COMPENDIO DE MANEJO DE ARCHIVOS

TEXAS INSTRUMENTS
ARGENTINA S.A.I.C.F.

COMPENDIO DE MANEJO DE ARCHIVOS

TEXAS INSTRUMENTS
ARGENTINA S.A.I.C.F.

PROLOGO DEL AUTOR

Al leer los manuales de la computadora TI - 99/4 A, particularmente en lo referido a manejo de archivos, se puede verificar la presentación de un material excelente en el tema de Instrucciones "BASIC' para el manejo de archivos

En este sentido, los textos son claros, concisos y pedagógicos, por ello, todas las partes de los manuales que tratan sobre archivos han sido traducidas al castellano tratándose de mantener el mismo sentido didáctico que los originales. Sin embargo, una cosa es saber la sintaxis de las instrucciones "BASIC" referidas a archivos y otra muy distinta es saber manejar archivos eficientemente mediante técnicas avanzadas de programación.

Las técnicas avanzadas de programación aplicadas a archivos adquieren una importancia cada vez mayor en la medida en que se aplican en sistemas de microcomputación. Estas técnicas no solo permiten una programación más racional y ordenada sino que también disminuyen los tiempos de ejecución de los programas, tendiendo además a unificar y universalizar ciertas técnicas comunes a todos los tratamientos de archivos.

*El aprendizaje de estas técnicas no está directamente relacionado con el conocimiento profundo de las instrucciones de manejo de archivos de "**TI - BASIC**" y sin un texto adecuado referido al tema seria largo, tedioso y penoso. Es más, de toda la bibliografía referida a la computación que invade nuestros mercados, se puede decir que no existe ninguna publicación que tome el tema de la "**lógica de manejos de archivos en microcomputadoras**" y lo desarrolle a un nivel aceptablemente profundo a la vez que didáctico. Todas las publicaciones referidas a estos temas son aplicables, en general, a máquinas cuya capacidad suele ser bastante mayor.*

Por todo esto, se ha creído conveniente agregar un modesto tratado de introducción a la lógica de manejo de archivos en microcomputadoras a continuación de la traducción de las instrucciones "BASIC" correspondientes.

Este agregado sin embargo exige que el lector conozca el funcionamiento o la lógica de los diagramas de flujo. Se ha enfocado el tema desde un punto de vista eminentemente práctico de tal forma que el lector pueda ir ejercitando y fijando los conocimientos de una forma muy personal.

ING. ALVARO MIRO
Buenos Aires, Diciembre de 1982

INSTRUCCIONES EN BASIC

Las instrucciones "Basic" utilizadas en el procesamiento de archivos son las siguientes:
open - close - input - restore

Instrucción "Open".

La gramática de esta instrucción es:
Open # (Nro. de archivo) : dispositivo, nombre de archivo, organización, tipo, modo, vitalidad.
La instrucción "Open" prepara al programa para la utilización de archivos almacenados en dispositivos externos.
La instrucción "Open" posibilita la apertura de una relación entre el número de archivo asignado por el usuario y el dispositivo donde se halla el archivo.
Por otra parte el "Open" le definirá a la computadora la estructura del archivo a tratar permitiendo así que el programa lo procese correctamente.
En algunos accesorios externos la computadora verificará la coincidencia entre lo declarado en el "Open" y el dispositivo externo. Si la computadora no encontrase ese archivo o no pudiese crear uno nuevo, entonces la instrucción Open no será ejecutada y se imprimirá un mensaje de error de entrada/ salida (I/O). Ver tabla de mensajes de error.
El número de archivo y nombre de archivo deben ser declarados en el "Open". En cambio; el resto de la información puede ser omitida o incluida en cualquier orden por constituir items informativos optativos, sin embargo si se omite cualquier información optativa, la computadora asumirá para ellos ciertos valores estándar o "defaults", como se verá más adelante.

1.- Número de Archivo:

Todas las instrucciones en TI—BASIC que se refieren a archivos lo hacen a través de un número de archivo cuyo valor puede estar comprendido entre O y 255 inclusive.
El número de archivo se le asigna a un archivo determinado a través de la instrucción "Open". El número cero corresponde a un archivo muy particular: la pantalla de video o display y la consola, por ello este número no podrá incluirse en ninguna instrucción "Open". El resto de los números puede asignarse como se desee siempre que no se ejecuten diferentes instrucciones "Open" (sobre archivos distintos) utilizando el mismo número.
El número de archivo se coloca en la instrucción "Open" a continuación del signo "#" (deberá quedar un espacio blanco entre el signo "#" y el número de archivo.

2.- Nombre del Archivo:

El nombre del archivo se refiere a un dispositivo externo o a un archivo guardado en él. Cada dispositivo externo tiene un nombre predeterminado el cual es reconocido por la computadora. Por ejemplo: Los nombres gue deben utilizarse para referirse a los "drives"de discos son: "DSK 1 " para el drive número 1, "DSK2" para el drive número 2 y "DSK3" para el drive número 3. Al incluir el nombre del archivo en el "Open"se está instruyendo a la computadora para que busque un archivo determinado cada vez que el programa (a través de alguna instrucción particular) haga referencia al número de archivo asociado. El nombre del archivo puede ser cualquier "string" adecuado fi las reglas establecidas para nombres de archivos. Si se utilizasen constantes "strings" entonces se las deberán colocar entre comillas.

3.- Organización del Archivo:

Los archivos de TI—BASIC pueden organizarse secuencialmente o aleatoriamente (Random). En los archivos secuenciales los registros solo pueden leerse uno después del otro sin posibilidad de saltear el orden de lectura.
Los archivos relativos (aleatorios; random) en cambio pueden leerse y escribirse en cualquier punto de su extensión sin limitación alguna. Además también pueden procesarse secuencialmente.
En la instrucción "Open" se deberá declarar (si se desea) el caracter del archivo mediante alguna de las siguientes palabras: "Sequential" o "Relative". También se puede declarar el número de registros existentes

en el archivo colocando este número a continuación de las palabras "Sequential" o "Relative". Si se decide omitir la información correspondiente a la organización del archivo la computadora asumirá como "Default" (valor estándar) que se trata de una organización secuencial.

4.- Tipo de Archivo:

Esta especificación designará el formato en que se hallan los datos del archivo: "Display" o "Internal". El formato,"Display" se refiere a archivos almacenados en caracteres imprimibles (ASCII). Este formato se utiliza cuando es necesario que el archivo sea leído por personas más que por la computadora. Cada registro de un archivo tipo "Display" corresponde generalmente a una línea de impresor.
El formato "Internal" corresponde a información almacenada en un formato interno de máquina, el cual no se halla en un estado directamente imprimible. La información que se halle en ese estado será fácilmente leída por la computadora pero no así por personas.
Se demuestra que el formato "Internal" es mucho más eficiente para el almacenamiento de información en dispositivos externos pues requiere menor cantidad de espacio y es fácilmente "formateable" mediante una instrucción "Print". Como la computadora comprende la información "Internal" en forma directa sin requerir traducción; todo aquello que se halle en este sistema será procesado más velozmente. Si se omite definir o aclarar esta información en la instrucción "Open" la máquina asumirá el tipo "Display" como estándar o "Default".

5.- Modo de Apertura del Archivo:

Esta parte del "Open" instruirá a la máquina el modo en que fluirá la información al y desde el archivo. Los modos posibles de trabajo son los siguientes.
Input, Output, Update, Append.
Si se omitiese esta entrada la computadora asumirá el estándar de: Update. Veamos ahora detalladamente lo que significa cada uno de estos modos:

- Input: El archivo solo podrá ser leído, o sea que sólo se le podrá extraer información y no habrá forma de grabar o ingresarle información.
- Output: El archivo sólo puede ser grabado o escrito. Es el caso contrario del anterior.
- Update: El archivo podrá ser ambos leído y escrito. Esta es una de las formas más utilizadas pues lo normal en un programa que maneja archivos es tener que leer información del mismo a la misma vez que también deberá ingresar alguna información generada por el mismo programa.
- Append: Mediante este modo se podrá agregar información nueva al final del archivo actual. Pero dentro de este modo hay que tener en cuenta que los registros "viejos" del archivo no podrán leerse o escribirse de ninguna manera.

6.- Tipo de Registro:

Esta entrada de la instrucción "Open' le especificará a la máquina si los registros del archivo tienen todos la misma longitud (Fixed) o si la longitud puede ser variable (variable). Esto se lleva a cabo mediante las palabras "Fixed" o "Variable" y pueden estar seguidas por un número que especifique la longitud máxima del registro en el archivo.
Cada dispositivo externo tiene su propia longitud máxima aceptable. Si se omitiese declarar el tipo de registro la máquina asumirá una longitud estándar máxima que será función del dispositivo mismo.
Se debe aclarar que si se ha definido a un archivo corno "relative" entonces se deberá utilizar el modo 'Fixed 'para los registros. Por ello si se omitiese declarar el modo "Fixed" habiendo declarado a un archivo ccomo "Relative" la máquina asumirá el modo "Fixed" de longitud de registro.
Los archivos secuenciales pueden poseer registros fijos o variables. Si se omite declarar el tipo de registro en el open de un archivo el cual ha sido declarado como secuencial entonces la máquina asumirá un tipo de registro "Variable" con una longitud máxima estándar en función del dispositivo correspondiente.

7.- Vitalidad o Vida del Archivo:

Los archivos creados por la TI-99 son considerados permanentes. Lo opuesto sería considerarlos no temporales o transitorios, esta entrada puede omitirse totalmente en la instrucción "Open" pues la computadora asumirá siempre un modo "permanente" en la vida de los archivos.

Instrucción "Close"

La gramática de esta instrucción es: CLOSE #(No de Archivo), (Delete).
La instrucción Close discontinua o desengancha a un archivo del programa que lo está utilizando. Una vez que se ha ejecutado la instrucción "**CLOSE**" sobre un archivo, el mismo no podrá llamarse o utilizarse de ninguna manera hasta tanto no se vuelva a ejecutar un "**OPEN**" sobre ese mismo archivo. Una vez ejecutado el **CLOSE** el número de archivo bajo el cual fue abierto quedará vacante, pudiendo reutilizarse en el manejo de otros archivos mediante nuevas instrucciones **OPEN**.
La instrucción "**DELETE**" es optativa .y significa borrar. Si se utilizase esta opción en la instrucción **CLOSE**, entonces la acción que tomará la computadora dependerá del dispositivo externo utilizado. Al aplicarse la opción "**DELETE**" a archivos de discos, el archivo completo será borrado.
Si se intentase ejecutar una instrucción de "**CLOSE**" sobre archivos no declarados en una instrucción "**OPEN**" anteriormente, entonces la computadora detendrá y se desenganchará del programa, imprimiendo un mensaje de error como ser: "**FILE ERROR**" seguido de un número código.
Con el fin de proteger los archivos, la computadora automáticamente cerrará (**CLOSE**) todos los archivos abiertos cada vez que se produzca una finalización compulsiva o por error del programa.
Si la finalización del programa ocurriese debido a una orden externa (**FCTN 4**) o por la inclusión de una instrucción "**BREAK**" en el programa entonces los archivos quedarán abiertos salvo que ocurra uno de los siguientes eventos:

- Se corrija el programa mediante el modo "Edit".
- Se salga del modo "Basic" mediante una instrucción "Bye".
- Se corre nuevamente el programa desde el principio mediante una instrucción "Run".
- Se ingresa el comando "New".

Si se utilizase la tecla (**FCTN =**) (Quit), la computadora no cerrará los archivos abiertos por lo que se podría perder información de los mismos.
Si se necesita salir de un programa antes que se termine de ejecutar naturalmente, entonces se deberán seguir los siguientes pasos de tal forma que no se puedan producir pérdidas de información (suponemos que el programa está corriendo):

1) Apretar la tecla (**FCTN 4**) hasta que la computadora devuelva un mensaje "Breakpoint at XXX". Esto puede tardar algunos segundos.
2) Cuando el cursor reaparezca en la pantalla se ingresará el comando "Bye" apretando luego la tecla "Enter".

Instrucción "Input"

La gramática de esta instrucción es:
Input # (Nro. de Archivo), rec. (Nro. de Registro): Lista de variables.
Esta forma de la instrucción "Input" permite la lectura de información almacenada en un dispositivo externo.
Esta instrucción puede ser aplicada solo a archivos que han sido previamente abiertos mediante una instrucción "Open" en los modos "Input" o "Update". El número de archivo que figure en la instrucción "Input" deberá ser el correspondiente al utilizado en la instrucción "Open". La lista de variables estará compuesta por una serie de nombres de variables ordenados y separados por comas. Estas variables son las que el Input extraerá del archivo para ingresarlas a la computadora. Estas variables podrán ser numéricas o strings.

Llenado de la lista de variables:

Cuando la computadora lee registros de un archivo situado en un dispositivo externo los va guardando temporariamente en una zona de la memoria llamada I/O Buffer (Input/Output Buffer). Los valores residentes en este Buffer irán siendo asignados a la lista de variables de izquierda a derecha. Cuando una lista de variables ha sido llenada con los datos de un registro situado en el Buffer, entonces el resto de los datos (si los hubiere) , serán descartados a no ser que la instrucción "Input" termine con una coma. Al hacer que el "Input" termine en coma se creará una situación de "Input" pendiente que veremos más adelante.
Si la lista de variables del "Input" fuese más larga que los datos disponibles en el registro del Buffer, entonces la computadora tomará el siguiente registro del archivo y usará los datos del mismo para satisfacer la demanda no completada de datos de la lista de variables del "Input".

Al ejecutar una instrucción "Input" la computadora tomará diversas acciones dependiendo estas del formato de almacenaje en que estén los datos (internal o display).
Si el archivo está en formato display: sus datos tendrán la misma forma que cuando fueron ingresados por la consola. En display la computadora conoce la longitud de cada dato simplemente por la coma que figura entre los mismos. Cada dato en un registro tipo display es chequeado para asegurar que los valores numéricos sean asignados a las variables numéricas (ver ejemplo 1). Si el tipo de dato no coincidiese con el tipo de variable, se producirá un mensaje de error "Input" y el programa dejará de correr inmediatamente.

Ejemplo 1:

```
100 OPEN #13: DSK1, SEQUENTIAL, DISPLAY, INPUT, FIXED 64
110 INPUT #13: A, B, STATES, D$, X, Y
```

Archivo en formato Internal:

Cada ítem o campo de un registro de un archivo en formato "Internal" está compuesto por 8 Bytes que contienen al número y uno adicional que contiene la longitud del registro. Cuando el ítem es numérico, entonces el largo del mismo será siempre de 9 Bytes o caracteres pero cuando se trate de un ítem compuesto por un string, entonces su largo podrá ser variable. Aquí sí que adquiere valor el primer caracter del ítem que define su longitud. En los registros que están en formato "Internal" pero de longitud fija (Fixed) el intentar leer más allá de los datos almacenados en los mismos causarán que la máquina lea una serie de ceros binarios por lo que si se tratase de asignar estos ceros a una variable numérica, se producirá un mensaje de error de Input y el programa dejará de correr inmediatamente.
La computadora conoce el largo de cada ítem interno mediante la evaluación del Indicador de Largo de Ítem existente al principio de cada uno.
Todos los Ítems Numéricos deben tener una longitud de nueve posiciones (ocho para los números y uno para especificación del largo del Ítem) y deben constituir representaciones válidas de números en punto flotante. Si no fuese así ocurrirá un error de entrada (Input) y el programa terminará inmediatamente
En registros de formato interno pero de longitud fija (Fixed), el leer más allá de los datos almacenados en cada registro causará que la máquina lea ceros binarios, por lo que si se tratase de asignar estos caracteres a una variable numérica ocurrirá un error Input y el programa dejará de correr inmediatamente. Si lo que se estuviesen leyendo fuesen Strings, entonces se asignará un String nulo a la variable String.

El Uso del Input con archivos aleatorios

Los archivos aleatorios pueden ser leídos secuencial o aleatoriamente.
La computadora crea un contador interno que indicará cual es el próximo registro a ser leído. El primer registro de un archivo es el número cero. Así, el contador arrancará desde cero y se irá incrementando en una unidad cada vez que se ingrese o use el archivo, ya sea para leer o escribir algo en él. En el siguiente ejemplo se la instruye a la computadora para que lea un archivo en forma secuencial.

```
100 OPEN #4: NAME$, RELATIVE, INTERNAL, INPUT, FIXED 64;
110 INPUT # 4: A, B, C$, D$, X
        :PROGRAMA
200 CLOSE #4
210 END
```

El contador interno puede ser modificado mediante la clausura **REC**. La expresión numérica colocada a continuación del **REC** será tomada por la computadora y asignada al contador que maneja los registros. Cuando la computadora ejecute un Input con cláusula **REC** leerá el registro especificado a continuación del "**REC**" y además situará al registro en el Buffer correspondiente.

La cláusula **REC** solo es permitida en instrucciones que se refieran a archivos aleatorios. El siguiente ejemplo muestra el acceso a un archivo aleatorio usando la cláusula **REC**:

```
100 OPEN # 6: NAME$, RELATIVE, INTERNAL, UPDATE, FIXED 72
110 INPUT K
120 INPUT # 6, REC K: A, B, C$, .D$
        :PROGRAMA
300 CLOSE # 6
310 END
```

Hay que asegurarse de usar la cláusula REC si se leen y escriben registros en un mismo archivo dentro de un mismo programa. Como el mismo contador se verá automáticamente incrementado por la lectura y jo escritura se puede, sin querer, saltear algunos registros y, lo que es peor, escribir sobre otros sobre los cuales no se deseaba efectuar correcciones. El ejemplo siguiente ilustra el problema:

```
100 OPEN # 3: NAME$, RELATIVE, INTERNAL, UPDATE, FIXED
110 FOR I = 1 To 10
120 INPUT # 3: A$, B$, C$, X, Y
  .
  .
  .
230 PRINT # 3: A$, B$, C$, X, Y
240 NEXT
250 CLOSE #3
260 END
```

Si hacemos correr el programa, en la línea 120 se leerán los registros N° 0, 2, 6, 8, etc.
Y en la instrucción 130 se escribirán los registros: 1, 3, 5, 7, 9… etc.
Si el contador interno apuntase a un número de registro más allá de la capacidad de datos almacenados entonces se producirá un mensaje de error de entrada (**INPUT**), causando la inmediata terminación del programa.

El Uso de INPUT PENDIENTE

La condición de **INPUT PENDIENTE** se produce cuando se ejecuta una instrucción Input conteniendo una coma final. Cuando se ejecute la próxima instrucción INPUT la computadora tomará uno de los siguientes cursos de acción.

- Si el **INPUT** nuevo no tiene **cláusula** REC, la computadora usará los datos del Buffer de entrada/salida empezando en el punto donde el INPUT anterior se habla detenido.
- Si el **INPUT** nuevo tiene cláusula REC, entonces elimina la condición de pendiente y lee el registro especificado por REC, introduciéndolo en el Buffer de entrada/salida.

Si existe una condición de **INPUT** pendiente y se ejecuta una instrucción **PRINT** sobre el mismo archivo, entonces la computadora anula la condición de pendiente y ejecuta la instrucción **PRINT** en forma normal. Si se usa un **INPUT** pendiente con el número de archivo cero se producirá la impresión del mensaje "**INCORRECT STATEMENT**" y la ejecución del programa será automáticamente terminada.

Fin de archivo

En procesamiento secuencia) se puede prevenir el error de intentar leer cuando el archivo está terminado. Para ello se deberá informar a la computadora que se ha arribado al fin de archivo. Para que esto resulte fácil **TI-BASIC** incluye una función de fin de archivo (EOF). Hay que asegurarse de colocar la instrucción EOF inmediatamente antes del **INPUT** que lee al archivo secuencialmente. De esta forma la lectura del archivo será detenida automáticamente cuando no exista más información para leer. Lo que se acostumbra es saltar a una rutina de cierre cuando se cumple con un EOF.

Ejemplo:

```
100 OPEN # 5: NAME$, SEQUENTIAL, INTERNAL, INPUT, FIXED
110 IF EOF (5) THEN 150
120 INPUT # 5: A, B
130 PRINT A;B
140 GOTO 110
150 CLOSE #5
160 END
```

La instrucción EOF no puede ser utilizada con archivos aleatorios o con algunos dispositivos externos. En estos casos se deberá crear una rutina especial para detectar el fm del archivo.
Un técnica común de fin de archivo es crear un último registro en el archivo que sirva como indicador de fin de archivo. A este registro se lo llama: "**DUMMY**" porque los datos que contiene sólo sirven a los efectos de acusar el fin de archivo. Por ejemplo, el registro podría estar lleno de nueves (999...). Cada vez que la computadora ejecute un **INPUT** sobre el archivo, se puede testear el registro ingresado y si se detectasen puros nueves entonces se podrá saltar a una rutina de cierre. El primer ejemplo que veremos crea un registro **DUMMY** y en el siguiente ejemplo la computadora testea los registros, transfiriendo el control a una rutina de cierre cuando se halla un registro **DUMMY**.

Ejemplo:

```
100 OPEN # 2: "CS1 ", SEQUENTIAL, FIXED, OUTPUT, INTERNAL
110 READ A,B,C,
120 IF A = 99 THEN 180
130 E=A+B+C
140 PRINT A;B;C;E
150 PRINT # 2: A,B,C,E
160 GO TO 110
170 DATA 5,10,15,20,30,100,200,300,99,99,99
180 PRINT # 2: 99,99,99,99
190 CLOSE # 2
200 END
```

```
100 OPEN # 1: "CS1", INTERNAL, INPUT, FIXED
110 INPUT # 1: A,B,C,E
120 IF A = 99 THEN 160
130 F = A * E
140 PRINT A;B;C;E;F
150 GO TO 110
160 CLOSE # 1
170 END
```

FUNCION - INSTRUCCION EOF (FIN DE ARCHIVO)

La gramática de esta instrucción es: EOF (expresión numérica).

La función EOF determina si se ha llegado al final de un archivo almacenado en algún dispositivo. El argumento es una expresión numérica correspondiente a algún archivo que haya sido abierto (OPEN). La función es el valor numérico generado cuando el argumento es evaluado por la función. Los valores que arroja la función EOF son los siguientes:

```
 0 Si no se llegó al fin de archivo
 1 Si se llegó a un fin de archivo lógico
-1 Si se llegó a un fin de archivo físico
```

Un archivo está posicionado en un fin de archivo lógico cuando se han procesado todos los registros del archivo. El fin de archivo físico se da cuando ya no queda más espacio para el archivo.

Ejemplo:

```
100 OPEN # 2: NAME, SEQUENTIAL, INTERNAL, INPUT, FIXED
110 IF EOF (2) THEN 160
120 REM: IF EOF GIVES ZERO
130 INPUT # 2: A,B,C
140 PRINT A,B,C
150 GO TO 110
160 CLOSE #2
170 END
```

INSTRUCCION PRINT

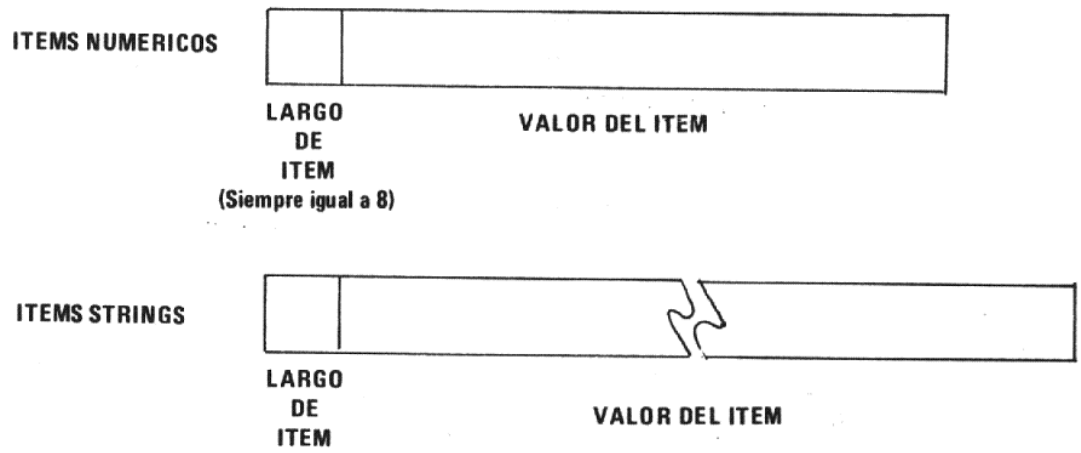
La gramática de esta instrucción es:

PRINT # N° de archivo, [REC expresión numérica]:[lista de variables]

Esta forma de instrucción PRINT permite escribir datos en un dispositivo externo. La instrucción PRINT sólo puede ser utilizada para escribir datos en archivos que hayan sido abiertos (OPEN) en los modos OUTPUT, UPDATE o APPEND. El N° de archivo debe ser el número del OPEN correspondiente a cuando se abrió el archivo.
Cuando la computadora ejecuta una instrucción PRINT guarda los datos en un sector de memoria transitorio llamado I/O BUFFER (Buffer de Entrada/Salida). Por cada archivo abierto existirá un Buffer separado. Si la instrucción PRINT no terminase con una coma, el registro es inmediatamente escrito en el fichero, partiendo del Buffer. Si la instrucción PRINT terminase con una coma entonces se creará una situación de PRINT pendiente mediante la cual los datos quedan retenidos en el Buffer sin procederse inmediatamente a su impresión. Más adelante se discuten las situaciones de un PRINT pendiente. Se discutirá aquí la información necesaria para crear una lista de impresión para grabar datos en un dispositivo de almacenamiento externo. La lista de impresión necesaria para imprimir líneas es exactamente igual a lo que se exige en un PRINT común. Se pueden usar datos en formato Display o Internal a pesar de que, como siempre, es más fácil y eficiente el generar archivos en formato internal.

Uso de la Instrucción PRINT con archivos en formato Internal

La lista de impresión consiste de expresiones numéricas o strings separadas por comas. Todos los elementos de separación en la lista de impresión tienen el mismo efecto y sentido para datos en formato internal: solamente sirven para separar los items uno de otros no indicando condiciones de espacia- miento como en un PRINT común.
Cuando los items de la lista de impresión son escritos en el accesorio externo en formato internal, los datos requieren las siguientes características:



En el ejemplo que sigue, la longitud total de datos grabados en formato internal es de 71 posiciones. Cada variable numérica consume 9 posiciones. A$ tiene una longitud de 18 caracteres (línea 110) más una posición para grabar el largo del string. B$ mide 17'caracteres más 1 (línea 120). Si los valores de A$ y B$ cambiasen durante el programa sus longitudes variarán de acuerdo al valor que tengan en el momento de ser escritas en el archivo. En el diseño de los registros, sin embargo, convendrá familiarizarse con la cantidad de datos que cada variable podrá contener, planeando siempre el registro de tal forma que siempre adquiera la mayor longitud posible.

```
100 OPEN # 5: "CS1 ", SEQUENTIAL, INTERNAL, OUTPUT, FIXED 128
110 A$ = "TEXAS INSTRUMENTS"
120 B$ = "PERSONAL COMPUTER"
130 READ X, Y, Z.
140 IF X = 99 THEN 190
150 A = X*Y*Z
160 PRINT # 5: A$, X, Y, Z, B$, A
170 GOTO 130
180 DATA 5, 6, 7, 1, 2, 3, 10, 20, 30, 40, 60, 1,5, 2,3, 7,6, 99, 99, 99
190 CLOSE # 5
200 END
```

Siempre que se especifiquen registros de longitud fija la computadora rellenará cada registro tipo internal con ceros binarios de tal forma que cada registro adopte la longitud necesaria.
La computadora no aceptará que un registro sea más largo que el máximo permitido por el dispositivo externo. Si por alguna razón se produjese esta situación en un registro de formato internal el programa terminará inmediatamente y se imprimirá un mensaje de error: "FILE ERROR IN xx".

Uso del PRINT con datos en formato display en dispositivos de almacenamiento externo

A pesar de que es más conveniente usar formato internal en datos almacenados en dispositivos externos, puede ocurrir que ocasionalmente se necesite recurrir a algún archivo que esté en formato display. A continuación se dan algunas indicaciones a tener en cuenta cuando se usa el formato display.

- Los registros se crean de acuerdo a las especificaciones que figuran en la instrucción PRINT de entrada y salida.

- Si al incluir un item-dato de la lista de impresión el mismo produjese un registro mayor en longitud que lo asignado o que el máximo aceptado por el dispositivo, el item no se dividirá pero pasará a ser el primer item del próximo registro. Si un item individual resultase de mayor longitud que la longitud del registro entonces el item será dividido en tantos registros como sean necesarios para abarcarlo y almacenarlo. El programa continuará corriendo normalmente y no se dará ningún aviso automático de la situación.

- Con objeto de leer archivos en formato display y a posteriori, los datos deben presentarse como si hubiesen sido entrados por la consola. Por ello, se deberá incluir explícitamente las comas separadoras y los dos puntos necesarios en el INPUT cuando se escribe un registro en el archivo. Estos sigros de puntuación no serán automáticamente insertados cuando se ejecuta la instrucción PRINT, y deben ser incluidos en la lista de variables de impresión tal como se muestra en la línea 170 del siguiente ejemplo:

```
100 OPEN # 10: "CS1 ", SEQUENTIAL, DISPLAY, OUTPUT, FIXED 128
170 PRINT #10: “””“;A$; “““,”; X; “ ,”; “,”; Y “,”; Z; “,””“; B$, “““ ,”; A
300 CLOSE #10
310 END
```

- Los items numéricos no tienen longitud fija como cuando se hallan en formato internal. En display, el largo de un item numérico es el mismo que tendría si fuese impreso en la pantalla mediante una instrucción PRINT o DISPLAY. Por ejemplo, el número de posiciones necesarias para imprimir 1,35 E - 10 es de diez.

Uso del PRINT en archivos aleatorios.

Los archivos aleatorios pueden ser procesados secuencialmente o en forma aleatoria. La computadora define un contador interno que indicará cuál registro ha de procesarse a continuación. El primer registro de un archivo es el número cero. Así, el contador se inicia en cero y se va incrementando en 1 después de cada acceso al archivo (leer o escribir). En el próximo ejemplo la instrucción PRINT dirige a la computadora a escribir en el archivo en forma secuencial. Luego podrá ser procesado en forma aleatoria o secuencial.

```
100 OPEN # 3: NAMES$, RELATIVE, INTERNAL, OUTPUT, FIXED 128
100 PRINT #3: A$, B$, C$, Z, Y, Z
200 CLOSE #3
210 END
```

El contador interno puede ser cambiado usando la claúsula REC.
La palabra REC debe estar seguida por una expresión numérica cuyo valor especificará en cual posición del archivo deberá escribir. Cuando la computadora efectúa un PRINT con clausura REC co- comienza a construir un registro de salida en el BUFFER, y cuando este registro sea escrito en el archivo lo será en la posición indicada por la expresión numérica de la claúsula REC. Esta claúsula solo puede usarse con archivos aleatorios el siguiente ejemplo muestra el uso PRINT con claúsula REC

```
100 OPEN # 3: NAME $, RELATIVE, INTERNAL, UPDATE, FIXED 128
110 INPUT K
120 INPUT # 3, REC K: A$, B$, C$, X, Y, Z
300 CLOSE #3
310 END
```

.....

```
100 OPEN #3: NAME$, RELATIVE, INTERNAL, UPDATE, FIXED
110 FOR I = 1 TO 10
120 INPUT # 3: A$, B$, C$, X, Y
130 PRINT  # 3: A$, B$, C$, X, Y
140 NEXT I
150 CLOSE # 3
160 END
```

Hay que asegurarse de utilizar la clausula REC si se leen y escriben registros de un mismo archivo dentro de un programa. Como el mismo contador interno se incrementa en 1 con cada acceso se podrian producir saltos o sobre escrituras sino se usa REC. El ejemplo escrito mas arriba nos muestra este detalle: La linea no produce las lecturas consecutivas de los registros 0, 2, 4, 6, 8 y la linea 130 escribe los registros 1, 3, 5, 7, 9

El uso de la instrucción PRINT pendiente

El registro es simpre inmediatamente escrito en el archivo cuando la computadora encuentra una instrucción PRINT, siempre que esta instrucción no finalice con una coma. Cuando una instrucción PRINT termina con una coma el PRINT queda pendiente o sea: no se lleva a cabo. Cuando aparece el proximo PRINT sobre el mismo archivo entonces la computadora podrá tomar diversos cursos de acción:

- Si la nueva instrucción PRINT no contiene claúsula REC la computadora colocará los datos en el BUFFER a continuación de los datos que ya se hallan allí.
- Si la nueva instrucción PRINT contiene la clausula REC la computadora escribira los datos pendientes de impresión en el registro del archivo que indique en ese momento el contador interno y efecutará al nuevo PRINT inmediatamente como es normal.

Si existe una condición de PRINT pendiente y se encuentra una instrucción INPUT sobre el mismo archivo entonces los datos pendientes serán escritos en el archivo inmediatamente en el número de registro indicado por la posición del contador interno, y el contador interno será incrementado. Luego se ejecutará la instrucción INPUT normalmente. Si existiese un PRINT pendiente y el archivo hubiese sido cerrado o restored (restituido) el registro pendiente será escrito antes de que se ejecute el restore ó el close

Instrucción Restore

La gramática de esta instrucción es:
RESTORE L Nº de archivo, REC Expresión numérica

La instrucción RESTORE reposiciona a un archivo abierto en su comienzo o a un registro particular si el archivo es aleatorio. Veamos los siguientes ejemplos:

```
100 OPEN #2: "CS1" SEQUENTIAL, INTERNAL, INPUT, FIXED 64
110 INPUT #2: A, B, C$, D$, X
400 RESTORE # 2
410 INPUT #2: A, B, C$, D$, X
500 CLOSE#2
510 END
```

```
100 OPEN # 4: NAME$, RELATIVE, INTERNAL, UPDATE, FIXED 128
110 INPUT # 4: A, B, C
200 PRINT # 4: A, B, C
300 RESTORE # 4; REC 10
310 INPUT # A, B, C
400 CLOSE # 4
410 END
```

Si el archivo especificado en el RESTORE no hubiese sido abierto entonces se generará un mensaje de error: FILE ERROR IN X X además el programa terminará inmediatamente.

Se puede usar la clausura REC solo en ficheros aleatorios. La computadora evaluará la expresión numérica que sigue a la palabra REC y utilizará este valor como puntero de un registro específico del archivo si se utiliza la instrucción RESTORE en un archivo aleatorio sin colocar la cláusula REC el archivo será restituido al registro cero.

Si existe una condición dé PRINT pendiente, esta será ejecutada antes de que se ejecute el RESTORE. Si en cambio existiese un INPUT pendiente entonces la información del BUFFER de entrada/salida será descartada.

Los archivos aleatorios no pueden ser manejados con cintas.

LOGICA DE MANEJO DE ARCHIVOS

El término "archivo" - Los ficheros administrativos - Los sistemas.

El tema de los archivos en computación ha sido siempre algo así como un tema "tabú", sobre todo para aquellas personas que se acercan por primera vez a las microcomputadoras. Sin embargo, el manejo de archivos en computación es conceptualmente sencillo a pesar de que puede presentar algunas facetas engorrosas en la realidad. Las reglas del manejo de archivos son pocas y simples, no yendo más allá de lo expresado por las instrucciones propias del lenguaje. Entonces, dónde está el "secreto" de manejo de los archivos, si todo es tan simple?. El secreto, si es que existe, reside en aprender a combinar estas instrucciones con ciertas técnicas de diagramación y con otras técnicas de manejo de arreglos (vectores, matrices, etc.) en la memoria de la máquina.

Todo esto lo iremos desarrollando ordenadamente a lo largo de este pequeño tratado.

Para comenzar, vamos a describir los alcances del término "archivo" y lo haremos por similitud con un término que, en líneas generales es sinónimo de éste: el término "fichero".

Todos sabemos qué es un fichero, pues de alguna manera o de otra en la vida común manejamos ficheros de stock, de personal, de cuentas corrientes, de proveedores, de direcciones (agenda), de repuestos, de medicamentos (vademecum), de calidad, etc.

Un fichero es un conglomerado de fichas o de tirillas (kardex) donde se guardan datos referentes a algún tema que sea de interés. Una particularidad de los ficheros es que están ordenados a través de alguno de sus datos, por ejemplo, cada ficha de un fichero de stock de materiales de un almacén puede contener los siguientes datos:

1 - Código de artículo
2 - Descripción de artículo
3 - Fecha de último movimiento
4 - Cantidad actual (stock)
5 - Precio medio de compra

Supongamos que este fichero contiene unas 2000 fichas. Cómo haríamos para hallar entre ellas a una determinada?. Muy fácil: bastaría con mantener a las fichas en orden ascendente por código de artículo, pudiendo entonces ubicar cualquier ficha de la misma forma en que ubicamos un nombre en una guía de teléfonos. Decimos entonces que este fichero está ordenado por la "clave": código de artículo. Ahora bien, qué sucedería si quisiera ubicar la ficha correspondiente a una descripción de artículo?. Bien, el hecho de que el fichero esté ordenado por una clave no implica necesariamente que lo esté por alguna otra en forma simultánea. Es más, no hace falta ningún esfuerzo mental para deducir que ningún fichero puede estar ordenado por más de una clave simultáneamente. Sería lo mismo que pedir una guía de teléfonos que estuviese ordenada por apellido, calle y número de teléfono simultáneamente

Por lo tanto, buscar una información en un fichero a través de una clave que no está ordenada, puede significar un esfuerzo realmente importante.

En los ficheros "manuales" no nos queda otro remedio que ordenarlos bajo la clave más utilizada en el manejo del mismo. Esto es lo que hace que sea sumamente engorroso extraer estadísticas de los ficheros de dimensiones relativamente elevadas.

Uno de los casos más dramáticos de ficheros de gran dimensión es el de los ficheros de manejo de stocks de productos perecederos, o sea: que poseen fechas de vencimiento, sobre todo cuando se trata de productos que pueden afectar a la salud humana. El ejemplo clásico en este caso es el fichero de manejo de stocks de productos medicinales llevado por todas las droguerías mayoristas de farmacias.

Este fichero se halla ordenado por código de medicamento por necesidades puramente administrativas aunque convendría que también lo estuviera por fecha de vencimiento del producto, por razones más que obvias. Este tipo de fichero suele superar las 10.000 fichas, por lo cual se hace virtualmente imposible efectuar un "barrido" periódico de todas las fichas para ir eliminando todos los productos vencidos o cercanos a vencer. También se hace muy engorroso en ficheros grandes, extraer estadísticas de movimientos (entradas/salidas) por artículo. Estas estadísticas tienen fundamental importancia pues son las que permiten el dimensionamiento óptimo de los mínimos de reposición a mantener en cada caso.

Sin embargo, estos temas (vencimientos, estadísticas) a pesar de ser indiscutiblemente importantes, no afectan al trabajo principal del fichero si no se realizan o no se tienen en cuenta. Simplemente nos podrán hacer perder cantidades elevadas de dinero por un mal dimensionamiento de nuestras compras ó por material echado a perder. Pero existen otros trabajos que sí deben realizarse como ser: los balances mensuales o periódicos de materiales. Esto en general no solamente exige "sumar" todo el fichero reagrupado por proveedores, tipos y subtipos, sinó que además exige generar un informe escrito cuya dimensión suele ser importante. Otra operación obligada es la verificación de los stocks versus el stock mínimo de reposición para la generación de órdenes de compra. En ficheros de gran dimensión este trabajo suele ser realmente importante en cuanto a las horas/empleado que insume, sin contar la infinidad de errores humanos generados en la trascripción de datos, números, nombres, etc. de un formulario a otro.

Son muchos los ejemplos que se podrían seguir citando. Pero hay algo que es cierto: la computadora, manejando estos ficheros en forma interna, podrá realizar estos trabajos en forma veloz y eficiente. Veloz por su velocidad de cálculo y manejo de datos, y eficiente por la probabilidad nula de que se cometan errores en esos cálculos y manipuleo de datos.

Como primera información entonces, podríamos decir: la computadora puede manejar todos nuestros ficheros individualmente en forma veloz y eficiente. A los ficheros en la jerga de la computación se los denomina "archivos", o sea, que los términos fichero y archivo son sinónimos. La única diferencia está por supuesto, en que un fichero administrativo común está compuesto por una serie de fichas de papel colocados en un mueble apropiado; en cambio, un archivo de computación estará grabado en discos, cintas, etc.

La computadora puede fácilmente "cargarse" con diferentes ficheros; esto es realmente fácil de lograr. Pero luego habrá que enseñarle a la máquina a aceptar información desde el exterior para así poder mantener los datos del fichero (archivo). Esto es lo que se llama el programa de manejo de archivo ó programa de instrucciones que permitan ejecutar diferentes operaciones sobre el fichero. Es justamente el tema más importante y que nos ocupará en este texto, el saber enseñarle a la máquina a administrar y manejar archivos.

Cada archivo que se crea y maneja no es más que un almacén de datos organizados según alguna clave o código. Pero esto no serviría de nada si no se conociesen las reglas de juego que permiten el manejo del fichero. Por ello, siempre que exista un archivo, existirá también un sistema o método de tratamiento de ese archivo. El programa a diseñar deberá tener en cuenta todas las reglas que se hallan natural y lógicamente incluidas dentro del método de tratamiento.

Operaciones básicas que se realizan sobre un archivo: creación, listado, altas, bajas y modificaciones.

Desde este punto en adelante se dan por conocidos dos temas: (a) la diagramación lógica ó diagramas de flujo y (b) el significado y la sintaxis de las instrucciones "BASIC" de manejo de archivos.

Existen ciertas reglas de tratamiento de archivos que son comunes a todos los programas que tratan con ellos y otras que dependerán exclusivamente del programa a desarrollar. El extraer la máxima cantidad de procedi-

mientos comunes hará que la programación con archivos se haga menos monótona y permitirá al programador concentrarse exclusivamente en la parte "no-standard" del programa a realizar.
Analizaremos todas aquellas operaciones que son comunes a la mayoría de los programas que tratan con archivos.

Creación de archivos - Diseño del registro - tipo: Cuando se decide confeccionar un programa que ejecute operaciones definidas sobre un archivo el primer pasó a dar, indudablemente, es la creación de un programa que permita al operador efectuar la carga inicial del archivo en discos.

Este programa auxiliar de carga inicial nada tienen que ver con el programa de trabajo que nos permitirá trabajar sobre el archivo, pues como su nombre lo indica, solo servirá para cargar una sola vez el archivo. y luego no será utilizado más para nada, salvo que, por alguna razón especial (pérdida de la información de los discos que contenían el archivo, inventario, etc.) se decida recargar todo el archivo. Para diagramar y programar la carga de un archivo en disco, en primer lugar se debe diseñar el archivo y evaluar si el disco es capaz de soportarlo desde el punto de vista de su tamaño. Veamos como se logra esto:

Sabemos que un disco de TI-99/4A poseee 358 sectores libres de 256 Bytes cada uno. Veremos como se hace para calcular cuantos "registros" de archivo caben enteros en un sector. Para esto, primero se debe definir la confección de un registro. La mejor forma de aprender este proceso es a través de un ejemplo.
Se requiere efectuar una carga de una lista de precios en un disco para su posterior tratamiento en forma computada. Los datos que lleva cada renglón de lista (cada registro) son: código de artículo, descripción, precio de lista y precio al consumidor final. Entonces: cada registro de nuestro archivo de precios estará compuesto por:

código de artículo:	8 caracteres como máximo (Bytes)
descripción de artículo:	59 caracteres como máximo (Bytes)
precio de lista:	9 caracteres como máximo (Bytes)
precio a consumidor:	9 caracteres como máximo (Bytes)

Total de Bytes (caracteres) por registro: 85.
Total de registros enteros por sector de 256 Bytes = INT (256/85) = 3
Total de artículos de la lista que pueden ingresar en un disco completo de 358 sectores = 358 x 3 = 1074

Esto quiere decir que el disco podrá contener un máximo de 1074 registros, y, una vez alcanzado este valor, el disco estará saturado.

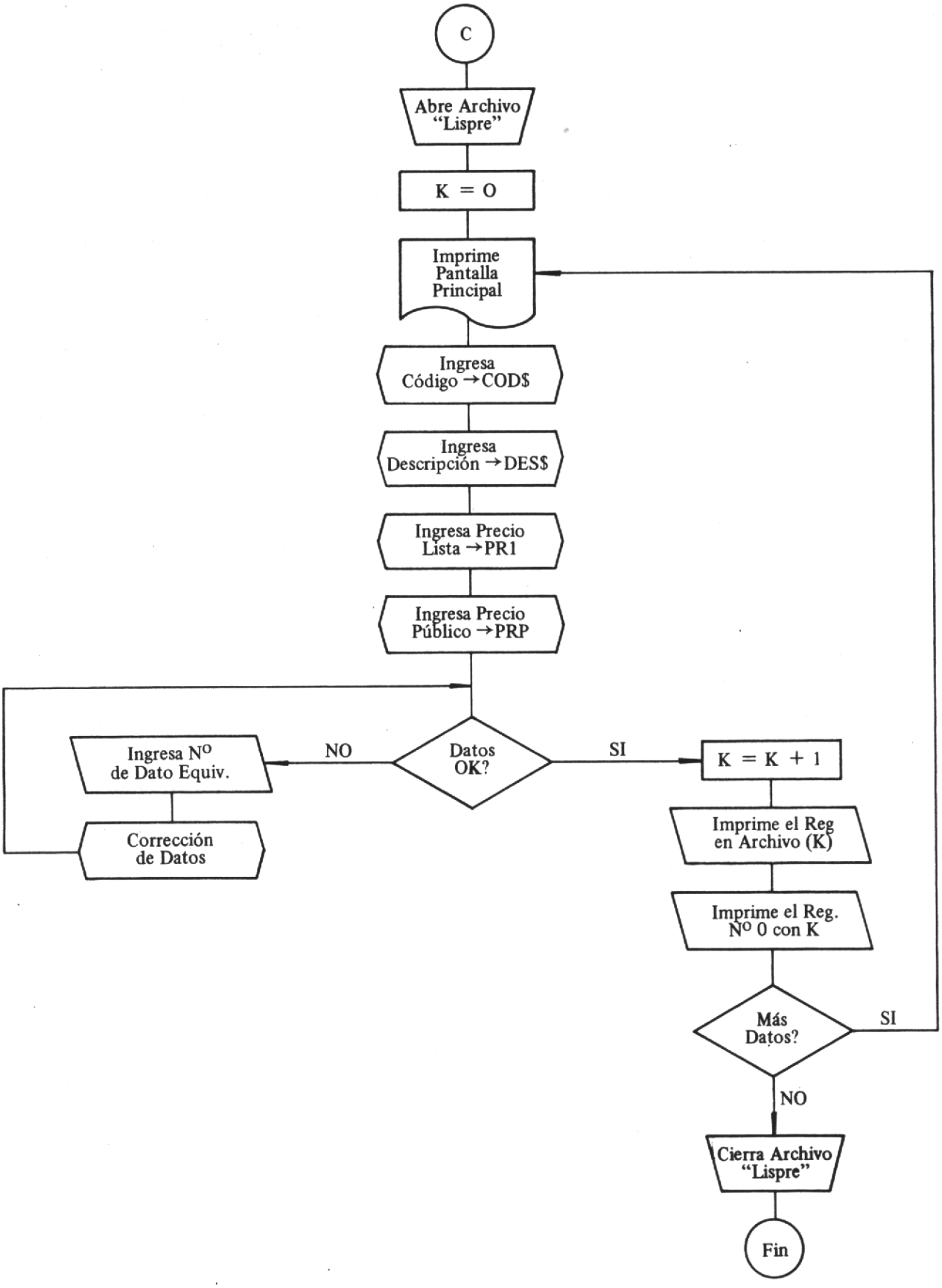
Este cálculo que acabamos de realizar es fundamental pues nos indicará si el archivo que queremos cargar "cabe" o no en el disco. Por ejemplo, si nuestra lista de precios estuviese formada por unos 5.000 artículos entonces el archivo de listas de precios no cabría en un solo disco, (pues solo pueden ingresar 1074 artículos) y por ello, habría que re-diseñar el archivo achicando los registros o mediante alguna otra técnica de tal forma que se pudiesen colocar los 5.000 en un solo disco.

NOTA La forma de cálculo de la cantidad de "fichas" o Registros que caben en un disco recién dada, en realidad no es la más precisa y se debe efectuar una aclaración respecto de la misma.
Supongamos que un Registro perteneciente a un archivo cualquiera está compuesto por dos ítems alfanuméricos y dos ítems numéricos, entonces el cargo total del Registro calculado según la técnica recién vista deberá incrementarse en 1 byte por cada ítem alfanumérico y bytes por cada ítem numérico.
Por ejemplo, en el caso de la lista de precios el cálculo adecuado sería:

CODIGO DE ARTICULO	8	BYTES + 1 BYTE	POR SER ALFANUMERICO
DESCRIPCION DE ARTICULO	59	BYTES + 1 BYTE	POR SER ALFANUMERICO
PRECIO DE LISTA	9	BYTES	POR SER NUMERICO
PRECIO A CONSUMIDOR	9	BYTES	POR SER NUMERICO

TOTAL DE BYTES POR REGISTRO = 85 + 2 = 87
REGISTRO ENTEROS POR SECTOR = INT (256 / 87) = 2
ARTICULOS DE LISTA POR DISCO = 2 X 358 = 716

Nótese lo importante de estos "Bytes Separadores" adicionales en lo que hace a la influencia sobre la capacidad de almacenamiento.


C
Abre Archivo "Lispre"
K = O
Imprime Pantalla Principal
Ingresa Código →COD$
Ingresa Descripción →DES$
Ingresa Precio Lista →PR1
Ingresa Precio Público →PRP
Datos OK?
NO
Ingresa Nº de Dato Equiv.
Corrección de Datos
SI
K = K + 1
Imprime el Reg en Archivo (K)
Imprime el Reg. Nº 0 con K
Más Datos?
SI
NO
Cierra Archivo "Lispre"
Fin

Como comentario aclaratorio se debe decir que es sumamente conveniente y recomendable que un archivo quepa en forma completa en un disco aunque no es obligatorio y los archivos que llevan más de un disco pueden "particionarse" y trabajarse en diferentes particiones. Lo que ocurre es que el trabajo con archivos particionados exige un esfuerzo de programación mucho mayor y una mayor atención por parte del operador bastante concentrada, pues el riesgo de cometer errores es aquí mayor por exigir el intercambio frecuente de discos.

Ahora bien, una vez que sabemos que el archivo cabe en un disco, nos dispondremos para diseñar un diagrama de flujo que nos permita efectuar la carga del archivo para luego codificarlo en BASIC y, finalmente ejecutarlo en la máquina. El diagrama de flujo está orientado hacia el BASIC extendido.

PROGRAMA

Veamos como podría traspasarse este diagrama a un programa en BASIC extendido. Arrancaremos desde la instrucción Nº 1000.

```
1000 OPEN # 1: "DSK1.LISPRE", RELATIVE, INTERNAL, UPDATE, FIXED 85 :: K = 0

1010 GOSUB 10000 :: REM IMPRIME PANTALLA PRINCIPAL DE PEDIDO DE DATOS

1020 GOSUB 10100 CARGA EN COD$ :: REM PIDE EL CODIGO DE ARTICULO Y LO VALIDA. CARGA EN COD$

1030 GOSUB 10110 CARGA EN DES$ :: REM PIDE LA DESCRIPCION Y LA VALIDA. CARGA EN DES$

1040 G O S U B 10120 EN PR1 :: REM PIDE EL PRECIO DE LISTA Y LO VALIDA. CARGA EN PR1

1050 GOSUB 10130 :: REM PIDE EL PRECIO A PUBLICO Y LO VALIDA. CARGA EN PRP

1060 GOSUB 10140 :: REM PIDE DATOS OK Y EFECTUA LAS CORRECCIONES NECESARIAS

1070 K = K + 1 :: PRINT # 1, REC K: COD$, DES$, PR1, PRP

1080 PRINT # 1, REC O:K

1090 F = 23 :: C = 18 :: GOSUB 10200 :: REM ACCEPT AT (F, C) Y VALIDA POR "SN"

1100 IF YN$ = "S" THEN 1010 ELSE CLOSE 1; :: END
```

SUBRUTINAS AL PROGRAMA DE CARGA INICIAL DE ARCHIVOS

```
10000 DISPLAY AT (3, 5) ERASE ALL: "DATOS P/LISTA DE PRECIOS"; TAB (5): RPT$ ("#", 24) :: DISPLAY AT(7, 2): "1. - CODIGO ART.:"

10002 DISPLAY AT (9, 2): "2. - DESCRIPCION ARTICULO:" :: DISPLAY AT (13, 2): "3. – PRECIO LISTA:" :: DISPLAY AT (15, 2): "4. - PRECIO PUBLICO:"

10004 DISPLAY AT (18, 2) : "DATOS OK? (S/N)  :: DISPLAY AT (20,2) : "DATO EQUIVOCADO?:" :: DISPLAY AT (23, 2): "MAS ARTICULOS?:"

10006 RETURN

10100 ACCEPT AT (7, 18) VALIDATE (DIGIT) SIZE (8) BEEP: COD$ :: RETURN

10110 ACCEPT AT (10, 2) VALIDATE (DIGIT, UALPHA) SIZE (59) BEEP: DES$ RETURN

10120 ACCEPT AT (13, 19) VALIDATE (DIGIT) SIZE (9) BEEP: PR1 :: RETURN

10130 ACCEPT AT (15, 20) VALIDATE (DIGIT) SIZE (9) BEEP: PRP :: RETURN

10140 F = 8 :: C = 18 :: GOSU B 10200

10142 IF YN$ = "S" THEN RETURN

10144 ACCEPT AT (20, 20) VALIDATE (DIGIT) SIZE (1) BEEP: EQ :: IF EQ < 1 OR EQ > 4 THEN 10144 ELSE EQ = INT (EQ)

10146 ON EQ GOSUB 10100, 10110, 10120, 10130

10148 GO TO 10140

10200 ACCEPT AT (F, C) VALIDATE ("SN") SIZE (1) BEEP: YN$ :: RETURN
```

Recomendamos al lector cargar este programa en su computadora y hacerlo correr ("RUN"). Se verá como el programa va pidiendo datos ordenadamente y, ante cualquier equivocación, permite que se efectuen las correcciones pertinentes. Se recomienda al lector muy especialmente seguir la mecánica del diagrama de flujo y luego del programa "sobre el papel" hasta comprender sin lugar a dudas el funcionamiento del programa y sus subrutinas. Se podrá deducir que este programa constituye un sistema universal de creación de archivos, y que, a través de unas pocas modificaciones puede servir para carga cualquier tipo de archivo de discos.

El registro número cero se utiliza para guardar el número de registros ocupados del archivo, o sea: su extensión (K).

Como se irá descubriendo más adelante, existe total justificación para la utilización de gran cantidad de subrutinas.

Agregado de datos o registros a un archivo ya creado con anterioridad

El programa anterior graba una lista de precios en un disco, y lo hace comenzando siempre desde el registro K = 1. De esta forma, cada vez que se haga correr este programa se recomenzará el proceso de carga inicial.

Qué sucedería si debiésemos cargar 700 items con sus descripciones y precios pero, debido a factores de tiempo sólo pudiésemos cargar 350 un día viernes y los restantes 350 el lunes siguiente? . La respuesta a esta pregunta es la siguiente:

Si el lunes siguiente se volviese a utilizar este mismo programa entonces se estarían re-grabando los registros del archivo desde el número 1 (K = 1) hasta el número 350 (K = 350). Este proceso destruiría los 350 registros cargados el día viernes. Lo que en realidad debería suceder el día lunes es que se debería comenzar a cargar el archivo iniciado a partir del registro número 350 (o -a partir del último registro grabado en el proceso anterior + 1). Veamos entonces como sería un programa de "agregue" de datos a un archivo ya creado. Este programa lo iniciaremos en el paso N° 2000.

```
2000 OPEN # 1: "DSK 1. LISPRE", RELATIVE, INTERNAL, UPDATE, FIXED 85
2010 INPUT 1, REC O:K
2020 GO TO 1010
```

y ya está. Por supuesto que ambos programas deberían "convivir", o sea, en realidad serian uno solo. Por lo tanto, si al programa anterior se le suman las instrucciones 2000, 2010 y 2020 entonces servirá para no solo crear inicialmente el archivo sino tambien para agregarle datos en cualquier momento que se desee.

Sin embargo, desgraciadamente esto no es suficiente pues al correr el programa se ejecutarían siempre las instrucciones tipo 1000, no ingresandose nunca a esta segunda opción. Por ello, es necesario ahora estudiar una técnica que, aunque no está relacionada con archivos, servirá para permitir el manejo de varios programas que actúan sobre el mismo archivo conviviendo estos en un solo "sistema" de programas. Esta técnica es la de los menús de opciones.

En nuestro ejemplo, no solo interesa crear el archivo y agregarle ítems sino también anular items (bajas), modificar items y listar el archivo.

Generación de un menú de opciones: Un menú de opciones nos presentará en la pantalla del monitor una colección numerada de operaciones que pueden realizarse y nos pedirá nuestra elección, en base a la cual el programa se direccionará al sector del programa que efectúa las operaciones seleccionadas. Veamos el siguiente gráfico referido al menú de opciones de nuestra lista de precios:

Pantalla del monitor

Menú principal

1 - Carga inicial
2 - Agregue de items (altas)
3 - Modificación de items
4 - Bajas
5 - Listados varios
6 - Fin de trabajo

Su elección?

En función de la elección que hagamos (1, 2, 3, ..., 6 6) el programa comenzará a ejecutarse desde el paso N° 5 1000, 2000, etc. Veamos como sería un programa que lograse esto: El programa de menú principal arrancará desde el paso número 100 y convivirá con las mismas instrucciones de los subprogramas de carga inicial y agregue

```
100 DISPLAY AT (3, 8) ERASE ALL: "MENU PRINCIPAL"; TAB (8); RPT$ ("$", 15) :: DISPLAY AT (6, 3): " 1.- CARGA INICIAL" :: DISPLAY AT (8, 3): " 2.- AGREGAR ITEMS"

110 DISPLAY AT (10, 3): "3.- MODIFICAR ITEMS" :: DISPLAY AT (12, 3): "4.- BAJAS DE ITEMS"

120 DISPLAY AT (14, 3) : "5.- LISTADOS VARIOS" :: DISPLAY AT (16, 3) : "6.- FIN DE TRABAJO" :: DISPLAY AT (20, 6): "SU ELECCION?:"

130 ACCEPT AT (20, 20) VALIDATE(DIGIT) SIZE (1) BEEP: ELEC :: IF ELEC < 1 OR ELEC > 6 THEN 130 ELSE ELEC = INT(ELEC)

140 IF ELEC = 6 THEN END

150 ON ELEC  GO TO 1000, 2000, 3000, 4000, 5000
```

El análisis de este programa es muy fácil. Simplemente imprime una lista de opciones en la pantalla del monitor (instrucciones 100-200) y luego pide que se ingrese el número de opción deseado (instrucción N° 130). El valor ingresado es validado y luego se prueba si corresponde a la última (sexta) opción (fin de trabajo). Si así fuese entonces se va directamente a una instrucción "END" (instrucción N° 140). Si la opción estuviese entre 1 y 5 inclusive entonces, a través de la instrucción N° 150 (ON ELEC GO TO) se enviará el control de programa a los sectores del mismo que ejecuten las tareas deseadas: paso 1000 para la opción N° 1 de creación inicial; paso 2000 para la opción N° 2 de agregado, etc.

La siguiente opción es la N° 3 y corresponde a modificaciones a realizar sobre el archivo. El programa de modificaciones debe comenzar en la instrucción N° 3000 según lo manda la instrucción N° 150.

Modificaciones del archivo: Muchas son las modificaciones que nos puede interesar llevar a cabo sobre el archivo. Entre otras, las más importantes son: modificar un ítem (registro) en cualquiera de sus campos; modificar todos los precios por medio de un porcentaje fijo; modificar los precios de una familia de ítems, seleccionable por medio del código del artículo; modificar la posición de los registros dentro del archivo para poder así presentarlos en un orden diferente a aquel en que se hallan ó, si estuviesen desordenados, poderlos ordenar, por ejemplo para que queden en un orden creciente de código de aplicación (sorts); etc.

En nuestro ejemplo solo desarrollaremos las modificaciones de registro, aumento masivo y aumento por ítem. Supondremos que el archivo ingresado está ordenado por código de artículo.

De todo lo dicho se desprende que la opción "3. MODIFICACIONES" no representa un único trabajo sino que, en sí contiene a otras tres opciones secundarias, por lo que el elegir la opción N° 3 ingresaremos a otro menú de opciones especificadas de las modificaciones y que llamaremos "Menú modificaciones". La pantalla correspondiente a este menú será como la que sigue:

Menú modificaciones

1 - Modificar un reg.
2 - Aumento masivo
3 - Aumento selectivo
4 - Volver a menú principal

Su elección?

El programa correspondiente a esta pantalla a las operaciones ligadas a ella deberá comenzar en el paso No 3000.

```
3000 DISPLAY AT (3,5) ERASE ALL: "MENU MODIFICACIONES";TAB (5); RPT$ ("#",19) :: DISPLAY AT (8, 2): " 1.- MODIFICAR UN REG".

3010 DISPLAY AT (11,2): "2.- AUMENTO MASIVO" :: DISPLAY AT (13,2):"3.- AUMENTO SELECTIVO" :: DISPLAY AT (16, 2): "4.- VOLVER A MENU PRINCIPAL"

3020 DISPLAY AT (20,5): "SU ELECCION?" :: ACCEPT AT (20, 19) VALIDATE(DIGIT) SIZE(1) BEEP: ELEC

3030 IF ELEC < 1 OR ELEC > 4 THEN 3020 ELSE ELEC = INT(ELEC)

3040 IF ELEC = 4 THEN 1000

3045 OPEN 1: "'DSK1, LISPRE", RELATIVE, INTERNAL, UPDATE, FIXED 85

3047 INPUT #1, REC O:K

3050 ON ELEC GO TO 3060, 3300, 3600
```

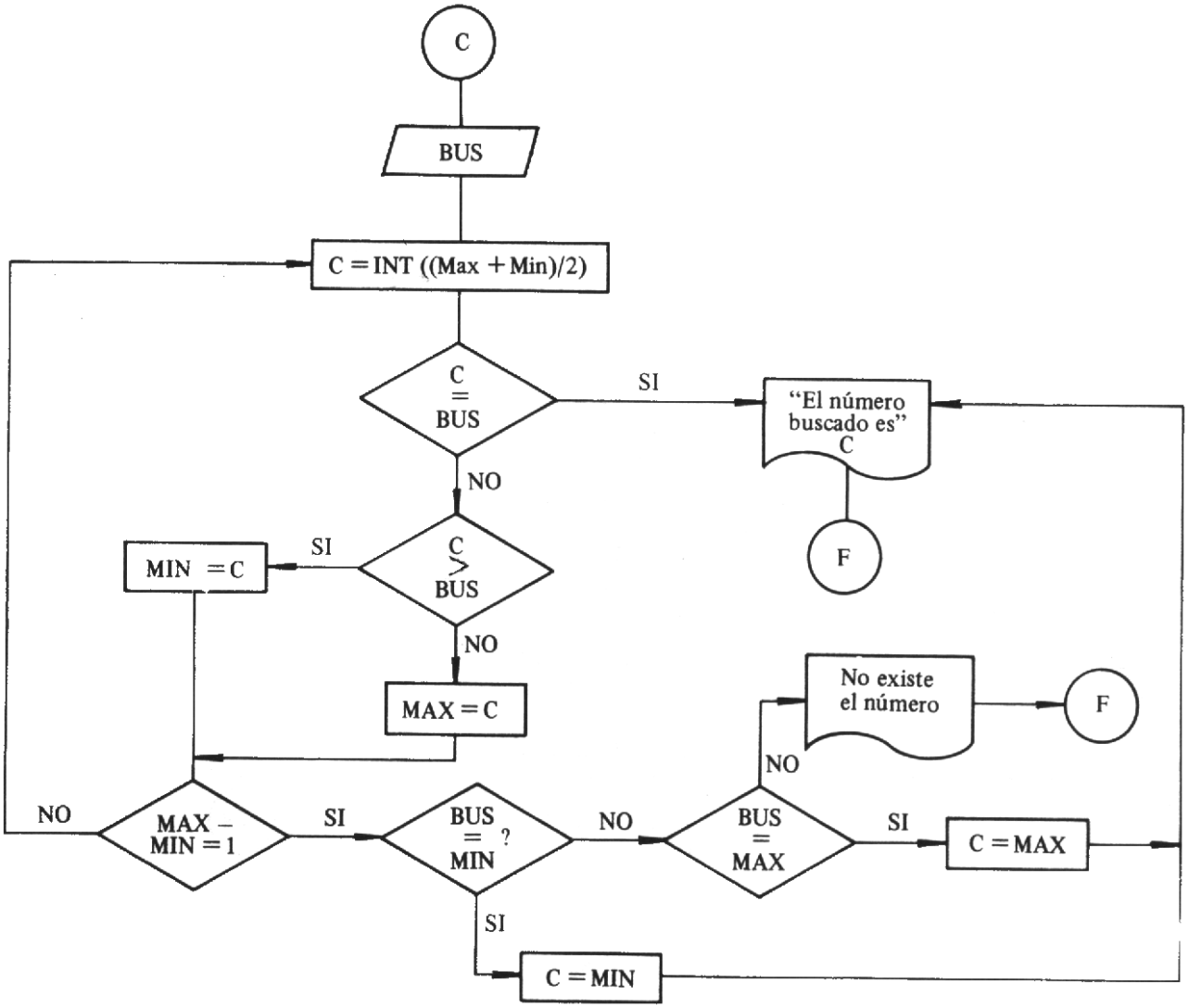
A partir de la instrucción 3060 comenzarían las operaciones destinadas a modificar alguno de los campos (código, descripción, precio de lista, precio de público) de algún registro preseleccionado del archivo. Para ello, hemos supuesto que el archivo se halla ordenado por código de artículo.

Este subprograma deberá pedir el código del artículo cuyo registro se desea modificar, buscar ese código en el archivo y, si lo halla, presentar los datos del registro en pantalla pidiéndole al operador le indique cual de los campos desea modificar. Al finalizar las modificaciones del registro se re-grabará el mismo en el mismo lugar del disco del cual fue tomado.

Si el código buscado por el operador no existiese en el archivo, entonces se deberá presentar un cartel en pantalla indicando tal circunstancia, y se requerirá el ingreso de un nuevo código de búsqueda.
Aquí el problema reside en como se busca un registro en un archivo. Si el archivo estuviese desordenado no nos quedaría otro remedio que ir tomando registro por registro del archivo e ir comparando sus códigos con el código de búsqueda. En archivos de poca cantidad de ítems este proceso es aceptablemente aplicable pero en aquellos donde la cantidad de ítems es muy elevada este sistema es sumamente ineficiente, recomendándose en tales casos el método conocido como de "búsqueda dicotómica". Este método solo es aplicable si el archivo se hallase ordenado por la clave de búsqueda seleccionada (en nuestro caso la clave de búsqueda será el código de artículo).

Hagamos aquí un alto en la confección de nuestra lista de precios y veamos un diagrama de flujo para lograr llevar a cabo una búsqueda dicotómica sobre un archivo ordenado por la clave de búsqueda y donde la cantidad de registros ocupados es conocida.

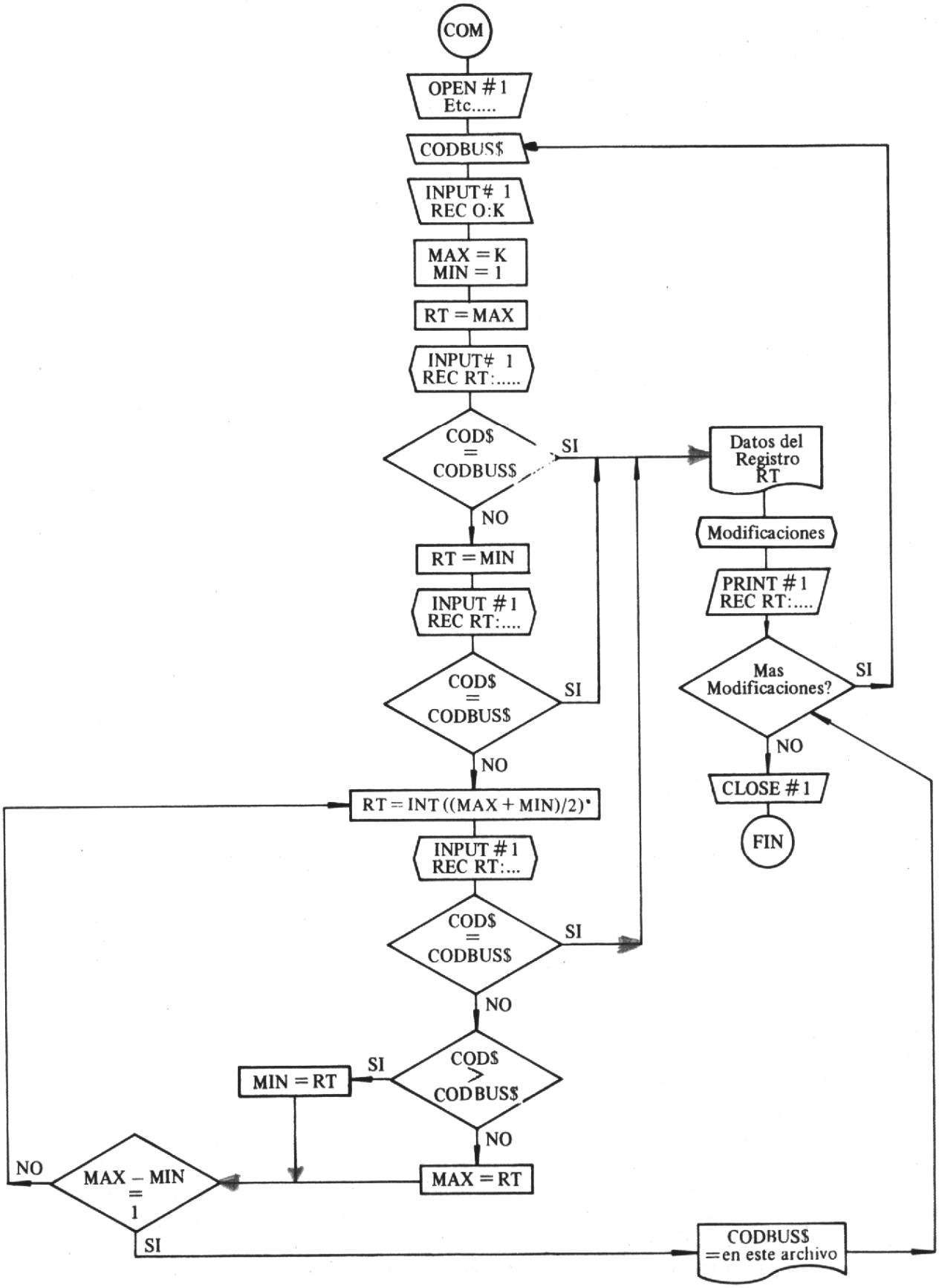
Técnica de búsqueda dicotómica en un archivo ordenado: Analizaremos primero un diagrama de flujo que permita efectuar la búsqueda de un número dentro de un vector numérico cuyos elementos consecutivos son números naturales consecutivos. El vector además está ordenado de menor a mayor. Supongamos que le alimentamos un número de la máquina y queremos comprobar si realmente ésta lo puede ubicar. Veamos primeramente el diagrama.



Supongamos que los límites (valor menor y mayor) de los elementos de este vector son MIN = 1, MAX = 10. Probemos en buscar diferentes números entre 1 y 10 inclusive y veremos que el diagrama los ubica rápidamente.

Si se alimentase el número 3, 5 (el cual no es natural, aunque está comprendido entre 1 y 10) veremos que el diagrama nos indicará que tal número no existe como perteneciente al vector antes definido. Una vez comprendido el proceso básico, es relativamente simple pasarlo a un archivo ordenado por una clave, como ser: el código de artículo de nuestra lista de precios e ingresando códigos hacer que la máquina los busque dentro del archivo.

Veamos como sería el diagrama de flujo equivalente al anterior pero aplicado a archivos.


COM
OPEN #1
Etc.....
CODBUS$
INPUT# 1
REC O:K
MAX = K
MIN = 1
RT = MAX
INPUT# 1
REC RT:.....
COD$
=
CODBUS$
SI
NO
Datos del
Registro
RT
Modificaciones
PRINT #1
REC RT:....
Mas
Modificaciones?
SI
NO
CLOSE #1
FIN
RT = MIN
INPUT #1
REC RT:....
COD$
=
CODBUS$
SI
NO
RT = INT ((MAX + MIN)/2)*
INPUT #1
REC RT:...
COD$
=
CODBUS$
SI
NO
COD$
>
CODBUS$
SI
MIN = RT
NO
MAX = RT
NO
MAX – MIN
=
1
SI
CODBUS$
=en este archivo

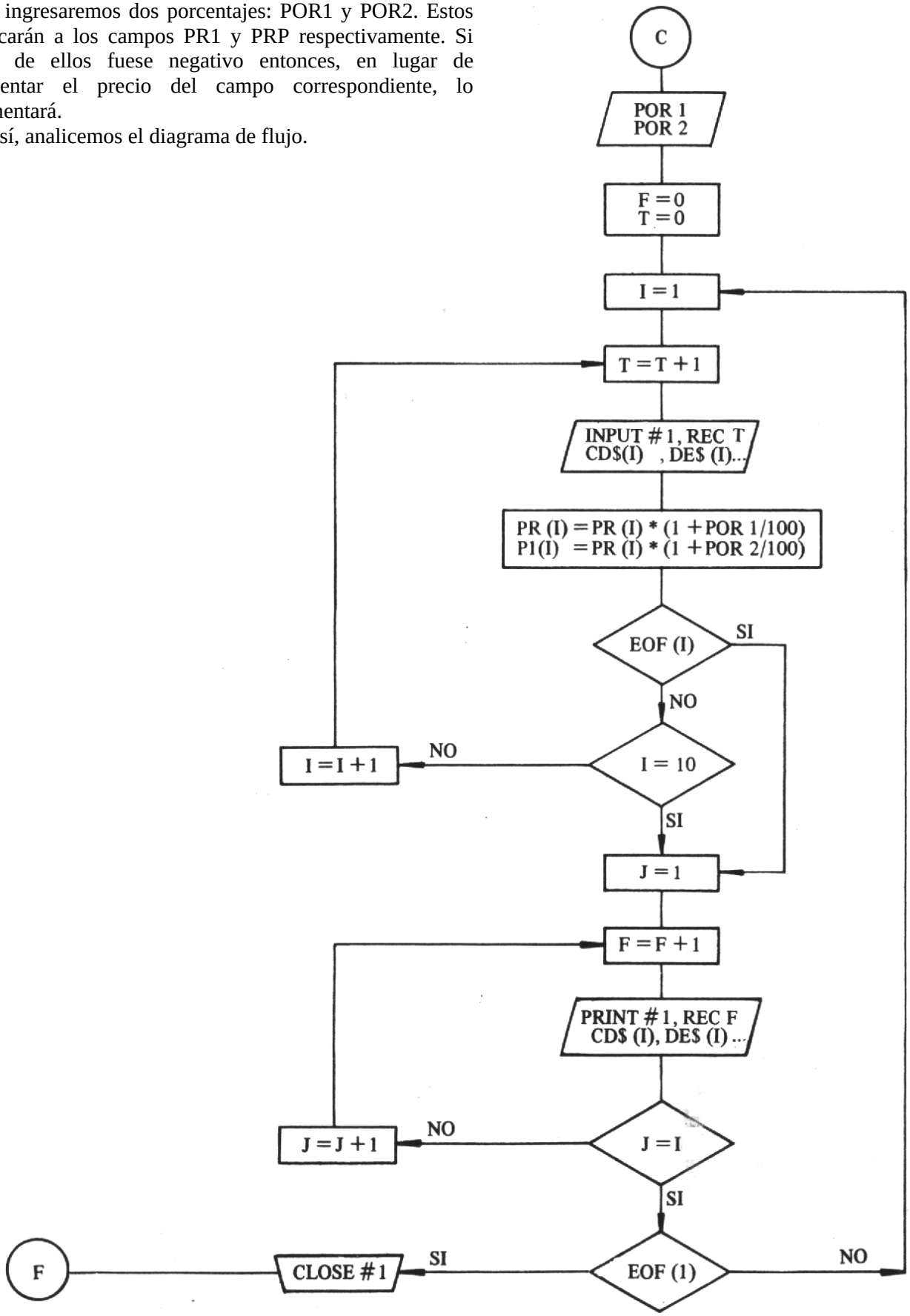
Pasando este diagrama a "BASIC", arrancando desde la instrucción 3060 habremos entonces solucionado la opción N° 1 del menú de modificaciones.

```
3060 DISPLAY AT (10,2) ERASE ALL: "INGRESE EL CODIGO DE":"ARTICULO CUYO REGISTRO":"DESEA MODIFICAR"
3070 ACCEPT AT (15, 10) VALIDATE (DIGIT) SIZE (8) BEEP: CODBUS$
3080 MAXI = K :: MINI = 1 :: RT = MAXI :: GOSUB 10210 :: REM EFECTÚA INPUT 1, REC RT
3090 IF COD$ = CODBUS$ THEN 3160 ELSE RT = MINI:: GOSUB 10210
3100 IF COD$ = CODBUS$ THEN 3160
3110 RT = INT (( MAXI + MINI)/2) :: GOSUB 10210
3120 IF COD$ = CODBUS$ THEN 3160
3130 IF COD$ > CODBUS$ THEN MINI = RT ELSE MAXI = RT
3140 IF (MAXI - MINI) = 1 THEN 3150 ELSE 3110
3150 DISPLAY AT (10,2) ERASE ALL: "NO EXISTE ARTICULO CON EL":"CO-DIGO:"; CODBUS$; "EN EL ARCHIVO" :: GOTO  3060
3160 GOSUB  10000
3170 GOSUB 10220 :: REM IMPRIME LOS DATOS DEL REGISTRO EN LA PANTALLA
3180 GOSUB 10140
3190 PRINT #1, REC RT : COD$, DES$, PRI, PRP
3200 DISPLAY AT (22, 2):"MAS MODIFICACIONES? (S/N):" :: F = 22 :: C = 27 :: GOSUB 10200
3210 IF YN$ = "S" THEN 3060
3220 CLOSE 1 :: GOTO  3000
10210 INPUT #1, REC RT; COD$, DES$, PR1, PRP :: RETURN
10220 DISPLAY AT (7,18): COD$ :: DISPLAY AT (10, 2): DES$ :: DISPLAY AT (13,19): PR1
10230 DISPLAY AT (15,20) : PRP :: RETURN
```

Según lo visto en el "ON ELEC" de la instrucción 3050, ahora correspondería permitir efectuar aumentos masivos o decrementos masivos en los precios.

Aquí también se podría optar por la técnica de tomar un registro, modificar su campo de precios y re-grabarlo ya modificado; luego tomar el siguiente y realizar el mismo trabajo, y así sucesivamente hasta agotar a todos los registros del archivo. Este método sería terriblemente lento en archivos de tamaño grande, por lo que se deberían utilizar técnicas algo más avanzadas. Una de las más eficaces consiste en trabajar sobre un campo del archivo "bajando bloques" de registros y almacenando el campo a modificar en un vector. Analicemos el siguiente diagrama de flujo que realiza el trabajo de modificación masiva. En este caso, la modificación masiva actuará sobre ambos precios (al público y de lista) aunque, con un poco de imaginación se podría solicitar al operador que defina sobre cual de los dos precios desea aplicar el porcentaje de incremento o decremento. Aún más: se podrían definir dos porcentajes de modificación: uno para el precio de lista y otro sobre el precio a público de tal forma que ambos precios se puedan modificar por medio de dos porcentajes diferentes en forma simultánea. Así, si no deseamos modificar a alguno de los campos bastaría con asignarle un porcentaje nulo de modificación con lo que los precios de ese campo no cambiarían.

O sea: ingresaremos dos porcentajes: POR1 y POR2. Estos modificarán a los campos PR1 y PRP respectivamente. Si alguno de ellos fuese negativo entonces, en lugar de incrementar el precio del campo correspondiente, lo decrementará.
Ahora sí, analicemos el diagrama de flujo.



El proceso que muestra este diagrama de flujo es el siguiente:

Se ingresan ambos porcentajes de aumento o decremento y se almacenan en las variables POR 1 y POR2 se traen a la unidad central de proceso 10 registros del archivo, almacenándose cada campo en un vector subindexado por I. A medida que van siendo traídos los registros se modifican los precios de lista y al público. Cuando se han traído y modificado un grupo de registros, entonces se deberán re-grabar en el archivo. De esto se encarga la segunda parte del diagrama (desde J = 1 hasta J = I?). Al detectarse el fin de archivo (EOF) se termina con la ejecución pues ya habrán sido re-grabados todos los registros del archivo 1. Mientras no se detecte un fin de archivo el control del diagrama volverá al principio del mismo para traer a la memoria el siguiente grupo de 10 registros.

Si se desease se podrían traer bloques de registro aún mayores (por ej.: 30 registros por vez), esto dependerá del espacio "STACK" que quede libre en la memoria de la máquina una vez cargado el programa así como también de si el programa se está trabajando en particiones o entero. Esto lo deberá analizar y decidir el programador en función de las circunstancias que lo rodean.

Ahora pasaremos el diagrama a un programa cuyo primer pase sera el número 3300.

```
3300 GOSUB 10250 :: REM PIDE EL INGRESO DE AMBOS PORCENTAJES
3310 F=0 :: T=0
3320 FOR I = 1 TO 10 ::  T = T + 1 :: GOSUB 10270 :: REM EFECTUA INPUT # 1, REC T
3330 GOSUB 10280 :: REM CALCULA LOS NUEVOS PRECIOS 3340 IF EOF (1) THEN 3360
3350 NEXT I :: I = I - 1
3360 FOR J = 1 TO T :: F = F + 1 :: GOSUB 10300 :: REM EFECTUA PRINT # 1, RECF
3370 NEXT J
3380 IF EOF  (1) THEN 3220 ELSE 3320
10250 DISPLAY AT (10, 2) ERASE ALL: "PORCENTAJE SOBRE PR. LISTA:?"
10252 DISPLAY AT (13, 2): "PORCENTAJE SOBRE PR. PUBLICO?:"
10254 ACCEPT AT (11, 10) VALIDATE (DIGIT) SIZE (5) BEEP: POR1
10256 ACCEPT AT (14, 10) VALIDATE (DIGIT) SIZE (5) BEEP: POR2
10258 DISPLAY AT (20, 3): "SON CORRECTOS? (S/N): " :: ACCEPT (20, 24) VA-
LIDATE ("SN") SIZE (1) BEEP: YN$
10260 IF YN$= "N" THEN 10254 ELSE RETURN
10270 INPUT # 1, RECT: CD$ (I), DE$ (I), PR(I), P1(I).
10272 RETURN
10280 PR(I) = PR(I) * (1 + POR 1/100) :: P1(I) = P1(I) * (1 + POR 2/100)
10282 RETURN
10300 PRINT # 1, RECF : CD$ (J), DE$ (J), PR(J), P1(J)
10302 RETURN
```

La siguiente opción dentro del menú de modificación es la de posibilitar la modificación solamente de algunos ítems.

Este es un tema que se maneja muy bien cuando se saben explotar eficientemente los códigos de items. Para mayor facilidad de interpretación de lo dicho analicemos esto desde el punto de vista de la lista de precios.

Supongamos que esta lista de precios corresponde a un comercio que vende artículos de audio. Estos artículos pueden subdividirse en diversos grupos:

1 - Familia o aplicación (Ej.: Televisores, Amplificadores, Accesorios etc.)
2 - Marca de fábrica
3 - Modelo

Entonces podríamos armar el código de cada artículo en función de la familia, marca y modelo.

Asignaremos el primer caracter del código a la familia o aplicación: los siguientes cuatro a la marca y finalmente los últimos tres al modelo.

Por ejemplo: Televisor Telefunken portátil podría armarse como A = familia de los televisores; TFKN por la marca Telefunken y PTL por ser un modelo portátil. El código entonces sería: "ATFKNPTL".

Por supuesto que esta no es la única ni la más eficiente forma de codificar artículos pero sí sirve para mostrar como ejemplo lo que se quiere enseñar.

Entonces, lo que podría interesar sería: aumentar en un porcentaje POR1 y POR2 los precios de los televisores ó también: aumentar el precio de lista de los amplificadores Philips, o modificar los precios de todos aquellos elementos portátiles de determinada familia o marca, etc.

Aquí se aprovecharán las funciones "**BASIC**" que permiten el manejo de constantes alfanuméricas o "Strings", que es justamente lo que son los códigos antes definidos.

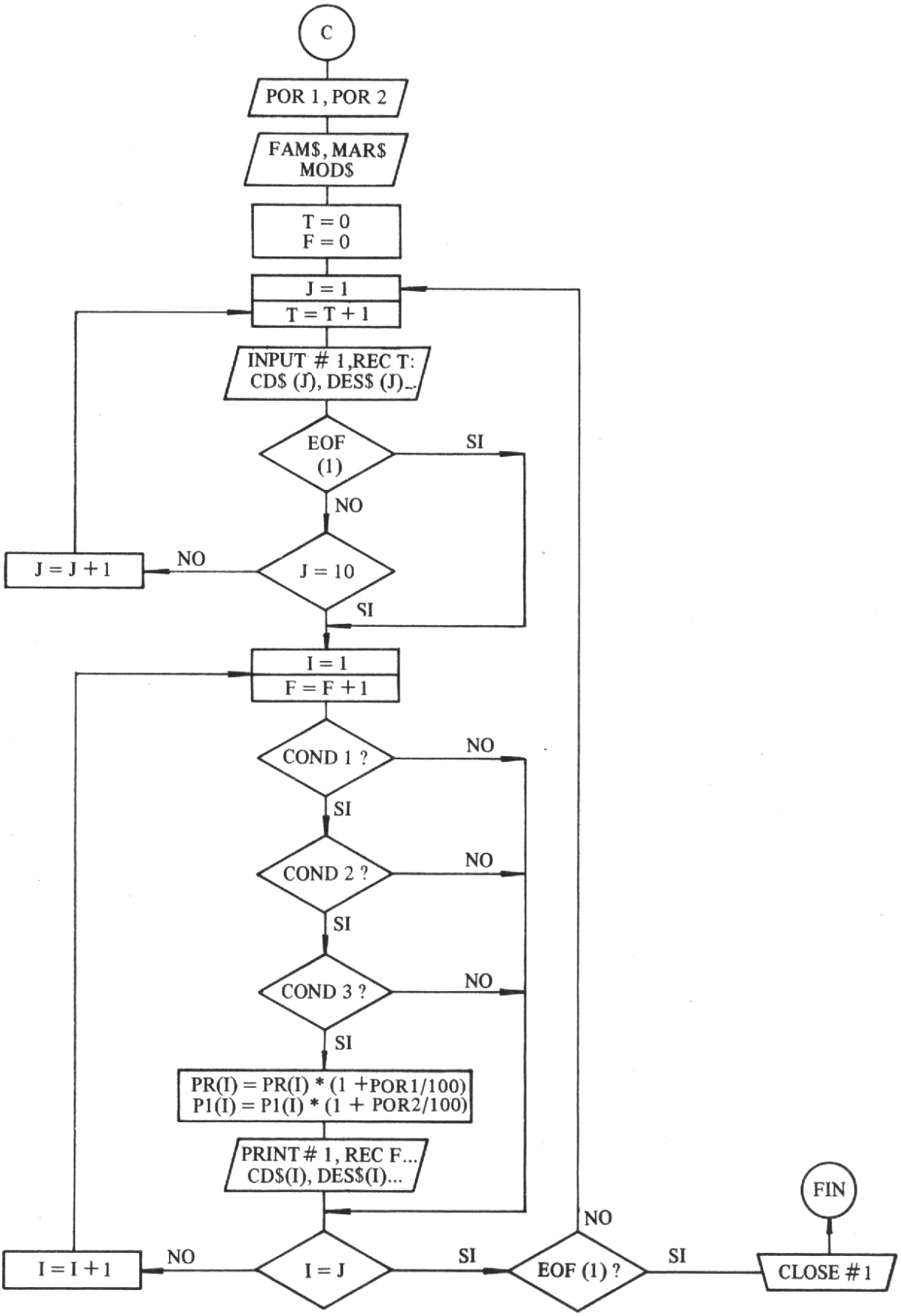
Supongamos entonces que podremos modificar familia, marca y modelo o sus combinaciones.

La máquina deberá pedir entonces tres "códigos" de modificación: uno para la familia, otro para la marca y otro para el modelo. Con esos tres códigos armará un código de búsqueda el cual se irá comparando con todos los códigos de artículos de la lista. Cuando ese código coincida con alguno de los de la lista, entonces se producirá la modificación, en caso contrario no habrá modificación de precios.

Veamos el siguiente diagrama de flujo:

Donde: COND1: IF SEG$ (CD$(I), 1, 1) = FAM$ OR FAM$ = ""
COND2: IF SEG$ (CD$(I),2,4) = MAR$ OR MAR$ = ""
COND3: IF SEG$ (CD$(I), 6, 3) = MOD$ OR MOD$ = ""

DIAGRAMA


C
POR 1, POR 2
FAM$, MAR$
MOD$
T = 0
F = 0
J = 1
T = T + 1
INPUT # 1,REC T:
CD$ (J), DES$ (J)...
EOF
(1)
SI
NO
J = J + 1
NO
J = 10
SI
I = 1
F = F + 1
COND 1 ?
NO
SI
COND 2 ?
NO
SI
COND 3 ?
NO
SI
PR(I) = PR(I) * (1 +POR1/100)
P1(I) = P1(I) * (1 + POR2/100)
PRINT # 1, REC F...
CD$(I), DES$(I)...
NO
I = I + 1
NO
I = J
SI
EOF (1) ?
SI
FIN
CLOSE #1

Analizando este diagrama vemos que pide se le ingresen los ya conocidos porcentajes "POR!" y "POR2". Luego se ingresarán las claves que se desean modificar o sea: la combinación de elementos que dirán exactamente a cuales registros se deberán afectar mediante los porcentajes POR 1 y POR2.

Estas claves se almacenan en las variables alfanuméricas MAR$, FAM$ y MOD$. Si no se desea utilizar alguna de las claves posibles (familia, marca y modelo), cuando la máquina solicite las claves en cuestión simplemente se ingresará un blanco (nulo). Por ejemplo: si se desease solo modificar los elementos cuya marcó codificada fuese "TFKN" entonces se ingresará la clave "TFKN" en la variable "MARS" y un blanco en FAM$ (familia) y MOD$ (modelo).

El diagrama bajará bloques de a 10 registros y, a través de su testeo a través de las condiciones 1, 2 y 3 decidirá si modifica el precio o no, luego de lo cual re-grabará solo aquellos registros que han sido modificados.

Las instrucciones correspondientes a este diagrama de flujo arrancarán desde la 3600 en adelante según lo indica la instrucción 3050 para la opción No 3, del menú de modificaciones.

Veamos el programa codificado en "**BASIC**" para el diagrama de flujo anterior.

```
3600 GOSUB 10250 :: REM PIDE AMBOS PORCENTAJES Y LOS VALIDA
3610 GOSUB 10320 :: REM ARMA PANTALLA DE PEDIDO DE MARCA, FAMILIA Y MODELO
3615 GOSUB 10340
3620 F = 0::T=0
3630 FOR J = 1 TO 10:: T = T + 1 :: GOSUB 10270
3640 I F EOF (1) THEN M =.J :: GOTO 3660
3650 NEXT J :: M = 10
3660 FOR I = 1 TOM:: F=F+1
3670 IF SEG$ (CD$(I), 1, 1) = FAM$ OR FAM$ = " " THEN 3680 ELSE 3720
3680 IF SEG$ (CD$(I), 2, 4) = MAR$ OR MAR$ = " " THEN 3690 ELSE 3720
3690 IF SEG$ (CD$(I), 6, 3) = MOD$ OR MOD$ = " " THEN 3780 ELSE 3720
3700 GOSUB  10280
3710 J = 1 :: GOSUB 10300
3720 NEXT
3730 IF EOF (1) THEN 3220 ELSE 3630
10320 DISPLAY AL (5,2) ERASE ALL: "1.- COD. FAMILIA:" :: DISPLAY AT (7, 2): " 2.- COD. MARCA:"
10322 DISPLAY AT (9,2):: "3.- COD. MODELO:" :: DISPLAY AT (14, 3):"DATOS OK? (S/N):"
10324 DISPLAY AT (16, 4): "DATO EQUIVOCADO:?" :: RETURN
10340 GOSUB 10500 :: REM PIDE FAMILIA Y LA VALIDA
10342 GOSUB 10510 :: REM PIDE MARCA Y LA VALIDA
10344 GOSUB 10520 :: REM PIDE MODELO Y LO VALIDA
10346 F = 14 :: C = 19 :: GOSUB 10200 :: REM PIDE OK Y VALIDA "SN"
```

```
10348 IF YN$ = "S" THEN RETURN
10350 ACCEPT AT (16, 22) VALIDATE (DIGIT) SIZE (1) BEEP: EQUI :: IF EQUI < 1 OR EQUI > 3 THEN 10350 ELSE EQUI = INT(EQUI)
10352 ON EQU I GOSUB 10500, 10510, 10520
10354 GO TO 10346
10500 ACCEPT AT (5, 8) VALIDATE (UALPHA) SIZE (1) BEEP: FAM$ :: RETURN
10510 ACCEPT AT (7,16) VALIDATE (UALPHA) SIZE (4) BEEP: MAR$ :: RETURN
10520 ACCEPT AT (9,16) VALIDATE (UALPHA) SIZE (3) BEEP: MOD$ :: RETURN
```

Con este tramo de programa se completan las opciones del menú de modificaciones y por ello pasaremos a analizar un método de producir bajas (anulaciones de registros) en el archivo.

Bajas en un archivo: En general los registros de los archivos tienen una vida útil. Por ejemplo, en nuestra lista de precios de equipos de audio cada tapio ocurrirá que determinados modelos o marcas dejan de fabricarse (se discontinuan) y por lo tanto deja de interesar el ocupar un lugar en la lista de precios (y por lo tanto en el disco) con esos modelos. Interesará entonces eliminar o "dar de baja" a esos artículos. liberando así espacios útiles en el disco. Por otra parte, si no pudiésemos dar de baja registros, como siempre se crean nuevos modelos, llegaría elmomento en que la capacidad del disco se saturaría (esto ocurrirá tarde o temprano sin importar la capacidad del mismo).

Por ello, todo archivo siempre debe poder ser "purgado" de aquellos registros que se vuelven obsoletos

Al eliminar un registro de un archivo quedará un lugar en blanco como si borraremos una línea escrita en la hoja de un libro, o sea: en realidad no se borra el registro propiamente dicho sino a su contenido. El objetivo del diagrama será entonces "cerrar" ese espacio vacío haciendo que todos los contenidos de los registros que siguen a continuación del eliminando se corran un lugar para "tapar" el espacio vacío, como se ve en la siguiente ilustración.

```
Códigos                              Códigos
Artículos                            Artículos
42832                                42832
51621                                51621
55742  ……registro a eliminar        ……registro eliminado
60221                                60221
61466                                61466
67678                                67678

Códigos
Artículos
42832
51621
60221    ……Todos los registros inferiores se han corrido un lugar "hacia arriba" para
61466          "cerrar" el espacio en blanco dejado por el registro eliminado
67678
```

Analicemos un diagrama de flujo en el cual, dado un código de artículo a ser dado de baja el computador lo buscará en el archivo mediante una búsqueda dicotómica y luego, a partir de ese registro, grabará el resto de los registros del archivo corriéndolos un lugar "hacia arriba".

Siendo este el último punto operativo del menú, una vez codificado el mismo, el programa quedará terminado. Por ello, la codificación o pasaje a "**BASIC**" de este diagrama de flujo se deja a cargo del lector como ejercicio. Se debe recordar que las instrucciones de este diagrama deben comenzar con el paso No 4000 y las subrutinas a partir del paso 10530. Decimos que este es el último punto operativo del menú, pues que resta (listados) no modifica el archivo de lista de precios, sino que lo volcará a un formulario are-opiado a través de un impresor. En el pasaje a "**BASIC**" del diagrama de bajas conviene también observar cuales son las subrutinas útiles aprovechables de la codificación ya realizadas para las opciones anteriores.

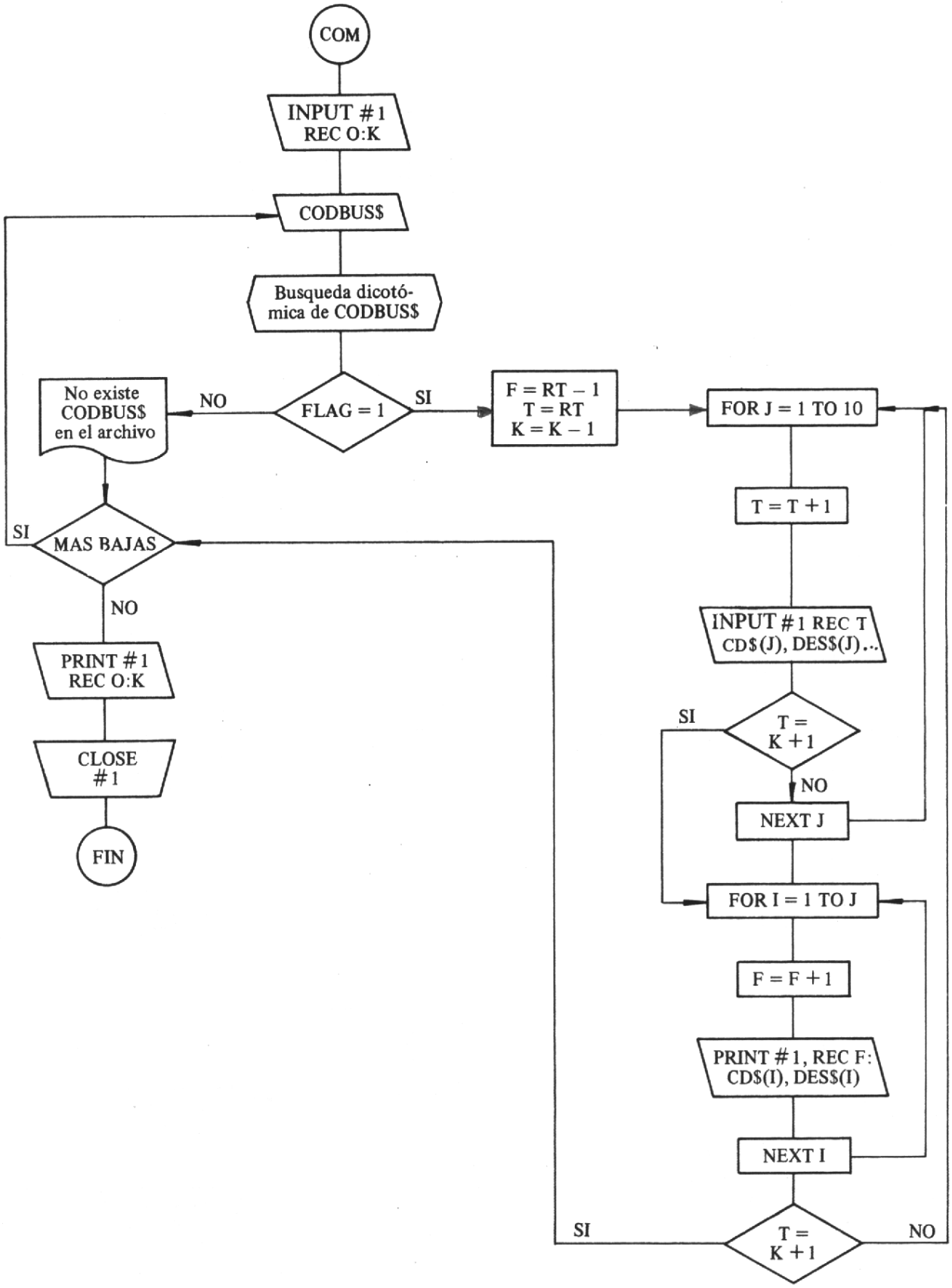
Por ejemplo, aquí se va a practicar una búsqueda dicotómica en un archivo y en la opción de modificaciones también se efectuó la misma búsqueda. Por lo tanto, sería sumamente interesante que, a través de una subrutina se unificarán ambas búsquedas. Si se piensa en el problema se verá que se trata de algo realmente simple pues bastará con saber definir cuales son los parámetros que gobiernan a la subrutina en cuestión.

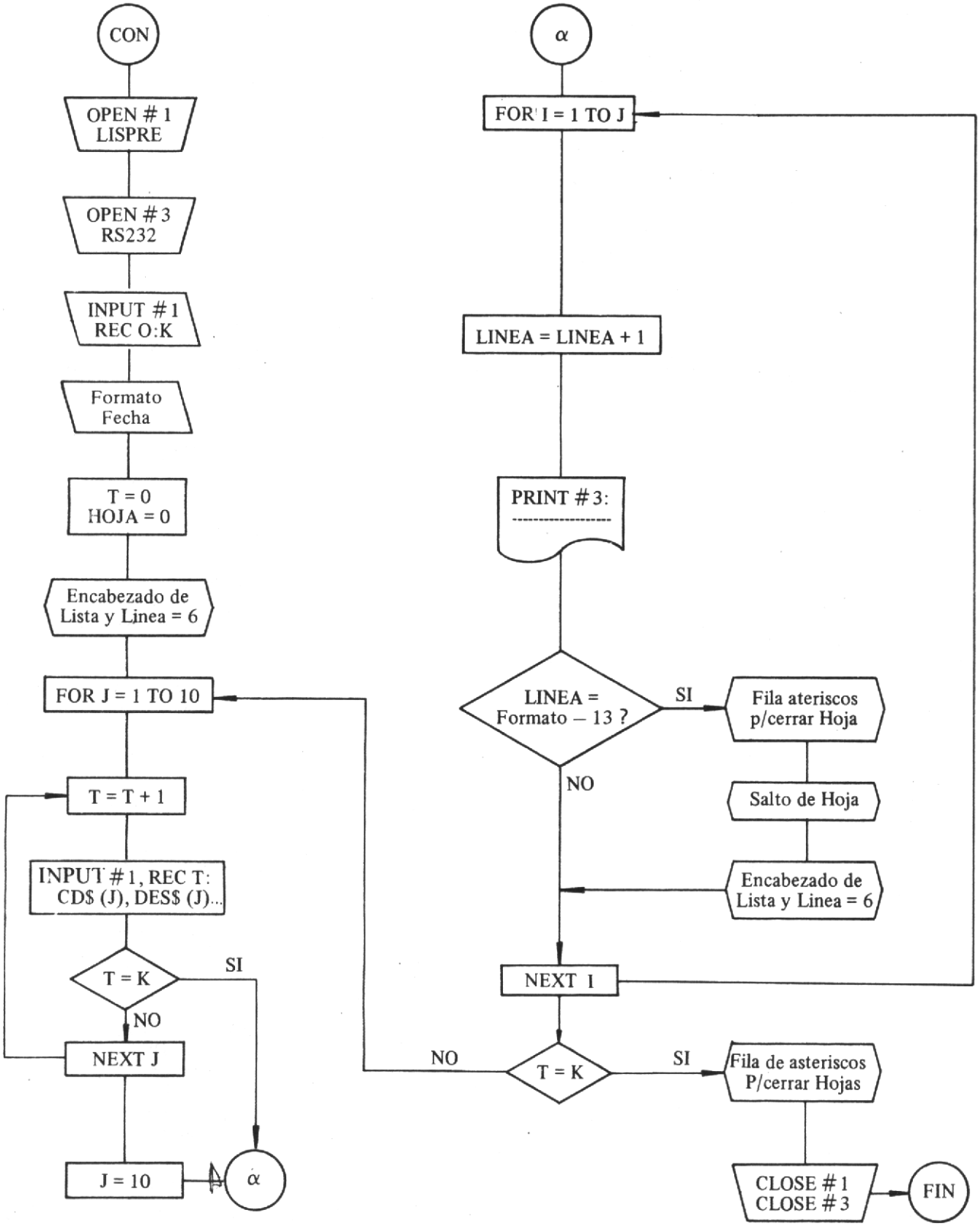
Listados varios del contenido de un archivo: Esta es la opción número 5 del menú principal del programa, y de todas las restantes, es una de las que más nos interesa pues es justamente la que nos permitirá producir finalmente a la expresión escrita o listado vía impresor de la lista de precios. Todos los trabajos que se han realizado sobre el archivo de lista de precios como ser: carga, modificación, etc..., tienen como finalidad el mantener la información al día del tal forma que pueda ser posteriormente listada y enviada ya sea a clientes o sucursales del comercio donde se aplica así como también puede ser "consultada" por el mismo computador a través de un programa de facturación que relacione a un archivo de remitos de mercadería con el de lista de precios. Se pueden pensar dos formas básicas muy útiles para generar listados. Estas son: (a) listado total de la lista de precios y (b) listado de algunos ítems solamente (seleccionables mediante un código ó combinación de códigos).

En este último caso estamos ante el mismo problema que cuando se deseaban modificar los precios de la lista a través de una selección de códigos que identificaban a la familia, la marca o el modelo. Más aún, el problema también se solucionará revisando todos los códigos de la lista y listando (en lugar de modificando) a todos aquellos registros cuyos código sea congruente con el definido en la clave selectiva de listado.

Por ello, dejaremos el desarrollo (programa y diagrama) de esta sub-operación en manos del lector que deberá repasar y compenetrarse con el trabajo que realiza el diagrama de flujo de la opción de modificaciones selectivas (por claves).

En cambio sí presentamos el diagrama de flujo y codificación del listado total de la lista de precios.


COM
INPUT #1
REC O:K
CODBUS$
Busqueda dicotó-
mica de CODBUS$
No existe
CODBUS$
en el archivo
NO
FLAG = 1
SI
F = RT – 1
T = RT
K = K – 1
FOR J = 1 TO 10
T = T + 1
INPUT #1 REC T
CD$(J), DES$(J)...
SI
T =
K + 1
NO
NEXT J
FOR I = 1 TO J
F = F + 1
PRINT #1, REC F:
CD$(I), DES$(I)
NEXT I
SI
T =
K + 1
NO
SI
MAS BAJAS
NO
PRINT #1
REC O:K
CLOSE
#1
FIN


CON
OPEN # 1
LISPRE
OPEN # 3
RS232
INPUT # 1
REC O:K
Formato
Fecha
T = 0
HOJA = 0
Encabezado de
Lista y Linea = 6
FOR J = 1 TO 10
T = T + 1
INPUT # 1, REC T:
CD$ (J), DES$ (J)...
T = K
SI
NO
NEXT J
J = 10
α
α
FOR I = 1 TO J
LINEA = LINEA + 1
PRINT # 3:

LINEA =
Formato – 13 ?
SI
Fila ateriscos
p/cerrar Hoja
NO
Salto de Hoja
Encabezado de
Lista y Linea = 6
NEXT I
NO
T = K
SI
Fila de asteriscos
P/cerrar Hojas
CLOSE # 1
CLOSE # 3
FIN

Si se analiza un poco el diagrama de impresión de resultados se verá que se utiliza una técnica de bloqueo de registros ya utilizada en otros diagramas (circuito del FOR J = 1 TO 10) y otro circuito de volcado de ese bloque al impresor el cual fue abierto (OPEN) como archivo 3. Su particularidad es que lleva un control de líneas impresas que le permite saltar de hoja cuando se ha llegado al final de formato. En este sentido, es conveniente aclarar que el contador de líneas impresas es la variable llamada "línea" y el contador de hojas es la variable llamada "hoja". La variable "linea" se inicializa cada vez que se imprime un nuevo encabezado (cada vez que se salta de una hoja a la siguiente).
También es conveniente aclarar que este programa permitirá listar la lista de precios en diferentes formatos (tamaños) de papel. Los papeles utilizados en computación pueden tener diferente cantidad de renglones, y esto se ingresará en la máquina en la variable "formato". Si, por ejemplo la hoja que se utiliza tuviese 72 líneas en total y descasemos dejar 3 líneas en blanco como margen superior y 3 como margen inferior, entonces, la cantidad de líneas útiles será de: 72 - (3 + 3) = 66. Además, en estas 66 líneas útiles habrá por ejemplo: 6 ocupadas por encabezados y títulos al principio de página y 1 al final del listado consistente en una fila de asteriscos que "cerrarán" prolijamente el listado de cada hoja.
O sea: habían: 72 - 6 - 6 - 1 = 59 líneas disponibles para impresión de datos propiamente dicho. Veamos el diagrama codificado en BASIC, arrancando desde el paso Nº 5000.

```
5000 OPEN #1: "DSK1.LISPRE", RELATIVE, INTERNAL, UPDATE, FIXED 85
5010 INPUT#1, REC O:K
5020 OPEN #3: "RS232.BA = 9600", VARIABLE 132
5030 DISPLAY AT (3,2) ERASE ALL: "FORMATO DEL FORMULARIO:?" :: DISPLAY AT (5,2):
     "FECHA DE HOY? (DD/MM/AA):"
5040 DISPLAY AT (11,2): "DATOS OK? (S/N):"
5050 ACCEPT AT (3,26) VALIDATE (DIGIT) SIZE (2) BEEP: FORMATO : : ACCEPT AT (6,10)
     VALIDATE (DIGIT, "/", "-") SIZE (8) BEEP: FECHA$
5060 ACCEPT AT (11,18) VALIDATE ("SN") SIZE (1) BEEP:YN$ :: IF YN$ = "N" THEN 5030
5070 T, HOJA = 0 :: GOSUB 12000 :: REM IMPRIME EL ENCABEZAMIENTO, CON N° DE
     HOJA, TITU LOS, ETC., INICIALIZA EL CONTADOR DE LINEA Y AVANZA AL CONTADOR
     DE HOJA EN UNA UNIDAD.
5080 FOR J = 1 TO 10 :: T = T + 1 :: GOSUB 10270
5090 IF T = K THEN 5110
5100 NEXTJ :: J=10
5110 FOR I = 1 TO J :: LINEA = LINEA + 1 :: GOSUB 12100 :: REM IMPRIME UNA LINEA DE
     DATOS
5120 IF LINEA = FORMATO - 13 THEN GOSUB 12200 :: REM IMPRIME ASTERISCOS, SALTA
     HOJA Y REENCABEZA
5130 NEXT I
5140 IF T = K THEN GOSUB 12300 ELSE 5080
5150 CLOSE # 1 :: CLOSE # 3 :: GO TO 100
```

Nota a las subrutinas de impresión de listado (pasos 12000 en adelante): Para poder encarar el listado de un informe, en primer lugar hay que definir cual va a ser la disposición de las columnas (tabulaciones), el diseño de encabezado, etc. Recién cuando se han resuelto estos problemas o interrogantes se puede proceder al programado del reporte propiamente dicho. Haremos entonces, un croquis o diseño de impresión de los datos a volcar.

Este croquis se puede apreciar en la siguiente ilustración; y en el mismo figuran las columnas que se utilizarán para imprimir los encabezados y los títulos de columnas. Las líneas llenas pueden representar asteriscos, numerales, etc.
Este papel especial para el diseño de impresiones presta ayuda muy importante en la posterior programación.
Ahora armaremos el programa de impresión:

```
12000  HOJA = HOJA + 1 ::  LINEA = 7 :: PRINT # 3: RPT$ ("*", 90)
12010  PRINT #3: "*"; "FECHA "; FECHA$; TAB(76); "HOJA NRO:"; HOJA; TAB(90); "*"
12020  PRINT #3: "*";       “---------"; TAB(36); "LISTA DE PRECIOS"; TAB(76); RPT$("-
       ",9); TAB(90); "*"
12030  PRINT #3: "*"; TAB(90); "*"; RPT$("*", 90)
12040  PRINT #3: "*"; TAB(3); "CODIGO"; TAB(36); "DESCRIPCION"; TAB(71);
       '"PREC.LIST"; TAB(81); "PREC.PUB."; TAB(90); "*"
12050  PRINT#3"*"; TAB(3); "------------"; TAB(36); "------------"; TAB(71); "------------";
       TAB(81); "-----------" ; TAB(90); "*"
12060  RETURN
12100  PRINT #3: "*"; TAB(2); CD$(I); TAB(11); DE$(I); TAB(71); PR(I); TAB(81); P1(I)
12110  RETURN
12200  PRINT #3: RPT$("*", 90) :: PRINT#3: " "; " " ; " “; “ “; “ “; “ “
12210  GOSUB 12000
12220  RETURN
12300  PRINT#3: RPT$ ("*", 90) :: RETURN
```

FECHA	**LISTA DE PRECIOS**		**HOJA Nº**
CODIGO	**DESCRIPCION**	**PREC. LIST.**	**PREC. PUB.**

Apéndice sobre los métodos de ordenamientos por claves en archivos

Existen muchos casos en que resulta útil el poder listar ciertos archivos en base a diversos ordenamientos. Veamos el siguiente ejemplo:
Supongamos que en un archivo de ventas se acarrean los siguientes datos:

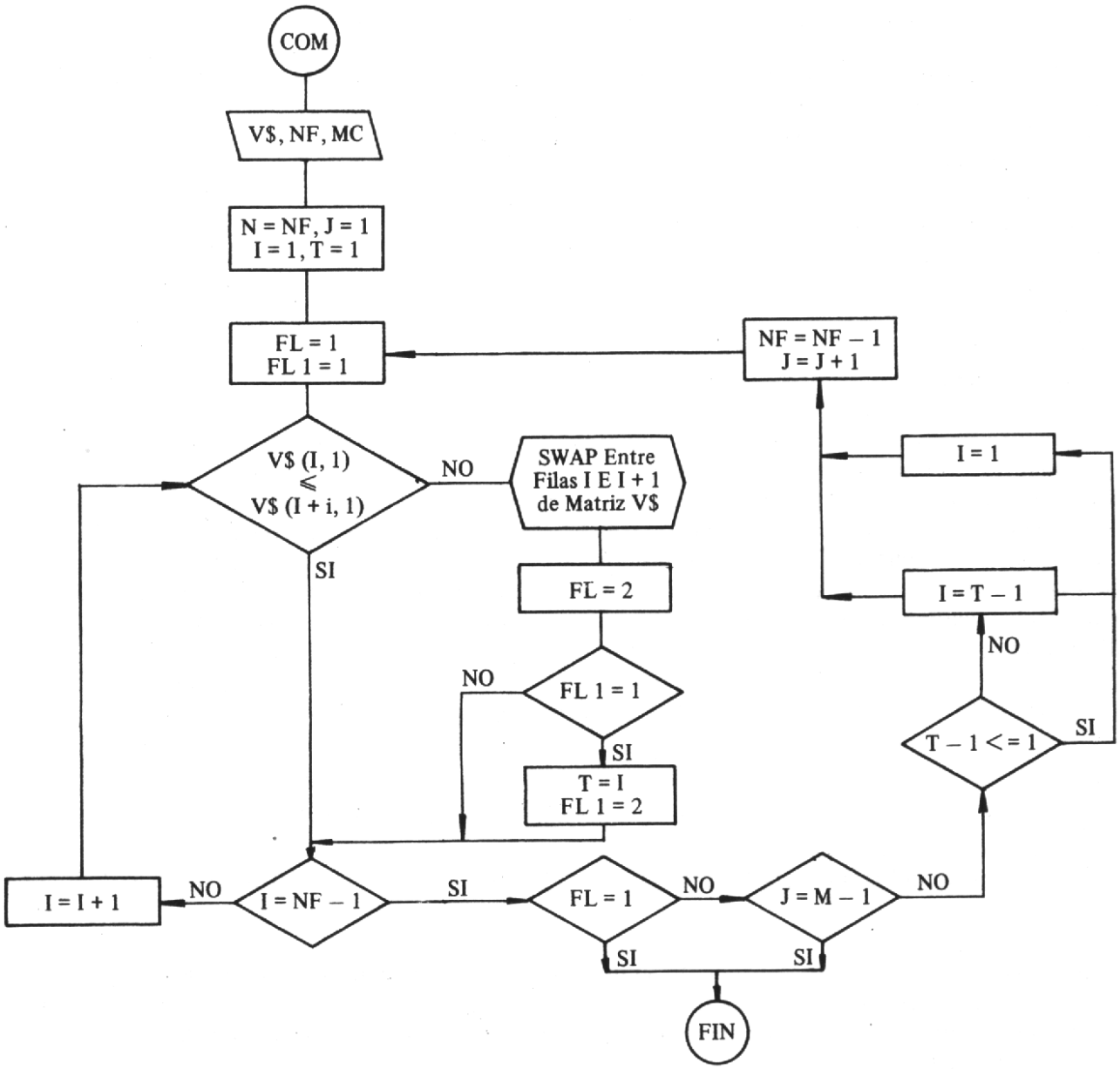
1 - Código del cliente
2 - Nombre del cliente
3 - Código del vendedor
4 - Código de artículo
5 - Cantidad vendida
6 - Precio
7 - Fecha de venta
8 - Condición de pago

Este archivo se va cargando diariamente y constituye la base para una posterior generación e impresión de facturas, trabajo que se lleva a cabo una vez por semana. Pero, además de la oficina de facturación al departamento de planificación industrial le interesan estos datos, ordenados o agrupados por número o código de artículo (a la facturación solo le interesan agrupados por código de cliente). Además a la gerencia de ventas le interesan los mismos datos pero ordenados de diversas formas: por código de artículo, por vendedor, por condición de pago, etc., pues son todos ordenamientos que podrían ayudar notablemente a la toma de decisiones en lo que hace a estrategias de mercado, artículos más vendidos, clientes y formas de pago más convenientes, etc.
Por otra parte, este archivo esta ordenado por código de cliente, por lo que, si desearemos algún otro tipo de ordenamiento deberemos re-ordenar sus registros siguiendo algún método a definir.

Cuando se desea ordenar un archivo residente en discos pueden ocurrir dos casos:

(1) Que todas las claves a ordenar "entren" o "quepan" en la unidad central de proceso de la computadora bajo la forma de un arreglo (vector o matriz).
(2) Que el tamaño del archivo sea tan grande que no "quepan" las claves en la UCP según lo definido en (1).

El primer caso nos exigirá un ordenamiento simple; en cambio el segundo caso nos obligará a un ordenamiento de mayor envergadura, trabajando los registros del archivo por bloques. Este último método es tema fundamental en un curso de técnicas avanzadas de programación, por lo que aquí solo describiremos el primer caso.
Ante todo, veamos como opera un diagrama de ordenamiento de una matriz (arreglo de 2 dimensiones) de dos columnas y "N" filas.
El diagrama presentado sirve para el sort en Ram de una matriz alfanumérica de 2 columnas. El diagrama ordenará a la primer columna de menor a mayor y, por inducción, a la segunda.
El procedimiento para aplicar este diagrama de flujo al ordenamiento de un archivo es como sigue: Se carga la primer columna de la matriz V$ con los valores correspondientes a la clave de ordenamiento seleccionada. Estos valores se toman de los registros del archivo a ser ordenado. Simultáneamente se carga la segunda columna de la matriz V$ con los números de registros de los cuales provienen esas claves. Luego se produce el ordenamiento de la matriz según lo muestra el diagrama presentado y, finalmente se produce el listado por impresor de los registros del archivo seleccionados por la segunda columna de la matriz ordenada. Veamos el programa que se generaría:



El registro tipo de nuestro archivo de ventas estará constituido por variables alfanuméricas (strings) y serán las siguientes:

- Código de cliente A$(1)	4	caracteres
- Nombre y apellido cite. A$(2)	30	caracteres
- Código de vendedor A$(3)	4	caracteres
- Código de artículo A$(4)	4	caracteres
- Cantidad vendida A$(5)	9	caracteres
- Precio A$(6)	9	caracteres
- Fecha de venta A$(7)	8	caracteres
- Condición de pago A$(8)	2	caracteres

Se desea poder producir listados ordenados por: código de cliente, código de vendedor, código de artículo y condición de pago.
El programa que desarrollaremos a continuación contendrá en su primer parte un menú donde se permitirá seleccionar el tipo de listado deseado.

```
100  DIM V$(500,2)
110  DISPLAY AT (2,2) ERASE ALL: "MENU DE LISTADOS ORDENADOS":RPT$ ("-",26)
120  DISPLAY AT (6,1): "1.- POR COD. DE CLIENTE": " ": "2.- POR COD. DE VENDEDOR:" " :
     "3.- POR COD. DE ARTICULO": “”
130  DISPLAY AT (12,1): "4.- POR CONDIC. DE PAGO": " ": "5.- FIN DE TRABAJO" " “ : ” " :  “ "
     :" TAB(5) ;  "SU ELECCION?:"
140  ACCEPT AT (18,19) VALIDATE (:"12345:") SIZE (1) BEEP: ELEC : : IF ELEC = 5 THEN
     END
150  OPEN # 1: "DSK1.VENTAS:", RELATIVE, INTERNAL, UPDATE, FIXED 70
160  INPUT # 1, REC O:K
170  ON ELEC GO TO 180, 190, 200, 210
180  F = 1 :: A$ = "LISTADO POR CODIGO DE CLIENTE" :: GO TO 220
190  F = 3 :: A$ = "LISTADO POR CODIGO DE VENDEDOR:" :: GO TO 220
200  F = 4 :: A$ = :"LISTADO POR CODIGO DE ARTICULO:" :: GO TO 220
210  F = 8 :: A$ = :"LISTADO POR CODIGO DE PAGO"
220  REM AQUI COMIENZA LA CARGA DE LA MATRIZ V$
230  FOR I= TO K
240  INPUT # 1, REC I: A$(1), A$(2), A$(3), A$(4), A$(5), A$(6), A$(7), A$(8)
250  V$(I, 1) = A$(F) :: V$(I, 2) = STR$ (I)
260  NEXT I::I=K
270  REM AQUI COMIENZA EL ORDENAMIENTO DE LA MATRIZ
280  M, NF = K : : J, I, T = 1
290  FL, FL1 = 1
300  IF V$(I, 1) = < V$(I + 1,1) THEN 340
310  A$ = V$(I, 1) :: V$(I,1) = V$(I + 1,1) :: V$ (I + 1,1) = A$ :: A$ = V$(I, 2) :: V$(I, 2) = V$(I + 1,2)
     :: V$ (I + 1,2) :: V$(I + 1,2) = A$ :: FL = 2
320  ON FL1 GO TO 330, 340
330  T = I :: FL1=2
340  IF I = NF --1 THEN 350 ELSE I = I + 1 :: GO TO 300
350  IF FL= 1 THEN 390
360  IF J = M-1 THEN 376
370  IF  T - 1 = < 1 THEN I =1 ELSE I = T - 1
380  NF = NF - 1 :: J = J + 1 ::GOTO290
390  REM COMIENZAN OPERACIONES DE IMPRESION DEL ARCHIVO ORDENADO
400  T = 0, HOJA= 0 :: DISPLAY AT (10,2) ERASE ALL: INGRESE LAS LINEAS
DEL PAPEL: " :: ACCEPT AT (13,10) VALIDATE (DIGIT) SIZE (2) BEEP: LIN
410  OPEN # 3: "RS232.BA = 9600", VARIABLE 132
420  GOSUB 10000 :: REM IMPRIME ENCABEZADOS, ACTUALIZA :"HOJA:" E INICIALIZA
     "LINEA"
```

```
430   GOSUB 10200 :: REM BLOQUEA 10 REGISTROS TOMANDO LOS SEGUN V$ (xx,2), HACE T=T+1
440   GOSUB 10300 :: REM IMPRIME EL BLOQUE, CONTROLA LINEAS Y SALTOS DE HOJA
450   IF T = M THEN 460 ELSE 430
460   PRINT #3: RPT$ ("*", 80)
470   CLOSE #1 :: CLOSE #3 :: GOTO 110
10200 FOR P = 1 TO 10 :: T = T + 1 :: RE = VAL (V$(T, 2) )
10210 INPUT #1, REC RE: Q$(P,1) Q$(P,2), Q$(P,3), Q$(P,4), Q$(P,5), Q$(P,6), Q$(P, 7), Q$(P, 8)
10220 IF T = M THEN RETURN
10230 NEXT P :: P = 10 :: RETURN
10000 HOJA = HOJA + 1
10002 LINEA = 6
10004 PRINT #3: RPT$("*",80); TAB(27); :"PARTE DIARIO DE VENTAS"; TAB(65); "HOJA NR:"; HOJA; TAB(80); "*"
10006 PRINT #3: "*"; TAB(65); "FECHA:"; FCA$; TAB(80); "*"; RPT$("*:")
10008 PRINT #3: "*"; "CODC"; TAB(10); "NOMBRE Y APELLIDO CLIENTE"; TAB(39); "VEND"; TAB(45); "ART."; TAB(50); "CANTID.:"; TAB(58); "PRECIO"; TAB(67); "FECHA"; TAB(76); "COND."; TAB(80);:"*"
10010 PRINT #3: RPT$ ("*", 80)
10012 RETURN
10300 FOR P5 = 1 TO P :: LINEA = LINEA + 1
10302 PRINT #3: "*"; TAB(2); Q$(P5,1); TAB(7); Q$(P5,2); TAB(10); Q$(P5,3); TAB(39); Q$(P5,4); TAB(45); Q$(P5,5); TAB(50); Q$(P5,6); TAB(58); 0$ (P5,7);
10304 PRINT #3 : TAB(67); Q$(P5,8); TAB(76); Q$(P5,9); TAB(80); "*"
10306 IF LINEA = LIN - 7 THEN 10310
10308 NEXT P5 :: RETURN
10310 PRINT #3 :: RPT$ ("*",80): " " : " " : " " : " " : " " : " " : " " : " " :
10320 GOSUB 10000
10330 GO TO 10302
```

Este programa resulta bastante eficiente para lograr el fin deseado. Se podría reemplazar toda la subrutina de ordenamiento de la matriz V $ por algún método de mayor velocidad. Esto se deja para la curiosidad e iniciativa del lector.

		PARTE DIARIO DE VENTAS			**HOJA Nº** **FECHA:**	
CODE	**NOMBRE Y APELLIDO CLIENTE**	**VEND.ART**	**CANT.**	**PRECIO**	**FECHA**	**COND**

INDICE

NOTA:

TEXAS INSTRUMENTS no garantiza que los programas estén libres de error o que coincidirán con los requerimientos específicos del consumidor. El consumidor asume completa responsabilidad por cualquier decisión o acción tomada en base a información obtenida usando los programas: Cualquier afirmación previa realizada con relación a la utilidad de los programas no debe tomarse como garantía expresa o implícita.

Este manual se terminó de imprimir el 15-11-83 en Gráfica Farinelli, Olegario Andrade 3848, J. León Suarez, Rep. Argentina.

Texas Instruments inventó el circuito integrado
el microprocesador y la microcomputadora
Ser primero es nuestra tradición.

TEXAS INSTRUMENTS

ARGENTINA S.A.I.C.F.

IMPRESO EN ARGENTINA

1983