS CONTADORES TIENEN DOS Y
*   EXPLORA LA TABLA PARA VER SI PUEDE PONER UNA TERCERA FICHA, EN
*   CASO DE QUE LO ENCUENTRE SE IRA A LA JUGADA CORRESPONDIENTE.

        IF LINEA1 = 2 IF JUGADA(1, 1) = 0 OR
                         JUGADA(2, 1) = 0 OR
                         JUGADA(3, 1) = 0 GO TO JUGADA9.

        IF LINEA2 = 2 IF JUGADA(1, 2) = 0 OR
                         JUGADA(2, 2) = 0 OR
                         JUGADA(3, 2) = 0 GO TO JUGADA10.

        IF LINEA3 = 2 IF JUGADA(1, 3) = 0 OR
                         JUGADA(2, 3) = 0 OR
                         JUGADA(3, 3) = 0 GO TO JUGADA11.

        IF COLUMNA1 = 2 IF JUGADA(1, 1) = 0 OR
                           JUGADA(1, 2) = 0 OR
                           JUGADA(1, 3)   0 GO TO JUGADA12.

        IF COLUMNA2 = 2 IF JUGADA(2, 1) = 0 OR
                           JUGADA(2, 2) = 0 OR
                           JUGADA(2, 3) = 0 GO TO JUGADA13.
```

```
           IF COLUMNA3 = 2 IF JUGADA(3, 1) = 0 OR
                              JUGADA(3, 2) = 0 OR
                              JUGADA(3, 3) = 0 GO TO JUGADA14.

           IF DIAGONAL1 = 2 IF JUGADA(3, 1) = 0 OR
                               JUGADA(2, 2) = 0 OR
                               JUGADA(1, 3) = 0 GO TO JUGADA15.

           IF DIAGONAL2 = 2 IF JUGADA(1, 1) = 0 OR
                               JUGADA(2, 2) = 0 OR
                               JUGADA(3, 3) = 0 GO TO JUGADA16.

      *********

      * SI NO TIENE NINGUNA POSIBILIDAD DE HACER TRES EN RAYA VA A VER
      * SI NOSOTROS TENEMOS ESA OPCON PARA IMPEDIRNOSLO.

      * PONE A 0 TODOS LOS CONTADORES.

           MOVE 0 TO LINEA1, LINEA2, LINEA3, COLUMNA1, COLUMNA2,
                     COLUMNA3, DIAGONAL1, DIAGONAL2.

      * INSPECCIONA LA TABLA BUSCANDO NUESTRAS JUGADAS.

           IF JUGADA(1, 1) = 1 ADD 1 TO LINEA1, ADD 1 TO COLUMNA1,
                               ADD 1 TO DIAGONAL2.

           IF JUGADA(2, 1) = 1 ADD 1 TO LINEA1, ADD 1 TO COLUMNA2.

           IF JUGADA(3, 1) = 1 ADD 1 TO LINEA1, ADD 1 TO COLUMNA3,
                               ADD 1 TO DIAGONAL1.

           IF JUGADA(1, 2) = 1 ADD 1 TO LINEA2, ADD 1 TO COLUMNA1.

           IF JUGADA(2, 2) = 1 ADD 1 TO LINEA2, ADD 1 TO COLUMNA2,
                               ADD 1 TO DIAGONAL1, ADD 1 TO DIAGONAL2.

           IF JUGADA(3, 2) = 1 ADD 1 TO LINEA2, ADD 1 TO COLUMNA3.

           IF JUGADA(1, 3) = 1 ADD 1 TO LINEA3, ADD 1 TO COLUMNA1,
                               ADD 1 TO DIAGONAL1.

           IF JUGADA(2, 3) = 1 ADD 1 TO LINEA3, ADD 1 TO COLUMNA2.

           IF JUGADA(3, 3) = 1 ADD 1 TO LINEA3, ADD 1 TO COLUMNA3,
                               ADD 1 TO DIAGONAL2.

      *******
```

```
* A CONTINUACION INSPECCIONA SI TENEMOS DOS FICHAS CON OPCION A
* PONER UNA TERCERA, SI ES ASI SE VA A LA JUGADA CORRESPONDIENTE
* PARA TAPARNOS ESA POSIBILIDAD OCUPANDO EL LA CASILLA.

     IF LINEA1 = 2 IF JUGADA(1, 1) = 0 OR
                      JUGADA(2, 1) = 0 OR
                      JUGADA(3, 1) = 0 GO TO JUGADA1.

     IF LINEA2 = 2 IF JUGADA(1, 2) = 0 OR
                      JUGADA(2, 2) = 0 OR
                      JUGADA(3, 2) = 0 GO TO JUGADA2.

     IF LINEA3 = 2 IF JUGADA(1, 3) = 0 OR
                      JUGADA(2, 3) = 0 OR
                      JUGADA(3, 3) = 0 GO TO JUGADA3.

     IF COLUMNA1 = 2 IF JUGADA(1, 1) = 0 OR
                        JUGADA(1, 2) = 0 OR
                        JUGADA(1, 3) = 0 GO TO JUGADA4.

     IF COLUMNA2 = 2 IF JUGADA(2, 1) = 0 OR
                        JUGADA(2, 2) = 0 OR
                        JUGADA(2, 3) = 0 GO TO JUGADA5.

     IF COLUMNA3 = 2 IF JUGADA(3, 1) = 0 OR
                        JUGADA(3, 2) = 0 OR
                        JUGADA(3, 3) = 0 GO TO JUGADA6.

     IF DIAGONAL1 = 2 IF JUGADA(3, 1) = 0 OR
                         JUGADA(2, 2) = 0 OR
                         JUGADA(1, 3) = 0 GO TO JUGADA7.

     IF DIAGONAL2 = 2 IF JUGADA(1, 1) = 0 OR
                         JUGADA(2, 2) = 0 OR
                         JUGADA(3, 3) = 0 GO TO JUGADA8.

***********

* SI EL NO TIENE OPCION A HACER TRES EN RAYA Y NOSOTROS TAMPOCO
* MIRARA SI TIENE EN ALGUNA CASILLA UNA FICHA Y SI PUEDE PONER OTRA
* EN LINEA.

* PONE A CERO DE NUEVO LOS CONTADORES.
```

```
        MOVE 0 TO LINEA1, LINEA2, LINEA3, COLUMNA1, COLUMNA2,
                  COLUMNA3, DIAGONAL1, DIAGONAL2.

* EXPLORA LA TABLA BUSCANDO SUS JUGADAS.

     IF JUGADA(1, 1) = 2 ADD 1 TO LINEA1, ADD 1 TO COLUMNA1,
                         ADD 1 TO DIAGONAL2.

     IF JUGADA(2, 1) = 2 ADD 1 TO LINEA1, ADD 1 TO COLUMNA2.

     IF JUGADA(3, 1) = 2 ADD 1 TO LINEA1, ADD 1 TO COLUMNA3,
                         ADD 1 TO DIAGONAL1.

     IF JUGADA(1, 2) = 2 ADD 1 TO LINEA2, ADD 1 TO COLUMNA1.

     IF JUGADA(2, 2) = 2 ADD 1 TO LINEA2, ADD 1 TO COLUMNA2,
                         ADD 1 TO DIAGONAL1, ADD 1 TO DIAGONAL2.

     IF JUGADA(3, 2) = 2 ADD 1 TO LINEA2, ADD 1 TO COLUMNA3.

     IF JUGADA(1, 3) = 2 ADD 1 TO LINEA3, ADD 1 TO COLUMNA1,
                         ADD 1 TO DIAGONAL1.

     IF JUGADA(2, 3) = 2 ADD 1 TO LINEA3, ADD 1 TO COLUMNA2.

     IF JUGADA(3, 3) = 2 ADD 1 TO LINEA3, ADD 1 TO COLUMNA3,
                         ADD 1 TO DIAGONAL2.

* SI EN ALGUNA DE LAS JUGADAS TIENE UNA FICHA Y TIENE OPCION A
* PONER OTRA SE VA A LA JUGADA CORRESPONDIENTE PARA HACERLO.


**********

     IF LINEA1 = 1 IF JUGADA(1, 1) = 0 OR
                      JUGADA(2, 1) = 0 OR
                      JUGADA(3, 1) = 0 GO TO JUGADA17.

     IF LINEA2 = 1 IF JUGADA(1, 2) = 0 OR
                      JUGADA(2, 2) = 0 OR
                      JUGADA(3, 2) = 0 GO TO JUGADA18.

     IF LINEA3 = 1 IF JUGADA(1, 3) = 0 OR
                      JUGADA(2, 3) = 0 OR
                      JUGADA(3, 3) = 0 GO TO JUGADA19.
```

```
      IF COLUMNA1 = 1 IF JUGADA(1, 1) = 0 OR
                         JUGADA(1, 2) = 0 OR
                         JUGADA(1, 3) = 0 GO TO JUGADA20.

      IF COLUMNA2 = 1 IF JUGADA(2, 1) = 0 OR
                         JUGADA(2, 2) = 0 OR
                         JUGADA(2, 3) = 0 GO TO JUGADA21.

      IF COLUMNA3 = 1 IF JUGADA(3, 1) = 0 OR
                         JUGADA(3, 2) = 0 OR
                         JUGADA(3, 3) = 0 GO TO JUGADA22.

      IF DIAGONAL1 = 1 IF JUGADA(3, 1) = 0 OR
                          JUGADA(2, 2) = 0 OR
                          JUGADA(1, 3) = 0 GO TO JUGADA23.

      IF DIAGONAL2 = 1 IF JUGADA(1, 1) = 0 OR
                          JUGADA(2, 2) = 0 OR
                          JUGADA(3, 3) = 0 GO TO JUGADA24.

***********

* SI NO ENCUENTRA TAMPOCO NINGUNA POSIBILIDAD DE PONER UNA FICHA
* EN LINEA EXPLORA QUE CASILLA HAY VACIA, SI ENCUENTRA ALGUNA PONE
* LA LINEA EN LA VARIABLE LINEA Y LA COLUMNA EN LA VARIABLE COLUMNA
* TRAS ELLO SE VA A LA ETIQUETA MIJUGADA.


      IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
         GO TO MIJUGADA.

      IF JUGADA(2, 1) = 0 MOVE 1 TO LINEA, MOVE 2 TO COLUMNA,
         GO TO MIJUGADA.

      IF JUGADA(3, 1) = 0 MOVE 1 TO LINEA, MOVE 3 TO COLUMNA,
         GO TO MIJUGADA.

      IF JUGADA(1, 2) = 0 MOVE 2 TO LINEA, MOVE 1 TO COLUMNA,
         GO TO MIJUGADA.

      IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
         GO TO MIJUGADA.

      IF JUGADA(3, 2) = 0 MOVE 2 TO LINEA, MOVE 3 TO COLUMNA,
         GO TO MIJUGADA.

      IF JUGADA(1, 3) = 0 MOVE 3 TO LINEA, MOVE 1 TO COLUMNA,
         GO TO MIJUGADA.
```

```
       IF JUGADA(2, 3) = 0 MOVE 3 TO LINEA, MOVE 2 TO COLUMNA,
          GO TO MIJUGADA.

       IF JUGADA(3, 3) = 0 MOVE 3 TO LINEA, COLUMNA,
          GO TO MIJUGADA.

  ************

  * SI SE LLEHA AQUI QUIERE DECIR QUE NO QUEDA NINGUNA CASILLA VACIA,
  * POR LO TANTO SE HA LLEGADO TABLAS.

   TABLAS.

  * SE NOS AVISA DE QUE LA PARTIDA QUEDA EN TABLAS Y SE VA A LA
  * ETIQUETA JUEGASOTRAVEZ.

       DISPLAY " LO DEJAMOS EN TABLAS " LINE 20 POSITION 25 REVERSE.
       GO TO JUEGASOTRAVEZ.

   LIMPIATABLA.

  * ESTA RUTINA SE UTILIZA PARA LIMPIAR LA TABLA Y PONERLA TODA A CERO.

  * SE LLAMA A LA ETIQUETA LIMPIA VARIANDO EL CAMPO LINEA DESDE 1
  * HASTA 3 Y EL CAMPO COLUMNA DE 1 HASTA 3, CON LO CUAL SE LIMPIARA
  * LA TABLA COMPLETA.

       PERFORM LIMPIA VARYING LINEA FROM 1 BY 1 UNTIL LINEA > 3
                      AFTER COLUMNA FROM 1 BY 1
                      UNTIL COLUMNA > 3.

   LIMPIA.

  * SE PONE UN CERO EN LA JUGADA INDICADA POR LINEA Y COLUMNA.

       MOVE 0 TO JUGADA(COLUMNA, LINEA).

   MEGANASTE.

  * SI SE LLEGA AQUI ES QUE HEMOS HECHO TRES EN RAYA, SE NOS INDICA
  * QUE LE HEMOS VENCIDO Y SE VA A LA ETIQUETA JUEGASOTRAVEZ.

       DISPLAY " ME GANASTE " REVERSE BLINK LINE 12
               POSITION 25 ERASE.
       PERFORM PITIDO 10 TIMES.
       GO TO JUEGASOTRAVEZ.

   PITIDO.
```

```
* ESTA RUTINA ES LLAMADA POR EL PARRAFO ANTERIOR PARA PROVOCAR
* VARIOS PITIDOS.

    DISPLAY " " LINE 25 POSITION 79 BEEP.

 MIJUGADA.

* A ESTA ETIQUETA SE LLEGARA CUANDO EL ORDENADOR HALLA DECIDIDO
* UNA JUGADA.

* MUEVE UN 2 A LA JUGADA INDICADA POR LINEA Y COLUMNA.

    MOVE 2 TO JUGADA(COLUMNA, LINEA).

* PONE LA JUGADA EN LA CORRESPONDIENTE CASILLA EN REVERSE.

    COMPUTE LINEA1 = LINEA * 2 + 6.
    COMPUTE LINEA2 = COLUMNA * 4 + 30.
    DISPLAY "*" LINE LINEA1 POSITION LINEA2 REVERSE.

* MUEVE 0 A LOS CONTADORES LINEA1 Y LINEA2.

    MOVE 0 TO LINEA1, LINEA2.

* SI EL SW1 CONTIENE UN 1 QUIERE DECIR QUE SE HA LLEGADO DE UNA
* JUGADA EN LA QUE SE HA VENCIDO, EN ESE CASO SE PONE A CERO Y
* SE VA A LA ETIQUETA TEGANE.

    IF SW1 = 1 MOVE 0 TO SW1, GO TO TEGANE.

* EN CASO CONTRARIO SE INSPECCIONA LA TABLA PARA VER SI HAY ALGUNA
* CASILLA LIBRE, YA QUE EN CASO DE NO HABERLA SE HABRE LLEGADO A
* TABLAS.

* SE LLAMA A LA RUTINA INSPECCIONA VARIANDO LINEA1 DESDE 1 HASTA 3
* Y LINEA2 DESDE 1 HASTA 3, CON LO CUAL SE INSPECCIONARA LA TABLA
* COMPLETA, SI EN ALGUNA DE ESTAS INSPECCIONES SE ENCUENTRA UN
* ESPACIO VACIO EN SW1 SE PONDRA UN 1.

    PERFORM INSPECCIONA VARYING LINEA1 FROM 1 BY 1
            UNTIL LINEA1 > 3 AFTER LINEA2 FROM 1 BY 1
            UNTIL LINEA2 > 3.

* SI AL LLEGAR AQUI SW1 CONTIENE 0 QUIERE DECIR QUE NO HAY NINGUNA
* CASILLA VACIA, EN ESE CASO SE VA A LA ETIQUETA TABLAS.
```

```
        IF SW1 = 0 GO TO TABLAS.

    * SI SE LLEGA AQUI QUIERE DECIR QUE AUN QUEDAN CASILLAS POR OCUPAR,
    * SE MUEVE 0 A SW1 Y SE VUELVE A LA ETIQUETA JUEGASTU.

        MOVE 0 TO SW1.
        GO TO JUEGASTU.


     INSPECCIONA.

    * ESTA ES LA RUTINA QUE

        IF JUGADA(LINEA1, LINEA2) = 0 MOVE 1 TO SW1.

    * A CONTINUACION ESTAN TODAS LAS JUGADAS QUE EL ORDENADOR HACE
    * DEPENDIENDO DE LA INSPECCION QUE HACE DE LA TABLA.

     JUGADA1.

        IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 1) = 0 MOVE 1 TO LINEA, MOVE 2 TO COLUMNA,
           GO TO MIJUGADA.

        MOVE 1 TO LINEA, MOVE 3 TO COLUMNA, GO TO MIJUGADA.

     JUGADA2.

        IF JUGADA(1, 2) = 0 MOVE 2 TO LINEA, MOVE 1 TO COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
          GO TO MIJUGADA.

        MOVE 2 TO LINEA, MOVE 3 TO COLUMNA, GO TO MIJUGADA.

     JUGADA3.

        IF JUGADA(1, 3) = 0 MOVE 3 TO LINEA, MOVE 1 TO COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 3) = 0 MOVE 3 TO LINEA, MOVE 2 TO COLUMNA,
           GO TO MIJUGADA.

        MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.
```

```
JUGADA4.

    IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
       GO TO MIJUGADA.

    IF JUGADA(1, 2) = 0 MOVE 2 TO LINEA, MOVE 1 TO COLUMNA,
       GO TO MIJUGADA.

    MOVE 3 TO LINEA, MOVE 1 TO COLUMNA, GO TO MIJUGADA.

JUGADA5.

    IF JUGADA(2, 1) = 0 MOVE 1 TO LINEA, MOVE 2 TO COLUMNA,
       GO TO MIJUGADA.

    IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
       GO TO MIJUGADA.

    MOVE 3 TO LINEA, MOVE 2 TO COLUMNA, GO TO MIJUGADA.

JUGADA6.

    IF JUGADA(3, 1) = 0 MOVE 1 TO LINEA, MOVE 3 TO COLUMNA,
       GO TO MIJUGADA.

    IF JUGADA(3, 2) = 0 MOVE 2 TO LINEA, MOVE 3 TO COLUMNA,
       GO TO MIJUGADA.

    MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

JUGADA7.

    IF JUGADA(3, 1) = 0 MOVE 1 TO LINEA, MOVE 3 TO COLUMNA,
       GO TO MIJUGADA.

    IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
       GO TO MIJUGADA.

    MOVE 3 TO LINEA, MOVE 1 TO COLUMNA, GO TO MIJUGADA.

JUGADA8.

    IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
       GO TO MIJUGADA.

    IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
       GO TO MIJUGADA.
```

```
           MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

       JUGADA9.

           MOVE 1 TO SW1.

           IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
              GO TO MIJUGADA.

           IF JUGADA(2, 1) = 0 MOVE 1 TO LINEA, MOVE 2 TO COLUMNA,
              GO TO MIJUGADA.

           MOVE 1 TO LINEA, MOVE 3 TO COLUMNA, GO TO MIJUGADA.

       JUGADA10.

           MOVE 1 TO SW1.

           IF JUGADA(1, 2) = 0 MOVE 2 TO LINEA, MOVE 1 TO COLUMNA,
              GO TO MIJUGADA.

           IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
              GO TO MIJUGADA.

           MOVE 2 TO LINEA, MOVE 3 TO COLUMNA, GO TO MIJUGADA.

       JUGADA11.

           MOVE 1 TO SW1.

           IF JUGADA(1, 3) = 0 MOVE 3 TO LINEA, MOVE 1 TO COLUMNA,
              GO TO MIJUGADA.

           IF JUGADA(2, 3) = 0 MOVE 3 TO LINEA, MOVE 2 TO COLUMNA,
              GO TO MIJUGADA.

           MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

       JUGADA12.

           MOVE 1 TO SW1.

           IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
              GO TO MIJUGADA.

           IF JUGADA(1, 2) = 0 MOVE 2 TO LINEA, MOVE 1 TO COLUMNA,
              GO TO MIJUGADA.
```

```
        MOVE 3 TO LINEA, MOVE 1 TO COLUMNA, GO TO MIJUGADA.

    JUGADA13.

        MOVE 1 TO SW1.

        IF JUGADA(2, 1) = 0 MOVE 1 TO LINEA, MOVE 2 TO COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 2) = 0 MOVE 2 TO COLUMNA, LINEA,
           GO TO MIJUGADA.

        MOVE 3 TO LINEA, MOVE 2 TO COLUMNA, GO TO MIJUGADA.

    JUGADA14.

        MOVE 1 TO SW1.

        IF JUGADA(3, 1) = 0 MOVE 1 TO LINEA, MOVE 3 TO COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(3, 2) = 0 MOVE 2 TO LINEA, MOVE 3 TO COLUMNA,
           GO TO MIJUGADA.

        MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

    JUGADA15.

        MOVE 1 TO SW1.

        IF JUGADA(3, 1) = 0 MOVE 1 TO LINEA, MOVE 3 TO COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
           GO TO MIJUGADA.

        MOVE 3 TO LINEA, MOVE 1 TO COLUMNA, GO TO MIJUGADA.

    JUGADA16.

        MOVE 1 TO SW1.

        IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
           GO TO MIJUGADA.
```

```
        MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

    JUGADA17.

        IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 1) = 0 MOVE 1 TO LINEA, MOVE 2 TO COLUMNA,
           GO TO MIJUGADA.

        MOVE 1 TO LINEA, MOVE 3 TO COLUMNA, GO TO MIJUGADA.

    JUGADA18.

        IF JUGADA(1, 2) = 0 MOVE 2 TO LINEA, MOVE 1 TO COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
           GO TO MIJUGADA.

        MOVE 2 TO LINEA, MOVE 3 TO COLUMNA, GO TO MIJUGADA.

    JUGADA19.

        IF JUGADA(1, 3) = 0 MOVE 3 TO LINEA, MOVE 1 TO COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(2, 3) = 0 MOVE 3 TO LINEA, MOVE 2 TO COLUMNA,
           GO TO MIJUGADA.

        MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

    JUGADA20.

        IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
           GO TO MIJUGADA.

        IF JUGADA(1, 2) = 0 MOVE 2 TO LINEA, MOVE 1 TO COLUMNA,
           GO TO MIJUGADA.

        MOVE 3 TO LINEA, MOVE 1 TO COLUMNA, GO TO MIJUGADA.

    JUGADA21.

        IF JUGADA(2, 1) = 0 MOVE 1 TO LINEA, MOVE 2 TO COLUMNA,
           GO TO MIJUGADA.
```

```
            IF JUGADA(2, 2) = 0 MOVE 2 TO COLUMNA, LINEA,
               GO TO MIJUGADA.

            MOVE 3 TO LINEA, MOVE 2 TO COLUMNA, GO TO MIJUGADA.

        JUGADA22.

            IF JUGADA(3, 1) = 0 MOVE 1 TO LINEA, MOVE 3 TO COLUMNA,
               GO TO MIJUGADA.

            IF JUGADA(3, 2) = 0 MOVE 2 TO LINEA, MOVE 3 TO COLUMNA,
               GO TO MIJUGADA.

            MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

        JUGADA23.

            IF JUGADA(3, 1) = 0 MOVE 1 TO LINEA, MOVE 3 TO COLUMNA,
               GO TO MIJUGADA.

            IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
               GO TO MIJUGADA.

            MOVE 3 TO LINEA, MOVE 1 TO COLUMNA, GO TO MIJUGADA.

        JUGADA24.

            IF JUGADA(1, 1) = 0 MOVE 1 TO LINEA, COLUMNA,
               GO TO MIJUGADA.

            IF JUGADA(2, 2) = 0 MOVE 2 TO LINEA, COLUMNA,
               GO TO MIJUGADA.

            MOVE 3 TO LINEA, COLUMNA, GO TO MIJUGADA.

        TEGANE.

      * SI EL ORDENADOR NOS VENCE LLEGARA AQUI DONDE NOS LO COMUNICA
      * POR SI NO NOS HABIAMOS DADO CUENTA.

            DISPLAY " TE HE GANADO " LINE 18 POSITION 25 REVERSE
                    BLINK BEEP.

        JUEGASOTRAVEZ.

      * ESTA RUTINA LA UTILIZAMOS CADA VEZ QUE NOS GANA EL ORDENADOR O
      * GANAMOS NOSOTROS PARA PREGUNTAR SI SE QUIERE SEGUIR JUGANDO.
```

```
DISPLAY "JUEGAS OTRA VEZ?" LINE 24 POSITION 25.
ACCEPT SINO LINE 24 POSITION 42 NO BEEP.
IF SINO = "S" OR SINO = "s" GO TO COMIENZO.
IF SINO NOT = "N" AND SINO NOT = "n" GO TO JUEGASOTRAVEZ.
DISPLAY " " ERASE.
STOP RUN.
```

14.- APLICACION DE GESTION DE SUPERMERCADOS.

Por último vamos a desarrollar una pequeña aplicación compuesta de varios programas, la finalidad de esta aplicación vá a ser la gestión de un supermercardo, estará compuesta de varios mantenimientos de ficheros, listados y algunos procesos más. Lo primero que vamos a hacer vá a ser un pequeño análisis de los programas que vamos a tener y la finalidad de cada uno.

MENU PRINCIPAL..: Este programa será el que nos dé acceso a uno de los sub-menús que a continuación veremos, en este primero se nos preguntará sobre opciones generales sin especificar el fichero sobre el que recaeran, por ejemplos mantenimientos.

SUB-MENU 1..: Este primer sub-menú será el de mantenimientos, en el se nos preguntará sobre que fichero vá a recaher la acción del mantenimiento.

SUB-MENU 2..: Este será el menú de listados, en él se nos preguntará que fichero es el que queremos listar.

SUB-MENU 3..: Este tercer sub-menú será el de movimiento diario, en él se nos preguntará que queremos efectuar, una entrada, una salida o el listado del diario.

SUB-MENU 4..: Este cuarto y ultimo sub-menu será el de obtención de informes, en él se nos dará paso a opciones tales como un listado de articulos bajo stock mínimo o un listado de artículos por proveedor.

El menú principal y los cuatro sub-menus irán incorporados en un mismo programa al que llamaremos 'PRUEBA60.CBL', todos los demás programas que tenemos a continuación constituirán un programa por sí sólo.

MANTENIMIENTO DE ARTICULOS..: Esta será la primera opción en el menú de mantenimientos, nos dará paso al programa de mantenimiento de los artículos que compodrán las existencias del supermercado. El funcionamiento de este programa será similar al de otros que ya hemos hecho, se nos preguntará el artículo, si este existe se imprimirá y se nos pedirá opción, en caso contrario se nos pedirán todos los campos.

MANTENIMIENTO DE PROVEEDORES..: Esta será la segunda opción en el sub menú de mantenimientos, en él mantendremos los proveedores que vaya a tener el supermercado, para poder hacer una entrada de artículos obligatoriamente tiene que estar dado de alta el proveedor que nos suministra ese artículo.

LISTADO DE ARTICULOS..: Será la primera opción del sub-menú de listados, se nos preguntará desde que artículo hasta que artículo queremos el listado, dando opción a efectuarlo desde el principio hasta el final del fichero. Como veremos en el programa correspondiente el listado se ha mejorado bastante, controlando la longitud del papel para efectuar el salto de página en el momento preciso.

LISTADO DE PROVEEDORES..: Muy parecido al anterior, será la segunda opción del sub menú de listados, en él se nos pedirá desde que proveedor hasta que proveedor se quiere efectuar el listado.

ENTRADA DE DIARIO..: Será la primera opción en el sub-menú de movimiento diario, en ella podremos dar entrada a los artículos que nos suministran nuestros proveedores.

SALIDA DE DIARIO..: Esta será la segunda opción en el sub menú de movimiento diario, en ella podremos dar salida a nuestras ventas diarias, al dar una salida el programa comprobará antes si hay existencias, en caso de no haberlas no nos permitirá efectuar esa salida.

LISTADO DE DIARIO..: Será la tercerá y última opción del sub menú de movimiento diario, en ella se listará el movimiento que haya habido hasta el momento en la fecha actual, tras listarlo se nos dará opción a actualizar o no, la actualización consiste en pasar las entradas y salidas a los ficheros de artículos y proveedores para

actualizar stocks y acumulados, una vez que un asiento de diario está actualizado no se podrá modificar.

LISTADO DE ARTICULOS BAJO MINIMO..: Esta será la primera opción en el sub-menú de obtención de informes, al ejecutarlá se efectuará un listado de los articulos de los que hay menos que el stock establecido como mínimo.

LISTADO DE ARTICULOS POR PROVEEDOR..: Esta será la segunda opción del sub-menú de obtención de informes, con ella podremos sacar un listado de artículos pero atendiendo al proveedor y no al artículo en sí, con lo que podremos tener un listado de todos los artículos de un determinado proveedor o grupo de proveedores.

Estos son el líneas generales los programas de los que se compondrá la aplicación. Las pantallas de esta aplicación las haremos con el editor que tenemos hecho, para ello las pantallas se exponen a continuación. Pero antes de esto vamos a estudiar el nombre que le vamos a poner a cada uno de los programas, ficheros y pantallas que vamos a utilizar, para ello vamos a hacer la siguiente división: la primera letra del nombre indicará que es el archivo, si es una 'P' el archivo será un programa, si es una 'F' el archivo será un fichero de datos, si es una 'L' el archivo será una libreria y por último si es una 'S' será una pantalla, ya que la 'P' está ocupada; las dos siguientes letras serán iniciales de la finalidad general de la aplicación, serán iguales en todos los archivos y en este caso vamos a poner 'GS' que son las iniciales de 'G'estión de 'S'upermercados; seguidamente en los programas tendremos un número de tres dígitos que nos servirá para saber que programa es en referencia al menú principal, el primero de estos dígitos será el sub-menú al que pertenece el programa, el segundo y tercero serán los indicadores del número de opción en el sub-menú que dá paso a ese programa, por ejemplo 301 sería la opción 1 del sub-menú 3; para nombrar las pantallas seguiremos la misma táctica, ya que cada pantalla tendrá su programa correspondiente a excepción de las correspondientes a menú y submenús; a los ficheros no los podemos identificar de esta forma, ya que un mismo fichero se vá a utilizar en muchos programas de sub-menus diferentes, en vista de esto en lugar de un número le pondremos tres letras que nos indiquen de que fichero se trata, por ejemplo CLI de clientes o PRO de proveedores; por último para nombrar a las librerias pondremos primero tres caracteres identificadores del nombre de fichero a que pertenece la libreria y a continuación tendremos dos caracteres mas que podrán ser SE si es una

declaración de fichero lo que se tiene o FD si es una declaración de registro. Con todo esto vamos a ver a continuación los nombres que daremos a cada uno de los archivos.

PRUEBA60 - Así se llamará el programa que contendrá el menú principal y los distintos sub-menús.

PGS101 - Será el programa uno del sub-menú 1, o sea el mantenimiento de artículos.

PGS102 - Será el nombre del programa de mantenimiento de proveedores.

PGS201 - Así llamaremos al programa de listado de artículos.

PGS202 - Este será el nombre del programa listado de proveedores.

PGS301 - De esta forma llamaremos al programa de entrada de diario.

PGS302 - El programa de salida de diario será el correspondiente a este nombre.

PGS303 - Así llamaremos al programa de listado de diario.

PGS401 - Este será el nombre del programa de listado de artículos bajo mínimo.

PGS402 - Así llamaremos al programa de listado de artículos por proveedor.

SGSPRE - De esta forma llamaremos a la primera pantalla que aparecerá al ejecutar el programa, será la pantalla de presentación.

SGS000 - Así llamaremos a la pantalla del menú principal.

SGS100 - SGS200 - SGS300 - SGS400 - De esta forma denominaremos a las pantallas de los cuatro sub-menús.

SGS101 - SGS102 - SGS201 - SGS202 - SGS301 - SGS302 - SGS303 - SGS401 - SGS402 - Así llamaremos a cada una de las pantallas correspondientes a cada uno de los programas.

FGSART Será el nombre que daremos al fichero de artículos.

FGSPRO Así llamaremos al fichero de proveedores.

FGSMOV - De esta forma llamaremos al ficheros de movimientos.

FGSPAN - Así llamaremos al fichero que utilizaremos para leer la pantalla correspondiente al programa.

LGSARTSE - Será el nombre de la librería de definición de fichero de artículos.

LGSARTFD - Así nombraremos a la librería de descripción de registro del fichero de artículos.

LGSPROSE - A la librería de descripción de fichero de proveedores lo llamaremos de esta forma.

LGSPROFD - De esta forma llamaremos a la descripción de registro del fichero de proveedores.

LGSMOVSE - Será el nombre de descripción de fichero de movimientos.

LGSMOVFD - Así llamaremos a la librería de descripción de registro del fichero de movimientos.

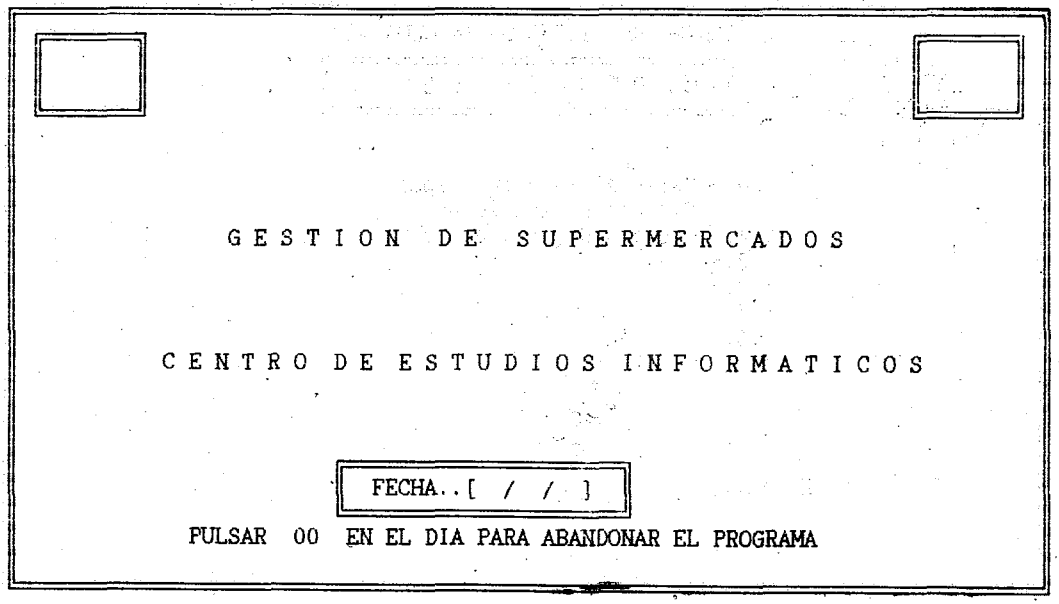
LGSPANSE - Será el nombre que recibirá la librería de descripción de fichero de pantalla.

LGSPANFD - Así llamaremos a la libreria de descripción de registro del fichero de pantalla.

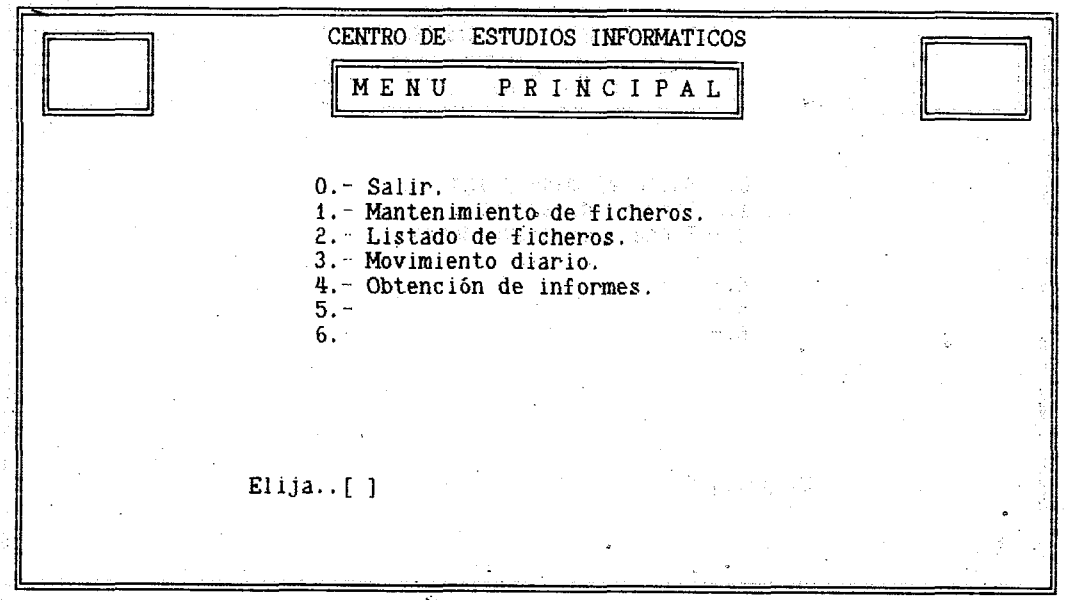
Ya tenemos los nombres de todos los archivos que nos van a hacer falta, el siguiente paso será hacer las pantallas que vá a tener la aplicación para ello como he dicho antes utlizaremos el editor de pantalla que tenemos, a continuación las tienes todas con su nombre.

APLICACION DE GESTION DE SUPERMERCADOS

PANTALLA 'PRESEN' DE PRESENTACION.

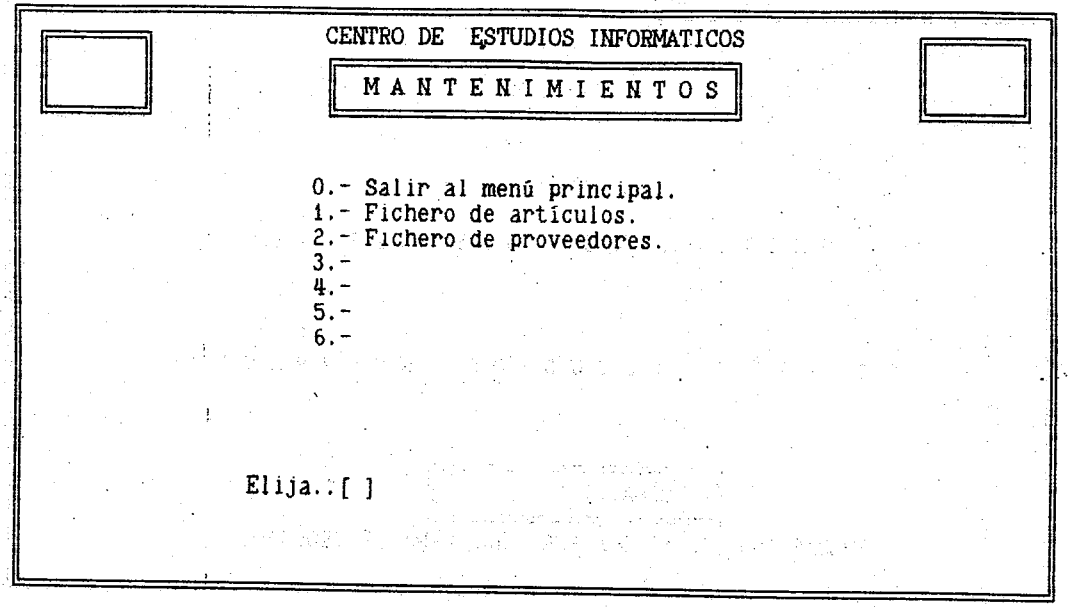


PANTALLA 'SGS000' DE MENU PRINCIPAL.

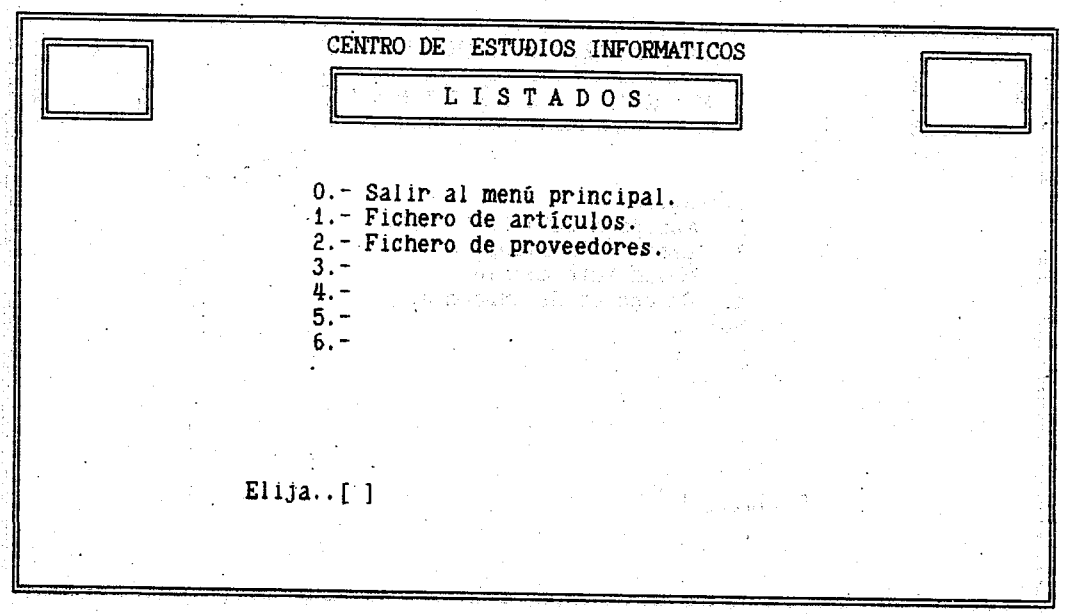


APLICACION DE GESTION DE SUPERMERCADOS

PANTALLA 'SGS100' DE SUB-MENU DE MANTENIMIENTOS.

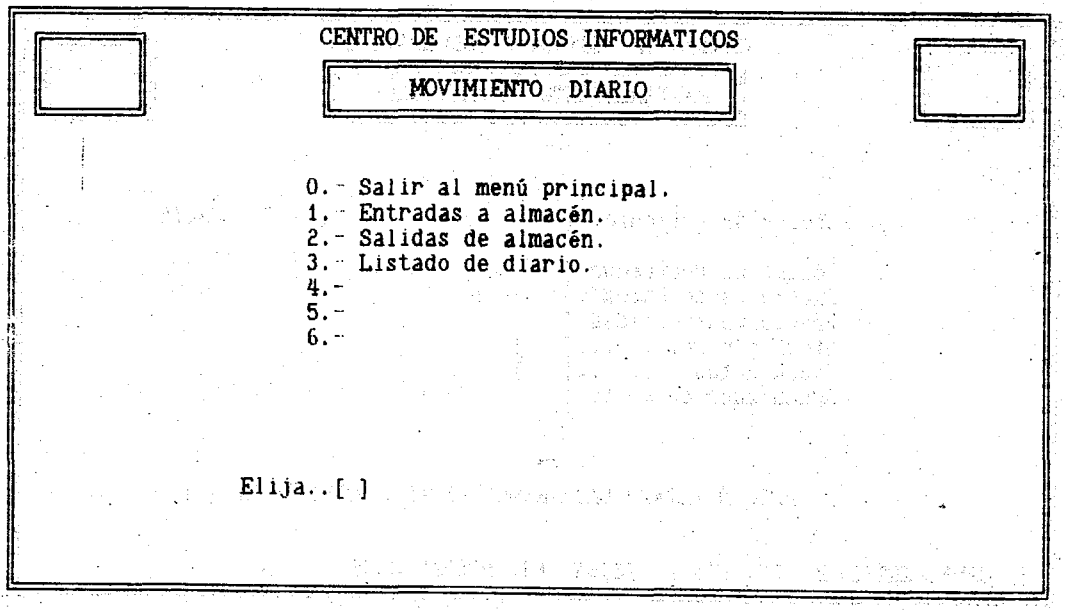


PANTALLA 'SGS200' DE SUB-MENU DE LISTADOS.

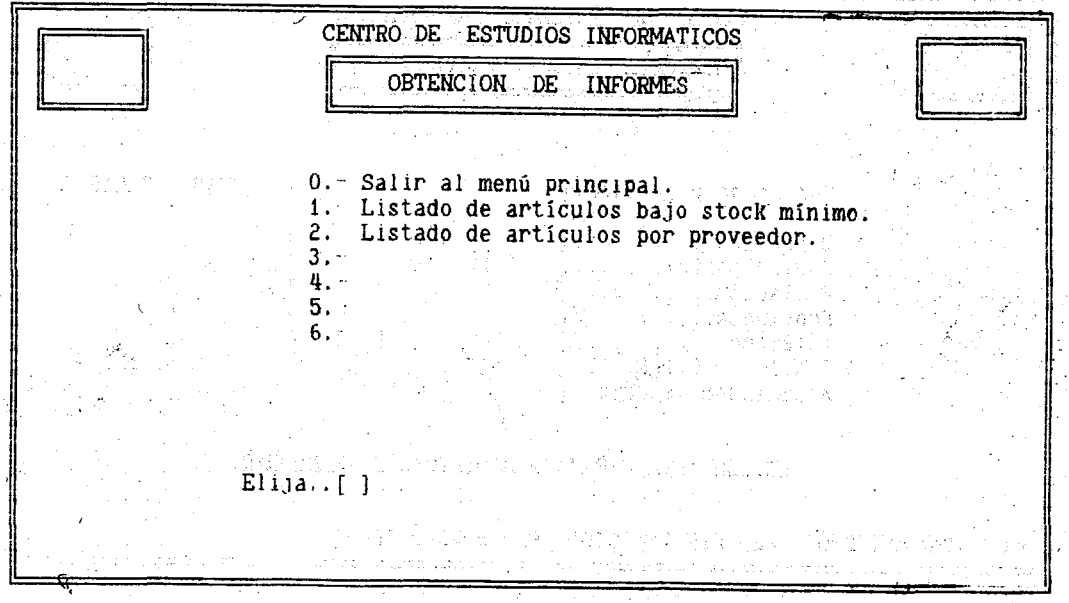


APLICACION DE GESTION DE SUPERMERCADOS

PANTALLA 'SGS300' DE MOVIMIENTO DIARIO.



PANTALLA 'SGS400' DE OBTENCION DE INFORMES.



PANTALLA 'SGS101' DE MANTENIM

```
                    CENTRO DE ESTUDIOS INFORMATICOS
                    MANTENIMIENTO DE ARTICULOS

          Código de artículo..[              ]    TAB = SALIR

          Código de proveedor.[                   ]
          Precio costo.unidad.[      ]
          Precio venta unidad.[      ]
          Stock mínimo.......[   ]
          Stock actual.......[   ]
          Acumulador de venta.[          ]

             RETURN=ALTA/B=BAJA/M=MODIFICAR/S=ABANDONAR..[ ]

F1.CAMPO ANTERIOR  F2. FIN PETICION  F3. BORRAR CAMPO
```

PANTALLA 'SGS102' DE MANTENIMIENTO DE PROVEEDORES.

```
                    CENTRO DE ESTUDIOS INFORMATICOS
                    MANTENIMIENTO DE PROVEEDORES

          Código de proveedor.[                   ]    TAB = SALIR

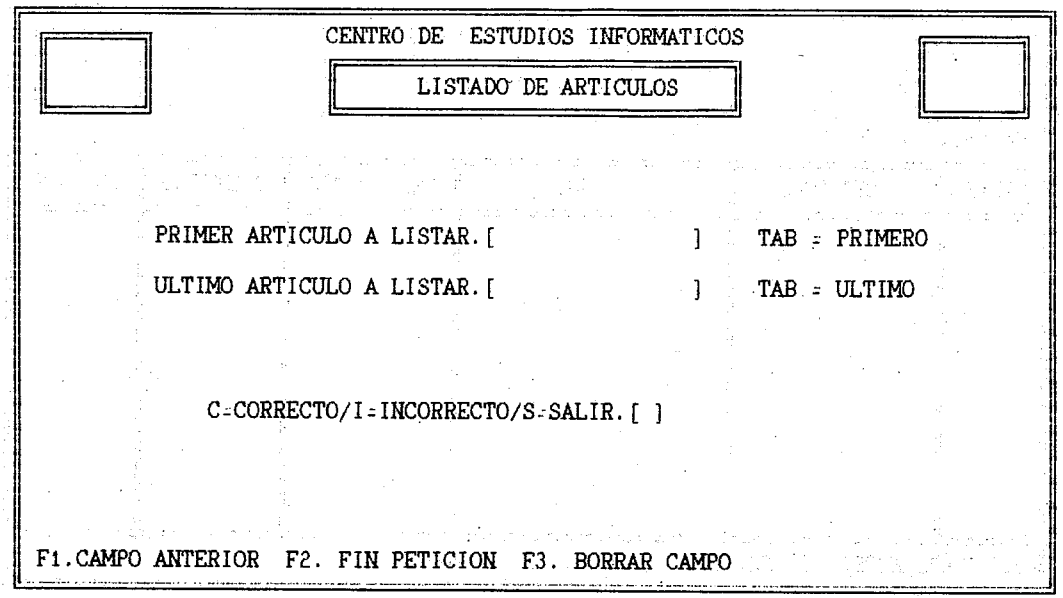
          Dirección..........[                   ]
          Código postal......[     ]
          Población..........[                   ]
          Provincia..........[                   ]
          Teléfono...........[             ]
          D.N.I. o  C.I.F....[               ]
          Acumulador de comp..[            ]

             RETURN=ALTA/B=BAJA/M=MODIFICAR/S=ABANDONAR..[ ]

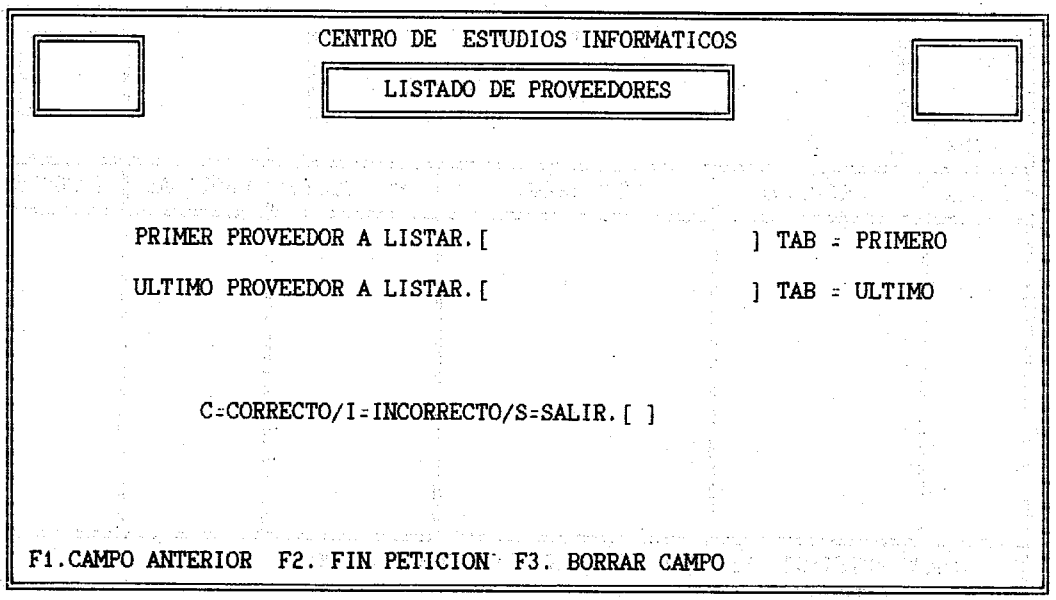
F1.CAMPO ANTERIOR  F2. FIN PETICION  F3. BORRAR CAMPO
```

APLICACION DE GESTION DE SUPERMERCADOS

PANTALLA 'SGS201' DE LISTADO DE ARTICULOS.

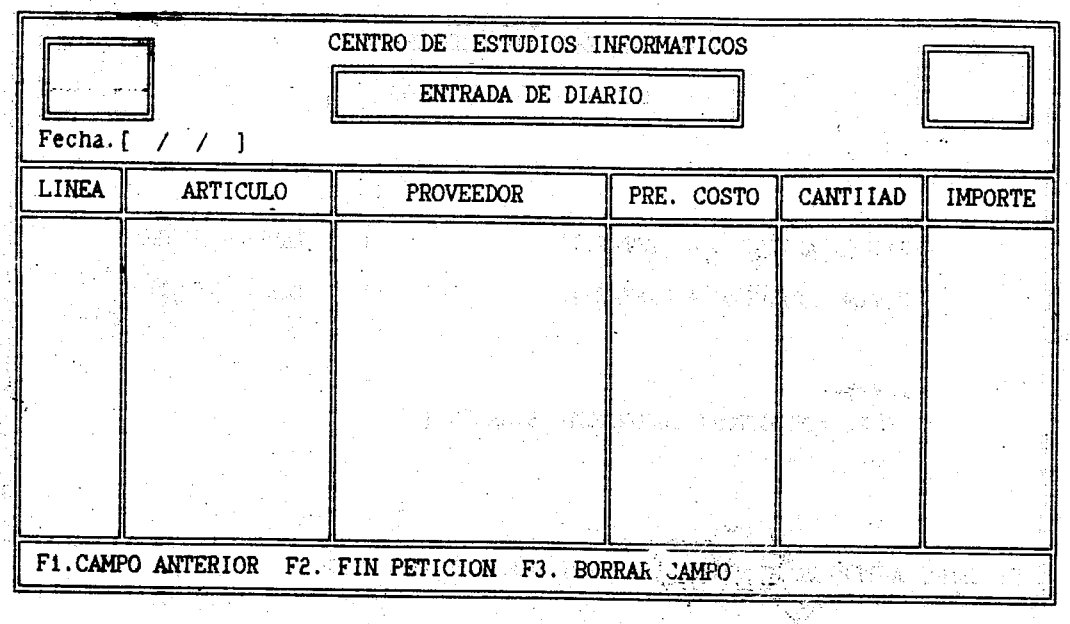


PANTALLA 'SGS202' DE LISTADO DE PROVEEDORES.

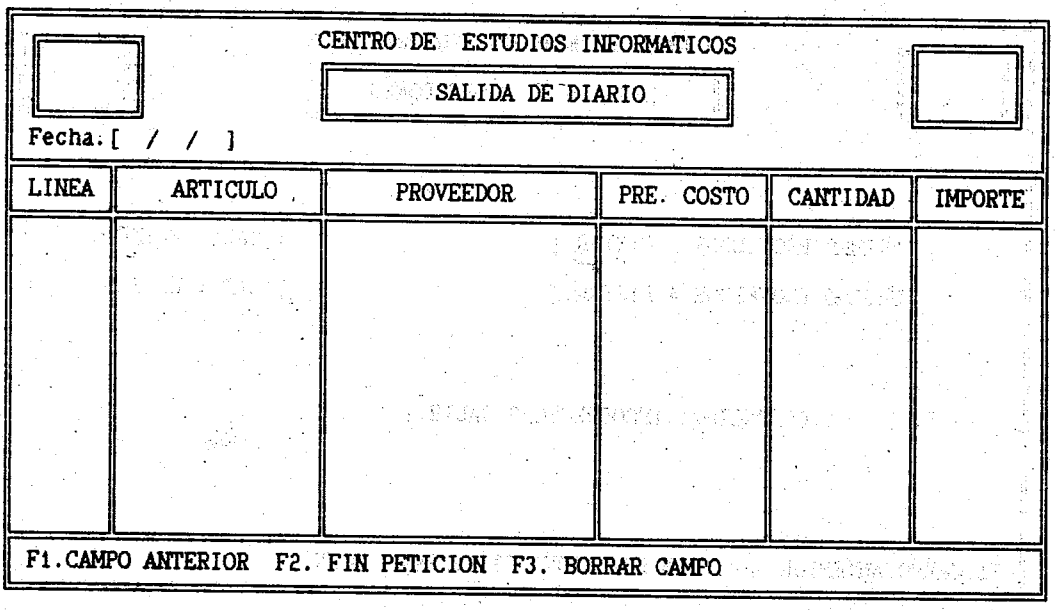


APLICACION DE GESTION DE SUPERMERCADOS

PANTALLA 'SGS301' DE ENTRADA DE MOVIMIENTOS DE DIARIO.

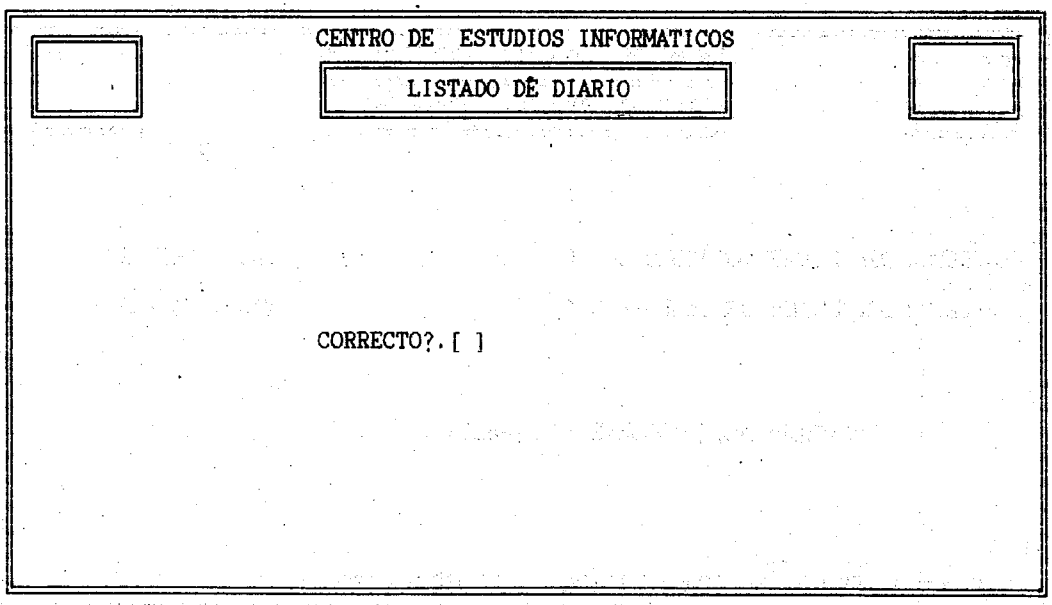


PANTALLA 'SGS302' DE SALIDA DE MOVIMIENTO DE DIARIO.

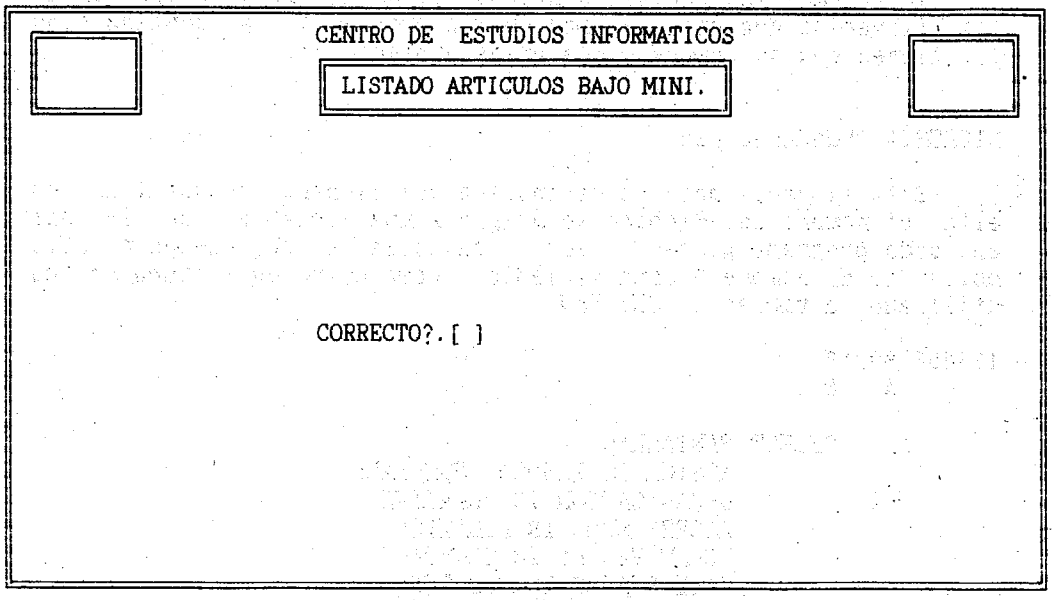


APLICACION DE GESTION DE SUPERMERCADOS

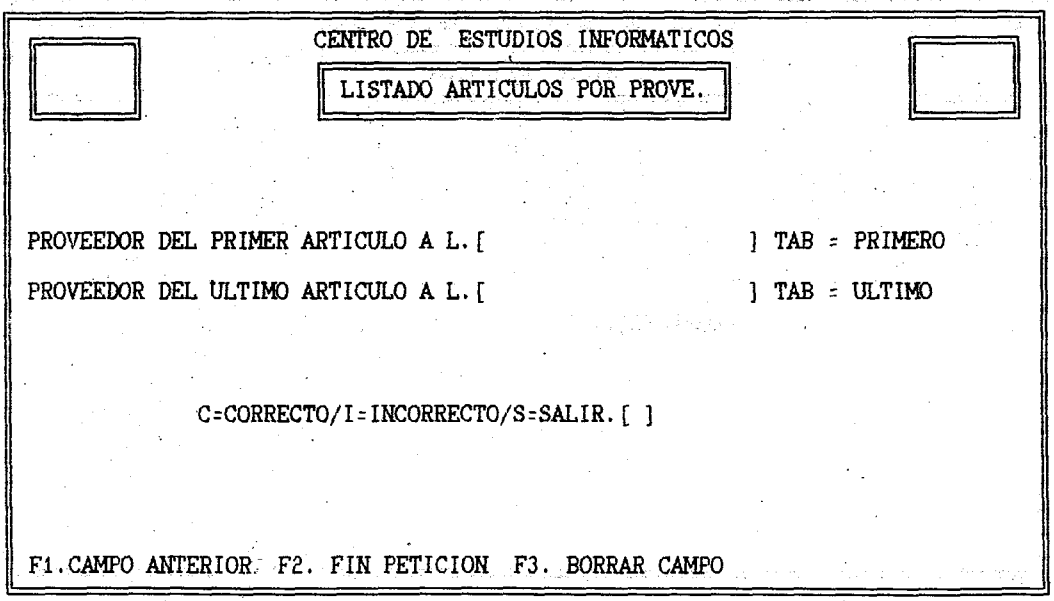
PANTALLA 'SGS303' DE LISTADO DIARIO.



PANTALLA 'SGS401' DE LISTADO DE ARTICULOS BAJO STOCK MINIMO.



PANTALLA 'SGS402' DE LISTADO PROVEEDOR.



Ya tenemos las pantallas hechas, el siguiente paso será crear las librerías que utilizaremos en los programas, a continuación las tienes con su nombre y una breve explicación.

LIBRERIA 'LGSPANSE.LIB' .

Esta librería hace la definición del fichero de pantalla, en ella el nombre del fichero se asigna a una variable, con lo cual en cada programa podremos leer la pantalla correspondiente sólo moviendo el nombre a esta variable, como clave de lectura se ha utilizado la variable 'CLV-PAN'.

```
123456789012
      A   B

          SELECT PANTALLA;
                 ASSIGN TO RANDOM, NOM-PAN;
                 ORGANIZATION IS RELATIVE;
                 ACCESS MODE IS DYNAMIC;
                 RELATIVE KEY IS CLV-PAN;
                 FILE STATUS IS ERRORES.
```

APLICACION DE GESTION DE SUPERMERCADOS

LIBRERIA 'LGSPANFD.LIB".

En esta librería se define el registro del fichero de pantalla, este registro está formado por una tabla de 24 elementos con 80 caracteres cada uno que es el formato en que el generador de pantallas las crea.

```
123456789012
      A   B

      FD  PANTALLA;
          LABEL RECORD IS STANDARD;
          DATA RECORD IS FIC-PAN.

      01  FIC-PAN.
          03  LIN-PAN                        PIC X(80) OCCURS 24 TIMES.
```

LIBRERIA 'LGSARTSE.LIB'.

En esta librería se hace la declaración del fichero de artículos y se asigna al fichero exterior 'FGSART.DAT', el fichero vá a ser indexado y su clave será la variable 'ART-ART' esta como veremos es el artículo al que nos referimos, se tiene un clave alternativa que es el nombre del proveedor, esto nos servirá a la hora de hacer un listado del fichero por orden de proveedor en vez de por la clave primaria.

```
123456789012
      A   B

          SELECT ARTICULOS;
                 ASSIGN TO RANDOM, "FGSART.DAT";
                 ORGANIZATION IS INDEXED;
                 ACCESS MODE IS DYNAMIC;
                 RECORD KEY IS ART-ART;
                 ALTERNATE RECORD KEY IS PRO-ART WITH DUPLICATES;
                 FILE STATUS IS ERRORES.
```

LIBRERIA 'LGSARTFD.LIB'.

En esta librería hacemos la declaración del registro del fichero de artículos, este está formado por la variable 'ART-ART' que será el nombre del artículo, es la clave primaria o principal, 'PRO-ART' es el proveedor que nos suministra este artículo, es la clave secundaria o alternativa; 'PRC-ART' es el precio costo del

artículo, 'PRV-ART' es el precio venta publico, 'STM-ART' es el stock mínimo que deseamos tener de ese artículo, 'STA-ART' almacenará el stock actual de dicho artículo y por último 'ACU-ART' es un acumulador que nos irá diciendo el importe de lo que se ha vendido de ese artículo hasta el momento.

```
123456789012
      A   B

      FD  ARTICULOS;
          LABEL RECORD IS STANDARD;
          DATA RECORD IS FIC-ART.

      01  FIC-ART.
          03  ART-ART                    PIC X(15).
          03  PRO-ART                    PIC X(20).
          03  PRC-ART                    PIC 9(05).
          03  PRV-ART                    PIC 9(05).
          03  STM-ART                    PIC 9(03).
          03  STA-ART                    PIC 9(03).
          03  ACU-ART                    PIC 9(10).
```

LIBRERIA 'LGSPROSE.LIB'.

En esta librería hacemos la declaración del fichero de proveedores, dicho fichero será indexado teniendo como clave el campo 'PRO-PRO' que será la denominación de ese proveedor.

```
123456789012
      A   B

          SELECT PROVEEDORES;
                 ASSIGN TO RANDOM, "FGSPRO.DAT";
                 ORGANIZATION IS INDEXED;
                 ACCESS MODE IS DYNAMIC;
                 RECORD KEY IS PRO-PRO;
                 FILE STATUS IS ERRORES.
```

LIBRERIA 'LGSPROFD.LIB'.

En ella haremos la definición del registro del fichero de proveedores, este tendrá los campos 'PRO-PRO' que será el nombre del proveedor, 'DIR-PRO' que será la dirección, 'COD-PRO' será el código postal, 'POB-PRO' la población, 'PRV-PRO' la provincia, 'TEL-PRO' el teléfono, 'DNI-PRO' el D.N.I. o C.I.F. de ese

proveedor y por último 'ACU-PRO' que será un acumulador de compras hechas a ese proveedor.

```
123456789012
      A   B

      FD  PROVEEDORES;
          LABEL RECORD IS STANDARD;
          DATA RECORD IS FIC-PRO.

      01  FIC-PRO.
          03  PRO-PRO                    PIC X(20).
          03  DIR-PRO                    PIC X(20).
          03  COD-PRO                    PIC 9(05).
          03  POB-PRO                    PIC X(20).
          03  PRV-PRO                    PIC X(20).
          03  TEL-PRO                    PIC X(13).
          03  DNI-PRO                    PIC X(15).
          03  ACU-PRO                    PIC 9(10).
```

LIBRERIA 'LGSMOVSE.LIB'.

Es la definición del fichero de movimientos, este al igual que los dos anteriores será indexado y tendrá como clave el grupo 'COD-MOV'.

```
123456789012
      A   B

          SELECT MOVIMIENTO;
                 ASSIGN TO RANDOM, "FGSMOV.DAT";
                 ORGANIZATION IS INDEXED;
                 ACCESS MODE IS DYNAMIC;
                 RECORD KEY IS COD-MOV;
                 FILE STATUS IS ERRORES.
```

LIBRERIA 'LGSMOVFD.LIB'.

En ella haremos la definición del registro del fichero de movimientos, tenemos el grupo 'COD-MOV' que es la clave de este fichero y que está compuesto por 'FEC-MOV' que será la fecha del movimiento, 'TIP-MOV' que será el tipo de ese movimiento, entrada o salida, 'LIN-MOV' que será la línea de movimiento dentro de esa fecha y ese tipo; fuera de la clave tenemos los campos 'COA-MOV' que es el artículo que sale o entra, 'CAN-MOV' que es la cantidad

de ese artículo, 'PRE-MOV' que será el precio y por último 'EST-MOV' será un campo de estado que como veremos le servirá al programa para saber si unos asientos ya han sido actualizados o no para evitar el doblar cantidades.

```
123456789012
      A   B

      FD  MOVIMIENTO;
          LABEL RECORD IS STANDARD;
          DATA RECORD IS FIC-MOV.

      01  FIC-MOV.
          03  COD-MOV.
              05  FEC-MOV.
                  07  ANO-MOV              PIC X(02).
                  07  MES-MOV              PIC X(02).
                  07  DIA-MOV              PIC X(02).
              05  TIP-MOV                  PIC X(01).
              05  LIN-MOV                  PIC 9(03).
          03  COA-MOV                      PIC X(15).
          03  CAN-MOV                      PIC 9(03).
          03  PRE-MOV                      PIC 9(05).
          03  EST-MOV                      PIC 9(01).
```

Bueno, pues hasta ahora tenemos programas que van a componer la aplicación, sus nombres, sus pantallas y las librerías que se van a utilizar; el siguiente paso será comenzar a hacer los programas y vamos a empezar por el programa que actuará de presentación, de menú principal y de sub-menús a este le vamos a llamar 'PRUEBA60.CBL' como vimos antes. Antes de comenzar con el listado del programa vamos a hacer un breve seguimiento del mismo.

El primer paso que daremos en este programa será sacar la pantalla de presentación, en ella pediremos la fecha controlando que esta sea válida, para ello controlamos al pedir el día que este esté entre 1 y 31, al pedir el mes se controla que esté entre 1 y 12 y además que si el mes es uno de los de 30 días el día que nos hallan dado no sea 31, por último al pedir el año se mira que este no sea menor que el 86, luego se divide entre 4, al dividir un año entre cuatro si hay resto quiere decir que no es bisiesto, en caso contrario si lo es, con esto controlamos si el mes de Febrero puede tener o no mas de 28 dias; tras haber dado la fecha pasamos al menú principal, en él se nos pide opción a entrar en

Mantenimientos, Listados, Movimiento diario u Obtención de informes, estas opciones están numerada de 1 a 4, con lo cual para elegir una tendremos que pulsar su número correspondiente, si se pulsa cero se volverá a la pantalla de presentación donde podremos modificar la fecha o salir si como día pulsamos cero; una vez que se ha elegido una opción se vá al sub-menú correspondiente donde se nos volverá a pedir opción respecto de ese sub-menú, si en cualquiera de ellos la opción es cero el resultado será la vuelta al menú principal, en caso contrario se llamará al correspondiente programa pasando la fecha que se ha pedido, cuando se vuelva de este programa cada opción volverá a su menú correspondiente.

Ya tenemos una idea bastante acertada de lo que debe ser el programa, a continuación tienes el listado que como siempre lleva incluidos bastantes comentarios para aclarar su funcionamiento.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PRUEBA60.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

      * SE HACE UN COPY A LA LIBRERIA DE SELECT DE LA PANTALLA.

           COPY "LGSPANSE.LIB".

       DATA DIVISION.
       FILE SECTION.

      * Y A LA FD DE LA MISMA.

       COPY "LGSPANFD.LIB".

       WORKING-STORAGE SECTION.

      * LA VARIABLE ERRORES SERA EL FILE STATUS EN TODOS LOS PROGRAMAS.

       77  ERRORES                          PIC X(02).
```

```
    * NOM-PAN ES EL NOMBRE AL QUE SE HA ASIGNADO EL FICHERO DE PANTALLA
    * DE ESTA FORMA PONIENDO EN ESTE CAMPO EL NOMBRE DE LA PANTALLA QUE
    * DESEAMOS LEER LO PODREMOS HACER.

     77  NOM-PAN                          PIC X(06).

    * OPCION SERA LA VARIABLE UTILIZADA PARA PEDIR OPCION EN LOS MENUS

     77  OPCION                           PIC 9(01).

    * CLV-PAN ES EL CODIGO EN EL FICHERO DE PANTALLA, SE PONE A 1 Y
    * NO SE MODIFICARA A LO LARGO DE LA PROCEDURE.

     77  CLV-PAN                          PIC 9(01) VALUE 1.

    * RESTO Y RESUL SON DOS VARIABLES QUE UTILIZAREMOS AL EFECTUAR UNA
    * DIVISION PARA COMPROBAR LA FECHA.

     77  RESTO                            PIC 9(02) VALUE 0.
     77  RESUL                            PIC 9(02).

    * EN EL GRUPO FECHA PONDREMOS LA FECHA ACTUAL PARA PASAR ESTA A
    * TODOS LOS DEMAS PROGRAMAS.

     01  FECHA.
         03  ANO                          PIC 9(02).
         03  MES                          PIC 9(02).
         03  DIA                          PIC 9(02).

     PROCEDURE DIVISION.
     PRINCIPIO.

    * PRIMERO SE BORRA LA PANTALLA.

         DISPLAY " " ERASE.

     COMIENZO.

    * SE ELIGE LA PANTALLA 'PRESEN' QUE ES LA PANTALLA DE PRESENTACION.

         MOVE "PRESEN" TO NOM-PAN.

    * SE ABRE SE LEE Y SE CIERRA EL FICHERO DE PANTALLA.

         OPEN INPUT PANTALLA.
         READ PANTALLA.
         CLOSE PANTALLA.
```

```
* SE IMPRIME EL REGISTRO QUE SERA LA PANTALLA INDICADA.

     DISPLAY FIC-PAN LINE 1 POSITION 1.

 PEDIRFECHA.

* EL SIGUIENTE PASO SERA PEDIR LA FECHA.

 PIDEDIA.

* LO PRIMERO QUE SE PIDE ES EL DIA.

     ACCEPT DIA LINE 20 POSITION 37 NO BEEP ECHO UPDATE.

* SE COMPRUEBA QUE SEA NUMERICO, EN CASO CONTRARIO SE VUELVE A PEDIR

     IF DIA NOT NUMERIC GO TO PIDEDIA.

* SI EL DIA HA SIDO 0 SE DETIENE EL PROGRAMA.

     IF DIA = 0 DISPLAY " " ERASE, STOP RUN.

* EN CASO CONTRARIO SE CONTROLA QUE EL DIA NO SEA MENOR DE 1
* NI MAYOR DE 31.

     IF DIA < 1 OR DIA > 31 GO TO PIDEDIA.

 PIDEMES.

* A CONTINUACION SE PIDE EL MES.

     ACCEPT MES LINE 20 POSITION 40 NO BEEP ECHO UPDATE.

* SE COMPRUEBA QUE SEA NUMERICO, SI NO LO ES SE VUELVE A PEDIR.

     IF MES NOT NUMERIC GO TO PIDEMES.

* SE CONTROLA QUE EL MES ESTE ENTRE 1 Y 12, SI NO LO ESTA ES
* INVALIDO Y POR LO TANTO SE VUELVE A PEDIR.

     IF MES < 1 OR MES > 12 GO TO PIDEMES.

* SE COMPRUEBA QUE SI EL MES DADO ES ALGUNA DE LOS DE 30 DIAS
* O ES FEBRERO EL DIA NO SEA 31, DE SER ASI SE VUELVE AL PRINCIPIO
* A COMENZAR A PEDIR LA FECHA YA QUE ESTA ES INVALIDA.

     IF MES = 2 OR MES = 4 OR MES = 6 OR MES = 9 OR MES = 11
        IF DIA = 31 GO TO PIDEDIA.
```

```
PIDEANO.
* POR ULTIMO PIDE EL AÑO.
    ACCEPT ANO LINE 20 POSITION 43 NO BEEP ECHO UPDATE.
* COMPRUEBA QUE SEA NUMERICO, DE NO SER ASI SE VUELVE A PEDIR.
    IF ANO NOT NUMERIC GO TO PIDEANO.
* SE COMPRUEBA QUE EL AÑO NO SEA MENOR DE 1986.
    IF AÑO < 86 GO TO PIDEANO.
* PARA AVERIGUAR SI EL AÑO ES BISIESTO SE DIVIDE ESTE ENTRE 4,
* EL RESULTADO SE ALMACENA EN RESUL Y EL RESTO EN EL CAMPO
* RESTO, SI EL RESTO DE DIVIDIR EL AÑO ENTRE 4 ES 0 QUIERE
* DECIR QUE EL AÑO ES BISIESTO EN CASO CONTRARIO NO LO ES.
    DIVIDE ANO INTO 4 GIVING RESUL REMAINDER RESTO.
* SE COMPRUEBA QUE SI EL AÑO NO ES BISIESTO, EL MES DADO HA SIDO
* FEBRERO Y EL DIA FACILITADO ES MAYOR DE 28 LA FECHA ES INVALIDA
* Y SE VUELVE A PEDIR.
    IF RESTO NOT = 0 AND MES = 2 AND DIA > 28 GO TO PIDEDIA.
MENUO.
* UNA VEZ QUE SE HA FACILITADO LA FECHA SE LLEGA AL MENU PRINCIPAL
* SE MUEVE EL NOMBRE DE LA PANTALLA DEL MENU PRINCIPAL A NOM-PAN.
    MOVE "SGS000" TO NOM-PAN.
* SE ABRE, LEE Y CIERRA EL FICHERO DE PANTALLA.
    OPEN INPUT PANTALLA.
    READ PANTALLA.
    CLOSE PANTALLA.
* SE IMPRIME EL REGISTRO QUE TENDRA LA PANTALLA DEL MENU.
    DISPLAY FIC-PAN LINE 1 POSITION 1.
PIDEOPCIONO.
* SE PIDE LA OPCION Y SE DEPOSITA EN EL CAMPO OPCION.
```

```
        ACCEPT OPCION LINE 20 POSITION 27 NO BEEP.
    * SE CONTROLA QUE LA OPCION NO SEA MENOR DE 0 NI MAYOR DE 4.
        IF OPCION < 0 OR OPCION > 4 GO TO PIDEOPCION0.
    * SI LA OPCION ES 0 SE VUELVE A LA ETIQUETA COMIENZO, CON LO CUAL
    * PODREMOS VOLVER DESDE EL MENU A VOLVER A CAMBIAR LA FECHA O A
    * SALIR.
        IF OPCION = 0 GO TO COMIENZO.
    * SI NO ES CERO SE VA A UNO DE LOS MENUS DEPENDIENDO DEL VALOR
    * QUE SE HALLA DEPOSITADO EN OPCION.
        GO TO MENU1, MENU2, MENU3, MENU4 DEPENDING ON OPCION.
     MENU1.
    * ESTE PRIMER SUB-MENU ES EL DE MANTENMIENTOS.
    * SE MUEVE EL NOMBRE DE LA PANTALLA A NOM-PAN.
        MOVE "SGS100" TO NOM-PAN.
    * SE ABRE, LEE Y CIERRA EL FICHERO DE PANTALLA.
        OPEN INPUT PANTALLA.
        READ PANTALLA.
        CLOSE PANTALLA.
    * SE IMPRIME EL REGISTRO QUE CONTENDRA LA PANTALLA.
        DISPLAY FIC-PAN LINE 1 POSITION 1.
     PIDEOPCION1.
    * TRAS ELLO PIDE OPCION.
        ACCEPT OPCION LINE 20 POSITION 27 NO BEEP.
    * CONTROLA QUE ESTA NO SEA MENOR DE 0 NI MAYOR DE 2.
        IF OPCION < 0 OR OPCION > 2 GO TO PIDEOPCION1.
    * SI LA OPCION PULSADA HA SIDO 0 SE VUELVE AL MENU PRINCIPAL.
        IF OPCION = 0 GO TO MENU0.
```

```
      * EN CASO CONTRARIO SE VA A UNA DE LAS ETIQUETAS MANTENIMIENTO
      * DEPENDIENDO DEL VALOR DEL CAMPO OPCION.

           GO TO MANTENIMIENTO1, MANTENIMIENTO2 DEPENDING ON OPCION.

       MENU2.

      * ESTE ES EL SEGUNDO SUB-MENU, LISTADOS.

      * SE MUEVE EL NOMBRE DE LA PANTALLA A NOM-PAN.

           MOVE "SGS200" TO NOM-PAN.

      * SE ABRE, LEE Y CIERRA EL FICHERO DE PANTALLA.

           OPEN INPUT PANTALLA.
           READ PANTALLA.
           CLOSE PANTALLA.

      * SE IMPRIME EL REGISTRO QUE CONTIENE LA PANTALLA.

           DISPLAY FIC-PAN LINE 1 POSITION 1.

       PIDEOPCION2.

      * SE PIDE LA OPCION.

           ACCEPT OPCION LINE 20 POSITION 27 NO BEEP.

      * SE CONTROLA QUE ESTA NO SEA MAYOR DE 2 NI MENOR DE 0.

           IF OPCION < 0 OR OPCION > 2 GO TO PIDEOPCION2.

      * SI LA OPCION ES CERO SE VUELVE AL MENU PRINCIPAL.

           IF OPCION = 0 GO TO MENU0.

      * EN CASO CONTRARIO SE VA A UNO DE LOS LISTADOS DEPENDIENDO DEL
      * VALOR DEPOSITADO EN EL CAMPO OPCION.

           GO TO LISTADO1, LISTADO2 DEPENDING ON OPCION.

       MENU3.

      * ESTE ES EL SUB-MENU 3, MOVIMIENTO DIARIO.

      * SE MUEVE EL NOMBRE DE LA PANTALLA A NOM-PAN.
```

```
        MOVE "SGS300" TO NOM-PAN.

    * SE ABRE, LEE Y CIERRA EL FICHERO DE PANTALLA.

        OPEN INPUT PANTALLA.
        READ PANTALLA.
        CLOSE PANTALLA.

    * SE IMPRIME EL REGISTRO QUE CONTIENE LA PANTALLA.

        DISPLAY FIC-PAN LINE 1 POSITION 1.

     PIDEOPCION3.

    * SE PIDE OPCION.

        ACCEPT OPCION LINE 20 POSITION 27 NO BEEP.

    * SE CONTROLA QUE ESTA ESTE ENTRE 0 Y 3.

        IF OPCION < 0 OR OPCION > 3 GO TO PIDEOPCION3.

    * SI ES CERO SE VUELVE AL MENU PRINCIPAL.

        IF OPCION = 0 GO TO MENU0.

    * EN CASO CONTRARIO SE VA A UNA DE LAS ETIQUETAS DEPENDIENDO
    * DEL VALOR DEPOSITADO EN OPCION.

        GO TO ENTRADAS, SALIDAS, LISTADO DEPENDING ON OPCION.

     MENU4.

    * ESTE ES EL CUARTO Y ULTIMO SUB-MENU, OBTENCION DE INFORMES.

    * SE MUEVE EL NOMBRE DE LA PANTALLA A LA VARIABLE NOM-PAN.

        MOVE "SGS400" TO NOM-PAN.

    * SE ABRE, LEE Y CIERRA EL FICHERO DE PANTALLA.

        OPEN INPUT PANTALLA.
        READ PANTALLA.
        CLOSE PANTALLA.

    * SE IMPRIME EL REGISTRO QUE CONTIENE LA PANTALLA.
```

```
        DISPLAY FIC-PAN LINE 1 POSITION 1.

    PIDEOPCION4.
    * SE PIDE OPCION.

        ACCEPT OPCION LINE 20 POSITION 27 NO BEEP.

    * SE CONTROLA QUE LA OPCION ESTE ENTRE 0 Y 2.

        IF OPCION < 0 OR OPCION > 2 GO TO PIDEOPCION4.

    * SI LA OPCION ES 0 SE VUELVE AL MENU PRINCIPAL.

        IF OPCION = 0 GO TO MENU0.

    * EN CASO CONTRARIO SE VA A UNA DE LAS ETIQUETAS DEPENDIENDO
    * DEL VALOR DE OPCION.

        GO TO INFORMES1, INFORMES2 DEPENDING ON OPCION.

    * A CONTINUACION ESTA CADA UNA DE LAS ETIQUETAS A LAS QUE SE LLAMA
    * DESDE LOS DISTINTOS MENUS, EN CADA UNA DE ELLAS SE LLAMA AL
    * PROGRAMA CORRESPONDIENTE PASANDO EL GRPO FECHA PARA PODER
    * UTILIZARLO EN LOS DEMAS PROGRAMAS, CUANDO SE VUELVE DEL CADA UNO
    * DE LOS PROGRAMAS SE VA A SU MENU CORRESPONDIENTE A VOLVER A PEDIR
    * OPCION.

     MANTENIMIENTO1.

         CALL "PGS101" USING FECHA.
         GO TO MENU1.

     MANTENIMIENTO2.

         CALL "PGS102" USING FECHA.
         GO TO MENU1.

     LISTADO1.

         CALL "PGS201" USING FECHA.
         GO TO MENU2.

     LISTADO2.

         CALL "PGS202" USING FECHA.
         GO TO MENU2.
```

```
ENTRADAS.

    CALL "PGS301" USING FECHA.
    GO TO MENU3.

SALIDAS.

    CALL "PGS302" USING FECHA.
    GO TO MENU3.

LISTADO.

    CALL "PGS303" USING FECHA.
    GO TO MENU3.

INFORMES1.

    CALL "PGS401" USING FECHA.
    GO TO MENU4.

INFORMES2.

    CALL "PGS402" USING FECHA.
    GO TO MENU4.
```

El siguiente programa que tenemos es el de mantenimiento del fichero de artículos 'PGS101'. Seguidamente tienes un breve seguimiento del funcionamiento de este programa.

La finalidad del programa es que pidiendo un artículo nos pida los demás campos si este artículo no existe o nos imprima los campos en pantalla si es que ya existe, despues de uno de los dos procesos nos pida opción y actúe en consecuencia. En la petición de campos hemos puesto tres teclas especiales controladas por medio de la opción ON EXCEPTION, estas teclas serán F1, el cometido de ella será llevarnos al campo anterior, F2 nos llevará directamente a pedir opción dejando la petición de campos, útil sobre todo cuando se ha de modificar un campo y F3 que borrará el campo donde estemos.

Al comenzar el programa lo primero que tiene son las DECLARATIVES, estas las usamos porque al principio los ficheros darán errores al no existir los mismos en el disco, cuando ocurra un error se nos dará el código de ese error y se nos preguntará si queremos crearlo, si el error es 94 es que el fichero todavía no existe y por lo tanto podemos responder afirmativamente a la

pregunta, si se produce un error diferente consulta el apéndice de errores para saber a que se debe. Tras las DECLARATIVES tenemos el programa en sí, primero saca la pantalla de petición de campos que es la SGS101, tras ello pone a cero los campos que no se van a pedir, ya que hay campos como stock actual o acumulado de ventas que se actualizan con el movimiento diario y no los podemos manipular, seguidamente imprimimos en pantalla la fecha actual que se paso del programa menú, se abren los ficheros de artículos y proveedores en modo I-O y se comienza la petición de campos. Lo primero que se pide es el artículo, cuando este se ha facilitado se comprueba si existe o no, en caso de que exista se imprimen los demás campos y se vá a pedirnos opción, en caso contrario nos irá pidiendo los demás campos. Como he dicho antes en la petición de un campo podemos pulsar las tecla F1, F2 y F3 para volver al campo anterior, irnos a la petición de opción o borrar el campo respectivamente.

Una vez se hallan pedido los campos o se halla pulsado la tecla F2 se llegará a la petición de opción, en ella se nos dará opción a dar de alta, dar de baja, modificar o abandonar el registro. Para dar de alta se escribe el registro en el fichero, tras ello se borra la pantalla y se vuelve a pedir otro artículo, con lo cual estará en un bucle de petición hasta que como artículo pulsemos return, en ese momento se abandonará el programa. En las bajas lo que se hace es borrar el registro con ayuda de DELETE, tras ello se borra la pantalla y se vuelve a pedir otro artículo. Si la opción elegida ha sido modificar el cursor se posicionará en el primer campo, a partir de ahí podremos ir bajando pulsando la tecla return o subiendo con F1, cuando acabemos de modificar podremos pulsar F2 para abandonar la petición e irnos de nuevo a la petición de opción. Por último si la opción tecleada ha sido la de salir se borra la pantalla y se vuelve a pedir otro artículo, con lo cual el registro con el que se estaba trabajando ha quedado tal y como estaba.

Bajo estas líneas tienes el listado del programa.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PGS101.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.
```

```
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.

     * SE HACE UN COPY A LAS LIBRERIAS DE SELECT DE LOS FICHEROS DE
     * ARTICULOS, PROVEEDORES Y PANTALLA.

          COPY "LGSPANSE.LIB".

          COPY "LGSARTSE.LIB".

          COPY "LGSPROSE.LIB".

      DATA DIVISION.
      FILE SECTION.

     * SE HACE LO MISMO A LAS LIBRERIAS DE FD.

      COPY "LGSPANFD.LIB".

      COPY "LGSARTFD.LIB".

      COPY "LGSPROFD.LIB".

      WORKING-STORAGE SECTION.

     * ERRORES SERA EL FILE STATUS DE TODOS LOS FICHEROS.

      77  ERRORES                          PIC X(02).

     * CLV-PAN ES EL CODIGO DEL FICHERO DE PANTALLA

      77  CLV-PAN                          PIC 9(01) VALUE 1.

     * NOM-PAN ES EL CAMPO DONDE SE DEBE PONER EL NOMBRE DE LA PANTALLA
     * A LEER.

      77  NOM-PAN                          PIC X(06).

     * LA VARIABLE OPCION NOS SERVIRA PARA PEDIR LA OPCION.

      77  OPCION                           PIC X(01).

     * EN EL GRUPO EXCEP PONDREMOS LA POSIBLES EXCEPCIONES, ESTAS
     * VENDRAN CAUSADAS POR LAS TECLAS F1, F2 Y F3.

      01  EXCEP                            PIC 9(02) COMP.
          88  F1 VALUE IS 00.
```

```
           88  F2 VALUE IS 01.
           88  F3 VALUE IS 02.

       LINKAGE SECTION.

      * EN ESTA SECCION SE DECLARA EL GRUPO FECHA QUE VA A PASAR DE UN
      * PROGRAMA A OTRO.

       01  FECHA.
           03  ANO                          PIC 9(02).
           03  MES                          PIC 9(02).
           03  DIA                          PIC 9(02).

       PROCEDURE DIVISION USING FECHA.

      * EN LAS DECLARATIVES PREVENIMOS UNA POSIBLE NO EXISTENCIA DE LOS
      * FICHEROS QUE SE VAN A TRATAR.

       DECLARATIVES.

       I-O-ARTICULOS SECTION.

      * LA PRIMERA SECCION ES LA DEL FICHERO DE ARTICULOS.

           USE AFTER STANDARD ERROR PROCEDURE ON ARTICULOS.

       I-O-CREACION1.

      * SE IMPRIME EN PANTALLA UN MENSAJE CON EL NUMERO DE ERROR QUE
      * SE HA PRODUCIDO.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "    EN EL FICHERO DE ARTICULOS. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION1.

      * SE PREGUNTA SI SE QUIERE CREAR EL FICHERO.

           ACCEPT OPCION LINE 25 POSITION 66 REVERSE.

      * SI LA OPCION ES 'S' SE ABRE EL FICHERO EN MODO OUTPUT PARA
      * CREARLO, SE CIERRA Y SE VUELVE A ABRIR EN MODO I-O; TRAS ELLO
      * SE VA A FIN1.

           IF OPCION   "S" OR OPCION - "s" OPEN OUTPUT ARTICULOS,
              CLOSE ARTICULOS, OPEN I-O ARTICULOS, GO TO FIN1.
```

```
* SI LA OPCION NO HA SIDO 'S' SE COMPRUEBA SI HA SIDO 'N', DE NOR
* SER ASI SE VUELVE A PEDIR LA OPCION DE CREACION.

     IF OPCION NOT = "N" AND OPCION NOT = "n"
        GO TO OPCIONCREACION1.

 FIN1.

* CUANDO SE LLEGA AQUI ES QUE EL FICHERO SE HA CREADO O NO LO HEMOS
* QUERIDO CREAR, EN CUALQUIER CASO SE BORRA EL MENSAJE DE LA LINEA
* 25 Y SE TERMINA LA SECCION DE ARTICULOS, CON LO CUAL SE VOLVERA
* AL SIGUIENTE PUNTO POR DONDE SE FUERA EN LA PROCEDURE.

     DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

 I-O-PROVEEDOR SECTION.

* LA SEGUNDA SECCION ES LA DE PROVEEDORES.

     USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.

 I-O-CREACION2.

* LA ACTUACION ES LA MISMA, SE COMUNICA EL ERROR Y SE PREGUNTA SI
* SE QUIERE CREAR EL FICHERO, EN CASO AFIRMATIVO SE CREA Y SE SALE
* Y EN CASO NEGATIVO SE ABANDONA LA SECCION.

     DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
     DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
     DISPLAY "    EN EL FICHERO DE PROVEEDORES. LO CREO?.[ ]"
             LINE 25 POSITION 24 REVERSE.

 OPCIONCREACION2.

     ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
     IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
        CLOSE PROVEEDORES, OPEN I-O PROVEEDORES, GO TO FIN2.
     IF OPCION NOT = "N" AND OPCION NOT = "n"
        GO TO OPCIONCREACION2.

 FIN2.

     DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

 END DECLARATIVES.

* SE TERMINAN LAS DECLARATIVES Y COMIENZO EL PROGRAMA.
```

```
APLICACION DE GESTION DE SUPERMERCADOS

     PROGRAMA SECTION.

     COMIENZO.

    * SE PONE EL NOMBRE DE LA PANTALLA EN NOM-PAN.

         MOVE "SGS101" TO NOM-PAN.

    * SE ABRE, LEE Y CIERRA EL FICHERO DE PANTALLA.

         OPEN INPUT PANTALLA.
         READ PANTALLA.
         CLOSE PANTALLA.

    * SE MUEVE CERO A LOS CAMPOS STA-ART Y ACU-ART, ESTO SE HACE
    * PORQUE ESTOS CAMPOS NO SE PIDEN POR TECLADO, SINO QUE SURGEN
    * DEL MOVIMIENTO DIARIO.

         MOVE 0 TO STA-ART.
         MOVE 0 TO ACU-ART.

    * SE IMPRIME EL REGISTRO QUE CONTIENE LA PANTALLA.

         DISPLAY FIC-PAN LINE 1 POSITION 1.

    * SE IMPRIME LA FECHA QUE SE PASO DEL PRIMER PROGRAMA.

         DISPLAY DIA LINE 23 POSITION 60.
         DISPLAY "/" LINE 23 POSITION 62.
         DISPLAY MES LINE 23 POSITION 63.
         DISPLAY "/" LINE 23 POSITION 65.
         DISPLAY ANO LINE 23 POSITION 66.

     APERTURA.

    * A CONTINUACION SE ABREN LOS FICHEROS DE ARTICULOS Y PROVEEDORES
    * EN MODO I-O, SI NO EXISTEN SE IRA A LAS DECLARATIVES DONDE SE
    * NOS DARA OPCION A CREARLOS.

         OPEN I-O ARTICULOS, PROVEEDORES.

     PIDEARTICULO.

    * A CONTINUACION SE PIDE EL ARTICULO QUE SE DESEA MANTENER,
    * ESTA PETICION SE HACE CON LA OPCION ON EXCEPTION, DETRAS DE
    * ELLA HEMOS PUESTO QUE MUEVA EL CAMPO TECLA A SI MISMO, ESTO
    * PARECE ILOGICO, PERO SE HACE PARA QUE A CONTINUACION DE LA
    * PETICION SE PUEDA PREGUNTAR LA TECLA QUE SE HA PULSADO.
```

```
           ACCEPT ART-ART LINE 9 POSITION 39 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SI LA TECLA PULSADA HA SIDO F3, SE MUEVEN ESPACIOS AL ARTICULO,
       * ESTE SE IMPRIME Y SE VUELVE A PEDIR.

           IF F3 MOVE SPACES TO ART-ART, DISPLAY ART-ART LINE 9
              POSITION 39, GO TO PIDEARTICULO.

       * SE PREGUNTA SI EL ARTICULO ES IGUAL A ESPACIOS, DE SER ASI
       * SE VA A LA ETIQUETA FINPROGRAMA.

           IF ART-ART = SPACES GO TO FINPROGRAMA.

        LEERARTICULO.

       * DESPUES DE PEDIR EL ARTICULO SE COMPRUEBA SI ESTE EXISTE, DE
       * NO SER ASI SE VA A LA ETIQUETA NOEXISTE.

           READ ARTICULOS INVALID KEY GO TO NOEXISTE.

       * EN CASO CONTRARIO SE LLAMA A LA ETIQUETA IMPRIMIR Y DESPUES SE
       * VA A PEDIR OPCION.

           PERFORM IMPRIME.
           GO TO PEDIROPCION.

        NOEXISTE.

       * A PARTIR DE ESTA ETIQUETA SE PEDIRAN LOS DEMAS CAMPOS.

        PIDEPROVEEDOR.

       * SE PIDE EL PROVEEDOR QUE NOS SUMINISTRA EL ARTICULO EN CUESTION.

           ACCEPT PRO-ART LINE 11 POSITION 39 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SE MUEVE EL CAMPO PRO ART QUE ES DONDE ESTA EL PROVEEDOR A
       * PRO-PRO QUE ES LA CLAVE DEL FICHERO DE PROVEEDORES.

           MOVE PRO-ART TO PRO-PRO.

       * SI SE HA PULSADO LA TECLA F1 SE VA A PEDIR DE NUEVO EL ARTICULO.

           IF F1 GO TO PIDEARTICULO.
```

```
* SI LA TECLA PULSADA HA SIDO F2 SE VA A LA ETIQUETA PEDIROPCION.

     IF F2 GO TO PEDIROPCION.

* SI HA SIDO F3 SE MUEVEN ESPACIOS, SE IMPRIME Y SE VUELVE A PEDIR

     IF F3 MOVE SPACES TO PRO-ART, DISPLAY PRO-ART LINE 11
        POSITION 39, GO TO PIDEPROVEEDOR.

 LEEPROVEEDOR.

* TRAS PEDIR EL PROVEEDOR SE COMPRUEBA SI ESTE EXISTE, PARA ELLO
* SE LEE DEL FICHERO DE PROVEEDORES, SI DA CLAVE INVALIDA ES QUE
* NO EXISTE, SE VA A LA ETIQUETA NOEXISTEPROVEEDOR.

     READ PROVEEDORES INVALID KEY GO TO NOEXISTEPROVEEDOR.

* SI SE LLEGA A ESTE PUNTO QUIERE DECIR QUE EL PROVEEDOR EXISTE,
* SE VA A LA ETIQUETA IMPRIMEPROVEEDOR.

     GO TO IMPRIMEPROVEEDOR.

 NOEXISTEPROVEEDOR.

* AQUI SE LLEGA EN EL CASO DE QUE EL PROVEEDOR NO EXISTA, SE IMPRIME
* UN MENSAJE EN LA LINEA 25 AVISANDOLO Y SE VUELVE A PEDIR.


     DISPLAY " ESE PROVEEDOR NO EXISTE " LINE 25 POSITION 30
             REVERSE BLINK BEEP.
     GO TO PIDEPROVEEDOR.

 IMPRIMEPROVEEDOR.

* AQUI SE LLEGARA CUANDO EL PROVEEDOR QUE HEMOS FACILITADO EXISTA,
* SE IMPRIMEN ESPACIOS EN LA LINEA 25 PARA BORRAR EL MENSAJE ANTERIOR
* EN PREVISION DE QUE SE HALLA IMPRESO.

     DISPLAY SPACES SIZE 30 LINE 25 POSITION 30.

 PIDEPRECIOCOSTO.

* PIDE EL PRECIO COSTO DEL ARTICULO EN CUESTION.


     ACCEPT PRC-ART LINE 12 POSITION 39 NO BEEP ECHO UPDATE
          TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.
```

```
    * SI SE HA PULSADO LA TECLA F1 SE VUELVE A PEDIRPROVEEDOR.

        IF F1 GO TO PIDEPROVEEDOR.

    * SI HA SIDO F2 LA PULSADA SE VA A LA ETIQUETA PEDIROPCION.

        IF F2 GO TO PEDIROPCION.

    * SI SE HA PULSADO F3 SE MUEVEN CEROS AL CAMPO, SE IMPRIME
    * Y SE VUELVE A PEDIR.

        IF F3 MOVE ZEROS TO PRC-ART, DISPLAY PRC-ART LINE 12
           POSITION 39, GO TO PIDEPRECIOCOSTO.

     PIDEPRECIOVENTA.

    * SI NO SE HA PULSADO NINGUNA DE LAS TECLAS ANTERIORES SE LLEGA
    * AQUI DONDE SE PIDE EL PRECIO DE VENTA DEL ARTICULO.

        ACCEPT PRV-ART LINE 13 POSITION 39 NO BEEP ECHO UPDATE
             TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSA LA TECLA F1 SE VA A PEDIR DE NUEVO EL PRECIO COSTO.

        IF F1 GO TO PIDEPRECIOCOSTO.

    * SI SE PULSA F2 SE VA A PEDIROPCION.

        IF F2 GO TO PEDIROPCION.

    * SI LA TECLA PULSADA HA SIDO F3 SE MUEVEN ZEROS AL PRECIO VENTA,
    * SE IMPRIME Y SE VUELVE A PEDIR.

        IF F3 MOVE ZEROS TO PRV-ART, DISPLAY PRV-ART LINE 13
           POSITION 39, GO TO PIDEPRECIOVENTA.

     PIDESTOCKMINIMO.

    * EL SIGUIENTE CAMPO A PEDIR ES EL STOCK MINIMO QUE SE DESEA
    * TENER DE ESTE ARTICULO.

        ACCEPT STM-ART LINE 14 POSITION 39 NO BEEP ECHO UPDATE
             CONVERT TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSA LA TECLA F1 SE VUELVE A PEDIR EL PRECIO DE VENTA.

        IF F1 GO TO PIDEPRECIOVENTA.
```

```
    * SI HA SIDO F2 LA PULSADA SE VA A PEDIROPCION.

        IF F2 GO TO PEDIROPCION.

    * SI HA SIDO F3 SE MUEVEN CEROS AL STOCK MINIMO, SE IMPRIME Y
    * SE VUELVE A PEDIR.

        IF F3 MOVE ZEROS TO STM-ART, DISPLAY STM-ART LINE 14
           POSITION 39, GO TO PIDESTOCKMINIMO.

     PEDIROPCION.

    * AQUI SE LLEGARA AL PULSAR F2 EN LA PETICION DE ALGUNO DE LOS
    * CAMPOS O TRAS HABERLOS PEDIDO TODOS.

    * SE PIDE OPCION.

        ACCEPT OPCION LINE 20 POSITION 56 NO BEEP.

    * SI LA OPCION ES ESPACIO SE VA A LA ETIQUETA ALTAS.

        IF OPCION = SPACE GO TO ALTAS.

    * SI ES UNA B SE VA A BAJAS.

        IF OPCION = "B" OR OPCION = "b" GO TO BAJAS.

    * SI ES UNA M SE VUELVE A PEDIRPROVEEDOR, A PARTIR DE HAY
    * PODREMOS IR MODIFICANDO Y NOS PODREMOS SALIR DE LA PETICION
    * CUANDO QUERRAMOS PULSANDO F2.

        IF OPCION = "M" OR OPCION = "m" GO TO PIDEPROVEEDOR.

    * SI LA OPCION ES S SE VA A LA ETIQUETA SALIR.

        IF OPCION = "S" OR OPCION = "s" GO TO SALIR.

    * SI NO ES NINGUNA DE LAS OPCIONES ANTERIORES ES QUE LA PULSADA
    * ES INVALIDA, POR LO TANTO SE VUELVE A PEDIR.

        GO TO PEDIROPCION.

     ALTAS.

    * CUANDO SE LLEGA AQUI ES QUE LA OPCION ELEGIDA ES LA DE ALTAS.
```

```
    * GRABAMOS EL REGISTRO, SI DA CLAVE INVALIDA LO REESCRIBIMOS
    * PORQUE SERA UN REGISTRO QUE EXISTIA Y SE HA MODIFICADO.

        WRITE FIC-ART INVALID KEY REWRITE FIC-ART.

    * DESPUES SE LLAMA A LA RUTINA DE BORRADO DE PANTALLA.

        PERFORM BORRAPANTALLA.

    * Y SE VUELVE A PEDIR ARTICULO.

        GO TO PIDEARTICULO.

     BAJAS.

    * PARA DAR DE BAJA BORRAMOS EL REGISTRO, SI DA CLAVE INVALIDA
    * ES QUE NO EXISTIA, NO SE HACE NADA PARA QUE EL PROGRAMA SIGA
    * YA QUE NO SE PUEDE BORRAR LO QUE NO EXISTE.

        DELETE ARTICULOS INVALID KEY MOVE OPCION TO OPCION.

     SALIR.

    * AQUI SE LLEGA CON LA OPCION SALIR O DESPUES DE LA OPCION DAR
    * DE BAJA, LO QUE SE HACE ES BORRAR LA PANTALLA Y VOLVER A PEDIR
    * ARTICULO.

        PERFORM BORRAPANTALLA.
        GO TO PIDEARTICULO.

     FINPROGRAMA.

    * CUANDO SE LLEGA AQUI ES QUE AL PEDIR ARTICULO HEMOS PULSADO
    * RETURN SE BORRA UN POSIBLE MENSAJE QUE HUBIERA EN LA LINEA 25

        DISPLAY SPACES SIZE 79 LINE 25 POSITION 1.

     SALIRPROGRAMA.

    * Y SE VUELVE AL PROGRAMA ANTERIOR.

        EXIT PROGRAM.

     IMPRIME.

    * ESTA RUTINA IMPRIME LOS CAMPOS DEL FICHERO EN SU POSICION
    * CORRESPONDIENTE. SE UTILIZA CUANDO SE HA DADO UN ARTICULO
    * EXISTENTE.
```

```
           DISPLAY PRO-ART LINE 11 POSITION 39.
           DISPLAY PRC-ART LINE 12 POSITION 39.
           DISPLAY PRV-ART LINE 13 POSITION 39.
           DISPLAY STM-ART LINE 14 POSITION 39.
           DISPLAY STA-ART LINE 15 POSITION 39.
           DISPLAY ACU-ART LINE 16 POSITION 39.

       BORRAPANTALLA.

      * PARA BORRAR LA PANTALLA SE IMPRIMEN ESPACIOS EN CADA UNA DE LAS
      * POSICIONES DE LOS DATOS QUE HAY EN PANTALLA.

           DISPLAY SPACES SIZE 15 LINE  9 POSITION 39.
           DISPLAY SPACES SIZE 20 LINE 11 POSITION 39.
           DISPLAY SPACES SIZE  5 LINE 12 POSITION 39.
           DISPLAY SPACES SIZE  5 LINE 13 POSITION 39.
           DISPLAY SPACES SIZE  3 LINE 14 POSITION 39.
           DISPLAY SPACES SIZE  3 LINE 15 POSITION 39.
           DISPLAY SPACES SIZE 10 LINE 16 POSITION 39.

      * TRAS ELLO TAMBIEN SE BORRAN TODAS LAS VARIABLES.

           MOVE SPACES TO ART-ART, PRO-ART.
           MOVE ZEROS TO PRC-ART, PRV-ART, STM-ART, STA-ART, ACU-ART.
```

El siguiente programa que tenemos es el de mantenimiento de proveedores, es muy parecido al de mantenimiento de artículos, se usa la misma táctica para pedir campos con las mismas teclas de función y las mismas opciones, lo que cambia lógicamente son los campos a pedir y el fichero con el que se vá a tratar. A continuación tienes el listado del programa.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PGS102.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
```

```
    * SE HACE UN COPY A LAS SELECTs DE LOS FICHEROS PANTALLA Y
    * PROVEEDORES QUE SON LOS QUE SE VAN A UTILIZAR.

        COPY "LGSPANSE.LIB".

        COPY "LGSPROSE.LIB".

     DATA DIVISION.
     FILE SECTION.

    * LO MISMO CON LAS LIBRERIAS DE FDs.

     COPY "LGSPANFD.LIB".

     COPY "LGSPROFD.LIB".

     WORKING-STORAGE SECTION.

    * LA VARIABLE ERRORES SERA EL FILE STATUS DE LOS FICHEROS.

     77  ERRORES                          PIC X(02).

    * CLV-PAN ES EL CODIGO DEL FICHERO DE PANTALLA.

     77  CLV-PAN                          PIC 9(01) VALUE 1.

    * EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA A LEER.

     77  NOM-PAN                          PIC X(06).

    * LA VARIABLE OPCION NOS SERVIRA PARA PEDIR OPCION.

     77  OPCION                           PIC X(01).

    * EN EL GRUPO EXCEP SE PONEN LOS CODIGOS DE LAS TECLAS QUE
    * VAN A CAUSAR EXCEPCION.

     01  EXCEP                            PIC 9(02) COMP.
         88  F1 VALUE IS 00.
         88  F2 VALUE IS 01.
         88  F3 VALUE IS 02.

     LINKAGE SECTION.

    * EN ESTA SECCION SE DECLARA EL GRUPO FECHA QUE NOS VIENE DEL
    * PROGRAMA ANTERIOR.
```

```
01  FECHA.
    03  ANO                          PIC 9(02).
    03  MES                          PIC 9(02).
    03  DIA                          PIC 9(02).

PROCEDURE DIVISION USING FECHA.

* SE PREPARAN LAS DECLARATIVES PARA EL FICHERO DE PROVEEDORES
* EN CASO DE QUE SE PRODUCZA UN ERROR PODREMOS CREAR EL FICHERO.

DECLARATIVES.

I-O-PROVEEDOR SECTION.

    USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.

I-O-CREACION1.

    DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
    DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
    DISPLAY "    EN EL FICHERO DE PROVEEDORES. LO CREO?.[ ]"
        LINE 25 POSITION 24 REVERSE.

OPCIONCREACION1.

    ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
    IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
       CLOSE PROVEEDORES, OPEN I-O PROVEEDORES, GO TO FIN1.
    IF OPCION NOT = "N" AND OPCION NOT = "n"
       GO TO OPCIONCREACION1.

FIN1.

    DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

END DECLARATIVES.

* PRINCIPIO DEL PROGRAMA.

PROGRAMA SECTION.

COMIENZO.

* AL COMENZAR SE MUEVE LA PANTALLA A NOM-PAN SE LEE Y SE IMPRIME.

    MOVE "SGS102" TO NOM-PAN.
    OPEN INPUT PANTALLA.
```

```
        READ PANTALLA.
        CLOSE PANTALLA.
        DISPLAY FIC-PAN LINE 1 POSITION 1.

    * SE IMPRIME LA FECHA QUE HA PASADO DEL ANTERIOR PROGRAMA.

        DISPLAY DIA LINE 23 POSITION 60.
        DISPLAY "/" LINE 23 POSITION 62.
        DISPLAY MES LINE 23 POSITION 63.
        DISPLAY "/" LINE 23 POSITION 65.
        DISPLAY ANO LINE 23 POSITION 66.

     APERTURA.

    * SE HABRE EL FICHERO EN MODO I-O.

        OPEN I-O PROVEEDORES.

     PIDEPROVEEDOR.

    * SE PIDE EL PROVEEDOR.

        ACCEPT PRO-PRO LINE 9 POSITION 39 NO BEEP ECHO UPDATE
             TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE HA PULSADO F3 SE MUEVEN ESPACIOS AL PROVEEDOR, SE
    * IMPRIME Y SE VUELVE A PEDIR.

        IF F3 MOVE SPACES TO PRO-PRO, DISPLAY PRO-PRO LINE 9
           POSITION 39,  GO TO PIDEPROVEEDOR.

    * SI COMO PROVEEDOR SE HA PULSADO RETURN SE VA A LA ETIQUETA
    * FINPROGRAMA.

        IF PRO-PRO = SPACES GO TO FINPROGRAMA.

     LEERPROVEEDOR.

    * SI SE HA FACILITADO EL PROVEEDOR SE LEE PARA COMPROBAR SI EXISTE.

    * EN CASO DE QUE CLAVE INVALIDA ES QUE NO EXISTE,  SE VA A LA
    * ETIQUETA NOEXISTE A PEDIR LOS DEMAS CAMPOS.

        READ PROVEEDORES INVALID KEY GO TO NOEXISTE.

    * EN CASO DE QUE SE LLEGUE AQUI ES QUE YA EXISTIA EL PROVEEDOR
    * INDICADO SE IMPRIMEN LOS CAMPOS Y SE VA A PEDIROPCION.
```

```
            PERFORM IMPRIMIR.
            GO TO PEDIROPCION.

        NOEXISTE.

       * AQUI SE LLEGARA CUANDO EL PROVEEDOR ESPECIFICADO NO EXISTA.

        PIDEDIRECCION.

       * SE NOS PIDE LA DIRECCION.

            ACCEPT DIR-PRO LINE 11 POSITION 39 NO BEEP ECHO UPDATE
                   TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SI SE PULSA F1 SE VUELVE A PEDIR EL PROVEEDOR.

            IF F1 GO TO PIDEPROVEEDOR.

       * SI HA SIDO F2 LA PULSADA SE VA A PEDIR OPCION.

            IF F2 GO TO PEDIROPCION.

       * SI LA TECLA FUE F3 SE MUEVEN ESPACIOS A LA DIRECCION, SE
       * IMPRIME Y SE VUELVE A PEDIR.

            IF F3 MOVE SPACES TO DIR-PRO, DISPLAY DIR-PRO LINE 11
               POSITION 39, GO TO PIDEDIRECCION.

       * SI NO SE HA PULSADO NINGUNA DE LAS TRES EL PROGRAMA SEGUIRA
       * SU FLUJO PIDIENDO CAMPOS.

        PIDECODIGOPOSTAL.

       * PIDE EL CODIGO POSTAL.

            ACCEPT COD-PRO LINE 12 POSITION 39 NO BEEP ECHO UPDATE
                   TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SI SE PULSA F1 SE VUELVE A PEDIR LA DIRECCION.

            IF F1 GO TO PIDEDIRECCION.

       * SI SE PULSO F2 SE VA A PEDIR OPCION.

            IF F2 GO TO PEDIROPCION.

       * SI FUE F3 LA TECLA SE MUEVEN CEROS AL CODIGO, SE IMPRIME Y SE
       * VUELVE A PEDIR.
```

```
           IF F3 MOVE ZEROS TO COD-PRO, DISPLAY COD-PRO LINE 12
              POSITION 39, GO TO  PIDECODIGOPOSTAL.

       * SI NO SE PULSO NINGUNA SIGUE POR AQUI.

        PIDEPOBLACION.

       * SE PIDE LA POBLACION.

           ACCEPT POB-PRO LINE 13 POSITION 39 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SI SE PULSA F1 SE VUELVE A PEDIR EL CODIGO POSTAL.

           IF F1 GO TO PIDECODIGOPOSTAL.

       * SI HA SIDO F2 SE VA A PEDIR OPCION.

           IF F2 GO TO PEDIROPCION.

       * SI LA TECLA FUE F3 SE MUEVEN ESPACIOS A LA POBLACION, SE
       * IMPRIME Y SE VUELVE A PEDIR.

           IF F3 MOVE SPACES TO POB-PRO, DISPLAY POB-PRO LINE 13
              POSITION 39, GO TO PIDEPOBLACION.

       * SI NO SE PULSO NINGUNA TECLA EL PROGRAMA SEGUIRA POR AQUI.

        PIDEPROVINCIA.

       * SE PIDE LA PROVINCIA.

           ACCEPT PRV-PRO LINE 14 POSITION 39 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SI SE PULSO F1 SE VUELVE A PEDIR LA POBLACION.

           IF F1 GO TO PIDEPOBLACION.

       * SI FUE F2 SE VA A PEDIR OPCION.

           IF F2 GO TO PEDIROPCION.

       * SI FUE F3 MUEVE ESPACIOS A LA PROVINCIA, LA IMPRIME Y LA
       * VUELVE A PEDIR.

           IF F3 MOVE SPACES TO PRV-PRO, DISPLAY PRV-PRO LINE 14
              POSITION 39, GO TO PIDEPROVINCIA.
```

```
    * SI NO SE PULSO NINGUNA DE LAS TECLAS SE SIGUE POR AQUI.

     PIDETELEFONO.

    * SE PIDE EL TELEFONO.

         ACCEPT TEL-PRO LINE 15 POSITION 39 NO BEEP ECHO UPDATE
                TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSO F1 SE VUELVE A PEDIR LA PROVINCIA.

         IF F1 GO TO PIDEPROVINCIA.

    * SI LA TECLA FUE F2 SE VA A PEDIR OPCION.

         IF F2 GO TO PEDIROPCION.

    * SI LA TECLA PULSADA FUE F3 SE MUEVEN ESPACIOS AL TELEFONO, SE
    * IMPRIME Y SE VUELVE A PEDIR.

         IF F3 MOVE SPACES TO TEL-PRO, DISPLAY TEL-PRO LINE 15
            POSITION 39, GO TO PIDETELEFONO.

    * SI NO SE PULSO NINGUNA EL PROGRAMA CONTINUA.

     PIDEDNIOCIF.

    * PIDE EL D.N.I.

         ACCEPT DNI-PRO LINE 16 POSITION 39 NO BEEP ECHO UPDATE
                TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSA LA TECLA F1 SE VUELVE A PEDIR EL TELEFONO.

         IF F1 GO TO PIDETELEFONO.

    * SI LA QUE SE PULSA ES F2 SE VA A PEDIR OPCION.

         IF F2 GO TO PEDIROPCION.

    * SI HA SIDO F3 SE MUEVEN ESPACIOS AL D.N.I., SE IMPRIME Y SE
    * VUELVE A PEDIR.

         IF F3 MOVE SPACES TO DNI-PRO, DISPLAY DNI-PRO LINE 16
            POSITION 39, GO TO PIDEDNIOCIF.

     PEDIROPCION.
```

```
    * AQUIE SE LLEGA TRAS HABER PEDIDO LOS CAMPOS O HABER PULSADO F2.

    * SE PIDE OPCION.

        ACCEPT OPCION LINE 20 POSITION 66 NO BEEP.

    * SI SE PULSO RETURN SE VA A ALTAS.

        IF OPCION = SPACE GO TO ALTAS.

    * SI SE PULSO B SE VA A BAJAS.

        IF OPCION = "B" OR OPCION = "b" GO TO BAJAS.

    * SI SE PULSO M SE VA A PEDIR DIRECCION, A PARTIR DE AHI PODREMOS
    * MODIFICAR EL CAMPO QUE DESEEMOS.

        IF OPCION = "M" OR OPCION = "m" GO TO PIDEDIRECCION.

    * SI FUE S LA OPCION PULSADA SE VA A SALIR.

        IF OPCION = "S" OR OPCION = "s" GO TO SALIR.

    * SI SE LLEGA AQUI QUIERE DECIR QUE LA OPCION PULSADA ES INVALIDA
    * POR LO TANTO SE VUELVE A PEDIR.

        GO TO PEDIROPCION.


     ALTAS.

    * AQUI SE LLEGA CUANDO LA OPCION ELEGIDA HA SIDO LA DE ALTAS.

    * SE ESCRIBE EL REGISTRO, SI DA CLAVE INVALIDA ES PORQUE YA EXISTIA
    * EN ESE CASO SE REESCRIBE.

        WRITE FIC-PRO INVALID KEY REWRITE FIC-PRO.

    * DESPUES SE BORRA LA PANTALLA Y SE VUELVE A PEDIR OTRO PROVEEDOR.

        PERFORM BORRAPANTALLA.
        GO TO PIDEPROVEEDOR.

     BAJAS.

    * PARA DAR DE BAJA SE BORRA EL REGISTRO DEL FICHERO, SI DA CLAVE
    * INVALIDA ES QUE NO EXISTIA, EN ESE CASO NO SE HACE NADA PORQUE
    * YA ESTA DADO DE BAJA.
```

```
        DELETE PROVEEDORES INVALID KEY MOVE OPCION TO OPCION.

    SALIR.

    * AQUI SE LLEGARA CON LA OPCION SALIR O DESPUES DE DAR DE BAJA, LO
    * QUE SE HACE ES BORRAR LA PANTALLA Y VOLVER A PEDIR OTRO PROVEEDOR.

        PERFORM BORRAPANTALLA.
        GO TO PIDEPROVEEDOR.

    FINPROGRAMA.

    * A ESTE PUNTO SE LLEGA CUANDO AL PEDIR EL PROVEEDOR SE PULSO RETURN
    * O TAB. SE BORRA UN POSIBLE MENSAJE QUE HUBIER EN LA LINEA 25.

        DISPLAY SPACES SIZE 79 LINE 25 POSITION 1.

    SALIRPROGRAMA.

    * Y SE SALE DEL PROGRAMA.

        EXIT PROGRAM.

    IMPRIMIR.

    * ESTA ES LA RUTINA DE IMPRESION DE CAMPOS, SE IMPRIME CADA
    * CAMPO EN SU POSICION CORRESPONDIENTE.

        DISPLAY DIR-PRO LINE 11 POSITION 39.
        DISPLAY COD-PRO LINE 12 POSITION 39.
        DISPLAY POB-PRO LINE 13 POSITION 39.
        DISPLAY PRV-PRO LINE 14 POSITION 39.
        DISPLAY TEL-PRO LINE 15 POSITION 39.
        DISPLAY DNI-PRO LINE 16 POSITION 39.
        DISPLAY ACU-PRO LINE 17 POSITION 39.

    BORRAPANTALLA.

    * ESTA RUTINA SE LLAMA CUANDO SE QUIERE BORRAR PANTALLA, LO
    * QUE HACE ES IMPRIMIR ESPACIOS EN LA POSICION DE LOS CAMPOS

        DISPLAY SPACES SIZE 20 LINE  9 POSITION 39.
        DISPLAY SPACES SIZE 20 LINE 11 POSITION 39.
        DISPLAY SPACES SIZE  5 LINE 12 POSITION 39.
        DISPLAY SPACES SIZE 20 LINE 13 POSITION 39.
        DISPLAY SPACES SIZE 20 LINE 14 POSITION 39.
        DISPLAY SPACES SIZE 13 LINE 15 POSITION 39.
```

```
           DISPLAY SPACES SIZE 15 LINE 16 POSITION 39.
           DISPLAY SPACES SIZE 10 LINE 17 POSITION 39.

       * Y BORRAR TAMBIEN LAS VARIABLES QUE SE UTILIZAN.

           MOVE SPACES TO PRO-PRO, DIR-PRO, POB-PRO,
                          PRV-PRO, TEL-PRO, DNI-PRO.
           MOVE ZEROS TO COD-PRO, ACU-PRO.
```

El siguiente programa que tenemos es el de listado de artículos, en él primero se nos pedirá desde que artículo queremos listar, y a continuación hasta que artículo queremos listar; si queremos que el listado se haga desde el principio hasta el final del fichero pulsaremos la tecla TAB en las dos peticiones. Seguidamente tenemos un breve seguimiento del programa.

Además de las variables normales que vamos teniendo en todos los programas en el listado tenemos algunas más, entre ellas tres cabeceras, estas nos van a servir para imprimir al principio de página los encabezamientos tales como fecha y campos que van a intervenir en el listado; tenemos un grupo al que hemos llamado CUERPO, este nos vá a servir para ir imprimiendo los distintos registros que vayamos leyendo, se ha dividido en forma tal que coincidan los campos con la cabecera; a continuacion tenemos un grupo al que hemos llamado PIE, este se imprimirá cada vez que se llegue al final de página. Para controlar cuando se llega al final de página y las páginas que se llevan impresas tenemos dos campos a los que hemos llamado LINEA y PAGINA.

Lo primero que tenemos en la PROCEDURE son las DECLARATIVES, en ellas prevenimos errores tanto de los ficheros de ARTICULOS y PROVEEDORES como de la impresora. Ya en el programa propiamente dicho lo primero que hacemos es sacar la pantalla e imprimir la fecha, esta fecha se pasa también a los campos de fecha de una de las cabeceras para imprimirla en cada principio de página; seguidamente se abren los ficheros de ARTICULOS y PROVEEDORES, se pide el primer artículo a listar y el último artículo listar; esta petición se hace forma análoga a los mantenmientos y podemos usar las teclas de función que usábamos en ellos; tras tener el principio y fin del listado se nos pide opción para saber si es correcto lo que hemos tecleado, es incorrecto o queremos abandonar el listado; si hemos tecleado incorrecto se nos volverán a pedir el principio y fin de listado, si hemos tecleado correcto se iniciará el listado y si la opción ha sido salir se abandonará el programa.

Al comenzar el listado lo primero que se hace es un START para posicionarnos en el registro se ha indicado que ha de ser el primero, si no se encuentra se nos comunica que no hay ningún registro a partir del indicado y se abandonará el listado, si se encuentra se imprimirán las cabeceras y se actualizarán las variables LINEA y PAGINA, a partir de ahí se irá leyendo en secuencial hasta que se llegue al final de fichero o el registro que se lea sea mayor que el indicado como final, tras leer se pasarán los campos del registro a los campos correspondientes del cuerpo de impresión, se imprimirá, se actualizará el campo LINEA y se controlará que si este ha llegado al límite de página se imprima el pie y se salte a la próxima página. Cuando se termine el listado se saltará hasta el principio de la próxima página, se cerrarán los ficheros y se volverá al programa menú.

Seguidamente tienes el listado del programa.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PGS101.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

      * SE HACE UN COPY A LAS LIBRERIAS DE SELECTs DE LOS FICHEROS DE
      * PANTALLA, ARTICULOS Y PROVEEDORES.

           COPY "LGSPANSE.LIB".

           COPY "LGSARTSE.LIB".

           COPY "LGSPROSE.LIB".

      * SE DECLARA EL FICHERO DE IMPRESORA.

           SELECT IMP;
                  ASSIGN TO PRINT, "PRINTER";
                  FILE STATUS IS ERRORES.
```

```
     DATA DIVISION.
     FILE SECTION.

    * SE HACE UN COPY A LAS LIBRERIAS DE LAS FDs DE LOS FICHEROS.

     COPY "LGSPANFD.LIB".

     COPY "LGSARTFD.LIB".

     COPY "LGSPROFD.LIB".

    * SE DECLARA EL REGISTRO DEL FICHERO DE IMPRESORA.

     FD  IMP;
         LABEL RECORD IS OMITTED;
         DATA RECORD IS FIC-IMP.

     01  FIC-IMP                              PIC X(80).

     WORKING-STORAGE SECTION.

    * A CONTINUACION SE DECLARAN LAS DISTINTAS CABECERAS QUE SE VAN
    * A UTILIZAR EN EL LISTADO ASI COMO EL CUERPO Y EL PIE DE PAGINA.

     01  CABECERA1.
         03  FILLER                           PIC X(71) VALUE "  CENTRO DE
         " ESTUDIOS INFORMATICOS   GESTION DE SUPERMERCADOS     FECHA:
         "".
         03  DIA-OUT                          PIC ZZ.
         03  FILLER                           PIC X(01) VALUE "/".
         03  MES-OUT                          PIC ZZ.
         03  FILLER                           PIC X(01) VALUE "/".
         03  ANO-OUT                          PIC ZZ.

     01  CABECERA2                            PIC X(80) VALUE ALL "-".

     01  CABECERA3.
         03  FILLER                           PIC X(02) VALUE SPACES.
         03  FILLER                           PIC X(15) VALUE "ARTICULO".
         03  FILLER                           PIC X(02) VALUE SPACES.
         03  FILLER                           PIC X(20) VALUE "PROVEEDOR".
         03  FILLER                           PIC X(02) VALUE SPACES.
         03  FILLER                           PIC X(06) VALUE "P.COS.".
         03  FILLER                           PIC X(02) VALUE SPACES.
         03  FILLER                           PIC X(06) VALUE "P.VEN.".
         03  FILLER                           PIC X(02) VALUE SPACES.
         03  FILLER                           PIC X(05) VALUE "ST.A.".
```

```
        03  FILLER                          PIC X(03) VALUE SPACES.
        03  FILLER                          PIC X(10) VALUE "ACUMULADO".

    01  CUERPO.
        03  FILLER                          PIC X(02) VALUE SPACES.
        03  ART-OUT                         PIC X(15).
        03  FILLER                          PIC X(02) VALUE SPACES.
        03  PRO-OUT                         PIC X(20).
        03  FILLER                          PIC X(03) VALUE SPACES.
        03  PRC-OUT                         PIC ZZZZ9.
        03  FILLER                          PIC X(03) VALUE SPACES.
        03  PRV-OUT                         PIC ZZZZ9.
        03  FILLER                          PIC X(03) VALUE SPACES.
        03  STA-OUT                         PIC ZZ9.
        03  FILLER                          PIC X(04) VALUE SPACES.
        03  ACU-OUT                         PIC ZZZZZZZZZ9.

    01  PIE.
        03  FILLER                          PIC X(50) VALUE "LISTADO DE
-       "ARTICULOS.".
        03  FILLER                          PIC X(07) VALUE "PAGINA:".
        03  PAG-OUT                         PIC ZZ.

    * LA VARIABLE PAGINA NOS SERVIRA PARA IR CONTROLANDO LAS PAGINAS
    * DE IMPRESION QUE SE LLEVAN.

     77  PAGINA                             PIC 9(02).

    * LA VARIABLE LINEA NOS VA A SERVIR PARA CONTROLAR LA LONGITUD DE
    * PAGINA Y SALTAR A OTRA EN EL MOMENTO PRECISO.

     77  LINEA                              PIC 9(02).

    * LAS VARIABLES PRI-ART Y ULT-ART NOS SERVIRAN PARA CONTROLAR DESDE
    * DONDE HASTA DONDE SE HA DE EFECTUAR EL LISTADO.

     77  PRI-ART                            PIC X(15).
     77  ULT-ART                            PIC X(15).

    * LA VARIABLE ERRORES SERA EL FILE STATUS DE LOS FICHEROS QUE SE
    * UTILIZAN.

     77  ERRORES                            PIC X(02).

    * CLV-PAN ES LA CLAVE DEL FICHERO DE PANTALLA.

     77  CLV-PAN                            PIC 9(01) VALUE 1.
```

```
    * EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA A LEER DEL FICHERO
    * DE PANTALLA.

     77  NOM-PAN                              PIC X(06).

    * OPCION NOS SERVIRA PARA PEDIR OPCION

     77  OPCION                               PIC X(01).

    * EN EL GRUPO EXCEP SE DECLARAN LAS POSIBLES EXCEPCIONES.

     01  EXCEP                                PIC 9(02) COMP.
         88  F1 VALUE IS 00.
         88  F2 VALUE IS 01.
         88  F3 VALUE IS 02.

     LINKAGE SECTION.

    * EN ESTA SECCION DECLARAMOS EL GRUPO FECHA QUE NOS VIENE DEL
    * PROGRAMA ANTERIOR.

     01  FECHA.
         03  ANO                              PIC 9(02).
         03  MES                              PIC 9(02).
         03  DIA                              PIC 9(02).

     PROCEDURE DIVISION USING FECHA.

    * SE PREPARAN LAS DECLARATIVES EN PREVISION DE POSIBLES ERRORES
    * EN LOS FICHEROS.

     DECLARATIVES.

     I-O-ARTICULOS SECTION.

    * LA PRIMERA SECCION SERA LA DEL FICHERO DE ARTICULOS.

         USE AFTER STANDARD ERROR PROCEDURE ON ARTICULOS.

     I-O-CREACION1.

    * SE IMPRIME EL ERROR QUE SE PRODUJO Y SE DA OPCION A CREAR EL
    * FICHERO.

         DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
         DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
         DISPLAY "    EN EL FICHERO DE ARTICULOS. LO CREO?.[ ]"
                 LINE 25 POSITION 24 REVERSE.
```

```
     OPCIONCREACION1.

         ACCEPT OPCION LINE 25 POSITION 66 REVERSE.

     * SI LA OPCION ES S SE CREA EL FICHERO.

         IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT ARTICULOS,
            CLOSE ARTICULOS, OPEN I-O ARTICULOS, GO TO FIN1.

     * SI NO ES S NI N SE VUELVE A PEDIR.

         IF OPCION NOT = "N" AND OPCION NOT = "n"
            GO TO OPCIONCREACION1.

     FIN1.

     * TRAS HABER CREADO EL FICHERO O SI HEMOS TECLEADO EN LA OPCION N
     * SE LLEGARA AQUI, SE BORRA EL MENSAJE DE LA LINEA 25 Y SE TERMINA
     * LA SECCION CORRESPONDIENTE A ARTICULOS.

         DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

     I-O-PROVEEDOR SECTION.

     * LA SIGUIENTE SECCION ES LA DE PROVEEDORES.

         USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.

     I-O-CREACION2.

     * ESTA FUNCIONES DE IGUAL FORMA QUE LA ANTERIOR.

         DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1, REVERSE.
         DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
         DISPLAY "    EN EL FICHERO DE PROVEEDORES. LO CREO?. [ ]"
                 LINE 25 POSITION 24 REVERSE.

     OPCIONCREACION2.

         ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
         IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
            CLOSE PROVEEDORES, OPEN I-O PROVEEDORES, GO TO FIN2.
         IF OPCION NOT = "N" AND OPCION NOT = "n"
            GO TO OPCIONCREACION2.

     FIN2.

         DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.
```

```
      I-O-IMP SECTION.

    * POR ULTIMO SE DECLARA LA SECCION PARA IMPRESORA.

          USE AFTER STANDARD ERROR PROCEDURE ON IMP.

      I-O-AVISO.

    * CUANDO SE PRODUCE UN ERROR EN EL FICHERO DE IMPRESORA LO MAS
    * POSIBLE ES QUE LA IMPRESORA NO ESTE CONECTADA, SE AVISA DE
    * QUE SE CONECTE Y SE PULSE UNA TECLA, TRAS LO CUAL SE ESCRIBE EL
    * REGISTRO QUE EN ESE MOMENTO HALLA Y SE TERMINA LA SECCION DE
    * IMPRESORA Y CON ELLA LAS DECLARATIVES.

          DISPLAY " CONECTE LA IMPRESORA Y PULSE UNA TECLA "
                  LINE 25 POSITION 15 REVERSE BLINK BEEP.
          ACCEPT OPCION OFF NO BEEP.

          WRITE FIC-IMP.
          DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.

      END DECLARATIVES.

    * COMIENZA EL PROGRAMA.

      PROGRAMA SECTION.

      COMIENZO.

    * SE MUEVE EL NOMBRE DE LA PANTALLA A LEER A NOM-PAN, SE ABRE, LEE
    * Y CIERRA EL FICHERO, TRAS ELLO IMPRIME EL REGISTRO DE PANTALLA.

          MOVE "SGS201" TO NOM-PAN.
          OPEN INPUT PANTALLA.
          READ PANTALLA.
          CLOSE PANTALLA.
          DISPLAY FIC-PAN LINE 1 POSITION 1.

    * SE IMPRIME LA FECHA QUE NOS VIENE DADA POR EL PROGRAMA ANTERIOR.

          DISPLAY DIA LINE 23 POSITION 60.
          DISPLAY "/" LINE 23 POSITION 62.
          DISPLAY MES LINE 23 POSITION 63.
          DISPLAY "/" LINE 23 POSITION 65.
          DISPLAY ANO LINE 23 POSITION 66.

    * SE MUEVE LA FECHA QUE NOS VIENE DEL PROGRAMA MENU A LA FECHA DE
    * LA CABECERA DE IMPRESION.
```

```
           MOVE DIA TO DIA-OUT.
           MOVE MES TO MES-OUT.
           MOVE ANO TO ANO-OUT.

       APERTURA.

      * SE ABREN LOS FICHEROS DE ARTICULOS Y PROVEEDORES.

           OPEN I-O ARTICULOS, PROVEEDORES.

       PIDEPRIMERO.

      * SE PIDE EL PRIMER ARTICULO A LISTAR Y SE PONE EN PRI-ART.

           ACCEPT PRI-ART LINE 10 POSITION 38 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

      * SI SE PULSA F2 SE VA A PEDIR OPCION

           IF F2 GO TO PEDIROPCION.

      * SI SE PULSA F3 SE MUEVEN SPACIOS A PRI-ART, SE IMPRIME Y SE
      * VUELVE A PEDIR.

           IF F3 MOVE SPACES TO PRI-ART, DISPLAY PRI-ART LINE 18
                 POSITION 38, GO TO PIDEPRIMERO.

       PIDEULTIMO.

      * A COTINUACION SE PIDE EL ULTIMO ARTICULO A LISTAR.

           ACCEPT ULT-ART LINE 12 POSITION 38 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

      * SI ULT-ART CONTIENE ESPACIOS QUIERE DECIR QUE SE HA PULSADO LA
      * TECLA TAB O POSIBLEMENTE RETURN, EN ESE CASO SE LLENA EL CAMPO
      * DE ZETAS PARA QUE LISTE HASTA EL FINAL DEL FICHERO.

           IF ULT-ART = SPACES MOVE "ZZZZZZZZZZZZZZZ" TO ULT-ART,
              DISPLAY ULT-ART LINE 12 POSITION 38.

      * SI SE HA PULSADO F1 SE VUELVE A PEDIR EL PRIMER ARTICULO
      * A LISTAR.

           IF F1 GO TO PIDEPRIMERO.

       * SI SE HA PULSADO F3 SE MUEVEN ESPACIOS A ULT-ART, SE IMPRIME
       * Y SE VUELVE A PEDIR.
```

```
           IF F3 MOVE SPACES TO ULT-ART, DISPLAY ULT-ART LINE 12
              POSITION 38, GO TO PIDEULTIMO.

       PEDIROPCION.

      * SE PIDE OPCION TRAS TENER EL COMIENZO Y FIN DEL LISTADO.

           ACCEPT OPCION LINE 17 POSITION 49 NO BEEP.

      * SI LA OPCION ES C ES CORRECTO, SE VA A LA ETIQUETA LISTAR.

           IF OPCION = "C" OR OPCION = "c" GO TO LISTAR.

      * SI LA OPCION ES I ES INCORRECTO, SE VUELVEN A PEDIR LOS LIMITES,
      * DEL LISTADO.

           IF OPCION = "I" OR OPCION = "i" GO TO PIDEPRIMERO.

      * SI LA OPCION NO ES S SE VUELVE A PEDIR OPCION YA QUE ES INVALIDA.

           IF OPCION NOT = "S" AND OPCION NOT = "s" GO TO PEDIROPCION.

       FINPROGRAMA.

      * SI SE LLEGA AQUI QUIERE DECIR QUE LA TECLA PULSADA FUE S, SE
      * CIERRAN LOS FICHEROS DE ARTICULOS Y PROVEEDORES.

           CLOSE ARTICULOS, PROVEEDORES.

       SALIRPROGRAMA.

      * TRAS LA CUAL SE VUELVE AL PROGRAMA ANTERIOR.

           EXIT PROGRAM.

       LISTAR.

      * AQUI SE LLEGARA CUANDO AL PEDIR OPCION SE HALLA PULSADO C.


      * SE MUEVE PRI-ART QUE TIENE EL PRIMER ARTICULO A LISTAR A ART-ART.

           MOVE PRI-ART TO ART-ART.

      * SE HACE UN START PARA POSICIONARNOS EN EL LUGAR INDICADO POR
      * ART-ART O EN EL SIGUIENTE REGISTRO A ESTE. SI DA CLAVE INVALIDA
```

```
    * ES QUE NO HAY NINGUN ARTICULO A PARTIR DEL ESPECIFICADO, EN ESE
    * CASO SE VA A LA ETIQUETA NOHAY.

        START ARTICULOS KEY IS NOT < ART-ART INVALID KEY GO TO NOHAY.

    * SI SE LLEHA AQUI QUIERE DECIR QUE HAY ALGUN REGISTRO A LISTAR.
    * SE ABRE EL FICHERO DE IMPRESORA, SE MUEVE 0 A PAGINA Y LINEA Y
    * SE LLAMA A LA ETIQUETA CABECERA PARA QUE IMPRIMA LA MISMA, ES DE
    * SUPONER QUE EN ESE MOMENTO TENDREMOS LA IMPRESORA CONECTADA Y A
    * PRINCIPIO DE PAGINA, EN CASO CONTRARIO EL PROGRAMA NOS COMUNICARA
    * QUE LA CONECTEMOS Y PULSEMOS UNA TECLA CUANDO ESTE PREPARA, ESTO
    * SE HARA GRACIAS A LAS DECLARATIVES.

        OPEN OUTPUT IMP.
        MOVE 0 TO LINEA, PAGINA.
        PERFORM CABECERA.

     LEER.

    * TRAS HABER IMPRESO LA CABECERA SE COMIENZA A LEER EN SECUENCIAL
    * EN EL FICHERO DE ARTICULOS, SI EN ALGUN MOMENTO SE LLEGA AL FIN
    * DE FICHERO SE VA A LA ETIQUETA FINLISTADO.

        READ ARTICULOS NEXT RECORD AT END GO TO FINLISTADO.

    * SI EL ARTICULO QUE SE ACABA DE LEER ES MAYOR QUE EL ESPECIFICADO
    * COMO ULTIMO A LISTAR SE BIFURCA A LA ETIQUETA FINLISTADO.

        IF ART-ART > ULT-ART GO TO FINLISTADO.

    * EN CASO CONTRARIO SE MUEVEN LOS CAMPOS DEL FICHERO DE ARTICULOS
    * A LOS DEL CUERPO DECLARADO PARA LA IMPRESION.

        MOVE ART-ART TO ART-OUT.
        MOVE PRO-ART TO PRO-OUT.
        MOVE PRC-ART TO PRC-OUT.
        MOVE PRV-ART TO PRV-OUT.
        MOVE STA-ART TO STA-OUT.
        MOVE ACU-ART TO ACU-OUT.

    * TRAS ELLO SE IMPRIME EL REGISTRO DE IMPRESORA CON EL CUERPO.

        WRITE FIC-IMP FROM CUERPO.

    * SE IMPRIME LA LINEA DE GUIONES QUE ESTA EN CABECERA2 PARA
    * SEPARAR UN ARTICULO DE OTRO.

        WRITE FIC-IMP FROM CABECERA2.
```

```
    * TRAS ESTO SE LE SUMA 2 A LINEA.

        ADD 2 TO LINEA.

    * SE PREGUNTA SI LA LINEA YA ES MAYOR DE 57, DE SER ASI SE LLAMA
    * A PIE Y CABECERA PARA VOLVER A COMENZAR EN LA PAGINA SIGUIENTE.

        IF LINEA > 57 PERFORM PIE, PERFORM CABECERA.
        GO TO LEER.

     CABECERA.

    * CUANDO SE LAMA A ESTA ETIQUETA SE IMPRIME LA CABECERA1 CON LA
    * FECHA Y UN TEXTO, A CONTINUACION SE IMPRIME LA CABECERA2 QUE ES
    * UNA LINEA DE GUIONES Y POR ULTIMO SE IMPRIME LA CABECERA3 QUE
    * TIENE LOS DISTINTOS CAMPOS QUE TIENE EL LISTADO.

        WRITE FIC-IMP FROM CABECERA1.
        WRITE FIC-IMP FROM CABECERA2.
        WRITE FIC-IMP FROM CABECERA3.

    * SE LLAMA A LA RUTINA LINBLAN PARA IMPRIMIR UNA LINEA EN BLANCO.

        PERFORM LINBLAN.

    * SE MUEVE 4 A LA LINEA PARA COMENZAR UNA NUEVA PAGINA.

        MOVE 4 TO LINEA.

     PIE.

    * CUANDO SE LLAMA A ESTA RUTINA PRIMERO SE SALTA HASTA EL FINAL DE
    * PAGINA, PARA ELLO A 64 SE LE RESTA LA LINEA EN LA QUE SE ESTA
    * ACTUALMENTE, CON EL RESULTADO SE LLAMA A LA RUTINA DE IMPRIMIR
    * LINEAS EN BLANCO, DE TAL FORMA QUE SE LLEGUE AL FINAL DE PAGINA.

        COMPUTE LINEA = 64 - LINEA - 2.
        PERFORM LINBLAN LINEA TIMES.

    * TRAS ELLO SE LE SUMA UNO A PAGINA, SE MUEVE LA PAGINA A PAG-OUT.

        ADD 1 TO PAGINA.
        MOVE PAGINA TO PAG-OUT.

    * SE IMPRIME EL PIE DE PAGINA, EN ESTE ESTA EL NUMERO DE PAGINA QUE
    * SE ACABA DE IMPRIMIR Y UN TEXTO EXPLICATIVO DEL LISTADO QUE SE
    * ESTA HACIENDO.
```

```
           WRITE FIC IMP FROM PIE.

       * TRAS ESTO SE LLAMA 3 VECES A LA RUTINA DE LINEA EN BLANCO PARA
       * PASAR A LA SIGUIENTE PAGINA.

           PERFORM LINBLAN 3 TIMES.

        FINLISTADO.

       * AL LLEGAR AL FIN DE LISTADO SE LLAMA A LA RUTINA DE IMPRIMIR
       * PIE PARA SALTAR LA ULTIMA PAGINA, TRAS ELLO SE CIERRA EL FICHERO
       * DE IMPRESORA, UTILIZAMOS LA OPCION WITH NO REWIND PARA QUE NO
       * SE SALTE OTRA PAGINA.


           PERFORM PIE.
           CLOSE IMP WITH NO REWIND.

       * TRAS ELLO SE VA A LA ETIQUETA FINPROGRAMA.

           GO TO FINPROGRAMA.

        NOHAY.

       * SI AL HACER EL START PARA COMENZAR EL LISTADO DA CLAVE INVALIDA
       * SE LLEGABA A ESTA ETIQUETA, EN ELLA AVISAMOS DE QUE NO HAY
       * NINGUN REGISTRO A PARTIR DEL INDICADO.


           DISPLAY " NO HAY NINGUN REGISTRO A PARTIR DEL INDICADO "
                   LINE 25 POSITION 15 REVERSE BLINK BEEP.

       * SE LLAMA 5000 VECES A LA RUTINA DE ESPERA, ESTO HARA QUE EL
       * MENSAJE PERMANEZCA DURANTE UN MOMENTO EN PANTALLA.

           PERFORM ESPERA 5000 TIMES.

       * SE BORRA EL MENSAJE Y SE VA A LA ETIQUETA FINPROGRAMA.

           DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.
           GO TO FINPROGRAMA.

        LINBLAN.

       * ESTA RUTINA ES LLAMADA PARA IMPRIMIR LINEAS EN BLANCO, LO QUE
       * SE HACE ES MOVER ESPACIOS AL REGISTRO DE IMPRESORA Y SE IMPRIME.
```

```
           MOVE SPACES TO FIC IMP.
           WRITE FIC IMP.

         ESPERA.

       * EN LA ETIQUETA ESPERA LO QUE SE HACE ES MOVER EL CAMPO OPCION
       * A SI MISMO CON LO CUAL SE PIERDE UN TIEMPO.

           MOVE OPCION TO OPCION.
```

El siguiente programa será el de listado de proveedores, este es muy parecido al anterior, la única diferencia es que el número de caracteres ocupado por los campos no tenía espacio en una sola línea y por lo tanto se ha hecho a dos líneas, por lo demás es practicamente igual.

```
123456789012
       A   B

         IDENTIFICATION DIVISION.
         PROGRAM-ID. PGS202.

         ENVIRONMENT DIVISION.
         CONFIGURATION SECTION.
         SOURCE-COMPUTER. PC.
         OBJECT-COMPUTER. PC.

         INPUT-OUTPUT SECTION.
         FILE-CONTROL.

       * SE HACE UN COPY A LAS LIBRERIAS DE SELECTs DE LOS FICHERO DE
       * PROVEEDORES Y PANTALLA.

           COPY "LGSPANSE.LIB".

           COPY "LGSPROSE.LIB".

       * SE DECLARA EL FICHERO DE IMPRESORA.

           SELECT IMP;
                  ASSIGN TO PRINT, "PRINTER";
                  FILE STATUS IS ERRORES.
```

APLICACION DE GESTION DE SUPERMERCADOS

```
       DATA DIVISION.
       FILE SECTION.

      * SE HACE UN COPY A LAS LIBRERIAS FDs DE LOS MISMOS FICHEROS.

       COPY "LGSPANFD.LIB".

       COPY "LGSPROFD.LIB".

      * SE DECLARA EL REGISTRO DE IMPRESORA

       FD  IMP;
           LABEL RECORD IS OMITTED;
           DATA RECORD IS FIC IMP.

       01  FIC-IMP                              PIC X(80).

       WORKING STORAGE SECTION.

      * SE DECLARAN LAS CABECERAS QUE SE VAN A USAR EN EL LISTADO.

       01  CABECERA1.
           03  FILLER                           PIC X(71) VALUE "  CENTRO DE
           " ESTUDIOS INFORMATICOS   GESTION DE SUPERMERCADOS      FECHA:
           "".
           03  DIA-OUT                          PIC ZZ.
           03  FILLER                           PIC X(01) VALUE "/".
           03  MES-OUT                          PIC ZZ.
           03  FILLER                           PIC X(01) VALUE "/".
           03  ANO-OUT                          PIC ZZ.

       01  CABECERA2                            PIC X(80) VALUE ALL "-".

       01  CABECERA3.
           03  FILLER                           PIC X(02) VALUE SPACES.
           03  FILLER                           PIC X(20) VALUE "PROVEEDOR".
           03  FILLER                           PIC X(02) VALUE SPACES.
           03  FILLER                           PIC X(13) VALUE "TELEFONO".
           03  FILLER                           PIC X(02) VALUE SPACES.
           03  FILLER                           PIC X(20) VALUE "POBLACION".
           03  FILLER                           PIC X(02) VALUE SPACES.
           03  FILLER                           PIC X(15) VALUE "D.N.I. o C.
               "I.F.".

       01  CABECERA4.
           03  FILLER                           PIC X(02) VALUE SPACES.
           03  FILLER                           PIC X(20) VALUE "DIRECCION".
```

```
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  FILLER                          PIC X(13) VALUE "CODIGO POST
               "AL".
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  FILLER                          PIC X(20) VALUE "PROVINCIA".
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  FILLER                          PIC X(10) VALUE "ACUMULADO".

       01  CUERPO1.
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  PRO-OUT                         PIC X(20).
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  TEL-OUT                         PIC X(13).
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  POB-OUT                         PIC X(20).
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  DNI-OUT                         PIC X(15).

       01  CUERPO2.
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  DIR-OUT                         PIC X(20).
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  COD-OUT                         PIC 9(05).
           03  FILLER                          PIC X(10) VALUE SPACES.
           03  PRV-OUT                         PIC X(20).
           03  FILLER                          PIC X(02) VALUE SPACES.
           03  ACU-OUT                         PIC ZZZZZZZZZ9.

       01  PIE.
           03  FILLER                          PIC X(50) VALUE "LISTADO DE
      -    "PROVEEDORES.".
           03  FILLER                          PIC X(07) VALUE "PAGINA:".
           03  PAG-OUT                         PIC ZZ.

      * PAGINA Y LINEA NOS SERVIRAN PARA CONTROLAR LA PAGINA DE Y LINEA
      * DE IMPRESION RESPECTIVAMENTE.

       77  PAGINA                              PIC 9(02).
       77  LINEA                               PIC 9(02).

      * LAS VARIABLES PRI-PRO Y ULT-PRO NOS SERVIRAN PARA CONTROLAR
      * EL PRINCIPIO Y FIN DEL LISTADO.

       77  PRI-PRO                             PIC X(20).
       77  ULT-PRO                             PIC X(20).

      * LA VARIABLE ERRORES SERA EL FILE STATUS DE LOS FICHEROS.
```

```
       77  ERRORES                          PIC X(02).
      * CLV-PAN ES LA CLAVE DEL FICHERO DE PANTALLA.
       77  CLV-PAN                          PIC 9(01) VALUE 1.
      * EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA QUE DESEAMOS LEER.
       77  NOM-PAN                          PIC X(06).
      * OPCION NOS SERVIRA PARA PEDIR EL CORRECTO SI O NO.
       77  OPCION                           PIC X(01).
      * EN EL GRUPO EXCEP SE DECLARAN LAS EXCEPCIONES.
       01  EXCEP                            PIC 9(02) COMP.
           88  F1 VALUE IS 00.
           88  F2 VALUE IS 01.
           88  F3 VALUE IS 02.
       LINKAGE SECTION.
      * EN ESTA SECCION SE DECLARA EL CAMPO FECHA QUE
      * NOS VIENE DADO POR EL PROGRAMA ANTERIOR.
       01  FECHA.
           03  ANO                          PIC 9(02).
           03  MES                          PIC 9(02).
           03  DIA                          PIC 9(02).
       PROCEDURE DIVISION USING FECHA.
      * TENEMOS LAS DECLARATIVES PARA PREVENIR POSIBLES ERRORES.
       DECLARATIVES.
      * LA PRIMERA SECCION QUE TENEMOS ES LA DE PROVEEDORES.
       I O PROVEEDOR SECTION.
           USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.
       I O CREACION1.
           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
```

```
        DISPLAY "     EN EL FICHERO DE PROVEEDORES. LO CREO?.[ ]"
                LINE 25 POSITION 24 REVERSE.

    OPCIONCREACION1.

        ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
        IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
           CLOSE PROVEEDORES, OPEN I O PROVEEDORES, GO TO FIN1.
        IF OPCION NOT   "N" AND OPCION NOT = "n"
           GO TO OPCIONCREACION1.

    FIN1.

        DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

    I O IMP SECTION.

    * LA SIGUIENTE SECCION ES LA DE LA IMPRESORA.

        USE AFTER STANDARD ERROR PROCEDURE ON IMP.

    I O AVISO.

        DISPLAY " CONECTE LA IMPRESORA Y PULSE UNA TECLA "
                LINE 25 POSITION 15 REVERSE BLINK BEEP.
        ACCEPT OPCION OFF NO BEEP.
        WRITE FIC IMP.
        DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.

    END DECLARATIVES.

    * AQUI COMIENZA EL PROGRAMA PROPIAMENTE DICHO.

    PROGRAMA SECTION.

    COMIENZO.

    * SE SACA LA PANTALLA CORRESPONDIENTE.

        MOVE "SGS202" TO NOM-PAN.
        OPEN INPUT PANTALLA.
        READ PANTALLA.
        CLOSE PANTALLA.

    * SE IMPRIME LA PANTALLA Y LA FECHA.

        DISPLAY FIC-PAN LINE 1 POSITION 1.
        DISPLAY DIA LINE 23 POSITION 60.
```

```
           DISPLAY "/" LINE 23 POSITION 62.
           DISPLAY MES LINE 23 POSITION 63.
           DISPLAY "/" LINE 23 POSITION 65.
           DISPLAY ANO LINE 23 POSITION 66.

       * SE PASA LA FECHA QUE NOS VIENE DEL PROGRAMA ANTERIOR A LA FECHA
       * DE LA CABECERA.

           MOVE DIA TO DIA-OUT.
           MOVE MES TO MES-OUT.
           MOVE ANO TO ANO-OUT.

        APERTURA.

       * SE ABRE EL FICHERO DE PROVEEDORES EN MODO DIRECTO.

           OPEN I-O PROVEEDORES.

        PIDEPRIMERO.

       * SE PIDE EL PRIMER PROVEEDOR A LISTAR.

           ACCEPT PRI-PRO LINE 10 POSITION 38 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SI SE PULSA F2 SE VA A PEDIR OPCION.

           IF F2 GO TO PEDIROPCION.

       * SI SE PULSA F3 SE MUEVEN ESPACIOS AL CAMPO, SE IMPRIME
       * Y SE VUEVE A PEDIR.

           IF F3 MOVE SPACES TO PRI-PRO, DISPLAY PRI-PRO LINE 18
                 POSITION 38, GO TO PIDEPRIMERO.

        PIDEULTIMO.

       * SE PIDE EL ULTIMO PROVEEDOR A LISTAR.

           ACCEPT ULT-PRO LINE 12 POSITION 38 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

       * SI SE HA PULSADL LA TECLA TAB O POSIBLEMENTE RETURN SE LLENA
       * EL CAMPO DE ZETAS PARA QUE SE LISTE HASTA EL FINAL DEL FICHERO.

           IF ULT-PRO = SPACES MOVE "ZZZZZZZZZZZZZZZZZZZZ" TO ULT-PRO,
              DISPLAY ULT-PRO LINE 12 POSITION 38.
```

```
    * SI SE HA PULSADO F1 SE VUELVE A PEDIR EL PRIMER PROVEEDOR A
    * A LISTAR.

        IF F1 GO TO PIDEPRIMERO.

    * SI SE PULSA F3 SE MUEVEN ESPACIOS AL CAMPO, SE IMPRIME Y SE
    * VUELVE A PEDIR.

        IF F3 MOVE SPACES TO ULT PRO, DISPLAY ULT-PRO LINE 12
           POSITION 38, GO TO PIDEULTIMO.

     PEDIROPCION.

    * TRAS PEDIR EL INICIO Y FIN DEL LISTADO SE PIDE OPCION.

        ACCEPT OPCION LINE 17 POSITION 49 NO BEEP.

    * SI SE PULSA C DE CORRECTO SE VA A LA ETIQUETA LISTAR.

        IF OPCION = "C" OR OPCION = "c" GO TO LISTAR.

    * SI SE PULSA I DE INCORRECTO SE VUELVEN A PEDIR EL INICIO Y FIN.

        IF OPCION = "I" OR OPCION = "i" GO TO PIDEPRIMERO.

    * POR ULTIMO SI LA OPCION NO ES S DE SALIR ES QUE ES INVALIDA, POR
    * LO TANTO SE VUELVE A PEDIR.

        IF OPCION NOT = "S" AND OPCION NOT = "s" GO TO PEDIROPCION.

     FINPROGRAMA.

    * SI SE LLEGA AQUI ES QUE LA OPCION TECLEADA HA SIDO S.

    * SE CIERRA EL FICHERO DE PROVEEDORES.

        CLOSE PROVEEDORES.

     SALIRPROGRAMA.

    * Y SE VUELVE AL PORGRAMA ANTERIOR.

        EXIT PROGRAM.

     LISTAR.

    * SI SE LLEGA AQUI ES QUE LA OPCION TECLEADA HA SIDO C.
```

```
* SE MUEVE EL CAMPO PRI-PRO QUE TIENE EL PRIMER PROVEEDOR A
* LISTAR A PRO-PRO. SEGUIDAMENTE SE HACE UN START PARA POSICIONARSE
* EN EL PROVEEDOR INDICADO, SI DA CLAVE INVALIDA QUIERE DECIR QUE
* NO HAY NINGUN REGISTRO A PARTIR DEL INDICADO, SE VA A LA ETIQUETA
* NOHAY.

     MOVE PRI-PRO TO PRO-PRO.
     START PROVEEDORES KEY IS NOT < PRO-PRO INVALID KEY
           GO TO NOHAY.

* SI SE LLEHA AQUI QUIERE DECIR QUE HAY REGISTROS PARA LISTAR.
* SE ABRE EL FICHERO DE IMPRESORA, SE MUEVE 0 A LA PAGINA Y A LA
* LINEA Y SE LLAMA A LA ETIQUETA CABECERA.

     OPEN OUTPUT IMP.
     MOVE 0 TO LINEA, PAGINA.
     PERFORM CABECERA.

 LEER.

* TRAS IMPRIMIR LA CABECERA SE COMENZARA A LEER EN SECUENCIAL DEL
* FICHERO A ESTE QUE SE LLEGUE AL FINAL DEL MISMO O HASTA QUE EL
* PROVEEDOR LEIDO SEA MAYOR QUE EL INDICADO COMO FINAL, EN ESE
* MOMENTO SE IRA A LA ETIQUETA FINLISTADO.

     READ PROVEEDORES NEXT RECORD AT END GO TO FINLISTADO.
     IF PRO-PRO > ULT-PRO GO TO FINLISTADO.

* SI NO SE HA LLEGADO AL FINAL SE PASAN TODOS LOS CAMPOS DEL FICHERO
* A LOS DEL CUERPO DEL LISTADO.

     MOVE PRO-PRO TO PRO-OUT.
     MOVE DIR-PRO TO DIR-OUT.
     MOVE COD-PRO TO COD-OUT.
     MOVE POB-PRO TO POB-OUT.
     MOVE PRV-PRO TO PRV-OUT.
     MOVE TEL-PRO TO TEL-OUT.
     MOVE DNI-PRO TO DNI-OUT.
     MOVE ACU-PRO TO ACU-OUT.

* SE IMPRIMEN LOS CUERPOS Y LA CABECERA SEPARADORA.

     WRITE FIC-IMP FROM CUERPO1.
     WRITE FIC-IMP FROM CUERPO2.
     WRITE FIC-IMP FROM CABECERA2.

* SE ACTUALIZA LA VARIABLE LINEA.
```

```
           ADD 3 TO LINEA.

      * SE CONTROLA QUE SI LA LINEA ES MAYOR DE 57 SE IMPRIMA EL PIE
      * DE PAGIA, SE SALTE A OTRA Y SE IMPRIMA LA CABECERA.

           IF LINEA > 57 PERFORM PIE, PERFORM CABECERA.

      * VUELVE A SEGUIR LEYENDO.

           GO TO LEER.

       CABECERA.

      * CADA VEZ QUE SE LLAMA AQUI SE IMPRIMEN LAS DISTINTAS CABECERAS.

           WRITE FIC IMP FROM CABECERA1.
           WRITE FIC IMP FROM CABECERA2.
           WRITE FIC IMP FROM CABECERA3.
           WRITE FIC IMP FROM CABECERA4.

      * SE DEJA UNA LINEA EN BLANCO Y SE ACTUALIZA LA VARIABLE LINEA.

           PERFORM LINBLAN.
           MOVE 5 TO LINEA.

       PIE.

      * CUANDO SE LLAMA A ESTA ETIQUETA, LO PRIMERO QUE SE HACE ES
      * SALTAR HASTA EL FINAL DE LA PAGINA.

           COMPUTE LINEA = 64 - LINEA + 2.
           PERFORM LINBLAN LINEA TIMES.

      * SE LE SUMA UNO A LA PAGINA.

           ADD 1 TO PAGINA.
           MOVE PAGINA TO PAG-OUT.

      * SE IMPRIME EL PIE.

           WRITE FIC-IMP FROM PIE.

      * SE SALTA HASTA LA SIGUIENTE PAGINA.

           PERFORM LINBLAN 3 TIMES.

       FINLISTADO.
```

```
      * CUANDO SE LLEGA AL FIN DEL LISTADO SE IMPRIME EL PIE DE PAGINA
      * SE CIERRA LA IMPRESORA Y SE VA A LA ETIQUETA FINPROGRAMA.

           PERFORM PIE.
           CLOSE IMP WITH NO REWIND.
           GO TO FINPROGRAMA.

       NOHAY.

      * SI A PARTIR DEL REGISTRO INDICADO NO SE ENCUENTRA NINGUNO
      * SE AVISA Y SE Y SE SALE DEL PROGRAMA.

           DISPLAY " NO HAY NINGUN REGISTRO A PARTIR DEL INDICADO "
                   LINE 25 POSITION 15 REVERSE BLINK BEEP.
           PERFORM ESPERA 5000 TIMES.
           DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.
           GO TO FINPROGRAMA.

       LINBLAN.

      * CADA VEZ QUE SE LLAMA A LINBLAN SE IMPRIME UNA LINEA EN BLANCO.

           MOVE SPACES TO FIC-IMP.
           WRITE FIC-IMP.

       ESPERA.

      * EN ESTA ETIQUETA SE MUEVE OPCION A SI MISMA PARA PERDER TIEMPO.

           MOVE OPCION TO OPCION.
```

El siguiente programa que tenemos es el de entradas de movimiento diario, en lo que consiste es en ir dando entrada a unas cantidades de un articulo determinado, al decirle nosotros el articulo nos sacará el proveedor y el precio costo, este se podrá modificar, al darle precio y cantidad nos dirá el importe y se pasará a otro movimiento. Los movimientos se irán numerando de tal forma que al comenzar el programa se pondrá en el siguiente asiento al último que hubiera en ese momento, para modifcar un movimiento tan sólo hemos de teclear su número y nos aparecerá en pantalla. Vamos a ver a continuación un breve seguimiento lógico del programa.

La clave del fichero de movimientos está constituida por la fecha de ese movimiento, el tipo de ese movimiento y el número de línea del mismo. Al comenzar el programa lo primero que hace es presentarnos las máscara de petición de datos, a continuación tomará como fecha para los movimientos la que hallamos tecleado a la entrada al programa, con esta fecha comenzará a buscar el último movimiento del día, ya que puede ser que halla movimientos del mismo día, tras buscarlo se pone tras el último movimiento, si no hay ninguno se pone en el movimiento 1; entonces se queda esperando a que le demos el número de línea de movimiento, si aceptamos el que él nos dá sólo tenemos que pulsar return, si pulsamos otro número se comprobará si existe o no, en caso de que exista se imprimirá y se nos dará opción a modificarlo, una vez que hallamos cambiado el número se seguirá a partir del que hallamos dado. Tras el número de movimiento se nos pedirá el artículo, se comprobará que este existe, en caso de que no existe se nos devolverá un aviso y se volverá a pedir, si todo vá bien se imprimirá en pantalla el proveedor que nos suministra dicho artículo y su precio, se nos dará opción a modificar el precio y por último se nos pedirá la cantidad, con estos últimos datos se hallará el importe y se imprimirá en pantalla. Una vez terminado este proceso se grabará el movimiento y se bajará a la siguiente línea con el número de movimiento incrementado en 1. Cuando se llegue al final de la máscara se borrará la pantalla y se volverá al principio otra vez. A lo largo de la petición de campos podemos usar las teclas de función con los mismo resultados que en los programas anteriores, la diferencia está en F2 que en vez de llavarnos a pedir opción lo que hace es sacarnos del programa, esta tecla se ha de usar obligatoriamente cuando querramos terminar de dar entradas.

Seguidamente tenemos el listado del programa con todos los comentarios que se han creido necesarios.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM ID. PGS301.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.
```

```
       INPUT OUTPUT SECTION.
       FILE CONTROL.

      * SE HACE UN COPY A LAS LIBRERIAS DE SELECTs DE LOS FICHEROS
      * DE PANTALLA, ARTICULOS, PROVEEDORES Y MOVIMIENTO.

           COPY "LGSPANSE.LIB".

           COPY "LGSARTSE.LIB".

           COPY "LGSPROSE.LIB".

           COPY "LGSMOVSE.LIB".

       DATA DIVISION.
       FILE SECTION.

      * SE HACE LO MISMO A LAS LIBRERIAS DE FDs CORRESPONDIENTES.

       COPY "LGSPANFD.LIB".

       COPY "LGSARTFD.LIB".

       COPY "LGSPROFD.LIB".

       COPY "LGSMOVFD.LIB".

       WORKING-STORAGE SECTION.

      * LA VARIABLE IMPORTE NOS VA A SERVIR PAR CALCULAR EL IMPORTE DE
      * UN MOVMIENTO.

       77  IMPORTE                          PIC 9(08) VALUE 0.

      * LA VARIABLE LINEA NOS SERVIRA PARA SABER EN QUE LINEA DE LA
      * PANTALLA NOS HEMOS DE SITUAR EN CADA MOMENTO.

       77  LINEA                            PIC 9(03).

      * ERRORES ES EL FILE STATUS DE LOS FICHEROS QUE TENEMOS.

       77  ERRORES                          PIC X(02).

      * CLV-PAN ES LA CLAVE DEL FICHERO DE PANTALLAS.

       77  CLV-PAN                          PIC 9(01) VALUE 1.
```

```
* EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA QUE DESEAMOS LEER.

 77  NOM-PAN                          PIC X(06).

* LA VARIABLE OPCION NOS SERVIRA EN ESTE CASO PARA LOS ON EXCEPTION

 77  OPCION                           PIC X(01).

* EN EL GRUPO EXCEP DECLARAMOS LOS CODIGOS DE LAS TECLAS QUE CAUSARAN
* EXCEPCION.

 01  EXCEP                            PIC 9(02) COMP.
     88  F1 VALUE IS 00.
     88  F2 VALUE IS 01.
     88  F3 VALUE IS 02.

 LINKAGE SECTION.

* EN ESTA SECCION DECLARAMOS EL GRUPO FECHA QUE NOS VIENE DEL
* PROGRAMA ANTERIOR.

 01  FECHA.
     03  ANO                          PIC 9(02).
     03  MES                          PIC 9(02).
     03  DIA                          PIC 9(02).

 PROCEDURE DIVISION USING FECHA.

* EN LAS DECLARATIVES PREVENIMOS LOS POSIBLES ERRORES EN FICHEROS

 DECLARATIVES.

 I-O-ARTICULOS SECTION.

* LA PRIMERA SECCION ES LA DEL FICHERO DE ARTICULOS.

     USE AFTER STANDARD ERROR PROCEDURE ON ARTICULOS.

 I-O-CREACION1.

     DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
     DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
     DISPLAY "     EN EL FICHERO DE ARTICULOS. LO CREO?.[ ]"
             LINE 25 POSITION 24 REVERSE.

 OPCIONCREACION1.
```

```
           ACCEPT OPCION LINE 25 POSITION 66 REVERSE.
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT ARTICULOS,
              CLOSE ARTICULOS, OPEN I-O ARTICULOS, GO TO FIN1.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION1.

       FIN1.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I-O-PROVEEDOR SECTION.

      * LA SEGUNDA ES LA DEL FICHERO DE PROVEEDORES.

           USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.

       I-O-CREACION2.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "    EN EL FICHERO DE PROVEEDORES. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION2.

           ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
              CLOSE PROVEEDORES, OPEN I-O PROVEEDORES, GO TO FIN2.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION2.

       FIN2.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I-O-MOVIMIENTO SECTION.

      * LA ULTIMA SECCION ES LA DEL FICHERO DE MOVIMIENTO.

           USE AFTER STANDARD ERROR PROCEDURE ON MOVIMIENTO.

       I-O-CREACION3.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "    EN EL FICHERO DE MOVIMIENTO.  LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.
```

```
     OPCIONCREACION3.

         ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
         IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT MOVIMIENTO,
            CLOSE MOVIMIENTO, OPEN I-O MOVIMIENTO, GO TO FIN3.
         IF OPCION NOT = "N" AND OPCION NOT = "n"
            GO TO OPCIONCREACION3.

     FIN3.

         DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

     END DECLARATIVES.

    * AQUI COMIENZA EL PROGRAMA PROPIAMENTE DICHO.

     PROGRAMA SECTION.

     COMIENZO.

    * SE SACA LA PANTALLA CORRESPONDIENTE.

         MOVE "SGS301" TO NOM-PAN.
         OPEN INPUT PANTALLA.
         READ PANTALLA.
         CLOSE PANTALLA.
         DISPLAY FIC-PAN LINE 1 POSITION 1.

    * TRAS IMPRIMIR LA PANTALLA IMPRIMIMOS LA FECHA QUE NOS VIENE
    * DEL PROGRAMA ANTERIOR.

         DISPLAY DIA LINE  6 POSITION 10.
         DISPLAY MES LINE  6 POSITION 13.
         DISPLAY ANO LINE  6 POSITION 16.

     APERTURA.

    * SE ABREN LOS FICHEROS DE ARTICULOS, PROVEEDORES Y MOVIMIENTO
    * EN MODO I-O, LA PRIMERA VEZ EN EL FICHERO DE MOVIMIENTO DARA
    * UN ERROR AL NO EXISTIR Y SE NOS DARA OPCION A CREARLO.

         OPEN I O ARTICULOS, PROVEEDORES, MOVIMIENTO.

     BUSCAFECHA.

    * SE MUEVE LA FECHA QUE NOS VIENE DEL PROGRAMA ANTERIOR AL CAMPO
    * FEC MOV QUE FORMA PARTE DE LA CLAVE DEL FICHERO DE MOVIMENTO.
```

```
        MOVE FECHA TO FEC-MOV.

    * SE MUEVE UNA E AL TIPO DE MOVIMIENTO YA QUE SON ENTRADAS.

        MOVE "E" TO TIP-MOV.

    * SE MUEVE 1 A LA LINEA DE MOVIMIENTO.

        MOVE 1 TO LIN-MOV.

    * SE HACE UN START EN BUSCA DE ESE MOVIMIENTO, SI NO EXISTE DARA
    * CLAVE INVALIDA LO QUE QUIERE DECIR QUE ES EL PRIMER MOVIMIENTO
    * DEL DIA, SE VA A LA ETIQUETA PRIMERO.

        START MOVIMIENTO KEY IS NOT < COD-MOV
              INVALID KEY GO TO PRIMERO.

    * SI LLEGA AQUI QUIERE DECIR QUE HA ENCONTRADO EL MOVIMIENTO
    * INDICADO O UNO POSTERIOR, EN ESE CASO NOS VAMOS A BUSCALINEA.

        GO TO BUSCALINEA.

     PRIMERO.

    * AQUI SE LLEGARA CUANDO SEA EL PRIMER MOVIMIENTO DEL DIA, MOVEMOS
    * UN CERO A LA LINEA Y NOS VAMOS A LA ETIQUETA EMPIEZA.

        MOVE 0 TO LINEA.
        GO TO EMPIEZA.

     BUSCALINEA.

    * EN ESTA ETIQUETA LO QUE VAMOS A HACER ES BUSCAR EL ULTIMO
    * MOVIMIENTO DEL DIA PARA EMPEZAR EN EL SIGUIENTE.

    * LEEMOS DEL FICHERO DE MOVIMIENTOS EN FORMA SECUENCIAL, SI SE
    * LLEGA AL FINAL DEL MISMO SE VA A LA ETIQUETA EMPIEZA.

        READ MOVIMIENTO NEXT RECORD AT END GO TO EMPIEZA.

    * SI LA FECHA DEL MOVIMIENTO QUE SE ACABA DE LEER ES MAYOR QUE
    * LA FECHA ACTUAL, O DE SI EL TIPO YA NO ES E, EN ESTOS CASOS
    * SE IRA A LA ETIQUETA EMPIEZA.

        IF FEC-MOV > FECHA OR TIP-MOV > "E" GO TO EMPIEZA.

    * SE MUEVE LA LINEA QUE SE ACABA DE LEER AL CAMPO LINEA.
```

```
        MOVE LIN-MOV TO LINEA.

    * SE IMPRIMEN EN LA PRIMERA LINEA DE LA MASCARA DE PETICION LA
    * LINEA Y EL ARTICULO.

        DISPLAY LIN-MOV LINE 10 POSITION 4.
        DISPLAY COA-MOV LINE 10 POSITION 10.

    * SE MUEVE EL ARTICULO AL CAMPO ART-ART QUE ES LA CLAVE DEL FICHERO
    * DE ARTICULOS.

        MOVE COA-MOV TO ART-ART.

    * SE LEE DEL FICHERO DE ARTICULOS PARA SABER EL PROVEEDOR DEL
    * ARTICULO INDICADO.

        READ ARTICULOS.

    * SE IMPRIME EL PROVEEDOR, EL PRECIO Y LA CANTIDAD.

        DISPLAY PRO-ART LINE 10 POSITION 26.
        DISPLAY PRE-MOV LINE 10 POSITION 50.
        DISPLAY CAN-MOV LINE 10 POSITION 64.

    * SE CALCULA EL IMPORTE Y SE IMPRIME.

        COMPUTE IMPORTE = PRE-MOV * CAN-MOV.
        DISPLAY IMPORTE LINE 10 POSITION 72.

    * SE VUELVE A SEGUIR BUSCANDO LA ULTIMA LINEA.

        GO TO BUSCALINEA.

     EMPIEZA.

    * CUANDO SE HALLA ENCONTRADO LA ULTIMA LINEA O SI ERA EL PRIMER
    * MOVIMIENTO SE LLEGA AQUI.

    * SE MUEVE LA FECHA ACTUAL AL CAMPO FEC-MOV.


        MOVE FECHA TO FEC-MOV.
    * SE MUEVE E AL TIPO DE MOVIMIENTO.

        MOVE "E" TO TIP-MOV.

    * SI LA ULTIMA LINEA ENCONTRADA ESTA EN LINEA, LA SIGUIENTE LINEA
```

```
* DE MOVIMIENTO SERA LINEA MAS 1, HACEMOS ESTA OPERACION Y ALMACE
* NAMOS EL RESULTADO EN LIN-MOV.

    COMPUTE LIN-MOV = LINEA + 1.

* MOVEMOS 10 AL CAMPO LINEA, 10 ES LA PRIMERA LINEA DE LA MASCARA
* DE PETICION DE MOVIMIENTOS.

    MOVE 10 TO LINEA.

 PIDEMOVIMIENTO.

* SE DA OPCION A MODIFICAR EL NUMERO DE LINEA DE MOVIMENTO.

    ACCEPT LIN-MOV LINE LINEA POSITION 4 NO BEEP ECHO UPDATE
           TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

* SI SE HA PULSADO F1 SE VA A LA ETIQUETA FINPROGRAMA.

    IF F2 GO TO FINPROGRAMA.

* SI SE HA PULSADO F3 SE MUEVEN CEROS AL CAMPO LIN-MOV, SE IMPRIME
* Y SE VUELVE A PEDIR.

    IF F3 MOVE ZEROS TO LIN-MOV, DISPLAY LIN-MOV LINE LINEA
       POSITION  4, GO TO PIDEMOVIMIENTO.

 LEELINEA.

* TRAS PEDIR EL NUMERO DE MOVIMIENTO SE LEE PARA COMPROBAR SI ES
* UNA LINEA YA EXISTENTE O ES NUEVA, SI DA CLAVE INVALIDA ES QUE
* EL MOVIMIENTO ES NUEVO, SE VA A PIDEARTICULO EN ESE CASO.


    READ MOVIMIENTO INVALID KEY GO TO PIDEARTICULO.

* SI YA EXISTIA LLEGARA AQUI, SE PREGUNTA SI EL CAMPO ESTADO DE
* ESE MOVIMIENTO TIENE UN 2, SI ES ASI QUIERE DECIR QUE EL ASIENTO
* YA ESTA ACTUALIZADO Y POR LO TANTO NO SE DEJA MODIFICAR.

    IF EST-MOV = 2 DISPLAY " ASIENTO YA ACTUALIZADO "
       LINE 25 POSITION 25 REVERSE BLINK BEEP,
    GO TO PIDEMOVIMIENTO.

* SI SE LLEGA QUIERE DECIR QUE EL MOVIMIENTO EXISTE PERO NO ESTA
* ACTUALIZADO, POR LO TANTO SE PUEDE MODIFICAR, LO PRIMERO SE IMPRIME

    PERFORM IMPRIMIR.
```

```
    PIDEARTICULO.

  * AQUI SE LLEGARA TRAS IMPRIMIR LOS CAMPOS DEL REGISTRO SI EXISTIA
  * O DIRECTAMENTE SI NO EXISTIA.

  * SE PIDE EL ARTICULO.

       ACCEPT COA-MOV LINE LINEA POSITION 10 NO BEEP ECHO UPDATE
              TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

  * SI SE HA PULSADO F1 SE VUELVE A PEDIR EL MOVIMIENTO.

       IF F1 GO TO PIDEMOVIMIENTO.

  * SI FUE F2 SE VA A LA ETIQUETA FINPROGRAMA.

       IF F2 GO TO FINPROGRAMA.

  * SI LA QUE SE PULSO FUE F3 SE MUEVEN ESPACIOS AL ARTICULO, SE
  * IMPRIME Y SE VUELVE A PEDIR.

       IF F3 MOVE SPACES TO COA-MOV, DISPLAY COA-MOV LINE LINEA
          POSITION 10, GO TO PIDEARTICULO.

    LEEARTICULO.

  * UNA VEZ QUE TENEMOS EL ARTICULO SE LEE DEL FICHERO CORRESPONDIENTE
  * PARA COMPROBAR SI EXISTE, SI NO EXISTE DARA CLAVE INVALIDA, LO
  * QUE QUIERE DECIR QUE EL ARTICULO NO EXISTE Y POR LO TANTO NO LO
  * PODEMOS COMPRAR, EN ESE CASO SE IRA A ARTINVALIDO, SI NO TIENE
  * PROBLEMAS AL LEERLO SE IRA A LA ETIQUETA IMPRIMEPROVEEDOR.

       MOVE COA-MOV TO ART-ART.
       READ ARTICULOS INVALID KEY GO TO ARTINVALIDO.
       GO TO IMPRIMEPROVEEDOR.

    ARTINVALIDO.

  * SI EL ARTICULO NO EXISTE SE AVISA Y SE VUELVE A PEDIR.

       DISPLAY " ESE ARTICULO NO EXISTE " LINE 25 POSITION 25
               REVERSE BLINK BEEP.
       GO TO PIDEARTICULO.

    IMPRIMEPROVEEDOR.

  * TRAS LEER EL ARTICULOS SE IMPRIME EL PROVEEDOR.
```

```
        DISPLAY SPACES SIZE 30 LINE 25 POSITION 25.
        DISPLAY PRO-ART LINE LINEA POSITION 26.

     IMPRIMEPRECIO.

    * Y TAMBIEN EL PRECIO, EL PRECIO SE MUEVE AL CAMPO PRE-MOV.

        DISPLAY PRC-ART LINE LINEA POSITION 50.
        MOVE PRC-ART TO PRE-MOV.

     PIDEPRECIO.

    * Y SE DA OPCION A MODIFICARLO.

        ACCEPT PRE-MOV LINE LINEA POSITION 50 NO BEEP ECHO UPDATE
               CONVERT TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSA F1 SE VUELVE A PEDIR EL ARTICULO.

        IF F1 GO TO PIDEARTICULO.

    * SI SE PULSO F2 SE VA A LA ETIQUETA FINPROGRAMA.

        IF F2 GO TO FINPROGRAMA.

    * SI FUE F3 EL PULSADO SE MUEVEN CEROS AL PRECIO, SE IMPRIME
    * Y SE VUELVE A PEDIR.

        IF F3 MOVE ZEROS TO PRE MOV, DISPLAY PRE MOV LINE LINEA
           POSITION 50, GO TO PIDEPRECIO.

     PIDECANTIDAD.

    * SE PIDE LA CANTIDAD.

        ACCEPT CAN-MOV LINE LINEA POSITION 64 NO BEEP ECHO UPDATE
               CONVERT TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSA F1 SE VUELVE A PEDIR EL PRECIO.

        IF F1 GO TO PIDEPRECIO.

    * SI SE PULSO F2 SE VA A LA ETIQUETA FINPROGRAMA.

        IF F2 GO TO FINPROGRAMA.

    * SI SE PULSO F3 SE MUEVEN CEROS A LA CANTIDAD, SE IMPRIME
    * Y SE VUELVE A PEDIR.
```

```
        IF F3 MOVE ZEROS TO CAN MOV, DISPLAY CAN MOV LINE LINEA
           POSITION 64, GO TO PIDECANTIDAD.

     IMPRIMEIMPORTE.

    * UNA VEZ TENEMOS LA CANTIDAD Y EL PRECIO LOS MULTIPLICAMOS
    * PARA HALLAR EL IMPORTE, EL RESULTADO SE DEJA EN EL CAMPO
    * IMPORTE Y SE IMPRIME.

        MULTIPLY CAN MOV BY PRE MOV GIVING IMPORTE.
        DISPLAY IMPORTE LINE LINEA POSITION 72.

     GRABAMOVIMIENTO.

    * UNA VEZ TENEMOS TODOS LOS CAMPOS SE MUEVE UN 1 AL ESTADO DEL
    * MOVIMIENTO, ESTO QUIERE DECIR QUE TODAVIA NO HA SIDO ACTUALIZADO.

        MOVE 1 TO EST-MOV.

    * SE GRABA EL MOVIMIENTO, EN CASO DE QUE DE CLAVE INVALIDA ES QUE
    * SE TRATABA DE UN MOVIMIENTO EXISTENTE QUE POSIBLEMENTE SE HALLA
    * MODIFICADO POR LO TANTO SE REESCRIBE.

        WRITE FIC-MOV INVALID KEY REWRITE FIC-MOV.

    * SE LE SUMA 1 AL NUMERO DE MOVIMIENTO Y A LA LINEA DE PANTALLA.

        ADD 1 TO LIN-MOV.
        ADD 1 TO LINEA.

    * SI LA LINEA ES MENOR DE 22 SE VUELVE A SEGUIR PIDIENDO
    * MOVIMIENTOS.

        IF LINEA < 22 GO TO PIDEMOVIMIENTO.

    * SI LLEGA AQUI QUIERE DECIR QUE YA SE HA LLENADO LA MASCARA DE
    * PETICION POR LO TANTO SE BORRA, SE PONE LINEA A 10 OTRA VEZ Y
    * SE VUELVE A SEGUIR PIDIENDO MOVIMIENTOS.

        PERFORM BORRA VARYING LINEA FROM 10 BY 1 UNTIL LINEA > 21.
        MOVE 10 TO LINEA.
        GO TO PIDEMOVIMIENTO.

     BORRA.

    * ESTA RUTINA ES LLAMADA PARA BORRAR LOS CAMPOS DE UNA LINEA.
```

```
           DISPLAY SPACES SIZE  3 LINE LINEA POSITION 4.
           DISPLAY SPACES SIZE 15 LINE LINEA POSITION 10.
           DISPLAY SPACES SIZE 20 LINE LINEA POSITION 26.
           DISPLAY SPACES SIZE  5 LINE LINEA POSITION 50.
           DISPLAY SPACES SIZE  4 LINE LINEA POSITION 64.
           DISPLAY SPACES SIZE  8 LINE LINEA POSITION 72.

       IMPRIMIR.

      * CUANDO SE ENCUENTRA UN REGISTRO QUE EXISTE SE LLAMA A ESTA
      * ETIQUETA PARA IMPRIMIR LOS CAMPOS, PARA ELLO SE LEE DEL FICHERO
      * DE ARTICULOS Y SE CALCULA EL IMPORTE.

           DISPLAY COA-MOV LINE LINEA POSITION 10.
           MOVE COA-MOV TO ART-ART.
           READ ARTICULOS.
           DISPLAY PRO-ART LINE LINEA POSITION 26.
           DISPLAY CAN-MOV LINE LINEA POSITION 64.
           DISPLAY PRE-MOV LINE LINEA POSITION 50.
           COMPUTE IMPORTE = CAN-MOV * PRE-MOV.
           DISPLAY IMPORTE LINE LINEA POSITION 72.

       FINPROGRAMA.

      * A ESTA ETIQUETA SE LLEHA CUANDO HEMOS PULSADO LA TECLA F2
      * SE CIERRAN LOS FICHEROS.

           CLOSE MOVIMIENTO, ARTICULOS, PROVEEDORES.

       SALIRPROGRAMA.

      * Y SE SALE DEL PROGRAMA.


           EXIT PROGRAM.
```

El siguiente programa será el de salidas de diario, este es muy parecido al anterior, la diferencia está en que el tipo ahora es 'S' de salida, y que controlamos que la cantidad que se dá de salida no sea mayor que la que tenemos en stock. Por lo demás el programa es practicamente igual. Seguidamente tienes el listado.

```
123456789012
      A   B
```

APLICACION DE GESTION DE SUPERMERCADOS

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PGS302.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

      * SE HACE UN COPY A LAS LIRBERIAS DE SELECTs DE LOS FICHEROS DE
      * PANTALLA, ARTICULOS, PROVEEDORES Y MOVIMIENTO.

           COPY "LGSPANSE.LIB".

           COPY "LGSARTSE.LIB".

           COPY "LGSPROSE.LIB".

           COPY "LGSMOVSE.LIB".


       DATA DIVISION.
       FILE SECTION.

      * SE HACE LO MISMO CON LAS LIBRERIAS DE FDs

       COPY "LGSPANFD.LIB".

       COPY "LGSARTFD.LIB".

       COPY "LGSPROFD.LIB".

       COPY "LGSMOVFD.LIB".

       WORKING-STORAGE SECTION.

      * EL CAMPO IMPORTE NOS SERVIRA PARA IR HALLANDO LOS IMPORTES.

       77  IMPORTE                          PIC 9(08) VALUE 0.

      * LINEA NOS SERVIRA PARA SABER EN LINEA DE LA PANTALLA HEMOS DE
      * IMPRIMIR EN CADA MOMENTO.

       77  LINEA                            PIC 9(03).
```

```
      * LA VARIABLE ERRORES SERVIRA DE FILE STATUS PARA LOS FICHEROS.

       77  ERRORES                          PIC X(02).

      * CLV-PAN ES EL CODIGO DEK FICHERO DE PANTALLA.

       77  CLV-PAN                          PIC 9(01) VALUE 1.

      * EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA A LEER.

       77  NOM-PAN                          PIC X(06).

      * LA OPCION NOS SERVIRA PARA LOS ON EXCEPTION.

       77  OPCION                           PIC X(01).

      * EN EL GRUPO EXCEP DECLARAMOS  LOS CODIGOS DE LAS TECLAS QUE
      * CAUSARAN EXCEPCION AL PEDIR UN CAMPO.

       01  EXCEP                            PIC 9(02) COMP.
           88  F1 VALUE IS 00.
           88  F2 VALUE IS 01.
           88  F3 VALUE IS 02.

       LINKAGE SECTION.

      * EN ESTA SECCION DECLARAMOS EL GRUPO FECHA QUE NOS VIENE DEL
      * PROGRAMA ANTERIOR.

       01  FECHA.
           03  ANO                          PIC 9(02).
           03  MES                          PIC 9(02).
           03  DIA                          PIC 9(02).

       PROCEDURE DIVISION USING FECHA.

      * EN LAS DECLARATIVES PREVENIMOS LOS POSIBLES ERRORES DE LOS
      * FICHEROS QUE UTILIZAMOS.

       DECLARATIVES.

       I-O-ARTICULOS SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON ARTICULOS.

       I-O-CREACION1.
```

```
           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "     EN EL FICHERO DE ARTICULOS. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION1.

           ACCEPT OPCION LINE 25 POSITION 66 REVERSE.
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT ARTICULOS,
              CLOSE ARTICULOS, OPEN I-O ARTICULOS, GO TO FIN1.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION1.

       FIN1.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I-O-PROVEEDOR SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.

       I-O-CREACION2.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "     EN EL FICHERO DE PROVEEDORES. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION2.

           ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
              CLOSE PROVEEDORES, OPEN I-O PROVEEDORES, GO TO FIN2.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION2.

       FIN2.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I-O-MOVIMIENTO SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON MOVIMIENTO.

       I-O-CREACION3.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
```

```
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "     EN EL FICHERO DE MOVIMIENTO.  LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION3.

           ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT MOVIMIENTO,
              CLOSE MOVIMIENTO,  OPEN I-O MOVIMIENTO,  GO TO FIN3.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION3.

       FIN3.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       END DECLARATIVES.
       PROGRAMA SECTION.

       COMIENZO.

      * SE PRESENTA LA PANTALLA DE SALIDAS DE DIARIO.

           MOVE "SGS302" TO NOM-PAN.
           OPEN INPUT PANTALLA.
           READ PANTALLA.
           CLOSE PANTALLA.

      * SE MUEVE 0 AL ACUMULADOR DE PROVEEDORES.

           MOVE 0 TO ACU-PRO.
           DISPLAY FIC-PAN LINE 1 POSITION 1.

      * SE IMPRIME LA FECHA EN PANTALLA.

           DISPLAY DIA LINE  6 POSITION 10.
           DISPLAY MES LINE  6 POSITION 13.
           DISPLAY ANO LINE  6 POSITION 16.

       APERTURA.

      * SE ABREN LOS FICHEROS DE ARTICULOS,  PROVEEDORES Y MOVIMIENTO.

           OPEN I-O ARTICULOS,  PROVEEDORES,  MOVIMIENTO.

       BUSCAFECHA.
```

```
* MUEVE LA FECHA ACTUAL AL CAMPO FEC-MOV.

     MOVE FECHA TO FEC-MOV.

* MUEVE UNA S A TIP-MOV QUE ES EL TIPO DE MOVIMIENTO.

     MOVE "S" TO TIP-MOV.

* MUEVE UN 1 LA LINEA DE MOVIMIENTO.

     MOVE 1 TO LIN-MOV.

* SE HACE UN START PARA BUSCAR EL MOVIMIENTO, SI DA CLAVE
* INVALIDA ES QUE ES EL PRIMER MOVIMIENTO DEL DIA, EN ESE
* CASO SE VA A PRIMERO.

     START MOVIMIENTO KEY IS NOT < COD-MOV
           INVALID KEY GO TO PRIMERO.

* EN CASO DE QUE SE LLEGUE AQUI QUIERE DECIR QUE SE HA ENCONTRADO
* EL PRIMER MOVIMIENTO Y SE VA A BUSCALINEA.

     GO TO BUSCALINEA.

 PRIMERO.

* SI SE LLEGA AQUI QUIERE DECIR QUE NO HABIA NINGUN MOVIMIENTO, POR
* LO TANTO ES EL PRIMERO DEL DIA, SE MUEVE 0 A LA LINEA Y SE VA A
* LA ETIQUETA EMPIEZA.

     MOVE 0 TO LINEA.
     GO TO EMPIEZA.

 BUSCALINEA.

* AQUI SE LLEGARA CUANDO HAY MAS MOVIMIENTOS ANTERIORES.

* SE LEE EN SECUENCIAL HASTA QUE SE LLEGUE AL FINAL DE FICHERO
* EN ESE MOMENTO SE IRA A LA ETIQUETA EMPIEZA.

     READ MOVIMIENTO NEXT RECORD AT END GO TO EMPIEZA.

* SI LA FECHA QUE SE ACABA DE LEER ES MAYOR QUE LA FECHA ACTUAL
* SE VA A LA ETIQUETA EMPIEZA.

     IF FEC MOV > FECHA GO TO EMPIEZA.
```

```
* SI SE LLEGA AQUI QUIERE DECIR        QUE EL MOVIMIENTO LEIDO ES DEL
* DIA ACTUAL,  SE PASA A LA VARIABLE LINEA.

     MOVE LIN-MOV TO LINEA.

* SE IMPRIMEN LA LINEA Y EL ARTICULO DEL MOVIMIENTO.

     DISPLAY LIN-MOV LINE 10 POSITION 4.
     DISPLAY COA-MOV LINE 10 POSITION 10.

* SE MUEVE EL ARTICULO A ART-ART Y SE LEE DEL FICHERO DE ARTICULOS.

     MOVE COA-MOV TO ART-ART.
     READ ARTICULOS.

* SE IMPRIMEN LOS DEMAS CAMPOS.

     DISPLAY PRO-ART LINE 10 POSITION 26.
     DISPLAY PRE-MOV LINE 10 POSITION 50.
     DISPLAY CAN-MOV LINE 10 POSITION 64.
     COMPUTE IMPORTE = PRE-MOV * CAN-MOV.
     DISPLAY IMPORTE LINE 10 POSITION 72.
     GO TO BUSCALINEA.

 EMPIEZA.

* CUANDO SE LLEGA AQUI ES QUE YA SE HA ENCONTRADO EL MOVIMIENTO
* O QUE ERA EL PRIMERO,  EL PROXIMO MOVIMIENTO SERA EL ULTIMO
* ENCONTRADO MAS 1,  SE HACE ESTA OPERACION Y SE DEPOSITA EN LIN-MOV

     COMPUTE LIN-MOV = LINEA + 1.

* SE MUEVE 10 AL CAMPO LINEA,  ESTE NOS INDICARA EN CADA MOMENTO
* EN QUE LINEA SE HA DE IMPRIMIR EN CADA MOMENTO.

     MOVE 10 TO LINEA.

 PIDEMOVIMIENTO.

* SE PIDE EL MOVIMIENTO,  EN CASO DE QUERER DEJAR EL QUE
* HAY PULSAREMOS RETURN.

     ACCEPT LIN-MOV LINE LINEA POSITION 4 NO BEEP ECHO UPDATE
            TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

* SI SE PULSA LA TECLA F2 SE VA A LA ETIQUETA FINPROGRAMA.

     IF F2 GO TO FINPROGRAMA.
```

```
* SI SE PULSO F3 SE MUEVEN CEROS AL NUMERO DE MOVIMIENTO, SE
* IMPRIME Y SE VUELVE A PEDIR.

     IF F3 MOVE ZEROS TO LIN-MOV, DISPLAY LIN-MOV LINE LINEA
        POSITION  4,  GO TO PIDEMOVIMIENTO.

 LEELINEA.

* UNA VEZ SE TIENE EL NUMERO DE MOVIMIENTO SE LEE DEL FICHERO
* DE MOVIMIENTOS PARA COMPROBAR SI EXISTE, EN CASO DE QUE
* NO EXISTE DARA CLAVE INVALIDA Y SE IRA A LA ETIQUETA PIDEARTICULO

     READ MOVIMIENTO INVALID KEY GO TO PIDEARTICULO.

* SI YA EXISTE SE PREGUNTA SI SU CAMPO DE ESTADO TIENE UN 2, DE SER
* ASI QUIERE DECIR QUE ESE MOVIMIENTO YA ESTA ACTUALIZADO Y POR LO
* TANTO NO SE PUEDE MODIFICAR.

     IF EST-MOV = 2 DISPLAY " ASIENTO YA ACTUALIZADO "
        LINE 25 POSITION 25 REVERSE BLINK BEEP,
        GO TO PIDEMOVIMIENTO.

* SI SE LLEGA AQUI QUIERE DECIR QUE EL MOVIMIENTO EXISTE Y ADEMAS NO
* ESTA ACTUALIZADO,  POR LO CUAL SE PUEDE MODIFICAR,  SE LLAMA A
* LA ETIQUETA IMPRIMIR PARA IMPRIMIRLO Y DESPUES SE SIGUE PIDIENDO.

     PERFORM IMPRIMIR.

 PIDEARTICULO.

* SE PIDE EL ARTICULO.

     ACCEPT COA-MOV LINE LINEA POSITION 10 NO BEEP ECHO UPDATE
            TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

* SI SE PULSO F1 SE VA A LA ETIQUETA PIDEMOVIMIENTO.

     IF F1 GO TO PIDEMOVIMIENTO.

* SI FUE F2 SE VA A FINPROGRAMA.

     IF F2 GO TO FINPROGRAMA.

* SI LA QUE SE PULSO FUE F3 SE MUEVEN ESPACIOS AL ARTICULO,  SE
* IMPRIME Y SE VUELVE A PEDIR.

     IF F3 MOVE SPACES TO COA-MOV, DISPLAY COA MOV LINE LINEA
        POSITION 10,  GO TO PIDEARTICULO.
```

```
      LEEARTICULO.

    * TRAS PEDIR EL ARTICULO SE LEE, SI DA CLAVE INVALIDA ES QUE
    * EL ARTICULO NO EXISTE Y POR LO TANTO NO SE LE PUEDE DAR SALIDA
    * EN ESE CASO SE IRA A LA ETIQUETA ARTINVALIDO.

          MOVE COA MOV TO ART-ART.
          READ ARTICULOS INVALID KEY GO TO ARTINVALIDO.

     * SI LLEGA AQUI ES PORQUE EL ARTICULO EXISTE, SE VA A LA ETIQUETA
     * IMPRIMEPROVEEDOR.

          GO TO IMPRIMEPROVEEDOR.

      ARTINVALIDO.

     * AQUI SE LLEGARA CUANDO EL ARTICULO TECLEADO NO EXISTA, SE AVISA
     * YSE VUELVE A PEDIR.


          DISPLAY " ESE ARTICULO NO EXISTE " LINE 25 POSITION 25
                  REVERSE BLINK BEEP.
          GO TO PIDEARTICULO.

      IMPRIMEPROVEEDOR.

    * SI EL ARTICULO EXISTE SE IMPRIME EL PROVEEDOR Y SE BORRA UN
    * POSIBLE MENSAJE DE LA LINEA 25.

          DISPLAY SPACES SIZE 30 LINE 25 POSITION 25.
          DISPLAY PRO-ART LINE LINEA POSITION 26.

      IMPRIMEPRECIO.

    * SE IMPRIME EL PRECIO QUE EL ARTICULO TIENE EN EL FICHERO.

          DISPLAY PRV-ART LINE LINEA POSITION 50.
          MOVE PRV-ART TO PRE-MOV.

      PIDEPRECIO.

    * Y SE DA OPCION A MODIFICAR ESE PRECIO.

          ACCEPT PRE-MOV LINE LINEA POSITION 50 NO BEEP ECHO UPDATE
                 CONVERT TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSA F1 SE VOVERA A PEDIR EL ARTICULO.
```

```
        IF F1 GO TO PIDEARTICULO.

    * SI ES F2 LA QUE SE PULSA SE VA A FINPROGRAMA.

        IF F2 GO TO FINPROGRAMA.

    * SI LA QUE SE PULSO FUE F3 SE MUEVEN CEROS AL PRECIO, SE IMPRIME
    * Y SE VUELVE A PEDIR.

        IF F3 MOVE ZEROS TO PRE-MOV, DISPLAY PRE-MOV LINE LINEA
           POSITION 50, GO TO PIDEPRECIO.

     PIDECANTIDAD.

    * TRAS EL PRECIO SE PIDE LA CANTIDAD.

        ACCEPT CAN-MOV LINE LINEA POSITION 64 NO BEEP ECHO UPDATE
               CONVERT TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.

    * SI SE PULSA F1 SE VUELVE A PEDIR EL PRECIO.

        IF F1 GO TO PIDEPRECIO.

    * SI SE PULSA F2 SE VA A FINPROGRAMA.

        IF F2 GO TO FINPROGRAMA.

    * SI LA QUE SE PULSO FUE F3 SE MUEVEN CEROS A LA CANTIDAD, SE
    * IMPRIME Y SE VUELVE A PEDIR.

        IF F3 MOVE ZEROS TO CAN-MOV, DISPLAY CAN-MOV LINE LINEA
           POSITION 64, GO TO PIDECANTIDAD.

    * SE CONTROLA QUE LA CANTIDAD QUE SE DA DE SALIDA NO SEA MAYOR
    * QUE EL STOCK ACTUAL DEL ARTICULO, DE SER ASI SE AVISA DE QUE
    * NO HAY SUFICIENTES. Y SE VUELVE A PEDIR.

        IF CAN-MOV > STA-ART DISPLAY " NO HAY SUFICIENTES "
           LINE 25 POSITION 25 REVERSE BLINK BEEP, GO TO
           PIDECANTIDAD.
        DISPLAY SPACES SIZE 30 LINE 25 POSITION 25.

     IMPRIMEIMPORTE.

    * CON EL PRECIO Y LA CANTIDAD SE HALLA EL IMPORTE Y SE IMPRIME.

        MULTIPLY CAN-MOV BY PRE-MOV GIVING IMPORTE.
        DISPLAY IMPORTE LINE LINEA POSITION 72.
```

```
       GRABAMOVIMIENTO.

      * SE MUEVE 1 AL ESTADO PARA COMUNIAR QUE ESTA SIN ACTUALIZAR Y
      * SE GRABA EL MOVIMIENTO.

           MOVE 1 TO EST-MOV.
           WRITE FIC-MOV INVALID KEY REWRITE FIC-MOV.

      * TRAS ELLO SE LE SUMA UNO A LA LINEA DE MOVIMIENTO Y A LA DE
      * PANTALLA.

           ADD 1 TO LIN-MOV.
           ADD 1 TO LINEA.

      * SI LA LINEA DE PANTALLA ES MENOR DE 22 TODAVIA QUEDA ESPACIO
      * PARA MAS ASIENTOS, SE VUELVE A PEDIR OTRO ARTICULO.

           IF LINEA < 22 GO TO PIDEMOVIMIENTO.

      * EN CASO DE QUE SE LLEGUE AQUI ES PORQUE LA PANTALLA ESTA LLENA
      * SE BORRA Y SE MUEVE 10 A LA LINEA PARA EMPEZAR OTRA VEZ A
      * LLENARLA.

           PERFORM BORRA VARYING LINEA FROM 10 BY 1 UNTIL LINEA > 21.
           MOVE 10 TO LINEA.
           GO TO PIDEMOVIMIENTO.

       BORRA.

      * ESTA RUTINA ES LLAMADA PAR BORRAR LA LINEA INDICADA POR LA
      * VARIABLE LINEA.

           DISPLAY SPACES SIZE  3 LINE LINEA POSITION 4.
           DISPLAY SPACES SIZE 15 LINE LINEA POSITION 10.
           DISPLAY SPACES SIZE 20 LINE LINEA POSITION 26.
           DISPLAY SPACES SIZE  5 LINE LINEA POSITION 50.
           DISPLAY SPACES SIZE  4 LINE LINEA POSITION 64.
           DISPLAY SPACES SIZE  8 LINE LINEA POSITION 72.

       IMPRIMIR.

      * CUANDO UN MOVIMIENTO EXISTE SE LLAMA A ESTA RUTINA PARA
      * IMPRIMIR SUS CAMPOS. PARA ELLO SE LEE DEL FICHERO DE ARTICULOS
      * Y SE HACEN LOS CALCULOS NECESARIOS.

           DISPLAY COA-MOV LINE LINEA POSITION 10.
           MOVE COA-MOV TO ART-ART.
```

```
           READ ARTICULOS.
           DISPLAY PRO-ART LINE LINEA POSITION 26.
           DISPLAY CAN-MOV LINE LINEA POSITION 64.
           DISPLAY PRE-MOV LINE LINEA POSITION 50.
           COMPUTE IMPORTE = CAN-MOV * PRE-MOV.
           DISPLAY IMPORTE LINE LINEA POSITION 72.

       FINPROGRAMA.

      * AQUI SE LLEGA CUANDO SE HA PULSADO F2, SE CIERRAN LOS FICHEROS
      * Y SE BORRA UN POSIBLE MENSAJE DE LA LINEA 25.

           CLOSE MOVIMIENTO, ARTICULOS, PROVEEDORES.
           DISPLAY SPACES SIZE 79 LINE 25 POSITION 1.

       SALIRPROGRAMA.

      * SE VUELVE AL PROGRAMA ANTERIOR.

           EXIT PROGRAM.
```

El programa que viene a continuación es el de listado y actualización de diario. Este programa consiste en hacer un listado de todos los movimientos en la fecha actual que no estuvieran ya actualizados, tras este proceso se nos pedirá opción a actualizar o no los movimientos listados, este proceso de actualización consiste en sumar las entradas de artículos a los stocks de los mismo, restar las salidas, actualizar los acumuladores de compras a proveedores y ventas de un artículo.

Seguidamente tienes el listado.

```
123456789012
       A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PGS303.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
```

```
      * SE HACE UN COPY A LAS LIBRERIAS DE SELETCs DE LOS FICHEROS DE
      * PANTALLA, MOVIMIENTO, ARTICULOS Y PROVEEDORES.

           COPY "LGSPANSE.LIB".

           COPY "LGSMOVSE.LIB".

           COPY "LGSARTSE.LIB".

           COPY "LGSPROSE.LIB".

      * SE DECLARA EL FICHERO PARA IMPRESORA.

           SELECT IMP;
                  ASSIGN TO PRINT, "PRINTER";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

      * SE HACE UN COPY A LAS FDs DE LOS MISMOS FICHEROS.

       COPY "LGSPANFD.LIB".

       COPY "LGSMOVFD.LIB".

       COPY "LGSARTFD.LIB".

       COPY "LGSPROFD.LIB".

      * SE DECLARA EL REGISTRO DE IMPRESORA.

       FD  IMP;
           LABEL RECORD IS OMITTED;
           DATA RECORD IS FIC-IMP.

       01  FIC-IMP                                PIC X(80).

       WORKING-STORAGE SECTION.

      * SE DECLARAN LAS CABECERAS Y CUERPOS A USAR EN EL LISTADO.

       01  CABECERA1.
           03  FILLER                             PIC X(68) VALUE "
      -    "                LISTADO DE DIARIO                FECHA..:".
           03  DIA-OUT                            PIC 9(02).
           03  FILLER                             PIC X(01) VALUE "/".
```

```
        03  MES-OUT                    PIC 9(02).
        03  FILLER                     PIC X(01) VALUE "/".
        03  ANO-OUT                    PIC 9(02).

    01  CABECERA2                      PIC X(80) VALUE ALL "-".

    01  CABECERA3.
        03  FILLER                     PIC X(01) VALUE SPACES.
        03  FILLER                     PIC X(03) VALUE "LI.".
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  FILLER                     PIC X(03) VALUE "TI.".
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  FILLER                     PIC X(15) VALUE "ARTICULO".
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  FILLER                     PIC X(05) VALUE "CANT.".
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  FILLER                     PIC X(06) VALUE "PRECIO".
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  FILLER                     PIC X(08) VALUE "IMPORTE".
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  FILLER                     PIC X(10) VALUE "SALIDAS".
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  FILLER                     PIC X(10) VALUE "ENTRADAS".

    01  CUERPO.
        03  FILLER                     PIC X(01) VALUE SPACES.
        03  LIN-OUT                    PIC ZZZ.
        03  FILLER                     PIC X(03) VALUE SPACES.
        03  TIP-OUT                    PIC X(01).
        03  FILLER                     PIC X(03) VALUE SPACES.
        03  ART-OUT                    PIC X(15).
        03  FILLER                     PIC X(03) VALUE SPACES.
        03  CAN-OUT                    PIC ZZ9.
        03  FILLER                     PIC X(03) VALUE SPACES.
        03  PRE-OUT                    PIC ZZZZ9.
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  IMP-OUT                    PIC ZZZZZZZ9.
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  SAL-OUT                    PIC ZZZZZZZZZ9.
        03  FILLER                     PIC X(02) VALUE SPACES.
        03  ENT-OUT                    PIC ZZZZZZZZZ9.

    01  PIE.
        03  FILLER                     PIC X(50) VALUE SPACES.
        03  FILLER                     PIC X(10) VALUE "PAGINA..:".
        03  PAG-OUT                    PIC ZZ.
```

```
      * LOS CAMPOS ENTRADAS Y SALIDAS ACTUARAN COMO ACUMULADORES DE
      * CADA UNO DE LOS CONCEPTOS A LOS QUE REPRESENTAN.

       77  ENTRADAS                        PIC 9(10).
       77  SALIDAS                         PIC 9(10).

      * CON EL CAMPO IMPORTE CALCULAREMOS LOS IMPORTES DE LOS MOVIMIENTOS.

       77  IMPORTE                         PIC 9(08).

      * LOS CAMPOS LINEA Y PAGINA NOS SERVIRAN PARA CONTROLAR AMBOS
      * CONCEPTOS.

       77  LINEA                           PIC 9(02).
       77  PAGINA                          PIC 9(02).

      * LA VARIABLE ERRORES SERA EL FILE STATUS DE LOS FICHEROS.

       77  ERRORES                         PIC X(02).

      * CLV-PAN SERA EL CODIGO DEL FICHERO DE PANTALLA.

       77  CLV-PAN                         PIC 9(01) VALUE 1.

      * EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA A LEER.

       77  NOM-PAN                         PIC X(06).

      * EL CAMPO OPCION NOS SERVIRA PARA PEDIR LA OPCION.

       77  OPCION                          PIC X(01).

       LINKAGE SECTION.

      * EN ESTA SECCION DECLARAMOS EL GRUPO FECHA QUE NOS VIENE DEL
      * PROGRAMA ANTERIOR.

       01  FECHA.
           03  ANO                         PIC 9(02).
           03  MES                         PIC 9(02).
           03  DIA                         PIC 9(02).

       PROCEDURE DIVISION USING FECHA.

      * EN LAS DECLARATIVES AVISAREMOS DE POSIBLES ERRORES EN LOS FICHEROS.

       DECLARATIVES.
```

```
    I-O-MOVIMIENTO SECTION.

  * LA PRIMERA SECCION ES LA DEL FICHERO DE MOVIMIENTO.

        USE AFTER STANDARD ERROR PROCEDURE ON MOVIMIENTO.

    I-O-CREACION1.

        DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
        DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
        DISPLAY "    EN EL FICHERO DE MOVIMIENTO,  LO CREO?.[ ]"
                LINE 25 POSITION 24 REVERSE.

    OPCIONCREACION1.

        ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
        IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT MOVIMIENTO,
           CLOSE MOVIMIENTO,  OPEN I-O MOVIMIENTO,  GO TO FIN1.
        IF OPCION NOT = "N" AND OPCION NOT = "n"
           GO TO OPCIONCREACION1.

    FIN1.

        DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

    END DECLARATIVES.
    PROGRAMA SECTION.

    COMIENZO.

  * SE SACA LA PANTALLA CORRESPONDIENTE.

        MOVE "SGS303" TO NOM-PAN.
        OPEN INPUT PANTALLA.
        READ PANTALLA.
        CLOSE PANTALLA.
        DISPLAY FIC-PAN LINE 1 POSITION 1.

    APERTURA.

  * SE ABRE EL FICHERO DE MOVIMIENTO.

        OPEN I-O MOVIMIENTO.

    PIDEOPCION.

  * SE PIDE EL CORRECTO SI O NO PARA COMENZAR O ABANDONAR EL LISTADO.
```

```
        ACCEPT OPCION LINE 14 POSITION 36 NO BEEP.

    * SI LA OPCION ES N SE VA A LA ETIQUETA FINPROGRAMA.

        IF OPCION = "N" OR OPCION = "n" GO TO FINPROGRAMA.

    * SI LA OPCION NO ES N NI S SE VUELVE A PIDEOPCION.

        IF OPCION NOT = "S" AND OPCION NOT = "s" GO TO PIDEOPCION.

    * SI SE LLEGA AQUI ES PORQUE LA OPCION PULSADA HA SIDO S

     LISTAR.

    * SE MUEVE CEROS A LOS CAMPOS PAGINA, LINEA, IMPORTE, SALIDAS
    * Y ENTRADAS PARA COMENZAR.

        MOVE 0 TO PAGINA, LINEA, IMPORTE, SALIDAS, ENTRADAS.

    * SE MUEVE LA FECHA ACTUAL A LA FECHA DE LA MASCARA DE IMPRESION DE
    * UNA DE LAS CABECERAS.

        MOVE DIA TO DIA OUT.
        MOVE MES TO MES-OUT.
        MOVE ANO TO ANO-OUT.

    * SE MUEVE LA FECHA AL CAMPO FECHA DEL FICHERO DE MOVIMIENTOS PARA
    * LISTAR LOS DEL DIA ACTUAL.

        MOVE FECHA TO FEC-MOV.

    * SE MUEVEN ESPACIOS AL TIPO DE MOVIMIENTO, YA QUE SE HAN DE LISTAR
    * TANTO LAS ENTRADAS COMO LAS SALIDAS.

        MOVE SPACES TO TIP-MOV.

    * SE MUEVE CERO AL NUMERO DE MOVIMIENTO.

        MOVE 0 TO LIN-MOV.

    * SE HACE UN START EN BUSQUEDA DEL PRIMER MOVIMIENTO DE LA FECHA
    * ACTUAL, SI SE PRODUCE CLAVE INVAIDA QUIERE DECIR QUE NO HAY NINGUN
    * MOVIMIENTO A LISTAR, EN ESE CASO SE VA A LA ETIQUETA NOHAY.

        START MOVIMIENTO KEY IS NOT < COD-MOV INVALID KEY
              GO TO NOHAY.

    * SI SE LLEGA AQUI ES PORQUE HAY MOVIMIENTOS PARA LISTAR.
```

```
* SE ABRE EL FICHERO DE IMPRESORA Y SE LLAMA A LA RUTINA DE IMPRESION
* DE CABECERA.

     OPEN OUTPUT IMP.
     PERFORM CABECERA.

 LEER.

* SE LEE EN SECUENCIAL, SI SE LLEGA AL FIN DE FICHERO SE VA A LA
* ETIQUETA FINLISTADO.

     READ MOVIMIENTO NEXT RECORD AT END GO TO FINLISTADO.

* SI LA FECHA DEL MOVIMIENTO QUE SE ACABA DE LEER ES MAYOR QUE LA
* FECHA ACTUAL TAMBIEN SE VA A LA ETIQUETA FINLISTADO.

     IF FEC-MOV > FECHA GO TO FINLISTADO.

* SI EL ESTADO DEL MOVIMIENTO QUE SE ACABA DE LEER ES 2 QUIERE DECIR
* QUE YA ESTABA ACTUALIZADO, POR LO TANTO NO LO LISTA, SE VUELVE A

* LEER OTRO.

     IF EST-MOV = 2 GO TO LEER.

* SI SE LLEGA A ESTE PUNTO ES PORQUE EL MOVIMIENTO LEIDO ES VALIDO
* PARA LA IMPRESION, SE PASAN TODOS LOS CAMPOS DEL FICHERO A LOS
* DEL CUERPO DEL FICHERO DE IMPRESORA.

     MOVE LIN-MOV TO LIN-OUT.
     MOVE TIP-MOV TO TIP-OUT.
     MOVE COA-MOV TO ART-OUT.
     MOVE CAN-MOV TO CAN-OUT.
     MOVE PRE-MOV TO PRE-OUT.

* SE CALCULA EL IMPORTE DEL MOVIMIENTO LEIDO.

     COMPUTE IMPORTE = CAN-MOV * PRE-MOV.
     MOVE IMPORTE TO IMP-OUT.

* SI EL TIPO DEL MOVIMIENTO ES E ESE IMPORTE SE SUMA AL ACUMULADOR
* DE ENTRADAS, EN CASO CONTRARIO AL DE SALIDAS.

     IF TIP-MOV = "E" ADD IMPORTE TO ENTRADAS ELSE
                      ADD IMPORTE TO SALIDAS.
     MOVE ENTRADAS TO ENT-OUT.
     MOVE SALIDAS TO SAL-OUT.
```

```
    * SE IMPRIME EL MOVIMIENTO.

        WRITE FIC-IMP FROM CUERPO.

    * SE LE SUMA UNO A LA LINEA

        ADD 1 TO LINEA.

    * SI LA LINEA ES MAYOR DE 57 SE LLAMA A LA RUTINA DE IMPRESION
    * DEL PIE Y DE LA CABECERA.

        IF LINEA > 57 PERFORM PIE, PERFORM CABECERA.

    * VUELVE A SEGUIR LEYENDO MOVIMIENTOS.

        GO TO LEER.

     CABECERA.

    * CADA VEZ QUE SE LLAMA A ESTA ETIQUETA SE IMPRIMEN LAS DISTINTAS
    * CABECERAS Y SE MUEVE 4 A LA  LINEA.

        WRITE FIC-IMP FROM CABECERA1.
        PERFORM LINBLAN
        WRITE FIC-IMP FROM CABECERA3.
        WRITE FIC-IMP FROM CABECERA2.
        MOVE 4 TO LINEA.

     PIE.

    * CADA VEZ QUE SE LLAMA AL PIE SE SALTA HASTA EL FIN DE PAGINA,

        COMPUTE LINEA = 62 - LINEA.
        PERFORM LINBLAN LINEA TIMES.

    * SE LE SUMA UNO A LA PAGINA Y SE IMPRIME EL PIE,

        ADD 1 TO PAGINA.
        MOVE PAGINA TO PAG-OUT.
        WRITE FIC IMP FROM PIE.

    * Y SE SALTA A LA SIGUIENTE PAGINA.

        PERFORM LINBLAN 3 TIMES.

     LINBLAN.

    * ESTA RUTINA ES LLAMADA PARA IMPRIMIR UNA LINEA EN BLANCO CADA VEZ.
```

```
           MOVE SPACES TO FIC-IMP.
           WRITE FIC-IMP.

       FINLISTADO.

      * CUANDO SE LLEGA A EL FIN DEL LISTADO SE LLAMA A LA RUTINA DE
      * IMPRESION DEL PIE Y DESPUES SE CIERRA EL FICHERO DE IMPRESORA.

           PERFORM PIE.
           CLOSE IMP WITH NO REWIND.

       ACTUALIZACION.

      * UNA VEZ SE HA LISTADO SE NOS PREGUNTA SI QUEREMOS ACTUALIZAR
      * LOS MOVIMIENTOS QUE SE ACABAN DE LEER.

           DISPLAY "ACTUALIZO?.[ ]" LINE 12 POSITION 25
           ACCEPT OPCION LINE 12 POSITION 37 NO BEEP.

      * SI LA OPCION ES N SE VA A LA ETIQUETA FINPROGRAMA.

           IF OPCION = "N" OR OPCION = "n" GO TO FINPROGRAMA.

      * SI LA OPCION NO ES N NI S SE VUELVE A PEDIR.

           IF OPCION NOT = "S" AND OPCION NOT = "s" GO TO ACTUALIZACION.

      * EN CASO DE QUE SE LLEGUE AQUI ES PORQUE LA OPCION TECLEADA
      * HA SIDO S.

      * SE MUEVE LA FECHA ACTUAL A LA FECHA DEL FICHERO DE MOVIMIENTOS.

           MOVE FECHA TO FEC-MOV.

      * SE MUEVEN ESPACIOS AL TIPO DE MOVIMIENTO.

           MOVE SPACES TO TIP-MOV.

      * SE MUEVE UN 0 A LA LINEA DE MOVIMIENTO,

           MOVE 0 TO LIN-MOV.

      * Y SE HACE UN START PARA POSICIONARNOS DE NUEVO EN EL PRIMER
      * MOVIMIENTO.

           START MOVIMIENTO KEY IS NOT < COD-MOV.

      * SE ABREN LOS FICHEROS DE ARTICULOS Y PROVEEDORES.
```

APLICACION DE GESTION DE SUPERMERCADOS

```
       OPEN I-O ARTICULOS, PROVEEDORES.

    LEEACTUALIZA.

  * SE LEE EN SECUENCIAL, CUANDO SE LLEGUE AL FINAL DE FICHERO SE
  * IRA A LA ETIQUETA FINACTUALIZA.

       READ MOVIMIENTO NEXT RECORD AT END GO TO FINACTUALIZA.

  * SI LA FECHA DEL MOVIMIENTO QUE SE ACABA DE LEER ES MAYOR QUE
  * LA FECHA ACTUAL SE VA A LA ETIQUETA FINACTUALIZA.

       IF FEC-MOV > FECHA GO TO FINACTUALIZA.

  * SI EL ESTADO DEL MOVIMIENTO ES 2 ES PORQUE YA ESTA ACTUALIZADO
  * POR LO TANTO NO SE ACTUALIZA OTRA VEZ, SE VUELVE A LEER OTRO
  * MOVIMIENTO.

       IF EST-MOV = 2 GO TO LEEACTUALIZA.

  * CUANDO SE LLEGUE AQUI ES PORQUE EL MOVIMIENTO ES ACTUALIZABLE.

  * SE MUEVE EL ARTICULO A LA CLAVE DEL FICHERO DE ARTICULOS Y SE LEE.

       MOVE COA-MOV TO ART-ART.
       READ ARTICULOS.

  * SI EL STOCK ACTUAL Y EL ACUMULADOR NO SON NUMERICOS SE LES MUEVE
  * CERO, ESTO PUEDE OCURRIR EN LA PRIMERA ACTUALIZACION CUANDO DICHOS
  * CAMPOS NO SE HAN ESCRITO NINGUNA VEZ AUN.

       IF STA-ART NOT NUMERIC MOVE 0 TO STA-ART.
       IF ACU-ART NOT NUMERIC MOVE 0 TO ACU-ART.

  * SE MUEVE EL PROVEEDOR A LA CLAVE DEL FICHERO DE PROVEEDORES Y SE
  * LEE DE DICHO FICHERO.

       MOVE PRO-ART TO PRO-PRO.
       READ PROVEEDORES.

  * SI EL ACUMULADOR NO ES NUMERICO SE LE MUEVE CERO.

       IF ACU-PRO NOT NUMERIC MOVE 0 TO ACU-PRO.

  * SE LE CALCULA EL IMPORTE AL MOVIMIENTO,

       COMPUTE IMPORTE = CAN-MOV * PRE-MOV.
```

```
* SI EL TIPO DEL MOVIMIENTO ES ENTRADA SE LE SUMA DICHA ENTRADA
* AL STOCK ACTUAL, Y SE LE SUMA EL IMPORTE AL ACUMULADOR DE COMPRAS
* DEL PROVEEDOR QUE NOS LO VENDE.

     IF TIP-MOV = "E" COMPUTE STA-ART = STA-ART + CAN-MOV,
                      COMPUTE ACU-PRO = ACU-PRO + IMPORTE.

* SI EL TIPO DEL MOVIMIENTO ES S SE LE RESTA LA SALIDA AL
* STOCK ACTUAL Y SE LE SUMA EL IMPORTE AL ACUMULADOR DE VENTAS
* DICHO ARTICULO.

     IF TIP-MOV = "S" COMPUTE STA-ART = STA-ART - CAN-MOV,
                      COMPUTE ACU-ART = ACU-ART + IMPORTE.

* SE MUEVE 2 AL ESTADO DE MOVIMIENTO PARA QUE SE SEPA QUE YA ESTA
* ACTUALIZADO.

     MOVE 2 TO EST-MOV.

* SE REESCRIBEN LOS REGISTROS DE MOVIMIENTO, ARTICULOS Y PROVEEDORES.

     REWRITE FIC-MOV.
     REWRITE FIC-ART.
     REWRITE FIC-PRO.

* SE VUELVE A SEGUIRLEYENDO.

     GO TO LEEACTUALIZA.

 FINACTUALIZA.

* CUANDO SE HA TERMINADO DE ACTUALIZAR SE CIERRAN LOS FICHEROS DE
* ARTICULOS Y PROVEEDORES Y SE VA A LA ETIQUETA FINPROGRAMA.

     CLOSE ARTICULOS, PROVEEDORES.
     GO TO FINPROGRAMA.

 NOHAY.

* SI SE LLEGA AQUI ES PORUQE AL COMENZAR A LISTAR NO SE ENCONTRO
* NINGUN MOVIMIENTO, SE NOS AVISA DE ELLO, SE ESPERA Y SE LLEGA A
* LA ETIQUETA FINPROGRAMA.

     DISPLAY " NO HAY MOVIMIENTOS EN LA FECHA ACTUAL "
             LINE 25 POSITION 15 REVERSE BLINK BEEP.
     PERFORM ESPERA 2000 TIMES.
     DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.
```

```
       FINPROGRAMA.

      * AQUI SE CIERRA EL FICHERO DE MOVIMIENTO.

           CLOSE MOVIMIENTO.

       SALIRPROGRAMA.

      * Y SE VUELVE AL PROGRAMA ANTERIOR.


           EXIT PROGRAM.

       ESPERA.

      * ESTA ETIQUETA ES LLAMADA PARA CAUSAR UNA ESPERA, LO QUE SE HACE
      * ES MOVER EL CAMPO OPCION A SI MISMO PARA PERDER TIEMPO.

           MOVE OPCION TO OPCION.
```

El programa que seguiría es el de listado de artículos bajo stock mínimo, este programa es muy parecido al de listado de artículos, la diferencia está en que no se listan todos los artículos, sino sólo aquellos en los cuales su stock actual es menor que su stock mínimo. El programa es fácil de entender y vá acompañado de comentarios.

```
123456789012
       A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PGS401.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

      * SE HACE UN COPY A LAS LIBRERIAS DE SELECTs DE LOS FICHEROS DE
      * PANTALLA, ARTICULOS Y PROVEEDORES.
```

APLICACION DE GESTION DE SUPERMERCADOS

```
           COPY "LGSPANSE.LIB".

           COPY "LGSARTSE.LIB".

           COPY "LGSPROSE.LIB".

      * SE DECLARA EL FICHERO DE IMPRESORA.


          SELECT IMP;
                 ASSIGN TO PRINT, "PRINTER";
                 FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

      * SE HACE UN COPY A LAS FDs DE LOS MISMOS FICHEROS.

       COPY "LGSPANFD.LIB".

       COPY "LGSARTFD.LIB".

       COPY "LGSPROFD.LIB".

      * SE DECLARA EL REGISTRO DE IMPRESORA.

       FD  IMP;
           LABEL RECORD IS OMITTED;
           DATA RECORD IS FIC-IMP.

       01  FIC-IMP                              PIC X(80).

       WORKING-STORAGE SECTION.

      * SE DECLARAN LAS CABECERAS Y CUERPOS A USAR EN EL LISTADO.

       01  CABECERA1.
           03  FILLER                           PIC X(71) VALUE "  CENTRO DE
      -    " ESTUDIOS INFORMATICOS   GESTION DE SUPERMERCADOS     FECHA:
      -    "".
           03  DIA-OUT                          PIC ZZ.
           03  FILLER                           PIC X(01) VALUE "/".
           03  MES-OUT                          PIC ZZ.
           03  FILLER                           PIC X(01) VALUE "/".
           03  ANO-OUT                          PIC ZZ.

       01  CABECERA2                            PIC X(80) VALUE ALL "-".
```

```
01  CABECERA3.
    03  FILLER                          PIC X(02) VALUE SPACES.
    03  FILLER                          PIC X(15) VALUE "ARTICULO".
    03  FILLER                          PIC X(02) VALUE SPACES.
    03  FILLER                          PIC X(20) VALUE "PROVEEDOR".
    03  FILLER                          PIC X(02) VALUE SPACES.
    03  FILLER                          PIC X(06) VALUE "P.COS.".
    03  FILLER                          PIC X(02) VALUE SPACES.
    03  FILLER                          PIC X(06) VALUE "P.VEN.".
    03  FILLER                          PIC X(02) VALUE SPACES.
    03  FILLER                          PIC X(05) VALUE "ST.A.".
    03  FILLER                          PIC X(03) VALUE SPACES.
    03  FILLER                          PIC X(15) VALUE "ST.M.".

01  CUERPO.
    03  FILLER                          PIC X(02) VALUE SPACES.
    03  ART-OUT                         PIC X(15).
    03  FILLER                          PIC X(02) VALUE SPACES.
    03  PRO-OUT                         PIC X(20).
    03  FILLER                          PIC X(03) VALUE SPACES.
    03  PRC-OUT                         PIC ZZZZ9.
    03  FILLER                          PIC X(03) VALUE SPACES.
    03  PRV-OUT                         PIC ZZZZ9.
    03  FILLER                          PIC X(03) VALUE SPACES.
    03  STA-OUT                         PIC ZZ9.
    03  FILLER                          PIC X(04) VALUE SPACES.
    03  STM-OUT                         PIC ZZ9.

01  PIE.
    03  FILLER                          PIC X(50) VALUE "LISTADO DE
-   "ARTICULOS BAJO STOCK MINIMO.".
    03  FILLER                          PIC X(07) VALUE "PAGINA:".
    03  PAG-OUT                         PIC ZZ.

* LOS CAMPOS PAGINA Y LINEA NOS SERVIRAN PARA CONTROLAR ESTOS
* DOS CONCEPTOS.

77  PAGINA                              PIC 9(02).
77  LINEA                               PIC 9(02).

* LA VARIABLE ERRORES SERA EL FILE STATUS DE LOS FICHEROS.

77  ERRORES                             PIC X(02).

* EL CAMPO CLV-PAN ES LA CLAVE DEL FICHERO DE PANTALLA.

77  CLV-PAN                             PIC 9(01) VALUE 1.
```

```
      * EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA A LEER.

       77  NOM-PAN                          PIC X(06).

      * LA VARIABLE OPCION NOS SERVIRA PARA PEDIR OPCION.

       77  OPCION                           PIC X(01).

      * EN EL GRUPO EXCEP SE DECLARAN LOS CODIGOS DE LAS TECLAS QUE
      * VAN A CAUSAR EXCEPCION.

       01  EXCEP                            PIC 9(02) COMP.
           88  F1 VALUE IS 00.
           88  F2 VALUE IS 01.
           88  F3 VALUE IS 02.

       LINKAGE SECTION.

      * EN ESTA SECCION SE DECLARA EL GRUPO FECHA QUE NOS VIENE DEL
      * PROGRAMA ANTERIOR.

       01  FECHA.
           03  ANO                          PIC 9(02).
           03  MES                          PIC 9(02).
           03  DIA                          PIC 9(02).

       PROCEDURE DIVISION USING FECHA.

      * EN LAS DECLARATIVES AVISAREMOS DE LOS POSIBLES ERRORES QUE
      * PUEDAN OCURRIR EN LOS FICHEROS.

       DECLARATIVES.

       I-O-ARTICULOS SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON ARTICULOS.

       I-O-CREACION1.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "    EN EL FICHERO DE ARTICULOS. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION1.

           ACCEPT OPCION LINE 25 POSITION 66 REVERSE.
```

```
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT ARTICULOS,
              CLOSE ARTICULOS, OPEN I-O ARTICULOS, GO TO FIN1.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION1.

       FIN1.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I-O-PROVEEDOR SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.

       I-O-CREACION2.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "    EN EL FICHERO DE PROVEEDORES. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION2.

           ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
              CLOSE PROVEEDORES, OPEN I-O PROVEEDORES, GO TO FIN2.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION2.

       FIN2.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I-O-IMP SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON IMP.

       I-O-AVISO.

           DISPLAY " CONECTE LA IMPRESORA Y PULSE UNA TECLA "
                   LINE 25 POSITION 15 REVERSE BLINK BEEP.
           ACCEPT OPCION OFF NO BEEP.
           WRITE FIC-IMP.
           DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.

       END DECLARATIVES.
       PROGRAMA SECTION.

       COMIENZO.
```

```
      * SACAMOS LA MASCARA EN PANTALLA.

           MOVE "SGS401" TO NOM-PAN.
           OPEN INPUT PANTALLA.
           READ PANTALLA.
           CLOSE PANTALLA.
           DISPLAY FIC-PAN LINE 1 POSITION 1.

      * SE IMPRIME LA FECHA.

           DISPLAY DIA LINE 23 POSITION 60.
           DISPLAY "/" LINE 23 POSITION 62.
           DISPLAY MES LINE 23 POSITION 63.
           DISPLAY "/" LINE 23 POSITION 65.
           DISPLAY ANO LINE 23 POSITION 66.
           MOVE DIA TO DIA-OUT.
           MOVE MES TO MES-OUT.
           MOVE ANO TO ANO-OUT.

       APERTURA.

      * SE ABREN LOS FICHEROS DE ARTICULOS Y PROVEEDORES.

           OPEN I-O ARTICULOS, PROVEEDORES.

       PEDIROPCION.

      * PEDIMOS EL CORRECTO SI O NO PARA EMPEZAR A LISTAR.

           ACCEPT OPCION LINE 14 POSITION 36 NO BEEP.

      * SI LA OPCION ES S SE VA A LA ETIQUETA LISTAR.

           IF OPCION = "S" OR OPCION = "s" GO TO LISTAR.

      * SI LA OPCION NO ES NI S NI N SE VUELVE A PEDIR.

           IF OPCION NOT = "N" AND OPCION NOT = "n" GO TO PEDIROPCION.

       FINPROGRAMA.

      * SI SE LLEHA AQUI ES PORQUE SE HA PULSADO N, SE CIERRAN LOS
      * FICHEROS,

           CLOSE ARTICULOS, PROVEEDORES.

       SALIRPROGRAMA.
```

```
    * Y SE SALE DEL PROGRAMA.

        EXIT PROGRAM.

     LISTAR.

    * CUANDO SE PULSE S SE LLEGARA  A ESTE PUNTO.

    * SE MUEVEN ESPACIOS A LA CLAVE DEL FICHERO DE ARTICULOS.

        MOVE SPACES TO ART-ART.

    * SE HACE UN START, SI DA CLAVE INVALIDA ES QUE NO HAY NINGUN
    * ARTICULO, EN ESE CASO SE VA A LA ETIQUETA NOHAY.

        START ARTICULOS KEY IS NOT < ART-ART INVALID KEY GO TO NOHAY.

    * SI HAY ARTICULOS SE ABRE EL FICHERO DE IMPRESORA, SE MUEVE 0 A
    * LA LINEA Y A LA PAGINA PARA COMENZAR Y SE LLAMA A LA RUTINA DE
    * IMPRESION DE CABECERA.

        OPEN OUTPUT IMP.
        MOVE 0 TO LINEA, PAGINA.
        PERFORM CABECERA.

     LEER.

    * SE LEE EN SECUENCIAL HASTA QUE SE LLEGUE AL FIN DE FICHERO, EN
    * ESE MOMENTO SE VA A LA ETIQUETA FINLISTADO.

        READ ARTICULOS NEXT RECORD AT END GO TO FINLISTADO.

    * SI EL STOCK ACTUAL DEL ARTICULO QUE SE ACABA DE LEER ES MAYOR O
    * IGUAL AL STOCK MINIMO VOLVEMOS A SEGUIR LEYENDO.

        IF STA-ART > STM-ART OR STA-ART = STM-ART GO TO LEER.

    * SI SE LLEHA AQUI ES PORQUE EL STOCK ACTUAL ES MENOR QUE EL STOCK
    * MINIMO, POR LO TANTO HAY QUE LISTARLO.

    * SE PASAN LOS CAMPOS DEL FICHERO A LOS DEL CUERPO DE IMPRESION.
        MOVE ART-ART TO ART-OUT.
        MOVE PRO-ART TO PRO-OUT.
        MOVE PRC-ART TO PRC-OUT.
        MOVE PRV-ART TO PRV-OUT.
        MOVE STA-ART TO STA-OUT.
        MOVE STM-ART TO STM-OUT.
```

```
* SE IMPRIME.

     WRITE FIC-IMP FROM CUERPO.
     WRITE FIC-IMP FROM CABECERA2.

* SE ACTUALIZA EL CAMPO LINEA.

     ADD 2 TO LINEA.

* SI LA LINEA ES MAYOR DE 57 SE LLAMA A LA RUTINA DE IMPRESION DE
* PIE Y CABECERA.

     IF LINEA > 57 PERFORM PIE, PERFORM CABECERA.

* SE VUELVE A SEGUIR LEYENDO.

     GO TO LEER.

 CABECERA.

* CADA VEZ QUE SE LLAMA A ESTA RUTINA SE IMPRIMEN LAS CABECERAS
* Y LA LINEA SE PONE CON EL VALOR 4.

     WRITE FIC-IMP FROM CABECERA1.
     WRITE FIC-IMP FROM CABECERA2.
     WRITE FIC-IMP FROM CABECERA3.
     PERFORM LINBLAN.
     MOVE 4 TO LINEA.

 PIE.

* CADA VEZ QUE SE LLAMA A ESTA RUTINA SE SALTA HASTA EL FIN DE PAGINA

     COMPUTE LINEA = 64 - LINEA - 2.
     PERFORM LINBLAN LINEA TIMES.

* SE LE SUMA UNO A LA PAGINA

     ADD 1 TO PAGINA.
     MOVE PAGINA TO PAG-OUT.

* Y SE IMPRIME EL PIE,

     WRITE FIC-IMP FROM PIE.

* DESPUES SE SALTA A LA SIGUIENTE PAGINA.
```

```
           PERFORM LINBLAN 3 TIMES.

       FINLISTADO.

      * CUANDO SE LLEGA AL FIN DE LISTADO SE LLAMA AL PIE Y SE CIERRA
      * EL FICHERO DE IMPRESORA, TRAS ELLO SE VA A LA ETIQUETA
      * FINPROGRAMA.

           PERFORM PIE.
           CLOSE IMP WITH NO REWIND.
           GO TO FINPROGRAMA.

       NOHAY.

      * AQUI SE LLEGA CUANDO EN EL FICHERO NO HAY NINGUN ARTICULO, SE
      * AVISA Y SE VA A LA ETIQUETA FINPROGRAMA.

           DISPLAY " NO HAY NINGUN ARTICULO EN EL FICHERO "
                   LINE 25 POSITION 15 REVERSE BLINK BEEP.
           PERFORM ESPERA 5000 TIMES.
           DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.
           GO TO FINPROGRAMA.

       LINBLAN.

      * ESTA RUTINA ES LLAMADA PARA IMPRIMIR UNA LINEA EN BLANCO.

           MOVE SPACES TO FIC IMP.
           WRITE FIC IMP.

       ESPERA.

      * A ESTA ETIQUETA SE LLAMA PARA CAUSAR UNA ESPERA, LO QUE SE
      * HACE ES MOVER EL CAMPO OPCION A SI MISMO PARA PERDER TIEMPO.

           MOVE OPCION TO OPCION.
```

A continuación tenemos el último programa de la aplicación, este es el de listado de artículos por proveedor, este difiere del listado normal en que en lugar de pedir artículo de inicio y fin se pide proveedor de inicio y fin, y al hacer el START para leer no especificamos la clave principal sino que ponemos la clave alternativa, una vez hecho esto podremos comenzar a leer en secuencial. El programa lo tienes bajo estas líneas.

```
123456789012
      A   B

       IDENTIFICATION DIVISION.
       PROGRAM-ID. PGS402.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. PC.
       OBJECT-COMPUTER. PC.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

      * SE HACE UN COPY A LAS LIBRERIAS SELECTs DE LOS FICHEROS DE
      * PANTALLA, ARTICULOS Y PROVEEDORES.

           COPY "LGSPANSE.LIB".

           COPY "LGSARTSE.LIB".

           COPY "LGSPROSE.LIB".

      * SE DECLARA EL FICHERO DE IMPRESORA.

           SELECT IMP;
                  ASSIGN TO PRINT, "PRINTER";
                  FILE STATUS IS ERRORES.

       DATA DIVISION.
       FILE SECTION.

      * SE HACE UN COPY A LAS FDs DE LOS FICHEROS CORRESPONDIENTES.

       COPY "LGSPANFD.LIB".

       COPY "LGSARTFD.LIB".

       COPY "LGSPROFD.LIB".

      * SE DECLARA EL REGISTRO DEL FICHERO DE IMPRESORA.

       FD  IMP;
           LABEL RECORD IS OMITTED;
           DATA RECORD IS FIC-IMP.

       01  FIC-IMP                          PIC X(80).
```

```
   WORKING-STORAGE SECTION.

  * SE DECLARAN LAS DISTINTAS CABECERAS Y CUERPOS QUE VAN A INTERVENIR
  * EN EL LISTADO.

   01  CABECERA1.

       03  FILLER                          PIC X(71) VALUE "  CENTRO DE
       " ESTUDIOS INFORMATICOS   GESTION DE SUPERMERCADOS     FECHA:
       "".
       03  DIA-OUT                         PIC ZZ.
       03  FILLER                          PIC X(01) VALUE "/".
       03  MES-OUT                         PIC ZZ.
       03  FILLER                          PIC X(01) VALUE "/".
       03  ANO-OUT                         PIC ZZ.

   01  CABECERA2                           PIC X(80) VALUE ALL "-".

   01  CABECERA3.
       03  FILLER                          PIC X(02) VALUE SPACES.
       03  FILLER                          PIC X(15) VALUE "ARTICULO".
       03  FILLER                          PIC X(02) VALUE SPACES.
       03  FILLER                          PIC X(20) VALUE "PROVEEDOR".
       03  FILLER                          PIC X(02) VALUE SPACES.
       03  FILLER                          PIC X(06) VALUE "P.COS.".
       03  FILLER                          PIC X(02) VALUE SPACES.
       03  FILLER                          PIC X(06) VALUE "P.VEN.".
       03  FILLER                          PIC X(02) VALUE SPACES.
       03  FILLER                          PIC X(05) VALUE "ST.A.".
       03  FILLER                          PIC X(03) VALUE SPACES.
       03  FILLER                          PIC X(10) VALUE "ACUMULADO".

   01  CUERPO.
       03  FILLER                          PIC X(02) VALUE SPACES.
       03  ART-OUT                         PIC X(15).
       03  FILLER                          PIC X(02) VALUE SPACES.
       03  PRO-OUT                         PIC X(20).
       03  FILLER                          PIC X(03) VALUE SPACES.
       03  PRC-OUT                         PIC ZZZZ9.
       03  FILLER                          PIC X(03) VALUE SPACES.
       03  PRV-OUT                         PIC ZZZZ9.
       03  FILLER                          PIC X(03) VALUE SPACES.
       03  STA-OUT                         PIC ZZ9.
       03  FILLER                          PIC X(04) VALUE SPACES.
       03  ACU-OUT                         PIC ZZZZZZZZZ9.

   01  PIE.
       03  FILLER                          PIC X(50) VALUE "LISTADO DE
```

```
           "ARTICULOS.".
           03  FILLER                          PIC X(07) VALUE "PAGINA:".
           03  PAG-OUT                         PIC ZZ.

       * PAGINA Y LINEA NOS VAN A SERVIR PARA CONTROLAR ESTOS DOS
       * CONCEPTOS.

        77  PAGINA                             PIC 9(02).
        77  LINEA                              PIC 9(02).

       * PRI-ART Y ULT-ART NOS VAN A SERVIR PARA CONTROLAR DESDE DONDE
       * HASTA DONDE SE QUIERE HACER EL LISTADO.

        77  PRI-ART                            PIC X(20).
        77  ULT-ART                            PIC X(20).

       * LA VARIABLE ERRORES ES EL FILE STATUS DE LOS FICHEROS

        77  ERRORES                            PIC X(02).

       * CLV-PAN SERA EL CODIGO DEL FICHERO DE PANTALLA.

        77  CLV-PAN                            PIC 9(01) VALUE 1.

       * EN NOM-PAN PONDREMOS EL NOMBRE DE LA PANTALLA A LEER.

        77  NOM-PAN                            PIC X(06).

       * LA VARIABLE OPCION NOS SERVIRA PARA PEDIR OPCION.

        77  OPCION                             PIC X(01).

       * EN EL GRUPO EXCEP SE DECLARAN LOS CODIGOS DE LAS TECLAS QUE
       * CAUSARAN EXCEPCION.

        01  EXCEP                              PIC 9(02) COMP.
            88  F1 VALUE IS 00.
            88  F2 VALUE IS 01.
            88  F3 VALUE IS 02.

        LINKAGE SECTION.

       * EN ESTA SECCION SE DECLARA EL GRUPO FECHA QUE NOS VIENE DEL
       * PROGRAMA ANTERIOR.

        01  FECHA.
            03  ANO                            PIC 9(02).
```

```
           03  MES                              PIC 9(02).
           03  DIA                              PIC 9(02).

       PROCEDURE DIVISION USING FECHA.

      * EN LAS DECLARATIVES AVISAREMOS DE POSIBLES ERRORES QUE SE
      * PRODUZCAN EN LOS FICHEROS.

       DECLARATIVES.

       I-O-ARTICULOS SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON ARTICULOS.

       I-O-CREACION1.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "    EN EL FICHERO DE ARTICULOS. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION1.

           ACCEPT OPCION LINE 25 POSITION 66 REVERSE.
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT ARTICULOS,
              CLOSE ARTICULOS, OPEN I O ARTICULOS, GO TO FIN1.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION1.

       FIN1.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I O PROVEEDOR SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON PROVEEDORES.

       I O CREACION2.

           DISPLAY " SE PRODUCE UN ERROR " LINE 25 POSITION 1 REVERSE.
           DISPLAY ERRORES LINE 25 POSITION 22 REVERSE.
           DISPLAY "    EN EL FICHERO DE PROVEEDORES. LO CREO?.[ ]"
                   LINE 25 POSITION 24 REVERSE.

       OPCIONCREACION2.

           ACCEPT OPCION LINE 25 POSITION 68 REVERSE.
```

```
           IF OPCION = "S" OR OPCION = "s" OPEN OUTPUT PROVEEDORES,
              CLOSE PROVEEDORES, OPEN I O PROVEEDORES, GO TO FIN2.
           IF OPCION NOT = "N" AND OPCION NOT = "n"
              GO TO OPCIONCREACION2.

       FIN2.

           DISPLAY SPACES SIZE 70 LINE 25 POSITION 1.

       I-O-IMP SECTION.

           USE AFTER STANDARD ERROR PROCEDURE ON IMP.

       I-O-AVISO.

           DISPLAY " CONECTE LA IMPRESORA Y PULSE UNA TECLA "
                   LINE 25 POSITION 15 REVERSE BLINK BEEP.
           ACCEPT OPCION OFF NO BEEP.
           WRITE FIC-IMP.
           DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.

       END DECLARATIVES.
       PROGRAMA SECTION.

       COMIENZO.

      * SE SACA LA MASCARA CORRESPONDIENTE.

           MOVE "SGS402" TO NOM-PAN.
           OPEN INPUT PANTALLA.
           READ PANTALLA.
           CLOSE PANTALLA.
           DISPLAY FIC-PAN LINE 1 POSITION 1.

      * SE IMPRIME LA FECHA EN PANTALLA.

           DISPLAY DIA LINE 23 POSITION 60.
           DISPLAY "/" LINE 23 POSITION 62.
           DISPLAY MES LINE 23 POSITION 63.
           DISPLAY "/" LINE 23 POSITION 65.
           DISPLAY ANO LINE 23 POSITION 66.

      * SE MUEVE LA FECHA ACTUAL A LA FECHA DE UNA DE LAS MASCARAS
      * DE IMPRESION.

           MOVE DIA TO DIA-OUT.
           MOVE MES TO MES-OUT.
           MOVE ANO TO ANO-OUT.
```

```
       APERTURA.
      * SE ABREN LOS FICHEROS DE ARTICULOS Y PROVEEDORES.
           OPEN I-O ARTICULOS, PROVEEDORES.
       PIDEPRIMERO.
      * SE PIDE EL PRIMER ARTICULO A LISTAR POR PROVEEDOR.
           ACCEPT PRI-ART LINE 10 POSITION 38 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.
      * SI SE PULSA F2 SE PASA A PEDIROPCION
           IF F2 GO TO PEDIROPCION.
      * SI SE PULSO F3 SE MUEVEN ESPACIOS A PRI-ART, SE IMPRIME Y SE
      * VUELVE A PEDIR.
           IF F3 MOVE SPACES TO PRI-ART, DISPLAY PRI-ART LINE 18
                 POSITION 38, GO TO PIDEPRIMERO.
       PIDEULTIMO.
      * SE PIDE EL ULTIMO A LISTAR.
           ACCEPT ULT-ART LINE 12 POSITION 38 NO BEEP ECHO UPDATE
                  TAB ON EXCEPTION EXCEP MOVE OPCION TO OPCION.
      * SI SE HA PULSADO TAB SE MUEVEN ZETAS AL CAMPO PARA QUE SE
      * LISTE HASTA EL FINAL DEL FICHERO.
           IF ULT-ART = SPACES MOVE "ZZZZZZZZZZZZZZZZZZZZ" TO ULT-ART,
              DISPLAY ULT-ART LINE 12 POSITION 38.
      * SI SE PULSA F1 SE VUELVE A PEDIR EL PRIMERO.
           IF F1 GO TO PIDEPRIMERO.
      * SI SE PULSO F3 SE MUEVEN ESPACIOS AL CAMPO, SE IMPRIME Y SE
      * VUELVE A PEDIR.
           IF F3 MOVE SPACES TO ULT-ART, DISPLAY ULT-ART LINE 12
              POSITION 38, GO TO PIDEULTIMO.
       PEDIROPCION.
```

```
    * DESPUES DE PEDIR EL PRINCIPIO Y FIN DEL LISTADO SE PIDE OPCION.

        ACCEPT OPCION LINE 17 POSITION 49 NO BEEP.

    * SI SE PULSA C SE VA A LA  ETIQUETA LISTAR.

        IF OPCION = "C" OR OPCION = "c" GO TO LISTAR.

    * SI SE PULSO I SE VUELVEN A PEDIR LOS LIMITES.

        IF OPCION = "I" OR OPCION = "i" GO TO PIDEPRIMERO.

    * SI LA OPCION NO ES S SE VUELVE A PEDIR LA OPCION.

        IF OPCION NOT = "S" AND OPCION NOT = "s" GO TO PEDIROPCION.

     FINPROGRAMA.

    * SI SE LLEGA A ESTE PUNTO ES PORUQE SE PULSO S. SE CIERRAN LOS
    * FICHEROS DE ARTICULOS Y PROVEEDORES.

        CLOSE ARTICULOS, PROVEEDORES.

     SALIRPROGRAMA.

    * Y SE SALE DEL PROGRAMA.

        EXIT PROGRAM.

     LISTAR.

    * AQUI SE LLEGARA CUANDO SE HALLA PULSADO C.

    * SE MUEVE EL PRIMER ARTICULO A LISTAR A EL CAMPO PRO ART.

        MOVE PRI ART TO PRO ART.

    * SE HACE UN START CON PRO ART QUE ES LA CLAVE SECUNDARIA DEL
    * FICHERO DE ARTICULOS, SI SE PRODUCE CLAVE INVLIDA SE VA A LA
    * ETIQUETA NOHAY.

        START ARTICULOS KEY IS NOT < PRO ART INVALID KEY GO TO NOHAY.

    * SI EXISTE ALGUN REGISTRO A PARTIR DEL ESPECIFICADO SE ABRE EL
    * FICHERO DE IMPRESORA,  SE MUEVE 0 A LA LINEA Y A LA PAGINA.

        OPEN OUTPUT IMP.
        MOVE 0 TO LINEA, PAGINA.
```

```
    * SE LLAMA A LA RUTINA DE IMPRESION DE CABECERA.

        PERFORM CABECERA.

     LEER.

    * SE LEE EN SECUENCIAL,  SI SE LLEGA AL FINAL DEL FICHERO SE VA A LA
    *.ETIQUETA FINLISTADO.

        READ ARTICULOS NEXT RECORD AT END GO TO FINLISTADO.

    * SI EL PROVEEDOR DEL REGISTRO QUE SE ACABA DE LEER ES MAYOR QUE
    * EL TOPE PUESTO,  DE SER ASI SE VA A FINLISTADO.

        IF PRO-ART > ULT-ART GO TO FINLISTADO.

    * SI SE LLEGA AQUI ES QUE EL REGISTRO SE PUEDE IMPRIMIR,  SE
    * PASAN TODOS LOS CAMPOS DEL FICHERO A LOS DEL CUERPO DE IMPRESION.

        MOVE ART-ART TO ART-OUT.
        MOVE PRO-ART TO PRO-OUT.
        MOVE PRC-ART TO PRC-OUT.
        MOVE PRV-ART TO PRV-OUT.
        MOVE STA-ART TO STA-OUT.
        MOVE ACU-ART TO ACU-OUT.

    * SE IMPRIME.

        WRITE FIC-IMP FROM CUERPO.
        WRITE FIC-IMP FROM CABECERA2.

    * SE LE SUMA 2 A LA LINEA.

        ADD 2 TO LINEA.

    * SI LA LINEA ES MAYOR DE 57 SE LLAMA A LA IMPRESION DE PIE Y
    * CABECERA.

        IF LINEA > 57 PERFORM PIE,  PERFORM CABECERA.

    * DESPUES SE VUELVE A SEGUIR LEYENDO.

        GO TO LEER.

     CABECERA.

    * CADA VEZ QUE SE LLAMA A ESTA RUTINA SE IMPRIMEN LAS DISTINTAS
    * CABECERAS Y SE MUEVE 4 A LA LINEA.
```

```
        WRITE FIC-IMP FROM CABECERA1.
        WRITE FIC-IMP FROM CABECERA2.
        WRITE FIC-IMP FROM CABECERA3.
        PERFORM LINBLAN.
        MOVE 4 TO LINEA.

     PIE.

    * CADA VEZ QUE SE LLAMA AL PIE SE VA HASTA EL FINAL DE PAGINA.

        COMPUTE LINEA = 64 - LINEA - 2.
        PERFORM LINBLAN LINEA TIMES.

    * SE LE SUMA 1 A LA PAGINA.

        ADD 1 TO PAGINA.
        MOVE PAGINA TO PAG-OUT.

    * SE IMPRIME EL PIE Y SE SALTA HASTA LA PROXIMA PAGINA.

        WRITE FIC-IMP FROM PIE.
        PERFORM LINBLAN 3 TIMES.

     FINLISTADO.

    * CUANDO SE LLEGA AQUI ES QUE SE HA TERMINADO EL LISTADO, SE LLAMA
    * A LA RUTINA DE IMPRSION DEL PIE Y SE CIERRA EL FICHERO DE
    * IMPRESORA.

        PERFORM PIE.
        CLOSE IMP WITH NO REWIND.

    * DESPUES SE VA A LA ETIQUETA FINPROGRAMA.

        GO TO FINPROGRAMA.

     NOHAY.

    * AQUI SE LLEGA CUANDO NO HA REGISTROS PARA LISTAR, SE NOS AVISA
    * Y SE VA A LA ETIQUETA FINPROGRAMA.

        DISPLAY " NO HAY NINGUN REGISTRO A PARTIR DEL INDICADO "
                LINE 25 POSITION 15 REVERSE BLINK BEEP.
        PERFORM ESPERA 5000 TIMES.
        DISPLAY SPACES SIZE 60 LINE 25 POSITION 15.
        GO TO FINPROGRAMA.
```

```
LINBLAN.

* ESTA RUTINA SE UTILIZA PARA IMPRIMIR LINEAS EN BLANCO.

    MOVE SPACES TO FIC-IMP.
    WRITE FIC-IMP.

ESPERA.

* ESTA RUTINA SE EMPLEA PARA PERDER TIEMPO, LO QUE SE HACE ES
* MOVER EL CAMPO OPCION A SI MISMO.

    MOVE OPCION TO OPCION.
```

15.- ERRORES RELATIVOS A LA UTILIZACION DE FICHEROS.

En este apéndice de errores podrás encontrar los errores de ejecución de un programa. Los siguiente errores son referentes a la utilización de ficheros.

ERRORES EN FICHEROS SECUENCIALES.

10 AT END.- Se produce al ejecutar un **READ** y no haber un registro lógico para leer, es decir, que se ha llegado al final del fichero.

30 PERMANENT ERROR.- En un error de E/S que se produce como resultado en un error de transmisión de datos, porque se produzca un error de paridad en el chequeo de datos, también se produce ante la ejecución de una instrucción no implementada.

34 PERMANENT ERROR BOUNDARY VIOLATION.- Se produce como resultado de una violación del fichero secuencial. Si ocurre al ejecutar la instrucción **OPEN** es que no hay memoria suficiente para abrir el fichero.

90 INVALID OPERATION.- Se produce al ejecutar una instrucción **READ, WRITE** o **REWRITE** que entra en conflicto con el modo en que ha sido abierto el fichero. También ocurre cuando se ejecuta un **REWRITE** que no ha sido precedido por un **READ.**

ERRORES RELATIVOS A FICHEROS

91 **FILE NOT OPENED.**- Este error se produce cuando se intenta ejecutar una instrucción **DELETE, READ, START, UNLOCK, WRITE, REWRITE** o **CLOSE** y el fichero en cuestión no está abierto.

92 **FILE NOT CLOSED.**- Este error se produce al ejecutar un **OPEN** y el fichero que se intentó abrir está ya abierto.

93 **FILE NOT AVAILABLE.**- Se produce al intentar abrir un fichero que fué cerrado con la opción **WITH LOCK.** También se produce a menudo cuando se vé imposibilitado para abrir el número de ficheros que se indican.

94 **INVALID OPEN.**- Se produce cuando el fichero que se intenta abrir no tiene correspondencia externa, también ocurre cuando algún parámetro es inválido, por ejemplo el nombre del fichero.

95 **INVALID DEVICE.**- Se produce al intentar operar con un fichero que se ha declarado de una forma conflictiva, por ejemplo si intentamos declarar el fichero de impresora como **RANDOM.**

96 **UNDEFINED CURRENT RECORD POINTER STATUS.**- Este error se produce cuando tras haber llegado al final del fichero se intenta ejecutar una instrucción **READ.**

97 **INVALID RECORD LENGTH.**- Se produce cuando se intenta reescribir un registro con una longitud mayor a la que tenía, también cuando se intenta escribir un registro con más longitud de la declarada, esto suele ocurrir cuando en el registro hay caracteres gráficos o caracteres españoles tales como eñe y vocales acentuadas.

98 **INVALID DATA IN RECORD.**- Este error se produce cuando en el registro existen una serie de caracteres no standard en el código ASCII.

99 **RECORD LOCKED.** Se produce cuando se intenta leer un registro que está bloqueado por otra estación.

FICHEROS RELATIVOS E INDEXADOS.

02 **SUCCESSFUL OPERATION BUT KEY HAS DUPLICATE.** Se produce en los ficheros indexados cuando se intenta escribir un registro con el mismo nombre de otro y se duplica la clave. En las claves alternativas esto tiene solución con el párrafo **WITH DUPLICATES,** en la clave principal es imposible tener duplicaciones.

ERRORES RELATIVOS A FICHEROS

10 **AT END.**- Se produce cuando se ha ido leyendo en secuencial y se ha llegado al final del fichero intentando leer un registro inexistente.

21 **SEQUENCE ERROR FOR A SEQUENTIALLY ACCESSED INDEXED FILE.**- Se produce cuando el orden ascendente de lectura secuencial ha sido violado o cuando se ha intentado hacer un **REWRITE** tras un **READ** habiendo cambiado el campo clave.

22 **DUPLICATE KEY VALUE.**- Se produce al ejecutar una instrucción **WRITE** con un registro que crea una clave duplicada.

23 **NO RECORD FOUND.**- Este error se produce cuando se intenta encontrar un registro inexistente, bien sea con **READ** o **START**.

24 **BOUNDARY VIOLATION.** Se produce al ejecutar un **WRITE** fuera de los límites de la definición externa del fichero.

30 **PERMANENT ERROR.**- Es un error de E/S que se produce como resultado en un error de transmisión de datos, porque se produzca un error de paridad en el chequeo de datos, también se produce ante la ejecución de una instrucción no implementada.

34 **PERMANENT ERROR INSUFFICIENT MEMORY.**- Una instrucción **OPEN** no puede ser ejecutada debido a que no hay memoria disponible para la apertura de más ficheros.

90 **INVALID OPERATION.**- Se produce al ejecutar una instrucción **READ**, **WRITE** o **REWRITE** que entra en conflicto con el modo en que ha sido abierto el fichero. También ocurre cuando se ejecuta un **REWRITE** que no ha sido precedido por un **READ**.

91 **FILE NOT OPENED.**- Este error se produce cuando se intenta ejecutar una instrucción **DELETE**, **READ**, **START**, **UNLOCK**, **WRITE**, **REWRITE** o **CLOSE** y el fichero en cuestión no está abierto.

92 **FILE NOT CLOSED.**- Este error se produce al ejecutar un **OPEN** y el fichero que se intentó abrir está ya abierto.

93 **FILE NOT AVAILABLE.**- Se produce al intentar abrir un fichero que fué cerrado con la opción **WITH LOCK**. También se produce a menudo cuando se vé imposibilitado para abrir el número de ficheros que se indican.

94 **INVALID OPEN.**- Se produce cuando el fichero que se intenta abrir no tiene correspondencia externa, también ocurre cuando algún parámetro es inválido, por ejemplo el nombre del fichero.

ERRORES RELATIVOS A FICHEROS

95 **INVALID DEVICE.**- Se produce al intentar operar con un fichero que se ha declarado de una forma conflictiva, por ejemplo si intentamos declarar el fichero de impresora como **RANDOM.**

96 **UNDEFINED CURRENT RECORD POINTER STATUS.**- Este error se produce cuando tras haber llegado al final del fichero se intenta ejecutar una instrucción **READ.**

97 **INVALID RECORD LENGTH.**- Se produce cuando se intenta reescribir un registro con una longitud mayor a la que tenía, también cuando se intenta escribir un registro con más longitud de la declarada, esto suele ocurrir cuando en el registro hay caracteres gráficos o caracteres españoles tales como eñe y vocales acentuadas.

98 **INVALID INDEX.**- Ocurre al trabajar con un fichero indexado en el cual el índice se ha desorganizado y el programa no es capaz de encontrar la clave solicitada.

ERRORES DE COMPILACION

16.- **ERRORES DE COMPILACION.**

Seguidamente tienes los errores de compilación más normales que nos podemos encontrar.

W 7: AREA A LEVEL.- Se ha encontrado un nivel 01 o 77 en el margen B, cuando estos niveles siempre han de comenzar en el margen A.

W 11: DATA DIVISION ORDER.- La cabecera indicadora de división, de sección, de párrafo o el indicador de nivel no están en el orden requerido por el compilador.

W 12: DECLARATIVE REFERENCE.- Se hace referencia en las DECLARATIVES a un nombre de párrafo que no está en las mismas. Si el párrafo se ha definido en otro lugar de la **PROCEDURE** se ejecutará correctamente.

W 13: END PROGRAM EXPECTED.- La palabra clave END no lleva a continuación la palabra PROGRAM, tampoco es interpretada por END DECLARATIVES ni como la instrucción AT END, por lo cual nos devuelve este aviso.

W 14: ENVIRONMENT DIVISION ORDER.- Las cabeceras de división,

sección o párrafo en la **ENVIRONMENT** no están en el orden requerido por el compilador.

W 15: FILE EXTERNAL NAME SIZE.- El nombre externo especificado en la cláusula **ASSIGN TO** excede de treinta caracteres. Solamente el primero de esos treinta caracteres será usado para acceder al fichero.

W 17: FILE RELATIVE KEY DATA TYPE.- La clave relativa declarada para la indicación de un fichero es numérica con signo. El standard requiere que sea numérica sin signo. El programa puede ser ejecutado pero si se dan claves negativas puede tener resultados no deseados.

W 18: FILE STATUS DATA SECTION.- El file status declarado para un fichero ha sido definido en la **FILE SECTION**. El standard no admite esta situación. El programa podrá ser ejecutado pero tendrá resultados inesperados.

W 19: GROUP VALUE CLASH.- La utilización de la cláusula **VALUE** es inválida a nivel de grupo.

W 20: IDENTIFICATION DIVISION ORDER.- La cabecera de división de la **IDENTIFICATION** no está en el orden requerido por el compilador.

W 21: INDICATOR AREA CHARACTER.- Se ha encontrado un caracter en la columna siete que no es identificado como uno de los caracteres válidos en esa columna, el caracter será omitido y tratado como un espacio en blanco.

W 22: INTENGER VALUE.- El entero indicado requiere más de dieciséis bits para la representación de su valor. La implementación de Cobol está limitada a dieciséis bits para la representación de algunos valores enteros. Se tomarán los dieciseís bits más bajos para ser utilizados.

W 23: LABEL CLASH.- La cláusula **LABEL RECORD OMITTED** ha sido especificada para un fichero asignado al tipo **RANDOM**. No tiene consecuencias en el programa objeto.

W 24: LITERAL SIZE.- El literal entre los indicadores de comillas no tiene ninguna longitud.

W 27: MULTIPLE PERIODS.- Se han encontrado dos puntos consecutivos en el programa fuente. No tiene repercusión en el programa objeto.

W 28: **MULTIPLE PUNCTUATION.**- Dos caracteres de puntuación, comas o punto y comas, se han encontrado consecutivos en el programa fuente. No afecta al programa objeto.

W 29: **NULL LITERAL.**- Las literales han de contener al menos un caracter.

W 30: **OPTION DATA TYPE CLASH.**- El parámetro de opción indicado para el dato asociado es inválido. La opción es ignorada.

W 31: **OPTION MULTIPLE DECLARATION.**- Se han especificado una o más opciones incompatibles, la segunda y siguientes de haberlas serán ignoradas.

W 32: **PROCEDURE NAME REQUIRED.**- Se requiere un nombre de párrafo en el area A debido a que la última instrucción fué de transferencia de control como **GO TO** o **STOP RUN.**

W 33: **PROCEDURE NAME SYNTAX.**- El nombre de párrafo contiene un punto decimal. El punto decimal es ignorado.

W 34: **RANDOM DEVICE ASSUMED.**- La cláusula **ASSIGN TO** no ha sido declarada para especificar el tipo de un fichero, el tipo **RANDOM** es asumido.

W 37: **RESERVED WORD CONFLICT.**- La palabra usada es alguna de las palabras reservadas en el Cobol Ansi 1974 pero no es reservada en esta implementación de Cobol. No tiene ningún efecto sobre el programa objeto.

W 38: **SEPARATOR REQUIRED.**- Un literal y una palabra definida por el usuario no tiene separador entre ellos.

W 39: **SEPARATOR SPACE.**- Se encuentra una coma o punto y coma sin un espacio a continuación. Los separadores siempre han de llevar un espacio a continuación.

W 40: **SUBSCRIPT ARITHMETIC.**- El standard permite la suma o resta en enteros declarados como índices de una tabla, pero no en subíndices.

W 41: **SYMBOL SIZE.**- Se ha definido una palabra que excede de treinta caracteres que es el máximo permitido.

W 45: **VERB IN AREA A.**- Se ha encontrado alguna instrucción en el area A en el programa fuente. La instrucción es tratada como si estuviera en el area B.

ERRORES DE COMPILACION

E 48: **ALL LITERAL SIZE.**- El literal especificado para la constante figurativa **ALL** contiene más de un caracter.

E 52: **ALTER MODIFIER DECLARATIVE.**- Una instrucción **ALTER** es inválida a causa de modificar un nombre de párrafo de las DECLARATIVES.

E 53: **ALTER MODIFIER QUALIFICATION.**- Una instrucción **ALTER** es inválida a causa de modificar un nombre de párrafo ambigüo. Es requerida una cualificación para producir una única referencia.

E 54: **ALTER MODIFIER UNDEFINED.**- Una instrucción **ALTER** es inválida por modificar un nombre de párrafo indefinido.

E 60: **ANS LITERAL CONTINUATION.**- Un literal alfanumérico se continua en la siguiente línea, pero no se han abierto unas nuevas comillas para continuarlo.

E 61: **ANS LITERAL REQUIRED.**- El contexto de la instrucción requiere un literal alfanumérico.

E 62: **ANS LITERAL SIZE.**- Una literal alfanumérica excede de 2047 caracteres en la longitud especificada en el programa fuente.

E 63: **ANS LITERAL TERMINATION.**- Una literal alfanumérica no se continua en la siguiente línea porque no hay guión en la columna siete y no se han cerrado las comillas al final de la línea.

E 64: **AREA A ENTRY REQUIRED.**- El contexto require un párrafo en el area A en el punto indicado en el programa fuente por el compilador.

E 65: **ASSIGN REQUIRED.**- La cláusula **ASSIGN** es requerida para cada fichero definido.

E 66: **BLANK WHEN ZERO CLASH.**- La cláusula **BLANK WHEN ZERO** solo se puede utilizar en descripciones de campos elementales numéricos del tipo **DISPLAY.**.

E 67: **CLASS DATA TYPE CLASH.**- El grupo condicional especificado entra en conflicto con el tipo de dato del elemento que ha sido indicado.

E 68: **COMPOSITE SIZE.** La composición de un operando numérico excede de dieciocho dígitos.

E 69: CONDITION-NAME DEFINED. El nombre condicional ya está definido y no se puede redefinir.

E 70: CONDITION-NAME ILLEGAL.- El contexto no admite un nombre condicional pero el indentificador indicado es un nombre condicional.

E 71: CONDITION-NAME VALUE TYPE. El valor especificado para un nombre condicional entra en conflicto con el tipo de la variable condicional asociada.

E 72: CONDITIONAL OPERATOR REQUIRED. El contexto requiere un operador relacional como >, < o u otra clase de operador como NUMERIC O ALPHABETIC en el punto indicado.

E 73: CONDITIONAL VARIABLE INVALID.- El nombre condicional indicado es inválido debido a un error de compilación en la descripción del mismo.

E 74: COPY STATEMENT.- La síntaxis del comando COPY indicado es incorrecta o falta el punto al final de la línea.

E 75: COPY TEXT.- El fichero indicado por la instrucción COPY es inaccesible.

E 76: CORRESPONDING GROUP REFERENCE.- El contexto requiere que el identificador a usar con la cláusula CORRESPONDING sea un nombre de grupo.

E 77: CORRESPONDING NUMERICS REQUIRED. Dos grupos han sido indicados para hacer una suma o resta con la cláusula CORRESPONDING y ninguno de sus elementos numéricos coinciden. Se necesita que halla al menos un elemento numérico coincidente.

E 78: CURRENCY LITERAL VALUE. El literal especificado en la cláusula CURRENCY SIGN es un caracter no permitido para este fin.

E 79: DATA DIVISION SYNTAX.- El elemento indicado es incorrecto en la síntaxis de la DATA DIVISION.

E 80: DATA ITEM REFERENCE.- El contexto requiere una referencia a un elemento de datos pero el identificador indicado no es un elemento de datos.

E 82: DATA MULTIPLE DESCRIPTION.- El indicador de descripción es repetido más de una vez para un mismo dato.

E 83: DATA NAME DEFINED.- Se ha creado una palabra de usuario que ya existía. Se necesita una cualificación para crear una referencia única.

E 85: DATA TYPE ALPHANUMERIC REQUIRED.- El contexto requiere un elemento de datos alfanumérico.

E 86: DATA TYPE CHARACTER REQUIRED.- El contexto requiere que se define una caracter para las reglas del funcionamiento de la instrucción INSPECT.

E 87: DATA TYPE CHARACTER SIZE.- Se ha especificado más de un caracter para una operación que requiere sólo uno. Solamente el primer caracter será usado.

E 88: DATA TYPE DISPLAY REQUIRED.- El contexto requiere que el elemento de datos sea del tipo DISPLAY

E 89: DATA TYPE NUMERIC INTEGER REQUIRED.- El contexto requiere un elemento de datos que sea un numérico entero.

E 90: DATA TYPE NUMERIC OR NSE REQUIRED.- El contexto requiere que el elemento de datos sea un campo numérico o numérico de edición.

E 91: DATA TYPE NUMERIC REQUIRED.- El contexto requiere un elemento de datos numérico, un elemento numérico de edición no está permitido.

E 92: DATA VALUE TYPE.- El valor especificado en la cláusula VALUE para un dato entra en conflicto con el tipo del mismo.

E 93: DATE, DAY OR TIME EXPECTED.- Al utilizar la opción FROM de la instrucción ACCEPT sólo está permitido indicar DATE, DAY o TIME.

E 94: DECLARATIVES REFERENCE.- Una instrucción GO TO o ALTER hace referencia a un nombre de párrafo ilegal dentro de las DECLARATIVES.

E 95: DECLARATIVES SECTION REQUIRED.- Las DECLARATIVES han de comenzar con la definición de una sección.

E 97: ELEMENTARY LEVEL.- El dato declarado antes del actual es un campo elemental y sin embargo el actual intenta ser una división de él.

E 98: **ENVIRONMENT DIVISION SYNTAX.**- El elemento indicado es incorrecto en la síntaxis de la **ENVIRONMENT DIVISION.**

E 99: **FILE ACCESS TYPE.**- Se ha indicado un tipo de acceso irreconocible para un fichero.

E 100: **FILE CONTROL DUPLICATE.**- La cláusula **FILE-CONTROL** es repetida para el mismo fichero.

E 101: **FILE DESCRIPTION.**- El fichero descrito para el nombre de fichero indicado ha sido omitido o es inválido. No se ha definido el registo para el fichero.

E 103: **FILE EXTERNAL NAME DATA TYPE.**- El nombre declarado como externo para un fichero no es alfanumérico.

E 104: **FILE EXTERNAL NAME INVALID.**- El campo declarado para el nombre externo de un fichero es inválido por un error en la descripción.

E 108: **FILE EXTERNAL NAME QUALIFIERS.**- El campo declarado para nombre externo de un fichero hace referencia a dos o más campos por lo cual es inválido.

E 109: **FILE EXTERNAL NAME UNDEFINED.**- El campo declarado como nombre externo de un fichero es indefinido.

E 110: **FILE LABEL DESCRIPTOR.**- La cláusula **LABEL RECORD** especificada es irreconocible.

E 117: **FILE MULTIPLE DEFINITION.**- El indicador de nombre de un fichero vuelve a ser encontrado en la FD cuando ya está definido.

E 118: **FILE MULTIPLE DESCRIPTION.**- La cláusula indicadora de descripción de fichero es repetida para un mismo fichero.

E 119: **FILE-NAME DEFINED.**- La palabra indicada para un nombre de fichero ya fué utilizada para otros propósitos.

E 120: **FILE-NAME ILLEGAL.**- El contexto no permite la referencia a este nombre de fichero.

E 121: **FILE-NAME INVALID.** El nombre de fichero indicado es inválido debido á un error en la descripción del mismo.

E 122: **FILE-NAME REQUIRED.** El contexto requiere el nombre de fichero.

ERRORES DE COMPILACION

E 123: FILE-NAME UNDEFINED.- El nombre de fichero indicado no ha sido definido.

E 124: FILE ORGANIZATION TYPE.- Se ha especificado un modo de organización irreconocible.

E 126: FILE RECORD KEY DATA LOCATION.- El nombre declarado como RECORD KEY en la definición de un fichero indexado no ha sido definido en el grupo de registro de ese fichero.

E 128: FILE RECORD KEY DATA SIZE.- El nombre declarado como RECORD KEY en la definición de un fichero indexado excede de 255 caracteres.

E 129: FILE RECORD KEY DATA TYPE.- El nombre declarado como RECORD KEY en la definición de un fichero indexado no es alfanumérico.

E 130: FILE RECORD KEY INVALID.- El nombre declarado como RECORD KEY en la descripción de un fichero indexado es inválido por un error en la descripción.

E 134: FILE RECORD KEY QUALIFIERS.- El nombre declarado como RECORD KEY en la definición de un fichero indexado hace referencia a dos o más campos y por lo tanto es inválido.

E 135: FILE RECORD KEY UNDEFINED.- El nombre declarado como RECORD KEY en la definición de un fichero indexado no ha sido definido.

E 136: FILE RECORD SIZE DECLARATION.- La longitud declarada bajo la cláusula RECORD CONTAINS de un fichero no coincide con la longitud real del registro.

E 137: FILE RELATIVE KEY DATA LOCATION.- El nombre declarado como RELATIVE KEY en la definición de un fichero directo ha sido declarado como parte del grupo de datos.

E 138: FILE RELATIVE KEY DATA TYPE.- El nombre declarado como RELATIVE KEY en la declaración de un fichero directo no es numérico entero.

E 139: FILE RELATIVE KEY INVALID.- El nombre declarado como RELATIVE KEY en la descripción de un fichero directo es inválido por causa de un error en la descripción del mismo.

E 143: FILE RELATIVE KEY QUALIFIERS.- El nombre declarado

como **RELATIVE KEY** en la definición de un fichero directo hace referencia a dos o más campos por lo cual es inválido.

E 144: FILE RELATIVE KEY UNDEFINED.- El nombre declarado como **RELATIVE KEY** en la definición de un fichero directo no ha sido definido.

E 145: FILE STATUS DATA SIZE.- El campo especificado como file status de un fichero no tiene dos caracteres de longitud en su declaración.

E 146: FILE STATUS DATA TYPE.- El campo especificado como file status de un fichero es de tipo no alfanumérico.

E 147: FILE STATUS INVALID.- El campo especificado como file status de un fichero es inválido debido a un error en la descripción del mismo.

E 151: FILE STATUS QUALIFIERS.- El campo declarado como file status de un fichero hace referencia a más de un campo por lo cual es inválido.

E 152: FILE STATUS UNDEFINED.- El campo declarado como file status de un fichero no ha sido definido.

E 153: FILLER LEVEL.- La cláusula **FILLER** ha sido usada con un campo a un nivel no elemental.

E 154: GO TO NOT LAST.- En una condición las instrucciones **GO TO** y **STOP RUN** han de ser las últimas en especificarse.

E 155: GROUP LEVEL.- El campo declarado previamente a este no es un campo elemental y sin embargo el actual tiene un número de nivel menor o igual a él.

E 157: GROUP USAGE CLASH.- Una cláusula **USAGE** contradice a otra cláusula **USAGE** declarada para el grupo en conjunto.

E 158: GROUP VALUE DECLARED.- La cláusula **VALUE** no puede ser utilizada a nivel de grupo.

E 159: IDENTIFICATION DIVISION SYNTAX.- El elemento indicado es incorrecto en la síntaxis de la **IDENTIFICATION DIVISION.**

E 160: IDENTIFIER INVALID.- El identificador indicado es inválido debido a un error en su descripción.

ERRORES DE COMPILACION

E 161: **IDENTIFIER QUALIFICATION.** El identificador referido hace referencia a dos o más elementos, la referencia es ambigüa.

E 162: **IDENTIFIER REQUIRED.** El contexto requiere un identificador en lugar de un literal.

E 163: **IDENTIFIER UNDEFINED.** El identificador indicado no ha sido difinido.

E 171: **INTEGER REQUIRED.** El contexto requiere un entero.

E 172: **INTEGER SIGN INVALID.** El contexto requiere un entero sin signo.

E 173: **INTEGER ZERO INVALID.** El contexto requiere un entero que no sea cero.

E 174: **IO FILE-NAME REQUIRED.** El contexto requiere el nombre de fichero que este propiamente definido en la cláusula **FD** de dicho fichero.

E 175: **IO FILE RECORD REQUIRED.** El contexto requiere el nombre de registro asociado con el fichero.

E 176: **JUSTIFY CLASH.** La cláusula **JUSTIFIED** usada en la descripción del dato entra en conflicto con otra cláusula de descripción usada en el mismo campo.

E 177: **LEVEL 77 NOT ELEMENTARY.** Se ha utilizado el nivel 77 como no elemental, este nivel siempre ha de ser elemental.

E 178: **LEVEL NUMBER INVALID.** El número de nivel indicado es inválido.

E 182: **LITERAL REQUIRED.** El contexto requiere un literal.

E 183: **MOVE ANSE OR ABS TO NSE OR NUMERIC.** La instrucción **MOVE** es inválida debido a que se intenta mover un campo de edición alfanumérico o un campo alfabético a un campo de edición numérico o un campo numérico.

E 185: **MOVE NONINTEGER TO NONNUMERIC.** Una instrucción **MOVE** es inválida debido a que se intenta pasar un campo no entero a un campo alfanumérico.

E 186: **MOVE NSE TO ABS.** Una instrucción **MOVE** es inválida ya que se intenta pasar un campo numérico de edición a un campo alfanumérico.

ERRORES DE COMPILACION

E 187: **MOVE NSE TO NSE OR NUMERIC.**- Una instrucción **MOVE** es inválida al intentar pasar un campo numérico de edición a otro campo numérico de edición a un campo numérico normal.

E 188: **MOVE NUMERIC TO ABS.**- Una instrucción **MOVE** es inválida al intentar pasar un campo numérico a un campo alfabético.

E 189: **NESTED CONDITIONAL STATEMENT.**- El condicional indicado es inválido a causa de estar anidado con otra instrucción condicional. Solamente la instrucción IF se puede anidar.

E 191: **NUMERIC LITERAL SIZE.**- El literal numérico indicado sobrepasa los dieciocho dígitos de longitud.

E 195: **OCCURS LEVEL.**- La cláusula **OCCURS** no se puede usar a niveles 01 ni 77.

E 196: **OCCURS VALUE CLASH.**- Un elemento de datos ha sido descrito utilizando las cláusulas **OCCURS** y **VALUE** situación que es inválida.

E 197: **OPEN MODE TYPE.**- El tipo de apertura indicado en una instrucción **OPEN** es irreconocible.

E 199: **PARAGRAPH-NAME DEFINED.**- El nombre indicado ya ha sido definido para usar como nombre de párrafo o sección.

E 200: **PARAGRAPH-NAME UNDEFINED.**- El nombre de párrafo especificado no existe.

E 203: **PERFORM ENTRY QUALIFICATION.**- La instrucción **PERFORM** es inválida a causa de que el nombre de párrafo es ambigüo, se hace necesaria una cualificación para crear una referencia única.

E 204: **PERFORM ENTRY UNDEFINED.**- La etiqueta indicada para una instrucción **PERFORM** no ha sido definida en el programa.

E 206: **PERFORM EXIT QUALIFICATION.**- La instrucción **PERFORM** es inválida a causa de que la etiqueta especificada para la instrucción **EXIT** es ambigüa.

E 207: **PERFORM EXIT UNDEFINED.**- La instrucción **PERFORM** especificada es inválida ya que la etiqueta especificada para la instrucción **EXIT** no ha sido definida.

E 208: **PERIOD SPACE REQUIRED.**- El contexto requiere un punto y un espacio de separación en el punto indicado.

E 210: PICTURE CHARACTER INVALID.- El caracter indicado en la definición de un campo es inválido.

E 220: PICTURE NUMERIC DIGIT SIZE.- La cadena de descripción para definir un campo numérico o numérico de edición excede de los dieciocho dígitos.

E 227: PROCEDURE NAME QUALIFICATION.- El nombre de párrafo indicado es ambigüo.

E 228: PROCEDURE NAME UNDEFINED.- El nombre de párrafo indicado no ha sido definido aún cuando se ha hecho referencia a él en el programa.

E 229: PROCEDURE SECTION REQUIRED.- Al terminar las DECLARATIVES en un programa es necesario continuar con un nombre de sección.

E 234: RANDOM ACCESS ILLEGAL.- El contexto no permite la definición del fichero con acceso RANDOM.

E 236: READ KEY ILLEGAL. El parámetro KEY es ilegal en una instrucción READ a causa de que el fichero es secuencial o se está leyendo en secuencial.

E 237: RECORD ENTRY LEVEL-NUMBER.- El número de nivel para la definición de un registro no es el 01.

E 238: RECORD ENTRY REQUIRED.- El contexto requiere la definición de el registro en el punto indicado

E 239: RECORD KEY COUNT.- Mas de catorce claves han sido especificadas para un fichero.

E 240: RECORD KEY INVALID. La cláusula RECORD KEY es inválida ya que el fichero no es de organización indexada.

E 243: REDEFINES NAME. El nombre de datos referenciado por la cláusula REDEFINES es inválida por no ser el anterior a él mismo, por referirse a un campo definido con la cláusula OCCURS o por referirse a un registro de fichero en la FILE SECTION.

E 246: RELATION DISPLAY DATA REQUIRED. La relación indicada hace referencia a un campo que ha de ser del tipo DISPLAY.

E 247: RELATION ILLEGAL.- El contexto no permite la relación indicada.

E 249: RELATION OPERATOR REQUIRED.- El contexto requiere un operador de relación.

E 250: RELATIVE KEY INVALID. - El parámetro RELATIVE KEY es inválido en un fichero cuya organización no es directa.

E 251: RELATIVE KEY REQUIRED.- La clave relativa es requerida al usar la instrucción START.

E 259: RESERVED COMPUTER NAME.- El nombre indicado para nombre de ordenador es una palabra reservada.

E 262: RIGHT PARENTHESIS ")" MISSING.- El paréntesis de cierre es requerido en la posición indicada.

E 272: SEQUENTIAL FILE ORG ILLEGAL.- El contexto no permite un fichero de organización secuencial.

E 273: SEQUENTIAL FILE ORG REQUIRED.- El contexto requiere un fichero de organización secuencial.

E 279: START RELATIVE KEY.- El nombre de campo usado en la instrucción START no hace referencia al campo clave.

E 282: SUBSCRIPT REQUIRED.- El campo necesita de un índice.

E 288: SYMBOL LETTER REQUIRED.- El nombre de datos indicado ha de contener una letra al menos.

E 289: SYNCHRONIZED CLASH.- La cláusula SYNC especificada entra en conflicto con otra cláusula utilizada para la descripción del campo.

E 290: TABLE DEPENDING DATA TYPE.- El nombre de campo especificado en la cláusula DEPENDING ON de OCCURS hace referencia a un campo que no es un numérico entero.

E 295: TABLE DEPENDING QUALIFIER.- El nombre de campo indicado hace referencia a dos o más campos por lo cual es inválido.

E 296: TABLE DEPENDING UNDEFINED.- El nombre de campo especificado en la cláusula DEPENDING ON no ha sido definido.

E 299: USAGE TYPE.- Se ha usado un tipo irreconocible en la cláusula USAGE IS.

ERRORES DE COMPILACION

E 301: VALUE CLASH. - La cláusula **VALUE** ha sido usada en conflicto con otra cláusula.

E 303: VALUE LITERAL SIZE. - La cadena que se intenta asignar por medio de la cláusula **VALUE** excede de la longitud del campo.

E 304: VERB REQUIRED. - El contexto requiere un verbo en el punto indicado

E 543: SYNTAX ERROR, EXPECTED WORD: "palabra". El contexto requiere la aparición de la palabra indicada entre comillas.

INDICE ALFABETICO

< 68
= 68
> 68
ACCEPT 33
ACCEPT 42
ACCESS MODE 108
ADD 52
AFTER ADVANCING 135
ALTER 63
ALTERNATE RECORD KEY 108
AND 68
Asignar 32
AT END 125
AT END 158
BEEP 51
BEFORE ADVANCING 135
Bifurcar 36
BLANK WHEN ZERO 24
BLINK 43
BLINK 51
BLOCK CONTAINS 113
Calcular 39
Cálculos 52
CALL 102
CANCEL 106
CLOSE 128
COBOL 10
CODE SET 114
COMP 24
COMP 1 24
COMP 2 24
COMP 3 24
Compilación 26
COMPUTATIONAL 24
COMPUTE 39
COMPUTE 58
CONFIGURATION SECTION 11
Constantes figurativas 14
Constantes y variables 14
CONVERT 44
COPY 163
CORRESPONDING 53
CORRESPONDING 78
CURRENCY SIGN 21
DATA DIVISION 11
DATA DIVISION 18
DATA RECORD 114

INDICE ALFABETICO

DATE 44
DATE WRITTEN 16
DAY 44
DECIMAL POINT IS COMMA 18
DECIMAL POINT IS COMMA 22
Decisión 37
DECLARATIVES 177
DELETE 139
DEPENDING ON 61
DISPLAY 24
DISPLAY 29
DISPLAY 51
DIVIDE 58
DIVISIONES 10
DYNAMIC 108
ECHO 44
ELSE 68
END DECLARATIVES 177
ENVIRONMENT DIVISION 11
ENVIRONMENT DIVISION 17
EOL 43
EOL 51
EOS 43
EOS 51
ERASE 43
ERASE 51
Estructura COBOL 10
EXIT PROGRAM 102
EXTEND 117
FD 113
FILE SECTION 12
FILE SECTION 19
FILE SECTION 113
FILE STATUS 108
FILE-CONTROL 107
FILLER 20
FROM 44
GIVING 53
GIVING 57
GIVING 57
GIVING 58
GO TO 36
GO TO 61
HIGH 42
HIGH 51
I-O 117
I-O-CONTROL 109

INDICE ALFABETICO

IDENTIFICATION DIVISION 10
IDENTIFICATION DIVISION 15
IF 37
IF 68
Imprimir 29
INDEX 24
INDEXED 107
Indice 7
INPUT 107
INPUT 117
INPUT-OUTPUT SECTION 11
INPUT-OUTPUT SECTION 18
INPUT-OUTPUT SECTION 106
INSPECT 79
INSTALLATION 16
INVALID KEY 139
INVALID KEY 140
JUSTIFIED 23
KEY IS 158
LABEL RECORD 114
LINE 31
LINE 51
LINES 135
LINKAGE SECTION 12
LOW 42
LOW 51
M rgenes 12
MOVE 32
MOVE 78
MULTIPLY 57
NEXT RECORD 158
NO BEEP 43
NOT 68
OBJECT-COMPUTER 18
OCCURS 24
OCCURS DEPENDING ON 25
OCCURS INDEXED BY 25
OF 53
OFF 43
OMITTED 114
ON EXCEPTION 44
ON OVERFLOW 102
ON SIZE ERROR 53
ON SIZE ERROR 57
ON SIZE ERROR 57
ON SIZE ERROR 58
ON SIZE ERROR 58

INDICE ALFABETICO

OPEN 116
OR 68
ORGANIZATION 108
OUTPUT 107
OUTPUT 116
PAGE 135
Palabras en COBOL 13
P rrafos 12
Pedir 33
PERFORM 85
PERFORM TIMES 89
PERFORM UNTIL 91
PERFORM VARYING 93
PICTURE 20
POSITION 31
POSITION 51
PRINT 107
PROCEDURE DIVISION 11
PROGRAM ID 16
PROMPT 43
Puntos 13
RANDOM 107
RANDOM 108
READ 123
READ INTO 126
RECORD CONTAINS 113
RECORD KEY 108
REDEFINES 23
RELATIVE 108
RELATIVE KEY 108
REMAINDER 58
RENAMES 26
REPLACING 83
REVERSE 43
REVERSE 51
REWRITE 140
RM 26
ROUNDED 53
ROUNDED 57
ROUNDED 57
ROUNDED 58
ROUNDED 58
SAME RECORD AREAS 109
Secciones 11
SECURITY 16
SELECT 107
SEQUENTIAL 108

INDICE ALFABETICO

SEQUENTIAL 108
SET 96
SET DOWN 97
SET UP 96
Signos de puntuación 14
SIZE 43
SOURCE COMPUTER 17
SPECIAL-NAMES 18
STANDARD 114
START 158
STOP 66
STOP RUN 66
SUBTRACT 56
SYNCHRONIZED 25
TAB 43
TALLYING 79
THRU 89
TIME 44
Transferencias de control 61
UNLOCK 162
UPDATE 43
USAGE 24
USE AFTER STANDARD ERROR 177
USE AFTER STANDARD ERROR 178
VALUE 22
WITH DUPLICATES 107
WITH LOCK 116
WITH LOCK 128
WITH NO REWIND 116
WITH NO REWIND 128
WORKING-STORAGE SECTION 11
WORKING-STORAGE SECTION 19
WRITE 119
WRITE FROM 121

7700
75/47

ta-ma
EDITORIAL Y LIBRERIA MICROINFORMATICA
Teléfonos: 200 97 46/47
Ctra. de Canillas, 144 28043 - MADRID.