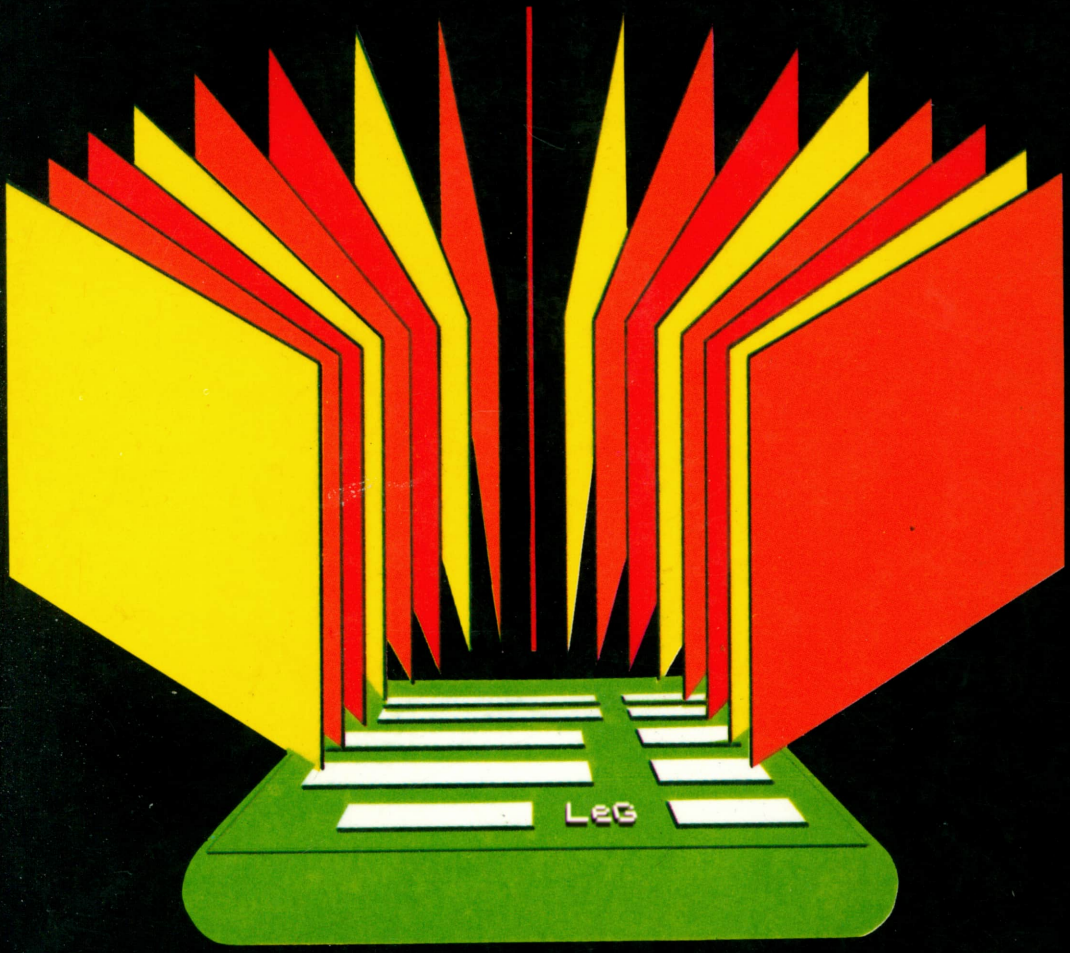
FICHEROS EN BASIC



C. Delannoy

PARANINFO SA

Ficheros en BASIC

Claude Delannoy

Ficheros en BASIC

SEGUNDA EDICION

1985

PARANINFO S.A.

MADRID

Traducido por:
LUIS CAMPOY GUILLEN
FELISA MATEO GARCIA

Esta obra es la traducción del libro francés de C. DELANNOY
"LES FICHIERS EN BASIC SUR MICRO-ORDINATEUR"
publicado por Editions EYROLLES, París (Francia)

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 84-283-1355-5

Depósito Legal: M-5.348-1985

PARANINFO S.A. Magallanes, 25 - 28015 MADRID (5-3512)

Impreso en Lavel. Los Llanos, nave. Humanes (Madrid)

Indice

Prefacio

El objeto de este manual de consulta, es proporcionar los medios necesarios para la creación y utilización de ficheros adaptados a sus necesidades. A través de él, descubrirá y conocerá un archivo o fichero informático y su diferencia con un archivo manual. Le iniciará en las técnicas de los accesos secuencial, directo e indexado. El conocimiento de sus respectivas ventajas e inconvenientes le permitirá escoger el método que mejor se adapte a los requerimientos de su aplicación.

Se ha adoptado un sistema muy progresivo, así los conceptos fundamentales se completan con ejemplos concretos y sencillos, escogidos de entre los más comunes, tales como una guía o repertorio telefónico y las fichas de los libros de una biblioteca.

Voluntariamente, nos limitaremos a tratar las instrucciones Basic utilizadas comúnmente sobre la mayoría de los micro-ordenadores. Podrá así utilizar y adaptar fácilmente los programas propuestos sobre su propia máquina. Para fijar ideas, diremos que hemos empleado el Basic de Microsoft con algunas peculiaridades específicas tales como: posibilidad de omitir la palabra LET en las instrucciones de mandato, sustitución de REM por el carácter especial ' , y la escritura de nombres de variable con más de 8 caracteres.

Cualquiera que sea la manera de consultar este manual, insistiremos en recomendarle que evite en lo posible la redacción de programas complejos. Comience por escribir pequeños ejemplos que le permitirán asimilar los conceptos básicos. Antes de abordar una realización personal, deberá haber estudiado los diferentes tipos de acceso, de este modo podrá escoger el método adecuado con conocimiento de causa.

Finalmente, trate no solamente de codificar los programas propuestos, sino de hacer uso de ellos lo más posible. Familiarizándose con ellos, descubrirá también las cualidades y los defectos del "diálogo" que establecerán con Vd. Así estará en mejores condiciones para definir sus propios programas.

1

Generalidades

1.1. INTRODUCCION

Seguramente Vd. ya conoce lo que es un fichero en el sentido más usual del término. Sin embargo, el concepto de fichero informático puede que le sea menos familiar. En este capítulo, intentaremos mostrarle, no sólo lo que tienen en común los ficheros informáticos con los ficheros usuales (que llamaremos manuales), sino también las diferencias existentes entre ambos.

Comprenderá fácilmente, que el término registro no es otro que el equivalente de la ficha manual. Por el contrario, descubrirá los conceptos de acceso secuencial o directo, propio de los ficheros informáticos.

1.2. FICHEROS MANUALES

Recordemos los principios esenciales, lo cual nos facilitará la introducción en los ficheros informáticos.

1.2.1. ¿Qué es un fichero manual?

Es un conjunto de fichas (de ahí el nombre de fichero), donde cada una contiene una cierta cantidad de información. Por ejemplo, un bibliotecario dispondrá para uno de sus libros de una ficha sobre la cual estará consignado:

– el número de la obra,
– el nombre del autor,
– el título de la obra,
– la naturaleza de la obra (novela, histórica, científica, etc...),
– la fecha de adquisición,
– etc...

Generalmente, y aunque no es obligatorio, todas las fichas contienen el mismo tipo de datos. (En nuestro ejemplo: nombre del autor, título de la obra, etc...).

1.2.2. ¿Qué operaciones se realizan sobre un fichero manual?

1. En principio es necesario "crear", es decir, rellenar todas las fichas que constituyen el fichero (en un momento dado).
2. A continuación se puede consultar el fichero con los fines siguientes:

   a) Para buscar la información contenida en una ficha, conociendo por ejemplo el número de libro o su título. En este punto se puede observar que la consulta será más o menos eficaz y rápida, dependiendo de la forma en que el fichero esté organizado:

   – Si las fichas se encuentran dispuestas de forma aleatoria, se deberán examinar una tras otra, hasta encontrar la deseada. Este concepto de acceso "secuencial" lo encontrará de nuevo en ciertos ficheros informáticos.

   – Si, por el contrario, las fichas se encuentran dispuestas en forma ordenada según el número de libro o por orden alfabético según el nombre del autor, la búsqueda será más fácil. Se podrá proceder por aproximaciones sucesivas, al igual que cuando se desea buscar una palabra sobre un diccionario.

   b) Para extraer o catalogar la información relativa a un cierto tipo de fichas, por ejemplo:

   – Contar el número de libros del tipo novela
   – Confeccionar una relación de las obras de un determinado autor
   – Confeccionar una relación de las obras adquiridas después de una determinada fecha
   – etcétera....

Hasta aquí la consulta se deberá realizar casi siempre de forma secuencial.

3. Se puede igualmente poner "al día" o actualizar el fichero:
   - Añadiendo nuevas fichas. En forma manual, esto se puede efectuar sin importar donde alojarlas. De hecho, siempre se puede situar una ficha nueva entre otras dos ya existentes.
   - Suprimiendo una ficha.
   - Modificando el contenido de una ficha.

1.3. CONCEPTO DE FICHERO INFORMATICO

El soporte de información de un fichero manual es generalmente el papel. Su manipulación es sencilla y el usuario puede:

- Acceder a cualquier ficha, siguiendo cualquier método (secuencial, al azar, ...),
- Examinar cualquier información detenidamente,
- Sustituir una información por otra sin preocuparse (demasiado) de su "dimensión". Así, podrá sustituir un título de 20 caracteres por otro de 125, sin que ello le reporte problema alguno.

En cuanto a los ficheros informáticos se refiere, su soporte es generalmente de tipo magnético, sean cintas o discos. Sin entrar demasiado en detalle sobre la codificación de la información, es evidente que estos últimos soportes no pueden manipularse directamente. Se hace pues necesario, el uso intermedio de los programas:

- Unos programas se encargarán de la creación del fichero a partir de los datos suministrados por el usuario (generalmente a través de un teclado). Diremos de tales programas que "escriben" los distintos registros del fichero. Observe la evolución del vocabulario: el término "ficha" está siendo sustituido por el de "registro".
- Otros programas tendrán como fin, la actualización o consulta del fichero, a partir de las normas o directrices suministradas por el propio usuario.

Por otra parte el tipo de soporte utilizado, (cinta magnética o casete, disco o diskete), influye de manera fundamental sobre el modo en que

estos programas podrán acceder a la información contenida en los ficheros de:

- Acceso secuencial para cintas o casetes.
- Acceso directo para discos o disketes.

1.4. EL ACCESO SECUENCIAL

1.4.1. Generalidades

La propia naturaleza de una cinta mangética o de un casete implica que los registros (fichas), sean almacenados (todavía diremos escritos) unos a continuación de los otros. Al consultar estos ficheros, se podría pensar que es posible posicionarse relativamente cerca de un registro solicitado (del mismo modo en que Vd. lo hace con su magnetófono o su reproductor de casetes). Desgraciadamente, existen razones esencialmente técnicas por las cuales esto no es posible con los soportes informáticos. Por tanto, para acceder a un determinado registro, es necesario "leer" todos los que le preceden. A título de ejemplo, esto se correspondería con el caso en que nuestro bibliotecario buscara la ficha relativa a un título determinado, examinando sucesivamente cada una de las fichas a partir de la primera. De otro lado, la marcha hacia atrás no es posible (al menos en Basic). Ello significa que si se desea leer un registro situado con anterioridad al último "leído", es indispensable comenzar la exploración o búsqueda sobre el fichero a partir del comienzo del mismo.

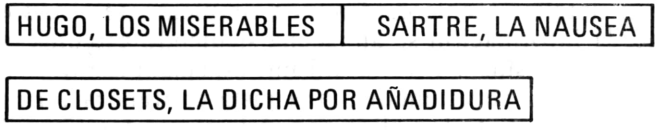
1.4.2. Ejemplo

Como ilustración, establezcamos una semejanza entre fichero manual y fichero informático para el caso en que nuestro bibliotecario disponga de tres fichas solamente.

El fichero manual aparecería así:

libro número 1 autor : HUGO título: LOS MISERABLES	libro número 2 autor: SARTRE título: LA NAUSEA	libro número 3 autor: DE CLOSETS título: LA DICHA POR AÑADIDURA

El fichero informático secuencial contendrá información codificada. De hecho, cada carácter se representará generalmente por un octeto (8 bits). Bajo estas condiciones, se puede considerar este fichero como una cadena de caracteres. De manera aproximada, los representaremos así:



Observaciones

1. Al diseñar los registros, no es preciso el determinar cómo se encuentran realmente separados dentro del fichero. Hablaremos de ello más adelante.

2. La ficha manual contiene detalles como el número de libro, autor, título, que no se encuentran generalmente dentro del registro. Es suficiente el establecer el orden en el cual se encuentra cada información. (Además, aquí el número de libro no figura ya; se ha supuesto que esta información correspondía al "número" o rango del registro.

1.4.3. Limitaciones

Según se ve, la "inserción" de un registro (correspondiente a la inserción de una ficha entre otras dos), es imposible al igual que la operativa de modificación. Ciertamente se podría pensar en nuestro ejemplo, en reemplazar o sustituir "LA NAUSEA" por "EL MURO" (que contiene menos caracteres). Por el contrario, no parece factible el sustituir "LOS MISERABLES" por "NUESTRA SEÑORA DE PARIS".

Para suprimir un registro, parecería suficiente el "borrarle". Por ahora, razones de tipo tecnológico, hacen imposible esa simple operativa.

En definitiva, los ficheros secuenciales en Basic no pueden ser actualizados. Entonces, pensará Vd., que estos ficheros son poco interesantes en general. Verá que la dificultad anterior se puede obviar con la creación de un segundo fichero, a partir del fichero inicial y de las informaciones de actualización.

1.5. EL ACCESO DIRECTO

1.5.1. Generalidades

Sin entrar en detalles técnicos, digamos simplemente que sobre un disco o diskete, la información se almacena siguiendo círculos concéntricos denominados "pistas". Un mecanismo especial permite acceder a una pista determinada. Seguramente, una pista contendrá varios registros. De todas formas, si todos ellos son del mismo tamaño, el Basic podrá localizar cualquier registro de una pista dada. Podrá encontrar, por ejemplo, el registro de orden 25 ó 58 de un fichero. (Para ello es necesario naturalmente conocer, en lo que se conoce como directorio, el número de pista donde comienza el fichero.

Resumiendo, en un fichero de acceso directo se puede localizar cualquier registro, sin necesidad de examinar otro. Esto constituye una gran ventaja (ahorro notable de tiempo de proceso) frente a los ficheros secuenciales. No obstante, es necesario que todos los registros sean de igual tamaño o longitud (igual número de caracteres); esta restricción no afecta a los ficheros secuenciales.

1.5.2. Ejemplo

Consideremos de nuevo nuestro fichero de biblioteca con tres libros. Esta vez, el fichero de acceso directo presentará una disposición en forma de tres registros del mismo tamaño, cada uno de los cuales contendrá el nombre del autor y el título. Por otra parte, dentro de cada registro se deberá prever un número de caracteres reservados para cada tipo de información (esta limitación no parece evidente, ya que se podría pensar el utilizar separadores como en secuencial). Veremos más adelante que la propia naturaleza de la codificación de la información impone en parte el uso de esta técnica.

Supongamos que escogemos registros de 64 caracteres desglosados de la siguiente forma:

- 16 caracteres para el nombre del autor,
- 48 caracteres para el título.

Nuestro fichero informático podría representarse así:

```
registro número 1
HUGO .......... LOS MISERABLES
registro número 2
SARTRE ......... LA NAUSEA
registro número 3
DE CLOSETS ..... LA DICHA POR AÑADIDURA
```

Los símbolos (.) representan los caracteres a "blancos" o "espacios".

1.5.3. Posibilidades

El acceso directo permite la modificación de un registro en la medida en que la información modificada se encuentre dentro de los límites impuestos en la creación del fichero. De esta forma, en el ejemplo precedente podríamos sustituir "LOS MISERABLES" por "NUESTRA SEÑORA DE PARIS".

Por otro lado y a diferencia de los ficheros secuenciales, los ficheros de acceso directo pueden agrandarse añadiendo registros al final del fichero. No obstante, la inserción en medio del fichero se hace imposible.

1.5.4. Observaciones importantes

a) Quien puede hacer lo difícil, puede hacer lo fácil. Dicho de otro modo, siempre es posible crear ficheros secuenciales sobre disco o diskete (no solamente sobre casete o cinta magnética).

b) Hasta aquí no hemos tratado más que sobre un fichero a la vez. En la práctica se pueden crear varios ficheros diferentes sobre un mismo diskete. Cada uno de ellos vendrá representado por un nombre. Por tanto puede conservar sus programas Basic en alguna parte también a través del uso del mandato "SAVE" (o su equivalente). Los programas por tanto no son otra cosa que ficheros secuenciales en los cuales cada registro corresponde a una línea de programa.

1.6. CONCLUSION

Si dispone de un sistema de disketes o discos y desea informatizar alguno de sus ficheros manuales (o crearlos de nuevo), necesitará para cada uno de ellos, escoger y decidir entre un fichero de acceso secuencial y un fichero de acceso directo. Como hemos intentado describir anteriormente, cada uno tiene sus ventajas e inconvenientes. Además, no se manipulan o tratan de la misma forma. Ninguno de ellos es comparable además a un fichero manual.

En el curso de la lectura de este libro, irá incrementando progresivamente sus conocimientos sobre los dos tipos de ficheros y sobre el uso que puede hacer de ellos. A partir de ahora ya sabe que los ficheros secuenciales son más fáciles de programar que los ficheros de acceso directo. Por el contrario, si tiene ficheros de gran volumen, para los cuales el acceso a un registro cualquiera es frecuente, se hace prácticamente necesario el uso de un acceso directo.

2

Ficheros secuenciales: Primeros pasos

2.1. INTRODUCCION

Es posible que Vd. disponga de un repertorio o "directorio telefónico" manual. En este caso, quizás deseará conservar la información que contiene, en un fichero informático.

El primer paso será crear el fichero con la ayuda de un programa. Este programa podrá solicitar, por ejemplo, la entrada sucesiva a través de un teclado de cada uno de los nombres y de los números de teléfono correspondientes. A continuación, con la ayuda de otro programa podrá "consultar" el fichero creado para encontrar el número de teléfono de una persona. Como se trata de un fichero secuencial, este programa deberá examinar cada nombre (y cada número asociado) a partir del primero, hasta encontrar el nombre deseado.

Un tercer programa, si lo desea, le podría suministrar la relación completa de cada uno de los nombres junto con su número de teléfono.

En fin, probablemente deba modificar su fichero, bien sea debido a que un amigo cambie de teléfono, o para añadir nuevos nombres. Esta posibilidad de actualización o "puesta al día" es fundamental. Sin embargo, no la estudiaremos hasta más adelante. En este capítulo vamos a aprender esencialmente:

- A crear el fichero directorio,
- A buscar el número de teléfono de un nombre dado,
- A confeccionar la relación o listado completo del fichero.

2.2. CREACION DE UN FICHERO DIRECTORIO

2.2.1. Un programa simplificado

Veamos seguidamente un primer ejemplo de programa de creación. Se ha reducido en extremo, con el fin de destacar los elementos más importantes. Supongamos que nuestro directorio contiene 50 nombres.

```
100 '**** programa simplificado de creación de un
110       fichero repertorio telefónico
120   OPEN "O", # 1, "REPERT"
130   FOR I = 1 TO 50
140       INPUT NOM$, NUM
150       PRINT # 1, NUM, NOM$
160   NEXT I
170   CLOSE # 1
```

La instrucción 120 se denomina instrucción de apertura de fichero. En ella se encuentran tres elementos:

"O" 1 "REPERT"

El primero que no es otro que la constante O (letra O como Output, palabra inglesa que singnifica salida), indica al Basic que se trata de un fichero que se va a crear. Dicho de otro modo, no existe todavía.

El segundo elemento permite elegir un número (aquí el 1) que se utilizará a lo largo del programa para designar al fichero. Observe que este número no tendrá ningún significado una vez concluida la ejecución del programa. Antes bien, el fichero será reconocido por el Basic por el nombre que Vd. le haya escogido.

Finalmente, el tercer elemento corresponde al nombre que se desea dar a este fichero. Aquí, se llamará REPERT. Observe la presencia de las comillas para expresar que se trata de la cadena constante REPERT y no de una variable con ese nombre.

Las instrucciones o sentencias 130 y 160 permiten el ejecutar 50 veces las instrucciones 140 y 150. La instrucción 140 almacena en las variables NOM$ y NUM un nombre y un número de teléfono. La instrucción 150 ordena "escribir" el contenido de las variables NUM y NOM$ en el fichero número 1. De hecho, el término escribir no respon-

de a la realidad. Digamos por el momento, que el contenido de estas variables se almacenará en el fichero en cuestión.

Por último, la instrucción 170 se considera como de "cierre" del fichero e indica al Basic que se ha terminado de trabajar con el fichero número 1. Aquí, como se trata de la creación de un fichero, el Basic situará al final, lo que se denomina como "marca de fin de fichero" (en inglés EOF – End Of File). Esta marca será de gran utilidad posteriormente cuando trate de consultar el fichero; en efecto, permitirá al Basic reconocer el sitio correcto donde el fichero finaliza.

Esta instrucción CLOSE tiene igualmente un papel menos evidente.

Para comprenderla, es preciso saber que los intercambios de información entre Basic y el fichero se efectúan por "bloques". Más exactamente, el Basic comienza por rellenar una zona denominada "memoria tampón" (en inglés buffer), tan pronto y a medida que encuentra las instrucciones PRINT relativas al fichero.

Solamente cuando esta memoria tampón está "llena", se recopiará sobre el fichero. En estas condiciones, cuando se ha terminado de crear un fichero, es necesario que exista una transferencia a ese fichero del contenido de la memoria tampón (que probablemente no está completamente llena). Esta operativa la realizará el Basic al encontrar la instrucción CLOSE. (No confunda el concepto tampón con el de registro. Generalmente la memoria tampón contendrá varios registros).

2.2.2 Un programa más general

Vamos a adaptar nuestro programa de creación, de forma que permita:

– Escoger el nombre que se desea dar al fichero.

Anteriormente nuestro programa estaba pensado para crear sólo un fichero llamado REPERT. Ahora, el mismo programa podrá utilizarse para crear repertorios o directorios diferentes (por ejemplo: comerciantes, amigos, empresas...),

– Entrar un número cualquiera de nombres (el programa anterior solicitaba 50 sistemáticamente).

Añadiremos igualmente algunas instrucciones PRINT para ayudar al usuario del programa, especificándole cuál es la información esperada.

```
100 '**** CREACION DE UN REPERTORIO TELEFONICO
110 '
120 '---- entrada del nombre del fichero y apertura
130       PRINT "CREACION DE UN REPERTORIO TELEFONICO"
140       INPUT "ENTRE EL NOMBRE DEL FICHERO "; NF$
150       OPEN "O", #1, NF$
160 '
170 '---- escritura del modo de empleo
180       PRINT "entre cada nombre seguido de una coma, "
190       PRINT "del número correspondiente (6 cifras) "
200       PRINT "para terminar, entre un nombre vacío o un número nulo"
210 '
220 '---- creación del fichero
230       INPUT NOM$, NUM
240       IF NOM$ = "" OR NUM = 0 THEN GOTO 270
250       PRINT #1, NUM; NOM$
260       GOTO 230
270 '
280 ' ---- cierre del fichero y final
290       CLOSE #1
300       PRINT "**** CREACION DEL FICHERO "; NF$; " TERMINADA"
```

Como observará, la instrucción OPEN (en 150) contiene ahora, como tercer elemento, el nombre de una variable (NF$) del tipo cadena o tira de caracteres. Así se permite a nuestro programa trabajar sobre un fichero con cualquier nombre suministrado previamente por medio de la instrucción 140.

La creación del fichero propiamente dicha se realiza como ya hemos citado con la ayuda de la instrucción INPUT y PRINT. Por el contrario, ahora estas instrucciones se repetirán hasta que el usuario indique que no se van a introducir más nombres. Esto lo realiza suministrando un nombre constituido por una cadena "vacía" o un número igual a cero.

Observación importante

Quizás piense que sería suficiente pulsar la tecla "return" en respuesta a la instrucción 230 para indicar que Vd. ha terminado. De hecho, esto puede inducir a error, ya que el Basic espera dos valores (uno para NOM$ y otro para NUM), mientras que Vd.le suministra simplemente una cadena vacía para NOM$. La solución consiste en suministrar dos

valores separados por una coma, el primero de los cuales podría eventualmente ser una cadena vacía.

Veamos las respuestas posibles:

?, 15 (NOM$ contiene una cadena vacía y NUM el valor 15)
? X, 0 (NOM$ contiene el caracter X y NUM el valor 0)
?, 0 (NOM$ contiene una cadena vacía y NUM el valor 0)

He aquí un ejemplo de ejecución de este programa:

```
RUN
CREACION DE UN REPERTORIO TELEFONICO
ENTRE EL NOMBRE DEL FICHERO ? REPERT
entre cada nombre seguido de una coma,
del número correspondiente (6 cifras)
para terminar, entre un nombre vacío o un número nulo
? TOMAS, 482337
? COLON, 274553
? LEDESMA, 672830
? BALBOA, 413355
? TABLADA, 896313
? , 0
**** CREACION DEL FICHERO REPERT TERMINADA
```

2.3. UTILIZACION DEL FICHERO

El acceso a la información clasificada en el fichero se efectuará por medio de una instrucción INPUT parecida a la utilizada para entrar información desde teclado. La instrucción OPEN se deberá utilizar, ahora, para indicar al Basic el nombre del fichero en cuestión. Se deberá mencionar igualmente que se trata esta vez de un fichero existente que se desea consultar solamente.

2.3.1. Programa de listado simplificado

Consideremos el fichero de 50 nombres creado por el programa del apartado 2.2.1. Podremos obtener un listado sobre pantalla por medio del siguiente programa:

```
100  ****  programa simplificado de listado del fichero REPERT
110        OPEN "I", #1, "REPERT"
120        FOR I = 1 TO 50
130          INPUT #1, NUM, NOM$
140          PRINT NOM$; NUM
150        NEXT I
160        CLOSE #1
```

La instrucción 110 indica siempre al Basic que el fichero REPERT será identificado a lo largo del programa por el número 1. Sin embargo, habrá observado la presencia del caracter (constante) I en lugar de O. Se corresponde con la palabra inglesa Input e indica al Basic que el fichero REPERT se utilizará sólo en modo de lectura (entrada); dicho de otro modo, se dedicará a acceder a la información que está almacenada. Por tanto, este fichero debe existir en el momento en que esta instrucción OPEN se vaya a ejecutar.

Las instrucciones 120 y 150 se encargan de ejecutar 50 veces las instrucciones 130 y 140. La 130 permite acceder al fichero REPERT y atribuir un nombre y el número de teléfono correspondiente a las variables NOM$ y NUM.

Finalmente, la instrucción CLOSE indica que se ha dejado de trabajar con el fichero número 1.

Observación: Si se desea obtener un listado sobre impresora en lugar de en pantalla, es posible (en ciertos Basic) utilizar la instrucción LPRINT en lugar de PRINT.

2.3.2. Programa general de listado de un fichero

Adaptemos el programa anterior de forma que nos proporcione el listado de una guía o repertorio telefónico cualquiera, que contiene un número cualquiera de nombres (por tanto, registros).

```
100  '****  LISTADO DE UN FICHERO REPERTORIO TELEFONICO
110' '
120  '----  entrada del nombre del fichero y apertura
130        PRINT "LISTADO DE UN FICHERO REPERTORIO"
140        INPUT "entre el nombre del fichero "; NF$
150        OPEN "I", #1, NF$
160  '
170  '----  listado del fichero
```

```
180         INPUT # 1, NUM, NOM$
190         PRINT NOM$; NUM
200         IF EOF (1) <> –1 THEN GOTO 180
210 '
220 '———— cierre y final
230         CLOSE # 1
240         PRINT "**** FIN DEL LISTADO DEL FICHERO "; NF$
```

Vemos de nuevo la instrucción OPEN (en 150) en la cual figura un nombre de variable en cadena o tira de caracteres NF$. En ella se ha situado previamente el nombre del fichero en cuestión (por la instrucción 140).

Las instrucciones de la 180 a 210 efectúan el listado de la totalidad del fichero. En la 190 verá que aparece una función denominada EOF. Su argumento (aquí un 1) es un número de fichero. Esta función toma el valor –1 si se encuentra la marca de fin de fichero (EOF) en el fichero mencionado en el argumento. En caso contrario, toma el valor 0. Así, si nuestro fichero contiene por ejemplo 60 nombres, las instrucciones 180 a 200 se ejecutarán 60 veces. Una vez ejecutada la lectura número 60, se detectará la marca de fin de fichero. La evaluación (en la línea 200) de la función EOF conducirá al valor –1; el Basic proseguirá la ejecución del programa a partir de 220.

Utilicemos este programa para listar el fichero REPERT que se creó anteriormente. El diálogo del usuario con el Basic sería así:

```
RUN
LISTADO DE UN FICHERO REPERTORIO
entre el nombre del fichero? REPERT
TOMAS 482337
COLON 274553
LEDESMA 672830
BALBOA 413355
TABLADA 896313
**** FIN DEL LISTADO DEL FICHERO REPERT
OK
```

2.3.3. Búsqueda del número de una persona

Ahora se encuentra Vd. más entrenado para escribir un programa que busque el número de una persona dada. Es suficiente, para ello, recorrer

el fichero (modalidad INPUT) hasta encontrar el nombre buscado o la marca EOF. (En este último caso, se ha suministrado al programa un nombre que no existe en el fichero).

```
100 '**** BUSQUEDA DE UN NUMERO TELEFONICO
110 '
120 '----entrada del nombre del fichero –apertura– entrada del nombre buscado
130       PRINT "BUSQUEDA DEL NUMERO DE UN ABONADO"
140       INPUT "entre el nombre del fichero repertorio "; NF$
150       OPEN "I", #1, NF$
160       INPUT "entre el nombre buscado "; NR$
170 '
180 '---- bucle de búsqueda (hasta encontrar el nombre o EOF)
190          INPUT #1, NUM, NOM$
200          IF NOM$ = NR$ THEN GOTO 270
210          IF EOF (1) <> -1 THEN GOTO 190
220 '
230 '---- nombre ausente
240       PRINT "el nombre "; NR$; " no figura en el fichero"
250       GOTO 300
260 '
270 '---- nombre encontrado
280       PRINT NR$; " tiene por número telefónico"; NUM
290 '
300 ' ---- cierre y final
310       CLOSE #1
```

Utilicemos este programa para encontrar el número de algunos abonados.

```
RUN
BUSQUEDA DEL NUMERO DE UN ABONADO
entre el nombre del fichero repertorio ? REPERT
entre el nombre buscado ? LEDESMA
LEDESMA tiene por número telefónico 672830
OK
RUN
BUSQUEDA DEL NUMERO DE UN ABONADO
entre el nombre del fichero repertorio ? REPERT
entre el nombre buscado ? SANCHEZ
```

```
el nombre SANCHEZ no figura en el fichero
OK
BUSQUEDA DEL NUMERO DE UN ABONADO
entre el nombre del fichero repertorio ? REPERT
entre el nombre buscado ? TABLADA
TABLADA tiene por número telefónico 896313
OK
```

2.4. ALGUNOS ERRORES A EVITAR

2.4.1. No abrir un fichero que no ha sido cerrado

Ciertamente, será poco frecuente que Vd. escriba un programa que contenga dos instrucciones OPEN para un mismo fichero. Sin embargo, el riesgo de error existe durante la puesta a punto de un programa. Si a continuación de una interrupción producida por un error, y después de su eventual corrección, trata de hacer ejecutar de nuevo su programa desde el principio, correrá el riesgo de volver a encontrar la instrucción OPEN ya ejecutada. Esto se traducirá en un mensaje del tipo:

```
FILE ALREADY OPEN IN XXXX

(Fichero abierto previamente en XXXX)
```

indicando que el fichero ya se abrió en XXXX (donde XXXX es el número de instrucción en cuestión).

Si por el contrario, después de las correcciones se relanza el programa a proceso con el mandato RUN, no debe ocurrir ese tipo de problema. En efecto, este mandato efectúa el cierre automático de todos los ficheros previamente abiertos.

2.4.2. No crear dos ficheros diferentes con igual nombre

La instrucción:

```
OPEN "O",.....
```

crea un nuevo fichero. Si el nombre mencionado se corresponde con un fichero ya existente, el Basic no le facilitará ningún diagnóstico. Sin embargo, su antiguo fichero se perderá totalmente y será sustituido por el que nuevamente crearía este programa. Le aconsejamos pues tomar muchas precauciones con los programas de creación de ficheros y que procure la utilización de nombres suficientemente significativos que eviten posibles confusiones. Finalmente, sepa que todos los sistemas la facilitan mandatos que le permitirán:

- Conocer la relación de ficheros existentes
- Destruir un fichero existente.

3

Un fichero repertorio más completo

3.1. INTRODUCCION

En el capítulo anterior Vd. ha aprendido a crear y utilizar un fichero repertorio sencillo que contiene, para cada nombre, el número de teléfono correspondiente. Es muy posible que desee confeccionar un fichero más completo que contenga por ejemplo la dirección de cada persona y piensa, en tal caso, en crear su fichero en la forma anteriormente expuesta, solicitando simplemente la información del teclado y escribiéndola en el fichero con un PRINT. No obstante, si procede así, correrá el riesgo de encontrar serias dificultades para la utilización posterior del fichero.

Supongamos por ejemplo que le haya creado a partir de tres variables:

```
NUM    : número de teléfono
NOM$   : nombre de la persona
CIU$   : ciudad
```

con:

```
PRINT #1, NUM; NOM$; CIU$
```

y que la información correspondiente al primer registro fuera:

```
482337
CARLOS
GARCIA
```

Si en otro programa, Vd. comienza a leer el primer registro de este fichero con:

```
READ #1, NUM, NOM$, CIU$
```

y escribe sobre pantalla el contenido de las dos primeras variables con:

```
PRINT NUM, NOM$
```

obtendrá:

```
482337  CARLOSGARCIA
```

(para mayor sencillez, omitiremos aquí lo que contendría la variable CIU$).

Como verá, las dos cadenas de caracteres que contienen NOM$ y CIU$ se agrupan en una durante la lectura del fichero.

Es posible que Vd. haya observado que ello es debido a que no hay nada en el fichero que permita separarlos. Vamos a estudiar este problema de los "separadores", después de lo cual, estaremos en condiciones de crear nuestro fichero repertorio como deseemos.

3.2. GRABACION DE INFORMACION EN UN FICHERO SECUENCIAL

3.2.1. ¿Qué ocurre en el caso de una pantalla?

Antes de ver qué ocurre en el caso de un fichero, vamos a hacer algunas consideraciones sobre la forma en que la información se escribe sobre pantalla (para impresión serían también válidas). Consideremos estas instrucciones:

```
100  NUM = 20
110  NOM$ = "ERNESTO"
120  PRINT NUM;NOM$
```

Su ejecución producirá:

```
@ 20@ERNESTO
```

(el signo @ indica un espacio. Recuerde que los números se escriben con un lugar reservado para el signo y un espacio a continuación).

Aquí, se puede afirmar que el Basic ha escrito 11 caracteres sobre la pantalla (4 para la variable NUM y 7 para NOM$).

De hecho, el Basic no transmite los 11 caracteres en la forma que Vd. los ve representados. La realidad es que envía simplemente el "código" de cada carácter. Quizás sepa que el código utilizado es el ASCII, en el cual cada carácter se codifica sobre un "octeto", es decir, sobre ocho bits (posiciones binarias).

3.2.2. ¡Y sobre disco!

Cuando Vd. utiliza una instrucción de escritura o grabación como:

```
PRINT # 1, NUM; NOM$
```

se efectúa igualmente la transmisión de los 11 octetos que contienen los códigos de los 11 caracteres. Pero ahora no se escriben en forma legible como en el caso de pantalla. Son copiados tal cual en el disco.

En resumen, se puede considerar que un fichero secuencial se compone de una sucesión "de octetos" representando cada uno un carácter. Diremos simplemente que es una sucesión de caracteres. Así pues, en nuestro ejemplo anterior, diremos que hemos escrito 11 caracteres en nuestro fichero:

@	2	0	@	E	R	N	E	S	T	O

(cada una de las casillas representa un "octeto")

3.2.3. Cuando se dispone de "tabulación automática"

En un orden de PRINT corriente, se pueden separar los nombres de las variables sea por un "punto y coma" (como anteriormente), sea por "comas". En este último caso, existe "tabulación", es decir, que la escritura tiene lugar a partir de una cierta ubicación, en lugar de efectuarse a continuación de la precedente. Así:

```
100  C1$ = "BAR"
110  C2$ = "CAZA"
```

```
120 PRINT C1$; C2$
130 PRINT C1$, C2$
```

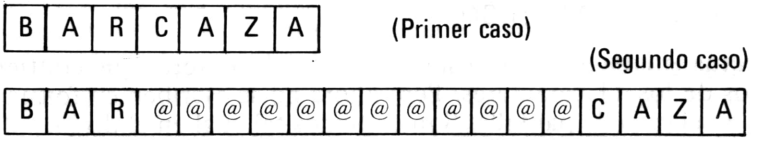
nos dará:

```
BARCAZA

BAR @@@@@@@@@@@@ CAZA
```

En el segundo caso, la cadena "CAZA" se ha escrito a partir de la columna 16. El Basic transmite 12 caracteres a espacios entre los caracteres BAR y los caracteres CAZA.

Sobre disco se obtendrían resultados semejantes. Utilicemos nuestra representación en casillas:



3.2.4. Cuando hay más de una línea

Ejecutemos este pequeño programa:

```
100 NUM = 20
110 NOM$ = "ERNESTO"
120 PRINT NUM
130 PRINT NOM$
```

Obtenemos:

```
RUN
20
ERNESTO
```

¿Qué se obtendrá si se escribe de forma semejante sobre un fichero secuencial? En realidad, hay que saber que al finalizar el primer PRINT, el Basic ha enviado sobre la pantalla dos caracteres especiales:

– Un primer carácter que significa volver al comienzo de línea que se denomina CR (como Carriage Return en inglés o sea "retorno de carro").

– Un segundo que significa pasar a la línea siguiente y se denomina LF (como Line Feed en inglés o sea "salto de línea"). En el caso de escritura en un fichero secuencial, el Basic transmite exactamente los mismos caracteres que habría enviado a pantalla. Se almacenan todos en el fichero incluyendo los caracteres especiales CR y LF.

En definitiva, estas instrucciones:

```
100 PRINT #1, NUM
110 PRINT #1, NOM$
```

escribirán en el fichero:

@	2	0	@	CR	LF	E	R	N	E	S	T	O	CR	LF

3.3. LECTURA DE INFORMACION DE UN FICHERO SECUENCIAL: LOS SEPARADORES

Cuando Vd. entra información desde el teclado, en respuesta a una instrucción INPUT, ya sabe que en ciertos casos el carácter "espacio" o la coma pueden hacer de separadores entre varias informaciones (destinadas cada una a una variable diferente). En otros casos, por el contrario, se pueden considerar como caracteres "ordinarios". Nos encontraremos con un mecanismo casi idéntico en la lectura de la información de un fichero secuencial.

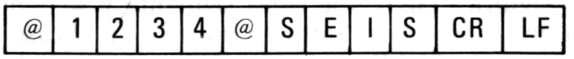
3.3.1. El blanco es a veces carácter separador

Supongamos que se ha creado un (pequeño) fichero secuencial con la ayuda de este programa:

```
100 OPEN "O", #1, "PRUEBA"
110 NUM = 1234
120 C$ = "SEIS"
```

```
130 PRINT #1, NUM; C$
140 CLOSE #1
```

El fichero denominado PRUEBA contendrá los siguientes caracteres:



Ejecutemos ahora este pequeño programa de lectura del fichero PRUEBA:

```
100 OPEN "I", #1, "PRUEBA"
110 INPUT #1, NUM, C$
120 PRINT "NUM = "; NUM
130 PRINT "C$ = "; C$
140 CLOSE #1
```

Obtendremos:

```
RUN
NUM = @ 1234
C$ = SEIS
```

Dicho de otro modo, el caracter espacio (@) situado entre el 4 y la S se ha considerado como separador. Permite al Basic detectar el final de los caracteres proporcionados para la variable numérica NUM. En cuanto a la cadena de caracteres a atribuir a N$, el Basic considera que termina, cuando se encuentra el carácter CR. (Igual ocurriría si hubiera leído la misma información del teclado —el hecho de pulsar la tecla "return" que envía los caracteres CR y LF al Basic).

Por el contrario, ejecutemos ahora otro programa de lectura sobre nuestro fichero PRUEBA:

```
100 OPEN "I", #1, "PRUEBA"
110 INPUT #1, C$
120 PRINT "C$ = "; C$
130 CLOSE #1
```

Ahora obtendremos (quizás se sorprenda):

```
RUN

C$ = 1234 @ SEIS
```

Esta vez, la totalidad de caracteres ha sido atribuida a C$ (con la excepción sin embargo del primer carácter en blanco). Incluso obtendríamos exactamente los mismos resultados leyendo la misma información desde el teclado con:

```
INPUT C$
```

En resumen, el espacio (@) se considera como separador cuando aparece después de un valor numérico, pero no cuando está situado en una cadena (entonces se considera como otro carácter cualquiera). Ya sabe Vd. que si tiene necesidad de separar varias cadenas entre sí (o una cadena de un valor numérico), es preciso utilizar otras soluciones.

3.3.2. Primer modo de separar cadenas: utilizar la coma

Se ejecuta el programa:

```
100 INPUT C1$, C2$
110 PRINT "C1$ = "; C1$
120 PRINT "C2$ = "; C2$
```

tecleando dos cadenas separadas por una coma, obtendrá:

```
RUN
?RESPUESTA NO 1, RESPUESTA NO 2
C1$ = RESPUESTA NO 1
C2$ = RESPUESTA NO 2
```

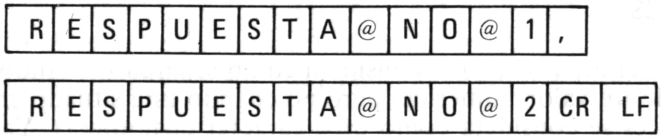
Pero para utilizar esta posibilidad en la lectura de un fichero secuencial, es preciso además que la coma (o comas) exista/n en la información almacenada en el fichero. Así pues, le va a ser necesaria la escritura de este carácter "coma" en las instrucciones PRINT relativas a su fichero.

Si por ejemplo, desea crear un fichero llamado PRUEBA2, que contenga dos cadenas cualesquiera leídas desde el teclado, podría utilizar este programa:

```
100 OPEN "O", #1, "PRUEBA 2"
110 INPUT C1$, C2$
120 PRINT #1, C1$; ", "; C2$
130 CLOSE #1
```

```
RUN
?RESPUESTA NO 1, RESPUESTA NO 2
```

El fichero PRUEBA2 contendrá ahora los siguientes caracteres:



Observe que la coma que figura en el fichero proviene de la instrucción 120 y que no tiene nada que ver con la coma que ha servido al usuario para separar sus dos respuestas.

La lectura del fichero se podrá naturalmente hacer así:

```
100  OPEN "I", #1, "PRUEBA2"
110  INPUT #1, C1$, C2$
120  PRINT "C1$ = "; C1$
130  PRINT "C2$ = "; C2$
RUN
C1$ = RESPUESTA NO 1
C2$ = RESPUESTA NO 2
```

3.3.3. Segunda solución para separar cadenas: las comillas

Las comillas son indispensables para manipular una cadena o tira de caracteres que contenga el carácter "coma". Supongamos, por ejemplo, que Vd. tiene que escribir dos cadenas, la primera conteniendo un nombre de persona y la segunda una dirección (que puede llevar una o varias comas).

He aquí un programa que crea un fichero que contiene esta información para 10 personas.

```
100  OPEN "O", #1, "DIRECCION"
110  FOR I = 1 TO 10
120       INPUT "NOM"; NOM$
130       INPUT "DIRECCION"; DIR$
140       PRINT #1, NOM$; ", "; CHR$ (34); DIR$, CHR$ (34)
150  NEXT I
160  CLOSE #1
```

```
RUN
NOM? CASTAÑEDA
DIRECCION? "2, AVENIDA NUEVA"
.....etc.....
```

Haremos varias observaciones necesarias:

1. En su respuesta en el teclado para DIR$, Vd. ha tenido que utilizar las comillas para evitar que la coma posterior al 2 sea interpretada como separador.

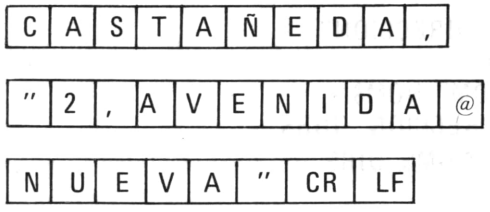
2. Después de la ejecución de la instrucción 130, la variable NOM$ contiene la cadena:

   2, AVENIDA NUEVA

   sin comillas.

3. En la 140, debe haber escrito sobre su fichero el contenido de DIR$ enmarcado entre comillas ("). Como quiera que este carácter es el mismo que se utiliza para delimitar las constantes cadenas, no puede aparecer en el interior de una de ellas. En tal caso, se hace necesario el utilizar la función CHR que, recordemos que produce como resultado, sin tener en cuenta el carácter que se ha suministrado, el código (decimal) en el argumento. Aquí, 34 es el código correspondiente al carácter comillas.

4. La primera ejecución de la instrucción 140, escribirá en el fichero DIRECCION:



Entonces será sencillo leer este fichero, por medio de instrucciones como:

```
INPUT #1, NOM$, DIR$
```

Observación: Quizás esta utilización de las comillas le resulte tediosa. De hecho, Vd. sólo las necesitará cuando sus cadenas contengan comas. Por otra parte, ciertos Basic facilitan una instrucción WRITE que pone automáticamente en su sitio correcto los separadores, durante la escritura de un fichero.

3.3.4. La instrucción "WRITE"

Desde que existe esta instrucción (este es el caso del Basic Microsoft, a partir de la versión 5.0), está disponible y es utilizable al igual que la instrucción PRINT tanto para pantalla como para un fichero secuencial.

Veamos un ejemplo sobre pantalla con este programa:

```
100  X = 123
110  C$ = "EJEMPLO"
120  WRITE X, C$
RUN
123, "EJEMPLO"
```

Observe que la coma aparece como separador entre los dos valores visualizados. Por otro lado, la cadena se encuadra entre comillas. Sin embargo, no se ha dejado ningún espacio delante o detrás del valor numérico 123 (contrariamente a lo que ocurriría con la instrucción PRINT).

Por tanto, nuestro ejemplo de creación del fichero DIRECCION (del apartado 3.3.3.), se podrá escribir más simplemente:

```
100  OPEN "O", #1, "DIRECCION"
110  FOR I = 1 TO 10
120       INPUT "NOM"; NOM$
130       INPUT "DIRECCION"; DIR$
140       WRITE #1, NOM$, DIR$
150  NEXT I
150  CLOSE #1
```

En definitiva, si dispone de la instrucción WRITE, le permitirá sortear el problema de los separadores en los ficheros secuenciales. Las instrucciones WRITE parecerán ahora más naturales y útiles que las ins-

trucciones PRINT. No obstante, como muchos Basic no disponen (aún) de esta facilidad, continuaremos durante este manual, utilizando la instrucción PRINT.

3.4. APLICACION PARA NUESTRO FICHERO REPERTORIO

3.4.1. Introducción

Al comienzo de este capítulo nos habíamos propuesto aprender a construir un fichero repertorio que contuviera para cada persona, su nombre, su número de teléfono y su dirección. Ahora ya posee los conocimientos necesarios para conseguirlo.

Sin embargo, previamente, debemos hacer un comentario sobre el número de teléfono. En el capítulo anterior se había considerado como numérico ya que no habíamos utilizado más que números de 6 cifras. Si por descuido hubiéramos entrado un número de 7 cifras, es muy posible que (a través del posterior listado) le hubiéramos encontrado algo erróneo. Ello es debido a que el Basic trabaja generalmente con seis cifras significativas. Así, el número 9417532 podría aparecer bajo la forma de 9.41753E6. Este problema puede ser obviado fácilmente considerando a ese número telefónico como una cadena de caracteres (ello es posible desde el momento en que no se va a efectuar cálculo sobre esta información). Por otra parte, vamos a tener en cuenta en lo sucesivo el indicativo o prefijo telefónico. Podríamos haberle incorporado en la cadena precedente, pero el hecho de constituir una información separada nos permitirá posteriormente efectuar búsquedas a través de este indicativo.

En cuanto a la dirección, sería conveniente descomponerla en tres partes:

- dirección propiamente dicha, de tipo cadena,
- código postal, de tipo numérico
- ciudad, de tipo cadena.

3.4.2. Creación del fichero repertorio

Veamos un programa simplificado de creación en la página siguiente.

```
100 '**** CREACION DE UN REPERTORIO TELEFONICO CON DIRECCIONES
110 '
120       PRINT "CREACION DEL FICHERO REPERTA"
130       OPEN "O",#1,"REPERTA"
140 '
150 '---- bucle hasta que nombre = cadena vacía
160          INPUT "nombre "; NOM$
170          IF NOM$ = " " THEN GOTO 260
180          INPUT "indicativo telefónico "; INDIC
190          INPUT "número de teléfono  "; NUMT$
200          INPUT "dirección  "; ADR$
210          INPUT "código postal  "; CODE
220          INPUT "ciudad  "; CIUDAD$
230          PRINT #1, NOM$; ", "; INDIC; NUMT$; ", "; CAR$ (34); DIR$;
240          PRINT #1, CAR$ (34); ", "; CODE: CIUDAD$
250       GOTO 160
260 '---- cierre
270       CLOSE #1
```

El elemento esencial de este programa está constituido por las instrucciones 230 y 240 que permiten:

- Situar las comas para separar una cadena de otra o de un valor numérico que le siga (por ejemplo: NUMT$ y DIR$ o NOM$ e INDIC)

- Situar entre comillas, la cadena contenida en DIR$, ya que puede que contenga una coma (como por ejemplo: 2, AV. BRUSELAS).

A título de ejemplo si se ejecuta este programa así:

```
RUN
CREACION DEL FICHERO REPERTA
nombre ? TOMAS
indicativo telefónico ? 86
número de teléfono ? 482337
dirección ? c/ granada, 2
código postal 58000
ciudad ? NERJA
nombre ? COLON
indicativo telefónico ? 63
```

```
número de teléfono ? 274553
dirección ? c/ baleares, 56
código postal ? 81000
ciudad ?AVILA
nombre ? LEDESMA
indicativo telefónico ? 75
número de teléfono ? 672830
dirección ? av. betanzos, 17
código postal ? 77780
ciudad ? CHINCHON
nombre ?
OK
```

el fichero REPERTA contendrá la información siguiente:

```
TOMAS, 86 482337, "c/ granada, 2", 58000 NERJA
COLON, 63 274553, "c/ baleares, 56", 81000 AVILA
LEDESMA, 75 672830, "av. betanzos, 17", 77780 CHINCHON
```

(los caracteres CR y LF no se representan aquí; se supone que están presentes al final de cada línea).

3.4.3. Listado del fichero repertorio

No reviste ningún problema en particular. Se abre el fichero al principio en modo lectura (instrucción 120). Se repiten a continuación un conjunto de instrucciones (160 a 210) que se encargan de leer la información del fichero y presentarla en forma legible en pantalla. Al detectar la marca de fin de fichero (EOF) al efectuar la última lectura, se interrumpe este proceso.

```
100 '**** LISTADO DEL FICHERO REPERTA
110 '
120       OPEN "I", #1, "REPERTA"
140       PRINT "LISTADO DEL FICHERO REPERTA"
150 '
160 '---- bucle hasta EOF
170          INPUT #1, NOM$, indic, NUMT$, DIR$, CODE, CIUDAD$
180          PRINT NOM$; TAB(20); " ("; INDIC; ") "; NUMT$
```

```
190          PRINT      TAB(20);DIR$;TAB (60);CODE;
                                          TAB (67);CIUDAD$
200          PRINT
210       IF EOF (1) <> -1 THEN GOTO 170
220  '---- cierre
230       CLOSE # 1
```

Observe que en la línea 180 se han insertado paréntesis para presentar el número de teléfono en pantalla en forma clásica como: (38) 584513. La instrucción 200 escribe una línea en blanco para espaciar así el listado obtenido.

Ejecutado sobre el pequeño fichero creado anteriormente, este programa proporciona los siguientes resultados:

```
RUN
LISTADO DEL FICHERO REPERTA

TOMAS       (86) 482327          58
            c) granada, 2        58000 NERJA

COLON       (63) 274553
            c/ baleares, 56      81000 AVILA

LEDESMA     (75) 672830
            av. betanzos, 17     77780 CHINCHON
```

3.5. PARA SIMPLIFICAR EL TRABAJO: LINE INPUT

Durante la creación del fichero repertorio utilizamos varias instrucciones INPUT para entrar la información correspondiente. En realidad es posible entrar el conjunto de información relativa a una persona con la ayuda de una sola instrucción : LINE INPUT. Así por ejemplo:

```
LINE INPUT CH$
```

permite entrar, en la variable cadena CH$, todos los caracteres tecleados (incluidas las comas y comillas) hasta que se pulse la tecla "return".

Así la creación de nuestro fichero REPERTA podría escribirse:

```
100 OPEN "O",#1,"REPERTA"
110    LINE INPUT CA$
120    IF CA$ = "" THEN GOTO 150
130    PRINT #1, CA$
140 GOTO 110
150 CLOSE #1
```

Veamos lo que podría ser la ejecución de este programa (con la misma información que anteriormente):

```
RUN
TOMAS, 86 482327, "c/ granada, 2", 58000 NERJA
COLON, 63 274553, c/ baleares, 56", 81000 AVILA
LEDESMA, 75 672830, "av. betanzos, 17", 77780 CHINCHON
OK
```

Como verá, la ventaja de LINE INPUT estriba en el hecho de que es suficiente entrar la información exactamente como desea que aparezca en el fichero. Sin embargo, si el programa de creación se encuentra simplificado, el trabajo del usuario se hace más costoso ya que en ese caso, él mismo debe prever sus separadores con exactitud. Además, una eventual omisión de un separador no le será señalada por el programa de creación, ya que por definición, LINE INPUT acepta "todo". Antes bien, los problemas aparecerán cuando se utilice posteriormente el fichero así creado.

Como norma general, no aconsejaríamos la utilización de LINE INPUT para entrar sus informaciones desde el teclado, por el contrario, según verá en el próximo capítulo, esta instrucción podrá utilizarse igualmente para leer la información de un fichero secuencial. En este caso, su interés es mucho mayor.

4

Concepto de registro en un fichero secuencial

4.1. INTRODUCCION

4.1.1. Comentarios sobre la instrucción "PRINT"

La ejecución de estas instrucciones:

```
100  A = 123
110  B = -48
120  C = 57
130  PRINT A, B
140  PRINT C
```

escribe sobre la pantalla:

```
123 @ -48
57
```

Obtenemos así dos líneas sucesivas. Por el contrario, si se sustituyen las instrucciones 130 y 140 por:

```
130  PRINT A; B; C
```

o

```
130  PRINT A; B;
140  PRINT C
```

no obtenemos más que una sola línea:

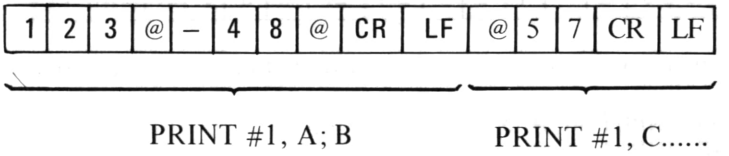
```
123  @ 48  @ @ 57
```

4.1.2. En el caso de un fichero secuencial

De igual forma, si 1 es el número asignado a un fichero secuencial (supuestamente abierto), las instrucciones:

```
PRINT #1 A; B
PRINT #1, C
```

escribirán en el fichero los siguientes caracteres:



Los caracteres CR y LF aparecen dos veces: diremos pues que hemos escrito dos "registros" en el fichero. De igual forma, se habían escrito dos líneas sobre la pantalla.

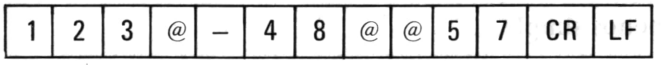
Por el contrario:

```
PRINT #1 A; B
```

o también:

```
PRINT #1, A; B;
PRINT #1, C
```

habrían escrito:



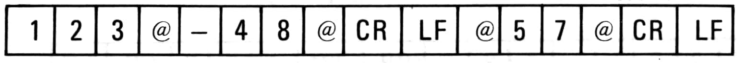
Los caracteres CR y LF no aparecen más que una vez: diremos pues que hemos escrito un solo registro en el fichero. De igual forma, habíamos escrito una sola línea sobre la pantalla.

4.2. LECTURA DE REGISTROS DE UN FICHERO SECUENCIAL

Hemos visto que la escritura de registros es en todo momento comparable a la escritura de líneas sobre pantalla. ¿Qué sucede con la lectura? Podría pensarse que se desarrolla de manera semejante a la lectura de información desde el teclado. Esto no es totalmente cierto.

4.2.1. Un caso simple

Supongamos que los primeros registros de un fichero son los siguientes:



y que las primeras instrucciones de lectura sobre este fichero son:

```
100  INPUT #1, A, B
110  INPUT #1, C
```

La instrucción 100 leerá el primer registro atribuyendo los valores 123 y –48 a las variables A y B. La instrucción 110 leerá el registro siguiente, atribuyendo el valor 57 a C.

Todo sucede como si a estas dos instrucciones de lectura de teclado:

```
100  INPUT A, B
110  INPUT C
```

hubiéramos respondido:

```
?123 @–48     (ó 123, – 48)
?57
```

(la tecla "return" corresponde como hemos visto a CR y LF)

4.2.2. Y uno menos simple....

Por el contrario, si leemos nuestros dos mismos registros con:

```
100  INPUT #1, A
110  INPUT #1, B, C
```

vemos que a A, B y C se les va a atribuir de nuevo los valores 123 –48 y 57.

Como puede ver, la instrucción 100 ha leído la primera información del primer registro. En cuanto a la 110, se ha hecho cargo de la segunda información de este primer registro. Dicho de otro modo, aquí los caracteres CR y LF se han considerado simplemente como un separador entre dos valores, como si fuera un espacio o una coma. (De hecho, en rigor, es el carácter LF el que sirve de separador).

Es preciso constatar que esto no sucede de igual modo para una lectura desde el teclado. En este caso, en efecto, a cada instrucción INPUT debe corresponder una línea de respuesta que contenga toda la información requerida. Por ejemplo, si a:

```
100 INPUT A
110 INPUT B, C
```

se responde:

```
?123, –48
```

se obtendrá un mensaje de error del tipo:

```
Redo from start
```

indicándole que no está facilitando el número requerido de valores (aquí ha dado demasiados)

4.2.3. Resumen

Nos hemos servido de una cierta analogía existente entre las instrucciones relativas al teclado y a la pantalla por una parte y a los ficheros secuenciales por otra. Esto nos ha permitido introducir de manera relativamente concreta, los conceptos de fichero y de registro. No obstante, como ha podido constatar, esta analogía tiene sus límites. En particular, la lectura en un fichero secuencial tiene lugar respetando las reglas siguientes:

– Los caracteres LF son siempre ignorados,
– El carácter CR es considerado como un separador.

Observación: Si estas consideraciones le parecen un poco complejas, le diremos que es posible utilizar simplemente los ficheros secuenciales.

Para ello, es suficiente como habíamos hecho en nuestros primeros ejemplos, leer sus ficheros de manera semejante a la que los creó (dicho en otras palabras, hacer que una instrucción INPUT lea todo un registro y sólo uno).

Por otra parte, ésta es la manera más natural de trabajar, sin embargo, es conveniente que esté prevenido de los riesgos que corre cuando, por ejemplo, por error Vd. proceda de forma diferente. En efecto, en este caso, puede obtener valores diferentes a los esperados sin que le sea facilitado ningún mensaje o diagnóstico de error. Esta situación tiene el riesgo de producirse si Vd. crea ficheros secuenciales en que los registros no están todos constituidos por el mismo número de valores.

4.3. PARA ACCEDER A UN REGISTRO DE UN FICHERO: LINE INPUT

4.3.1. Introducción

Suponga que acaba de crear un fichero secuencial y que utilizando un programa descubre que los resultados obtenidos no son los esperados. Entonces es que existe al menos un error en uno de los dos programas (creación o utilización) ¿Cómo disipar la duda? Con toda seguridad podrá repasar atentamente cada uno de sus programas. Sería sin embargo mucho más agradable el disponer de los medios precisos para verificar que su fichero se creó como Vd. lo deseaba.

Ahora bien, la instrucción LINE INPUT, aplicada a los ficheros secuenciales le ofrece esta posibilidad. En efecto, le permite leer su fichero, registro a registro, sin tener en cuenta otros separadores que no sean los CR.

4.3.2. Un programa muy útil

He aquí un programa sencillo que realiza el listado del conjunto de registros de un fichero secuencial cualquiera:

```
100 '* LISTADO POR REGISTRO DE UN FICHERO SECUENCIAL CUALQUIERA
110 '
120 '----  lectura del nombre del fichero y apertura
           PRINT "LISTADO POR REGISTRO DE UN FICHERO SECUENCIAL"
```

```
140           INPUT " entre el nombre del fichero "; NOMF$
150           OPEN "I", # 1, NOMF$
160 '
170 '---- listado del fichero
180           LINE INPUT # 1, ENREG$
190           PRINT ENREG$
200       IF EOF (1) <> -1 THEN GOTO 180
210 '
220 '---- cierre y final
230       PRINT "*** FIN DE FICHERO"
240       CLOSE # 1
```

No vamos a detallar el funcionamiento de este programa, muy semejante a los anteriormente vistos. Solamente es nueva la instrucción 180. Su cometido es el de leer en el fichero abierto bajo el número 1, un registro (cadena de caracteres) en la variable cadena llamada aquí ENREG$. Esta es escrita inmediatamente en la pantalla por la instrucción 190.

Así cada registro aparece sobre una líneą de pantalla. El control del contenido del fichero es así facilitado.

4.3.3. Ejemplo

Utilicemos este programa para listar los registros de nuestro fichero REPERTA creado en el capítulo anterior.

```
RUN
LISTADO POR REGISTRO DE UN FICHERO SECUENCIAL
entre el nombre del fichero ? REPERTA
TOMAS, 86 482337, "c/ granada, 2", 58000 NERJA
COLON, 63 274553, "c/ baleares, 56", 81000 AVILA
LEDESMA, 75 672830, "av. betanzos","17", 77780 CHINCHON
OK
```

5

Un solo programa, varias funciones. Concepto de menú

5.1. INTRODUCCION

En los capítulos precedentes hemos aprendido a escribir varios programas que realizaban varias operaciones sobre un mismo fichero (creación, listado completo, listado por registros, etc...). Puede ser conveniente el reagrupar estas diferentes posibilidades en un solo programa. La elección de una de las funciones la realizará entonces el usuario, a partir de un "menú" ofrecido por el programa. Varias operaciones pueden ser eventualmente ejecutadas sucesivamente.

5.2. CONCEPTO DE MENU

Un programa con "menú" se compone pues de una primera parte que tiene por objeto el solicitar del usuario una selección entre varias operaciones. En función de su respuesta, se ejecutará un conjunto dado de instrucciones (o lo que es más práctico, un sub-programa o rutina). A continuación se ejecuta de nuevo la primera parte para solicitar del usuario una nueva selección. Naturalmente se debe prever en el menú la posibilidad de parada. Cada rutina puede a su vez contener un sub-menú, etc...

5.3. PREPAREMOS NUESTRO MENU

Vamos a confeccionar un programa que efectúe diversas operaciones sobre un fichero repertorio, donde cada registro contiene:

- Un nombre (cadena),
- Un indicativo telefónico (numérico),
- Un número telefónico de 6 ó 7 cifras (cadena),
- Un código postal (numérico),
- Un nombre de departamento (cadena).

Se desea escoger entre las siguientes funciones:

- Creación del fichero.
- Búsqueda de la información relativa a una persona dada, a partir de su nombre.
- Listado selectivo, es decir, el listado de los registros correspondientes, por ejemplo, a un código postal dado, a un departamento dado, etc...

La primera parte de nuestro programa podría ser así:

```
1000 '**** GESTION DE UN FICHERO REPERTORIO SECUENCIAL
1010 '      cada registro contiene:
1020 '      - nombre (cadena)
1030 '      - indicativo telefónico (numérico)
1040 '      - número telefónico (cadena)
1050 '      - dirección (cadena)
1060 '      - código postal (numérico)
1070 '      - ciudad (cadena)
1080 '      las funciones posibles son:
1090 '      - creación
1100 '      - listado completo
1110 '      - búsqueda de una persona
1120 '      - listado selectivo
1130 '
1140 '----   entrada del nombre del fichero
1150        PRINT "GESTION DE UN FICHERO REPERTORIO"
1160        PRINT
1170        INPUT "entre el nombre del fichero"; NOMF$
```

```
1180  '
1190  '----- repetición de la selección de una función y
1200  '       ejecución hasta petición de parada
1210        PRINT
1220        PRINT "seleccione una de las siguientes funciones"
1230  PRINT "    CR creación"
1240  PRINT "    LI listado de todo el fichero"
1250  PRINT "    RE búsqueda de una persona"
1260  PRINT "    LS listado selectivo"
1270  PRINT "    FI fin de trabajo"
1280  INPUT D$
1290  IF D$ = "CR"  THEN GOSUB 2010 : GOTO 1210
1300  IF D$ = "LI"  THEN GOSUB 3010 : GOTO 1210
1310  IF D$ = "RE"  THEN GOSUB 4000 : GOTO 1210
1320  IF D$ = "LS"  THEN GOSUB 5010 : GOTO 1210
1330  IF D$ = "FI"  THEN END
1340  PRINT " *** mala función, entre de nuevo" : GOTO 1280
```

Después de solicitar el nombre del fichero en 1170, las instrucciones 1220 a 1270 presentarán un menú al usuario, indicándole para cada función:

- Un código de dos letras que constituye la respuesta a suministrar si escogió esa función.
- Un texto especificando el cometido o función.

La respuesta se lee en 1280. Las instrucciones 1290 a 1330 permiten la ejecución de una rutina o sub-programa particular dependiendo de la respuesta aportada. Aquí, es preciso indicar que hemos empleado una forma práctica de instrucciones múltiples (muchos Basic permiten varias instrucciones por línea). Así:

```
1290 IF D$ = "CR" THEN GOSUB 2010 : GO TO 1210
```

significa: si D$ contiene la cadena CR, ejecutar la rutina situada en 2010 y después ir a la instrucción 1210. En caso contrario ejecutar la línea siguiente (aquí será la 1300). En fin, en el caso de que el usuario no haya escogido ninguna de las respuestas propuestas, se le indica en la línea 1340, solicitándole una nueva selección.

Observaciones

Si su Basic no dispone de instrucciones múltiples, deberá tomar algunas precauciones en lo que respecta a las instrucciones 1290 a 1320. En efecto, 1290 no debe ser sustituida apresuradamente por:

```
1290 IF D$ = "CR" THEN GOSUB 2010
1295 GO TO 1210
```

Por el contrario puede utilizar:

```
1290 IF D$ <> "CR" THEN GO TO 1300
1292    GOSUB 2010
1295    GO TO 1210
1300 .....
```

Hemos escogido un modelo de dos letras con un carácter mnemotécnico. Entre otras soluciones, son posibles:

- Una respuesta más clara (más larga)
- Un código numérico: no tiene más que el carácter mnemotécnico de las letras. Es por tanto poco práctico para el usuario. Por el contrario, este caso es más fácil de programar la selección de la rutina en cuestión. Si por ejemplo, se escogen los códigos 1, 2, 3, 4 y 5 en lugar de CR, LI, RE, LS y FI, las instrucciones 1280 a 1330 se pueden sustituir por:

```
1290 INPUT R
1300 IF R > 1 AND R < 4 THEN ON R GOSUB 2010, 3010, 4010, 5010:
     GOTO 1280
1310 IF R = 1 THEN END
```

Vamos ahora a ver cómo realizar los cuatro sub-programas o rutinas en cuestión.

5.4. LA RUTINA DE CREACION DEL FICHERO

Debemos prever dos formas de crear nuestro fichero:

- Una primera denominada "conversacional" donde se indica al usuario qué información debe aportar (nombre, indicativo...).

– Una segunda denominada "por registro" en la que el usuario proporciona íntegramente cada registro, con los separadores necesarios.

Incluso, la selección entre estas dos posibilidades se efectúa a partir de un sub-menú presentado al usuario.

Se debe tener en cuenta que, generalmente, la apertura en modo escritura ("O") del fichero, si existe ya, le destruye, por tanto, la operación presenta un cierto riesgo. Es por esto por lo que se comienza (de 2050 a 2090) por hacer confirmar al usuario que él desea crear efectivamente el fichero en cuestión. En caso de no confirmación, se limita a salir de la rutina.

En la 2130 se abre el fichero. Después, se hace elegir (2150 a 2210) al usuario una de las dos opciones CV (creación conversacional) o ER (creación por registro). Según el caso, se bifurcará a una de las dos secuencias de instrucción (comenzando en 2230 ó 2350), en función de la opción elegida.

```
2000  '
2010  '****************************************
2020  '*** RUTINA DE CREACION DEL FICHERO ***
2030  '****************************************
2040  '
2050  PRINT "ATENCION, si el fichero  "; NOMF$; "  existe"
2060  PRINT "                 será destruido."
2070  PRINT "¿quiere crearlo?"; NOMF$;" (SI o NO)"
2080  INPUT R$
2090  IF R$ <> "SI" THEN RETURN
2100  '
2110  '---- apertura del fichero y selección de opción
2120  '           (conversacional o por registro)
2130  OPEN "O", # 1, NOMF$
2140  '
2150  PRINT "seleccione el modo de creación deseado :"
2160  PRINT "   CV conversacional"
2170  PRINT "   ER por registro"
2180  INPUT R$
2190  IF R$ = "CV" THEN GOTO 2230
2200  IF R$ = "ER" THEN GOTO 2350
2210  PRINT "*** mala selección –entre de nuevo" : GOTO 2180
```

```
2220 '
2230 '---- creación conversacional
2240 INPUT "NOM (para acabar pulsar return)", NOM$
2250 IF NOM$ = "" THEN GOTO 2470
2260 INPUT "INDICATIVO TELEFONICO (1 ó 2 cifras)"; INDIC
2270 INPUT "NUMERO DE TELEFONO (6 ó 7 cifras)"; NUMT$
2280 LINE INPUT "DIRECCION": DIR$
2290 INPUT "CODIGO POSTAL": CODE
2300 INPUT "CIUDAD"; CIUDAD$
2310 PRINT # 1, NOM$; ", "; INDIC; NUMT$; ", "; CAR$ (34); DIR$;
2320 PRINT # 1, CAR$ (34); ", "; CODE, CIUDAD$
2330 GOTO 2240
2340 '
2350 '---- creación por registro
2360 PRINT "entre, en orden:"
2370 PRINT "NOMBRE, INDIC TEL, DIRECCION, CODIGO POSTAL,";
2380 PRINT "CIUDAD"
2390 PRINT "separados por comas"
2400 PRINT "PARA ACABAR entrar una línea vacía"
2410 '
2420 LINE INPUT ENR$
2430 IF ENR$ = "" THEN GOTO 2470
2440 PRINT # 1, ENR$
2450 GOTO 2420
2460 '
2470 '---- cierre del fichero
2480 CLOSE #1
2490 RETURN
```

A partir de la 2230, se realiza (como en el apartado 4.2. del capítulo 3), la creación conversacional del fichero. Las instrucciones de lectura desde el teclado (2240 a 2300) y de escritura sobre el fichero (2310 a 2320) se repiten hasta que el usuario suministre una cadena vacía para el nombre. Hemos utilizado la instrucción LINE INPUT (en 2280) para entrar la dirección. Esto evita al usuario el situar su respuesta entre comillas, ya que contiene una coma.

En 2340 se halla la creación por registro (como en el apartado 3.5. del capítulo 3). Se limita a leer una cadena (en la variable ENR$) y la reescribe tal como está en el fichero. Además, se repiten estas instrucciones hasta que el usuario suministre una cadena vacía.

En ambos casos, la ejecución continúa en 2470 donde se cierra el fichero antes de salir de la rutina.

5.5. RUTINA DE LISTADO

```
3000  '
3010  '****************************************
3020  '**** RUTINA DE LISTADO DEL FICHERO ****
3030  '****************************************
3040  '---- apertura del fichero y selección de opción
3060  '          (por registro o con formato)
3070  OPEN "I", # 1, NOMF$
3080  PRINT "seleccione el modo de listado deseado : "
3090  PRINT "      ER    por registro"
3100  PRINT "      MF    con formato"
3110  INPUT R$
3120  IF R$ = "ER" THEN GOTO 3160
3130  IF R$ = "MR" THEN GOTO 3210
3140  PRINT "*** mala selección – entre de nuevo": GOTO 3110
3150  '
3160  '---- listado por registro
3170  LINE INPUT # 1, ENR$
3180  PRINT ENR$
3190  IF EOF (1) <> –1 THEN GOTO 3170
3200  GOTO 3280
3210  '---- listado con formato
3220  INPUT #1, NOM$, INDIC, NUMT$, DIR$, CODE, CIUDAD$
3230  PRINT NOM$; TAB (20); " ("; INDIC; ") "; NUMT$
3240  PRINT TAB (20); DIR$; TAB (60); CODE; TAB (67); CIUDAD$
3250  IF EOF (1) <> –1 THEN GOTO 3220
3260  '
3270  '---- cierre del fichero
3280  CLOSE # 1
3290  RETURN
```

Después de abrir el fichero (en 3070), aparece un "sub-menú" (3080 a 3150) que permite al usuario seleccionar entre:

- Listado común por registro (3160 a 3200): cada registro se escribe tal como aparece en el fichero,
- Listado "formateado" (3210 a 3260): la información de cada registro se presenta según un determinado formato, gracias especialmente a la función TAB. Encontrará en el apartado 8, un ejemplo de listado obtenido con esta opción.

En ambos casos, el fichero se cierra y se sale de la rutina.

5.6. RUTINA DE BUSQUEDA DE UNA PERSONA

```
4000  '
4010  '*********************************************
4020  '****RUTINA DE BUSQUEDA DE UNA PERSONA****
4030  '*********************************************
4040  '
4050  '---- apertura del fichero y entrada del nombre a buscar
4060  OPEN "I", #1, NOMF$
4070  INPUT "entre el nombre buscado"; NOMR$
4080  '
4090  '---- búsqueda del nombre hasta encontrarle o eof
4100  INPUT #1, NOM$, INDIC, NUMT$, DIR$, CODE, CIUDAD$
4110  IF NOM$ = NOMR$ THEN GOTO 4150
4120  IF EOF(1)<>-1 THEN GOTO 4100
4130  '---- nombre ausente
4140  PRINT "*** nombre ausente del fichero" : GOTO 4190
4150  '---- nombre encontrado
4160  PRINT "nombre buscado : "; NOMR$
4170  PRINT "teléfono       : " ("; INDIC; ")":NUMT$
4180  PRINT "dirección      :"; DIR$; CODE; CIUDAD$
4190  '---- cierre del fichero
4200  CLOSE #1
4210  RETURN
```

No presenta ninguna dificultad en particular. Una vez que se ha solicitado al usuario el entrar el nombre a buscar, se examina sucesivamente cada registro hasta encontrar el nombre solicitado o hasta encontrar el

fin de fichero. En el primer caso, se escribe toda la información relativa a esa persona. En el segundo caso, se indica que el nombre solicitado no se encuentra en el fichero.

5.7. RUTINA DE LISTADO SELECTIVO

Se trata de una función que todavía no hemos realizado en el curso de los capítulos precedentes. Consiste en suministrar el listado de los registros que responden a un cierto criterio. Por ejemplo, puede que se desee listar todos aquellos cuyo indicativo telefónico sea 38.

Hemos previsto los criterios de selección (no existe ningún impedimento para añadir otros):

- Indicativo telefónico (IN),
- Código postal (CP),
- Número de departamento (DE): este último no figura explícitamente en el fichero, pero se puede calcular a partir del código postal.

```
5000  '
5010  '****************************************
5020  '**** RUTINA DE LISTADO SELECTIVO ****
5030  '****************************************
5040  '
5050  '---- apertura del fichero, selección de opción
5060  '          y de la información a seleccionar
5070  OPEN "I", #1, NOMF$
5080  PRINT "seleccione el modo de listado deseado"
5090  PRINT " IN por indicativo telefónico"
5100  PRINT " DE por número de departamento"
5110  PRINT " CP por código postal"
5120  INPUT R$
5130  IF R$ = "IN" OR R$ = "DE" OR R$ = "CP" THEN GOTO 5160
5140  PRINT "*** mala selección - entre de nuevo" : GOTO 5120
5150  '
5160  IF R$ = "IN" THEN INPUT "entre el indicativo"; INDICR
5170  IF R$ = "DE" THEN INPUT "entre el número de dept"; NUMDR
```

```
5180  IF R$ = "CP" THEN INPUT "entre el código postal"; CODER
5190  '
5200  '---- búsqueda en el fichero
5210        INPUT # 1, NOM$, INDIC, NUMT$, DIR$, CODE, CIUDAD$
5220        '---- el registro interesa
5230        IF R$ = "IN" AND INDIC = INDICR THEN GOTO 5280
5240        IF R$ = "DE" AND INT (CODE/1000) = NUMDR THEN
            GOTO 5280
5250        IF R$ = "CP" AND CODE = CODER THEN GOTO 5280
5260        '---- no
5270        GOTO 5310
5280        '---- sí – listado del registro seleccionado
5290        PRINT NOM$; TAB (20); " ("; INDIC; ") ";NUMT$
5300        PRINT TAB (20); DIR$; TAB (60); CODE; TAB (67); CIUDAD$
5310  IF EOF (1) <> -1 THEN GOTO 5210
5320  '
5330  '---- cierre del fichero
5340  CLOSE # 1
5350  RETURN
```

Aquí, después de abrir el fichero (5070), se hace elegir al usuario una de las tres funciones IN, DE o CP (5080 a 5140). A continuación se le solicita (5160 a 5180) que entre el "valor" del criterio elegido (es decir, el indicativo seleccionado si ha elegido IN, el número de departamento para DE o el código postal para CP).

Desde la 5200 a 5320, se encuentra un bucle que permite examinar todos los registros del fichero. De la 5220 a 5250, se investiga si el registro leído es un registro a considerar. Si es así, se escribe (5290 a 5300) en caso contrario no se hace nada.

5.8. EJEMPLO DE UTILIZACION

A título de ejemplo, le proporcionamos una pequeña ejecución de este programa. Por brevedad, se ha suprimido una parte a nivel de creación del fichero. Este "recorte" viene materializado por una columna de puntos suspensivos.

```
RUN
GESTION DE UN FICHERO REPERTORIO

entre el nombre del fichero? REPERTA

seleccione una de las siguientes funciones
        CR creación
        LI listado de todo el fichero
        RE búsqueda de una persona
        LS listado selectivo
        FI final del trabajo

? CR
ATENCION, si el fichero REPERTA existe
              será DESTRUIDO
¿quiere crear REPERTA? (SI o NO)
? SI
seleccione el modo de creación deseado:
        CV conversacional
        ER por registro

? CV
NOMBRE (para acabar pulsar return) ? TOMAS
INDICATIVO TELEFONICO (1 ó 2 cifras) ? 86
NUMERO DE TELEFONO (6 ó 7 cifras) ? 482337
                              .
                              .
                              .

NOMBRE (para acabar pulsar return)?
seleccione una de las siguientes funciones
        CR creación
        LI listado de todo el fichero
        RE búsqueda de una persona
        LS listado selectivo
        FI final de trabajo
? LI
seleccione el modo de listado deseado :
        ER por registro
        MF con formato
? MF
```

```
TOMAS       (86) 482337
            c/ granada, 2                  58000 NERJA
COLON       (63) 274553
            c/ baleares, 56                81000 AVILA
LEDESMA     (75) 672830
            av) betanzos, 17               77780 CHINCHON
BALBOA      (94) 413355
            c/ donoso cortes, 17           83600 FRAGA
TABLADA     (77) 896313
            c/ hermosilla, 12              45510 MADRID
MONTIEL     (1) 9332765
            c/ gaudí, 147                  75008 BARCELONA
VIDAL       (6) 6333947
            bul. las ruedas, 29            91310 MANRESA
GRIÑON      (1) 2481530
            c/ sierras, 41                 75008 BARCELONA

seleccione una de las siguientes funciones
        CR creación
        LI listado de todo el fichero
        RE búsqueda de una persona
        LS listado selectivo
        FI final de trabajo
? RE
entre el nombre buscado? TABLADA
nombre buscado : TABLADA
teléfono       : (77) 896313
dirección      : c/ hermosilla, 12 45510 MADRID

seleccione una de las siguientes funciones
        CR creación
        LI listado de todo el fichero
        RE búsqueda de una persona
        LS listado selectivo
        FI final de trabajo
? LS
seleccione el modo de listado deseado
        IN por indicativo telefónico
        DE por número de departamento
        CP por código postal
```

```
? DE
entre el número de dept? 75
MONTIEL      (1) 9332765
             c/ gaudí, 147              75008 BARCELONA
GRIÑON       (1) 2481530
             c/ sierras, 41             75008 BARCELONA
seleccione luna de las siguientes opciones
        CR creación
        LIS listado de todo el fichero
        RE búsqueda de una persona
        LS listado selectivo
        FI final de trabajo
? FI
OK
```

6

Otro programa con menú: "consulta" de un fichero cualquiera

6.1. INTRODUCCION

En el capítulo 4 dijimos que un programa de listado completo de los registros de un fichero podría ser útil para descubrir posibles errores de programación. Sin embargo, cuando se va a tratar con ficheros de gran volumen, el listar todos los registros puede resultar tedioso. En este caso, es más cómodo el poder examinar solamente uno o varios registros.

Por otro lado, al margen de cualquier tipo de error, puede ser interesante el disponer de un programa capaz de examinar cualquier registro de cualquier fichero (secuencial).

6.2. PROGRAMA DE "CONSULTA" DE UN FICHERO

Vamos pues a confeccionar un programa que permita consultar ciertos registros a partir de las peticiones expresadas por el usuario. Se han previsto tres funciones (además de la de parada):

- Listar un cierto número de registros sucesivos (a partir del sitio en que estamos dentro del fichero).
- Saltar un cierto número de registros (permite consultar ciertos registros sin listarlos todos).

– Situarse al comienzo del fichero: esta función es muy útil si se necesita listar un registro situado delante del lugar en el que nos encontramos. En efecto, como la lectura debe ser secuencial, no se puede volver hacia atrás. La única solución consiste en cerrar el fichero y volverle a abrir.

Por otra parte, se ha previsto un "contador de registro" que permitirá precisar cual es el rango del registro que se está examinando.

```
1000  '**** INVESTIGACION DE UN FICHERO SECUENCIAL
1010  '
1020  '---- lectura del nombre del fichero y apertura
1030  PRINT "INVESTIGACION DE UN FICHERO SECUENCIAL" : PRINT
1040  INPUT "NOMBRE DEL FICHERO "; NOMF$
1050  OPEN "I", # 1, NOMF$
1060  COMPT = 0
1070  '
1080  '---- repetición de la selección de una función y ejecución
1090  ' hasta petición de parada
1100  PRINT "seleccione la función deseada :"
1110  PRINT " LE listado de registro"
1120  PRINT " SA salto de registros"
1130  PRINT " DB situarse al comienzo del fichero"
1140  PRINT " FI fin"
1150  INPUT R$
1160  IF R$ = "LE" THEN GOSUB 2000 : GOTO 1100
1170  IF R$ = "SA" THEN GOSUB 2000 : GOTO 1100
1180  IF R$ = "DB" THEN GOSUB 3000 : GOTO 1100
1190  IF R$ = "FI" THEN CLOSE # 1 : END
1200  PRINT "*** mala función – entre de nuevo" : GOTO 1150
2000  '
2010  '************************************************
2020  '*** RUTINA DE LISTADO O SALTO DE REGISTROS ***
2030  '************************************************
2040  '
2050  '----entrada número registros en cuestión y controles
2060  PRINT " ¿cuántos registros quiere? ";
2070  IF R$ = "LE" THEN PRINT "listar"
2080  IF R$ = "SA" THEN PRINT "saltar"
2090  INPUT NENR
```

```
2100  IF NENR < = 0 THEN PRINT "*** mal valor" : RETURN
2110  IF EOF (1) = –1 THEN PRINT "*** FIN DE FICHERO" : RETURN
2120  '
2130  '———— listado de los registros pedidos (para si EOF)
2140  FOR I = 1 TO NENR
2150      LINE INPUT # 1, ENREG$
2160      COMPT = COMPT + 1
2170      IF R$ = "LE" THEN RRINT "ENREG NO"; COMPT : ENREG$
2180      IF EOF (1) = –1 THEN PRINT "*** EOF ENCONTRADO" : RETURN
2190  NEXT 1
2200  RETURN
3000  '
3010  '*************************************************************
3020  '*** RUTINA DE POSICIONAMIENTO AL COMIENZO DEL FICHERO ***
3030  '*************************************************************
3040  '
3050  CLOSE # 1
3060  OPEN "I", # 1, NOMF$
3070  COMPT = 0
3080  RETURN
```

El programa principal tiene una apariencia que le resultará ahora familiar. Hemos utilizado además unas rutinas para la realización de las diferentes funciones propuestas en el menú.

No obstante, como la función de salto de registro nos ha parecido semejante a la de listar, no hemos utilizado más que una sola rutina para ambas. Por otro lado, la cuenta de los registros obliga a la inicialización a cero de un contador (instrucción 1060). Finalmente, la apertura del fichero se ha realizado en el programa principal. Si se hubiera realizado, por ejemplo, en la rutina de salto, ello implicaría para cada llamada un posicionamiento al comienzo del fichero, contrariamente a lo deseado.

La rutina de listado o de salto comienza por solicitar del usuario el número de registros a tratar (2050 a 2090). Se comprueba (2100) que este número es positivo; en caso contrario está prevista una salida (se hubiera podido prever igualmente el realizar de nuevo la pregunta). Por otra parte, se ha verificado si nos encontramos al final del fichero (2110); en efecto, en este caso, la primera lectura por INPUT entrañaría un error (con parada del programa). Además, después de denunciar esta anomalía, se sale del sub-programa. Observemos que ante esta situación, el usuario no tiene otra solución que solicitar el posicionamiento al comienzo del fichero (o parar).

Cuando la petición es válida, se realiza entonces un bucle (2140 a 2190) que permite examinar el número de registros requerido. Dentro de este bucle es donde se necesita conocer si la función elegida es LE (en cuyo caso se lista el registro) o SA. En cualquier caso, el contador de registros se incrementa en uno (2160). Además, es indispensable verificar (2180) después de cada lectura de un nuevo registro, que no se ha detectado el fin de fichero. Si fuera así, un mensaje le señalaría esta situación al usuario y se saldría de la rutina.

Finalmente, la rutina de reposicionamiento al principio del fichero, se encarga de cerrar y volver a abrir el fichero. Bajo estas condiciones, el próximo registro eventualmente leído, será el primero. Obviamente, el contador se restaura nuevamente a cero.

Observación: En el capítulo anterior, las aperturas y cierres del fichero se realizaban dentro de las rutinas relativas a las distintas funciones del menú. Aquí, se realizan una sola vez en el programa principal y eventualmente en la rutina de posicionamiento al comienzo del fichero. No se pueden en realidad establecer reglas sobre el sitio donde se deben situar estas instrucciones. Sin embargo Vd. debe cuidar su manipulación. Deberá evitar especialmente el abrir un fichero ya abierto o el cerrar un fichero que aún no ha sido abierto.

Veamos ahora un ejemplo de utilización de nuestro programa de consulta:

```
RUN
INVESTIGACION DE UN FICHERO SECUENCIAL

NOMBRE DEL FICHERO ? REPERTA
seleccione la función deseada:
        LE    listado de registro
        SA    salto de registros
        DB    situarse al comienzo del fichero
        FI    fin
? LE
¿cuántos registros quiere listar?
? 1
ENREG NO 1 TOMAS, 86 482337, "c/ granada, 2", 58000 NERJA
seleccione la función deseada:
        LE    listado de registro
        SA    salto de registros
        DB    situarse al comienzo del fichero
        FI    fin
```

```
? SA
¿cuántos registros quiere saltar?
? 3
seleccione la función deseada:
        LE   listado de registro
        SA   salto de registros
        DB   situarse al comienzo del fichero
        FI   fin
? LE
¿cuántos registros quiere listar?
? 2
ENREG NO 5 TABLADA, 77 896313, "c/ hermosilla, 12", 45510 MADRID
ENREG NO 6 MONTIEL, 1 9332765, "c/ gaudí, 147", 75008 BARCELONA
seleccione la función deseada:
        LE   listado de registro
        SA   salto de registros
        DB   situarse al comienzo del fichero
        FI   fin
? SA
¿cuántos registros quiere saltar?
? 5
*** EOF ENCONTRADO
seleccione la función deseada:
        LE   listado de registro
        SA   salto de registros
        DB   situarse al comienzo del fichero
        FI   fin

? DB
seleccione la función deseada:
        LE   listado de registro
        SA   salto de registros
        DB   situarse al comeinzo del fichero
        FI fin
? LE
¿cuántos registros quiere listar?
? 1
ENREG NO 1 TOMAS, */ 482337, "c/ granada, 2", 58000 NERJA
seleccione la función deseada:
        LE listado de registro
```

```
SA   salto de registros
DB   situarse al comienzo del fichero
FI   fin

? FI
OK
```

7

Actualización de un fichero secuencial

7.1. INTRODUCCION

Hasta aquí, hemos aprendido a crear y después a utilizar un fichero, sin introducirle modificaciones. Generalmente, es indispensable el permitir la evolución del contenido de un fichero. Así, por ejemplo, en nuestro repertorio de los anteriores capítulos, es muy probable que con el paso del tiempo, tengamos necesidad de:

- Añadir nuevos registros correspondientes a nuevos nombres,
- Suprimir registros que ya no se necesitan,
- Modificar ciertos registros, en caso, por ejemplo de que una persona cambie de número de teléfono o de dirección.

Esta función puede ser igualmente indispensable si cuando se creó el fichero, se introdujeron errores (de ortografía o de número equivocado); en este caso, sería deplorable el tener que volver a crear la totalidad del fichero (teniendo en cuenta que es probable que volvieran a aparecer nuevos errores).

En general, estas funciones se denominan "ACTUALIZACIONES".

En este capítulo, vamos a aprender cómo efectuar la actualización de ficheros secuenciales.

7.2. PROBLEMAS DERIVADOS DE LA ACTUALIZACION DE UN FICHERO SECUENCIAL

Como hemos dejado entrever, es imposible añadir, modificar o suprimir un registro cualquiera de un fichero secuencial. Por otra parte, observe que cuando Vd. abre un fichero por la instrucción OPEN, sólo selecciona entre dos posibilidades:

- "O" creación del fichero
- "I" utilización de un fichero existente. Se trata de consultas puramente pasivas, o sencillamente de lecturas.

Dicho de otro modo, la misma naturaleza de la instrucción OPEN demuestra que la actualización es imposible.

¡No obstante tranquilícese! Existe una posibilidad indirecta de actualización. Consiste simplemente en crear un nuevo fichero, utilizando los registros del antiguo y la información de las modificaciones aportadas por el usuario. Por ejemplo, para corregir el registro número 25 de un fichero llamado REPERT, se creará un segundo fichero llamado REPERT2. Se copiarán los 24 primeros registros de REPERT y a continuación se le presentará al usuario el número 25 de REPERT, solicitándole las modificaciones que desea incorporar. Estas se realizarán sobre las variables correspondientes y después el resultado se escribirá en REPERT2. Si no se requieren más modificaciones, se copiará sobre REPERT2 el resto de registros de REPERT.

Este mismo procedimiento se puede utilizar para añadir nuevos registros. Para ello, en ciertos momentos de la copia, se solicitará al usuario, que facilite la información relativa al registro a añadir. Finalmente, la "supresión" de un registro se realizará simplemente evitando la copia del registro (o registros) a eliminar, sobre el nuevo fichero.

En cualquier caso, cuando finaliza la actualización, el usuario se encuentra con dos ficheros. Para evitar la proliferación de ficheros inútiles, se deberá destruir el fichero antiguo. Este es el cometido del mandato "KILL". Si llega el caso, se puede renombrar el nuevo fichero con el nombre del antiguo, por medio del mandato "NAME".

Así, en nuestro ejemplo anterior, después de haber creado REPERT2 a partir de REPERT, podíamos teclear estas dos instrucciones:

```
KILL "REPERT"
NAME "REPERT2" as "REPERT"
```

(la segunda instrucción cambiaría el nombre REPERT2 por REPERT). Estas instrucciones, presentadas aquí en "modo directo", pueden figurar igualmente en el contexto de un programa.

7.3. PROGRAMA PARA AÑADIR REGISTROS AL FINAL DEL FICHERO

Se trata de la forma más simple de actualización. Vamos pues a ampliar nuestro fichero repertorio (con dirección), de los capítulos anteriores. Como ya hemos dicho, necesitamos recopiar nuestro antiguo repertorio en un nuevo fichero al que añadiremos a continuación los registros deseados.

Esto es lo que realiza el programa:

```
1000  '**** EXTENSION DE UN FICHERO REPERTORIO
1010  '
1020  '---- entrada nombres ficheros fuente y destino – apertura
1030  PRINT "EXTENSION DE UN FICHERO REPERTORIO"
1040  INPUT "nombre del fichero fuente "; NOMF1$
1050  INPUT "nombre del fichero destino "; NOMF2$
1060  OPEN "I", # 1, NOMB1$
1070  OPEN "O", # 2, NOMF2$
1080  '
1090  '---- recopia del fichero fuente en fichero destino
1100  LINE INPUT # 1, ENR$
1110  PRINT # 2, ENR$
1120  IF EOF (1) <> –1 THEN GOTO 1100
1130  '
1140  '---- adición de registros
1150  PRINT "entre ahora los registros a añadir"
1160  PRINT "para acabar, responda return para el nombre"
1170  '
1180  PRINT
1190  INPUT          "nombre          : " ; NOM$
1200  IF NOM$ = " "  THEN GOTO 1320
1210  INPUT          "indicativo      : " ; INDIC
1220  INPUT          "número teléfono : " ; NUMT$
1230  LINE INPUT     "dirección       : " ; DIR$
1240  INPUT          "código postal   : " ; CODE
```

```
1250  INPUT          "ciudad              : " ; CIUDAD$
1260  '
1270  PRINT # 2, NOM$; ", "; INDIC; NUMT$; ", "; CAR$ (34);
1280  PRINT # 2, DIR$; CAR$ (34); ", "; CODE; CIUDAD$
1290  '
1300  GOTO 1180
1310  '
1320  '---- cierre
1330  CLOSE # 1 : CLOSE # 2
1340  '
1350  '---- destrucción eventual del fichero fuente
1360  '      y cambio de nombre al fichero destino
1370  PRINT
1380  PRINT "quiere destruir el fichero fuente "; NOMF1$
1390  PRINT " y que"; NOMF2$; "tome el nombre"; NOMF1$
1400  PRINT " SI o NO"
1410  INPUT R$
1420  IF R$ = "NO" THEN END
1430  IF R$ <> "SI" THEN GOTO 1400
1440  KILL NOMF1$
1450  NAME NOMF2$ as NOMF1$
```

Desde la instrucción 1030 a 1050, se solicitan los nombres del fichero a actualizar (fuente) y del fichero destinado a ser el actualizado (destino). El primero debe existir previamente, mientras que el segundo es creado por el programa.

El conjunto de registros del fichero fuente se vuelve a copiar íntegramente sobre el fichero destino (1100 a 1120). Después (de 1140 a 1300) se solicita del usuario que facilite la información correspondiente a los registros a añadir (como se haría en un programa de creación). Se ha previsto una parada cuando el nombre facilitado sea una cadena vacía (1200).

Los dos ficheros se cierran en 1330. A continuación, se solicita al usuario la elección entre dos posibilidades:

- Conservar los dos ficheros (fuente y destino),
- Destruir el primer fichero y dar al segundo el nombre del primero.

En el segundo caso, las operaciones necesarias son realizadas por las instrucciones KILL (1440) y NAME (1450) de las que ya hablamos en el apartado 7.2.

Vamos a ejecutar este programa de añadir dos registros a nuestro fichero REPERTA, creado en el capítulo 5.

```
RUN
EXTENSION DE UN FICHERO REPERTORIO
nombre del fichero fuente ? REPERTA
nombre del fichero destino ? TEMPO
entre ahora los registros a añadir
para acabar, responda return para el nombre

nombre          : ? LOPEZ
indicativo      : ? 84
núm. teléfono   : ? 612842
dirección       : : plaza mayor, 2
código postal   : ? 39400
ciudad          : ? MALAGA

nombre          : ? FALCON
indicativo      : ? 92
núm. teléfono   : ? 134521
dirección       : c/ silva, 19
código postal   : ? 14200
ciudad          : ?SANTIAGO

nombre          : ?

quiere destruir el fichero fuente REPERTA
y que TEMPO tome el nombre REPERTA
      SI o NO
? SI
OK
```

Verifiquemos que la actualización se ha realizado satisfactoriamente, para lo cual, haremos un listado completo de los registros del fichero. Tendríamos pues ésto:

```
TOMAS, 86 482337, "c/ granada, 2", 58000 NERJA
COLON, 63 274553, "c/ baleares, 56", 81000 AVILA
LEDESMA, 75 672830, "Av. vetanzos, 17 ", 77780 CHINCHON
BALBOA, 94 413355, "c/ donoso cortés, 17", 83600 FRAGA
TABLADA, 77 896313, "c/ hermosilla, 12", 45510 MADRID
```

```
MONTIEL, 1 9332765, "c/ gaudí, 147", 75008 BARCELONA
VIDAL, 6 6333947, "bul. las ruedas', 29", 91310 MANRESA
GRIÑON, 1 2481530, "c/ sierras, 41", 75008 BARCELONA
LOPEZ, 84 612842, "plaza mayor, 2", 39400 MALAGA
FALCON, 92 134521, "c/ silva, 19", 14200 SANTIAGO
```

Las dos últimas líneas corresponden efectivamente a los registros añadidos.

7.4. PROGRAMA GENERAL DE ACTUALIZACION

Considerando siempre nuestro fichero repertorio con direcciones, vamos ahora a escribir un programa que permita cualquier tipo de actualización: supresión, añadido o modificación de registros. Como veremos en el siguiente apartado, existen varias formas de dar las directrices para la actualización. Aquí, procederemos así:

- Cada registro del fichero fuente se presentará (en la pantalla) al usuario, que escogerá entre:
  - conservarle como está,
  - suprimirle,
  - modificarle.
- Además, en cada momento, tendrá la posibilidad de escoger entre:
  - pasar al registro siguiente,
  - insertar un registro (podrá añadir varios consecutivos, repitiendo esta función).

7.4.1. El programa principal

```
1000 '**** ACTUALIZACION DE UN FICHERO REPERTORIO SECUENCIAL
1020 '
1030 ' las funciones posibles son *
1040 '  – añadir
1050 '  – suprimir
1060 '  – modificar
```

```
1070  '
1080  '---- entre nombres ficheros fuente y destino – apertura
1090  PRINT "PROGRAMA DE ACTUALIZACION DE UN REPERTORIO"
1100  INPUT "nombre del fichero fuente"; NOMF1$
1110  INPUT "nombre del fichero destino"; NOMF2$
1120  OPEN "I", #1, NOMF1$ : OPEN "O", #2, NOMF2$
1130  '
1140  '---- seleccione, ejecución de una función hasta petición de
1150  '          parada
1160  PRINT
1170  PRINT "seleccione una de las funciones : "
1180  PRINT " AJ : añadir un registro"
1190  PRINT " ES : consultar el registro siguiente"
1200  PRINT " FI : fin"
1210  INPUT R$
1220  IF R$ = "AJ" THEN GOSUB 2000 : GOTO 1170
1230  IF R$ = "ES" THEN GOSUB 3000 : GOTO 1170
1240  IF R$ = "FI" THEN GOTO 1270
1250  PRINT " *** mala función – entre de nuevo" : GOTO 1210
1260  '
1270  '---- fin – recopia del resto del fichero fuente en destino
1280  IF EOF (1) = –1 THEN GOTO 1320
1290     LINE INPUT #1, ENR$
1300     PRINT #2, ENR$
1310  GOTO 1280
1320  CLOSE #1 : CLOSE #2
1330  END
```

Se comienza por abrir los dos ficheros (fuente y destino) cuyos nombres son proporcionados por el usuario (1080 a 1120).

A continuación viene un bucle (1140 a 1250) en el cual se ofrece la elección entre la consulta del registro siguiente (código ES) o la inserción de un registro (código AJ) y esto, hasta que el usuario indique que ha finalizado (código FI). La parte no examinada todavía del fichero fuente si existe, se copia en el fichero destino (1260 a 1310). Observe en 1290, la utilización de LINE INPUT que permite, en este caso, simplificar la lectura del fichero fuente.

Cada una de las dos funciones AJ y ES se realizan por medio de una rutina.

7.4.2. Rutina de inserción de registros

```
2000  '
2010  '************************************
2020  '*** RUTINA DE AÑADIR REGISTROS ***
2030  '************************************
2040  '
2050  PRINT "entre el registro a añadir"
2060  INPUT "nombre"; NOM$
2070  INPUT "INDICATIVO"; INDIC
2080  INPUT "NUMERO TEL ": NUMT$
2090  LINE INPUT "DIRECCION"; DIR$
2100  INPUT "CODIGO POSTAL"; CODE
2110  INPUT "CIUDAD"; CIUDAD$
2120  '
2130  PRINT # 2, NOM$; ", "; INDIC; NUMT$; ", "; CAR$ (34); DIR$;
2140  PRINT # 2, CAR$ (34); ", "; CODE; CIUDAD$
2150  RETURN
```

Es una rutina clásica. Observe en ella solamente la utilización en 2090 de LINE INPUT para la entrada de la dirección desde el teclado. Ello permite al usuario, responder con una cadena que eventualmente contenga una coma, sin tener que situarla entre comillas.

7.4.3. Rutina de consulta del registro siguiente

```
3000  '
3010  '*****************************************************
3020  '*** RUTINA DE CONSULTA DEL REGISTRO SIGUIENTE ***
3030  '*****************************************************
3040  '
3050  '  funciones posibles: conservar, suprimir, modificar
3060  '
3070  '---- verificar fin del fichero fuente
3080  IF EOF (1) = -1 THEN PRINT "*** FIN FICHERO FUENTE" : RETURN
3090  '
3100  '---- lectura de un registro y presentación en pantalla
3110  INPUT # 1, NOM$, INDIC, NUMT$, DIR$, CODE, CIUDAD$
```

```
3120  PRINT "he aquí el registro siguiente:
3130  PRINT "NOMBRE "; NOM$
3140  PRINT "TELEFONO ("; INDIC; ") "; NUMT$
3150  PRINT "DIRECCION "; DIR$; CODE; CIUDAD$
3160  '
3170  '---- presentación del menú y lectura de selección
3180  PRINT "quiere Vd.:"
3190  PRINT " conservarle – teclee CO"
3200  PRINT " suprimirle – teclee SU"
3210  PRINT " modificarle – teclee MO"
3220  INPUT R$
3230  IF R$ <> "CO" THEN GOTO 3260
3240     PRINT # 2, NOM$;", "INDIC; NUMT$; ", "; CAR$ (34); DIR$;
3250     PRINT # 2, CAR$ (34); ", "; CODE; CIUDAD$
3250     RETURN
3260  IF R$ = "SU" THEN RETURN
3270  IF R$ = "MO" THEN GOSUB 4000 : RETURN
3280  PRINT " *** mala selección – entre de nuevo : GOTO 3220
```

Se comienza por verificar que la solicitud formulada es factible, asegurándose de que no nos encontramos en el final del fichero fuente (3080). Si así fuera, se escribe un mensaje de aviso y se sale de la rutina. Ello significa que en tal caso el usuario deberá efectuar de nuevo una elección entre AJ, ES o FI. Como es lógico, si solicita nuevamente ES, volverá a la misma situación; solamente la selección de ES y FI serán realmente significativas.

En caso contrario (el caso más general), es decir, si existen todavía registros en el fichero fuente, se leerá el siguiente en las variables correspondientes (3110) y se presentará su contenido sobre pantalla (3120 a 3150). Se ofrecen ahora tres posibilidades (3180 a 3210): conservar (CO), suprimir (SU) o modificar (MO).

La función CO se trata simplemente en 3230 a 3250, re-escribiendo el último registro leído en el fichero destino y saliendo de la rutina.

La última SU se realiza de forma aún más sencilla, saliendo de la rutina sin hacer nada (3260).

Por último, MO entraría (en 3270) la llamada a una rutina cuyo contenido es el siguiente:

```
4000  '
4010  '*********************************************
4020  '*** RUTINA DE MODIFICACION DE UN REGISTRO ***
4030  '*********************************************
4040  '
4050  PRINT "para cada información : "
4060  PRINT " – entre el nuevo valor para modificar"
4070  PRINT " – pulse RETURN para no modificar"
4080  '
4090  INPUT "NOMBRE "; NOMR$
4100     IF NOMR$ <> " " THEN NOM$ = NOMR$
4110  INPUT "INDICATIVO "; INDICR
4120     IF INDICR <> 0 THEN INDIC = INDICR
4130  INPUT "NUMERO TELEFONO "; NUMTR$
4140     IF NUMTR$ <> " " THEN NUMT$ = NUMTR$
4150  LINE INPUT "DIRECCION "; DIRR$
4160     IF DIRR$ <> " " THEN DIR$ = DIRR$
4170  INPUT "CODIGO POSTAL "; CODER
4180     IF CODER <> 0 THEN CODE = CODER
4190  INPUT "CIUDAD "; CIUDADR$
4200     IF CIUDADR$ <> " " THEN CIUDAD$ = CIUDADR$
4210  '
4220  PRINT #2, NOM$; ", "; INDIC; NUMT$; " , "; CAR$ (34); DIR$;
4230  PRINT #2, CAR$ (34); " , "; CODE; CIUDAD$
4240  RETURN
```

Con esto se permite al usuario que proporcione, de entre las seis informaciones que constituyen un registro, sólo aquellas que desea modificar. Para ello se utilizan seis variables suplementarias (diferentes de las seis relativas a un registro): NOMR$, INDICR, NUMTR$, DIRR$, CODER, CIUDADR$. Estas albergarán el valor propuesto. Se conviene en que si este valor es una cadena vacía para las variables cadenas o cero para las variables numéricas (en ambos casos, se obtienen pulsando "return" únicamente), la información se conservará tal como figuraba en el registro. Esta forma de proceder permite una actualización relativamente rápida, dado que no es necesario teclear más que las correcciones.

Observación: No hemos tenido en cuenta aquí la posibilidad de dar al fichero destino el nombre del fichero fuente.

7.4.4. Ejemplo de utilización

He aquí una actualización del fichero REPERTA; suponemos que al comienzo se encuentra en el estado descrito al final del apartado 7.3 (contenido diez registros).

```
? RUN
PROGRAMA DE ACTUALIZACION DE UN REPERTORIO
nombre del fichero fuente? REPERTA
nombre del fichero destino? TEMPO

seleccione una de las siguientes funciones:
        AJ : añadir un registro
        ES : consultar el registro siguiente
        FI : fin
? ES
he aquí el registro siguiente :
NOMBRE TOMAS
TELEFONO ( 86 ) 482337
DIRECCION c/ granada, 2 58000 NERJA
quiere Vd.:
conservarle – pulse CO
suprimirle – pulse SU
modificarle – pulse MO
? CO

seleccione una de las siguientes funciones :
        AJ : anadir un registro
        ES : consultar el registro siguiente
        FI : fin
? ES
he aquí el registro siguiente:
NOMBRE COLON
TELEFONO ( 63 ) 274553
DIRECCION c/ baleares, 53 81000 AVILA
quiere Vd. :
conservalre – pulse CO
suprimirle – pulse SU
modificarle – pulse MO
? MO
```

```
para cada información:
– entre el nuevo valor para modificar
– teclee RETURN para no modificar
NOMBRE?
INDICATIVO?
NUMERO TELEFONO ? 284553
DIRECCION ? c/ baleares, 46
CODIGO POSTAL ?
CIUDAD ?
seleccione una de las siguientes funciones:
        AJ: añadir un registro
        ES : consultar el registro siguiente
        FI : fin
? AJ
entre su registro a añadir
NOMBRE ? VIÑAS
INDICATIVO ? 51
NUMERO TEL ? 634827
DIRECCION ? 123, c/ darío, 23
CODIGO POSTAL ? 85400
CIUDAD ? LUGO
seleccione una de las siguientes funciones:
        AJ : añadir un registro
        ES : consultar el registro siguiente
        FI : fin
? FI
OK
```

A título indicativo, he aquí el listado total de los registros del fichero así actualizado. (Atención, aquí se llama TEMPO).

```
TOMAS, 86 482337, "c/ granada, 2", 58000 NERJA
COLON, 63 284553, "c/ baleares, 46", 81000 AVILA
VIÑAS, 51 634827, "c/ darío, 23", 85400 LUGO
LEDESMA, 75 672830, "av. betanzos, 17", 77780 CHINCHON
BALBOA, 94 413355, "c/ donoso cortés, 7", 83600 FRAGA
TABLADA, 77 896313, "c/ hermosilla, 12", 45510 MADRID
MONTIEL, 1 9332765, "c/ gaudí, 167", 75008 BARCELONA
VIDAL, 6  6333947, "bul. las ruedas, 29", 91310 MANRESA
```

```
GRIÑON, 1 2481530, "c/ sierras, 41", 75008 BARCELONA
LOPEZ, 84 612842, "pl. mayor, 2", 39400 MALAGA
FALCON, 92 134521, "c/ silva, 19", 14200 SANTIAGO
```

Compruebe que el segundo registro ha sido corregido como estaba previsto (a nivel del número de teléfono y de la dirección). Así mismo se ha insertado correctamente un registro (VIÑAS...) entre el segundo y el tercero.

7.5. OTRAS POSIBILIDADES DE ACTUALIZACION

El programa anterior muestra cada registro al usuario. Si el número de modificaciones a efectuar sobre el fichero es pequeño, quizás encuentre algo tediosa esta técnica. Imagine que tenga que corregir el registro número 241.

Sin proponerle nuevos programas, digamos que las técnicas siguientes, serán más o menos eficaces según las circunstancias:

*** Búsqueda del registro de un número dado**

El usuario debe disponer de un listado de su fichero (y por tanto, de una impresora).

*** Búsqueda de un nombre dado**

El listado aquí no es necesario. Sin embargo, si se desean buscar varios registros, se ahorrará tiempo si se consultan en el orden en que aparecen en el fichero. Si se dispone del listado, no plantea (demasiados) problemas. En caso contrario, no existe apenas otra salida más que ordenar sus registros por orden alfabético de nombres. Pero en tal caso, dirá Vd. ¿va a ser preciso clasificar el fichero en cada actualización? No es indispensable si preve la creación del fichero por orden alfabético y tiene cuidado de insertar en el sitio correcto, cada nuevo registro.

8

Ficheros de acceso directo: primer paso

8.1. GENERALIDADES

8.1.1. Ventajas e inconvenientes del secuencial

Ya ha visto Vd. que los ficheros secuenciales son relativamente fáciles de crear y consultar. Ello es debido, fundamentalmente, a que las instrucciones de lectura o escritura correspondientes (INPUT# y PRINT#) son semejantes a las utilizadas para leer del teclado o para escribir sobre pantalla. En contrapartida a esta relativa facilidad, el acceso puramente secuencial a estos ficheros, impone severas restricciones:

- La búsqueda de un registro dado es generalmente larga en tiempo,
- La actualización implica la creación de un nuevo fichero.

8.1.2. Concepto de acceso directo

Ahora bien, como dejamos entrever en el capítulo de introducción, la naturaleza misma de los discos o diskettes, permite el acceso (casi directo) a un registro dado. Técnicamente, esto se realiza por un mecanismo que permite a los dispositivos (cabezas) de lectura o de escritura,

situarse rápidamente sobre una pista dada. En cuanto al acceso al registro propiamente dicho, se realiza por el Basic que, una vez encontrada la pista, va a buscar (secuencialmente) el registro en cuestión.

Ciertamente, dirá Vd. entonces que no se accede realmente al registro Sin embargo, para el usuario todo sucede como si tuviera acceso directo. El se limita a decir qué registro (indicado por un número) desea leer o escribir, sin preocuparse de la forma en que el Basic lo encuentra en realidad.

8.1.3. Restricciones del acceso directo

Por lo tanto, para que el Basic sea capaz de encontrar el registro perteneciente a un número dado, es necesario:

- Que conozca las pistas sobre las que se encuentra el fichero. Esto es gestionado por el Basic que mantiene esta información en un repertorio (en inglés "directory").

- Que sepa cuántos registros hay por pista, para poder así "calcular" el número de la pista correspondiente a un registro dado. Bajo estas condiciones, vemos que cada pista debe albergar el mismo número de registros que, por lo tanto, deben ser de la misma longitud. Esta longitud puede variar de un fichero a otro (ciertos sistemas sin embargo, –especialmente el TRS 80– imponen registros de longitud fija.)

8.1.4. El acceso directo no es la panacea

En efecto, para acceder directamente a un registro, es preciso conocer el número. Esto significa que el acceso directo no ofrecerá ningún interés para el usuario más que cuando ese número tenga sentido para él. Por ejemplo, si consideramos uno de los ficheros "repertorio" de los capítulos precedentes, es posible que hacer un fichero de acceso directo aporte pocas ventajas. Efectivamente, ¿sabrá a qué corresponde el registro de orden ciento veinticinco o el cuarenta y nueve? En tal caso, se tendrá más bien, tendencia a "designar" el registro por el nombre que contiene. Pero entonces no estamos en las condiciones del acceso directo. Veremos, sin embargo (en el Capítulo 13), que a partir de un fichero de acceso directo, podemos realizar un acceso llamado "indexado" que responderá al problema planteado.

8.2. CREACION DE UN FICHERO BIBLIOTECA

8.2.1. Introducción

Quizás recuerde nuestro ejemplo del fichero biblioteca manual del capítulo 1. Ahora vamos a aprender a construir un fichero de acceso directo. El número de registros corresponderá al número del libro (obtenido, por ejemplo, dando el número 1 al primero comprado, el 2 al segundo y así sucesivamente).

Por ahora, no alojaremos en cada registro más que dos informaciones (de tipo cadena):

- Nombre del autor,
- Título del libro.

8.2.2. Apertura del fichero

Al igual que en el secuencial, se indica en una instrucción OPEN:

- El nombre del fichero,
- Un número que servirá para designarle a lo largo del programa.

Por otro lado, se da el tamaño de los registros. Aquí, hemos escogido 64 caracteres (octetos).

La instrucción se escribirá:

```
OPEN "R", #1, NOMF$, 64
```

Esto significa que al fichero cuyo nombre está contenido en la variable NOMF$, se le va a asignar el número 1 y que sus registros son de 64 caracteres de longitud.

Por otra parte, se dará cuenta de que se encuentra con el carácter "R" allí donde en secuencial se encontraba una de las letras "O" o "I". R es la abreviatura de "Random access file" (fichero de acceso directo). Notará que no existe selección entre creación y lectura. Volveremos sobre este punto que se puede considerar como una de las características de estos ficheros: teniendo un acceso directo, nada impide, en efecto, el leer el registro número doce, el escribir el cuarenta y ocho o modificar el treinta y siete.

8.2.3. Desglose del registro

Quizás piense que, habiendo definido el tamaño de sus registros, se le permite rellenarlos como quiera. En realidad, el Basic impone igualmente la definición de la ubicación destinada para cada información dentro de un registro. Esto nos obliga aquí a seleccionar cuántos caracteres (máximo) deseamos utilizar para el nombre de autor. Supongamos que son 16; nos quedan ahora pues 48 (64-16) caracteres para el título.

La instrucción FIELD (en inglés: campo), es la que nos permitirá decir al Basic como vamos a desglosar nuestro registro. Aquí escribiremos:

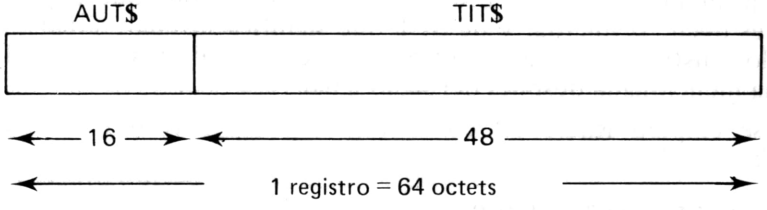
```
FIELD #1, 16 AS AUT$, 48 AS TIT$
```

#1 indica que se van a escribir los registros correspondientes al fichero número 1 (por lo tanto, la instrucción OPEN debe ser ejecutada antes de la instrucción FIELD relativa al mismo fichero).

16 AS AUT$ significa que un primer campo de 16 caracteres será designado por el símbolo AUT$.

Análogamente, 48 AS TIT$ indica que un segundo campo de 48 octetos estará designado por TIT$.

Lo representaremos esquemáticamente:



8.2.4. La zona tampón asociada al fichero

Podrá pensar, a priori, que AUT$ y TIT$ son nombres de variables ordinarias. En realidad, para trabajar con un fichero de acceso directo el Basic utiliza (también para los ficheros secuenciales), una zona tampón destinada a contener un registro (en realidad, al menos uno). Los símbolos AUT$ y TIT$ designan los emplazamientos en este tampón. No son variables en el sentido común del término.

Particularmente vamos a ver que para situar valores en estas zonas, necesitará utilizar una instrucción especial diferente a la de asignación.

8.2.5. Preparación de un registro

En el caso de los ficheros secuenciales, Vd. podía escribir un registro designando cualesquiera variables en una orden de PRINT#. Aquí, por el contrario, debe llenar cada uno de los "campos" del tampón con la ayuda de instrucciones especiales (LSET, RSET). Por ejemplo, para preparar un registro que contenga:

```
HUGO                como nombre de autor
LOS MISERABLES      como título
```

podremos escribir:

```
LSET   AUT$ = "HUGO"
LSET   TIT$  = "LOS MISERABLES"
```

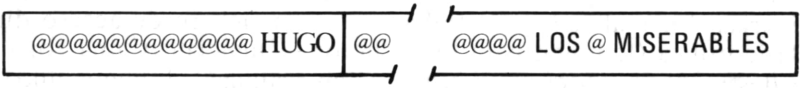
y obtendremos en el tampón:

La primera instrucción LSET (su nombre se compone del L como Left, es decir, izquierda, y de SET que significa instalar, situar), ha ordenado, justificado a la izquierda, en el campo llamado AUT$, la cadena mencionada después del signo igual.

Si se empleó RSET (R como Right, o sea derecha), en lugar de LSET:

```
RSET   AUT$ = "HUGO"
RSET   TIT$  = "LOS MISERABLES"
```

obtendríamos las cadenas ajustadas por la derecha, así:



donde el signo @ determina espacios a blancos.

A la derecha del signo igual de estas instrucciones (LSET, RSET), se podrá encontrar, no solamente una constante, sino cualquier expresión de tipo cadena.

Por el contrario, es ABSOLUTAMENTE INDISPENSABLE el utilizar estas instrucciones para llenar el tampón o buffer. Observe que si por error, Vd. escribió:

```
AUT$ = "HUGO"
```

sería una variable llamada AUT$ la que recibiría la cadena indicada y no el campo AUT$ descrito en la instrucción FIELD. Por otro lado, en la mayoría de los Basic existe una anomalía: toda referencia posterior a AUT$ (incluso por LSET o RSET), concernería en adelante a esta nueva variable y nunca más (dentro del programa) al campo correspondiente al tampón.

8.2.6. Escritura de un registro

Cuando el tampón está completamente lleno, Vd. puede ordenar al Basic que lo escriba en el fichero por medio de una instrucción PUT (en inglés: situar, poner) y deberá precisar:

- El número del fichero correspondiente
- El número del registro que va a escribir.

En efecto, contrariamente a lo que ocurre en el secuencial, Vd. puede escribir los registros sin importar el orden. Nada le obliga a comenzar por el primero.

Incluso, puede escribir sólo una parte de ellos (profundizaremos sobre este punto que comporta algunos problemas).

Así:

```
PUT #1, 25
```

escribe en el registro número 25 del fichero, el contenido del tampón asociado.

Igualmente:

```
PUT #1, I
```

permite escribir un registro cuyo número viene dado por el valor contenido en la variable I.

8.2.7. Programa de creación

He aquí un programa simplificado de creación de un fichero biblioteca compuesto de diez registros:

```
1000  '**** CREACION DE UN FICHERO BIBLIOTECA DE ACCESO DIRECTO
1010  '
1020  '---- entrada nombre de fichero, apertura, descr. de reg.
1030  PRINT "CREACION DE UN FICHERO BIBLIOTECA" : PRINT
1040  INPUT "entre el nombre del fichero  "; NOMF$
1050  NENR% = 10
1060  OPEN "R", # 1, NOMF$, 64
1070  FIELD # 1, 16 AS AUT$, 48 AS TIT$
1080  '
1090  '---- creación del fichero
1100  FOR I = 1 TO NENR
1110     PRINT : PRINT "registro NO"; I
1120     INPUT "autor  "; AUTOR$
1130     INPUT "título  "; TITULO$
1140     LSET AUT$ = AUTOR$
1150     LSET TIT$ = TITULO$
1160     PUT # 1, I
1170  NEXT I
1180  '
1190  CLOSE # 1
```

Aquí, los registros se crean en orden gracias a un bucle FOR (1100 a 1170) en el que el contador I hace de número de registro (en 1160).

Observe, en 1190, la presencia de una instrucción CLOSE, que juega el mismo papel que en secuencial y es indispensable.

Veamos un ejemplo de ejecución de este programa (cuidando la brevedad, hemos suprimido una parte del listado):

```
RUN
CREACION DE UN FICHERO BIBLIOTECA

entre el nombre del fichero ? BIBLI

REGISTRO NO 1
```

```
autor ? HUGO
título ? los miserables

REGISTRO NO 2
autor ? ALAIN
título ? propósitos sobre la dicha

REGISTRO NO 3
autor ? BAZIN
título ? la víspera

REGISTRO NO 4
      ,
      ,
      ,
autor ? TROYAT
título ? la cabeza sobre las espaldas

REGISTRO NO 10
autor ? MESSEGUE
título ? la naturaleza tiene razón

OK
```

8.3. UTILIZACION DEL FICHERO BIBLIOTECA

8.3.1. Introducción

Vamos a aprender aquí, cómo buscar un registro dado o cómo listar todo el fichero. Se trata pues de "consultas" puramente pasivas. En el Capítulo 10 estudiaremos la actualización en general.

La apertura del fichero se realiza exactamente como cuando su creación. (Ya que no existe diferencia entre lectura o escritura para el acceso directo). La misma instrucción FIELD sirve para describir sus registros.

De otra parte, se utiliza una nueva instrucción (GET) para recopiar un registro dado en el tampón (operación simétrica de PUT).

Así:

```
GET #1, I
```

recopia en el tampón asociado al fichero 1, el contenido del registro de número I.

8.3.2. Programa de consulta

Este programa simplificado, busca el registro cuyo número ha sido facilitado por el usuario.

```
1000  ' **** CONSULTA DE UN FICHERO BIBLIOTECA
1010  '
1020  '---- entrada nombre fichero, apertura, descrip. de reg.
1030  '        solicita número reg. buscado y verificación
1040  PRINT "CONSULTA DE UN FICHERO BIBLIOTECA"
1050  INPUT "entre el nombre del fichero  "; NOMF$
1060  NENR = 10
1070  OPEN "R", # 1, NOMF$, 64
1080  FIELD # 1, 16 AS AUT$, 48 AS TIT$
1090  INPUT "qué registro busca Vd.  "; NUM
1100     IF NUM > 0 AND NUM < = NENR THEN GOTO 1140
1110     PRINT "*** mal número – entre de nuevo" : GOTO 1090
1120  '
1130  '---- acceso al registro buscado y escritura
1140  GET # 1, NUM
1150  PRINT
1160  PRINT "libro número : "; NUM
1170  PRINT "autor        : "; AUT$
1180  PRINT "título       : "; TIT$
1190  '
1200  '---- fin
1210  CLOSE # 1
```

después de haber pedido el nombre del fichero (1020 a 1050), éste se abre en 1070 y sus registros son descritos por una instrucción FIELD (1080). Como anteriormente, hemos supuesto que el fichero contenía diez registros; este valor se ha situado en una variable llamada NENR (en 1060) a fin de facilitar una eventual modificación.

En 1090 se solicita del usuario que indique el número de libro a buscar y se verifica que se encuentre dentro de los límites permitidos (entre 1 y NENR); en caso contrario, se solicita que facilite un nuevo valor.

El acceso al registro en cuestión se efectúa por una instrucción GET (1140). A partir de este momento, el tampón asociado al fichero 1 está disponible y se pueden consultar cada una de sus zonas. Aquí, las instrucciones 1150 a 1180 escriben su contenido (precedido del número de libro).

Observación muy importante

Para situar valores en pseudo-variables como AUT$ y TIT$, es indispensable emplear LSET o RSET, y no es necesaria ninguna instrucción similar para consultarlas. Pueden aparecer en cualquier expresión, bien sea en una asignación o en una instrucción PRINT (es el caso de 1170 y 1180). Por el contrario, recordemos que no pueden aparecer a la izquierda de una instrucción LET o en una instrucción INPUT.

He aquí tres ejemplos de ejecución de este programa: hemos buscado los libros números 1, 5 y 8:

```
RUN
CONSULTA DE UN FICHERO BIBLIOTECA
entre el nombre del fichero ? BIBLI
qué registro busca Vd. ? 1

libro número : 1
autor        : HUGO
título       : los miserables
OK

RUN
CONSULTA DE UN FICHERO BIBLIOTECA
entre el nombre del fichero ? BIBLI
qué registro busca Vd. ? 5

libro número  : 5
autor         : DE CLOSETS
```

```
título          : la dicha por añadidura
OK

RUN
CONSULTA DE UN FICHERO BIBLIOTECA
entre el nombre del fichero ? BIBLI
qué registro busca Vd ? 8

libro número  : 8
autor         : SARRAZIN
título        : el astrágalo
OK
```

8.3.3. Programa de listado

Ahora es fácil de realizar un programa que confeccione el listado completo de las obras de nuestro fichero completo de las obras de nuestro fichero biblioteca:

```
1000  '**** LISTADO DE UN FICHERO BIBLIOTECA
1010  '
1020  '---- entrada nombre fichero, apertura, descrip. reg.
1030  PRINT "LISTADO DE UN FICHERO BIBLIOTECA"
1040  INPUT "entre el nombre del fichero "; NOMF$
1050  NENR = 10
1060  OPEN "R", # 1, NOMF$, 64
1070  FIELD # 1, 16 AS AUT$, 48 AS TIT$
1080  '
1090  '---- listado
1100  FOR I = 1 TO NENR
1110     GET # 1, I
1120     PRINT
1130     PRINT "libro número : "; I
1140     PRINT " autor       : "; AUT$
1150     PRINT "título       : "; TIT$
1160  NEXT I
1170  '
1180  '---- fin
1190  CLOSE # 1
```

El bucle FOR (1100 a 1160) permite leer sucesivamente cada uno de los registros del fichero.

Ejecutemos este programa sobre el fichero BIBLI, creado en el apartado 8.2.; obtendremos:

```
RUN
LISTADO DE UN FICHERO BIBLIOTECA
entre el nombre del fichero ? BIBLI

libro número  : 1
autor         : HUGO
título        : los miserables

libro número  : 2
autor         : ALAIN
título        : propósitos sobre la dicha

libro número  : 3
autor         : BAZIN
título        : la víspera

libro número  : 4
autor         : ROUSSEAU
título        : la astronomía

libro número  : 5
autor         : DE CLOSETS
título        : la dicha por añadidura

libro número  : 6
autor         : SYLVERE
título        : toinou

libro número  : 7
autor         : ROY
título        : Stendhal por él mismo

libro número  : 8
autor         : SARRAZIN
título        : el astrágalo
```

```
libro número  : 9
autor         : TROYAT
título        : la cabeza sobre las espaldas

libro número  : 10
autor         : MESSEGUE
título        : la naturaleza tiene razón

OK
```

Observación importante:

Hemos definido también el número de registros dentro del programa. Esta manera de proceder difiere de la que se ha utilizado con los ficheros secuenciales: en este caso, sin conocer necesariamente cuántos registros había en el fichero, le recorrimos hasta encontrar una marca de fin de fichero (EOF). Puede que Vd., con toda razón, pregunte por qué no hemos utilizado aquí esta técnica. Ello es debido a que en la mayoría de los Basic existe otra anomalía a saber:

- O la función EOF no está prevista por el acceso directo
- O es utilizable pero puede tomar el valor –1 cuando ha intentado leer uno o varios registros inexistentes.

(Esto es debido, en parte al desglose de cada pista de un diskete en sectores de longitud fija (128 ó 256 octetos); el Basic detectará no el último registro, sino solamente el último sector del fichero.

8.4. PRIMERAS CONCLUSIONES

Estos pequeños ejemplos, le muestran el interés que ofrece el acceso directo en la consulta de un registro de número dado. En este caso, el ahorro de tiempo obtenido en relación con la utilización de un fichero secuencial, es mayor cuanto mayor sea el número de registros.

Por el contrario, no existe diferencia significativa entre ambos tipos de acceso, para el listado completo de un fichero. Finalmente, si debe confeccionar un programa de búsqueda de un libro a partir de un nombre de autor o de un título, estará obligado a efectuar una búsqueda se-

cuencial. Más exactamente, sería preciso recorrer los registros en orden, a partir del primero, hasta encontrar el buscado. Esto es exactamente lo que haría con un fichero secuencial. En este caso, dirá Vd. que no ofrece ningún interés el utilizar el acceso directo. ¡Conclusión errónea y un poco apresurada! Cuando se estudie el acceso indexado podrá convencerse de lo contrario.

Capítulo 9

Un fichero biblioteca con más información

9.1. INTRODUCCION

En el capítulo anterior, hemos creado un fichero biblioteca donde cada registro contenía simplemente el nombre de autor y el título del libro correspondiente. Vamos a aprender aquí a confeccionar un fichero más completo, en el cual cada registro contenga además el año de adquisición y el número de páginas del manual correspondiente, así como su precio de compra.

Pensará, naturalmente, que para ello es suficiente el describir tres nuevas zonas en la instrucción FIELD. Ahora bien, el Basic tiene una restricción importante que reside en el hecho de que no se pueden definir más que cadenas de caracteres.

Así, por ejemplo, la instrucción:

```
FIELD #1, 10 AS C$, 4 AS NUM
```

será rechazada, puesto que NUM no es del tipo cadena.

9.2. ¿COMO TRABAJAR CON VALORES NUMERICOS

9.2.1. Manipulándoles en forma de cadena de caracteres

Vd. puede utilizar, siempre dentro de su programa, variables tipo cadena para encuadrar estos tres nuevos datos o informaciones que son: año, número de páginas y precio. Por ejemplo, si la descripción del nuevo registro es:

```
FIELD #1, 16-AS AUT$, 48 AS TIT$, 4 AS AÑ$, 5 AS NP$, 8 AS PR$
```

podrá, a tiempo de creación, entrar (desde teclado) las diferentes informaciones para:

```
INPUT "autor"; AUTOR$
INPUT "título"; TITULO$
INPUT "año"; AÑO$
INPUT "no pág."; NPAG$
INPUT "precio"; PREC$
```

y después situarlas en el tampón con:

```
LSET AUT$ = AUTOR$
LSET TIT$ = TITULO$
LSET AÑ$ = AÑO$
LSET NP$ = NPAG$
LSET PR$ = PREC$
```

Sin embargo, se imponen varios comentarios:

1. No importa la respuesta que pueda ser facilitada a INPUT para una cadena. Así, el Basic aceptará perfectamente que a:

```
INPUT "año"; AÑO$
```

responda con:

```
? AÑ 81
```

o:

```
? JUN 82
```

incluso:

```
? –1982
```

Dicho de otro modo, le será difícil entonces, el verificar que la respuesta es realmente numérica. Igualmente para PREC$,estas respuestas:

```
?564
?564P
?280.
?P 720
```

serían aceptadas.

2. No se puede efectuar ningún cálculo con estas informaciones (no se pueden multiplicar o sumar dos cadenas de caracteres). Entonces ¿cómo obtener, por ejemplo, el coste total de los libros adquiridos en 1982? Incluso las comparaciones, que sí son posibles entre cadenas, se hacen aquí laboriosas. ¡Imagine simplemente que "123.45" < "21.9"! Bajo estas condiciones, sería igualmente imposible seleccionar las adquisiciones realizadas entre 1979 y 1981 o incluso las de menos de 100 páginas.

En definitiva, ya ve Vd. que aunque las informaciones se escriban en el fichero en forma de cadenas, debemos poder manipularlas en forma numérica en el programa o programas que trabajen con este fichero. Aparece la necesidad de las conversiones.

9.2.2. Primera manera de convertirlos

Ciertamente Vd. pensará entonces en trabajar con variables numéricas que convertirá en cadenas de caracteres por la función STR$ antes de situarlas en el tampón. La conversión recíproca se realiza por la función VAL.

Ello es factible, en efecto. Necesitará, sin embargo, prever el tamaño máximo de cada una de las cadenas (ya que debe Vd. definir en FIELD el tamaño de cada campo).

Por ejemplo, si utiliza este desglose:

```
FIELD # 1, 16 AS NOM$, 48 AS TIT$, 5 AS AÑ$, 6 AS NP$, 9 AS PR$
```

sus informaciones durante la creación del fichero, pueden leerse como anteriormente en 9.2.1. El tampón ahora se llenará con:

```
LSET AUT$ = AUTOR$
LSET TIT$ = TITULO$
LSET AÑ$ = STR$ (AÑO)
LSET NP$ = STR$ (NUPAG)
LSET PR$ = STR$ (PRECIO)
```

No olvide que la conversión por STR$ reserva siempre un lugar para el signo. Por ello, en nuestra instrucción FIELD, hemos previsto para AÑ$, NP$ y PR$, un carácter más que en el ejemplo del apartado anterior.

Recíprocamente, cuando se utiliza el fichero, después de haber accedido a un registro (por GET), puede obtener los valores numéricos correspondientes con:

```
AÑO = VAL (AÑ$)
NUPAG = VAL (NP$)
PREC = VAL (PR$)
```

No es necesaria ninguna conversión para las cadenas AUT$ y TIT$ que están directamente disponibles, como en el capítulo anterior.

9.2.3. Otra manera "más económica"

No vamos a entrar en el detalle de la representación binaria de los números. Nos conformaremos con recordar que, generalmente, el Basic configura los números sobre 32 bits (4 octetos) bajo la forma llamada flotante, que permite expresar bien sea:

```
0.0000452 o bien 325.576 ó 34528000000
```

(existiendo, sin embargo, una limitación en la precisión).

En estas condiciones, comprenderá que la mayor parte de los Basic ofrecen la posibilidad de colocar un valor numérico cualquiera en 4 octetos (caracteres) del tampón. Sin embargo, los campos de este tam-

pón, definidos por una instrucción FIELD, no pueden ser más que del tipo cadena de caracteres. No es cuestión pues, de asignarles directamente un valor numérico. Así la instrucción:

```
LSET AÑ$ = 1984
```

entrañaría un error.

Es preciso, de hecho, utilizar una función especial MKS$ que produce un resultado de tipo cadena. Así:

```
LSET AÑ$ = MKS$ (1984)
```

sitúa el valor 1984, codificado en binario, en el campo AÑ$ del tampón.

Volvamos a nuestro fichero biblioteca. Sus registros podrían describirse por:

```
FIELD # 1, 16 AS NOM$, 48 AS TIT$, 4 AS AÑ$, 4 AS NP$, 4 AS PR$
```

(observe bien que los campos se mantienen siempre del tipo cadena, pero los destinados a recibir valores numéricos, son ahora de longitud 4).

Durante su creación, las informaciones pueden ser leídas siempre por:

```
INPUT "autor"; AUTOR$
INPUT "título"; TITULO$
INPUT "año"; AÑO$
INPUT "no pág"; NUPAG$
INPUT "precio"; PREC$
```

y el tampón ahora se llena como sigue:

```
LSET AUT$ = AUTOR$
LSET TIT$ = TITULO$
LSET AÑ$ = MKS$ (AÑO)
LSET NP$ = MK$ (NUPAG)
LSET PR$ = MKS$ (PRECIO)
```

Recíprocamente, además de la utilización de un fichero biblioteca, obtendrá los valores numéricos contenidos en un registro procediendo así (después de haberle leído con GET):

```
AÑO = CVS (AÑ$)
NUPAG = CVS (NP$)
PRECIO = CVS (PR$)
```

Observaciones:

1. Estas funciones MKS$ facilitan un resultado que ocupa (generalmente) menos espacio que el producido por STR$. Veremos un poco más tarde que ciertos valores numéricos pueden incluso ser ubicados en dos octetos (apareciendo como uno sólo).

2. La utilización de MKS$ puede parecer un poco artificial, en la medida en que, por ejemplo:

   ```
   LSET AÑ$ = MKS$ (AÑO)
   ```

   conduce a dos variables (AÑO y AÑ$) que contienen la misma cosa, (con la diferencia, sin embargo, de que una es considerada como numérica y la otra como una cadena). No obstante, en ciertos Basic, los números no se representan en memoria en forma binaria flotante mencionada antes: en este caso MKS$ (que produce siempre un resultado en binario flotante), realiza verdaderamente una conversión.

9.3. APLICACION DE UN FICHERO BIBLIOTECA

Según nos propusimos en la introducción, vamos a crear ahora y a utilizar un fichero biblioteca en el que los registros contendrán, además de las informaciones de autor y título, las de año de adquisición, el número de páginas y el precio. Para ello, utilizaremos la última técnica descrita (conversiones MKS$ y CVS).

9.3.1. Programa de creación

```
1000  '**** CREACION DE UN FICHERO BIBLIOTECA
1010  '
1020  '---- entrada nombre fichero, apertura, descrip. reg.
1030  PRINT "CREACION DE UN FICHERO BIBLIOTECA"
1040  INPUT "entre el nombre del fichero  "; NOMF$
1050  NENR = 10
1060  OPEN "R", # 1, NOMF$, 76
1070  FIELD # 1, 16 AS AUT$, 48 AS TIT$, 4 AS AÑ$, 4 AS NP$, 4 AS PR$
1080  '
1090  '---- creación del fichero
1100  FOR I = 1 TO NENR
1110  '--- entrada de información y verificación
1120  PRINT
1130  PRINT "libro número " ; I
1140  INPUT "autor              : "; AUTOR$
1150  INPUT "título             : "; TITULO$
1160  INPUT "año de compra   : "; AÑO
1170     IF INT (AÑO) <> AÑO THEN GOTO 1160
1180     IF AÑO <= 0 OR AÑO > 1983 THEN GOTO 1160
1190  INPUT "número de páginas : " ; NUPAG
1200     IF INT (NUPAG) <>  NUPAG THEN GOTO 1190
1210     IF NUPAG <=  0 OR NUPAG > 10000 THEN GOTO 1190
1220  INPUT "precio de compra    : " ; PREC
1230     IF PREC <= 0 THEN GOTO 1220
1240  '
1250  '--- preparación del tampón y escritura
1260  LSET AUT$ = AUTOR$
1270  LSET TIT$ = TITULO$
1280  LSET AÑ$ = MKS$ (AÑO)
1290  LSET NP$ = MKS$ (NUPAG)
1300  LSET PR$ = MKS$ (PREC)
1310  PUT # 1, I
1320  NEXT
1330  '
1340  '---- fin
1350  CLOSE # 1
```

Este programa es semejante al del capítulo anterior. La toma en consideración de tres nuevas informaciones, implica ahora registros de 76

octetos en lugar de 64 (instrucción 1060). La descripción se efectúa en 1070, por medio de una instrucción FIELD como expusimos en el apartado 9.2.3.

Durante la entrada de informaciones "numéricas", hemos tenido cuidado de verificar que eran verosímiles. Concretamente, en 1170 se comprueba que el valor aportado para el año es un valor entero y en 1180 se verifica que sea inferior o igual al año 1983, en este caso. Observe que si este programa va a ser usado durante algún tiempo, puede ser razonable el utilizar una variable (en lugar de una constante), que contenga el valor del año en curso. En 1200 y 1210 se realizan controles semejantes para el número de páginas. En cuanto al precio de compra, solamente se verifica que sea positivo (1230), sin imponer un límite máximo.

La preparación del tamón (1250 a 1300) le será ahora familiar. Hemos utilizado MKS$ para convertir nuestros tres valores numéricos en cadenas.

He aquí un "listing" parcial de ejecución de este programa:

```
RUN
CREACION DE UN FICHERO BIBLIOTECA
entre el nombre del fichero ? BIBLI

libro número 1
autor              : ? HUGO
título             : ? los miserables
año de compra      : ? 1976
número de páginas  : ? 854
precio compra      : ? 850

libro número 2
autor              : ? ALAIN
título             : ? propósitos sobre la dicha
año de compra      : ? 1978
número de páginas  : ? 235
        "
        "
        "
precio compra      : ? 115

libro número 10
autor              : ? MESSEGUE
```

```
título              : ? la naturaleza tiene razón
año de compra       : ? 1979
número de páginas   : ? 358
precio de compra    : ? 385
OK
```

9.3.2. Utilización del fichero

Como hicimos para nuestro fichero repertorio secuencial, podemos encontrar numerosas utilizaciones de nuestro fichero biblioteca. A fin de no hacer muy pesado nuestro ejemplo, hemos decidido presentarle un programa para un menú que contenga las tres siguientes funciones:

- Listado de un registro de número dado,
- Listado de todos los registros relativos a un autor dado,
- Listado de todos los registros correspondientes a un precio de compra situado en un entorno dado.

Otras funciones como:

- Listado de las adquisiciones efectuadas después de una fecha determinada.
- Listado de los registros cuyo número de páginas está en un entorno dado,
- etc...

podrían añadirse fácilmente, y su programación utilizaría los mismos principios que los de las tres funciones propuestas.

```
1000  '**** CONSULTA DE UN FICHERO BIBLIOTECA
1010  '
1020  ' ---- funciones utilizables:
1030  '          - listado de un registro de número dado
1040  '          - listado de los libros de un autor dado
1050  '          - listado de las adquisiciones de un precio
1060  '            situado en un entorno dado
1070  '
1080  '---- entrada nombre fichero, apertura, descr. reg.
1090  PRINT "CONSULTA DE UN FICHERO BIBLIOTECA
1100  INPUT "entre el nombre del fichero" ; NOMF$
1110  NENR = 10
```

```
1120  OPEN "R", #1, NOMF, 76
1130  FIELD # 1, 16 AS AUT$, 48 AS TIT$, 4 AS AÑ$, 4 AS NP$, 4 AS PR$
1140  '
1150  '---- repetición de selección de una función y ejecución
1160  '        hasta una petición de parada
1170  PRINT
1180  PRINT "seleccione una de las siguientes funciones  : "
1190  PRINT "LI listado de un registro de número dado"
1200  PRINT "LA selección por nombre de autor"
1210  PRINT "LP selección por precio"
1220  PRINT " FI fin"
1230  '
1240  INPUT R$
1250  IF R$ = "LI" THEN GOSUB 2000 : GOTO 1170
1260  IF R$ = "LA" THEN GOSUB 3000 : GOTO 1170
1270  IF R$ = "LP" THEN GOSUB 4000 : GOTO 1170
1280  IF R$ = "FI" THEN CLOSE # 1 : END
1290  PRINT "*** mala selección – entre de nuevo" : GOTO 1240
1300  '
2000  '*****************************************************
2010  '*** RUTINA DE LISTADO DE LIBROS DE NUMERO DADO ***
2020  '*****************************************************
2030  '
2040  PRINT
2050  INPUT " número del libro buscado" ; NUM
2060     IF NUM <= 0 OR NUM > NENR THEN GOTO 2050
2070  GET # 1, NUM
2080  GOSUB 8000                    ' lista registro
2090  RETURN
2100  '
3000  '*************************************************
3010  '*** RUTINA DE SELECCION POR NOMBRE DE AUTOR ***
3020  '*************************************************
3030  '
3040  INPUT " autor escogido "; AUTOR$
3050  ' ---- se completa la cadena AUTOR$ por blancos
3060  AUTO$ = AUTOR$ + SPACE$ (16 – LEN (AUTOR$) )
3070  FOR NUM = 1 TO NENR
3080     GET # 1, NUM
3090     IF AUT$ = AUTOR$ THEN GOSUB 8000
```

```
3100  NEXT NUM
3110  RETURN
3120  '
4000  '*************************************
4010  '*** RUTINA DE SELECCION POR PRECIO ***
4020  '*************************************
4030  '
4040  INPUT "precio mínimo : " ; PRMIN
4050     IF PRMIN < 0 THEN GOTO 4040
4060  INPUT "precio máximo : " ; PRMAX
4070     IF PRMAX < 0 THEN GOTO 4060
4080     IF PRMAX < PRMIN THEN GOTO 4040
4090  '
4100  FOR NUM = 1 TO NENR
4110     GET # 1, NUM
4120     PREC = CVS (PR$)
4130  IF PREC >= PRMIN AND PREC <= PRMAX THEN GOSUB 8000
4140  NEXT NUM
4150  RETURN
4160  '
8000  '*****************************************
8010  '*** RUTINA DE LISTADO DE UN REGISTRO ***
8020  '*****************************************
8030  '
8040  PRINT
8050  PRINT "Libro número  : " ; NUM
8060  PRINT "autor         : " ; AUT$
8070  PRINT "título        : " ; TIT$
8080  PRINT "año de compra : " ; CVS (AÑ$)
8090  PRINT "número de pag.: " ; CVS (NP$)
8100  PRINT "precio compra : " ; CVS (PR$)
8110  RETURN
```

La escritura de este programa es muy clásica.

a) "El programa principal". En él se encuentra:

* La selección del nombre del fichero, su apertura y la descripción del registro (1080 a 1130). Observe la instrucción 1110 que ubica el número de registros en NENR. Esto es indispensable, teniendo en cuenta que no existe una verdadera función EOF en el acceso directo.

* Un bucle (1150 a 1300) en el cual:

– Se solicita del usuario que seleccione una de las tres funciones referidas (LI, LA, LP) o una cuarta correspondiente a la parada del programa (FI).

– Se ejecuta la función seleccionada mediante la llamada a una rutina (2000, 3000 ó 4000).

b) "Rutina de listado de un registro"

Una vez solicitado el número de libro (2050), se verifica que es compatible con el número de registros del fichero (2060). Se accede entonces al registro (2070) y se pide a una rutina (situada en 8000) que confeccione el listado.

Observe que las tres funciones (LI, LA y LP) necesitan listar un registro. Por eso, es razonable utilizar una rutina para realizar esta operación común.

c) "Rutina de selección por nombre de autor"

En 3040, se pide entrar en la variable AUTOR$, el nombre elegido. Para seleccionar los libros en cuestión, se podría pensar que es suficiente entonces el comparar el campo AUT$ de cada uno de los registros con esta variable AUTOR$. En realidad, es preciso recordarlo, dos cadenas son iguales si (y solamente si) :

– Son de la misma longitud

– Están formadas por el mismo conjunto de caracteres

Ahora bien, AUT$ es siempre de longitud 16. En efecto, cuando al crear su fichero, escogió, por ejemplo, el nombre HUGO, éste se completó con 12 blancos (situados a su derecha si empleó (LSET).

Por el contrario, si Vd. respondió con este mismo nombre (HUGO) en la instrucción 3040, AUTOR$ contiene ahora una cadena de longitud 4.

Para hacer posible la comparación, se utilizan varias soluciones.

Aquí, completamos AUTOR$ con caracteres "espacio".

Más exactamente, añadimos (en 3060): 16 – LEN (AUTOR$) ya que LEN (AUTOR$) es la longitud de la cadena contenida en AUTOR$. Si

se toman estas precauciones, la selección ahora se hace más fácil. Es suficiente leer todo el fichero (3070 a 3100) llamando a la rutina de listado (8000) cada vez que un registro contenga el nombre de autor elegido.

Observaciones:

1. Desde la constitución del fichero, será preciso ser prudente en la forma de escribir los nombres de autor. Concretamente:

```
HUGO
HUGO V
HUGO VICTOR
VICTOR HUGO
```

quede bien entendido, son cadenas diferentes.

2. Ciertos Basic corren el riesgo de detectar un error en la 3060 cuando LEN (AUTOR$) sea > ó = a 16 (siendo el argumento de la función SPAG$ < ó = que cero).

d) "Rutina de selección por precio"

En 4040 a 4080, se solicita entrar el entorno de precio escogido y asegurarse que los valores aportados son "verosímiles".

Se recorre el conjunto del fichero (4100 a 4140) y los registros para los cuales el precio se encuentra dentro del entorno fijado se listan. Observe que el campo PR$, que contiene el precio codificado sobre 4 octetos, no es utilizable directamente. Es necesario crear una variable numérica PREC, utilizando la función de conversión CVS. (Si la omite, escribiendo por ejemplo PR$ > ó = que PRMIN, el Basic le facilitará un mensaje del tipo: "type mismatch in").

e) "Rutina de listado de un registro"

Escribe simplemente cada uno de los campos, poniendo cuidado de convertir los que corresponden a valores numéricos.

f) "Ejemplo de utilización"

He aquí los resultados obtenidos en este programa aplicado al fichero biblioteca creado en el apartado 9.3.1.:

```
RUN
CONSULTA DE UN FICHERO BIBLIOTECA
entre el nombre del fichero ? BIBLI

seleccione una de las siguientes funciones:
        LI listado de un registro de número dado
        LA selección por nombre de autor
        LP selección por precio
        FI fin
? LI

número del libro buscado ? 8

libro número    : 8
autor           : SARRAZIN
título          : el astrágalo
año de compra   : 1980
número páginas  : 192
precio compra   : 365

seleccione una de las siguientes funciones :
        LI listado de un registro de número dado
        LA selección por nombre de autor
        LP selección por precio
        FI fin
? LP
precio mínimo   : ? 10
precio máximo   : ? 150

libro número    : 2
autor           : ALAIN
título          : propósitos de la dicha
año de compra   : 1978
número páginas  : 235
precio compra   : 125

libro número    : 9
autor           : TROYAT
título          : la cabeza sobre las espaldas
año de compra   : 1976
```

```
número páginas : 244
precio compra  : 115

selecciones una de las siguientes funciones :
        LI listado de un registro de número dado
        LA selección por nombre de autor
        LP selección por precio
        FI fin
? LA
autor escogido ? ALAIN

libro número   : 2
autor          : ALAIN
título         : propósitos de la dicha
año de compra  : 1978
número páginas : 235
precio compra  : 125

seleccione una de las siguientes funciones :
        LI listado de un registro de número dado
        LA selección por nombre de autor
        LP selección por precio
        FI fin
? FI
OK
```

9.4. LAS DIFERENTES CONVERSIONES

9.4.1.

Hasta aquí, hemos empleado MKS$ para convertir variables llamadas numéricas en cadenas, y CVS para la función inversa. Ahora bien, en realidad, la mayoría de los Basic permiten tres tipos de variables numéricas: entera, real de simple precisión y real de doble precisión). El empleo del término numérico (sólo) corresponde, en la realidad, al tipo real de simple precisión. Veamos el cuadro de conversiones numérica-cadena y cadena-numérica que existen en los "Basic extendidos".

Tipo numérico	Conversión de numérico a cadena	No de octetos de la cadena del tampón	Conversión de cadena a numérico
Entero	MKI$	2	CVI
Real simple precisión	MKS$	4	CVS
Real doble precisión	MKD$	8	CVD

Nota: MKI es abreviatura de Make from Integer
MKS es abreviatura de Make from Single
MKD es abreviatura de Make from Double

Igualmente:

CVI es abreviatura de ConVert to Integer
CVS es abreviatura de ConVert to Single
CVD es abreviatura de ConVert to Double

Recordemos que el tipo de una variable puede definirse "implícitamente" (instrucción DEF...) o "explícitamente poniendo a continuación de su nombre, un símbolo particular ($ para las cadenas, % para los enteros, !para las reales de simple precisión y # para las reales de doble precisión). En fin, en ausencia de toda especificación, el tipo se considera como real de simple precisión. Ello explica que hayamos podido abstenernos hasta aquí de hacer estas distinciones, utilizando simplemente MKS$ y CVS.

9.4.2. Aplicación para nuestro fichero biblioteca

Podemos utilizar variables enteras para el número de páginas y el año de adquisición. Esto sólo es posible cuando:

- los valores son efectivamente enteros,
- no sobrepasan el valor 32767 (valor máximo representable sobre dos octetos).

Por otro lado, si consideramos libros de "gran valor" (precio superior a 9999.99 Ptas., podemos utilizar una variable real de doble precisión

para situar el precio. Esto es, ciertamente, un poco ficticio, pues generalmente un libro no costará 12595.75 Pts., pero probablemente pueda costar 12500,- Ptas; dicho de otro modo, no tendremos ciertamente verdadera necesidad de una precisión superior a la simple. Debe Vd. considerar lo que sigue, como un ejemplo de aplicación de diferentes conversiones más que un caso real.

El registro se describirá por:

```
FIELD #1, 16 AS AUT$, 48 AS TIT$, 2 AS AÑ$, 2 AS NP$, 8 AS PR$
```

La creación de un registro de clase I se podrá hacer así:

```
PRINT "libro número"; I
INPUT "autor"; AUTOR$
INPUT "título"; TITULO$
INPUT "año"; AÑO%
INPUT "no. pag"; NUPAG%
INPUT "precio"; PREC#
LSET  "AUT$ = AUTOR$
LSET  "TIT$" = TITULOS
LSET  "AÑ$ = MKI$ (AÑO%)
LSET  "NP$" = MKI$ (NUPAG%)
LSET  "PR$ = MKD$ (PREC#)
PUT   #1, I
```

Del mismo modo, el listado de un registro se hará con:

```
GET     #1, I
PRINT   "libro número      "; I
PRINT   "autor             "; AUT$
PRINT   "título            "; TIT$
PRINT   "año de compra     "; CVI (AÑ$)
PRINT   "no. pag           "; CVI (NP$)
PRINT   "precio compra     "; CVI (PR$)
```

9.4.3. Interés de las distintas conversiones

Para muchas aplicaciones, Vd. se puede conformar con utilizar únicamente el tipo numérico estándar. En algunos casos, se hará necesario el pasar a otros tipos:

a) Cuando debe conservar valores con precisión superior a la obtenida de manera estándar (generalmente el equivalente de 6 cifras significativas).

b) Cuando desee "optimizar" el tamaño de sus ficheros; este problema no surgirá realmente más que cuando estos últimos sean voluminosos. En este caso, si sus registros contienen valores enteros, ahorrará dos octetos cada vez que utilice MKI$ en vez de MKS$. Observe que incluso es posible ubicar en un solo octeto, un valor entero, con tal de que esté contenido entre 0 y 255, gracias a la conversión CHR$ (la conversión inversa está asegurada por ASC).

10

Actualización de ficheros de acceso directo: primeros conceptos

10.1. INTRODUCCION

El estudio de los ficheros secuenciales nos han mostrado que en general su actualización implicaba la creación de un segundo fichero. Los programas correspondientes eran entonces costosos en tiempo de ejecución.

Por definición, los ficheros de acceso directo permiten, por el contrario, la modificación de un registro sin que sea necesario acceder a los otros (con el condicionante, sin embargo, de que se conozca el número de registro en cuestión).

En cuanto a la adición o incorporación de nuevos registros, se puede efectuar teóricamente de cualquier lugar. Es más, la creación de un fichero de acceso directo no es sino una sucesión de adiciones. Así, si Vd. ha creado en un primer paso un fichero de 100 libros numerados de 1 a 100, podrá añadir posteriormente el 101, luego el 102, etc... Además, nada le obliga a escribir sus registros de manera "consecutiva". Por ejemplo, comience por crear un fichero para los libros con números 1, 2, 3, 5, 6, 7, 11, 12, etc (por lo tanto, están ausentes los números 4, 8, 9 y 10).

Posteriormente, podrá crear el registro número 4 ó 9. Sin embargo, de esta forma, aparece una dificultad que no existía para los ficheros secuenciales ni para los "manuales": ¿Cómo saber si un registro de nú-

mero dado existe o no? Veremos que la solución de este problema nos dará también la forma de realizar las supresiones de registros. Por último, si todos los registros tienen asignados números consecutivos (por ejemplo: 100 libros numerados de 1 al 100), es entonces imposible insertar uno nuevo entre otros dos ¿Cómo vamos entonces a asignar un número de libro comprendido, por ejemplo, entre el 47 y el 48?

Vamos a aprender a resolver progresivamente estos problemas que comporta la actualización de ficheros de acceso directo. En este capítulo, abordaremos:

- La modificación de registros ya existentes,
- La adición de registros al final del fichero.

La actualización en general (con adiciones, supresiones), será tratada en el siguiente capítulo.

10.2. MODIFICACION DE REGISTROS

Esta necesidad surge cuando se desea:

- Efectuar correcciones (la "toma de información" suele estar raramente exenta de errores),
- Llevar en cuenta la evolución de cierta información. Esto no tiene razón de ser para nuestro fichero biblioteca ya que la información que en él se ha almacenado, tiene un carácter prácticamente permanente. Por el contrario, sería indispensable, en el caso por ejemplo, del fichero "stock" de un librero. Este último estaría, con seguridad, relacionado con el anterior, pero en éste se encontraría información con cierta evolución, tal como: precio de venta, número de artículos en stock, etc...

En ambos casos, la función a realizar es la misma. Se trata simplemente de re-escribir un registro de número dado, después de haber modificado una parte de sus elementos. Existe una sola dificultad, aunque pequeña: es necesario leer el registro en cuestión, antes de efectuar las modificaciones requeridas.

He aquí un programa de corrección, bajo demanda, de nuestro fichero biblioteca:

```
1000  '**** MODIFICACION DE UN FICHERO BIBLIOTECA
1010  '
1020  '---- entrada nombre fichero, apertura, descrp. reg.
1030  PRINT "MODIFICACION DE UN FICHERO BIBLIOTECA
1040  INPUT "entre el nombre del fichero "; NOMF$
1050  NENR = 10
1060  OPEN "R", # 1, NOMF$, 76
1070  FIELD # 1, 16 AS AUT$, 48 AS TIT$, 4 AS AÑO$, 4 AS NP$, 4 AS PR$
1080  '
1090  '---- bucle para corrección de registro
1100  '       hasta que número = 0
1110  PRINT
1120  INPUT "num de registro – 0 para acabar "; NUM
1130     IF NUM = 0 THEN CLOSE # 1 : END
1140     IF NUM < 0 OR NUM > NENR THEN GOTO 1120
1150  GET # 1,  NUM
1160  PRINT "para cada campo, entre la nueva información"
1170  PRINT " o pulse return para no modificar nada"
1175  '
1180  PRINT "autor                : " ; AUT$;
1190  INPUT AUTOR$
1200     IF AUTOR$ <> " " THEN LSET AUT$ = AUTOR$
1210  '
1220  PRINT "título               : "; TIT$
1230  INPUT "                       "; TITULO$
1240     IF TITULO$ <> " " THEN LSET TIT$ = TITULO$
1250  '
1260  PRINT "año de compra :      : " ; CVS (AÑO$);
1270  INPUT AÑO
1280     IF AÑO > 0 THEN LSET AÑO$ =  MKS$ (AÑO)
1290  '
1300  PRINT "número páginas       : " ; CVS (NP$) ;
1310  INPUT NUPAG
1320     IF NUPAG > 0 THEN LSET NP$ = MKS$ (NUPAG)
1330  '
1340  PRINT "precio               : " ; CVS (PR$) ;
1350  INPUT PREC
1360     IF PREC > 0 THEN LSET PR$ = MKS$ (PREC)
1370  '
1380  PUT # 1, NUM
1390  GOTO 1110
```

Las instrucciones 1000 a 1070 ya le son muy familiares: la entrada del nombre del fichero, apertura y descripción del registro.

El resto del programa está constituido por la repetición de la corrección de un registro para el que el usuario aporta el número.

Cuando el número elegido es cero se para el programa. La modificación de un registro dado se desarrolla así:

- El registro es leído (en 1150).
- Cada uno de sus campos se presenta en pantalla al usuario al que se le pide entrar la nueva información cuando desea una modificación; en caso contrario, es suficiente con que teclee "return" (el valor leído por INPUT es entonces cero o cadena vacía, en el supuesto de que se trate de una variable numérica o una cadena).

  Cuando se requiere una modificación al campo en cuestión, se le atribuye un nuevo valor a través de la instrucción LSET y si llega el caso por una conversión MKS$.
- El registro así corregido, se re-escribe en el fichero (1380).

Vamos a utilizar este programa para efectuar algunas correcciones en nuestro fichero biblioteca:

```
RUN
MODIFICACION DE UN FICHERO BIBLIOTECA
entre el nombre del fichero ? BIBLI 1

número registro – 0 para acabar ? 5

para cada campo, entre la nueva información
   o pulse return para no modificar nada
autor              : DE CLOSETS    ?
título             : la dicha por añadidura
                     ?
año de compra      : 1980 ? 1979
número páginas     : 279 ? 282
precio             : 488 ?

número registro – 0 para acabar ? 5
para cada campo, entre la nueva información
   o pulse return para no modificar nada
autor              : SILVERE ?
```

```
título              : toinou
                      ? Toinou
año de compra       : 1981 ?
número páginas      : 396 ?
precio              ; 448 ? 445

número registro – 0 para acabar ? 0
OK
```

Hemos modificado el año de compra (1979 en lugar de 1980) y el número de páginas (282 en lugar de 279) del libro número 5, así como el título (Toinou en vez de toinou) y el precio 270 en vez de 265 del libro número 6.

10.3. AÑADIR UN REGISTRO AL FINAL DEL FICHERO

10.3.1. Introducción

Supongamos que se han comprado dos nuevos libros y deseamos añadir la información correspondiente en nuestro fichero biblioteca que actualmente se compone de 10 registros.

Esto no parece plantear ningún problema en particular, ya que nos es suficiente con escribir dos nuevos registros que tengan los números 11 y 12. Sin embargo, los programas que hemos realizado para consultar este fichero contienen todos una instrucción que indica el número de registros (NENR = 10). Si nuestro fichero se hace más grande, puede parecer difícil el tener que modificar esta instrucción en todos los programas que estén involucrados. Existirá entonces una probabilidad importante de error. Hubiera sido preferible que esta información (número de registros), constituyera una parte del propio fichero. (Observe, de pasada, que así funcionaba implícitamente para los ficheros secuenciales, gracias a la función EOF).

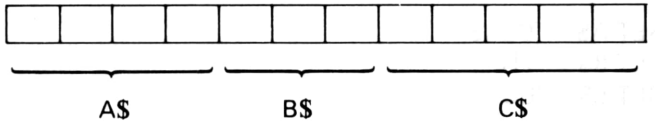
Ahora bien, en realidad, nada impide escribir este valor en uno de los registros, que se destinaría solamente para este fin (no servirá pues para describir un libro). Lo más sencillo es escoger el primero. Este registro tendrá el mismo tamaño que los demás (el acceso directo impone esta condición); por el contrario, su estructura será diferente. ¿Cómo traducir ésto en Basic? Vamos a introducir un nuevo concepto que consiste en utilizar varias instrucciones FIELD para un mismo fichero.

10.3.2. La técnica del "Field múltiple"

Si en un programa escribimos:

```
FIELD #1, 4 AS A$, 3 AS B$, 5 AS C$
```

significa que el tampón correspondiente al fichero número 1 se estructura así:

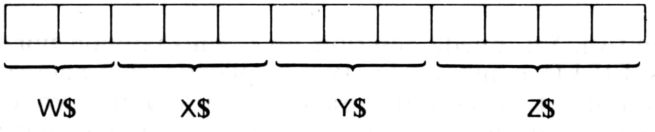


(cada casilla corresponde a un carácter).

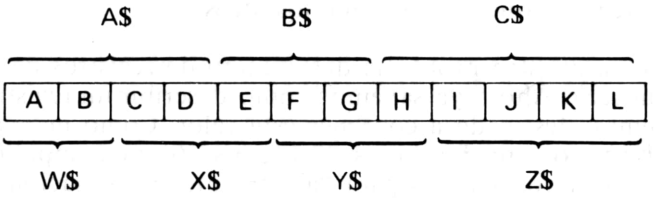
Si, en este mismo programa, escribimos igualmente:

```
FIELD #1, 2 AS W$, 3 AS X$, 3 AS Y$, 4 AS Z$
```

este mismo tampón, estaría estructurado así:



Así, hemos descrito el mismo tampón de dos formas diferentes. Supongamos que en un momento dado (por ejemplo, después de la lectura de un registro con GET), el tampón contiene:



Si está interesado en las pseudo-variables A$, B$, C$, W$, X$, Y$ y Z$, contendrán las siguientes cadenas:

```
A$: "ABCD"  B$: "EFG"  C$: "HIJKL"
W$: "AB"  X$: "CDE"  Y$: "FGH"  Z$: "IJKL"
```

REcíprocamente, para llenar el tampón, podrá utilizar indiferentemente el conjunto A$, B$, C$ o el conjunto W$, X$, Y$, Z$. Es decir, así:

```
LSET A$ = "CONS"
LSET B$ = "TIT"
LSET C$ = "UCION"
```

o bien:

```
LSET W$ = "CO"
LSET X$ = "NST"
LSET Y$ = "ITU"
LSET Z$ = "CION"
```

configuran un tampón que contendrá los 12 caracteres:

```
C O N S T I T U C I O N
```

En definitiva, Vd. puede describir un mismo tampón por tantas instrucciones FIELD como desee. Para explicarle el funcionamiento de este mecanismo, le hemos mostrado un mismo registro descrito de varias maneras. Sin embargo, en la práctica, el interés esencial de esta técnica reside sobre todo en la posibilidad de alojar en un mismo fichero, registros de diferente estructura.

Precisamente, es lo que vamos a hacer ahora.

10.3.3. Un registro que "no es como los otros"

Volvamos a nuestro problema del número de registros de nuestro fichero biblioteca. Ahora ya sabemos cómo describir un registro diferente a los demás, destinado a contener este valor. Como necesita un máximo de 4 octetos, quedan libres 72 (de los 76 que componen el registro). Podemos aprovecharlos para "alojar" otra información tal como:

– Fecha de creación del fichero,
– Fecha de la última actualización

Si escogemos escribir cada fecha en forma de 3 enteros (día, mes, año), la instrucción FIELD que describa este primer registro que llamaremos "cabecera" podría ser:

```
FIELD #1, 4 AS NE$, 2 AS DC$, 2 AS MC$, 2 AS AC$,
                  2 AS DM$, 2 AS MM$, 2 AS AM$, 60 AS X$
```

(donde el campo X$ serviría ficticiamente para completar a 76 caracteres la descripción del tampón).

10.3.4. Añadir registros a nuestro fichero biblioteca

Vamos pues a dotar a nuestro fichero de un registro "cabecera" a fin de realizar fácilmente la adición o añadido de registros al final del fichero.

Para ello, observe que no podemos utilizar ya el fichero BIBLI1 de los apartados y capítulos anteriores; el programa de creación del capítulo 8 debería ser por tanto modificado y adaptado.

Aquí, le proponemos un programa nuevo con menú, dándole a escoger entre las opciones de:

– Creación
– Incluir al final del fichero.

```
1000  '**** GESTION (CREACION, ADICION) DE UN FICHERO BIBLIOTECA
1010  '
1020  '----- entrada nombre fichero, apertura
1030  '       descripción registro cabecera
1040  '       y registro normal
1050  PRINT "GESTION DE UN FICHERO BIBLIOTECA
1060  INPUT "nombre fichero "; NOMF$
1070  OPEN "R", #1, NOMF$, 76
1080  FIELD #1, 4 AS NE$, 2 AS DC$, 2 AS MC$, 2 AS AC$,
                          2 AS DM$, 2 AS MM$, 2 AS AM$,
                          60 AS XX$
1090  FIELD #1, 16 AS AUT$, 48 AS TIT$, 4 AS AÑO$, 4 AS NP$, 4 AS PR$
1100  '
```

```
1110  '---- entrada fecha del día
1120  INPUT "entre la FECHA : día, mes, año  "; DIA, MES, AÑO
1130  '
1140  '---- menú propuesto
1150  PRINT "seleccione una de las funciones  :"
1160  PRINT "   CR creación"
1170  PRINT "  AJ adición"
1180  INPUT R$
1190  IF R$ = "CR" THEN GOSUB 2000 : GOTO 1230
1200  IF R$ = "AJ" THEN GOSUB 3000 : GOTO 1230
1210  PRINT "*** mala selección – entre de nuevo  " : GOTO 1180
1220  '
1230  '---- fin
1240  CLOSE # 1
1250  END
1260  '
2000  '****************************
2010  '*** RUTINA DE CREACION ***
2020  '****************************
2030  '
2040  '---- bucle para toma de información y escritura
2050  '         hasta petición de parada
2060  NUM = 1
2070     GOSUB 9000              ' toma de información
2080     IF AUTOR$ = "  " THEN GOTO 2120
2090     GOSUB 9500              ' escritura de un registro
2100  NUM = NUM + 1 : GOTO 2070
2110  '
2120  '---- preparación del registro de cabecera y escritura
2130  LSET NE$ = MKS$ (NUM)
2140  LSET DC$ = MKI$ (DIA)
2150  LSET MC$ = MKI$ (MES)
2160  LSET AC$ = MKI$ (AÑO)
2170  LSET DM$ = DC$
2180  LSET MM$ = MC$
2190  LSET AM$ = AC$
2200  PUT # 1, 1
2210  '
2220  RETURN
2230  '
```

```
3000  '*************************************
3010  '*** RUTINA DE ADICION DE REGISTROS ***
3020  '*************************************
3030  '
3040  '---- ACCESO AL REGISTRO CABECERA
3050  GET #1, 1
3060  NUM = CVS (NE$)
3070  DIAC = CVI (DC$)
3080  MESC = CVI (MC$)
3090  ÑAOC = CVI (AC$)
3100  DIAM =  CVI (DM$)
3110  MESM = CVI (MM$)
3120  AÑOM = CVI (AM$)
3130  PRINT "fecha de creación del fichero : " ; DIAC; MESC; AÑOC
3140  PRINT "fecha de últimos añadidos :  " ; DIAM, MESM, AÑOM
3150  '
3160  '---- bucle para toma de información y escritura
3170  '        hasta petición de parada
3180  GOSUB 9000                ' toma de información
3190  IF AUTOR$ = "  " THEN GOTO 3230
3200  GOSUB 9500                ' escritura de un registro
3190  NUM = NUM + 1 : GOTO 3180
3220  '
3230  '---- actualización de cabecera
3240  GET #1, 1
3245  LSET NE$ = MKS$ (NUM)
3250  LSET DM$ =  MKI$ (DIA)
3260  LSET MM$  = MKI$ (MES)
3270  LSET AM$ = MKI$ (AÑO)
3280  PUT #1, 1
3290  '
3300  RETURN
3310  '
9000  '*************************************
9010  '*** RUTINA DE TOMA DE INFORMACION ***
9020  '*************************************
9030  '
9040  PRINT
9050  PRINT "libro número "; NUM
9060  INPUT "autor              : "; AUTOR$
```

```
9070     IF AUTOR$ = "  " THEN RETURN
9080  INPUT "título                : " ; TITULO$
9090  INPUT "año de compra         : " ; AÑO
9100  INPUT "número páginas        : " ; NUPAG
9110  INPUT  " precio compra       : " ; PREC
9120  '
9130  RETURN
9140  '
9500  '*******************************************
9510  '*** RUTINA DE ESCRITURA DE UN REGISTRO ***
9520  '*******************************************
9530  '
9540  LSET AUT$ = AUTOR$
9550  LSET TIT$ =  TITULO$
9560  LSET AÑO$ = MKS$ (AÑO)
9570  LSET NP$ =  MKS$ (NUPAG)
9580  LSET PR$ = MKS$ (PREC)
9590  '
9600  PUT # 1, NUM + 1
9610  '
9620  RETURN
```

a) El programa principal

La primera parte (1000 a 1100), es ya clásica, con la excepción de la descripción del registro de cabecera (1080). Aquí, la instrucción FIELD correspondiente se ha escrito sobre varias líneas. Hemos utilizado, para ello, la tecla "line feed" (abreviado LF), que hace saltar a la línea siguiente de la pantalla sin que la instrucción Basic se considere terminada (para esto habría que pulsar "return"); algunos blancos suplementarios han permitido una disposición del texto que mejora su legibilidad. La mayor parte de los Basic permiten las instrucciones de 255 caracteres. Si éste no es su caso, puede sustituir la instrucción 1080 por:

```
1080 FIELD #1, 4 AS NE$, 2 AS DC$, 2 AS MC$, 2 AS AC$, 66 AS XX$
1082 FIELD #1, 10 AS YY$, 2 AS DM$, 2 AS MM$, 2 AS AM$, 60 AS ZZ$
```

(la segunda instrucción FIELD permitiría describir la parte no detallada en la primera).

Después de haber solicitado la fecha del día (1120), se propone al usuario escoger entre creación e inclusión; las funciones correspondientes se ejecutan por sendas rutinas.

b) La rutina de creación

* La parte esencial consiste en un bucle en el cual el número de libro NUM, inicializado a 1, aumenta en una unidad a cada pasada. Dentro, se encuentra la llamada a dos rutinas: una para la extracción de la información y otra para la escritura en el fichero. Se "sale" del bucle cuando el nombre de autor entrado desde el teclado es una cadena vacía (2080).

* Se prepara a continuación el registro de cabecera (2130-2190) situando en él la fecha del día, a la vez que como fecha de creación (DC$, MC$, AC$) y (arbitrariamente) como fecha de la última actualización (DM$, MM$, AM$). Observe la utilización de MKI$ (en lugar de MKS$) para las variables reales DIA, MES, AÑO. Podríamos, igualmente, haber utilizado las variables enteras DIA% , MES % , AÑO % . Finalmente el valor de NUM se sitúa en NE$. Esta variable indica el número total de registros, incluída la cabecera; en efecto, cuando se satisface la prueba de parada (2080), NUM contiene el número del próximo libro a incluir.

c) La rutina de adición o inclusión de registros

El acceso a la cabecera (3050), permite recoger las informaciones (3060 a 3120) y presentar al usuario las fechas de creación y de última actualización (3130-3140).

A continuación se encuentra un bucle de la toma de información y de escritura (3170-3210) análogo al encontrado en la rutina de creación; la única diferencia reside en la inicialización de la variable NUM con el número de registros contenidos en el fichero (3060). En efecto, si hay "n" registros, habrá "n-1" libros, por tanto, el próximo libro a entrar llevará el número "n".

Finalmente, la cabecera se actualiza (3240 a 3280). Para ello, es necesario reconstruir la información correspondiente de los siete campos que contiene; en efecto, han desaparecido del tampón por el que pasaron los registros añadidos. Aquí, hemos decidido re-leer la cabecera del fichero, modificar los campos afectados (número de registros, fecha de la última actualización) y re-escribirla.

d) La rutina de toma de información

No presenta dificultades en particular. Señalemos simplemente que, para mayor brevedad, no hemos puesto ninguna instrucción de verificación (como en el caso de los programas de los capítulos precedentes).

e) La rutina de escritura de un registro

Solamente una particularidad a destacar: como NUM contiene el número de libro que acaba de entrar (por el teclado), el número de registro correspondiente (contando la cabecera), será NUM + 1.

Observaciones:

1. Este programa no permite efectuar más que la creación del fichero o la adición de registros. Generalmente, necesitará de otras funciones (listado, correcciones, selecciones, etc...). Estas funciones deberán igualmente tener en cuenta la presencia del registro "cabecera". Más exactamente, los programas realizados en los capítulos anteriores, no podrán utilizarse con el fichero creado aquí, sin hacer algunas ligeras modificaciones.

2. Sensibilización sobre los problemas de seguridad de los ficheros.

Aquí, no hemos actualizado la cabecera hasta el final de la ejecución del programa. Esto significa que si, por desgracia, éste último ha sido interrumpido (instrucción errónea, corte de energía,...), las informaciones de la cabecera no corresponderán ya al contenido del fichero. Será necesario recomenzar totalmente la operación así interrumpida (creación de todo el fichero, inclusión de un conjunto de registros). Si esta "pérdida de tiempo" le parece excesiva, siempre es posible el prever la actualización de la cabecera (en el fichero), después de la escritura de cada registro .

Este problema de "seguridad" es difícil de resolver de forma general.

Cuando Vd. deba crear ficheros que presenten un interés real, le aconsejamos disponer sistemáticamente de al menos una "copia" que deberá "actualizar" de cuando en cuando. En efecto, incluso las precauciones draconianas dentro del programa, no le evitarán accidentes tales como:

– Escritura involuntaria sobre el diskette (después de una intervención desafortunada, por ejemplo por POKE).

– Destrucción física de un diskette (polvo, huellas depositadas en una zona donde existe un campo magnético importante, envejecimiento, incendio, etc...).

11

Actualización de los ficheros de acceso directo: caso general

11.1. INTRODUCCION

Hasta aquí hemos llenado nuestro fichero escribiendo sus registros según un orden natural. Dicho de otra forma, el segundo se escribía después del primero, el octavo después del séptimo, etc... Ahora bien, en realidad nada nos obliga a proceder así. La naturaleza misma del acceso directo permite crear registros sin importar el orden.

Volvamos, por ejemplo, a nuestro fichero biblioteca. Suponga que en un momento dado no dispone de entre los 10 libros que posee más que los que tienen de número el 1, 2, 5, 6 y 10 (estando el resto, por ejemplo, prestados). Vd. puede entonces comenzar a escribir en un fichero los registros de orden 1, 2, 5, 6 y 10 sin preocuparle los de orden 3, 4, 7, 8 y 9. Posteriormente, a medida que disponga de sus otros libros, podrá escribir los registros correspondientes.

Pero ¿qué ocurrirá si intenta acceder a un registro que aún no ha sido escrito, por ejemplo, el correspondiente al número 7?. Con seguridad, sobre un número tan pequeño de libros, este caso es poco probable, pero le podrá ocurrir con más frecuencia con un fichero de 100 ó de 500 libros. La respuesta a esta pregunta depende del sistema con el que trabaje. La diferencia entre ellos reside esencialmente en la forma en que el espacio del fichero se ha reservado en disco. Para mayor sencillez, digamos que, generalmente, el espacio necesario para un fichero se asigna a medida de las necesidades (esto significa que, puesto que siempre es

posible agrandar un fichero de acceso directo, las pistas asignadas no son necesariamente consecutivas. Sin embargo, esto es "tenido en cuenta" por el Basic y el usuario no debe preocuparse de ello).

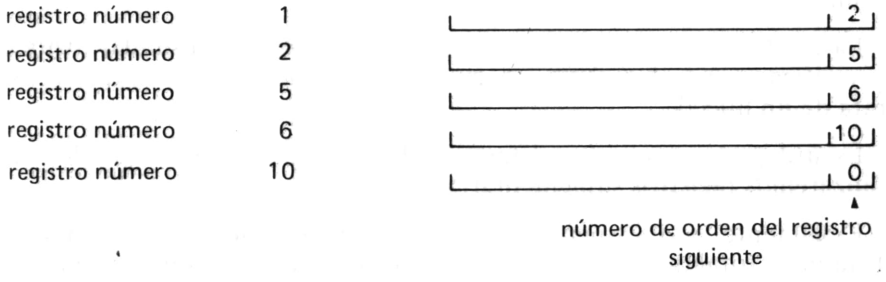
Así, cuando comience por escribir el registro 75, ciertos sistemas reservan al mismo tiempo, el espacio de los 74 precedentes; otros por el contrario, no reservan espacio más que para algunos (su número exacto depende de su tamaño). Si se lee entonces un registro inexistente, se obtendrán generalmente valores nulos (código ASCII : 0) en todos los octetos. Por el contrario, la lectura de un registro todavía no escrito, pero con su espacio reservado, conduce en principio que contendrán valores aleatorios.

Bajo estas condiciones, Vd. ve que es preferible disponer de un medio de conocer, con seguridad, los registros inexistentes (les llamaremos "huecos").

11.2. VARIAS MANERAS DE RECONOCER LOS HUECOS

Se pueden utilizar muchas soluciones. Por ejemplo, Vd. puede servirse de una tabla que le facilite los números de los huecos. Esta tabla puede ser ubicada en el fichero (utilizando una técnica de "FIELD múltiples").

Igualmente, puede encadenar los registros existentes: esto consiste en prever, en cada uno de ellos, un campo suplementario destinado a contener el número del registro siguiente. En nuestro ejemplo de fichero biblioteca (del apartado 1), se presentaría así:



Una tercera solución consiste en escribir, desde el comienzo, los registros "huecos" situando uno o varios valores que no puedan aparecer en

los registros verdaderos. Así, en nuestro fichero biblioteca, podríamos comenzar por crear 10 registros huecos, situando un valor nulo (código ASCII 0) en el primer octeto del campo AUT$.

Estos tres métodos permiten suprimir un registro existente y para ello es necesario sustituirle por uno hueco.

El último método descrito es, sin embargo, con mucho, el más sencillo. Los dos primeros pueden parecer delicados de utilizar si se desea conservar una gran seguridad frente a la destrucción de los ficheros; es preciso prever actualizaciones permanentes, bien sea de la tabla de huecos, sea de los encadenamientos, a cada incorporación de un nuevo registro. Su principal interés reside (en ciertos sistemas), en el ahorro de espacio obtenido en el caso de ficheros con un gran porcentaje de huecos. Aquí, no le presentaremos más que la programación para el tercer método.

11.3. PROGRAMA GENERAL DE ACTUALIZACION

El fichero inicialmente se rellena a "huecos" por una operación que llamaremos de "inicialización". Esto no es en realidad una creación ya que los registros así escritos no tiene significación precisa.

El usuario podrá sustituir a continuación algunos de estos huecos por registros verdaderos. Hablaremos aquí de inclusiones de registros ya que en realidad, no hay creación efectiva de nuevos registros. Antes de cada "inclusión" será preciso verificar que el registro en cuestión es realmente un hueco.

La modificación de registros será realizada de manera similar. Ahora, se verificará que el registro en cuestión es válido, es decir, que no se trata de un hueco.

Es útil poder aumentar el tamaño de nuestro fichero. Para ello, inicializaremos registros suplementarios.

Finalmente, es indispensable, como en el capítulo anterior, conocer el número total de registros (incluidos los huecos). Utilizaremos además un registro cabecera. Este deberá ser actualizado con seguridad por las distintas funciones. Veamos ahora en detalle el mecanismo de este programa.

11.3.1. Programa principal

```
1000  '**** GESTION DE UN FICHERO BIBLIOTECA
1010  '
1020  '---- Cada libro se localiza por un número que es
1030  '       el número de orden del reg. correspondiente - 1
1040  '
1050  '---- Funciones : - inicialización
1060  '                 - añadido de registros
1070  '                 - extensión del fichero
1080  '                 - supresión de registros
1090  '                 - corrección de registros existentes
1100  '
1110  '---- entrada nombre fichero - apertura
1120  '       descripción registro de cabecera y normal
1130  PRINT "GESTION DE UN FICHERO BIBLIOTECA"
1140  INPUT "nombre del fichero  "; NOMF$
1150  OPEN "R", #1, NOMF$, 76
1160  FIELD #1, 16 AS AUT$, 48 AS TIT$, 4 AS AÑO$, 4 AS NP$, 4 AS PR$
1170  FIELD #1,  4 AS NE$, 2 AS DC$, 2 AS MC$, 2 AS AC$,
                          2 AS DM$, 2 AS AM$, 2 AS MM$
                          60 AS XX$
1180  INPUT "entre la fecha : día, mes, año  "; DIA, MES, AÑO
1190  CABELE = 0
1200  '
1210  '---- repetición de la selección de una función y ejecución
1220  '       hasta petición de parada
1230  PRINT
1240  PRINT "seleccione una de las siguientes funciones : "
1250  PRINT " IN inicialización"
1260  PRINT " AJ añadido de registros"
1270  PRINT " EX extensión del fichero"
1280  PRINT " SE supresión de registros"
1290  PRINT "CO corrección de registros"
1300  PRINT "FI fin"
1310  '
1320  INPUT R$
1330  '
1340  IF R$ = "IN" THEN GOSUB 2000 : GOTO 1230
1350  IF R$ = "AJ" THEN GOSUB 3000 : GOTO 1230
```

```
1360  IF R$ = "EX" THEN GOSUB 4000 : GOTO 1230
1370  IF R$ = "SE" THEN GOSUB 5000 : GOTO 1230
1380  IF R$ = "CO" THEN GOSUB 6000 : GOTO 1230
1390  IF R$ <> "FI" THEN PRINT "*** mala respuesta" : GOTO 1320
1400  '
1410  '---- fin - escritura de cabecera si ha lugar
1420  IF CABELE = 1 THEN GOSUB 7500
1430  CLOSE # 1
1440  END
1450  '
```

Las instrucciones de 1130 a 1180 son análogas a las encontradas en el capítulo anterior.

La instrucción 1190 no es muy expresiva en sí misma. En realidad, es preciso ver que si el usuario escogió la función "inicialización", no se podrá leer ninguna cabecera ya que el fichero todavía no existe. En los demás casos será necesario hacerlo. Hemos decidido aquí, no escribir la cabecera hasta el final del programa y por tanto no leerla más que una sola vez. La variable CABELE sirve para indicar si la cabecera ha sido leída ya en el curso de la ejecución del programa (tomará el valor 1 cuando sea leída).

Las instrucciones 1210 a 1400 permiten al usuario encadenar varias funciones entre: inicialización, añadido, extensión, supresión y corrección.

Cada una de ellas se ejecuta por medio de una rutina.

El programa finaliza (1420, 1430) con el cierre del fichero precedido de la escritura de la cabecera. La comprobación sobre CABELE que condiciona esta escritura, puede parecer inútil. En realidad, es preciso prever el caso en que el usuario escoja inmediatamente la función FI. Escribir la cabecera, en este caso, le llevaría a situar cualquier valor en el fichero.

11.3.2. Rutina de inicialización

```
2000  '*********************************
2010  '*** RUTINA DE INICIALIZACION ***
2020  '*********************************
2030  '
2040  '---- confirmación de petición de creación
2050  '      entrada del número de registros deseado
```

```
2060  PRINT " si el fichero "; NOMF$ ; " existe ya, será destruido"
2070  PRINT "quiere inicializar (SI o NO)"
2080  INPUT R$ : IF R$ <> "SI" THEN RETURN
2090  INPUT "cuánto espacio quiere reservar "; NB
2100     IF NB < 0 THEN GOSUB 2090
2110  '
2120  '---- inicialización fichero (0 en primer octeto de cada reg.)
2130  NENR = NB + 1
2140  FOR I = 2 TO NENR
2150     LSET AUT$ = CHR$ (0)
2160     PUT # 1, I
2170  NEXT I
2180  '
2190  '---- preparación de la información de cabecera
2200  DIAC = DIA : MESC = MES : AÑOC = AÑO
2210  DIAM = DIA : MESC = MES : AÑOM = AÑO
2215  NENR = NB + 1
2220  '
2225  CABELE = 1
2230  RETURN
```

De la 2040 a 2100, se solicita al usuario que confirme su petición (esta función sirve para prevenir el riesgo de destrucción de un fichero ya existente). Se le solicita también que indique el número de registros deseados.

La inicialización propiamente dicha se efectúa en 2130 a 2170. Observe que se comienza a escribir a partir del segundo registro, reservando el primero para la cabecera. Aquí, se ha decidido situar simplemente un valor nulo en el primer octeto de cada registro. Es preciso observar que si el primer campo era numérico, será necesario asegurarse que ningún registro pueda contener este valor (por ejemplo, el mismo cero, escrito en forma entera, ocupa dos octetos que contienen este valor nulo (no es lo mismo para un valor significativo). Lo más prudente será entonces, o bien escoger otro valor absolutamente improbable o bien utilizar otro campo que tenga tipo cadena.

De 2200 a 2220, se actualizan las variables que contienen la información destinada a la cabecera, a saber:

NENR: número de registros del fichero

DIAC, MESC, AÑOC: fecha de creación del fichero (se asigna la fecha del día)

DIAM, MESM, AÑOM: fecha de la última actualización (se asigna convencionalmente, la misma fecha del día).

Es necesario indicar que esta actualización se realiza en las variables internas del programa. Efectivamente, no se puede llenar en este instante los diferentes campos de la cabecera (NE$, DC$, ...); su valor correría el riesgo de ser "machacado" por cualquier lectura de un registro del fichero. (No olvide que las dos funciones FIELD 1160 y 1170 corresponden a descripciones diferentes del mismo tampón).

Finalmente, el valor 1 se sitúa en la variable CABELE para señalar que la información relativa a la cabecera, existe efectivamente (aquí, no ha habido lectura previa de cabecera ya que el fichero no existía antes).

11.3.3. Rutina de incluir registros

```
3000  '*****************************************
3010  '*** RUTINA DE AÑADIDO DE REGISTROS ***
3020  '*****************************************
3030  '
3032  IF CABELE = 0 THEN GOSUB 7000 ' lectura de cabecera
3035  '
3040  '---- entrada el número de libro a añadir (0 para acabar)
3050  INPUT "número de libro a añadir (0 para acabar) "; NUM
3060     IF NUM = 0 THEN RETURN
3070     IF NUM < 1 OR NUM > NERN - 1 THEN GOTO 3050
3080  '
3090  '---- verificación de registros disponibles
3090  GET # 1, NUM + 1
3120     IF ASC (AUT$) = 0 THEN GOTO 3150
3130     PRINT "libro número "; NUM: " existe ya"; GOTO 3050
3140  '
3150  '---- toma de información del libro a crear desde teclado
3160  '         escritura en el registro de orden NUM + 1
3170  GOSUB 9000 : GOSUB 9500
3180  GOTO 3050
3190  '
```

```
9000  '****************************************
9010  '*** RUTINA DE TOMA DE INFORMACION ***
9020  '****************************************
9030  '
9040  PRINT
9050  PRINT "libro número "; NUM
9060  INPUT "autor                : "; AUTOR$
9070  INPUT "título               : "; TITULO$
9080  INPUT "año de compra        : "; AÑO
9090  INPUT "número de páginas: "; NUPAG
9100  INPUT "precio compra        : "; PREC
9110  '
9120  '  RETURN
9130  '
9500  '*********************************************
9510  '*** RUTINA DE ESCRITURA DE UN REGISTRO ***
9520  '*********************************************
9530  '
9540  LSET AUT$ = AUTOR$  : LSET TIT$ = TITULO$
9550  LSET AÑO$ = MKS$ (AÑO) : LSET NP$ =  MKS$ (NUPAG)
9560  LSET PR$ = MKS$ (PREC)
9570  '
9580  PUT # 1, NUM + 1
9590  '
9600  RETURN
OK
```

Primero se lee la cabecera, si no se ha hecho ya (3032). Las instrucciones siguientes (3040 a 3190) se repiten hasta que el usuario indique que no tiene más registros que añadir. Se pueden encontrar tres partes:

- Entrada del número de libro a incluir (3040-3070): se verifica que es compatible con el tamaño del fichero. Observe que el libro con número NUM corresponde, contando la cabecera, al registro de orden NUM + 1.

- Verificación de que el registro en cuestión es uno hueco (3090-3130): se asegura, para ello, que su primer octeto contenga el valor cero (ASC (AUT$), da el valor decimal del código ASCII relativo al primer carácter de la cadena AUT$).

– Toma de información correspondiente y escritura en el fichero: se utilizan dos rutinas clásicas que no necesitan más comentarios.

11.3.4. Rutina de extensión del fichero

```
4000  '*****************************************
4010  '*** RUTINA DE EXTENSION DEL FICHERO ***
4020  '*****************************************
4030  '
4040  IF CABELE = 0 THEN GOSUB 7000   ' lectura de cabecera
4050  '
4060  '---- entrada del número de espacio suplementario deseado
4070  PRINT "su fichero contiene ya "; NENR - 1; "libros"
4080  INPUT "cuántos más desea "; NEPLUS
4090     IF NEPLUS = 0 THEN RETURN
4100     IF NEPLUS < 0 THEN GOTO 4080
4110  '
4120  '---- inicialización de registros suplementarios
4130  N1 = NENR + 1 : N2 = N1 + NEPLUS - 1
4140  FOR I = N1 TO N2
4150     LSET AUT$ = CHR$ (0)
4160     PUT # 1, I
4170  NEXT I
4180  '
4190  '---- actualización del número total de registros
4200  NENR = N2
4210  '
4220  RETURN
4230  '
```

En 4040 tiene lugar la eventual lectura de la cabecera. A continuación se solicita al usuario que indique cuanto espacio suplementario desea, después de haber indicado el espacio que contenía ya el fichero (4060-4100). En esta situación, el usuario puede "salir" de la rutina indicando que quiere cero.

La inicialización de registros suplementarios (4120-4170) se efectúa como vimos anteriormente. Finalmente la variable NENR ve actualizado su valor en 4200.

11.3.5. Rutina de supresión de registros

```
5000  '*****************************************
5010  '*** RUTINA DE SUPRESION DE REGISTROS ***
5020  '*****************************************
5030  '
5035  IF CABELE = 0 THEN GOSUB 7000  ' lectura de cabecera
5040  '---- bucle para supresión hasta petición de parada
5050  '
5060  '---- entrada del número de libro a suprimir
5070  INPUT "número de libro a suprimir (0 para acabar) "; NUM
5080     IF NUM = 0 THEN RETURN
5090     IF NUM < 1 OR NUM > = NENR THEN GOTO 5070
5100  '
5120  '
5130  '---- lectura registro a suprimir – presentación en pantalla
5140  'confirmación de petición de supresión
5150  GET # 1, NUM + 1
5160     IF ASC (AUT$) = 0 THEN PRINT "** ya vacío": GOTO 5070
5170  PRINT "información sobre el libro número  "; NUM
5180  GOSUB 8000
5190  INPUT "quiere suprimirlo (SI o NO)"; R$
5200     IF R$ <> "SI" THEN GOTO 5070
5210  '
5220  ' ---- supresión
5230  LSET AUT$ =  CHR$ (0)
5240  PUT # 1, NUM + 1
5250  '
5260  GOTO 5070
5270  '

8000  '*************************************************
8010  '*** RUTINA DE PRESENTACION DE UN REGISTRO ***
8020  '*************************************************
8030  '
8040  PRINT " autor              :": AUT$
8050  PRINT "título              : " ; TIT$
8060  PRINT "año de compra       : " ; CVS (AÑO$)
8070  PRINT "número páginas      : " ; CVS (NP$)
8080  PRINT "precio compra       : " ; CVS (PR$)
```

```
8090  '
8100  RETURN
8110  '
```

Después de la eventual lectura de la cabecera (5035), las instrucciones siguientes se repiten hasta que el usuario indique que no tiene nada más que añadir. Se encuentran tres partes:

- Entrada del número de libro a suprimir (5060 a 5090): se verifica que es compatible con el tamaño del fichero.
- Lectura en el fichero, del registro a suprimir: se verifica que exista efectivamente (que no es un hueco). Después de presentar su contenido en pantalla por la rutina (8000), se solicita al usuario que confirme su petición de supresión.
- Si ha lugar, el registro se re-escribe con valor nulo en el primer octeto (5230-5240).

11.3.6. Rutina de lectura de cabecera

```
7000  '************************************************************
7010  '*** RUTINA DE LECTURA Y ACTUALIZACION DE CABECERA ***
7020  '************************************************************
7030  '
7040  GET # 1, I
7050  NENR = CVS (NE$)
7060  DIAC = CVI (DC$) : MESC = CVI (MC$) : AÑOC = CVI (AC$)
7070  DIAM = CVI (DM$) : MESM = CVI (MM$) : AÑOM = CVI (AM$)
7080  '
7090  CABELE = 1
7100  '
7110  PRINT "fecha de creación fichero  : "; DIAC, MESC, AÑOC
7120  PRINT "fecha de últimas modificac. : "; DIAM, MESM, AÑOM
7130  '
7140  DIAM = DIA : MESM = MES : AÑOM = AÑO
7150  '
7160  RETURN
7170  '
```

Después de la lectura, los valores del registro cabecera se ubican en las variables del programa: NENR, DIAC, ... Esto es necesario, ya que el contenido del tampón será machacado cuando se realice el próximo GET o cuando se prepare un nuevo registro. Estas variables se presentan en pantalla, a título de información.

11.3.7. Rutina de escritura de cabecera

```
7500 '*****************************************
7510 '*** RUTINA DE ESCRITURA DE CABECERA ***
7520 '*****************************************
7530 '
7540 LSET NE$ = MKS$ (NENR)
7550 LSET DC$ = MKI$ (DIAC) : LSET MC$ = MKI$ (MESC)
7560    LSET AC$ = MKI$ (AÑOC)
7500 LSET DM$ = MKI$ (DIAM): LSET MM$ = MKI$ (MESM)
7580    LSET AM$ = MKI$ (AÑOM)
7590 PUT #1, 1
7600 RETURN
7610 '
```

Consiste en situar los valores de fecha de las variables NENR, DIAC,... en el tampón y escribir su contenido en el registro de orden 1.

11.4. CUANDO EL NUMERO DE REGISTRO NO TIENE SENTIDO

Hasta ahora hemos supuesto que el número de registro tenía una significación para el usuario. Así, podíamos acceder directamente a un libro a partir de su número. Ahora bien, en la práctica, la situación no es siempre tan ideal. Pensemos, por ejemplo, en el comerciante que administra un stock de piezas. Con seguridad, cada una de ellas posee una "referencia", pero ésta es generalmente inutilizable como número de registro: en el mejor de los casos, contribuirá a crear un fichero desmesurado, lleno de huecos. Vd. podría entonces estar tentado de sacar la conclusión de que el acceso directo ofrece finalmente pocas ventajas en comparación con el secuencial. En realidad, existen varias posibilidades:

a) Encontrar una fórmula que permita establecer una correspondencia entre un número de pieza (o un nombre) y un número de orden en el fichero. Con seguridad, generalmente existirán unos huecos (registros que no corresponden a ninguna pieza) y existirán dobles (es decir, que varias piezas podrán corresponder al mismo registro). Será necesario entonces, prever el situar estos "dobles" en ubicaciones suplementarias. El arte consiste en encontrar una fórmula que ofrezca una tasa pequeña de huecos, suministrando un número razonable de dobles.
No detallaremos esta técnica aquí, conocida por el nombre de "hash-code".

b) Crear su fichero de acceso directo de modo secuencial: la información relativa a las distintas piezas, se ubica en registros consecutivos, sin preocuparse de la significación de su orden. En apariencia, este procedimiento no aporta ninguna ventaja; en efecto, la búsqueda de una pieza de número dado se debe efectuar de forma puramente secuencial.

Sin embargo, la técnica del "acceso indexado" permite "acelerar" esta búsqueda. Consiste en crear en memoria una tabla de correspondencia entre el nombre de pieza y el número de registro. Con seguridad, para construir la tabla, será necesario leer una primera vez todo el fichero. Por otra parte, es suficiente a continuación, efectuar una simple búsqueda en esta table (en memoria), para encontrar el registro correspondiente a una pieza dada. Si la tabla está además clasificada por orden alfabético, la búsqueda puede ser todavía más rápida.

En el Capítulo 13, volveremos sobre el acceso indexado. A partir de ahora, sin embargo, ve Vd. que los programas realizados en este capítulo no son todos utilizables directamente. Sin embargo, podrá aplicar siempre las técnicas que le hemos expuesto en este capítulo, a saber:

- Registro cabecera,
- Señalización de los registros inutilizados (huecos).

12

Las clasificaciones

12.1. INTRODUCCION

Cuando Vd. efectúa un listado de su fichero repertorio, puede que desee obtenerle por orden alfabético de nombres. Incluso, puede que desee un listado del fichero biblioteca donde los nombres de los autores aparezcan en orden alfabético.

Para obtener este resultado a partir de un fichero en el que los registros no están ordenados según el orden que le interesa, es necesario efectuar una operación denominada "clasificación".

Vamos a ver que se pueden utilizar varias técnicas, según que el fichero sea secuencial o de acceso directo. En general, todas utilizan una operación más sencilla que es la clasificación en memoria, es decir, la ordenación de los elementos de una tabla. Debido a ello, haremos algunas consideraciones al respecto.

12.2. CLASIFICACION DE UNA TABLA

Tengamos una tabla de cinco elementos, que contiene los valores numéricos siguientes:

T

15	8	25	10	7
T(1)	T(2)	T(3)	T(4)	T(5)

El objeto es obtener una tabla ordenada, es decir, que aparezca así:

	7	8	10	15	25
T	T(1)	T(2)	T(3)	T(4)	T(5)

Para ello, el método más fácil de programar sería el siguiente: se compara el primer elemento T(1) de la tabla (aquí el valor es 15), con el segundo T(2) (de valor 8). Si el segundo es menor que el primero (éste es el caso), se intercambian sus valores. Lo cual nos conduce a:

T	8	15	25	10	7

Se compara ahora T(1) (el valor ya se ha transformado), con T(3). Como $8 < 25$, no se efectúa ningún cambio. A continuación se compara T(1) con T(4) y luego con T(5). En este caso, como $8 > 7$ se intercambian los valores de T(1) y de T(5). Nuestra tabla se presentará así:

7	15	25	10	8

Así, después de haber efectuado esta serie de comparaciones entre el primer elemento y los demás, T(1) contiene el elemento más pequeño de la tabla.

Hemos pues progresado, ya que nuestro problema consiste ahora en ordenar la parte de T formada por los elementos T(2), T(3), T(4) y T(5).

Se aplica ahora la misma técnica con T(2) que se compara sucesivamente con cada uno de los elementos siguientes. Se ve que $T(2) < T(3)$ y después que $T(2) > T(4)$; se intercambian los valores de estos últimos, lo cual da:

7	10	25	15	8

Finalmente, como $T(2) < T(5)$, se intercambian de nuevo teniendo ahora:

7	8	25	15	10

Al concluir esta segunda serie de comparaciones, T(2) es el menor de los cuatro últimos elementos de la tabla. Por tanto T ha quedado clasificada hasta el elemento de orden 2.

Una tercera serie de comparaciones, esta vez sobre T (3), nos hace ver que T(3) > T(4), por tanto se efectúa un nuevo intercambio:

7	8	15	25	10

después T(3) > T(5), por tanto:

7	8	10	25	15

Finalmente, una cuarta serie de comparaciones sobre T(4) que no debe compararse más que con T(5), nos llevaría a un último intercambio:

7	8	10	15	25

Nuestra tabla ahora está clasificada.

He aquí un programa que realiza lo que acabamos de describir:

```
100  '**** CLASIFICAR NUMEROS
110  '
120  DATA 15, 8, 25, 10, 7
130  DIM T (5)
140  '
150  '--- lectura de números a clasificar en T y escritura
160  FOR I = 1 TO 5 : READ T (I) : NEXT I
170  PRINT "números a clasificar : " ;
180  FOR I = 1 TO 5 : PRINT T (I); : NEXT I
190  '
200  ' --- clasificación de la tabla T en ella misma
210  FOR I = 1 TO 4
220  '--- comparación de T (I) con todos los siguientes
230  FOR J = I + 1 TO 5
240     IF T (I) < T (J) THEN GOTO 270
250  '  --- intercambia T (I) y T (J)
```

```
260   X = T (I) : T (I) = T (J) : T (J) = X
270   NEXT J
280   NEXT I
290   '
300   '--- escritura de la tabla clasificada
310   PRINT
320   PRINT "tabla clasificada  : ";
330   FOR I = 1 TO 5 : PRINT T (I); NEXT I
```

la ejecución del programa será:

```
RUN
números a clasificar : 15 8 25 10 7
tabla clasificada    :  7 8 10 15 25

OK
```

Los valores de la tabla T se leen en una instrucción DATA y se imprimen. El bucle del contador I (210 a 280) permite considerar cada uno de los elementos T (I) de la tabla, con excepción del último. Para un valor dado de I, el bucle del contador J (230 a 270) realiza la comparación de T(I) con cada uno de los elementos siguientes: cuando T(I) > T(J), los valores de estos dos elementos se intercambian (260).

Observaciones

1) Si su Basic posee la instrucción SWAP, podrá sustituir ventajosamente las instrucciones de la línea 260 por:

```
SWAP T(I), T(J)
```

Si, como a menudo será el caso de los ficheros, Vd. debe clasificar tablas de cadenas, la instrucción SWAP le hará ganar tiempo de ejecución. En efecto, se limita a intercambiar las "direcciones" de las dos cadenas, sin "trasladar" realmente su contenido. El ahorro de tiempo obtenido en comparación con un programa de intercambio de valores (como en 260), será tanto más considerable, cuanto más largas sean las cadenas (puede incluso sobrepasar un factor 10).

2) La técnica de clasificación que le hemos mostrado es una de las más sencillas. Se denomina de "burbuja" (en inglés: bubble). Otras técnicas son más rápidas, pero también más complejas de comprender y de poner en funcionamiento (Shell, Shell/Metzner, Quicksort,...). Mientras sus ficheros no sean muy voluminosos o si no tiene problemas de consumo de tiempo de ejecución, Vd. puede limitarse a utilizar el método de la burbuja. Este es el método que utilizaremos en este capítulo.

Veamos ahora como clasificar un fichero. Para ello, distinguiremos el caso de un fichero secuencial y el de acceso directo.

12.3. CLASIFICACION DE UN FICHERO SECUENCIAL

12.3.1. Lectura del fichero

Supongamos que se desea obtener un listado clasificado por los nombres de su fichero repertorio. Si le interesan sólo los nombres, puede leer solamente el campo de "nombre" en memoria, de cada uno de los registros. Así construirá una tabla para la cual el método descrito anteriormente, le será suficiente.

No obstante, en general, deseará que en su listado aparezca toda o parte de la información asociada a cada nombre. Ahora bien, es difícil ir a buscar esta información al fichero, una vez clasificada la tabla de nombres. En efecto, se necesitaría una búsqueda secuencial para cada nombre (ya que después de la clasificación, su orden no tiene nada que ver ya con el orden de los registros); los tiempos de ejecución serían entonces desorbitantes.

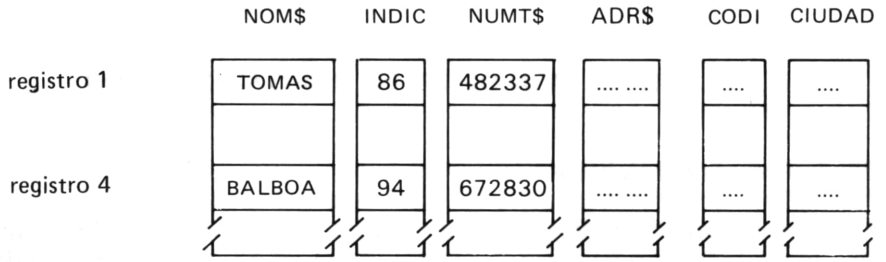
Es preciso pues, "aprovecharse" de la primera (y única) lectura del fichero para llevar a memoria toda la información que contiene (o, al menos, la que le interesa). Podría pensar en crear una única tabla donde cada elemento fuese un registro del fichero (leído por LINE INPUT). Sin embargo, en este caso, no podría acceder únicamente al nombre, elemento éste sobre el que Vd. desea efectuar la clasificación. Será necesario pues, el constituir tantas tablas como informaciones diferentes. En el caso de nuestro fichero repertorio, tendremos una tabla para cada una de estas informaciones: nombre, indicativo telefónico, número de teléfono, dirección, código postal y ciudad.

Como ve, la clasificación de ficheros secuenciales sólo será posible cuando sean poco voluminosos (ya que en definitiva, se recopiarán íntegramente en memoria).

12.3.2. La clasificación

a) El problema

Supongamos que hemos leído nuestro fichero repertorio, constituyendo las siguientes tablas:



Así, NOM$ (1) contiene el nombre del primer registro, INDIC (1) su indicativo y así sucesivamente.

Entonces, estamos tentados de aplicar la misma técnica de clasificación con la tabla NOM$. Así, en un momento dado, deberemos, por ejemplo, intercambiar el contenido de NOM$ (1) (TOMAS) con el de NOM$(4) (BALBOA). Pero entonces las informaciones contenidas en las otras tablas, no estarían ya de acuerdo con NUM$. Así, TOMAS tendría el número de teléfono de BALBOA (672830) y viceversa. Se hace pues necesario efectuar los intercambios de los elementos de orden 1 y 4 de todas las tablas.

Tal método nos conduciría a un resultado correcto. Sin embargo, es tedioso y consume mucho tiempo para realizar los intercambios de información.

b) Clasificación por índice

De hecho ¿es necesario clasificar todas estas tablas? No olvidemos que ese no es nuestro verdadero objetivo; en efecto, deseamos solamen-

te estar en condiciones de encontrar los diferentes registros siguiendo el orden alfabético de los nombres. Para ello, es suficiente saber donde está el primero (dicho de otro modo, cuál es su número de orden dentro de la tabla), donde está el segundo, el tercero, ... El método de "clasificación por índice" consiste en situar esta información del orden en una nueva tabla denominada índice. Antes de aplicarlo a nuestro fichero repertorio, vamos a explicar su funcionamiento sobre el simple ejemplo de una tabla de cinco números, ya descrita en el párrafo 12.2.:

T

15	8	25	10	7
T(1)	T(2)	T(3)	T(4)	T(5)

El problema de su clasificación se dará por resuelto cuando se pueda decir, por orden creciente que:

- El primer elemento está en la posición 5
- El segundo elemento está en la posición 2
- El tercer elemento está en la posición 4
- El cuarto elemento está en la posición 1
- El quinto elemento está en la posición 3

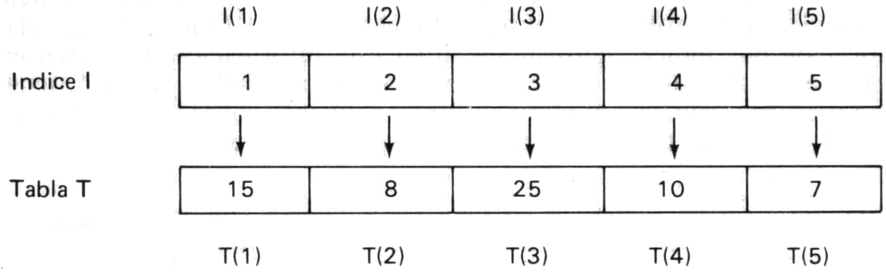
Esto se puede "resumir" en una tabla índice I que describa estas posiciones. Aquí I deberá contener:

5	2	4	1	3
↑ orden ler elemento	↑ orden 2º	↑ orden 3º	↑ orden 4º	↑ orden 5º

Así, I(1) dará el orden del primer elemento de T; I(3) el del tercer elemento, etc...

¿Cómo constituir este índice I? Podemos inspirarnos en la técnica descrita en el párrafo 12.2; pero allí donde los intercambios se efectuaban sobre dos elementos de la tabla T, aquí nosotros los efectuaremos

en la tabla índice. Al comienzo, situaremos los cinco valores 1, 2, 3, 4 y 5 en I:

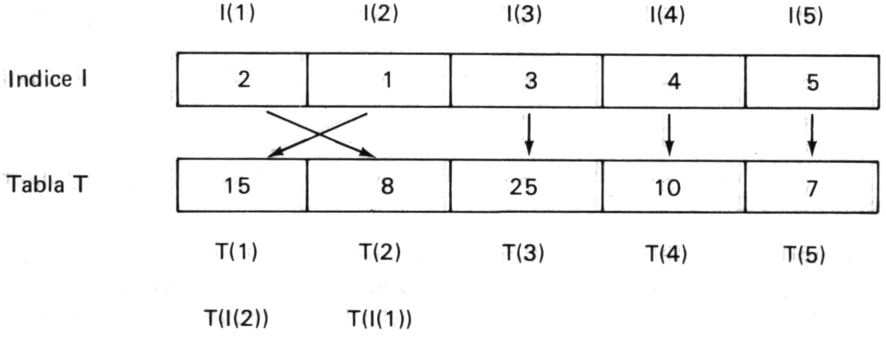


Podemos decir que cada elemento de I "apunta" hacia un elemento de T; bien entendido que, hasta el momento no se ha realizado ninguna clasificación; habríamos podido, por descontado, distribuir en forma distinta los cinco valores 1, 2, 3, 4 y 5 en I (sin embargo, es más sencillo proceder como lo hemos hecho).

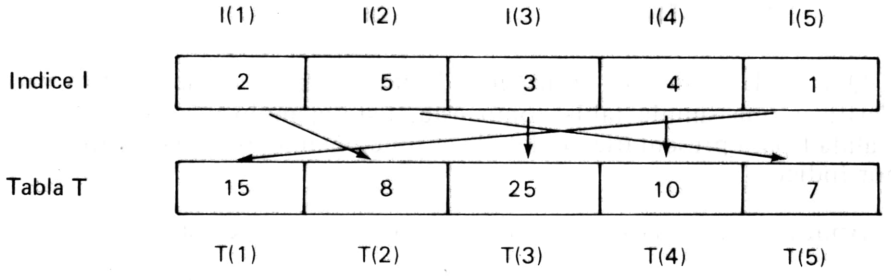
Veamos ahora el desarrollo de la clasificación propiamente dicho. Se comienza por efectuar una serie de comparaciones entre el elemento de T sobre el cual apunta el primer elemento I(1) del índice y los que son apuntados por I(2), I(3), I(4) e I(5). Dicho de otro modo, se compara T(I(1)) con T(I(2)), T(I(3)), etc... Desde la primera comparación encontramos que:

$$T(I(1)) > T(I(2))$$

Se intercambian entonces los CONTENIDOS CORRESPONDIENTES DEL INDICE I (y ninguno más de la tabla T), lo cual nos lleva a:



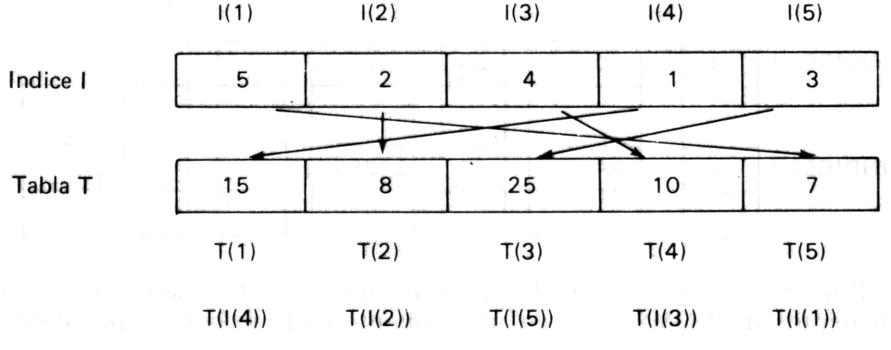
Se compara entonces T(I(1)) con T(I(3)), es decir T(2) con T(3), o dicho de otra forma, los valores 8 y 23, lo cual no implica ningún intercambio. Sucede lo mismo para la comparación de T(I(1)) con T(I(4)), es decir de T(2) con T(4), o sea entre 8 y 10. Por el contrario, la comparación de T(I(1)) con T(I(5)), es decir, entre T(2) y T(5), o lo que es lo mismo entre 8 y 7 implica intercambio de los contenidos de I(2) e I(5):



Al finalizar esta primera serie de comparaciones, el primer elemento del índice, es decir I(1), contiene el ORDEN del elemento más pequeño de T (observe que los valores de T quedan en su sitio).

Se procede de forma similar comparando T(I(2)) con T(I(3)), T(I(4)) con T(I(5)), invirtiendo si ha lugar los valores en I y así sucesivamente con T(I(3)) y finalmente con T(I(4)).

Al concluir estas cuatro series de comparaciones, tendremos:



Recorriendo "secuencialmente" el índice I, se encuentran los elementos de T, ordenados por orden creciente, a saber:

```
T(I(1)) = T(5) = 7
T(I(2)) = T(2) = 8
T(I(3)) = T(4) = 10
T(I(4)) = T(1) = 15
T(I(5)) = T(1) = 25
```

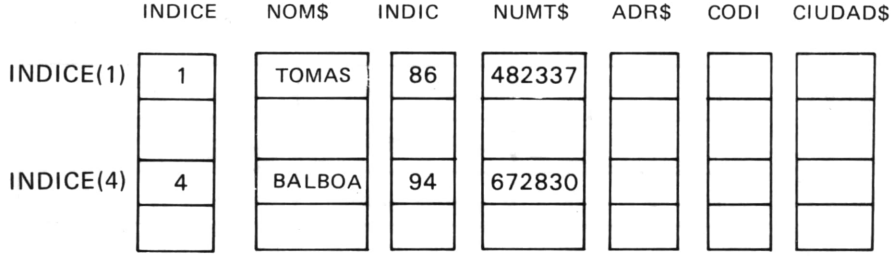
Quede claro que, la utilización de un "índice" no se justifica para clasificar una simple tabla como ésta. Este ejemplo nos ha servido en realidad para presentarle en detalle el funcionamiento de la clasificación por índice.

c) Aplicación a la clasificación de nuestro fichero repertorio

Aplicaremos el mismo método, pero esta vez, la tabla T queda sustituida por las tablas (NOM$, INDIC, ...) de las cuales hemos hablado en el párrafo a).

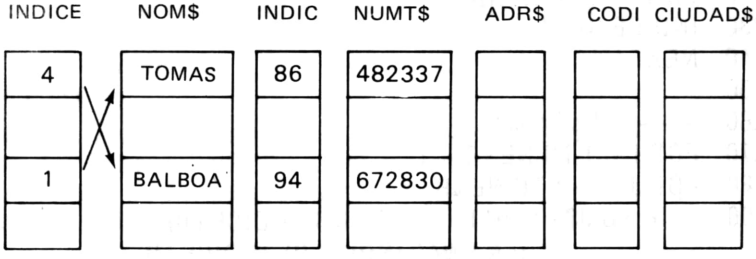
La tabla índice, que llamaremos INDICE, contendrá tantos elementos como registros haya en el fichero. Apuntará simultáneamente a cada una de las seis tablas ya que su contenido no cambiará en el curso de la clasificación (sucedía anteriormente lo mismo para T).

No vamos a describir la clasificación en detalle, pero vamos a dar una ilustración: para ello, volvamos a tomar el ejemplo del párrafo 12.3.2., añadiendo ahora la tabla INDICE:



Durante la clasificación, las comparaciones se efectuarán sobre los elementos de NOM$. Así, en un momento dado se comparará NOM$

(INDICE(1)) con NOM$ (INDICE(4)), es decir NOM$ (1) con NOM$ (4), dicho de otro modo, TOMAS y BALBOA. Existirá intercambio de SOLAMENTE LOS VALORES INDICE (1) e INDICE (4), lo que nos conducirá a la siguiente situación:



Programa de clasificación de nuestro fichero repertorio

Esta es la aplicación directa de lo que acabamos de ver:

```
1000  ' **** CLASIFICACION DE UN FICHERO REPERTORIO
1010  '
1020  '---- entrada nombre fichero – apertura
1030  PRINT "CLASIFICACION DE UN FICHERO REPERTORIO "
1040  INPUT "nombre del fichero"; NOMF$
1050  OPEN "I", # 1, NOMF$
1060  '
1070' '---- entrada número max. registros – reserva de tabla
1080  INPUT "entre el número máximo de registros "; NB
1090  DIM NOM$ (NB),  INDIC (NB), NUMT$ (NB), DIR$ (NB)
1100  DIM CODE (NB), CIUDAD $ (NB)
1110  DIM INDICE (NB)
1120  '
1130  ' ---- lectura del fichero
1140  K = 1
1150     INPUT # 1, NOM$ (K), INDIC (K), NUMT$ (K), DIR$ (K),
                    CODE (K), CIUDAD$ (K)
1160     IF EOF (1) = – 1 THEN GOTO 1220
1170  K = K + 1
```

```
1180  IF K < = NB THEN GOTO 1150
1190  PRINT "*** más registros de los previstos" : END
1200  '
1210  ' ---- inicialización de la tabla de índices
1220  NENR = K - 1
1220  FOR I = 1 TO NENR
1230  INDICE (I) = I
1240  NEXT I
1250  '
1260  '---- clasificación
1270  FOR I =  1 TO NENR - 1
1280  FOR J = I +1 TO NENR
1290     IF NOM$ (INDICE (I)) > NOM$ (INDICE (J))
                  THEN SWAP INDICE (I), INDICE (J)
1300     NEXT J
1310     NEXT I
1320  '
1330  '---- listado
1340  FOR I = 1 TO NENR
1350     K = INDICE (I)
1360     PRINT NOM$ (K); TAB (20); " ("; INDIC (K); ") "; NUMT$ (K)
1370     PRINT TAB (20); DIR$ (K); TAB (60); CODE (K); CIUDAD$ (K)
1380  NEXT I
```

De 1000 a 1060, se encuentra la entrada del nombre del fichero a clasificar y su apertura. En 1080, se solicita al usuario una estimación del número máximo de registros. Ello es necesario para reservar los emplazamientos de las tablas correspondientes (1090 a 1110); entre ellas, se encuentra la tabla INDICE cuyo significado ya se explicó anteriormente.

La lectura del fichero se realiza en 1130 a 1190. Un contador K inicializado a 1, se incrementa en uno a cada lectura de un nuevo registro y sirve de índice para las diferentes tablas sobre las cuales está contenida la información correspondiente. Estas instrucciones se repiten hasta encontrar el fin de fichero, a menos que las tablas se llenen antes; en este caso el programa se interrumpe.

En el caso "normal" cuando se encuentre la marca EOF, se sabe que el número de registros leídos está contenido en K.

Por razones de legibilidad del programa, se asigna este valor a una nueva variable NENR.

La tabla INDICE se inicializa con valores consecutivos (1222-1240). Después viene la clasificación propiamente dicha en 1260-1310. El elemento más importante es la instrucción 1290 donde se comparan los nombres (de NOM$) que tengan de orden INDICE (I) e INDICE (J); si ha lugar se intercambian los contenidos de INDICE(I) e INDICE(J). Finalmente, se realiza el listado (por orden alfabético de nombre de autor) en un bucle con contador I (1330-1380). La exploración de las diferentes tablas se efectúa en el orden indicado por el índice: para un valor dado del contador I, se listan todos los elementos de orden K = = INDICE (I).

Utilicemos este programa para obtener el listado clasificado de los registros del fichero REPERTA tal y como se presentaba al final del capítulo 7 (allí se llamaba TEMPO. Aquí le hemos dado de nuevo el nombre de REPERTA).

```
RUN
CLASIFICACION DE UN FICHERO REPERTORIO
nombre del fichero ? REPERTA
entre el número máximo de registros ? 12

BALBOA      ( 94) 413355
            c/ donoso cortés, 17        83600 FRAGA
COLON       ( 63) 274553
            c/ baleares, 56             81000 AVILA
GRIÑON      ( 1 ) 2481530
            c/ sierras, 41              75008 BARCELONA
LEDESMA     ( 75) 672830
            Av. betanzos, 17            77780 CHINCHON
LOPEZ       ( 84 ) 612842
            plaza mayor, 2              39400 MALAGA
MONTIEL     ( 1 ) 9332765
            c/ gaudí. 147               75008 BARCELONA
TABLADA     (77) 896313
            c/ hermosilla, 12           45510 MADRID
TOMAS       ( 86 ) 482337
            c/ granada, 2               58000 NERJA
VIDAL       ( 6 ) 6333947
            bul. las ruedas, 29         91310 MANRESA

OK
```

12.4. CLASIFICACION DE UN FICHERO DE ACCESO DIRECTO

12.4.1. Primer método para ficheros sin "huecos"

Supongamos que se desea clasificar un fichero biblioteca sin huecos (es decir, que existen todos sus registros). Por descontado, al igual que para los ficheros secuenciales, se puede leer el conjunto de registros en memoria, creando así tantas tablas como campos contenga cada registro. Sin embargo, es suficiente si leemos solamente los nombres de autor (si deseáramos una clasificación por este campo). Entonces, es posible efectuar una clasificación por índice. Por ejemplo, si nuestro fichero consta de 4 libros, crearemos inicialmente dos tablas, como sigue:

INDICE
1
2
3
4

AUTOR$
HUGO
ALBA
BAZAN
DEMETRIO

Se tiene una situación análoga a la encontrada en el caso de clasificación por índice de un fichero secuencial. Cada elemento de INDICE contiene el orden de un elemento de la tabla AUTOR$. Por otro lado, se ve que este número es también el del registro correspondiente al autor indicado. Esto se cumple, porque el fichero no tiene huecos. Sin embargo, si contiene cabecera, será preciso añadir 1 para obtener el número correcto de registro. Una clasificación por índice de AUTOR$ nos conduce entonces a la siguiente situación:

INDICE
2
3
4
1

AUTOR$
HUGO
ALBA
BAZAN
DEMETRIO

Está claro que la tabla AUTOR$ no ha cambiado. El listado clasificado del fichero, se realizará entonces leyendo los registros en el orden que indica el índice. Así, aquí, se leerá primeramente el registro de orden INDICE (1), es decir, el 2 (autor: ALBA), después el de orden INDICE (2) es decir, el 3, y así sucesivamente.

12.4.2. Cuando el fichero contiene huecos

Supongamos que nuestro fichero biblioteca se presenta ahora como sigue:

registro 1: cabecera
registro 2: HUGO.....
registro 3: "hueco"
registro 4: ALBA.....
registro 5: BAZAN
registro 6: "hueco"
registro 7: DEMETRIO

Se podría pensar en crear la tabla de autores de los registros existentes, con el índice correspondiente. Obtendríamos ésto:

INDICE
1
2
3
4

AUTOR$
HUGO
ALBA
BAZAN
DEMETRIO

Encontramos de nuevo lo mismo que en el ejemplo anterior. Pero si ahora es posible clasificar los autores, nada nos permite ya encontrar el registro correspondiente; en efecto, los valores situados en INDICE ya no son los órdenes de los registros relativos a los autores a los cuales "apuntan".

a) Primera solución

Podemos crear una tercera tabla llamada NUME que contenga los órdenes de los registros asociados a la tabla AUTOR$. Al comenzar, tendremos esto:

INDICE
1
2
3
4

NUME
2
4
5
7

AUTOR$
HUGO
ALBA
BAZAN
DEMETRIO

La clasificación se efectúa como anteriormente, por intercambio de los valores en INDICE; los contenidos de las tablas NUME y AUTOR$ quedan sin cambios. Obtenemos pues:

INDICE
2
3
4
1

NUME
2
4
5
7

AUTOR$
HUGO
ALBA
BAZAN
DEMETRIO

Observe que NUME no se utiliza para realizar la clasificación. El listado se obtiene, de nuevo, explorando "secuencialmente" la tabla índice. Por el contrario, en un momento dado, el orden del registro a leer no será ya (como anteriormente) INDICE(I) sino NUME (INDICE(I)). Así, en nuestro ejemplo, listaremos primero el registro de orden:

NUME (INDICE(1)), es decir, NUME(2) que es 4

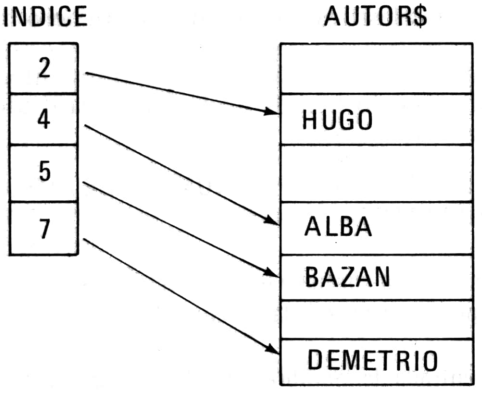
después el de orden:

NUME (INDICE(2)), es decir NUME(3) que es 5

y así sucesivamente.

b) Segunda solución

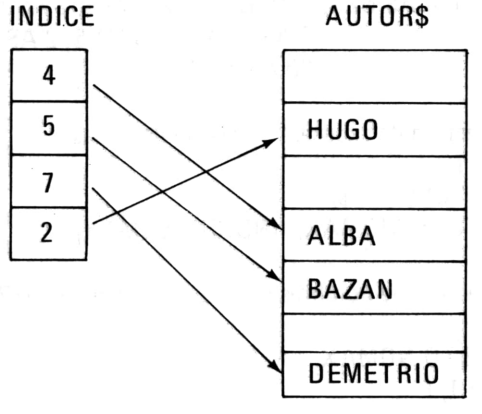
Podemos también crear una tabla AUTOR$ que contenga tantos elementos como registros haya en el fichero (huecos y cabecera incluídos). En cuanto a la tabla índice, situamos en ella los verdaderos órdenes de los registros existentes; sus elementos apuntan pues a la vez a AUTOR$ y al fichero. Así nos encontramos de nuevo en una situación análoga a la de clasificación de un fichero sin huecos. Al comienzo, tenemos esto:



Algunos elementos de la tabla AUTOR$ no contienen ningún valor; son los que corresponden a un hueco o a la cabecera.

La clasificación se efectúa mediante cambios en INDICE. .

Obtendríamos lo que sigue:



El listado se efectúa considerando los registros en el orden indicado por INDICE. Aquí, listaremos primero, el registro de orden o rango INDICE (1), es decir el 4, después el de orden INDICE(2), o sea el 5 y así sucesivamente.

Este segundo método es más costoso en memoria generalmente que el primero (salvo cuando el fichero contiene pocos huecos). Por el contrario, es más sencillo de comprender y de programar.

12.4.3. Programa de clasificación de nuestro fichero biblioteca

Utiliza el segundo método descrito anteriormente. Se aplica la fichero biblioteca tal como lo hemos hecho en el capítulo 11 (es decir, que contiene huecos y cabecera).

```
1000  ' **** CLASIFICACION POR NOMBRE DE AUTOR DE UN FICHERO
1005  '       BIBLIOTECA
1010  '
1020  '----- el fichero contiene una cabecera y ciertos registros
1030  '         pueden estar señalados como huecos
1040  '
1050  ' ----- entrada del nombre del fichero - apertura
1060  '         descripción registro normal y cabecera
1070  PRINT "CLASIFICACION POR NOMBRE DE AUTOR DE UN FICHERO"
1075  PRINT "BIBLIOTECA"
1080  INPUT "nombre del fichero : "; NOMF$
1090  OPEN "R", # 1, NOMF$, 76
1100  FIELD # 1, 16 AS AUT$, 48 AS TIT$, 4 AS AÑO$, 4 AS NP$, 4 AS PR$
1110  FIELD # 1, 4 AS NE$, 2 AS DC$, 2 AS MC$, 2 AS AC$,
                          2 AS DM$, 2 AS MM$, 2 AS AM$,
                          60 AS XX$
1120  '
1130  '----- lectura de cabecera - reserva de tablas
1140  GET # 1, I
1150  NOMAX = CVS (NE$)
1160  DIM AUTOR$ (NOMAX), INDICE (NOMAX)
1170  '
1180  '----- lectura de nombres de autor en tabla AUTOR$
1190  K = 0
1200  FOR I = 2 TO NOMAX
1210     GET # 1, I
```

```
1220     IF ASC (AUT$) = 0 THEN GOTO 1260
1230     K = K + 1
1240     AUTOR$ (I) = AUT$
1250     INDICE (K) = 1
1260   NEXT I
1270   '
1280   ' ---- clasificación de la tabla INDICE
1290   NENR = K
1300   FOR = 1 TO NENR - 1
1310   FOR J = I + 1 TO NENR
1320     IF AUTOR$ (INDICE(I)) > AUTOR$ (INDICE(J)) THEN
           SWAP INDICE (I), INDICE (J)
1330     NEXT J
1340     NEXT I
1350   '
1360   '---- listado del fichero clasificado
1370   FOR I = 1 TO NENR
1380     GET # 1, INDICE (I)
1390     PRINT
1400     PRINT AUT$; TAB (20); TIT$
1410     PRINT TAB (20); "año : "; CVS (AÑO$) ;
1420     PRINT TAB (35); "no páginas : " ; CVS (NP$);
1430     PRINT TAB (35); "precio : " ; CVS (PR$)
1440   NEXT I
```

Las instrucciones 1000 a 1110 no necesitan ya comentarios. En 1140 se lee la cabecera, que permite determinar el número total de registros (NOMAX), comprendidos los huecos y la cabecera. Se puede ahora reservar los espacios para las tablas AUTOR$ e INDICE (1160). De 1180 a 1260 se llenan las dos tablas. Como hemos visto, INDICE no debe contener más que los órdenes de los registros útiles. Esto justifica la utilización de un índice K, el cual aumenta en 1 con cada registro encontrado y no vacío. Por el contrario, cada nombre se ordena en AUTOR$ en un espacio en el que el orden I es el del registro correspondiente.

De 1280 a 1340, se clasifica INDICE. Finalmente, el listado se realiza de 1370 a 1440. Para ello, se leen los registros siguiendo el orden indicado por el índice.

Observación: Aquí hemos realizado una clasificación por nombre de autor. El mismo método se aplicaría para una clasificación por título, precio, año de adquisición, etc...

12.5. CONCLUSION

En este capítulo, no hemos ofrecido más que los conceptos básicos relativos a la clasificación de ficheros. Los numerosos métodos existentes y su estudio detallado podrían ser objeto de un manual. Por otro lado, no hemos examinado más que el caso de clasificación por un criterio (el nombre de autor en nuestro ejemplo). Es posible realizar una clasificación por dos o más criterios: por ejemplo, el nombre de autor y después el título (así los libros de un mismo autor aparecerían clasificados por orden alfabético de títulos).

Finalmente, no hemos realizado más que "listados clasificados". En ciertos casos, podría desear "clasificar un fichero", es decir, organizar sus registros en un cierto orden. (Verá que esto puede ser útil en el acceso indexado). Ciertamente, no le hemos propuesto programas en este sentido. Sin embargo, la adaptación de los programas anteriores es muy sencilla, ya que es suficiente sustituir las instrucciones PRINT por instrucciones de escritura sobre el fichero (PRINT # o PUT).

13

El acceso indexado

13.1. INTRODUCCION

En el estudio de los ficheros de acceso directo, hemos comentado en varias ocasiones, la dificultad que podía existir, en ciertos casos, para asignar una significación precisa a un número de registro. Incluso en nuestros simples ejemplos de ficheros "biblioteca", no es fácil conocer un libro por su número. Será, ciertamente, más cómodo el encontrarle, por ejemplo, a partir de un nombre de autor (mediante una selección eventual entre varios libros de un mismo autor). Para ello, no conocemos actualmente más que un solo método: examinar secuencialmente cada uno de los registros del fichero hasta encontrar el/los que nos interesan. Procediendo así, no haremos uso de las ventajas del acceso directo: un fichero secuencial permitía obtener los mismos resultados de manera sencilla y fácil.

De forma general, este problema surgirá cada vez que, habiendo creado un fichero de acceso directo, deseemos acceder a un registro a partir del conocimiento del contenido de uno de sus campos. El acceso indexado, para este caso, nos va a permitir encontrar el registro en cuestión más rápidamente que nos permitía el acceso secuencial (sin embargo, más lentamente que con el acceso directo).

Hablar de "el acceso indexado" hace suponer que se trata de un método único. La realidad es otra: en rigor, deberíamos hablar de uno o varios métodos de acceso indexado, entre otros. En efecto, a partir de un principio común que vamos a tratar de despejar, se permiten numerosas variantes; algunas pueden, por otra parte, ser relativamente com-

plejas. Su estudio detallado se sale del ámbito del estudio de este manual de iniciación. Vamos aquí, a conformarnos con "sensibilizarle" sobre las posibilidades que pueden ofrecer estas técnicas.

13.2. UN METODO SENCILLO: INDICE EN MEMORIA

Se inspira en el que vimos a propósito de la clasificación de un fichero de acceso directo. La idea básica consiste en crear, en un primer paso, una tabla índice (en memoria). En un segundo paso, la búsqueda de un registro se hace por consulta a esta tabla que facilita el orden.

13.2.1. Caso de un fichero "sin huecos"

Examinemos primeramente este método en el caso de un fichero biblioteca sin huecos, donde se desea realizar un acceso indexado sobre el nombre del autor.

La creación del índice es exactamente la misma que la preparación de la tabla INDICE destinada a la clasificación del fichero (capítulo 12). Por ejemplo, para un fichero de cuatro libros, esta tabla podría ser:

INDICE

HUGO
ALBA
BAZAN
DEMETRIO

Si se desea entonces, encontrar el libro del autor BAZAN, es suficiente recorrer secuencialmente la tabla INDICE hasta encontrarle. Aquí, sabemos que el registro buscado es el de orden 3 (ya que en INDICE (3) es donde se ha encontrado el nombre BAZAN). Si a continuación, deseamos encontrar el libro del autor DEMETRIO, una nueva búsqueda en INDICE nos proporcionará el orden buscado: 4.

13.2.2. Caso de un fichero sin huecos

Como para las clasificaciones, existen dos maneras (al menos) de proceder:

a) Crear una tabla índice sin huecos. Pero entonces el orden en el índice ya no corresponde al número de registro; es necesario crear una tabla suplementaria (NUME en el capítulo 12). La búsqueda de un registro de autor dado se realiza siempre por exploración secuencial del índice; si K es la posición sobre la cual se ha encontrado el autor en INDICE, el número de registros correspondiente es NUME(K).

b) Crear un índice sin huecos. La búsqueda de un registro de autor dado se realiza entonces en forma tan sencilla como en 2.1.

13.2.3. Primeras conclusiones

a) Como ve, este método implica una fase de preparación que es la creación del índice; éste necesita de una lectura completa del fichero. El método no ofrecerá por tanto un cierto interés (ahorro de tiempo), más que cuando se deban buscar a continuación varios registros. Aplicar esta técnica para la búsqueda de un sólo libro no tendría sentido: una exploración secuencial sería el método más sencillo y más rápido.

b) Decir "acceso indexado sobre un fichero biblioteca" no precisa nada. En efecto, en nuestro anterior ejemplo, podíamos encontrar libros a partir de un nombre de autor, pero no a partir del título. En rigor, deberíamos especificar que hemos realizado un acceso indexado SOBRE EL NOMBRE DE AUTOR, al igual que habíamos efectuado una clasificación sobre el nombre de autor en el capítulo 12. Por el contrario nada impide ciertamente, realizar un acceso indexado sobre el título: es suficiente simplemente con crear el índice apropiado durante la fase de preparación.

c) Caso de "dobles": hemos escogido un ejemplo donde no había más que un solo libro de cada autor. En el caso contrario, ve Vd. que sería preciso explorar todo el índice para encontrar los diferentes libros de un autor dado. Veremos más adelante que con un índice clasificado, este problema de "dobles" se trata más fácilmente.

d) Actualización: el hecho de utilizar el acceso indexado sobre un fichero no impone ninguna modificación de los programas de creación.

Quede entendido que los programas realizados anteriormente para la actualización, son utilizables si se sabe acceder a un registro por su número. En caso contrario, el acceso indexado le permitirá:

- La modificación de registros (que el fichero tenga huecos o no)
- La supresión de registros en el caso en que se ha previsto un índice para huecos; el índice facilita el orden del registro a suprimir, es decir, a sustituir por un hueco. Al mismo tiempo, el elemento correspondiente del índice se sustituye por el que representa (convencionalmente) un hueco: código ASCII 0, por ejemplo.
- La adición o inserción de registros: queda facilitada en el caso de un índice para huecos. Es suficiente buscar el primer hueco disponible (en el índice), para saber en qué orden escribir el registro a añadir. Observamos, sin embargo, una dificultad cuando no existen ya huecos disponibles; esta dificultad está ligada a la imposibilidad en Basic de aumentar la dimensión de una tabla. Para tratar convenientemente esta situación, ha sido necesario al comienzo, dar al índice una dimensión superior al número de registros del fichero.

13.2.4. Algunas mejoras posibles

a) Clasificación del índice

En el método anterior, la búsqueda era secuencial para el índice; esto puede ser lento para el caso de ficheros grandes. Ahora, es posible clasificar el índice (por orden alfabético), por ejemplo.

La búsqueda de un elemento podrá ser acelerada por una técnica denominada "de dicotomía". Consiste en comparar el elemento buscado (llamado CLAVE) con el elemento medio del índice. Según el resultado, se sabe entonces, estando clasificado el índice, si la clave buscada está situada en la primera o en la segunda mitad del índice. Se aplica de nuevo este procedimiento sobre la mitad seleccionada y así sucesivamente. A título indicativo, para un índice de 128 elementos, esta técnica permite encontrar la clave en un máximo de siete comparaciones; una búsqueda secuencial implicaría 64 de media.

Es preciso observar que, paralelamente a la clasificación del índice, es necesario constituir una tabla que facilite los órdenes de los registros en el fichero; en efecto, el número de un elemento del índice, después de la clasificación, ya no es el del orden del registro correspondiente. La situación es idéntica a la obtenida en la clasificación por índice (capítulo 12).

Por supuesto, si este método es interesante para la búsqueda o la modificación de un registro, es delicado de utilizar para la inserción o la supresión. Necesitaría entonces una reorganización completa del índice (con cada operación), para que quede siempre clasificado; se puede eventualmente prever sistemáticamente huecos para limitar el número de elementos a desplazar en caso de inserción de un registro.

b) Indice parcial

Cuando los elementos del índice son cadenas relativamente largas, se puede ahorrar espacio en memoria, conservando en el índice los primeros caracteres solamente. Ciertamente, esto favorecerá la aparición de dobles. Así, los nombres GALDOS y GALIANO aparecerán idénticos, en un índice de tres caracteres (GAL). Si el índice está clasificado, la búsqueda de una clave dada (GALDOS, por ejemplo), se efectúa ahora por una primera búsqueda de sus tres primeros caracteres (GAL) en el índice. A partir de lo cual se deduce que es suficiente examinar todos los registros que tengan GAL como clave: esto queda facilitado ya que están consecutivos en el índice (pero no en el fichero). De entre ellos no nos interesa más que los que tengan GALDOS como nombre de autor.

13.3. INDICE CONSERVADO EN EL FICHERO

Si se aprecia demasiado largo el tiempo de creación del índice al comienzo del programa, es posible conservarle en el fichero mismo: para ello se utilizará una técnica de FIELD múltiple.

Ahora, la fase preparatoria se resume a la lectura del o de los registros que contiene el índice.

Sin embargo, es preciso constatar que mientras se utilice este método para consultar el fichero, no existe ningún problema. No ocurre lo mismo para la actualización, ya que se trata de modificaciones, inserciones o supresiones. En efecto, si como en el ejemplo anterior se conforma con efectuar las modificaciones correspondientes del índice en memoria, se corre un riesgo: en caso de interrupción inesperada del programa, el índice ordenado en el fichero ya no estará de acuerdo con el contenido del propio fichero (el índice en memoria, lo estaba, pero se ha perdido). Se puede pensar en escribir el índice en el fichero, después de cada actualización. Este método es, sin embargo, caro en tiempo de ejecución y por otro lado, no es completamente infalible: es su-

ficiente que exista una interrupción entre el momento en que se ha modificado el registro y el que se debe escribir el índice. Cualquiera que sea la solución adoptada, es pues indispensable prever un programa de "reconstrucción del índice"; éste confecciona un nuevo índice, a partir del estado efectivo del fichero. Esto no es otra cosa que el equivalente de la fase preparatoria del acceso indexado con índice en memoria. Finalmente, además, el índice puede conservarse clasificado. Las dificultades ya mencionadas subsisten en la actualización; se hacen necesarias las reorganizaciones: parciales o totales según que el índice esté o no con huecos.

13.4. ¡EN DEFINITIVA!

El acceso indexado puede presentar por lo tanto numerosos aspectos. Algunos incluso, no se han citado aquí; mencionamos:

- El índice a varios niveles (comparable a la clasificación sobre varios criterios)
- El acceso secuencial indexado

Hemos querido mostrarle aquí solamente las posibilidades de esta técnica. Sin embargo, para evitar el que se "adentre", puede que inútilmente en programas sofisticados, pensamos que es bueno ofrecerle algunos consejos.

Para empezar, si sus ficheros de acceso directo evolucionan poco en el tiempo, puede crearlos de manera sencilla: por registro, si el número tiene un significado -de manera consecutiva en caso contrario. Para consultarles, podrá entonces:

- Crear un índice al comienzo del programa. Vd. le utilizará "tal cual" para una razonable cantidad de consultas. Podrá igualmente clasificarle y efectuar una búsqueda por dicotomía si el tiempo de acceso le parece excesivo.
- Conservar su índice sobre disco (o diskette) si tiene que consultar a menudo su fichero.

En caso (supuestamente poco frecuente) de actualización, podrá realizarlas directamente en el fichero, sin tener en cuenta el índice. Vd. deberá reconstruirle simplemente al final del programa, por medio de

una nueva lectura íntegra del fichero. Este método es ciertamente lento, pero tiene la ventaja de ser sencillo.

Si, por el contrario, sus ficheros evolucionan rápidamente, Vd. deberá prever la actualización del índice al mismo tiempo que el fichero (esté o no conservado en disco). Se trata de una solución difícil de realizar y muy sensible a las posibles interrupciones del programa. Deberá ser capaz de reconstruir el índice a partir del fichero.

De cualquier forma, le aconsejamos encarecidamente que no intente adentrarse en una programación demasiado complicada. Deberá, en principio familiarizarse progresivamente con estos diversos conceptos a fin de tener un cierto control sobre los programas que escriba.

Necesita saber, en efecto, que los programas donde intervienen apuntadores (como los de los elementos de la tabla NUME), son difíciles de poner a punto y de mantenerlos. Imagine las consecuencias de un error de una unidad sobre un índice de una tabla que apunta hacia los registros de un fichero.

Por otra parte, y como en la mayoría de los problemas de programación, podrá a veces obtener soluciones relativamente satisfactorias, pero no "óptimas" (en tiempo de ejecución, por ejemplo). Por el contrario, al desear presurosamente llegar a un punto óptimo, corre el riesgo de adentrarse en dificultades desmesuradas en comparación con la ganancia eventual conseguida. Para poner un ejemplo real, digamos que es frecuente realizar un programa "satisfactorio" en una jornada de trabajo y pasarse un mes intentando hacerle más eficaz. En este terreno, como en muchos otros, lo mejor es enemigo de lo bueno.

Anexo
Lista de programas

1. ACCESO SECUENCIAL

2. ACCESO DIRECTO

3. CLASIFICACIONES

Indice alfabético

Otros libros sobre lenguajes informáticos publicados por

PARANINFO S.A.

CHECROUN.– BASIC. Programación de microordenadores. 4ª edición. 112 páginas.

Para quienes carecen de conocimientos sobre ordenadores. Elementos que constituyen el sistema de información en torno a un microordenador. Importancia del lenguaje de programación y sus niveles.

GALAN PASCUAL.– Programación con el lenguaje COBOL. 3ª edición. 328 páginas.

El COBOL como herramienta de trabajo. Imprescindible para quienes han de introducirse en la programación de ordenadores aplicados a la gestión empresarial.

LARRECHE.– BASIC. Introducción a la programación. 4ª edición. 132 páginas.

Con numerosos ejercicios y ejemplos prácticos para aprender a escribir los propios programas y dar instrucciones al ordenador.

MONTEIL.– Primeros pasos en LOGO. 96 páginas.

El LOGO, es un lenguaje sencillo por sus caracteres, textos y símbolos. Con un Glosario de Términos de gran utilidad práctica.

De ROSSI.– BASIC. Curso acelerado. 4ª edición. 224 páginas.

Para aprender con rapidez este lenguaje. Más de 350 ejemplos y ejercicios para practicar el BASIC en cualquier ordenador.

SANCHIS LLORCA.– Programación con el lenguaje PASCAL. 5ª edición. 368 páginas.

Para que el lector pueda llegar a construir sus propios programas con calidad, buen estilo y fiabilidad. Numerosos ejercicios, ejemplos y programas completos.

WATT y MANGADA.– BASIC para niños. Con notas didácticas para padres y educadores. 4ª edición. 128 páginas.

Una obra eminentemente didáctica en la que los dibujos juegan un importante papel. Parte de conocimientos básicos sobre la utilización del ordenador y de conceptos elementales de programación BASIC. Un libro para los niños que deben conocer todos los padres y educadores.

WATT y MANGADA.– BASIC avanzado para niños. Con notas didácticas para padres y educadores. 160 páginas.

Dirigido a los pequeños que ya tienen conocimientos de BASIC para que vayan profundizando en el conocimiento y utilización de este lenguaje de programación.

WATT y MANGADA.– BASIC para niños con el microordenador Dragón. Con notas didácticas para padres y educadores. 128 páginas.

En el se analizan procesos y programas de forma muy amena. El niño aprenderá y jugará a la vez y, sin darse cuenta, sentará las bases para un estudio más profundo del tema que le permitirá llevar a cabo desarrollos más complejos.

De próxima aparición

BELLIDO.– Amaestra tu Dragón.

BELLIDO y SANCHEZ.– BASIC para maestros.

GARCIA MERAYO.– FORTRAN 77.

HART.– Diccionario en BASIC.

MARSHALL.– Lenguajes de programación para micros: BASIC, PASCAL, LIPS, COBOL, FORTH, COMAL, FORTRAN, PILOT, PROLOG y LOGO.

FICHEROS EN BASIC

Proporciona los medios necesarios para la creación y utilización de ficheros. A través de esta obra, descubrirá y conocerá un archivo o fichero informático y su diferencia con un archivo manual y se iniciará en la técnicas de los accesos secuencial, directo e indexado. El conocimiento de sus respectivas ventajas e inconvenientes le permitirá escoger el método que mejor se adapta a los requerimientos de su aplicación.

El sistema de exposición es progresivo, y los conceptos fundamentales se complementan con ejemplos concretos y sencillos, tales como una guía o repertorio telefónico y las fichas de los libros de una biblioteca.

Las instrucciones Basic tratadas son las más comúnmente utilizadas en la mayoría de los micro-ordenadores. Así, los programas propuestos se pueden adaptar y utilizar fácilmente sobre cualquier máquina.

EDITORIAL PARANINFO SA
Magallanes, 25 - 28015 Madrid

ISBN: 84-283-1355-5