sinclair

QL

Archive de QL

CAPITULO 1
QL ARCHIVE

QL Archive es un programa de base de datos, que le permitirá crear sistemas de ficheros para almacenar toda clase de información. Le ofrece plena libertad para decidir cómo desea almacenar y recuperar la información.

Descubrirá rápidamente cómo puede utilizar Archive para crear sistemas sencillos de índice de tarjetas, tales como listas de direcciones o registros de clientes. Después de familiarizarse con la creación de estos sistemas sencillos, podrá interesarle desarrollar sistemas más complejos de múltiples ficheros relacionados, donde se comparte la información, por ejemplo, entre registros de control de existencias y de compras.

Usted puede presentar la información en el formato de la pantalla provisto por Archive, o podrá diseñar su propio formato. Puede crear formularios e informes impresos en el formato que le parezca, a partir de la información en los ficheros.

Una de las características más poderosas de Archive es su estructura de comandos. Después de crear un fichero y almacenar registros en el mismo, podrá utilizar estos comandos para hallar ciertos registros, para buscar o seleccionar información y para presentar la información en el fichero en un orden determinado.

Los comandos se combinan para formar un lenguaje de programación poderoso, similar a SuperBASIC, que puede emplearse para crear muchas aplicaciones especiales.

El programa le guía continuamente mediante una serie de mensajes que nunca le dejan en duda acerca de las opciones que tiene a su disposición, o de lo que debe hacer. Si requiere más información podrá siempre obtenerla en los ficheros de Ayuda. Puede pedir Ayuda en cualquier momento, sea cual fuere lo que esté haciendo, y el programa siempre le provee la información más a propósito a sus necesidades actuales.

El poder real de Archive se pone de manifiesto cuando Vd. escribe sus propios procedimientos en el lenguaje de comandos. Puede crear un procedimiento nombrado para hacer exactamente lo que desea y luego emplearlo como un comando adicional, exactamente de la misma forma que utiliza los comandos normales de Archive.

Las operaciones de escribir y modificar un programa se simplifican con el *editor de procedimientos*, que junto con el *editor de la línea de entrada* (que está siempre disponible), facilitan enormemente las correcciones.

Los ficheros de datos tienen campos y registros de longitud variable. Esto no sólo conduce al aprovechamiento más eficaz de la memoria disponible y espacio del cartucho, sino que también simplifica la creación de los ficheros. Usted nunca precisa decidir de antemano lo grande que precisa ser un registro.

Este manual contiene varios ejemplos prácticos. Pruebe los ejemplos para ver la cantidad de cosas que puede hacer. Contienen muchos procedimientos de uso general que podrá Vd. incluir en sus propios programas.

Si, en cualquier momento, no está seguro de lo que debe hacer, recuerde que puede solicitar Ayuda pulsando **F1**. Podrá también cancelar una operación no completada, tal como escribir un número o utilizar un comando, pulsando **ESC**.

Archive ha sido diseñado para darle la mayor flexibilidad posible. Por consiguiente, no le ofrece tanta ayuda como los restantes programas QL a la hora de seleccionar las opciones. Si no está familiarizado con los computadores (también llamados ordenadores) y con su programación, le resultará aconsejable estudiar la Guía de SuperBASIC para Principiantes antes de intentar escribir programas con Archive.

CAPITULO 2
PARA COMENZAR

CARGA DE QL ARCHIVE

La carga de QL Archive se describe en la Introducción a los Programas QL. Después de cargar Archive, aparece lo siguiente en la pantalla:

```
CARGANDO QL ARCHIVE
Base de datos
Versión x.xx
Copyright ©1984 PSION Ltd
Reservados todos los derechos
```

donde x.xx representa el número de versión (por ej. 2.00).

El programa aguarda entonces unos segundos antes de comenzar.

La información de Ayuda no se carga en la memoria del computador junto con el programa. Solamente se lee del Cartucho Archive al precisarla. **Por tanto, no debe extraer el cartucho Archive del Microdrive 1 si cree que precisará la información de Ayuda.**

ASPECTO GENERAL

Al cargar Archive, la pantalla tiene el aspecto que se muestra en la Figura 2.1.

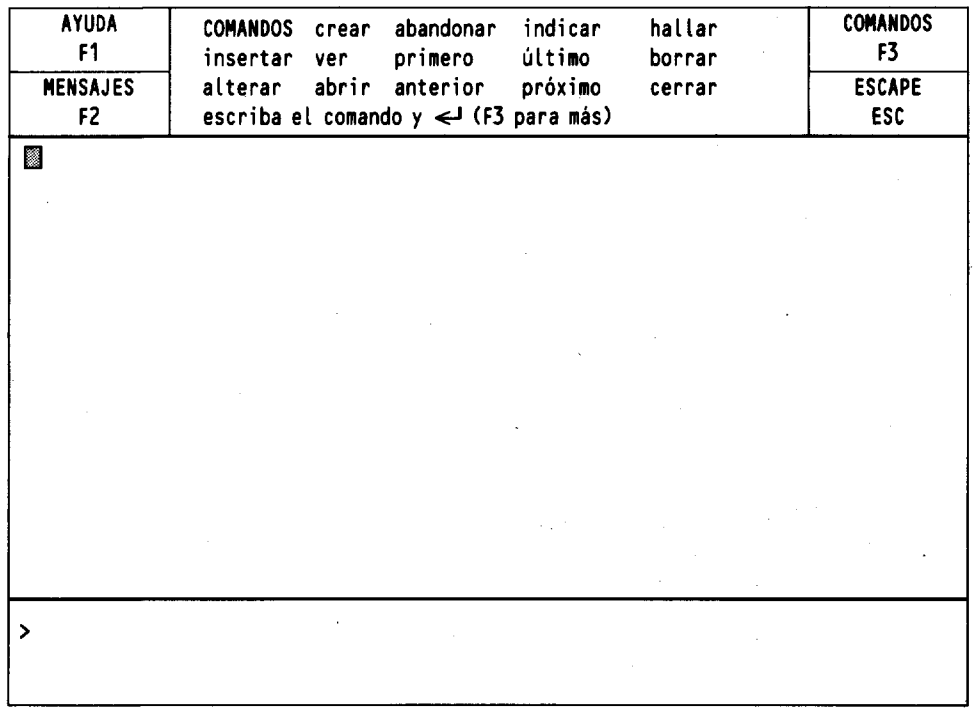

Figura 2.1 La pantalla al emplear un monitor (80 caracteres).

Si Vd. utiliza un televisor, la pantalla tiene un aspecto ligeramente distinto. Esto se debe a que un televisor no puede normalmente mostrar con claridad 80 caracteres por línea. Por consiguiente, en este caso Archive sólo muestra 64 caracteres por línea.

La pantalla está dividida en tres zonas: la zona de presentación, la zona de trabajo y la zona de control.

La zona de presentación y la zona de trabajo

Como su nombre indica, la zona de presentación es donde aparece toda la información producida por Archive.

La zona de trabajo ocupa las cuatro últimas líneas de la pantalla. Aparecen en esta zona todos los comandos que Vd. escribe, junto con los mensajes de error.

Estas dos zonas actúan casi siempre juntas, ya que los comandos escritos en la zona de trabajo producen resultados en la zona de presentación.

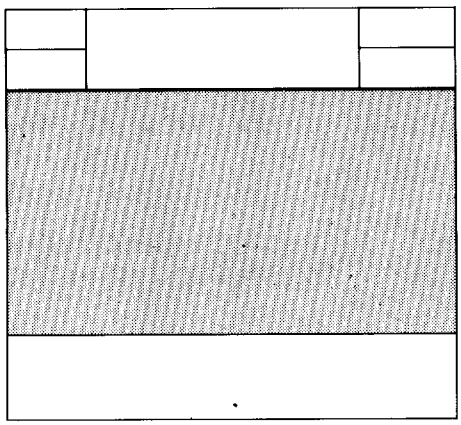

Figura 2.2 La zone de presentación

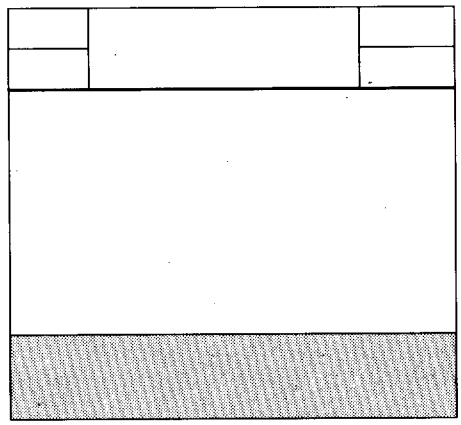

Figura 2.3 La zone de trabajo

Para servirle de ejemplo, escriba el breve programa siguiente, exactamente como se muestra.

```
haz x=13:mientras x_>0:escribir x:haz x=x-l:finmientras ↵
```

El texto de este programa aparecerá en la primera línea de la zona de trabajo. Al pulsar ↵, los números entre trece y uno aparecerán en líneas sucesivas de la zona de presentación. La línea inferior de la zona de presentación se dejará en blanco, a excepción del cursor rojo que indica la próxima posición en que aparecerá el texto. Aparecen los números entre uno y trece, que junto con la línea en blanco inferior, ocupan las quince líneas de la zona de presentación.

El comando:

```
limpiar ↵
```

borrará todos los datos en la zona de presentación.

La zona de control

La zona de control ocupa las primeras líneas de la pantalla y se indican aquí las opciones normales para obtener Ayuda (**F1**), para activar y desactivar las indicaciones (**F2**), para cancelar una operación incompleta (**ESC**) y para seleccionar un comando (**F3**).

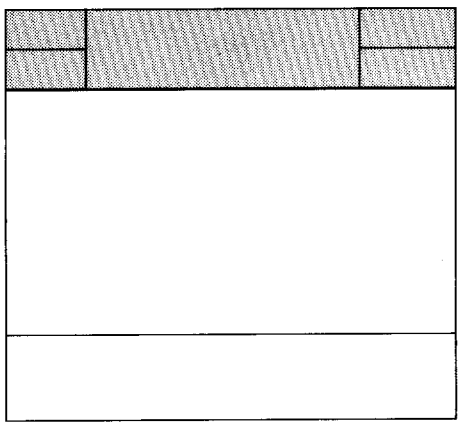

Figura 2.4 La zona de control

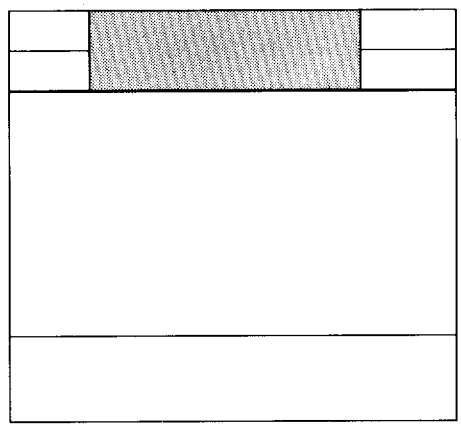

Figura 2.5 Los comandos

EMPLEO DE LOS COMANDOS

Debido a que los comandos de Archive forman un lenguaje de programación, precisará escribir sus nombres completos. Esto podrá parecerle un proceso muy largo, pero más adelante verá cómo puede crear procedimientos para introducir comandos pulsando solamente una tecla.

Hay cuatro listas de comandos que aparecen en la pantalla pulsando **F3**. Si ya se encuentra una lista de comandos en la pantalla, al pulsar **F3** aparece la lista siguiente. Puede utilizar la mayor parte de estos comandos escribiendo el nombre del comando y pulsando ↵. Sin embargo, ciertos comandos precisan más información y le indicarán que la escriba.

Puede utilizar cualquiera de los comandos, aun cuando el nombre del mismo no aparezca en ese momento en la zona de control.

EL COMANDO MODO

El programa le ofrece la posibilidad de combinar las zonas de control, presentación y trabajo en una sola zona, utilizando para ello el comando **modo**. Utilizando **modo** por si sólo combinará las tres zonas en una sola y obtendrá el mismo efecto si escribe **modo 0**. Pruebe con:

```
modo 0 [↵]
```

y los datos introducidos desde el teclado y todo lo que aparece en la pantalla –como resultado de un comando o de un programa– comparten la totalidad de la pantalla. El valor 1 vuelve a dividir la pantalla en las tres zonas.

Puede también utilizar el comando **modo** para cambiar el número de caracteres que se presentan en cada línea de la pantalla. Para esto, precisa dar un segundo número separado con una coma del primero. El segundo número debe ser 4, 6 u 8 para seleccionar pantalla de 40, 64 u 80 columnas. Escriba:

```
modo 0,4 [↵]
```

para cambiar la pantalla a 40 caracteres y para combinar las tres zonas de la pantalla en una sola. Observe que para cambiar el tamaño de la pantalla precisa escribir el 0, que antes era opcional.

Pruebe con distintas combinaciones para ver el efecto que tienen en la pantalla. Termine con un comando que deje la pantalla dividida en las tres zonas, pero elija el número de caracteres por línea que le provea unas indicaciones claras en el monitor o televisor.

CAPITULO 3
FICHEROS DE QL ARCHIVE

REGISTROS Y CAMPOS DE LOS FICHEROS

Un fichero de Archive se comporta como un índice de tarjetas. Un índice de tarjetas verdadero consta de una caja que contiene una serie de tarjetas de registro. Cada tarjeta tiene varios artículos (elementos) de información escritos en ella. Para que este índice de tarjetas tenga utilidad, se precisan reglas para determinar dónde se escribe en la tarjeta cada trozo de información.

Supongamos, por ejemplo, que se trata de un índice de nombres y direcciones. Normalmente, Vd. escribiría el nombre de la persona en la parte superior de la tarjeta, seguido de la dirección y el número de teléfono (si lo tiene). Resultaría muy difícil de utilizar si ciertas tarjetas tuvieran el nombre escrito en la parte superior y otras en la parte inferior. La forma en que se utiliza normalmente este índice es recorriendo las tarjetas con el dedo y leyendo solamente la línea superior, hasta hallar el nombre buscado.

Si Vd. tuviera dos juegos de tarjetas, el primero con registros de nombres y direcciones, y el segundo con registros de existencias, no guardaría entonces todas las tarjetas en la misma caja, sino que utilizaría dos cajas con los nombres, por ejemplo, "Registros de Clientes" y "Registros de Existencias".

Este sistema de índice de tarjetas contiene la mayor parte de las ideas necesarias para comprender cómo actúa un *fichero* de Archive. Un fichero es como la caja del índice de tarjetas y se le da un nombre para identificarlo. El fichero está formado por una colección de *registros*, cada uno de ellos con la misma finalidad que una tarjeta del índice. Por tanto, un fichero es, simplemente, una colección de registros relacionados entre sí.

Como ocurre con el índice de tarjetas, la información en cada registro está organizada en una forma regular. Los elementos de datos individuales, tales como números de teléfono, podrán tenerse en cierta parte de la tarjeta. Un registro en un fichero de Archive está organizado de la misma forma. Cada elemento de datos se almacena en una parte distinta del registro, denominada un *campo*. Un registro en un fichero de clientes, como el descrito anteriormente, contendrá un campo de nombre, un campo de dirección, un campo de descuento, etc.

Si esto fuera todo no merecería la pena utilizar un fichero de datos de Archive en lugar del índice de tarjetas físico. Pero el empleo de los registros electrónicos de un computador acarrea muchas ventajas. Un índice de tarjetas de clientes se dispondría normalmente por el orden alfabético de los nombres de los clientes, para facilitar la búsqueda de la información relacionada con un cliente dado. Pero supongamos que desea enviar una carta a todos los clientes que no han hecho un pedido en los últimos seis meses. Resultaría muy trabajoso recorrer todo el contenido de un índice de tarjetas para recopilar esa lista. Con Archive, puede realizar esta búsqueda con unos pocos comandos sencillos. Además, resulta fácil imprimir al mismo tiempo las etiquetas de direcciones para los sobres.

Esto le demuestra la gran cantidad de tiempo y trabajo que podrá ahorrarse utilizando Archive para almacenar y manipular los datos archivados.

CAPITULO 4
EXAMEN DE UN FICHERO

La mejor forma de aprender cómo actúa Archive es examinando el fichero de demostración, llamado **gazet**, que se provee en el cartucho Archive. Este fichero contiene información referente a varios países –el continente, la capital, la moneda, el idioma, el número de habitantes, el área que ocupa y la renta nacional per capita.

La mayoría de los ejemplos en los Capítulos 4 y 5 se refieren al fichero "gazet". Antes de utilizarlo, haga una copia del fichero siguiendo el procedimiento a continuación.

Después de cargar Archive, inserte un cartucho formateado en el Microdrive 2 y escriba:

```
salvaguardar [↵]
mdv1_gazet_dbf [↵]
mdv2_gazet_dbf [↵]
```

Aguarde a que se detengan los dos Microdrives; no se impaciente, ya que el fichero es bastante largo y tardará cierto tiempo en copiarse. Utilice la copia de este fichero, que se encuentra ahora en el Microdrive 2, para experimentar.

A partir de aquí, no siempre le indicaremos que escriba [↵] al final de cada comando, pero recuerde que debe siempre hacerlo.

El comando **ver** abre un fichero para que Vd. pueda ver su contenido, pero con este comando no puede hacer modificaciones o adiciones en el fichero. Este es un comando más seguro que **abrir** si desea simplemente examinar un fichero, ya que en este caso estará protegido el fichero contra las modificaciones accidentales. Puede examinar la copia del fichero "gazet" en el Microdrive 2 escribiendo:

```
ver "gazet"
```

EXAMEN DE UN REGISTRO

Para examinar el primer registro, escriba:

```
primero indicar
```

No se olvide de pulsar [↵] después de cada comando y aparecerá entonces en la pantalla el primer registro del fichero.

Observe que en la primera línea aparece el nombre lógico del fichero. Archive provee automáticamente el nombre "maestro" cuando se trata de un solo fichero. Los nombres lógicos de fichero se emplean al utilizar más de un fichero a la vez, como se describe más adelante.

EXAMEN DE OTROS REGISTROS

Después de ver el primer registro del fichero, podrá pasar al registro siguiente escribiendo:

```
próximo
```

y aparecerá en la pantalla el próximo registro del fichero. Cuando Vd. escribe otros comandos después del comando **indicar**, se actualiza continuamente la zona de presentación para mostrar el contenido del registro actual. Puede utilizar el comando **próximo** para recorrer el fichero, registro por registro, hasta llegar al final del mismo (no se pasará del último registro).

Hay otros tres comandos que puede utilizar para controlar qué registro del fichero se presenta en la pantalla.

`anterior`	presenta el registro anterior,
`primero`	presenta el primer registro,
`último`	presenta el último registro del fichero.

Pruebe estos comandos para recorrer las varias partes del fichero y presentar en la pantalla el registro que le parezca. Observe que los cuatro comandos **primero, último, próximo** y **anterior** no presentan de por si el registro. Simplemente pasan de un registro a otro, independientemente de si Vd. utilizó el comando **indicar**.

BUSQUEDA DE UN FICHERO

Hallar

El primero y el más sencillo de los comandos de búsqueda es **hallar**. Este comando buscará desde el principio del fichero hasta el primer caso en que aparece el trozo de texto especificado en cualquiera de los campos de texto. Por ejemplo:

```
hallar "África"
```

Cuando Vd. pulsa [↵], tiene lugar una breve pausa y aparece entonces el primer registro que contiene la palabra "África" en cualquiera de sus campos de texto. Observe que esta búsqueda es independiente de si el texto es en mayúsculas o en minúsculas, hallando por tanto "Africa", "AFRICA" o "África".

Continuar

Si el primer registro hallado que contiene el texto no es el que Vd. desea, podrá saltar al próximo caso escribiendo:

```
continuar
```

El comando **continuar** repite la búsqueda para hallar el próximo caso en que aparece el texto en cualquiera de los campos de los registros siguientes.

Puede que tenga que repetir varias veces una búsqueda hasta hallar el registro que desea. Pulse **F5** y Archive volverá a presentar el comando previo en la línea de comandos. Pulse [↵] y se ejecutará el comando.

Buscar

Otro método para localizar un registro dado es con el comando **buscar**. Esto le permitirá hallar un registro especificando el contenido de uno o más campos dados. Por ejemplo:

```
buscar continente$="EUROPA" yy idiomas$="FRANCES"
```

hallará el primer registro del fichero que cumpla con ambas condiciones. Usted debe escribir el nombre completo del campo.

A diferencia del comando **hallar**, que examina todos los campos de texto de los registros, **buscar** solamente examina los campos que Vd. especifique. Además, distingue entre mayúsculas y minúsculas. Si desea que **buscar** no distinga entre mayúsculas y minúsculas, podrá utilizar las funciones mayúsc() o minúsc(). A fin de no tener problemas en reconocer los acentos, es preferible convertir a mayúsculas. Por ejemplo:

```
buscar mayús(continente$)="EUROPA"
```

Aquí también, podrá emplear el comando **continuar** para hallar el próximo caso en que aparece este texto.

Elegir

En muchos casos, podrá interesarle examinar un subgrupo de registros en un fichero. Supongamos, por ejemplo, que sólo desea ver los detalles de los países de Europa. Puede emplear el comando **elegir** para seleccionar del fichero todos aquellos registros que satisfagan cierta condición. El fichero se comportará entonces como si sólo tuviera esos registros seleccionados. Pruebe este comando en el fichero de datos "gazet" para ver como actúa. Escriba primeramente:

```
escribir cuenta()
```

que le indicará cuántos registros tiene el fichero. Escriba entonces:

```
elegir continente$="EUROPA" escribir cuenta()
```

y verá cuántos registros han sido seleccionados. Los registros retirados del fichero continúan en la memoria del computador y podrá restablecerlos al fichero en cualquier momento con el comando **restaurar**. Escriba:

```
restaurar
```

y vuelva a escribir el valor de **cuenta()**, para comprobar que se ha restaurado el fichero a su estado original.

Cuando utiliza el comando **escribir** desde el teclado, se borrará el fichero que aparece en la pantalla. Esto se debe a que, en general, **indicar** y **escribir** utilizan áreas superpuestas de la memoria. Después de utilizar **escribir**, debe volver a utilizar **indicar** para restablecer la presentación en la pantalla.

CLASIFICACION DE UN FICHERO

Los registros de un fichero podrán no estar en el orden que Vd. desea. Puede ordenar el fichero por el contenido de los campos numéricos o de los campos de texto. **Al ordenar un fichero sólo se tienen en cuenta los ocho primeros caracteres de texto.**

Supongamos, por ejemplo, que desea clasificar los registros del fichero "gazet" en orden alfabético por capitales. Podrá hacerlo con el comando **ordenar** como sigue:

```
ordenar capital$;a
```

La letra "a" que sigue al signo de punto y coma especifica que Vd. desea clasificar el fichero en orden ascendente. Si desea clasificar el fichero en orden descendente utilice en su lugar la letra "d". El campo capital$ se convierte en la *clave de clasificación* para el fichero. Puede especificar hasta cuatro campos para la clave de clasificación –dando una lista de campos a continuación del comando **ordenar**. Para cada clave precisará especificar si la clasificación va a efectuarse en orden ascendente ó descendente. El comando que sigue, por ejemplo, clasificará el fichero en orden descendente por habitantes y en orden ascendente por capitales.

```
ordenar hab;d,capital$;a
```

Observe que cada nombre de campo está separado con el signo (;) de la letra "a" o "d" que especifica el orden ascendente o descendente, pero cada par (nombre del campo y letra) está separado del próximo con una coma.

Cuando se especifica más de un campo a los fines de la clasificación, los registros se clasifican inicialmente por el contenido del primer campo en la lista. Si dos o más registros tienen el mismo contenido en este campo, se clasifican por el próximo campo en la lista. Si hay registros que son iguales en cuanto al contenido de estos dos campos,se clasifican por el contenido del tercer campo y así sucesivamente.

SITUAR

Después de clasificar (ordenar) un fichero, podrá utilizar el comando **situar** para hacer que un registro dado sea el registro actual en el fichero. Este comando halla el primer registro cuyo primer campo de clasificación es mayor o igual que la expresión dada. Este registro se convierte en el registro actual del fichero.

Por ejemplo, si el fichero "gazet" ha sido clasificado según lo mencionado en el ejemplo anterior, el comando:

```
situar "100"
```

localiza el primer país en el fichero clasificado que tenga 100 millones de habitantes. Si no existe un país con este número de habitantes, Archive localizará el primer país con menos de 100 millones de habitantes (recuerde que el fichero fue clasificado en orden descendente).

El comando **situar** va seguido de una *expresión* que podrá ser de texto o numérica, pero debe ser del mismo tipo que el campo utilizado para ordenar el fichero. (Vea la Guía de Consulta).

Puede localizar un registro en cuanto al contenido de más de un campo de clasificación, utilizando **situar** con varias expresiones separadas por comas. Por ejemplo:

```
haz a=100 haz b$="D" situar a,b$
```

hallará el primer país con 100 (millones) de habitantes o menos y cuyo nombre de la capital comience con la letra "D", o se encuentre después de la D en el alfabeto. En este ejemplo, Archive localizará Bangladesh, con una población de 76.1 millones y capital Dacca.

La única restricción en el número de expresiones que pueden utilizarse con **situar** es el número de campos utilizados para clasificar el fichero.

No se puede emplear **continuar** después de **situar**, ya que la repetición del comando **situar** con la misma condición siempre localizará el mismo registro.

Situar ofrece el método más rápido de localizar un registro en un fichero grande ya clasificado. Debido a la incertidumbre en el registro localizado, precisará, generalmente, efectuar otra comprobación en el registro para cerciorarse de que es el que desea.

CIERRE DE UN FICHERO

Cuando haya terminado de examinar un fichero, debe indicárselo a Archive. Puede hacerlo escribiendo:

```
cerrar
```

Esto sólo actúa en los ficheros y deja intactos los programas y el formato de la pantalla. Puede cerrar todos los ficheros y borrar los datos y la pantalla con el comando:

```
nuevo
```

Esto deja Archive en el estado inicial que tiene al cargarlo.

O bien, si ha terminado de trabajar con Archive, podrá regresar a SuperBASIC con el comando **abandonar**. Este comando cierra automáticamente todos los ficheros abiertos antes de salir de Archive.

Recuerde que no debe jamás extraer del Microdrive un cartucho que contenga ficheros abiertos.

CAPITULO 5
MODIFICACION DE UN FICHERO

Antes de escribir los ejemplos en este capítulo, escriba primeramente **nuevo** para borrar todos los datos de Archive y dejarlo listo para empezar de nuevo.

El comando **abrir** prepara un fichero para poder leerlo y escribir en el mismo.

Si Vd. abre un fichero con el comando **abrir**, en lugar de con **ver**, podrá escribir en el fichero para cambiar su contenido, además de leerlo. Esto significa que las adiciones, supresiones o modificaciones producirán un cambio permanente en la copia del fichero al cerrarlo. Escriba:

```
abrir "gazet"
```

Si Vd. abrió un fichero para leerlo con el comando **ver**, no debe entonces utilizar comandos que traten de modificar los datos. Si lo hace, Archive reportará un error. Los comandos descritos en este capítulo modifican los ficheros de datos, debiendo por tanto utilizarse solamente en un fichero abierto con el comando **abrir**.

Ponga en la pantalla el primer registro del fichero con:

```
primero indicar
```

CIERRE DE UN FICHERO

Cuando haya terminado de hacer modificaciones en el fichero, debe cerrarlo (con **cerrar** o **nuevo**) para que se registren todos los cambios.

Si no cierra el fichero como es debido (por ejemplo, si se limita a apagar el computador cuando haya terminado), podrá alterarse el fichero y no se registrarán los cambios más recientes. **Cerciórese siempre de que no hay ficheros abiertos en el cartucho antes de extraerlo del Microdrive. No apague el computador sin antes cerrar todos los ficheros abiertos y extraer los cartuchos de los Microdrives.**

INSERTAR

El comando **insertar** sirve para añadir uno o más registros al fichero actual. Al utilizar este comando, le pide que escriba el contenido de cada campo del nuevo registro. Escriba:

```
insertar
```

Aparece ahora en la zona de presentación:

```
Nombre lógico : maestro
pais$         :
continente$   :
capital$      :
moneda$       :
idiomas$      :
hab           :
área          :
rnb           :
```

Escriba ahora el contenido de cada campo. Puede pasar de un campo al próximo pulsando [↵] o **TAB**, así como retroceder al campo anterior pulsando **SHIFT** y **TAB** juntas. Puede hacer tantos cambios en los campos como le parezca, hasta que esté satisfecho. Para insertar el nuevo registro en el fichero pulse **F5**. Para salir de **insertar** pulse **F4**. Escriba, por ejemplo:

```
ESCOCIA           TAB
EUROPA            TAB
EDINBURGO         TAB
LIBRA ESTERLINA   TAB
INGLES            TAB
10                TAB
30                TAB
50                TAB
```

La zona de presentación mostrará:

```
Nombre lógico     : maestro
pais$             : ESCOCIA
continente$       : EUROPA
capital$          : EDINBURGO
moneda$           : LIBRA ESTERLINA
idiomas$          : INGLES
hab               : 10
área              : 30
rnb               : 50
```

Cuando se haya asegurado de que ha escrito correctamente la nueva información, pulse **F5** para insertar el nuevo registro en el fichero. Los campos que acaba de escribir quedarán entonces en blanco, listos para insertar un nuevo registro. Pulse **F4** para terminar con el comando **insertar**.

Puede también terminar la inscripción en cada campo y pasar al próximo pulsando [↵]. El nuevo registro se añade automáticamente al fichero al pulsar [↵] después del último valor.

Si el fichero está ordenado, el nuevo registro se inserta en la posición correcta para mantener el orden.

BORRAR

Puede utilizar el comando **borrar** para retirar un registro de un fichero. Este comando retira del fichero el registro actual (el que se muestra con el comando **indicar**). Todo lo que tiene que hacer para retirar un registro dado es presentarlo en la pantalla con **indicar** y, después de cerciorarse de que se trata del registro que desea borrar, escriba:

```
borrar
```

MODIFICACION DE UN REGISTRO

También resulta muy fácil modificar el contenido de uno o todos los campos de un registro existente. Hay dos métodos para esto.

Alterar

El primero es utilizando el comando **alterar**. Seleccione el registro que desea modificar (con **indicar** y **hallar**) antes de usar el comando **alterar**. Actúa de la misma forma que **insertar**, a excepción de que en cada campo se muestra el contenido previo. Puede saltar aquellos campos cuyo contenido no desea modificar (con **TAB** o [↵]). Escriba un nuevo valor, o utilice las teclas del cursor para modificar el valor previo. Pulse **F5** para sustituir el registro.

Como ocurre con **insertar**, se sustituye automáticamente el registro pulsando [↵] después de escribir un valor para el último campo del registro.

Actualiz

Seleccione primeramente el registro que desea modificar y cambie entonces el contenido de los campos hasta dejar el registro mostrado exactamente como lo desea. Escriba **actualizar** para modificar el registro.

Supongamos, por ejemplo, que decide clasificar Islandia formando parte de Europa, en lugar de en el Artico. Localice primeramente el registro, escribiendo:

```
hallar "islandia"
indicar
```

Luego utilice el comando **haz** y modifique el contenido del campo continente$:

```
haz continente$ = "EUROPA"
```

Finalmente, introduzca este cambio en el registro, escribiendo **actualizar**.

Con estos dos métodos, se insertará el nuevo registro en la posición correcta si el fichero estaba ordenado. En caso contrario, el nuevo registro se inserta en una posición no especificada del fichero.

El comando **alterar** es más sencillo de usar, pero siempre afecta al registro actual. El comando **actualizar** resulta útil al trabajar con varios ficheros a la vez.

Antes de apagar el computador, recuerde que debe cerrar el fichero con uno de los comandos **cerrar**, **nuevo** o **abandonar**.

CAPITULO 6
CREACION DE UN FICHERO

Si ha estado siguiendo los ejemplos hasta aquí, sólo habrá utilizado Archive para examinar el fichero suministrado. En este capítulo se le indica cómo puede crear su propio fichero y elegir los nombres de fichero que le parezcan.

En caso necesario, escriba **nuevo** para borrar el contenido en la memoria del computador y cerrar los ficheros que estén abiertos. Cerciórese de que tiene insertado en el Microdrive 2 un cartucho formateado para el fichero que Vd. va a crear.

Supongamos que desea utilizar Archive para preparar un catálogo de libros. Para esto, precisará crear un nuevo fichero llamado, por ejemplo, "libros". El primer paso para crear un fichero es decidir qué información va a contener, es decir, los campos que Vd. va a utilizar en cada registro. En este caso precisará, evidentemente, registrar el autor, el título y la materia; podrá también interesarle incluir otros detalles, tales como el tipo de libro (novela o biografía), ISBN (Número de Libro Estándar Internacional), lugar en la estantería, una descripción breve, etc. En este ejemplo utilizaremos, simplemente, tres campos de texto para contener el autor, el título y la materia, junto con un campo numérico para el dato ISBN.

CREAR

Para crear un fichero se emplea el comando **crear**. Usted debe especificar el nombre del fichero que desea crear y los nombres de los campos que va a utilizar en cada registro (el símbolo $ indica que el campo contiene texto). Cuando haya acabado de definir los campos de un registro, termine el comando **crear** con **fincrear**. Podrá crear un fichero para un sencillo catálogo de libros, según lo explicado, escribiendo la siguiente secuencia.

```
crear "libros"
autor$
titulo$
materia$
isbn
fincrear
```

Tenga en cuenta que no precisa escribir el comando final **fincrear**. Puede hacerlo si lo desea, pero puede también terminar la creación de un fichero simplemente pulsando [↵] en una línea de entrada en blanco. (Pero debe siempre incluir **fincrear** cuando utilice el comando **crear** en un programa de Archive).

ADICION DE REGISTROS

Cuando Vd. haya creado un fichero según lo mencionado, quedará abierto en la memoria del computador para leer y escribir en el mismo, pero no contiene aún ningún registro. Puede añadir registros con el comando **insertar**. Escriba:

```
insertar
```

y se mostrará en la zona de presentación:

```
nombre lógico : maestro
autor$        :
titulo$       :
materia$      :
isbn          :
```

Todo lo que tiene que hacer ahora es escribir el contenido de cada campo. Por ejemplo:

```
Pérez, J
Manual Técnico
Fabricación de Cañones
1234567
```

Aparece entonces en la zona de presentación:

```
nombre lógico : maestro
autor$        : Pérez, J
titulo$       : Manual Técnico
materia$      : Fabricación de Cañones
isbn          : 1234567
```

Pulse **F5** para insertar el registro en el fichero mostrado con el comando **ver**. Se borra el contenido del campo, listo para insertar otro registro.

Recuerde también que puede terminar de introducir datos para cada campo y pasar al próximo pulsando [↵], y que al pulsar [↵] después del último valor se añade el registro al fichero.

Cuando haya terminado pulse **F1** y acuérdese de cerrar el fichero con **cerrar** o **abandonar**.

CAPITULO 7
FORMATOS DE LA PANTALLA

Cuando Vd. utiliza el comando **indicar** en un fichero que ha creado, los registros se muestran en el formato estándar de Archive.

DEFINICION DE UN FORMATO DE LA PANTALLA

Usted puede diseñar su propio formato de la pantalla que mejor vaya para la información en su fichero de datos. Abra un fichero existente con el comando **abrir** y escriba:

```
indicar
```

Para modificar el formato de la pantalla utilice el comando **peditar**. Escriba:

```
peditar
```

En la zona de presentación aparece el formato actual de la pantalla –el que Archive crea automáticamente. Si no hay un formato de la pantalla en la memoria del computador, la zona de presentación podrá estar en blanco.

Si aparece un formato, verá que no se incluyen los valores de los campos para los ficheros. Los espacios en que aparecen normalmente estos valores están marcados con filas de puntos. El formato de la pantalla puede considerarse como un plano de fondo, sobre el cual se muestran los valores de un número de variables en ciertas posiciones. Archive muestra el formato de la pantalla en dos etapas –en primer lugar, traza el texto de fondo y luego indica los valores de las variables en las posiciones marcadas en la pantalla.

Después de seleccionar **peditar** se encontrará al *nivel maestro* del comando y tiene entonces tres opciones:

escribir texto de fondo en la pantalla,
pulsar **ESC** para salir de **peditar**,
pulsar **F3** para emplear un comando de edición de la pantalla.

Para diseñar un formato de la pantalla, pulse **F3** y luego la tecla L para borrar (limpiar) la pantalla y comenzar de nuevo. Pulse [↵] para confirmar su elección; si pulsa cualquier otra tecla regresará al nivel maestro de **peditar**.

Elija los colores del papel y de la tinta pulsando P o T y luego pulsando cualquier tecla para pasar de un color a otro de los cuatro. Pulse [↵] para regresar al nivel maestro e introducir el texto de fondo.

El texto de fondo podrá ser significativo, tal como:

Diccionario geográfico de José Cuesta

o podrá estar comprendido por un nuevo nombre para cada uno de los campos del fichero:

habitantes (millones):

Puede mover el cursor a cualquier parte de la zona de presentación mediante las cuatro teclas del cursor. Lo que Vd. escribe aparece inmediatamente en la zona de presentación, en la posición del cursor, formando parte del fondo para el formato de la pantalla. La única excepción es cuando se posiciona el cursor en una parte de la pantalla que está reservada para mostrar una variable. En este caso, Archive indica el nombre de la variable en la zona de trabajo, en la parte inferior de la pantalla. Usted no puede escribir texto de fondo en esta zona, a no ser que la deje antes libre como se explica más adelante.

Los cuatro comandos para editar la pantalla le permiten obtener unos formatos de buen aspecto para presentar los datos. La operación de limpiar la pantalla ya se ha explicado. Podrá tener que experimentar un poco antes de familiarizarse con los tres comandos, con lo cual le convendrá utilizar una copia del fichero de datos que no le importe perder.

COMANDOS PARA EDITAR LA PANTALLA

Marcar Variable (V)

Supongamos que desea mostrar el valor de la variable país$ en cierta posición de la pantalla. Mueva el cursor a esa posición, pulse **F3** y luego la tecla V. Archive le pide el nombre de la variable. Escriba:

```
pais$
```

Observe que este nombre no aparece en la pantalla –en este momento Vd. sólo está marcando el punto en que se presentará el *valor*. Al pulsar [↵] Archive le pide que indique cuánto espacio desea reservar para mostrar el valor. Pulse cualquier tecla, excepto [↵], para marcar el espacio con una fila de puntos. Puede utilizar **CTRL** y la tecla "izquierda" del cursor para borrar espacio reservado. Cuando haya reservado el espacio suficiente, pulse [↵] y Archive vuelve a dejarle al nivel maestro de **peditar**.

Si mueve el cursor a una de las zonas reservadas (marcadas con puntos), Archive muestra, en la zona de trabajo, el nombre de la variable para la cual está reservado el espacio.

Si reserva espacio para una variable en una parte de la pantalla que se superpone con una zona ya reservada, tendrá la opción de cancelar la zona previa. Podrá entonces volver a utilizar la opción para asignar el espacio a una nueva variable.

Tinta (T)

Supongamos que desea cambiar el color de la tinta. Mueva el cursor al punto en que desea que comience el texto en el nuevo color, pulse **F3** y luego la tecla T. Archive indica los cuatro colores disponibles en la zona de trabajo, con el color seleccionado resaltado. Pulse cualquier tecla para cambiar el color seleccionado y pulse entonces [↵] para registrar su selección. Todo lo que Vd. escriba ahora aparecerá del color que Vd. seleccionó hasta que vuelva a usar el comando **tinta** para cambiar a otro color.

Papel (P)

El cambio del color del papel actúa de la misma forma –excepto en que Vd. pulsa **F3** y luego la tecla P.

Si desea que el cambio de color solamente afecte a parte de una línea, mueva el cursor al comienzo de la parte afectada y seleccione el papel y tinta que desea. Mueva entonces el cursor al final de esa parte y vuelva a seleccionar el color y tinta para restaurarlos a sus valores originales.

ACTIVACION DE UN FORMATO DE LA PANTALLA

Una vez que haya diseñado el formato de la pantalla y haya salido de **peditar**, el formato de la pantalla quedará en el estado *activo*. Esto significa que se presentarán automáticamente los valores de todas las variables en el formato de la pantalla cada vez que Archive complete un comando (o un programa). Si escribe, por ejemplo, el comando **próximo**, Archive pasa al próximo registro del fichero actual y muestra los campos incluidos en el formato de la pantalla. El formato activo de la pantalla se desactiva cada vez que Vd. utiliza el comando **limpiar**.

Si un formato no está activo, podrá activarlo con el comando **pantalla**. Esto presenta el texto de fondo del formato de la pantalla, pero no indica los valores actuales de las variables.

COMO SALVAR Y CARGAR FORMATOS DE LA PANTALLA

Usted puede salvar su propio diseño de la pantalla en un cartucho del Microdrive con el comando **psalvar**:

```
psalvar "nombfich"
```

donde "nombfich" es el nombre que Vd. decide dar al fichero. El formato se salva exactamente como aparece en la pantalla.

Puede volver a cargar el formato de la pantalla con el comando:

```
pcargar "nombfich"
```

Cuando Vd. carga un formato de la pantalla, se activa y aparece automáticamente en la pantalla.

Archive no actualiza automáticamente un formato de la pantalla activa desde el interior de un programa. Supongamos que desea indicar todos los registros del fichero actual, uno tras otro, y ha intentado hacerlo escribiendo el siguiente programa de una línea:

```
primero: haz x=0: mientras x<cuenta():próximo:haz x=x+1:finmientras
```

(Los comandos **mientras** y **finmientras** hacen que se ejecute repetidamente la sección del programa incluida entre ellos mientras sea verdadera –no sea cero– la condición que sigue a **mientras**. Para que actúe correctamente, cada comando **mientras** debe ir acompañado del correspondiente comando **finmientras**.)

Este programa no haría lo que Vd. quiere, ya que Archive sólo actualiza el contenido del formato de la pantalla al final del programa.

EL COMANDO PESCRIBIR

Pero Vd. puede forzar la presentación de los valores de las variables en una pantalla activa desde el interior de un programa, utilizando el comando **pescribir**. El siguiente programa de una línea hará que se muestren todos los registros requeridos:

```
primero:haz x=0:mientras x<cuenta():pescribir:próximo:
    haz x=x+1: finmientras
```

Si no hay ninguna pantalla activa, **pescribir** no tendrá ningún efecto.

EL COMANDO INDICAR

Recuerde que el comando **indicar** tiene su propio formato. Siempre sustituye a cualquier formato de la pantalla por su propia lista simplificada de los campos en el registro actual del fichero actual. Por tanto, Vd. debe **psalvar** su propio formato de la pantalla antes de que vuelva a utilizar **indicar**. De no hacerlo, el formato de la pantalla de Vd. será sustituido por el formato de **indicar** y no podrá entonces recuperarlo (a no ser que vuelva a diseñarlo con **peditar**).

CAPITULO 8
PROCEDIMIENTOS

Para utilizar los ejemplos en este capítulo, escriba primeramente **nuevo** para borrar la memoria del computador y luego escriba **ver "gazet"** para abrir el fichero de muestra en el cartucho de datos, que se supone Vd. habrá insertado en el Microdrive 2.

Los comandos y funciones de Archive forman, juntos, un lenguaje de programación que le servirá para escribir programas que le permitan manejar los ficheros. Verá que los programas de Archive son fáciles de escribir.

Un programa de Archive está formado por una o más secciones. Cada sección se denomina un *procedimiento*, que es, simplemente, una sección del programa a la que se le da un nombre. Esto le permite referirse al procedimiento por su nombre, lo mismo que con los procedimientos que Vd. escribe y utiliza en SuperBASIC. En Archive podrá ejecutar un procedimiento escribiendo el nombre del mismo en el teclado. Cuando Vd. escribe un procedimiento, tiene el efecto de añadir un nuevo comando al programa Archive.

Un procedimiento no puede tener más de 255 líneas y cada línea no debe contener más de 160 caracteres.

CREACION DE UN PROCEDIMIENTO

Para escribir o modificar un procedimiento debe emplear el *editor de programas*. Este editor le permite añadir, borrar o alterar el texto de los procedimientos.

Se describe detalladamente en el Capítulo 9, pero en este capítulo examinaremos brevemente algunas de sus características maestros para que podamos escribir unos pocos procedimientos cortos. Vamos a suponer que, inicialmente, no hay ningún procedimiento en la memoria del computador.

Escriba:

```
editar
```

para invocar el editor de programas. Verá que cambia la zona de control para indicarle que escriba el nombre de un nuevo procedimiento. Al invocar el editor siempre le permite crear un nuevo procedimiento si Vd. no ha definido o cargado aún otros procedimientos.

Por tanto, lo primero que tiene que hacer es definir el nuevo procedimiento. Vamos a comenzar con una tarea muy sencilla –facilitarnos el trabajo cambiando el nombre del comando **indicar**. Le daremos el nombre "i" para ahorrarnos el trabajo de escribirlo completo.

Escriba solamente:

```
i
```

El lado izquierdo de la zona de presentación contendrá el nombre del procedimiento. En el lado derecho de dicha zona se muestra una lista del procedimiento, el cual no contiene aún ningún comando. Las palabras **proc** y **finproc** que marcan el comienzo y el fin del procedimiento las pone Archive automáticamente.

Precisará ahora añadir el *cuerpo* del procedimiento –es decir, la secuencia de acciones que va a realizar.

La zona de control le muestra que puede insertar líneas de texto en el nuevo procedimiento. En lo referente al ejemplo anterior, el texto es el comando **indicar**. Escriba:

```
indicar
```

y Archive insertará el nuevo texto en el procedimiento, bajo la línea resaltada. Si ha estado siguiendo este ejemplo hasta aquí, la zona de presentación contendrá lo siguiente:

```
i    proc i
     indicar
     finproc
```

Podría añadir aquí más líneas de texto y cada línea se insertaría bajo la línea resaltada.

Pero en este caso, el procedimiento está completo y podrá ahora salir del comando **editar** pulsando **ESC** dos veces.

Todo lo que precisa hacer para usar el procedimiento es escribir el nombre del mismo, seguido de [↵]. Este nuevo procedimiento actuará exactamente igual que si escribe completo el comando **indicar**.

LISTADO E IMPRESION DE PROCEDIMIENTOS

Siempre que invoque el comando **editar** verá una lista de los nombres de todos los procedimientos definidos que se encuentran en la memoria del computador.

Puede listar cualquiera de estos procedimientos desde el comando **editar** pulsando la tecla **TAB** (para recorrer la lista hacia abajo), o pulsando juntas las teclas **SHIFT** y **TAB** (para recorrer la lista hacia arriba) hasta resaltar el nombre del procedimiento que le interesa. El procedimiento se lista automáticamente en el lado derecho de la pantalla. Si el procedimiento es demasiado largo para que quepa en la zona de presentación, aparece la primera parte del mismo. Podrá entonces recorrer el procedimiento hacia arriba o hacia abajo con las teclas "arriba" y "abajo" del cursor. Cuando haya terminado podrá salir de **editar** pulsando **ESC**.

Si desea obtener una lista impresa de los procedimientos, podrá utilizar para esto el comando **listar**. Todo lo que precisa hacer es escribir:

```
listar
```

y obtendrá en la impresora una lista de todos los procedimientos que se encuentran en la memoria del computador.

AVISO: No utilice este comando a no ser que esté conectada la impresora, ya que de lo contrario quedará "colgado" el programa.

COMO SALVAR Y CARGAR PROCEDIMIENTOS

Si desea conservar los procedimientos que ha definido para emplearlos en el futuro, podrá hacerlo con el comando **salvar**. Este comando almacena todos los procedimientos definidos en un fichero nombrado en el cartucho del Microdrive. Si desea salvar los nuevos procedimientos de Indicar que acaba de definir, con el nombre de fichero "misproc", escriba:

```
salvar "misproc"
```

Más adelante podrá volver a cargar estos procedimientos en la memoria del computador, escribiendo:

```
cargar "misproc"
```

El comando **cargar** borra los procedimientos existentes en la memoria antes de cargar los nuevos procedimientos desde el cartucho del Microdrive. Si desea añadir los nuevos procedimientos a los que ya se encuentran en la memoria, podrá hacerlo con el comando **unir**. Por ejemplo:

```
unir "misproc"
```

Este comando actúa como el comando **cargar**, excepto en que no se borran los procedimientos existentes. Si un nuevo procedimiento tiene el mismo nombre que uno existente, el nuevo sustituye a la versión previa.

COMO EXAMINAR LOS REGISTROS DE UN FICHERO

La operación de renombrar los comandos más comúnmente usados para darles nuevos nombres de una sola letra es una de las formas de facilitarle el trabajo. Otra forma sería escribiendo un procedimiento más largo para sustituir varios comandos por pulsaciones de una tecla. Utilice el comando **editar** para definir el siguiente procedimiento –le permitirá abrir y examinar cualquiera de sus ficheros de datos, a condición, naturalmente. de que el fichero que desea utilizar no esté ya cargado.

Si ya ha definido un procedimiento, al escribir:

```
editar
```

no le dará automáticamente la opción de crear un nuevo procedimiento. Al encontrarse en **editar**, debe pulsar **F3** y luego la tecla N para iniciar un Nuevo procedimiento.

No se preocupe si comete errores al escribir el ejemplo –aprenderá a corregirlos en el próximo capítulo.

NOTA: En este ejemplo se usa el número "1" para "primero" –en lugar de la letra P– a fin de distinguirlo de próximo que comienza con la misma letra. También se usa la letra "d" (delantero) para "anterior" -en lugar de la letra A– a fin de distinguirlo del comando "a" (Abandonar).

```
proc verfich
limpiar
leer "qué fichero? ";fichero$
ver fichero$
indicar
haz letra$="z"
mientras letra$_<_>"a"
  pescribir
  haz letra$=minúsc(tecla())
  si letra$="1":primero:finsi
  si letra$="ú":último:finsi
  si letra$="p":próximo:finsi
  si letra$="d":anterior:finsi
  finmientras
cerrar
finproc
```

Recuerde que para salir de **editar** tiene que pulsar **ESC** dos veces.

Puede utilizar el procedimiento, escribiendo:

```
verfich
```

Primeramente limpia la zona de presentación y luego le pedirá que escriba el nombre del fichero, tal como "gazet"; pero si ya estuviera cargado "gazet" recibirá un mensaje de error. Para solucionar esto, escriba **nuevo** y vuelva a cargar y ejecutar el procedimiento. Cuando haya escrito el nombre de uno de sus ficheros de datos, el procedimiento abrirá ese fichero en el modo de lectura solamente e indicará el primer registro del mismo. Aguardará entonces a que Vd. pulse una tecla y responderá a las teclas 1, ú, p, d, a. Las cuatro primeras causarán la correspondiente acción de indicar (**primero**, **último**, **próximo** o **anterior**) y al pulsar la tecla a (abandonar) cerrará el fichero y dará fin al procedimiento.

Por ser éste el primer programa de cierta longitud que hemos escrito, convendrá dar unos pocos comentarios. Observe que el ejemplo está sangrado (a distintos niveles) para aclarar la estructura del procedimiento. Usted no precisa escribirlo así, con todas los sangrados. Se añaden automáticamente al escribir, listar o imprimir el procedimiento.

La parte maestro del procedimiento (esperar a que sea pulsada una tecla y ejecutar la correspondiente acción) está comprendida entre **mientras** y **finmientras**. Sólo saldrá de este bucle repetidor cuando sea falsa la condición que sigue a **mientras** (en este caso, cuando Vd. pulsa la tecla a).

El comando **si**, utilizado varias veces dentro de este bucle, también requiere que cada **si** tenga el correspondiente **finsi** para marcar el final de la secuencia de instrucciones que deben ejecutarse si lá condición es verdadera. **Si** y **finsi** son comandos independientes y pueden emplearse en distintas líneas. Podríamos haber escrito, por ejemplo, la primera de las sentencias **si** en este procedimiento como sigue:

```
si letra$="1"
primero
finsi
```

Puede incluir varias líneas de sentencias entre **si** y**finsi**; se ejecutarán todas ellas, a condición de que sea verdadera la condición que sigue a **si**. En el procedimiento**verfich** estas sentencias son lo suficientemente cortas para poder escribir cada una de ellas en una línea, separándolas con el signo de dos puntos (:).

Como verá, se emplea un comando **pescribir** dentro del bucle maestro de este procedimiento para asegurar que se muestre en la pantalla cada nuevo registro. Recuerde que, si bien los comandos de Indicar (**primero**, **último** etc.) siempre pasan al registro correcto, los datos en la zona de presentación no se cambian automáticamente hasta llegar al fin del procedimiento. Si no hubiéramos incluido el comando **pescribir**, no habría aparecido ninguna información en la zona de presentación hasta que Vd. pulsara la tecla 'a' para abandonar el procedimiento (salir del mismo). En ese caso, solamente vería el resultado de la última de la secuencia de pulsaciones de teclas que Vd. ha efectuado.

CAPITULO 9
EDICION

En este capítulo se describe el editor de programas. Se incluyen aquí unos pocos ejemplos sencillos y la mejor forma de aprender es practicando con ellos. Para comenzar, escriba **nuevo** para borrar la memoria del computador.

Después de leer este capítulo podrá tratar de escribir unos pocos programas sencillos, o modificar los procedimientos con que trabajó en el capítulo anterior. Si desea emplear unos ejemplos más largos, podrá utilizar el editor para escribir los programas completos (o en parte) que aparecen en los capítulos siguientes.

EL EDITOR DE PROGRAMAS

Utilice el comando **editar** para entrar al *nivel maestro* del editor de programas.

Para servir de ejemplo, podemos crear un procedimiento y añadir un par de sentencias al mismo. Desde el nivel maestro de **editar**, pulse **F3** y la tecla N para crear un nuevo procedimiento. Cuando el programa se lo pida, escriba **prueba** para el nombre del procedimiento.

Pulse **ESC** dos veces para salir del editor sin añadir ninguna sentencia al procedimiento. Vuelva a utilizar entonces el comando **editar**. Si no tiene cargados otros procedimientos, aparecerá en la pantalla:

```
prueba    proc prueba
            finproc
```

Si continúan cargados los procedimientos que Vd. creó en el último capítulo, se resalta entonces a la izquierda el procedimiento **prueba** para distinguirlo de los otros. Pulse **F4** para insertar líneas de texto. Se resaltará la línea que contiene **proc**.

Escriba ahora:

```
escribir "ésta es una prueba" [↵]
escribir "tiene dos sentencias" [↵] [↵]
```

Pulsando [↵] dos veces seguidas hará que salga de **insertar**. Cuando haya terminado, la pantalla tendrá el siguiente aspecto:

```
prueba    proc prueba
            escribir "ésta es una prueba"
            escribir "tiene dos sentencias"
            finproc
```

Se resalta la línea que contiene la segunda sentencia de escribir.

Si comete un error, a condición de que lo note antes de pulsar [↵] podrá corregirlo con el editor de línea. Pero después de pulsar [↵] la línea se inserta en el procedimiento. Para volver a sacarla precisará pulsar **F5**. Si pulsa ahora [↵] se sustituye la línea previa por la nueva.

El programa no le permite editar la sentencia **finproc** al final del procedimiento, ni tampoco la palabra **proc** en la primera línea del procedimiento, pero podrá editar el resto del contenido de esta línea. Por tanto, podrá renombrar un procedimiento utilizando el editor de línea para borrar el nombre previo y escribir en su lugar el nuevo nombre. La lista de procedimientos a la izquierda de la pantalla se altera de conformidad, automáticamente, para mantener los procedimientos en orden alfabético.

Comandos de edición

Hay cuatro comandos de edición, que Vd. habrá notado en la sección de comandos al crear un nuevo procedimiento. Para seleccionar uno de estos comandos, pulse **F3** y a continuación la primera letra del nombre del comando.

Nuevo Procedimiento (N)

Usted escribe el nombre del procedimiento que desea crear. Si escribe el nombre de un procedimiento existente, se creará un nuevo procedimiento que sustituye al original.

Pulsando [↵] al final del nombre, el nuevo procedimiento se convierte en el procedimiento actual y aparece en la lista a la derecha de la pantalla. El programa le presenta un procedimiento vacío, es decir, uno que sólo contiene las sentencias **proc** y **finproc**.

Borrar Procedimiento (B)

Este comando borra del programa el procedimiento actual. En primer lugar, Vd. debe seleccionar el procedimiento que desea borrar, pulsando las teclas **SHIFT** y **TAB** según lo mencionado, para convertirlo en el procedimiento actual. Seleccione entonces el comando pulsando **F3** y luego la tecla B.

Debe pulsar entonces [↵] para confirmar que realmente desea borrar el procedimiento. Si cambia de idea en este momento, pulse cualquier otra tecla para regresar a **editar** sin borrar el procedimiento.

Tenga cuidado al utilizar este comando pues no se puede restaurar un procedimiento borrado, aparte de volviendo a escribirlo.

Quitar (Q)

Este comando retira una o más líneas de texto, del procedimiento actual. El texto retirado puede insertarse en otra posición, o incluso en otro procedimiento, con el comando **mezclar**

Antes de seleccionar el comando, utilice las teclas "arriba" y "abajo" del cursor para hacer que la línea actual sea la primera o la última línea de la sección que desea retirar. Seleccione entonces el comando pulsando **F3** y luego la tecla Q.

Si pulsa entonces [↵], se retira del procedimiento la línea actual. Puede también utilizar las teclas "arriba" o "abajo" del cursor para mover el cursor al otro extremo de la sección de texto que desea retirar. Se resalta el bloque de texto que va a ser retirado. Después de marcar así el texto que desea retirar, pulse [↵]. Archive retira inmediatamente el texto marcado y lo coloca en una parte reservada de la memoria, denominada memoria de edición.

Mezclar (M)

Este comando inserta en el procedimiento actual, bajo la línea actual, el texto retirado la última vez que se utilizó el comando **quitar**. El texto puede insertarse en otro lugar, o incluso en otro procedimiento.

Antes de seleccionar el comando utilice, en caso necesario, las teclas **SHIFT** y **TAB** para seleccionar el procedimiento en que desea insertar el texto. Debe también utilizar las teclas "arriba" y "abajo" del cursor para resaltar la línea inmediatamente encima de la posición en que desea insertar el texto.

Archive inserta inmediatamente el texto, bajo la línea actual. Después de utilizar **mezclar** para insertar el texto, la memoria de edición quedará vacía. Por tanto, no puede insertar el mismo texto en más de un lugar.

CAPITULO 10
PROGRAMACION EN ARCHIVE

En este capítulo se describe el desarrollo de un ejemplo práctico y se explica cada nueva técnica al utilizarla.

Supongamos que Vd. participa en la administración de un club o sociedad que carga una subscripción y produce una circular. Precisará enviar a cada socio una copia de cada edición. También precisará enviar a cada socio un aviso para recordarle la fecha en que debe pagar la subscripción.

Este ejemplo le permitirá preparar una lista de direcciones e imprimir las correspondientes etiquetas de direcciones cuando le interese. La etiqueta de la dirección incluye el recordatorio de cuándo debe pagarse la subscripción. En este ejemplo se supone que Vd. envía seis ediciones de la circular al año y que la subscripción de una persona vence cuando haya recibido seis ediciones. Podría fácilmente adaptarse a cualquier caso en que Vd. envía con regularidad una circular a varias personas en una lista de direcciones.

LISTA DE DIRECCIONES

En este ejemplo utilizaremos lo más posible los recursos existentes e introduciremos otros nuevos. Si precisa ayuda con alguna de las funciones o con un comando no explicado aún, o uno que parece hacer algo que Vd. no comprende, podrá resultarle más fácil dar ahora un vistazo a la Guía de Consulta o pulsar **F1** para pedir **Ayuda**. Utilizamos los comandos **insertar** y **alterar** para todas las adiciones y modificaciones en los registros del fichero. Sin embargo, precisaremos escribir unas rutinas (programas) especiales para imprimir las etiquetas de direcciones.

Habrá que tener en cuenta las siguientes operaciones:

Añadir un nuevo registro al fichero.
Borrar un registro.
Modificar un registro.
Anotar los pagos de las subscripciones.
Preparar las etiquetas de direcciones.
Salir del programa.

Para esto, escribiremos un procedimiento para cada una de estas tareas y las enlazaremos mediante otro procedimiento que nos permita seleccionar cualquiera de las opciones.

En esta aplicación, son evidentes los campos que debe contener cada registro. Precisaremos el nombre y dirección, más un campo para anotar el número de ediciones que ha recibido la persona. Podemos crear el fichero necesario inmediatamente, como se muestra a continuación.

```
crear "correo"
  titulo$
  nombre$
  apellido$
  calle$
  ciudad$
  provincia$
  distpostal$
  ediciones
  fincrear
```

Hemos empleado tres campos alfanuméricos para el nombre de la persona: uno para el título o tratamiento de cortesía (Dr., Sr., Sra., etc.), otro para el nombre de pila y el tercero para el apellido. Probablemente que podríamos habernos arreglado con un solo campo.

Tenemos otros cuatro campos alfanuméricos para la dirección: calle, ciudad, provincia y distrito postal. No siempre precisan utilizarse de esta forma, pero pueden tratarse como cuatro campos generales que contienen la dirección completa. Cuatro campos serán normalmente más que suficientes.

Solamente hay un campo numérico, para contener la información referente al número de ediciones que se han enviado.

Ahora que tenemos el fichero, podemos utilizarlo para probar los varios procedimientos al irlos escribiendo. Es aconsejable comprobar cada procedimiento, en lo posible, al crearlo. Podrá entonces localizar los errores al producirse y corregirlos inmediatamente. Si deja todas las comprobaciones hasta el final resultará mucho más complicado, ya que podrán darse varios problemas al mismo tiempo. Evite las complicaciones mientras se encuentra en la fase de comprobar los procedimientos –cerciórese de que cada procedimiento actúa como es debido antes de pasar al próximo. Verá así que el programa final funcionará como es debido tan pronto como haya escrito el último procedimiento.

Inserción

No es necesario escribir un procedimiento para añadir un registro, ya que para esto podemos utilizar el comando **insertar**. Recuerde que debe usar **pescribir** para mostrar en la pantalla el contenido de un registro desde dentro de un procedimiento. Puede hacer uso ahora de **insertar** para añadir unos pocos registros al fichero y poder así comprobar los otros procedimientos en un fichero real.

Supresiones

A veces, deseará remover los registros de aquellas personas que no hayan pagado la subscripción anual. Para esto escribiremos un procedimiento, **cancelar**, que le permitirá recorrer el fichero para examinar los registros de todas las personas que no han pagado y decidir si deben darse de baja.

Utilizaremos el campo variable **ediciones** para contener el número de ediciones que tiene derecho a recibir una persona. Todos aquellos registros en que sea cero el valor en **ediciones** serán por tanto candidatos para darles de baja.

```
proc cancelar
  nota ***** borrar socios que no han pagado *****
  limpiar
  indicar
  elegir ediciones =0
  todos
    pescribir
    escribir en 10,0; "BORRAR (s/n)? ";
    haz ok$ =mayúsc(tecla())
    escribir ok$
    escribir en 10,0;
    si ok$ ="s"
      borrar
      escribir "BORRADO"; tab 15
      sino
      escribir tab 15
      finsi
    fintodos
  restaurar
  finproc
```

```
NOTA: ok = correcto
```

Debido a que no puede recuperarse un registro borrado, se presenta en la pantalla todo el contenido del registro y el programa le pide confirmación de que desea realmente borrarlo. Empleamos la función **tecla()**, que aguarda a que sea pulsada una tecla y devuelve entonces el código ASCII para esa tecla. Observe que **mayúsc()** convierte el código a mayúsculas, con lo cual Vd. podrá escribir la letra en mayúsculas o en minúsculas.

Una vez que esté satisfecho de que ha escrito correctamente este procedimiento, podrá probarlo en su fichero (a condición, naturalmente, de que Vd. haya introducido algunos registros de prueba). En primer lugar, salga de **editar** pulsando **ESC** (dos veces en caso necesario) y salve el procedimiento en un fichero llamado "Listadirec".

Escriba:

```
salvar "listadirec"
```

El procedimiento **cancelar** quedará ahora almacenado y podrá invocarlo cuando cargue el fichero "Listadirec".

Después de introducir cada uno de los procedimientos siguientes, repita estos pasos para almacenar cada nuevo procedimiento en "Listadirec".

Pagos

Normalmente le interesará anotar los pagos de las cuotas, a partir de una lista de nombres y direcciones de aquellas personas que han enviado las subscripciones. Por tanto, precisará localizar el registro de una persona dada. El enfoque óptimo es escribir un procedimiento aparte, **locreg**, para localizar un registro dado y luego incorporarlo en el procedimiento **pago**.

Este procedimiento solicita una serie de texto (n$) y luego localiza el primer registro del fichero que contiene ese texto. Si Vd. responde pulsando [↵], se da a n$ el valor de la serie vacía (" ") y no se efectúa la búsqueda. Debe utilizar este método para indicar que ha terminado de registrar los pagos.

Desde el nivel de **editar**, pulse **F3** y la tecla N para introducir el procedimiento **locreg**.

```
proc locreg
  nota *****localizar un registro dado *****
  limpiar
  haz ok$ ="n"
  leer "quién? "; n$
  si n$ <>""
    hallar n$
    mientras ok$ <>"s" yy hallado()
      escribir titulo$ ; " "; nombre$(1); " "; apellido$
      escribir calle$
      escribir "OK (s/n)? ";
      haz ok$ =mayúsc(tecla())
      limpiar
      si ok$ <>"s"
        continuar
        finsi
      finmientras
    si no hallado()
      escribir n$ ; " no hallado"
      finsi
    finsi
  finproc
```

En esta búsqueda se emplea el comando **hallar** para localizar el texto en cualquier campo alfanumérico. Puede, por tanto, identificar un registro por el nombre o por la dirección. Naturalmente, el primer registro que cumpla las condiciones podrá no ser el que Vd. desea y precisaremos poder continuar la búsqueda. Esta es la finalidad del bucle **mientras finmientras**. Este bucle escribe el nombre y la primera línea de la dirección, para identificar el registro, preguntando entonces si se trata del registro correcto. Si Vd. no responde pulsando la tecla S continúa la búsqueda. Este bucle termina cuando Vd. responda pulsando la tecla S o cuando no se halle el texto en los registros restantes. Observe que la función **hallado()** devuelve un valor que no es cero (verdadero) si la búsqueda da resultado.

Debido a que ok$ podría ser inicialmente "s" (como resultado de una búsqueda previa que ha dado resultado), debemos darle otro valor al comienzo del procedimiento, antes de pasar al bucle. Esto asegurará que el bucle sea utilizado una vez como mínimo.

Podemos ahora escribir el procedimiento **pago**:

```
proc pago
  nota *****anotar el pago de la cuota *****
  limpiar
  haz n$ ="x"
  mientras n$ <>""
    locreg
    si ok$ ="s"
      haz ediciones =ediciones+6
      actualizar
      finsi
    finmientras
  finproc
```

El bucle en este procedimiento continuará hasta que n$ sea una serie vacía (" "). Esto le permitirá anotar varios pagos sin tener que seleccionar la opción **pago** para cada uno. Cuando haya terminado, pulse [↵] en respuesta a la pregunta "quién?". Si el valor de ok$ es "s" después de invocar el procedimiento **locreg**, se anota entonces el pago marcando que es válido para otras seis ediciones.

Aquí también, tenemos que ajustar el valor inicial de n$ a uno que sea apropiado (que no sea " ") para asegurar que el procedimiento no esté afectado por una operación previa.

Modificaciones

El procedimiento para permitirle cambiar el contenido de un registro se ha simplificado ahora mucho. Nuevamente, Vd. debe poder seleccionar un registro dado para modificarlo, con lo cual la estructura general será idéntica a la del procedimiento **pago**.

```
proc modif
  nota ***** alterar registro *****
  limpiar
  haz n$ ="x"
  limpiar
  mientras n$ <>""
    locreg
    si ok$ ="s"
      alterar
      limpiar
      finsi
    finmientras
  finproc
```

PARAMETROS

Vamos a dejar ahora brevemente el desarrollo del programa para describir el empleo de *parámetros* con los procedimientos. Puede utilizar un *parámetro* para pasar un valor a un procedimiento, en lugar de emplear el valor de una variable. Le daremos unos pocos ejemplos para que vea cómo se emplean.

Pruebe con el siguiente ejemplo sencillo. Puede añadir el parámetro a la línea que contiene el nombre del procedimiento, utilizando para ello el editor de línea.

```
proc comprobar; a
  escribir 5*a
finproc
```

Esto define un procedimiento llamado **comprobar** que requiere un parámetro, "a". Observe que el parámetro está separado del nombre del procedimiento por el signo (;). Siempre que vaya a utilizar este procedimiento, precisará dar un valor al parámetro. Por ejemplo, podrá escribir:

```
comprobar; 3
```

que presentará el valor 15 –el número (3) ha sido pasado al procedimiento como valor del parámetro "a".

Puede especificar cualquier número de parámetros para un procedimiento, a condición de que los separe con comas. Por ejemplo:

```
proc ensayo; a,b,c
  escribir a * b * c
finproc
```

que Vd. podrá invocar con:

```
ensayo; 3,4,5
```

Los valores que Vd. provee no precisan ser literales; podrán también ser variables, como se indica a continuación:

```
haz x = 2
haz y = 5
haz z = 7
ensayo; x,y,z
```

Observe que los nombres de las variables no precisan ser iguales que los nombres de los parámetros utilizados en el procedimiento. Podemos distinguir entre los *parámetros formales* (tal como a,b,c) en la definición del procedimiento, y los *parámetros reales* que son los valores reales que se pasan al procedimiento.

También podrá pasar los resultados de expresiones:

```
ensayo; x*2,z/y,(z-y)*x
```

No está limitado a utilizar variables numéricas, pudiendo también pasar series (expresiones alfanuméricas) como parámetros, a condición de que especifique variables alfanuméricas en la definición del procedimiento. Por ejemplo:

```
proc prueba; a$
  escribir a$
finproc

haz p$ = "mensaje"
prueba; p$
```

Lo único que se requiere es que el número y tipos de los parámetros suministrados coincidan con la lista de parámetros de pura forma en la definición del procedimiento.

Etiquetas de direcciones

La razón de esta pausa para introducir la explicación de los parámetros es que proveen una forma muy conveniente de escribir el procedimiento para imprimir etiquetas de direcciones. Para poder probarlo, escribiremos primeramente el procedimiento para mostrar las direcciones en la pantalla y más adelante lo convertiremos para enviar los datos a la impresora.

Supongamos que las etiquetas tienen ocho líneas que se imprimen. Si esto no va bien para la impresora y la combinación de etiquetas, precisará cambiar en el procedimiento el número de líneas de espacio para adaptarlo a los requisitos de Vd. Comience salvando otra vez los procedimientos en "Listadirec".

En primer lugar, escribiremos un procedimiento que presenta una línea en la pantalla, el contenido de la cual se pasa por medio de un parámetro.

```
proc linea; x$
  escribir x$
  finproc
```

Podemos ahora utilizar este procedimiento para presentar las ocho líneas de texto para las etiquetas de direcciones.

```
proc etiqueta
  nota ***** imprimir etiquetas *****
  si ediciones
    si ediciones =1
      linea; "RECUERDE pagar la suscripción"
      sino
      linea; ""
      finsi
    linea; ""
    linea; titulo$ +" "+nombre$ (1)+". "+apellido$
    linea; calle$
    linea; ciudad$
    linea; provincia$
    linea; distpostal$
    linea; ""
    haz ediciones =ediciones -1
    actualiz
    finsi
  finproc
```

Este procedimiento incluye un recordatorio en la etiqueta de la dirección si la persona está a punto de recibir la última edición. Cada vez que se imprime una etiqueta, se reduce en uno la cuenta de ediciones para esa persona. Si este número llega a cero, no se imprime entonces la etiqueta.

Podrá ver ahora lo útiles que resultan los parámetros –sin ellos, este procedimiento sería mucho más largo. Observe lo fácil que es combinar el título, el nombre y el apellido para la primera línea de la dirección.

Quizás se pregunte la razón de que hayamos definido **línea** en lugar de utilizar **escribir** en todo el procedimiento **etiqueta**. Lo hemos hecho para mostrar las líneas de la dirección en la pantalla. Podemos convertir este procedimiento para enviar la salida a la impresora, simplemente cambiando una línea en el procedimiento **línea**, en lugar de tener que cambiar todas las sentencias de escribir en el procedimiento **etiqueta**. Todo lo que tenemos que hacer es cambiar **línea** a lo siguiente:

```
proc linea; x$
  imprimir x$
  finproc
```

Finalmente, podemos escribir un procedimiento para imprimir todas las etiquetas de direcciones:

```
proc despachar
  limpiar
  todos
    etiqueta
    fintodos
  finproc
```

Para salir del programa

La opción final es salir del programa cuando Vd. haya terminado. Este procedimiento puede resultar muy sencillo –todo lo que se requiere del mismo es asegurar que se cierra debidamente el fichero antes de devolver el control al teclado. Hemos añadido también un breve mensaje de despedida para poner de relieve que ha terminado el programa.

```
proc adiós
  cerrar
  escribir "adiós"
  stop
  finproc
```

ERRORES

Es muy probable que más tarde o más temprano Vd. cometa un error al utilizar éste u otro programa. Podrá, por ejemplo, pulsar accidentalmente la tecla **ESC** o podrá escribir letras cuando debe escribir números. Este tipo de error lo detecta Archive y resulta normalmente en un mensaje de error y el regreso desde el programa al teclado.

Puede utilizar el comando **error** y marcar un procedimiento para que sea tratado de una forma especial si se detecta un error. Al producirse un error en el procedimiento marcado, o en cualquier otro procedimiento que sea invocado por éste, resulta en un retorno inmediato, prematuro, desde el procedimiento marcado.

El método normal de resolver los errores se inactiva entonces para el procedimiento marcado y Vd. podrá decidir cómo solucionarlo. Puede hallar el número del último error que se ha producido utilizando para ello la función **númerr()**. Puede emplear esta función para leer el número del error más de una vez, ya que el valor sólo se pone a cero la próxima vez que utilice el comando **error**. Si no se ha producido ningún error desde el comienzo del programa, o desde la última vez que utilizó **error**, entonces la función **númerr()** devolverá el valor cero.

Este método, si bien le resultará difícil de entender al principio, le proporciona un medio muy poderoso y flexible de solucionar los errores. En el ejemplo a continuación se muestra el empleo del comando **error**. Esto le provee un método "a prueba de errores" para introducir un número.

```
proc probar
  leer x
  finproc

proc comprobar
  haz n =1
  mientras n
    error probar
    haz n =númerr()
    si n
      escribir "Cometió el número de error " ;n ;", repita"
      finsi
    finmientras
  finproc
```

El primer procedimiento aguarda, simplemente, a que Vd. introduzca un valor para la variable x. El segundo procedimiento se encarga de los errores al ejecutar el procedimiento de leer. Si se produce un error en **probar**, terminará prematuramente este procedimiento y se pondrá el número del error. Este número se lee entonces por la función **númerr()** y, de no ser cero, se imprime el mensaje de error (este mensaje de error podrá ser, naturalmente, en la forma que a Vd. le parezca). Debido a que estas

sentencias están incluidas en un bucle **mientras finmientras**, si se produce un error hará que vuelvan a ejecutarse. El número del error se borra por el comando **error**, quedando listo para la próxima tentativa. No podrá Vd. salir de **comprobar** hasta que haya escrito un número válido.

Este ejemplo reporta el número del error que fue detectado. En la mayoría de las ocasiones no le interesará saber qué error ha ocurrido. El uso maestro de la función **númerr()** es para ver si no hay ningún error o si se ha detectado un error de cualquier tipo. En el capítulo titulado Guía de Consulta se incluye una lista de los números de error y las causas posibles.

Podemos ahora escribir un procedimiento que le permita elegir una de las seis opciones en la aplicación de la lista de direcciones, pulsando una sola tecla. Es lo suficientemente sencillo para no precisar de explicaciones.

```
proc elegir
   nota ***** elegir una opción *****
   limpiar
   escribir
   escribir " Insertar Despachar Pago Modif Cancelar Abandonar"
   escribir "? ";
   haz opción$ =mayúsc(tecla())
   escribir opción$
   si opción$ ="i": insertar : finsi
   si opción$ ="d": despachar : finsi
   si opción$ ="p": pago : finsi
   si opción$ ="m": modif : finsi
   si opción$ ="c": cancelar : finsi
   si opción$ ="a": adiós : finsi
   finproc
```

Todo lo que resta para completar nuestro programa es escribir un procedimiento de comienzo para abrir el fichero e invocar el procedimiento **elegir**. Debemos incluir **elegir** en un bucle para que vuelva a ofrecerle las opciones cada vez que Vd. complete la selección previa.

En el procedimiento a continuación, verá que el bucle **mientras finmientras** no se termina nunca. Este bucle sólo terminará cuando tenga un valor cero la expresión que sigue a **mientras**. En este procedimiento la expresión siempre tiene el valor 1, con lo cual continuará indefinidamente el bucle. La única forma de salir de este bucle es eligiendo la opción Abandonar (adiós). El comando **stop** en **adiós** devuelve inmediatamente el control al teclado.

```
proc comienzo
   nota ***** para comenzar proc *****
   limpiar
   abrir "nvocorreo.dbf"
   mientras 1
      error elegir
      haz n =númerr()
      si n
         escribir "Error - Pulse cualquier tecla para continuar"
         haz m$=tecla()
      finsi
   finmientras
finproc
```

Dentro de este bucle hay una secuencia de sentencias que se encarga de los errores, siguiendo un método similar al descrito en la sección previa. Si Vd. comete un error, no continuará el programa hasta que Vd. pulse una tecla. Esto le permitirá examinar lo que acaba de hacer para ver cómo cometió el error.

EL COMANDO EJECUTAR

Al procedimiento maestro en el programa de la lista de direcciones lo hemos llamado **comienzo**. Esto le permitirá utilizar el comando **ejecutar** al usar el programa.

Salve este procedimiento final con el nombre "listadirec". Cuando desee ejecutar el programa precisará cargar los procedimientos en la memoria del computador y luego ejecutar el procedimiento maestro, que invocará a todos los otros. Una forma de lograr esto es utilizando el comando **cargar** y luego escribiendo el nombre del procedimiento maestro. Por ejemplo:

```
cargar "listadirec"
comienzo
```

El comando **ejecutar** cargará un programa nombrado y luego ejecutará automáticamente el procedimiento llamado **comienzo** (si lo hay). Puede ejecutar el programa exactamente como en el ejemplo anterior, escribiendo simplemente:

```
ejecutar "listadirec"
```

Las dos secciones restantes de este capítulo incluyen unos procedimientos de uso general que podrán resultarle muy útiles.

VARIABLES LOCALES

La mayoría de las variables que aparecen en los procedimientos son *globales*. Esto significa que se reconocen en todo el programa. Pueden utilizarse o modificarse en cualquier procedimiento, en lugar de solamente en aquél en que primero se les asigna un valor.

Las variables que se emplean como parámetros de pura forma en un procedimiento son *variables locales*, ya que no se reconocen fuera del procedimiento en el cual se definen.

El ejemplo a continuación le aclarará la distinción entre variables globales y locales. Antes de continuar, escriba **nuevo** para borrar la memoria del computador. En primer lugar creamos un procedimiento que tiene dos variables locales, **a** y **b$**, además de asignar valores a dos variables normales (globales) **u** y **v$**.

```
proc demo; a,b$
   escribir a;b$
   haz u = 3
   haz v$ = "texto"
   escribir u;v$
   finproc
```

Luego utilizamos **demo**:

```
demo 5;"frase"
```

Aparecen entonces los cuatro valores, indicando que se reconocen las cuatro variables dentro de **demo**.

Si escribimos:

```
escribir u;v$
```

indicará que estas dos variables son también reconocidas fuera del procedimiento. Pero si escribimos:

```
escribir a;b$
```

resulta en un error, ya que estas dos variables no son reconocidas fuera de **demo**. Todos los parámetros de pura forma son variables locales, pero Vd. puede también declarar locales otras variables, como en el siguiente ejemplo:

```
proc momia
   escribir "dentro de momia"
   escribir p; q; r
   finproc

proc manta
   local q,r
   haz p = 2
   haz q = 3
   haz r = 4
   escribir "dentro de manta"
   escribir p; q; r
   momia
   finproc
```

Si trata de usar **manta** escribiendo:

```
manta
```

verá que los valores de 'p','q' y 'r' se reconocen todos (y por tanto se escriben) en **manta**, pero **momia** no reconoce los valores de 'q' y 'r', que son valores locales para **manta**.

Los valores de las variables locales no se definen más que en el procedimiento en el cual se declaran –ni siquiera en los procedimientos invocados por aquél. La variable 'p' es global y se reconoce en todas partes.

Podrá preguntarse la razón de que se precisen variables locales. Para ilustrar la utilidad de las mismas, supongamos que Vd. escribe un programa y éste contiene varios procedimientos que Vd., u otra persona, escribió originalmente para utilizarlos en otros programas. Es muy posible que dos o más de estos procedimientos tengan variables que, si bien tienen el mismo nombre, se emplean con fines totalmente distintos. Si estas variables fueran globales, un procedimiento podría alterar un valor de tal forma que fuera incorrecto para otro. En este caso, habría que verificar todos los procedimientos utilizados y, si procede, cambiar los nombres de las variables. Pero si las variables fueran locales, no importará que tengan el mismo nombre. A condición de que se encuentren en distintos procedimientos,si se cambia una no tendrá ningún efecto en la otra.

Además, no importa si un procedimiento invoca a otro que tiene una variable del mismo nombre –a condición de que una de ellas sea local. Por ejemplo, en el procedimiento **elegir**, en la sección titulada Errores de este capítulo, se declaró que la variable **opción$** era local. Esto significa que no hay necesidad de verificar si también se emplea **opción$** en cualquiera de los muchos procedimientos invocados por **elegir** –los procedimientos invocados no pueden alterar el valor de **opción$** en **elegir**.

MENSAJES

La presentación de un mensaje y el aguardar a que se pulse una tecla es una de las operaciones que más se precisan, mereciendo por tanto escribir para esto un procedimiento de uso general. Este procedimiento ha de permitir la indicación de una gran variedad de mensajes. Una forma muy sencilla de permitir que un procedimiento escriba (muestre en la pantalla) cualquier mensaje es pasando el mensaje al procedimiento en la forma de un parámetro.

```
proc mensaje; m$
   escribir m$+":";
   haz x$ =mayúsc(tecla())
   escribir x$
   finproc
```

El mensaje que va a presentarse en la pantalla se pasa al procedimiento como un parámetro en la variable local m$. La función **tecla()** aguarda a que se pulse una tecla y devuelve el código ASCII para la tecla. En este procedimiento, el código ASCII es convertido a mayúsculas por la función **mayúsc()**, para que el resultado no dependa de que sea en mayúsculas o minúsculas. Finalmente, el valor resultante se asigna a la variable x$. Esta es una variable global, con lo cual la tecla pulsada continuará disponible para cualquier otro procedimiento del programa.

PAUSA

Un procedimiento muy útil es el de **pausa**. Utiliza el procedimiento **mensaje** para presentar un mensaje y luego aguarda simplemente a que sea pulsada una tecla. Como a Vd. no le interesará generalmente conocer cuál fue la tecla pulsada, se emplea una variable local, y$, para conservar el contenido original de x$.

```
proc pausa
   nota ***** esperar por una tecla *****
   local y$
   haz y$ =x$
   escribir
   mensaje; "pulse cualquier tecla para continuar"
   haz x$ =y$
   finproc
```

INTRODUCCION DE DATOS

Texto

La aceptación de texto desde el teclado es muy sencilla. Cualquier conjunto de caracteres es una serie de texto válida (aun cuando no tenga ningún significado) y no causa un error del sistema. Normalmente no precisará tomar precauciones especiales al introducir texto. Generalmente, basta con una línea tal como la indicada a continuación, que le pide que escriba su nombre.

```
leer "Sirvase escribir su nombre: ";nombre$
```

Observe que se incluye un espacio como último carácter del texto del mensaje. Este pequeño detalle influye mucho en el aspecto del programa a la hora de utilizarlo.

Usted puede introducir varios elementos con una sentencia de leer. Todo lo que precisa hacer es incluir todos los mensajes y nombres variables, separados por el signo (;). Por ejemplo:

```
leer "Su nombre?";nombre$;"Su apellido?";apellido$;
```

Esta última sentencia de leer también termina con el signo (;) -esto impide que el cursor pase a la próxima línea después que Vd. haya escrito los datos de entrada.

Números

Al emplear el comando **leer** para introducir texto para una variable alfanumérica, el computador aceptará, sin quejarse, todo lo que Vd. escriba en el teclado. Pero si trata de introducir texto para una variable numérica, obtendrá un mensaje de error si escribe algo que no sea un número válido. Dado que Vd. no querrá salir del programa cada vez que le resbale el dedo al escribir un número, precisará cerciorarse de que el programa se las arregla con tales errores.

La forma más útil de lograr esto es utilizando el comando **error**, descrito anteriormente. El procedimiento que sigue, por ejemplo, aceptará cualquier número válido que esté dentro de una escala especificada. Produce incluso cualquier mensaje que a Vd. le interese presentar en la pantalla.

```
proc número; m$,min,máx
  nota ***** obtener número en la escala *****
  local mal
  haz mal=1
  mientras mal
    escribir m$; "? ";
    error leernúm
    haz mal=númerr()
    sino mal
        si núm<min oo núm>máx
      haz mal=1
      escribir "La escala permitida es "; min; " hasta "; máx
    finsi
  finsi
  si mal
    escribir "Vuelva a probar"
    finsi
  finmientras
finproc
```

Debido a que **error** debe ir seguido del nombre de un procedimiento, definimos ahora **leernúm** para introducir un valor para la variable **núm**.

```
proc leernúm
  leer núm
  finproc
```

Supongamos que desea usar un procedimiento para verificar si un número está dentro de la escala de 1 a 10. Podrá hacerlo con el procedimiento **número**, de la siguiente forma:

```
proc verificar
  número; "Valor numérico?",1,10
  finproc
```

CAPITULO 11
EMPLEO DE MULTIPLES FICHEROS

NOMBRES LOGICOS DE FICHERO

En este capítulo se complementa la explicación de cómo usar el lenguaje de programación de Archive, describiendo la forma de trabajar con dos o más ficheros abiertos. Al trabajar con más de un fichero abierto a la vez, Vd. debe poder identificar el fichero que desea utilizar para una operación dada. Para esto, debe dar a cada fichero un nombre lógico distinto al abrirlo o crearlo, refiriéndose entonces a ese nombre en todos los comandos relacionados con ese fichero.

Archive suministra automáticamente el *nombre lógico de fichero* "maestro" cuando Vd. abre un solo fichero. Se le llama nombre lógico para distinguirlo del *nombre físico del fichero* –el nombre que Vd. da al fichero al salvarlo.

Debido a que los programas se refieren a un fichero por su nombre lógico, Vd. puede escribir un programa que actúe con varios ficheros.

Los nombres lógicos de fichero son esenciales para operaciones con múltiples ficheros, ya que Vd. sólo puede abrir un segundo fichero utilizando su nombre físico y su nombre lógico. Observe que el nombre lógico no se salva con el fichero al cerrarlo y debe especificarse cada vez que se abre un fichero.

Dos o más ficheros de datos podrán contener campos que tienen el mismo nombre. Al ocurrir esto, Vd. puede identificar el fichero al cual pertenece el campo añadiendo el nombre lógico del fichero al nombre del campo. Por ejemplo, si el campo país$ aparece en dos ficheros cuyos nombres lógicos son "maestro" y "b", podrá Vd. referirse a cada uno de ellos con los nombres "maestro.país$" y "b.país$".

MODIFICACION DE LOS REGISTROS DE UN FICHERO

En nuestro primer ejemplo se indica la forma de añadir, borrar o renombrar los campos en un fichero existente.

Supongamos que desea hacer cambios en el fichero "gazet", para crear un nuevo fichero que sólo contenga los países de Europa. En este caso, no precisará incluir el campo continente$. Cambiaremos también el nombre del campo "hab" a "habitantes".

La forma más conveniente de modificar el fichero es creando un segundo fichero que contenga los campos que le interesan, para luego copiar los registros requeridos desde el fichero previo al nuevo fichero, que le llamaremos "europa". El resto del trabajo se efectuará con el siguiente procedimiento:

```
proc comienzo
nota ***** crear el fichero europa *****
  crear "europa" lógico "e"
    pais$
    capital$
    idiomas$
    moneda$
    habitantes
    rnb
    área
    fincrear
  ver "gazet" lógico "g"
  elegir continente$="EUROPA"
  todos "g"
    escribir en 0,0;g.pais$;tab 30
    haz e.pais$=g.pais$
    haz e.capital$=g.capital$
    haz e.idiomas$=g.idiomas$
    haz e.moneda$=g.moneda$
    haz e.habitantes=g.hab
    haz e.rnb=g.rnb
    haz e.área=g.área
    añadir "e"
    fintodos
```

```
cerrar "e"
cerrar "g"
escribir
escribir "HECHO"
finproc
```

EL FICHERO ACTUAL

Por el ejemplo anterior, verá que puede utilizar el mismo nombre para un campo en ambos ficheros –pueden distinguirse incluyendo el nombre lógico del fichero. Si Vd. no incluye el nombre lógico, se asume entonces que se utilizará el *fichero actual*. El último fichero que fue abierto se convierte automáticamente en el fichero actual. En este ejemplo, el fichero actual será "gazet" (con el nombre lógico "g"), con lo cual podremos utilizar este procedimiento, simplemente no escribiendo la g antes del nombre del campo en el programa anterior.

Si Vd. no incluye el nombre lógico del fichero en aquellos casos en que es opcional, Archive asumirá que el comando se refiere al fichero actual. Para evitar la posibilidad de confusiones, conviene incluir el nombre lógico del fichero.

Podrá, en cualquier momento, especificar el fichero actual mediante el comando **usar**. Si Vd. incluye el comando:

```
usar "e"
```

en el ejemplo anterior, entonces "europa" será el fichero actual hasta que vuelva a cambiarlo, ya sea abriendo otro fichero o con el comando **usar**.

CONTROL DE EXISTENCIAS

Vamos a ver ahora un ejemplo más complejo. En un sistema de control de las existencias, precisará:

Hallar información acerca de un artículo dado en existencia. Obtener un informe de los niveles actuales de las existencias de todos los artículos.
Anotar las ventas y modificar de conformidad los registros de existencias. Pedir nuevos suministros para mantener los niveles de las existencias.
Anotar las entregas de existencias recibidas.

Evidentemente, precisará un fichero para contener los detalles de todos los artículos en existencia y también resulta conveniente tener un segundo fichero para los detalles de todos los suministradores. Precisará tener acceso a uno de estos ficheros desde el otro –por ejemplo, podrá interesarle conocer todos los posibles suministradores de un artículo dado, o hallar los artículos que son suministrados por cierta compañía,

Para simplificar lo más posible la aplicación, no vamos a utilizar el enfoque a base de menús de los ejemplos dados en el capítulo anterior. En su lugar, escribiremos el programa en la forma de una serie de comandos independientes que puedan utilizarse –como los comandos normales– escribiendo sus nombres.

Debido a que los procedimientos dependerán en gran parte de la estructura de los ficheros que utilizamos, debemos primeramente considerar la forma que van a tener.

El fichero de existencias

El fichero de existencias debe contener todos los detalles referentes al nivel actual de cada artículo en existencia. En la lista siguiente se da la explicación de todos los campos que vamos a utilizar.

Nombre del campo	Aplicación	Ejemplo
`númstock$`	El código interno de existencias	`A101`
`descripión$`	La descripción del artículo	`Aparato, grande`
`cant`	La cantidad en existencia	`500`
`precven`	El precio de venta	`1.25`
`nivelped`	Hacer nuevo pedido cuando el nivel de las existencias llegue a este valor.	`200`
`cantcomp`	La cantidad que debe pedirse	`400`

Podemos crear el fichero con:

```
crear "stock" lógico "sto"
  númstock$
  descripción$
  cant
  nivelped
  precven
  cantcomp
  fincrear
```

El fichero de suministradores

Este fichero contiene los nombres, direcciones y números de teléfono de las compañías que suministran las mercancías que Vd. vende. También le resultará conveniente incluir el nombre de la persona con quien se hace contacto en la compañía. Para obtener acceso a esta información eficazmente, incluiremos un código para cada compañía. Utilizaremos aquí los siguientes campos:

Nombre del campo	Aplicación	Ejemplo
nombrcomp$	El nombre de la compañía	Eléctrica del Sur, S.A.
calle$	Primera línea de la dirección	Plaza Mayor 4
ciudad$	Segunda " " " "	Marbella
provincia$	Tercera " " " "	Málaga
distrpost$	Ultima " " " "	MA15
contacto$	Nombre de la persona a quien dirigirse.	Fermín Braña
tel$	Número de teléfono	952-820251 Ext 15
código$	El código que Vd. da a la compañia.	a

Podemos crear el fichero con:

```
crear "suminist" lógico "sum"
  nombrcomp$
  calle$
  ciudad$
  provincia$
  distrpost$
  contacto$
  tel$
  código$
  fincrear
```

El fichero de pedidos

Este fichero constituye el enlace entre los dos ficheros anteriores. Tiene los siguientes campos:

Nombre del campo	Aplicación	Ejemplo
númstock$	El código de existencias de Vd.	A101
código$	El código que Vd. da al suministrador	a
códsum$	El código del suministrador para el artículo	123-456
precio	El precio de venta del suministrador	0.87
entrega	El plazo de entrega del suministrador (en días)	28

Cada registro de este fichero enlaza un registro en el fichero de existencias con un registro en el fichero de suministradores. En el ejemplo anterior se muestra que Eléctrica del Sur (código de suministrador "a") puede suministrarle aparatos grandes (código de stock "A101"). Además, también se incluyen detalles del precio, plazo de entrega y el código de existencias del suministrador. Estos datos resultan útiles a la hora de hacer más pedidos.

Este fichero le permitirá tener en cuenta los casos en que un suministrador provee más de un artículo (los mismos valores para código$, pero distintos valores para númstock$) y aquellos en que puede obtenerse un artículo de varios suministradores (igual númstock$ pero distinto código$).

Puede crear ahora el fichero con:

```
crear "pedidos" lógico "ped"
  númstock$
  código$
  códsum$
  precio
  entrega
  fincrear
```

Indagaciones

Verá que lo que se precisa con mayor frecuencia es hallar información acerca de un artículo dado de las existencias, como resultado de indagaciones de los clientes. Precisará hallar la información con la mayor rapidez posible, pero podrá tener que hallar un registro dado a partir del número de referencia del artículo o de su descripción. Por tanto, utilizaremos el comando **hallar**, que le permitirá dar un texto válido para comenzar la búsqueda.

El procedimiento debe pedirle que Vd. confirme que se trata del registro que desea hallar. Esta tarea vamos a delegarla a otro procedimiento, ya que pudiéramos desear utilizarlo en otros casos.

```
proc confirmar
  escribir : escribir "Confirmar (s/n)";
  haz si=minúsc(tecla())="s"
  finproc
```

Esto hace que la variable **sí** contenga 1 si Vd. pulsa la tecla S -en caso contrario el valor es cero. Observe el empleo del signo = para asignar el valor y también para la condición lógica.

```
proc indagar
  nota ***** consultas de articulos en stock *****
  escribir
  leer "Articulo Stock? "; nombre$
  usar "sto"
  hallar nombre$
  haz si=0
  mientras hallado() yy no si
    indicar
    pescribir
    confirmar
    si no si
      continuar
      finsi
    finmientras
  si no hallado()
    escribir
    escribir nombre$; " no existe"
  finsi
finproc
```

Este procedimiento permite, simplemente, localizar el registro correcto. Un procedimiento más útil para interrogar el fichero de existencias es **consulta**:

```
proc consulta
  indagar
  limpieza
  finproc
```

Se utiliza aquí otro procedimiento, **limpieza**, que aguarda a que Vd. pulse una tecla, limpia la pantalla y escribe entonces una lista de los comandos que Vd. puede utilizar. Vamos a dejar este procedimiento hasta que hayamos escrito los procedimientos que precisa listar. Acuérdese de salir de **editar** de vez en cuando para salvar estos procedimientos al introducirlos.

Informe de existencias

También podemos escribir un procedimiento sencillo para obtener un informe general de las existencias.

```
proc informe
  nota ***** informe de existencias *****
  limpiar
  escribir tab 2; "ARTICULO"; tab 11; "CODIGO";
  escribir tab 20; "CANTIDAD"; tab 31; "PRECIO";
  escribir tab 40; "VALOR DEL STOCK"
  escribir
  haz total=0
  usar "sto"
  todos
    escribir descripción$( hasta 10); tab 11; sto.númstock$;
      tab 20;cant;
    escribir tab 31;"$";precven; tab 40;"$";precven*cant
    haz total=total+precven*cant
    fintodos
  escribir
  escribir "Valor total del stock =$"; total
  limpieza
  finproc
```

Anotación de las ventas

Todo lo que precisamos hacer para anotar una venta es sustraer el número de artículos vendidos, del correspondiente registro de existencias. Es aconsejable incluir una confirmación de que nos estamos refiriendo al artículo correcto y que el stock (existencias) es suficiente para cumplimentar el pedido.

```
proc cantidad
  nota ***** imprimir articulos en stock *****
  indagar
  limpiar
  leer "Cuántos? "; núm
  escribir
  limpiar
  escribir núm; " * "; sto.númstock$;" ("; sto.descripción$;")"
  finproc

proc ventas
  nota ***** procesar la venta *****
  cantidad
  si núm<=sto.cant
    escribir "Valor del pedido:- $"; núm*sto.precven
    confirmar
    si si
      haz sto.cant=sto.cant-núm
      actualizar
      pescribir: nota *** mostrar el registro modificado ***
      finsi
    sino
    escribir "Stock insuficiente"
    finsi
  limpieza
  finproc
```

Anotación de las existencias recibidas

El procedimiento que sigue le permitirá anotar las existencias que recibe. Este procedimiento también solicita confirmación de los detalles que Vd. escribe antes de aceptarlos y actualizar el correspondiente registro de existencias.

```
proc entrega
  nota ***** existencias recibidas *****
  cantidad
  confirmar
  escribir
  si si
    escribir "Aceptado"
    haz sto.cant=sto.cant+núm
    actualizar
```

```
    pescribir
    sino
    escribir "Entrega no anotada"
    finsi
  limpieza
  finproc
```

Nuevos pedidos de existencias

Hasta aquí, nuestros procedimientos solamente se refieren al fichero de existencias. Cuando deseemos pedir más existencias, tendremos que referirnos a los ficheros de suministradores y de pedidos para hallar el nombre y dirección de la compañía, el precio, etc.

Suponiendo que ya hemos localizado el artículo en el fichero de existencias (con **indagar**), seleccionaremos entonces, del fichero de pedidos, aquellos registros que tienen el código de stock correcto. Estos registros contienen los códigos para todas las compañías que pueden suministrar el artículo que nos interesa. Puesto que los registros también contienen el precio y plazos de entrega para cada suministrador, podremos decidir si nos interesa más el artículo más barato o el plazo de entrega más corto.

Utilizaremos **situar** para hallar rápidamente el registro del suministrador requerido. Esto significa que el fichero de suministradores debe estar ordenado (respecto al código del suministrador –código$) antes de que podamos utilizar el procedimiento **hacerped**.

```
proc hacerped
  nota ***** pedir nuevo stock *****
  indagar
  usar "ped"
  elegir sto.númstock$=ped.númstock$
  escribir
  escribir "rápido o barato (r/b)";
  si minúsc(tecla())="r"
    rápido
    sino : barato
    finsi
  haz micód$=códsum$
  restaurar
  usar "sum"
  situar comp$
  impreso
  escribir
  escribir "Entrega prevista "; ent; " dias"
  limpieza
  finproc
```

El procedimiento **barato** halla el suministrador que ofrece el precio más bajo, mientras que **rápido** halla el suministrador con el plazo de entrega más corto.

```
proc barato
  nota ***** hallar el más barato *****
  usar "ped"
  haz pre=precio
  haz comp$=código$
  haz ent=entrega
  todos
    si precio<pre
    haz pre=precio
    haz comp$=código$
    haz ent=entrega
    finsi
  fintodos
  finproc

proc rápido
  nota ***** entrega rápida *****
  usar "ped"
  haz ent=entrega
  haz comp$=código$
```

```
haz pre=precio
todos
  si entrega<ent
    haz ent=entrega
    haz comp$=código$
    haz pre=precio
    finsi
  fintodos
finproc
```

El procedimiento **impreso** produce el formulario para hacer el pedido. Usted debe modificarlo conforme a sus propias necesidades. Vamos a utilizar aquí una versión sencilla que muestra en la pantalla los detalles del pedido.

```
proc impreso
  nota ***** hacer formulario de pedido *****
  limpiar
  escribir
  escribir sum.nombrcomp$
  escribir sum.calle$
  escribir sum.provincia$
  escribir sum.distrpost$
  escribir
  escribir "Sirvase suministrar "; sto.cantcomp;
  escribir " * número de ref ";
  escribir micód$
  escribir "("; sto.descripción$; ") ";
  escribir "a $"; pre; " cada."
  escribir
  escribir "Valor total: $"; sto.cantcomp*pre
  finproc
```

Finalmente, precisamos un comando para cerrar todos los ficheros cuando hayamos terminado con ellos.

```
proc adiós
  confirmar
  si si
    limpiar
    escribir : escribir "adiós"
    cerrar "sto"
    cerrar "sum"
    cerrar "ped"
    limpiar
    finsi
  finproc
```

Podemos escribir ahora un procedimiento corto para ejecutar esta aplicación. Este procedimiento debe abrir los tres ficheros con los nombres lógicos correctos, limpiar la pantalla y mostrar los comandos adicionales que tiene a su disposición. Observe que, normalmente, el fichero de existencias es el único cuyos registros precisarán modificarse. Los otros dos ficheros se abren para lectura solamente. También ordena (clasifica) el fichero de suministradores para que podamos **situar** (localizar) una compañía por su código de referencia.

```
proc comienzo
  limpiar
  escribir en 5,5; "DEMOSTRACION DEL CONTROL DE EXISTENCIAS"
  escribir
  abrir "stock" lógico "sto"
  ver "suminist" lógico "sum"
  ver "pedidos" lógico "ped"
  usar "sum"
  ordenar código$; a
  limpieza
  finproc
```

Finalmente, podemos escribir el procedimiento **limpieza**, que limpia la pantalla y muestra una lista de los comandos adicionales que están disponibles:

```
proc limpieza
   nota ***** limpiar pantalla y ver comandos *****
   local x$
   escribir
   escribir "Pulse cualquier tecla para continuar ";
   haz x$=tecla()
   limpiar
   escribir
   escribir "consulta informe entrega hacerped ventas adiós"
   escribir "Escriba su opción"
finproc
```

CAPITULO 12
GUIA DE CONSULTA PARA QL ARCHIVE

VARIABLES

Los nombres de variables pueden tener hasta trece caracteres y no deben comenzar con un dígito (0 a 9). Pueden contener cualquier combinación de caracteres alfabéticos en mayúsculas o minúsculas, así como dígitos. No se admiten otros caracteres, excepto "$" y ".", que tienen significados especiales. Si un nombre de variable termina con el signo "$" es una variable alfanumérica. Estas variables pueden tener hasta 255 caracteres. Si el nombre no termina con el signo $ la variable es numérica. Un nombre de variable puede referirse al contenido de un registro en un fichero y se conoce entonces como una variable de campo. Las variables de campo se asume normalmente que se refieren al fichero actual, pero puede hacerse que se refieran a otro fichero abierto incluyendo el nombre lógico del fichero, separado del nombre de la variable por un punto (.). Esta variable de campo se escribe en el formato:

nombre_lógico.nombre del_campo

Por ejemplo:

```
maestro.continente[
```

Si un nombre de variable incluye un punto, se refiere entonces a un campo en un fichero abierto. Si no tiene un punto, el programa trata entonces de casar el nombre con una variable existente, por el siguiente orden:

1) un campo del fichero actual
2) una variable local (un parámetro en el procedimiento actual, si lo hay)
3) una variable global

El programa presenta un mensaje de error si no encuentra una equivalencia.

SINTAXIS

El vocablo *sintaxis* se refiere a la estructura exacta de un comando o función. La sintaxis de un comando, por ejemplo, especifica los parámetros que precisa el comando, el orden en que deben aparecer y los símbolos utilizados (si procede) para separarlos.

En esta sección se describe la numeración escrita (notación) utilizada para expresar la sintaxis del lenguaje de programación de Archive.

EXPRESIONES

Una *expresión* es una combinación de valores literales, variables, funciones y operadores que resulta en un valor. Una *expresión numérica* da como resultado un valor numérico, mientras que una *expresión alfanumérica* da como resultado un valor de texto. Por ejemplo:

```
3 * y * seno (x) + long (a$)        {numérica}
"abc" + a$ + rept (" - ", 5)        {alfanumérica}
```

Como se indica en estos ejemplos, una expresión puede contener varias sub-expresiones. En ese caso, no se pueden mezclar sub-expresiones de distintos tipos. Deben ser todas alfanuméricas o todas numéricas.

Convenios de sintaxis

Las definiciones de sintaxis son similares a las utilizadas para definir la sintaxis de SuperBASIC.

Símbolo	Significado
itálica	indica una entidad sintáctica
[]	incluye un parámetro opcional
**	incluye parámetros que pueden repetirse
|	o bien
{}	comentario

Entidades sintácticas

s.lit	serie literal
s.exp	expresión alfanumérica
n.exp	expresión numérica
exp	expresión, alfanumérica o numérica
ptm	parámetro de impresión
var	nombre de variable, alfanumérica o numérica
lfn	nombre lógico de fichero
fnm	nombre físico de fichero (máximo de 8 caracteres)
pnm	nombre de procedimiento

Una serie literal es texto entre comillas o apóstrofes; por ejemplo "texto" o 'texto'.

Una expresión alfanumérica es una serie literal, o una combinación de series literales, variables alfanuméricas y funciones alfanuméricas que resulta en un valor de texto. Por ejemplo:

```
"paco"+a$+car(72)
```

Una expresión numérica es un número, o una combinación de números, variables numéricas y operadores (+, −, *, /, etc.) que resulta en un valor numérico. Por ejemplo:

```
(3+x)/seno(y)
```

Un parámetro de impresión es una de cuatro posibilidades: **en, tab, tinta, papel**. En nuestra notación de sintaxis, la descripción completa de un parámetro de impresión es:

parám_impres := | **en** *n.exp,n.exp*
| **tab** *n.exp*
| **tinta** *n.exp*
| **papel** *n.exp*

Los nombres lógicos de fichero y los nombres de procedimientos tienen las mismas restricciones que los nombres de variables. Los nombres físicos de fichero no deben además exceder de 8 caracteres.

Como ejemplo de una *definición de sintaxis*, veamos la sintaxis del comando ordenar. En nuestra notación, aparece en la forma:

espec ordenar:= var; a | d
ordenar *espec ordenar* *[,*espec ordenar*]*

Por tanto, **ordenar** precisa ir seguido de una especificación de ordenar como mínimo, que a su vez consiste en una *variable* separada por el signo (;) de una letra que debe ser **a** o **d**. Además, Vd. podrá opcionalmente incluir hasta otras 3 especificaciones, a condición de que separe cada par con comas. Evidentemente, la notación de sintaxis provee una descripción mucho más compacta.

Tenga en cuenta que la notación de sintaxis no le indica el significado o la finalidad de los símbolos –precisará leer el resto de la descripción para cada comando. La sintaxis solamente le da una descripción de la forma y la clase de parámetros que componen un comando válido. La notación de sintaxis tampoco le indica el máximo admisible de repeticiones para los parámetros que se repiten. **Ordenar** acepta hasta cuatro pares de variable y letra.

FICHEROS DE DATOS DE ARCHIVE

Campo

Un *campo* es el espacio reservado para contener una serie o un número.

En Archive, cada campo se identifica por un nombre de variable. El hecho de que un campo pueda contener una serie o un número depende del nombre dado al campo al crearlo –los campos de series (alfanuméricos) tienen un nombre que acaba con $. Un campo alfanumérico de Archive puede contener un máximo de 255 caracteres. Un campo numérico tiene un nombre que no acaba con $. Todos los números se almacenan en la misma cantidad de espacio, independientemente del valor que tengan. La escala posible para un número es la misma que la escala numérica válida para los operadores aritméticos.

Registro

Un *registro* es una colección de campos cuyo contenido está relacionado de alguna forma. Los campos de un registro pueden, por ejemplo, emplearse para contener el nombre, la dirección y el número de teléfono de una persona. En Archive, los registros son de *longitud variable*, con lo cual cada registro solamente ocupa el espacio necesario para almacenar la información contenida en sus campos. Un registro de Archive puede tener un máximo de 255 campos.

Fichero

Un *fichero de datos* está formado por varios registros relacionados. Continuando con el ejemplo anterior, un fichero de datos podrá constar de una colección de registros de nombre, dirección y número de teléfono para muchas personas. El número de registros en un fichero de datos de Archive está limitado a unos 15000 registros, pero en la práctica este límite depende de la capacidad de un cartucho del Microdrive, que puede contener unos 1000 registros de 100 caracteres. El fichero es la unidad básica que Vd. puede salvar a un cartucho del Microdrive o cargar desde el mismo. Cada fichero tiene un nombre para identificarlo. En Archive, Vd. da un nombre físico al fichero al crearlo, pero puede cambiar el nombre lógico en cualquier momento.

Operaciones de abrir y cerrar ficheros

Cuando desee leer de un fichero de datos o escribir en el mismo, debe primeramente *abrirlo*. La operación de abrir un fichero de datos transfiere una copia de parte del mismo, desde el cartucho en el Microdrive a la memoria, si bien en el caso de un fichero muy largo, es posible que sólo se encuentre en la memoria parte del fichero en un momento dado.

Puede abrir un fichero de datos en el modo de *lectura solamente* (con **ver**), que, como su nombre sugiere, no le permite cambiar el contenido del fichero. También tiene la opción de abrir un fichero de datos en el modo de *actualizar* (con el comando **abrir**), pudiendo entonces leerlo y alterar su contenido.

Cada vez que Vd. abre un fichero de datos, Archive reserva espacio para las variables de campos que se precisan para un registro del fichero. Las variables de campos siempre contienen los valores del registro actual en el fichero.

Al cerrar un fichero de datos con **cerrar** o **abandonar** los cambios que Vd. ha hecho se copian al fichero almacenado en el cartucho del Microdrive, desechándose la copia contenida en la memoria. El cierre de un fichero es la única forma de asegurar que la copia en el cartucho contiene las modificaciones más recientes. Debido a que un fichero abierto utiliza parte de la memoria del computador, Vd. no debe dejar ficheros abiertos cuando no los utiliza.

Al salir de Archive mediante el comando **abandonar**, se cierran automáticamente todos los ficheros que están abiertos.

No apague el computador, ni extraiga un cartucho del Microdrive, mientras el cartucho contenga ficheros abiertos.

Nombres lógicos de fichero

Cada fichero de datos abierto recibe un *nombre lógico* al abrir el fichero. Si Vd. no especifica un nombre lógico al abrir el fichero, Archive le da automáticamente el nombre lógico "maestro".

El nombre lógico del fichero se emplea para identificar un fichero dado cuando se utilizan varios ficheros a la vez.

PROCEDIMIENTOS

Un procedimiento es una sección nombrada de un programa, que comienza con una declaración de la siguiente forma:

proc *pnm*[; *var* *[, *var*]*]

y termina con:

finproc

Puede hacerse referencia al mismo por su nombre desde cualquier otro programa o procedimiento, incluso desde si mismo. Actúa como si se hubiera insertado su código en el punto desde el cual se invoca.

En Archive, los comandos **proc** y **endproc** no pueden introducirse directamente desde el teclado, pero se añaden automáticamente cuando Vd. utiliza el editor de programas para crear un procedimiento.

EL EDITOR DE PROGRAMAS

El editor de programas se introduce con el comando **editar**.

Si no hay ningún procedimiento en la memoria, Archive le ofrece inmediatamente la opción de crear un nuevo procedimiento. En caso contrario, le provee una lista de todos los procedimientos que se encuentran en la memoria, en el lado izquierdo de la zona de presentación. El primer procedimiento aparece resaltado y se lista completo a la derecha de la pantalla. La primera línea del procedimiento también aparece resaltada para marcar el procedimiento actual y la línea actual del procedimiento.

Una vez que se encuentre en **editar** tiene las cinco opciones siguientes:

Seleccionar un procedimiento
Pulse **TAB** para recorrer hacia abajo la lista de procedimientos, o las teclas **SHIFT** y **TAB** para recorrer la lista hacia arriba. En la lista de la pantalla siempre se muestra el procedimiento actual.

Seleccionar una línea
Utilice las teclas "arriba" y "abajo" del cursor para seleccionar una línea del procedimiento actual. Se resalta la línea actual.

Pulsar F3 para el menú de comandos de edición
Hay cuatro comandos, que se seleccionan pulsando la tecla correspondiente a su primera letra.

Borrar Pulse [↵] para borrar el procedimiento resaltado a la izquierda de la pantalla. Pulse cualquier otra tecla para salir del comando sin borrar el procedimiento.

Nuevo Escriba el nombre del nuevo procedimiento y pulse [↵]. Si ya existe un procedimiento con ese nombre, el comando le ofrece la oportunidad de editarlo.

Quitar Retira texto del procedimiento actual y lo transfiere a la memoria de edición. Antes de invocar este comando, utilice las teclas "arriba" o "abajo" del cursor para hacer que la primera (o la última) línea de la zona que va a retirarse sea la línea actual. Utilice entonces las teclas "arriba" y "abajo" del cursor para marcar la zona de texto que desea retirar. Pulse [↵] para pasar el texto a la memoria de edición.

Mezclar Copia el contenido de la memoria de edición al procedimiento actual, bajo la línea actual. Al utilizar **mezclar** se borra la memoria de edición.

Insertar texto
Pulse **F4** para insertar una o más líneas de texto bajo la línea actual en el procedimiento actual. Escriba el texto y pulse [↵]. Si pulsa [↵] sin haber escrito ningún texto, saldrá de la opción de insertar.

Editar texto
Pulse **F5** para editar la línea actual del procedimiento actual. La línea de texto se copia a la línea de entrada y podrá entonces modificarla con el editor de línea. Pulse [↵] para sustituir la línea previa por la nueva.

EL EDITOR DE LA PANTALLA

El editor de la pantalla se invoca con el comando **peditar**. Le permite diseñar un nuevo formato de la pantalla, o modificar uno existente. Después de diseñar un nuevo formato, podrá salvarlo al cartucho con el comando **psalvar**, para volver a cargarlo posteriormente desde el cartucho con el comando **pcargar**.

El formato de la pantalla puede considerarse que consta de dos partes –el texto fijo de fondo y los valores variables que aparecen en el mismo. El comando **pantalla** muestra el fondo en la pantalla, mientras que **pescribir** añade los valores actuales de las variables que contiene.

Peditar le ofrece dos opciones:

escribir texto en el fondo de la pantalla
pulsar **F3** para usar el comando de editar la pantalla.

Después de pulsar **F3** tiene a su disposición cuatro comandos para modificar la pantalla:

L – Limpiar la pantalla
V – Marcar una zona para mostrar una Variable
T – Seleccionar el color de la Tinta
P – Seleccionar el color del Papel

El formato de la pantalla se activa con:

pcargar
pantalla

Al estar activado un formato dado de la pantalla, muestra los valores actuales de sus variables después de un comando **pescribir**, o cuando se devuelve el control al teclado después de ejecutar un programa (o un comando). El formato de la pantalla se desactiva borrando (limpiando) la pantalla con **limpiar**. Si no está activado ningún formato, el comando **pescribir** no surte efecto. Sólo puede haber un formato de la pantalla a la vez en la memoria del computador.

El comando **indicar** crea y utiliza su propio formato de la pantalla. Por tanto, sustituirá a otro formato existente.

LOS COMANDOS

Archive le ofrece los siguientes comandos:

TODOS

Explora todos los registros presentes, lógicamente, en el fichero, en el tiempo más corto posible.

Sintaxis: **todos** [*lfn*]:...:**fintodos**

En general, esta exploración no se hace por un orden determinado. El nombre lógico del fichero, opcional, hace que **todos** se refiera a un fichero abierto especificado. Al no dar el nombre lógico del fichero, explora el fichero actual.

El bucle **todos** sirve para aquellos casos en que Vd. desea examinar los registros de un fichero, en lugar de modificarlos. No utilice **actualizar** en un bucle **todos**, a no ser que esté seguro de que no cambiará la longitud del registro. Podrá, por ejemplo, cambiar el valor de un número o convertir un campo de texto a mayúsculas. En caso de duda, utilice un bucle **mientras** –empleando la función **finf()** para detectar el fin del fichero. Por ejemplo:

```
primero
mientras no finf()
   ...
   actualizar
   ...
   próximo
   finmientras
```

ALTERAR

Altera el formato actual de la pantalla para presentar los valores actuales de las variables.

Sintaxis: **alterar**

Usted puede alterar el contenido de uno o más campos del fichero actual, cuyos valores aparecen en el formato de la pantalla. Tenga en cuenta que podrán no mostrarse todas las variables de los campos. No se puede cambiar un campo que no aparezca en la pantalla. Si no aparece en la pantalla ninguna de las variables de los campos, Archive hace que se **indique** el fichero.

Seleccione primeramente el campo que desea alterar, pulsando **TAB** o [↵] hasta dejar el cursor en el campo correcto (se saltan aquellas variables que no sean campos del fichero). Podrá entonces escribir un nuevo valor o utilizar el editor de línea para modificar el valor existente. Pulse **TAB** o [↵] para pasar al próximo campo. (Pulsando juntas las teclas **SHIFT** y **TAB** se retrocede al campo anterior).

Cuando haya hecho todos los cambios que desea, pulse **F5** para sustituir el registro previo por el nuevo. Pulsando [↵] al final del último campo se sustituye automáticamente el registro. Si el fichero está ordenado, la nueva versión del registro se inserta por orden.

AÑADIR

Añade un registro al fichero especificado, o al fichero actual si Vd. no da el nombre lógico del fichero.

Sintaxis: **añadir** [*lfn*]

Los campos del registro toman los valores actuales de las variables de los campos. Si está ordenado el fichero, la inserción se hace por orden.

ANTERIOR

Retrocede un registro en el fichero especificado, o en el fichero actual si Vd. no da el nombre lógico del fichero.

Sintaxis: **anterior** [*lfn*]

SALVAGUARDAR

Hace una copia del fichero especificado. Le convendrá hacer copias de todos sus ficheros, para el caso de que se dañen o se borren por error.

Sintaxis: **salvaguardar** previofnm **como** nuevofnm

CERRAR

Cierra el fichero especificado, o el fichero actual si Vd. no da el nombre lógico del fichero.

Sintaxis: **cerrar** [*lfn*]

LIMPIAR

Limpia la zona de presentación y desactiva el formato de la pantalla. Vea los comandos **pantalla**, **pcargar** y **pescribir**.

Sintaxis: **limpiar**

CONTINUAR

Continúa la operación previa de **buscar** o **hallar**, desde el registro que sigue al registro actual en el fichero actual.

Sintaxis: **continuar**

CREAR

Crea un fichero abierto nombrado, cuyos registros contienen los campos dados en la lista de variables especificada en el comando. Usted tiene la opción de especificar un nombre lógico para el fichero –si no lo hace, el fichero se crea con el nombre lógico "maestro".

Sintaxis: **crear** *fnm* [**lógico**:*lfn*]: *var* *[: *var*]* : **fincrear**

BORRAR

Borra el registro actual del fichero especificado, o del fichero actual si Vd. no da un nombre lógico para el fichero.

Sintaxis: **borrar** [*lfn*]

Tenga cuidado al emplear este comando, ya que no podrá recuperar el registro borrado.

DIR

Presenta una lista (directorio) de los ficheros en el cartucho del Microdrive.

Sintaxis: **dir** [*Microdrive*]

Puede especificar el Microdrive **mdv1** o **mdv2**. Si Vd. no incluye el nombre del Microdrive, Archive lista automáticamente los ficheros en el cartucho del Microdrive 2.

Antes de mostrar la lista de los ficheros, Archive indica el nombre de volumen del cartucho (el nombre que Vd. le dio al formatearlo).

INDICAR

Muestra el nombre lógico del fichero actual y una lista de los nombres de los campos y los valores de las variables de los campos para el registro actual. Si está ordenado el fichero, muestra también los campos de clasificación y su prioridad.

Sintaxis: **indicar**

Este comando sustituye el formato de la pantalla existente –definido por Vd.– por esta lista, que pasa entonces a ser el formato activo.

VOLCAR

Sintaxis: **volcar** [;*var*]*[,*var*]*

Imprime en la impresora los campos especificados de los registros seleccionados del fichero actual, en forma de tabla. Si Vd. no da una lista de las variables de los campos, se imprimen *todos* los campos.

Podrá dirigir la salida a un fichero del Microdrive con el comando **vía**.

EDITAR

Invoca el editor de procedimientos para crear un nuevo procedimiento o para modificar uno existente.

Sintaxis: **editar**

FINTODOS

Vea **todos**.

FINCREAR Vea **crear**.

ERROR Marca un procedimiento para tener en cuenta los errores. Si se produce un error al ejecutar este procedimiento, o cualquier otro procedimiento invocado por éste, causa el regreso prematuro desde el procedimiento marcado. El procedimiento puede determinar la naturaleza del error utilizando la función **númerr()** para leer el número del error. Este número de error se borra cada vez que se ejecuta el comando **error**.

Sintaxis: **error** *pnm*[;*exp* * [,*exp*] *]

EXPORTAR Salva a un cartucho los campos nombrados en los registros seleccionados del fichero actual de Archive, en una forma adecuada para importarlos a QL Abacus o QL Easel.

Sintaxis: **exportar** *fnm* [; *var*] *[,**var**] * [**quill**]

Si Vd. no especifica una lista de variables de los campos, se exportan *todos* los campos. Si incluye el parámetro opcional**quill** (separado por un espacio como mínimo del último nombre de la variable), el fichero se exporta en una forma que sirva para importarlo en QL Quill.

El fichero de exportar se nombra *fnm* y, a no ser que Vd. especifique una extensión para el nombre del fichero, Archive utiliza la extensión **_exp**.

Consulte la *sección de Información* para la explicación detallada de las operaciones de Importar y Exportar.

HALLAR Comienza al principio de un fichero y busca el primer registro que contenga el texto especificado en cualquiera de los campos de texto. Esto es independiente de si el texto es en mayúsculas o en minúsculas.

Sintaxis: **hallar** *s.exp*

Podrá continuar la búsqueda con el comando **continuar**, pudiendo también ver si la búsqueda ha dado resultado examinando el valor devuelto por la función **hallado()**.

PRIMERO Selecciona el primer registro del fichero especificado, o del fichero actual si Vd. no da un nombre lógico para el fichero.

Sintaxis: **primero** [*lfn*]

FORMATEAR Formatea el cartucho en el Microdrive 2 (La unidad del lado derecho). Da al cartucho el nombre especificado por Vd. Este nombre se reporta cuando Vd. utilice posteriormente **dir** para ver el directorio de los ficheros en ese cartucho.

Sintaxis: **Formatear** *nombre dado por Vd.*

SI Permite que una condición especificada controle el proceso siguiente.

Sintaxis: **si** *n.exp* : ... [: **sino** : ...] : **finsi**

Sin el parámetro opcional **sino**: Si la expresión no es cero, se ejecutan las sentencias a continuación. Si la expresión es cero, se transfiere la ejecución a la sentencia que sigue a **finsi**.

Con el parámetro opcional **sino**: Si la expresión numérica no es cero, se ejecutan las sentencias entre **si** y **sino**. En caso contrario, se ejecutan las sentencias entre **sino** y **finsi**. En ambos casos, la ejecución continúa con las sentencias que siguen a **finsi**.

IMPORTAR Lee un fichero, *nombre1*, exportado desde QL Abacus o QL Easel, y crea un fichero de datos de Archive, *nombre2*. Como ocurre con los comandos **abrir** y **ver**, Vd. tiene la opción de especificar un nombre lógico para el fichero de datos.

Sintaxis: **importar** *nombre1* **como** *nombre2* [**lógico***lfn*]

donde: *nombre1*:= *fnm*
nombre2:= *fnm*

Consulte la *sección de Información* para la explicación detallada de las operaciones de Importar y Exportar.

TINTA

Selecciona el color de primer plano para todo el texto a continuación, al color especificado en el valor de la expresión.

Sintaxis: **tinta** *n.exp*

Los colores son: 0 y 1 Negro
2 y 3 Rojo
4 y 5 Verde
6 y 7 Blanco

Si el resultado de la expresión es mayor que 7, el valor tomado es el resto después de dividir por 8. Por ejemplo, **tinta** 9 es equivalente a **tinta** 1 (ambos seleccionan el color negro). Si utiliza tinta en un comando **escribir**, el color sólo cambia mientras dure ese comando.

LEER

Solicita que se introduzcan (lean) datos desde el teclado para las variables listadas en el comando. Cada variable en una lista puede ir precedida de una serie de texto inicial que se presenta como un mensaje para la operación de leer. Todos los valores introducidos deben separarse con signos (;). Si la lista termina con un signo de punto y coma, el cursor no pasa a una nueva línea después de introducir los datos.

Sintaxis: **leer** [*var* | *s.lit* | *ptm* *[; *var* | *s.lit* | *ptm*]*];]

La lista de datos leídos puede incluir los siguientes parámetros para posicionar el cursor:

en *línea,columna*
tab *columna*

donde: *línea*:=*n.exp*,
columna:=*n.exp*

El primero de éstos posiciona el cursor en la posición de línea y columna especificada, mientras que **tab** mueve el cursor a la columna especificada de la línea actual. Si el cursor ya ha sobrepasado de la columna especificada, tab no tiene ningún efecto.

Estos dos parámetros no pueden utilizarse fuera de un comando **leer**, o **escribir**.

Puede también utilizar **tinta** y **papel** formando parte del comando **leer**. En este caso sólo afectan a los colores de la tinta y del papel hasta el final de dicho comando, en cuyo momento vuelven los colores a los valores que tenían.

INSERTAR

Añade un nuevo registro a un fichero.

Sintaxis: **insertar**

Utiliza el formato actual de la pantalla para mostrar los valores actuales de las variables. Usted puede escribir un nuevo valor para uno o más campos del fichero actual, cuyos valores aparecen en la pantalla. Tenga en cuenta que podrán no mostrarse todas las variables de los campos. No puede escribir un nuevo valor para un campo que no se muestra. Si no aparece en la pantalla ninguna de las variables de los campos, Archive hace que se **indique** el fichero.

Seleccione primeramente un campo pulsando **TAB** o [↵] hasta dejar el cursor en el campo correcto (se saltan los valores que no son campos del fichero). Escriba entonces un nuevo valor. Pulse **TAB** o [↵] para pasar al próximo campo. (Pulsando juntas las teclas **SHIFT** y **TAB** regresará al campo previo).

Cuando haya escrito todos los valores que desea, pulse **F5** para añadir el nuevo registro al fichero. También se añade el registro al fichero si pulsa [↵] al estar el cursor en el último campo. Los campos a los que Vd. no haya dado un valor serán cero (si se trata de un campo numérico) o una serie de texto vacía (si es un campo de texto). Si está ordenado el fichero, la nueva versión del registro se inserta en secuencia. En caso contrario, se inserta en una posición indeterminada.

TIRAR

Borra el fichero especificado del cartucho en el Microdrive.

Sintaxis: **tirar** *fnm*

Tenga cuidado al utilizar este comando, ya que no podrá recuperar el fichero borrado.

ULTIMO

Selecciona el último registro del fichero especificado, o del fichero actual si Vd. no da un nombre lógico para el fichero.

Sintaxis: **último** [*lfn*]

HAZ Sirve para asignar un valor a una variable (igual que en SuperBASIC).

Sintaxis: **haz** *var = exp*

LISTAR Lista en la impresora todos los procedimientos que se encuentran en la memoria.

Sintaxis: **listar**

CARGAR Carga el fichero de programas especificado, desde el cartucho del Microdrive a la memoria.

Sintaxis: **cargar** [**objeto**] *fnm*

Si Vd. incluye el parámetro opcional **objeto**, Archive asumirá que el fichero tiene el formato binario, en lugar de ASCII. (Vea **salvar**).

LOCAL En un procedimiento, este comando hace que la lista de variables a continuación sean variables locales. Estas variables sólo existen dentro del procedimiento en que se declaran, no definiéndose en ningún otro procedimiento. Sus valores se destruyen al salir del procedimiento.

Sintaxis: **local** *var* *[,*var*]*

SITUAR Localiza –en un fichero ordenado– el primer registro cuyo campo (o campos) coincide con la expresión (o expresiones).

Sintaxis: **situar** *exp* *[.*exp*]*

El registro se localiza mucho más rápidamente que con el comando **hallar**, pero **el fichero precisa estar ordenado**. Cada expresión debe referirse explícitamente al contenido de un campo de clasificación dado. En el caso de un campo de texto, a la hora de hallar el equivalente exacto se distingue entre mayúsculas y minúsculas.

Si Vd. ha clasificado (ordenado) el fichero en relación a más de un campo, podrá entonces especificar varias expresiones (una para cada campo de clasificación). Las expresiones deben ir separadas por comas y deben referirse a los campos empleados para ordenar el fichero. Deben seguir el mismo orden que en el comando **ordenar**. Por ejemplo:

```
ordenar animal$ ;a,peso;a
situar "Elefante",2000
```

hallará el primer registro en que el campo animal$ contenga el texto "Elefante" y un peso igual (o superior) a 2000.

Si no encuentra un registro que coincida exactamente, **situar** halla de todas formas un registro, que será el primero cuyo campo exceda de los valores especificados –en el sentido en que ha sido ordenado (es decir, la "d" viene después de la "e" si el fichero ha sido clasificado en orden descendente).

VER Abre el fichero nombrado solamente para leerlo. Si Vd. no especifica el nombre lógico, Archive provee el nombre "maestro".

Sintaxis: **ver** *fnm* [**lógico** *lfn*]

IMPRIMIR Envía a una impresora conectada a SER1 los valores en la lista que le sigue, de la misma forma que con el comando **listar**.

Sintaxis: **imprimir** [*exp*|*ptm* *[;*exp*|*ptm*] *] [;]

UNIR Añade los procedimientos del fichero de programas especificado, a los procedimientos que ya se encuentran en la memoria del computador. Si el fichero ya contiene un procedimiento con el mismo nombre que otro en la memoria, el nuevo procedimiento sustituye al procedimiento previo.

Sintaxis: **unir** [**objeto**] *fnm*

Si Vd. incluye el parámetro opcional **objeto**, Archive asumirá que setrata de un fichero en el formato binario, en lugar de ASCII. (Vea **salvar**).

MODO Cambia el formato de la pantalla.

Sintaxis: **modo** *var,var*

La primera variable puede tener un valor de 0 ó 1. El valor 0 une la zona de control, la zona de presentación y la zona de trabajo en una sola zona. El valor 1 vuelva a dividir la pantalla en tres zonas.

La segunda variable puede tener un valor de 4, 6 u 8, para seleccionar la pantalla de 40, 64 u 80 caracteres por línea.
Inicialmente, el formato de la pantalla es equivalente a:

```
modo 1,8
```

NUEVO

Borra todos los datos de la memoria del computador, dejándola lista para empezar de nuevo. Cierra todos los ficheros abiertos. (Este comando no borra los ficheros almacenados en un cartucho del Microdrive).

Sintaxis: **nuevo**

PROXIMO

Selecciona el próximo registro del fichero especificado, o del fichero actual si Vd. no da un nombre lógico para el fichero.

Sintaxis: **próximo** [*lfn*]

ABRIR

Abre el fichero especificado para poder leerlo y también modificarlo. Si Vd. no da un nombre lógico para el fichero, Archive le da el nombre lógico "maestro".

Sintaxis: **abrir** *fnm* [**lógico** *lfn*]

ORDENAR

Ordena los registros del fichero conforme al contenido de los campos especificados.

Sintaxis: **ordenar** *espec_ord* *[,*espec_ord*]*

donde: *espec_ord*:= *var*; **a** | **d**

El primer campo especificado en la lista es el campo de clasificación primario. Los registros cuyo campo de clasificación primario sea igual se clasifican entonces conforme al contenido del próximo campo de clasificación en la lista (si se especifica uno) y así sucesivamente. Por cada campo de clasificación especificado debe darse también la dirección en que va a efectuarse la clasificación. Esta debe ser **a** o **d** para indicar orden ascendente o descendente, respectivamente.

El comando **ordenar** solamente tiene en cuenta los 8 primeros caracteres de un campo de texto y Vd. no puede especificar más de cuatro campos para **ordenar** el fichero.

PAPEL

Establece el color de fondo para todo el texto que sigue, al color especificado por el valor de la expresión.

Sintaxis: **papel** *n.exp*

Los colores son: 0 y 1 Negro
2 y 3 Rojo
4 y 5 verde
6 y 7 blanco

Si el resultado de la expresión es mayor que 7, el valor tomado es el resto después de dividir por 8, es decir, **papel** 11 es equivalente a **papel** 3, seleccionando ambos el color rojo.

Si utiliza **papel** en un comando **escribir**, el color de fondo solamente cambia mientras dure ese comando.

POSICIONAR

Hace que el registro cuyo número viene dado por la expresión sea el registro actual.

Sintaxis: **posicionar** *n.exp*

ESCRIBIR

Muestra en la pantalla los valores de la lista de parámetros que le sigue –los cuales deben separarse con signos (;). Si la lista termina con un signo de punto y coma (;), el cursor no pasa a una nueva línea después de mostrar la lista. Vea asimismo **imprimir**.

Sintaxis: **escribir** [*exp*|*ptm*]_*[;*exp*|*ptm*]*][;]

ABANDONAR

Cierra todos los ficheros y regresa a SuperBASIC.

Sintaxis: **abandonar**

NOTA Al emplearse en un procedimiento, marca que el resto de la línea contiene un comentario. El texto que sigue en esa línea se ignora al ejecutar el procedimiento.

Sintaxis: **nota**

RESTAURAR Este comando restablece todos aquellos registros en el fichero actual que fueron removidos la última vez que utilizó **seleccionar**. Inutiliza el orden del fichero.

Sintaxis: **restaurar**

REGRESO Se emplea en un procedimiento para causar la terminación inmediata del mismo y regresar al procedimiento que lo invocó.

Sintaxis: **regreso**

EJECUTAR Carga en la memoria el fichero con el procedimiento especificado e inicia la ejecución del procedimiento **comienzo**.

Sintaxis **ejecutar** [**objeto**] *fnm*

Si Vd. incluye el parámetro opcional **objeto**, Archive asume que el fichero estará en el formato binario, en lugar de ASCII. (Vea **salvar**).

SALVAR Salva al cartucho en el Microdrive, en un solo fichero nombrado, todos los procedimientos que se encuentran en la memoria.

Sintaxis: **salvar** [**objeto**] *fnm*

Si Vd. incluye el parámetro opcional **objeto**, Archive salva el fichero en el formato binario, en lugar de ASCII. Esto significa que Archive no precisa convertir el programa a caracteres ASCII antes de salvarlo, resultando por tanto mucho más rápido. Usted puede **cargar**, **ejecutar** o **unir** este programa añadiendo el parámetro opcional **objeto** al comando correspondiente. Estas operaciones también son más rápidas, ya que no precisa hacerse ninguna conversión. Estos ficheros tienen la extensión _pro, en lugar de la extensión _prg normal.

Puede también salvar este programa **objeto** en una forma que le proteja contra los exámenes o modificaciones. Para esto, en lugar de **objeto** incluya el parámetro opcional **protec**. Un programa salvado de esta forma sólo puede cargarse, ejecutarse o unirse utilizando el parámetro opcional **objeto** con el correspondiente comando.

Un programa protegido no puede listarse, editarse o salvarse. Si Vd. une un programa protegido con otro programa, la combinación resultante estará igualmente protegida. La única forma de quitar la protección es con el comando **nuevo**.

La operación de salvar una versión protegida no afecta a la copia del programa en la memoria del computador. Usted podrá listar, editar o salvar dicho programa en la forma normal.

PANTALLA Presenta el formato de la pantalla previamente cargado con pcargar. No tiene ningún efecto si no está presente un formato de la pantalla. No presenta ninguna de las variables en la pantalla.

Sintaxis: **pantalla**

BUSCAR Busca el fichero actual desde el principio hasta hallar un registro en que se cumpla la expresión especificada. Este registro pasa a ser el actual.

Sintaxis: **buscar** *n.exp*

PEDITAR Invoca el editor de la pantalla para poder definir un nuevo formato de la pantalla. Vea el Capítulo 7.

ELEGIR Recorre todo el fichero y selecciona solamente aquellos registros en que se cumpla la expresión especificada. El fichero se comporta entonces como si sólo tuviera los registros seleccionados.

Sintaxis: **elegir** *n.exp*

Podrá restablecer todos los registros desechados con el comando **restaurar**.

PLEER

Aguarda a que sean leídas las variables en la lista que sigue a este comando, utilizando el orden especificado en dicha lista. Todas las variables en la lista deben hallarse en el formato activo de la pantalla.

Sintaxis: **pleer** *var* *[, *var*]*

PCARGAR

Carga un formato de la pantalla previamente definido y salvado. También presenta este formato de la pantalla y activa la indicación de las variables en la pantalla.

Sintaxis: **pcargar** *fnm*

Los valores presentados se actualizan entonces automáticamente siempre que el control regrese desde un procedimiento al intérprete del teclado.

NORMAL

Dirige a la impresora la salida como resultado de un comando **imprimir** y **listar**. Esto cancela el efecto del comando **vía**.

Sintaxis: **normal**

VIA

Dirige al fichero especificado –o a la pantalla– la salida como resultado de un comando **imprimir**, **listar** o **volcar**, en lugar de dirigirla a la impresora.

Sintaxis: **vía** *fnm* [**exportar** | *volcar*] o bien: **vía pantalla**

Si Vd. dirige la salida a un fichero, ésta va a través de la definición de la impresora actualmente instalada para que contenga todos los códigos especiales que precisa la impresora.

Si Vd. incluye el parámetro opcional **exportar**, Archive asegura que el fichero contenga solamente códigos ASCII imprimibles, caracteres de retorno y de avance de línea. El fichero resultante sirve para importarlo a Quill.

El parámetro opcional **volcar** permite transmitir el texto al fichero sin que sea procesado por la definición de impresora. En este caso, todos los códigos ASCII (incluidos los códigos de control) pasan directamente al fichero.

A no ser que Vd. especifique una extensión para el nombre del fichero, Archive asume la extensión **__lis** (o la extensión **__exp** o **__dmp** si Vd. incluye el parámetro opcional **exportar** o **volcar**).

La otra forma del comando –**vía pantalla**– dirige la salida a la pantalla en lugar de a la impresora.

PESCRIBIR

Se emplea dentro de un procedimiento para forzar la presentación de los campos del registro actual.

Sintaxis: **pescribir**

Debe existir un formato activo de la pantalla (el formato de la pantalla puede activarse previamente con el comando **pantalla**, **pcargar** o **indicar**). Si no hay un formato activo de la pantalla, el comando no tiene ningún efecto.

PSALVAR

Salva la zona de presentación actual a un fichero nombrado en un cartucho del Microdrive y constituye entonces un formato definido de la pantalla.

Sintaxis: **psalvar** *fnm*

Este comando salva el texto de la pantalla y una lista de variables en la pantalla, junto con sus posiciones.

STOP

Termina la ejecución de todos los procedimientos y devuelve el control al teclado.

Sintaxis: **stop**

TRAZAR

Activa y desactiva el modo de trazar.

Sintaxis: **trazar**

Escriba:

```
trazar
```

para activar la opción de trazar. En el modo de trazar, cada línea del programa se presenta en la zona de trabajo de la pantalla, al ejecutarse. Mantenga pulsado el espaciador para obtener una pausa. Continuará la operación de trazar cuando Vd. suelte el espaciador. Para volver a desactivar la opción de trazar, escriba:

```
trazar
```

ACTUALIZAR

Sustituye el registro actual en el fichero especificado (o en el fichero actual si Vd. no da un nombre lógico para el fichero), por un registro que contiene los valores actuales de las variables de los campos.

Sintaxis: **actualizar**

USAR

Hace que el fichero especificado sea el fichero actual.

Sintaxis: **usar** *lfn*

MIENTRAS

Ejecuta repetidamente las sentencias entre **mientras** y **finmientras** siempre que no sea cero (que sea verdadero) el valor de la expresión.

Sintaxis: **mientras** *n.exp* : ... : **finmientras**

FUNCIONES

Considere una función como una receta que convierte uno o más valores iniciales, denominados los *argumentos* de la función, en un valor distinto, el cual se dice que es el valor *devuelto* por la función.

Las funciones provistas en Archive pueden tener tres, dos, uno o ningún argumento. Los argumentos de una función se escriben entre paréntesis a continuación del nombre de la función. No debe dejar un espacio entre el nombre y el comienzo del paréntesis, pero podrá dejar espacios entre los parámetros que están dentro del paréntesis. Si una función tiene más de un argumento, los argumentos se separan con comas. Todas las funciones deben ir seguidas del paréntesis, aun cuando no tengan ningún argumento. El paréntesis le recordará que se trata de una función, permitiéndole distinguir entre una variable y una función, aun cuando tengan el mismo nombre.

Archive le provee las siguientes funciones:

ABS(*n.exp***)** Devuelve el valor absoluto del argumento, es decir, ignora el signo menos.

ARCTG(*n.exp***)** Devuelve el ángulo, en radianes. cuya tangente es n.exp.

CAR(*n.exp***)** Esta función devuelve el carácter ASCII cuyo código es n.exp. Un carácter con un código ASCII menor que 32 sólo se envía a la impresora si va precedido del código ASCII cero. Por ejemplo:

```
imprimir car(0)+car(13)
```

pasa a la impresora el carácter ASCII para el retorno del carro. Esto es útil si su impresora precisa secuencias de códigos de control para producir efectos especiales -consulte el manual de la impresora para los códigos especiales que requiere.

Usted puede, por ejemplo, enviar una "A" a la pantalla con:

```
escribir car(65).
```

CODIGO(*s.exp***)** Devuelve el valor ASCII del primer carácter hallado en el texto especificado.

COS(*n.exp***)** Devuelve el coseno del ángulo dado (en radianes).

CUENTA([*lfn* **])** Devuelve la cuenta del número de registros en el fichero actual.

FECHA(*n.exp***)** Devuelve la fecha de hoy en una serie de texto de tres formas:

n.exp	serie de texto
0	"AAAA/MM/DD"
1	"DD/MM/AAAA"
2	"MM/DD/AAAA"

Usted debe primeramente poner en hora el reloj del sistema, como se indica en la *Guía de Palabras Clave de SuperBASIC*.

DIAS(*s.exp*) Devuelve el número de días, desde el uno de enero de 1583 hasta la fecha dada en la expresión de texto, en la forma "AAAA/MM/DD". En la conversión se asume que se utiliza el calendario Gregoriano (moderno). Por tanto, la fórmula sólo es válida para fechas posteriores a 1582.

DEC(*valor,ld,ancho*)
valor:=(n.exp)
ld:= (n.exp)
ancho:= (n.exp)

Convierte el *valor* numérico dado en una serie de texto equivalente, en formato decimal, con los lugares decimales indicados en *ld*. El texto está justificado por la derecha en un campo con el número de carácteres indicado en *ancho*. Por ejemplo:

dec(1.23e1,3,10) devuelve el texto " 12.300" (con 4 espacios a la izquierda).

GRAD(*n.exp*) Toma un ángulo, medido en radianes, y lo convierte al mismo ángulo en grados.

FINF([*lfn*]) Devuelve un valor que indica si Vd. ha tratado de leer más allá del final del fichero actual, o del fichero especificado si incluye un identificador de fichero. Devuelve el valor 1 si Vd. trató de leer más allá del final del fichero. En caso contrario es cero.

NUMERR() Devuelve el número del último error que se produjo (el número de error cero indica que no hubo errores). El número de error es el que aparece junto con el mensaje de error cuando Archive reporta un error detectado.

EXP(*n.exp*) Devuelve el valor de e (aprox. 2.718) elevado a la potencia dada en (*n.exp*). El valor devuelto será erróneo si *n.exp* es mayor que +88, ya que el resultado excederá entonces de la escala numérica de Archive.

NOMCAM(n.exp|, *lfn*|)
Devuelve el nombre del campo especificado en el registro actual del fichero especificado (o del fichero actual si Vd. no dio un nombre lógico de fichero). Observe que **nomcam(0)** devuelve el nombre del primer campo.

TIPOCAM(*n.exp* [, *lfn*])
Devuelve el tipo del campo especificado en el registro actual del fichero especificado (o del fichero actual si Vd. no dio un nombre lógico de fichero). Observe que **tipocam(0)** devuelve el tipo del primer campo.

Devuelve el valor 0 si el campo es numérico; de lo contrario devuelve 1.

VALCAM(*n.exp*[, *lfn*])
Devuelve el valor del campo especificado en el registro actual del fichero especificado (o del fichero actual si Vd. no dio un nombre lógico de fichero). Observe que **valcam(0)** devuelve el valor del primer campo.

HALLADO() Devuelve el valor 1 si se halla un registro como resultado del comando **buscar** o **hallar**; de lo contrario devuelve 0.

GEN(*valor,ancho*)
valor:=n.exp
ancho:=n.exp

Convierte el *valor* numérico dado a la serie de texto equivalente, en el formato general. El texto está justificado por la derecha en un campo con el número de caracteres indicado en *ancho*. Por ejemplo:

gen(1.23e1,10)

devuelve el texto " 12.3" (con 6 espacios a la izquierda).

TECLA() Aguarda a que se pulse una tecla y devuelve un carácter de texto que corresponde a la tecla pulsada.

PULSADA() Devuelve un carácter de texto que corresponde a la tecla que fue pulsada en el momento de invocar la función. No aguarda a que se pulse una tecla, pero devuelve una serie vacía ("") si no se pulsa una tecla.

ENSERIE(*larga,corta*)
larga:= s.exp
corta:= s.exp

Halla el primer caso en que aparece *corta* dentro de *larga* y devuelve la posición del primer carácter de *corta* en *larga*. Devuelve el valor cero si no la localiza. Esta operación diferencia entre mayúsculas y minúsculas.

```
enserie("Febrero","Feb")     devuelve 1
enserie("Febrero","eb")      devuelve 2
enserie("Febrero","EB")      devuelve 0
```

ENT(*n.exp*) Devuelve el valor entero del número, truncándolo en el punto decimal. Este truncado siempre actúa hacia cero. Así pues:

```
ent(3.7)      {devuelve 3}
ent(-4.8)     {devuelve -4}
```

LONG(*s.exp*) Devuelve el número de caracteres en el texto especificado.

LN(*n.exp*) Devuelve el logaritmo natural (base e) de n.exp. Se produce un error si n.exp es negativo o cero, ya que no se definen los logaritmos en esta escala.

MINUS(*s.exp*) Convierte el texto especificado a minúsculas.

MEMORIA( ) Devuelve el número de bytes libres que restan en la memoria.

MES(*n.exp*) Devuelve el nombre del mes, en texto.

Por ejemplo, **mes(3)** devuelve el texto "Marzo".

Si se emplea un argumento mayor que 12, se sustituye por el resto después de dividir por 12, con lo cual, por ejemplo, **mes(13)** y **mes(1)** dan ambos el resultado "Enero".

NUM(*valor, ancho*)
valor:= n.exp
ancho:= n.exp

Convierte el *valor* numérico dado a la serie de texto equivalente, en el formato de enteros. El texto está justificado por la derecha en un campo con el número de caracteres indicado en *ancho*. Por ejemplo:

num(1.23e1,10) devuelve el texto " 12" (con 8 espacios a la izquierda).

NUMCAM([*lfn*])
Devuelve el número de campos en los registros del fichero especificado (o del fichero actual si Vd. no dio un nombre lógico de fichero).

PI() Devuelve el valor de la constante matemática π.

RAD(*n.exp*) Toma un ángulo, medido en grados, y lo convierte al mismo ángulo en radianes.

NUMREG([*lfn*])
Devuelve el número (contando el primer registro como cero) del registro actual del fichero especificado (o del fichero actual si Vd. no dio un nombre lógico de fichero).

REPT(*s.exp,n.exp*)
Esta función devuelve una serie que consiste en un número de copias del primer carácter del texto dado. El texto resultante puede ser de hasta 255 caracteres. Por ejemplo:

```
escribir rept("*",5)      {escribe cinco asteriscos}
escribir rept("abc",3)      {escribe "aaa"}
```

SIGNO(*n.exp*) Devuelve +1, −1 ó 0, dependiendo de si el argumento es positivo, negativo o cero.

SENO(*n.exp*) Devuelve el valor del seno del ángulo especificado (en radianes).

RAIZ(*n.exp*) Devuelve la raíz cuadrada del argumento, que no debe ser negativo.

SERIE(*n,tipo,ld*)

n:=*n.exp*
tipo:=*n.exp*
ld:=*n.exp*

Convierte un número, n, en la serie de texto equivalente.

El segundo parámetro, tipo, indica la forma de la serie convertida, como sigue:

0 decimal (punto flotante)
1 notación exponencial, o científica
2 enteros
3 formato general

El tercer parámetro, ld, indica el número de dígitos a continuación del punto decimal en la serie convertida. Debe siempre especificarse este parámetro, si bien se ignora su valor para el formato general y de enteros.

Por ejemplo:

```
haz a$=serie(12.3456,0,2) {da el valor "12.35" para a$}
haz a$ serie(12.3456,1,4) {da el valor "1.2346e1" para a$}
```

TG(*n.exp*) Devuelve la tangente del ángulo especificado (en radianes).

HORA( ) Devuelve, en la forma de texto, la hora del día en el formato "HH:MM:SS". Usted debe primeramente poner en hora el reloj del sistema, como se indica en la Guía de Palabras Clave de SuperBASIC.

MAYUS(*s.exp*) Convierte el texto especificado a mayúsculas.

VAL(*s.exp*) Convierte el texto a su valor numérico equivalente. Sólo convierte el texto formado por caracteres numéricos válidos y se detiene la conversión al hallar el primer carácter que no pueda ser interpretado como un dígito. Por ejemplo, **val("1.1ABC")** devuelve el valor numérico 1.1, mientras que **val("ABC")** devuelve 0.0.

VALVAR(*s.exp*) Devuelve el valor de la variable cuyo nombre viene dado por *s.exp* – por ejemplo:

```
haz a$="long"
haz longitud=15
escribir valvar(a$+"itud")
```

escribirá el valor 15.

Observe que **valvar(nomcam(y))** equivale exactamente a **valcam(y)**.

ERRORES

Cuando Archive detecta un error en un comando escrito desde el teclado, o en un procedimiento, presenta en la pantalla un número de error y un mensaje breve. Como ejemplos de errores que se detectan cabe mencionar:

cuando se trata de dividir por cero
si sin el correspondiente **finsi**
cuando se suministra un procedimiento con un número erróneo de parámetros.

Si el error viene del teclado, el texto en la sentencia permanece visible en la zona de trabajo. Puede pulsar **F5** para invocar el texto y emplear el editor de línea para corregir el error. Podrá entonces pulsar [↵] para ejecutar la sentencia corregida.

Si el error viene de una sentencia de un programa, Archive muestra el nombre del procedimiento y la línea en que se produjo el error. Podrá entonces utilizar el editor de programas para corregir el error.

Cuando Vd. utiliza el comando **error** en sus programas, Archive reportará los errores que detecte en un procedimiento que Vd. ha marcado con **error**. Podrá entonces solucionar el error en la forma que le parezca (incluso ignorarlo). Puede hallar qué error se ha producido examinando el valor devuelto por la función **númerr()**. Este número es el mismo que da Archive cuando presenta un mensaje de error.

En la lista que sigue se enumeran los números de error de Archive, junto con los mensajes correspondientes. En lo posible, se incluye en la lista un breve ejemplo de una sentencia que causaría un error. Los mensajes de error no están diseñados para localizar exactamente el error, sino para dar una idea del tipo de error que debe buscarse.

Los mensajes de error para los cuales no se incluye un breve ejemplo se marcan con un asterisco. Se trata de ellos en las notas que siguen a la lista.

Núm.	Mensaje	Ejemplo
0	(ningún error)	
1	comando no reconocido	altera (en lugar de alterar)
2	se esperaba fin de sentencia	`haz x=3 haz y=4`
3	se esperaba nombre de variable	`haz 31=x`
4	parámetro de escribir no reconocido	escribir crear
5	tipo incorrecto de datos	* (1)
6	se esperaba expresión numérica	`haz x="paco"`
7	se esperaba expresión alfanumérica	`haz x$=4`
8	variable no hallada	`haz x=qq` (qq no definido)
9	variable no definida	escribir qq
10	falta separador	escribir en 5
11	nombre demasiado largo	`haz estenombremuylargo=4`
12	nombre duplicado	`crear:n$:n$:fincrear`
13	se esperaba literal alfanuméroco	* (2)
14	falta finproc	* (3)
15	sentencia proc incorrecta	* (3)
16	fin prematuro de sentencia	`crear"prueba":fincrear`
17	mala estructura del programa	* (4)
18	demasiados números	* (5)
50	falta signo de comillas final	`haz x$="paco`
51	falta exponente después de 'E'	`haz x=1.2E`
52	número fuera de límites	`haz x=1.2E100`
53	símbolo desconocido	`haz x=%`
70	error en sintaxis de evaluador	`haz x=3+`
71	falta parte de un paréntesis	`haz x=(3+5)/7)`
73	tipos distintos	`haz x$="paco"+3`
74	número incorrecto de argumentos	`haz x$=serie(1,2)`
75	serie demasiado larga	`haz x$=rept("*",256)`
76	división por cero	`haz a=0: haz x=5/a`
77	argumentos de función incorrectos	`haz x$=raiz(-4)`
78	error en subíndice de serie	`haz x$="paco"` (hasta 97)
80	memoria agotada	* (6)
90	no hay espacio para abrir un fichero	* (7)
91	transferencia incompleta de fichero	* (8)
93	fuera de límites	escribir en 100,100;37
94	fichero no abierto	añadir (sin antes abrir un fichero)
100	no se puede abrir fichero	`ver"xxx"` (inexistente)
101	intento de escribir en un fichero de lectura solamente	`ver"nombres":insertar`
103	tipo incorrecto de fichero	`pcargar"nombres"` (fichero de datos)
104	nombre incorrecto de fichero	`salvar"3prueba"`
105	error al leer el fichero	* (9)

Notas

(1) La causa más probable del error 5 –'tipo incorrecto de datos'– es si Vd. introduce un texto literal cuando se esperaba un número, tal como en respuesta a una sentencia de **leer**. Por ejemplo:

```
leer x
```

(2) El error 13 –'se esperaba literal alfanumérico'– puede ocurrir, por ejemplo, al importar un fichero creado por Vd. (sin utilizar uno de los comandos **exportar** en los programas QL). Significa que Archive ha hallado un número, o una expresión numérica o de texto, donde esperaba hallar un valor de texto literal. En la mayoría de los casos en que Archive halla datos numéricos cuando esperaba texto, o viceversa, se producirá el error 7 o el error 8.

(3) Los errores 14 –'falta finproc'– y 15 –'sentencia proc incorrecta'– no se producirán normalmente. Indican que Archive ha detectado que falta un **finproc** o un error en la estructura de una sentencia **proc** en un procedimiento. Sólo ocurrirán si Vd. crea un fichero de programas con un editor distinto al suministrado con Archive.

(4) El error 17 –'mala estructura del programa'– suele indicar que un comando **todos**, **si** o **mientras** no va acompañado del correspondiente **fintodos**, **finsi** o **finmientras** en un procedimiento. Podrá también ocurrir este error si Vd. incluye un **finproc** dentro de otra estructura del programa, o si utiliza **regreso** directamente desde el teclado.

(5) El error 18 –'demasiados números'– indica que Vd. está tratando de introducir más números de los que caben en la memoria reservada para ello. Este error puede ocurrir en una línea de entrada desde el teclado, o al cargar un programa que incluye un procedimiento con demasiados números en una de sus líneas. El límite exacto depende de las circunstancias –un límite característico sería de 15 a 20 números, así que será muy poco probable que Vd. se encuentre con este error.

(6) El error 80 –'memoria agotada'– es muy improbable que ocurra. Podrá solamente darse si Vd. utiliza un programa muy grande. El tamaño de un fichero normal de datos no está limitado por la cantidad de memoria en el computador, ya que solamente se almacena en la memoria parte de un fichero grande en un momento dado. Si Archive presentara alguna vez este error, Vd. precisará reducir el tamaño del programa que utiliza antes de continuar. En caso necesario, podrá dividir el programa en varias secciones, en ficheros distintos, y emplear entonces **unir** para cargar cada sección al precisarla. Para efectuar este proceso precisará una buena pericia de programación.

(7) El error 90 –'no hay espacio para abrir un fichero'– ocurre cuando se llena el área de la memoria que Archive reserva para almacenar la información interna acerca de los ficheros que se encuentran en la memoria. Podrá ocurrir esto aun cuando continúe habiendo espacio en la memoria, es decir, cuando no esté próximo a cero el valor devuelto por la función memoria().

(8) El error 91 –'transferencia incompleta de fichero'– significa que ha fallado, por alguna razón, la operación de cargar o salvar un fichero. Esto podrá suponer que se han corrompido los datos, o que se ha dañado el cartucho o el Microdrive.

(9) El error 105 –'error al leer el fichero'– significa que parte de los datos en el fichero están en el formato incorrecto, en el orden incorrecto o que se han corrompido. Esto sólo es probable que ocurra si Vd. crea su propio fichero de importar –o su propio fichero de programas– sin utilizar el editor de programas de Archive (usos avanzados).