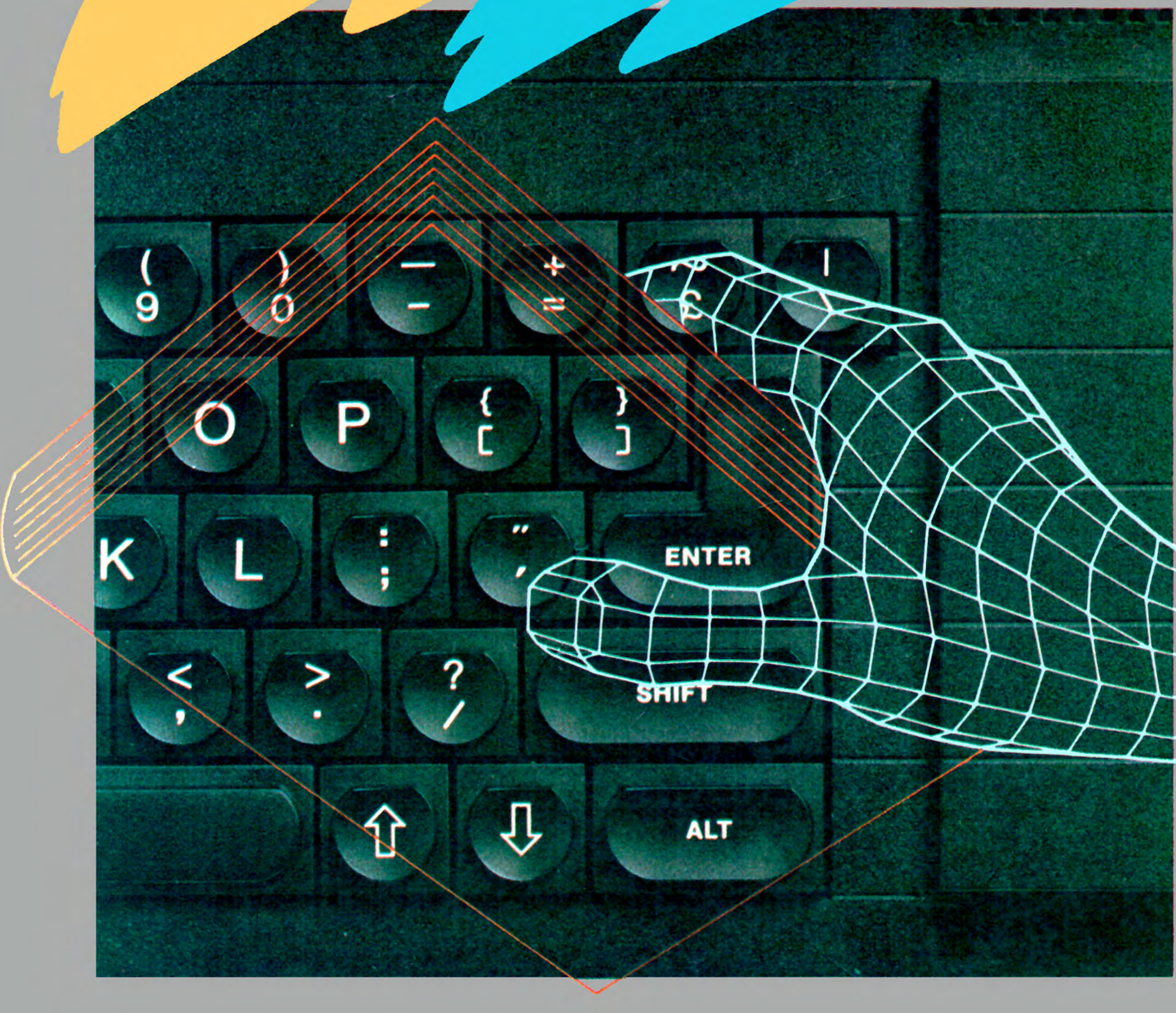
ARCHIVE

Manejo de la Base de Datos en el QL

Ian Murray/Blueprint



ANAYA

ARCHIVE

Manejo de la base de datos en el QL

Ian Murray
Blueprint

ANAYA MULTIMEDIA

INFORMATICA PERSONAL-PROFESIONAL

Título de la obra original:
QL ARCHIVE

Traducción: Santiago Gala
Diseño de cubierta: Narcís Fernández

Villafranca, 22. 28028 Madrid
Depósito legal: M. 10.332-1986
ISBN: 84-7614-077-0
Printed in Spain
Imprime: Anzos, S. A. Fuenlabrada (Madrid)

Indice

Les presentamos a Maya; la señora de Alcoy. Es la esposa de Juan, y está intentando establecer su propio negocio de joyería (jades, etc.). Maya está ilusionada con el proyecto. Su marido no está seguro de que pueda salir adelante.

Este señor es Juan Alcoy. Es director de producción de una compañía dedicada a juguetes: Juguetes Mas. Tiene exceso de peso (le sobran sus dos tónicas con ginebra diarias).

Y Eva Alcoy. Es hija de Maya y Juan. Es opositora y también prometida de Salva Calderón (por alguna razón que nadie acierta a comprender).

También les presentamos a Salva Calderón. Es el prometido de Eva. Está convencido de ser el nuevo Fernando Fernán-Gómez (aunque sobre ese punto están permitidas todas las opiniones).

Y lo mejor de la familia. Tere y Chona Alcoy. Son las mellizas de la familia Alcoy. A Tere le gusta Toyah, y Chona prefiere a Boy George. Aparte de esto, no encontramos muchas más diferencias.

Dibujos de Derek Alder, del *Sunday Times*.

ENTER

SHIFT

ALT

1

Primeros pasos

La mayor parte del contenido de este capítulo le resultará familiar si ha leído ya los *Primeros pasos* de alguno de los libros de esta misma serie sobre los programas del QL. Sin embargo, parte de la información es nueva y se relaciona específicamente con ARCHIVE: consulte las secciones llamadas *La pantalla, Comandos* y *Edición de texto*.

Si tiene problemas

La manera de abandonar ARCHIVE se explica al final de este capítulo, y el uso del botón de reset en la página siguiente. Si el ordenador deja de responderle, o bien si le envía mensajes extraños, consulte el apéndice *Si tiene problemas*.

EL QL

Bajo su estilizado teclado yace un ordenador muy complicado. Es una máquina potente, atenta a trabajar y obedecer sus órdenes. Se llama *software* a aquellos programas que, mediante su

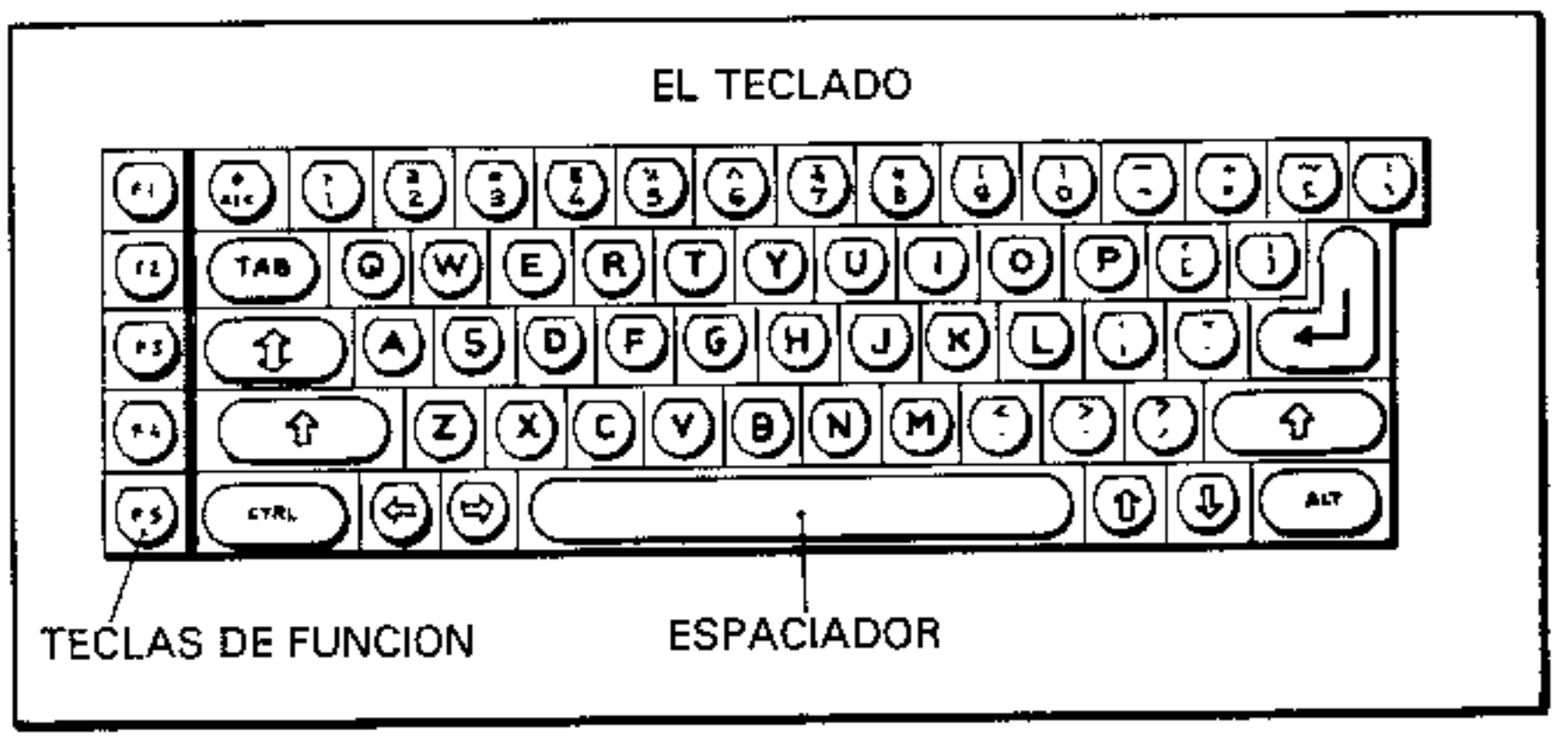
ayuda, le indican al ordenador qué hacer. Ejemplos de *software* son ARCHIVE, QUILL, ABACUS y EASEL, los cuatro programas incluidos con el QL. Para almacenar el *software* usamos los cartuchos de Sinclair, de una manera parecida al almacenamiento de música en un cassette.

La principal diferencia es que se puede apagar un magnetófono en cualquier momento sin estropear la cinta. Si se saca el enchufe del ordenador sin grabar el trabajo, se perderá lo hecho. Asegúrese de salvar su trabajo en cinta con regularidad, y grabe las copias en más de una cinta.

Instale el ordenador y conecte la pantalla, teclado y fuente de alimentación (véanse los manuales apropiados). Mire ahora al teclado: se parece a una máquina de escribir convencional; la principal diferencia es que tiene unas cuantas teclas adicionales.

EL TECLADO

A continuación mostramos el teclado, con una explicación de la función de cada tecla. A lo largo del libro irán siendo explicadas en detalle.



↵ Le indica a ARCHIVE que ha terminado de introducir una línea.

ESC Tecla de **ESCAPE.** Sirve para volver a la pantalla principal, o bien para abandonar una línea.

CTRL Tecla de **CONTROL.** Se usa en conjunción con otras teclas para borrar líneas de texto.

ALT Ignórela. No tiene ninguna función en ARCHIVE.

⇧ Si la pulsa al tiempo que alguna otra tecla obtendrá la letra mayúscula o el símbolo de la parte superior de la tecla.

⇧ Cuando se pulsa, bloquea el teclado en mayúsculas. Usela de nuevo para volver a minúsculas. Sólo tiene efecto sobre las letras.

TAB **TABULADOR.** Sirve para moverse entre los campos de un registro o los procedimientos de un programa.

RESET No es una tecla normal, sino un botón bajo el extremo derecho del QL. Tiene casi el mismo efecto que apagar y encender de nuevo el ordenador. Usela sólo si fallan todas las alternativas para escapar de una situación en la que el programa se ha "bloqueado" o "colgado".

Teclas del **cursor**

⇧ ⇩ Mueven el cursor al extremo izquierdo o derecho de una línea.

⇦ ⇨ Mueven el cursor una posición a la izquierda o la derecha.

Teclas de **función**

F1 Tecla de AYUDA.

F2 Presenta o elimina el AREA DE CONTROL.

F3 Avanza una página en la lista de comandos en el AREA DE CONTROL.

F4 Abandona los comandos alterar/insertar, y el modo inserción en el editor.

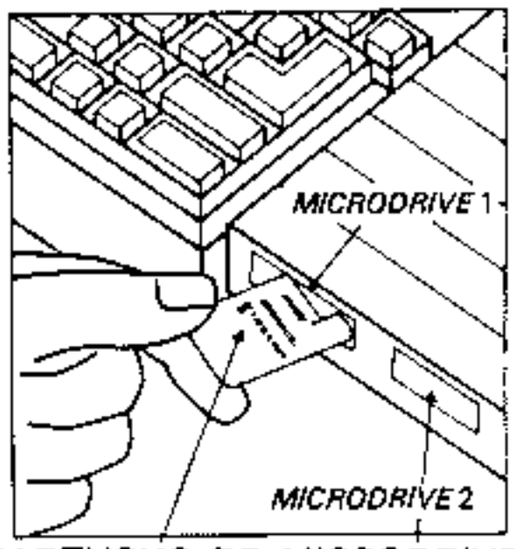
F5 La tecla de edición de línea.

CARTUCHOS DE *MICRODRIVE*

Los cartuchos son los microcassettes negros que se introducen en la unidad de *microdrive*.

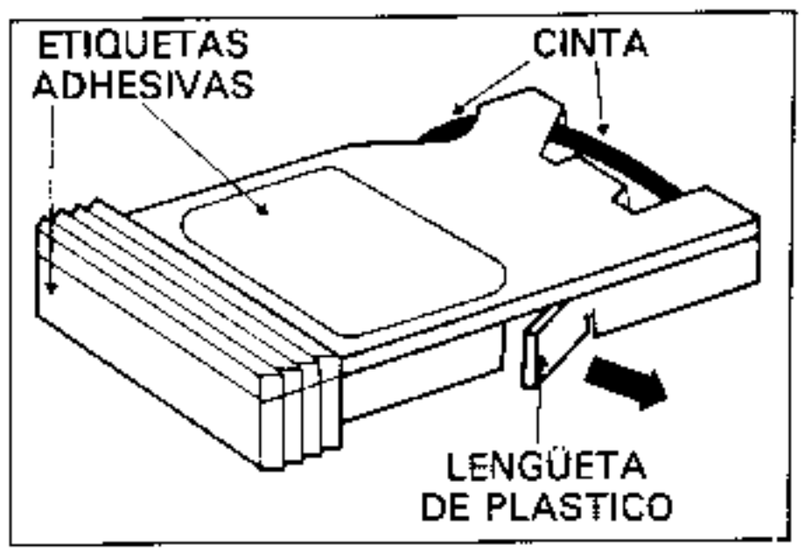
Son parecidos a una cinta de cassette, pero han sido diseñados especialmente para el QL. Uno de sus cartuchos contendrá el programa ARCHIVE (está en una caja junto con EASEL, ABACUS y QUILL). Va a necesitar varios cartuchos vacíos para grabar información y programas: como los cassettes, se pueden borrar y volver a grabar.

MICRODRIVES



CARTUCHO DE *MICRODRIVE*
(con su etiqueta en
la parte superior)

CARTUCHO DE *MICRODRIVE*



Las etiquetas de los cartuchos

Cada cartucho tiene un pequeño rectángulo en uno de los lados para la etiqueta *(véase diagrama)*. Es importante mantenerla actualizada para saber qué contiene el cartucho de un vistazo. Esta etiqueta debe mirar hacia arriba cuando inserte el cartucho en la unidad.

Como las etiquetas de los cartuchos son pequeñas, es útil etiquetar cada cartucho con un número, y distinguir entre cartuchos *de programa* y *de trabajo* (de datos). Haga una anotación de cada cartucho en un cuaderno. Use una página para cada cartucho, y anote los cambios cada vez que añada o borre algún fichero. Si no lo hace, se sorprenderá de lo fácil que resulta perderle la pista a su trabajo.

Protección de cartuchos

Observe la diferencia entre el cartucho de ARCHIVE y uno nuevo. La pequeña lengüeta del lateral derecho falta en el cartucho de ARCHIVE. Esto es un dispositivo de seguridad muy útil. Si se elimina la lengüeta, el ordenador no escribe ni borra nada de ese cartucho; sólo lee la información previa.

De esta manera ni usted ni nadie podrá borrar accidentalmente su copia de ARCHIVE, ni hacer ningún daño si pulsa las teclas equivocadas.

Si tiene necesidad de volver a escribir un cartucho protegido, basta con reemplazar la lengüeta con un trozo de cinta adhesiva. Asegure la fijación, para que no se despegue dentro del ordenador.

No saque el cartucho mientras esté encendida la luz roja que hay junto a la unidad. El cartucho de programa se puede sacar en cuanto acabe la carga de ARCHIVE, siempre que no necesite recurrir a la ayuda de ARCHIVE. Los cartuchos de datos se pueden intercambiar sin problemas, siempre que ningún programa esté haciendo uso de ellos, y que cualquier archivo de datos haya sido cerrado. Si extrae el cartucho en el momento incorrecto, se puede destruir la información previamente almacenada. Para una discusión más detallada, véase el capítulo 3.

Precauciones de carácter general

- No toque la cinta con los dedos, ni introduzca nada en el cartucho.
- Devuelva siempre el cartucho a su funda cuando no esté usándolo.
- No mueva el QL con un cartucho én su interior, ni siquiera si está apagado.
- Nunca toque un cartucho si la unidad está en funcionamiento, es decir, si la luz roja está encendida.
- Si queda expuesto un bucle de cinta en alguna de las dos partes que se indican en el diagrama, devuelva cuidadosamente la cinta al interior del cartucho. Para ello se puede usar una varilla limpia y suave, como la de un lápiz o bolígrafo. Nunca toque la cinta con los dedos.
- Los cartuchos de *microdrive* no durarán toda la vida, y necesitan ser reemplazados periódicamente. Es muy importante conservar copias de seguridad de los programas y archivos útiles en otro cartucho.
- Proteja los programas y datos que no quiere sobregrabar, eliminando la lengüeta de plástico del cartucho como se muestra en el diagrama.
- Si quiere volver a escribir en un cartucho protegido, pegue cinta adhesiva en lugar de la lengüeta.
- No pulse el botón de inicialización o **RESET** en el lateral del ordenador si hay cartuchos en las unidades. Sáquelos antes de inicializar. Si el ordenador está "colgado" y está encendida la luz de un *microdrive*, deje el cartucho y pulse el **RESET**.

COPIADO DEL CARTUCHO DE PROGRAMA

Antes de usar ARCHIVE por primera vez es vital que haga una copia de seguridad del programa, ya que los cartuchos no tienen una duración ilimitada, y pueden estropearse. Aparte de los peligros normales del uso y la fatiga, al perro le pueden resultar un bocado agradable o un niño puede sacar toda la cinta. Un paso en falso sobre el cable de alimentación también puede dañar el cartucho del programa. En cualquiera de estos casos, los daños pueden hacerle perder el programa almacenado en el cartucho, en este caso ARCHIVE.

Para hacer una copia de ARCHIVE necesita seguir unas instrucciones concretas. Sígalas exactamente: no importa que no las entienda. El proceso usa el idioma incorporado en el QL, que se conoce como SUPERBASIC.

Puede olvidar lo que ha hecho en cuanto acabe: no tendrá necesidad de volver a usar el SUPERBASIC en ARCHIVE.

1. Conecte el ordenador.
   Asegúrese de que el cable de alimentación está conectado y enchufado a la red (la luz anaranjada a la izquierda del QL se lo confirma). No inserte todavía ningún cartucho en la máquina. Tras un momento, la pantalla mostrará el mensaje siguiente:

```
F1 ..... Monitor
F2 ..... TV

© 1983 Sinclair Research Ltd.
```

Nota: Si utiliza un televisor, es necesario sintonizarlo al canal adecuado para el QL. Si tiene un número suficiente de presintonías, sintonice alguno de los botones libres para el QL.

F1 o F2

2. Pulse **F1** si tiene un monitor especial para ordenadores.
   F2 si utiliza una televisión doméstica.

El ordenador busca primero un cartucho de programa en la unidad de la izquierda (en adelante unidad 1), y, como no lo encuentra, le presenta la pantalla de BASIC del QL (una pantalla vacía en una TV). El cursor (un pequeño rectángulo de color claro) parpadea en la parte inferior de la pantalla esperando su próxima orden.

3. Ponga el cartucho original de ARCHIVE en la unidad de la derecha (unidad 2).

4. Introduzca un cartucho virgen en la unidad izquierda.

lrun mdv2_clone

5. Teclee lrun mdv2_clone.
   Tenga en cuenta que se trata de la letra "l", no del *número* "1". Observe también el espacio en blanco entre lrun y mdv2, así como el subrayado " _" (⇑ + guión) antes de clone. En otras palabras, le pide al ordenador que lea y ejecute un programa en BASIC llamado clone desde la segunda unidad (la derecha), que contiene el original de ARCHIVE.
   Si comete algún error al escribir, pulse la tecla **CTRL** con la flecha izquierda para borrar un carácter a la izquierda del cursor.

↵

6. Pulse ↵.
   El *microdrive* comienza a girar. El QL responde "Formatear mdv1---Pulse la espaciadora para continuar".

> *Nota:* En la jerga informática, "formatear" significa marcar la cinta del *microdrive* de tal manera que el QL pueda saber su posición rápidamente. Tenga cuidado: este proceso borra cualquier información previa de la cinta. Por el momento supondremos que usa un cartucho virgen.

ESP

7. Pulse la BARRA ESPACIADORA.
   Los *microdrives* suenan de nuevo, "formateando" el cartucho de la izquierda. A continuación, se copia el programa ARCHIVE, fichero por fichero, en el nuevo cartucho. El proceso tarda algunos minutos, y, una vez finalizado, el cursor (que marca el lugar donde se introducen los caracteres) volverá a aparecer en la parte inferior de la pantalla.

> *Nota:* De ahora en adelante use siempre la copia de ARCHIVE, y no el original. Así se asegura de que si algo sale mal todavía tendrá un cartucho original en perfecto estado.

Más adelante, cuando se haya asegurado de que la nueva copia de ARCHIVE funciona y de que su impresora está adaptada correctamente, elimine la lengüeta a la derecha del cartucho. Así protegerá el cartucho y evitará un borrado accidental de su copia de ARCHIVE.

FORMATEO DE UN CARTUCHO DE *MICRODRIVE*

Para utilizar ARCHIVE en los próximos capítulos, necesitará una copia de ARCHIVE y un cartucho vacío y formateado, que servirá para almacenar los archivos que cree.

Si no dispone de un cartucho formateado, asegúrese de que el ordenador le presenta la pantalla del BASIC, y siga estas instrucciones:

1. Introduzca un cartucho de *microdrive* nuevo en la unidad de la derecha.

format mdv2_arcdata

2. Escriba format mdv2_arcdata.
   Así se utiliza el comando format del QL; le pide al ordenador que formatee un cartucho en la unidad 2, la de la derecha, y que le ponga al cartucho el nombre arcdata. Tenga cuidado con el "_" y el ESPACIO. Cada vez que formatee un cartucho se le debe dar un nombre, que puede tener hasta diez caracteres, sin incluir espacios.
   Es útil que el nombre de cada cartucho (también llamado "nombre de volumen", como si fueran libros) describa su contenido. Una copia de seguridad de la cinta de datos podría llamarse, por ejemplo, "arcdata2".

3. Pulse ↵ para ejecutar la orden. La unidad 2 girará unos segundos, y una vez haya acabado volverá a la pantalla del BASIC.
   El número de sectores libres (en otras palabras, la cantidad de espacio libre en la cinta) aparece en la pantalla, por ejemplo, 218/219 sectores. Quiere decir que la cinta tiene 219 sectores, de los cuales, 218 quedan libres para el usuario.

Se puede utilizar también el comando formatear desde dentro de ARCHIVE. Sea muy cuidadoso al formatear las cintas, ya que el proceso elimina cualquier información previa. Una buena idea es dejar sólo la cinta a formatear en la unidad, para asegurarse de no borrar por accidente un cartucho equivocado.

ARRANCANDO ARCHIVE

Ya está preparado para usar ARCHIVE. Lo primero es que el ordenador "lea" el programa. Para ello le hará falta la copia de trabajo de ARCHIVE y un cartucho formateado. Si el QL

está conectado y se encuentra en la pantalla del BASIC, pase a la sección siguiente. En caso contrario:

1. Conecte el ordenador.
Después de un momento aparecerá en la pantalla el mensaje:

F1 ..... Monitor
F2 ..... TV

2. Inserte una copia de ARCHIVE en el *microdrive* izquierdo (unidad 1).

F1 o F2

3. Pulse **F1** si tiene un monitor.
F2 si usa una pantalla de TV.

Mire en el diagrama del teclado si no está seguro de la tecla que debe pulsar. Si pulsa **F1** le indica al QL que utilice 80 columnas para presentar la información, mientras que **F2** le hará usar una presentación de 64 columnas.

Tras unos segundos se borrará la pantalla, mostrando un mensaje parecido a éste:

CARGANDO QL ARCHIVE
Base de datos
Versión X.XX
Copyright © 1984 Psion Ltd.
Reservados todos los derechos

El número indicado como X.XX es el número de la versión del programa. Indica lo reciente que es la versión que posee. Los programas, como los libros, se revisan y reeditan continuamente, en nuevas ediciones o versiones. Cuanto más alto sea el número, tanto más reciente es su copia. La primera edición española corresponde a la versión 2.21.

En este libro se indicarán las diferencias entre la versión española y las anteriores, cuando sea relevante.

Si posee una versión anterior a la 2.21, tenga en cuenta que los nombres de los comandos aparecen en inglés. Esta diferencia no se marcará en el libro.

Después de una corta espera, aparecerá la pantalla principal de ARCHIVE, dividida en tres zonas.

ARRANCANDO ARCHIVE DESDE EL BASIC

Si seleccionó un tipo de pantalla antes de insertar el cartucho, o si acaba de abandonar otro programa, estará en la pantalla de BASIC del QL.

Para comenzar ARCHIVE:

1. Inserte la copia del cartucho ARCHIVE en el cartucho izquierdo (unidad 1).
   Asegúrese de introducir un cartucho vacío y formateado en la unidad 2.

lrun mdv1_boot ↵

2. Escriba lrun mdv1_boot y pulse ↵.
   Así le pide al QL que busque un fichero llamado boot en el cartucho de la unidad 1, que lo lea desde la cinta y que lo ejecute. Este programa (boot) a su vez busca ARCHIVE y lo lee al QL.
   Recuerde el espacio tras lrun, y el "_".
   El *microdrive* se pondrá en funcionamiento y aparecerá la presentación inicial de ARCHIVE, como se describió anteriormente.
   En la jerga informática, cuando se habla de "botar" un ordenador se indica que el ordenador busca y lee automáticamente el primer programa de un disco, de manera que se puedan usar desde él programas mayores. El programa "botador" (en inglés "boot") es el primer programa ejecutado en la mayoría de los ordenadores. El comando lrun del SUPERBASIC es una abreviatura de "load" (carga) y "run" (literalmente, corre).

Ancho de la visualización

Las televisiones domésticas se fabrican para ver películas (y anuncios), pero se pueden usar también con el QL. Los monitores, en cambio, se diseñan para trabajar con los ordenadores y muestran la información con mayor claridad. Por eso, al monitor "le caben" 80 caracteres en cada línea. Una televisión lo puede hacer también, pero sus ojos le agradecerán que use sólo 64. El programa dispone también de la posibilidad de una visualización de 40 caracteres de anchura (véase el comando modo en el capítulo 10). Esta presentación es innecesaria, a menos que su TV esté en un estado realmente malo. La visualización de ARCHIVE varía ligeramente según el tipo de pantalla que utilice. Su esencia, sin embargo, será la misma, utilice la opción que utilice.

Nota: Los ejemplos se han escrito suponiendo que la visualización es de 64 columnas en una TV. El ancho de la visualización no cambia la forma de trabajar de ARCHIVE, pero podría serle necesario cambiar la disposición de alguna de las pantallas que diseñe.

LA PANTALLA

De momento se encuentra en la primera pantalla (y principal) de ARCHIVE. Si la imagen no está bien definida, ajuste los controles de "color", "brillo" y "contraste" de su televisor. En ocasiones la posición idónea para el QL no es la que usted elige para su programa favorito.

Observe que la pantalla se divide en tres áreas. Son el AREA DE CONTROL, el AREA DE TRABAJO y el AREA DE VISUALIZACION.

Area de control

El rectángulo de la parte superior de la pantalla es el AREA DE CONTROL. Está ahí para recordarle qué hacer. De momento se puede:

- Obtener AYUDA pulsando **F1.**
- Eliminar y volver a presentar el AREA DE CONTROL pulsando **F2.**
- Pasar la página de comandos pulsando **F3.**
- Escribir un comando.
- Pulsar la tecla **ESCAPE** (aunque no tendrá ningún efecto en este momento).

Area de trabajo

El área de la parte inferior de la pantalla es el AREA DE TRABAJO. Aquí se escriben los comandos, y ARCHIVE le presenta mensajes de error si cometió alguna equivocación. También sirve para introducir y editar las líneas de programa.

Comandos

Como todos los programas de ordenador, ARCHIVE necesita que alguien le diga qué hacer: le debe dar órdenes. Un comando es una instrucción que usted escribe. Le dice a ARCHIVE que cumpla una tarea concreta. ARCHIVE vuelve a leer otro comando cuando acaba el anterior. Tenga en cuenta que en ARCHIVE los nombres de los comandos se deben introducir completos, y no la inicial como en QUILL, ABACUS y EASEL.

ARCHIVE entiende todos los comandos que aparecen en la lista del AREA DE CONTROL, y usted puede, además, crear comandos propios.

Algunos comandos necesitan más información (otras palabras o números) que especifiquen la orden completamente. La longitud máxima de una línea de comandos es de 128 caracteres, incluyendo los espacios entre palabras.

Petición de comando

Cuando introduce un comando, las letras aparecen junto a un signo > en el AREA DE TRABAJO. El símbolo ">" (en inglés *prompt*) le indica que ARCHIVE está esperando que escriba un comando. La línea que se está escribiendo se llama línea de entrada o línea de comando.

Area de visualización

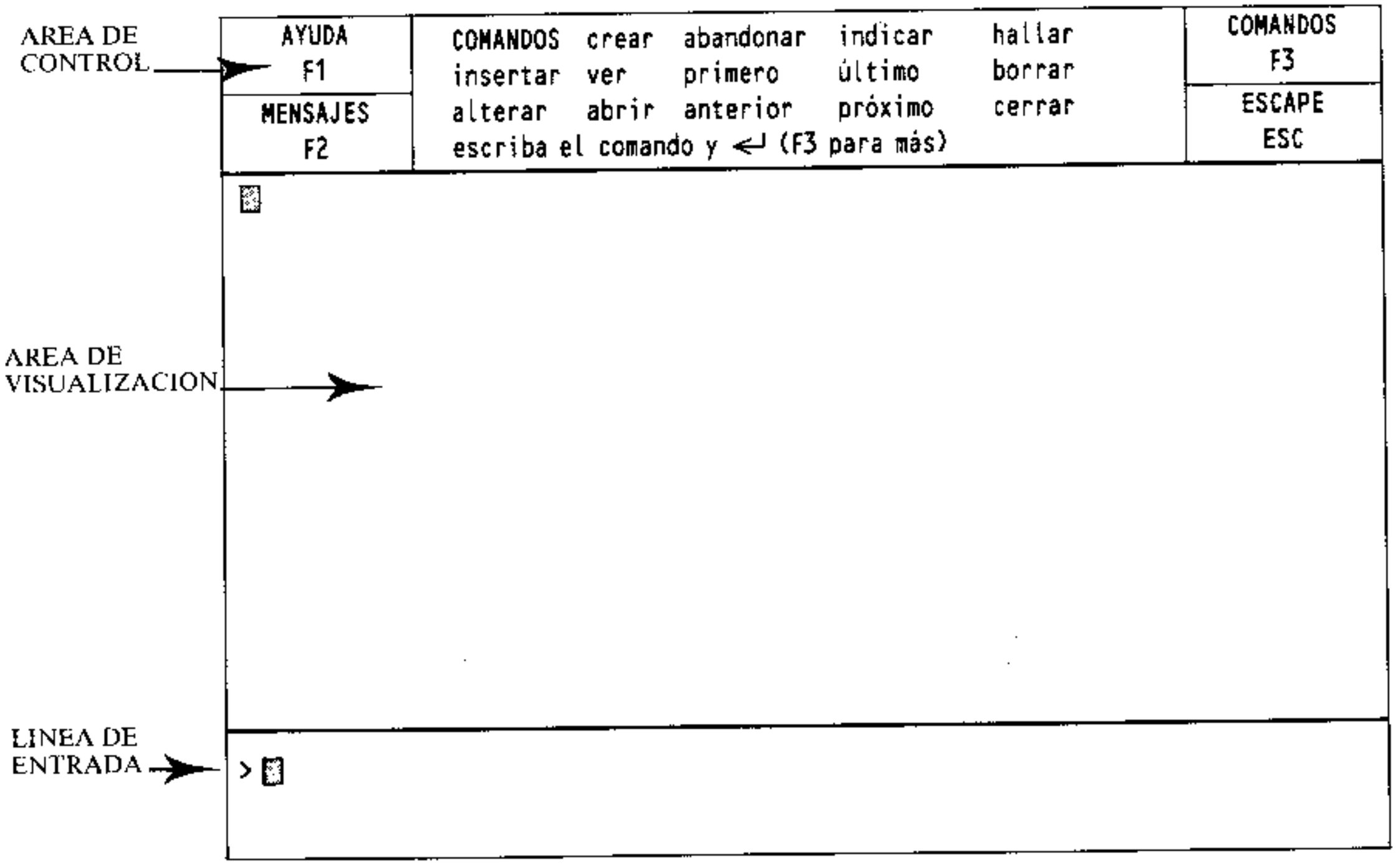
La parte central de la pantalla es el AREA DE VISUALIZACION, y en ella se muestran los resultados de las instrucciones.

Notas sobre la pantalla

AREA DE CONTROL
Le recuerda las opciones de que dispone; indica qué se puede hacer.

AREA DE TRABAJO
En ella se introducen los comandos y líneas de programa.

AREA DE VISUALIZACION
Imprime los resultados de los comandos, y también la información que solicite.



CARACTER DE PETICION

Le indica que ARCHIVE espera la introducción de una orden.

EL CURSOR

Un rectángulo rojo (gris si su TV es en blanco y negro) que se sitúa donde aparecerá el texto que escriba a continuación.

USO DE LAS TECLAS DE FUNCION

Las teclas de la parte izquierda del teclado se llaman TECLAS DE FUNCION. Llevan escrito **F1, F2, F3, F4** y **F5.** Damos a continuación una corta indicación de su misión.

F1: la tecla de AYUDA

Los mensajes de AYUDA se almacenan en la copia de ARCHIVE; asegúrese de que hay un cartucho ARCHIVE en la unidad 1 (si quiere usar la AYUDA).

F1 Se le puede pedir ayuda al programa en cualquier momento pulsando **F1.** Hágalo ahora. El *microdrive* gira, la pantalla cambia y aparece una lista de temas. El cursor se queda en la parte inferior de la lista, junto a un signo de interrogación. ARCHIVE le pregunta sobre qué parte de sus posibilidades quiere ser informado.

En este momento la mayor parte de los elementos de la lista no significan nada para usted. Para saber más, seleccione uno de los asuntos.

Basta con pulsar suficientes letras para distinguir la palabra del resto de la lista.

Para saber más sobre "Ficheros (Comandos)":

fi **1.** Introduzca fi o FI (valen las dos).
Se necesitan dos letras para distinguir esta opción de FUNCIONES.

↵ **2.** Pulse ↵.
La pantalla se vacía y aparece información sobre los comandos asociados con ficheros. Se dispone de ayuda adicional sobre cada uno de los comandos. Para saber más sobre create:

cr **3.** Escriba cr.

↵ **4.** Pulse ↵.
La pantalla vuelve a vaciarse y ARCHIVE muestra información sobrc crear.

Algunas opciones le proporcionan nuevas listas. Haga unas incursiones por las PANTALLAS DE AYUDA, si lo desea. No se preocupe si no entiende todo lo que ve: irá adquiriendo sentido según avanza por el libro.

Vuelva a la pantalla principal de una de las siguientes maneras:

bien

ESC Pulse la tecla **ESCAPE.**
Así volverá inmediatamente al punto en que solicitó ayuda.

o bien

↵ ← ↵ Pulse ↵ y volverá a la PANTALLA DE AYUDA anterior. Cada vez que pulse ↵, ARCHIVE le devuelve a la pantalla anterior, hasta que alcance la pantalla principal de ARCHIVE, donde pulsó **F1.**

Se puede pulsar la tecla de AYUDA en cualquier momento, sin ningún miedo a perder lo que está haciendo. La PANTALLA DE AYUDA que aparece siempre tiene una relación con lo que

está haciendo. Si, por ejemplo, pulsa AYUDA mientras utiliza el comando insertar, aparece información sobre ese comando. Al acabar pulse **ESC** o ↵, y ARCHIVE le devuelve al sitio exacto en que se encontraba cuando pulsó **F1.**

Si el cartucho de ARCHIVE no se encuentra en la unidad izquierda, ARCHIVE le responde con el mensaje "***NO HAY AYUDA*** Pulse ESPACIADOR para continuar". Pulse la barra espaciadora y volverá a lo que estaba haciendo. En ningún caso pierde el trabajo realizado.

F2: eliminando el AREA DE CONTROL

Cuando comience a estar familiarizado con ARCHIVE, no tendrá necesidad de que le recuerden qué hacer a continuación. Se puede borrar temporalmente el AREA DE CONTROL pulsando **F2.**

F2 Hágalo ahora.
El AREA DE CONTROL desaparece, y el AREA DE VISUALIZACION ocupa ahora toda la parte superior de la pantalla. Vuelva a pulsar **F2** y verá cómo reaparece el AREA DE CONTROL. Se puede utilizar esta tecla en cualquier momento, no importa lo que esté haciendo.

F3: la lista de COMANDOS

Si pulsa **F3,** la parte central del AREA DE CONTROL mostrará más comandos de ARCHIVE.

F3 Pulse **F3:** aparece otro conjunto de comandos.
F3 F3 Púlsela dos veces más.
Cada vez que lo haga aparecen nuevos comandos en el rectángulo. Púlsela por cuarta vez y volverá a aparecer la primera lista (que comienza por crear).

Se pueden usar todos los comandos en cualquier momento. Las cuatro páginas se utilizan para facilitar la impresión, así que, al revés que en QUILL, no es necesario cambiar de página.

F4

Esta tecla cumple misiones distintas dependiendo del momento en que se use. Se discuten más adelante, en los capítulos pertinentes.

F5

Esta tecla sirve para tres cosas, en función del momento de uso:

- Almacena la información que usted introduce y que debe ser grabada en el cartucho de *microdrive*.
- Permite la edición de una línea cuando se introducen programas.
- Devuelve a la línea de comandos (tras la demanda ">") la última línea introducida, para que pueda ser corregida o reutilizada.

EDICION DE TEXTO EN ARCHIVE

Introducción y edición de líneas

Sitúese en la pantalla principal de ARCHIVE, con el cursor inmediatamente a continuación de la demanda ">".
Escriba lo que se indica en los ejemplos siguientes.

Luis

1. Escriba su nombre.
Las letras aparecen tras el >. El cursor se mueve a la derecha cada vez que pulse una letra. El siguiente carácter aparece en la posición del cursor, y éste se vuelve a desplazar a la derecha.

2. Pulse ↵.
Así le indica a ARCHIVE que ha finalizado la línea, y que está dispuesto a que éste la interprete y actúe en consecuencia. Como su nombre no es un comando, ARCHIVE escribe un mensaje de error. Con este mensaje el programa le indica que no entiende el comando que usted le ha introducido. Hay una descripción detallada de los mensajes de error en el apéndice "Si tiene problemas".

3. Pulse la tecla de función **F5.**
Reaparece su nombre. **F5** siempre vuelve a escribir el último comando o línea de comandos introducida. Esto resulta muy útil: si introdujo algún error, puede volver a escribir la línea, corregirlo (mediante las teclas de edición que se muestran a continuación), y continuar. Practíquelo con su nombre.

4. Pulse la flecha hacia abajo.
El cursor salta al final de la línea.

[⇧] **5.** Pulse la flecha hacia arriba.
El cursor vuelve al principio.

[⇳] **6.** Pulse ⇳.
Cualquier letra que pulse a continuación aparece en mayúsculas hasta que desbloquee las mayúsculas (mediante ⇳ otra vez). La tecla sólo afecta a las letras.

NOMBRE **7.** Escriba la palabra NOMBRE.

[ESP] **8.** Pulse la **BARRA ESPACIADORA** para insertar un espacio entre la palabra NOMBRE y su nombre, para facilitar la lectura. Observe cómo las letras mayúsculas aparecen a la izquierda, empujando a las letras existentes hacia la derecha.

[⇳] **9.** Pulse el bloqueo de mayúsculas de nuevo.

[⇨] **10.** Pulse ⇨ y manténgala apretada.
El cursor se mueve sobre las letras y espacios sin alterarlos. Cuando llega al final de la línea se detiene, ya que sólo se mueve a través del texto. Levante el dedo de la tecla. Si mantiene una tecla pulsada, el QL actúa como si la pulsara repetidamente.

[⇧][⇦] **11.** Pulse la flecha izquierda mientras mantiene pulsada la tecla de mayúsculas. El cursor salta al comienzo de la palabra.

Borrado de texto

[CTRL][⇦] **12.** Mantenga pulsada la tecla de **CONTROL** y pulse la flecha izquierda una vez. Si no está al comienzo de la línea, se borrará el carácter inmediatamente a la izquierda del cursor, y la línea se cierra para tapar el hueco.

[CTRL][⇨] **13.** Mantenga la tecla de **CONTROL** y pulse la flecha derecha una vez. Esta vez se borra el carácter bajo el cursor.

Inserción de texto

14. Para recuperar los caracteres borrados basta con teclearlos de nuevo. Aparecen bajo el cursor, moviéndose hacia la derecha el carácter que ocupaba esta posición.

Movimiento a través del texto

[⇧][⇨] **15.** Para saltar una palabra hacia la derecha, pulse mayúsculas y la flecha derecha.

CTRL ⇧ **16.** Para borrar toda la línea a la izquierda del cursor, pulse **CONTROL** y la flecha superior. La línea entera desaparece.

ESC **17.** Pulse **ESCAPE.**
ARCHIVE escribe <ESC> bajo la linea y se le presenta una nueva demanda > más abajo.
Esta vez no aparece ningún mensaje de error, ya que pulsar **ESC** en una introducción de línea causa el abandono de la línea. El programa ignora lo que haya introducido.

TECLAS USADAS EN EDICION. EL EDITOR DE LINEA

TECLAS DEL CURSOR

Movimiento

⇩ Mueve el cursor al fin de línea.
⇧ Mueve el cursor al comienzo de línea.
⇨ Mueve el cursor un carácter a la derecha.
⇦ Mueve el cursor un carácter a la izquierda.
⇧ ⇨ Mueve el cursor una palabra a la derecha.
⇧ ⇦ Mueve el cursor una palabra a la izquierda.

Borrado

CTRL ⇨ Borra el carácter bajo el cursor.
CTRL ⇦ Borra el carácter a la izquierda del cursor.
CTRL ⇩ Borra todo el texto a la derecha del cursor.
CTRL ⇧ Borra todo el texto a la izquierda del cursor.

ABANDONAR ARCHIVE

El comando ABANDONAR

Cuando desee abandonar ARCHIVE, no se limite a apagar el ordenador. El comando abandonar "apaga" ARCHIVE sin riesgo de pérdida de ningún dato que haya introducido.

abandonar ↵

1. Escriba abandonar.
2. Pulse ↵.
   ARCHIVE vuelve a la pantalla que vio cuando conectó el QL (la que le pide que pulse **F1** o **F2**), en el caso de la versión 1, o bien a la pantalla BASIC del QL (ARCHIVE versión 2).
3. Saque ahora los cartuchos del *microdrive* y apague, o bien utilice otro programa.

> *Nota:* Si utiliza abandonar en la versión 2, no puede volver a comenzar con "lrun". Debe sacar los cartuchos, pulsar **RESET** y comenzar de nuevo.

Vuelva a este capítulo si siente la necesidad de repasar algo. El significado de todo lo dicho se hará más claro a medida que avance por el libro.

2

El primer archivo

El propósito principal de ARCHIVE es almacenar y recuperar información. Esta información se almacena en la cinta en archivos o ficheros, donde se agrupan datos relacionados entre sí. Al final de este capítulo habrá aprendido, de un modo sencillo, cómo almacenar y recuperar información de un fichero.

Se presentan las siguientes características de ARCHIVE:

- Comandos de manejo de ficheros de ARCHIVE:

crear	— crea una estructura de archivo
dir	— lista los ficheros que hay en el cartucho
cerrar	— cierra y graba un archivo

- Comandos de manejo de registros de ARCHIVE:

indicar	— muestra un archivo
insertar	— inserta información en un fichero
alterar	— altera la información de un fichero
próximo	— avanza por el archivo
anterior	— retrocede por el archivo
primero	— se sitúa en el primer registro de un fichero
último	— se sitúa en el último registro de un fichero
hallar	— busca un dato
continuar	— continúa la búsqueda

Introducción

Este capítulo explica la función primaria de ARCHIVE: el almacenamiento y recuperación de información. Usaremos ARCHIVE para crear y utilizar un fichero de base de datos simple. Los detalles del uso de estos comandos serán cubiertos más ampliamente en el próximo capítulo.

Según avance por los ejemplos es posible que no entienda algunos mensajes del AREA DE CONTROL. También se explican más adelante. No es importante que los entienda en este momento.

No se preocupe si comete errores. Si hace algo que ARCHIVE no comprende, aparecerá un mensaje de error. Basta con introducir el texto correcto de nuevo.

Ejemplo: Organización de la boda de Eva

Eva utiliza ARCHIVE como ayuda en la organización de su Gran Día cuando no contempla extasiada a Salva. Tiene la lista de invitados en el ordenador, y su idea es usar ARCHIVE para enviar las invitaciones de boda, agradecer las respuestas y llevar la cuenta de la gente que viene y los regalos. Juega con la idea de permitir a ARCHIVE la colocación de los invitados en el banquete: peor que su madre no lo puede hacer, y, además, seguro que organiza menos enredos.

Sigue pensando en lo que tiene que hacer, antes y después de la boda, y recordando cosas que ya debería haber hecho. Mirando su revoltijo de papeles, con listas que ha vuelto a escribir varias veces, decide informatizar sus notas además de su libro de direcciones.

CREACION DE UN ARCHIVO

La primera tarea de Eva es usar ARCHIVE para crear un archivo llamado agenda.

El proceso es parecido a pegar una etiqueta en una carpeta, para llenarla a continuación de fichas y almanecarla en un archivador: un archivo mantiene información sobre un asunto, y separada de otros archivos. El nombre del fichero permite recuperar la información con rapidez.

Ver al sacerdote
Pedirle a Felicidad que lea la cartilla
Decirle a Juana que traiga los trajes
Recordarle a Salva que me traiga su agenda
Decirle al Sr. Hort que se encargue de las flores
Que no se olvide hacer los ojales
Zapatos blancos
¿Compro un traje o llevo el de Mamá?
¿Ha comprado Salva un traje de sport?
fotógrafo
Coche
Llamar a Sara. ¿Quién le organizó el banquete?
Capacidad de la sala Azul del Hotel Almirante
Encargar la tarta

Figura 2.1. La agenda de Eva

Instrucciones

1. Asegúrese de tener en memoria el programa ARCHIVE (véase el capítulo anterior), y de tener un cartucho en blanco formateado en la unidad 2.

crear ↵

2. Escriba crear y pulse ↵.
ARCHIVE coloca un par de comillas tras crear.

agenda

3. Escriba agenda.
Este es el nombre del nuevo archivo. Lo que escriba aparece entre las comillas. Si comete algún error, borre los caracteres introducidos mediante el editor de línea (véase el final del capítulo 1).

Una vez corregidos los errores:

4. Pulse ↵.
El cursor pasa a la línea siguiente, sin mostrar el ">", y el AREA DE CONTROL cambia. Le indica las opciones disponibles en este momento.
ARCHIVE espera la información adicional necesaria para crear un nuevo fichero de datos: un nombre o etiqueta para la información que se va a almacenar. En este caso queremos almacenar notas.

notas$ **5.** Teclee notas$.
La información se almacenará bajo este nombre. Fíjese en el signo del dólar "$" al final del nombre (⇑ y 4). Le indica a ARCHIVE que bajo el nombre 'notas$' se almacenarán letras y números. Si falta el dólar, ARCHIVE sólo le dejará almacenar números.

6. Pulse ↵.
El cursor baja una línea. El AREA DE CONTROL no cambia, ya que ARCHIVE espera otro nombre, o bien una indicación de que la creación del fichero está terminada. No queremos almacenar nada más, y debemos indicarle al programa que hemos terminado. Si comete un error, vaya a la sección titulada "Arreglando los errores en crear" a continuación.

7. Pulse ↵ otra vez.
De esta manera se acaba el comando crear. ARCHIVE supone que se ha acabado si se pulsa ↵ en una línea en blanco. La palabra fincrear aparece automáticamente, y el *microdrive* gira. ARCHIVE está salvando los detalles del archivo en el cartucho.

> *Nota:* Podría haber introducido fincrear, pero es más fácil dejar que lo haga ARCHIVE por usted. De cualquier manera, crear debe terminar con fincrear, o el archivo no se creará.

Reaparece el signo ">", listo para otro comando. El AREA DE CONTROL muestra de nuevo la lista de comandos.

ARCHIVE no indica el éxito en la creación del fichero. La única indicación es la ausencia de errores. ARCHIVE tampoco hace ningún tipo de suposiciones sobre lo que usted quiere hacer con el fichero que acaba de crear.

El fichero está ahora "abierto", y listo para recibir la información que quiera almacenar.

Arreglando los errores en CREAR

Errores en una línea:

Si nota que ha introducido un error en una línea antes de pulsar ↵, se puede editar la línea mediante las teclas normales (véase el final del primer capítulo).

Errores en una línea anterior:

No se puede volver una línea atrás en crear. Si pulsa **F5** para reeditar una línea previa, o no funcionará o tendrá algún otro efecto. Debe abandonar crear y comenzar de nuevo.

Para abandonar:

Para abandonar crear completamente, basta pulsar **ESC** antes de haber escrito fincrear. ARCHIVE escribe ‹ESC› y reaparece el "›". Todo lo introducido en crear se ignora y el fichero no se crea ni se almacena en el cartucho. Para volver a empezar, escriba crear tras el "›". La estructura del archivo se salva sólo si se introduce fincrear.

Mensajes de error:

ARCHIVE necesita que se le indique qué hacer de una manera muy precisa. Si ARCHIVE no entiende lo que usted teclea, aparece un mensaje de error en el AREA DE TRABAJO, se abandona el último comando introducido y reaparece el "›". En este caso, todo lo introducido en crear es ignorado y no tienen ningún efecto. Para comenzar, escriba crear tras el "›".

INSERCION DE INFORMACION EN UN ARCHIVO

Habiendo creado su archivo, Eva está lista para añadirle información.

insertar 

1. Escriba insertar y pulse ↵.
   El AREA DE CONTROL cambia para mostrar algunas de las teclas que ARCHIVE espera que use. Ignórelas de momento.

El cursor del AREA DE TRABAJO se quita de en medio y se queda quieto encima del "›". Un segundo cursor aparece en el AREA DE VISUALIZACIÓN. Durante insertar los caracteres

que escriba aparecerán sobre él. Hasta que reaparezca el ">", ARCHIVE espera que inserte registros (fichas) en el fichero (ARCHIVE llama registros a los elementos que componen un fichero; en este caso, cada nota es un registro).

En la línea superior del AREA DE VISUALIZACION aparece:

nombre lógico: maestro

Es un nombre temporal que ARCHIVE le asigna al archivo. No tiene nada que ver con la lógica; su utilidad queda clara más adelante. De momento puede ignorarlo.

La pantalla contiene la siguiente información:

```
AREA DE VISUALIZACION:      nombre lógico: maestro
                       :    notas$         : [ ]
AREA DE TRABAJO        :    >crear "agenda"
                       :    notas$
                       :    fincrear
                       :    >insertar
```

ARCHIVE está esperando que introduzca su primera anotación.

Ver al sacerdote

4. Escriba Ver al sacerdote.
Mientras teclea, aparecen las letras tras notas$: en el AREA DE VISUALIZACION.

F5

5. Pulse **F5** (o ↵ en la versión 2) para salvar el registro y continuar con el siguiente. (Si pulsa ↵ en la versión 1, el cursor se limita a volver al comienzo de la línea.) La nota que acaba de introducir desaparece. El cursor se sitúa de nuevo junto a los dos puntos. El registro que contiene la nota Ver al sacerdote ha sido insertado en el archivo agenda. ARCHIVE no le indica que el registro ha sido añadido.

ARCHIVE espera que inserte más registros. El comando insertar sólo permite añadir nuevas fichas al fichero. No se puede mirar los registros existentes.

Pedirle a Felicidad que sea la madrina

6. Escriba Pedirle a Felicidad que sea la madrina.
Si comete un error, edite la línea mediante las teclas del cursor.

F5

7. Cuando la línea sea correcta, pulse **F5**.
La línea recién introducida desaparece y el cursor vuelve a los dos puntos.

Decirle a Juana que haga los trajes F5

8. Escriba Decirle a Juana que haga los trajes y pulse **F5**.

Las teclas de función y ↵ en INSERTAR

Las versiones 1 y 2 usan teclas distintas en insertar.

Para salvar un registro

Versión 1	**F5**
Versión 2	**F5** o ↵ en el último campo

Si pulsa ↵ en el último campo en la versión 1, el cursor vuelve al primer campo.

Para dejar INSERTAR

Versión 1	**ESC**
Versión 2	**F4**

F4 es mejor para salir de insertar que **ESC**, ya que la última tecla implica un error.

El fichero agenda debe contener tres anotaciones a estas alturas:

Ver al sacerdote
Pedirle a Felicidad que sea la madrina
Decirle a Juana que haga los trajes

Para abandonar INSERTAR

Un cliente que se enreda con el rosal invasor de los vecinos, obliga a Eva a dejar de engordar el archivo. Pero, antes de correr a deshacer el agravio, quiere echarle una ojeada a lo que acaba de escribir. Antes debe indicarle a ARCHIVE que acabó de insertar.

F4

1. Pulse **F4** (en la versión 1, pulse **ESC**).
   El AREA DE VISUALIZACION muestra ahora un registro vacío. El ">" y el cursor reaparecen en el AREA DE TRABAJO. ARCHIVE espera otro comando.

Nota: **ESC** abandona insertar en ambas versiones de ARCHIVE. En la versión 2 se puede abandonar insertar pulsando **F4**. Es mejor hacerlo así, ya que **ESC** origina un "error" que puede detener los programas que usted escriba (se discute en el capítulo 9).

Para mostrar información

Eva usa el comando indicar para examinar su hábil trabajo.

indicar ↵

1. Escriba indicar y pulse ↵.
Aparece el último registro que introdujo. Es el registro activo (el último utilizado): el que siempre presenta indicar.

MOVIMIENTO POR EL FICHERO

El comando ANTERIOR

Para ver los otros datos introducidos:

anterior ↵

1. Introduzca anterior y pulse ↵.
ARCHIVE presenta inmediatamente el registro anterior, que se convierte, además, en el registro activo.
Reaparece el ">" en la línea siguiente del AREA DE TRABAJO.

Para volver atrás sin tener que escribir anterior:

F5

2. Pulse **F5.**
Vuelve a aparecer la palabra anterior tras el ">". **F5** presenta de nuevo la última línea que se introdujo.

Para ejecutar el comando:

↵

3. Pulse ↵.
ARCHIVE retrocede un registro por el archivo y lo muestra en el AREA DE VISUALIZACION.

F5

4. Vuelva a pulsar **F5**.
La palabra anterior vuelve a aparecer.

↵

5. Pulse ↵.
Esta vez la pantalla parpadea un poco, pero no cambia su contenido. Está mirando al primer registro del archivo. No existe un registro anterior. ARCHIVE no lo indica, y se limita a escribir de nuevo el primer registro.

Los comandos ULTIMO, PROXIMO y PRIMERO

Para ver el último registro del archivo activo:

último ⏎

1. Escriba último y pulse ⏎.
El AREA DE VISUALIZACION muestra ahora el último registro introducido.

próximo ⏎

2. Introduzca próximo y pulse ⏎.
El AREA DE VISUALIZACION no cambia. No hay ningún registro detrás del último. ARCHIVE, de nuevo, no informa de que se ha llegado al final de archivo.

primero ⏎

3. Escriba primero y pulse ⏎.
Se encontrará mirando nuevamente el primer registro del archivo.

próximo ⏎

4. Introduzca próximo y pulse ⏎.
El AREA DE VISUALIZACION muestra el siguiente registro del archivo. En términos de ARCHIVE, éste pasa a ser el registro activo.

El registro activo

El registro activo es el que ARCHIVE está manipulando en un momento determinado; no tiene por qué ser el que se está viendo en la pantalla. Es como la página por la que está abierto un libro. Al abrir por primera vez el libro, la primera página es la activa. Al ir volviendo las páginas, acaba por serlo la última. En ARCHIVE el registro activo es el último que se ha cambiado, impreso, añadido o cambiado. Algunos comandos actúan sobre una secuencia de registros; en ese caso, el último afectado al acabar el comando se convertirá en registro activo, aunque hay algunas excepciones.

Los comandos que afectan registros actúan sobre el registro activo, a menos que usted especifique lo contrario.

ADICION DE NOTAS AL FICHERO AGENDA

Eva quiere introducir algunas notas más en su fichero.

Nota: ARCHIVE debería permitir introducir un máximo de 160 caracteres, espacios incluidos, en cada línea mediante insertar. Sin embargo, introducir tantos caracteres no resulta práctico. Si añade algún carácter tendrá problemas al escribir la nota sobre las flores. En la versión 1 lo que escriba continuará en la línea siguiente al llegar a la derecha de la pantalla. Al mostrar el siguiente registro, la parte sobreimpresa puede permanecer en la pantalla. En algunas versiones de ARCHIVE no se le permite sobrepasar una línea. Mantenga las líneas tan cortas como sea necesario.

Se puede introducir hasta 255 caracteres en una línea de un fichero de base de datos de ARCHIVE, pero no mediante insertar. Más adelante se describen otros métodos.

insertar ⊡

1. Introduzca insertar y pulse ↵ .

Recordarle a Salva que me traiga su agenda F5

2. Introduzca la siguiente nota de Eva.

Decirle al Sr. Hort que se encargue de las flores F5
Que no se olvide hacer los ojales F5
Zapatos blancos F5
¿ Compro un traje o llevo el de Mamá ? F5
¿ Ha comprado Salva un traje de sport ? F5
Fotógrafo F5
Coche F5
Llamar a Sara. ¿ Quién les organizó el banquete ? F5
Capacidad de la sala Azul del Hotel Almirante F5
Encargar la tarta F5

3. Escriba el resto de las notas de Eva, pulsando **F5** cuando acabe cada una para salvarla.

Una vez acabado el trabajo:

F4

4. Pulse **F4** para dejar de insertar (o **ESC** si tiene la versión 1).

Nota: En la versión 1, si pulsa ↵ en lugar de **F5** se pierde el último registro.

El fichero agenda contiene ahora las siguientes anotaciones:

```
notas$
Ver al sacerdote
Pedirle a Felicidad que sea la madrina
Decirle a Juana que haga los trajes
Recordarle a Salva que me traiga su agenda
Decirle al Sr Hort que se encargue de las flores
Que no se olvide hacer los ojales
Zapatos blancos
¿ Compro un traje o llevo el de Mamá ?
¿ Ha comprado Salva un traie de sport ?
Fotógrafo
Coche
Llamar a Sara ¿ Quién les organizó el banquete ?
Capacidad de la sala Azul del Hotel Almirante
Encargar la tarta
```

BUSQUEDA Y MODIFICACION DE REGISTROS

Después de pelearse con la labor desaliñada de Maya, Eva decide hacer una apresurada visita a *Pronupcias*. Para cambiar la anotación sobre el traje de boda,

hallar "tra" ↵

1. Escriba hallar "tra" y pulse ↵.
La pantalla mostrará la nota Decirle a Juana que haga los trajes.

El comando hallar va al principio del archivo y examina cada registro. ARCHIVE intenta hallar las letras entre comillas dentro del texto del registro y se detiene en el primer registro en que las encuentre.

continuar ↵

2. Introduzca continuar y pulse ↵.
La pantalla muestra el siguiente registro que contiene las letras "tra":

Recordarle a Salva que me traiga su agenda.

El comando continuar sigue la búsqueda emprendida por el último comando hallar, buscando en los restantes registros. Si no logra encontrar una coincidencia, permanece activo el último registro hallado (así como hallar deja activo el primer registro del archivo si no encuentra una ocurrencia).

F5 ↵

3. Pulse **F5** seguido por ↵.
ARCHIVE sigue buscando y llega al registro que quiere EVA: ¿ Compro un traje o llevo el de Mamá ? En este ca-

so, habría hecho mejor buscando " traje" o " Traje". El espacio en blanco delante de la palabra evita que aparezcan cosas como "DisTRAJEron", estrechando las posibilidades. El comando hallar encontrará las coincidencias independientemente de que estén escritas con mayúsculas o minúsculas.

Nota: Cuando utilice el comando hallar, introduzca el número mínimo de letras que distingan el registro de los demás. No es necesario introducir una palabra o frase entera.

Para alterar los registros

Maniática del detalle, Eva quiere cambiar la anotación en el registro:

alterar ⊡

1. Escriba alterar y pulse ↵.
   Observe que las indicaciones del programa son las mismas del comando insertar. El cursor del área de trabajo se queda sobre el ">" y aparece un nuevo cursor en el área de visualización, situado al principio del registro.

⇩

2. Pulse la flecha hacia abajo para mover el cursor al final de la línea.

CTRL ⇦ *(20 veces)*

3. Pulse **CTRL** a la vez que la flecha izquierda, para borrar o llevo el de Mamá ?

⇧

4. La flecha arriba nos devuelve al comienzo de la línea.

CTRL ⇨ *(2 veces)*

5. Borre la "¿" y el espacio en blanco.

↑ ⇨ ↵ CTRL ⇦ ar

6. Use mayúsculas con la flecha derecha para moverse al comienzo de la segunda palabra. La flecha izquierda le sitúa en el espacio en blanco. Borre la o e inserte ar, quedando Comprar un traje.

F5

7. Pulse **F5**. Reaparece el ">" en el AREA DE TRABAJO, esperando el siguiente comando. Al contrario que en insertar, no es necesario pulsar **F4** o **ESC** para abandonar el comando. ARCHIVE nos devuelve automáticamente al modo comandos tras modificar un registro.

Nota: Si decide (antes de pulsar **F5**) que la modificación no es válida, use **F4** en la versión 2, o **ESC** en la versión 1, para abandonar alterar.

Buscar y modificar más registros

Mientras se imagina vestida de tafetán rosa y beige, Eva recuerda el nombre del fotógrafo de Villanueva de Segre.

hallar ⏎

1. Escriba hallar y pulse ⏎.
Aparece un par de comillas tras el comando; el cursor se sitúa para que introduzca la palabra que quiere buscar. El AREA DE CONTROL le indica cómo usar hallar y le pide que introduzca la palabra o grupo de letras (espacios incluidos) que busca.

> *Nota:* Ciertos comandos, como crear y hallar, van seguidos por letras entre comillas. Si pulsa ⏎ tras el comando, ARCHIVE proporciona automáticamente las comillas, para que no las tenga que escribir usted.

fot

2. Escriba fot.
Aparecen las letras "fot" entre comillas.

⏎

3. Pulse ⏎.
El registro con la anotación "Fotógrafo" aparece en el AREA DE VISUALIZACION.

alterar ⏎

4. Escriba alterar y pulse ⏎.
De nuevo aparece un cursor al comienzo del registro.

C

5. Pulse C.
Al pulsar la primera letra desaparece el resto de la línea. ARCHIVE siempre borra una línea si se introduce una letra al comenzar a editarla.

> *Nota:* Si quiere insertar caracteres al comienzo de una línea en alterar, pulse primero la flecha hacia arriba seguida por las letras que quiere insertar.

ontratar a Pérez Fotógrafos F5

6. Escriba ontratar a Pérez Fotógrafos y pulse **F5**.

> *Nota:* Para abandonar alterar sin grabar los cambios realizados, pulse **F4** o **ESC**. La versión modificada del registro permanece en pantalla, pero no se graba en el cartucho. El registro original permanece intacto. Para comprobarlo, use los comandos anterior y próximo; así volverá a imprimirse el registro en la pantalla.

El fichero agenda contiene ahora las siguientes anotaciones:

```
notas$
Ver al sacerdote
Pedirle a Felicidad que sea la madrina
Decirle a Juana que haga los trajes
Recordarle a Salva que me traiga su agenda
Decirle al Sr Hort que se encargue de las flores
Que no se olvide hacer los ojales
Zapatos blancos
Comprar el traje
¿ Ha comprado Salva un traje de sport ?
Contratar a Pèrez Fotògrafos
Coche
Llamar a Sara ¿ Quièn les organizò el banquete ?
Capacidad de la sala Azul del Hotel Almirante
Encargar la tarta
```

PARA GRABAR EL ARCHIVO

Eva, contemplando orgullosa su trabajo, quiere grabar lo hecho, no sea que un apagón destruya el fruto de varias horas. Primero comprueba que los ficheros están en su *microdrive*. Entonces basta cerrar los archivos abiertos para que se grabe la información.

La memoria

El QL tiene dos clases de almacenamiento de datos y de programas: memoria temporal y memoria permanente.

La **memoria temporal** permanece mientras el QL está conectado. El QL y ARCHIVE hacen su trabajo en memoria temporal, ya que ésta almacena y recupera la información mucho más rápido que el almacenamiento permanente del QL: el *microdrive*.

La **memoria permanente** del QL es la cinta que hay en el interior del cartucho. En ella se almacenan los archivos igual que se usan los cassettes para guardar música. La información puede borrarse y regrabarse. Pese al nombre de memoria permanente, los cartuchos pueden perder su información por varias causas, desde daño físico hasta exposición a campos magnéticos.

La memoria temporal

En la jerga informática se suele llamar RAM a la memoria temporal. Son las siglas de *Random Access Memory* (Memoria de Acceso Aleatorio). La información se guarda mediante impulsos eléctricos en circuitos integrados, y se puede recuperar ("acceder") a ella en cualquier orden, es decir, al azar. La cinta, en cambio, tiene que esperar a que la información pase por delante de la cabeza lectora para recuperarla. Por otro lado, siendo el *microdrive* un dispositivo mecánico en vez de electrónico, las cintas son mucho más lentas que las señales eléctricas en la RAM.

La cantidad de memoria temporal es limitada, y todo lo que hace ARCHIVE usa parte de ella. El propio programa ocupa una buena cantidad, y cada archivo que se abre necesita algo.

Cada vez que se apaga el QL, o cada vez que abandonamos ARCHIVE, se pierde el contenido de la memoria temporal. Para mantener esta información debemos almacenarla en la cinta.

Otras clases de memoria permanente son los discos (más rápidos que las cintas) y los cartuchos de ROM, que son tan rápidos como la RAM, pero contienen información fija, como programas.

Listado de los archivos en un cartucho: el comando DIR

dir ⊡

1. Escriba dir y pulse ↵.
El AREA DE CONTROL cambia, indicándole que introduzca un nombre de unidad. Aparecen comillas tras el comando. ARCHIVE espera que le indique de qué *microdrive* debe leer el directorio.

⊡

2. Pulse otra vez ↵.
Si no le indica a ARCHIVE qué unidad debe mirar, el programa buscará los ficheros existentes en la unidad 2 (la derecha). Si no quiere que ARCHIVE piense por usted, introduzca un nombre de unidad entre las comillas, por ejemplo mdv1_.
El AREA DE VISUALIZACION muestra a continuación el directorio del cartucho en la unidad 2. Consiste en una lista de los ficheros almacenados en él. Muestra también el espacio libre que queda en el cartucho. El archivo de Eva aparece como agenda_dbf. Las letras finales indican *Data Base File* (Fichero de Base de Datos).

Consulte el recuadro sobre nombres de archivos si quiere más información.

Nota: dir muestra el espacio libre en el cartucho en términos de sectores. Cuando inicializa un cartucho (véase el capítulo 1), la cinta se divide en alrededor de 220 sectores. El mensaje 13/218 sectores indica que quedan 13 sectores libres en la cinta.

Para cerrar archivos

Cuando se creó el fichero agenda (en memoria temporal), la estructura vacía, que contiene los nombres de los datos (notas$), se grabó en el cartucho, pero sin datos, ya que aún no los había introducido.

A partir de entonces, usted introdujo varios registros al fichero, siempre en memoria temporal. Algunos, pero no todos, pueden haberse almacenado en la cinta a medida que se iba llenando el área de memoria reservada para agenda. La copia del archivo en la cinta no está completa.

ARCHIVE puede mirar, alterar o añadir información a un fichero cuya estructura está en memoria temporal. De estos ficheros se dice que están *abiertos*.

Los cambios y adiciones hechos a un fichero que está abierto tienen efecto en la memoria temporal. Para grabar estos cambios, los datos en memoria tienen que actualizar la información del archivo en cinta. Para hacerlo de una forma correcta hay que *cerrar* el archivo.

Cuando se cierra un archivo, toda la información sobre él en *memoria temporal* (incluyendo los registros modificados o insertados) desaparece. No se puede, por tanto, volver a mirar o modificar el fichero sin abrirlo de nuevo. Para abrir un archivo, el programa lee alrededor de 500 caracteres del fichero a la memoria temporal. Esto no afecta al fichero en cinta. Se queda exactamente igual que estaba, y contiene exactamente la misma información que el fichero que se acaba de abrir en memoria temporal.

Cerrar un archivo en ARCHIVE es como salvar un programa en SUPERBASIC, o unos datos de ABACUS, EASEL o QUILL.

Nota:

- El comando crear abrió el fichero agenda automáticamente; fincrear salvó el fichero vacío al cartucho. Si ya existía un fichero llamado agenda_dbf, habría sido sustituido por el fichero vacío sin ninguna advertencia.

- Al crear ficheros, sea cuidadoso de que el nombre elegido no coincida con ningún fichero existente, o perderá el fichero antiguo al ser sobreimpreso por el nuevo.

cerrar ⊡

1. Escriba cerrar y ↵.
El *microdrive* gira, y todos los registros del fichero se graban en el cartucho. El fichero en cinta está siendo sustituido por la copia del fichero con los registros que acaba de añadir. Igual que fincrear le indica a ARCHIVE que salve la estructura del fichero, cerrar le dice que salve los registros (el cuerpo del fichero) a la cinta.
Reaparece el ">" tras unos segundos. ARCHIVE está listo para una nueva instrucción, aunque el *microdrive* esté todavía funcionando.

Nota: Salve su trabajo cerrando los ficheros con frecuencia. Una vez abiertos, los cambios y adiciones iniciales sólo tienen lugar en la memoria del QL. Algunos registros pueden almacenarse en cinta si las necesidades de memoria lo requieren.

Si se va la luz o la máquina se desconecta por cualquier razón, todo el contenido de la memoria se pierde. Sólo la información que quede almacenada en la cinta se podrá usar de nuevo cuando vuelva a conectar el QL. Los ficheros que no hayan sido cerrados correctamente, mediante cerrar o abandonar, puede que no conserven sus datos.

Es sorprendente lo frustrante que resulta perder algo de lo ya hecho, aunque sea el trabajo de diez minutos.

Nombres de unidad

Cualquier nombre de fichero puede venir precedido de un nombre de unidad, para que ARCHIVE sepa en qué unidad debe buscar la información. Si no se indica un nombre, ARCHIVE busca en el *microdrive* de la derecha, unidad 2. Un fichero en el *microdrive* 2 se puede especificar como mdv2_agenda_dbf, donde mdv2_ es el nombre de unidad, agenda el nombre de fichero y _dbf es la extensión.

Observe que un nombre de unidad está incompleto si no lleva un subrayado tras él, incluso cuando lo utiliza sólo (por ejemplo, dir "mdv1_").

Nombres de archivo

En ARCHIVE, un nombre de fichero completo consta de tres partes, separadas por un "_", el carácter de subrayar. Las tres partes (en el orden correcto) son:

(nombre de unidad)_(nombre de fichero)_(extensión)

Nombres de fichero:

- Se usan para identificar los ficheros almacenados en los cartuchos.
- Pueden tener hasta ocho caracteres de longitud.
- Deben comenzar por una letra.
- No pueden incluir espacios en blanco.

Extensiones a los nombres de fichero:

- Tienen hasta tres caracteres de longitud (letras o números).
- Van unidos al nombre de fichero por un subrayado "_".
- Se usan para identificar tipos o grupos de ficheros en el cartucho.

"agenda_dbf" es un fichero de base de datos, que se distingue de un fichero de comandos, ya que este último acaba en _prg.

Normalmente no es necesario escribir la extensión, ya que ARCHIVE la proporciona automáticamente. Si usted quiere, puede indicar una extensión. Por ejemplo, la extensión _vie (en vez de _dbf) puede servir para marcar viejas versiones de sus ficheros, que no están actualizadas, pero puede ser interesante conservar. Sus extensiones tienen prioridad sobre las de ARCHIVE, pero debe recordar indicarlas cada vez que haga referencia al fichero.

Los comandos siguientes crean o esperan las siguientes extensiones:

_dbf:	crear, abrir, mirar, importar
_prg:	salvar, leer, unir, ejecutar
_scn:	psalvar, pleer
_lis:	via
_exp:	importar, exportar

RESUMEN

Eva ha creado un nuevo fichero ARCHIVE llamado agenda mediante los comandos crear y fincrear. Prepara un encabezamiento llamado notas$, y utiliza insertar para introducir una serie de anotaciones bajo esa cabecera. El signo del dólar ($) tras notas indica que los elementos del registro contendrán texto.

Las instrucciones anterior, primero, último y próximo le sirven a Eva para moverse por el archivo y ver las anotaciones. Usa también los comandos alterar y hallar para encontrar una posición determinada en el archivo y cambiar un registro. El comando cerrar se utiliza para transferir el fichero ARCHIVE desde la memoria temporal del QL a su almacenamiento permanente.

Trucos útiles

- Utilice mayúsculas y minúsculas para los datos (excepto en códigos especiales). Es mucho más fácil leer el texto que si escribe en mayúsculas.
- No use líneas que no quepan en la pantalla. ARCHIVE puede almacenar más caracteres, pero es más difícil modificar los datos. No suele haber necesidad de introducir líneas largas. La mayor parte de la información se almacena con el propósito de imprimirla alguna vez, y el papel normal sólo tiene 80 caracteres de ancho.
- Pulse **F5** para repetir los comandos. Use las teclas de edición para cambiar el comando si causó un error por estar mal escrito.

3

Archivos múltiples

En este capítulo aprenderá cómo crear ficheros más complejos y a manejar varios archivos simultáneamente.
Se explican los siguientes comandos de ARCHIVE:

- Se examinan con detalle:

crear	— crea una estructura de archivo
indicar	— muestra el fichero activo
insertar	— inserta registros
alterar	— modifica registros
cerrar	— cierra un fichero de datos

- Se presentan los siguientes:

abrir	— carga un archivo en memoria temporal
ver	— carga un fichero en memoria temporal sin permitir modificaciones
nuevo	— limpia la memoria temporal
borrar	— borra el registro activo
salvagrdr	— copia un archivo
formatear	— inicializa un cartucho

pescribir	— muestra el registro activo de un fichero indicado
añadir	— inserta un registro utilizando los valores de campo

Puntos a recordar

- La instrucción crear se termina con su correspondiente fincrear.
- insertar sigue añadiendo registros hasta que pulse **F4** (o **ESC** en la versión 1).
- alterar afecta sólo al registro activo.
- Las inserciones y correcciones no tienen lugar hasta que se pulsa **F5**.
- Recuerde cerrar los ficheros para salvar los cambios o adiciones en el cartucho.
- Si pulsa **ESC**, interrumpirá lo que estaba haciendo ARCHIVE.

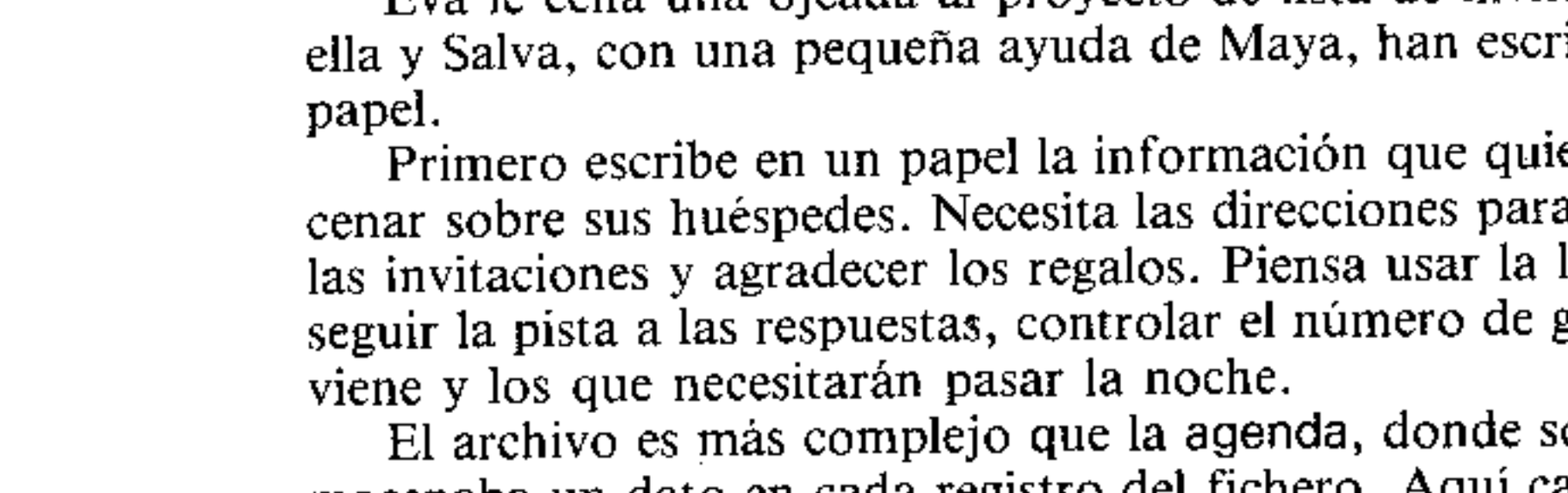
EL SEGUNDO FICHERO

Ejemplo: La lista de invitados

Eva le echa una ojeada al proyecto de lista de invitados que ella y Salva, con una pequeña ayuda de Maya, han escrito en un papel.

Primero escribe en un papel la información que quiere almacenar sobre sus huéspedes. Necesita las direcciones para mandar las invitaciones y agradecer los regalos. Piensa usar la lista para seguir la pista a las respuestas, controlar el número de gente que viene y los que necesitarán pasar la noche.

El archivo es más complejo que la agenda, donde sólo se almacenaba un dato en cada registro del fichero. Aquí cada nombre y dirección, junto con la información asociada, constituye un registro. Los elementos de información que hay en cada archivo se llaman *campos*.



Necesito Nombre, dirección, número
Nombre
Primera línea de letras
Segunda línea de letras
Tercera línea de letras
Cuarta línea de letras
¿Cuántos vienen? VIENEN$ (S/N) (texto)
¿Cuántos serán en la fiesta? CUANTOS (mnemónico)

Figura 3.1. El fichero de invitados

Archivos, registros, campos

Archivos Un archivo o fichero es una colección de registros sobre un tema dado. Se almacena en un cartucho bajo un nombre de archivo, de forma que se pueda volver a llamar. Por ejemplo, un cocinero puede crear un archivo llamado "recetas", un bibliotecario uno llamado "libros" y un médico "pacientes".

Registros Son las partes que componen un fichero, como una receta, detalles sobre un libro, o una ficha de paciente; los tres tienen un aspecto similar, aunque contienen distinta información.

Campos Son las partes componentes de un registro. Por ejemplo, los ingredientes e instrucciones de una receta, el autor y título de un libro, el nombre y dirección de un paciente.

Límites Se pueden crear tantos ficheros como se quiera, si se tienen suficientes cartuchos para almacenarlos. Un campo que contenga texto tiene hasta 255 caracteres (pero sólo se pueden cambiar

160 con alterar e insertar). Un registro puede tener hasta 255 campos (aunque alterar e insertar sólo pueden actuar sobre tantos campos como líneas hay en la pantalla).

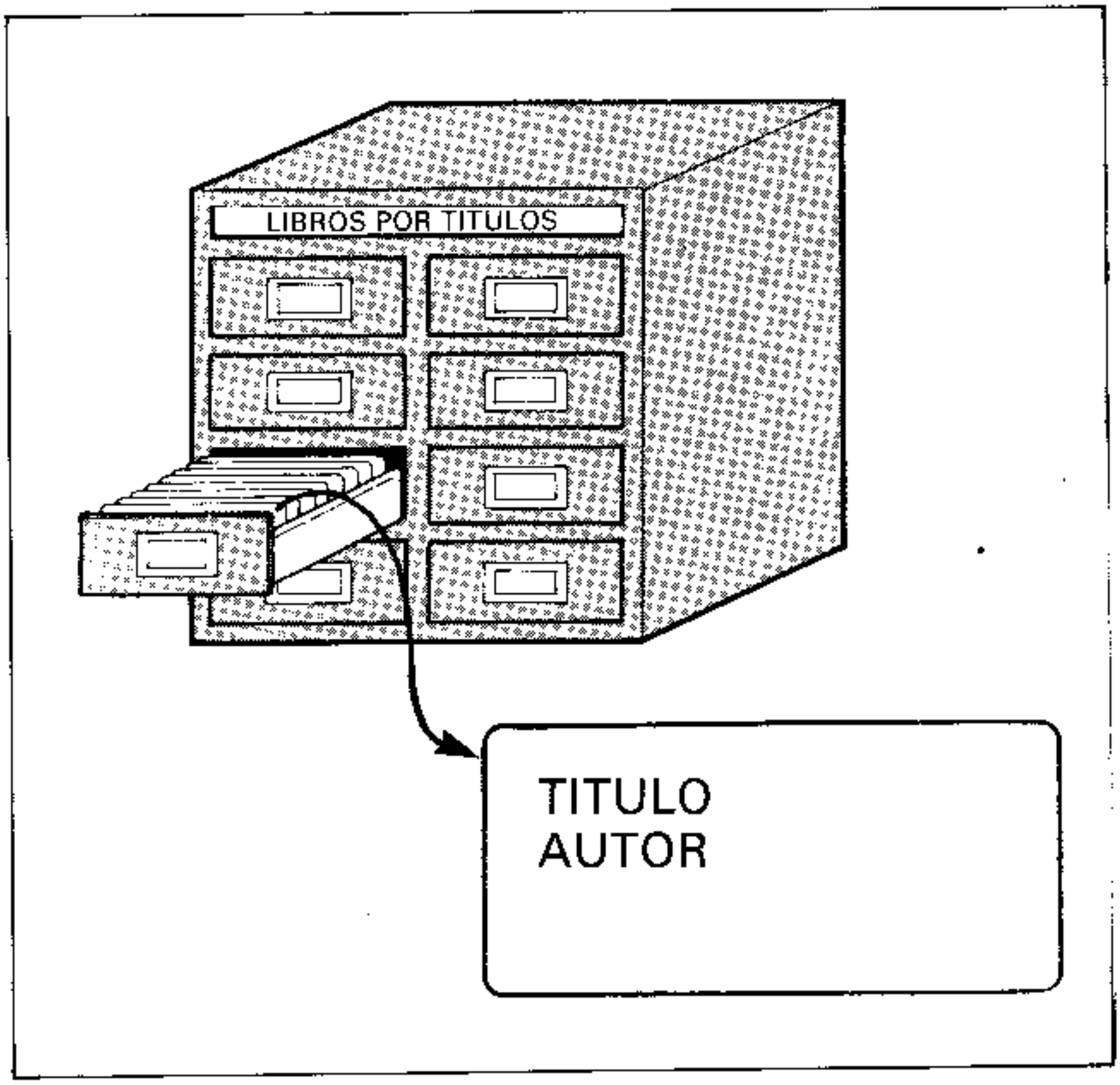


Figura 3.2. Archivos, registros y campos.

Empezando

Si no ha cargado ARCHIVE, cárguelo como se indica en el capítulo 1. Debe introducir en la unidad 2 el cartucho de datos con el fichero agenda que creó en el capítulo anterior. Si ignora el contenido del cartucho, escriba dir y pulse ↵ dos veces para ver una lista de los archivos en la unidad 2. Cambie el cartucho

si no es el correcto. Inserte otro, y repita el comando dir hasta que encuentre el fichero agenda dbf.

Si tiene algún fichero de datos abierto, ciérrelo.

LOS INVITADOS

Eva decide llamar a su archivo invitado.

Creación del archivo

crear ⏎

1. Escriba crear y pulse ⏎.
   Observe los cambios en el AREA DE CONTROL, en la parte superior de la pantalla. ARCHIVE sustituye la lista de comandos por información sobre la instrucción crear. Se le sugiere: ESCRIBA EL NOMBRE DEL ARCHIVO.

⏎

2. Pulse ⏎ otra vez.
   ARCHIVE imprime el mensaje de error nombre de fichero no válido, y reaparece el ">". No se preocupe si no acaba de comprender los mensajes de error. A menudo no son muy indicativos. Se explican con detalle en el apéndice.

> *Nota:* El comando crear debe ir seguido por un nombre de fichero válido. ARCHIVE intentó interpretar el comando completo, pero no encontró ningún nombre. Los errores hacen que se abandone el último comando introducido, y el ">" reaparece. Cuando le ocurra algo parecido, vuelva a empezar.

crear ⏎
invitado

1. Escriba crear y pulse ⏎.
2. Escriba invitado.
   La palabra aparece entre las comillas.

⏎

3. Pulse ⏎.
   El cursor se mueve a la línea siguiente. Esta vez el comando fue aceptado.

Desde este momento hasta que reaparezca el ">" ARCHIVE espera que introduzca en cada nueva línea un nombre de campo o una instrucción fincrear.

El fichero agenda tenía un solo campo: notas$: nuestro nuevo fichero tiene varios.

Si comete un error y pulsa ↵ después de escribir una línea, no es posible volver atrás a corregirlo.

Si es un error que ARCHIVE no acepte, como un nombre de campo demasiado largo, ARCHIVE imprimirá un mensaje de error y abandonará crear. Tendrá que volver a comenzar desde el principio.

Si termina crear sin querer pulsando ↵ dos veces, no se puede continuar añadiendo campos ni volver a corregir algún nombre. Escriba cerrar y comience de nuevo.

Sea muy cuidadoso al introducir nombres de campo en crear. Escribalos siempre en papel antes de empezar.

Nombres de campo y tipos

Nombres de campo

- Pueden tener hasta 13 caracteres de longitud.
- Deben comenzar por una letra.
- No pueden contener espacios.
- No pueden ser nombres de comando ni palabras claves que ARCHIVE reserva para su propio uso.

Hay dos tipos de campo:

Campos numéricos

- Sólo pueden contener números.
- Se pueden sumar, multiplicar, dividir o utilizar para cálculos matemáticos.

Campos alfanuméricos

- Pueden contener texto en cualquier combinación de letras y números. Se llaman también "cadenas", ya que consisten en caracteres encadenados.
- Los nombres de campos alfanuméricos acaban siempre por un signo dólar, "$".
- Los números almacenados en campos alfanuméricos se pueden utilizar para cálculos, pero sólo si se convierten primero en un formato numérico (mediante la función val( ), descrita en el capítulo 9).

Nota: Si escribe un nombre de campo que incluya un espacio, ARCHIVE no imprime un mensaje de error, pero el nombre de campo es sólo la primera palabra. Puesto que ésta nunca acaba en "$", el campo será tomado como numérico, lo quisiera usted o no. Por ejemplo, Comun Aut$ se convierte en Comun.

Introduzca ahora los nombres de campo para la información sobre invitados.

nombre$ ⊟ **1.** Escriba nombre$ y pulse ↵.

Errores en CREAR

Errores en la introducción de un campo:

Antes de pulsar ↵ puede usar las teclas de edición para corregir los errores.

Errores en un campo de una línea previa:

No se puede volver a la línea anterior a corregirlo. Si es importante, debe abandonar crear (pulse **ESC**) y comenzar de nuevo. Todo lo que había introducido en crear será ignorado y se perderá.

Si acaba CREAR demasiado pronto:

Si pulsa ↵ accidentalmente en una línea en blanco, y ha introducido al menos un campo, aparece fincrear, el fichero se crea y se salva al cartucho, quedando abierto. No se puede volver a comenzar y corregir los errores. Ni se puede volver a empezar de la misma manera: ARCHIVE no admite dos ficheros en memoria con el mismo nombre lógico. El programa escribirá un error (versión 1), o le pedirá que introduzca un nombre lógico para el fichero (versión 2).

Para que no le ocurra, escriba cerrar y pulse ↵. Asi cerrará el archivo que acaba de crear. Para borrar este fichero incorrecto grabado en la cinta, pulse tirar "invitado_dbf" (tirar se explica en el capítulo 5). Entonces puede volver a empezar crear, abriendo un nuevo archivo en memoria temporal con el mismo nombre.

Nota: ARCHIVE (versión 1) no comprueba en el cartucho si el fichero que va a crear ya existe, y no le avisará si borra un fichero existente con el mismo nombre.

Errores:

Si pulsa algo que ARCHIVE no entiende, aparece un

mensaje de error. Los errores hacen que el programa abandone lo que estaba haciendo, y reaparece el ">". Vuelva a comenzar. Se explica el significado de los mensajes de error en el apéndice.

Direc1$ ↵ Direc2$ ↵ Direc3$ ↵ Direc4$ ↵ Vienen$ ↵ Cuantos ↵

2. Escriba la lista de campos, pulsando ↵ tras cada palabra. Observe que Cuantos no termina en un carácter "$".

```
Direc1$
Direc2$
Direc3$
Direc4$
Vienen$
Cuantos
```

Tras escribir un par de nombres, el cursor estará en la parte inferior de la pantalla, y la línea superior del AREA DE TRABAJO desaparecerá cada vez que pulse ↵. El movimiento de las líneas de la pantalla hacia arriba se llama desplazamiento vertical *(scroll)*.

Este desplazamiento es una de las razones por las que debe guardar una copia escrita de la estructura del fichero.

Una vez escritos todos los campos:

↵

3. Pulse ↵ otra vez.
Aparece fincrear. El *microdrive* vuelve a girar.
ARCHIVE salva la estructura del archivo al cartucho.
El fichero está abierto en la memoria temporal, y se puede mostrar la estructura o insertar registros.
Reaparece el ">", preparado para otro comando.

Para mostrar la estructura

Eva sabe que si se equivoca al indicar la estructura del archivo (los nombres y tipos de los campos), puede copiar más tarde los datos que contenga el fichero mal estructurado en un nuevo fichero (capítulo 5). No es, sin embargo, un trabajo fácil, así que comprueba que ha creado la estructura correcta.

indicar ↵

1. Escriba indicar y pulse ↵.
El AREA DE VISUALIZACION muestra la estructura de registros del nuevo fichero. No se muestra ningún registro, ya que todavia no se ha introducido ninguno.

Observe el cero tras los dos puntos de Cuantos. ARCHIVE pone un cero en los campos numéricos vacíos, y nada en los alfanuméricos.

> *Nota:* El cero se representa en el QL mediante un "0" cruzado con un "/". Se hace así para distinguirlo con más facilidad de la letra "O".

Listar el directorio

Para asegurarse de que su fichero se ha grabado realmente en el cartucho, Eva usa la instrucción dir.

dir ↵ ↵

1. Escriba dir y pulse dos veces ↵.
   El AREA DE VISUALIZACION se borra, y el directorio del *microdrive* 2 muestra dos archivos: invitado_dbf y agenda_dbf

Cambiar de fichero activo: nombres lógicos

Pensando en su lista de invitados, Eva se recuerda de que tiene que anotar algunas otras cosas antes de escribir los nombres y direcciones. Quiere hacer una lista de regalos, y comienza a pensar en arreglar las habitaciones del hotel para los invitados que tengan que pasar la noche. Antes de escribir las anotaciones tiene que abrir el fichero agenda.

Nombres lógicos

Los nombres lógicos son necesarios para distinguir los diferentes archivos cuando hay más de uno abierto al mismo tiempo. Las instrucciones que actúan sobre ficheros abiertos, como cerrar y próximo, afectan sólo a un fichero. Se pueden utilizar con un nombre lógico para especificar el fichero que se verá afectado; por ejemplo:

cerrar "maestro"	cierra el fichero que tenga el nombre lógico maestro.
próximo "maestro"	pasa un registro en el fichero con nombre lógico "maestro".

Un nombre lógico es un nombre temporal que dura sólo mientras el fichero está abierto. No afecta de ninguna manera al nombre de archivo "fisicamente" grabado en el cartucho. No tiene por qué ser parecido, ni necesita ser el mismo la siguiente vez que abra el archivo. El único requerimiento es que en ningún momento haya dos nombres lógicos iguales.

Los nombres lógicos pueden ser de sólo una letra, lo que ahorra tiempo al escribirlos. El nombre de archivo en la cinta es largo normalmente, para recordarnos el contenido del archivo.

ARCHIVE automáticamente proporciona el nombre lógico maestro al primer fichero que se abre si no se especifica otro, o si no hay otro fichero abierto con ese nombre. Evite siempre usar maestro como nombre lógico, y abra siempre los archivos con nombres de su elección.

Los nombres lógicos pueden tener hasta 13 caracteres; comienzan con una letra y no deben incluir espacios.

Los nombres lógicos son una abreviatura útil para especificar a qué fichero nos referimos. Su uso se verá más claramente según avance en el libro. Apunte en un papel los nombres lógicos que esté utilizando.

abrir "agenda" ↵

1. Escriba abrir "agenda" y pulse ↵.
En el AREA DE TRABAJO, el programa añade lógico "" a la línea, y espera a que usted escriba un nombre lógico (versión 2).
El motivo es algo complicado: cuando se crea o abre un fichero se le debe dar un nombre temporal por el que le conoce ARCHIVE. Cada archivo abierto tiene un nombre temporal distinto. Este *nombre temporal* no tiene ninguna relación con el nombre que tiene el *archivo* en el cartucho.
El nombre temporal se llama *nombre lógico*, aunque no tenga nada que ver con la lógica. Es un término informatico para distinguir un fichero de todos los demás que, estando abiertos, comparten la memoria temporal.
Si no especifica un nombre lógico cuando crea o abre un archivo, ARCHIVE proporciona uno; pero ARCHIVE sólo tiene uno en su repertorio: "maestro".
Cuando creó invitado no indicó ningún nombre lógico, y ARCHIVE le dio el nombre maestro. Como dos ficheros abiertos no pueden tener el mismo nombre lógico, ARCHIVE le pide un nombre distinto para agenda. En la versión 1 habría aparecido el error nombre duplicado.
Para abrir correctamente el fichero:

a ⊡

2. Escriba a entre las comillas que le ha proporcionado ARCHIVE y pulse ↵.
Reaparece el ">", indicando que el comando tuvo éxito. El archivo agenda está abierto. El AREA DE VISUALIZACION, sin embargo, no ha cambiado.

USO DE FICHEROS MULTIPLES

Presentación del archivo activo

Eva tiene ahora dos ficheros abiertos: invitado, con el nombre lógico maestro, y agenda. El fichero agenda es el fichero activo (ya que lo acaba de abrir). Para verlo:

indicar ⊡

1. Escriba indicar y pulse ↵.
El AREA DE VISUALIZACION se borra, desapareciendo el directorio del *microdrive*, y aparece el primer registro del fichero agenda. Observe cómo el nombre lógico es ahora a.
El primer registro del fichero agenda es el registro activo. El fichero activo, desde que lo abrimos, es agenda. El comando indicar siempre muestra el fichero activo (a menos que se le indique otra cosa).
Usted no ha cerrado el fichero invitado, y por tanto sigue abierto.

Adición de notas al fichero **agenda**

Eva añade dos notas más al fichero agenda usando el comando insertar.

insertar ⊡ Buscar un Hotel para los invitados F5 Hacer una Lista de Boda F5 F4

1. Escriba las dos notas siguientes. No olvide pulsar **F5** o ↵ (versión 2) tras cada una. Si olvidó el uso de insertar, vuelva al capítulo anterior, o guíese por las indicaciones de ARCHIVE y la tecla de ayuda, **F1**.

Buscar un Hotel para los invitados
Hacer una Lista de Boda

Recuerde salvar la segunda nota, pulsando **F5** antes de abandonar insertar.

Abandone insertar mediante **F4** (versión 2) o **ESC** (ambas versiones).

Borrado de registros

Eva borra la anotación sobre la agenda de Salva, ya que ahora es redundante.

hallar "Sal" ↵

1. Escriba hallar "Sal" y pulse ↵.
El registro buscado aparece en el AREA DE VISUALIZACION.

borrar ↵

2. Escriba borrar y pulse ↵.
El registro desaparece, y el AREA DE VISUALIZACION muestra el próximo registro del fichero.
Sea cuidadoso cuando utilice el comando borrar. Elimina el registro activo del fichero activo, sea o no visible en la pantalla. Recuerde que la pantalla muestra el último registro indicado, que puede no ser el activo (por ejemplo, si después escribió un comando usar). No se puede recuperar un registro borrado de la memoria temporal.

Nota: Si pierde información importante es posible recuperarla si hizo copias de seguridad del archivo. El uso de las copias de seguridad se explica al final de este capítulo.

Apertura de ficheros con VER

El comando ver también abre ficheros de datos. Se usa de la misma manera que abrir. La única diferencia es que los ficheros abiertos con ver no se pueden modificar, es decir, no se puede insertar, borrar o alterar registros.

ARCHIVE no salva los archivos abiertos con ver al cerrarlos, ya que las copias en memoria y cinta deben ser idénticas. Simplemente elimina la copia en memoria.

Utilice ver para evitar modificaciones accidentales a los ficheros; también ahorra tiempo al cerrar los ficheros, ya que ver no desperdicia tiempo restaurando los ficheros en el cartucho.

El fichero agenda contiene ahora las siguientes anotaciones:

```
notas$
Ver al sacerdote
Pedirle a Felicidad que sea la madrina
Decirle a Juana que haga los trajes
Decirle al Sr Hort que se encargue de las flores
Que no se olvide hacer los ojales
Zapatos blancos
Comprar el traje
¿ Ha comprado Salva un traje de sport ?
Contratar a Pérez Fotógrafos
Coche
Llamar a Sara ¿ Quién les organizó el banquete ?
Capacidad de la sala Azul del Hotel Almirante
Encargar la tarta
Buscar un Hotel para los invitados
Hacer una lista de bodas
```

Adición de registros al fichero **invitado**

Eva está ya lista para introducir algunos nombres y direcciones, que se conservarán en el fichero invitado, al que le ha dado el nombre lógico maestro. En primer lugar debe convertirlo en fichero activo.

USAR para cambiar el fichero activo

Para convertir invitado en fichero activo:

usar "maestro" ↵

1. Escriba usar "maestro" y pulse ↵.
   (El fichero invitado se abrió automáticamente con el nombre lógico maestro.)

Nota: usar es una excepción a la regla de que ARCHIVE proporciona siempre las comillas cuando son necesarias. Póngalas usted mismo.

El comando usar seguido de un nombre lógico convierte este fichero en el fichero activo. Si no sabe cuál es el fichero activo, escriba un comando usar. No se le hace daño al programa si se le pide que use el fichero que ya está usando.

Registros y ficheros activos

La mayor parte de los comandos de ARCHIVE afectan al registro activo del fichero activo, si no se especifica otra cosa.

ARCHIVE trabaja siempre con una unidad de información cada vez. Los comandos que afectan ficheros o registros afectan sólo un fichero o un registro. Los comandos que actúan sobre varios registros lo hacen sucesivamente, uno cada vez. No importa el número de registros que tenga un fichero, o cuántos ficheros haya abierto, sólo un registro de un fichero será el registro activo.

Imagínese que está usando varios libros de consulta en su mesa. Cada libro está abierto por una página, pero usted mira sólo una página en cada momento. El libro que usted mira en cada momento es el libro activo.

ARCHIVE lee los registros de los ficheros de la misma manera. El registro activo de cada fichero es el último registro al que se ha movido el programa. Cuando cambia de registro el fichero activo no afecta a los registros activos de otros ficheros abiertos. El fichero activo es el último abierto, creado o usado por los comandos abrir, ver, crear o usar.

Cuando se abre un fichero, su primer registro se convierte en el registro activo. Si se cierra un archivo, se convierte en fichero activo el último que se usó con anterioridad.

Para alterar un *registro* que no sea el activo, debe primero convertirlo en activo moviéndose a él. *No* se puede hacer que un comando afecte a un registro que no sea el registro activo.

Para actuar sobre un *fichero* que no sea el activo, debemos convertirlo en el fichero activo. Sin embargo, algunos comandos permiten especificar esta operación como parte de la instrucción. Para indicar que un comando afecta a un fichero que no es el activo, debe utilizar el nombre lógico. El fichero indicado se convierte entonces en el activo.

Ejemplo: Uso de un nombre lógico en un comando

Se puede conseguir el mismo efecto de dos maneras: el fichero activo antes y después de los ejemplos siguientes es fich1, y el resultado es que se borra el registro activo de fich2.

Ejemplo 1

El fichero fich1 es el activo.

```
>usar "fich2"        convierte en activo a fich2
>borrar              borra el registro activo
>usar "fich1"        vuelve a hacer activo fich1
```

Ejemplo 2

El fichero fich1 es el activo.

>borrar "fich2"	borra el registro activo de fich2. Este queda como fichero activo
>usar "fich1"	vuelve a hacer activo fich1

insertar ↵

1. Escriba insertar y pulse ↵ para comenzar a insertar nombres en el fichero invitado.

El AREA DE CONTROL le indica las opciones válidas del comando insertar. La tecla **TAB** (el tabulador) mueve el cursor de campo en campo del registro; las teclas de función sirven para grabar los registros o abandonar el comando.

Observe que ↵ tiene el mismo efecto que **TAB.** El comando insertar sigue ofreciéndole registros vacios para la inserción hasta que lo abandone con **F4** o **ESC.**

Juan y Josefa Martínez Rebollo ↵

2. Escriba Juan y Josefa Martínez Rebollo y pulse ↵. El cursor pasa a la línea siguiente.

Maja **3.** Escriba Maja.
Eva se detiene, y piensa que no es correcto tratar con tanta formalidad a sus amigos. Decide cambiar "Josefa" por "Pepa".

Edición de un campo

⇧ TAB **1.** Pulse **TAB** mientras mantiene pulsada la tecla de mayúsculas (⇧). El cursor se mueve al comienzo del campo nombre$.

⇧ ⇨ **2.** Manteniendo la tecla ⇧ pulsada, use la flecha derecha. El cursor salta al comienzo de la siguiente palabra.

⇧ ⇨ **3.** Hágalo otra vez para llegar a la "J" de "Josefa".

CTRL ⇨ *(5 veces)* **4.** Borre Josef mediante CTRL y la flecha derecha.

Pep ↵ **5.** Escriba Pep y pulse ↵.
El cursor vuelve al comienzo de la siguiente línea.

⇩ **6.** El cursor va al final de la línea.

da del Viento, 25 ↵ **7.** Acabe la introducción de la dirección y pulse ↵.

28009 Madrid ↵ MADRID ↵ ↵ s ↵ **8.** Complete la ficha como se indica, pulsando ↵ tras cada línea. Pulse ↵ sin escribir para dejar una línea vacía:

```
direc2$      :28009 Madrid
direc3$      :MADRID
direc4$      :
vienen$      :s
```

La idea de Eva es usar el campo vienen$ con la letra s si los invitados han confirmado la asistencia, y la letra n si no pueden venir. Deja la línea en blanco si no ha recibido la respuesta, o no saben si podrán venir. Así resulta más fácil seleccionar listas para impresión.
ARCHIVE nunca comprueba la información de un campo alfanumérico. El programa no plantea ningún problema si existen nombres duplicados o mal escritos, o si aparece una dirección en el campo nombre$. Se puede hacer que ARCHIVE compruebe lo que introducimos, pero sólo si escribimos el programa nosotros mismos.
El cursor está en el campo cuantos.

dos ↵ **9.** Escriba dos y pulse ↵.
Si tiene la versión castellana de ARCHIVE, éste no le dejará introducir letras en un campo numérico. Si tiene versiones más antiguas, aparece el mensaje de error variable no reconocido.

Introducción de texto en campos numéricos

ARCHIVE no permite introducir caracteres en campos numéricos. Todos los campos cuyo nombre no acabe en $ son considerados numéricos por el programa.

- En la versión 2, permite escribir texto, pero el cursor se queda en la línea cuando pulsamos ↵. Ni siquiera pulsando **ESC** nos permite movernos al siguiente campo. La única manera de salir del problema es borrar el texto o escribir un número: en este ejemplo, corrija la línea hasta que quede un 2 y pulse ↵.
- La versión 1 permite la introducción de texto, pero se detiene con un mensaje de error si se pulsa ↵.
Si aparece un mensaje de error, ARCHIVE abandona insertar, y se pierde el registro que estaba escribiendo.

Recuperación de errores en INSERTAR (versión 1)

Afortunadamente (si tiene versión 1), no tiene que volver a escribir el registro entero. Existe un pequeño truco para recuperar los registros que pierda mientras inserta.

añadir ↵

1. Escriba añadir y pulse ↵.
Igual que ocurre en insertar cuando pulsa **F5,** este comando utiliza la última información de los nombres de campo para crear un registro. El comando añadir se explica más detalladamente en el capítulo 5. El nuevo registro se graba en cinta.

alterar ↵

2. Escriba alterar y pulse ↵.
Corrija ahora la información del campo cuantos:

↵ (o TAB) *(6 veces)*

3. Pulse ↵ o **TAB** para saltar los campos correctos.

2 ↵

4. Introduzca el número 2 en el campo cuantos y pulse ↵.

Compruebe si es correcta la información que acaba de introducir.

Si hay errores, utilice ⇧ y **TAB** para volver atrás por los campos; corrija el error mediante los cursores o escribiendo el campo de nuevo.

En la versión 1 se puede recorrer el registro en las dos direcciones mediante ↵ o ⇧/**TAB.** Si pulsa ↵ cuando está en el último campo, el cursor vuelve al primero. Si ahora pulsa ⇧/**TAB,** el cursor vuelve al último campo.

El registro no se añade al fichero automáticamente cuando llega al último campo. No se insertará hasta que usted se lo indique a ARCHIVE.

En la versión 2, en cambio, el registro se salva si pulsa ↵ en el último registro. La única manera de volver al primer campo es usando ⇧/**TAB** a la vez.

Grabación de un registro

La manera de salvar una inserción o una alteración es la misma en esencia:

F5

1. Cuando acabe de modificar los campos, pulse **F5** (o, en la versión 2, pulse ↵ en el último campo).
Así, ARCHIVE graba la información como parte del fichero en la memoria del QL, pero no salva necesariamente el registro en el cartucho. Esto sucede cuando la memoria temporal se llena, o bien al cerrar el fichero. Observe que **F5** se puede pulsar en cualquier posición de cualquier campo de un registro.

Nota: Pulse **F5** sólo cuando ha escrito todos los campos que necesite en el registro. Si pulse **F5** antes, el registro se grabará incompleto, y la única manera de arreglarlo es salir de insertar mediante **F4** (o **ESC**) y usar el comando alterar. El registro activo es el último que se ha insertado, así que no es necesario el comando anterior.

Para abandonar un registro

F4 (o ESC)

1. Si decide que el registro no es adecuado antes de salvarlo, basta pulsar **F4** (o **ESC** en la versión 1) para eliminar el registro incorrecto como si nunca hubiera existido, aunque se siga viendo en la pantalla.

Añada más invitados

Escriba el siguiente registro de la lista de Eva:

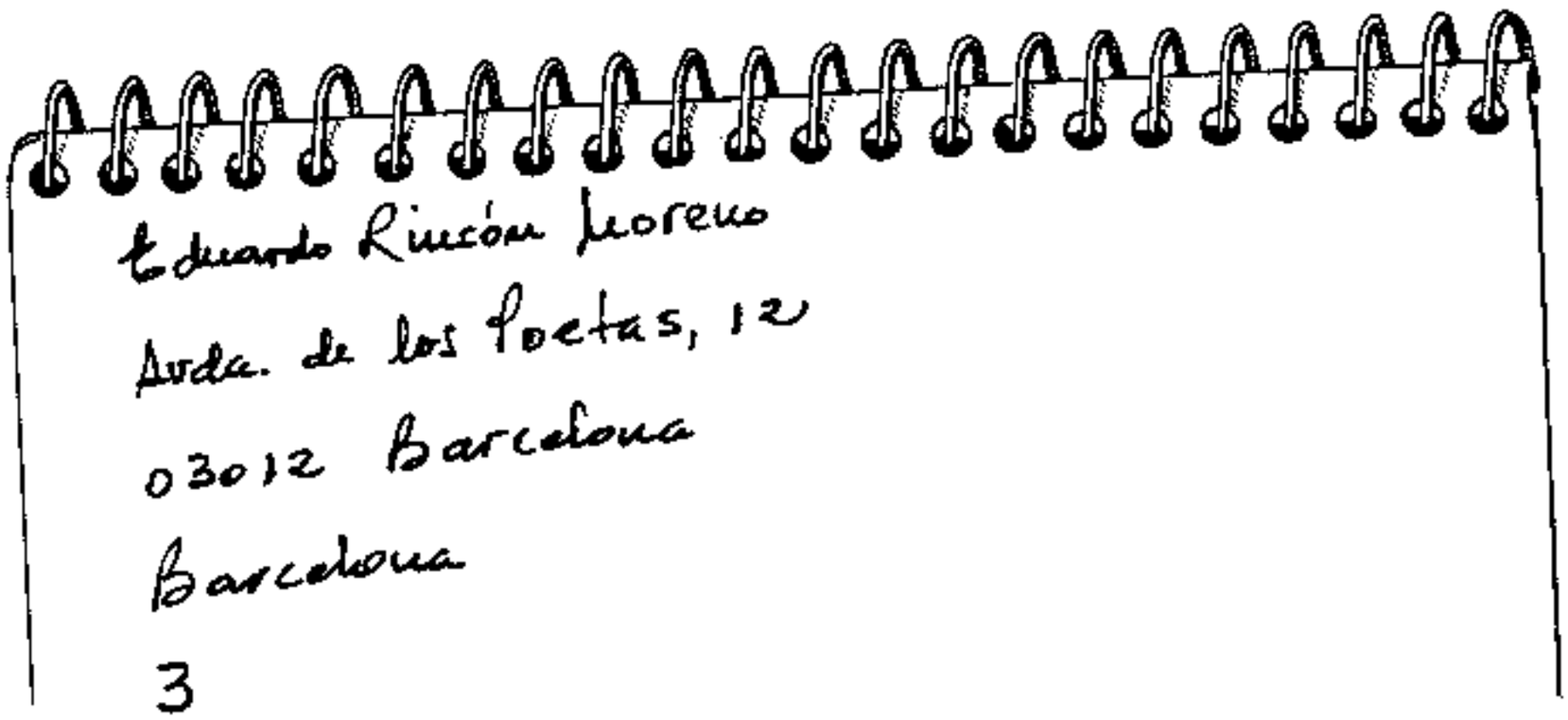
insertar ↵

1. Escriba insertar y pulse ↵, a menos que ya esté insertando.

Luisa Sala y Carmen García Carrión ↵ Gran Vía, 34 ↵ Orihuela ↵ Alicante ↵ ↵ s ↵ 2 ↵

2. Introduzca la ficha siguiente de la lista de Eva; recuerde que debe pulsar ↵ tras cada campo y, si tiene versión 1, **F5** para salvar el registro.
3. Inserte los registros restantes de la lista de Eva y Maya. Deje vacíos los campos sobre los que no haya datos.

Nota: Aunque Eva puso "s" en el campo vienen$ de las dos primeras fichas, ya que han confirmado su asistencia, deje en blanco el campo para los invitados que aún no han contestado. Sea cuidadoso e introduzca el número de invitados en la posición correcta.



Teatro Principal
Pza. Mayor, 3
Hellín
Albacete
3

Maibel Gutiérrez
Paseo de las Acacias, 34
Villanueva de Segre
Tarragona
1

Felipe y Luisa González
Jardines Dorados, 12
c. de las Flores
Bangkok
Tailandia
2

Antonio Pérez de las Heras y Sra.
Teniente Coronel
Prat d'en Joaquim
03102 Roses
Gerona
2

Arturo y Teresa López Audión
Magnolias, 110
05002
Bilbao
7

Srta. Felisa de los Monteros
Vista Cielo
Montes
Villanueva de Segre
Tarragona
1

Figura 3.3. Lista de invitados

F4

4. Cuando pulse **F5** o ↵ en el último campo del último registro que quiera insertar, pulse **F4** (**ESC** en la versión 1) sobre un registro en blanco para abandonar insertar.

> *Nota:* Si pulsa **F5** mientras tiene en pantalla un registro vacío, el registro se inserta vacío en la base de datos. Es difícil darse cuenta, ya que la pantalla no cambia, al presentar un nuevo registro vacío en la pantalla. Si encuentra registros vacíos en sus ficheros, puede llenarlos con nuevos registros mediante alterar o bien eliminarlos del fichero mediante borrar.

Para ver el último registro insertado

Para ver el último registro insertado:

pescribir ↵

1. Escriba pescribir y pulse ↵.
   El comando pescribir (abreviatura de pantalla y escribir) muestra el registro activo. No tiene efecto si no se ha usado el comando indicar antes, o si la presentación en pantalla no es la del fichero activo.
   Si piensa que el registro que se ve en pantalla es el activo, use pescribir para estar seguro.

INSERTAR y ALTERAR

Recuerde que insertar añade registros a un fichero, y no información a un registro existente. Sólo añade registros; nunca puede cambiar los existentes. Para cambiar un registro existente debe usar alterar (como se explicó en el capítulo anterior). Alterar funciona exactamente de la misma manera que insertar, pero sólo actúa sobre un registro cada vez: el registro activo.

> *Nota:* alterar e insertar no pueden añadir o modificar información a más campos de los que caben en la pantalla. En otras palabras: si la pantalla tiene quince líneas, no se puede insertar o modificar los campos decimosexto, decimoséptimo, etc. Se puede hacer quitando el AREA DE CONTROL, dibujando una pantalla de usuario o utilizando comandos indirectos. Evite este problema manteniendo el número de campos razonablemente pequeño.

El fichero invitado debe contener ahora los siguientes nombres y direcciones:

```
nombre$                                          direc1$                                  direc2$
            direc3$                    direc4$   vienen$ cuantos
Juan y Pepa Martinez Rebollo                     Majada del Viento, 25                    28029 Ma
drid        MADRID                                  s                 2
Luisa Sala y Carmen Garcia Carrión               Gran Via, 34                             Orihuela
            ALICANTE                                s                 2
Eduardo Rincón Moreno                            Avda. de los Poetas, 12                  03012 Ba
rcelona     BARCELONA                                                 3
Teatro Principal                                 Pza. Mayor, 3                            Hellin
            ALBACETE                                                  3
Lorenzo Sánchez López, critico teatral           Revista del Escenario                    Avda. Pr
incipal, 23 Villanueva de Segre TARRAGONA                             1
Juan Echarri                                     Cia de Danza Moderna "Break out"         Melancól
icos, 35    28013 Madrid        MADRID                                1
Milena de Pablo Martinez                         Diaz de Carreras, s.n.                   Tarancón
            CUENCA                                                    1
Maribel Gutiérrez                                Paseo de las Acacias, 34                 Villanue
va de Segre TARRAGONA                                                 1
Felipe y Luisa González                          Jardines Dorados, 12                     C. de la
s Flores    Bangkok             TAILANDIA                             2
Antonio Pérez de las Heras y Sra.                Teniente Coronel                         Prat d'e
n Joaquim   03102 Roses         GIRONA                                2
Arturo y Teresa López Andión                     Magnolias, 110                           05002
            BILBAO                                                    7
Srta. Felisa de los Monteros                     Vista Cielo                              Urb. Los
 Montes     Villanueva de Segre TARRAGONA                             1
```

Cerrar varios ficheros

Incluso el trabajo más apasionante puede llegar a cansar. Tras escribir varios nombres y direcciones, Eva salva la tarea realizada cerrando el fichero.

cerrar ↵

1. Escriba cerrar y pulse ↵.
El *microdrive* derecho gira, y los últimos registros se graban en la cinta. La copia del archivo en memoria temporal desaparece.
El comando cerrar sin un nombre de fichero cierra el fichero activo, en este caso maestro. Si hay más de un fichero abierto, queda como fichero activo el último que se utilizó. El fichero agenda, a, es el fichero activo.

Nota: El AREA DE VISUALIZACION muestra todavía el último registro del fichero invitado. La pantalla no se altera al cerrar un fichero. Si ve un fichero en pantalla no quiere decir necesariamente que está en uso, ni siquiera abierto. La pantalla muestra todavía los últimos valores de las variables de campo del fichero invitado. Si volviera a abrir el fichero, la pantalla cambiaría para enseñar el primer registro del fichero.

Para cerrar también el archivo agenda:

cerrar ↵

2. Escriba cerrar y pulse ↵.
El *microdrive* gira de nuevo, grabando la última versión del fichero agenda en el cartucho. El archivo está siendo sustituido por la nueva versión ampliada.
El comando cerrar salva en la cinta los ficheros abiertos mediante el comando abrir, ya que podrían haber sido modificados. Si se abre un fichero con mirar, no se permite hacer cambios, y por ello la copia del fichero en la cinta no se actualiza.

Para cerrar varios ficheros

El comando cerrar actúa sobre el fichero activo si no se indica ningún nombre. Para cerrar un fichero que no sea el activo, hay que escribir el comando con un nombre lógico de archivo, por ejemplo, cerrar "maestro".

Si hay más de un fichero abierto simultáneamente, los que no sean activos sólo se pueden cerrar indicando su nombre.

Si no recuerda los nombres de los ficheros abiertos, basta con cerrar varias veces hasta que ARCHIVE escriba el mensaje no hay fichero abierto. Así sabrá que todos los ficheros están cerrados.

Los comandos abandonar y nuevo cierran todos los archivos abiertos. Pero ambos comandos borran toda la información que hubiera en ARCHIVE, y el primero, además, sale de ARCHIVE.

Se evitan problemas con los ficheros abriendo tantos como sea posible mediante ver, y cerrando todos aquellos que ya no son necesarios. Además, así se libera memoria que puede ser necesaria para otras tareas, ya que cada fichero abierto necesita algo de la memoria temporal.

Si se cierran y abren los ficheros con frecuencia la ejecución puede ser lenta, ya que esta operación consume tiempo (grabando las modificaciones en la cinta) y la última sugerencia puede no ser práctica.

COPIAS DE SEGURIDAD DE FICHEROS

Es muy importante hacer copias de seguridad de sus ficheros. La cinta de los cartuchos de *microdrive* es muy vulnerable, y se puede perder muchas horas de trabajo por poner el dedo en la cinta o entrarle una partícula de polvo. Además, se puede borrar un fichero por error, o formatear un cartucho que contiene información importante. Otras posibles causas de fallo se listan en el apéndice.

La solución a este problema es hacer copias de sus ficheros con la mayor frecuencia posible.

ARCHIVE tiene el comando salvagrdr para copiar ficheros. Sólo actúa sobre ficheros en cinta. No copiará ficheros abiertos.

Copia el mismo cartucho mediante SALVAGRDR

salvagrdr "invitado_dbf" como "invit1_dbf" ↵

1. Escriba la línea superior y pulse ↵.
   Observe que los dos archivos tienen la misma extensión, _dbf.
   El cartucho de datos comienza a girar. ARCHIVE está haciendo un duplicado del fichero invitado en el mismo cartucho donde está el original.

dir ↵

2. Escriba dir y pulse ↵.

↵

3. Pulse ↵ otra vez para decirle a ARCHIVE que quiere el directorio del cartucho de datos (mdv2). ARCHIVE supone mdv2 si no se le indica otra cosa (por ejemplo, escribiendo dir "mdv1 ").

Debe haber tres ficheros en el cartucho: agenda dbf, invitado_dbf e invit1 dbf.

salvagrdr "invitado_dbf" como "invitado bak" ↵

4. Escriba la línea anterior y pulse ↵.

dir ↵ ↵

5. Pida un nuevo directorio.
   Observe cómo aparece el fichero invitado_bak. Ahora hay tres copias de nuestro fichero invitado en el cartucho del *microdrive* 2: invitado_dbf, invit1_dbf e invitado_bak.

Nota: Sea cuidadoso y no le pida a ARCHIVE una copia de un fichero con el nombre de un archivo existente. ARCHIVE versión 1 lo borrará sin advertirle del problema. Como protección, ninguna versión de ARCHIVE copiará un archivo con el

mismo nombre en la misma cinta (salvagrdr "agenda_dbf" como "agenda dbf" no funciona). En la versión 2 aparece un mensaje de error cada vez que intente copiar un fichero con un nombre que ya existe. Para hacerlo hay que tirar el fichero viejo antes de "salvagrdr" (véase el capítulo 5).

Si salvaguarda un fichero en la misma cinta, se evitará problemas de borrados accidentales, y protección contra daño parcial de la cinta. Para más seguridad, haga siempre las copias de seguridad en un cartucho distinto.

Formateo de un cartucho

Sólo se puede usar un cartucho para almacenar información si éste ha sido formateado previamente. Al hacerlo, el QL marca en la cinta las zonas utilizables; así sabe después dónde buscar la información.

- Atención: al formatear un cartucho se borra toda la información que contiene. Haga siempre un directorio del cartucho antes de formatear.

Si no tiene un cartucho formateado, utilice el comando formatear para prepararlo. Esta instrucción es muy parecida al comando del SUPERBASIC que se vio en el primer capitulo.

Como precaución, retire el cartucho de ARCHIVE y el de datos antes de comenzar.

1. Introduzca un cartucho vacio en la unidad 1.

formatear "mdv1_ arcdatseg1" ↵

2. Escriba formatear "mdv1_arcdatseg1" y pulse ↵.
Así formatea el cartucho de la unidad izquierda, dándole el nombre (nombre de volumen) de arcdatseg1, abreviatura de "ARCHIVE datos seguridad 1".
Naturalmente, puede darle el nombre que desee, siempre que no sobrepase los diez caracteres (excluyendo el nombre de la unidad). Evite el carácter de subrayado "_", ya que tiene un significado especial cuando se utiliza el QL en una red local.
Una vez formateado un cartucho, no es necesario repetir la operación, a menos que desee borrar los viejos ficheros y volver a utilizar la cinta. También puede formatear para rescatar cintas "corrompidas", que dan error en las operaciones de lectura/escritura, con errores como "medio incorrecto" o "E/S incompleta". La cinta es reutilizable si vuelve a formatearla, ya que el QL reorganiza los sectores evitando la zona estropeada.

- Recuerde: cuando formatea un cartucho borra toda la información que hubiera en éste.

No olvide etiquetar los cartuchos, usando una de las etiquetas autoadhesivas.

Copias a otro cartucho

El cartucho recién formateado debe estar en la unidad 1.

1. Devuelva el cartucho de datos a su posición usual en la unidad 2.

salvagrdr "invitado _dbf" como "mdv1_ invitado_dbf" ↵

2. Escriba la línea anterior y pulse ↵. Así copia el fichero de la unidad 2 a la 1, manteniendo el mismo nombre. Debe hacer una copia de seguridad exacta del cartucho de datos, de forma que si le sucede algo a la cinta original se pueda recuperar toda la información de la copia.

dir ↵ mdv1_ ↵

3. Escriba dir y pulse ↵, y después escriba mdv1_ y vuelva a pulsar ↵, para comprobar que el fichero se ha copiado correctamente.
   Observe el "_" al final del nombre de la unidad.

Compruebe siempre que tiene la cinta exacta en cada cartucho al hacer una copia de seguridad. Si deja la cinta de ARCHIVE en la unidad 1 y no ha eliminado la lengüeta de protección de escritura, los ficheros se salvarán en él hasta que no quede espacio en la cinta, cosa que sucede rápidamente. Si ha eliminado la lengüeta, los ficheros no se copiarán, y el comando dará error, con resultados impredecibles.

salvagrdr "agenda _dbf" como "mdv1_ agenda_dbf"

4. Haga una copia de seguridad del fichero agenda de la misma manera. Recuerde indicar el nombre completo con extensión, ya que este comando no hace ninguna suposición sobre los tipos de ficheros que copia.

Copias rotatorias

Para trabajar con seguridad, debe tener al menos dos cintas para copias de seguridad, etiquetadas 1 y 2, ya que es posible que una operación de copia dé errores, perdiéndose el fichero original y la copia en la operación.

Haga su primera copia en la cinta 1, y la siguiente vez utilice la cinta 2. Vuelva a usarlas en ese orden. Así, si ocurre lo peor, tendrá todavía una copia no más antigua que la última operación de copia.

Anote en una lista la fecha de cada copiado y el número de la cinta que usó. El último número en la lista es la cinta que NO hay que usar para las siguientes copias de sus archivos.

Si quiere usar las posibilidades de AYUDA de ARCHIVE no olvide volver a introducir el cartucho de programa en el *microdrive* 1 antes de seguir con su trabajo.

Fechado de los cartuchos

El sistema de volcados rotatorios es efectivo, pero depende de saber con precisión cuál fue la última cinta que se usó para un volcado.

Es fácil perder los papeles donde se apuntan las fechas. Si le ocurre, tendrá problemas para saber qué cartucho de copia debe usar.

La solución es incluir la fecha en el mismo cartucho.

Use el comando salvar para crear un fichero de programa con un nombre que le sirva como recordatorio de la fecha. No debe tener ningún procedimiento en memoria cuando haga esta operación (los programas, procedimientos y el comando salvar se presentan en el próximo capítulo).

nuevo ↵

1. Escriba nuevo y pulse ↵ para limpiar todos los procedimientos de memoria.

salvar ↵ "nov10_eti" ↵

2. Introduzca salvar "nov10_eti" y pulse ↵.
La extensión _eti identifica el fichero como una etiqueta.

F5 mdv1_ ↵

3. Repita el comando mediante **F5**, pero añada mdv1_ al nombre del fichero. Así guarda la fecha en la cinta de seguridad.

La próxima vez que haga una copia de seguridad observe si el cartucho que va a usar tiene un fichero _eti con la misma fecha que el de trabajo. En ese caso es muy probable que esté usando el cartucho incorrecto.

Ponga fecha a los cartuchos cada vez que haga copias de seguridad, y borre las etiquetas de fecha antiguas mediante tirar.

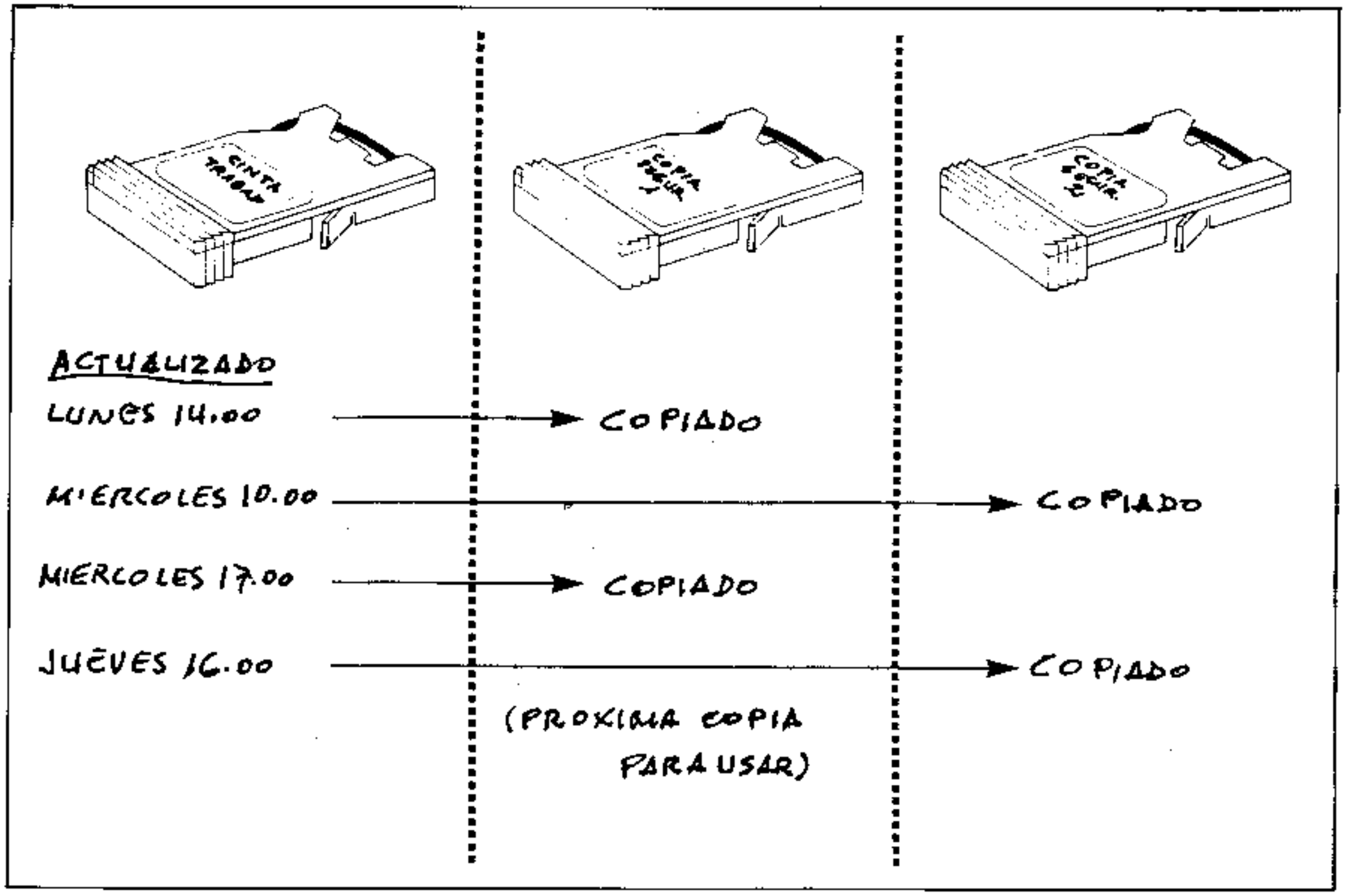


Figura 3.4. Rotación de copias

RESUMEN

Eva ha creado su segundo fichero, invitado, con más de un campo. Abre su primer archivo, agenda, usando el comando abrir. Tiene que darle un nombre lógico a agenda, ya que tiene otro fichero abierto. Inserta algunos registros y borra otro redundante usando el comando borrar. Esta instrucción borra el registro activo del fichero activo, a menos que se especifique un nombre lógico. Después Eva convierte a invitado en fichero activo, utilizando el comando usar con un nombre lógico. El comando usar convierte un fichero abierto en memoria temporal en el fichero activo.

Eva inserta y corrige información en sus dos ficheros con los comandos insertar y alterar. Cierra después los ficheros mediante el comando cerrar, y salvaguarda copias de ambos en la misma cinta, y en un cartucho de seguridad, que formatea antes.

Trucos útiles

- Apunte un plan de trabajo antes de comenzar a usar ARCHIVE. Así pondrá en claro sus ideas y tendrá una referencia útil de lo que piensa hacer.
- Use dir para comprobar que no va a borrar ningún fichero al crear o salvagrdr. Si no lo hace, ARCHIVE versión 1 los borrará sin advertirle.
- Haga copias de seguridad de sus ficheros en cartucho aparte con tanta frecuencia como le sea posible.
- Grabe su trabajo con regularidad cerrando el fichero de datos que esté corrigiendo o ampliando.
- Haga copias de seguridad rotatorias; así perderá poca información aunque se destruya una copia de seguridad.
- Use la flecha superior antes de insertar un carácter al comienzo de un campo en alterar. Así ARCHIVE no borra la línea.
- En el comando insertar, no pulse nunca **F5** en un registro vacío. En ese momento no nota nada, pero el registro vacío se salva en la cinta.
- Salve ficheros vacíos con la fecha de la copia de seguridad como nombre para recordar el día que hizo la copia.

4

Funciones y procedimientos

La potencia real de ARCHIVE aparece al poderle añadir los comandos que desee. Así se puede hacer trabajar al programa en la tarea especifica que necesite, y se ahorra tener que escribir continuamente largas series de comandos para hacer el mismo trabajo una y otra vez.

En este capítulo aprenderá a usar funciones y cálculos simples en sus problemas, y a crear una serie de comandos específicos, llamados *procedimientos,* mediante el editor de programas.

Se presentan las siguientes funciones y comandos de ARCHIVE:

- Comandos:

editar	— edita procedimientos
escribir	— imprime información en la pantalla
cargar	— lee un fichero de programa en memoria temporal
salvar	— graba programas en memoria permanente
unir	— combina más de un fichero de programas
ejecutar	— ejecuta las instrucciones de un programa
nota	— permite una línea de comentarios en un programa

- Funciones:

cuenta()	—	calcula el número de registros de un archivo
mes()	—	muestra el nombre del mes
mayús()	—	convierte minúsculas en mayúsculas

Ideas a recordar

- El ordenador está para trabajar. Use teclas como **F5.** Escriba comandos mediante el editor.
- El editor de procedimientos es un editor de línea parecido al de EASEL y ABACUS.
- Salve los procedimientos con frecuencia para no perderlos.
- Si se atasca en algún momento en el editor, pulse **ESC.** Lo más que puede perder es la línea que está introduciendo.
- Es mejor que los procedimientos realicen una sola tarea; así podrá usarlos solos o como parte de otros procedimientos.
- No se puede salvar procedimientos selectivamente, ni unir sólo partes de un programa.

Ejemplo: Uso de la lista de invitados

Comienzo

Si no ha cargado ARCHIVE, hágalo ahora como se describe en el primer capítulo. Asegúrese de tener el fichero de datos con los ficheros agenda_dbf e invitado_dbf en la unidad 2. Use el comando dir para ver el contenido del cartucho si no está seguro.

CUENTA DEL FICHERO

El comando ESCRIBIR y la función CUENTA()

Eva quiere saber cuántos nombres y direcciones tiene que escribir todavía. Las direcciones que tiene en sus notas hacen un total de 45. Le pide a ARCHIVE que le indique cuántos registros tiene invitado.

```
ver "invitado"lógico "i"  ↵
      indicar  ↵
  escribir cuenta()
```

1. Abra el fichero invitado con el comando ver.
2. Escriba indicar y pulse ↵ para ver el fichero.
3. Introduzca escribir cuenta().

Observe que la palabra cuenta va seguida inmediatamente por un par de paréntesis, sin espacios intermedios. Si no hay nada entre los paréntesis, ARCHIVE escribe el número de registros del fichero activo.

⊡ 4. Pulse ↵.
El AREA DE VISUALIZACION se borra y, en la esquina superior izquierda, aparece escrito 12, que es el número de registros que contiene nuestro fichero. No representa el número de personas, ya que ARCHIVE no tiene en cuenta cuántos nombres aparecen en cada registro, ni la gente del campo cuántos. Como hay una sola dirección en cada registro, el número representa el número de cartas que habrá que enviar.

Para ver el resultado de CUENTA()

En algunos aparatos de televisión, si selecciona el modo monitor, la parte izquierda de la pantalla queda fuera de la visualización. Por ejemplo, puede que el número 12 del ejemplo aparezca como un 2.

Para mostrar el número en la columna 5:

escribir" ";cuenta( ) ⊡ *(5 espacios)*

1. Escriba escribir" ";cuenta() y pulse ↵.
ARCHIVE muestra el número 12 desplazado a la derecha cinco espacios.
El comando de esta línea es escribir. El comando escribir muestra lo que se le pida en la pantalla, no en la impresora.
Observe que si añade espacios entre comillas hace que ARCHIVE los imprima literalmente en la pantalla.
Para escribir texto, introdúzcalo entre comillas. Si no incluye comillas, ARCHIVE espera un nombre de campo o variable, un número o una función.
Observe el punto y coma ";" que se usa para separar los elementos que componen la sentencia. Si añadiera más elementos al comando, tendrían que ir también separados por un punto y coma. En la pantalla aparecerán todos en la misma línea.

Funciones

En términos informáticos, a cuenta() se le llama una *función*. Al utilizar funciones en una línea de instrucciones, éstas siempre producen un resultado en forma numérica o

de texto. Se dice que una función *devuelve un valor*. Observe que este "valor" no tiene por qué ser numérico; a menudo es una cadena de caracteres.

Por ejemplo, otras funciones "devuelven" el nombre de un mes si se les da un número, o convierten minúsculas en mayúsculas:

```
escribir mes(2)    escribe Febrero, el segundo mes
escribir mayús("palabra")    escribe PALABRA
```

Las funciones no son comandos: no se pueden usar solos en una línea de instrucciones. Se utilizan siempre en conjunción con un comando, por ejemplo, escribir cuenta().

Se puede reconocer una función por los paréntesis que lleva siempre al final de la palabra. Algunas funciones, como cuenta(), no tienen por qué llevar nada entre los paréntesis.

A la información que hay entre los paréntesis de una función se le llama técnicamente "argumentos" de la función.

Para contar los registros de un fichero abierto pero no activo, basta teclear el nombre lógico del fichero entre los paréntesis, por ejemplo:

```
escribir cuenta("maestro").
```

En cualquier otro caso, ARCHIVE supone que usted desea la cuenta del archivo activo.

escribir nombre$ ⊡

1. Introduzca escribir nombre$ y pulse ↵.
   El campo de nombre del registro activo aparece debajo de la cuenta del fichero en el AREA DE VISUALIZACION. ARCHIVE escribe

```
Juan y Pepa Martínez Rebollo
```

escribir 45-cuenta() ⊡

2. Escriba la línea anterior y pulse ↵.
   Aparece bajo el nombre de nuestros invitados un número que indica las direcciones que le faltan por escribir a Eva (45 es el número total de nombres en la lista). ARCHIVE escribe 33.
   Observe que el símbolo para el signo menos es el guión.

El resultado como fracción

Eva quiere saber qué proporción de su trabajo ha hecho hasta ahora.

escribir cuenta()/45 ↵

3. Escriba escribir cuenta()/45 y pulse ↵.
   El nuevo número en la pantalla representa el número de registros del fichero dividido por el número total de direcciones.

- Observe que el símbolo para la división es la barra inclinada "/".
  ARCHIVE escribe 0.2666666666667 *(algo más de un cuarto)*.

El resultado como porcentaje

Para verlo como un porcentaje:

F5 ↓ *100 ↵

4. Pulse **F5** para editar la última línea y añada *100 al final.
   La línea queda:

   escribir cuenta()/45*100

   Pulse ↵.

- Observe que el símbolo para la multiplicación es en ARCHIVE el asterisco "*", y no la equis "x".
  ARCHIVE escribe 26.6666666667.

Para ver el gasto en sellos

escribir cuenta()*17 ↵

5. Teclee escribir cuenta()*17 y pulse ↵ para multiplicar el número de cartas a enviar por el precio de un sello.
   ARCHIVE escribe 204, es decir, 204 pesetas.

F5 ↓ ↵ ↵ . ↵

6. Pulse **F5** para reeditar la anterior línea, y añada una coma a 17 para saber el precio en monedas de 100 pesetas.
   ARCHIVE muestra 2.04. Como norma general, los números decimales se indican con punto "." en vez de coma ",".

Expresiones y operadores aritméticos

Los signos de la multiplicación, división, adición y sustracción (*, /, +, —, ^) se llaman *operadores*. Se usan para realizar cálculos matemáticos con números, campos numéricos o funciones que devuelvan números. "Operan" sobre los valores que tienen a ambos lados. Por ejemplo:

```
escribir  1 + 2
escribir  cuántos/2
escribir  cuenta()*.17
```

No hace falta dejar espacios a los lados de los operadores, aunque se puede hacer para más claridad, por ejemplo, escribir 1+2 es lo mismo que escribir 1 + 2.

Un conjunto de valores y operadores se llama *expresión*. Veamos algunos ejemplos de expresiones:

```
1 + 2
cuenta/45*100
```

El término expresión se puede referir a un sólo valor.

cuenta() es una expresión numérica, ya que devuelve un valor numérico.

El símbolo ^ (mayúsculas y acento grave) indica exponenciación, que significa "elevar a la potencia". Por ejemplo,

```
escribir 2^2   muestra 4 (2 al cuadrado)
escribir 2^3   muestra 8 (2 al cubo)
escribir 2^4   muestra 16 (2⁴).
```

MIRANDO EL FICHERO DE INVITADOS

Restaurar la pantalla: el comando PANTALLA

Eva quiere echarle una ojeada a su lista de invitados.
Para restaurar la presentación:

pantalla ↵

1. Escriba pantalla y pulse ↵.
   Los números de la pantalla se borran, y en su lugar apa-

rece la presentación estándar de ficheros, que muestra el registro activo.
El comando pantalla muestra la presentación que ARCHIVE ha conservado en memoria temporal.
Si se ha usado alguno de los comandos indicar, alterar o insertar, se conserva en memoria una copia de la presentación estándar.
Si no existe una pantalla en memoria, el comando pantalla no tiene ningún efecto.
Se pueden crear las presentaciones que se desee, como se trata en el capítulo 10.

Para conseguir la presentación estándar

ARCHIVE presenta en *pantalla* una copia de la presentación de registros y guarda en *memoria* una copia siempre que se use el comando indicar. El comando indicar crea una presentación sencilla, con un campo en cada línea, y el nombre lógico del fichero en la línea superior.

Lo hace para cada fichero activo para el que se ejecute el comando indicar. Esta copia de pantalla permanece en memoria hasta que se cambia escribiendo otro comando indicar con otro fichero, o hasta que se vacía la memoria por el comando nuevo.

La pantalla se puede borrar temporalmente por los comandos limpiar o dir, o usando el comando escribir tras la petición ">". Así no se elimina la copia de memoria, y se puede restaurar el contenido de la pantalla usando el comando pantalla.

El comando pantalla vuelve a copiar en la pantalla la imagen guardada en memoria. No tiene ningún efecto si no hay una imagen de pantalla en memoria. Este comando actúa más rápidamente que indicar.

La presentación estándar de ficheros

La presentación estándar muestra los ficheros actuales de las variables de campo que representa, sea su fichero el activo en ese momento o no.

Esta situación puede llevar a ciertas situaciones confusas, por ejemplo:

- Si, por tener el mismo nombre, se muestran campos de otros ficheros.
- Si aparece en la pantalla un registro de un fichero que acaba de ser cerrado, o del que se acaban de borrar todos los registros.
- Si se muestra un registro a medio insertar que se abandonó (*este problema desaparece si se introduce* pescribir).

Cada vez que cambia el registro activo se imprime el nuevo en la pantalla, si los comandos se introducen desde el teclado tras el ">".

Los comandos insertar y alterar tienen el mismo efecto que el comando indicar, creando en memoria y pantalla la imagen de pantalla estándar (si no existía). Sin embargo, insertar no muestra el último registro del archivo, sino un conjunto de campos vacíos, listos para una inserción.

Se pueden crear formatos de pantalla, que se almacenan en cinta y se pueden cargar en cualquier momento, en lugar de utilizar el formato estándar que proporciona indicar. Permanecen en memoria hasta que se cambian por otro o por el formato estándar, o hasta que se borran al hacer nuevo. Desaparecen temporalmente de la pantalla por el comando limpiar, dir y las instrucciones de escritura directa, apareciendo de nuevo al hacer pantalla. Su manejo se discute en detalle en el capítulo 10.

Revisión del fichero

Eva se asegura de estar viendo el primer registro que introdujo en el fichero.

primero ⏎

1. Escriba primero y pulse ⏎.
   Eva revisa el fichero comprobando las direcciones que ha escrito mediante próximo (y de vez en cuando anterior), y usa **F5** para repetir el último comando que introdujo. Se cansa rápidamente de escribir esas instrucciones cada vez que quiere cambiar de comando.
   Intentando escribir más deprisa comienza a cometer errores. ARCHIVE, concienzudo en sus deberes, le indica que no reconoce los comandos prójimo y anterror. Los repite, en ocasiones más de una vez.

CREACION DE INSTRUCCIONES (PROCEDIMIENTOS)

Creación de un procedimiento para avanzar por el fichero

Decide acelerar las cosas creando nuevos comandos, o procedimientos, que hagan ese trabajo. Su nombre tendrá sólo una letra (a de anterior y p de próximo).

editar ⏎

1. Escriba editar y pulse ↵.
El AREA DE VISUALIZACION se borra, y la petición de comando ">" y la palabra editar desaparecen del AREA DE TRABAJO. En cambio aparece la palabra proc en la línea de entrada, y el cursor a continuación.

Desde este momento hasta que vuelva a aparecer la petición ">", el comando editar tiene el control de todo lo que hace ARCHIVE. El ">" sólo vuelve a aparecer cuando hayamos terminado con editar. Para abandonar editar, basta pulsar **ESC** hasta que reaparezca el ">".

> *Nota:* editar es el único sitio de ARCHIVE en que la tecla **ESC** no hace que el programa abandone automáticamente todo lo que estaba haciendo. En el editor, la tecla **ESC** sirve para dejar el modo de comandos y para abandonar el editor sin perder nada de lo hecho. El único sitio en que **ESC** pierde información es al pulsarla mientras se introduce una línea en el AREA DE TRABAJO. Se irá familiarizando con estas peculiaridades cuando aprenda a usar el editor.

En el AREA DE CONTROL se puede leer:

CREAR nuevo procedimiento. Escriba el nombre del procedimiento

ARCHIVE llama *procedimientos* a los comandos que usted crea. Se le pide que le diga a ARCHIVE el nombre que ha elegido para este procedimiento.

> *Nota:* Si comete un error al escribir una línea, corríjala usando las teclas de edición como hasta ahora. Las teclas del cursor funcionan en editar de la misma manera que cuando introduce un campo o un comando.

p ↵

2. Pulse p y luego ↵.
El AREA DE TRABAJO se vacía, y aparecen dos líneas en el AREA DE VISUALIZACION:

```
p       proc p
          finproc
```

Las palabras proc y finproc las incluye ARCHIVE, y no se pueden editar. El programa no funcionará sin ellas. Observe la forma que tiene ARCHIVE de insertar espacios en blanco (indentar). Se dará cuenta más adelante que resulta muy útil para saber la estructura de control de los programas complejos.
La p de la columna de la izquierda es el nombre del procedimiento. El texto de la zona principal del AREA DE VISUALIZACION representa el contenido del procedimiento.
El AREA DE CONTROL le pide que inserte una línea. El cursor espera en el AREA DE TRABAJO.

próximo ↵

3. Escriba próximo y pulse ↵.
La línea de entrada vuelve a desaparecer del AREA DE TRABAJO y reaparece en el AREA DE VISUALIZACION entre las líneas proc y finproc:

```
proc p
    próximo
    finproc
```

La línea próximo ha sido introducida en el procedimiento.

ESC

4. Pulse **ESC** para dejar de insertar líneas en el editor.

ESC

5. Pulse **ESC** para dejar de editar. El AREA DE VISUALIZACION se borra, y reaparece el ">". Ya no está bajo el control del editor, y ARCHIVE espera la siguiente instrucción.

pantalla ↵

6. Si la pantalla está vacía escriba pantalla y pulse ↵.
Así vuelve a aparecer el último registro que visualizó. El comando editar no altera nada de lo que el programa estaba haciendo cuando lo utilizó. Simplemente le permite editar y crear procedimientos (sus comandos personalizados).

Procedimientos

Un procedimiento es un conjunto de comandos de ARCHIVE, en secuencia y dotados de un nombre. Este nombre se puede utilizar exactamente de la misma manera que los comandos de ARCHIVE, para realizar una serie de tareas. Cualquier comando que se pueda introducir en la línea de entrada funcionará en un procedimiento.

Los procedimientos se utilizan igual que los comandos. Escriba el nombre del procedimiento tras el ">" y pulse ↵. ARCHIVE ejecuta los comandos que encuentra en el procedimiento en secuencia, de la misma forma que si los escribiera uno tras otro.

Cuando pulsa ↵ tras escribir una palabra en la línea de entrada, ARCHIVE mira su lista de comandos para ver si se corresponde a alguno. En caso contrario, ARCHIVE prueba los nombres de los procedimientos que ha creado, si hay alguno en memoria. Si los hay, y el nombre es el mismo, ARCHIVE ejecuta las instrucciones que encuentra entre las líneas proc y finproc. En caso contrario aparece el error comando no reconocido.

- Un nombre de procedimiento puede tener hasta 13 caracteres, debe comenzar con una letra y no contener espacios. No puede ser el mismo que una palabra clave de ARCHIVE.
- Un procedimiento puede contener hasta 255 líneas de comandos.
- Se pueden modificar los procedimientos una vez creados.

Los procedimientos se pueden almacenar en *microdrives* para su uso posterior. Al crearlos no se convierten en parte permanente de los comandos de ARCHIVE. Hay que grabarlos a la cinta al crearlos o cada vez que se cambien, o se perderán al borrar la memoria temporal del QL. Se deben "cargar" cada vez que se quiera utilizarlos.

Uso de los procedimientos

p ↵

1. Pulse p y después ↵.
   La presentación en pantalla cambia, presentando el siguiente registro como si hubiera escrito próximo.

p ↵

2. Hágalo otra vez. Ahora puede moverse por el fichero tan deprisa como si usara **F5**, pero puede utilizar otras instrucciones sin tener que volver a escribir el comando, y es más difícil cometer errores de escritura.

Otro procedimiento para moverse hacia atrás por el fichero

Para crear una versión abreviada de anterior, utilice el comando editar. Así vuelve a llamar al "editor de procedimientos".

El editor de procedimientos

El comando editar proporciona un *editor de texto* especial para escribir procedimientos. Un editor de texto procesa texto de una manera parecida a QUILL. Permite añadir, cambiar o borrar texto, desde palabras a bloques enteros. La diferencia principal entre editar y QUILL es que el editor de ARCHIVE es un editor de líneas, que modifica sólo una línea cada vez.

El comando editar permite crear, modificar y borrar procedimientos. Puede insertar líneas en los procedimientos, moverlas, borrarlas o modificarlas, y se pueden usar las teclas normales de edición, más las teclas **F1** y **F2**.

El editor de texto de ARCHIVE ayuda a escribir procedimientos proporcionando el comando inicial y el final, indentando las líneas para visualizar la estructura, y comprobando (hasta cierto punto) que las líneas son correctas. Observe que el editor no proporciona automáticamente comillas en los comandos que las necesitan.

ARCHIVE ordena los procedimientos por orden alfabético de nombres.

Es posible el uso de otro procesador de texto para escribir procedimientos de ARHIVE.

editar ↵

1. Escriba editar y pulse ↵.
El AREA DE VISUALIZACION se borra de nuevo, y la palabra editar desaparece del AREA DE TRABAJO. Esta vez, sin embargo, no se le pide que cree un nuevo procedimiento.
Si hay procedimientos ya creados en memoria, ARCHIVE los presenta en pantalla en primer lugar. El comando editar supone que usted va a crear un nuevo procedimiento sólo si no existe ninguno en memoria.
Se presenta en pantalla el comando p, y el AREA DE

CONTROL ofrece algunas opciones. Las del bloque central se refieren principalmente a la edición de los procedimientos existentes. La opción para crear un nuevo procedimiento no está a la vista. Para ver más opciones disponibles:

F3 **2.** Pulse **F3**, como le indica el bloque de la derecha. El AREA DE CONTROL cambia, mostrando ahora cuatro opciones más de edición que se pueden utilizar pulsando una sola tecla. No es necesario pulsar ↵ tras ninguna de ellas.

n **3.** Pulse n (nuevo procedimiento).
La pantalla se parece ahora a la que había cuando introdujo el primer procedimiento (excepto por la lista de los procedimientos existentes en el AREA DE VISUALIZACION). El AREA DE CONTROL le pide que introduzca un nombre, y proc está escrito en el AREA DE TRABAJO.

a ↵ **4.** Escriba a y pulse ↵.
El AREA DE CONTROL le pide que inserte una línea. El AREA DE TRABAJO aparece vacía, y el AREA DE VISUALIZACION muestra ahora:

```
a          proc a
p          finproc
```

La columna de la izquierda muestra los nombres de todos los procedimientos que hay en memoria. Uno de ellos, el procedimiento activo, se muestra a la derecha. ARCHIVE ha creado el procedimiento a, y lo ha insertado en la lista de procedimientos a la izquierda de la pantalla. La lista está en orden alfabético. La letra a aparece resaltada, ya que éste es el procedimiento activo.
El contenido del procedimiento activo se muestra en la parte derecha de la pantalla. La línea que acaba de introducir, proc a, aparece resaltada: es la línea activa.
La *línea activa* y el *procedimiento activo* son conceptos parecidos a los de registro activo y fichero activo. Son los últimos creados o modificados, y son los que se verán alterados por la próxima acción que realice.

anterror ↵ **5.** Escriba anterror (el error es intencionado) y pulse ↵. La línea de entrada desaparece del AREA DE TRABAJO, y se inserta en el procedimiento:

```
a          proc a
n             anterror
              finproc
```

Observe que la palabra anterror aparece resaltada. Es ahora la línea activa. ARCHIVE espera que inserte más líneas. Para corregir la línea activa, debe primero indicarle a ARCHIVE que ya no quiere insertar más líneas.

ESC **6.** Pulse **ESC**.
El bloque central del AREA DE CONTROL vuelve al menú principal del editor.
Se puede abandonar el modo de inserción pulsando ↵, o las flechas hacia arriba o abajo, en un momento en que el AREA DE TRABAJO esté vacía.

F5 **7.** Pulse **F5**.
La palabra anterror reaparece en el AREA DE TRABAJO. Puede usar ahora el *editor de línea.*

⇨ *(5 veces)* · CTRL ⇨

8. Cambie anterror por anterior. Modifique la línea mediante las teclas de edición.

↵ **9.** Pulse ↵ cuando haya acabado.
El AREA DE TRABAJO se borra y la línea corregida aparece en el procedimiento en el AREA DE VISUALIZACION. Ha terminado de crear el procedimiento a.
El procedimiento queda como sigue:

```
proc a
    anterior
    finproc
```

Nota: Si ARCHIVE no responde a las teclas de función en el editor, es probable que haya olvidado salir del modo de inserción: pulse **ESC**.

Para cambiar de procedimiento

Para ver el contenido del procedimiento p:

TAB **1.** Pulse la tecla de tabulación.
El AREA DE VISUALIZACION muestra ahora el primer procedimiento que escribió. La letra p aparece resaltada en el margen izquierdo, y la primera línea del procedimiento es la línea activa.
Observe que, aunque escribió p en primer lugar, ARCHIVE coloca los procedimientos en orden alfabético.
Cuando pulsa **TAB**, el programa avanza un procedimiento, y con ⇧ /**TAB** ARCHIVE retrocede al anterior.

Para abandonar el editor:

ESC 2. Pulse **ESC**.

Pruebe ahora los nuevos comandos a y p.

Borrado de líneas

Si inserta por error una línea de más al crear un procedimiento, se puede borrar abandonando el modo de inserción y pulsando **F3** seguido de q (por *quitar*). Este comando no tiene ningún efecto inmediato: se le pide que confirme, pulsando ↵, que desea borrar la línea activa. Si no quiere eliminarla, pulse **ESC**.

Inserción de líneas

Mientras aparezca la palabra INSERCION en el AREA DE CONTROL, se puede seguir introduciendo líneas en el AREA DE TRABAJO e insertándolas en el procedimiento cada vez que pulse ↵. Las nuevas líneas aparecen inmediatamente debajo de la línea activa. La línea activa siempre aparece resaltada en color blanco. Cada nueva línea se convierte en la activa a medida que se escribe.

En términos de ARCHIVE, usted se encuentra en modo de *inserción*. Esto quiere decir que todo lo que se puede hacer es insertar líneas. Como en el comando insertar, no se puede cambiar ninguna línea introducida sin salir primero del modo de inserción. Hasta el momento en que pulse ↵ se pueden cambiar las líneas mediante las teclas de cursor.

ARCHIVE le sitúa automáticamente en modo inserción cada vez que crea un nuevo procedimiento: es obvio, ya que no existe ninguna línea que editar. En otros momentos pulse **F4** si quiere insertar líneas, siempre que se encuentre en el menú principal del editor.

Para dejar de insertar líneas debe abandonar el modo de inserción. No se puede editar una línea previa, ni utilizar los comandos accesibles con **F3** mientras esté en modo inserción. Para salir pulse **ESC** (o ↵ o ⇧ o ⇩ con una línea vacía). La línea en la que estaba trabajando se abandona cuando pulsa **ESC**.

Pulse siempre ↵ para introducir las líneas. Para volver a modo inserción tras **ESC** pulse **F4**.

Creación de más procedimientos

Utilice el comando editar para crear procedimientos sencillos que usen los comandos indicar, alterar, insertar y editar, pero dándoles nombres de una sola letra, igual que anterior y próximo.

editar ↵

1. Escriba editar y pulse ↵.

F3

2. Pulse **F3**.

n

3. Pulse n para comenzar un nuevo procedimiento.

m ↵

4. Introduzca el nombre del nuevo procedimiento.

indicar ↵ nota mostrar ↵

5. Escriba el comando. El comando de la segunda línea (nota) sirve sólo para recordarle la abreviatura que se ha utilizado. ARCHIVE ignora las líneas que comienzan por nota.

↵

6. Recuerde siempre pulsar ↵ para salir de modo inserción tras cada procedimiento. Si pulsa **F3** o **F5** en modo inserción no ocurre nada.

F3 n c ↵ alterar ↵ nota cambiar ↵ ↵ F3 n i ↵ insertar ↵
↵ F3 n e ↵ editar ↵ ↵

Los procedimientos quedarán como sigue:

```
proc a
anterior
finproc
proc c
alterar :nota cambiar
finproc
proc e
editar
finproc
proc i
insertar
finproc
proc m
indicar :nota mostrar
finproc
proc p
prximo
finproc
```

ESC

7. Pulse **ESC** para dejar el editor.

Grabación de procedimientos: la instrucción SALVAR

Es importante que recuerde que mientras los ficheros de datos se graban automáticamente al cerrarlos, los procedimientos

se deben *salvar* explícitamente. ARCHIVE no le pide que lo haga y, por tanto, es fácil perder procedimientos por error.

Los procedimientos se pueden perder usando nuevo o abandonar, apagando la máquina o cargando otros procedimientos del cartucho.

Antes de seguir, Eva salva una copia de sus procedimientos en cinta en un cartucho llamado fácil. Hasta ahora sólo han sido creados en memoria temporal. Se perderán si se va la luz o se borra la memoria por cualquier razón.

salvar ↵ fácil ↵

1. Escriba salvar y pulse ↵ y después fácil y pulse ↵. El *microdrive* derecho gira; todos los procedimientos se salvan en cinta en un fichero de nombre fácil_prg. Es un *fichero de programa,* distinto de un fichero de datos. Al salvar los procedimientos, en contraposición a los ficheros de datos, éstos no quedan "cerrados" ni borrados de memoria temporal (RAM).

   En la terminología de ARCHIVE existe una distinción entre un procedimiento y un programa. Un programa es un conjunto de procedimientos que realizan ciertas tareas conjuntamente.

   Es una buena idea escribir los programas y procedimientos tan cortos como sea posible. Así se ahorra memoria, y resulta más fácil modificar los programas.

dir ↵ ↵

2. Observe el directorio de la unidad 2: fácil_prg aparece en la lista si ha sido grabado correctamente. La cinta debe contener los siguientes ficheros: agenda_dbf, invitado_dbf, invit1_dbf, invitado_bak y fácil_prg, y un fichero etiqueta como nov10 eti.

UN PROCEDIMIENTO INICIAL PARA ABRIR FICHEROS

Eva crea un procedimiento para abrir los ficheros que usa cada vez que arranca ARCHIVE.

Comienza a escribir un programa nuevo con los procedimientos necesarios, ya que está directamente relacionado con los ficheros de la boda, y lo quiere salvar bajo un nombre diferente a fácil, que se puede usar con cualquier fichero.

Quiere escribir procedimientos para abrir, cerrar y hacer copias de seguridad de ficheros que usa, ya que los comandos que hacen esta tarea son largos, llenos de comillas y es fácil introducir errores.

Para abrir el fichero de **invitados** y la **agenda**

Le da a los comandos nombres sin sentido pero cortos, como ai, que sustituye a abrir "invitado".

No son fáciles de recordar fuera del contexto del fichero de boda, pero son fáciles de usar.

nuevo ↵

1. Escriba nuevo y pulse ↵.
Si abrió los ficheros de datos con ver se borrarán con rapidez de la memoria. Si fueron abiertos mediante abrir se salvarán a la cinta, con los cambios realizados. La pantalla se borra, así como los procedimientos que hubiera en memoria.
Asegúrese de salvar los procedimientos si son nuevos o han sufrido cambios, antes de escribir el comando nuevo.

editar ↵

2. Escriba editar y pulse ↵.
Como no existe ningún procedimiento en memoria temporal, ARCHIVE espera que introduzca el nombre de un nuevo procedimiento.

ai ↵

3. Escriba ai y pulse ↵.
ARCHIVE pasa automáticamente a modo de inserción.

escribir "Abrir Fichero de Invitados" ↵

4. Introduzca escribir "Abrir Fichero de Invitados" y pulse ↵.

abrir "invitado" lógico "i" ↵

5. Escriba abrir "invitado" lógico "i" y pulse ↵.

ESC

6. Pulse **ESC** para abandonar el modo de inserción.

El procedimiento ai queda como sigue:

```
proc ai
    escribir "Abrir Fichero de Invitados"
    abrir "invitado" lógico "i"
finproc
```

F3

7. Pulse **F3**.

n

8. Pulse n para comenzar otro procedimiento.

aa ↵ escribir "Abrir Fichero Agenda" ↵ abrir "agenda" lógico "a" ↵

9. Llame aa al procedimiento y escriba las dos líneas siguientes:

```
escribir "Abrir Fichero Agenda"
abrir "agenda" lógico "a"
```

Nota: Utilice el comando escribir para que la pantalla presente información que permita seguir los pasos del programa. Si un procedimiento falla, será más fácil saber qué está hecho y dónde surgió el problema.

ESC **10.** Abandone el modo de inserción pulsando **ESC**.

El procedimiento aa debe quedar como sigue:

```
proc aa
    escribir "Abrir Fichero Agenda"
    abrir "agenda" lógico "a"
    finproc
```

Un procedimiento que abre ambos ficheros

F3 n
abrir ↵ **1.** Comience un nuevo procedimiento mediante **F3** y n.

2. Escriba abrir y pulse ↵.
No sucede nada: ARCHIVE rehúsa darle un nombre reservado a un procedimiento.

ambos ↵ **3.** Añada las letras ambos al nombre, que se convierte en abrirambos, y pulse ↵.

aa ↵ **4.** Inserte aa en la primera línea y pulse ↵.
Recuerde que sigue en modo inserción. Puede seguir insertando líneas escribiéndolas y pulsando ↵ hasta que salga del modo inserción. Las líneas aparecerán en el AREA DE TRABAJO según las escribe.

ai ↵ **5.** Escriba ai y pulse ↵.

El procedimiento queda como sigue:

```
proc abrirambos
    aa
    ai
    finproc
```

ESC **6.** Pulse **ESC** para dejar el modo de inserción. La línea anterior a la última queda resaltada. Es la línea activa.

⇧ **7.** Pulse la flecha arriba. La línea anterior se convierte en activa.

⇧ ⇩ ⇧ **8.** Pulse las flechas arriba y abajo para ver cómo funcionan. Observe que la línea finproc no puede hacerse nunca activa, ni tampoco cambiarla o borrarla. Se puede editar la primera línea, pero no la palabra proc.

> *Nota:* Para cambiar de nombre un procedimiento edite la línea superior.
>
> Observe que los tres procedimientos que acaba de crear se pueden usar por sí solos, o puede usar abrirambos para llamar a los otros dos.

Los procedimientos que contienen comandos con los nombres de otros procedimientos se dice que los *llaman*. Los comandos del procedimiento "llamado" se ejecutan por orden antes de devolver el control al procedimiento que lo llamó.

Procedimientos para cerrar los ficheros

Eva crea otros procedimientos de la misma manera que antes.

F3 n ca ↵

1. Escriba un procedimiento llamado ca (para cerrar agenda).

escribir "Cerrar Fichero Agenda" ↵

2. En primer lugar indíquele a ARCHIVE que imprima un mensaje diciendo qué fichero va a cerrar. Escriba la línea indicada y pulse ↵.

cerrar "a" ↵

3. Dígale ahora a ARCHIVE que cierre el fichero agenda, escribiendo cerrar "a" y pulsando ↵. Se usa a, ya que agenda se abrió con el nombre lógico a.

↵

4. Pulse ↵ para abandonar el modo de inserción.

El procedimiento debe quedar:

```
proc ca
    escribir "Cerrar Fichero Agenda"
    cerrar "a"
    finproc
```

F3 n ci

5. Pulse **F3** y n para crear un procedimiento similar para cerrar el fichero invitado, llamado ci.

escribir "Cerrar Fichero de Invitados" ↵

6. Introduzca la línea anterior y pulse ↵.

cerrar "i" ↵

7. Escriba cerrar "i" y pulse ↵.

↵

8. Pulse ↵ para abandonar el modo de inserción.

El procedimiento está acabado:

```
proc ci
    escribir "Cerrar Fichero de Invitados"
    cerrar "i"
    finproc
```

Un procedimiento que cierra ambos ficheros

F3 n cerrarambos ↵

1. Escriba ahora un procedimiento llamado cerrarambos que llame (es decir, utilice) los dos procedimientos que acaba de crear, ca y ci.

ca ↵

2. Escriba ca y pulse ↵.
   Así llama al procedimiento ca que cierra el fichero agenda.

ci ↵

3. Escriba ci y pulse ↵.
   Cuando ARCHIVE acaba de ejecutar los comandos de ca, devuelve el control a cerrarambos. Este llama en ese momento al procedimiento ci.

↵

4. Pulse ↵ para abandonar el modo de inserción.

El procedimiento cerrarambos debe quedar como sigue:

```
proc cerrarambos
    ca
    ci
    finproc
```

Un procedimiento para hacer copias de seguridad

Escriba ahora otros tres procedimientos que salvaguarden los ficheros agenda e invitado.

F3 n

1. Pulse **F3** y n para crear un nuevo procedimiento.

sala ↵

2. Llame al procedimiento sala *(SALvaguarda·Agenda)*.

escribir "Copiando el Fichero Agenda" ↵

3. Escriba un mensaje en la pantalla que indique lo que copia ARCHIVE.

salvagrdr "agenda_dbf" como "mdv1_agenda_dbf"

4. Indíquele a ARCHIVE que copie el fichero agenda a la unidad 1.

↵

5. Abandone el modo de inserción.

El procedimiento debe quedar como sigue:

```
proc sala
    escribir "Copiando el Fichero Agenda"
    salvagrdr "agenda_dbf" como "mdv1_agenda_dbf"
    finproc
```

F3 n sali ↵ escribir "Copiando Fichero de Invitados" ↵ salvagrdr "invitado_dbf" como "mdv1_invitado_dbf" ↵

6. Cree un segundo procedimiento como sala, pero haga que copie el fichero invitado a la unidad 1. No olvide

escribir un mensaje en la pantalla que indique qué fichero se va a copiar.

↵ **7.** Abandone el modo de inserción pulsando ↵.

El procedimiento debe quedar así:

```
proc sali
    escribir "Copiando Fichero de Invitados"
    salvagrdr "invitado" como "mdv1_invitado_dbf"
    finproc
```

F3 n copiambos ↵ sala ↵ sali ↵ ESC

8. Para acabar, cree un procedimiento que llame a los dos anteriores, sala y sali, con el nombre copiambos.

Este debe quedar como sigue:

```
proc copiambos
    sala
    sali
    finproc
```

Estos procedimientos están limitados a su uso con los ficheros de la boda, y perderán su utilidad a medida que usted desarrolle sus programas. Sin embargo, son útiles, ya que evitan cansancio y algunas posibilidades de error en las operaciones relacionadas con *microdrives*.

Un procedimiento general de copia

Para realizar copias de seguridad en medio de una sesión de trabajo necesitará cerrar antes los ficheros de datos, copiarlos a la unidad 1, y volverlos a abrir antes de continuar el trabajo.

F3 n hazcopia ↵ cerrarambos ↵ copiambos ↵ abrirambos ↵ ESC

1. Escriba un procedimiento que haga el trabajo por usted. Debe quedar parecido a éste:

```
proc hazcopia
    cerrarambos
    copiambos
    abrirambos
    finproc
```

Recuerde que un cartucho en mal estado puede causar fallos en las operaciones de cinta, y que no se pueda

cerrar los ficheros que no están abiertos. Si falla cualquier comando de un procedimiento, los que le siguen no se ejecutarán.

> *Nota:* El comando nuevo cierra todos los ficheros de datos abiertos y borra la memoria, pero no se puede usar en un procedimiento, ya que limpia también el resto del procedimiento que lo contiene.

UNION DE LOS NUEVOS PROCEDIMIENTOS

El procedimiento START y el comando EJECUTAR

F3 ~ start ↵ Nota FichBoda_prg 20 Nov 1985 ↵ Nota procedimientos para abrir, cerrar y copiar ↵ Nota ficheros de datos ↵ abrirambos ↵ ESC

1. Cree, para acabar, un procedimiento llamado start, que queda como sigue:

```
proc start
  nota Ficheros de la Boda, 11 Nov 1985
  nota Procedimientos para abrir, cerrar y copiar
  nota ficheros de datos
  abrirambos
  finproc
```

Un procedimiento llamado start tiene un significado especial para ARCHIVE. El comando ejecutar, seguido de un nombre de fichero de programa, carga los procedimientos de ese programa y busca uno llamado start, que se ejecuta automáticamente.

Observe el uso de nota para identificar el programa en uso, y la fecha de las últimas modificaciones. ARCHIVE ignora todo el resto de una línea que comienza por "nota".

ESC 2. Abandone el editor pulsando **ESC**.

> *Nota:* No se puede utilizar el comando ejecutar si el programa a cargar no contiene un procedimiento llamado start. Los procedimientos se cargan en memoria, pero aparece el error comando no reconocido.

Mezcla de procedimientos en memoria: el comando UNIR

unir ⏎ fácil ⏎

1. Traiga el programa fácil a memoria temporal, de forma que pueda usarse junto a los procedimientos que acaba de crear. Hágalo mediante el comando unir. Escriba unir y pulse ↵; a continuación escriba fácil y pulse ↵.

El comando unir le indica a ARCHIVE que mezcle los procedimientos en memoria con los contenidos en el fichero indicado. Si ARCHIVE encuentra dos procedimientos con el mismo nombre, los del fichero sustituyen a los residentes en memoria.

Sea cuidadoso para no perder procedimientos de esta manera.

Este comando no se debe utilizar desde un procedimiento (en la versión 1), ya que puede causar errores.

Almacenamiento de procedimientos

Salve los nuevos procedimientos al cartucho con el nombre FichBoda.

salvar ⏎ FichBoda

1. Escriba salvar y pulse ↵, y a continuación FichBoda y pulse ↵. Los procedimientos se almacenan en la cinta del *microdrive* 2 con el nombre FichBoda_prg.

Haga una copia de seguridad

Ponga el cartucho de seguridad en el *microdrive* 1 para copiar los procedimientos que acaba de escribir. Utilice **F5** para editar la línea anterior, y añada mdv1_ al principio del nombre. Así, los procedimientos se almacenan en la unidad 1 con el mismo nombre que en el 2.

F5 ↑ ⇨ ⇨ mdv1 ⏎

1. Añada mdv1_ al nombre de fichero y pulse ↵.
   Es la manera más fácil de realizar copias de seguridad de los ficheros de programa con los que trabaje. Se puede usar el comando salvagrdr. Si usa este comando recuerde indicar la extensión _prg con el nombre de fichero. El comando salvagrdr copia los programas, aunque estén activos en memoria.

> *Nota:* Como crear, cerrar y salvagrdr, el comando salvar borra, en la versión 1, cualquier programa previo del mismo nombre sin ninguna advertencia.

Es buena idea salvar los programas con distintos nombres según los escribe. Por ejemplo, nombre versiones sucesivas con procs1, procs2, procs3, etc., hasta estar seguro de su buen funcionamiento. Salve la versión final con el nombre procs. Así, si realiza cambios inadecuados, puede volver a la versión anterior.

FichBoda_prg contiene ahora los procedimientos que se ven a continuación. Use la tecla **TAB** (o ↑↑ /**TAB**) para moverse por la lista. El capítulo 6 explica cómo listar los procedimientos en papel, si dispone de impresora.

> *Nota:* No se puede LISTAR a la pantalla a menos que utilice antes el comando VIA PANTALLA; en cualquier caso, no es necesario hacerlo.

Procedimientos en FichBoda_prg

El programa quedará como sigue:

```
proc a
  anterior
  finproc
proc aa
  escribir "Abrir Fichero Agenda"
  abrir "agenda" lógico "a"
  finproc
proc abrirambos
  aa
  ai
  finproc
proc ai
  escribir "Abrir Fichero de Invitados"
  abrir "invitado" lógico "i"
  finproc
proc c
  alterar :nota cambiar
  finproc
proc ca
  escribir "Cerrar Fichero Agenda"
  cerrar "a"
  finproc
proc cerrarambos
  ca
  ci
  finproc
proc ci
  escribir "Cerrar Fichero de Invitados"
  cerrar "i"
  finproc
proc copiambos
  sala
  sali
  finproc
```

```
proc e
  editar
  finproc
proc hazcopia
  cerrarambos
  copiambos
  abrirambos
  finproc
proc i
  insertar
  finproc
proc m
  indicar :nota mostrar
  finproc
proc p
  próximo
  finproc
proc sala
  escribir "Copiando el Fichero Agenda"
  salvagrdr "agenda_dbf" como "mdv1_agenda_dbf"
  finproc
proc sali
  escribir "Copiar Fichero de Invitados"
  salvagrdr "invitado_dbf" como "mdv1_invitado_dbf"
  finproc
proc start
  nota Ficheros de la Boda, 11 Nov 1985
  nota Procedimientos para abrir, cerrar y copiar
  nota ficheros de datos
  abrirambos
  finproc
```

dir ↵ ↵

1. Haga un directorio para comprobar que FichBoda se ha salvado en la unidad 2. La cinta debe contener los ficheros agenda_dbf, invitado_dbf, invit1_dbf, invitado bak, fácil_prg y FichBoda_prg, y la etiqueta de fecha.

UNA RUTINA DE COPIA DE UTILIDAD GENERAL

El comando LEER

No es posible sobreestimar la importancia de hacer copias de seguridad de los programas. Copiar es aburrido, y cuanto más fácil le resulte más veces lo hará. En esta sección se escribe un programa general para realizar copias en cualquier momento y en cualquier situación.

El número de procedimientos escritos para copiar crecerá con rapidez cuando siga creando ficheros de datos y de programas. El procedimiento siguiente le pide que indique el nombre del fichero que se debe copiar, y lo copia con el mismo nombre en el

microdrive 1. Tenga cuidado de colocar el cartucho de seguridad en la unidad 1.

Añada el proccdimiento copia a su colección. Le pedirá que introduzca un nombre de fichero y enviará el comando necesario a ARCHIVE.

editar ↵

1. Escriba editar y pulse ↵.

F3 n copia

2. Pulse **F3**, luego n y el nombre: copia.

leer "¿Nombre completo del fichero? "; fn$ ↵

3. Escriba la línea anterior y pulse ↵. Observe el espacio en blanco entre la interrogación y las comillas.

El comando leer tiene la sintaxis leer "mensaje";variable. Espera que introduzca información por la pantalla, y supone que ha terminado cuando pulsa ↵. Imprime el mensaje entre comillas como petición, para recordarle que tiene que escribir algo.

Observe que el mensaje le recuerda que escriba el nombre completo, y que es el nombre del fichero origen el que debe introducirse.

La información que escriba se almacena en memoria bajo el nombre que sigue al comando leer; en este caso fn$. fn$ es una *variable,* ya que su contenido varía según lo que escriba.

salvagrdr fn$ como "mdv1_" + fn$ ↵ ESC

4. Escriba ahora la línea que copia el fichero cuyo nombre acaba de leer a la unidad 1.

Observe que, aunque no escriba un espacio entre fn$ y como, ARCHIVE lo coloca automáticamente. ARCHIVE sabe que en un comando el dólar aparece sólo al final del nombre de variables, así que la letra siguiente debe ser el comienzo de otra palabra.

Observe el uso del signo +, para añadir las letras mdv1_ al nombre de la variable. Observe también que fn$ no lleva comillas, mientras que "mdv1_" sí.

Uso de las VARIABLES

La variable es, en este caso, fn$, que significa Fichero-Nombre, y acaba en $ porque los nombres de ficheros son "cadenas" de caracteres, y no números. Se utiliza para almacenar el nombre de fichero que usted escriba, de manera que se pueda utilizar a continuación por el comando salvagrdr.

El comando salvagrdr requiere normalmente que los nombres de ficheros aparezcan entre comillas. ARCHIVE lee a continuación los caracteres entre las comillas para saber qué fichero debe

usar. Si la palabra no está entre comillas y acaba en un $, ARCHIVE sabe que la palabra es una variable. Busca en memoria una variable con ese nombre, y usa los caracteres que contenga como nombre de fichero.

En este caso, los dos nombres que necesita salvagrdr deben ser distintos, así que al segundo se le añade el literal mdv1_ delante de los caracteres para crear un nuevo nombre. Así, le dice a ARCHIVE que copie el fichero al cartucho de la unidad 1, la izquierda.

El procedimiento debe quedar como sigue:

```
proc copia
    leer "¿Nombre completo del fichero? ";fn$
    salvagrdr fn$ como "mdv1_" + fn$
    finproc
```

ESC **5.** Abandone el editor.

Intente ejecutar el nuevo procedimiento ahora:

copia ↵ **6.** Escriba copia y pulse ↵.
Aparece en el AREA DE VISUALIZACION el mensaje "¿Nombre completo del fichero? ", con el cursor tras él. ARCHIVE espera que escriba algo a continuación.
Asegúrese de tener la cinta de seguridad en la unidad 1.

fácil_prg **7.** Escriba fácil_prg. Las letras que introduce se escriben junto al mensaje en el AREA DE VISUALIZACION.

↵ **8.** Pulse ↵.
El fichero se copia de la unidad 2 a la 1.

Variables

Las variables son como las variables de campo (véase el capítulo 2), pero sin conexión con ningún registro o fichero. El nombre de una variable es siempre el mismo, pero su contenido varía según su definición.

Las variables sirven para almacenar caracteres y números en memoria temporal, pero no en registros o ficheros.

Nombres de variable:

- Deben comenzar por una letra.
- Pueden tener hasta 13 caracteres de longitud.
- No pueden contener espacios.

Si el nombre acaba en un dólar, "$", el contenido puede ser texto o números. En ese caso, se les llama *variables de cadena* (o alfanuméricas), ya que contienen cadenas de texto.

Si el nombre no acaba en un dólar, el contenido debe ser numérico. Se permite el signo menos y el "punto" decimal (.). Si intenta almacenar otra cosa se origina un error. Este tipo de variable se llama *variable numérica.*

No se pueden utilizar las palabras claves de ARCHIVE como nombre de variable: por ejemplo, abrir$ o lógico$ son inaceptables.

Se puede elegir arbitrariamente, con las limitaciones indicadas, el nombre de cada variable; pero es mejor que los nombres le recuerden para qué sirve la variable, y que sean cortos, para ahorrar tiempo de escritura y mantener cortos los programas.

Nota: Es sorprendente lo fácil que es olvidar el cartucho de programa ARCHIVE, en la unidad 1, cuando hace una copia de seguridad. Si el cartucho está protegido de escritura por habérsele quitado la lengüeta de protección, el cartucho girará durante un rato, y es posible que "cuelgue" el ordenador. Si no está protegido, el cartucho se llenará enseguida y perderá el tiempo de nuevo.

Advertencia: Si el procedimiento no funciona, puede ser porque algunas versiones de ARCHIVE se niegan a escribir sobre un fichero ya existente. En este caso, modifique el procedimiento añadiendo la línea 'tirar "mdv1_" + fn$ *antes* de la línea "salvagrdr". El comando tirar elimina el fichero indicado de la unidad 1 antes de hacer la copia. Para más información vea el próximo capítulo.

Adición de cadenas de caracteres

Observe que se pueden juntar cadenas de caracteres utilizando el signo +. Por ejemplo, escribir "Sr. " + nombre$ + apell$ podría escribir Sr. Luis Martínez. Se puede usar la adición con cualquier variable o texto literal. El proceso se llama *concatenación* en la jerga informática.

Un procedimiento más comunicativo

El procedimiento copia funciona como está, pero se puede mejorar mucho haciéndolo un poco más comunicativo.

editar ↵ TAB *(8 veces)* F4 escribir "Copia de un fichero (2 a 1)." ↵ escribir "Sitúe el cartucho en la unidad 1" ↵ escribir ↵

1. Use el editor para añadir las líneas siguientes al comienzo del procedimiento:

```
escribir "Copia de un fichero (2 a 1)."
escribir "Sitúe el cartucho en la unidad 1"
escribir
```

El comando escribir sin literales imprime una línea en blanco en la pantalla, y se utiliza por razones estéticas, para hacer más legible la presentación.

Cuando acaba la ejecución de un comando, el cursor se limita a volver a la línea de comandos. No hay ninguna otra indicación de que el comando ha finalizado, y a veces ARCHIVE está listo para otro comando bastante antes de que se detengan los *microdrives*.

⇩ ↻ F4 escribir "Copiado el Fichero '" + fn$ + "' a la unidad 1" ↵

2. Añada la línea siguiente al final del procedimiento para saber que ha acabado:

```
escribir "Copiado el Fichero '" + fn$ + "' a la unidad 1"
```

Observe el apóstrofo junto a las comillas. Es una manera de escribir entre comillas sin que el ordenador dé el error Faltan comillas de cierre. La construcción "'" crea un literal con un apóstofo en el interior. El apóstrofo se consigue mediante CTRL + acento.
Si el comando salvagrdr falla por alguna razón, el procedimiento se detiene y el mensaje final no se imprime.
El procedimiento queda ahora como sigue:

```
proc copia
  escribir "Copia de un Fichero (2 a 1)."
  escribir "Sitúe el cartucho en la unidad 1"
  escribir
  leer "¿Nombre completo del Fichero? ";fn$
  salvagrdr fn$ como "mdv1_" + fn$
  escribir "Copiado en Fichero '" + fn$ + "' a la unidad 1"
  finproc
```

↵ ESC

3. Abandone el modo inserción pulsando ↵, y pulse **ESC** para salir del editor.

salvar ↵ FichBod1 ↵

4. Escriba salvar y pulse ↵; escriba luego FichBod1 y de nuevo ↵ para salvar los procedimientos.

copia ↵

5. Escriba copia y pulse ↵.

FichBod1 ↵

6. Escriba FichBod1 y pulse ↵. El fichero de programa FichBod1_prg contiene los siguientes procedimientos:

a, aa, abrirambos, ai, c, ca, cerrarambos, ci, copia, copiambos, e, hazcopia, i, m, p, sala, sali, start

nuevo ↵

7. Escriba nuevo y pulse ↵ para borrar toda la memoria.

GRABACION Y CARGA DE PROCEDIMIENTOS

Carga de procedimientos: el comando CARGAR

Si acaba de borrar la memoria o de arrancar ARCHIVE no habrá ningún procedimiento en memoria. Antes de utilizar los procedimientos que acaba de definir, debe hacer una copia en memoria temporal leyéndolos de la cinta. El comando cargar lee un fichero de procedimientos a la memoria.

cargar ↵ FichBoda ↵

1. Escriba cargar "FichBoda" y pulse ↵ antes de usar los nuevos comandos.

start ↵

2. Escriba start y pulse ↵ para abrir los ficheros de boda.

También puede usar el comando ejecutar "FichBoda", que hace exactamente lo mismo que estos dos comandos. Utilice el comando hazcopia periódicamente para salvar los ficheros de datos.

> *Nota:* Los programas largos se cargan en memoria por trozos, con pausas en las que el cartucho no gira. No se preocupe, es normal. Sea paciente con el comando cargar.

Notas sobre la grabación de procedimientos

- Si une varios programas, los procedimientos que los componen no se podrán volver a salvar por separado. A menos

que borre procedimientos, se grabarán todos juntos en el fichero que indique en el comando salvar.
- Para mantener los programas separados y de un tamaño razonable, borre los procedimientos en memoria antes de comenzar a escribir uno nuevo, o de cargar uno que quiere cambiar.

Un procedimiento que borra los programas, pero no los datos

A menudo es necesario borrar los procedimientos de memoria tras salvarlos, de forma que se pueda comenzar a escribir un nuevo programa.

El comando nuevo cierra todos los ficheros de datos abiertos, además de borrar los procedimientos. Así se pierde el tiempo mientras se salva a la cinta el fichero de datos, y debe reabrirlos a continuación.

Evite este problema creando un programa con un procedimiento vacío en él. Lo puede cargar en cualquier momento para borrar los procedimientos en memoria.

> *Nota:* Al cargar un programa se borran todos los procedimientos en memoria, sustituyéndolos por los propios.

Para escribir el procedimiento hay que limpiar la memoria con nuevo.

nuevo ↵ editar ↵ vacío ↵ ESC ESC

1. Borre toda la memoria temporal mediante nuevo y ↵.

2. Cree un programa vacío con el procedimiento:

```
proc vacío
finproc
```

salvar ↵ vacío ↵

3. Utilice salvar para crear un fichero de programa llamado vacío.

Uso del fichero de programa VACIO

cargar ↵ vacío ↵

1. Cuando quiera borrar los procedimientos de la memoria, escriba cargar "vacío" y pulse ↵. Así limpiará los procedimientos de memoria.

editar ↵

2. Escriba editar y pulse ↵.

F3 b ↵

3. Antes de comenzar los nuevos procedimientos borre vacío pulsando **F3**, b, y pulsando ↵.

Notas al salvar y cargar procedimientos

- Los procedimientos, como los ficheros de datos, se pueden crear y modificar sólo en memoria, y se deben salvar a la cinta si no quiere perderlos cuando se apague la máquina o se borre la memoria.
- Del mismo modo, se debe cargar de cinta cualquier procedimiento que quiera utilizar.
- Salvar y cargar procedimientos es como cerrar y abrir ficheros de datos. Un comando los graba en cinta, el otro los lee de nuevo a memoria. La diferencia es que al salvar programas éstos quedan en memoria. ARCHIVE hace una copia en cinta, pero no los borra de la memoria.
- Los procedimientos se almacenan en cinta como ficheros. Se trata de ficheros de programa, que contienen procedimientos, en lugar de registros. Se pueden salvar hasta 255 procedimientos en un sólo fichero, si hay espacio libre en el cartucho.
- Los procedimientos se almacenan en cinta con un nombre de fichero. Las reglas para nombres de ficheros de programa son las mismas que para ficheros de datos: hasta ocho caracteres de longitud, comienzo por una letra y sin espacios en blanco. El nombre de fichero no tiene por qué ser el nombre de ninguno de los procedimientos.
- ARCHIVE proporciona automáticamente una extensión _prg al nombre (de PRoGrama). Se puede usar otra, pero debe recordar usarla siempre que cargue o salve el fichero.
- Los procedimientos se salvan en cinta con el comando salvar; por ejemplo, salvar "fichproc".
- Los programas se cargan en memoria temporal mediante el comando cargar; por ejemplo, cargar "fichproc". El comando cargar borra cualquier procedimiento en memoria temporal y los sustituye por los del fichero.
- El comando ejecutar carga un programa de la misma manera, pero ejecuta el procedimiento llamado start al terminar.
- Para cargar un fichero de programa sin perder los procedimientos presentes use el comando unir; por ejemplo, unir "fichpro2". Si algún procedimiento en el fichero tiene el mismo nombre que otro en memoria, el *nuevo sustituye* al *antiguo*.

RESUMEN

La función cuenta( ) con el comando escribir le sirvió a Eva para saber cuántos nombres y direcciones le quedan por escribir, y cuánto va a gastar en sellos de correos.

Para hacer las cosas fáciles, crea procedimientos para moverse fácilmente por el fichero. Tienen nombres de una sola letra, y llaman a los comandos editar, indicar, alterar, etc. Salva el programa en cinta con el nombre fácil_prg.

Eva sigue escribiendo procedimientos que facilitan la apertura, copia y grabación de ficheros. Estos se almacenan con el nombre FichBoda_prg. Recupera los procedimientos fácil mediante el comando unir, para poderlos usar junto a los que acaba de definir. Escribe un procedimiento llamado start que se ejecutará en primer lugar cuando use el comando ejecutar para cargar los procedimientos.

Tras esto Eva escribe un procedimiento general de copia, que obtiene el nombre del fichero a copiar mediante una variable y el comando leer. Para acabar, crea un procedimiento "vacío" que limpia la memoria temporal al cargarlo, sin cerrar los ficheros abiertos.

Trucos útiles

- Mantenga cortos los procedimientos.
- Los nombres de procedimiento deben tener significado, de forma que pueda recordar qué hacen. Así, los procedimientos que los usen serán fáciles de seguir.
- Use nombres de variable tan cortos y significativos como sea posible.
- Si los programas son pequeños (sólo unos pocos procedimientos), serán fáciles de unir para realizar tareas más complejas (sin tener que borrar o reescribir).
- Use sentencias de escritura en los procedimientos para saber qué están haciendo, sobre todo si requieren tiempos de espera apreciables.
- Use comandos nota para recordarle el objeto de sus procedimientos y programas.
- Cuando escriba y experimente con programas, salve las diferentes etapas de su desarrollo con nombres distintos. Así puede recuperar una versión anterior si se arrepiente de algún cambio. Para ello basta con cargar una versión anterior a los últimos cambios.

5

Bucles

Una de las ventajas de usar un ordenador para manejar grandes cantidades de datos es su capacidad para realizar tareas repetitivas.

ARCHIVE puede estar hora tras hora haciendo exactamente lo mismo sin protestar. Basta escribir unos pocos comandos una vez para hacer que ARCHIVE compruebe o cambie todos los registros de un fichero.

Este capítulo presenta los bucles, los comandos que permiten hacerlo. Puede crear un procedimiento que imprima todos los registros de un fichero. También escribirá un programa que copia datos de un fichero a otro, permitiendo cambiar las estructuras de datos y campos de una manera eficiente.

Se presentan las siguientes características de ARCHIVE:

- Comandos:

todos/fintodos	— repite un grupo de comandos con todos los registros de un fichero
mientras/finmientras	— repite un conjunto de instrucciones mientras persista cierta condición

posición — convierte un determinado registro en el registro activo

trazar — activa y desactiva la traza de los programas línea a línea

haz — asigna valor a una variable

actualiz — modifica un registro usando los valores de las variables de campo

tirar — borra un fichero

- Funciones:

recnum( ) — muestra el número de registro

finf( ) — indica que el registro activo es el último del fichero

Puntos de interés

- El bucle es la característica más importante, más útil y que más trabajo ahorra en cualquier lenguaje de programación.

  Las causas más corrientes de fallo usando bucles son:

- Comenzar en el punto inadecuado.
- Olvidarse de incrementar el contador de bucle (puntero de registro).
- Olvidar la condición de salida.

Ejemplo: Maya lo comprueba todo (avance rápido)

Maya sospecha que Eva ha eliminado a algunos de sus amigos de la lista y le va a echar la culpa a ARCHIVE. Maya inspecciona la lista para estar segura de que no se ha excluido a ninguna viuda desaliñada pero imprescindible, ni se ha incumplido ningún compromiso social inexcusable.

En cuanto acaba *Tocata* y las mellizas dejan tranquila la televisión, Maya conecta el QL, sintoniza el canal adecuado y carga ARCHIVE.

Si no acaba de cargar ARCHIVE, salve los procedimientos que haya en memoria y use nuevo para borrar los datos y programas de memoria.

ejecutar "fichboda" ↵

1. Maya pone el cartucho de datos en la unidad derecha y escribe ejecutar "fichboda".

indicar

2. Escribe indicar, y aparece en pantalla el primer registro del último fichero abierto.

Maya sospecha...

UN PROCEDIMIENTO PARA MOVERSE RAPIDAMENTE POR EL FICHERO

A pesar de la comodidad del procedimiento p de Eva, o de usar próximo y **F5**, Maya cree que se pueden hacer las cosas más fáciles escribiendo un procedimiento que recorra los registros automáticamente.

En primer lugar borra los procedimientos que se cargaron automáticamente al ejecutar "fichboda". Para ello carga el fichero vacío.

Instrucciones

cargar ↵ vacío ↵

1. Escriba cargar "vacío" y pulse ↵.
El procedimiento vacío se carga en memoria, borrándose el resto.

editar ↵

2. Escriba editar y pulse ↵.

F3 b ↵

3. Borre el procedimiento vacío pulsando **F3** y luego b seguido por ↵.
El procedimiento desaparece, dejando vacía la memoria de programa. ARCHIVE ofrece inmediatamente la oportunidad de crear un nuevo procedimiento, como si hubiera pulsado **F3** y n.

El comando TODOS

El comando todos, seguido siempre por el comando fintodos, va al primer registro de un fichero y se mueve consecutivamente hasta recorrer todo el fichero. Tras el último registro vuelve al principio del fichero y se detiene. Cualquier comando entre todos y fintodos se ejecutará con todos los registros del fichero.

Uso de TODOS en un procedimiento

zoom ↵

1. Escriba el nombre del nuevo procedimiento y pulse ↵.

todos ↵

2. Escriba todos y pulse ↵.

pescribir ↵

3. Introduzca pescribir y pulse ↵.
Este comando fuerza la impresión de cada registro en la presentación estándar, ya que la impresión no es automática en los procedimientos.

fintodos ↵

4. Escriba fintodos y pulse ↵.

↵

5. Pulse de nuevo ↵ para abandonar el modo de inserción.

El procedimiento debe quedar como sigue:

```
proc zoom
    todos
        pescribir
        fintodos
    finproc
```

ESC

6. Pulse **ESC** para salir del editor.

zoom ↵

7. Escriba zoom y pulse ↵.
Los registros del fichero aparecen en la pantalla uno tras otro, tan rápido como ARCHIVE y el QL logran locali-

zarlos y mostrarlos. De vez en cuando puede girar el *microdrive,* buscando más registros.

La secuencia de acontecimientos es:

- ARCHIVE lee el comando todos por primera vez y se sitúa en el primer registro del fichero activo. ARCHIVE espera un comando fintodos antes del final del procedimiento.
- Se ejecuta el comando pescribir para el primer registro.
- Al llegar al comando fintodos, ARCHIVE pasa al siguiente registro y vuelve por el procedimiento al comando siguiente a todos.
- Pescribir muestra el segundo registro.
- ARCHIVE pasa otro registro y el procedimiento realiza el ciclo todos, pescribir, fintodos de nuevo.
- El proceso sigue hasta que ARCHIVE pasa por todos los registros del fichero. Una vez alcanzado el último registro, al tratar fintodos de pasar al registro siguiente se alcanza el fin de fichero. ARCHIVE termina la secuencia todos/fintodos y lee el comando siguiente a fintodos por primera vez. En este caso es finproc, y el programa se detiene.

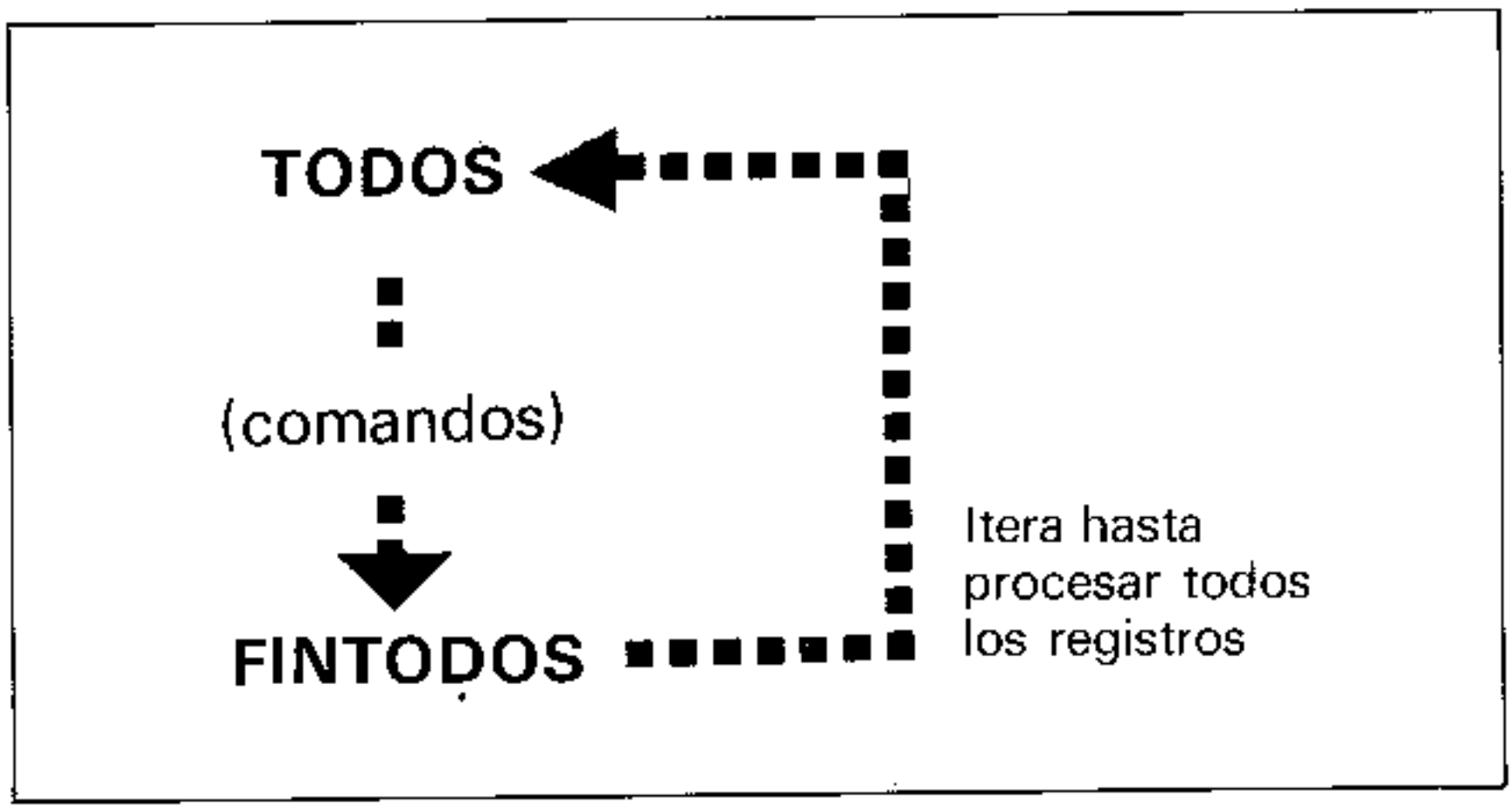


Figura 5.1. El bucle TODOS

Los programadores llaman bucles a las estructuras como ésta. Los bucles iteran (repiten) instrucciones cierto número de veces. En este caso, el número de veces es una vez para cada registro del fichero.

El comando todos debe ir seguido por un fintodos, o el procedimiento dará el error mala estructura de programa.

Por esta razón, el comando todos no se puede utilizar en la línea de comandos, ya que ARCHIVE se liará y puede llegar a "colgarse". Asegúrese de usar un fintodos en la misma línea si escribe todos en la línea de entrada.

> *Nota:* El comando todos está pensado para moverse por un fichero de la manera más rápida posible y, por regla general, muestra los registros en el orden en que fueron introducidos. ARCHIVE toma algunos atajos para hacerlo; por tanto, no se debe cambiar la secuencia o el contenido de los registros durante el bucle; por ejemplo, con los comandos alterar o borrar. El comando es, sobre todo, para búsquedas rápidas.

Escritura del número de registro

La memoria de Maya ya no es lo que era, y no se ha dado cuenta de que no están los Garciadejo (¡qué pesados!). Queda, pues, contenta con el trabajo de Eva. Quiere echarle otra ojeada a los registros que introdujo; desgraciadamente, el comando todos hace exactamente lo que indica su nombre, y pasa por todos los registros. Eva tecleó la lista de su madre tras la suya (corrigiendo sobre la marcha), así que Maya tiene que empezar en la parte central del fichero, y seguir hasta el final.

Sabe que hay 12 registros en el archivo, y que los suyos comienzan alrededor de la mitad.

posición 6 ↵

1. Escriba posición 6 y pulse ↵.
   El primer nombre de la lista de Maya aparece en la pantalla. Es el séptimo registro del archivo, pero ARCHIVE le da el número 6, ya que empieza a contar por cero. ARCHIVE escribe:

   Milena de Pablo Martínez

Un procedimiento que muestra el número de registro

editar ↵ F3 n escreg ↵ escribir "Número de registro: ";recnum( ) ↵ ESC

1. Utilice el editor para crear un nuevo procedimiento que muestra el número del registro activo:

```
proc escreg
    escribir "Número de registro: ";recnum( )
    finproc
```

Observe el punto y coma que separa el texto entre comillas del nombre de la función recnum( ). Le indica a ARCHIVE que escriba los dos elementos sin ninguna separación en la misma línea. Tenga en cuenta los dos puntos tras la palabra registro; si no los pone, no habrá separación entre el texto y el número.

La función RECNUM( )

Esta función devuelve el número del registro activo. Los registros se numeran consecutivamente, más o menos en la secuencia en que se insertaron, empezando por cero.

> *Nota:* ARCHIVE no siempre sitúa los registros en el archivo en el orden en que se insertan. Si se borran registros puede insertar otros nuevos en su lugar para ahorrar espacio.

ESC

2. Vuelva a la línea de comandos.

escreg

3. Escriba escreg y pulse ↵.
   Debe aparecer la línea Número de registro:6 en la parte inferior del AREA DE VISUALIZACION, sin borrar la pantalla.

> *Nota:* Si se introduce un comando escribir en la línea de comandos, se borra el registro que se estuviera mostrando en pantalla. Si se utiliza en un procedimiento, sin embargo, la visualización permanece y el comando lo escribe todo en la parte inferior de la pantalla, a menos que se especifique lo contrario.

La pantalla y los procedimientos

La visualización estándar, en memoria, actúa de diferente manera cuando se trabaja desde un procedimiento.

La visualización no se actualiza automáticamente cuando cambia el registro activo. Es preciso indicárselo al programa mediante el comando pescribir. Es una abreviatura de Pantalla-ESCRIBIR, y escribe los valores de los campos en la pantalla de visualización.

Además, el comando escribir no borra la visualización estándar de la pantalla. Los comandos escribir envían los datos a la parte inferior de la pantalla, a menos que especifique lo contrario.

El comando pescribir no hace nada si la visualización estándar no está en pantalla. Utilice el comando pantalla para reactivar la visualización estándar si está ya en memoria, o el comando indicar en caso contrario.

Uso del procedimiento ESCREG

primero ↵

1. Escriba primero y pulse ↵.
La pantalla muestra el primer registro.

escreg ↵

2. Escriba escreg y pulse ↵.
El número de la parte inferior de la pantalla debe cambiar a 0. ARCHIVE escribe la línea entera de nuevo, pero, como el resto de los caracteres no cambia, no hay ningún otro efecto.

Nota: Los números de registro no son fijos ni absolutos. Permanecen siempre en secuencia, incluso cuando se cambia la ordenación o se borran registros. El número de todos los registros puede cambiar.

Maya decide editar el procedimiento zoom de manera que comience en el registro activo, sea el que sea, y se siga moviendo hacia el final del archivo.

El comando MIENTRAS

editar ↵
TAB

1. Escriba editar y pulse ↵.
2. Pulse el tabulador hasta llegar al procedimiento zoom. La palabra zoom aparecerá resaltada en la parte izquierda de la pantalla.

⇩ F5
CTRL ⇩

3. Cursor abajo hasta la línea todos. Pulse entonces **F5.**
4. Pulse **CTRL** y la tecla ⇩ para borrar la línea entera.

mientras no finf() ↵

5. Escriba mientras no finf() y pulse ↵.
6. Baje el cursor hasta la palabra fintodos y pulse **F5.**

⇩ ⇩ F5
⇨ ⇨ ⇨ CTRL ⇩

7. Pulse la tecla ⇨ hasta pasar sobre la palabra fin. Mantenga pulsada la tecla **CTRL** y pulse ⇩ para borrar las letras todos.

mientras ↵
⇧ F4

8. Escriba mientras y pulse ↵.
9. Suba el cursor hasta la línea pescribir y pulse **F4** para insertar una línea a continuación.

escreg ↵

10. Escriba escreg y pulse ↵.

El procedimiento acabado queda como se ve a continuación.

```
proc zoom
    mientras no finf()
        pescribir
        escreg
        finmientras
    finproc
```

Los dos comandos, mientras y finmientras, crean un bucle en el procedimiento de una manera parecida a todos y fintodos. La función finf() se utiliza para comprobar si se ha alcanzado el final del fichero. El bucle continúa mientras (o hasta que) la función finf() devuelve el valor 0. En otras palabras, hasta el fin del archivo.

Conviene notar la indentación automática que realiza ARCHIVE de los comandos entre el mientras y el finmientras. Es parecida a la que realiza entre proc y finproc. Así es más fácil identificar los comandos incluidos en el bucle y ver la estructura del procedimiento.

↵ 11. Pulse ↵ para salir del modo de inserción.

ESC 12. Pulse **ESC** para salir del editor.

La función FINF()

Utilizamos la función finf() para detectar el momento en que se alcanza el fin de un fichero.

finf() devuelve el valor 1 si se ha alcanzado el fin del fichero, y 0 en caso contrario.

El comando último convierte al último registro del fichero en activo, y pone inmediatamente después finf() a 1.

El comando próximo pone a 1 finf() sólo cuando se ejecuta estando ya en el último registro. En otras palabras, escribir próximo en el penúltimo registro convierte en activo el último registro, pero no pone finf() a 1. Hace falta volver a escribir el comando.

Uso del procedimiento ZOOM corregido

zoom ↵ 1. Escriba zoom y pulse ↵.
No ocurre nada. El cursor se queda sobre el ">", indicando que el comando no ha acabado de ejecutarse. Tampoco ocurre nada si pulsa alguna tecla (excepto **ESC**).

ESC

2. Pulse **ESC.**
   Aparece ‹**ESC**› y a continuación el cursor, esperando una nueva línea.

Está claro que hay algún error en el procedimiento zoom. O bien se está moviendo por el fichero, pero sin mostrar en pantalla los cambios, o bien no se mueve del primer registro.

PARA SEGUIR LA PISTA A LOS ERRORES

Existe una manera de observar cómo ejecuta ARCHIVE los procedimientos:

trazar ↵

1. Escriba trazar y pulse ↵.
   No parece ocurrir nada especial. Simplemente le ha indicado a ARCHIVE que deseamos que le siga la pista a la ejecución de los procedimientos.

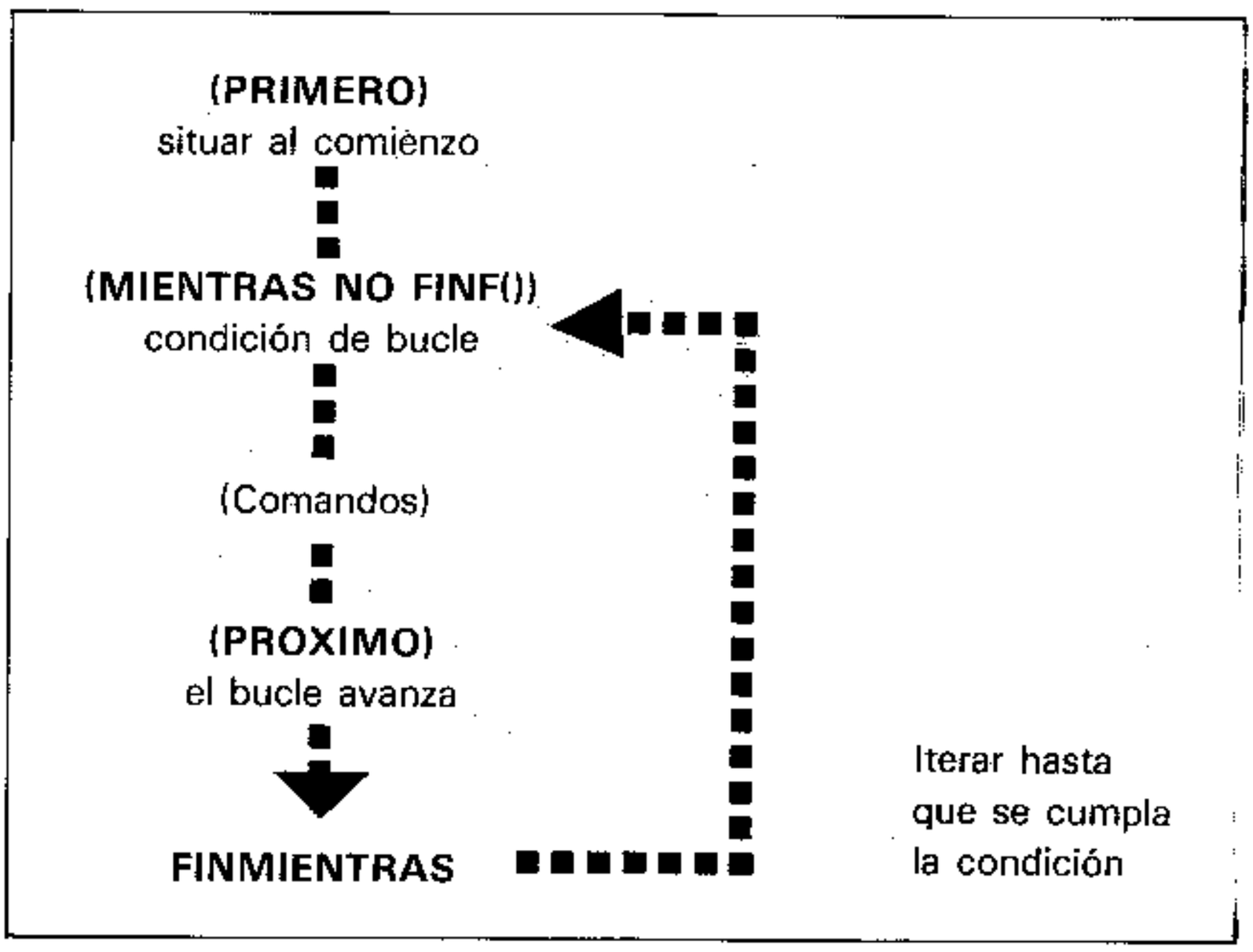


Figura 5.2. El bucle MIENTRAS NO FINF()

zoom ↵

2. Escriba zoom y pulse ↵.
   Las líneas del procedimiento aparecen en el AREA DE

TRABAJO a medida que ARCHIVE las procesa. El comando pescribir permanece algo más que los otros, que pasan demasiado deprisa para leerlos con facilidad.

ESP (manténgala pulsada)

3. Mantenga pulsada la barra espaciadora para detener temporalmente la ejecución de un procedimiento.

ESP (libérela)

4. Suelte la barra espaciadora para dejar que el procedimiento continúe su ejecución.

ESP (púlsela de nuevo)

5. Utilice la barra espaciadora para ver el procedimiento en acción.

La secuencia de acontecimientos es como sigue:

```
zoom      :mientras no finf()
zoom      :pescribir
zoom      :escreg
escreg    :escribir "Número de registro: ";recnum()
escreg    :finproc
zoom      :finmientras
zoom      :pescribir ...
```

y así la secuencia comienza de nuevo. La única línea que no reaparece es la primera: mientras no finf().

Observe lo fácil que resulta ver el efecto de la llamada por zoom al segundo procedimiento, escreg.

ESC

6. Pulse **ESC** para salir del procedimiento.

De la manera en que está, el procedimiento no acabará nunca, ya que está atrapado en un "bucle sin fin".

Nunca llegará al final del fichero, ya que ARCHIVE no recibe ninguna orden de avanzar al registro próximo. El bucle mientras tiene otros usos además de pasar por todos los registros de un fichero (como todos), y esto se le debe indicar explícitamente.

editar ↵ TAB

1. Use el comando editar y el tabulador hasta el procedimiento zoom, para modificarlo.

⇩ *(3 veces)*

2. Sitúese en la línea escreg mediante la tecla ⇩.

F4 próximo ↵

3. Pulse **F4** e inserte la línea próximo, que le indica a ARCHIVE que se mueva al registro siguiente después de ejecutar el procedimiento escreg.

El procedimiento queda como sigue:

```
proc zoom
    mientras no finf()
```

```
        pescribir
        escreg
        próximo
        finmientras
    finproc
```

↵ ESC

4. Pulse ↵ para salir del modo de inserción, y pulse entonces **ESC** para abandonar el editor.

zoom .↵

5. Introduzca el comando zoom modificado en la línea de entrada. Esta vez la visualización cambia para mostrar un registro cada vez, y el número de registro se incrementa en la línea inferior.
Se pueden ver también las líneas del procedimiento pasar por el AREA DE TRABAJO, ya que el comando trazar está activo todavia.
Cuando el procedimiento muestra el último registro, se detiene y reaparece el cursor tras el ">" en la línea de comandos.

Para desconectar la TRAZA

El procedimiento se ha ejecutado más lentamente de lo que habría sido posible, ya que la función de *traza* estaba conectada. Para apagarla:

trazar .↵

1. Escriba trazar y pulse ↵.
El comando trazar es como el botón de la luz: si se escribe el comando una vez, la *traza* se conecta. Para desconectarla hace falta escribirlo de nuevo. A un control de este tipo se le llama *interruptor*. Otro ejemplo de interruptor es la tecla **F2.**

El comando TRAZAR

El comando trazar imprime las líneas de los programas o procedimientos en el AREA DE TRABAJO a medida que las ejecuta.

La primera vez que se escribe el comando se conecta la *traza* y permanece conectada hasta que se vuelve a escribir el comando.

Se puede conectar y desconectar la *traza* mientras un procedimiento está ejecutándose, pulsando **F3** a la vez. Esta tecla actúa exactamente igual que el comando trazar.

Cuando se pulsa la barra espaciadora mientras se traza un procedimiento se suspende la ejecución del procedimiento; al soltarla continúa la ejecución.

Es útil dejar conectada la *traza* mientras se escribe un procedimiento. Sin embargo, disminuye la velocidad de ejecución de los programas, por lo que debe desconectarse una vez funcionen correctamente.

zoom ↵

2. Escriba zoom y pulse ↵.
El cursor vuelve a una nueva línea sin que haya ocurrido nada. El fichero está ya al final, así que el comando mientras no finf() ni siquiera llega a ejecutarse una vez.

posición 6 ↵

3. Escriba posición 6 y ↵.
Aparece en pantalla el séptimo registro, pero el número de registro que aparece impreso sigue siendo el 11. Para cambiar este número haría falta ejecutar el procedimiento escreg.

escreg ↵

4. Escriba escreg y pulse ↵.
La línea inferior ahora indica: Número de registro: 61. El nuevo número de registro ha sobreimpreso el primer 1 de 11, pero no el último.

Modificación de ESCREG

editar ↵

1. Utilice el editor para realizar una pequeña corrección a escreg.

↓

2. Pulse la flecha inferior para llegar a la línea escribir.

F5 ↓

3. Pulse **F5** para editar la línea, y pulse la flecha abajo para llegar al final de la línea.

;" " ↵

4. Añada ;" " al final de la línea, que quedará:

```
escribir "Número de registro: ";recnum();" "
```

Así cada número va seguido por un espacio en blanco, y nos aseguramos de que los números de registro hasta 99 se borrarán al sobreescribirlos.

ESC

5. Pulse **ESC** para abandonar el editor.

El procedimiento debe quedar como sigue:

```
proc escreg
    escribir "Número de registro: ";recnum();" "
    finproc
```

zoom ↵

6. Escriba zoom y pulse ↵.
Esta vez el procedimiento recorre los registros uno por uno, imprimiendo el número de registro en la parte inferior de la pantalla a continuación. El procedimiento se detiene al llegar al final del fichero, tras el registro número 11.

Uso de ZOOM con otro fichero

Se puede usar el mismo procedimiento, sin ningún cambio, para recorrer el fichero agenda.

usar "a" ↵
zoom ↵

1. Escriba usar "a" y pulse ↵.
2. Escriba zoom y pulse ↵.
El *microdrive* puede comenzar a girar, buscando la información del fichero "agenda" y los números de registro de la parte inferior de la pantalla se incrementan desde 0 a 14. Aparte de esto, no ocurre nada más, ya que la visualización está todavía preparada para el fichero "invitado".

indicar ↵

3. Escriba indicar y pulse ↵ para sustituir la visualización estándar del fichero "invitado" por la del fichero activo, ahora "agenda". El registro activo es ahora el último, hacer una lista de invitados.

primero ↵
zoom ↵

4. Escriba primero y pulse ↵.
5. Escriba zoom y pulse ↵.
Los registros pasan mucho más deprisa que en el fichero de invitados y no son tan fáciles de leer. Se puede adaptar fácilmente el bucle que acabamos de crear para que escriba las líneas una a continuación de otra en la pantalla.

CREACION DE UN PROCEDIMIENTO PARA LISTAR EL FICHERO

El procedimiento zoom se puede adaptar para que liste el fichero. Antes de hacer cambios en zoom:

salvar ↵ zoom ↵

1. Salve los procedimientos en un fichero llamado zoom.

Este contiene los procedimientos escreg y zoom, como sigue:

```
proc escreg
    escribir "Número de registro: ";recnum();" "
    finproc
proc zoom
    mientras no finf()
        pescribir
        escreg
        próximo
        finmientras
    finproc
```

editar ↵ TAB

2. Escriba editar y pulse ↵.
3. Pulse **TAB** para editar el procedimiento zoom.

F5 CTRL ⇩ listagenda ↵

4. Cambie el nombre del procedimiento a listagenda.

⇩ ⇩ F5 CTRL ⇨ ⇩ notas$ ↵

5. Cambie la línea pescribir hasta que se lea: escribir notas$.

⇩ F3 q ↵

6. Quite la línea escreg.

El procedimiento estará así:

```
proc listagenda
    mientras no finf()
        escribir notas$
        próximo
        finmientras
    finproc
```

ESC

7. Pulse **ESC** para abandonar el editor.

Duplicación de un procedimiento

Acabamos de convertir el procedimiento zoom en el procedimiento listagenda, pero no hemos perdido zoom. Está en cinta, bajo el nombre "zoom_prg". Se puede volver a introducir en memoria sin afectar a listagenda ni a escreg usando el comando unir.

unir "zoom" ↵
editar ↵

1. Escriba unir "zoom" y pulse ↵.
2. Escriba ahora editar y pulse ↵ para ver el efecto de este último comando.

Ambos procedimientos, zoom y listagenda, existen, ya que salvamos a cinta zoom antes de convertirlo en listagenda. Cuando este fichero se unió a la memoria, escreg fue sustituido por la versión idéntica que venía en la cin-

ta, pero ninguno de los otros dos procedimientos se vio afectado.
Utilice esta técnica para escribir procedimientos similares, pero ligeramente distintos, sin tener que introducirlos desde el principio.

Si salva los procedimientos por separado podrá volver a combinarlos de distinta manera en varios programas sin tener que volver a teclearlos cada vez.

Estudie la posibilidad de crear una biblioteca de procedimientos en un cartucho de *microdrive*.

Resumiendo, se puede duplicar un procedimiento salvándolo en cinta, cambiándole el nombre en memoria y uniéndolo de nuevo a la memoria.

ESC **3.** Pulse **ESC** para salir del editor.

Uso del procedimiento de listado

primero ↵ **1.** Escriba primero y pulse ↵.
listagenda ↵ **2.** Escriba listagenda y pulse ↵.
Los 14 registros pasan rápidamente por la línea inferior de la visualización, sobreimprimiéndose unos a otros.
Como ya explicamos, las instrucciones ESCRIBIR en los procedimientos se comportan de esa manera cuando la visualización estándar está activa.

editar ↵ F4 limpiar ESC
3. Use el EDITOR para insertar el comando limpiar en la primera línea tras el nombre en el procedimiento listagenda.
Este comando borra el AREA DE VISUALIZACION.
Muchos procedimientos importantes necesitan comenzar por un comando que limpie o prepare la pantalla.

ESC primero ↵ **4.** Abandone el editor y haga que la primera anotación sea el registro activo del fichero.
listagenda ↵ **5.** Escriba listagenda y pulse ↵. Cada anotación del fichero se imprime en una nueva línea en el AREA DE VISUALIZACION.

El resultado debe quedar como sigue:

```
notas%
Ver al sacerdote
Pedirle a Felicidad que sea la madrina
Decirle a Juana que haga los trajes
```

```
Decirle al Sr Hort que se encarque de las flores
Aue no se olvide hacer los ojales
Zapatos blancos
Comprar el traje
¿ Ha comprado Salva un traje de soort ?
Contratar a Perez Fotògrafos
Coche
Llamar a Sara ¿ Quièn les organizò el banquete ?
Capacidad de la sala Azuï del Hotel Almirante
Encargar la tarta
Buscar un Hotel para los invitados
Hacer una lista de bodas
```

Impresión del listado en papel

Si dispone de impresora resulta muy fácil obtener un listado en papel: basta cambiar el comando escribir por imprimir. El uso de impresoras se explica con más detalle en el próximo capítulo. El procedimiento queda así cuando se adapta para imprimir su salida sobre papel:

```
proc listagenda
    limpiar
    mientras no finf()
        imprimir notas$
        próximo
        finmientras
    finproc
```

Varias instrucciones en la misma línea

Se pueden escribir varios comandos en la misma línea, si se separan con dos puntos (:).

```
:odos:escribir notas$:fintodos
```

1. Escriba la línea anterior y pulse ↵.
   El resultado es exactamente igual al que habría obtenido escribiendo primero seguido por listagenda. El uso de esta técnica permite introducir directamente comandos que necesitan ir seguidos por su correspondiente fin..., como fintodos o finmientras. Sin embargo, el número máximo de caracteres de una línea es de 255, y puede resultar inconveniente, especialmente al no poderse salvar la línea.
   La escritura de varios comandos en la misma línea puede ser también una buena manera de hacer que los procedimientos queden más compactos y legibles.

Grabación de los procedimientos

Salve ahora los procedimientos existentes como "zoom_prg". La memoria contendrá los siguientes procedimientos:

```
proc escreg
  escribir "Número de registro:";recnum();" "
  finproc
proc listagenda
  limpiar
  mientras no finf()
    escribir notas$
    próximo
    finmientras
  finproc
proc zoom
  mientras no finf()
  pescribir
  escreg
  próximo
  finmientras
finproc
```

tirar ↵ zoom_prg ↵

1. Elimine el viejo zoom_prg del cartucho usando tirar.

salvar ↵ zoom ↵

2. Escriba salvar "zoom" y pulse ↵.

F5 ⇧⇨ ⇨ mdv1_ .↵

3. Tras comprobar que la cinta de seguridad está en la unidad 1, salve los procedimientos editando la línea.

ADICION DE CAMPOS A UN ARCHIVO

Mirando sus notas, Maya se da cuenta de que Eva podía haber seguido apuntando sus notas en trozos de papel, ya que, como está el fichero "agenda", ARCHIVE no podrá serle de gran utilidad. No resulta fácil saber qué tareas son urgentes, o bien si requieren una escalofriante expedición de compras o basta con una llamada telefónica.

Decide añadir dos nuevos campos al fichero "agenda", uno con la fecha límite de los recados y otro que contenga un código que indique qué hace falta hacer.

No se puede alterar la estructura de registros del fichero "agenda". Por tanto, se debe crear un nuevo fichero y transferir las anotaciones.

Creación de la nueva estructura de archivo

Si ha olvidado cómo se utiliza crear, o bien si tiene algún problema, acuda a la sección inicial de los capítulos 2 y 3.

```
crear "nueagen" lógico "n"  ↵
```

1. Escriba la línea anterior y pulse ↵.

```
        notas$  ↵
```

2. Escriba notas$ y pulse ↵.
   Este es el campo que contenía el fichero "agenda".

```
fecha$  ↵   accion$  ↵
```

3. Introduzca además los dos campos siguientes:

   fecha$
   accion$

```
                ↵
```

4. Finalice el comando crear pulsando ↵ en una línea en blanco.

Adición de registros del fichero antiguo al nuevo

```
          editar  ↵
F3   n transferir  ↵
          limpiar  ↵
```

1. Escriba editar y pulse ↵.
2. Cree un nuevo procedimiento llamado transferir.
3. La primera línea, limpiar, borra la pantalla para que sea más fácil ver qué ocurre.

```
      todos "a"  ↵
```

4. Escriba todos "a" y pulse ↵.
   Al no haber ningún registro en el nuevo fichero, el comando todos se refiere al antiguo fichero "agenda".

```
haz n.notas$ = a.notas$  ↵
```

5. Escriba la línea anterior y pulse ↵.
   Este comando hace el campo notas$ del nuevo fichero igual al del antiguo.

```
      añadir "n"  ↵
```

6. Escriba añadir "n" y pulse ↵.
   Este comando inserta un registro en el nuevo archivo, con el valor de notas$ asignado en la línea anterior.
   Es parecido al uso del comando insertar para añadir registros. Entonces pulsábamos **F5** cuando queríamos añadir el registro al fichero. Añadir, sin embargo, se puede utilizar sin necesidad de esperar los datos del teclado.

```
fintodos  ↵
```

7. Finalice el bucle escribiendo fintodos.

El procedimiento debe quedar como sigue:

```
proc transferir
    limpiar
    todos "a"
        haz n.notas$ = a.notas$
        añadir "n"
        fintodos
    finproc
```

ESC **8.** Abandone el editor.

Observe que se ha utilizado explícitamente el nombre lógico a lo largo de todo el programa.

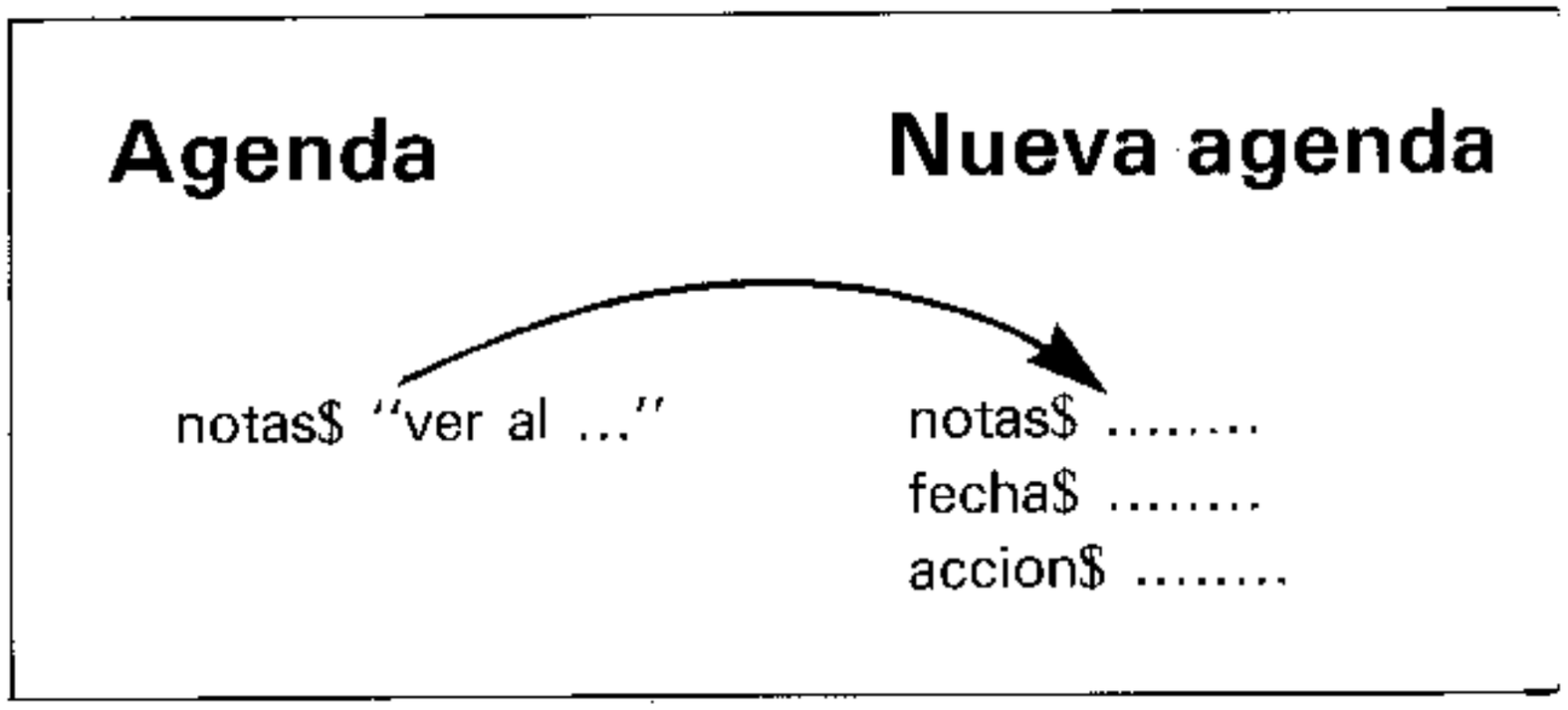


Figura 5.3. Adición de registros del viejo fichero al nuevo

El comando HAZ

El comando haz le da a la variable a la izquierda del signo de igualdad el mismo valor (es decir, contenido) de la variable de campo a la derecha. Puesto que ambas tienen el mismo nombre, se utiliza el nombre lógico seguido por un punto junto al nombre para distinguirlos.

La linea se podría parafrasear como "haz el campo notas$ del fichero "n" tener un valor igual al campo notas$ del fichero "a"". De una manera más general, "haz el primer campo igual al segundo".

Puliendo el procedimiento TRANSFERIR

El procedimiento funciona como está, pero no hay manera de saber lo que ha hecho hasta que acaba. Introduzca algunas sentencias escribir que le permitan observar cómo progresa la transferencia.

⇩ *(4 veces)* F4 escreg ↵ escribir n.notas$ ↵

1. Inserte las líneas siguientes en el procedimiento transferir, tras la línea añadir:

```
escreg
escribir n.notas$
```

Observe que no se utiliza ningún nombre lógico en escreg. No importa, ya que el registro enésimo del nuevo fichero se corresponde con el del antiguo fichero.

El procedimiento quedará como sigue:

```
proc transferir
    limpiar
    todos "a"
        haz n.notas$ = a.notas$
        añadir "n"
        escreg
        escribir n. notas$
        fintodos
    finproc
```

ESC

transferir ↵

2. Abandone editar.

3. Escriba transferir y pulse ↵.
La pantalla se borra, y las nuevas anotaciones aparecen escritas en la pantalla a medida que se añaden al fichero. El procedimiento termina cuando no hay ya más registros en el antiguo fichero agenda.

indicar ↵

4. Escriba indicar para ver la nueva estructura de fichero. La pantalla se borra y aparece el último registro.

El procedimiento de transferencia no tiene más utilidad en su forma actual, y los otros procedimientos en memoria ya se han salvado como "zoom_prg", así que cierre los ficheros de datos abiertos y borre la memoria mediante nuevo.

nuevo ↵

5. Escriba nuevo y pulse ↵.

Para cambiar de nombre un fichero

No existe una instrucción explícita para cambiar de nombre los archivos. Para hacerlo se debe copiar el fichero que queremos cambiar de nombre, dándole a la copia el nuevo nombre. A continuación se borra el fichero de nombre incorrecto.

tirar "agenda_dbf" ↵

1. Los poseedores de ARCHIVE versión 2.21 no pueden copiar un fichero sobre otro existente sin borrar primero el anterior. Esta línea no es necesaria para los poseedores de la versión 1.

salvagrdr "nueagen_dbf" como "agenda_dbf" ⏎

2. Escriba salvagrdr "nueagen_dbf" como "agenda_dbf" y pulse ⏎.
   Así guarda una copia del nuevo fichero con el nombre de "agenda".

F5 ⇧⇨ ⇨ CTRL⇨ *(3 veces)* ⇨ *(4 veces)* da ⇧⇨ ⇧⇨ ⇨ mdv1_ ⏎

3. Compruebe que el cartucho para copias de seguridad está en la unidad 1 y haga una copia del nuevo fichero "agenda" antes de borrar el original nueagen.

Borrado de un fichero con TIRAR

Antes de borrar ningún fichero haga un directorio del cartucho para ver qué contiene.

dir ⏎ ⏎

1. Escriba dir y pulse ⏎ dos veces. Aparecerá en pantalla una lista de los ficheros de la unidad 2.

tirar ⏎ nueagen_dbf ⏎

2. Escriba tirar y pulse ⏎, escriba entonces nueagen_dbf entre las comillas y pulse ⏎.

dir ⏎ ⏎

3. Escriba dir y pulse ⏎ dos veces. Podrá ver cómo ha desaparecido el fichero nueagen de la unidad 2.

El comando TIRAR

Sea muy cuidadoso cuando utilice el comando tirar, ya que no se puede recuperar un fichero una vez que ha sido borrado. Si no conserva una copia de seguridad y comete un error, puede ser un verdadero desastre.

El comando tirar necesita que se le dé el nombre completo, con extensión.

Nota: Si utiliza el nombre incompleto, o el nombre de un fichero que no existe, ARCHIVE no le indica que no ha borrado nada. Compruebe el directorio del cartucho si tiene dudas.

- ARCHIVE se negará a borrar un fichero de datos abierto.

El comando tirar devuelve el control a la línea de comandos muy rápidamente, aunque el *microdrive* continúa girando.

Cambio de todos los registros

Ahora que el nuevo fichero está a salvo, podemos añadir información a los campos nuevos, que están en blanco.

Eva planea una boda a finales de junio, pensando en rosales y fresas, y en los días largos y agradables. Maya recuerda su propia boda, en que llovió a mares.

¡Tanta gente, y tantas cosas que hacer! Decide poner la fecha de hoy, 5 de abril, en todas las notas.

En vez de introducirla a mano intenta que ARCHIVE ponga la fecha en los registros y también una "t" (de telefonear) como código de acción. Las excepciones que queden las puede cambiar a mano.

Un buen candidato para la tarea es el bucle todos, pero la manera especial en que funciona este comando hace que no se pueda usar cuando se modifica la longitud de los registros. Por ello hay que usar la secuencia primero, mientras no finf(), próximo, finmientras.

> *Recuerde:* No utilice el bucle todos si va a modificar el fichero con comandos como alterar, insertar, borrar, añadir o actualiz.

Actualización de registros

El comando actualiz funciona de una manera muy parecida a añadir, pero su efecto es cambiar un registro existente, como alterar. Se les dan nuevos valores a los campos con el comando haz, pero los valores del registro no se alteran hasta que se escribe el comando actualiz.

Cree un procedimiento llamado actagen utilizando la sentencia actualiz.

editar ↵ F3 n actagen ↵

1. Utilice el editor para crear un nuevo procedimiento llamado actagen.

primero ↵

2. Escriba primero y pulse ↵ para hacer que el bucle comience en el primer registro.

mientras no finf() ↵

3. Escriba mientras no finf() y pulse ↵.
   Este bucle hace a ARCHIVE ejecutar los siguientes comandos sobre todos los registros hasta que se acabe el fichero.

haz fecha$ = "04/05" ↵

4. Escriba haz fecha$ = "04/05" y pulse ↵.
   Así el campo fecha$ adquiere el valor "04/05" (la fecha del dia 5 de abril).
   Observe que la fecha se ha escrito al estilo de Estados Unidos, con el mes en primer lugar. Esto se debe a que ARCHIVE se puede utilizar para ordenar los registros por fecha si se introduce de esta manera.

Variables de campo y memoria temporal

Los comandos actualiz y añadir utilizan una característica de ARCHIVE: las variables de campo existen independientemente del registro que las contiene.

Cada vez que un registro pasa a ser el activo, el contenido de sus campos se copia en memoria bajo los nombres de campo adecuados. Así los comandos alterar e insertar le permiten introducir información sin afectar el registro hasta que se salvan las variables, por ejemplo, pulsando **F5.**

Los comandos actualiz y añadir actúan como la tecla **F5,** salvando los contenidos de las variables de campo al registro. Actualiz afecta al registro activo, mientras añadir crea un nuevo registro.

Se puede alterar el valor de las variables de campo usando sentencias como haz o leer.

Las variables de campo de todos los ficheros abiertos permanecen en memoria hasta que se cambian con comandos como haz, alterar o insertar, o hasta que se borran con cerrar o nuevo. Por eso la visualización sigue mostrando el último registro de un fichero cuando se han borrado todos.

Nota: La diagonal ("/" o simbolo de división) que hemos colocado entre el mes y el día es sólo para hacer la combinación más legible; no tiene ningún objetivo en ARCHIVE. Si utiliza algún símbolo para separar días, meses y años, asegúrese de usar siempre el mismo. En caso contrario, ARCHIVE no ordenará correctamente por fecha.

haz accion$ = "t" ↵
actualiz ↵

5. Ponga el valor del campo accion a t.
6. Utilice el comando actualiz para darle los valores a los campos fecha$ y accion$ del fichero.

próximo
finmientras

7. Escriba próximo y pulse ↵.
8. Acabe el bucle mientras escribiendo finmientras y ↵.

El procedimiento debe quedar como sigue:

```
proc actagen
    primero
    mientras no finf()
        haz fecha$ = "04/05"
        haz accion$ = "t"
        actualiz
        próximo
        finmientras
    finproc
```

↵ ESC

9. Pulse ↵ para abandonar el modo de edición, y pulse **ESC** para dejar el editor.

Nota: Sea cuidadoso si utiliza actualiz y añadir, ya que ambos reemplazan el contenido de todos los campos con el contenido de todas las variables de campo. Por ejemplo, si añade registros al final de un fichero existente y utiliza haz para asignar valores a parte de los campos, el resto de los campos en los registros añadidos tendrá el valor de los campos del registro activo en el momento en que comenzó a añadir. Actualiz también actualiza la información de todos los campos con los valores para el registro activo en memoria.

Funciones de hora y fecha

La hora y la fecha desempeñan un papel muy importante en muchos programas, desde diarios y planificaciones de proyectos a contabilidad y coste del tiempo.

ARCHIVE proporciona varias funciones muy útiles que le permitirán saber la fecha y la hora, o manipular fechas.

mes( ) devuelve, si se le proporciona un número, el nombre del mes correspondiente. Si el número es mayor de 12, ARCHIVE utiliza el resto de dividir el número por doce; por ejemplo, escribir mes(15) presentará en pantalla Marzo.

hora( ) devuelve la hora como una cadena alfanumérica que representa las horas, minutos y segundos. Si quiere usarla, debe poner en hora el reloj del QL mediante los comandos sdate o adate del SUPERBASIC.

fecha( ) devuelve la fecha del sistema del QL (puesta

mediante el comando sdate) en un formato a elegir entre tres.

Si, por ejemplo, ha puesto la fecha del sistema a cinco de abril de 1984, se puede representar como sigue:

```
escribir fecha(0) muestra 1984/04/05
escribir fecha(1) muestra 05/04/1984   (Como en España: el día en primer lugar)
escribir fecha(2) muestra 04/05/1984   (Como en Norteamérica: el mes primero)
```

días( ) se puede usar para calcular el número real de días pasados entre dos fechas. Puede ser muy útil en programas de contabilidad o financieros.

ARCHIVE y las fechas

Por lo que respecta a ARCHIVE, una fecha no es más que una serie de caracteres, y 04/05 (5 de abril) se clasifica antes de 05/01 (1 de mayo), como debe ser.

Si se almacenan las fechas con la notación de Estados Unidos, 05/01 (1 de mayo) viene después de 04/05 (5 de abril), como si fueran los números 501 y 405. Si se almacena con la notación española, 01/05 (1 de mayo) viene antes que 05/04 (5 de abril), mientras que 06/05 (6 de mayo) viene después, como si se tratara de los números 105, 504 y 605.

Adición de nuevos datos

```
abrir "agenda" lógico "a"  ↵  primero  ↵  actagen  ↵
```

1. Abra el fichero agenda, comenzando en el primer registro, y ejecute el procedimiento actagen.

```
indicar  ↵
```

2. Puesto que no hay sentencias pescribir o escribir en él, escriba indicar y pulse ↵ para saber si ha funcionado correctamente cuando los *microdrives* se detengan.

El fichero agenda debe quedar como sigue:

```
lógico    :  a
notas$    :  Ver al Sacerdote
fecha$    :  04/05
acción$   :  t
```

editar ↵ F3 b ↵ ESC

3. Borre ahora el procedimiento actagen, ya que no volverá a usarse.

Alteración de todos los registros de un fichero

Maya escribe ahora un conjunto de sentencias en un bucle que le permitirán alterar todos los registros del fichero. Elige un bucle mientras, ya que va a alterar el fichero.

alterar afecta sólo a un registro cada vez, lo que le va a permitir a Maya trabajar sobre cada registro tanto como desee, bastando finalizar alterar para pasar al siguiente.

Observe el uso de pescribir, ya que la presentación automática no funciona en procedimientos, y cada registro parece tener el contenido del anterior si no se fuerza una impresión de pantalla.

Escriba los comandos en una sola línea tras el indicador ">", separándolos mediante dos puntos ":". No pulse ↵ hasta estar seguro de haber acabado.

primero:mientras no finf():pescribir:alterar:próximo:finmientras() ↵

1. Escriba la línea anterior y pulse ↵.
   Si cometió un error al escribir la línea, y ARCHIVE responde con un mensaje de error, recuerde que puede editar la línea pulsando **F5.**
   No pulse **ESC** en alterar si no quiere que se detenga el bucle completo.
   Si utiliza la versión 1, pulse **F5** para pasar al registro siguiente.
   Si utiliza la versión 2, pulse **F4** para moverse sin conservar los cambios, o pulse **F5** o ↵ para salvar los cambios y pasar al registro siguiente.

> *Nota:* En alterar, si dispone de ARCHIVE versión 1, pulse **F5** para pasar al registro siguiente, haya hecho cambios en el registro o no.

2. Cambie el código de acción y la fecha de los siguientes campos:

```
notas$                                          fecha$ accion$
Ver al sacerdote                                04/08
Que no se olvide hacer los ojales               06/23
Zapatos blancos                                 04/14  c
Comprar el traje                                04/14  c
¿ Ha comprado Salva un traje de sport ?         04/05
Hacer una lista de bodas                        04/05
```

insertar ↵ No olvidar aguja e hilo ↵ 06/23 ↵ F5 ESC (*o* F4)

3. Añada la siguiente nota al fichero:
No olvidar aguja e hilo

```
notas$                                                fecha$ accion$
Ver al sacerdote                                      04/08
Pedirle a Felicidad que sea la madrina                04/05  t
Decirle a Juana que haga los trajes                   04/05  t
Buscar un Hotel para los invitados                    04/05  t
Decirle al Sr Hort que se encargue de las flores      04/05  t
Que no se olvide hacer los ojales                     06/23
Zapatos blancos                                       04/14  c
Comprar el traje                                      04/14  c
¿ Ha comprado Salva un traje de sport ?               04/05
Contratar a Pèrez Fotògrafos                          04/05  t
Coche                                                 04/05  t
Llamar a Sara ¿ Quièn les organizò el banquete ?      04/05  t
Capacidad de la sala Azul del Hotel Almirante         04/05  t
Encargar la tarta                                     04/05  t
Hacer una lista de bodas                              04/05
no olvidar aguja e hilo                               06/23
```

cerrar ↵

4. Cierre ahora el fichero agenda.

salvagrdr "agenda_dbf" como "mdv1_agenda_dbf" ↵

5. Haga una copia de seguridad del fichero agenda en la cinta de seguridad en la unidad 1.

RESUMEN

Maya decide escribir un procedimiento que le ayude a mirar los ficheros rápida y fácilmente. El procedimiento, llamado zoom, utiliza el comando pescribir para mostrar registros en la pantalla. Mediante un bucle se le indica a ARCHIVE que repita un conjunto de comandos, en este caso hasta que se llegue al final del archivo (usando el bucle mientras no finf()).

Maya escribe también un procedimiento que muestra el número de registro del registro activo en pantalla. El procedimiento se llama escreg, y utiliza la función recnum().

Maya decide hacer más útil el fichero agenda añadiéndole dos campos más. Para hacerlo, crea primero una nueva estructura de fichero con los nuevos campos incluidos (fecha$ y acción$), además del campo que ya existía (notas$). Escribe entonces un procedimiento para transferir todos los campos notas$ del fichero original a la copia. Actualiza el campo fecha$ en el nuevo fichero mediante un procedimiento, actagen, que usa las sentencias haz y actualiz.

Trucos útiles

- Asegúrese de que la palabra finproc está indentada dos espacios al finalizar un procedimiento que contenga un bucle todos o mientras. En caso contrario ha olvidado un fintodos o un finmientras.
- Inserte la línea finmientras inmediatamente tras su correspondiente mientras cuando escriba bucles complejos en sus procedimientos. Así no lo olvidará, y preservará la indentación correcta en las líneas siguientes. Inserte las líneas interiores del bucle subiendo con el cursor y pulsando **F4.**
- Realice copias de los procedimientos en memoria salvándolos a cinta, cambiando los nombres de los procedimientos y uniendo de nuevo el fichero salvado.
- Limpie los cartuchos periódicamente borrando los ficheros viejos e innecesarios. En caso contrario el directorio de la cinta acabará desorganizándose y le costará trabajo localizar sus ficheros. Además, los *microdrives* funcionan peor si se llenan.
- Use una libreta para conservar detalles sobre el cometido de cada fichero. Se le olvidará muy deprisa.
- Para desconectar el comando trazar cuando se está ejecutando un procedimiento, pulse **F3.**

6

Impresión y listado de la información

Algunas tareas, como anotar listas, se realizan mejor sobre papel que en el ordenador. Los listados en papel proporcionan también una seguridad "a prueba de bomba" contra lo "imposible": la pérdida de todos los datos en cinta.

Este capítulo explica la impresión de listados y etiquetas en impresora y pantalla.

Presentamos las siguientes características de ARCHIVE:

- Sentencias de ARCHIVE:

imprimir	— envía la información a la impresora
papel	— pone el color de fondo de la pantalla
tinta	— pone el color del texto en pantalla

- Funciones de ARCHIVE:

numcam()	— número de campos de un registro
valcam()	— contenido del campo enumerado
val()	— convierte una cadena en un valor numérico

Puntos a recordar

- No utilice los comandos que envían información a la impresora si no la tiene conectada. El ordenador se bloqueará o "colgará" si lo hace.
- Si quiere usar una impresora, asegúrese de que está "adaptada" correctamente antes de usarla. Si no lo está, el ordenador podría bloquearse; en el mejor caso, la impresora escribirá cosas sin sentido.
- Un uso incorrecto de los puntos y comas en los comandos escribir o imprimir es una de las causas más corrientes de salida con formato incorrecto en pantalla o impresora.

Ejemplo: Impresión de listados

Un montón de bisutería junto al ordenador alerta a Eva: Maya ha estado con uno de sus trucos. Para comprobar qué ha hecho su madre con el fichero agenda, lo lista todo por pantalla.

UN PROCEDIMIENTO PARA LISTAR EL FICHERO AGENDA

Instrucciones

Si ARCHIVE no está cargado, cárguelo ahora (véase el capítulo 1), y ponga el cartucho de datos que contenga los ficheros invitado y agenda en la unidad 2. Este cartucho contiene también los programas zoom, fácil y FichBoda.

abrir "agenda" lógico "a" ↵

1. Abra el fichero agenda con el nombre lógico a.

cargar "zoom" ↵

2. Escriba cargar "zoom" y pulse ↵ para cargar los procedimientos del programa zoom.

editar ↵

3. Escriba editar y pulse ↵.

TAB F4 primero ↵

4. Pulse **TAB** y **F4** para insertar la línea primero al principio del procedimiento listagenda.

↓ *(3 veces)* F5 ↓;" ";fecha$;" ";acción$ ↵

5. Añada los nuevos campos escribiendo ;" ";fecha$;" "; acción$ al final de la línea escribir notas$.
   Observe que el comando escribir va seguido por una lista de elementos, cada uno separado por un punto y coma.

Los tres campos se separan por un espacio. Sin él los campos aparecerían unidos en una cadena de caracteres larga y sin sentido.

El procedimiento debe quedar como sigue:

```
proc listagenda
    primero
    limpiar
    mientras no finf()
        escribir notas$;" ";fecha$;" ";acción$
        próximo
        finmientras
    finproc
```

ESC

listagenda ↵

6. Pulse **ESC** para dejar de editar.
7. Escriba listagenda y pulse ↵.
   Aparece listado el fichero línea por línea en la pantalla, desapareciendo por la parte superior de la pantalla tan pronto como la lista llega a la línea inferior:

```
Ver al sacerdote 04/08
Pedirle a Felicidad que sea la madrina 04/05 t
Decirle a Juana que haga los trajes 04/05 t
Buscar un Hotel para los invitados 04/05 t
Decirle al Sr Hort que se encargue de las flores 04/05 t
Que no se olvide hacer los ojales 06/23
Zapatos blancos 04/14 c
Comprar el traje 04/14 c
¿ Ha comprado Salva un traje de sport ? 04/05
Contratar a Pérez Fotógrafos 04/05 t
Coche 04/05 t
Llamar a Sara ¿ Quién les organizó el banquete ? 04/05 t
Capacidad de la sala Azul del Hotel Almirante 04/05 t
Encargar la tarta 04/05 t
Hacer una lista de bodas 04/05
no olvidar aguja e hilo 06/23
```

La linea inferior está vacía, y el cursor queda en la primera posición de la nueva línea.

Retornos de carro

Para introducir más información en la pantalla, Eva cambia el procedimiento para que la visualización no cambie de línea cada vez que lista un nuevo registro.

editar ↵

TAB ⇩ *(4 veces)* F5 ⇩ ; ↵

1. Edite el procedimiento listagenda.
2. Añada un punto y coma al final de la línea escribir, de manera que la línea quede:

```
escribir notas$;" ";fecha$;" "acción$;
```

ESC

3. Pulse **ESC** para dejar el editor.

El procedimiento debe quedar como sigue:

```
proc listagenda
    primero
    limpiar
    mientras no finf()
        escribir notas$;" ";fecha$;" ";acción$;
        próximo
        finmientras
finproc
```

listagenda ↵

4. Escriba listagenda y pulse ↵.
Esta vez los registros se escriben uno tras otro, pasando de línea cuando queda llena la anterior, y así hasta el final:

```
Ver al sacerdote 04/08 Pedirle a Felicidad que sea la madrina 04/05 tDecirle a J
uana que haga los trajes 04/05 tBuscar un Hotel para los invitados 04/05 tDecirl
e al Sr Hort que se encargue de las flores 04/05 tQue no se olvide hacer los oja
les 06/23 Zapatos blancos 04/14 cComprar el traje 04/14 cé Ha comprado Salva un
traje de sport ? 04/05 Contratar a Pèrez Fotògrafos 04/05 tCoche 04/05 tLlamar a
 Sara ¿ Quièn les organizò el banquete ? 04/05 tCapacidad de la sala Azul del Ho
tel Almirante 04/05 tEncargar la tarta 04/05 tHacer una lista de bodas 04/05 no
olvidar aguja e hilo 06/23
```

El cursor queda posicionado inmediatamente detrás del último carácter impreso. Ese es el efecto de poner un punto y coma al final de una sentencia escribir: no se comienza una nueva línea tras cada comando.

Retornos de carro

El retorno de carro recibió su nombre en honor de las viejas máquinas de escribir mecánicas, que devolvían el carro de papel físicamente a la posición correspondiente a

una nueva línea cada vez que se pulsaba la barra de retorno. En las máquinas de escribir eléctricas esto se hace pulsando una gran tecla con forma de "J", muy parecida a la tecla marcada ↵ en el QL.

La tecla ↵ se llama frecuentemente tecla **INTRO (ENTER),** y también **RETURN** (o **RETORNO)** por esta razón. Tiene aún el efecto de finalizar la introducción de texto en la mayor parte de los programas.

Un "retorno de carro" se compone realmente de dos acciones distintas: por un lado se pasa el papel (en las impresoras o máquinas de escribir) hacia arriba una línea. Esta acción se llama paso de linea (**LF,** del inglés *Line Feed*). La otra acción devuelve la cabeza de impresión o el carro al margen izquierdo, y se conoce como retorno de carro o **CR** *(Carriage Return).*

La expresión "retorno de carro" se utiliza con frecuencia en sentido laxo para describir el efecto de ambas acciones. La combinación de ambas comienza una nueva línea. El comando escribir envía un **LF** y un **CR** juntos como un "retorno de carro", a menos que se suprima finalizando la sentencia con un punto y coma.

Puntos y comas finales y el comando ESCRIBIR

Un comando escribir sin un punto y coma al final comienza una nueva línea al terminar. Envía una señal a la pantalla para que mueva el cursor una línea abajo y lo devuelva al margen izquierdo. Esto se llama un "retorno de carro".

Un comando escribir sin argumentos envía un retorno de carro a la pantalla. Tiene como resultado la producción de una línea en blanco.

El retorno de carro se suprime si la sentencia de escritura termina en punto y coma. La línea de comandos escribir; no tiene, pues, ningún efecto.

La escritura comienza siempre en la posición del cursor.

Tabulación

Eva se da cuenta de que su última idea no fue exactamente inspirada, y se dispone a rectificar su error y a mejorar la presentación de los listados.

El procedimiento listagenda tal y como está produce un desastre. Incluso sin el punto y coma el listado no queda muy claro, ya que las fechas y códigos no están alineados, y no resultan

fáciles de distinguir, y el código acción$ se mezcla con la siguiente notas$.

editar ⏎ TAB

1. Edite el procedimiento listagenda.
2. Cambie la línea **escribir** sustituyendo los espacios con instrucciones de tabulación, de manera que quede:

```
escribir notas$; tab 55; fecha$; tab 61; acción$
```

⏎

3. Pulse ⏎ para salvar la nueva línea.

El procedimiento debe quedar como sigue:

```
proc listagenda
    primero
    limpiar
    mientas no finf( )
        escribir notas$; tab 55; fecha$; tab 61; acción$
        próximo
        finmientras
finproc
```

ESC

4. Pulse **ESC** para dejar de editar.

listagenda ⏎

5. Escriba listagenda y pulse ⏎.
Debe aparecer lo siguiente en pantalla, con la fecha y el código correctamente alineados en columnas:

```
Ver al sacerdote                                          04/08
Pedirle a Felicidad que sea la madrina                    04/05 t
Decirle a Juana que haga los trajes                       04/05 t
Buscar un Hotel para los invitados                        04/05 t
Decirle al Sr Hort que se encargue de las flores          04/05 t
Que no se olvide hacer los ojales                         06/23
Zapatos blancos                                           04/14 c
Comprar el traje                                          04/14 c
¿ Ha comprado Salva un traje de sport ?                   04/05
Contratar a Pèrez Fotógrafos                              04/05 t
Coche                                                     04/05 t
Llamar a Sara ¿ Quièn les organizò el banquete ?          04/05 t
Capacidad de la sala Azul del Hotel Almirante             04/05 t
Encargar la tarta                                         04/05 t
Hacer una lista de bodas                                  04/05
no olvidar aguja e hilo                                   06/23
```

Tabulaciones

La tabulación es, en un ordenador o una máquina de escribir, la acción de hacer que el cursor o cabeza de impresión salte a una posición determinada antes de imprimir más caracteres.

La palabra viene de la forma latina de tabla *(tabula)*, ya que a las columnas de cifras alineadas se las conoce por tablas.

La palabra clave de ARCHIVE tab no es un comando. Se llama un elemento de impresión, y se puede usar sólo junto a los comandos escribir, imprimir y leer.

Los tabuladores saltan a una posición absoluta de la pantalla, y no una relativa al último carácter impreso. Por ejemplo, escribir nombre$; tab 50 se mueve a la columna 50, y no a una columna 50 espacios más allá del contenido de nombre$.

Si la columna especificada está ya más allá del cursor o cabeza de impresión cuando se escribe el tabulador, no tiene ningún efecto.

Por ejemplo, la sentencia escribir "La fecha de hoy:"; tab 5; fecha$ daría como resultado La fecha de hoy:04/03. Tras escribir el primer elemento de impresión, el cursor ha pasado ya la columna 5.

tab debe ir seguido por un número no mayor que la anchura de pantalla en el momento de usarlo.

Observe que los elementos tab dejan la posición de impresión en la columna especificada de la pantalla, y que posicionan el comienzo del siguiente elemento de impresión en esa columna.

Nota: Las cifras usadas (55 y 61) suponen que trabaja con una visualización de 64 u 80 columnas.

Si alguna de las notas tiene más de 64 caracteres, sigue imprimiéndose en la línea siguiente. La fecha y el código, sin embargo, se escriben todavía en la posición correcta, ya que los números de columna son mayores que la columna del cursor en el momento en que se emite el tab.

ADICION DE COLOR. SALVA LO PONE BONITO

Los comandos PAPEL y TINTA

Salva se queda alelado con las tabulaciones de Eva, pero está mucho menos impresionado con lo que llama "la brutal tiranía mecánica" de las letras blancas sobre una pantalla negra.

Se propone ejercitar su juicio estético sobre el esquema de colores de ARCHIVE usando los comandos papel y tinta.

Pruebe estos comandos, aunque tenga una televisión en blanco y negro, ya que afectan al tono de la visualización, además del color.

papel 4 ↵

1. Escriba papel 4 y pulse ↵.
De momento no ocurre nada. El comando le indica a ARCHIVE que utilice el color 4 (el verde) como color de fondo de los siguientes comandos escribir.

listagenda ↵

2. Escriba listagenda y pulse ↵.
El AREA DE VISUALIZACION se vuelve instantáneamente de color verde y se listan las notas en color blanco. El resultado es bastante ilegible, así que Salva decide cambiar el color de tinta.

tinta 0 ↵

3. Escriba tinta 0 y pulse ↵.
Este comando le dice a ARCHIVE que escriba los caracteres siguientes en el color 0 (negro).

listagenda ↵

4. Escriba listagenda y pulse ↵.
Aparecen escritas las notas en las letras negras sobre fondo verde.

Eva duda si "arañas sobre un tapete" es una mejora sobre la "brutal tiranía mecánica", pero Salva ya ha tenido una idea mejor. ¡Cuán glorioso sería si cada nota se imprimiera de un color distinto, según el mes!

papel 0: tinta 6 ↵

5. Escriba papel 0: tinta 6 y pulse ↵ para devolver sus colores normales a la pantalla: blanco sobre negro.

Los comandos PAPEL y TINTA

Las palabras clave papel y tinta son poco usuales, ya que se utilizan como comandos por sí mismos, y como elementos de impresión junto al comando escribir.

El comando papel cambia el color de fondo para el texto que se escriba a continuación al color representado por el valor del número siguiente.

El comando tinta cambia el color de los caracteres escritos para el texto subsiguiente al color especificado por el valor del número siguiente.

Colores:	0 y 1	negro
	2 y 3	rojo
	4 y 5	verde
	6 y 7	blanco

No se puede utilizar un número mayor que 7 para cambiar de color, excepto en la versión 1, en cuyo caso el programa divide el número por 8 y usa el resto de la división.

Por ejemplo, papel 2 pone el papel de color rojo, y papel 16 en la versión 1:

16/8 = 2 resto 0 = 0 = negro

Cambia, por tanto, el color a negro; en la versión 2 da el error "Elemento de impresión no reconocido".

Este tipo de aritmética se llama aritmética modular. "Módulo 8" significa que la parte significativa de un número es el resto después de dividirlo por 8.

No utilice el mismo color para tinta y papel, ya que el resultado será una pantalla aparentemente vacía.

Nota: El comando papel tiene un efecto ligeramente distinto en función de donde comience el siguiente comando escribir.

Si el cursor está en la parte inferior de la pantalla, todos los comandos escribir siguientes harán que la línea entera tenga el nuevo color de fondo.

Si el cursor está en cualquier otro lugar, sólo la anchura del texto impreso se verá afectada del nuevo color.

Para evitarlo, use el comando limpiar para poner inmediatamente toda la pantalla al color de papel.

Elementos de impresión

Salva planea utilizar el número del mes para determinar el color de la tinta.

Puede hacerlo incorporando el comando tinta como un elemento de impresión en las sentencias escribir. A menos que se

use tinta como un comando, los colores cambiados en una sentencia de impresión sólo actúan sobre los siguientes elementos de impresión de la misma sentencia. Tan pronto como el comando acaba, se restauran los colores anteriormente activos.

Por ejemplo, los siguientes cuatro comandos escriben primero un nombre en rojo (color 2) y luego otro en blanco (color 6):

```
papel 0
tinta 6
escribir tinta 2; "Alfredo"
escribir "Jaime"
```

Todo lo que Salva tiene que hacer ahora es averiguar cómo hacer cambiar el color de cada línea en función del mes.

Elementos de impresión

Los comandos escribir se componen de la palabra clave escribir, seguida por una lista de elementos separados por puntos y comas. Estos elementos pueden incluir:

- tab
- en
- papel
- tinta
- información que será impresa, como cadenas de caracteres o expresiones numéricas.

Estos elementos se pueden usar también en las sentencias leer.

Un solo comando leer puede llevar varias cadenas de caracteres y variables. Por ejemplo, el siguiente comando lee los valores de las variables fecha$ y hora$ una tras otra:

```
leer "¿Fecha? ";fechas$;" ¿Hora? "; hora$
```

El comando imprimir admite tab, pero no los otros elementos de impresión.

Uso de VAL( ) para obtener números de las cadenas

El mayor problema de Salva es que el comando tinta tiene que ir seguido por un número o valor numérico, y los meses es-

tán almacenados como una cadena de caracteres en el campo fecha$.

Decide usar la función val( ) para convertir el mes en un valor numérico.

Si una cadena alfanumérica comienza con números, la función val( ) devuelve el valor numérico de ese número.

Por ejemplo, el comando escribir val("06/23") presentará en pantalla el número 6.

Observe que val( ) convierte sólo el primer número de la cadena, ignorando el / y caracteres siguientes, sean o no numéricos.

La función VAL( )

La función val( ) convierte cadenas de caracteres en un único valor numérico.

- Devuelve el valor 0 si la cadena no comienza por números:

  val("hola") devuelve el valor 0

- Devuelve el valor de los números de la cadena si ésta comienza por un número:

  val("123") devuelve el valor 123
  vall("123hola") devuelve el valor 123

- Acepta espacios por la izquierda, el punto decimal y el signo menos, así como las cifras 0 a 9, como caracteres numéricos válidos:

  val(" −23.45") devuelve el valor −23.45

- Tan pronto como val( ) llega a un carácter no numérico, lo ignora, así como los caracteres restantes:

  val("352 + 345") devuelve el valor 352

El segundo problema de Salva es que los valores de los meses de abril a junio son 4, 5 y 6, separados por uno, mientras los distintos colores están separados por dos.

Como están, junio producirá letras blancas, pero abril y mayo producirán ambos letras verdes.

La solución es multiplicar el número del mes por 2. Los valores resultantes serán 8, 10 y 12, lo que plantea un nuevo proble-

ma, ya que la versión 2 no los acepta como colores válidos. Podríamos restarle 8, pero aparecería un nuevo problema, ya que 8 — 8 = 0 (negro), y el mes de abril no se distinguiría del fondo. Por tanto, decidimos restarle 6.

editar ↵ TAB ⇩ *(4 veces)* F5 ⇧⇨

1. Edite el procedimiento listagenda.
/tinta val(fecha$) * 2 – 6; ↵

2. Cambie la línea escribir insertando el elemento de impresión tinta val(fecha$) * 2 – 6; tras la palabra escribir, de forma que la línea quede:

escribir tinta val(fecha$) * 2 — 6; notas$; tab 55; fecha$; tab 61; acción$

ESC

3. Pulse **ESC** para dejar el editor.

El procedimiento debe quedar como sigue:

```
proc listagenda
  primero
  limpiar
  mientras no finf()
    escribir tinta val(fecha$)*2-6;notas$; tab 55;fecha$; tab 61;accion$
    próximo
    finmientras
  finproc
```

listagenda ↵

4. Escriba listagenda y pulse ↵.
Todas las notas se listan en el AREA DE VISUALIZACION, las de abril en rojo, y las de junio de color blanco.

A Salva le gustaría que hubiese alguna nota de mayo, para añadir algo de verde a una pantalla monótonamente roja.

salvar "zoom" ↵

5. Escriba salvar "zoom".
Así salvará los tres procedimientos: escreg, listagenda y zoom a cinta.

LISTADO EN IMPRESORA

El comando IMPRIMIR

Eva quiere estudiar una copia en papel de la lista, de manera que toda la familia pueda observarla y sugerir comentarios.

editar ↵ TAB
F5 CTRL ⇨ *(4 veces)* impr ↵

1. Edite el procedimiento listagenda.

2. Cambie el nombre del procedimiento listagenda a impragenda.

↓ ⇩ F3 q ↵
↓ ⇩ F5 CTRL ⇨ *(6 veces)* imprim ↵

3. Borre la línea limpiar.

4. Cambie el comando escribir por imprimir.

El comando imprimir es casi exactamente lo mismo que escribir, excepto que envía su salida a la impresora en lugar de la pantalla.

> *Nota:* No utilice imprimir o ningún otro comando que envíe información a la impresora si no tiene una conectada. El QL se "colgará" o bloqueará, ya que no será capaz de ejecutar la orden de ARCHIVE, y tendrá que reinicializar la máquina (véase *"En línea"*, más abajo).

↑ ⇨ CTRL ⇨ *(24 veces)* ↵

5. Cambie algo más la línea de impresión eliminando el elemento de impresión tinta, ya que los elementos de impresión de color no se pueden utilizar con la impresora. La línea debe quedar así:

```
imprimir notas$; tab 55; fecha$; tab 61; acción$
```

↵ **6.** Pulse ↵ para salvar la nueva línea.

El procedimiento quedará así:

```
proc impragenda
    primero
    mientras no finf( )
        imprimir notas$;tab 55;fecha$; tab 61;acción$
        próximo
        finmientras
    finproc
```

ESC

unir "zoom" ↵

7. Pulse **ESC** para salir del editor.
8. Una el programa zoom para restaurar listagenda junto a impragenda.

salvar ↵ listados ↵

9. Salve los procedimientos a un fichero llamado listados.

salvar ↵ mdv1_listados ↵

10. Salve los procedimientos a la unidad 1 también, tras asegurarse de colocar la cinta de seguridad en la unidad 1.

El programa listados contiene los siguientes procedimientos:

```
proc escreg
  escribir "Número de registro:";recnum();" "
  finproc
proc impragenda
  primero
  mientras no finf()
    imprimir notas$; tab 55;fecha$; tab 61;accion$
    próximo
    finmientras
  finproc
proc listagenda
  primero
  limpiar
  mientras no finf()
    escribir tinta val(fecha$)*2-6;notas$; tab 55;fecha$; tab 61;accion$
    próximo
    finmientras
  finproc
proc zoom
  mientras no finf()
    pescribir
    escreg
    próximo
    finmientras
  finproc
```

En línea

Asegúrese de que la impresora está preparada antes de utilizar un comando de impresora. Debe estar enchufada, conectada al QL, y *en línea*.

Se dice que una impresora está en línea cuando está lista para recibir información e instrucciones del ordenador, más que de sus propios interruptores de control. En muchas impresoras esto se indica por una luz marcada SEL u ON LINE. Si la impresora no está preparada, el QL "se colgará" hasta que lo esté.

Además, el fichero printer_dat debe estar en el cartucho de ARCHIVE, y debe contener información precisa sobre su impresora. Así, ARCHIVE puede saber qué señales particulares necesita su impresora para funcionar correctamente (si necesita alguna).

Si su impresora imprime *basura* cuando se le envían datos desde ARCHIVE, este fichero necesita probablemente una corrección. El proceso de asegurarse de que el programa sabe las peculiaridades de su impresora se llama, en inglés, *printer installation,* y se describe en el apéndice del manual de ARCHIVE, y en los libros de esta misma colección QL QUILL, QL EASEL y QL ABACUS.

Necesitará el programa SUPERBASIC y los ficheros de datos install_bas e install_ dat para poder preparar una impresora.

Si no están en el cartucho de ARCHIVE, hágalo mediante los cartuchos de QUILL, EASEL o ABACUS, que sí lo tienen.

Copie el fichero printer_dat resultante en el cartucho de ARCHIVE mediante salvagrdr.

Recuerde que debe eliminar la protección de escritura del cartucho eliminando la lengüeta si es preciso.

El fichero printer_dat

El programa de preparación almacena información que usted da sobre su impresora en un fichero llamado printer_dat en el cartucho de programa. Esta información se carga para uso de ARCHIVE al cargarse ARCHIVE.

El fichero se debe llamar printer_dat; si no, ARCHIVE no lo reconocerá. No existe una manera fácil de decir para qué impresora sirve.

Solucione este problema creando un fichero vacío que sirva de etiqueta (como los ficheros _eti que usó en las copias de seguridad), y sálvelo en el cartucho de programa. Si tiene una impresora Epson MX-80, por ejemplo, podría llamarse "mx80_imp". Puede usar el comando dir para saber qué fichero printer_dat está usando.

Si alguna vez necesita enviar salida de impresora a más de una impresora, o para más de un tipo de papel, o incluso a otro ordenador, deberá preparar el fichero printer_dat de nuevo.

Evite tener que repetirlo continuamente teniendo varios ficheros printer_dat, uno para cada necesidad. Almacénelos con diferentes nombres (por ejemplo, mx80_dat o apricot_dat) en el cartucho de datos de la unidad 2.

Cuando necesite cambiar de impresora, copie el fichero _dat relevante al cartucho de programa, por ejemplo, salvagrdr "mx80_dat" como "mdv1_printer_dat".

No olvide volver a etiquetar el cartucho borrando el viejo fichero _pri y salvando uno nuevo, de manera que sepa en el futuro para qué impresora está preparado printer_dat.

Impresión de las anotaciones

impragenda ↵

1. Escriba impragenda y pulse ↵.
La impresora lista todas las notas del fichero agenda, línea por línea, y se detiene tan pronto como acaba con la última.

Salto de página

Saque el papel girando el carro a mano, o bien presionando el botón de *salto de página,* marcado normalmente FF o TOF. Probablemente funcionará sólo si la impresora está "fuera de linea", es decir, preparada para recibir señales de sus interruptores, y no del ordenador.

Un salto de página "alimenta" el papel hasta el punto en que la impresora cree que está el principio de la página siguiente.

Sería muy útil si el ordenador pudiera enviar la instrucción para saltar la página directamente a la impresora, evitando asi nuestra intervención manual.

Use editar para crear el procedimiento siguiente:

```
proc ff
    imprimir car(12);
finproc
```

La función CAR( )

La función car( ) se utiliza aquí para enviar un código a la impresora, que lo interpreta como una señal para pasar de página. Observe el punto y coma del final, que suprime el retorno de carro que, en caso contrario, se enviaría con el carácter.

El mismo código exactamente se puede enviar a la pantalla de TV o monitor, usando el comando escribir y la función car( ). Borra la pantalla, como si hubiera usado el comando limpiar. Es el equivalente, en la pantalla, de comenzar una nueva página.

El número entre corchetes se llama un código ASCII. No es

necesario que conozca esos códigos o que entienda cómo funcionan para utilizar los dos o tres que necesitará ocasionalmente. Cuando aparezcan, estarán por regla general en sentencias escribir, y siempre usando la función car( ), como aquí. Para más información sobre el ASCII y car( ) véase el apéndice al final del libro.

ESC

1. Pulse **ESC** para salir del modo de inserción.

TAB

2. Pulse **TAB** hasta llegar al procedimiento impragenda.

↓ *(5 veces)* F4 ff ⇦

3. Inserte una línea ff tras la línea finmientras para ir al comienzo de la página siguiente.

ESC

4. Pulse **ESC** para dejar el modo de inserción.

TAB *(2 veces)*

5. Pulse **TAB** para llegar a zoom.
   Este procedimiento está ya en el fichero de programa zoom, y no se necesita en listados.

F3 b ↵

6. Borre el procedimiento zoom usando **F3** seguido de b y luego ↵.

TAB *(3 veces)* F3 b ↵

7. Borre también el procedimiento escreg.

Los procedimientos restantes quedan como sigue:

```
proc ff
  imprimir car(12);
  finproc
proc impragenda
  primero
  mientras no finf()
    imprimir notas$; tab 55;fecha$; tab 61;accion$
    próximo
    finmientras
  ff
  finproc
proc listagenda
  primero
  limpiar
  mientras no finf()
    escribir tinta val(fecha$)*2-6;notas$; tab 55;fecha$; tab 61;accion$
    próximo
    finmientras
  finproc
```

ESC salvar ↵ listad2 ↵ salvar ↵ mdv1_listad2 ↵

8. Abandone el editor y salve los procedimientos a listad2, y también a mdv1_listad2.

LISTADO DE DIRECCIONES

Cambio del programa para listar el fichero de invitados

El fichero invitado se puede listar e imprimir de la misma manera exactamente, ya que la única diferencia es que este fichero tiene más campos que agenda.

editar ↵ TAB *(2 veces)*

1. Utilice el editor y pulse **TAB** hasta editar el procedimiento listagenda.

F5 ⇧⇨ CTRL⇦ *(6 veces)* invit

2. Cambie el nombre del procedimiento de listagenda a listinvit.

⇦ ⇩ *(4 veces)* F5 CTRL⇩ escribir nombre$ ↵ F4 escribir direc1$ ↵ escribir direc2$ ↵ escribir direc3$ ↵ escribir direc4$ ↵

3. Sustituya la línea de impresión por las siguientes líneas:

```
escribir nombre$
escribir direc1$
escribir direc2$
escribir direc3$
escribir direc4$
```

escribir ↵

4. Introduzca escribir y pulse ↵. Así escribe una línea en blanco entre las direcciones, haciéndolas más fáciles de leer.

↵

5. Pulse ↵ para dejar el modo de inserción.

El procedimiento debe quedar como sigue:

```
proc listinvit
   primero
   limpiar
   mientras no finf( )
      escribir nombre$
      escribir direc1$
      escribir direc2$
      escribir direc3$
      escribir direc4$
      escribir
      próximo
      finmientras
   finproc
```

ESC

6. Abandone el editor.

unir ↵ listad2 ↵

7. Una el fichero de programa listad2 para recuperar el procedimiento listagenda.

salvar ↵ listad3 ↵

8. Salve los procedimientos como listad3; el fichero contendrá los procecimientos siguientes:

```
impragenda
ff
listagenda
listinvit
```

abrir ↵ invitado ⇨ lógico "i" ↵

9. Abra el fichero de invitados con el nombre lógico i.

listinvit ↵

10. Escriba listinvit y pulse ↵.
Los campos seleccionados de cada registro se escriben en la pantalla. Algunos están más separados que otros, ya que la cuarta línea de direcciones está vacía a veces.

Juan y Pepa Martínez Rebollo
Majada del Viento, 25
28029 Madrid
MADRID

Luisa Sala y Carmen García Carrión
Gran Vía, 34
Orihuela
ALICANTE

... y así sucesivamente.

Paso de parámetros

Sería tedioso tener que cambiar los comandos escribir por imprimir (o al revés) cada vez que un campo de un registro necesite imprimirse o listarse.

Eva escribe un procedimiento que contiene el comando, de manera que sólo sea necesario un cambio.

editar ↵ F3 n línea; l$ ↵ escribir l$ ↵ ESC

1. Cree el procedimiento siguiente:

```
pro línea; l$
  escribir l$
  finproc
```

ESC

2. Abandone el editor.
   Observe el punto y coma seguido por una variable tras el nombre del procedimiento. Actúa como el comando leer que utilizamos en los procedimientos de copia en el capítulo 4. Tiene un valor diferente (en este caso, una cadena de caracteres) cada vez que se utiliza el procedimiento, y lo almacena como una variable para su uso en el procedimiento.

Sin embargo, a diferencia de leer, ARCHIVE no espera a que escriba nada a continuación.

La información se le pasa al procedimiento cuando se usa como un comando. A la información pasada se le llama un *parámetro*.

Observe que, como está, el procedimiento línea sólo funciona si se le pasa una expresión alfanumérica, ya que el nombre de variable I viene seguido por un signo dólar.

Uso de parámetros

línea ⊡

1. Escriba línea y pulse ↵.
   ARCHIVE muestra la primera línea del procedimiento y un mensaje de error: Error 2: esperado fin de instrucción.

Si un procedimiento espera que se le pase un valor, debe proporcionárselo. ARCHIVE cree que el punto y coma y el nombre de variable no deberían estar ahí (y que la línea proc debería haber acabado), ya que usted no le pasó una cadena de caracteres como parámetro.

línea; nombre$ ⊡

2. Escriba línea; nombre$ y pulse ↵.
   El campo nombre$ del registro activo se muestra en pantalla:

Srta. Felisa de los Monteros

Parámetros

Los elementos de información que se le pasan a los procedimientos se llaman *parámetros*. Le proporcionan al procedimiento información que necesita para cumplir su misión. Hacen posible que un solo procedimiento se utilice de varias maneras por varios programas y con distintos da-

tos. Esto ahorra tener que escribir procedimientos nuevos continuamente, y ahorra espacio en los programas.

El número de parámetros pasados a un procedimiento se determina en la definición, donde siguen al nombre del procedimiento. Pueden ser variables de cadena o numéricas, y se separan por comas.

Los datos pasados a un procedimiento deben encajar en cantidad y tipo (cadena o numérico) con los parámetros esperados.

Por ejemplo, un procedimiento que comience como sigue:

```
proc ejemplo; car$,num,text$
```

se podría usar como sigue:

```
ejemplo; "Conejo",6,alimento$
```

Todas las variables definidas como parámetros en una línea proc se borran de memoria tan pronto como acaba el procedimiento. Se llaman variables *locales*, ya que son "locales" a ese procedimiento.

Es importante distinguir entre el valor pasado como parámetro y la variable que se usa para almacenarlo. En la jerga de la programación, la variable en el procedimiento que usa el parámetro se llama parámetro "formal" (es parte de la forma del procedimiento), y el dato real que se le pasa se llama parámetro "real".

nea; "Nombre " + nombre$

3. Escriba línea; "Nombre " + nombre$ y pulse ↵.

Observe que las cadenas están unidas y no separadas por punto y coma.

Si un procedimiento espera un solo parámetro y le intenta pasar más, resultará un mensaje de error.

Las dos cadenas se imprimen en pantalla:

```
Nombre Srta. Felisa de los Monteros
```

editar ↵ TAB TAB ⇩ *(4 veces)* F5 ⇧⇨ CTRL⇧ línea; ↵ [⇩ F5 ⇧⇨ CTRL⇧ línea; ↵] *(5 veces)* ⇩ F5 CTRL⇩ línea; " " ␣ ESC

4. Use ahora el editor para cambiar el procedimiento listinvit hasta que quede como sigue:

```
proc listinvit
   primero
limpiar
mientras no finf( )
   línea; nombre$
   línea; direc1$
   línea; direc2$
   línea; direc3$
   línea; direc4$
   línea; ""
   próximo
   finmientras
finproc
```

Observe que los dos pares de comillas en línea; "" están juntos, con nada entre ellos. Esto se llama una cadena vacía o, en jerga todavía más abstracta, una cadena nula.

Se utiliza aquí para enviar algo hacia el procedimiento **línea**, a la que se le debe pasar un parámetro, pero en este caso lo único que se desea imprimir es un retorno de carro.

La cadena nula

Una cadena nula, "", es una cadena de texto que no contiene nada.

Una cadena nula es muy distinta de, por ejemplo, la cadena " ", que no contiene nada cuando la imprimimos, pero, para ARCHIVE, contiene un espacio, que es un carácter por derecho propio.

La cadena nula tiene un valor ASCII de 0, mientras que el espacio tiene el valor ASCII de 32.

El uso más común de la cadena nula es para limpiar o declarar una variable de texto de manera que no tengan accidentalmente un efecto sobre el programa al poseer un valor previo no deseado.

Es también el valor de una variable de leer si no se escribe nada en respuesta y se pulsa ↵.

Salvando los procedimientos

salvar ↵ listad4 ↵

editar ↵ TAB TAB ⇩ F5 CTRL⇨ *(6 veces)* imprim ↵ ESC

1. Salve los procedimientos a listad4. Contendrá los procedimientos impragenda, ff, línea, listagenda y listinvit.
2. Para cambiar el procedimiento entero de manera que pase de escribir en la pantalla a imprimir en la impreso-

ra, cambie el comando escribir por imprimir en el procedimiento línea:

```
proc línea; l$
   imprimir l$
   finproc
```

Edición de procedimientos: cortar y mezclar

Eva se da cuenta de que los procedimientos listinvit y listagenda son esencialmente el mismo, salvo por los campos impresos, que son distintos para cada fichero.

Decide cortar las líneas línea; de listinvit y ponerlas en un procedimiento distinto.

Cortar

editar ↵

1. Escriba editar y pulse ↵.

TAB *(4 veces)* F5 ⇧⇨ CTRL⇦ *(5 veces)* a ↵

2. Cambie el nombre del procedimiento listinvit a lista.

⇩ ⇩ ⇩

3. Haga que la línea mientras sea la activa, de color blanco.

F4 muestinvit ↵

4. Inserte la línea muestinvit para llamar al nuevo procedimiento.

⇩

5. Haga la línea línea; nombre$ la línea activa.

F3 q

6. Pulse F3 y la letra q.
   Lea las indicaciones de la caja. Le indican que use las flechas del cursor. Debe hacerlo sólo si quiere quitar más de una linea, como ahora.

⇩ ⇩ ⇩ ⇩ ⇩

7. Pulse la flecha hacia abajo 5 veces, de manera que un bloque de líneas hasta, e incluyendo, línea; " " esté de color blanco. Si se pasa, vuelva atrás con la flecha arriba.

↵

8. Pulse ↵.
   El bloque de líneas completo desaparece y las líneas restantes se cierran:

```
proc lista
   primero
   limpiar
   mientras no finf( )
      muestinvit
      próximo
      finmientras
   finproc
```

Cuando se quitan lineas de un procedimiento no desaparecen para siempre de inmediato. Se almacenan temporalmente en un área de memoria llamada un *buffer* (memoria intermedia).

Permanecen ahí hasta que se reemplazan por el siguiente conjunto de líneas quitadas, hasta que abandona el editor o hasta que se transfieren a otro sitio por una operación de "mezcla".

Ahora vamos a "mezclarlas" en un nuevo procedimiento. Hágalo inmediatamente por si algún accidente vacía el *buffer* de edición.

Buffer

Son zonas de memoria que almacenan información secuencialmente (por regla general) y que la liberan en el mismo orden exacto cuando es necesario. Esto se llama un *buffer* FIFO *(First In First Out)* o una cola.

La mayor parte de las impresoras tienen memoria, llamada *buffer* de la impresora, que almacena la información enviada por el ordenador hasta que la impresora está lista para procesarla. Por eso muchas impresoras siguen imprimiendo durante bastante tiempo después de que el ordenador haya acabado de enviar datos. Las impresoras, por regla general, imprimen mucho más despacio de lo que envía datos el ordenador.

La mayor parte de los teclados de ordenador (incluyendo el QL) tienen memorias de teclado, aunque, por regla general, en ellos caben sólo unos pocos caracteres. Por eso puede comenzar a teclear un comando antes de que haya finalizado la ejecución del anterior. Cuando vuelve a aparecer el >, con él lo hacen las letras de golpe.

El *buffer* de edición de ARCHIVE funciona de una manera muy parecida. La información se almacena en él cada vez que se quitan líneas, y salen exactamente en el mismo orden cuando se utiliza el comando de "mezcla". La diferencia principal es que cada nueva operación de "cortado" borra el *buffer antes* de añadir las nuevas líneas. Si, por ejemplo, quiere mover tres partes de un procedimiento separadas, debe:

```
CORTAR MEZCLAR CORTAR MEZCLAR CORTAR MEZCLAR
```

y *no*

```
CORTAR CORTAR CORTAR MEZCLAR
```

Mezclar

F3 n
muestinvit ↵

1. Pulse **F3** y n para comenzar un nuevo procedimiento.
2. Llámelo muestinvit.
   La línea proc es ahora la línea resaltada.

ESC

3. Pulse **ESC** para dejar de insertar.

F3 m

4. Pulse **F3** y m. El último bloque de líneas cortado se une inmediatamente después de la línea activa; eliminándolas del *buffer* al hacerlo. Una vez usado, el comando mezclar no se puede utilizar otra vez hasta que se corten más líneas.

Se pueden mover líneas en un solo procedimiento, o de uno a otro.

ESC

5. Abandone el editor.

unir "listad4" ↵

6. Una el programa listad4.

salvar ↵ listad5 ↵

7. Salve los procedimientos a listad5:
   impragenda, ff, línea, lista, listagenda, listinvit, muestinvit.

trazar ↵

8. Conecte la *traza* si no lo está, para ver los tres procedimientos, línea, lista y muestinvit en acción.

lista ↵

9. Escriba lista y pulse ↵ para ver el resultado.

Juan y Pepa Martínez Rebollo
Majada del Viento, 25
28029 Madrid
MADRID

Luisa Sala y Carmen García Carrión
Gran Vía, 34
Orihuela
ALICANTE

... etcétera.

trazar ↵

10. Desconecte de nuevo la traza.

Uso de la función VALCAM( )

Puede utilizar ahora los procedimientos lista y línea para mostrar cualquier fichero registro a registro, siempre que escriba un procedimiento que determine qué campos hay que usar.

Eva piensa que esto podría dar problemas a la larga. No le divierte la idea de reescribir los procedimientos de listado para cada nuevo fichero que cree.

Escribe un procedimiento que listará un registro de cualquier fichero, sin conocer siquiera los nombres de los campos.

escribir valcam(0) ↵

1. Escriba escribir valcam(0) y pulse ↵.
ARCHIVE muestra el contenido del primer campo del registro activo:

Srta. Felisa de los Monteros

F5 ⇧⇦ ⇧⇨ ⇦ CTRL⇦ 1 ↵

2. Pulse **F5** y cambie el 0 por un 1.
Se escribe el segundo campo:

Vista Cielo

- La función valcam( ) devuelve el contenido del campo cuyo número aparece entre los paréntesis. Los campos se numeran comenzando por el cero. El último (séptimo) campo es, por tanto, el campo número 6.

UN PROCEDIMIENTO ESTANDAR DE LISTADO

editar ↵ n muestreg ↵ haz cuent = 0 ↵ mientras cuent < 7 ↵ escribir valcam(cuent) ↵ haz cuent = cuent + 1 ↵ finmientras ↵ ESC

1. Cree el procedimiento siguiente:

```
proc muestreg
   haz cuenta = 0
   mientras cuenta<7
   escribir valcam(cuenta)
   haz cuenta = cuenta + 1
   finmientras
finproc
```

Este bucle es el primero que vemos que no determina su salida en relación con el movimiento de registro en registro. Su ejecución tendrá lugar un número finito de veces, en este caso siete.

La palabra cuenta es una variable numérica, y se utiliza como un contador que determina el número de veces que se ejecuta el bucle.

Cuando cuenta llega a ser igual a 7, se detiene el bucle. ARCHIVE no se confunde con la función cuenta( ), ya que ésta lleva paréntesis tras el nombre.

Recuerde que las variables numéricas se identifican por no llevar un signo dólar al final, y que se utilizan para almacenar números que se van a utilizar en los cálculos.

No se pueden utilizar cadenas de caracteres como variables numéricas. Comandos como haz cuenta = "0" o haz cuenta = nombre$ darán lugar a errores.

ESC **2.** Abandone el editor.

trazar ↵ **3.** Use trazar para observar el procedimiento.

muestreg ↵ **4.** Escriba muestreg y pulse ↵.

El registro activo se lista campo a campo, uno en cada línea:

```
Srta. Felisa de los Monteros
Vista Cielo
Urb. Los Montes
Villanueva de Segre
TARRAGONA
1
```

trazar ↵ **5.** Vuelva a desconectar la *traza*.

Cómo funciona el procedimiento

Veamos lo que sucede, paso a paso:

haz cuenta = 0:

Cuando comienza el procedimiento, el comando haz asigna el valor a la derecha del signo igual, 0, a la variable nombrada a la izquierda. El valor 0 es ahora almacenado en memoria bajo el nombre cuenta. Podría, por supuesto, haberse llamado contador, o x, o cualquier nombre de variable válido.

Esta línea es el equivalente del comando primero en el bucle mientras no finf( ). Si no hace cero el contador antes de comenzar un bucle de cuenta, podría tener ya un valor alto, y el bucle no se ejecutaría ni una sola vez. Si la variable cuenta no está definida, el procedimiento se detiene cuando se usa en el comando mientras.

mientras cuenta<7:

La primera vez que ARCHIVE lee esta línea, el valor de cuenta es 0. el símbolo < significa "sea menor que", y la línea se puede leer como "Mientras (el valor de) cuenta sea menor que 7 (ejecutar los comandos hasta el comando finmientras)".

escribir valcam(cuenta):
En esta línea, ARCHIVE sustituye el valor de cuenta, 0, en el paréntesis, y escribe el contenido del campo 0, es decir, el primer campo.

haz cuenta = cuenta + 1:
Esta línea le añade uno al valor de cuenta. ARCHIVE calcula primero el valor de los números de la parte derecha del signo igual. Mira en cuenta y sustituye un cero, y a continuación le suma 1. Almacena el resultado, 1, en memoria bajo el nombre de cuenta, sustituyendo el valor anterior.

finmientras:
Al llegar a este comando, ARCHIVE vuelve a la primera línea del bucle y comprueba para ver si el bucle debe o no seguir.
El valor del contador es ahora 1, que sigue siendo menor que 7, y el bucle continúa. ARCHIVE vuelve a ejecutar el comando siguiente a mientras.
Cuando el valor de cuenta llegue a 7, el test mientras que gobierna el bucle fallará y el resultado es que el programa saltará fuera del bucle. El comando inmediatamente posterior a finmientras se ejecutará.
Observe que, aunque el bucle no se ejecuta cuando el contador llega a 7, se realizan siete pasadas del bucle, ya que la cuenta comienza en cero.

Uso del procedimiento de listado estándar

Eva prueba su nuevo éxito sobre el fichero agenda.

editar ↵ TAB TAB TAB ⇩ *(4 veces)* F5 CTRL ⇩ muestreg ↵ ESC

1. Utilice el editor para cambiar el procedimiento lista de manera que llame al procedimiento de uso general muestreg en lugar del antiguo muestinvit.

El procedimiento quedará como sigue:

```
proc lista
    primero
    limpiar
    mientras no finf()
        muestreg
```

```
    próximo
    finmientras
finproc
```

usar "a" ↵

lista ↵

2. Use el fichero agenda.
3. Escriba lista y pulse ↵.
   Se muestra el contenido de los tres campos del primer registro, tras lo cual aparece el error "Argumentos no válidos" (si tiene versión 2). Si tiene versión 1, el programa da error a partir del tercer campo, ya que trata de buscar campos que no existen, y saca otra información de la memoria.

```
Ver al sacerdote
04/08
```

La función NUMCAM()

Afortunadamente, el programa de listado de propósito general que acaba de escribir Eva no será inútil para los ficheros de menos de siete campos.

La función numcam() devuelve el número de campos de un fichero (el fichero activo, a menos que aparezca un nombre lógico entre los paréntesis).

escribir numcam() ↵

1. Escriba escribir numcam() y pulse ↵.
   Aparece el número 3 en la pantalla.

editar ↵ TAB *(7 veces)* ⇩ ⇩ F5 ⇩ CTRL ⇦ numcam() ↵ ESC

2. Cambie el 7 en el procedimiento muestreg a numcam(), de manera que la línea quede:

```
mientras cuenta<numcam()
```

El procedimiento debe quedar como sigue:

```
proc muestreg
    haz cuenta=0
    mientras cuenta<numcam()
        escribir valcam(cuenta)
        haz cuenta=cuenta+1
        finmientras
    finproc
```

El procedimiento lista funciona ahora con cualquier fichero.

lista ⏎

3. Escriba **lista** y pulse ⏎ para verlo funcionar perfectamente con el fichero agenda:

```
Ver al sacerdote
04/08

Pedirle a Felicidad que sea la madrina
04/05
t
Decirle a Juana que haga los trajes
04/05
t
```

... etcétera.

salvar ⏎ listad6 ⏎

4. Salve los procedimientos a **listad6**.
El fichero contendrá los procedimientos: ff, impragenda, linea, lista, listagenda, listinvit, muestivint y muestreg.

Impresión de procedimientos con LISTAR

El comando listar imprime el contenido de los procedimientos presentes a la impresora. El nombre procede de la costumbre, en la jerga informática, de llamar "listado" a la impresión de los comandos de un programa.

Los procedimientos aparecen indentados y en orden alfabético, como en editar.

La impresión a papel de los procedimientos es útil, ya que es más fácil trabajar en papel si se está usando un programa complicado. Se le puede añadir notas y seguir su ejecución paso a paso, para buscar errores y *bugs* (chinches). Este término, ya clásico, denota los errores difíciles de encontrar y molestos, y deriva de las averías de los grandes ordenadores, causadas con frecuencia por cucarachas que anidaban entre los circuitos.

De ahí que los programadores le echaran la culpa de sus errores a los *bugs,* eludiendo su responsabilidad. El término ha perdurado con el significado de errata, sobre todo para aquellas difíciles de encontrar o las que se muestran sólo bajo circunstancias excepcionales.

Los listados se pueden usar para analizar los programas paso a paso anotando los valores de las variables (se suele llamar *ejecución en seco*).

La figura 6.1 muestra cómo el error de la versión anterior de muestreg se podía haber detectado "ejecutando en seco".

```
proc muestreg
    haz cuenta = 0
    mientras cuenta < 7
        escribir valcam(cuenta)
        haz cuenta = cuenta + 1
        finmientras
    finproc
```

Pasos por el bucle

1	2	3	4
0			
0<8	1<8	2<8	3<8
0 = notas$	1 = fecha$	2 = accion$	3 = ???
0 + 1 = 1	1 + 1 = 2	2 + 1 = 3	
			no hay campo n.° 4

Figura 6.1. Ejecución en seco (muestra)

Asegúrese de que la impresora está preparada antes de usar listar.

Borrado de ficheros de programa inútiles

El programa listado ha pasado por muchos estadios, actualizándose a cada paso, y salvándose con un nuevo número de versión cada vez. Ahora hay seis versiones: listado a listad6. Esta última es la más actualizada, y contiene todos los procedimientos relevantes de las otras versiones.

tirar ↵ listado_prg ↵ tirar ↵ listad2_prg ↵ tirar ↵ listad3_prg ↵ tirar ↵
listad4_prg ↵ tirar ↵ listad5_prg ↵

1. Borre todas las versiones de listado excepto listad6 para abrir sitio en el cartucho.

salvagrdr ↵ listad6_prg ↵ listado_prg ↵ tirar ↵ listad6_prg

2. Cambie el nombre de listad6 a listado. Para ello debe primero salvagrdr con el nuevo nombre, y luego tirar el fichero antiguo.

dir ↵ ↵

3. Haga un directorio de la unidad 2 para comprobar que tenemos los siguientes programas en cinta: fácil_prg, FichBoda_prg, vacío_prg y listado_prg.
Compruebe la fecha en el fichero _eti de la unidad 2.

dir ↵ mdv1_ ↵

4. Haga un directorio de la cinta de seguridad para ver si la fecha en el fichero _eti indica que es el cartucho correcto. Si coincide con la fecha del cartucho de trabajo, debe usar la otra cinta.

nuevo ↵

5. Borre todos los procedimientos y ficheros de datos de memoria temporal escribiendo nuevo y ↵.

salvagrdr...

6. Haga una copia de seguridad de los ficheros cambiados o creados desde la última vez que hizo una copia.

salvar ↵ Abr6_eti ↵ F5 ⇧⇨ ⇨ mdv1_ ↵

7. Salve ficheros de etiqueta con la fecha de hoy en ambos cartuchos.

tirar ↵ ????_eti ↵

8. Borre los ficheros _eti antiguos de ambos cartuchos.

Véase el capítulo 12 para más información sobre el listado de datos, incluyendo la impresión de etiquetas.

RESUMEN

En este capítulo, Eva utiliza distintos comandos y procedimientos para imprimir información de sus dos ficheros agenda e invitado en la pantalla y la impresora.

En primer lugar crea un procedimiento llamado listagenda que lista los campos del fichero agenda en pantalla, línea a línea. Utiliza un punto y coma al final de una sentencia de escritura para evitar que se envíe un retorno de carro. Usa a continuación el comando tab para hacer que los campos se impriman en un punto específico de la pantalla.

Salva se da cuenta de que la lista de Eva necesita un poco de vida, y usa los comandos papel y tinta para añadirle color. Usa la función val() para cambiar el color de cada anotación en función de su fecha.

Eva crea ahora un procedimiento para listar sus notas por impresora, de manera que pueda verlas en papel. Llama al procedimiento impragenda, y cambia el comando escribir que usó antes por imprimir. Este comando envía la información a la impresora, en vez de hacerlo a la pantalla. Utiliza la función car() en un procedimiento llamado ff para hacer que la impresora salte de página (haga un *"form feed"*).

Tras ello, Eva vuelve su atención al fichero invitado, y cambia el procedimiento listagenda para que funcione con este archi-

vo. Escribe entonces un nuevo procedimiento llamado línea, que evitará que tenga que cambiar todas las instrucciones escribir por imprimir cada vez que quiera imprimir el fichero en impresora en lugar de la pantalla. El nuevo procedimiento recibe "parámetros", lo que permite su uso con distintos objetivos.

A continuación Eva utiliza el "cortado" y "mezcla" del editor para crear un procedimiento que se llamará muestinvit, y lista el fichero invitado en pantalla. Utiliza la función valcam() para permitir que el procedimiento se use para cualquier fichero. Para ello crea un contador de bucle, al que se le suma 1 cada vez que se imprime un campo, y sale del bucle al finalizar el registro. Usa también la función numcam() para que ficheros con distinto número de campos puedan usar el procedimiento.

Finalmente, Eva imprime una lista de todos los procedimientos mediante el comando listar. Borra a continuación las versiones del programa listado que han sido mejoradas por la última, listad6.

Trucos útiles

- Si no sucede nada cuando envía un comando de impresora a una impresora bien inicializada, probablemente ha olvidado pulsar el botón ON-LINE de la impresora.
- Asegúrese de imprimir la fecha en todos los listados y ficheros de programa. Si no, no sabrá cuál es la última cuando tenga varias versiones.

7

Selección de información

No tiene ningún sentido almacenar gran cantidad de información en un ordenador, a menos que se pueda recuperar cada vez que se necesite. ARCHIVE recupera información más deprisa y con más precisión que cualquier sistema de archivo manual en papel. La búsqueda de registros por uno o varios de sus contenidos es fundamental para el uso de una base de datos, y se puede hacer de muchas maneras.

En este capítulo y el próximo aprenderá los métodos disponibles de selección y búsqueda de datos, así como los pros y contras del uso de diferentes comandos para ello, y escribirá un procedimiento para hacer más fácil el uso de continuar.

Aprenderá también sobre métodos para comprobar y validar la entrada de datos (por ejemplo, fechas) mediante expresiones lógicas.

Se presentan las siguientes características de ARCHIVE:

- Comandos de ARCHIVE:

buscar	— encuentra un registro por un campo específico
elegir	— selecciona registros que cumplen ciertos criterios
restaurar	— devuelve el fichero a su estado normal

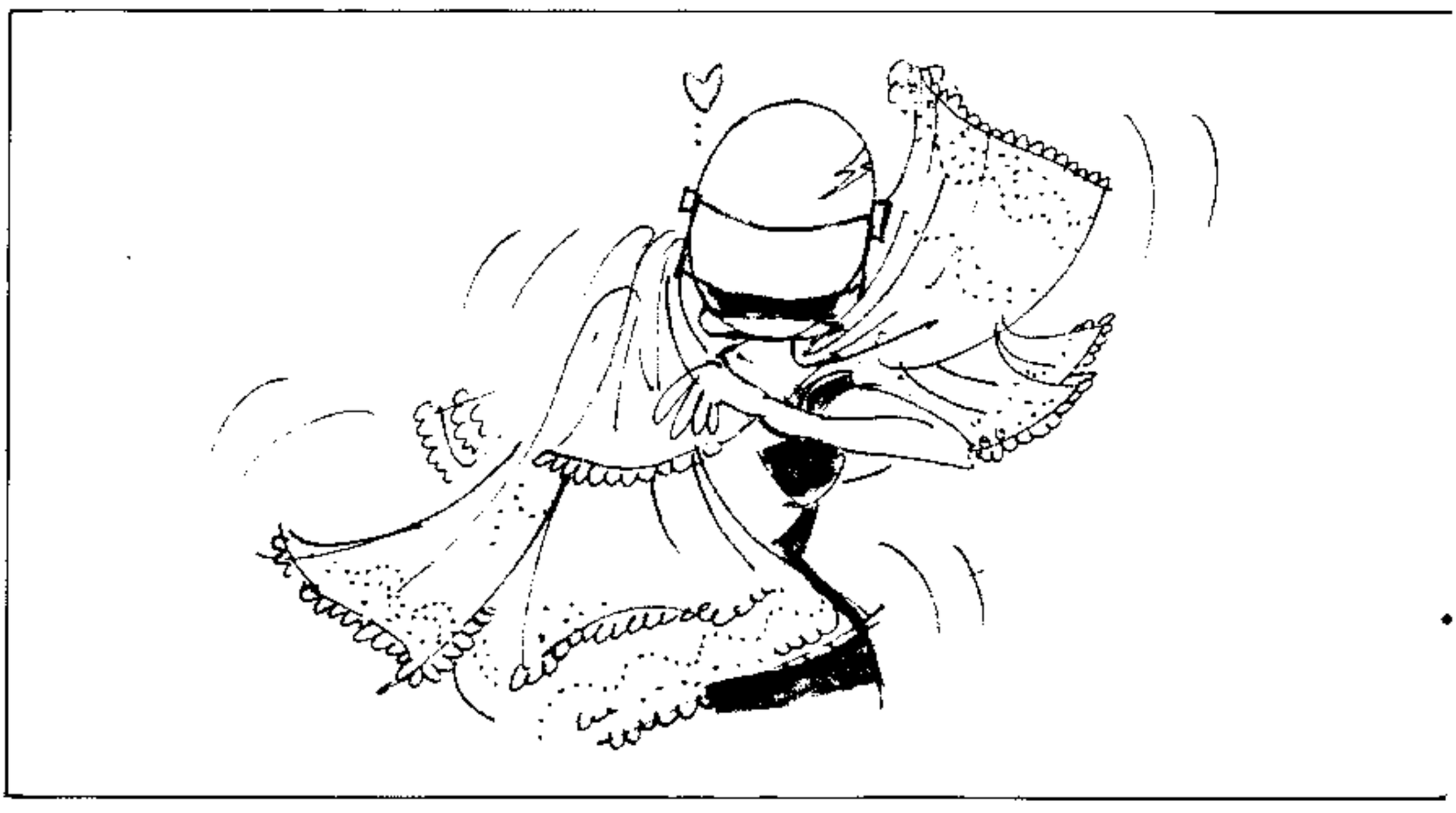
- Funciones de ARCHIVE:

mayús()	— convierte letras a mayúsculas
minúsc()	— convierte letras a minúsculas
hallado()	— comprueba si un hallar ha tenido éxito
enserie()	— busca ciertos caracteres en cualquier posición en una serie alfanumérica
long()	— devuelve la longitud de una cadena de caracteres como un número que representa el número de caracteres en la serie

- Los operadores >< (mayor y menor que), yy, oo y no.
- Fragmentación de variables alfanuméricas.

Puntos a recordar

- continuar sigue aplicando el último comando buscar o hallar, no importa el tiempo que hace que se usó, y comienza tras el registro activo.
- Utilice restaurar cuando termine una selección, o las búsquedas siguientes pueden ser incorrectas.
- ARCHIVE ignora totalmente los registros no elegidos (es como si no existieran temporalmente). Comandos como todos sólo afectan a todos los registros seleccionados.



Muy ajustado

Ejemplo: Preparaciones de boda: ¿qué y cuándo?

Comienzo

Si no ha cargado ARCHIVE, hágalo ahora. No debe tener ningún fichero de datos abierto.

Ponga un cartucho de seguridad en la unidad 1 y utilice dir para comprobar que el fichero _eti (véase el capítulo 3) no es del mismo que en el cartucho de trabajo.

BUSQUEDA DE REGISTROS ESPECIFICOS

Instrucciones

ver ↵ agenda"lógico "a ↵

1. Escriba ver y pulse ↵. Escriba entonces agenda" lógico "a y pulse ↵ para abrir el fichero agenda. Observe que ARCHIVE sólo proporciona el primer conjunto de comillas automáticamente. Para evitar pasar por encima de ella con el cursor, simplemente no escriba la última de cierre tras el nombre lógico.

indicar ↵

2. Escriba indicar y pulse ↵.

cargar ↵ zoom ↵

3. Cargue el programa zoom.

Eva quiere ver todas las notas que le recuerdan hacer una llamada telefónica.

El comando hallar, presentado en el capítulo 2, no tiene utilidad para hallar las notas con una t en el campo de acción, ya que todos los registros con una t en cualquier parte de cualquier campo, incluido notas$, aparecerá si escribe hallar "t". Por ejemplo, Pérez Fotógrafos.

El comando BUSCAR

El comando buscar halla los registros comparando campos específicos con palabras o valores específicos. Como el comando hallar, comienza con el primer registro del archivo y comprueba si cumple la condición pedida. Si no lo hace, busca en los registros siguientes hasta que lo hace. Si se encuentra un registro que coincide, se puede continuar la búsqueda con el comando continuar. Este conserva el último registro que funcionó si no hay más coincidencias.

buscar accion$ = "t" ↵

1. Escriba buscar accion$ = "t" y pulse ↵.
La presentación cambia hasta mostrar el segundo registro:

Decirle a Fecilidad que sea la madrina

continuar ↵

2. Escriba continuar y pulse ↵.
Así encontrará el siguiente registro con accion$ = "t":

Pedirle a Juana que haga los trajes

F5 ↵

3. Pulse **F5** seguido por ↵ para ver más.
Sólo los registros que tengan la letra t en el campo accion$ se muestran, hasta que no quedan más. Deben aparecer los siguientes registros:

```
notas$
Pedirle a Felicidad que sea la madrina
Decirle a Juana que haga los trajes
Buscar un Hotel para los invitados
Decirle al Sr Hort que se encargue de las flores
Contratar a Pèrez Fotògrafos
Coche
Llamar a Sara o Quièn les organizò el banquete ?
Capacidad de la sala Azul del Hotel Almirante
Encargar la tarta
```

El comando buscar es absolutamente pedante sobre lo que busca, tanto en el caso de comparar mayúsculas o minúsculas como en la longitud de las líneas. Por ejemplo, el comando buscar accion$ = "t" ignorará los registros cuyo campo contenga los dos caracteres "t " (la letra t seguida por un espacio en blanco).

El comando buscar es distinto de hallar, ya que tiene muy en cuenta si las letras están en mayúsculas o minúsculas. El comando que acabamos de usar no habría encontrado los registros con la letra T en el campo accion$. Esta búsqueda se llama "dependiente de la caja".

MAYUS() y MINUSC()

Si duda si los datos están almacenados en mayúsculas o minúsculas, cambie los caracteres a unas u otras mediante las funciones mayús() o minúsc(). La función mayús() convierte las letras en mayúsculas, sin afectar al resto de los caracteres (por ejemplo, números). La función minúsc() hace la tarea contraria.

Por ejemplo:

buscar mayús(accion$) = "T" encontrará las tes mayúsculas o minúsculas. Pruebe para asegurarse.

Así no se cambia los contenidos de los campos, sólo quiere decir que, en tanto trata de encontrar un campo, ARCHIVE convertirá todo a mayúsculas antes de comparar con las letras que buscamos.

UN PROCEDIMIENTO QUE SIGUE BUSCANDO REGISTROS ESPECIFICOS

Bucles sin fin deliberados

Eva quiere escribir un procedimiento que le ahorre el uso de continuar en la línea de comandos cada vez que quiere ver el siguiente registro que cumple la condición.

editar ↵ F3 n vermás ↵ mientras 1 1 ↵ continuar ↵ pescribir ↵ finmientras
ESC

1. Use el editor para crear el procedimiento vermás, y escríbala como sigue:

```
proc vermás
    mientras 1 = 1
        continuar
        pescribir
        finmientras
    finproc
```

Este uso del comando mientras causa un bucle que no se detiene nunca, ya que la ecuación es siempre cierta. El valor 1 será siempre igual a 1.
La única manera de romper la ejecución de un bucle sin fin es pulsar **ESC**.

ESC
2. Abandone editar.

salvar ↵ vermás ↵
3. Salve el procedimiento a vermás.

Búsqueda de un campo vacío con una cadena nula

Antes de usar el nuevo procedimiento, Eva quiere buscar alguna otra cosa. Quiere ver ahora las notas generales (no relacio-

nadas con teléfono o compras, que se almacenaron sin nada en el campo accion$).

buscar accion$ = "" ↵

1. Escriba la línea anterior y pulse ↵.
   Observe el uso de una cadena vacía. Recuerde que es distinto de " ", que contiene un espacio, un carácter como otro cualquiera. Una cadena vacía tiene el valor ASCII 0, mientras un espacio tiene el valor ASCII 32.
   Una cadena vacía sólo funciona con campos alfanuméricos. Los campos numéricos vacios contienen el valor cero. No pueden estar vacíos del todo.
   Aparece el primer registro del fichero:

```
Ver al sacerdote
```

vermás ↵

2. Escriba vermás y pulse ↵.
   Aparecen uno tras otro los cuatro registros restantes que cumplen la condición.

```
Que no se olvide hacer los ojales
¿ Ha comprado Salva un traje de sport ?
Hacer una lista de bodas
No olvidar aguja e hilo
```

ESC

3. Pulse **ESC** para detener el bucle infinito.

El comando SI

Es desde luego inconveniente y chapucero dejar vermás como está.

editar ↵ TAB TAB ↓ ↓ F4 si hallado() = 0 ↵

1. Inserte estas líneas en el procedimiento:

```
si hallado() = 0
```

   Esta línea comprueba si la función hallado() devuelve un 0 (no hallado) o un 1 (hallado un registro). La línea se podría haber escrito también si no hallado().

stop ↵

2. stop.
   El comando stop hace que se detengan todos los procedimientos, y devuelve el control a la línea de comandos.

finsi ↵

3. finsi.
   Como mientras ... finmientras, el comando si debe ir se-

guido por un finsi, de manera que ARCHIVE sepa qué comandos están relacionados con él.

El procedimiento debe quedar parecido al siguiente:

```
proc vermás
    mientras 1=1
        continuar
        si hallado()=0
            stop
            finsi
        pescribir
        finmientras
    finproc
```

↵ ESC

4. Pulse ↵ para dejar el modo de inserción, y **ESC** para salir del editor.

salvar ↵ vermás ↵ F5 ⇧⇨ ⇨ mdv1 ↵

5. Salve el procedimiento al programa vermás y, tras comprobar que el cartucho de seguridad está en la unidad 1, sálvelo a mdv1_vermás, usando **F5** para ahorrar pulsaciones.

Los dos nuevos comandos, parar y si seguido por finsi han sido introducidos al cambiar el procedimiento, además de una nueva función, hallado().

La sentencia si se puede parafrasear como "Si no (registro hallado), stop".

La función HALLADO()

La función hallado() es parecida a finf() en que devuelve un valor de 1 si es cierta, y 0 en caso contrario. Es decir, que si los comandos buscar o continuar tienen éxito, el valor de hallado() es 1, o cierto. En caso contrario, si no se encontró un registro, hallado() será igual a 0.

La función hallado() funciona con el comando hallar de la misma manera exactamente.

En la versión 2 de ARCHIVE, hallado() funciona también con situar (descrito en el próximo capítulo).

SI y FINSI

La construcción si ... finsi es similar a mientras ... finmientras en que los comandos están emparejados y no deben existir por

separado. También se parecen en que los comandos que haya entre ellos se ejecutan sólo si la condición que sigue al primer comando es cierta.

Por ejemplo, mientras no finf() ejecuta los comandos del bucle mientras es cierto que no se ha alcanzado el fin del fichero.

El comando(s) entre si y finsi en este ejemplo se ejecuta cuando es cierto que hallado() = 0, es decir, cuando no se ha encontrado ningún registro. Hasta entonces, el comando en el comando si se ignora y el procedimiento actúa sobre la primera línea tras finsi.

Cierto y falso

Comandos como si y finsi determinan si una expresión que les sigue es cierta o falsa en función de que tenga un valor de 1 ó 0.

Las comparaciones y búsquedas (usando operadores como =), las pruebas condicionales (si y mientras), y estados (como fin de fichero) todos se basan en la idea de cierto y falso. Las búsquedas y los comandos se hacen si la expresión relevante es cierta, y no si es falsa.

Los ordenadores representan la salida de estas evaluaciones como 0 ó 1. Conceptos como éxito y fracaso, verdad y mentira, cierto y falso, sí y no, se reducen simplemente a valores distintos de cero (1) y cero (0).

Esto explica por qué, por ejemplo, el manual de ARCHIVE dice que la sintaxis de la sentencia buscar requiere una expresión numérica (buscar <expr.n.>). No quiere decir que buscar funcione sólo con campos numéricos, sino que requiere una expresión que evalúe a un valor numérico que represente cierto o falso (encontrado o no).

La noción de una expresión cierta o falsa es crucial para entender el funcionamiento de los comandos si y mientras. Los comandos evalúan las expresiones siguientes, y ejecutan los comandos hasta finsi o finmientras sólo si el resultado es 1, o cierto. Si el resultado es 0, o falso, los comandos tras el si se ignoran, y los bucles mientras acaban.

Evaluación de expresiones

Una expresión es cualquier combinación de valores, sea texto, números o funciones, que puede evaluarse para obtener un solo resultado.

El término valor se usa tanto para el valor numérico de

un campo numérico como para el contenido de una cadena de caracteres.

A continuación se pueden ver algunos ejemplos de expresiones:

```
16 + 1
cuenta()/50*.16
```

Ambos ejemplos son expresiones numéricas, y no contienen cadenas de caracteres (la función CUENTA() devuelve un número).

```
"Don " + nombre$ puede evaluar como "Don José
    Pérez"
fecha$ + " " + mayús(accion$) evalúa como "04/03 T"
accion$ = "t" evalúa como cierto si son iguales
```

Los tres ejemplos anteriores son expresiones literales, y no contienen números a menos que sean parte de la cadena de caracteres.

Evaluación de expresiones lógicas y de relación

Observe que si el último ejemplo evalúa como cierto, devuelve el valor 1. Si el resultado fuera falso, devolvería el número 0.

El comando escribir 10 = 13 − 3 muestra el número 1, ya que la expresión es correcta (cierta).

La sentencia escribir = 10 = 2*3 escribe el número 0. La expresión está equivocada (falsa).

La sentencia haz x = nombre$ = "Salva" parece incorrecta a primera vista, ya que parece mezclar números con texto. De hecho, la expresión es nombre$ = "Salva", que evalúa como 1 si el campo contiene Salva, y como 0 en caso contrario. x toma el valor del resultado.

Observe aquí los dos sentidos distintos del signo "=". El primero es parte del comando haz, y asigna el valor de la expresión siguiente a la variable anterior. El segundo es un operador de relación en una comparación entre cadenas, y sirve para devolver un valor cierto o falso, que se asigna entonces a x.

Uso del procedimiento VERMAS

primero ↵
pantalla ↵
vermás ↵

1. Use primero para ir al principio del fichero agenda, y
2. Use pantalla para mostrar el primer registro.
3. Escriba vermás y pulse ↵.
   Pese a que no se ha escrito ningún comando buscar, el procedimiento muestra todos los registros que encajan en el último comando de búsqueda (buscar accion$ = ""), y se detiene cuando no encuentra más. Debe seleccionar los registros siguientes:

```
notas$
Ver al sacerdote
Que no se olvide hacer los ojales
¿ Ha comprado Salva un traje de sport ?
Hacer una lista de bodas
no olvidar aguja e hilo
```

El comando continuar en vermás actúa sobre el último comando buscar o hallar, no importa el tiempo que hace que se utilizó, mientras la memoria no haya sido borrada. Continuar comienza su búsqueda en el primer registro tras el activo, no importa si cumple la condición o no.

Mayor y menor que

buscar accion$ >"" ↵

1. Escriba buscar accion$ >"" y pulse ↵.
   Aparece el primer registro que tiene una t en el campo accion$:

Decirle a Felicidad que sea la madrina

vermás ↵

2. Use vermás para ver el resto de las coincidencias. El símbolo > quiere decir "mayor que", y ARCHIVE ha buscado todas las notas con t o c, es decir, campos accion$ que tienen más de nada:

```
notas$
Pedirle a Felicidad que sea la madrina
Decirle a Juana que haga los trajes
Buscar un Hotel para los invitados
Decirle al Sr Hort que se encargue de las flores
Zapatos blancos
```

```
Comprar el traje
Contratar a Pèrez Fotògrafos
Coche
Llamar a Sara ¿ Quièn les organizò el banquete ?
Capacidad de la sala Azul del Hotel Almirante
Encargar la tarta
```

El símbolo opuesto es <, que quiere decir "menor que".

buscar accion$< >"t" ↵

3. Escriba buscar accion$< >"t" y pulse ↵.
Aparece el primer registro con algo distinto de t en el campo accion$:

```
Ver al sacerdote
```

vermás ↵

4. Use vermás para mostrar los registros siguientes, que deben ser:

```
Que no se olvide hacer los ojales
Zapatos blancos
Comprar el traje
¿ Ha comprado Salva un traje de sport ?
Hacer una lista de bodas
no olvidar aguja e hilo
```

Los símbolos <> juntos quieren decir "menor que o mayor que", en otras palabras, "distinto".

Operadores de relación

Los operadores de relación se pueden utilizar para comparar la relación de un valor (numérico o literal) con otro.

Por ejemplo:

= comprueba si los dos valores son *iguales.*
> comprueba si el valor de la izquierda es *mayor que* el de la derecha.

El conjunto completo de operadores de relación es:

=	igual a
>	mayor que
<	menor que
>=	mayor o igual que
<=	menor o igual que
<>	distinto

Observe que >< , => y =< no funcionarán como combinaciones.

Si tiene problemas para distinguir < de >, recuerde que el valor del lado estrecho (la punta) debe ser menor que el del lado ancho (abierto) para que la expresión tenga un valor cierto.

Mejoras de VERMAS

El procedimiento vermás es todavía bastante chapuza: la sentencia si comprueba una condición (hallado o no), que podría comprobarse en el comando mientras. En otras palabras, el bucle debe continuar mientras hallado() es cierto, y devuelve el valor 1.

editar ↵ TAB TAB TAB

1. Use el editor y **TAB** para resaltar vermás.

↓ ↓ ↓ F3 q
↓ ↓

2. Cursor abajo hasta la línea si y pulse **F3** y q.
3. Cursor abajo dos veces más para marcar las otras dos líneas de la sentencia si para borrado. Si se pasa, vuelva en dirección opuesta.

↵

4. Pulse ↵ para borrar las tres líneas, si, stop, finsi.

⇧ F5 ⇧⇨ CTRL ↓ hallado() ↵ ESC

5. Use **F5** para cambiar la linea mientras 1 = 1 hasta que quede mientras hallado(). Esto equivale a mientras hallado() = 1, ya que la función y la expresión devuelven el valor 1 si hallado() es cierto.

El procedimiento queda ahora así:

```
proc vermás
    mientras hallado()
        continuar
        pescribir
        finmientras
    finproc
```

salvar ⏎ vermás ⏎ F5 ⇧⇨ ⇨ mdv1_ ⏎

6. Salve el procedimiento al programa vermás, y a mdv1_vermás.

El procedimiento comienza sólo si hallado() es cierto y se ha encontrado un registro. Sigue entonces ejecutando el bucle sólo mientras hallado() sea cierto tras cada continuar. Entonces para, tras el último registro que encaja.

Búsqueda de parte de un campo mediante subcadenas

Eva quiere saber qué tendrá que hacer el mes de junio.

buscar fecha$ = "06" ⏎

1. Escriba buscar fecha$ = "06" y pulse ⏎.
   Aparece el primer registro, con 04 al comienzo de la fecha.
   El comando buscar ha fallado pese a que hay notas de junio en el archivo. Es porque el valor que se busca debe coincidir con el campo entero *exactamente.* La cadena 06 no es igual que 06/01.

Se puede buscar ocurrencias parciales de una cadena alfanumérica diciéndole a ARCHIVE las posiciones de los caracteres que nos interesan. El campo fecha$ tiene el mes en la primera y segunda posiciones, y el día en la cuarta y quinta. La tercera posición la ocupa el separador, un *slash* (/).

El separador sólo tiene un cometido estético, ya que 06/01 es más fácil de leer como fecha que 0601. No tiene otra función, pero debe tomarse en consideración.

buscar fecha$(1 hasta 2) = "06" ⏎

2. Escriba buscar fecha$(1 hasta 2) = "06" y pulse ⏎.

vermás ⏎

3. Use vermás para comprobar el resultado.
   Sólo los dos registros que tienen los dos primeros caracteres de fecha$ iguales a 06 se muestran.

Subcadenas

Las subcadenas son al fin y al cabo una función (aunque no estén clasificadas así) que devuelve un valor que representa sólo parte de una cadena más larga.

Los números entre paréntesis que siguen la cadena (o variable de cadena) designan los caracteres que forman la

subcadena. El primer carácter de una cadena es el 1. Las posiciones del comienzo y el final de la cadena se pueden designar de varias maneras:

Por ejemplo, si el campo literal notas$ tiene el valor Ver al sacerdote:

```
escribir notas$(5)              escribirá una a
escribir notas$( hasta 3)       escribirá Ver
escribir notas$(2 hasta 5)      escribirá er a
escribir notas$(8 hasta)        escribirá sacerdote
```

Búsqueda de intervalos de registros

Eva le pide ahora a ARCHIVE que le muestre sus notas para la primera semana de abril.

Para encontrar los registros cuyas fechas cubran un período de más de un mes o de menos, es necesario especificarle a ARCHIVE un intervalo de fechas.

Un intervalo de valores incluye todos los valores desde uno dado (inicial), hasta el valor final del periodo.

Se especifica un intervalo dando la primera y última fecha que se deben incluir, utilizando los signos > y < para *capturar* cualquier fecha intermedia.

```
buscar fecha$> ="04/02" yy fecha$< ="04/08" ↵
```

1. Escriba la línea buscar fecha$> = "04/02" yy fecha$< = "04/08" y pulse ↵.

Las fechas representan la primera semana de abril. El primer registro, con fecha del 8 de abril, aparece:

```
Ver al sacerdote                                    04/08
```

```
vermás ↵
```

2. Escriba vermás y pulse ↵ para ver el resto de los registros seleccionados. Aparecen todas las notas con fecha 5 de abril:

```
Pedirle a Felicidad que sea la madrina              04/05
Decirle a Juana que haga los trajes                 04/05
Buscar un Hotel para los invitados                  04/05
Decirle al Sr Hort que se encargue de las flores    04/05
¿ Ha comprado Salva un traje de sport ?             04/05
Contratar a Pérez Fotógrafos                        04/05
Coche                                               04/05
```

```
Llamar a Sara ¿ Quién les organizó el banquete ? 04/05
Capacidad de la sala Azul del Hotel Almirante    04/05
Encargar la tarta                                04/05
Hacer una lista de bodas                         04/05
```

El uso de yy en una expresión hace que la fecha elegida sea *a la vez* mayor o igual que el día 2, y (y además) menor o igual al día 8. El 14, por ejemplo, cumple la primera condición, pero no es menor que el día 8, y por tanto no se le elige.

Búsqueda de cadenas con la función ENSERIE()

Salva quiere comprobar si es necesario que haga algo en la preparación de la boda. Tiene que encontrar todas las notas del fichero agenda en que aparezca su nombre.

El comando buscar es distinto de hallar por otra característica significativa. Compara el texto exactamente con un campo entero (o parte de un campo, si usamos subcadenas). El comando hallar busca texto en cualquier parte de cualquier campo.

Para buscar texto en cualquier parte de un campo especifico, utilice la función enserie(). La palabra es una abreviatura de en una serie.

La función actúa con buscar de la misma manera que hallar, pero sobre un solo campo.

buscar enserie(notas$,"Salva") ↵

1. Escriba buscar enserie(notas$, "Salva") y pulse ↵.
   Aparece la primera nota con el nombre de Salva:

   ¿ Ha comprado Salva un traje de sport ?

vermás ↵

2. Use vermás para ver las restantes coincidencias. No hay ninguna.

La función ENSERIE()

La función enserie():

- Debe llevar dos cadenas de caracteres entre los paréntesis, separadas por una coma. Pueden ser cualquier expresión alfanumérica: variables, como un nombre de campo; constantes literales, entre paréntesis; fun-

ciones que devuelvan cadenas como resultado; o combinaciones de los anteriores;
- devuelve un valor de cero si la segunda cadena no existe en alguna parte de la primera. Si la segunda aparece en la primera, enserie() devuelve la posición del primer carácter coincidente en la cadena más larga.

Por ejemplo, escribir enserie("Ver al sacerdote","sac") mostrará el número 8.

- Evalúa como cierta siempre que se halle la subcadena.

Observe que la función enserie() distingue entre mayúsculas y minúsculas como buscar pero al revés que hallar. Se dice que es "dependiente de la caja".

El comando ELEGIR

Eva quiere ver los registros que le recuerdan cosas que debe hacer antes de mayo.

Hasta ahora, el comando buscar se ha utilizado para hallar registros que cumplieran un cierto requerimiento de manera que pudieran verse. Para utilizarlos para algo más (como listados o actualizaciones) habria que utilizar comandos que comprobaran el valor de hallado() en un bucle. Así se podría procesar cada registro coincidente de la manera apropiada.

Con frecuencia es más fácil elegir los registros coincidentes en primer lugar, y procesarlos a voluntad a continuación.

elegir fecha$< "05/01" ↵

1. Escriba elegir fecha$< "05/01" y pulse ↵.
   Aparece la siguiente anotación:

   Ver al sacerdote

primero ↵

2. Escriba primero y pulse ↵.
   El registro no cambia. El comando elegir deja el fichero posicionado en el primer registro que cumple la condición, y no en el último, como hace buscar al acabar su búsqueda.

zoom ↵

3. Escriba zoom y pulse ↵.
   Observe que los números de registro impresos en la parte inferior de la pantalla comienzan en 0 y siguen en secuencia hasta 13.

escribir cuenta()

4. Introduzca escribir cuenta() y pulse ↵.
ARCHIVE borra la pantalla y muestra el número 14, aunque de hecho había 16 registros en el fichero.

El comando elegir excluye todos los registros que no cumplen la condición; y ARCHIVE simplemente los ignora. Es como si nunca hubieran existido.

pantalla ↵
elegir accion$ = "t" ↵

5. Escriba pantalla y pulse ↵.

6. Escriba elegir accion$ = "t" y pulse ↵.

zoom ↵

7. Escriba zoom y pulse ↵.
Esta vez sólo quedan elegidos los registros que estaban seleccionados y tienen una t en el campo accion$. Quedan 9 registros numerados del 0 al 8.
La selección resultante se podría haber conseguido con una sola orden:

elegir fecha$<"05/07" yy accion$ = "t"

primero ↵

8. Utilice primero para volver al primer registro de la selección.

listagenda ↵

9. Escriba listagenda y pulse ↵.
Los registros seleccionados se listan línea a línea en el AREA DE VISUALIZACION:

```
notas$                                                fecha$ accion$
Pedirle a Felicidad que sea la madrina                04/05  t
Decirle a Juana que haga los trajes                   04/05  t
Buscar un Hotel para los invitados                    04/05  t
Decirle al Sr Hort que se encarque de las flores      04/05  t
Contratar a Pèrez Fotògrafos                          04/05  t
Coche                                                 04/05  t
Llamar a Sara ¿ Quièn les organizó el banquete ?      04/05  t
Capacidad de la sala Azul del Hotel Almirante         04/05  t
Encargar la tarta                                     04/05  t
```

Una vez se selecciona un fichero, los registros elegidos son los únicos registros accesibles hasta que el fichero se cierra o restaura.
Una selección que no incluya ningún registro hace todos los registros inaccesibles, como si no hubiera registros en el fichero.
Las selecciones afectan a la mayoría de los comandos, incluso a todos, pero no afectan a cerrar. Todos los regis-

tros originales se salvan al cartucho (si estaban abiertos con abrir) cuando se cierra un fichero seleccionado.
elegir hace que los registros que cumplen la condición se consideren lógicamente presentes. Aunque todos los registros queden físicamente en el archivo tras elegir, ARCHIVE se comporta como si pensase que sólo quedan los registros elegidos.

Restaurando el archivo

restaurar ↵
listagenda ↵

1. Escriba restaurar y pulse ↵.
2. Escriba listagenda y pulse ↵.
   Esta vez aparecen en la pantalla los 16 registros del fichero original. El orden puede no ser el mismo en el que se introdujeron (antes de seleccionar el archivo), pero todos deben estar ahí.

El comando restaurar tiene el efecto de cancelar cualquier selección previa.

Al revés que elegir, el comando restaurar no convierte en activo el primer registro del fichero.

Recuerde usar restaurar si escribió por error un comando elegir que no se cumple, ya que el fichero no tendrá ningún registro a partir de ese momento: las elecciones siguientes, corregido el error, fallarán al no quedar registros, como el resto de los comandos de ficheros.

> *Nota:* La visualización estándar mostrará el último registro del archivo si ocurre una selección fallida. Esto resulta equivoco, ya que de hecho no hay registros seleccionados en el fichero. La función cuenta( ) devuelve el valor 0.

Cerrar un fichero que se abrió con abrir tiene el efecto de restaurarlo. Todos los registros, no sólo los elegidos, se salvan de nuevo en la cinta.

> *Nota:* restaurar también cancela el efecto del comando ordenar, que se presenta en el próximo capítulo.

MAS SELECCIONES

Selección con YY

Eva quiere ver los nombres de los huéspedes que vienen a la boda desde Madrid. Una compañera de colegio vive allí y no tiene coche, y podria venir con alguien que lo tenga.

ver ↵ invitado"lógico "g ↵

1. Abra el fichero invitado con el comando ver.

indicar ↵ 2. Escriba el primer registro.

El fichero de invitados no puede revelar quién tiene o no un coche, ya que no se ha almacenado allí ese tipo de información; en lugar de ello, Eva selecciona a los que vivan en Madrid y vengan en un grupo de menos de cuatro. Puede entonces recordar los que es probable que vengan en coche.

La disposición de las distintas direcciones significa que la palabra Madrid no tiene por qué aparecer siempre en el mismo campo. Es probable que aparezca en una de las líneas de dirección a partir de la primera, asi que Eva opta por comprobar las líneas 2 y 3.

Operadores lógicos

Los operadores lógicos son no, yy y oo.

Los operadores lógicos son distintos que los de relación (=, <, >) y los aritméticos (+, —, *, /, ^) que ya hemos descrito.

no se llama un operador *monario,* ya que se utiliza igual que el signo menos. Por ejemplo, no hallado( ) y —6 son dos expresiones válidas, pero otros operadores (=, +, <, *, etc.) necesitan tener una expresión a cada lado.

no se puede utilizar en medio de expresiones, además de en su comienzo.

La palabra *lógicos* se refiere a un tipo de álgebra en que los resultados de las ecuaciones son siempre cierto o falso. Se llama álgebra de Boole, en honor de su inventor. La lógica booleana, que utiliza operadores como no, yy y oo, se utiliza para resolver esas ecuaciones.

reg cuantos<4 yy direc3$="MADRID" yy direc4$="MADRID"

3. Escriba la línea anterior.

Observe que el campo cuantos es numérico, de manera que el valor a comparar no está entre comillas.

⊡ **4.** Pulse ↵.
Se muestra el último registro del archivo, y no es un residente en Madrid.
La selección ha fallado, ya que la palabra MADRID aparece sólo una vez en la dirección. No puede estar a la vez en las dos líneas.
Para que un yy sea cierto, lo deben ser sus dos lados.

La palabra yy se llama un operador lógico. Como de costumbre, esto no tiene nada que ver con el sentido habitual de la palabra "lógico".

Selección con OO

restaurar ⊡ **1.** Escriba restaurar y pulse ↵.

elegir cuantos<4 yy direc3$ = "MADRID" oo direc4$ = "MADRID" ⊡ **2.** Escriba elegir cuantos<4 yy direc3$ = "MADRID" oo direc4$ = "MADRID" ↵ y pulse ↵ o use **F5** para editar la línea.
Basta que una mitad de una expresión oo sea cierta para que lo sea el total. Es decir, la expresión es cierta si (o bien) direc3$ es igual a MADRID OO (bien) direc4$ es igual a MADRID.
En este caso, si ninguna línea es igual a MADRID, el registro no se selecciona. Si lo hace, al menos una se elige el registro. Una dirección de Madrid aparece en pantalla:

```
Juan / Pepa Martinez Rebollo Majada del Viento, 25          28029 Madrid      M
ADRID                    s              2
```

zoom ⊡ **3.** Use zoom para ver el resto:

```
Juan Echarri                    Cia de Danza Moderna "Break out" Melancólicos, 35 2
8013 Madrid MADRID                   1
```

Recuerde que, si introdujo accidentalmente espacios al final de un campo de direcciones (por ejemplo, "MADRID "), los espacios no serán visibles, pero la selección se verá afectada por ellos y no aparecerán. Puede comprobar si hay espacios al final mediante subcadenas y usando long( ), por ejemplo:

```
buscar direc1$(long(direc1$)) = "  "
```

La función long( ) devuelve un número que representa la longitud de la cadena entre paréntesis. Por ejemplo:

escribir long("Hola") muestra un 4

COMPROBACION DE DATOS MEDIANTE EXPRESIONES LOGICAS

Muchas clases de datos (como, por ejemplo, fechas y códigos) necesitan introducirse con mucha precisión si queremos que sean de utilidad.

Podemos usar expresiones lógicas para comprobar que los datos son del tipo, longitud e intervalo correcto de valores. Si son incorrectos, se pide que se introduzca de nuevo.

Un procedimiento que comprueba intervalos

```
proc entraletra
  haz ok = 0
  mientras no ok
    leer "¿Inicial ?";inic$
    si inic$ >"A" yy inic$< = "Z" yy long(inic$) = 1
      haz ok = 1
      sino
      escribir "Inicial debe ser una mayúscula"
      finsi
    finmientras
  finproc
```

Este procedimiento itera hasta que se le introduce el dato correcto (una sola letra). La variable ok, puesta inicialmente a falso (0), se utiliza como un marcador que indica datos correctos o incorrectos. El bucle acaba cuando ok es cierto.

Una versión más compacta del mismo procedimiento usa el valor de la expresión misma (cierto o falso, 1 ó 0) para poner ok:

```
proc entraletra
  haz ok = 0
  mientras no ok
    leer "¿Inicial ?";inic$
    haz ok = (inic$ > = "A" yy inic$< = "Z" yy long (inic$) = 1)
    finmientras
  finproc
```

La versión final del procedimiento se deshace también de la variable ok: la misma variable del bucle se prepara con un valor incorrecto y el bucle sigue mientras esté equivocado:

```
proc entraletra
   haz inic$ = ""
   mientras inic$<"A" oo inic$>"Z" oo long(inic$)< >1
      leer "¿Inicial ?";inic$
      finmientras
   finproc
```

Dos posibilidades

Si los datos introducidos deben tener un valor a elegir entre dos posibles, recuerde usar oo:

```
si inic$ = "S" oo inic$ = "N"
    haz ok = 1
    finsi
```

Así pasa inic$ correcto si es igual a S o N. Si utilizase yy aquí, haría que se rechazara cualquier entrada, ya que un carácter no puede tener dos valores simultáneamente.

Observe que si le da la vuelta a la pregunta (y usa < > en lugar de =), debe cambiar también el operador lógico:

```
si inic$< >"S" yy inic$< >"N"
   haz ok = 1
   finsi
```

Así se pasa todo excepto S o N como correcto.

Varias posibilidades

Si hay varias posibilidades y no están en secuencia como un intervalo de valores, utilice la función enserie( ):

```
proc autorizar
   haz ok = 0
   mientras no ok
      leer "Iniciales del director: ";inic$
      haz inic$ = mayús(inic$)
```

```
    si enserie("JB/HS/AM/MD/SS",inic$) yy long(inic$) = 2
      haz ok = 1
      finsi
    finmientras
  finproc
```

Observe el uso de la barra vertical para que no se pasen como correctas combinaciones como BH, SA, MM, etc. (la última letra de una opción correcta junto a la primera de otra).

Para ser verdaderamente pedante debería excluir las barras de la variable leída con otra comprobación al comienzo:

```
si enserie (inic$,"/") yy ...
```

En el capítulo 11 se discuten otros métodos de hacer comprobaciones de datos si hay muchas introducciones correctas posibles.

Comprobación de fechas

Se puede usar una sola expresión compleja para comprobar una fecha:

```
proc compfecha
  haz pidefech = 1
  mientras pidefech
  leer "Fecha (AA/MM/DD) ";fech$
  haz a = val(fech$( hasta 2))
  haz m = val(fech$(3 hasta 5))
  haz d = val(fech$(7 hasta))
  haz pidefech = 0
  si (d>29 yy m = 2) oo (d>28 yy m = 2 yy y/4< >ent(y/4))
  oo (d>30 yy (m = 4 oo m = 6 oo m = 9 oo m = 11) oo
  m<1 oo m>12 oo d<1 oo d>31
    haz pidefech = 1
    finsi
  finmientras
finproc
```

La expresión se puede parafrasear como:

Si el día > 29 y el mes es febrero.
O el día > 28 y el mes es febrero y el año no es bisiesto (divisible por cuatro).
O el día > 30 y el mes es abril, junio, septiembre o noviembre.

O el mes < 1 o el mes > 12 o el día < 1 o el día > 31.
ENTONCES está mal, pide una nueva fecha.

Observe que el procedimiento utiliza también el marcador (pidefech en este caso) de una manera distinta. Esta vez el bucle finaliza cuando el marcador es falso. Se pone a falso en el bucle suponiendo que la entrada será correcta, y sólo se cambia cuando no lo es.

Inversión de la salida lógica

Se puede invertir completamente una expresión compleja sin alterar ningún operador de varias maneras.

- Usando marcadores de distintas maneras (como arriba).
- Usando paréntesis con no al frente.
- Insertando un sino al comienzo de la sentencia si.

Un ejemplo: Para invertir

```
si tecla$ = "X"
   stop
   finsi
```

añada una sentencia sino como sigue:

```
si tecla$ = "X"
   sino
   stop
   finsi
```

El último comando para sólo si tecla$ *no* es X, y equivale a:

```
si no (tecla$ = "X")
   stop
   finsi
```

Notas sobre la evaluación de expresiones lógicas

- Cuando evalúe expresiones como cierto o falso, ARCHIVE utiliza cada operador por turno y lo evalúa antes de pasar al siguiente.
- Cada operador tiene una prioridad en el proceso. ARCHIVE no se limita a comenzar con el primero de la línea.

- Los operadores aritméticos (*, /, +, —) vienen antes quc los operadores de relación (=, >, <), que, a su vez, vienen antes que los operadores lógicos (no, yy, oo).
- El operador no se evalúa antes que yy, y éste antes que oo. Evite problemas en cualquier caso poniendo los operadores entre paréntesis (así resulta más fácil leer las expresiones complejas).
- Las expresiones entre paréntesis tienen prioridad sobre las expresiones que no los llevan. Si hay paréntesis dentro de los paréntesis, se evalúa primero la expresión más interna.
- Los paréntesis tienen el efecto de eliminar las prioridades normales de los operadores.

Ejemplos detallados

Cuando se manejan selecciones complicadas en las que intervienen operadores lógicos y de relación, puede ayudar a escribir las expresiones y evaluarlas en abstracto, una operación cada vez. Compruebe las expresiones mediante datos medios, parte de los cuales debería fallar la prueba (si se ha escrito correctamente), y parte del cual debe pasarlo.

En los ejemplos siguientes observe cómo cada operador tiene conectadas dos expresiones. Cada pareja se evalúa por turno, de acuerdo con la prioridad del operador. El resultado es siempre cierto (mayor que cero) o falso (cero).

```
Datos ejemplo:
direc3$      :Madrid
direc4$      :
cuantos      :3
```

En todos los ejemplos siguientes, los operadores de relación (<, >, =) se evalúan en primer lugar, pero no se comentan para poner más énfasis en los operadores lógicos.

Ejemplo 1: OO

Evaluación de oo

```
direc3$ = "MADRID" oo direc4$ = "MADRID"
cierto          oo          falso
        = cierto
```

ARCHIVE pregunta: ¿Es al menos una de las ecuaciones cierta? Si lo es, la expresión entera es cierta.

Un lado, o ambos, de una expresión oo debe ser cierto para que la expresión sea cierta.

Ejemplo 2: YY

Evaluación de yy

```
direc3$ = "MADRID" yy direc4$ = "MADRID"
cierto       yy       falso
        = falso
```

ARCHIVE se pregunta: ¿Son ambas ecuaciones ciertas? En caso afirmativo, la expresión global es cierta. Recuerde que ambos lados deben ser ciertos para que la expresión evalúe como cierta.

Ejemplo 3: OO, YY

Evalúa primero yy y luego oo

```
direc3$ = "MADRID" oo direc4$ = "MADRID" yy cuantos<4
     falso          yy        cierto
     cierto         oo        = falso
     = cierto
```

ARCHIVE comienza por preguntarse: ¿Son las comparaciones a ambos lados del yy ciertas? Si lo son, la expresión yy entera es cierta.

A continuación ARCHIVE se pregunta si la expresión yy entera o la expresión al otro lado del oo son ciertas. Basta con que lo sea una para que lo sea el resultado total.

Ejemplo 4: ( ), OO, YY

Evalúa paréntesis ( ), luego yy y finalmente oo

```
(direc3$ = "MADRID" oo direc4$ = "MADRID") yy cuantos
     cierto          oo       falso
     = cierto        yy       cierto
     = cierto
```

ARCHIVE se hace primero la pregunta de si es alguna de las ecuaciones entre paréntesis cierta. Si lo es al menos una, la expresión oo es cierta.

La siguiente pregunta es si la expresión oo y la expresión del otro lado del yy son ambas ciertas. Si lo son, la expresión completa es cierta.

Recuerde que los paréntesis "le pueden" a las prioridades habituales. Si se coloca una expresión entre paréntesis, se evalúa antes que las expresiones exteriores.

Formas de evitar los operadores lógicos

Si encuentra que los operadores lógicos resultan confusos, hay maneras de evitar el problema.

Una técnica es usar elegir repetidamente, seleccionando sobre la última selección cada vez. No utilice restaurar entre las selecciones.

Otra manera es, por ejemplo, formar una cadena temporal con todas las líneas de dirección y buscar mediante enserie( ). Así ahorra también introducir la palabra clave, MADRID, varias veces. Por ejemplo:

```
elegir enserie(direc3$ + direc4$,"MADRID") yy cuantos<4
```

Así localiza cualquier ocurrencia de MADRID en la tercera y cuarta línea de direcciones si vienen menos de cuatro. Tiene también la ventaja de hallar la palabra si no es la única en la línea.

Recuerde que se pueden añadir textos para formar textos mayores. La función evalúa las dos líneas de direcciones como si fueran una sola cadena de texto. Por ejemplo, enserie("ColmenarMADRID","MADRID") es cierto.

Hay otra manera más rápida de hallar y agrupar registros, especialmente en ficheros con muchos registros. Se hace usando los comandos ordenar y situar, que son la materia del próximo capítulo.

Pongamos en orden la cinta

dir ↵ ↵

dir ↵ mdv1_ ↵

1. Haga un directorio de la unidad 2 para comprobar la fecha del fichero _eti.
2. Pida el directorio de la cinta de seguridad, en la unidad 1, para comprobar la fecha del fichero _eti y estar seguros de que es la cinta de copia correcta. Si la fecha coincide con la de la cinta de trabajo, use el otro cartucho.

nuevo ⏎

3. Borre todos los procedimientos y ficheros de datos de memoria temporal escribiendo nuevo y pulsando ⏎.

salvagrdr...

4. Haga copia de los ficheros creados o cambiados desde la última vez que usó la cinta.

salvar "Abr6 eti" ⏎ F5 ⇧⇨ ⇨ mdv1_ ⏎

5. Salve ficheros _eti en ambas cintas.

tirar...

6. Use tirar... para borrar los ficheros viejos de ambas cintas.

RESUMEN

Eva utiliza el comando buscar para encontrar las anotaciones del fichero agenda que dicen que tiene que llamar por teléfono (donde el campo accion$ es t). Escribe el procedimiento vermás para ver qué registro cumple la condición sin escribir continuar repetidamente. El procedimiento comienza usando un bucle sin fin, que es un bucle (por ejemplo, mientras/finmientras, todos/ fintodos) que siempre se cumple. En este caso, se le pide a ARCHIVE que siga usando el comando continuar mientras 1 = 1, es decir, siempre.

Eva mejora el procedimiento vermás utilizando la función hallado( ), que tiene dos valores, 1 ó 0. Será 1 si se halla el registro, y 0 en caso contrario. En vermás se usa con el bucle mientras, finmientras. vermás usa el comando continuar mientras que hallado( ) es igual a 1. El procedimiento termina tan pronto como se vuelve 0 (cuando no hay registros que cumplan la condición).

Tras esto, Eva utiliza subcadenas de texto para hallar notas fechadas en junio. La subcadena puede comparar texto en posiciones especificadas de un campo. La función enserie( ) halla texto en cualquier posición dentro de un campo. Eva sigue usando el comando elegir para limitar los registros disponibles a aquellos que cumplen ciertos criterios. Cuando se usa el comando elegir, sólo son accesibles para búsqueda o modificación los registros seleccionados. Para volver a tener acceso al resto de registros del fichero es preciso utilizar el comando restaurar.

El uso del comando elegir y el fichero invitado le permiten ahora a Eva saber qué huéspedes viven en Madrid y vienen menos de cuatro a la fiesta. Eva tiene que usar los operadores lógicos yy y oo para hacerlo.

Trucos útiles

- Si no parece tener el número correcto de registros en el fichero, puede haber olvidado restaurar tras el uso del comando elegir.
- Ponga cada par de valores entre paréntesis cuando use expresiones lógicas. Es más fácil de leer y asegura una evaluación correcta.
- Use validación de los datos que introduzca. Si comprueba los datos según se escriben, puede evitar errores que causarían problemas ocultos más adelante.
- La función enserie( ) se puede usar como sustituto del operador lógico oo en la validación de datos.

8

Ordenación de la información

Una de las tareas más aburridas que se ve obligado a realizar el género humano es la ordenación de fichas en secuencias específicas; por ejemplo, en orden alfabético.

La ordenación es vital para la recuperación y análisis rápido de grupos de registros similares.

Cuantos más registros haya, más dura la faena y mayor probabilidad de errores. En un gran sistema, un registro mal archivado puede quedar perdido para siempre. Todos los nuevos registros tienen que ordenarse antes de archivarlos eficientemente y, una vez hecho, nadie quiere cambiar el orden.

Si se necesita ver los registros en un orden distinto ocasionalmente, la regla general es establecer un sistema completo de referencias cruzadas, ya que volver a ordenar los registros es poco práctico, si no imposible, por regla general.

ARCHIVE ordena nuestros registros eficientemente, en tantas secuencias como queramos y con tanta frecuencia como sea necesario.

Este capítulo le muestra cómo debe poner los datos en un orden específico, y cómo recuperarlos con facilidad.

Se presentan las siguientes características de ARCHIVE:

- Comandos de ARCHIVE:

ordenar — El uso de ordenar para clasificar los registros en un orden determinado.
situar — El uso de situar para encontrar un registro en un fichero ordenado.

Puntos a recordar

- Use "claves compuestas" para evitar la penalización en memoria que sufriría si ordena por muchos niveles. Hágalo almacenando las claves como campos separados antes de ordenar.
- Una búsqueda indexada, usando situar con un fichero ordenado, es mucho más rápida que el uso de hallar o buscar.

Ejemplo: Poner en orden a los invitados

Preparación

Si no está cargado ARCHIVE, cárguelo y ponga su cartucho de datos en el *microdrive* 2.

cargar ↵ vermás ↵

1. Cargue el fichero de programa vermás, creado en el último capítulo.

ver ↵ agenda "lógico"a ↵

2. Abra el fichero agenda con el comando ver.

ver ↵ invitado "lógico"i ↵

3. Abra el fichero invitado con el comando ver.

ORDENACION Y COLOCACION

Uno de los mayores dolores de cabeza de Eva y Maya es la colocación de los invitados en sus asientos durante el banquete. ¿Cómo deben colocar a los huéspedes de manera que todos tengan cerca a alguien interesante para charlar?

Para cumplir ese propósito intentan hacer que nadie tenga cerca a ningún compañero de viaje.

Por tanto, sacan una lista, ordenada por el número de invitados, para poder simplificar la separación de los grandes grupos.

EL COMANDO ORDENAR

A veces se necesita ordenar información en secuencias específicas, sin importar el orden inicial.

Las guías de teléfonos están ordenadas, por regla general, por apellidos. Las sacas de correo que aguardan su reparto en la oficina de clasificación están ordenadas por casas, calles y barrios. Los diarios y calendarios guardan un orden temporal. Las listas de morosos de los banqueros pueden estar en orden de volumen de la deuda, con las más pequeñas en primer lugar. Las clasificaciones del fútbol están por puntos, con las más altas en primer lugar.

Esto es especialmente útil cuando se trata de leer listas impresas, ya que se pueden buscar los registros con rapidez. Si no aparecen en la lista donde deberían estar, no están en la lista (o están mal escritos y, por tanto, mal ordenados).

- **La misma información necesita, a veces, verse en más de un orden.**

La información contable se suele listar una vez por clientes y otra por fechas. Así es más fácil comprobar un pedido telefónico de un cliente, además de hacer más fácil el trabajo de un auditor.

Otros órdenes típicos permiten, por ejemplo, hallar un libro por autor o por título, o hallar un número telefónico por apellido o por profesión (como en las *Páginas Amarillas*).

La ordenación de registros tiene también el efecto de agrupar los registros parecidos juntos, haciendo más fácil responder a preguntas como: ¿cuántos libros tenemos sobre esta materia, o de tal autor?, ¿cuántos negocios hicimos con tal cliente tal día?

Aunque los comandos hallar, buscar y elegir ofrecen gran cantidad de maneras de acceder y seleccionar registros, ninguno de ellos los sitúa en una secuencia útil y reconocible.

Esto es especialmente interesante trabajando con ARCHIVE, ya que los registros se disponen en el orden más eficiente en términos de espacio, y no siempre estarán en la secuencia en que se insertaron. Su posición relativa puede cambiar tras comandos como elegir o restaurar.

El comando ordenar pone los registros en orden mediante uno o más campos.

restaurar ⊡ indicar ⊡

1. Restaure el fichero de invitados y véalo en pantalla si no está ahí.

ordenar ↵ cuantos;a

2. Escribir ordenar cuantos;a. Observe el punto y coma (;) tras el nombre de campo, y la letra a a continuación. Así ordenamos el fichero por los contenidos del campo cuantos, en orden ascendente. Esto quiere decir que los números menores quedan en primer lugar, y los mayores al final.

↵

3. Pulse ↵. La presentación cambia para mostrar un registro que tiene sólo una persona que viene de esa dirección.

zoom ↵

4. Use zoom para ver el resto del fichero, observando el campo cuantos y también el número de registros. Los registros deben aparecer en el orden siguiente:

```
nombre$                                    cuantos
Srta. Felisa de los Monteros                     1
Milena de Pablo Martinez                         1
Maribel Gutièrrez                                1
Lorenzo Sànchez Lòpez, crìtico teatral           1
Juan Echarri                                     1
Antonio Pèrez de las Heras y Sra.                2
Juan y Pepa Martinez Rebollo                     2
Felipe y Luisa Gonzàlez                          2
Luisa Sala y Carmen Garcìa Carriòn               2
Eduardo Rincòn Moreno                            3
Teatro Principal                                 3
Arturo y Teresa Lòpez Andiòn                     7
```

Cuando ARCHIVE ordena sobre un campo, el contenido de ese campo en registros sucesivos está siempre en orden ascendente o descendente. Sin embargo, entre los grupos de registros con el mismo valor en el campo de ordenación (por ejemplo, todos los registros con cuantos = 3) los registros pueden estar en un orden distinto al que se usó para introducirlos en el fichero.

> *Nota:* Algunas de las primeras versiones de ARCHIVE no ordenan correctamente los números negativos, que se sitúan tras los positivos en orden ascendente. Si tiene la versión española, no debe tener ningún problema.

Como el comando elegir, ordenar numera los registros desde 0 a partir del primero de acuerdo con la nueva secuencia, sin importar el número original.

- El comando restaurar deshace las ordenaciones del fichero, además de las selecciones.

Ordenación automática

Una vez ordenado un fichero, los registros añadidos con insertar o añadir se colocan automáticamente en la posición correcta. Esto es también válido si un campo CLAVE (un campo que se usó en el comando ordenar) se altera o actualiza.

Juan ve la factura

EL COMANDO SITUAR

El comando situar busca registros en un fichero ordenado.

Ordenación por campos numéricos

situar 3 ⏎

1. Escriba situar 3 y pulse ⏎.
Observe que el 3 no está entre comillas, ya que cuantos es un campo numérico.

Aparece el primer registro del fichero con una expedición de tres personas:

```
Eduardo Rincón Moreno Avda. de los Poetas, 12 03012 Barcelona BARCELONA
                3
```

Puntos a recordar

- situar sólo sirve para buscar en el contenido del campo que se usó para ordenar el archivo y, por tanto, no es necesario especificar el campo de nuevo.
- Como hallar y buscar, el comando situar busca la primera ocurrencia de un cierto valor. Como buscar, la comparación es específica en un determinado campo. Si el valor es una cadena literal, las mayúsculas y minúsculas se tratan de la misma manera.
- situar, al revés que buscar, busca sólo tantas letras como aparezcan en el comando; por ejemplo, buscar "s" hallaría Salva.

  Esto es cierto sólo para textos. En el caso de números, éstos deben coincidir enteramente.
- Si situar no encuentra un número o el comienzo de un texto, el registro siguiente en la secuencia se convierte en el registro activo. En orden ascendente, es el siguiente mayor. En orden descendente, es el siguiente menor.
- La función hallado( ) funciona con situar en ARCHIVE versión 2 de la misma manera que con hallar y buscar. Si se encuentra una coincidencia exacta, devuelve el valor 1.
- El comando continuar no funciona con situar. Para buscar más registros con el mismo comienzo se puede utilizar un bucle mientras-próximo. Estarán siempre inmediatamente detrás del registro hallado, ya que ordenar ha agrupado previamente todos los registros con valores parecidos en secuencia.

Para ver los registros siguientes con un 3 en cuantos:

```
mientras cuantos = 3: pescribir: próximo: finmientras ↵
```

1. Escriba la línea anterior y pulse ↵.

   Aparecen los registros siguientes hasta que aparece el primer cuantos mayor que 3. El primer registro que tiene cuantos mayor que 3 causa un fallo en la condición de bucle y se muestra automáticamente al acabar la línea.

   El comando situar es también distinto de hallar y buscar en que, si no encuentra una coincidencia exacta, encuen-

tra el registro que sigue a la posición donde hubiera estado el registro buscado.

ordenar ↵ cuantos;d ↵

2. Escriba ordenar cuantos;d y pulse ↵.
En este caso, la d quiere decir descendente, por oposición a ascendente.

zoom ↵

3. Use zoom para ver el resto del fichero.
Una vez más, los registros han cambiado de secuencia y de numeración, esta vez en orden decreciente, con el valor más alto en primer lugar. Los registros deben quedar en una secuencia parecida a ésta:

```
nombre$                                          cuantos
Arturo y Teresa Lòpez Andiòn                           7
Eduardo Rincòn Moreno                                  3
Teatro Principal                                       3
Antonio Pèrez de las Heras y Sra.                      2
Juan y Pepa Martinez Rebollo                           2
Felipe y Luisa Gonzàlez                                2
Luisa Sala y Carmen Garcia Carriòn                     2
Juan Echarri                                           1
Lorenzo Sànchez Lòpez, critico teatral                 1
Srta. Felisa de los Monteros                           1
Milena de Pablo Martinez                               1
Maribel Gutièrrez                                      1
```

En el fichero de invitados hay gente que viene a la boda en grupos de 1, 2, 3 y 7 personas. Pruebe los siguientes comandos:

situar 8 halla el primer registro, con 7 en el campo cuantos.
situar 5 halla el primer registro menor que 5, que es 3.
situar 0 busca el último registro, ya que no hay ninguno menor.

> *Nota:* Un comando situar que no encuentre una correspondencia exacta busca el siguiente valor mayor si el fichero está ordenado en orden ascendente.

El comando situar es mucho más rápido que hallar o buscar, especialmente con ficheros grandes, ya que estos últimos comienzan siempre en el primer registro y examinan todos los registros uno tras otro para buscar coincidencias, ya que el registro seleccionado podría estar en cualquier parte del fichero.

situar, en cambio, salta inmediatamente al lugar donde debe estar el fichero buscado. Lo puede hacer porque el archivo se ha

puesto en secuencia, y los registros que coinciden no pueden estar en cualquier parte del archivo. ARCHIVE sabe ya dónde debe buscar el registro.

Ordenación de archivos de texto

ordenar ↵ nombre$;a ↵

zoom ↵

1. Escriba ordenar nombre$;a y pulse ↵.
2. Use zoom para ver la secuencia.
   Los registros se ordenan en orden alfabético, pero no estrictamente por nombre, ya que la primera palabra en nombre$ puede ser un título como Sr. o Sra.:

```
nombre$                                  direc1$
Antonio Pérez de las Heras y Sra.        Teniente Coronel
Arturo y Teresa López Andión             Magnolias, 110
Eduardo Rincón Moreno                    Avda. de los Poetas, 12
Felipe y Luisa González                  Jardines Dorados, 12
Juan Echarri                             Cía de Danza Moderna "Break out"
Juan y Pepa Martínez Rebollo             Majada del Viento, 25
Lorenzo Sánchez López, crítico teatral   Revista del Escenario
Luisa Sala y Carmen García Carrión       Gran Vía, 34
Maribel Gutiérrez                        Paseo de las Acacias, 34
Milena de Pablo Martínez                 Practicante Díaz de Carreras, s.n.
Srta. Felisa de los Monteros             Vista Cielo
Teatro Principal                         Pza. Mayor, 3
```

Es más útil ordenar sobre un campo uniforme, como puede ser un apellido. Pero, para hacerlo, el apellido debe aparecer al comienzo de un campo. Mejor que introducir el apellido en primer lugar y luego el nombre, se puede conseguir una buena ordenación almacenando nombre y apellidos en campos separados. Para imprimir el nombre completo se pueden concatenar los campos.

Nota: ARCHIVE ordena sólo los primeros ocho caracteres de una cadena alfanumérica. *Cualquier carácter tras el octavo no afectará al orden.* Por ejemplo, dos empleados llamados "Martínez" de primer apellido pueden estar en orden incorrecto, si el segundo apellido no se usa como un campo separado de ordenación. Podría aparecer, por ejemplo, "Martínez Mediero" delante de "Martínez Asur".

Ordenación por fechas

ver agenda"lógico"a" ↵

1. Abra el fichero agenda si no lo está ya.

usar "a" ↵ indicar ↵

2. Use y muestre las anotaciones.

ordenar fecha$;a ↵

3. Escriba ordenar fecha$;a y pulse ↵.
Las anotaciones aparecen ahora en secuencia de fecha, siempre que introdujera las fechas al estilo de Estados Unidos, con el mes en primer lugar, como se explicó en el capítulo 5:

```
notas$                                              fecha$
Capacidad de la sala Azul del Hotel Almirante       04/05
Encargar la tarta                                   04/05
¿ Ha comprado Salva un traje de sport ?             04/05
Coche                                               04/05
Contratar a Pérez Fotógrafos                        04/05
Hacer una lista de bodas                            04/05
Pedirle a Felicidad que sea la madrina              04/05
Llamar a Sara ¿ Quién les organizó el banquete ?    04/05
Decirle a Juana que haga los trajes                 04/05
Decirle al Sr Hort que se encargue de las flores    04/05
Buscar un Hotel para los invitados                  04/05
Ver al sacerdote                                    04/08
Comprar el traje                                    04/14
Zapatos blancos                                     04/14
Que no se olvide hacer los ojales                   06/23
no olvidar aguja e hilo                             06/23
```

Nota: No se pueden utilizar subcadenas en el comando ordenar, así que no intente ordenar fechas introducidas a la española. Por ejemplo, la sentencia:

ordenar fecha$(4 hasta 5) + fecha$(1 hasta 2)

no funciona. Por esa razón, la fecha se debe almacenar con el mes en la primera posición. Si hay fechas de varios años, introduzca éste en primera posición.

situar "06" ↵

1. Escriba situar "06" y pulse ↵ para hallar la primera anotación fechada en junio:

   no olvidar aguja e hilo

situar "04/06" ↵

2. Escriba situar "04/06" para buscar el primer registro posterior al 5 de abril:

   Ver al sacerdote

> *Recuerde:* Se deben introducir exactamente los primeros ocho caracteres de los campos que quiera ordenar.
>
> Esto impide que se le pueda hacer a ARCHIVE ordenar campos creados con la función fecha( ). Esta función produce cadenas de fecha (incluyendo el siglo) de más de ocho caracteres, por ejemplo, 1985/04/05. Use subcadenas para eliminar los primeros dos caracteres antes de actualizar el campo, por ejemplo: haz fecha$ = fecha(0)(3 hasta): actualiz.

Ordene el fichero por el campo acción$:

ordenar accion$;a ↵

3. Escriba ordenar accion$;a y pulse ↵. Los registros aparecen ahora en secuencia, de acuerdo con el contenido del campo accion$.
Aparecen los campos vacios en primer lugar, seguidos por "c" de comprar y "t" de telefonear:

```
notas$                                                    accion$
no olvidar aguja e hilo
Hacer una lista de bodas
Ver al sacerdote
Que no se olvide hacer los ojales
¿ Ha comprado Salva un traje de sport ?
Zapatos blancos                                           c
Comprar el traje                                          c
Decirle al Sr Hort que se encargue de las flores          t
Buscar un Hotel para los invitados                        t
Llamar a Sara ¿ Quién les organizó el banquete ?          t
Capacidad de la sala Azul del Hotel Almirante             t
Encargar la tarta                                         t
Coche                                                     t
Decirle a Juana que haga los trajes                       t
Pedirle a Felicidad que sea la madrina                    t
Contratar a Pérez Fotógrafos                              t
```

situar "t" ↵

4. Escriba situar "t" y pulse ↵.
Aparece el primer campo acción$ que tiene una t. Use zoom para ver las demás llamadas telefónicas que debe hacer.

situar "S" ↵

5. Escriba situar "S" (observe la "S" mayúscula) y pulse ↵.
Aparece en pantalla el primer registro con t (versión 2) o el primer registro que no tiene el campo vacío (versión 1). Aquí aparece una importante diferencia entre las versiones anteriores y la versión española de ARCHIVE. Las versiones 1.XX ordenan los registros de acuerdo con el código ASCII, lo que hace que todas las mayúsculas

vayan delante de todas las minúsculas, y que las vocales acentuadas vayan detrás de la z.
En la versión 2.XX este orden no sólo se ha cambiado, sino que el usuario lo puede elegir mediante el programa config bas, presente en el cartucho de ABACUS. El orden estándar que se encuentra en la copia comercial de ARCHIVE ordena las letras sin tener en cuenta mayúsculas o acentos.
Observe que, en la versión 1, si usa situar con una letra minúscula y el fichero está ordenado por nombres, que suelen comenzar por mayúscula, los resultados pueden ser paradójicos. Por ejemplo, buscando "andrés" se puede encontrar "Zacarías".

Ordenación sobre más de un campo

Eva preferiría una secuencia más útil, que ordene las notas por fechas y, dentro de cada día, por código de acción.

ordenar fecha$;a,accion$;a ↵

1. Escriba ordenar fecha$;a,accion$;a y pulse ↵.
   Observe la coma que separa los dos campos, y que cada uno se ha marcado por separado para su ordenación ascendente. Se puede mezclar con libertad los campos numéricos y los literales, en orden ascendente o decreciente, hasta un total de cuatro campos, siempre que estén separados por comas.

zoom ↵

2. Escriba zoom y pulse ↵ para ver el resto de los registros, que deben quedar en el orden siguiente:

```
notas$                                                fecha$ accion$
¿ Ha comprado Salva un traje de sport ?               04/05
Hacer una lista de bodas                              04/05
Encargar la tarta                                     04/05  t
Capacidad de la sala Azul del Hotel Almirante         04/05  t
Coche                                                 04/05  t
Llamar a Sara ¿ Quién les organizó el banquete ?      04/05  t
Pedirle a Felicidad que sea la madrina                04/05  t
Decirle a Juana que haga los trajes                   04/05  t
Decirle al Sr Hort que se encargue de las flores      04/05  t
Buscar un Hotel para los invitados                    04/05  t
Contratar a Pérez Fotógrafos                          04/05  t
Ver al sacerdote                                      04/08
Zapatos blancos                                       04/14  c
Comprar el traje                                      04/14  c
Que no se olvide hacer los ojales                     06/23
no olvidar aguja e hilo                               06/23
```

Las anotaciones para el 5 de abril que no tienen código de acción quedan antes que las que tienen una "t" del mismo dia.

situar "04/05","t" ↵

3. Escriba situar "04/05","t" y pulse ↵ para localizar la primera nota del 5 de abril que requiere telefonear.

Ordenaciones secundarias

El comando situar se puede usar para señalar a un nivel secundario de ordenación dentro de uno principal, siempre que los valores que se deben localizar se correspondan con la secuencia de campos usada en ordenar. En 3, arriba, la fecha debe preceder al código de acción, ya que así se ordenó el fichero en 1.

No es necesario usar más de un campo con un comando situar en los ficheros que estén ordenados por más de uno. Se pueden localizar registros indicándole sólo el valor del primer campo.

Claves compuestas

No es muy buena idea ordenar sobre campos múltiples si tiene montones de registros o varios ficheros abiertos. Cada nivel de ordenación usa más memoria y reduce la cantidad disponible para otros ficheros. También limita el número máximo de registros que admite un fichero (véase los apéndices para una explicación más detallada).

Se puede crear una clave compuesta (de hasta ocho caracteres) duplicando partes de algunos campos en un campo clave especial (usando concatenación de subcadenas), por ejemplo:

```
haz clave$ = compañía( hasta 4) + apell$( hasta 4): actualiz:
ordenar clave$;a
```

Así se desperdicia algo de espacio, pero no se interfiere con el número máximo de registros ni el número máximo de ficheros abiertos.

Para salvar el orden

Si cierra un fichero ordenado y lo abrió con el comando abrir, el fichero se salvará en el cartucho en la secuencia ordenada. Cuando lo vuelva a abrir, estará de nuevo en el orden salvado, pero se puede volver a la secuencia primitiva con el comando restaurar.

La ordenación automática de los registros insertados o cambiados continúa si el fichero ordenado se ha salvado y vuelto a abrir.

Esto no es cierto si se cierra un fichero abierto con ver, ya que nada se salva de nuevo en el cartucho.

Cómo funciona ORDENAR

El comando ordenar usa un índice para hallar los registros. Este se almacena en memoria, aparte de los registros, y funciona exactamente igual que el índice de un libro.

Para encontrar cualquier mención al comando leer en este libro, podría buscar página por página (una búsqueda secuencial), pero tardaría mucho. En cambio, se tarda sólo un momento si se utiliza un índice. El índice contiene la palabra leer en orden alfabético, y lista los números de página que hay que consultar.

ARCHIVE hace exactamente lo mismo con los registros. Almacena los primeros ocho caracteres de los campos alfanuméricos en orden, junto con el lugar del archivo en que se puede encontrar el registro, en un índice. Por eso situar es tan rápido con grandes ficheros.

El precio que se paga por esa facilidad es que las entradas del índice gastan memoria, y ésta no es ilimitada. Cuantos más ficheros abiertos, o cuantos más niveles de ordenación, más espacio se malgasta.

Limpieza del cartucho

zoom prg ↵

1. Borre el programa zoom de la cinta, ya que ha sido actualizado por vermás.

dir . ↵ ↵

2. Haga un directorio de la unidad 2 para comprobar que tiene en ella los siguientes ficheros de programa: fácil_prg, FichBoda_prg, vacío prg, listados_prg y vermás_prg.
   Compruebe la fecha en el fichero _eti de la unidad 2.

mdv1_ ↵

3. Haga un directorio de la cinta de seguridad, en la unidad 1, para comprobar mediante la fecha del fichero eti que es la cinta de seguridad correcta. Si la fecha coincide con el cartucho de trabajo, debe utilizar la otra.

nuevo ↵

4. Borre todos los procedimientos y ficheros de datos de memoria temporal escribiendo nuevo y pulsando ↵.

salvagrdr ...

5. Copie los ficheros creados o modificados desde la última vez que usó la cinta de seguridad.

salvar "Abr6_eti" ↵ F5 ⇧⇨ ⇨ mdv1_ ↵

6. Salve los nuevos ficheros de etiqueta con la fecha de hoy en la cinta de trabajo y la cinta de seguridad de la unidad 1.

tirar ... 7. Tire los viejos ficheros _eti de ambos cartuchos.

RESUMEN

Eva prueba el comando ordenar, usándolo para poner los registros en varias secuencias. Primero pone el fichero de invitados por orden, según el tamaño de la "expedición" (cuántos). Poniendo un ;a tras el nombre del campo por el que se va a ordenar, el fichero se ordena en orden ascendente. Una d forzaría el orden decreciente.

Eva usa entonces el comando situar para encontrar los huéspedes con un número específico en el campo cuántos. situar sólo puede usarse en un fichero que ha sido ordenado, y sólo en el campo que se usó para ordenar.

El segundo campo que Eva intenta ordenar es fecha$. Lo puede hacer porque introdujo las fechas de la forma que la escriben en Estados Unidos (con el mes delante). Tras esto, utiliza el campo accion$ para ordenar los registros. Todos los registros con una cadena vacía en el campo vienen antes que los que llevan una c, que a su vez vienen antes que los que llevan t.

ordenar se puede usar con campos numéricos o de texto, y se puede usar con más de un campo, así que Eva acaba por ordenar el fichero sobre los dos campos, fecha$ y accion$, de manera que pueda ver los grupos de tareas que tiene que hacer cada día.

Trucos útiles

- Almacene las fechas como en Estados Unidos: año, mes, día (sin las dos primeras cifras del año); así podrá ordenar por orden alfabético las fechas, y el resultado corresponderá también con el orden cronológico.

9

Mantenimiento de registros

La mayor parte de las aplicaciones de bases de datos repiten las mismas pequeñas tareas una y otra vez. Estos trabajos son específicos del fichero o tarea, pero suelen ser muy parecidos. Una biblioteca, por ejemplo, envía recordatorios sobre los libros cuyo préstamo ha vencido, y los hombres de negocios los envían sobre retrasos en los pagos.

Todos los ficheros necesitan, en un momento u otro, que se le añadan o eliminen registros, que éstos se ordenen o cuenten, que se impriman o se listen en pantalla. Estas tres áreas se suelen citar en la jerga informática como introducción, proceso de datos y salida de información.

Los beneficios reales de ARCHIVE aparecen cuando disponemos de todos los procedimientos específicos para un trabajo particular, simultáneamente y con el mínimo esfuerzo. Este tipo de programa debe ser adaptable a cualquier archivo o tarea en su parte esencial.

En este capítulo se construye un programa que maneja las funciones de búsqueda, edición y actualización que se suelen necesitar cuando se mantienen registros.

Se presentan las siguientes características de ARCHIVE:

- Comandos de ARCHIVE:

error	— Atrapa errores sin detener la ejecución del programa.
regreso	— Hace que un procedimiento se pare y devuelva el control al programa que lo llamó.
sino	— Se usa con si para ofrecer una alternativa.

- Funciones de ARCHIVE:

tecla( )	— Hace que ARCHIVE espere a que se pulse una tecla.
numerr( )	— Devuelve el número del último error.
rept( )	— Repite una cadena un número determinado de veces.
serie( )	— Convierte una expresión numérica en una cadena.

Ejemplo: Rogamos confirmen asistencia

Comienzan a llegar respuestas a las invitaciones de boda, y Eva quiere actualizar el fichero invitado para tener una idea clara de las cifras.

Considerando cada contestación por turno, primero tiene que hallar y luego alterar los registros para actualizar los campos vienen$ y cuantos.

Ya tiene unos cuantos procedimientos para realizar algunas de las tareas requeridas, pero quiere unirlos para facilitar la tarea.

Tiene la idea de utilizar a las gemelas para facilitarle parte del trabajo. Necesita que el programa sea informativo para las gemelas y seguro para los datos.

En lugar de introducir comandos directamente en la línea de comandos, ARCHIVE ejecutará varios procedimientos en respuesta a la pulsación de letras sueltas.

Comienzo

1. Cargue ARCHIVE, si aún no está en memoria.

abrir ↵ invitado"lógico"i ↵

2. Abrir el fichero invitado con el nombre lógico i.

indicar ↵

3. Mostrar el fichero en la pantalla.

PROCEDIMIENTO PARA ANOTAR A LOS HUESPEDES ESPERADOS

Antes de crear ningún procedimiento, Eva se plantea qué es lo que quiere que hagan. Comienza por escribir sus planes en una hoja de papel, en términos generales, y los traduce a continuación a una aproximación a los términos y pasos habituales de ARCHIVE.

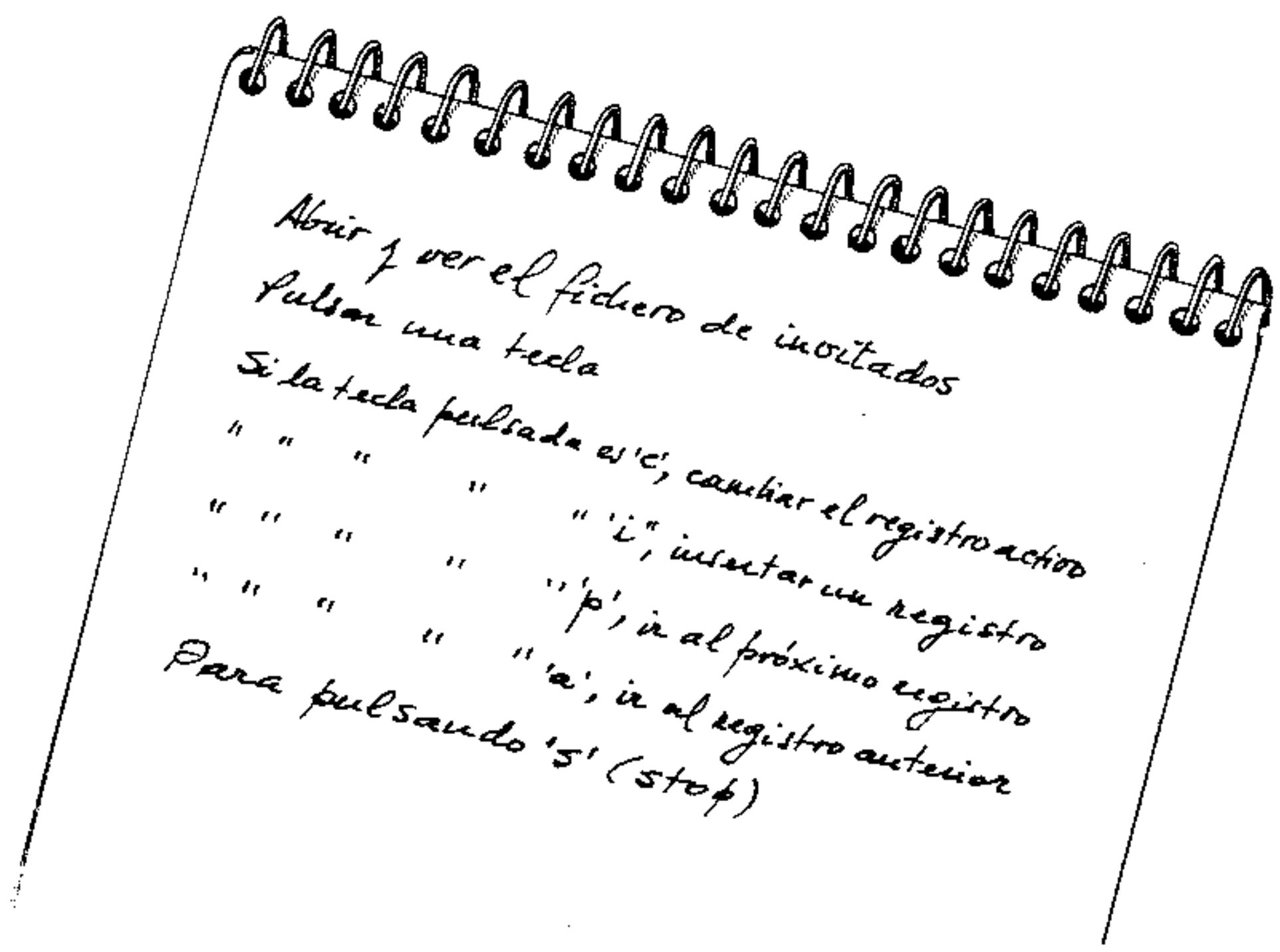


Figura 9.1. Análisis paso a paso

Decide no abrir y cerrar los ficheros cada vez que se usa el programa, ya que puede dejarlo para realizar otra tarea con los ficheros, y no quiere perder tiempo.

El esquema que ha escrito ya se parece mucho a algunos de los comandos de ARCHIVE, o cambia para que quede:

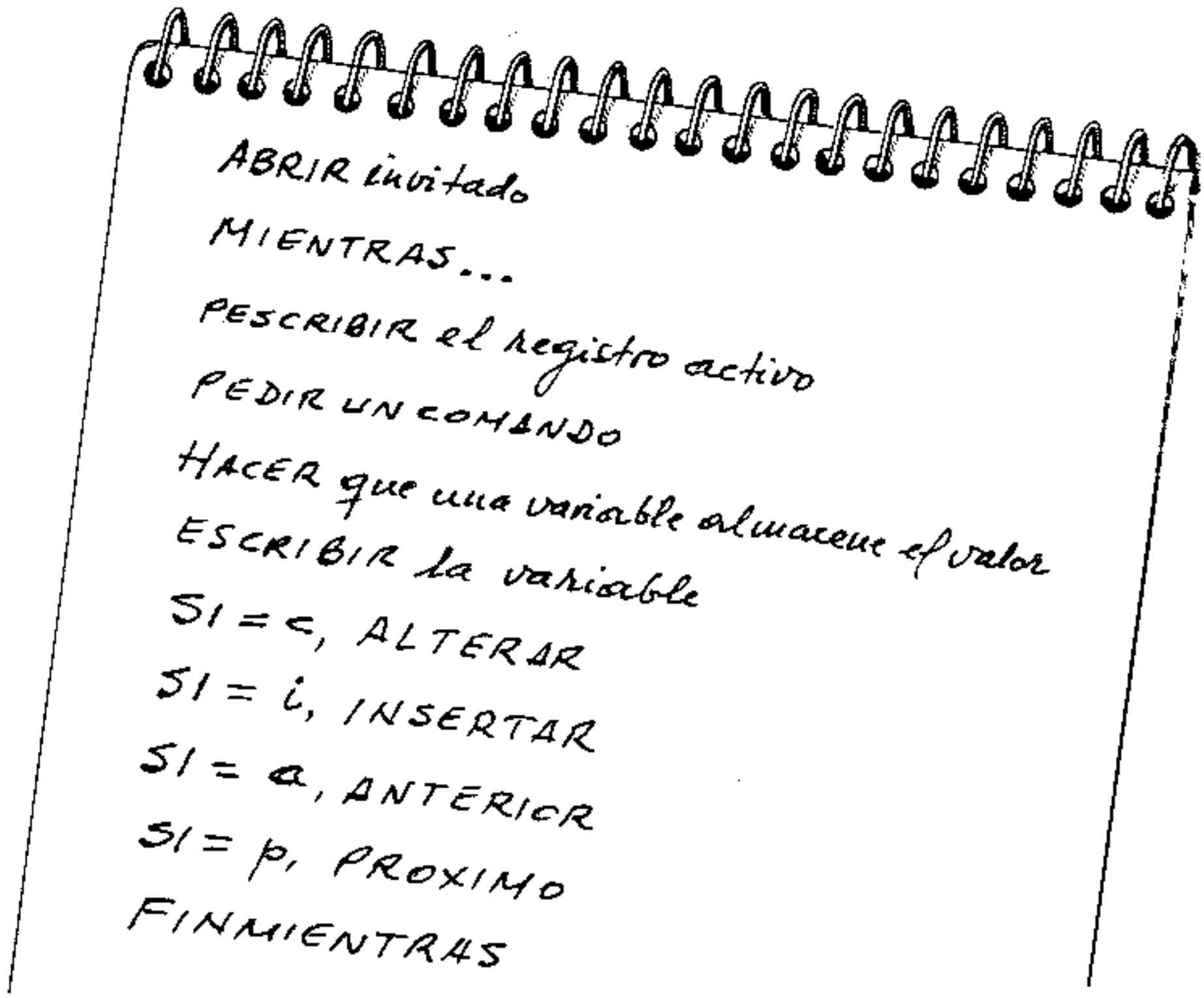


Figura 9.2. Un análisis más formal

Eva sabe que necesitará más opciones, y decide dividir la tarea en dos procedimientos para más claridad.

Comienza a trabajar sobre los procedimientos para mostrar los registros y pedir la tecla. Aquí es donde volverá, antes o después, el resto de los procedimientos del fichero.

Un procedimiento que lee un comando de una letra

editar ↵ pidecom ↵

1. Utilice el editor para crear un nuevo procedimiento llamado pidecom.

mientras 1 ↵ escribir en 12,5; "¿Comando?: "; ↵ haz c$ = tecla( ) ↵
escribir c$ ↵

2. Escriba las líneas siguientes:

```
mientras 1
escribir en 12,5;"¿Comando?: ";
```

Observe el punto y coma al final de la línea escribir, que sirve para separar los dos elementos de impresión. El elemento de impresión en se puede usar para posicionar el cursor directamente, con los comandos escribir y leer.

```
haz c$ = tecla( )
```

La línea haz c$ = tecla( ) espera a que se pulse una tecla, y almacena el carácter en la variable c$.

El nombre de la variable c$ se ha elegido por ser abreviatura de comando, y es corto para ahorrar al escribir. También ahorra algo de memoria en los procedimientos.

```
escribir c$
```

Al revés que leer, tecla( ) no muestra en pantalla la tecla pulsada. Es útil verla para saber si hemos pulsado un comando incorrecto. Así nadie se quedará preguntándose por qué no funciona el programa.

mientras ↵

↵

ESC

3. Finalice el procedimiento cerrando el bucle mientras.
4. Pulse ↵ para dejar el modo de inserción y, a continuación...
5. Salga del editor.

La función TECLA( )

La función tecla( ) hace esperar al procedimiento que la llama hasta que se pulsa una tecla. La función devuelve el valor de la tecla pulsada, evaluado como un carácter. Es decir, devuelve el valor "A" cuando se pulsa ⇑ y A, y "9" (como una cadena de caracteres) si se pulsa 9.

Es distinta de leer en que no hace falta pulsar la tecla ↵ para finalizar la introducción, y el carácter no aparece en la pantalla si no se usa escribir.

escribir tecla( ) ↵

1. Introduzca escribir tecla( ) y pulse ↵.
   El cursor del AREA DE TRABAJO espera sobre la petición ">". El comando no finaliza hasta que se pulsa una tecla.

(*excepto* ESC)

2. Pulse cualquier tecla, numérica o de letra. La tecla o número se imprime en la pantalla, y ARCHIVE espera el siguiente comando.
   No pulse **ESC** cuando utilice tecla( ), o el procedimiento que ha hecho la llamada se detendrá. Cualquier otra tecla, sin embargo, hará que continúe el procedimiento.

Prueba del procedimiento PIDECOM

trazar ↵
pidecom ↵

1. Conecte la *traza* de programas.

2. Escriba pidecom y pulse ↵.
El mensaje ¿Comando? : aparece en la línea 12, cinco columnas a la derecha.

< > (*excepto* ESC)

3. Pulse cualquier tecla excepto **ESC**.
El carácter aparece tras el mensaje, y el cursor sobre él. Al pulsar una tecla hacemos que el carácter aparezca inmediatamente detrás de la petición de comando.
Si pulsamos otra, aparecerá en el mismo sitio.
Observe que el cursor en el AREA DE VISUALIZACION no sigue el mensaje de petición. No indica dónde aparecerá el carácter siguiente si se usa tecla( ).

ESC

4. Pulse **ESC** para salir del procedimiento y...

trazar ↵

5. Desconecte la *traza*.

Está claro que el procedimiento no sirve para nada como está, y hace falta otro procedimiento para tomar alguna decisión en función de la tecla pulsada.

Los nombres de procedimiento

Los nombres de procedimiento usados en este capítulo pueden parecer un poco raros, pero son el resultado de un compromiso entre varios requerimientos:

- Los procedimientos que comienzan por la misma letra aparecen juntos en el editor, y los procedimientos que manejan la misma tarea se pueden referenciar más fácilmente.
- Los nombres de procedimiento deben recordar lo que hace el procedimiento.
- Los nombres de procedimiento deben ser tan cortos como sea posible, para ahorrar espacio en memoria.

Un procedimiento que toma decisiones

editar ↵ F3 n compcom ↵

1. Cree un nuevo procedimiento llamado compcom.

si c$ = "S": stop:finsi ↵ si c$ = "P":próximo:finsi ↵ si c$ = "A": anterior:finsi ↵
si c$ = "C":alterar: finsi ↵ si c$ = "I": insertar: finsi ↵ ESC

2. Inserte la línea siguiente:

```
si c$ = "S":stop:finsi
```

Si se pulsa la tecla S, el procedimiento se detendrá.

```
si c$ = "P":próximo:finsi
si c$ = "A":anterior:finsi
```

Si se pulsan las teclas P o A, se realiza próximo o anterior. Tan pronto como acaban, el bucle en pidecom vuelve a comenzar, y ARCHIVE espera otra tecla.

```
si c$ = "C":alterar:finsi
si c$ = "I":insertar:finsi
```

Los comandos alterar e insertar llamados mediante C e I permiten cambiar registros de la misma manera que si se hubiesen tecleado directamente. Se usan los nombres completos (en lugar de las abreviaturas), ya que funcionan más rápido. Las TECLAS DE FUNCION que se utilizan normalmente para salvar o abandonar registros funcionan como siempre. El AREA DE CONTROL sigue cambiando sus mensajes.
Si se pulsa cualquier tecla distinta de las indicadas, el bucle devuelve al procedimiento al comando tecla( ) de nuevo.

El procedimiento debe quedar como sigue:

```
proc compcom
    si c$ = "S": stop: finsi
    si c$ = "P": próximo: finsi
    si c$ = "A": anterior: finsi
    si c$ = "C": alterar: finsi
    si c$ = "I": insertar: finsi
    finproc
```

ESC salvar ↵ BodaCom ↵

3. Abandone el editor y salve los procedimientos en un fichero de programas llamado BodaCom. Es una abreviatura de "comandos de la boda", para distinguirlos de bodafich, que contenía los procedimientos para la gestión del archivo.

Modificaciones del procedimiento PIDECOM

Muchos de los comandos anteriores cambian el registro activo o lo modifican. Es útil, por tanto, presentar en pantalla el

registro activo cada vez que ARCHIVE vuelve dc compcom y ejecuta el bucle en pidecom para obtener otro comando.

A veces el registro actual no será distinto, pero un pescribir superfluo es preferible a salpicar nuestro programa de sentencias, y posiblemente olvidarlas en algunos sitios.

Incluya pescribir en pidecom:

editar ↵ TAB ↓ F4 pescribir ↵

1. Edite pidecom.
2. Añada pescribir tras la línea mientras.
   Así, la pantalla mostrará el registro activo cada vez que comience el bucle.
   Recuerde que la presentación de registros no es automática si se está ejecutando un procedimiento. pescribir también hace que el último registro modificado se muestre al acabar insertar.

↓ ↓ F5 ⇧⇨ ⇨ *(3 veces)* mayús (↓) ↵

3. Cambie la sentencia haz para que quede:

```
haz c$ = mayús(tecla( ))
```

   Así se convierte cualquier entrada en minúsculas a mayúsculas. Cualquier referencia futura a c$ puede suponer con seguridad que no está en mayúsculas.

↓ F4 compcom ↵ ESC

4. Inserte la línea compcom antes del finmientras.

El procedimiento queda como sigue:

```
proc pidecom
    mientras 1
    pescribir
    escribir en 12,5;  "¿Comando?:  ";
    haz c$ = mayús(tecla( ))
    escribir c$
    compcom
    finmientras
    finproc
```

ESC

5. Deje el editor y
6. Una los procedimientos fácil, ya que algunos de ellos se utilizarán en compcom.

unir ↵ fácil ↵

editar ↵ TAB *(3 veces)* F3 b ↵ TAB F3 b ↵ ESC

7. Borre los procedimientos b y e, ya que no harán falta en este procedimiento, y salga del editor.

Menús

Los procedimientos y programas que ofrecen una serie de opciones que se eligen con una sola tecla se llaman normalmente *menús*. Igual que el menú de un restaurante, se elige de entre lo que hay disponible.

Los menús de ordenador actúan como una plataforma donde todas las opciones comienzan, y donde todas deben terminar. Son un punto de referencia útil y estable.

Uso de PIDECOM con el fichero de **invitados**

indicar ↵

1. Muestre el registro activo del fichero invitado.

pidecom ↵

2. Escriba pidecom y pulse ↵.
Aparece la petición: ¿Comando?:

p

3. Pulse p.
La letra aparece tras el mensaje, y se muestra el siguiente registro.

pp

4. Pulse p dos veces rápidamente.
Aparece el siguiente registro, y luego otro después.
Las teclas pulsadas se almacenan en una memoria temporal, y se guardan para su procesamiento posterior (versión 2).
Téngalo en cuenta y espere hasta que un comando acabe antes de pulsar la tecla siguiente, ya que, si no, puede llenar la memoria intermedia de comandos sin sentido.
De momento no es fácil, ya que la última letra que se usó permanece en la pantalla. No se puede decir en qué momento está el procedimiento listo para una nueva tecla.

s

5. Pulse s para detener el procedimiento.

Un cursor para TECLA( )

Cambie pidecom de manera que el carácter anterior se borre al finalizar el comando.

editar ↵ TAB *(5 veces)*

1. Use editar y seleccione el procedimiento pidecom.

↓ *(3 veces)* F5 ↓ papel 6;" " ↵

2. Añada un espacio en blanco al mensaje de petición de la primera sentencia escribir, para que quede así:

```
escribir en 12,5;"¿Comando?: "; papel 6;" "
```

Este espacio actúa como un cursor que muestra que se espera un comando. Será sobreimpreso por el siguiente comando que se pulse.

⇩ ⇩ F5 ⇧⇨ en 12,16; ↵ ESC

3. Cambie la línea escribir c$ para que quede:

```
escribir en 12,16;c$
```

Los caracteres se imprimen en la pantalla tras el último carácter impreso. Así, el carácter almacenado en c$ se imprimiría tras (en vez de sobre) el nuevo "cursor", a menos que se lo indiquemos (con el elemento de impresión en).
Comenzando por la apertura de interrogación ¿ en la columna 5, el cursor está en la decimosexta posición, así que la nueva tecla se imprime sobre él.
Si no lo hace, probablemente no habrá copiado el mensaje de nuestro ejemplo literalmente. No se preocupe, limítese a ajustar la columna en para que encaje con su propio mensaje.

El procedimiento queda ahora como sigue:

```
proc pidecom
  mientras 1
    pescribir
    escribir en 12,5;"¿Comando?: "; papel 6;" "
    haz c$=mayús(tecla())
    escribir en 12,16;c$
    compcom
    finmientras
  finproc
```

ESC **4.** Abandone el editor.

pidecom **5.** Pruebe de nuevo pidecom.
Esta vez reaparece el espacio en blanco tan pronto como acaba un comando.
Pulse una letra de comando cuando esté presente el espacio blanco. Si no lo está, aguarde a que finalice el último comando.

Para evitar que los errores detengan un programa

i **1.** Pulse i.
La visualización muestra un registro vacío.

ESC

2. Pulse **ESC** para abandonar insertar.
   No sólo se detiene insertar, sino que los procedimientos pidecom y compcom se detienen también.

editar ↵ TAB TAB

3. Use el editor para cambiar compcom.

⇩ *(5 veces)* F4 ⇧⇨ ⇧⇨ CTRL ⇩ error i: finsi ↵

4. Cambie el comando insertar para que la línea quede: si c$ = "|": error i: finsi

El procedimiento queda como sigue:

```
proc compcom
    si c$ = "S": stop: finsi
    si c$ = "P": próximo: finsi
    si c$ = "A": anterior: finsi
    si c$ = "C": alterar: finsi :nota cambiar
    si c$ = "I": error i: finsi
finproc
```

ESC

5. Abandone el editor.

pidecom ↵

6. Pruebe de nuevo pidecom.

i

7. Pulse i.

ESC

8. Pulse **ESC** para dejar de insertar.
   Esta vez el procedimiento no se detiene.
   Observe que vuelve a escribir el registro activo.

> *Nota:* En la versión 1 de ARCHIVE, la única manera de dejar de insertar es pulsar la tecla **ESC**. No se puede usar la tecla **F4**. Esta no hace que se detengan los programas.

s

9. Pulse s para detener el procedimiento.

salvar ↵ BodaCom1 ↵

10. Salve los procedimientos en el fichero de programa BodaCom1.
    Contiene ahora los procedimientos a, c, compcom, i, p, pidecom.

Diagramas de flujo

Todos los profesores de programación repiten una y otra vez que se escriben mejores programas y procedimientos si se analiza el "flujo" del programa mediante un *diagrama de flujo*. Un diagrama de flujo es un dibujo en el que se utilizan formas geo-

métricas y flechas para indicar los pasos y conexiones de un programa. Casi todos los programadores le dirán que jamás han perdido el tiempo haciendo uno, pero en esencia el análisis "paso a paso" de Eva es muy parecido a un diagrama de flujo. Su propósito es hacer al programador considerar cada paso por turno.

Si la idea le atrae, busque un libro que explique el método en detalle.

El comando ERROR

El comando error absorbe los errores que ocurran en el procedimiento que le sigue. Si el procedimiento se detiene por alguna razón que causaría normalmente un error o mensaje ‹**ESC**›, el procedimiento se detendrá, pero sin escribir ningún mensaje de error, y el procedimiento que lo llamó no detendrá su ejecución.

El comando error amortigua el procedimiento que lo usa de los efectos de un error en los procedimientos que llame.

En el ejemplo anterior, si el procedimiento i se detiene por un fallo en el comando insertar, el procedimiento que lo llamó (con la línea error i) no se parará.

error sólo funciona con procedimientos, sin embargo, y no puede usarse con los comandos de ARCHIVE. Por eso tuvimos que usar el procedimiento i, de fácil_prg. La línea error insertar es incorrecta, y causaría ella misma un error.

Un procedimiento que cierra todos los ficheros abiertos

editar ↵ F3 n cerrartodo ↵ mientras 1 ↵ cerrar ↵ finmientras ↵ ESC

1. Cree el procedimiento siguiente:

```
proc cerrartodo
        mientras 1
            cerrar
            finmientras
        finproc
```

ESC **2.** Salga del editor.

Probando CERRARTODO

cerrartodo ↵ **1.** Escriba cerrartodo y pulse ↵. Se cierra el fichero invitado, salvándose al cartucho, y se muestra el mensaje.

Error 94: fichero no abierto. El procedimiento falla automáticamente tan pronto como no hay más ficheros abiertos que cerrar.
El procedimiento sigue en el bucle, cerrando ficheros abiertos hasta que no hay más, y el procedimiento se detiene a causa del error.

La función NUMERR( )

error cerrartodo ⏎

1. Escriba error cerrartodo y pulse ⏎.
No hay ficheros que cerrar, así que el procedimiento se detiene inmediatamente, pero no causa un mensaje de error. Si la línea error cerrartodo se hubiera usado en un procedimiento, el procedimiento hubiera seguido sin interrupción.

escribir numerr( ) ⏎

2. Escriba escribir numerr( ) y pulse ⏎.
Aparece el número 94 en el AREA DE VISUALIZACIÓN. Es el número del error que hizo que se detuviera cerrartodo, aunque no se escribió ningún mensaje de error.
La función numerr( ) devuelve el número del último error, o 0 si el procedimiento usado con el comando error funcionó (es decir, no produjo ningún error).

La función NUMERR( )

Cada vez que ocurre un error, la función numerr( ) devuelve el número del error.

Normalmente los errores hacen detenerse la ejecución de procedimientos y comandos, con la impresión de un mensaje de error en el AREA DE TRABAJO. El comando error actúa de otra manera.

numerr( ) se usa, sobre todo, para comprobar si hubo error tras la llamada de un procedimiento con el comando error. Si numerr( ) no se utiliza, no hay manera de saber si se ha producido un error. No hay indicación de que ha ocurrido un error si éstos se filtran mediante error.

La función numerr( ) devuelve el valor 0 si no ocurrió error en una función llamada por error.

La tecla **ESC** hace que numerr( ) devuelva 27. Observe que este valor es el mismo que el código ASCII de escape.

Indicación de errores

No es buena idea utilizar indiscriminadamente error para evitar los errores, ya que éstos cumplen una misión muy útil.

Por ejemplo, si utiliza alterar con un fichero que se abrió con ver, no sabrá al pulsar **F5** que el registro no se ha grabado correctamente, ya que ARCHIVE no permite escribir en un fichero abierto sólo para lectura, como ocurre cuando se abren con ver.

Si el comando se utilizó en un procedimiento llamado con error, podría no darse cuenta de que el comando alterar no tuvo ningún efecto.

De la misma manera, cerrartodo podría fallar al intentar salvar a cinta un fichero de datos abierto, pero no dará ninguna advertencia.

La naturaleza de un error se puede comprobar utilizando el comando si, y escribir el mensaje adecuado, tomar las medidas oportunas. Por ejemplo:

editar ↵ F3 n comperr ↵ si numerr( ) > 0 ↵ escribir en 14,5; tinta 2; papel 6;"Error ";numerr( ) ↵ finsi ↵ ESC

1. Cree el procedimiento siguiente:

```
proc comperr
    si numerr( ) > 0
        escribir en 14,5; tinta 2; papel 6;"Error ";numerr( )
        finsi
    finproc
```

⇧TAB *(4 veces)* F5 ⇧⇨ ⇧⇨ CTRL ⇩ error a: comperr: finsi ↵

2. Añada el comando error, y el procedimiento comperr a la línea que llama al procedimiento a en compcom.

⇩ F5 ⇧⇨ *(4 veces)* compcom: ↵

3. Inserte la palabra compcom en la línea que llama a i.

Ambas líneas deben quedar como sigue:

```
si c$ = "A": error a: comperr: finsi
si c$ = "I": error i: comperr: finsi
```

ESC 4. Abandone el editor.

Uso de ERROR y NUMERR()

- Use siempre error y numerr() juntas.
- Llame primero a un procedimiento usando error, y, cuando finalice el procedimiento, compruebe el numerr() resultante.
- Recuerde que siempre es una sentencia de ARCHIVE la que causa un problema, pero sólo se puede llamar con error a un procedimiento que contenga el comando, y no al propio comando.
- Un error causa la parada del procedimiento que la contiene. Este detiene a su vez al procedimiento que lo llamó, y éste a su vez al que lo llamó, y así sucesivamente.
  El proceso sigue hasta que se alcanza la línea que se introdujo en el teclado, o hasta que se llega a un procedimiento que llamó al siguiente con el comando error. Así se atrapa el error, y se evita que continúe la cadena de paradas.

Los ejemplos siguientes muestran el efecto del uso de error en posiciones diferentes de tres procedimientos que se llaman sucesivamente por turno, con una llamada final a un procedimiento que contiene un error.

Ejemplo 1:

```
proc1          proc2          proc3          proc4
  proc2          proc3          proc4          ecribir "Hola"
  escribir "1"   escribir "2"   escribir "3"   escribir "4"
  finproc        finproc        finproc        finproc
```

Si escribe proc1 en la línea de comandos hará que aparezca el mensaje de error Error 1: Comando no reconocido. Es debido a que escribir está incorrectamente escrito como escribir en proc4.

Ejemplo 2:

```
proc1          proc2          proc3          proc4
  proc2          proc3          error proc4    ecribir "Hola"
  escribir "1"   escribir "2"   escribir "3"   escribir "4"
  finproc        finproc        finproc        finproc
```

Si escribe proc1 en la línea de comandos se atrapa el error en proc3. Sólo proc4 se detendrá. El AREA DE VISUALIZACION mostrará:

```
3
2
1
```

```
Ejemplo 3:
proc1          proc2          proc3          proc4
  error proc2    proc3          proc4          ecribir "Hola"
  escribir "1"   escribir "2"   escribir "3"   escribir "4"
  finproc        finproc        finproc        finproc
```

Si ahora escribe proc1 en la línea de comandos, el error se atrapará en proc1. Los procedimientos 2, 3 y 4 se detendrán. El AREA DE VISUALIZACION muestra sólo un 1.

> *Nota:* Las líneas que siguen a un procedimiento o comando fallido no se ejecutan. ARCHIVE vuelve inmediatamente al procedimiento que lo llamó.

Creación de un nuevo procedimiento inicial

Eva quiere crear un procedimiento inicial (llamado start), que abra el fichero invitado y muestre el primer registro.

Este procedimiento sustituye a los procedimientos, muy rudimentarios, que creó antes (capítulo 4). Incluye todos los comandos que se usan sólo una vez al comienzo del programa.

editar ↵ F3 n start ↵

1. Use editar para crear el procedimiento start.

limpiar ↵

2. Escriba limpiar para borrar la pantalla antes de comenzar los nuevos programas.

escribir en 5,10; tinta 2; papel 6; "Abriendo fichero invitados..." ↵

3. Siempre es útil que quien esté usando un programa sepa qué está ocurriendo. El mensaje disminuye también las preocupaciones sobre el tiempo que tardan los *microdrives* en hacer su trabajo. Observe el uso de colores para resaltar el mensaje.

error cerrartodo ↵

4. Escriba error cerrartodo y pulse ↵.
Así cierra cualquier fichero que haya podido quedar abierto accidentalmente.

abrir "invitado"lógico "i" ↵ indicar ↵

5. Escriba abrir "invitado"lógico "i" y pulse ↵. Se usa indicar aquí porque sólo hace falta usarlo una vez en el programa.

pidecom ↵

6. Escriba pidecom.
Así se llama al procedimiento menú, que será en adelante la raíz de todos los procedimientos. Si el programa se detiene, Eva puede volver a arrancarlo usando pidecom en

vez de start, que gasta un montón de tiempo usando los *microdrives* para abrir y cerrar ficheros.

El procedimiento queda como sigue:

```
proc start
    limpiar
    escribir en 5,10; tinta 2; papel 6; "Abriendo fichero
        Invitados..."
    error cerrartodo
    abrir "invitado"lógico "i"
    indicar
    pidecom
    finproc
```

El núcleo del procedimiento son los comandos abrir, indicar y pidecom, pero los demás tienen también funciones importantes.

Recuerde que el comando ejecutar busca un procedimiento llamado start en el programa que carga.

Identificación de programas y procedimientos

F3 n a1 ↵ nota BODACOM_PRG 5 de Abril de 1985 ↵ ESC

1. Cree el procedimiento **a1**, con la fecha y nombre del fichero de programa en él. Use el comando **nota** al comienzo de la línea, para indicarle a ARCHIVE que no trate lo que sigue en la misma línea como un comando, sino que lo ignore.
   El procedimiento debe quedar como sigue:

```
proc a1
  nota COMBODA_PRG 5 de Abril 1985
finproc
```

Así se identifica al programa, y la fecha de la versión. El procedimiento se llama a1 para que aparezca cerca de la parte superior de la lista de procedimientos en editar, y sea más fácil referirse a él.

Añada otros comentarios para recordar qué procedimientos están relacionados, y lo que hacen.

```
nota pidecom: menú, pantalla, lee tecla c$
nota compcom: llamada pidecom, comprueba c$
nota i: llamada por compcom, (error) insertar
nota comperr: escribe un mensaje con numerr()
```

Quizá crea ahora que no necesita que le recuerden cómo funcionan sus programas. Puede olvidarlo pronto, ya que los programas se escriben de distinta manera a medida que pasa el tiempo.

Este procedimiento guía o índice ayudará también a cualquiera que necesite usar o modificar sus procedimientos si usted no está a mano para explicarle su funcionamiento y cómo están relacionados.

Leer una lista de comentarios es más fácil que repasar uno a uno los procedimientos.

Los comentarios se escriben a menudo en los procedimientos a los que se refieren, pero así se hace más lenta su ejecución, sobre todo en los bucles.

> *Nota:* Cuando hay muchos procedimientos en un programa, el procedimiento a1, como el procedimiento ayuda del que hablaremos más adelante, se hacen muy largos, y desperdician espacio en memoria temporal.

La solución es guardar dos versiones del programa final en cinta. Una totalmente documentada (con, por ejemplo, la extensión _not), y la otra, más compacta, como copia de trabajo.

ESC

2. Abandone el editor.

salvar "BodaCom2" ↵

3. Salve los procedimientos como BodaCom2.

Este programa contiene los procedimientos a1, start, pidecom, compcom, cerrartodo, comperr, y los procedimientos a, c, i, y p de fácil.

BUSCAR A LA PERSONA ADECUADA

Eva sigue ahora dispuesta a crear un procedimiento que busque gente en el fichero de invitados. Antes de cambiar un registro debe localizarlo, y próximo es una manera un tanto lenta de hacerlo.

unir ↵ vermás ↵

1. Una el procedimiento vermás.

editar

2. Use el editor.

TAB *(6 veces)* F3 b ↵ TAB F3 b ↵

3. Borre los procedimientos listagenda y escreg, que no serán necesarios.

F3 n b ↵

4. Cree un procedimiento llamado b (abreviatura de buscar).

leer en 13,5; tinta 4; "¿Buscar?: ";txt$

5. Esta línea pregunta qué letras hay que buscar, y las almacena en txt$.

si txt$>"" ↵

6. Si sólo se pulsa ↵, la variable txt$ será una cadena nula, es decir, vacía. En ese caso, el procedimiento supone que no quiere buscar ningún registro y devuelve control a pidecom.

hallar txt$ ↵ si hallado() ↵

7. si hallado() es una segunda sentencia si (totalmente incluida en la primera) que comprueba el resultado de hallar.

pescribir ↵ vermás ↵

8. Así se llama al procedimiento que usa continuar mientras los registros coincidan, ya que el primero podría no ser el correcto.

sino

9. El comando sino forma parte de la sentencia si, a la que sigue, y proporciona un camino alternativo para las ocasiones en que falla la expresión lógica de si. En este ejemplo, esto sucede cuando hallado() devuelve un 0, es decir, no es cierto.

escribir en 14,5; papel 6; tinta 2; txt$ + " no hallado" ↵

10. Esta línea imprime un mensaje si no se halla la palabra.

finsi ↵ finsi ↵

11. Acabe el procedimiento cerrando las dos sentencias si. El procedimiento debe quedar como sigue:

```
proc b
  leer en 13,5; tinta 4;"¿Buscar?: ";txt$
  si txt$>""
    hallar txt$
    si hallado()
      pescribir
      vermàs
      sino
      escribir en 14,5; papel 6; tinta 2;txt$+" no hallado"
      finsi
    finsi
finproc
```

Sentencias SI anidadas

Observe que se ha usado una sentencia si dentro de otra. La segunda se ejecutará sólo si la primera es cierta. En la jerga informática a esto se le llama si anidados. Las sentencias si se pue-

den anidar hasta cualquier profundidad, es decir, se pueden tener tantos comandos si dentro de si..., como se desee.

Asegúrese de posicionar los finsi correctamente cuando utilice varios si juntos. Recuerde comprobar que finproc está indentada dos espacios. Asi comprueba que el número de comandos fin... es correcto.

El comando SINO

La palabra sino es realmente parte de la sentencia si ... finsi. Cualquier comando que siga a sino, hasta finsi, se ejecuta si la expresión que acompaña a si es falsa.

Su uso se puede parafrasear como: "si esto es cierto, hacer esto; sino (en caso contrario) hacer los comandos hasta finsi".

Adición de nuevas opciones a COMPCOM

TAB *(3 veces)* ⇩ F4 si c$ = "B":b: finsi ↵ ESC

12. Añada la línea siguiente como una de las sentencias si de compcom:

```
si c$ = "B":b: finsi
```

El procedimiento queda ahora como sigue:

```
proc compcom
    si c$ = "S": stop: finsi
    si c$ = "B":b: finsi
    si c$ = "P": próximo: finsi
    si c$ = "A": anterior: finsi
    si c$ = "C": alterar: finsi: nota cambiar
    si c$ = "I": error i: comperr: finsi
finproc
```

ESC **13.** Abandone el editor y pruebe los nuevos comandos.

Modificación del procedimiento VERMAS

El procedimiento vermás sigue usando el comando continuar hasta que no hay más coincidencias.

La única manera de parar en el registro que desee es pulsar **ESC**, un poco cuestión de puntería si el listado pasa muy rápido.

Cámbielo para que espere a que se pulse una tecla antes de seguir.

editar ↵ TAB *(12 veces)* ⇩ F4 escribir en 13,5; tinta 4; "Pulse cualquier tecla para ver más, o P para parar" ↵ si mayús(tecla()) = "P" ↵ regreso ↵

1. Añada las siguientes líneas al procedimiento vermás tras el comando mientras:

```
escribir en 13,5; tinta 4; "Pulse cualquier tecla para ver
    más, o P para parar"
si mayús(tecla()) = "P"
regreso
```

Este comando devuelve control inmediatamente a b, ignorando las líneas siguientes.

sino

2. sino.
Si se pulsa otra tecla que no sea P, se ejecuta el comando tras sino, y vermás sigue su curso.

⇩ ⇩ F4 finsi ↵ ESC

3. Añada un finsi antes del finmientras.

⇩ F4 escribir en 14,5; papel 6; tinta 2; "No encuentro más " + txt$

4. Ponga una linea de advertencia al final del procedimiento para mostrar que no hay más registros que cumplan la condición.

```
proc vermás
  mientras hallado()
    escribir en 13,5; tinta 4;"Pulse cualquier tecla para ver más, o P para para
r"
    si mayús(tecla())="P"
      regreso
      sino
      continuar
      pescribir
      finsi
    finmientras
  escribir en 14,5; papel 6; tinta 2;"No encuentro más "+txt$
  finproc
```

ESC

5. Abandone el editor.

El comando REGRESO

El comando regreso hace que el procedimiento en el que se usa se detenga inmediatamente, y devuelva el control al procedimiento que lo llamó. Un procedimiento normalmente devuelve el control sólo cuando ha finalizado.

Si el procedimiento se llamó desde la línea de comandos, la

petición de comando > aparece de nuevo, lista para otro comando.

regreso no es como stop, que hace que se detenga el programa entero y todos sus procedimientos.

Uso del programa para buscar algunos nombres

start ↵

1. Use el comando start para ver la ejecución del programa.

b

2. Pulse b.
   El mensaje "¿Buscar?: " aparece en la línea inferior, con el cursor esperando a continuación.

Felisa ↵

3. Escriba Felisa y pulse ↵.
   La pantalla cambia para mostrar Felisa de los Monteros, y aparece el mensaje "Pulse cualquier tecla para ver más, o P para parar".

(*cualquier tecla que no sea* P)

4. Pulse cualquier tecla excepto P.
   Aparece el mensaje "No encuentro más Felisa", ya que no hay más registros con las letras "Felisa" en ellos.

b

5. Pulse b de nuevo.

Fred ↵

6. Escriba Fred y pulse ↵.
   Aparece el mensaje "No encuentro más Fred", ya que ningún registro contiene las letras fred.

P

7. Pulse P para detener el procedimiento.

Un procedimiento que borra las líneas de la pantalla

Hay ahora tantos mensajes en la pantalla que Eva empieza a encontrar difícil leer lo que está ocurriendo, y no sabe qué se supone qué debe hacer a continuación. Decide añadir un procedimiento que borre las líneas que ya no son necesarias de la pantalla.

editar ↵ F3 n blanco;x,y ↵

1. Use editar para crear un procedimiento que borre una línea. El procedimiento se llama blanco y usa dos parámetros, x e y:

   ```
   proc blanco;x,y
   ```

   El procedimiento necesita que se le pasen dos parámetros numéricos que determinan qué línea hace falta borrar, y comenzando en qué columna.
   El nombre del procedimiento se separa de los parámetros

x e y por un punto y coma, y éstos se separan uno de otro por una coma.

escribir en x,y;rept(" ",64−y+1); ↵

2. Observe el punto y coma al final de la sentencia escribir. Sin él, el procedimiento haría que la pantalla suba una línea si se usa para borrar la línea inferior del AREA DE VISUALIZACION.
La función rept() repite el espacio " " y se usa con escribir para escribir espacios sobre el resto de la línea, es decir, (64−y+1) espacios.

↵ 3. Abandone el modo de inserción.

El procedimiento queda como sigue:

```
proc blanco;x,y
    escribir en x,y;rept(" ",64−y+1);
    finproc
```

La función REPT()

La función rept() contiene dos elementos de información entre los paréntesis, separados por una coma. El primero es el carácter o caracteres que se deben repetir, y el segundo es el número de veces que se debe repetir.

Por ejemplo, escribir rept("x",20) muestra:

xxxxxxxxxxxxxxxxxxxx

Las variables x e y se usan para posicionar el cursor en la fila x y la columna y. Así la función rept() escribe 64 (la anchura de la pantalla en modo TV) espacios menos el número de columna en que empieza la impresión. Si la impresión comienza en la columna 5, entonces hay 5—1 (es decir, 4) espacios delante. Por tanto, se deben borrar 64—4, o 60, columnas.

En otras palabras, cuando se le pasan dos números como parámetros, blanco borra la línea x desde la columna y hasta el final. Si quiere usar el procedimiento con otra anchura de pantalla, por ejemplo, 40, sustituya 64 por 40.

Inserción de BLANCO en los demás procedimientos

TAB *(7 veces)* 1. Use el tabulador hasta el procedimiento pidecom.

⇩ ⇩ F4 blanco;13,5 ↵ ESC

2. Inserte una línea justo tras la línea pescribir del procedimiento pidecom:

```
blanco;13,5
```

Así se borra desde la columna 5 al final de la línea cada vez que acaba un comando, se use o no la línea 13. Los mensajes de error también necesitan borrarse cuando se carga un nuevo comando en pidecom.

⇩ ⇩ F4 blanco;14,5 ↵ ESC

3. Inserte la línea anterior tras la linea tecla() en el procedimiento pidecom:

```
blanco;14,5
```

El procedimiento queda ahora como sigue:

```
proc pidecom
  mientras 1
    escribir
    blanco;13,5
    escribir en 12,5;"¿Comando?: "; papel 6;" "
    haz c$=mayús(tecla())
    blanco;14,5
    escribir en 12,16;c$
    compcom
    finmientras
finproc
```

Observe que los mensajes o preguntas necesarios durante la ejecución de un comando (por ejemplo, b o vermás) aparecen en la línea 13, y las advertencias o mensajes de error aparecen en la línea 14.

Las preguntas y los mensajes se deben borrar tan pronto como finaliza el comando, y antes de que se pulse la siguiente tecla. Los mensajes de error, en cambio, deben permanecer hasta que se pulse la siguiente tecla, para que permanezcan en la pantalla suficiente tiempo para ser leídos.

ESC

4. Abandone el editor.

salvar ↵ BodaCom3 ↵

5. Salve los procedimientos a BodaCom3.
BodaCom3 contiene ahora los procedimientos:

a	i
a1	p
b	pidecom
blanco	start
cerrartodo	vermás
compcom	zoom
comperr	

pidecom ↵

6. Escriba pidecom y pulse ↵.

b

7. Pulse b.

↵

8. Cuando se le pida que introduzca los caracteres que se van a buscar, pulse simplemente ↵.
La pregunta ¿Buscar?: se borra y el procedimiento espera una nueva tecla de comando.

s

9. Salga de pidecom pulsando S.

La desprendida familia Alcoy

Preguntas y advertencias estándar

Eva se da cuenta de que probablemente necesitará mensajes, preguntas y advertencias en muchos otros procedimientos, y escribir las líneas completas desperdicia espacio y tiempo.

También, si decide cambiar la posición de los mensajes en la pantalla, o su color, tendrá que cambiar todas las líneas de preguntas o mensajes de todos los procedimientos.

Crea dos procedimientos para posicionar y dar el color de las preguntas y advertencias. Se le pasará el mensaje relevante como un parámetro cada vez que se usen.

editar ↵ F3 n pregunta;mens$ ↵ escribir en 13,5; tinta 4;mens$ ↵ ESC

1. Cree el procedimiento pregunta como sigue:

```
proc pregunta;mens$
    escribir en 13,5; tinta 4;mens$;
    finproc
```

F3 n advierte;mens$ ↵ escribir en 14,5; papel 6; tinta 2;" ";mens$; " " ↵ ESC

2. Crear el procedimiento advierte;mens$ como sigue:

```
proc advierte;mens$
    escribir en 14,5; papel 6; tinta 2;" ";mens$;" "
    finproc
```

Observe los espacios literales a ambos lados del mensaje. Así se mejora la legibilidad del mensaje si se imprime en un fondo blanco.

TAB *(hasta* vermás*)* ⇩ ⇩ F5 CTRL ⇨ *(25 veces)* pregunta ↵ ⇩
(8 veces) F5 ⇧⇨ *(6 veces)* ⇨ CTRL ⇧ advierte ↵

3. Adapte vermás para que use los dos nuevos procedimientos.

La línea:

escribir en 13,5; tinta 4;"Pulse cualquier tecla para ver más, o P para parar"

se convierte en:

pregunta;"Pulse cualquier tecla para ver más, o P para parar".

La línea

escribir en 14,5; papel 6; tinta 2;"No encuentro más " + txt$

se convierte en:

advierte:"No encuentro más " + txt$

El procedimiento queda como sigue:

```
proc vermás
    mientras hallado()
        pregunta;"Pulse cualquier tecla para ver más, o P para
            parar"
        si mayús(tecla()) = "P"
            regreso
            sino
            continuar
            pescribir
            finsi
        finmientras
    advierte;"No encuentro más " + txt$
    finproc
```

⇧TAB *(11 veces)* ⇩ F5 ⇧⇨ *(4 veces)* ⇨ CTRL⇧ pregunta ⇧⇨ ⇨
CTRL⇩ ↵ F4 leer txt$ ↵ ESC ⇩ *(7 veces)* F5 ⇧⇨ *(6 veces)* ⇨
CTRL⇧ advierte ↵

4. Adapte el procedimiento b, para que la línea:

leer en 13,5; tinta 4;"¿Buscar?:";txt$

se convierta en

pregunta;"¿Buscar?:"

y añada la línea leer txt$

Cambie la línea:

escribir en 14,5; papel 6; tinta 2; txt$ + " no hallado"

para que quede:

advierte;txt$ + " no hallado"

ESC **5.** Abandone el editor.

El procedimiento queda ahora como sigue:

```
proc b
    pregunta;"¿Buscar?:"
    leer txt$
    si txt$>""
        hallar txt$
        si hallado()
```

```
                pescribir
                vermás
                sino
                advierte;txt$ + " no hallado"
                finsi
            finsi
        finproc
```

Conversión de números a series de caracteres

Eva descubre que tiene un ligero problema con el procedimiento comperr. La función numerr() devuelve un resultado numérico. Quiere usar el procedimiento advierte para indicar el número de error, pero sólo acepta cadenas alfanuméricas como parámetros.

Debe convertir en primer lugar el número de error en una cadena alfanumérica. Para hacerlo usa la función serie().

La función SERIE()

editar ↵ TAB *(8 veces)* ↓ F5 ↓ yy numerr()< 27 ↵ ↓ F5
⇧⇨ *(6 veces)* ⇨ CTRL ⇧ advierte ⇧⇨ CTRL ⇨ + serie(↓,3,0) ↵

1. Adapte comperr para que la linea:

```
si numerr()>0
```

se convierta en:

```
si numerr()>0 yy numerr()<>27
```

y la linea:

```
escribir en 14,5; tinta 2; papel 6;"Error ";numerr()
```

se convierta en:

```
advierte;"Error " + serie(numerr(),3,0)
```

ESC **2.** Abandone el editor.

El procedimiento debe quedar como sigue:

```
proc comperr
    si numerr()  0 yy numerr()   27
```

```
    advierte;"Error " + serie(numerr(),3,0)
    finsi
finproc
```

La función serie() requiere que aparezcan entre los paréntesis tres valores separados por comas. Deben ser los tres expresiones numéricas.

Los valores que se le pasan a una función para definir su tarea se llaman *argumentos* en jerga informática. Son como los parámetros pasados a los procedimientos.

Con serie(), sólo el primero es el número que se va a convertir en una cadena de caracteres.

Los otros dos controlan la manera en que se va a representar el número (por ejemplo, si se trata de un número entero, o con cifras decimales), y el número de cifras decimales, si se requieren.

Los tres argumentos deben estar presentes, aunque no tengan ninguna utilidad.

En este caso, Eva utiliza un 3 en el segundo parámetro, que indica la representación en "formato general", es decir, como se imprimiría en pantalla.

Usa un 0 en el segundo argumento, que indica las cifras decimales, ya que no se requiere en la representación general (que las muestra si las hay, y tantas como sean necesarias). Se trata de un argumento *inútil*.

Los argumentos que se pueden usar con serie(), junto a su significado, se explican más detalladamente en el capítulo 12, y en la sección de *Palabras Clave* al final de este libro.

Uso de procedimientos para introducir el número de invitados

Eva tiene ya bastantes opciones de comando disponibles para comenzar a actualizar los registros de huéspedes que han contestado. Sería más simple, sin embargo, si (en vez de pulsar a para alterar y pasar 5 campos para llegar a cuantos cada vez) pudiera pulsar directamente un número en el campo cuantos.

editar ↵ F3 n vienen ↵ pregunta;"¿Cuántos vienen?: " ↵ leer cuantos ↵ haz vienen$ = "s" ↵ actualiz ↵ ESC

1. Utilice editar para crear un procedimiento vienen como sigue:

```
proc vienen
    pregunta;"¿Cuántos vienen?: "
    leer cuantos
```

```
haz vienen$ = "s"
actualiz
finproc
```

El procedimiento espera que se escriba un número, y actualiza a continuación el campo cuantos del registro activo con ese número, poniendo simultáneamente una "s" de si en el campo vienen$.

⇧TAB *(8 veces)* ⇩ F4 si c$ = "V":vienen:finsi ↵ ESC

2. Añada otra opción a compcom:

```
si c$ = "V":vienen: finsi
```

El procedimiento queda ahora como sigue:

```
proc compcom
    si c$ = "S": stop: finsi
    si c$ = "V":vienen: finsi
    si c$ = "B":b: finsi
    si c$ = "P": próximo: finsi
    si c$ = "A": anterior: finsi
    si c$ = "C": error c: finsi: nota cambiar
    si c$ = "I": error i: finsi
    finproc
```

ESC 3. Salga del editor.

Introduciendo la primera confirmación

Eva tiene la respuesta de Lorenzo Sánchez López, crítico teatral de la *Revista del Escenario,* que está encantado de poder asistir a la boda. Para ejecutar el programa:

pidecom ↵ 1. Escriba pidecom y pulse ↵.

b 2. Pulse b.
Aparece la pregunta.

lorenzo ↵ 3. Escriba lorenzo y pulse ↵.
La pantalla cambia para mostrar el registro de Lorenzo Sánchez López.

p 4. Pulse p para evitar que vermás encuentre más *"Lorenzos"* si los hubiera.

v 5. Cuando se le pregunte el número de asistentes:

1 ↵ 6. Escriba 1 y pulse ↵.
La presentación cambia para mostrar los nuevos contenidos de los campos cuantos y vienen$, y pidecom se prepara para otra tecla de comando.

s 7. Pulse s para salir de pidecom.

Un procedimiento que ayuda a identificar las letras de comando

Eva está satisfecha de momento con los resultados del programa BodaCom, pero decide añadirle una pantalla de ayuda para recordarse (y recordarles a las gemelas si consigue que hagan algo) qué teclas pulsar.

editar ↵ TAB *(11 veces)* ⇩ *(4 veces)* F5 ⇩ ; papel 0;"(Pulse ? para obtener ayuda)" ↵

1. Cambie pidecom para que la línea de petición de comando quede así:

```
escribir en 12,5;"¿Comando?: "; papel 6;" "; papel 0;
    "(Pulse ? para obtener ayuda)"
```

El procedimiento queda ahora como sigue:

```
proc pidecom
  mientras 1
    pescribir
    blanco;13,5
    escribir en 12,5;"¿Comando?: "; papel 6;" "; papel 0;"(Pulse ? para obtener
ayuda)"
    haz c$=mayus(tecla())
    blanco;14,5
    escribir en 12,16;c$
    compcom
    finmientras
  finproc
```

⇧ TAB *(4 veces)* F4 si c$ = "?": ayuda: finsi ↵ ESC

2. Añada un nuevo si en compcom para llamar al procedimiento ayuda cada vez que se pulse "?".

```
si c$ = "?":ayuda: finsi
```

El procedimiento queda ahora como sigue:

```
proc compcom
    si c$ = " ?":ayuda: finsi
    si c$ = "S": stop: finsi
    si c$ = "V":vienen: finsi
    si c$ = "B":b: finsi
    si c$ = "P": próximo: finsi
    si c$ = "A": anterior: finsi
    si c$ = "C": error c: finsi: nota cambiar
    si c$ = " I ": error i: finsi
    finproc
```

3. Cree el procedimiento ayuda como sigue:

```
proc ayuda
  limpiar
  escribir "PULSE UNA DE LAS TECLAS SIGUIENTES PARA ..."
  escribir "A  volver al registro Anterior"
  escribir "P  ir al Próximo registro"
  escribir "C  Cambiar un registro en pantalla"
  escribir "I  Insertar un nuevo registro"
  escribir "B  Buscar registros"
  escribir "V  marcar una respuesta, y decir cuàntos Vienen"
  escribir "S  dejarlo (Stop)"
  pantalla
  finproc
```

4. Actualice el procedimiento a1 para que incluya los nuevos procedimientos. Abrevie la frase llamado por mediante ().
5. Abandone el editor.

ESC
salvar ↵ BodaCom ↵ salvar ↵ mdv1 BodaCom ↵

6. Salve los procedimientos en bodacom, y de nuevo en el mdv1 también, tras comprobar que está usando la cinta de seguridad correcta mediante la etiqueta de cinta.
7. Haga una copia de seguridad extra y etiquétela "para ejercicios". La necesitará en seguida.

BodaCom contiene ahora los procedimientos siguientes:

```
proc a
  anterior
  finproc
proc a1
  nota COMBODA_PRG 5 de Abril 1985
  nota pidecom: menú, pantalla, lee tecla c$
  nota compcom: llamada pidecom, comprueba c$
  nota i: llamada por compcom, (error) insertar
  nota comperr: escribe un mensaje con numerr()
  finproc
proc advierte;mens$
  escribir en 14,5; papel 6; tinta 2;" ";mens$;" "
  finproc
proc ayuda
  limpiar
  escribir "PULSE UNA DE LAS TECLAS SIGUIENTES PARA ..."
  escribir "A  volver al registro Anterior"
  escribir "P  ir al Próximo registro"
  escribir "C  Cambiar un registro en pantalla"
  escribir "I  Insertar un nuevo registro"
  escribir "B  Buscar registros"
  escribir "V  marcar una respuesta, y decir cuàntos Vienen"
  escribir "S  dejarlo (Stop)"
  pantalla
  finproc
```

```
proc b
  pregunta;"¿Buscar?: "
  leer txt$
  si txt$>""
    hallar txt$
    si hallado()
      pescribir
      vermàs
      sino
      advierte;txt$+" no hallado"
      finsi
    finsi
  finproc
proc blanco;x,y
  escribir en x,y;rept(" ",64-y+1)
  finproc
proc c
  alterar :nota cambiar
  finproc
proc cerrartodo
  mientras 1
    cerrar
    finmientras
  finproc
proc compcom
  si c$="?":ayuda: finsi
  si c$="S": stop : finsi
  si c$="V":vienen: finsi
  si c$="B":b: finsi
  si c$="P": pròximo : finsi
  si c$="A": anterior : finsi
  si c$="C": error c:comperr: finsi :nota cambiar
  si c$="I": error 1: finsi
  finproc
proc comperr
  si numerr()>0 yy nàmerr()<>27
    advierte;"Error "+serie(numerr(),3,0)
    finsi
  finproc
proc i
  insertar
  finproc
proc p
  proximo
  finproc
proc pioecom
  mientras 1
    pescribir
    blanco;13,5
    escribir en 12,5;"¿Comando?: "; papel 6;" "; papel 0;"(Pulse ? para obtener
ayuda)"
    haz c$=mayùs(tecla())
    blanco;14,5
    escribir en 12,16;c$
    compcom
    finmientras
  finproc
proc pregunta;mens$
  escribir en 13,5; tinta 4;mens$;
  finproc
proc start
  limpiar
  escribir en 5,10; tinta 2; papel 6;"Abriendo fichero Invitados ..."
  error cerrartodo
  abrir "invitado" lògico "i"
```

```
  indicar
  pidecom
  finproc
proc vermás
  mientras hallado()
    pregunta;"Pulse cualquier tecla para ver más, o P para parar"
    si mayús(tecla())="P"
      regreso
      sino
      continuar
      pescribir
      finsi
    finmientras
  advierte;"No encuentro más "+txt$
  finproc
proc vienen
  pregunta;"¿Cuantos vienen?: "
  leer cuantos
  haz vienen$="s"
  actualiz
  finproc
proc zoom
  mientras no finf()
    pescribir
    escreg
    próximo
    finmientras
  finproc
```

tirar ↵ BodaCom1 prg ↵ tirar ↵ BodaCom prg ↵ tirar ↵
BodaCom3 prg ↵

pidecom ↵

s

8. Borre las versiones antiguas del programa, que han perdido ya su vigencia.
9. Pruebe la nueva pantalla de ayuda: escriba pidecom y pulse ↵, y a continuación ?.
10. Pulse s para salir de pidecom.
11. Saque el cartucho de la unidad 2, y etiquételo "*antes de hacer los ejercicios*".
12. Inserte la cinta de seguridad que acaba de crear etiquetada "para ejercicios".

EJERCICIOS

Los siguientes ejercicios le añaden más tareas al programa BodaCom. Todos se llaman desde la pregunta ¿Comando? pulsando una sola tecla.

La mayoría se usarán en el siguiente capítulo para proporcionarle el conjunto completo de opciones de un primer programa profesional.

Recuerde añadir las nuevas opciones a los procedimientos compcom y ayuda.

Recuerde que las preguntas y las introducciones de datos adicionales deben escribirse en la línea 13, y las advertencias en la línea 14.

Al final de la sección podrá encontrar respuestas sugeridas a los ejercicios.

1. Comando *p*

Adapte el procedimiento p para que muestre el mensaje Fin de fichero si se alcanza el final del fichero. Cambie la línea de compcom que llama a próximo para que llame a p.

2. Comando *a*

Adapte el procedimiento a para que produzca el mensaje Principio de fichero cuando se utilice sobre el primer registro. Cambie compcom.

3. Comando *z*

Añada el procedimiento zoom al programa bodacom. Adáptelo para que se pueda detener pulsando **ESC**, sin que se pare el programa entero. Recuerde el procedimiento p. Escriba un mensaje en la línea 13 que recuerde a la persona que lo use que **ESC** es una posibilidad de salida. Así se puede buscar un registro, y parar cuando se localice.

4. Comando *r*

Exactamente lo mismo que zoom, pero recorre el fichero en sentido contrario.

5. Comando *l* (uno)

Ir al primer registro (la letra p ya se ha usado).

6. Comando *u*

Ir al último registro.

7. Comando *n*

Ir a un registro especificado. Requerirá un número como entrada adicional (como en b y vienen).

8. Comando +

Salta un número especificado de registros hacia adelante o atrás desde el registro actual. Lea cuantos como un número positivo o negativo. Recuerde que las variables numéricas no terminan en S.

9. Comando *e*

Borre (elimine) un registro. Por si la tecla se pulsa accidentalmente, añada una linea donde se pida confirmación de que se debe borrar el registro.

- Asegúrese de que salva las respuestas en un cartucho especial "para ejercicios".

RESPUESTAS SUGERIDAS

Los procedimientos siguientes no son necesariamente los mejores, y desde luego no son las únicas soluciones a los problemas planteados al final del capitulo 9.

La mayoría se usarán en el siguiente capítulo, así que debe leerlos y añadirlos de la manera exacta en que están a la cinta que contiene bodacom etiquetada "antes de los ejercicios".

1. Comando *p*

```
proc p
   proximo
   si finf()
     advierte;"Fin de fichero"
     finsi
   finproc
```

2. Comando *a*

```
proc a
   anterior
   si recnum()=0
     advierte;"Comienzo de fichero"
     finsi
   finproc
```

3. Comando z

```
proc zoom
  escmens
  mientras no finf()
    pescribir
    escreg
    p
    finmientras
  finproc
```

Se usa el procedimiento p en lugar de próximo, ya que escribe un mensaje de advertencia cuando se llega al fin de fichero.

```
proc escmens
  pregunta;"Pulsa ESC para parar"
  finproc
```

Se ha escrito el procedimiento escmens porque se puede utilizar en otros procedimientos, por ejemplo, el siguiente.

Añada una línea a compcom que diga:

si c$ = "Z": error zoom: finsi

4. Comando *r*

```
proc inverso
  escmsg
  mientras recnum()>0
    a
    pescribir
    finmientras
finproc
```

Añada una línea a compcom que diga:

si c$ = "R": error inverso: finsi

5. Comando *l* (uno)

Añada una línea a compcom que diga:

si c$ = "1": primero: finsi

6. Comando *u*

Añada una línea a compcom como la siguiente:

si c$ = "U": último: finsi

7. Comando *n*

```
proc ir
  pregunta;"¿Ir a registro no.?"
  leer num
  posición num
  finproc
```

Observe que num es una variable numérica.
Añada a compcom una línea como ésta:

si c$ = "N":ir: finsi

8. Comando +

```
proc saltar
  pregunta;"¿Registros a saltar?"
  leer num
  posición recnum()+num
  finproc
```

Añada a compcom una línea como la siguiente:

si c$ = "+":saltar: finsi

La variable num se usa aquí de la misma manera que en el procedimiento anterior. Su función es puramente temporal, y parecida a txt$ para introducción de texto: al mantener el mismo nombre en los procedimientos se evita que la memoria se llene de variables inútiles.

Se podría haber usado el comando local para conseguir el mismo efecto.

Observe que, si se introduce en num un número positivo, el fichero se posiciona esos registros a continuación del actual. Si se escribe un número negativo, ocurre lo contrario, ya que un signo + y un — da un .

9. Comando *e*

```
proc elimina
  pregunta:"Borrar. ¿Seguro? (s/n)"
  leer txt$
  si minusc(txt$)="s"
    borrar
    finsi
  finproc
```

Añada también una línea como la siguiente a compcom:

si c$ = "E":elimina: finsi

AYUDA

El procedimiento ayuda debe quedar ahora parecido al siguiente:

```
proc ayuda
  limpiar
  escribir "PULSE UNA DE LAS TECLAS SIGUIENTES PARA ..."
  escribir "A  volver al registro Anterior"
  escribir "P  ir al Próximo registro"
  escribir "C  Cambiar un registro en pantalla"
  escribir "I  Insertar un nuevo registro"
  escribir "B  Buscar registros"
  escribir "V  marcar una respuesta, y decir cuàntos Vienen"
  escribir "S  dejarlo (Stop)"
  escribir "E  eliminar el registro actual"
  escribir "1  Ir al primer registro"
  escribir "U  Ir al Último registro"
  escribir "Z  zoom / avance ràpido"
  escribir "R  inverso / rebobinado"
  escribir "+  saltar un número de registros"
  escribir "N  ir al registro Número"
  escribir
  escribir "Pulse cualquier tecla para seguir";tecla()
  pantalla
  finproc
```

Si la pantalla de ayuda queda demasiado grande para la pantalla de que disponemos, elimine la cabecera o ponga dos comandos en cada línea (por ejemplo, A/P registro Anterior/Próximo).

SALVE EL TRABAJO REALIZADO

Cambie sus respuestas a los ejercicios hasta que sean las mismas que hemos sugerido, o simplemente escriba las respuestas sugeridas antes de seguir.

salvar ↵ BodaCom ↵

1. Salve los procedimientos que hay ahora en memoria a Bodacom.

dir ↵ ↵

2. Haga un directorio de la unidad 2 para comprobar la fecha del fichero _eti.

dir ↵ mdv1_ ↵

3. Haga un directorio de la cinta de seguridad, en la unidad 1, para comprobar la fecha del fichero _eti, asegurándose así de que se trata de la cinta de seguridad correcta. Si es la misma que en su cinta de trabajo, use el otro cartucho de seguridad.

salvar ↵ mdv1_bodacom ↵

4. Salve los mismos procedimientos a mdv1_BodaCom.

nuevo ↵

5. Borre todos los procedimientos y ficheros de datos de memoria escribiendo nuevo y pulsando ↵.

salvagrdr ...

6. Copie los ficheros que se han creado o cambiado desde que se usó la cinta por última vez.

salvar "Abr6 _eti" ↵ F5 ⇧⇨ ⇨ mdv1_ ↵

7. Salve ficheros _eti con la fecha de hoy en ambos cartuchos, tanto el de trabajo como la cinta de seguridad en la unidad 1.
8. tirar los ficheros _eti viejos de ambas cintas.

RESUMEN

Mientras comienzan a llegar respuestas a las invitaciones de boda, Eva decide escribir un conjunto de procedimientos que le permitan ver quién ha contestado y quién no, y cuánta gente viene en cada grupo.

En primer lugar, crea dos procedimientos, pidecom y compcom que funcionan juntos, recogiendo un comando de una sola letra y realizando una tarea en función de esa letra. El procedimiento pidecom usa la función tecla() y la variable c$ para leer un comando del teclado. El procedimiento compcom comprueba qué tecla se ha pulsado, y actúa en consecuencia.

Eva salva los dos procedimientos al fichero llamado boda com (comandos de boda). Arregla ligeramente pidecom añadien

dole una línea escribir, que hace que el registro actual se escriba en la pantalla. Usa también la función mayús() para convertir la letra obtenida en respuesta a tecla() en mayúsculas.

Deseosa de probar sus nuevos procedimientos, Eva usa unir para añadir los procedimientos de fácil a la memoria, ya que algunos son llamados por compcom. Prueba el conjunto sobre el fichero invitado. Se da cuenta de que sería más fácil de usar si tecla() tuviera un cursor para indicar que ARCHIVE aguarda que se pulse una tecla. Usa el elemento de impresión papel para proporcionar un cursor blanco en la posición donde se introduce la letra de comando.

Después usa el comando error para evitar que el programa colapse si hay algún error (como la pulsación de **ESC** en insertar). Usa también este comando en un nuevo procedimiento, llamado cerrartodo, que cierra todos los ficheros abiertos. Mediante error, el procedimiento no da error cuando ya no quedan más ficheros por cerrar. Usa luego la función numerr() para, en un procedimiento llamado comperr, imprimir el número de error cuando se usa el comando error, y los errores que podrían causar problemas no pasen inadvertidos.

Eva crea ahora un procedimiento que busca una persona en particular. Uniendo este procedimiento, b, con el procedimiento vermás, creado en el capítulo 7, ARCHIVE sigue buscando nombres que contengan las letras indicadas.

Tras verificar el trabajo, Eva añade un procedimiento llamado blanco que borre las líneas de mensaje de la pantalla.

Crea también dos procedimientos estándar que impriman mensajes y preguntas, así como advertencias sobre los errores.

Crea también un procedimiento que, al ser llamado, lista las opciones disponibles en la línea de comandos. Lo llama ayuda.

10

Construcción de pantallas

El comando indicar es rápido y fácil de usar, pero tiene varias limitaciones profundas:

- Si un fichero tiene más campos de los que caben en la pantalla, los campos no visibles no se pueden alterar o insertar.
- No se puede indicar dos ficheros juntos en la misma pantalla.
- Se muestra el nombre lógico, en lugar del nombre de fichero completo o una cabecera.
- indicar muestra los nombres de campo (abreviados a menudo a palabras casi sin significado), en lugar de mensajes indicativos.

Este capítulo muestra cómo construir sus propias pantallas.

- Se presentan los siguientes comandos de ARCHIVE:

peditar	— edita una pantalla
psalvar	— salva una pantalla a *microdrive*
pcargar	— carga una pantalla desde cartucho
pleer	— lee información desde una pantalla.

Puntos a recordar

- Las pantallas "a la medida" funcionan exactamente igual que la visualización de indicar.
- Una pantalla creada con peditar, como los procedimientos en editar, se debe salvar a cartucho si no queremos que se pierda.
- Se puede editar la visualización estándar si se desea, pero se puede limpiar en peditar mediante **F3** y l.
- Los programas de Eva suponen que los mensajes y las preguntas usan las líneas 12 a 14.

CONSTRUYA SU PROPIA PANTALLA

Eva cree que la visualización estándar de registros proporcionada por ARCHIVE es funcional, pero poco atractiva. No permite colocar los campos más que en la parte izquierda de la pantalla.

Se puede construir una presentación en pantalla con una serie de comandos leer en. Así cargaría sólo un campo cada vez, sin embargo, y el comando leer no muestra el valor previo del campo. Resultaría también lento.

En lugar de eso, Eva decide usar el comando peditar para crear una pantalla que funcione de una manera parecida a la pantalla de indicar cuando se use con comandos como insertar y alterar.

Quedará parecida a la pantalla siguiente:

```
                    LISTA DE INVITADOS A LA BODA

Nombre
Dirección

                                                ¿Vienen?
                                                No. Vienen
```

El comando PEDITAR

La palabra peditar es una abreviatura de pantalla y editar. El comando le permite crear y modificar la visualización en pantalla. Es distinto que editar, ya que le permite moverse con li-

bertad por la pantalla, y escribir directamente en las posiciones donde quiera situar un texto.

El comando peditar le permite crear una pantalla de presentación que se comportará exactamente como la visualización estándar de ARCHIVE.

> *Nota:* Si se ha usado el comando indicar, la visualización estándar para ese fichero está en memoria, y se puede cambiar mediante peditar. Como probablemente querrá comenzar de cero, borre la visualización estándar desde la pantalla de peditar usando **F3** seguido por | y ↵.

Por otra parte, puede encontrar útil mantener la visualización estándar temporalmente en la pantalla para recordarle los nombres de campo en uso. Se pueden borrar a continuación mediante la BARRA ESPACIADORA.

Creación de una visualización en pantalla mediante PEDITAR

peditar ↵

1. Escriba peditar y pulse ↵.
Se borra el AREA DE VISUALIZACION, y el AREA DE CONTROL le indica que use las teclas del cursor y ↵ para moverse por la pantalla.
Puede situar ahora preguntas y mensajes para la visualización e introducción de datos en la pantalla, y decirle a ARCHIVE dónde cargar e imprimir variables y campos. El resultado se puede salvar y usar en vez de indicar cuando sea necesario.

F3 | ↵

2. Borre la visualización estándar de la pantalla.

El título de la pantalla

⇨ *(20 veces)*

1. Pulse ⇨ hasta que el cursor recorra aproximadamente un tercio de la línea superior.

F3 p ESP ↵

2. Ponga de color rojo el fondo para el título de la pantalla. Para hacerlo, seleccione la opción papel pulsando p tras pulsar **F3**. El nombre del color aparece en el AREA DE TRABAJO, y el número del papel en el AREA DE CONTROL. Pulse la barra espaciadora hasta que la palabra rojo aparezca en el AREA DE TRABAJO; en ese momento pulse ↵ para indicarle a ARCHIVE que hemos seleccionado ese color.

F3 t ↵

3. Seleccione la tinta (el color de las letras) blanca de la misma manera que puso el papel rojo.

⇧ LISTA DE INVITADOS A LA BODA ↵ ⇧

4. Escriba LISTA DE INVITADOS A LA BODA y pulse ↵.
El cursor se mueve al comienzo de la línea siguiente.
El fondo es de color rojo, y las letras son blancas.
El fondo y las letras seguirán de esos colores hasta que seleccione otros diferentes.

↵ 5. Pulse de nuevo ↵.

Uso de los colores en PEDITAR

Puede llenar la pantalla entera de los colores negro, verde, blanco o rojo mediante las opciones papel y tinta del menú **F3** de peditar, si lo desea.

Una vez elegidos, tinta y papel colorean cualquier texto que escriba, hasta que seleccione otros distintos. El fenómeno es temporal más que espacial: puede seleccionar un color y moverse a cualquier posición en la pantalla, y no sólo hacia adelante.

Si pulsa la barra espaciadora, aparece un rectángulo del color del papel. Para moverse por la pantalla sin cambiar el color de fondo, use las teclas con flechas.

Puede elegir el color de las variables, además del texto (aunque no en la versión 1).

La primera variable

F3 p ESPACIO *(3 veces)* ↵

F3 t ESPACIO *(4 veces)* ↵

1. Ponga el papel negro de nuevo.
2. La tinta de nuevo a blanco.

Nombre

3. Escriba Nombre.

⇨ *(4 veces)*

4. Pulse ⇨ cuatro veces.

F3

5. Pulse **F3**.
El AREA DE CONTROL cambia, mostrando las cuatro teclas que se pueden pulsar para realizar distintas acciones. El uso de **F3** en peditar es muy parecido a su uso en editar.

v

6. Pulse v.
La caja de mensajes del AREA DE CONTROL cambia de nuevo, y la pregunta: "Nombre de Variable" aparece en el AREA DE TRABAJO.
ARCHIVE espera que escriba el nombre de una variable que se cargará de la pantalla en esa posición, como si usara un comando leer.

nombre$

7. Escriba nombre$.

↵ **8.** Pulse ↵.
La caja de mensajes cambia de nuevo. ARCHIVE espera que marque la zona en que se presentará e introducirá la variable.

□ **9.** Pulse cualquier tecla, excepto ↵.
Aparece un punto en el AREA DE VISUALIZACION. Observe que no es el carácter de la tecla que pulsó. La pulsación de cualquier tecla (excepto ↵) producirá una punto.

□ *(39 veces)* **10.** Pulse cualquier tecla otras 39 veces.

↵ **11.** Pulse ↵.
El cursor no va a una nueva línea, pero el AREA DE CONTROL vuelve al modo principal del editor de pantallas. El AREA DE TRABAJO se borra.

Ha finalizado el marcado de la zona de variable.

↵ **12.** Pulse ↵.

Los nombres de variables entre paréntesis no aparecen en la pantalla; están ahí para mostrarle la posición en que se colocará la variable en relación a otras.

A medida que coloca las variables, aparecen puntos que indican la longitud que tendrá su presentación. Estos puntos desaparecen cuando abandona peditar.

```
.....................................................
                LISTA DE INVITADOS A LA BODA


Nombre      ..................................... (nombre$)
Dirección ..................................... (direc1$)
            ..................................... (direc2$)
            ..................... (direc3$) ¿Vienen?   .. (vienen$)
            ..................... (direc4$) No. Vienen .. (cuentos)
```

Figura 10.1. Diagrama del diseño de pantalla propuesto

La segunda variable

Dirección **1.** Escriba Dirección.

⇨ **2.** Pulse ⇨ para llevar el cursor debajo del primer punto.

F3 **3.** Pulse **F3**.

v **4.** Pulse v.

direc1$ ↵ **5.** Escriba direc1$ y pulse ↵.

Observe que ARCHIVE no comprueba si el nombre que introdujo es un nombre de variable válido.

□ *(35 veces)*
6. Pulse cualquier tecla, excepto ↵, 35 veces.

↵
7. Pulse ↵ para dejar de asignar espacio a la variable.

.↵
8. Pulse ↵ para comenzar una nueva línea.

Poniendo más variables

⇨ *(déjela pulsada)*
1. Deje ⇨ pulsada hasta que el cursor llegue bajo el primer punto de las líneas de puntos.

F3
2. Pulse **F3**.

v
3. Pulse v.

direc2$
4. Escriba direc2$.

.↵
5. Pulse ↵.

. *(30 veces)* ↵
6. Escriba 30 puntos.
7. Pulse ↵ dos veces.

⇨ *(déjela pulsada)* F3 v direc3$ ↵ . *(20 veces)* ↵ ↵
8. Sitúe la variable direc3$ bajo direc2$, y asígnele 20 puntos de longitud.

⇨ *(déjela pulsada)* F3 v direc4$ ↵ . *(20 veces)* ↵ ↵
9. Sitúe la variable direc4$ bajo direc3$, y asígnele 20 espacios de longitud.

⇦
10. Mueva el cursor sobre uno de los puntos. Aparece inmediatamente el nombre de la variable en el AREA DE TRABAJO.

⇧
11. Mueva el cursor a los puntos de otras variables. Cada nueva variable se muestra en el AREA DE TRABAJO. Tran pronto como el cursor sale de los puntos, el AREA DE TRABAJO se limpia.

Observe que sólo se puede asignar una variable usando **F3** y v. Si escribe puntos en la pantalla mediante la tecla correspondiente, sólo conseguirá que los puntos aparezcan en la pantalla.

PEDITAR y la tecla F3

Al pulsar la tecla **F3** usando peditar se nos ofrecen cuatro opciones, cada una de las cuales se puede utilizar pulsando una sola tecla seguida por ↵.

v — marca la posición inicial para mostrar o introducir una variable en pantalla. Pulse cualquier tecla, excepto ↵, para situar la longitud de la visualización. Si ya hay una variable, el uso de **F3** y v la borra, en lugar de situar otra.

Cuando se usa la pantalla, las variables situadas en pantalla *no causan un error* si no existen realmente.

l	borra (limpia) toda la pantalla. Sea cuidadoso al usarla, ya que no se puede recuperar la pantalla si la borra antes de salvarla a cinta.
t	asigna el color de la tinta, que sigue activo hasta que se asigne de nuevo, o se abandone peditar.
p	asigna el color de papel que, como tinta, permanece activo hasta que se reasigna, o se abandona peditar. El uso de las teclas de cursor después de poner un nuevo papel deja inalterada la pantalla. Si usa la **BARRA ESPACIADORA**, "pinta" la posición del color actual de papel.

Observe que, en la versión 1, no se puede usar t y p para asignar el color de variables, que son siempre blancas sobre fondo negro.

Antes de seguir, Eva decide salvar su trabajo.
Para salir de peditar:

ESC

12. Pulse **ESC** cuando esté en el modo principal de peditar. Tan pronto como abandona el editor de pantallas, se borra la pantalla. La presentación que acaba de crear reaparece inmediatamente, con los campos del registro activo en la posición de los puntos.

Para salvar ficheros de pantalla: el comando PSALVAR

psalvar ↵

1. Escriba psalvar y pulse ↵.
Observe que la p inicial distingue la palabra de la relativa a programas.

boda ↵

2. Escriba boda y pulse ↵. El *microdrive* de la derecha gira, y una copia de la presentación de pantalla recién creada se salva en el cartucho bajo el nombre boda_scn.

El comando psalvar utiliza la extensión _scn a menos que se le especifique otra.

Solución a los errores en PEDITAR

- La mayor parte de las teclas de edición normales no funcionan de la manera normal en peditar (si hacen algo).
- La única manera de corregir cualquier cosa que escriba es pasando de nuevo hacia atrás mediante **CTRL** + ⇦, o pasando encima con la espaciadora y escribiendo de nuevo.
- Las teclas ⇨ y ⇦ funcionan como se espera, pero las teclas ⇧ y ⇩ nos mueven arriba y abajo de la pantalla, en vez de llevarnos a la izquierda y derecha de las líneas.
- Mientras pone una variable, cualquier tecla, excepto ↵ y **CTRL** ⇦ produce un punto en la pantalla.
- No se puede dejar de situar una variable mediante **ESC**. Esta tecla produce simplemente un punto. Debe pulsar ↵ para volver al modo normal, y borrar entonces la variable puesta.
- Las variables, una vez puestas, se pueden borrar situando el cursor sobre cualquiera de los puntos y pulsando **F3**, v y ↵.
- La única manera de cambiar la longitud de una variable una vez puesta es borrarla y ponerla de nuevo.
- Se puede mover una variable, una vez puesta, a una posición completamente distinta de la pantalla sin borrarla primero. Basta situarla directamente en la nueva posición.
- La posición en que situamos la variable originalmente se borrará automáticamente. Los puntos, sin embargo, quedan expuestos en la pantalla de peditar, y se pueden borrar con **CTRL** ⇦ o **ESPACIO.**

USO DE SUS PANTALLAS

abrir ↵ invitado''lógico''i ↵

1. Abra el fichero invitado.

cargar ↵ BodaCom ↵

2. Cargue los procedimientos BodaCom.

pidecom ↵

3. Escriba pidecom y pulse ↵.
Aparece la petición !Comando?: en la pantalla.

p

4. Pulse p para cambiar de registro.
El registro se muestra inmediatamente, exactamente como si hubiera usado la presentación estándar de indicar.

s

5. Pulse s para detener el procedimiento.

limpiar ↵

6. Escriba limpiar y pulse ↵ para borrar la pantalla.

pantalla ↵

7. Escriba pantalla y pulse ↵.
Reaparece su presentación en pantalla.

indicar ↵

8. Escriba indicar y pulse ↵.
Su presentación es sustituida por la visualización estándar de ARCHIVE.

El comando indicar siempre muestra la pantalla estándar de ARCHIVE. El comando pantalla vuelve a mostrar la visualización que hubiera en pantalla, sea la de ARCHIVE o alguna creada por peditar.

La presentación estándar reemplaza su pantalla en memoria temporal.

Para utilizar su pantalla de nuevo o para editarla, hace falta cargarla de nuevo a memoria.

Carga de ficheros de pantalla: el comando PCARGAR

pcargar ↵ **1.** Escriba pcargar y pulse ↵. Observe la p al comienzo del comando.

boda ↵ **2.** Escriba boda y pulse ↵.
La pantalla se borra y la presentación estándar se sustituye por la presentación salvada en boda_scn. El registro activo se muestra automáticamente.

peditar ↵ **3.** Escriba peditar y pulse ↵.
El registro activo se sustituye por puntos que representan las variables en pantalla.

↵ *(4 veces)* **4.** Pulse ↵ cuatro veces y mueva el cursor al final del campo direc2$.

⇨ *(5 veces)* **5.** Pulse ⇨ 5 veces.

¿Vienen? ⇨ *(2 veces)* **6.** Escriba ¿Vienen? seguido por dos pulsaciones de ⇨.

F3 **7.** Pulse **F3**.

v **8.** Pulse v.

vienen$ ↵ **9.** Escriba vienen$ y pulse ↵.

. . ↵ **10.** Escriba dos puntos y pulse ↵.

↵ **11.** Pulse ↵ de nuevo para situar la otra variable en la línea siguiente.

⇨ *(hasta la ¿ de* vienen) No. vienen ⇨ ⇨ F3 v cuantos ↵ . . ↵ ↵

12. Sitúe la otra variable bajo la anterior, alineando el mensaje y poniendo la variable en la misma columna (véase diagrama).

La pantalla debe qucdar ahora como la siguiente:

```
                    LISTA DE INVITADOS A LA BODA

Nombre      ....................................
Direccion   ....................................
            ....................................
            ....................................          ¿Vienen?   ..
```

ESC psalvar ↵ boda ↵

13. Abandone peditar y psalve la nueva visualización al archivo de pantallas boda.

Nota: El número de caracteres que se pueden insertar o imprimir en una variable viene limitado por el número de puntos (escribir) o el número de puntos menos uno (pleer) en la versión 2. Así, si quiere introducir a través de la pantalla los códigos vienen$, debe dar, al menos, dos caracteres de ancho, uno para la cifra y el otro para el cursor.

Las líneas más largas que la anchura de la variable en la pantalla existen, pero no se imprimen con pantalla. Esto puede tener consecuencias molestas si no tiene cuidado, ya que, por ejemplo, un 10 se confunde con un 1 en una pantalla con una sola posición.

ADICION DE LA NUEVA VISUALIZACION A SUS PROCEDIMIENTOS

editar ↵ TAB *(hasta* start)

↓ *(5 veces)* F5 CTRL ↓ pcargar "boda" ↵

1. Use editar para alterar el procedimiento start.

2. Cambie la línea indicar para que quede pcargar "boda".

El procedimiento debe quedar como sigue:

```
proc start
  limpiar
  escribir en 5,10; tinta 2; papel 6;"Abriendo fichero Invitados ..."
  error cerrartodo
  abrir "invitado" logico "i"
  pcargar "boda"
  pidecom
  finproc
```

ESC

salvar ↵ bodacom ↵

3. Deje el editor.

4. Salve el programa bodacom recién cambiado.

Resumen de archivos de pantalla

- Las presentaciones de pantalla se comportan exactamente de la misma manera que el comando indicar de ARCHIVE, pero se pueden diseñar a la medida.
- peditar sirve para crear y modificar visualizaciones en pantalla, igual que editar con los procedimientos.
- Otros comandos de edición se llaman mediante **F3**, seguido por

v	para poner o borrar una posición de variable
l	para limpiar la pantalla de peditar
t	para poner el color de la tinta de los mensajes y variables
p	para poner el color del papel.

- psalvar salva pantallas a un archivo tipo _scn.
- pcargar carga una pantalla del cartucho a memoria.
- pleer le permite introducir variables dispuestas en la pantalla. Se usa en el próximo capítulo.

Versiones incompatibles de ficheros de pantalla

Los archivos de pantalla creados con la versión 1 de ARCHIVE no se pueden usar por la versión 2, ya que la estructura del archivo es distinta*.

RESUMEN

Eva decide mejorar la tarea de introducción de información al fichero invitado creando una visualización de pantalla que le permita poner sus propios mensajes y preguntas en la pantalla, y

---

* *N. del T.:* La documentación indica que la versión 2 viene con un programa en SUPERBASIC llamado scrcon_bas que realiza la conversión del formato antiguo al nuevo. Esto no es cierto en la versión 2.21 (primera española), por lo que, si quiere utilizar una vieja pantalla con el nuevo programa, debe editarla de nuevo en ARCHIVE versión 2.

poner los nombres de los campos en las posiciones de la pantalla que prefiera.

Lo hace mediante el comando peditar, para crear una visualización de pantalla. Borra la pantalla estándar usando la tecla **F3**, seguida por l.

En primer lugar, crea un título para la pantalla, que coloca en un color diferente, usando los comandos de papel y tinta.

Coloca entonces las variables por turno, tras preparar un mensaje delante de cada una, de manera que se sepa el tipo de información que contienen.

Salva el fichero de pantalla, mediante psalvar, para que no se almacene sólo en memoria temporal, y se pierda al salir de ARCHIVE.

Para acabar, añade el archivo de pantalla a los procedimientos de bodacom, de manera que se llame por start a abrir el fichero invitado.

Trucos útiles

- Dibuje una visualización de pantalla en papel antes de intentar poner mensajes en la pantalla. Marque los números de línea en un lado, para que pueda ver de un vistazo dónde escribirá cuando use en al escribir.
- Escriba los nombres de los campos y considere qué longitud máxima va a necesitar cada uno de ellos. Recuerde: un campo con pocos caracteres asignados muestra sólo parte de su contenido.
- No nombre los ficheros scn con el mismo nombre que los ficheros _prg, ya que resulta fácil escribir psalvar como salvar, borrando uno de ellos.
- Cuando sitúe las variables de un color y los mensajes de otro, ponga primero todos los mensajes, moviéndose por la pantalla mediante las teclas del cursor. A continuación ponga todas las variables. Así se evita tener que cambiar de color de papel y de tinta para cada mensaje y cada variable.

11

Un nuevo programa hecho de viejos bloques

Se puede mejorar sorprendentemente la facilidad, rapidez y precisión de una tarea usando la habilidad de ARCHIVE para hacer automáticamente conexiones complejas, referencias cruzadas, verificaciones y cálculos.

Muchos trabajos requieren buscar información en varias listas, tablas, catálogos o ficheros diferentes. Por ejemplo, cuando un cliente pide un producto, el encargado de pedidos busca el precio del producto y la dirección del cliente, calcula las ventas, factura al cliente y reduce el stock antes de archivar una copia en el fichero de pedidos. Este proceso es tan tedioso como erróneo.

Este capítulo muestra cómo:

- Usar los programas de boda para manejar otros grupos de datos muy distintos.
- Utilizar dos ficheros simultáneamente.
- Calcular automáticamente campos a partir de otras entradas de datos.
- Se presenta el siguiente comando ARCHIVE:

pleer — espera la entrada de variables específicas en una pantalla de presentación.

Punto a recordar

- No ponga demasiados campos en un fichero, probablemente lo que necesite es crear un fichero separado, aunque ligado.

Nota: A partir de ahora, las instrucciones y métodos utilizados en este libro se van haciendo más complejos y no podrán explicarse con tanto cuidado como en capítulos anteriores.

Incluso si no desea utilizar ahora las técnicas o no comprende enteramente su significado, trate de seguirlas.

Más tarde, cuando se encuentre con un problema que no sepa cómo resolver, puede acordarse de haber visto algo parecido aquí, y encontrará estos capítulos particularmente útiles.

Ejemplo: Las ventas de juguetes

Juan es director de producción de Juguetes Mas *("Donde los sueños son lo primero").* Su último hallazgo, el lagarto "Lito" (apodado el Litosaurio por Salva), no se está vendiendo demasiado bien. Las grandes ventas en la campaña de Navidad de la recién lanzada llamada "Lola" amenazan con derribar a "Lito".

Ha pensado asombrar al mundo, y a su jefe, demostrándole que los beneficios de las ventas del lagarto superan a los de las llamas, un caso claro de calidad contra cantidad.

Como esto requiere cálculos repetitivos y moderadamente complejos, decide utilizar ARCHIVE para toda la aritmética.

La mayoría del trabajo lo tiene ya hecho Eva. Todo lo que Juan tiene que hacer es adaptar el programa.

INSTRUCCIONES

La primera tarea de Juan es actualizar los ficheros de datos. Su idea es utilizar dos ficheros, uno para guardar los pormenores de los juguetes y el otro para los registros de ventas. Así elimina la repetición de datos (pueden existir muchas partidas de ventas de un solo juguete).

Esta relación de "uno a muchos" es común en numerosas actividades:

- Bibliotecas, donde existen muchos préstamos de cada libro.
- Médicos, donde cada paciente realiza muchas visitas.
- Vendedores, donde se dan muchas ventas por cada cliente.

- Revistas profesionales, donde cada corporación tiene muchas suscripciones.

Es una pérdida de tiempo teclear la información detallada de cada libro, paciente, cliente o la dirección de los suscriptores cada vez que se haga referencia a ellos, puesto que la información entra una sola vez y aparece cada vez que el proceso se repite.

Desde luego es posible que un libro no se preste nunca, que una persona nunca se ponga enferma, etc., y existan entonces pocos registros en vez de muchos.

El hecho de que pueda haber muchos, o ningún registro, hace poco práctico crear un campo para cada préstamo, visitas, etc.

La solución es crear dos archivos y encadenarlos, buscando los datos de uno de acuerdo con los del otro.

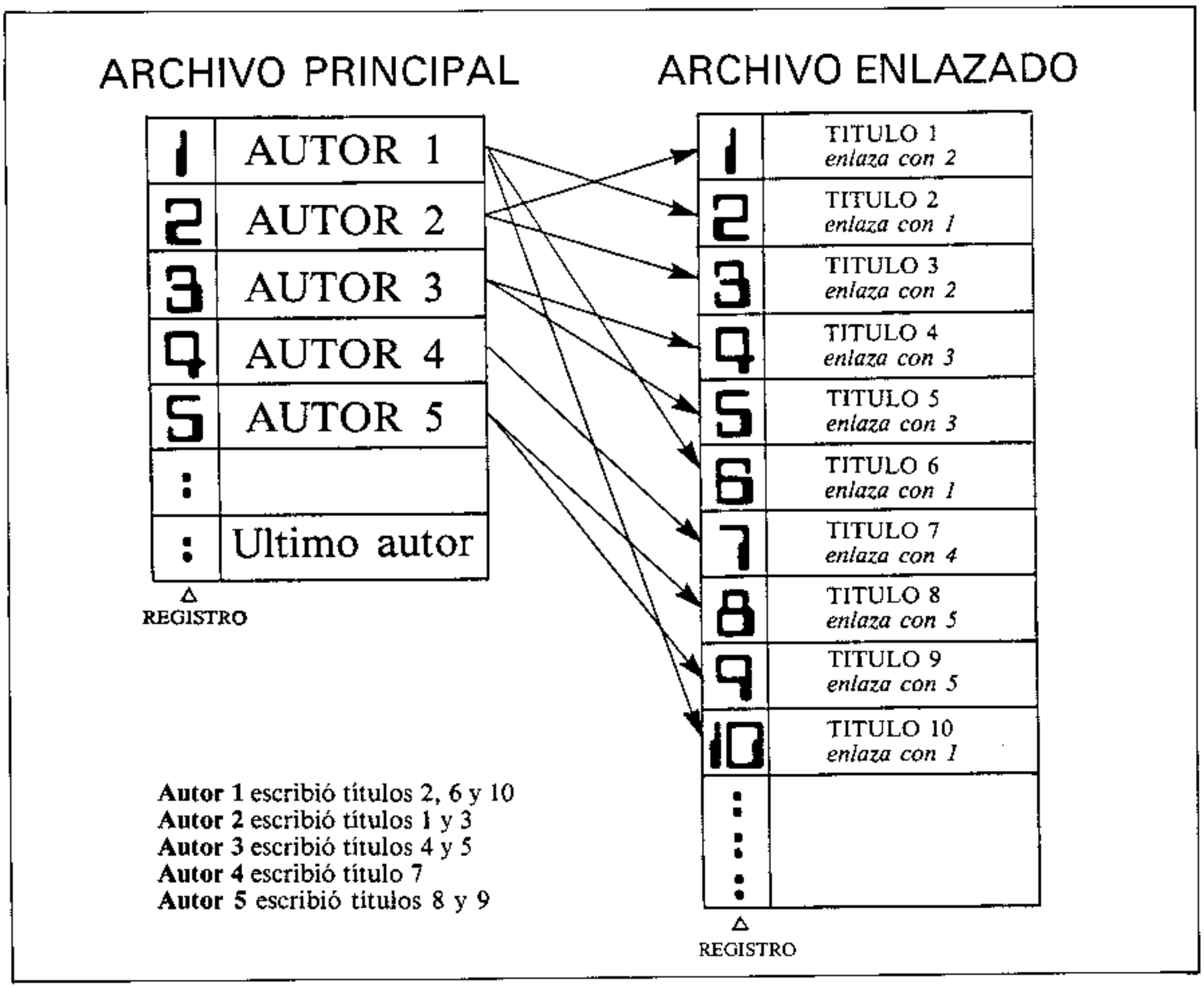


Figura 11.1. Una relación "uno-a-muchos"

Múltiples relaciones "uno a muchos"

Este proceso permite manejar fácilmente relaciones mucho más complejas. En cada uno de los ejemplos anteriores se puede crear un segundo fichero con relación "uno a muchos":

- Muchos préstamos a cada lector.
- Muchas prescripciones (a pacientes) por cada fármaco.
- Muchos productos en cada venta.
- Diferentes títulos enviados al mismo suscriptor.

Este capítulo sólo trata de un caso simple de búsqueda y relación, pero el principio es el mismo.

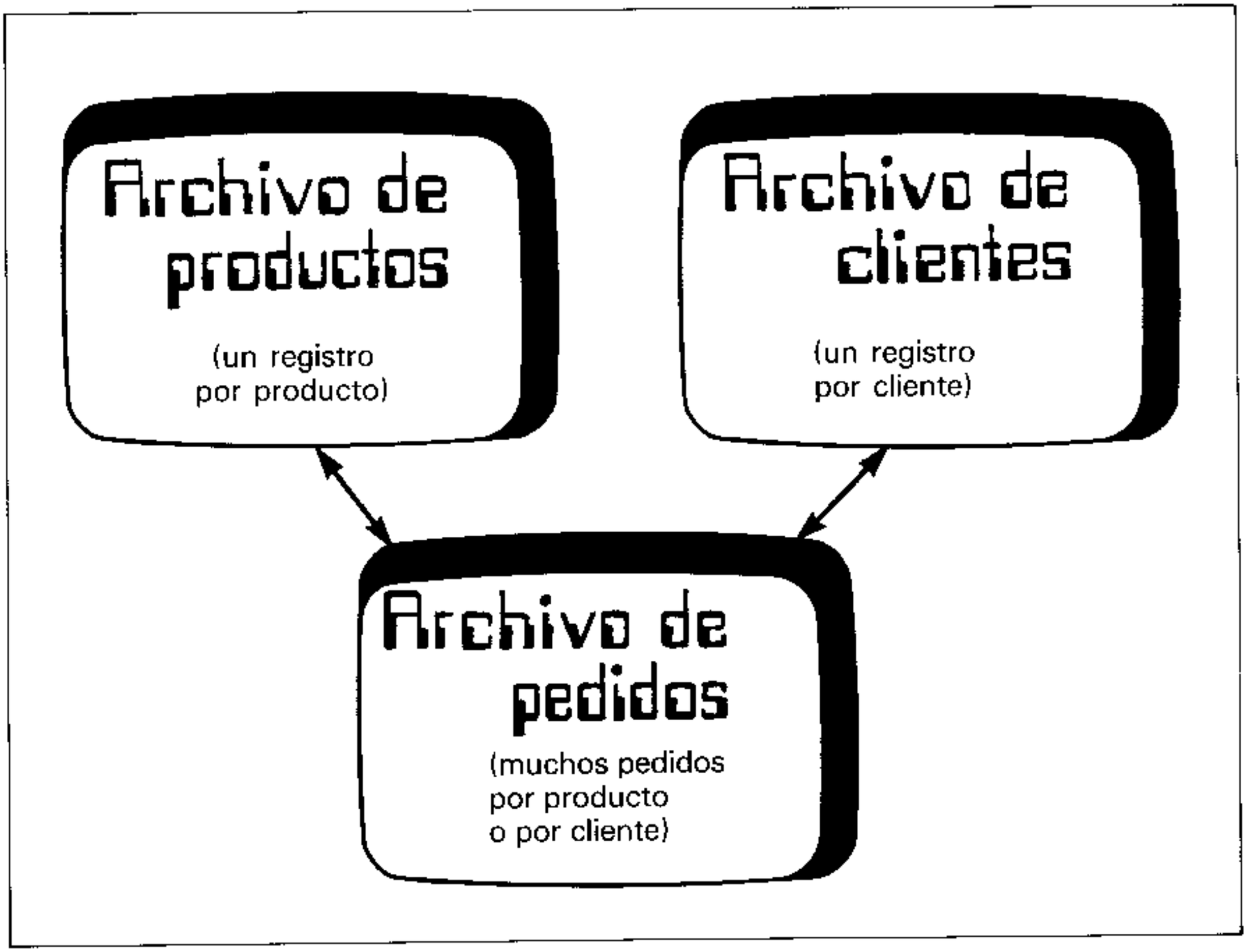


Figura 11.2. Múltiples relaciones "uno-a-muchos"

Comienzo

Si ARCHIVE no está cargado, cárguelo ahora. Asegúrese de que todos los ficheros de datos y procedimientos han sido borrados de la memoria.

> *Nota:* De ahora en adelante no se darán los comandos tecla a tecla; sólo aparecerán a la izquierda las nuevas líneas a escribir o cambiar.

El fichero "juguet"

El fichero juguet contiene el código, nombre, coste y precio de cada juguete.

crear "juguet" lógico "j" ↵

1. Comienza la creación del fichero juguet.

codigo$ ↵ **2.** El campo codigo$ contiene un nombre de tres letras por cada juguete. Se usa como clave para relacionar con los registros del fichero ventas, que contendrán la misma clave.

nombre$ ↵ **3.** El campo nombre$ contiene el nombre del juguete; por ejemplo, Pablo el Pingüino.

coste ↵ **4.** El campo coste contiene el valor numérico del coste de fabricación. Es necesario para los cálculos.

precio ↵ **5.** El campo precio contiene el valor del juguete en la tienda.

↵ **6.** Pulse ↵ y aparecerá la palabra fincrear Se ha creado el fichero.

indicar ↵ **7.** Escriba indicar y pulse ↵ para ver la estructura final.

Debe aparecer lo siguiente:

```
nombre lógico   : j
      código$   :
      nombre$   :
      coste     :0
      precio    :0
```

El fichero "ventas"

A continuación creamos el fichero que contiene los registros de ventas mensuales por cada juguete.

crear "ventas" lógico "v" ↵

1. Cree el fichero ventas.

código$ ↵

2. El campo código$ contiene el código del juguete en el fichero juguet, que permite buscar información en el fichero juguet cada vez que se introduzca una venta.

mes ↵

3. El campo mes contiene el número del mes en que se han hecho las ventas; por ejemplo, 4 = abril.

cant ↵

4. El campo cant contiene la cantidad total vendida durante el mes.

desc ↵

5. El campo desc contiene el porcentaje de descuento ofrecido ese mes en todas las ventas.

benef ↵

6. El campo benef contiene el beneficio total del mes, calculado automáticamente por el programa.

↵

7. Pulse ↵ y aparecerá la palabra fincrear. El fichero ha sido creado.

indicar ↵

8. Escriba indicar y ↵ para ver la estructura final.

Debe aparecer de esta manera:

```
nombre lógico : v
mes           : 0
cant          : 0
desc          : 0
benef         : 0
```

Juan opina que su programa debería permitir los registros de juguetes sin ninguna unión con los de ventas, pero no al contrario. Después de todo, es posible tener un juguete que no se venda, al menos todavía, pero no la venta de un juguete que no exista.

Otras aplicaciones de las búsquedas

Búsquedas para validación

Es extremadamente útil detectar errores comprobando la entrada de datos en el momento de introducirse por teclado.

Un procedimiento sencillo es mediante la sentencia si con expresiones lógicas, como se ilustró en el capítulo 7.

Si el margen de posibles entradas es más amplio, se pueden agrupar en un fichero, y comprobar el dato de entrada. Si se encuentra, la entrada es correcta.

Búsqueda para explicación del código

Los códigos son rápidos de escribir y ocupan poco espacio cuando están almacenados, pero son difíciles de leer y recordar, especialmente si no los definió o no utiliza el sistema frecuentemente.

Cree un fichero de búsqueda ordenada de códigos, junto con una descripción completa de lo que significan, y utilícelo cuando se listen o impriman los registros codificados.

La pantalla de venta de juguetes

Borre los ficheros de datos de la memoria tecleando nuevo y ↵.

Escriba peditar para crear un formato de pantalla que muestre el contenido de los dos ficheros.

La variable f$ aparece también en pantalla, aunque no forma parte de un fichero. Se utiliza para guardar el nombre del fichero actualmente en uso.

Escriba el formato de la pantalla según el diagrama. Los puntos indican dónde deberán colocarse las variables utilizando **F3** y V, y los nombres entre paréntesis las variables que deberán colocarse allí.

No escriba los nombres de las variables. Se muestran simplemente para ayudarle a crear el formato.

- Fíjese en los dos puntos que hay a uno y otro lado de los campos de código$ y nombre$. Cuando se utilice la pantalla mostrará a aquel que esté usando el ordenador, la longitud máxima del campo.
  La novena línea (número 8, pues ARCHIVE cuenta desde 0) contiene todas las variables numéricas para el fichero ventas. Están todas sobre la misma línea, de modo que puedan borrarse fácilmente.
  La línea punteada al final de la pantalla se reserva para los comandos, indicadores y mensajes de aviso.

```
.....................................................................
                        LISTADO DE JUGUETES
                        -------------------

Código de jug. :....: (codigo$)                  Archivo ......... (f$)
Nombre         :................: (nombre$)
Coste  ...... Ptas (coste)
Precio ...... Ptas (precio)

Mes        Cantidad vendida              Descuento %     Benef Ptas
...             ......                       .....          .......
(mes)           (cant)                       (desc)         (benef)

.....................................................................
```

Salve con psalvar el formato en un fichero llamado venjug.

- No dé el mismo nombre a ficheros _scn y _prg. Un día puede olvidar accidentalmente una p en psalvar y borrar un conjunto de procedimientos en cinta con cualquier otra cosa que esté en memoria en ese momento. Incluso si no hay procedimientos en memoria, salvar salvará un fichero vacío _prg.

Cuando acabe la pantalla deberá ser como ésta:

```
.....................................................
                LISTADO DE JUGUETES
                -------------------
Código de jug. :     :                    Archivo
Nombre         :                 :
Coste          Ptas
Precio         Ptas

Mes        Cantidad vendida          Descuento %    Benef Ptas


.....................................................
```

MODIFICACION DE "BODACOM" PARA LAS VENTAS DE JUGUETES

Juan hace una copia de los programas de Eva cargándolos, haciendo una o dos alteraciones superficiales, y recuperándolos con un nombre nuevo.

cargar ↵ bodacom ↵

1. Cargue los procedimientos de la boda de Eva.

editar ↵ TAB ↓ F5 ↑→ CTRL ↓ juguet_prg 12 May 84 ↵

2. Cambie el comentario del procedimiento a1 para elegir el nuevo programa; por ejemplo:

   Nota juguet_prg 12 MAY 84

   Borre las líneas que hacen referencia a vienen e ir en a1.

TAB (*repita hasta* vienen) F3 b ↵

3. Borre el procedimiento vienen, que no va a ser necesario.

TAB (*repita hasta* ir) F3 b ↵

4. Borre el procedimiento ir, creado en los ejercicios del úl-

F4 ordenar código$; a, mes; a ↵

3. Inserte una línea para ordenar, en el fichero ventas, los campos código$ y mes, en orden ascendente. Escriba ordenar código$;a,mes;a y ↵. Ambos ficheros necesitan estar ordenados por código$, porque el código del juguete se utiliza como clave para unir sus ventas a cada juguete. Y también porque situar (usado con ficheros ordenados) es más rápido que los comandos hallar o buscar. Las ventas pueden ser introducidas en el fichero en cualquier orden para cualquier juguete, pero las ventas de un mismo juguete tienen que aparecer en secuencia cuando se unen al registro de juguete.

abrir "juguet" lógico"j" ↵ ordenar código$;a ↵

4. Insertar otras dos líneas para abrir el fichero juguet como "j", y ordenarlo por código$ en orden ascendente.

> *Nota:* Los dos comandos ordenar se incluyen aquí por claridad, aunque son redundantes, puesto que los ficheros se han ordenado una vez. El orden de un fichero se salva cuando se cierra (si se abrió con la sentencia abrir), y se carga en el mismo orden cuando el fichero es reabierto.

Juan inicializa el valor de una variable que se utiliza para seguir la pista, en todo momento, del fichero que maneja el programa. El fichero cambiará en los procedimientos a medida que se cambie de fichero, debe declararse aquí, antes de referirse a ella en otra parte.

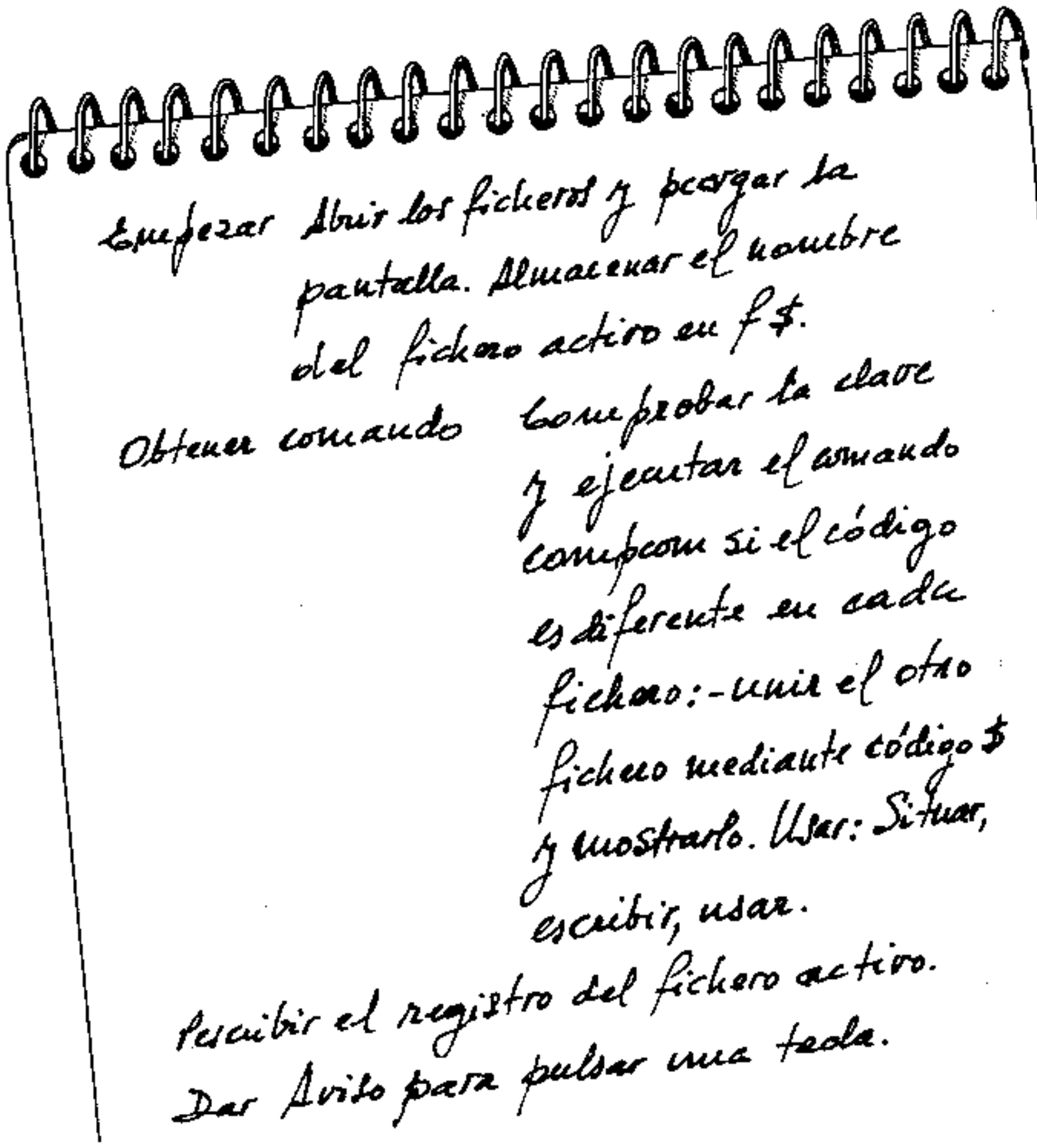
haz f$ = "juguetes"

5. Inserte la línea haz f$ = "juguetes".
La variable f$ se utiliza como un mensaje en la pantalla para recordarle qué fichero está activo. La mayoría de los comandos afectan al fichero activo, y no a otros, por lo que es necesario saber cuál es éste.
Esta variable se utiliza también por los procedimientos como una señal que determina cuál es el fichero activo.

6. Cambie la línea que carga el fichero de pantalla, sustituyendo el fichero boda por venjug.
El formato de pantalla se carga al final de todo, para que su aparición nos indique que el programa está listo para recibir comandos.

El procedimiento final deberá aparecer como éste:

Mirando en la descripción, ve que la búsqueda inicial y la final son simplemente uniones entre el fichero activo y el no activo. Reescribe la descripción porque esto ocurre una sola vez en el proceso.



A estas alturas, la única diferencia significativa respecto al método de Eva (start y pidecom) es la necesidad de guardar dos ficheros ordenados unidos bajo la misma clave.

Juan empieza en serio desde el principio, con el procedimiento start.

editar ↵ TAB

1. Escriba editar y pulse la tecla **TAB** hasta llegar al procedimiento start.
2. Adapte start para que en lugar de abrir el fichero invitado como i, abra el fichero ventas como v.

F4 ordenar código$; a, mes; a ↵

3. Inserte una línea para ordenar, en el fichero ventas, los campos código$ y mes, en orden ascendente. Escriba ordenar código$;a,mes;a y ↵. Ambos ficheros necesitan estar ordenados por código$, porque el código del juguete se utiliza como clave para unir sus ventas a cada juguete. Y también porque situar (usado con ficheros ordenados) es más rápido que los comandos hallar o buscar. Las ventas pueden ser introducidas en el fichero en cualquier orden para cualquier juguete, pero las ventas de un mismo juguete tienen que aparecer en secuencia cuando se unen al registro de juguete.

abrir "juguet" lógico"j" ↵ ordenar código$;a ↵

4. Insertar otras dos líneas para abrir el fichero juguet como "j", y ordenarlo por código$ en orden ascendente.

> *Nota:* Los dos comandos ordenar se incluyen aquí por claridad, aunque son redundantes, puesto que los ficheros se han ordenado una vez. El orden de un fichero se salva cuando se cierra (si se abrió con la sentencia abrir), y se carga en el mismo orden cuando el fichero es reabierto.

Juan inicializa el valor de una variable que se utiliza para seguir la pista, en todo momento, del fichero que maneja el programa. El fichero cambiará en los procedimientos a medida que se cambie de fichero, debe declararse aquí, antes de referirse a ella en otra parte.

haz f$ = "juguetes"

5. Inserte la línea haz f$ = "juguetes".
La variable f$ se utiliza como un mensaje en la pantalla para recordarle qué fichero está activo. La mayoría de los comandos afectan al fichero activo, y no a otros, por lo que es necesario saber cuál es éste.
Esta variable se utiliza también por los procedimientos como una señal que determina cuál es el fichero activo.

6. Cambie la línea que carga el fichero de pantalla, sustituyendo el fichero boda por venjug.
El formato de pantalla se carga al final de todo, para que su aparición nos indique que el programa está listo para recibir comandos.

El procedimiento final deberá aparecer como éste:

```
proc start
   limpiar
   escribir en 5,10; tinta 2; papel 6;"Abriendo ficheros ..."
   error cerrartodo
   abrir "ventas" lógico "v"
   ordenar codigo$;a,mes;a
   abrir "juguet" lógico "j"
   ordenar codigo$;a
   haz f$="juguetes"
   pcargar "venjug"
   pidecom
   finproc
```

Cambio de fichero activo

Juan crea un procedimiento para cambiar el fichero activo. De otra forma, el fichero juguet siempre será el activo, y ninguna de las opciones de comandos disponibles afectarán al fichero ventas.

Para ello necesita un procedimiento que invierta la situación de los ficheros. Utiliza la variable f$ para localizar el fichero activo y cambiarlo por el otro.

F3 n cambiar ↵ si f$ = "ventas" ↵

1. Cree un procedimiento llamado cambiar. La primera línea será si f$ = "ventas", esto es, preguntará si el fichero activo es el fichero "ventas".

usar "j" ↵

2. La segunda línea, usar "j".
   Si ventas es el fichero activo, entonces use el fichero juguet.

haz f$ = "juguet"

3. Añada una línea para volver a actualizar la variable f$ con el valor del fichero activo juguet.

sino ↵

4. La siguiente línea sino (significa de otro modo) quiere decir: si el fichero activo no es ventas, ejecuta el siguiente comando.

usar "v" ↵

haz f$ = "ventas" ↵

5. Añada una línea que actualice el fichero ventas.

6. Añada una línea que cambie el valor de f$: haz f$ = "ventas".

finsi ↵

7. Fin de la sentencia si: finsi. Abandone el modo de inserción.

El procedimiento acabado deberá quedar así:

```
proc cambiar
   si f$ = "ventas"
      usar "j"
```

```
        haz f$ = "juguet"
        sino
        usar "v"
        haz f$ = "ventas"
        finsi
    finproc
```

El procedimiento que cambia las letras del comando, junto con ayuda y a1, han de ser alterados para tener en cuenta el nuevo procedimiento.

8. Inserte si c$ = "X":cambiar:finsi en compcom.

El procedimiento deberá aparecer de esta manera:

```
proc compcom
  si c$="X":cambiar: finsi
  si c$="E":elimina: finsi
  si c$="+":saltar: finsi
  si c$="R": error inverso: finsi
  si c$="1": primero : finsi
  si c$="U": último : finsi
  si c$="Z": error zoom: finsi
  si c$="?":ayuda: finsi
  si c$="S": stop : finsi
  si c$="B";b: finsi
  si c$="P": próximo : finsi
  si c$="A": anterior : finsi
  si c$="C": error c:comperr: finsi :nota cambiar
  si c$="I": error i: finsi
  finproc
```

9. No olvide actualizar los procedimientos ayuda y a1 añadiendo una opción para X:
Añada escribir "X cambiar de fichero activo" al fichero ayuda.

10. Añada nota cambiar:(compcom), usa el otro fichero, cambia f$.

11. Introduzca la línea de comentario en el procedimiento a1 llamándolo juguet1_prg y salve los procedimientos en juguet1.

Unir los ficheros

Ahora ya se podrán utilizar todos los procedimientos de Eva sobre cualquier fichero de forma independiente, pero la información obtenida raramente será sobre el mismo juguete, a no ser que se obligue a los registros a permanecer unidos.

Lógicamente esto no es deseable, ya que los dos registros de la pantalla han de ser relativos al mismo juguete, principalmente porque el objetivo principal de tener dos ficheros es buscar información sobre los juguetes para cada venta.

Es necesario añadir tres líneas más al procedimiento pidecom para reflejar el uso de dos ficheros.

Después de cada comando, ARCHIVE vuelve a pidecom. Si como resultado de algún comando, los registros de ventas y juguetes dejan de estar relacionados con el mismo juguete, deberá corregirse la situación verificando la relación y rehaciéndola si es necesario.

1. Edite el procedimiento pidecom.
2. Inserte haz otrocod$ = " " antes de la línea mientras 1:
   La variable otrocod$ se utiliza para almacenar el código$ actual en el fichero que no está en uso. Así detecta si el código del fichero activo (el valor de código$) no es el mismo que el código del otro fichero (almacenado en otrocod$).
   La primera vez que se utiliza pidecom, otrocod$ no sirve para nada, porque los ficheros todavía no están unidos.
3. Baje una línea e inserte las tres líneas siguientes:

```
si código$< >otrocod$
    unereg
    finsi
```

La sentencia si llama al procedimiento unereg si el campo código$ no es el mismo que otrocod$ (que representa código$ en el fichero que no está en uso).

La primera vez que se ejecute pidecom, unirá un registro de ventas con el registro activo de juguetes, dado que otrocod$ comienza como " ", que no puede ser un código$ en ningún fichero.

Más tarde, si una acción tomada desde compcom hace referencia a otro fichero, se localiza éste, para guardar los registros de ventas y juguetes unidos por el mismo código.

El procedimiento pidecom resulta ahora:

```
proc pidecom
  haz otrocod$=""
  mientras 1
    si codigo$<>otrocod$
      unereg
      finsi
    pescribir
```

```
    blanco;13,5
    escribir en 12,5;"¿Comando?: "; papel 6;" "; papel 0;"(Pulse ? para obtener
ayuda)"
    haz c$=mayùs(tecla())
    blanco;14,5
    escribir en 12,16;c$
    compcom
    finmientras
  finproc
```

Y el programa contiene los siguientes procedimientos:

```
a                   cerrartodo          pidecom
al                  compcom             pregunta
advierte            comperr             saltar
ayuda               elimina             start
b                   escmens             vermàs
blanco              i                   zoom
c                   inverso
cambiar             p
```

El procedimiento de unir registros

Qué fichero unir

Juan comprende que, cualquiera que sea el fichero activo, el otro tendrá que estar unido a él. Si no están unidos por el mismo código, esto es detectado en pidecom. El procedimiento unereg realiza la unión, haciendo uso de la variable f$ (que siempre contendrá el nombre del fichero activo) para determinar si tiene que unir ventas con juguetes, o al contrario.

F3 n unereg ↵ haz otrocod$ = código$ ↵

1. Cree un procedimiento llamado unereg y escriba la primera línea haz otrocod$ = código$. Observe que otrocod$ toma el valor del código del fichero activo antes de que los ficheros sean cambiados. Dando por supuesto que será encontrado un código similar en el otro fichero. Si no es así, otrocod$ se cambia.

si f$ = "ventas" ↵

2. La segunda línea deberá ser: si f$ = "ventas"
   La variable f$ determina el fichero activo. El otro entonces es el que se usa y se une.

unejug; código$ ↵

3. La línea tres es unejug; código$.
   El valor actual del campo código$ es utilizado como un parámetro en el procedimiento que une juguetes a ventas.

sino ↵ uneven; código$ ↵

4. Escriba las siguientes líneas para unir ventas a juguetes en el caso de que el fichero ventas no sea el fichero activo:

```
sino
uneven; código$
```

finsi ↵ ↵

5. Cierre la sentencia si: finsi y abandone el modo inserción.

El procedimiento queda de la siguiente forma:

```
proc unereg
    haz otrocod$ = código$
    si f$ = "ventas"
        unejug; código$
        sino
        uneven; código$
        finsi
finproc
```

Unir "juguetes" con "ventas"

El procedimiento unereg tiene dos opciones posibles: unir juguetes a ventas o unir ventas a juguetes. Son distintos, ya que pueden usarse por otros procedimientos posteriores, y porque (por suposición de Juan) puede (al principio) no haber ventas para un juguete. El procedimiento que une ventas a juguetes deberá contemplar esta posibilidad.

Por ello, Juan comienza por el más fácil: crear el procedimiento que une un registro de juguetes a uno de ventas. Necesita localizar en el fichero juguet el código que tiene el registo activo de ventas. Primero tendrá que convertir a juguet en el fichero activo.

proc unejug; busca$ ↵

1. El procedimiento para hacer esto se llama unejug; busca$. El valor de busca$ será el mismo que código$ cuando al procedimiento se le llame con código$ como parámetro.

usar "j" ↵

2. La segunda línea convierte a juguet en el fichero activo, que en ese momento es ventas, para encontrar el registro de juguet que se va a unir.

situar busca$ ↵

3. La siguiente línea es situar busca$.
Puede utilizarse el comando situar cuando se ha ordenado un fichero. Salta inmediatamente a la posición correc-

ta en éste, y, por tanto, es mucho más rápido que hallar o buscar.

Se supone que el fichero tiene un juguete para cada venta, así es que situar no debe, en principio, fallar nunca.

pescribir ↵

4. Utilice pescribir para mostrar el campo activo cuando el registro ha sido encontrado.

pescribir sólo afecta al registro activo, así que los valores de ventas no vuelven a mostrarse.

usar "v" ↵

5. Una vez que la unión ha sido realizada y mostrada, se vuelve a convertir el fichero ventas en fichero activo, tal y como estaba al principio.

El procedimiento queda así:

```
proc unejug; busca$
usar "j"
situar busca$
pescribir
usar "v"
finproc
```

Unir "ventas" con "juguet"

Este procedimiento trabaja exactamente de la misma forma que el anterior, pero es significativamente diferente. Ahora es posible que no existan registros de ventas para un juguete dado.

Si se da ese caso, el comando situar, al no encontrar en ventas el registro adecuado, se posicionará en el siguiente, que no tendrá el mismo código que el registro de los juguetes.

El comando pescribir, utilizado con el fichero ventas, mostraría entonces un registro equivocado, sin relación con el juguete que está en pantalla. Esto podría tener consecuencias desafortunadas.

Juan resuelve este problema colocando espacios en blanco en la línea de ventas en lugar de utilizar pescribir.

Deja vacío otrocod$ si no hay registro de ventas.

uneven; busca$ ↵ usar "v" ↵ situar busca$ ↵ si v.código$< >busca$ ↵

1. Cree un procedimiento llamado uneven; busca$ y escriba las siguientes líneas:

```
uneven; busca$
usar "v"
situar busca$
si v.código$< >busca$
```

Si no existe registro unido al fichero ventas, situar dejará uno con diferente código$ como registro activo.
Eso no es lo que se busca.

En la versión 2 de ARCHIVE se puede comprobar si situar se ha satisfecho mediante el comando hallado( ) con la sentencia si.

blanco;8,0

2. Añada la siguiente línea: blanco;8,0.
Esta línea pone a blancos los valores previos del juguete mostrado en la pantalla. Un pescribir ahora nos presentaría una venta no relacionada con el juguete.

haz otrocod$ = "⇩"

3. La siguiente línea es haz otrocod$ = "⇩".
Cuando no existe código del juguete activo en el fichero de ventas, se introduce un carácter nulo en la variable otrocod. Esto evita problemas al unir en pidecom.

sino ↵ pescribir ↵ finsi ↵ usar "j" ↵

4. Añada las líneas siguientes al procedimiento:

```
sino
pescribir
finsi
usar "j"
```

5. Abandone el modo inserción.

El procedimiento acabado queda de esta forma:

```
proc uneven; busca$
    usar "v"
    situar busca$
    si v.código$< >busca$
        blanco; 8,0
        haz otrocod$ = " "
        sino
        pescribir
        finsi
    usar "j"
finproc
```

6. Actualice el procedimiento a1 para que la línea de comentarios lea juguet2_prg. Añada información sobre unereg, uneven y unejug.

El procedimiento ahora deberá aparecer así:

```
proc a1
  nota JUGUET2_PRG 12 de Mayo de 1985
  nota unereg: (pidecom), sitúa código en el otro fichero
  nota uneven: (unereg),localiza j.codigo$ en ventas (si no blancos)
  nota unejug: (unereg) (ventaent), sitúa v.codigo$ en juguet
  nota pidecom: menú, pantalla, lee tecla c$
  nota compcom: llamada pidecom, comprueba c$
  nota i: llamada por compcom, (error) insertar
  nota comperr: escribe un mensaje con numerr()
  nota c: (compcom), (error) alterar
  nota blanco: (pidecom), pone en blanco la línea desde x,y
  nota cerrartodo: (pidecom), cierra todos los ficheros
  nota b: (compcom), busca texto. almacenado en txt$
  nota ayuda: (compcom), lista las opciones disponibles
  nota pregunta: Procedimiento estándar, escribe línea en 13,5
  nota vermás: (b), sigue hallando registros
  nota start: cierra los ficheros, abre y muestra "invitado"
  nota advierte: proc estándar para escribir advertencias en 14,5
  nota cambiar: (compcom), usa el otro fichero, cambia f$
  finproc
```

7. Salve los procedimientos como juguet2.

Advierta que todos los procedimientos relativos a unir empiezan con las mismas letras. De esta forma, siguiendo el curso de una unión, o intentando descubrir errores, no se necesita listar todos los procedimientos del fichero.

La siguiente figura muestra cómo están relacionados los ficheros.

```
proc pidecom
  haz otrocod$=""
  mientras 1
    si codigo$<>otrocod$
      unereg ──────────────► proc unereg
      finsi                    haz otrocod$=codigo$
    pescribir                  si f$="ventas"
    blanco;13,5                  unejug;codigo$ ──┐
    escribir en 12,5;...         sino             │
    haz c$=mayús(tecla())        uneven;codigo$ ──┼──► proc uneven;busca$
    blanco;14,5                  finsi            │      usar "v"
    escribir en 12,16;c$       finproc            │      situar busca$
    compcom                                       │      si v.codigo$<>busca$
    finmientras                                   │        blanco;8,0
  finproc                                         │        haz otrocod$=""
                                                  │        sino
                                                  │        pescribir
                                                  │        finsi
                                                  │      usar "j"
                                                  │      finproc
                                                  │
proc unejug;busca$ ◄──────────────────────────────┘
  usar "j"
  situar busca$
  pescribir
  usar "s"
  finproc
```

Algunos registros para empezar

Es posible que Juan tenga grandes deseos de poner a prueba su inspirado programa, a pesar de que todavía no está acabado.

Comienza ejecutando el comando start para conseguir que "la bola empiece a rodar", luego pulsa i mientras utiliza el fichero juguet para insertar algunos juguetes.

Pone mucho cuidado para asegurarse de que los códigos de los juguetes (código$) sean letras mayúsculas, para que situar las considere siempre iguales.

Teclea los detalles de los siguientes juguetes:

LOL	Llama Lola	690	1700
LIT	Lagarto Lito	578	1990
PIP	Pingüino Pablo	356	1100

Los datos estarían almacenados en los campos con este formato:

```
código$     :PIP
nombre$     :Pingüino Pablo
coste       :356
precio      :1100
```

Juan no añade ningún registro de ventas más en este momento, porque tendría que calcular el beneficio (el punto principal de este ejercicio) a mano.

No está demasiado contento con la forma en que los juguetes se insertan en ambos ficheros. insertar no pone en mayúsculas los códigos, haciéndole usar la tecla ⇧, y no hay nada que le impida definir dos veces el mismo código, lo que resulta muy poco seguro.

El comando PLEER

Juan busca un método para introducir los datos de uno en uno, y no todos a la vez, como con insertar. Así, puede comprobar cada campo introducido antes de proceder con una nueva entrada (si hay alguna). Decide utilizar el comando pleer.

Este comando es muy similar al comando leer. Mediante pleer, ARCHIVE espera a que se introduzca los datos de entrada en la pantalla para almacenarlos en una variable.

pleer sólo trabaja con variables definidas en el formato de pantalla, si es un fichero propio tipo _scn, o en el formato estándar.

pleer no necesita de comandos especificos de impresión (tales como en o tinta). Hay que tenerlo en cuenta al definir el formato de pantalla.

A diferencia de leer, el comando pleer no borra el contenido de la variable antes de comprobar la entrada desde pantalla.

Este comando se introduce acompañado de aquellos campos o variables que se quieren introducir, por ejemplo:

```
pleer nombre$, coste, precio
```

Para insertar un nuevo juguete

Juan escribe un conjunto de procedimientos que comprueben los códigos de juguete introducidos antes de insertar un registro en el fichero. Los pasa a mayúsculas y comprueba que no existen ya. Después, mediante pleer, toma los campos que quedan en la pantalla uno a uno. Y graba el registro.

De esta forma, evita que el mismo código corresponda a diferentes juguetes.

Plantea los procedimientos de la siguiente forma:

jugins	— llamado desde compcom, inserta un registro de juguetes si el código$ es nuevo. Lo llama
intcodjug	— pide un código de juguete y lo convierte en mayúsculas.
juglim	— borra las variables del campo de entrada.
jugleer	— utiliza pleer para poder introducir los datos en los campos correspondientes.

```
proc jugins ↵   intcodjug ↵
```

1. Cree el primero de estos procedimientos, jugins. Las primeras dos líneas son:

```
proc jugins
    intcodjug
```

El procedimiento intcodjug pide el código de juguete, y lo convierte en mayúsculas.

```
    situar nuecod$
```

2. Escriba la línea 3: situar nuecod$.
El comando situar se utiliza para encontrar el código, si existe.

```
si código$ = nuecod$
```

3. La siguiente línea es si código$ = nuecod$.
Esta línea comprueba si situar ha encontrado un regis-

tro. Si código$ no es el mismo que nuecod$, que era el código deseado, entonces no lo ha encontrado.

En ARCHIVE versión 2, la misma comprobación puede ser realizada con la función hallado( ), pero en la versión 1 esto no funciona con el comando situar.

advierte;"código ya existente"

4. Es necesario añadir una línea que advierta al operador que el código dado ya existe:

```
advierte;"código ya existente"
```

- Cada código debe ser único, o aparecerán errores, con ventas unidas a juguetes equivocados. Si el código ya existe, no podrá insertar más registros con el mismo código.

sino ↵ haz código$ = nuecod$ ↵

5. Las siguientes dos líneas son:

```
sino
haz código$ = nuecod$
```

Que iguala el valor actual de código$ con el de nuecod$, una vez que ha sido comprobado.

juglim **6.** Realiza un llamada al procedimiento juglim, que pone el campo de variables a cero, y llama a su vez a pescribir.

error jugleer **7.** La siguiente línea llama al procedimiento jugleer, que busca en la pantalla los datos de entrada que faltan. Para ello utiliza el comando error, porque la entrada puede ser abandonada con la tecla **ESC**, y así los errores (como poner letras en un campo numérico) no interrumpen todo el programa.

comperr **8.** Luego añade una línea para comprobar los errores:

```
comperr
```

Compruebe siempre los posibles errores después de llamar a un procedimiento con error. Comperr informará de cualquier error.
Si algunos campos ya han sido introducidos cuando se abandona un nuevo registro, éste no es añadido. El mensaje de error indicará por qué.

finsi **9.** Finalice el procedimiento cerrando la sentencia si:finsi.

El procedimiento acabado queda:

```
proc jugins
    intcodjug
    situar nuecod$
    si código$ = nuecod$
       advierte; "código ya existente"
       sino
       haz código$ = nuecod$"
       juglim
       error jugleer
       comperr
       finsi
    finproc
```

10. Pulse ↵ para abandonar el modo inserción y crear el siguiente procedimiento.

Listo para un nuevo código

El siguiente procedimiento, intcodjug, es muy educado, y antes que nada pregunta por el código del juguete, mediante el procedimiento pregunta.

Es esencial que los códigos de juguete sean almacenados siempre consistentemente, de forma que pueda hacerse la unión correcta entre los ficheros.

El procedimiento es así:

```
proc intcodjug
    pregunta; " ¿Código de juguete? "
    leer nuecod$
    haz nuecod$ = mayús(nuecod$)
    finproc
```

Limpiar los campos de entrada

1. Cree un procedimiento, juglim, que borre y muestre el valor de los campos activos.

```
juglim
haz nombre$ = " "
haz coste = 0
haz precio = 0
pescribir
finproc
```

A menos que se utilice el comando pescribir, la pantalla muestra siempre el valor del último registro indicado. Si antes de realizar una entrada de datos no borramos los valores de los campos, da la impresión de estar superponiendo los datos sobre otro registro.
Tenga en cuenta que, aunque ponga las variables a cero, esto no afectará al fichero hasta que no se utilicen los comandos añadir o actualiz.

Valores de entrada para el fichero "juguetes"

pleer nombre$,coste, precio

1. Finalmente cree el procedimiento jugleer para conseguir los valores del fichero juguet. La primera línea es pleer nombre$,coste, precio.
El campo código$ ya ha colocado el valor correcto en jugins. Los otros campos son investigados ahora.

añadir

2. La segunda línea añade las variables de entrada al fichero juguet: añadir.
Si pleer es abandonado por causa de un error, o por pulsar la tecla **ESC**, el procedimiento se para antes de que alcance esta línea, y el registro no es añadido.
Sin embargo, algunas de las variables de campo mostradas estarán equivocadas hasta que se realice el próximo pescribir.

El procedimiento acabado es el siguiente:

```
proc jugleer
    pleer nombre$,coste, precio
    añadir
    finproc
```

Y el procedimiento total queda:

```
proc jugins
  intcodjug
  situar nuecod$
  si codigo$=nuecod$
    advierte;"Código en uso"
    sino
    haz codigo$=nuecod$
    juglim
    error jugleer
    comperr
    finsi
  finproc

proc intcodjug
  pregunta;"¿Código de juguete?
  leer nuecod$
  haz nuecod$=mayús(nuecod$)
  finproc

proc juglim
  haz nombre$=""
  haz coste=0
  haz precio=0
  pescribir
  finproc

proc jugleer
  pleer nombre$,coste,precio
  añadir
  finproc
```

Actualizando

1. Como ha sido descrito un procedimiento que añade registros al fichero juguet, ya no es necesario el procedimiento i, así que elimínelo.
2. Cambie el procedimiento compcom borrando la línea:

   ```
   si c$ = "I":error i:comperr:finsi.
   ```

3. Inserte en su lugar las tres líneas siguientes:

   ```
   si c$ = "I"
   si f$ = "ventas":ventins:sino:jugins:finsi
   finsi
   ```

   Ahora que se ha borrado i, habrá que crear un nuevo procedimiento para insertar registros en ventas.
   Si la variable f$ indica que ventas es el fichero activo, ventins inserta una venta. En caso contrario, se utiliza jugins.
4. Actualiza el procedimiento a1 para incluir jugins, intcodjud, juglim y jugleer:

```
nota intcodjug: (jugins) (venins). pide un código de jug.
nota juglim: (jugins), limpia las var. de campo
nota jugleer: (jugins) (comcheck). intr. campos menos codigo$
nota jugins: (comcheck), añade si codigo es nuevo
finproc
```

Insertar y calcular una nueva venta

Ahora comienza para Juan la parte difícil del trabajo: el cálculo automático de beneficios cada vez que se introduzcan ventas mensuales.

Utiliza pleer para introducir las ventas, de forma equivalente que para juguet.

El procedimiento inicial para insertar una nueva venta, ventins, es muy similar a jugins. La diferencia clave es que esta vez el código debe existir ya en el fichero juguet. De lo contrario, nos faltarían los datos del coste y precio del juguete para poder hacer los cálculos.

intcodjug ↵ unejug; nuecod$ ↵

1. Las dos primeras líneas del procedimiento ventins son:

```
intcodjug
unejug; nuecod$
```

Se utiliza el procedimiento unejug para ejecutar el necesario situar en el fichero juguet.
Originalmente se creó para que lo llamara unereg, pero sirve aqui para el mismo propósito.

```
si j.código$ = nuecod$
```

2. La tercera línea es: si j.código$ = nuecod$.
Advierta el uso del nombre lógico del fichero j con la variable código$. Esto es necesario porque el fichero activo es v, que tiene también un campo llamado código$.
Si el código$ activo no es el buscado, el procedimiento dará un aviso y no continúa con la inserción de una venta.

```
haz código$ = nuecod$ ↵ ventlim ↵ error ventleer ↵
```

3. Las tres líneas siguientes son:

```
haz código$ = nuecod$
ventlim
error ventleer
```

El procedimiento ventleer utiliza pescribir para buscar los campos de mes, descuento y la cantidad de juguetes vendidos, calcula el beneficio y lo añade al registro.

```
comperr ↵ sino ↵ advierte;"No existe el juguete" ↵ finsi ↵
```

4. Finalice el procedimiento añadiendo una sentencia para escribir un mensaje de aviso, y el fin de la condición si.

El procedimiento final quedará:

```
proc ventins
intcodjug
unejug;nuecod$
si j.código$ = nuecod$
    haz código$ = nuecod$
    ventlim
    error ventleer
    comperr
    sino
    advierte;"No existe el juguete"
    finsi
finproc
```

Teclea en ventlim el procedimiento para limpiar los valores de los campos y mostrarlos.
El procedimiento deberá quedar:

```
proc ventlim
    haz mes=0
    haz cant=0
    haz desc=0
    haz benef=0
    pescribir
    finproc
```

Los cálculos

El procedimiento ventleer, al igual que jugleer, utiliza el comando pleer para registrar los valores de los campos en la pantalla. Con estos datos junto con los del registro activo de juguetes calcula el beneficio en las ventas.

Antes de empezar, Juan se plantea estos cálculos. El beneficio de cualquier venta será el precio del juguete, menos el descuento, menos el coste de fabricación.

Precio — Descuento — Coste = Beneficio

El descuento debe ser calculado primero, al ser introducido como un porcentaje; por ejemplo, 20 %. Un descuento del 20 % es lo mismo que el 80 % (100— 20) de la cantidad, así que Juan ajusta la fórmula así:

Precio × (1 — Descuento %/100) — Coste = Beneficio

Si el porcentaje de descuento es del 20 %, el precio ha de ir multiplicado por 0,80, que es lo mismo que restarle 0,20 veces su precio.

Finalmente, este beneficio por juguete ha de multiplicarse por la cantidad vendida para que dé el beneficio total de la venta.

```
pleer mes,cant,desc
```

1. Cree el procedimiento ventleer y teclee la primera línea:

```
pleer mes,cant,desc.
```

Fíjese que el campo de beneficio, al igual que codigo$, no está en esta lista, y por eso nunca se carga directamente. Se calcula en la siguiente línea.

```
haz benef=cant*(j.precio*(1—desc/100)—j.coste)
```

2. Teclea la siguiente línea que realiza el cálculo:

```
haz benef=cant*(j.precio*(1—desc/100)—j.coste)
```

Los dos campos del fichero juguet que son utilizados en el cálculo se identifican por el nombre lógico j.

- Advierta que ARCHIVE hace los cálculos aritméticos de acuerdo con ciertas prioridades (igual que lo hace con expresiones lógicas).
- Las expresiones entre paréntesis se hacen primero, siendo prioritarias la multiplicación y división frente a la suma y la resta.
- Sin los paréntesis, este orden de prioridades en los operadores haría que el cálculo estuviese equivocado.
- Para mayor seguridad, procure poner paréntesis en las expresiones complejas. Si lo hace, será más fácil de leer.

añadir ↵ pescribir ↵

3. Finalice el procedimiento con estas dos líneas:

añadir
pescribir

El procedimiento quedará:

```
proc ventleer
  pleer mes,cant,desc
  haz benef=cant*(j.precio*(1-desc/100)-j.coste)
  añadir
  pescribir
  finproc
```

4. Actualice la línea de comentarios en el procedimiento a para que lea juguet3_PRG y añada líneas de notas para cada procedimiento recién definido:

```
nota ventlim: (ventins), limpia campos
nota ventleer: (ventins), usa pleer para leer campos, calc benef
nota ventins: (compcom), unejug
finproc
```

5. Abandone el editor y salve el procedimiento para el fichero juguet3.

Prioridades en las operaciones

ARCHIVE calcula las expresiones complejas paso a paso, pero no en una secuencia arbitraria. Cada parte en una expresión tiene una prioridad determinada:

Operaciones	Prioridad
Subíndices y cadenas	12
Todas las funciones	11
^ (exponenciación)	10
Negación	9
()	8
*, /	7
+, —	6
=, <, >, <=, >=, <>	5
no	4
yy	3
oo	2

Las operaciones con un número alto se realizan antes que las que tienen menos prioridad.

Advierta que efectivamente los paréntesis anulan todas las prioridades en las operaciones matemáticas, relativas y lógicas. Uselos para asegurar precisión y hacer las expresiones más fáciles de leer.

Ventas del lagarto "Lito"

Juan introduce los datos sobre las ventas del lagarto "Lito" y la llama "Lola". Sólo existen datos de seis meses para "Lola", que fue lanzada al mundo en julio.

De hecho, Juan hace responsable de las pérdidas de "Lito" al Departamento de Marketing por la promoción en julio de la llama "Lola". Confía en que su comparación del promedio de los beneficios ponga a "Lito", y a él mismo, de nuevo en el candelero.

1. Escriba el comando pidecom para conseguir que el programa comience de nuevo.
   No es necesario utilizar el comando start de nuevo, y esperar mientras todos los ficheros son cerrados y reabiertos, porque ya están abiertos y todas las variables necesarias para los procedimientos actualizadas.

2. Pulse X para pasar del fichero juguet al fichero ventas.
3. Pulse i para insertar una venta.

lit ↵

4. Teclee lit y ↵.
5. Introduzca las siguientes cantidades:

Mes	Cantidad	Descuento %
Lagarto Lito - código LIT		
Ene	10	
Feb	180	
Mar	310	
Abr	320	
May	230	
Jun	260	
Jul	—20	(incluyendo lagartos devueltos)
Ago	210	20
Sep	180	
Oct	135	
Nov	120	
Dic	90	
Llama Lola - código LOL		
Jul	525	50
Ago	475	50
Sep	330	20
Oct	260	20
Nov	210	
Dic	300	50

Juan tiene ahora las cantidades y beneficios mensuales para los dos juguetes. En el siguiente capítulo adapta los procedimientos para listar de Eva sobre la impresora o la pantalla.

1. Cambie la línea de comentarios en el procedimiento a para que lea juguet_prg.
2. Salve los procedimientos como juguet.
3. Borre los programas juguet1, juguet2 y juguet3.

dir ↵ ↵

4. Haga un directorio del *microdrive* 2 para comprobar los datos en el fichero _lab.

dir ↵ mdv1 ↵

5. Haga un directorio de su cinta de seguridad, en el *microdrive,* para comprobar los datos del fichero _eti y asegurarse de que es correcta la cinta de seguridad. Si es la misma que la cinta con la que ha trabajado, utilice la otra cinta de seguridad.

nuevo

6. Borre todos los procedimientos y ficheros de datos de la memoria temporal tecleando nuevo y ↵.

salvagrdr ...

salvar "Mayo12_eti" ↵

7. Haga una copia de seguridad de los ficheros creados o cambiados desde que la cinta se usó por última vez.

F5 ⇧⇨ mdv1_ ⇦

8. Salve los ficheros etiqueta en la cinta de trabajo y en la de seguridad del *microdrive* 1.
9. Elimine los antiguos ficheros _eti en ambas cintas.

RESUMEN

Juan piensa que sería una buena idea adaptar algunos de los procedimientos de Eva para sus propios fines. Quiere comparar las ventas del lagarto "Lito" y de la llama "Lola". Para ello crea dos nuevos ficheros, juguet y ventas, que contienen un campo común, codigo$, para poder así ser enlazados. Diseña también su propio formato pantalla mediante peditar.

Adapta el procedimiento bodacom de Eva. También el procedimiento start para abrir sus dos nuevos ficheros ordenados por codigo$ (y mes en el caso ventas). Crea un procedimiento capaz de cambiar de un fichero a otro pulsando una tecla. Lo llama cambiar.

Necesita unir ambos ficheros para relacionar las cantidades de ventas para cada juguete de forma correcta. Escribe tres procedimientos para hacer esto, unereg, unejug y uneven. Salva sus procedimientos y los prueba introduciendo algunas cantidades de ventas para "Lola" y "Lito".

Utiliza el comando pleer para actualizar el fichero juguet cuando todos los detalles han sido introducidos por pantalla.

Finalmente, Juan escribe un procedimiento de entrada para el fichero ventas que calcula automáticamente el beneficio a partir del precio, el coste y el descuento.

Indicaciones útiles

- Utilice más ficheros, y menos campos por fichero. Una los ficheros.
- Utilice variables (y sus procedimientos) como señales para estar informado sobre condiciones de importancia, como el nombre del fichero activo, o el registro que se une a otros ficheros.
- Utilice paréntesis para dar claridad y precisión en las expresiones complejas.
- Utilice pleer (incluso con la pantalla estándar) para controlar la entrada de datos campo a campo.

- Capture los datos erróneos en el momento de introducirse por la pantalla. Más tarde, serán mucho más difíciles de encontrar y corregir, y pueden no ser advertidos hasta que hayan causado muchos problemas.
- Utilice ficheros de búsqueda ordenados para los registros unidos, validando las entradas y expandiendo los códigos.
- Ponga dos puntos u otro tipo de delimitación a ambos lados del campo definido en la pantalla para mostrar cuál es la longitud máxima del dato permitida.

Listado del programa

juguet_prg contiene ahora los siguientes procedimientos:

```
proc a
  anterior
  si recnum()=0
    advierte;"Comienzo de fichero"
    finsi
  finproc
proc aj
  nota JUGUET2_PRG 12 de Mayo de 1985
  nota ventlim: (ventins), limpia campos
  nota ventleer: (ventins), usa pleer para leer campos, calc benef
  nota ventins: (compcom), unejug
  nota intcodjug: (jugins) (venins), pide un código de jug.
  nota juglim: (jugins), limpia las var. de campo
  nota jugleer: (jugins) (comcheck), intr. campos menos codigo$
  nota jugins: (compcom), añade si codigo es nuevo
  nota unereg: (pidecom), sitúa código en el otro fichero
  nota uneven: (unereg),localiza j.codigo$ en ventas (si no blancos)
  nota unejug: (unereg) (ventaent), sitúa v.codigo$ en juguet
  nota pidecom: menú, pantalla, lee tecla c$
  nota compcom: llamada pidecom, comprueba c$
  nota i: llamada por compcom, (error) insertar
  nota comperr: escribe un mensaje con numerr()
  nota c: (compcom), (error) alterar
  nota blanco: (pidecom), pone en blanco la linea desde x,y
  nota cerrartodo: (pidecom), cierra todos los ficheros
  nota b: (compcom), busca texto, almacenado en txt$
  nota ayuda: (compcom), lista las opciones disponibles
  nota pregunta: Procedimiento estàndar, escribe linea en 13,5
  nota vermàs: (b), sigue hallando registros
  nota start: cierra los ficheros, abre y muestra "invitado"
  nota advierte: proc estàndar para escribir advertencias en 14,5
  nota cambiar: (compcom), usa el otro fichero, cambia f$
  finproc
proc advierte;mens$
  escribir en 14,5; papel 6; tinta 2;" ";mens$;" "
  finproc
proc ayuda
  limpiar
  escribir "PULSE UNA DE LAS TECLAS SIGUIENTES PARA ..."
  escribir "A  volver al registro Anterior"
  escribir "P  ir al Próximo registro"
  escribir "C  Cambiar un registro en pantalla"
```

```
  escribir "I  Insertar un nuevo registro"
  escribir "B  Buscar registros"
  escribir "S  dejarlo (Stop)"
  escribir "E  eliminar el registro actual"
  escribir "1  Ir al primer registro"
  escribir "U  Ir al último registro"
  escribir "Z  zoom / avance rápido"
  escribir "R  inverso / rebobinado"
  escribir "+  saltar un número de registros"
  escribir "X  Cambiar de fichero activo"
  escribir
  escribir "Pulse cualquier tecla para seguir"¡tecla()
  pantalla
  finproc
proc b
  pregunta¡"¿Buscar?: "
  leer txt$
  si txt$>""
    hallar txt$
    si hallado()
      escribir
      vermás
      sino
      advierte¡txt$+" no hallado"
      finsi
    finsi
  finproc
proc blanco¡x,y
  escribir en x,y¡rept(" ",64-y+1)
  finproc
proc c
  alterar :nota cambiar
  finproc
proc cambiar
  si f$="ventas"
    usar "j"
    haz f$="juguetes"
    sino
    usar "v"
    haz f$="ventas"
    finsi
  finproc
proc cerrartodo
  mientras 1
    cerrar
    finmientras
  finproc
proc compcom
  si c$="X":cambiar: finsi
  si c$="E":elimina: finsi
  si c$="+":saltar: finsi
  si c$="R": error inverso: finsi
  si c$="1": primero : finsi
  si c$="U": último : finsi
  si c$="Z": error zoom: finsi
  si c$="?":ayuda: finsi
  si c$="S": stop : finsi
  si c$="B":b: finsi
  si c$="P": próximo : finsi
  si c$="A": anterior : finsi
  si c$="C": error c:comperr: finsi :nota cambiar
  si c$="I"
    si f$="ventas":ventins: sino :jugins: finsi
    finsi
  finproc
```

```
proc comperr
  si nùmerr()>0 yy nùmerr()<>27
    advierte;"Error "+serie(nùmerr(),3,0)
    finsi
  finproc
proc elimina
  pregunta;"Borrar. ¿Seguro? (s/n)"
  leer txt$
  si minùsc(txt$)="s"
    borrar
    finsi
  finproc
proc escmens
  pregunta;"Pulsa ESC para parar"
  finproc
proc i
  insertar
  finproc
proc intcodjug
  pregunta;"¿Código de juguete? "
  leer nuecod$
  haz nuecod$=mayùs(nuecod$)
  finproc
proc inverso
  escmsg
  mientras recnum()>0
    a
    escribir
    finmientras
  finproc
proc jugins
  intcodjug
  situar nuecod$
  si codigo$=nuecod$
    advierte;"Código en uso"
    sino
    haz codigo$=nuecod$
    juglim
    error jugleer
    comperr
    finsi
  finproc
proc jugleer
  pleer nombre$,coste,precio
  añadir
  finproc
proc juglim
  haz nombre$=""
  haz coste=0
  haz precio=0
  escribir
  finproc
proc p
  pròximo
  si finf()
    advierte;"Fin de fichero"
    finsi
  finproc
proc pidecom
  haz otrocod$=""
  mientras 1
    si codigo$<>otrocod$
      unereg
      finsi
    escribir
```

```
    blanco;13,5
    escribir en 12,5;"¿Comando?: "; papel 6;" "; papel 0;"(Pulse ? para obtener
ayuda)"
    haz c$=mayùs(tecla())
    blanco;14,5
    escribir en 12,16;c$
    compcom
    finmientras
  finproc
proc pregunta;mens$
  escribir en 13,5; tinta 4;mens$;
  finproc
proc saltar
  pregunta;"¿Registros a saltar?"
  leer num
  posición recnum()+num
  finproc
proc start
  limpiar
  escribir en 5,10; tinta 2; papel 6;"Abriendo ficheros ..."
  error cerrartodo
  abrir "ventas" lógico "v"
  ordenar codigo$;a.mesia
  abrir "juguet" lógico "j"
  ordenar codigo$;a
  haz f$="juguetes"
  pcargar "venjug"
  pantalla
  pidecom
  finproc
proc unejug;busca$
  usar "j"
  situar busca$
  escribir
  usar "v"
  finproc
proc unereg
  haz otrocod$=codigo$
  si f$="ventas"
    unejug;codigo$
    sino
    uneven;codigo$
    finsi
  finproc
proc uneven;busca$
  usar "v"
  situar busca$
  si v.codigo$<>busca$
    blanco;8,0
    haz otrocod$=""
    sino
    escribir
    finsi
  usar "j"
  finproc
proc ventins
  intcodjug
  unejug;nuecod$
  si j.codigo$=nuecod$
    haz codigo$=nuecod$
    ventlim
    error ventleer
    comperr
    sino
    advierte;"No existe el juguete"
```

```
    finsi
  finproc
proc ventleer
  pleer mes,cant,desc
  haz benef=cant*(j.precio*(1-desc/100)-j.coste)
  añadir
  pescribir
  finproc
proc ventlim
  haz mes=0
  haz cant=0
  haz desc=0
  haz benef=0
  pescribir
  finproc
proc vermas
  mientras hallado()
    pregunta;"Pulse cualquier tecla para ver más, o P para parar"
    si mayús(tecla())="P"
      regreso
      sino
      continuar
      pescribir
      finsi
    finmientras
  advierte;"No encuentro mas "+txt$
  finproc
proc zoom
  escmens
  mientras no finf()
    pescribir
    p
    finmientras
  finproc
```

EJERCICIOS

Nota: Utilice una cinta de seguridad especial para los ejercicios.

1. Altere el procedimiento borrar para que no elimine un registro de juguetes si todavía existen los registros de ventas de ese juguete.
2. Divida el procedimiento compcom en dos, llamando a ambos desde pidecom de acuerdo con el valor de c$. Esto mejorará la respuesta del programa al apretar una tecla, y es especialmente útil con comandos como p o a. ARCHIVE sólo tiene que leer la mitad de líneas del tipo si c$ cada vez.
3. Cambie la forma en que se alteran los registros para que el campo código$ no se pueda cambiar, e incluya el

cálculo del beneficio en los registros de ventas. Utilice ventleer y jugleer para hacer esto, unido con actualiz.

RESPUESTAS SENCILLAS

1. Añada unas cuantas líneas a borrar para que busque un registro de ventas si el fichero actual es juguet:

```
proc borrar
   pregunta; "Confirme borrado (s/n)"
   leer txt$
   si minusc(txt$) = "s"
      si f$ = "juguet"
         uneven; código$
         si v.código$ = j.código$
            advierte; "existen ventas para ese juguete"
            sino
            borrar
            finsi
         sino
         borrar
         finsi
      finsi
   finproc
```

2. Divida físicamente compcom en dos procedimientos (usando cortar y mezclar), y compruebe c$ en pidecom utilizando enserie( ) para determinar a cuál llamar.

```
proc compcom
  si c$="S": stop : finsi
  si c$="B":b: finsi
  si c$="P": pròximo : finsi
  si c$="A": anterior : finsi
  si c$="C": error c:comperr: finsi :nota cambiar
  si c$="I"
    si f$="ventas":ventins: sino :juginsi: finsi
    finsi
  finproc
```

```
proc compcom2
  si c$="X":cambiar: finsi
  si c$="E":elimina: finsi
  si c$="+":saltar: finsi
  si c$="R": error inverso: finsi
  si c$="1": primero : finsi
  si c$="U": último : finsi
  si c$="Z": error zoom: finsi
  si c$="?":ayuda: finsi
  finproc
```

Alterar la línea compcom en pidecom para leer:

```
proc pidecom ......
  si enserie("PVAIRZS",c$):compcom2:sino:compcom:finsi
  finproc
```

3. Cree nuevos procedimientos para añadir si c$ es I, o actualiz si es C. Llame a este procedimiento desde jugleer y ventleer (en lugar de usar añadir directamente).

```
proc jugleer
  pleer nombre$,coste,precio
  salvcambio
  finproc

proc salvcambio
  si c$ = "I"
    añadir
    sino
    actualiz
  finsi
finproc
```

```
proc ventleer
  pleer mes,cant,desc
  haz benef=cant*(j.precio*(1-desc/100)-j.coste)
  salvcambio
  escribir
  finproc
```

En el procedimiento compcom2: borre error a de la línea desde la cual se llama al procedimiento a, y en su lugar añada:

```
proc compcom2
  si c$="C"
    si f$="ventas": error ventleer: sino ; error jugleer:comperr: finsi
    finsi
  finproc
```

Borre el procedimiento a.

Nota: Recuerde añadir estas soluciones a su cinta de trabajo.

12

Listado, paginación y recuento

Las listas escritas a mano y mecanografiadas sufren de un inconveniente mayor que el entumecimiento y la fatiga de la muñeca del escritor: si se necesita un cambio total, lo más probable es que deba empezarse por tirarlas.

ARCHIVE imprime los mismos datos en tantos formatos, secuencias y selecciones como se precise, y tan a menudo como sea necesario. Todo lo que se desperdicia es algo de papel, cinta de impresora y mucho menos tiempo que el que llevaría hacerlo manualmente.

ARCHIVE también realiza referencias, cálculos, promedios, totales y subtotales con las listas impresas de cualquier forma que se precise, prescindiendo efectivamente de las largas horas perdidas buscando en tablas de números y tecleándolos de nuevo en una calculadora.

Este capítulo explica cómo:

- Escribir programas de listado que funcionan con pantalla o impresora.
- Determinar el tipo (textual o numérico) de un campo.
- Listar dos ficheros unidos.
- Detener el paso de la información por pantalla.
- Usar campos para subtotales y promedios.

- Se presentan los siguientes comandos de ARCHIVE:

local	— marca una variable como local para el procedimiento en que se usa.
modo	— cambia el ancho de pantalla, y permite elegir si se muestran las tres áreas o no.

- Se presentan las siguientes funciones ARCHIVE:

tipocam( )	— se usa para descubrir si un campo es de tipo numérico o de caracteres.
pulsada( )	— comprueba si se ha pulsado una tecla y devuelve su valor en caso afirmativo.

Puntos a recordar

- Cuando hay varios bucles anidados, la acción que ocurre más a menudo debe estar en el bucle central.
- Los totales se calculan usando bucles con contadores. Estos deben ponerse a cero al entrar, incrementarlos, y acabados cuando se alcanza el fin de los registros que deben totalizarse.

Ejemplo: Listado de ventas de juguetes

Juan ha completado los procedimientos para la inserción, modificación y visualización de los registros de ventas y juguetes.

La mayoría de los programas de ordenador constan de tres partes distintas: entrada, proceso de datos y salida. El proceso de datos es el propósito principal del uso de un ordenador: trabajos tales como el ordenamiento de datos y la ejecución de cálculos se hacen más deprisa y con mayor seguridad en ordenador que a mano.

El proceso tiende a tener lugar tanto en el momento de introducción de datos (por ejemplo, calculando el beneficio) como en la salida (por ejemplo, totalizando las ventas), así que los procedimientos de Juan se dividen en sólo dos grupos: los que conciernen a la entrada y los que conciernen a la salida de datos.

Ahora presta atención a los procedimientos de salida, diseñados para listar los registros (con totales) sobre la pantalla o la impresora.

INSTRUCCIONES

1. abrir el fichero juguet con el nombre lógico j.
2. abrir el fichero ventas con el nombre lógico v.

Los dos ficheros deberían estar ya ordenados, ya que fueron salvados tras una ordenación.

Si los ficheros ya están abiertos, salve todos los procedimientos en memoria antes de seguir.

3. usar el fichero ventas, si no es ya el fichero actual.

¿Listar sobre pantalla o impresora?

Juan decide usar uno o dos de los procedimientos de listado de Eva como piedra angular de su programa, pero borra los que no va a necesitar.

cargar "listado" ↵

1. Cargue el programa listado.
2. Borre los procedimientos listinvit, listagenda, impreg, impragenda y zoom, que no serán ya de utilidad.
   Los procedimientos restantes deben ser escrec, ff, línea y lista.

Los procedimientos de listado de Juan se llamarán, desde pidecom, con una simple pulsación de tecla, igual que todos los otros comandos. Asigna la letra l para enviar las listas a la impresora, y v, para enviarlas a la pantalla, o *VDU (Visual Display Unit)*, ya que la letra p ya está en uso para próximo.

Mejor que escribir conjuntos separados de procedimientos para estas dos condiciones diferentes, adapta línea a las dos condiciones distintas: escribir o imprimir.

Por el momento, línea aparece así:

```
proc línea;1$
    imprimir 1$
    finproc
```

3. Haga línea el procedimiento actual.

si c$ = "L" ↵ imprimir l$; ↵

4. Añade una cláusula si para que si c$ (la variable comando) vale l, el procedimiento mande a la impresora los caracteres pasados en l$. Si no, escribirá en la pantalla. Advierta el punto y coma después de l$ para suprimir el

retorno de carro automático, porque las ventas se imprimirán en un formato tabular, un registro en cada línea, mediante muestreg.

sino ↵ escribir l$; ↵

5. Introduce una cláusula sino para enviar la información a la pantalla si c$ no es L. Esta línea también acaba con un punto y coma.

finsi **6.** Fin de la cláusula si.

Debería quedar así:

```
proc línea;l$
    si c$ = "P"
    imprimir l$;
    sino
    escribir l$
    finsi
finproc
```

Cambio del procedimiento MUESTREG

1. Haz muestreg el procedimiento actual de nuestra atención.
Por el momento aparecerá esto:

```
proc muestreg
    haz cuenta = 0
    mientras cuenta< numcam( )
        escribir valcam(cuenta)
        haz cuenta = cuenta + 1
        finmientras
finproc
```

proc listreg **2.** Cambia el nombre del procedimiento a listreg para clarificar su propósito.

línea;valcam(cuenta) + " " **3.** Cambia la línea escribir valcam(cuenta) en el nuevo procedimiento listreg para que llame a línea de la siguiente manera:

```
línea;valcam(cuenta) + " "
```

Observe que se ha añadido un blanco al campo usando un signo más en vez de un punto y coma, porque a línea se le debe pasar un solo parámetro.

linearc **4.** Añada una línea linearc después del bucle para llamar

a un procedimiento (descrito más adelante) que imprimirá un retorno de carro después de que el registro ha sido impreso en una línea.

El procedimiento quedará así:

```
proc listreg
    haz cuenta = 0
    mientras cuenta< numcam( )
        línea; valcam(cuenta) + " "
        haz cuenta = cuenta + 1
        finmientras
    linearc
    finproc
```

El procedimiento LINEARC

```
proc linearc ↵
línea; car(13) + car(10) ↵
finproc ↵
```

1. Cree el procedimiento linearc como sigue:

```
línea;car(13) + car(10)
finproc
```

El procedimiento linearc envía los caracteres ASCII CR y LF a la impresora o a la pantalla, dado que el retorno de carro se ha suprimido en *línea* usando punto y coma.

Cambio de LISTA

1. Haga lista el procedimiento actual de nuestra atención.

Por el momento aparece esto:

```
proc lista
    primero
    limpiar
    mientras no finf( )
        muestreg
        próximo
        finmientras
    finproc
```

listreg

2. Cambie la línea muestreg en el procedimiento lista para que ahora llame a listreg.

ESC

3. Abandone editar.
4. Salve el procedimiento como listjug.

haz c$ = "V"

5. Teclee haz c$ = "V" para declarar la variable y preparar línea para usar la pantalla.

lista

6. Escriba lista para ejecutar el procedimiento.

La pantalla se borra y empieza listar el primer registro del fichero activo, pero inmediatamente se para con un mensaje de error.

```
Error 72 : Valores incompatibles
listreg  : línea;1$
```

El procedimiento línea espera que se le pase un carácter como parámetro. El valor de campo devuelto por valcam( ) es numérico tan pronto como se pide el segundo campo (mes).

La función TIPOCAM( )

La forma de superar este problema y continuar el uso de listreg y línea de la manera deseada es convertir cualquiera de los campos numéricos o caracteres usando la función serie( ).

Primero, sin embargo, hace falta poder decir qué campos son numéricos.

Se puede usar la función tipocam( ) para determinar el tipo (cadena de caracteres o número) de cualquier campo.

Como la función valcam( ), tipocam( ) devuelve un valor para el campo cuyo número aparece entre los paréntesis.

Por ejemplo, escribir tipocam(0) muestra un número indicando el tipo del primer campo (campo 0) del fichero activo.

Si el campo es un conjunto de caracteres, tipocam( ) devuelve el valor 1. Si el campo es numérico, devuelve el valor 0.

Cambie el procedimiento listreg para que el valor devuelto por valcam( ) se convierta en una cadena de caracteres en formato general si el campo es numérico. Añada una sentencia si que use tipocam( ) para comprobar cuándo es necesaria la conversión.

Recuerde que serie( ) en el formato general muestra las cifras decimales que tenga el número.

1. Use editar y TAB hasta el procedimiento listreg.
2. Añada la siguiente línea al principio del bucle:

```
si tipocam(cuenta)
```

Recuerde que la variable cuenta comienza en 0 y se incrementa hasta que iguala al número de campos del fichero.

La línea si tipocam(cuenta) es una forma abreviada de

si tipocam(cuenta) = 1. Ejecuta las sentencias siguientes si la función devuelve el valor 1 (el campo es una cadena de caracteres).

3. Inserte estas líneas a continuación:

```
sino
línea; serie(valcam(cuenta),3,0) + " "
finsi
```

Si el número devuelto por tipocam( ) es 0 (el campo es numérico), el procedimiento ejecuta la línea después de sino. Así convierte el valor de campo en una serie de caracteres antes de añadir un espacio y pasarlo como un parámetro a línea.

El procedimiento alterado aparece así:

```
proc listreg
    haz cuenta = 0
    mientras cuenta< numcam( )
        si tipocam(cuenta)
            línea; valcam(cuenta) + " "
            sino
            línea; serie (valcam(cuenta),3,0) + " "
            finsi
        haz cuenta = cuenta + 1
         finmientras
     linearc
     finproc
```

4. Abandone editar y escriba el comando lista.
Esta vez se listan todos los registros del fichero activo, una línea para cada uno:

```
codigo$ mes  cant  desc  benef
LIT       1    10     0   14120
LIT       2   180     0  254160
LIT       3   310     0  437720
LIT       4   320     0  451840
LIT       5   230     0  324760
LIT       6   260     0  367120
LIT       7   -20     0  -28240
LIT       8   210    20  212940
LIT       9   180     0  254160
LIT      10   135     0  190620
LIT      11   120     0  169440
LIT      12    90     0  127080
LOL       7   525    50   97125
LOL       8   475    50   87875
```

```
LOL        9   330    20 234300
LOL       10   260    20 184600
LOL       11   210     0 222600
LOL       12   300    50  55500
```

El listado sale de la pantalla, sin embargo, y los primeros registros no se leen adecuadamente.

Interrupción del listado: cuenta de líneas

Juan decide crear un procedimiento que use tecla( ) para detener el listado cuando la pantalla esté llena. Una simple pulsación de tecla permite continar el listado.

editar

1. Escriba el comando editar.
2. Cree el siguiente procedimiento:

```
proc pausa
   escribir "Pulse una tecla para continuar";tecla( )
   finproc
```

El AREA DE VISUALIZACION vacía acepta 14 líneas de impresión antes de llenarse. El procedimiento pausa se usará cuando se impriman 13 líneas en la pantalla. Para saber cuándo hacerlo, el programa cuenta el número de líneas impresas. Una vez que la pantalla está llena, la cuenta tendrá que ponerse a cero para empezar a contar la siguiente pantalla completa.

Añada algunas líneas a linearc para contar las líneas e investigar si es el momento de parar.

haz c1 = c1 + 1

3. Haga **TAB** hasta el procedimiento linearc e inserte la línea haz c1 = c1 + 1 al final. Cada vez que se produce un retorno de carro, comienza una nueva línea.
   La variable c1, abreviatura de cuenta líneas, se usa para almacenar el número de líneas impresas. Se incrementa cada vez que comienza una nueva línea.

si c1 = 13 ↵ líneamax ↵ finsi ↵

4. Inserte la siguiente sentencia si:

```
si c1 = 13
   líneamax
   finsi
```

El procedimiento líneamax (descrito luego) se activa cada vez que c1 alcanza el valor 13.

El procedimiento acabado queda así:

```
proc linearc
    línea; car(13) + car(10)
    haz c1 = c1 + 1
    si c1 = 13
        líneamax
        finsi
    finproc
```

Interrupción de listados con PULSADA()

Una manera alternativa de parar el listado es hacer que éste continúe hasta que se pulsa una tecla, mejor que ir parando a intervalos regulares.

La función pulsada( ) es como la función tecla( ), en que busca un solo carácter desde el teclado. A diferencia de tecla( ), pulsada( ) no detiene la ejecución del procedimiento, sino que "mira" si se ha pulsado una tecla.

pulsada( ) se puede usar cada vez que se imprime una línea para comprobar si se ha pulsado una tecla y detener el listado; por ejemplo, si pulsada( ) >"":pausa: finsi.

Observe que pulsada( ) no funciona muy bien en las primeras versiones de ARCHIVE, por lo que no es muy de fiar y no la usaremos. En la versión 2 no debe tener ningún problema.

Si se llenó la pantalla o página

```
haz c1 = 0
```

```
si c$ = "L" ↵ ff ↵
```

1. Cree el procedimiento lineamax.
2. Escriba la primera línea haz c1 = 0.
   Así pone el contador de línea a cero, preparado para empezar a contar de nuevo.
3. Ahora escriba las dos líneas siguientes:

```
si c$ = "L"
    ff
```

Así, lanza el papel al principio de la siguiente hoja si está usando la impresora.
El procedimiento ff ya existe, proviene de listado.

sino ↵ pausa ↵

4. Escriba ahora dos líneas que llaman al procedimiento pausa:

```
sino
pausa
```

Si el valor de c$ no es L (el listado sale por la pantalla), detiene el programa.

finsi

5. Finalice de escribir el procedimiento con una línea finsi.

El procedimiento acabado queda como sigue:

```
proc lineamax
    haz c1 = 0
    si c$ = "L"
    ff
    sino
    pausa
    finsi
finproc
```

si c1 = max1

6. Cambie el número 13 en el procedimiento linearc por la variable max1; la línea queda: si c1 = max1.
Así permite asignar un número distinto de máximo de línea en función de si los listados se van a sacar por impresora o por pantalla.
Un número típico de líneas sobre papel continuo de impresión de ordenador es 66, por lo que una cantidad de 55 deja márgenes satisfactorios arriba y abajo.

> *Nota:* Los distintos modos de presentación de ARCHIVE (con o sin el AREA DE TRABAJO y el AREA DE CONTROL) requerirán también distintos máximos de línea. La presentación se puede cambiar usando el comando modo.

El procedimiento queda ahora así:

```
proc linearc
    línea; car(13) + car(10)
    haz c1 = c1 + 1
    si c1 = max1
        lineamax
        finsi
    finproc
```

La sentencia MODO

modo cambia el número de columnas mostradas en la pantalla, y también muestra o no las AREAS DE CONTROL y DE TRABAJO.

modo debe ser seguido por dos números (o expresiones), separados por una coma.

Si el primer número es 1, las áreas aparecen unidas: si es 0, están separadas.

Hay 14 líneas de visualización con las áreas activas, y 24 cuando no lo están. Comandos y líneas de impresión aparecen uno a continuación de otro cuando no hay AREA DE TRABAJO.

Una errata en las primeras versiones de ARCHIVE hace que editar "cuelgue" el programa si se usa sin AREA DE TRABAJO y más de 19 procedimientos.

El segundo número usa la primera cifra de los tres posibles anchos de pantalla (40, 64 y 80) para designar cuál debe ser usado.

Ejemplos:

modo 1,6	3 áreas separadas y 64 columnas (como pulsando **F2** para cargar ARCHIVE).
modo 1,8	3 áreas y 80 columnas (como pulsando **F1** para cargar ARCHIVE).
modo 0,4	las tres áreas juntas y 40 columnas.

Cuando sólo se elimina el AREA DE CONTROL (usando la tecla **F2**), hay 19 líneas en el AREA DE VISUALIZACION.

Declaración de variables de trabajo

1. Cree un procedimiento llamado inicia para definir temporalmente los valores iniciales de las variables necesarias para estos procedimientos.
Cuando se usen más tarde en un programa se controlarán por otros procedimientos, e inicia llegará a ser redundante.

haz c$ = "V"

2. Teclee la siguiente línea para poner la variable c$ que determina enviar los listados a pantalla o impresora ("V" las envía a la pantalla).

haz c1 = 0 ↵ haz max1 = 13 ↵ haz c$ = "V"

3. Ponga la variable c1 a 0, y max1 a 13:

```
haz c1 = 0
haz max1 = 13
```

lista **4.** Haga que inicia llame al procedimiento lista.

El procedimiento deberá quedar así:

```
proc inicia
    haz c$ = "V"
    haz c1 = 0
    haz max1 = 13
lista
finproc
```

5. Abandone editar y...
6. Salve el procedimiento como listjug2.

inicia **7.** Teclee inicia para iniciar las variables, y ver los resultados.

Los registros aparecen en pantalla hasta casi el final, entonces el programa para y aparece el mensaje de pausa. Pulse una tecla para seguir con el listado.

Los procedimientos funcionan juntos de la siguiente manera:

```
)proc linea;i$
  si c$="L"
    imprimir i$;
    sino
    escribir i$;
    finsi
  finproc
proc lineamax
  haz c1=0
  si c$="L"
    ff
    sino
    pausa
    finsi
  finproc
proc linearc
  linea;car(13)+car(10)
  haz c1=c1+1
  si c1=max1
    lineamax
    finsi
  finproc
proc listreg
  haz cuenta=0
  mientras cuenta<numcam()
    si tipocam(cuenta)
```

```
      linea;valcam(cuenta)+" "
      sino
      linea;serie(valcam(cuenta),3,0)+" "
      finsi
    haz cuenta=cuenta+1
    finmientras
  linearc
  finproc
```

Totalización de beneficios

Juan se aproxima al último objetivo de todo su esfuerzo: subtotalizar y comparar los beneficios de lagartos y llamas.

Añade unas pocas líneas a los procedimientos para usar un acumulador que almacene un total (acumulado) de los beneficios. Este se incrementa a medida que se imprime cada registro, y el resultado se imprime al final.

Un total actualizado trabaja exactamente como un contador en un bucle. Empieza con un valor de la variable igual a cero antes del bucle. Cada registro añade su valor a la variable hasta que el bucle finaliza.

Por ejemplo, la siguiente línea totaliza el campo benef en el fichero ventas:

```
haz tot = 0: todos "v": haz tot = tot + v.benef: fintodos:
escribir tot
```

1. Edite los procedimientos en memoria y haga a lista el procedimiento activo.
2. Inserta la línea haz tot = 0 antes del bucle en lista. Así pone a cero el acumulador.
3. Inserte esta línea en el bucle:

   ```
   haz tot = tot + benef
   ```

   Así se incrementa el acumulador.
4. Inserte la siguiente línea después del bucle:

   ```
   línea;"*BENEFICIO TOTAL:" + serie(tot,3,0)
   ```

   para imprimir el total acumulado.

El procedimiento aparecerá así después de los cambios:

```
proc lista
  primero
```

```
limpiar
haz tot = 0
   mientras no finf()
   listreg
   haz tot = tot + benef
   próximo
   finmientras
línea; "*BENEFICIO TOTAL: " + serie(tot,3,0)
finproc
```

5. Abandone el editor y escriba inicia para comprobar la totalización.

Subtotales por juguetes

Los subtotales funcionan exactamente de la misma manera, pero se imprimen y se ponen a cero de nuevo cada vez que cambia el grupo que se está calculando.

Juan crea un nuevo procedimiento que lista y subtotaliza las ventas sólo cuando tengan el mismo código de juguete. Dado que el fichero está ordenado por código$, se puede hacer fácilmente almacenando el código inicial y usando un bucle hasta que el código actual sea distinto.

1. Use editar para modificar los procedimientos y haga lista el procedimiento activo.
2. Inserte la línea haz ultcod$ = código$ al principio del bucle. Así almacena el valor actual de código$.
3. Cambie la línea listreg por listventas.
   Así llama a un procedimiento que lista los registros mientras código$ sea igual a ultcod$.
4. Cambie la línea haz tot = tot + benef para que quede:

   haz tot = tot + subtot

   La variable subtot se acumula en listventas, y se añade a tot cada vez que acaba.
5. Quite la línea próximo del procedimiento lista.
   Cuando hay más de un bucle en uso, el comando próximo debe ejecutarse en el más interior; en este caso, listventas.

El procedimiento queda ahora así:

```
proc lista
   primero
```

```
limpiar
haz tot = 0
mientras no finf( )
    haz ultcod$ = código$
    listventas
    haz tot = tot + subtot
    finmientras
línea; "* BENEFICIO TOTAL: " + serie(tot,3,0)
finproc
```

Listado y totalización de ventas de un juguete

1. Use editar para crear el procedimiento listventas.

haz subtot = 0

2. La primera línea debe ser: haz subtot = 0.
   Así pone a cero la variable subtot, el acumulador del subtotal.

mientras no finf( ) yy código$ = ultcod$

3. La siguiente línea, mientras no finf( ) yy código$ = ultcod$, comienza un bucle que continúa mientras el código$ en el registro de ventas siga igual que ultcod$. Advierta que no finf( ) aparece tanto en este bucle como en todos los bucles de lista. Sin esta condición, el programa se quedaría para siempre en este bucle, imprimiendo el último registro, que tendrá el valor del último ultcod$.

listreg

4. Escriba una línea que llame al procedimiento listreg.

haz subtot = subtot + benef

5. La siguiente línea, haz subtot = subtot + benef incrementa el acumulador.

próximo

6. Escriba ahora próximo.
   Recuerde que, si los bucles están anidados, próximo debe estar en el bucle central.

finmientras

7. Cierre el bucle mientras con la línea finmientras.
8. Escriba la línea:

   ```
   línea; "*BENEFICIO de" + ultcod$ + ":" +
   serie(subtot,3,0)
   ```

   que imprime el subtotal.

linearc

9. Ahora escriba linearc.
   Así empieza una nueva línea después de imprimir el subtotal, e incrementa el contador de líneas.

El procedimiento final quedará así:

```
proc ventas
  haz subtot = 0
  mientras no finf( ) yy código$ = ultcod$
     listreg
     haz subtot = subtot + benef
     próximo
     finmientras
   línea;''*BENEFICIO de'' + ultcod$ + :'' + serie
   (subtot,3,0)
   linearc
   finproc
```

10. Abandone editar.
11. Pruebe estos procedimientos de listado llamando al procedimiento inicia.

```
LIT 1 10 0 14120
LIT 2 180 0 254160
LIT 3 310 0 437720
LIT 4 320 0 451840
LIT 5 230 0 324760
LIT 6 260 0 367120
LIT 7 -20 0 -28240
LIT 8 210 20 212940
LIT 9 180 0 254160
LIT 10 135 0 190620
LIT 11 120 0 169440
LIT 12 90 0 127080
* BENEFICIO de LIT: 2775720
LOL 7 525 50 97125
LOL 8 475 50 87875
LOL 9 330 20 234300
LOL 10 260 20 184600
LOL 11 210 0 222600
LOL 12 300 50 55500
* BENEFICIO de LOL: 882000
* BENEFICIO TOTAL: 3657720
```

Promedio de ventas

Los subtotales mostrados no son muy significativos, porque la llama ''Lola'' solamente ha estado en venta durante seis meses, mientras que el lagarto ''Lito'' ha estado un año.

Juan añade un último contador que cuenta el número de registros de ventas de cada juguete para que el programa calcule el beneficio medio.

1. Use editar y haga listventas el procedimiento actual.
2. Inserte haz cuenta = 0 antes del bucle.
3. Inserte haz cuenta = cuenta + 1 en el bucle.

4. Inserte después del bucle:

   línea; "* PROMEDIO: " + serie(subtot/cuenta,3,0)
   linearc

   El beneficio medio mensual es el valor de los beneficios subtotalizados dividido por el número de meses.

El procedimiento quedará así:

```
proc listventas
haz subtot = 0
haz cuenta = 0
mientras no finf( ) yy código$ = ultcod$
   listreg
   haz subtot = subtot + benef
   haz cuenta = cuenta + 1
   próximo
   finmientras
línea;"*BENEFICIO de" + ultcod$ + ":" serie(subtot,3,0)
linearc
línea; "*PROMEDIO: "serie(subtot/cuenta,3,0)
linearc
finproc
```

Variables locales

Observe que la variable cuenta tiene el mismo nombre que una variable de **listreg**, que es llamada por **listventas** una vez por cada registro de ventas con el mismo código de juguete.

Para empezar, **listreg** corrige el valor de cuenta en **listventas**, dejándole con el valor del número de campos en el registro más 1. Obviamente así se arruina la posibilidad de contar con seguridad el número de registros.

Para superar este problema, use el comando **local** en el procedimiento **listreg**.

local cuenta

1. **TAB**ule hasta el procedimiento **listreg** e inserte la línea **local cuenta** al principio del procedimiento.

   El procedimiento queda así:

```
proc listreg
   local cuenta
   haz cuenta = 0
   mientras cuenta<numcam( )
```

```
    si tipocam(cuenta)
      línea; valcam(cuenta) + " "
      sino
      línea; serie(valcam(cuenta),3,0) + " "
      finsi
    haz cuenta = cuenta + 1
    finmientras
  linearc
  finproc
```

2. Abandona editar, y...
3. Salve los procedimientos en listjug3.
4. Escriba inicia para comprobar que los procedimientos funcionan:

inicia

```
LIT 1 10 0 14120
LIT 2 160 0 254160
LIT 3 310 0 437720
LIT 4 320 0 451840
LIT 5 230 0 324760
LIT 6 260 0 367120
LIT 7 -20 0 -28240
LIT 8 210 20 212940
LIT 9 180 0 254160
LIT 10 135 0 190620
LIT 11 120 0 169440
LIT 12 90 0 127080
* BENEFICIO de LIT: 2775720
* PROMEDIO: 231310
LOL 7 525 50 97125
LOL 8 475 50 87875
LOL 9 330 20 234300
LOL 10 260 20 184600
LOL 11 210 0 222600
LOL 12 300 50 55500
* BENEFICIO de LOL: 882000
* PROMEDIO: 147000
* BENEFICIO.TOTAL: 3657720
```

El comando LOCAL

El comando local hace que cualquier nombre de variable que siga a la declaración local se use como local en el procedimiento.

Se llama variables *locales* de un procedimiento (opuestas a *globales*), a las que cualquier cosa que les suceda en el procedimiento se ignora por los otros procedimientos del programa.

Las variables, con el mismo nombre en otros procedimientos, no son afectadas por variables locales a los procedimientos que llaman, o por los que son llamadas.

En el caso de listventas y listreg, el valor de cuenta varía como sigue cuando hay (por ejemplo) tres ventas con el mismo código de juguete:

En el procedimiento:	*Valor de cuenta:*
listventas	:0
listreg	:0,1,2,3,4,5(primer registro)
listventas	:1
listreg	:0,1,2,3,4,5(segundo)
listventas	:2
listreg	:0,1,2,3,4,5(tercero)
listventas	:3

Unión de ficheros

Juan comienza a considerar la unión de los dos ficheros en los listados, uniendo los procedimientos listjug con los de juguet.

Primero altera el procedimiento principal de listado para hacer juguet el fichero de datos actual y listar cada juguete antes de moverlo a listventas.

Cambia entonces listventas para que use el fichero ventas completo y encuentre los registros de ventas unidos al juguete actual.

1. Use edit y haga lista el procedimiento activo.

```
usar "j" ↵   haz f$ = "juguetes" ↵
```

2. Inserte estas dos líneas al principio del procedimiento para asegurar que el fichero juguet es el fichero actual cuando comience el listado.

```
usar "j"
haz f$ = "juguetes"
```

listreg

3. Inserte listreg al principio del bucle para imprimir el registro de juguete actual antes de ir a listar las ventas del juguete.
4. Borre la línea haz ultcod$ = código$, ya que ultcod$ será ahora el código$ del fichero juguetes.

próximo

5. Escriba próximo, que en este caso afecta al fichero de juguetes.

El procedimiento cambiado queda así:

```
proc lista
   usar "j"
   haz f$ = "juguetes"
   primero
   limpiar
   haz tot = 0
     mientras no finf( )
     listreg
     listventas
     haz tot= tot + subtot
     próximo
     finmientras
   línea; "* BENEFICIO TOTAL: " + serie(tot,3,0)
   finproc
```

6. Use **TAB** hasta el procedimiento listventas.

usar "v"

7. Inserta la línea usar "v" al principio del procedimiento. El fichero ventas es el fichero actual en este procedimiento.

situar j.código$

8. Escriba situar j.código$.
   Si hay registros de ventas del juguete actual, situar encuentra el primero.
   Si no hay ninguno, el bucle falla antes de comenzar, dado que el código$ de las ventas no es el mismo que j. código$.

j. código$

9. Cambie ultcod$ por j. código$ en el resto del procedimiento.

usar "j"

10. Inserte la línea usar "j" al final del procedimiento para hacer juguet de nuevo el fichero actual.

El procedimiento quedará así:

```
proc listventas
   usar "v"
   situar j.código$
   haz subtot = 0
   haz cuenta = 0
   mientras no finf( ) yy código$ = j.código$
      listreg
      haz subtot = subtot + benef
      haz cuenta = cuenta + 1
      próximo
      finmientras
   línea; "* BENEFICIO de " + j.código$ + ":" + serie
      (subtot,3,0)
```

```
linearc
línea; "* PROMEDIO: " + serie(subtot/cuenta,3,0)
linearc
usar "j"
finproc
```

11. Borre el procedimiento inicia, que no será necesario una vez que las variables se coloquen en los procedimientos pidecom y compcom.
12. Abandone editar, y
13. Salve los procedimientos en el fichero de programa listjug.

Unión de los dos programas

Juan está ya listo para unir los procedimientos de juguet y crea un procedimiento para unirlos.

```
unir "juguet"
```

1. Teclee unir "juguet" para crear un gran programa con todos los procedimientos de entrada y salida.

```
editar
```

2. Escriba editar y use **TAB** hasta el procedimiento compcom.

```
si c$ = "L": haz max1 = 55:listselec: finsi ↵
si c$ = "V": haz max1 = 14:listselec: finsi ↵
```

3. Inserte dos líneas para crear los comandos 1 (de impresora de líneas) y V (de VDU).
Observe que las dos líneas llaman al mismo procedimiento, pero con valores diferentes de max1.

```
linearc ↵ pausa ↵
```

4. Use **TAB** hasta el procedimiento lista e inserte estas dos líneas al final del procedimiento.
Así, asegura una pausa al final del listado para que se puedan leer los totales en pantalla antes de volver a pidecom.

```
usar "j" ↵ haz f$ = "juguet" ↵ primero ↵ limpiar ↵
```

5. Quite estas cuatro líneas del procedimiento para que se puedan pasar a uno nuevo.
Tenga cuidado de no borrar lo quitado (en la memoria intermedia de edición), ya que va a formar la base de un nuevo procedimiento.

El procedimiento queda así:

```
proc lista
  haz tot = 0
  mientras no finf( )
     listreg
     listventas
```

```
    haz tot = tot + subtot
    próximo
    finmientras
línea; "* BENEFICIO TOTAL: " + serie(tot,3,0)
linearc
pausa
finproc
```

6. Cree el procedimiento listselec.
Este procedimiento se usará para definir los valores iniciales y leer el intervalo de registros necesario.
Recuerde: pulse **ESC** para dejar el modo inserción.

usar "j" ↵ haz f$ = "juguet" ↵ primero ↵ limpiar ↵

7. Incluya las líneas quitadas de **lista**.

haz c1 = 0 ↵ lista ↵ pantalla ↵

8. Inserte estas tres líneas para inicializar el contador de líneas, llamar al procedimiento de listado y volver a mostrar la pantalla de presentación.

El procedimiento acabado aparece así:

```
proc listselec
  usar "j"
  haz f$ = "juguetes"
  haz c1 = 0
  primero
  limpiar
  lista
  pantalla
  finproc
```

9. Actualice el procedimiento a1 para que tenga en cuenta los nuevos procedimientos. Añada las líneas siguientes:

```
nota ff: (lineamax) envía FF a la impresora.
nota línea: (listreg) escribe o imprime 1$, según el valor de c$.
nota linearc: (listreg) CR y LF a impresora o pantalla.
nota lineamax: (linearc) página a impresora, pausa a pantalla.
nota lista: (listselect) bucle de control, escribe el beneficio total.
nota listreg: (listventas) lista los campos, uno a uno en una línea.
nota listventas: (lista) lista las ventas de un juguete, subtotal y promedio.
nota listselec: (compcom) parámetros iniciales del listado.
nota pausa: (lineamax) pausa para seguir el listado.
```

10. Actualice el procedimiento ayuda para incluir los nuevos comandos L y V.
11. Abandone editar y
12. Salve los procedimientos unidos en un fichero llamado todojug.

Selección y listado parcial

Juan quiere que el listado de programas sea distinto: intenta listar sólo parte de los ficheros, en lugar de todos los registros.

Elegir el registro inicial es fácil. Añade líneas para leer el código de juguete inicial y lo sitúa antes de comenzar el listado.

La elección del registro final no es una gran preocupación. Borra la pregunta del registro inicial, pregunta por el registro final, y añade una expresión al bucle del fichero para que continúe sólo mientras el código actual sea menor o igual que el código elegido.

Serán impresos todos los códigos entre el inicial y el final.

El proceso de una secuencia de registros es simple si el fichero está ordenado por el campo elegido para la secuencia.

editar

1. Edite el procedimiento listselec.

```
pregunta;"¿Desde qué código?" ↵ leer desde$ ↵
```

2. Inserte las líneas siguientes al principio del procedimiento para leer la letra(s) desde la que comenzará el listado:

```
pregunta;"¿Desde qué código?"
input desde$
```

Se usará situar para localizar el principio; así, si se responde A se empezará con el primer registro.
De igual modo, Ll comienza con el primer registro que empiece por Ll o siguientes letras.

si desde$>""

3. Escriba la línea siguiente: si desde$>"".
Si se pulsa L o V accidentalmente, basta pulsar ↵. Así, desde$ es un carácter nulo y no se hace el listado. El procedimiento devuelve el control a pidecom.

```
haz desde$ = mayús(desde$) ↵ blanco;13,0 ↵ pregunta;¿Hasta qué código?" ↵
leer acod$ ↵ haz acod$ = mayús(acod$) ↵
```

4. Si desde$ no es un carácter nulo, estas líneas ponen a blancos la línea de mensajes, y convierten la elección inicial y final en mayúsculas.

situar desde$

5. Inserte la línea situar desde$ tras las líneas que referencian usar para el fichero juguet.
El registro inicial debe hacerse el registro activo antes de comenzar el listado.

6. Quite la línea primero por la misma razón.
7. Inserte la línea finsi al final del procedimiento para cerrar la sentencia si desde$ >"".

finsi

El procedimiento queda así:

```
proc listselec
   pregunta;"¿Desde qué código?"
   leer desde$
   si desde$ >""
      haz desde$ = mayús(desde$)
      blanco;13,0
      pregunta;"¿Hasta qué código?"
      leer acod$
      haz acod$ = mayús(acod$)
      usar "j"
      haz f$ = "juguetes"
      situar desde$
      haz c1 = 0
      limpiar
      lista
      pantalla
      finsi
  finproc
```

mientras no finf( ) yy código$< = acod$

8. Haga **TAB** hasta el procedimiento lista y añada la expresión yy código$ < = acod$ al final de la línea mientras, para que ponga:

   mientras no finf( ) yy código$< = acod$

   Así conseguirá que el listado continúe hasta el primer código mayor que el código elegido para finalizar o hasta el fin de fichero.

El procedimiento lista aparece así:

```
proc lista
   haz tot = 0
   mientras no finf( ) yy código$< = acod$
     listreg
     listventas
     haz tot = tot + subtot
     próximo
     finmientras
```

```
línea; "* BENEFICIO TOTAL: " + serie(tot,3,0)
linearc
pause
finproc
```

ESC

salvar "todojug"

start

9. Abandone editar y
10. Salve los procedimientos en todojug.
11. Teclee start para ver funcionar el programa completo:

```
LOL 7 525 50 97125
LOL 8 475 50 87875
LOL 9 330 20 234300
LOL 10 260 20 184600
LOL 11 210 0 222600
LOL 12 300 50 55500
* BENEFICIO de LOL: 882000
* PROMEDIO: 147000
* BENEFICIO TOTAL: 3657720
```

Formato de la salida numérica

Por el momento, las listas de Juan son funcionales, pero no agradables a la vista. Lo serían más si todos los números se alinearan en columnas de la misma anchura y las cifras estuvieran correctamente ajustadas a la derecha (por ejemplo, la cifra 9 de la columna mes apareciera sobre el 0 del 10 mejor que sobre el 1).

Esto puede hacerse fácilmente en ARCHIVE, versión 2, usando la función gen( ).

Conversión y formato de números

ARCHIVE versión 2 incluye tres funciones que realizan una tarea similar a la función serie( ), exceptuando que se puede fijar la longitud de la cadena de caracteres resultante, y que los números están ajustados a la derecha (la forma normal de representar números en columnas).

gen( ), formato general, es similar a serie( ) con valor 3 como segundo argumento.

Por ejemplo, gen(2*4,10) devuelve " 8"

DEC( ), formato decimal, es similar a serie( ) con valor 0 como segundo argumento.

Por ejemplo, dec (13.56897,3,15) devuelve " 13.568"

NUM( ), formato de número entero, es similar a serie( ) con 2 como segundo argumento.

Por ejemplo, num(23,10) devuelve " 23"

La función GEN( )

La función gen( ) es parecida a la función serie( ) con valor 3 (formato general) como argumento central. Convierte un número a una cadena de caracteres en formato general, es decir, lo hará sin limitar el número de cifras decimales.

La diferencia es que gen( ) puede calcular la longitud total de la cadena de caracteres (incluyendo los espacios blancos que se necesiten), y que el número aparece a la parte derecha de la cadena.

gen( ) necesita dos argumentos numéricos entre paréntesis. El primero es el número a convertir y el segundo es la longitud total de la cadena.

1. Edita el procedimiento listreg.

línea; gen(valcam(cuenta)),10) + " "

2. Cambia la segunda llamada a línea para que ponga:

```
línea; gen(valcam(cuenta)),10) + " "
```

El procedimiento ahora queda así:

```
proc listreg
  local cuenta
  haz cuenta = 0
  mientras cuenta< numcam( )
  si tipocam(cuenta)
    línea; valcam(cuenta) + " "
    sino
    línea; gen(valcam(cuenta),10) + " "
    finsi
  haz cuenta = cuenta + 1
  finmientras
  linearc
  finproc
```

3. Abandone la edición y escriba pidecom, seguido de v para ponerlo en marcha.

RESUMEN

Juan usa el programa listado de Eva para listar sus informes de ventas por pantalla o impresora. Usa la función tipocam( ) para indicar si el campo a imprimir es numérico o de texto, y luego la función serie( ) para convertir el campo a texto si es numérico. Crea otro procedimiento que detiene el listado en pantalla o hace saltar el papel cuando se llena la página.

Para completar el programa, Juan añade procedimientos para calcular el total, subtotal y promedio para cada juguete; así, facilita la comparación.

Entre Juan y Eva han creado un conjunto de procedimientos que se puede adaptar fácilmente para el mantenimiento y listado de cualquier fichero.

Cree ficheros y formatos de pantalla, haga unos pocos cambios al programa todojug (sálvelo bajo un nuevo nombre), y los procedimientos funcionarán de la misma manera exacta, permitiéndole encontrar, borrar, insertar, modificar y listar registros.

Truco útil

- Cree procedimientos tan generales como sea posible, usando variables, para que se puedan adaptar rápida y fácilmente para su uso con cualquier fichero.

Listado del programa

El programa todojug debe contener los siguientes procedimientos nuevos o actualizados, que son los procedimientos creados en listjug.

```
proc a
  anterior
  si recnum()=0
    advierte;"Comienzo de fichero"
    finsi
  finproc
proc a1
  nota JUGUET2_PRG 12 de Mayo de 1985
  nota ff: (lineamax), envia FF a la impresora
  nota linea: (listreg), escribe o imprime l$ según el valor de c$
  nota linearc: (listreg), CR y LF a impresora o pantalla
  nota lineamax: (linearc), página a impresora, pausa a pantalla
  nota lista: (listselec), bucle de control, escribe el beneficio total
  nota listreg: (listventas) lista los campos uno a uno en una línea
  nota listventas: (lista), lista las ventas de un juguete, subtotal y promedio
```

```
  nota listselec: (compcom), parâmetros iniciales del listado
  nota pausa: (lineamax) tecla para seguir el listado
  nota ventlim: (ventins), limpia campos
  nota ventleer: (ventins), usa pleer para leer campos, calc benef
  nota ventins: (compcom), unejug
  nota intcodjug: (jugins) (venins), pide un código de jug.
  nota juglim: (jugins), limpia las var. de campo
  nota jugleer: (jugins) (comcheck), intr. campos menos codigo$
  nota jugins: (compcom), añade si codigo es nuevo
  nota unereg: (pidecom), sitúa código en el otro fichero
  nota uneven: (unereg),localiza j.codigo$ en ventas (si no blancos)
  nota unejug: (unereg) (ventaent), sitúa v.codigo$ en juguet
  nota pidecom: menú, pantalla, lee tecla c$
  nota compcom: llamada pidecom, comprueba c$
  nota i: llamada por compcom, (error) insertar
  nota comperr: escribe un mensaje con numerr()
  nota c: (compcom), (error) alterar
  nota blanco: (pidecom), pone en blanco la línea desde x,y
  nota cerrartodo: (pidecom), cierra todos los ficheros
  nota b: (compcom), busca texto, almacenado en txt$
  nota ayuda: (compcom), lista las opciones disponibles
  nota pregunta: Procedimiento estándar, escribe línea en 13,5
  nota vermas: (b), sigue hallando registros
  nota start: cierra los ficheros, abre y muestra "invitado"
  nota advierte: proc estándar para escribir advertencias en 14,5
  nota cambiar: (compcom), usa el otro fichero, cambia f$
  finproc
proc advierte;mens$
  escribir en 14,5; papel 6; tinta 2;" ";mens$;" "
  finproc
proc ayuda
  limpiar
  escribir "PULSE UNA DE LAS TECLAS SIGUIENTES PARA ..."
  escribir "V/L listar por pantalla o impresora"
  escribir "A  volver al registro Anterior"
  escribir "P  ir al Próximo registro"
  escribir "C  Cambiar un registro en pantalla"
  escribir "I  Insertar un nuevo registro"
  escribir "B  Buscar registros"
  escribir "S  dejarlo (Stop)"
  escribir "E  eliminar el registro actual"
  escribir "1  Ir al primer registro"
  escribir "U  Ir al último registro"
  escribir "Z  zoom / avance rápido"
  escribir "R  inverso / rebobinado"
  escribir "+  saltar un número de registros"
  escribir "%  Cambiar de fichero activo"
  escribir
  escribir "Pulse cualquier tecla para seguir";tecla()
  pantalla
  finproc
proc b
  pregunta;"¿Buscar?: "
  leer txt$
  si txt$>""
    hallar txt$
    si hallado()
      escribir
      vermas
      sino
      advierte;txt$+" no hallado"
      finsi
    finsi
  finproc
proc blanco;x,y
```

```
  escribir en x,y;rept(" ",64-y+1)
  finproc
proc c
  alterar :nota cambiar
  finproc
proc cambiar
  si f$="ventas"
    usar "j"
    haz f$="juguetes"
    sino
    usar "v"
    haz f$="ventas"
    finsi
  finproc
proc cerrartodo
  mientras 1
    cerrar
    finmientras
  finproc
proc compcom
  si c$="L": haz maxl=55:listselec: finsi
  si c$="V": haz maxl=13:listselec: finsi
  si c$="X":cambiar: finsi
  si c$="E":elimina: finsi
  si c$="+":saltar: finsi
  si c$="R": error inverso: finsi
  si c$="1": primero : finsi
  si c$="U": último : finsi
  si c$="Z": error zoom: finsi
  si c$="?":ayuda: finsi
  si c$="S": stop : finsi
  si c$="B":b: finsi
  si c$="P": próximo : finsi
  si c$="A": anterior : finsi
  si c$="C": error c:comperr: finsi :nota cambiar
  si c$="I"
    si f$="ventas":ventins: sino :jugins: finsi
    finsi
  finproc
proc comperr
  si numerr()>0 yy numerr()<>27
    advierte;"Error "+serie(numerr(),3,0)
    finsi
  finproc
proc elimina
  pregunta;"Borrar. ¿Seguro? (s/n)"
  leer txt$
  si minusc(txt$)="s"
    borrar
    finsi
  finproc
proc escmens
  pregunta;"Pulsa ESC para parar"
  finproc
proc ff
  imprimir car(12);
  finproc
proc i
  insertar
  finproc
proc intcodjug
  pregunta;"¿Código de juguete? "
  leer nuecod$
  haz nuecod$=mayús(nuecod$)
  finproc
```

```
proc inverso
  escmsg
  mientras recnum()>0
    a
    pescribir
    finmientras
  finproc
proc jugins
  intcodjug
  situar nuecod$
  si codigo$=nuecod$
    advierte¡"Código en uso"
    sino
    haz codigo$=nuecod$
    juglim
    error jugleer
    comperr
    finsi
  finproc
proc jugleer
  pleer nombre$,coste,precio
  añadir
  finproc
proc juglim
  haz nombre$=""
  haz coste=0
  haz precio=0
  pescribir
  finproc
proc linea¡l$
  si c$="L"
    imprimir l$¡
    sino
    escribir l$¡
    finsi
  finproc
proc lineamax
  haz cl=0
  si c$="L"
    ff
    sino
    pausa
    finsi
  finproc
proc linearc
  linea¡car(13)+car(10)
  haz cl=cl+1
  si cl=maxl
    lineamax
    finsi
  finproc
proc lista
  haz tot=0
  mientras no fin() yy codigo$<=acod$
    listreg
    listventas
    haz tot=tot+subtot
    próximo
    finmientras
  linea¡"* BENEFICIO TOTAL: "+serie(tot,3,0)
  linearc
  pausa
  finproc
proc listreg
  local cuenta
```

```
  haz cuenta=0
  mientras cuenta<númcam()
    si tipocam(cuenta)
      linea;valcam(cuenta)+" "
      sino
      linea;gen(valcam(cuenta),10)+" "
      finsi
    haz cuenta=cuenta+1
    finmientras
  linearc
  finproc
proc listselec
  pregunta;"¿Desde el código? "
  leer desde$
  si desde$>""
    haz desde$=mayús(desde$)
    blanco;13,0
    pregunta;"¿Hasta código?"
    leer acod$
    haz acod$=mayús(acod$)
    usar "j"
    haz f$="juguetes"
    haz cl=0
    situar desde$
    limpiar
    lista
    pantalla
    finsi
  finproc
proc listventas
  usar "v"
  situar j.codigo$
  haz subtot=0
  haz cuenta=0
  mientras no finf() yy codigo$=j.codigo$
    listreg
    haz subtot=subtot+benef
    haz cuenta=cuenta+1
    próximo
    finmientras
  linea;"* BENEFICIO de "+j.codigo$+": "+serie(subtot,3,0)
  linearc
  linea;"* PROMEDIO: "+serie(subtot/cuenta,3,0)
  linearc
  usar "j"
  finproc
proc p
  próximo
  si finf()
    advierte;"Fin de fichero"
    finsi
  finproc
proc pausa
  escribir "Pulse una tecla para seguir";tecla()
  finproc
proc pidecom
  haz otrocod$=""
  mientras 1
    si codigo$<>otrocod$
      unereg
      finsi
    pescribir
    blanco;13,5
    escribir en 12,5;"¿Comando?; "; papel 6;" "; papel 0;"(Pulse ? para obtener
ayuda)"
```

```
    haz c$=mayús(tecla())
    blanco;14,5
    escribir en 12,16;c$
    compcom
    finmientras
  finproc
proc pregunta;mens$
  escribir en 13,5; tinta 4;mens$;
  finproc
proc saltar
  pregunta;"¿Registros a saltar?"
  leer num
  posición recnum()+num
  finproc
proc start
  limpiar
  escribir en 5,10; tinta 2; papel 6;"Abriendo ficheros ..."
  error cerrartodo
  abrir "ventas" lógico "v"
  ordenar codigo$;a,mes;a
  abrir "juguet" lógico "j"
  ordenar codigo$;a
  haz f$="juguetes"
  pcargar "venjug"
  pantalla
  pidecom
  finproc
proc unejug;busca$
  usar "j"
  situar busca$
  pescribir
  usar "v"
  finproc
proc unereg
  haz otrocod$=codigo$
  si f$="ventas"
    unejug;codigo$
    sino
    uneven;codigo$
    finsi
  finproc
proc uneven;busca$
  usar "v"
  situar busca$
  si v.codigo$<>busca$
    blanco;9,0
    haz otrocod$=""
    sino
    pescribir
    finsi
  usar "j"
  finproc
proc ventins
  intcodjug
  unejug;nuecod$
  si j.codigo$=nuecod$
    haz codigo$=nuecod$
    ventlim
    error ventleer
    comperr
    sino
    advierte;"No existe el juguete"
    finsi
  finproc
proc ventleer
```

```
  pleer mes,cant,desc
  haz bene+=cant*(j.precio*(1-desc/100)-j.coste)
  añadir
  pescribir
  finproc
proc ventlim
  haz mes=0
  haz cant=0
  haz desc=0
  haz bene+=0
  pescribir
  finproc
proc vermàs
  mientras hallado()
    pregunta;"Pulse cualquier tecla para ver màs, o P para parar"
    si mayûs\tecla())="P"
      regreso
      sino
      continuar
      pescribir
      finsi
    finmientras
  advierte;"No encuentro màs "+txt$
  finproc
proc zoom
  escmens
  mientras no finf()
    pescribir
    p
    finmientras
  finproc
```

13

Salida pura y simple

Este capítulo muestra:

- Cómo desviar la salida impresa a un fichero de cinta, para uso por QUILL, ABACUS o EASEL.
- Cómo enviar la salida impresa a la pantalla.
- Cómo manipular estas características para cambiar la estructura de un fichero.
- Cómo listar un fichero con un solo comando.
- Cómo imprimir una fila de etiquetas.
- Cómo crear un menú de programas (por oposición a los menús de procedimientos).
- Cómo emular vectores en ARCHIVE.
- Cómo crear procedimientos que escriben procedimientos.
- Se presentan los siguientes comandos de ARCHIVE:

volcar	— produce un listado de un fichero, un registro por línea.
vía	— desvía la salida impresa a la pantalla o a un fichero.
normal	— cancela el anterior comando vía.
exportar	— crea ficheros de datos adecuados para su uso por otros programas.

importar	— crea ficheros de datos a partir de ficheros creados por otros programas.
salvar objeto	— salva los procedimientos de una forma compactada.
cargar objeto	— carga un programa salvado con la misma opción.

Puntos a recordar

- Si usa bucles anidados y comprueba una condición final que debe detenerlos a todos (por ejemplo, finf( )), no olvide comprobar la condición en cada uno de los bucles mientras.
- Si anida bucles que pasan sobre todos los registros de un fichero, asegúrese de que el comando próximo esté en el interior del bucle más profundo, y no se repita en los exteriores.
- Los ficheros ASCII deben terminar con el carácter ASCII número 26 (que, si lo imprime desde ARCHIVE, debe ir precedido por el carácter 0) cuando cree un fichero con vía e imprimir.
- Cuando imprima comillas literales en ficheros, la combinación ''''' vale, mientras que '''' no funciona.

Ejemplo: Juan se pone las botas

Juan está encantado con los buenos resultados globales del lagarto "Lito" y se dispone inmediatamente a preparar un informe completo, con gráficas, para presentárselo a su jefe.

Piensa escribir el informe con QUILL y dibujar las gráficas con EASEL. Mejor que escribir las cifras cada vez, usa ARCHIVE para crear ficheros que puedan usarse en QUILL y EASEL.

Si acaba de entrar en ARCHIVE, use ver para abrir juguet (lógico j) y ventas (lógico v).

usar "v" ↵ limpiar ↵

1. Haga juguet el fichero actual y borre la pantalla si es necesario.

F2

2. Pulse **F2** para borrar el AREA DE CONTROL.
   Si no lo hace, el siguiente comando escribirá sobre ella, haciendo la salida difícil de leer.

exportar "_scr"quill

3. El comando exportar seguido por quill le dice a ARCHIVE que copie los registros del fichero activo a otro fichero en un formato adecuado para su uso por QUILL.

El fichero se exportará de manera que QUILL pueda importarlo como un fichero de texto (documento).

El AREA DE VISUALIZACION muestra:

```
codigo$ nombre$              coste precio
LIT     Lagarto Lito           578   1990
LOL     Llama Lola             690   1750
PIP     Pablo el Pinguino      356   1100
```

El comando muestra en la pantalla lo que se habría copiado en el cartucho si se hubiera utilizado un nombre distinto de _scr. Así es como queda el texto cuando se importa en QUILL.

Nota: El símbolo que se ve al final de cada línea no aparece en QUILL. Es un código de fin de línea usado internamente por QUILL.

El fichero "_scr"

Este nombre de fichero es un nombre especial reservado por el QL para designar la pantalla. El subrayado delante le indica a ARCHIVE que al nombre que va a continuación no debe añadirle unidad por defecto o extensión, sino que debe abrir un fichero con ese nombre *directamente*. Permite, por tanto, acceder a todos los dispositivos de comunicación del QL. En este caso, en vez de copiar juguet a un fichero en cartucho, la salida se ha redirigido hacia la pantalla.

La pantalla no es, desde luego, un fichero en cinta, pero se usa aqui como un fichero *lógico* y no uno real.

Desde el punto de vista del QL, el proceso de transferencia es similar si la salida se envía a pantalla, a la impresora o al *microdive*. Los mismos datos se pueden dirigir a cualquier dispositivo *extraño* cambiando ligeramente las reglas.

Las reglas para comunicar dispositivos distintos se guardan en archivos llamados controladores *(drivers)*. Los datos en printer dat, por ejemplo, los usa el controlador de impresora.

Juan quiere impresionar

Exportación a QUILL

Para crear realmente un fichero de exportación hacia QUILL, escriba:

```
exportar "jugcifr"quill
```

1. Así crea un fichero llamado jugcifr_exp en la unidad 2, que contiene el fichero ventas en formato de texto (como se mostró previamente).
   Se puede importar en QUILL cargando este programa y pulsando **F3**, D, I, y escribiendo jugcifr.
   Se puede incorporar como parte de otro documento, en este caso un informe.

Exportación a EASEL y ABACUS

La manera de crear copias de ficheros de datos en un formato aceptable para su uso por EASEL y ABACUS es exactamente la misma, excepto que no se debe escribir QUILL.

```
exportar "juggraf"
```

1. Así, se crea un fichero llamado juggraf_exp. Se puede importar a EASEL o ABACUS cargando primero este programa, pulsando **F3**, D, I, y escribiendo el nombre del fichero.

```
exportar "  _scr"
```

2. Para ver en la pantalla el formato de los datos, escriba:

```
exportar "_scr"
```

Así se desvía la salida impresa del fichero de *microdrive* a la pantalla.

El AREA DE VISUALIZACION muestra:

```
"código$","nombre$","coste","precio"
"LIT","Lagarto Lito",578,1990
"LOL","Llama Lola",690,1750
"PIP","Pablo el Pingüino",356,1100
```

El formato común

La presentación en pantalla muestra la manera en que EASEL y ABACUS necesitan recibir datos.

La primera línea contiene los nombres de campo, entre comillas, seguidos por los registros cada uno en una nueva línea. Cada campo viene delimitado (separado del siguiente) por comas, y los campos textuales están entre comillas.

El símbolo al final de la lista representa aquí el marcador de fin de fichero, el carácter ASCII 26.

Es el mismo formato exacto que ARCHIVE necesita para importar ficheros. Se puede exportar también desde EASEL y ABACUS, o crearlos en SUPERBASIC o incluso desde el mismo ARCHIVE, e importarlos en ARCHIVE como ficheros de datos, siempre que sigan este formato.

Exportación de ventas a EASEL

Los ficheros de datos no siempre tienen una estructura adecuada para la realización de gráficas. El fichero ventas es un buen ejemplo.

```
usar "v"
```

1. Haga ventas el fichero activo.

```
exportar "_scr"
```

2. Muestre el formato de exportación en la pantalla:

```
"codigo$","mes","cant","desc","benef"
"LIT",1,10,0,14120
"LIT",2,180,0,254160
"LIT",3,310,0,437720
"LIT",4,320,0,451840
"LIT",5,230,0,324760
"LIT",6,260,0,367120
"LIT",7,-20,0,-28240
"LIT",8,210,20,212940
"LIT",9,180,0,254160
"LIT",10,135,0,190620
"LIT",11,120,0,169440
"LIT",12,90,0,127080
"LOL",7,525,50,97125
"LOL",8,475,50,87875
"LOL",9,330,20,234300
"LOL",10,260,20,184600
"LOL",11,210,0,222600
"LOL",12,300,50,55500
```

Cuando se importe a EASEL como un diagrama de barras, producirá un gráfico en el que las letras de código se convierten en nombres de celda de cuatro conjuntos de figuras (mes, cant, desc y benef):

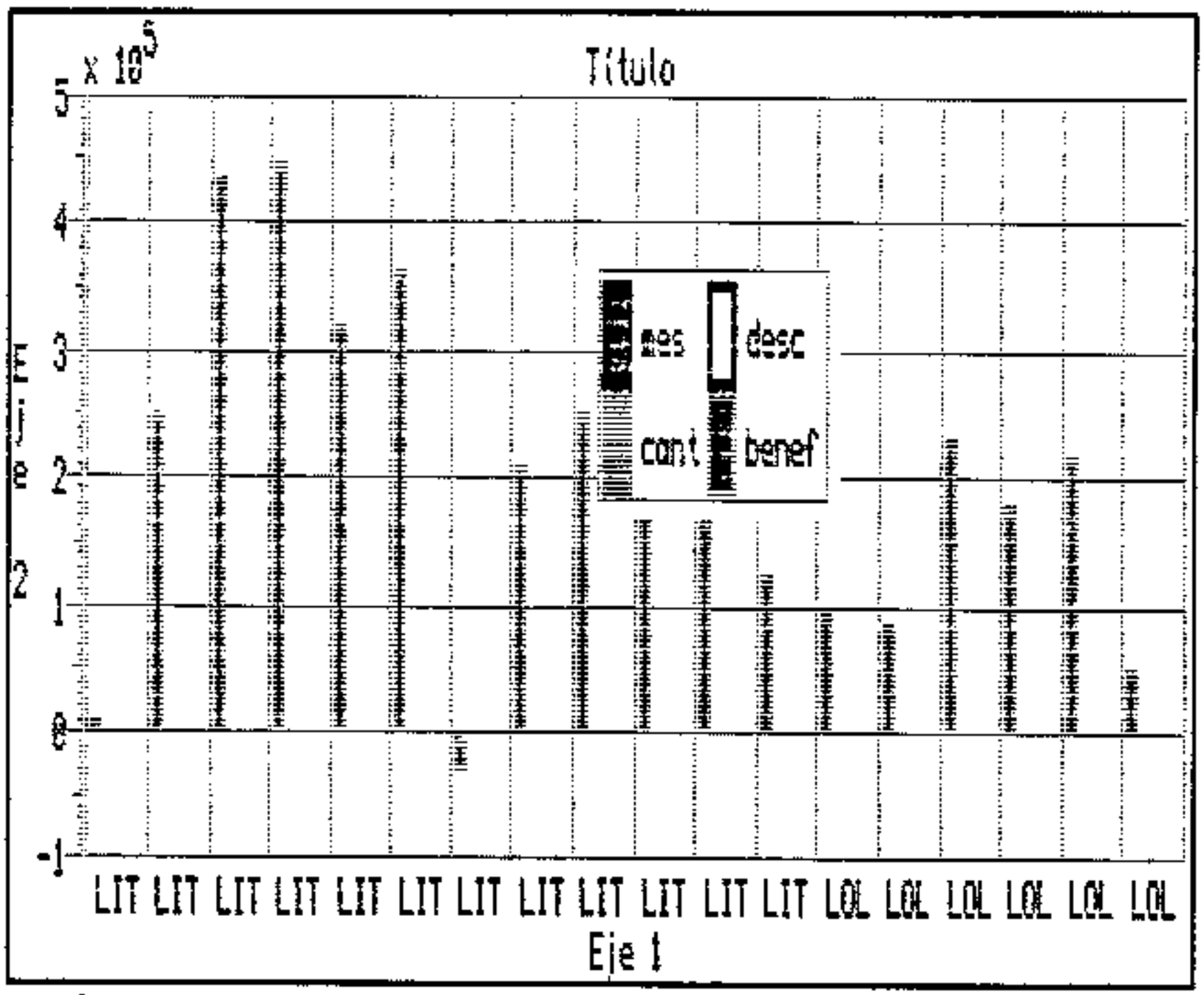


Figura 13.1. Una mala exportación a EASEL

Ficheros ASCII

Los archivos ASCII contienen sólo caracteres imprimibles ASCII (32-127) y los caracteres de retorno de carro (13-10). Los ficheros deben acabar con un marcador de fin de fichero (26). Vea los apéndices para tener más detalles sobre los códigos ASCII.

Los ficheros de programa y los ficheros de exportación son ficheros ASCII. Los ficheros de pantalla y los ficheros de datos (que contienen mucha información adicional sobre los datos para uso de ARCHIVE) no lo son.

Se pueden imprimir otros caracteres ASCII (para ficheros vía o control de su impresora), pero debe preceder cada uno con un carácter nulo; por ejemplo, imprimir car(0) + car(26). Si no lo hace, los caracteres se filtran por el controlador de impresora.

La gráfica anterior no significa mucho como está. Para comenzar, no tiene mucho sentido mostrar el número de mes en una gráfica. La figura de interés real es el beneficio, y el objetivo del ejercicio es comparar los juguetes.

```
elegir mes>6 yy código$(1) = "L"
```

3. Limite los registros a los meses para los que hay cifras de ambos juguetes.

```
ordenar mes;a,código;a
```

4. Ordene el fichero en primer lugar por fecha.

```
exportar "_scr";código$,benef
```

5. Exporte sólo dos campos a la pantalla.
   Observe el punto y coma antes de los campos, y la coma que los separa.

Ahora aparece:

```
"codigo$","benef"
"LIT",-28240
"LOL",97125
"LIT",212940
"LOL",87875
"LIT",254160
"LOL",234300
"LIT",190620
"LOL",184600
"LIT",169440
"LOL",222600
"LIT",127080
"LOL",55500
```

EASEL presentaría, a partir de este fichero, una gráfica mucho más significativa, como se muestra en la figura 13.2. Todavía no es perfecta, ya que las barras de los dos juguetes son del mismo color, y los meses no se escriben, pero Juan puede arreglar esos detalles a mano cuando imprima el gráfico.

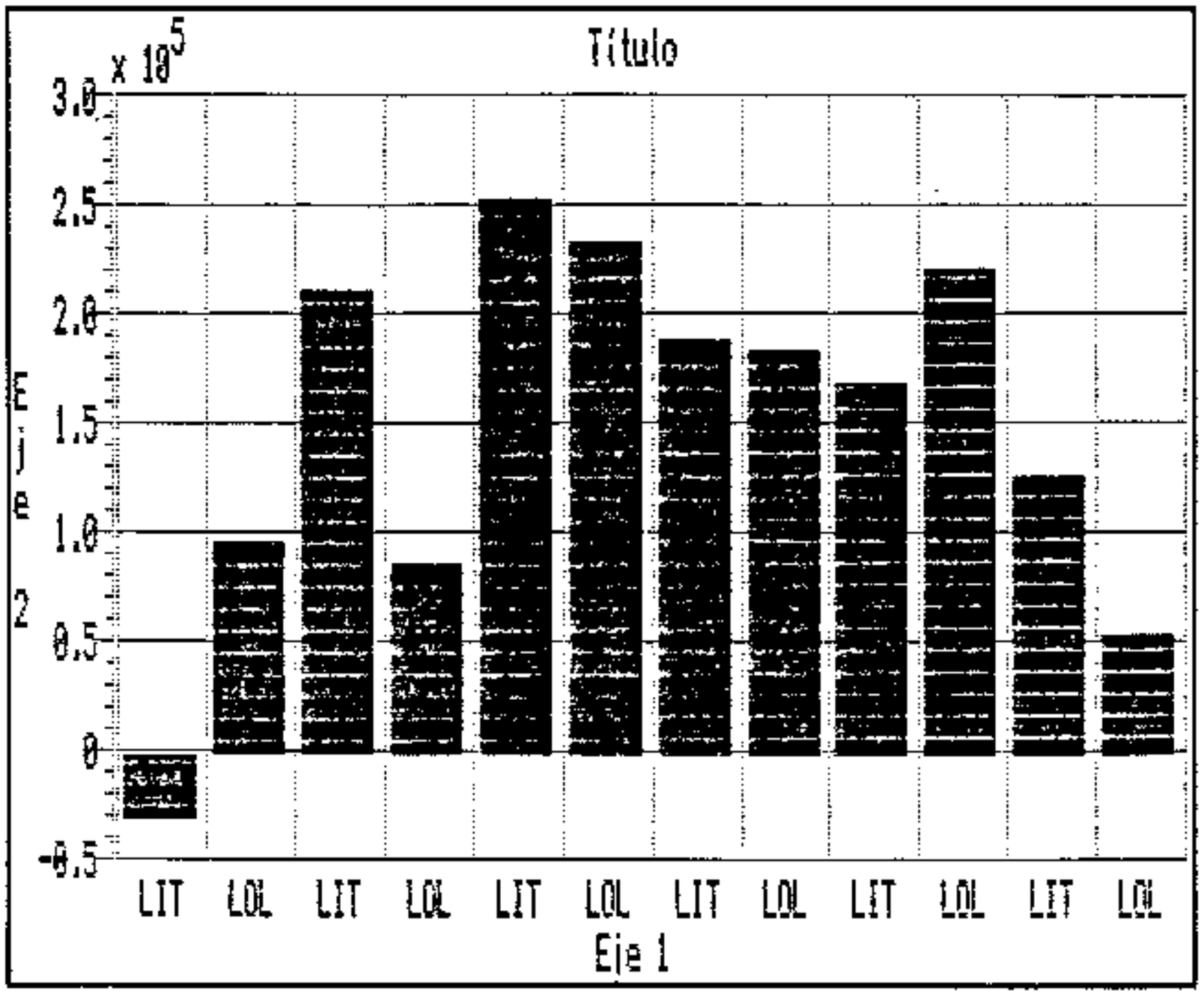


Figura 13.2. Una exportación mejor a EASEL

F3 export "vengraf";código$,benef
F3 export "vencifr";código$,benef quill

6. Repita y edite la línea dos veces, para crear ficheros reales en cinta, con un formato adecuado para EASEL y QUILL.
   Observe el espacio delante de quill en el segundo comando.

restaurar ↵ ordenar código$;a,mes;a ↵

7. Devuelva el fichero a su estado anterior.

Cuando exporte de ARCHIVE a EASEL, recuerde:

- Los campos (tras el primero) se convierten en conjuntos de cifras en las gráficas, si son numéricos.

- El contenido del primer campo alfanumérico se convierte en las etiquetas de celda en el eje horizontal.

CAMBIO DE ESTRUCTURA DE FICHEROS

Puede utilizar los comandos exportar e importar para eliminar campos de ficheros de datos.

importar "vengraf_exp" como "benef"lógico"b"

1. Así crea una copia del fichero exportado vengraf_exp como un fichero de datos, y lo abre en memoria.
2. El fichero tiene sólo dos campos, código$ y benef.
3. Cierre el fichero benef_ dbf.

indicar
cerrar

Más detalles sobre el formato común

Las reglas del formato común (que excluye a QUILL) son las siguientes:

- Cada línea del fichero acaba con un retorno de carro (carácter ASCII 13 y 10). Las líneas no terminan con coma.
- El fin de fichero se marca con el carácter ASCII 26.
- Todos los demás caracteres son caracteres imprimibles ASCII (letras, números y signos de puntuación).
- La primera línea debe contener nombres entre comillas, separados por comas uno de otro. Cada nombre determina el tipo de los datos (texto o numérico) que aparece en la misma columna en las restantes líneas.
- Las demás líneas contienen tantos elementos (campos) como nombres existan, cada uno separado por comas y con las cadenas de caracteres entre comillas.
- Las cadenas vacías se representan por " ", y los campos numéricos por cero.

Impresión en cualquier sitio con VIA

Se pueden usar estas reglas para crear un fichero con más campos que el original.

vía "porcent_imp"

1. vía desvía la salida que normalmente va a la impresora hacia un fichero.

escribir "'código$','benef','porcent'"

2. Así determina los campos que tendrá el fichero. (La comilla simple se consigue en el teclado español mediante **CTRL** y ' —acento—.)
Observe que toda la línea aparece entre comillas dobles, y cada nombre de campo (con o sin dólar) aparece entre comillas simples, separados por comas.

> *Nota:* Se puede usar comillas simples como literales si van inmersas en comillas dobles.

F5 imprimir "'código$','benef','porcent'"

3. Si la línea aparece correcta al imprimirla en la pantalla, edítela para imprimirla.
Así se envía una línea al fichero de impresión.

> *Nota:* Si realmente imprime un error usando este método, hay que escribir normal y comenzar de nuevo.

usar "v"

4. Use el fichero ventas e introduzca la línea siguiente:

primero: mientras no finf(): escribir "'" + codigo$ + "','"; benef;"','";0: próximo: finmientras.

VIA, NORMAL

Como el comando exportar, via está pensado sobre todo para crear ficheros en *microdrive*.

Existe en la jerga una palabra, spool, que se aplica a este tipo de situaciones. La palabra tiene su origen en los tiempos en que los ordenadores eran grandes y caros, y mucha gente compartía la misma impresora.

En ese tipo de situaciones no se puede imprimir cuando uno quiere. Los trabajos de impresión se "imprimen" en cinta o disco (en el *spool*) y se unen a una cola de ficheros similares que esperan turno. Incluso, en muchos casos, los ficheros (en el *spool*) se podían imprimir via consolas o elegir entre varias impresoras.

Via le dice a ARCHIVE que los comandos como imprimir, listar y volcar (véase la próxima sección) van a enviar su salida por un camino distinto de la impresora. Si el nombre de fichero es pantalla, la salida va a la pantalla.

La redirección de la salida impresa sigue hasta que se usa el comando normal, que cierra el fichero.

El fichero es un fichero ASCII como los que crea exportar.

Haga un fichero adecuado para su importación por QUILL, añadiendo quill al final del comando. Por ejemplo, via "document"quill.

via crea ficheros con la extensión _lis si no se le indica otra cosa.

5. Sea cuidadoso al escribir esta línea. Si se sitúa una coma incorrectamente, o se mezclan las comillas simples con las dobles, o las comas con punto y comas, no funcionará.
   Observe que el campo literal se rodea de comillas mediante "'", mientras que benef viene seguido sólo de una coma.
   Se imprime un cero literal como contenido del nuevo campo porcent.
   Recuerde que escribir envía su carácter de retorno de carro si no se evita.

La pantalla queda ahora:

```
'codigo$','benef','porcent'
'LIT',14120,0
'LIT',254160,0
'LIT',437720,0
'LIT',451840,0
'LIT',324760,0
'LIT',367120,0
'LIT',-28240,0
'LIT',212940,0
'LIT',254160,0
'LIT',190620,0
'LIT',169440,0
'LIT',127080,0
'LOL',97125,0
'LOL',87875,0
'LOL',234300,0
'LOL',184600,0
'LOL',222600,0
'LOL',55500,0
```

F5 primero: mientras no finf(): imprimir "'" + codigo$ + "',"; benef;",";0: próximo: finmientras

6. Si la salida parece correcta, reedite la línea y cambie escribir por imprimir.
   Esta vez los datos van al archivo.

imprimir car(0) + car(26)

7. Este comando pone un marcador de fin de fichero (ASCII 26) en el cartucho.
Viene precedido por un carácter nulo (ASCII 0), ya que normalmente el controlador de impresora filtra todos los caracteres no imprimibles de los comandos de impresión. Se puede imprimir un carácter especial o código de control precediéndolo por un car(0). Así le indica al controlador de impresora que deje pasar el próximo carácter "tal cual".

normal **8.** Así acaba el último via, cierra el fichero y lo salva a la cinta.

importar "porcent_imp" como "porcent"lógico"p"

9. Así hace una copia de porcent_imp como un fichero de datos abierto, porcent_dbf.

indicar **10.** Muestre el fichero. Hay ahora tres campos: codigo$, benef y porcent.

cerrar **11.** Cierre el fichero.

Tiene ahora una buena colección de versiones de los ficheros de juguetes y ventas en el *microdrive* 2.

ventas_dbf	vengraf_exp
juguet_dbf	benef_dbf
juggraf_exp	porcent_imp
jugcifr_exp	porcent_dbf
vencifr_exp	

Recuerde borrar los ficheros innecesarios mientras todavía puede recordar si sirven o no para algo.

Puede usar este método combinado con técnicas normales de listado para crear combinaciones de campos o ficheros para enviar a EASEL o ABACUS imposibles de conseguir con exportar.

QUILL necesita símbolos especiales de fin de línea, y, naturalmente, no comillas o comas, a menos que tengan que aparecer en algún texto.

Si tiene ARCHIVE versión 2, puede utilizar también este método con bucles y las funciones nomcam(), tipocam() y valcam() para escribir procedimientos que añadan nuevas cadenas de caracteres o campos numéricos a cualquier estructura de fichero.

Errores de importación

Hay algunos errores muy corrientes (y desorientadores al principio) que se pueden encontrar cuando se importan ficheros

en ARCHIVE. La mayor parte se refieren al proceso de copia del fichero, y no a la sintaxis de la sentencia propiamente dicha.

- No se puede importar ficheros de QUILL.

Para más información sobre importar/exportar y la integración de los programas, véase *QL: la integración del software*, o el apéndice en el manual del QL.

He aquí algunos errores, y las causas más comunes:

Error - Esperada serie de texto

Uno de los nombres de campo de la primera línea no está entre comillas.

Error - Tipo de datos incorrecto

Se ha encontrado un elemento numérico de datos (sin comillas) en una columna cuyo nombre acaba en $ (o viceversa). Recuerde que los campos alfanuméricos vacíos son "⇩", y los numéricos son 0.

Error - Incapaz de abrir fichero

Recuerde nombrar el fichero de importación completamente, incluyendo la extensión.

Error - Nombre duplicado

El último comando tuvo un éxito parcial, y dejó el fichero que quería importar abierto en memoria. Ciérrelo.

Ejemplo: Etiquetas y otra manera de imprimir ficheros

Salva ha pedido prestadas unas cuantas hojas de papel de etiquetas de ordenador de la caja que compró Maya para enviar catálogos a sus clientes. Se trata de papel continuo, con filas de agujeros, papel "pijama" con dos columnas de etiquetas autoadhesivas a lo largo de la página.

Salva se dispone a preparar un programa que usará para enviar cartas de agradecimiento por los regalos de boda y (espera que en seguida) tarjetas de cambio de domicilio.

Comienza por abrir y examinar el fichero invitado, comprobando las líneas que son demasiado anchas para las etiquetas.

INSTRUCCIONES

Si acaba de comenzar, cargue ARCHIVE. En caso contrario, salve los procedimientos en memoria y use nuevo para limpiar la memoria.

El comando VOLCAR

El comando volcar es otro comando de impresora, que lista los registros de un fichero línea tras línea.

usar "i" via pantalla

1. Haga invitado el fichero activo.
2. Si no tiene una impresora, envíe la salida a la pantalla. Esto es muy útil para comprobar la salida de impresora.

volcar

3. Tras una corta pausa mientras ARCHIVE prepara el fichero para imprimir, se listan todos los registros del fichero:

```
nombre$                                    direc1$                                    direc2
$              direcó$              direc4$   vienen$ cuantos
Juan y Pepa Martinez Recoilo               Majada del Viento, 25                      29029
Madrid         MADRID                           s          2
Luisa Sala y Carmen Garcia Carrión         Gran Via, 34                               Orihue
la             ALICANTE                         s          2
Eduardo Rincón Moreno                      Avda. de los Poetas, 12                    03012
Barcelona      BARCELONA                                   8
Teatro Principal                           Pza. Mayor, 3                              Hellin
               ALBACETE                                    3
Lorenzo Sánchez López, critico teatral     Revista del Escenario                      Avda.
Principal, 23  Villanueva de Segre TARRAGONA               1
Juan Echarri                               Cia de Danza Moderna "Break out"           Melanc
ólicos, 85     28013 Madrid        MADRID                  1
Milena de Pablo Martinez                   Practicante Diaz de Carreras, s.n.         Teranc
ón             CUENCA                                      1
Maribel Gutiérrez                          Paseo de las Acacias, 34                   Villan
ueva de Segre  TARRAGONA                                   1
Felipe y Luisa González                    Jardines Dorados, 12                       Camino
 de las Flores Bangkok             TAILANDIA               2
Antonio Pérez de las Heras y Sra.          Teniente Coronel                           Prat d
'en Joaquim    08102 Roses         GIRONA                  2
Arturo y Teresa López Andión               Magnolias, 110                             05002
               BILBAO                                      7
Srta. Felisa de los Monteros               Vista Cielo                                Urb. L
os Montes      Villanueva de Segre TARRAGONA               1
```

Observe que se imprime el contenido completo de todos los campos en columnas en la pantalla, con el nombre de campo en la parte superior de cada columna.

Eva y Salva cuentan sus "bendiciones"

El ancho de cada columna se asigna a la longitud del ejemplo más largo de ese campo en el fichero. Si el ancho resultante es menor que el nombre del campo, el nombre se abrevia.

El resultado es hacer el ancho de cada registro igual a la suma de los ejemplos máximos de cada campo en el fichero. A veces es demasiado ancho para la pantalla, y los registros continúan en la línea siguiente.

La palabra "volcar" es un viejo ejemplar de la jerga informática. Describe la acción de coger el contenido íntegro de un fichero o parte de la memoria y listarlo "tal cual" en una impresora o terminal.

Salva mejora el volcado

El resultado no es exactamente lo que Salva quería, ya que es casi imposible usarlo para nada. Mejora algo la cosa limitando los campos que va a imprimir volcar.

```
volcar;nombre$,direc1$
```

1. Observe que el comando va seguido por un punto y coma, y los nombres de los campos separados por comas. Si no escribe el punto y coma tras el comando, volcar imprimirá todos los campos.

Esta vez la impresora imprime sólo los campos pedidos. Están aún en columnas tan anchas como el campo más largo.

```
nombre$                                  direc1$
Juan y Pepa Martinez Rebollo             Maiada del Viento, 25
Luisa Saia y Carmen Garcia Carriòn       Gran Via, 34
Eduardo Rincòn Moreno                    Avda. de los Poetas. 12
Teatro Principal                         Pza. Mayor, 3
Lorenzo Sànchez Lòpez, critico teatral   Revista del Escenario
Juan Echarri                             Cia de Danza Moderna "Break out"
Milena de Pablo Martinez                 Practicante Diaz de Carreras, s.n.
Maribel Gutièrrez                        Paseo de las Acacias, 34
Felipe y Luisa Gonzàlez                  Jardines Dorados, 12
Antonio Pèrez de las Heras y Sra.        Teniente Coronel
Arturo y Teresa Lòpez Andiòn             Magnolias, 110
Srta. Felisa de los Monteros             Vista Cielo
```

El comando volcar es muy cómodo para ficheros cuyos datos se manejen fácilmente bajo forma tabular, como agenda, pero ofrece muy poco control sobre la presentación de los datos.

```
todos: si lon(nombre$)>35: escribir nombre$: finsi: fintodos
```

2. Salva observa algunos nombres largos y hace que ARCHIVE los imprima. No se va a molestar ahora en cambiarlos, sino que va a probar su programa de etiquetas.

normal

3. Cancele el comando via pantalla.

> *Nota:* volcar imprime todos los registros de un fichero a menos que se haya hecho una selección previa.

Papel de etiquetas de ordenador

El papel autoadhesivo de ordenador viene normalmente con dos o más etiquetas a lo largo de la página. Es prácticamente imposible imprimir etiquetas mediante una impresora de hojas sueltas, ya que las etiquetas resbalan en el mecanismo de fricción. Use el alimentador de tracción con papel perforado para que las etiquetas y la impresora marchen en buena armonia.

Las etiquetas en varias columnas plantean un problema al imprimir direcciones. Es fácil imprimir un nombre y una dirección a la vez, uno tras otro. No es tan fácil imprimir los nombres de varios registros, seguidos por la primera línea de direcciones de los mismos registros, y asi sucesivamente.

Una solución es almacenar los nombres y direcciones de cada registro en variables, e imprimir las variables usando tab para situar la impresión en cada columna de etiquetas.

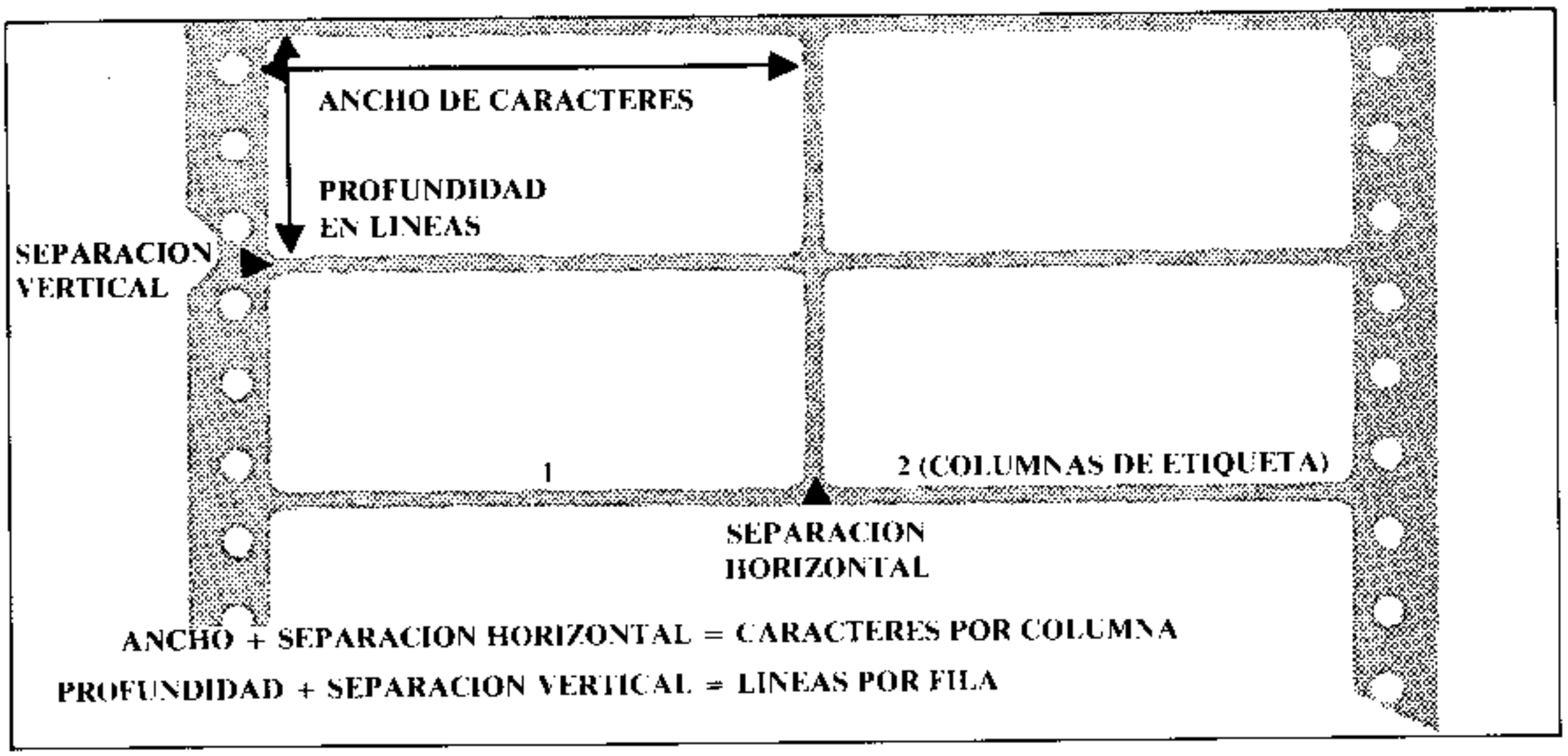


Figura 13.3. Dimensiones significativas del papel de etiquetas

Un programa según esta idea que imprima tres etiquetas a lo largo de la página necesitará los siguientes pasos:

```
mientras no finf()
    haz a1$ = direc1$
    haz a2$ = direc2$
    haz ...
    (etc.)

próximo
    haz b1$ = direc1$
    haz b2$ = direc2$
    haz ...
    (etc.)

próximo
    haz c1$ = direc1$
    haz c2$ = direc2$
    haz ...
    (etc.)

próximo
    imprimir a1$; tab 38; b1$; tab 76; c1$
    imprimir a2$; tab 38; b2$; tab 76; c2$
    imprimir ...
    (etc.)

finmientras
```

Salva decide no adoptar este método. Aparte de ser tedioso de escribir, resulta muy difícil de adaptar. Sabe que puede utilizar más adelante el programa con otros ficheros, o con etiquetas de forma y número de columnas distinto.

Uso de bucles para procesar etiquetas

Salva puede escribir un programa adaptable usando bucles para almacenar e imprimir las etiquetas. Esto quiere decir que cada línea se identificará posteriormente por el mismo nombre de variable.

Salva decide utilizar un campo llamado linea$ en un fichero de datos para almacenar las líneas de etiqueta. Cada registro almacena una línea mientras se prepara para su impresión. Por ejemplo, el primer registro contendrá el campo nombre$ de dos registros de direcciones.

Antes de escribir los procedimientos, Salva hace un esquema del programa en papel en términos generales:

fichero lineas - registro 1 - nombre$ x2
registro 2 - direc1$ x2
registro 3 - direc2$ x2
registro 4 - direc3$ x2
registro 5 - direc4$ x2

comienzo: abrir ficheros
- ficheros de datos
- fichero que almacenará las líneas
definir las variables de formato de etiquetas
- ancho de una etiqueta (31)
- número de líneas por etiqueta (9)
- número de etiquetas por fila (2)
avanzar por los registros de datos, almacenar e imprimir etiquetas
almacenar dos registros de direcciones como líneas de impresión
- almacenar campos de dirección, una línea cada vez para cada etiqueta
- completar los campos hasta la longitud total con blancos
imprimir 5 líneas de etiquetas
imprimir líneas en blanco hasta el comienzo de la siguiente fila de etiquetas

El fichero de líneas de etiquetas

Salva crea primero un fichero de datos que almacenará las líneas de impresión:

```
crear "invitado_eti"lógico"eti" ↵  linea$ ↵  fincrear
```

1. El fichero tiene el mismo nombre que los datos, pero una extensión distinta para diferenciarlos en la cinta. Así puede usar el programa con distintos ficheros, cada uno con su fichero _eti fácilmente identificable.

```
haz x=0: mientras x<5: añadir: haz x=x+1: finmientras
```

2. Esta línea le añade cinco registros completamente vacíos al fichero. El fichero debe contener sólo tantos registros linea$ como líneas se van a imprimir del fichero de direcciones (nombre$, direc1$, direc2$, direc3$ y direc4$).

```
cerrar
```

3. Cierre el fichero para asegurarse de que se ha salvado correctamente al cartucho. Lo abrirá cada vez que necesite etiquetas.

Estadística vital

El primer procedimiento de Salva abre los ficheros y prepara las variables relacionadas con el formato de las etiquetas:

```
editar
```

1. Use el editor para crear un procedimiento.

```
proc start ↵  haz fichdat$="invitado"
```

2. Recuerde que el uso de variables le permite adaptar el programa fácilmente a otros ficheros y formatos.

```
haz fd$="i"
```

3. El nombre lógico del fichero de datos se usará para distinguirlo del fichero de líneas de etiqueta.

```
nota abrir fichdat$ lógico fd$
```

4. Inserte una línea que le recuerde lo que significan las dos variables y que debe abrir el fichero de datos.

```
abrir fichdat$+"_ eti" lógico "eti"
```

5. Abra el fichero de líneas de etiqueta relacionado con el fichero de datos en uso.
   Recuerde que el fichero de líneas debe tener exactamente tantos registros como campos del fichero de datos se van a imprimir.
   Recuerde que si usa una extensión no estándar al crear o salvar un fichero debe usarla siempre que lo vuelva a abrir.

```
haz colum=2
```

6. La variable colum determina el número de columnas de etiquetas presentes en el papel.

```
haz ancho=31
```

7. La variable ancho almacena el número de caracteres desde el comienzo de una etiqueta hasta el comienzo de la siguiente.

El fichero de líneas de etiquetas

Salva crea primero un fichero de datos que almacenará las líneas de impresión:

```
crear "invitado_eti"lógico"eti" ↵  linea$ ↵  fincrear
```

1. El fichero tiene el mismo nombre que los datos, pero una extensión distinta para diferenciarlos en la cinta. Así puede usar el programa con distintos ficheros, cada uno con su fichero _eti fácilmente identificable.

```
haz x=0: mientras x<5: añadir: haz x=x+1: finmientras
```

2. Esta línea le añade cinco registros completamente vacíos al fichero. El fichero debe contener sólo tantos registros linea$ como líneas se van a imprimir del fichero de direcciones (nombre$, direc1$, direc2$, direc3$ y direc4$).

```
cerrar
```

3. Cierre el fichero para asegurarse de que se ha salvado correctamente al cartucho. Lo abrirá cada vez que necesite etiquetas.

Estadística vital

El primer procedimiento de Salva abre los ficheros y prepara las variables relacionadas con el formato de las etiquetas:

```
editar
```

1. Use el editor para crear un procedimiento.

```
proc start ↵  haz fichdat$="invitado"
```

2. Recuerde que el uso de variables le permite adaptar el programa fácilmente a otros ficheros y formatos.

```
haz fd$="i"
```

3. El nombre lógico del fichero de datos se usará para distinguirlo del fichero de líneas de etiqueta.

```
nota abrir fichdat$ lógico fd$
```

4. Inserte una línea que le recuerde lo que significan las dos variables y que debe abrir el fichero de datos.

```
abrir fichdat$+"_ eti" lógico "eti"
```

5. Abra el fichero de líneas de etiqueta relacionado con el fichero de datos en uso.
   Recuerde que el fichero de líneas debe tener exactamente tantos registros como campos del fichero de datos se van a imprimir.
   Recuerde que si usa una extensión no estándar al crear o salvar un fichero debe usarla siempre que lo vuelva a abrir.

```
haz colum=2
```

6. La variable colum determina el número de columnas de etiquetas presentes en el papel.

```
haz ancho=31
```

7. La variable ancho almacena el número de caracteres desde el comienzo de una etiqueta hasta el comienzo de la siguiente.

Observe que no tiene en cuenta el hecho de que hay normalmente una separación de dos o tres caracteres entre las etiquetas. Desde luego, imprimir en el hueco no tiene ningún sentido.

Salva introduce un valor temporal de 31, de manera que las dos etiquetas no pasen del tamaño de la pantalla mientras hace sus pruebas.

haz lineas=9

8. La variable lineas almacena el número de líneas entre la parte superior de una etiqueta y la parte superior de la siguiente fila de etiquetas (es decir, la altura de una etiqueta).

Incluye la separación de una línea entre etiquetas.

etiquetas finproc

9. Etiquetas es el nombre del procedimiento que comienza y gobierna el resto de los procedimientos de etiquetas.

El procedimiento ahora queda como sigue:

```
proc start
    haz fichdat$ = "invitado"
    haz fd$ = "i"
    nota abrir fichdat$ lógico fd$
    abrir fichdat$ + "_eti" lógico "eti"
    haz colum = 2
    haz ancho = 31
    haz lineas = 9
    etiquetas
    finproc
```

Almacenamiento de una columna

Salva comienza directamente por la rutina que almacena e imprime una fila, de manera que pueda ver algún resultado sin tener que acabar toda la tarea.

Antes de imprimir las líneas de etiquetas que se almacenan en invitado_eti, debe crearlas a partir de los campos de dos direcciones distintas.

editar ↵ F3 n etilin ↵

1. Use el editor y comience el procedimiento etilin.

primero "eti" ↵ mientras no finf()

2. El comando primero "eti" hace simultáneamente eti el fichero activo y hace la primera linea$ el registro activo. No hay necesidad de usar un contador, ya que el fichero eti sólo contiene tantos registros como líneas se van a imprimir. Cada línea se almacena en secuencia según su número de registro.

```
haz cam$ = valcam(recnum("eti"),fd$)
```

3. La función recnum() devuelve el número del registro linea$ activo, y asigna a cam$ el valor del campo del mismo número de fd$.
   Observe que se puede usar un nombre lógico como segundo argumento de valcam().
   Por ejemplo, con la primera linea$, el registro 0, cam$ se asigna al valor valcam(0), que es el campo nombre$ en el fichero de invitados.

> *Nota:* Las primeras versiones de ARCHIVE pueden producir un error si se usan nombres lógicos con valcam(). Haga activo el fichero mediante usar.

```
haz cam$ = cam$ + rept(" ",ancho-lon(cam$))
```

4. Esta línea añade tantos espacios como sea necesario para hacer la línea tan larga como el ancho de una etiqueta.

```
    etisal;cam$
```

5. Se llama a un procedimiento que almacena (salva) el campo y los espacios (pasados como un parámetro) en linea$.

```
próximo ↵    finmientras ↵
```

6. Así nos movemos al siguiente registro y el proceso comienza de nuevo con los contenidos del próximo registro de datos.
   Recuerde que primero "eti" hace a eti el fichero activo.

El procedimiento etilin debe quedar como sigue:

```
proc etilin
    primero "eti"
    mientras no finf()
    haz cam$ = valcam(recnum("eti"),fd$)
    haz cam$ = cam$ + rept(" ",ancho-long(cam$))
    etisal;cam$
    próximo
    finmientras
finproc
```

Almacenamiento de una línea

```
    proc etisal;txt$
```

1. Cree un procedimiento etisal que actualice linea$ con la cadena txt$ que se le pasa.

```
si n = 0 ↵    haz linea$ = "⇩" ↵    finsi ↵
```

2. Así limpia linea$ siempre que se vaya a almacenar la primera etiqueta de una fila.

Se hace mediante un bucle si aquí. Si se hubiese hecho antes del bucle en etilin habríamos tenido que usar varios comandos actualiz para salvar los registros vacíos.

haz linea$ = linea$ + txt$

3. Esta línea añade el parámetro, en este caso parte de la dirección, a linea$.

actualiz

4. Se actualiza el registro linea$.

El procedimiento debe quedar así:

```
proc etisal;txt$
    si n=0
        haz linea$=""
        finsi
    haz linea$=linea$+txt$
    actualiz
finproc
```

Impresión de una fila

Tras almacenar diez campos de dos registros de direcciones en los cinco registros de línea, el programa usa un bucle para imprimir las líneas:

etimprim ↵ primero "eti" ↵ mientras no finf() ↵ linimpr;linea$ ↵
próximo ↵ finmientras ↵

1. Escriba el procedimiento etimprim.

linimpr; txt$ ↵ imprimir txt$ ↵

2. Escriba el procedimiento linimpr para imprimir realmente el texto.

Finalmente, tenga en cuenta la diferencia entre las cinco líneas impresas y la posición inicial de la siguiente fila de etiquetas.

etinuefil ↵ haz cuenta=0 ↵ mientras cuenta<lineas-recnum() ↵

3. Comience un nuevo procedimiento llamado etinuefil. Use un bucle con contador para imprimir tantos retornos de carro como la profundidad total de la etiqueta menos el número de líneas ya impresas.
El número es igual al número de registro del fichero activo eti, que representa la última línea impresa.

> *Nota:* Recuerde que los números de campo (y de registro) comienzan a contar por cero.

linimpr;"" ↵ haz cuenta = cuenta + 1 ↵ finmientras ↵

4. Acabe el procedimiento usando linimpr sin texto para imprimir un retorno de carro, incremente el contador y acabe el bucle.

El procedimiento debe quedar como sigue:

```
proc etinuefil
    haz cuenta = 0
    mientras cuenta < lineas - recnum()
        linimpr;"⇩"
        haz cuenta = cuenta + 1
        finmientras
    finproc
```

ESC ESC salvar "etiquet1" ↵ salvar "mdv1_etiquet1" ↵

5. Salve los programas de etiqueta, en primera generación, a las dos cintas.
Asegúrese de tener un cartucho de seguridad en la unidad 1, y salve los procedimientos por si algún accidente los destruye antes de tener siquiera la oportunidad de usarlos.
Use dos líneas de comandos, mejor que dos puntos para separarlos. Así, si algún comando falla, sabremos cuál de los dos.

start

6. Defina las variables de formato y abra el fichero de etiquetas introduciendo start.
Ignore el mensaje de error: se debe a que aún no hemos escrito etiquetas.

etilin: etimpr

7. Escriba estos dos comandos para probarlos.
Tras una corta pausa mientras el programa almacena dos direcciones, aparecen dos etiquetas, una al lado de otra, por la pantalla. Como los campos linea$ se actualizan, el cartucho de datos puede girar intermitentemente al principio. Una vez las líneas se mantengan de una longitud constante, esto debe parar.
La etiqueta aparece como sigue:

```
Juan y Pepa Martinez Rebollo        Luisa Sala y Carmen Garcia Carrión
Mayaga del viento, 25               Gran Via, 34
28029 Madrid                        Orihuela
MADRID                              ALICANTE
```

Procedimiento de control

Salva escribe ahora el procedimiento principal, que controlará el trabajo pasando por todos los registros y desde el que son llamados los demás. El procedimiento se llama etiquetas, y se llama inicialmente desde start:

```
etiquetas ↵ limpiar ↵ mientras no finf(fd$) ↵
```

1. Use el nombre lógico del fichero de datos (almacenado en fd$) por claridad y para asegurar que el test de fin de fichero se haga sobre el fichero correcto.

```
haz n=0 ↵ mientras no finf(fd$) yy n<colum
```

2. La variable n es el contador del número de etiquetas en una fila. El bucle falla cada dos pasadas (cuando n es igual a colum).
Observe que se usa también finf() en este bucle. Si no se hiciera, el programa seguiría el bucle hasta que se acabe la fila, incluso si el fichero se ha acabado.
Así que, si el último registro cae en la primera etiqueta de una fila, se repetiría hasta que acabe la fila.

```
etilin
```

3. Así se llama el procedimiento descrito anteriormente, que almacena una dirección como parte de las líneas de etiquetas.

```
próximo fd$
```

4. La acción que ocurre con más frecuencia (aparte de almacenar parte de una línea) es avanzar por los registros de direcciones; próximo, por tanto, se sitúa en el bucle más interno.
Errores muy corrientes en los procedimientos que usan finf() incluyen el olvido de próximo o su repetición en todos los bucles.
El nombre lógico se usa por claridad y por hacer invitado el fichero activo.

```
haz n=n+1
```

5. Incremente el contador de bucle.
Aunque hay un finf() en la condición del mientras, n (que está también en la expresión del mientras) hace que el bucle falle cada dos registros, mientras que próximo fd$ sólo falla una vez cada fichero.

```
etimpr
```

6. Llame al procedimiento que imprime las líneas almacenadas en las etiquetas de una fila.

```
etinuefil ↵ finmientras
```

7. Llame al procedimiento que imprime retornos de carro desde la última línea hasta la parte superior de la siguiente etiqueta.

```
cerrar "eti"
```

8. Cierre el fichero de etiquetas tan pronto como acabe el programa.

El procedimiento queda ahora como sigue:

```
proc etiqueta
    limpiar
    mientras no finf(fd$)
        mientras no finf(fd$) yy n<colum
            etilin
            próximo fd$
            haz n=n+1
            finmientras
        etimpr
        etinuelin
        finmientras
    cerrar "eti"
finproc
```

ESC ESC salvar "etiquet2" ↵ salvar "mdv1_etiquet2"

9. Salve la versión del programa de segunda generación.

Observe que el programa no usa en ningún momento un carácter de salto de página (ASCII 12) para pasar de página. El papel de etiquetas suele tener exactamente la misma separación entre dos páginas que entre dos etiquetas normales. La longitud de la página es, por tanto, irrelevante. Una vez alineadas correctamente, la paginación se realiza automáticamente.

Impresión de prueba

Para ayudar a alinear correctamente las etiquetas antes de seguir con la impresión, ofrezca la opción de imprimir una fila de etiquetas de prueba. Imprimiendo asteriscos en las posiciones donde irán las líneas de la dirección, podrá ajustar el papel si hay alguna fuera de sitio. Ajuste moviendo el carro o las posiciones de los orificios del papel y haga otra prueba.

editar **1.** Use editar para crear el procedimiento etialin.

```
proc etialin
  haz ok=0
  mientras no ok
    escribir en 13,10;"¿Impresión de prueba? (s/n)";
    haz prueba$=mayús(tecla())
    escribir prueba$
    si prueba$="N"
      haz ok=1
      sino
      si prueba$="S"
```

```
        haz n=0
        mientras n<colum
          etilin
          haz n=n+1
          finmientras
        etimpr
        etinuelin
        finsi
      finsi
    finmientras
finproc
```

F4 si prueba$ = "N" ↵ ↵ ⇩ sino ↵ haz cam$ = "*" ↵ finsi

2. Use **TAB** hasta el procedimiento etilin e inserte estas líneas de manera que se almacene un asterisco en vez de un campo si testimpr es S.

La sentencia si que almacena cam$ queda como sigue:

```
si prueba$ = "N"
    haz cam$ = valcam(recnum("eti"),fd$
    sino
    haz cam$ = "*"
    finsi
```

etialin

3. Use **TAB** hasta el procedimiento start y llame a etialin justo antes de etiqueta.

progmenú

4. Añada la línea progmenú justo después de etiqueta. Este procedimiento se explica en la siguiente sección.

ESC ESC primero "i" ↵ start

5. Muévase atrás hasta el primer registro de invitados y pruebe la opción de impresión de prueba. Ignore el mensaje de error sobre progmenú cuando acaba el programa. El procedimiento no lo hemos escrito todavía.

La rutina de impresión imprime en la pantalla algo parecido a la línea siguiente:

```
¿Impresión de prueba? (s/n) S

*                                *
*                                *
*                                *
*                                *
*                                *

¿Impresión de prueba? (s/n) N
```

Se borra entonces la pantalla y se imprime lo siguiente:

```
Juan y Pepa Martinez Rebollo    Luisa Sala y Carmen Garcia Carrión
Majada del Viento, 25           Gran Via, 34
28029 Madrid                    Orihuela
MADRID                          ALICANTE
```

salvar "etiquet3" ↵ salvar "mdv1_etiquet3"

5. Salve el programa como etiquet.

Un programa de menús

Salva ha llamado todos los procedimientos con nombres que comienzan por eti, de manera que permanezcan juntos cuando se unan a un programa mayor.

Se da cuenta, sin embargo, que el programa abre otro fichero, y que podría quedarse sin memoria si se utiliza con programas grandes, una presentación en pantalla y otros ficheros de datos abiertos, sobre todo si están ordenados. La solución es ejecutar, mejor que unir, los nuevos programas. Al unir programas se incrementa el número de procedimientos en memoria, si los ejecuta cambia los que había por un conjunto nuevo.

Escribe un menú que ofrezca la elección entre los distintos programas que se pueden ejecutar (cargar y comenzar automáticamente). Para los ficheros de boda el menú incluirá bodacom, listado, etiquet (para otra impresión) y bodafich (para las copias de seguridad).

cargar "vacío"

1. Cargue el programa vacío para borrar los procedimientos de etiquetas de memoria.

El procedimiento siguiente se debe salvar sólo por una duplicación más fácil.

editar ↵ F5 progmenú ↵

2. Cambie de nombre al programa vacío por progmenú.

```
proc progmenu
  limpiar
  escribir en 0,20;"PROGRAMA DE BODA"
  escribir en 5,5;"1. Mantenimiento de la lista de Invitados"
  escribir en 6,5;"2. Listado e impresión de Invitados"
  escribir en 7,5;"3. Impresión de etiquetas"
  escribir en 8,5;"4. Copias de Seguridad"
```

```
escribir en 9,5;"A. Salir de Archive"
escribir en 12,5;"Pulse la opción elegida"
mientras 1
  haz c$=mayús(tecla())
  escribir c$
  si c$="1": ejecutar "bodacom": finsi
  si c$="2": ejecutar "listado": finsi
  si c$="3":start: finsi
  si c$="4": ejecutar "bodafich": finsi
  si c$="A": abandonar : finsi
  finmientras
finproc
```

3. Escriba el resto del procedimiento.

ESC ESC salvar "progmenú" ↵ salvar "mdv1_progmenú"

4. Salve el procedimiento.

cargar "etiquet3" ↵ unir "progmenú"

5. Vuelva a cargar el programa de etiquetas y una el programa progmenú.
6. Cree un procedimiento de comentarios, a:

```
proc a
  nota etiquet_prg   10 5 25
  nota start: abre fichero de datos y tabla. prepara valores de etiqueta
  nota etialin: imprime asteriscos en el comienzo de la etiqueta
  nota etiquetas: bucle de fichero principal y de etiquetas; almacena, imprime
  nota etilin: preparar los campos de datos para las lineas de etiqueta
  nota etisal: se le pasa un campo que se añade a linea
  nota etimpr: imprime linea de etiquetas (tabla de registros)
  nota etinuelin: imprime lineas hasta la siguiente fila
  nota linimpr: se le pasa un texto que se imprime o escribe
  finproc
```

salvar "etiquet" ↵ salvar "mdv1_etiquet"

7. Salve la versión final del programa.

tirar "etiquet1_prg" ↵ tirar "etiquet2_prg" ↵ tirar "etiquet3_prg"

8. Borre las versiones de desarrollo de la cinta. Use **F5** para repetir la línea, las teclas de edición ⇧ ⇨ para cambiar el número, y se ahorrará volver a escribir la línea cada vez.

Uso del menú de programas

Para que el menú de programas funcione con otros programas, el procedimiento progmenú se debe unir con todos ellos, y llamarse al finalizar la ejecución (por ejemplo, cuando c$ valga S en BodaCom).

Esto hace falta porque al ejecutar un fichero se borran los procedimientos actuales, incluido el que usó el comando.

Los programas listado y bodafich deben ser "arreglados", escribiendo un menú y procedimiento start antes de poderse usar con este sistema. Recuerde también que ejecutar sin start causa un error "comando irreconocible".

Este nuevo método sacrifica la velocidad por el espacio y la facilidad de trabajo.

Tablas y manejo de texto

Si usted está familiarizado con los lenguajes de programación como el BASIC, probablemente se dio cuenta de que Salva estaba usando el archivo eti como una tabla monodimensional en el programa de etiquetas.

No hay ninguna necesidad de entender lo que es una tabla o vector para usar una de sus mayores ventajas: la posibilidad de designar variables por el número de su posición en una lista. En términos prácticos, significa que se puede usar contadores para cambiar las variables que se van a usar en los bucles.

Tablas

El diagrama siguiente ilustra una tabla bidimensional (llamada también matriz) de datos.

	1 CODIGO	2 JUGUETE	3 COSTE	4 PRECIO
1	LIT	Lagarto Lito	578	1990
2	LOL	Llama Lola	690	1750
3	PIP	Pablo el Pingüino	356	1100

Los ficheros de datos de ARCHIVE son tablas bidimensionales (2 ejes de coordenadas bastan para describirlos). Cualquier elemento se puede describir de una manera única por nombre (por ejemplo, coste del lagarto) o por números (tercer campo del primer registro).

Los archivos de ARCHIVE tienen grandes sobrecargas en términos de memoria, y además usan de vez en cuando los *microdrives*, ralentizando inaceptablemente algunas operaciones. Es preferible poder crear tablas de variables normales.

Una "tabla unidimensional" es simplemente una manera elegante de describir una lista de elementos relacionados.

En el programa de etiquetas de Salva hay varias variables llamadas linea$ a la vez: linea$ registro 1, linea$ registro 2, y así sucesivamente. Todas se llaman linea$, de manera que un solo comando pueda modificarlas a todas. Se puede poner en juego las distintas linea$ usando un número (el número de registro) para especificar cuál.

La alternativa es crear un número fijo de variables con nombres distintos y escribir un comando por línea, por ejemplo: escribir linea1$, escribir linea2$, escribir linea3$, etc.

Así se desperdicia un montón de espacio en memoria, es largo de introducir, difícil de adaptar e inherentemente inflexible.

ARCHIVE se construyó originalmente como una base de datos con potentes facilidades de lenguaje, más que como un lenguaje con potentes facilidades de manejo de ficheros (lo que también es). De momento no incluye facilidades para el manejo de tablas en su repertorio de instrucciones.

Las tablas, sin embargo, tienen muchas propiedades muy útiles, que se pueden emular de varias maneras.

Cadenas como tablas de tamaño fijo

Otro método de imitar las ventajas de una tabla monodimensional usa cadenas de caracteres para almacenar las variables.

El siguiente procedimiento extrae subcadenas de longitud fija de una serie de texto mayor de acuerdo con su posición (número) en la cadena:

```
proc listames
  haz m=0
  haz m$="EneFebMarAbrMayJunJulAgoSepOctNovDic"
  mientras m<12
    haz comienzo=(3*m)+1
    escribir m+1;" ";m$(comienzo hasta comienzo+2)
    haz m=m+1
    finmientras
finproc
```

Se muestra 1 Ene 2 Feb 3 Mar y así sucesivamente.

Cada mes es una subcadena fija de tres caracteres de longitud, y esta técnica se puede usar con cualquier longitud fija, siempre que las constantes numéricas (3, 2 y 12) se cambien a la medida.

Hay un límite de 255 caracteres para la suma de las longitudes parciales (la longitud máxima de una cadena) que debe tener en cuenta cuando use esta técnica. Esta opción, por ejemplo, no

puede usarse para las etiquetas (por ejemplo, 5 líneas × 3 columnas × 35 caracteres = 525 caracteres).

El ejemplo anterior usa una cadena alfanumérica proporcionada al principio. En la práctica se podría crear y mantener en otra rutina y contener información cambiante.

Así, para cambiar la séptima subcadena del ejemplo anterior:

```
haz comienzo = (3*6) + 1
haz m$ = m$(hasta comienzo — 1) + nuevo$ + m$
    (comienzo + 3 hasta)
```

La nueva subcadena debe, naturalmente, tener exactamente 3 caracteres de longitud.

Subcadenas de longitud variable

Se pueden almacenar y leer subcadenas de longitud variable de un "vector" de texto sin necesidad de disponer de una longitud fija equivalente al máximo:

```
proc listalcoy
  haz n$="Juan*Maya*Eva*Tere*Chona*"
  haz ultcom=0
  mientras ultcom<long(n$)
    haz proxcom=ultcom+enserie(n$(ultcom+1 hasta ),"*")
    escribir n$(ultcom+1 hasta proxcom-1);" Alcoy"
    haz ultcom=proxcom
    finmientras
  finproc
```

Este procedimiento muestra:

```
Juan Alcoy
Maya Alcoy
Eva Alcoy
Tere Alcoy
Chona Alcoy
```

El procedimiento busca un separador (en este caso un asterisco) que marca el comienzo y final de las subcadenas.

> *Recuerde:* enserie() no devuelve simplemente 1 si se encuentra una subcadena, sino el número de la posición inicial.

Cada búsqueda de un nuevo asterisco comienza tras el último hallado, y el nombre es la subcadena entre el último asterisco y el próximo.

El bucle se detiene cuando nuevaster es igual a la posición del último asterisco de la serie. Es obvio que el carácter que se use como separador no debe aparecer nunca como parte de una subcadena.

Una ejecución en seco del procedimiento queda como sigue:

Pasadas por el bucle	1	2	3	4	5
haz ultast =	0				
mientras ultstart< long(n$)					
haz nuevaster =	0+5=	5+5=	10+4=	14+5=	19+6=
	5	10	14	19	25
escribir parte de n$	1—4	6—9	11—13	15—18	20—24
haz ultast =	5	10	14	19	25
finmientras					

Se pueden almacenar valores numéricos en una cadena de texto convirtiéndolos primero mediante serie(). Este sistema se puede usar para incrementar efectivamente un contador bucle en pasos irregulares. Por ejemplo, almacene una cadena de caracteres "0 4 5" y extraiga los valores por turno para usarlos en valcam() en un bucle. Así afecta sólo al primer, quinto y sexto campos.

Troceando las líneas de dirección

El procedimiento que acabamos de describir puede ser útil para manejar los registros de direcciones. Muchas listas de correo necesitan ordenarse en distintos momentos por distintos campos, como el nombre, nombre de la empresa, ciudad, barrio o provincia. ARCHIVE puede ordenar sólo sobre un campo, así que cada elemento necesitará un campo. Sin embargo, puede haber de una a tres líneas de casa, calle y ciudad, y, si se permiten tres campos, algunos estarán vacios con frecuencia, causando problemas al imprimir etiquetas.

Una solución es almacenar todas las líneas de dirección en un campo, delimitadas por comas, y trocear este campo como se considere conveniente.

```
Empresa$: Plásticos Martínez, S.A.
Direc$  : Calle 34, Bloque 6, Polígono Industrial Dos Piedras.
ciudad$ : Villamás
provinc$: Soria
```

```
proc cortadirec
  haz separ$=","
  haz ult=0
  mientras ult<long(l$)
    haz prox=ult+enserie(l$(ult+1 hasta ),separ$)
    escribir l$(ult+1 hasta prox-1)
    haz ult=prox
    finmientras
  finproc
```

Observe que este procedimiento usa las comas sólo como separadores, y que no las imprime.

Más sobre los campos de dirección vacíos

Otra manera de acabar con el problema de las líneas en blanco usa un contador en etilin para contar los campos, y sólo almacena las líneas que contienen algo. Otro bucle almacena blancos en los campos linea$ que quedaron en blanco. Por ejemplo:

```
proc etilineas
  primero 'eti"
  mientras cuenta<muncam("eti")
    haz cam$=valcam(recnum(cuenta),fd$)
    si cam$>""
      etilin;cam$+rept(" ",ancho+long(cam$))
      próximo
      finsi
    haz cuenta=cuenta+1
    próximo
    finmientras
mientras no finf()
  etilin;rept(" ",ancho)
  próximo
  finmientras
finproc
```

Líneas demasiado largas

La técnica de subcadenas se puede usar para "partir" en dos las líneas demasiado largas para entrar en el máximo ancho de una etiqueta.

El siguiente procedimiento comienza por buscar en la posición máxima, y se mueve hacia atrás hasta que encuentra un espacio. El espacio se usa aquí como un separador que permite identificar el principio y el final de las palabras. La línea se corta en el primer espacio, y el resto de la línea se sigue investigando mientras resulta demasiado larga.

Al procedimiento se le pasan dos parámetros: la cadena que debe ajustarse (linea$) y la longitud máxima de una línea (max).

```
proc ajusta;linea$;max
haz comienzo = 1
haz pos = max
haz long = long(linea$)
mientras pos<long
    mientras linea$(pos)<>" "
        haz pos = pos — 1
        finmientras
    escribir linea$(comienzo hasta pos — 1)
    haz comienzo = pos + 1
    haz pos = pos + max
    finmientras
finproc
```

Observe que el procedimiento no contiene sentencias si. Las condiciones que gobiernan la ejecución del bucle fallan automáticamente cuando no hay más trabajo que hacer.

Salva podría usar una mezcla de este procedimiento y el ejemplo anterior (con unas pocas modificaciones) para cortar las líneas demasiado largas de su programa de etiquetas.

Separadores numerados

El método que acabamos de tratar sirve sobre todo para extraer subcadenas en secuencia.

Se pueden buscar por número si se almacena un número tras el primer delimitador de la subcadena. Por ejemplo:

```
proc listaloy
  haz n$="*1Juan*2Maya*3Eva*4Tere*5Chona*6"
  haz max=val(n$(long(n$)))
```

```
haz cuenta=1
mientras cuenta<max
  haz buscnum$="*"+serie(cuenta,2,0)
  haz ultast=enserie(n$,buscnum$)
  haz proxast=ultast+enserie(n$(ultast+1 hasta ),"*")
  escribir n$(ultast+2 hasta proxast-1);" Alcoy"
  haz cuenta=cuenta+1
  finmientras
finproc
```

Observe que la posición inicial de la subcadena impresa se ha incrementado para eliminar la impresión del número, y que se ha almacenado un número al final de la cadena para proporcionar un número máximo de pasadas por el bucle.

Hay un límite para la longitud del texto total, pero se pueden almacenar varias cadenas como campos de un fichero. El bucle de búsqueda de una subcadena se puede anidar dentro de otro bucle que recorra los registros.

Este es un caso donde resulta, por lo general, más práctico usar campos en vez de subcadenas, pero la mezcla de ambos métodos puede ser útil.

Por ejemplo, podría formar la base de una tabla tridimensional con los ejes siguientes: registros, campos y subcadenas.

Las tablas tridimensionales pueden servir para imprimir gráficas, mover gráficas por la pantalla, menús de selección múltiple en una sola pantalla (usando escribir en) o niveles múltiples de subtotalización.

En la práctica, con mucha frecuencia se pueden usar registros y campos en lugar de cadenas para la mayoría de los propósitos. Decida qué sistema usar de acuerdo con los requerimientos de la tarea que debe realizar.

Estos métodos de almacenamiento y uso de variables son más lentos que usar una variable por cada elemento, pero los procedimientos son más compactos y flexibles.

La función VALVAR()

En ARCHIVE versión 2 se puede usar la función valvar() para sacar datos de tablas de variables.

valvar() devuelve el contenido de la variable nombrada en su argumento.

Por ejemplo, escribir valvar(nombre$) podría presentar Juan.

Este sistema permite almacenar información en una serie de variables (var1$, var2$, var3$, etc.) y ser luego extraída por un bucle con contador con líneas como la siguiente:

```
escribir valvar("var" + str(cuenta,3,0) + "$")
```

Si cuenta fuera 2, ARCHIVE leería la línea como:

```
escribir valvar("var2$")
```

Procedimientos que escriben procedimientos

Los comandos y los nombres de campos mismos no se pueden almacenar en variables, para luego referirnos a ellos por el nombre de la variable. Por ejemplo, no se puede pedir que se elija el fichero invitado con los comandos siguientes:

```
haz elec$ = "direc3$": elegir elec$ = "Barcelona"
```

El comando ordenar es otro ejemplo. Un fichero puede contener muchos campos, pero escribir un procedimiento que ofrezca una elección de ordenaciones implica una repetición de comandos como el siguiente:

```
si elec$ = "1": ordenar nombre$;a: finsi
si elec$ = "2": ordenar empresa$;a: finsi
si elec$ = "3": ordenar ciudad$;a: finsi
```

Hay una manera de evitar este problema y usar un solo procedimiento para ordenar, de acuerdo con su elección.

Cree primero un fichero ASCII en cinta (mediante via) que parezca un fichero _prg, y use unir para traerlo de nuevo a memoria.

El método, de nuevo, es lento pero muy flexible. El siguiente prodecimiento ofrece la ordenación de cualquier fichero por cualquier campo.

El procedimiento de control reordena llama a cuatro procedimientos que ejecutan cada paso por turno:

```
proc eligecampos
  haz ok=0
  mientras no ok
    leer en 13,10;"Ordenar. ¿Sobre qué campo?";num
    si num>=0 yy num<=númcam()
      haz ok=1
      finsi
    finmientras
  finproc
```

```
proc escproc
  vla "secuenc_prg"
  imprimir "proc ponenorden"
  imprimir "ordenar "+nomcam(num)+";a"
  imprimir "finproc";
  imprimir car(0)+car(26);
  normal
  unir "secuenc"
  ponenorden
  finproc
```

```
proc listacampos
  haz cuenta=0
  mientras cuenta<númcam()
    escribir tab 10;cuenta;".";nomcam(cuenta)
    haz cuenta=cuenta+1
    finmientras
  finproc
```

```
proc reordena
  limpiar
  listacampos
  eligecampos
  escproc
  pantalla
  finproc
```

Si se usa con el fichero invitado y se elige el número 5, se crea el siguiente procedimiento:

```
proc ponenorden
    ordenar vienen$;a
    finproc
```

Selecciones complejas y variables

Un candidato más lógico para este tratamiento es elegir o buscar. Muchos programas de listado o producción de informes necesitan ofrecer un conjunto muy complejo de selecciones, búsquedas con enserie(), búsquedas exactas, elecciones exclusivas, etcétera. Cuantos más campos tenga un archivo, más complejo se puede hacer.

El siguiente ejemplo ilustra algunos pasos que se podrían dar para escribir un programa de selección variable:

Obtención de la opción:

```
proc seleccion
  escribir "ELIJA CRITERIO DE SELECCION (<! para todos)"
  leer "¿Què código?";selcod$
  leer "¿Què mes?";selmes
```

Almacenamiento de la expresión:

```
haz selexp$=""
si selcod$>""
  haz selexp$=selexp$+" yy código$='"+selcod$+"'"
  finsi
si selmes>0
  haz selexp$=selexp$+" yy mes="+serie(selmes,2,6)
  finsi
```

Escritura, unión y uso del procedimiento:

```
si selexp$>""
  via "selec_prg"
  imprimir "proc selecsi"
  imprimir "elegir 1=1 "+selexp$
  imprimir "finproc"+car(0)+car(26)
  normal
  unir "selec"
  selecsi
  finsi
finproc
```

Si se eligen LIT y 6, el procedimiento producido quedará así:

```
proc selecsi
  elegir 1=1 yy código$="LIT" yy mes=6
  finproc
```

Observe que elegir 1 = 1 es un truco que permite que cualquier condición comience por yy. Uno siempre es igual a uno, y evalúa como cierto, así que elegir 1 = 1 es lo mismo que elegirlo todo (no seleccionar nada), dejando que las otras condiciones gobiernen el comando.

Los procedimientos se pueden escribir así porque se almacenan como archivos ASCII. Cada vez que se carga un procedimiento, ARCHIVE lee los comandos letra a letra, y los convierte en instrucciones que entiende el QL.

Con la versión 2 se puede salvar y cargar procedimientos cuyos procedimientos se han convertido en códigos únicos (llamados token). De esta manera ocupan mucho menos espacio en cinta y cargan más deprisa. Estos códigos se conocen también como "código objeto".

La palabra clave objeto salva los procedimientos de esta manera. Por ejemplo: salvar "bodacom" objeto.

Los ficheros de programa salvados con objeto tienen por defecto la extensión _pro en vez de _prg, y se pueden leer poniendo la palabra objeto tras cargar, unir o ejecutar. Por ejemplo, ejecutar "bodacom" objeto.

Se puede editar todavía los procedimientos salvados de esta manera, pero sólo con el editor de programas de ARCHIVE.

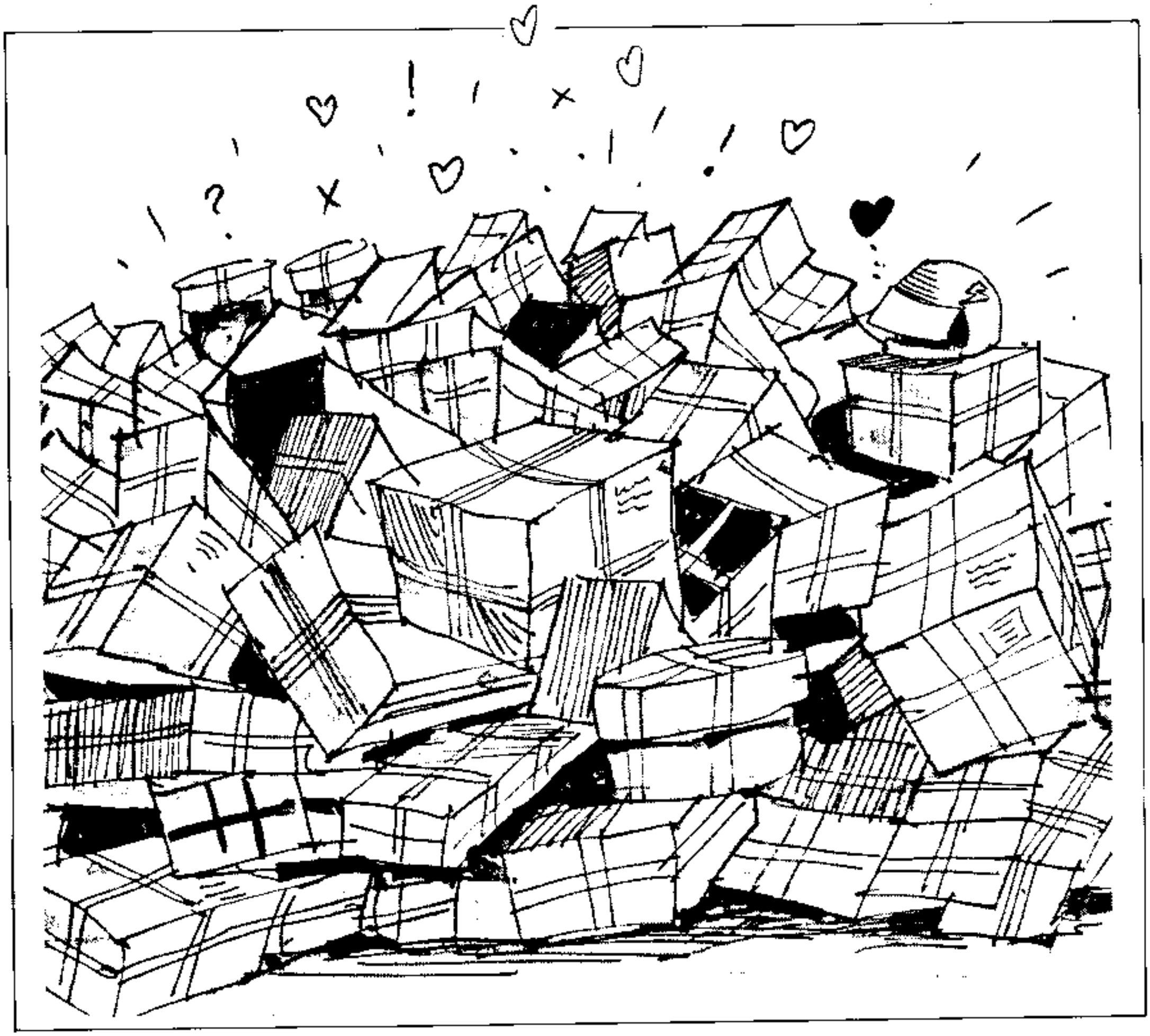
Se puede evitar hasta esto. Si desea proteger sus programas, use la palabra protec con salvar. Por ejemplo: salvar "bodacom" protec.

Al volver a cargar usando objeto los programas protegidos no se pueden listar ni editar. Y ¡tenga cuidado de no proteger su única versión de un importante programa!

Trucos útiles

Hay varias maneras de asegurarse que los programas serán fáciles de adaptar y manejables:

- Use variables globales mejor que textos literales para los mensajes y formatos de impresora, y las condiciones de los bucles. Definalos al comienzo del programa.
- Use variables o ficheros como tablas. Gestione su manejo con bucles.
- Mantenga los procedimientos tan pequeños como sea posible y limitados a un solo propósito. Ponga las líneas relacionadas con otras funciones en otro procedimiento (y llámelo), o déjelas en el procedimiento "padre" si es ahí donde deben estar.
- Use parámetros para estar seguro de que las variables necesarias están disponibles para una tarea particular.
- Use el mismo nombre de variable para tareas comunes, como señales, salida y contadores. Hágalas *locales* al procedimiento que las usa.



¡Al fin solos!

Listado del programa de impresión de etiquetas

El procedimiento de listado de etiquetas debe quedar como sigue:

```
proc a
  nota etiquet_prg   10 5 85
  nota start: abre fichero de datos y tabla. prepara valores de etiqueta
  nota etialin: imprime asteriscos en el comienzo de la etiqueta
  nota etiquetas: bucle de fichero principal y de etiquetas: almacena, imprime
  nota etilin: preparar los campos de datos para las lineas de etiqueta
  nota et:sal: se le pasa un campo que se añade a linea
```

```
  nota etimpr: imprime linea de etiquetas (tabla de registros)
  nota etinuelin: imprime lineas hasta la siguiente fila
  nota linimpr: se le pasa un texto que se imprime o escribe
  finproc
proc etialin
  haz ok=0
  mientras no ok
    escribir en 13,10;"¿Impresión de prueba? (s/n)";
    haz prueba$=mayús(tecla())
    escribir prueba$
    si prueba$="N"
      haz ok=1
      sino
      si prueba$="S"
        haz n=0
        mientras n<colum
          etilin
          haz n=n+1
          finmientras
        etimpr
        etinuelin
        finsi
      finsi
    finmientras
  finproc
proc etilin
  primero "eti"
  mientras no finf()
    si prueba$="N"
      haz cam$=valcam(recnum("eti"),fd$)
      sino
      haz cam$="*"
      finsi
    haz cam$=cam$+rept(" ",ancho-long(cam$))
    etisal;cam$
    próximo
    finmientras
  finproc
proc etimpr
  primero "eti"
  mientras no finf()
    linimpr;linea$
    próximo
    finmientras
  finproc
proc etinuelin
  haz cuenta=0
  mientras cuenta<lineas-recnum()
    linimpr;""
    haz cuenta=cuenta+1
    finmientras
  finproc
proc etiquetas
  limpiar
  mientras no fint(fd$)
    haz n=0
    mientras no fint(fd$) yy n<colum
      etilin
      proximo fd$
      haz n=n+1
      finmientras
    etimpr
    etinuelin
    finmientras
  cerrar "eti"
```

```
  finproc
proc etisai;txt$
  si n=0
    haz linea$=""
    finsi
  haz linea$=linea$+txt$
  actualiz
  finproc
proc linimpri;txt$
  escribir txt$
  finproc
proc start
  haz fichdat$="invitado"
  haz fd$="i"
  nota abrir fichdat$ lògico fd$
  abrir fichdat$+"_eti" lògico "eti"
  haz colum=2
  haz ancho=31
  haz lineas=9
  etialin
  etiquetas
  finproc
```

Apéndice A

Si tiene problemas

GENERALIDADES

Recuerde la tecla de **AYUDA, F1**. Está siempre disponible, y no le hace perder lo que estaba haciendo.

Si el ordenador no responde al teclado

- Pulse la tecla **ESC** o ↵ si tiene problemas; le devuelven a territorio conocido.
- Si el ordenador persiste en su actitud tras un tiempo prudente, es posible que necesite reinicializar la máquina. Perderá cualquier material en memoria temporal que no haya salvado a *microdrive*, y puede que también cualquier fichero abierto que se haya modificado. Para inicializar, saque los cartuchos y pulse el botón del costado derecho del QL. La máquina hace la comprobación de la memoria y vuelve a la pantalla inicial, donde se puede volver a llamar a ARCHIVE insertando el cartucho de programa y pulsando **F1** o **F2**.

Si el ordenador no carga ARCHIVE

- ¿Está la pantalla conectada al ordenador?
- ¿Están ordenador y pantalla conectados a la fuente de alimentación?
- ¿Tiene un cartucho con el programa ARCHIVE en la unidad de la izquierda?

Si la respuesta a las tres preguntas es *sí*, reinicialice la máquina.

Si un cartucho gira y gira sin parar

Sucede cuando pide el directorio de un *microdrive* que no está formateado. Saque el cartucho de la unidad que *no* gira y pulse el botón de **RESET**. Formatee el cartucho incorrecto.

Nombres de archivos

- No use espacios en los nombres. No use tampoco '_' en un nombre de archivo, ya que ARCHIVE no comprueba los nombres que incluyen ese carácter al inicio. El carácter '_' se usa, por ejemplo, para acceder a un sistema en red local (por ejemplo, "_neto_1") o a otros periféricos suplementarios.
- Los nombres de fichero no deben tener más de ocho caracteres y deben contener sólo letras o números. Deben comenzar siempre por una letra.

Tiempos de carga crecientes

- Los *microdrives* se vuelven poco eficientes a medida que se llenan. A medida que la cinta se acerca al 90 por 100 de ocupación, hay menos espacios (y más separados) para alojar archivos. Los ficheros se vuelven más y más fragmentados y el cartucho debe dar varias vueltas para cargar el fichero completo.
- En cuanto note que sus ficheros tardan mucho en cargar, use el comando salvagrdr para copiarlos en un cartucho recién formateado. Tras copiarlos todos, borre el cartucho (preferiblemente formateando). Puede entonces reutilizar el cartucho. El proceso de copia reorganiza los ficheros en zonas contiguas del cartucho, y disminuye el tiempo necesario para cargarlos.

Dificultades en la carga o copia de cartuchos

- Si no puede cargar cualquiera de los cuatro programas suministrados con el QL, intente hacer una copia de mdv2_ a mdv1_. Si hace la copia de esta manera produce una copia que ejecutará satisfactoriamente en el mdv1_.
  Se puede hacer situando el cartucho original en la unidad 2 y un cartucho virgen en mdv1_. Escriba lrun mdv2_clone (véase el capítulo 1, *Primeros Pasos*).
- Si clone no consigue una copia completa, inténtelo de nuevo con un cartucho distinto en mdv1_, o intente copiar el fichero que falló con el comando copy del SUPERBASIC, o el comando salvagrdr de ARCHIVE:

  copy mdv2_nomfichero to mdv1_nomfichero

  Si tiene éxito la copia del archivo, copie los ficheros restantes de mdv2 a mdv1 (para preparar un cartucho vacío, use format mdv1_).

Archivos estropeados

- Los archivos salvados en cinta resultan a veces imposibles de cargar o abrir de nuevo. Mensajes de error como error de lectura, E/S incompleta, Tipo de fichero incorrecto o Sin memoria son indicativos, aunque no exclusivos, de ficheros estropeados.

Citemos entre las causas posibles:

- Apagar el QL o encenderlo con cartuchos en la unidad.
- Reinicializar el QL con cartuchos en las unidades.
- Reinicializar el QL en insertar.
- Picos de tensión en el QL o los *microdrives.*
- Daño físico al cartucho o la cinta (pliegues, grasa o polvo, tensión excesiva).
- Salvar ficheros en unas unidades excesivamente calientes.
- Exposición de los cartuchos a campos magnéticos fuertes.
- Exposición a cargas estáticas (carpetas de nilón, pantallas de TV, etc.).

La única manera de prevenir estos problemas es mantener varias copias de seguridad constantemente.

Rescate de un fichero de datos estropeado

Si no tiene copias de seguridad de un fichero de datos y éste se estropea, puede todavía salvar los registros en buen estado.

Intente abrir el archivo con ver e indicar. Si funciona, busque el primer registro estropeado con el comando siguiente:

```
todos: pescribir: escribir recnum( ): fintodos
```

Busque ahora el último registro estropeado así:

```
último: mientras recnum( ) > 0: anterior: pescribir: escribir
recnum( ): finmientras
```

Use el método de transferencia del capítulo 5 para crear una copia vacía de la estructura del archivo, añada los registros hasta y desde los estropeados.

Tendrá que recrear los registros estropeados a partir de listados o del material fuente original.

MENSAJES DE ERROR

Mensajes de error del SUPERBASIC

Hay varios mensajes de error que vienen del SUPERBASIC, y aparecen normalmente durante la carga de ARCHIVE o las copias del cartucho de programas mediante clone. (Los programas que se llaman con lrun, como boot o clone, son programas BASIC.)

Son mensajes como:

```
en línea 200 línea incorrecta
en línea 10 medio incorrecto
```

Los mensajes significan, por regla general, que el programa está estropeado, y necesita copiarse de nuevo (mediante clone) a partir de la copia original de ARCHIVE.

Mensajes de error de ARCHIVE

Pueden aparecer dos tipos de mensaje de error cuando se utiliza ARCHIVE: los que tienen número de error, y los que no.

Mensajes de error sin número

Línea incorrecta

ARCHIVE hace ciertas comprobaciones sobre las líneas que se introducen durante la edición. Si encuentra algo inaceptable, aborta la línea completamente e imprime ese mensaje en el AREA DE TRABAJO. Por ejemplo,

```
escribir nota
```

ya que nota es una palabra reservada, que significa un comentario, y la línea no tiene sentido.

No hay fichero en la unidad

El mensaje *no* aparece si realmente no hay ficheros en la unidad, ya que dir muestra siempre el nombre de volumen y los sectores libres. Si la cinta no está formateada, el *microdrive* gira sin detenerse.

Este mensaje quiere decir que se ha usado un nombre de unidad incorrecto, por ejemplo, mdv1 (sin el subrayado), o mdv3_ sin tener una tercera unidad. Aparecerá también si no hay cinta en la unidad, o si el cartucho no está insertado correctamente.

A veces aparece este mensaje incorrectamente si se usa dir tras un mensaje como E/S incompleta (error 91) o cuando cambia los cartuchos mientras hay ficheros abiertos. Posiblemente la única solución sea reinicializar el ordenador.

No hay ayuda

Este mensaje quiere decir que el cartucho de programa ARCHIVE no está en la unidad 1, o el cartucho de programa no contiene el archivo archive_hob en él. No quiere decir que la función de ayuda se haya quedado sin información sobre un comando o situación particular.

Mensajes de error numerados

Cuando ARCHIVE detecta un error en un comando o en una línea de un procedimiento, muestra un número de error y un mensaje de error, con el formato siguiente:

Error 1: comando no reconocido

Este número de error es el que devuelve la función numerr( ).

1. Error: comando no reconocido
   Causa: Normalmente se trata de un comando mal escrito, o de una llamada a un procedimiento que no está en memoria.

   por ejemplo: añadr
   debe ser: añadir

2. Error: esperado fin de instrucción
   Causa: ARCHIVE encuentra más información de la que espera en una línea.

   por ejemplo: haz x = 3 haz y = 4
   debe ser: haz x = 3: haz y = 4

   Ocurre también cuando se le olvida pasar un parámetro a un procedimiento que debe llevarlo. Es porque ARCHIVE no espera que la línea proc del procedimiento llamado tenga ningún parámetro.

3. Error: esperado nombre de variable
   Causa: ARCHIVE espera que lo siguiente sea una variable y no la encuentra.

   por ejemplo: ordenar
   debe ser: ordenar nombre$;a

4. Error: elemento de impresión no reconocido
   Causa: Tras un comando escribir, ARCHIVE espera un nombre de variable, una expresión, o caracteres entre comillas. Una palabra clave no puede ser elemento de impresión.

   por ejemplo: escribir crear

5. Error: tipo incorrecto de datos
   Causa: Se trató de pasar una variable alfanumérica a un procedimiento que espera un parámetro numérico, o viceversa.
6. Error: esperada expresión numérica
   Causa: Se intentó asignar texto a una variable numérica.

   por ejemplo: haz x = "si"
   debe ser: haz x = 25

7. Error: esperada expresión alfanumérica
Causa: Tratar de asignar números a una expresión alfanumérica, o el uso de una variable numérica como un nombre de fichero.

por ejemplo: haz x$ = 25
debe ser: haz x$ = "si"

por ejemplo: abrir x
debe ser: abrir x$

8. Error: variable no hallada
Causa: Se ha nombrado una variable que no existe. A menudo se trata de una palabra clave que se ha escrito incorrectamente, haciendo pensar a ARCHIVE que se trataba de una variable.

por ejemplo: mientras no finf
debe ser: mientras no finf( )

por ejemplo: escribir tab61;notas$
debe ser: escribir tab 61;notas$

9. Error: variable indefinida
Causa: Se ha tratado de hacer actuar un comando sobre una variable que no ha sido creada o inicializada. Es improbable que llegue a ver este mensaje, ya que las variables indefinidas producen, por regla general, el mensaje anterior (error núm. 8).

por ejemplo: escribir z (z indefinida)

10. Error: falta separador
Causa: Dos expresiones en la misma línea deben ir separadas por un separador (coma o punto y coma). Por regla general aparece por la falta de comas en listas (de parámetros o de leer), o por punto y coma en listas de impresión.

por ejemplo: escribir en 5;notas$
debe ser: escribir en 5,1;notas$

11. Error: nombre demasiado largo
Causa: La asignación de un nombre de más de 13 caracteres a una variable.

por ejemplo: haz zzzzzzzzzzzzzzzzzzzzzzz = 13
debe ser: haz zzzzzzzzzzzz = 13

12. Error: nombre duplicado.
Causa: Si se abre un fichero con un nombre que ya existe, o si se asigna un nombre de campo dos veces.

por ejemplo: abrir "agenda"lógico"a" ↵ abrir "activo" lógico "a"
debe ser: abrir "agenda"lógico"a" ↵ abrir "activo" lógico "ac"

por ejemplo: crear "a": direc$: direc$: fincrear
debe ser: crear "a": direc1$: direc2$: fincrear

13. Error: esperada serie literal
Causa: Se intentó importar un fichero que no tiene el formato que espera ARCHIVE. Por ejemplo, ARCHIVE espera que la primera información en un fichero importado sean los nombres de los campos del nuevo fichero de datos, que deben ir entre comillas.

14. Error: falta finproc
Causa: Debe ocurrir sólo en ficheros de programa construidos mediante un editor de textos distintos del de ARCHIVE, ya que el editor de ARCHIVE proporciona automáticamente el finproc.

15. Error: sentencia proc incorrecta
Causa: Véase error 14.

16. Error: fin de instrucción prematuro
Causa: ARCHIVE espera más expresiones o comandos.

por ejemplo: crear "agenda": fincrear
debe ser: crear "agenda": notas$: fincrear

17. Error: error en estructura de programa
Causa: Por regla general significa que falta un fintodos, finmientras o finsi en un procedimiento. Se puede originar también añadiendo un finproc superfluo.

18. Error: demasiados números
Causa: Indica que está intentando introducir más números de los que caben en la memoria reservada para entrada. El error puede ocurrir en una línea de entrada del teclado, o mientras carga un programa con muchos números en una de las líneas. El límite exacto depende de las circunstancias (un número típico está entre 15 y 20 números), así que es improbable que vea este mensaje.

por ejemplo: escribir 1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + (1 + 1 + (1 + (1)))))))))))))))))))

21. Error: fichero existente
Causa: Sólo en la versión 2. Los comandos salvagrdr y crear no pueden borrar un fichero existente con el mismo nombre.

27. *Causa:* Cuando se pulsa la tecla **ESC**, mientras se ejecuta un procedimiento, la función numerr( ) devuelve el valor 27. **ESC** hace que los procedimientos se detengan exactamente igual que si se produjera un error. 27 es el código ASCII de la tecla **ESC**.

50. Error: faltan comillas de cierre
Causa: No es necesariamente que falten las comillas de cierre, sino que una línea de comando tiene un número impar de comillas. ARCHIVE espera que las comillas aparezcan en pares rodeando a las constantes alfanuméricas.

por ejemplo: abrir "agenda"lógico a"
debe ser: abrir "agenda" lógico "a"

51. Error: falta exponente después de "E"
Causa: E en un número debe ir seguido por un exponente.

por ejemplo: haz x + 1.2e
debe ser: haz x + 1.2e5
ya que, en notación científica, el valor tras la E representa la potencia de 10 por la que se debe multiplicar la parte de antes de la E. En el ejemplo x sería 120000.

52. Error: número demasiado grande
Causa: ARCHIVE puede manejar sólo números cuyos valores caigan en el intervalo 1E-38 a 1E + 38.

por ejemplo: escribir 10000000000000000000000000000000000000000000

53. Error: símbolo desconocido
Causa: Si se pone un símbolo en una expresión en una

posición en que ARCHIVE espera un número o un carácter.

por ejemplo: haz x = %

70. Error: error de sintaxis
Causa: La expresión que se debe evaluar (sumar, restar, etcétera) está incompleta o incorrecta.

por ejemplo: haz x = 3 +

71. Error: falta paréntesis
Causa: No hay el mismo número de paréntesis abiertos que cerrados en una expresión.

por ejemplo: haz x = (3 + 5)/7)
debe ser: haz x = ((3 + 5)/7)

72. Error: valores incompatibles
Causa: La mezcla de elementos numéricos y alfanuméricos en la misma expresión. Las expresiones no deben mezclar ambos tipos.

por ejemplo: haz x = 3 + "5"
debe ser: haz x = 3 + 5

Apéndice B
Memoria y número máximo de registros

El número máximo de registros que caben en un fichero depende de la memoria permanente (cartucho o *diskette*) y también de la memoria temporal (RAM).

Límite teórico

ARCHIVE puede almacenar 65535 registros en un fichero si hay suficiente memoria.

Límite de un *microdrive*

Cada campo numérico en un archivo ocupa 8 bytes de memoria.

Un byte es la medida de memoria, y representa (a efectos prácticos) la memoria necesaria para almacenar un carácter.

Cada campo alfanumérico ocupa tantos bytes como su longitud más uno. Debe estimar la longitud media de las series de texto de cada registro.

Un *microdrive* tiene aproximadamente 100K (un disco, 720K), así que la fórmula siguiente da el número máximo de registros que puede almacenar un sólo fichero.

```
102400/( (cn * 8) + (ct * 1) + prom)
```

cn es el número de campos numéricos
ct es el número de campos de texto
prom es el promedio que ocupa un campo de texto

Multiplique la respuesta por 0.8 para reducir el tamaño máximo (y en el caso del disco cambie 102400 por 720 * 1024). Los cartuchos llenos tardan tiempos demasiado largos para salvar mientras buscan trozos de espacio libre.

Consideremos la RAM

Cuando se acaba de cargar ARCHIVE queda libre una cierta cantidad de RAM para su uso por procedimientos, pantallas y ficheros de datos.

En la versión 1, la cantidad oscila alrededor de 10K, y en la versión 2, 15K.

Para verlo, escriba:

```
escribir memoria( )
```

Así escribe la cantidad de memoria libre en bytes.

Esta cantidad disminuye cada vez que carga o abre un archivo. Los archivos de pantalla ocupan aproximadamente 500 bytes, y se reservan inicialmente hasta 500 bytes por cada fichero de datos. Los ficheros de programa ocupan un espacio proporcional a su tamaño. Las variables también gastan memoria libre.

La consideración clave que gobierna el número máximo de registros de los ficheros abiertos es la cantidad de memoria disponible para los índices de los ficheros.

Estructura de datos e índice

Cuando se abre un fichero, ARCHIVE no carga sólo los datos, sino también información relativa al fichero.

ARCHIVE organiza tres zonas en memoria:

- definición de fichero, que incluye los nombres de los campos,
- índice del espacio libre en el fichero,
- índice de los registros.

El último es el más importante. ARCHIVE sólo sabe dónde buscar un registro (tenga en cuenta que pueden ser todos de distinto tamaño), porque el índice tiene la posición del registro en el fichero y su longitud.

Requerimientos del índice

Cuando se ordena un fichero, se añaden 8 bytes de cada campo clave al índice, para su uso por situar.

Así, cada nivel de ordenación incrementa el espacio total de memoria que necesita un fichero de datos.

Existe un límite superior de 8K (8 × 1024) para la zona de índices. Dividiendo este tamaño por las necesidades de índice se puede calcular el máximo número de registros de un archivo.

Cada registro de un archivo crea una carga en el índice de los siguientes bytes:

sin ordenar	6 bytes
ordenado por 1 campo	14 bytes
ordenado por 2 campos	22 bytes
ordenado por 3 campos	30 bytes
ordenado por 4 campos	38 bytes

La fórmula para el número máximo de registros de un fichero ordenado por 2 campos es, por tanto, 8192/14, que es 585.

Si se queda sin memoria, puede bien reducir el número de ficheros abiertos a la vez, o bien use claves compuestas (véase el capítulo 8) para reducir el número de claves de ordenación.

Puede reducir la memoria necesaria para los procedimientos ahorrando en las variables, acortando nombres y textos, eliminando las sentencias nota, usando "tablas" y bucles para no repetir comandos, o ejecutando programas menores uno tras otro (véase el capítulo 13).

LIMITACIONES

Longitud máxima de texto:	Campo — 160 Variable — 255
Precisión matemática:	13 cifras
Número entero menor/mayor:	$\pm 1.7^{\pm 38}$
Ordenación:	Hasta 4 niveles de campos clave Primeros 8 caracteres de texto significantes.

Máximo número de procedimientos por programa en memoria: 255.

Máximos campos en un registro: 255.

Variables de campo y nombres de procedimiento: 13 caracte-

res, el primero letra, sin espacios ni signos de puntuación
Las variables de texto pueden terminar en $
No pueden ser palabras clave de ARCHIVE

Máximo número de registros en un fichero: Hasta 65535, pero mucho menos con frecuencia

Nombres de fichero: Los nombres de unidad en el formato mdv1_.
Los nombres de fichero hasta 8 caracteres, el primero alfabético, sin espacios
La extensión separada por subrayado, hasta tres caracteres

Apéndice C
Diferencias entre versiones

Nuevas funciones en la versión 2

ARCTG( ), COS( ), DEC( ), GRAD( ), EXP( ), NOMCAM( ), LN( ), . SEN( ), TAN( )

Nuevas palabras clave de la versión 2

OBJETO, PROTEC

Estructura de fichero incompatible

ficheros _scn

Memoria disponible al cargar

Versión 1: aprox. 12.000 caracteres
Versión 2: aprox. 16.000 caracteres

Comillas

'"' es inaceptable en la versión 2

pleer, insertar, alterar, leer

Versión 1: los caracteres alfanuméricos en campos numéricos causan error
Versión 2: el programa no permite introducir caracteres alfanuméricos en campos numéricos

Versión 1: ↵ cicla de nuevo entre los campos
F4 ningún efecto
160 caracteres máximo por línea

Versión 2: ↵ salva el registro si está en último campo
F4 abandonar
El máximo es el ancho de la pantalla

Nuevo mensaje de error en versión 2

Error 21: fichero existente

Apéndice D

El código ASCII

El nombre ASCII viene de las iniciales, en inglés, de "Código Americano Estándar para el Intercambio de Información".

El código pensado para permitir que diferentes marcas y elementos de equipo, como ordenadores, impresoras y monitores, se entiendan uno a otro. Hay códigos para los números, las letras y los signos de puntuación, además de señales especiales como el "salto de página" (FF).

Este código es el estándar internacional para microordenadores. Algunos grandes ordenadores y miniordenadores (por ejemplo, los de IBM) y algunos micros, por ejemplo los Commodore, no siguen ese código, pero son la excepción que confirma la regla.

Conversión de los códigos ASCII a caracteres

Los códigos son números entre 0 y 127, y se pueden convertir en los caracteres (incluyendo los números del 0-9) o señales que representan usando la función car( ).

La función car( ) puede escribir también caracteres para los códigos entre 128-255, pero éstos producen resultados distintos en los distintos ordenadores y no forman parte del estándar. La

"ñ" y las vocales acentuadas están entre los caracteres del "segundo conjunto".

Cómo funcionan

Los ordenadores manejan toda la información como flujos de impulsos eléctricos, que son alto o bajo, positivo o cero. Los manipula mediante la aritmética binaria, donde todos los valores se pueden representar por una serie de unos y ceros.

El sistema de numeración que usamos habitualmente es el decimal, que representa todos los números por una serie de cifras entre 0 y 9.

Cada número binario tiene su equivalente en decimal:

binario		**decimal**
00000001	es igual a	1
10110001	es igual a	177
11111111	es igual a	255

y así sucesivamente

La letra **A** mayúscula se almacena en memoria y en los cartuchos como una serie de señales que representan el número binario 1000001, ó 65 en decimal.

El comando escribir "A" muestra la letra A en la pantalla, igual que escribir car(65). En otras palabras, la función car( ) le permite usar datos de una manera muy cercana a la representación interna del ordenador. Esto puede ser extremadamente útil cuando se manejan periféricos como impresoras, *modems* u otros ordenadores.

Conversión de caracteres a ASCII

La función código( ) devuelve el código ASCII en decimal del primer carácter de una serie de texto. Por ejemplo:

```
escribir código("A")          escribe el número 65
escribir código("Botella")    escribe el número 66
```

Así, **A** es la representación alfabética o de carácter de 65, que es la representación decimal de 1000001, que es una representación binaria de las señales eléctricas que se usan en el ordenador para representar la letra **A** en el texto.

Códigos corrientes

Los códigos que hacen que se imprima una letra, un número o un signo de puntuación se llaman *códigos "imprimibles"*. Los principales son:

32-47	Espacio y signos de puntuación
48-57	Números del 0 al 9
65-90	Letras mayúsculas (A a la Z)
97-122	Letras minúsculas (a a la z)

Por ejemplo, la letra **I** se almacena como el número ASCII 49, y el espacio ' ' como el número 32. La **M** mayúscula se almacena como 77, y la **m** minúscula como 109. Los valores ASCII para las mismas letras mayúsculas y minúsculas difieren *siempre* en 32.

Los otros códigos que es más probable que encuentre son:

car(0)	carácter nulo
car(27)	**ESC** o escape

Estos dos se usan como códigos de control en combinación con otros para cambiar su significado. Se usan a menudo para enviar información a impresoras y terminales.

car(10)	**LF** o salto de línea
car(12)	**FF** o salto de página
car(13)	**CR** o retorno de carro
car(7)	**BEL** o campana. Por lo general causa un ruido, pero ARCHIVE no lo reconoce como tal
car(26)	**EOF** o marcador de fin de fichero, también llamado ⇧ Z o **CTRL-Z**

Hexadecimal

Algunas teclas ASCII muestran los números en dos notaciones: decimal y hexadecimal. (Hexadecimal significa, en griego, 16: usa el número 16 en lugar de las 10 cifras normales. 16 es más útil en un ordenador que 10.)

La notación hexadecimal incluye las letras A a F, además de los números, pero algunos números hexadecimales se parecen a los decimales (por ejemplo, 12 en decimal es en hex 0C, pero 65 decimal = 41 en hex). Compruebe antes de usar un número el tipo de números de que se trata.

Apéndice E

Proveedores de Software para el QL

UTILIDADES Y HERRAMIENTAS

Adder Publishing	68000 Macro Assembler.
Ariolasoft	Q Doctor.
Bedsoft	Screen Editor.
Computer One	Monitor. Forth. Assembler. Typing Tutor.
Computability	Expresso Coppee. Cartridge copier.
DA Bandoo	Assembler. Screen Editor.
Data Management	SBUTIL. MBACKUP. TERMINAL. CHARGEN.

	SBEXTRAS.
	FM.
	QLLIFE.
	QLFED.
Digital Precision	QL Super Monitor & Disassembler.
	QL Sprite Generator.
	QL SuperBASIC Compiler.
Eidersoft	QLART.
	QL Dump.
	QL Archiver.
	QL Sketch Pad.
Goose Software	Printing Format Utilities.
Hisoft	MONQL.
J & D Software	Artist QL.
Micrologic Consultants Ltd.	QL Terminal Emulator.
Miracle Systems	Screen Dump.
PCS	PCS Utilities.
Posi-tron Computing	Hi-res graphics screendump.
Pscientific Software	Q Keydefine.
Q Code	Terminal Emulation
	68000 Assembler/Editor
	Games - Sprite Shooting.
	Lander.
	Moon Landing.
Saltgrade Software	File Manager.
	Font Editor.
Sigma Research	Sketchpad.
Software 2000	QL Backup/Restore.
Synapse Software	QL Debugger.
Tasman Software	Tasprint QL.
	Tascopy QL.
WD Sotfware	WD Utilities.
	Ref QL.

LENGUAJES

Computer One	Forth. Pascal.
Digital Precision	QL Super Forth 83.
RE Jackson	Forth 79.
Metacomco	BCPL. LISP. ASSEMBLER.
MicroAPL Ltd.	APL.
Micro Proccessor Engineering Ltd.	QL Forth 83.
TDI Ltd.	UCSD PASCAL. UCSD FORTRAN-77. Advanced Development Toolkit. UCSD P System. UCSD Prolog.

DOMESTICO Y ENTRETENIMIENTO

Bedsoft	Gambler. Beat the Clock. Autodraw.
Blain Software	Merry Muncher. Fire Tower. Advance Invaders.
Brainstorm	Westmonster Palace.
Cam Mawr Products	QL Pools. QL Snake.
Cenprime Software	QL Bank Account.
Computer One	Typing Tutor.
CP Software	Bridge Player.
D B Microservices	QMAIL. Cassette Index.
Dialog Software	Home Accounts Manager.

Digital Precision	QL Super Backgammon. QL Astrologer. QL Arcade Game. QL Reversi.
Eidersoft	Zapper. Archiver. QL ART. QSpell.
Equate	Solar Invaders. Wall Breaker. Draughts. Mind Your Path. Statistical Averages. Calendar.
Games Workshop	D Day.
Intersoft	Executive Adventure.
Key Software	QL Early Learner.
New Horizon Software	Pacmen. QBERT. Golf.
Medic Data Systems	Metropolis. M-Paint.
Megacycal Software Ltd	Gunshoe Logic.
Microdeal	Lands of Havoc. Hopper. Cuthbert in Space.
Newtech Publishing Ltd	Tycoon.
Paddy Software	Q-Rev.
Peak Electronics	QL Colour Quest.
Portfolio Software	Stockmarket Manager.
Psion	QL Match Point.
Quality Leader Software	QL Crossword.
Quest Automation	Blackjack. West. Zappit. Quest - the Adventure.
Rodent Software	Adventure Writer. QL Artist. 2 × 7 Games Cartridges.

S B Software	Fantasia Adventure.
Shadowsoft	Area Radar Control.
	Strategy Game.
	Quazimono.
	Space Paranoids.
	Night Nurse.
	Paint Master.
	Galactic Invaders.
Snowsoft	Hungry Harry in the Haunted House.
Summit Software	Frogger.
	Dungeon.
Superplant Software	Plant and Gardening Software.
Swansoft	Space Trek.
Synapse Software	QL Maths Package.
Talent Software	ZKUL.
	West.
	Graphic QL.
	Cartridge Doctor.
Victory Software	Pirate Island.
	Advance Attack.
	Fire Tower.
	Food Freak.
WD Software	WD Morse Tower.
Westway Ltd.	EVA.
Yorkshire Mails	QL Poolswimmer.

SISTEMAS OPERATIVOS

PCM L	CPM/80.
Lattice	"C" Cross Compiler for IBM PC - QL.
GST	68K/OS.
	QC (C Compiler).
Quest	CP/M 68K
	68000 Version of CP/M
	- microdrive and floppy disk.

NEGOCIOS

Data Management	QL diary. Databoss.
Dialog Software	TRANSACT. Sales Ledger. Invostat.
Quantum Mechanics	QSpell.
TR Computer Systems	QL Payroll.
Q-Soft	Agenda.

PROVEEDORES DE *HARDWARE* PARA EL QL

Broomspring	QL Computer Desk.
Care Electronics	Centronics Printer Interface. Printer Selector.
Compak Data	Modem and Printer Controller.
Computamate Data Products	Q-Disc Controller. Q-Disc Drive System.
Computer Supplies CST	Joysticks. Disc Interface. Q Dsic 40/80 track drives 200K 5.25'' 800K '' Centronics Interface. IEEE Interface.
Data Efficiency Design Consultancy	Sinclair Vision Monitor. OASIS MADC12LQ.
Eidersoft	QL Joystick adaptor. QL Joystick.
Eprom Serovces	Serial to Parallel Adaptor. QL Eprom Cartridge. Various Eprom Software.
Kempston Micro Electronics Ltd.	Centronics Interface. Disk Interface.

Lightwave Leisure	The Stick.
Micropost	Joystick Interface. Joystick. Interface and Joystick.
Microperipherals	Disc Interface. 3.5'' disc drives. Serial to parallel lead.
Microvitec	Monitor.
Miracle Systems	Parallel Printer Interface. Double Expander. Joystick Adaptor.
Modem House	QL Bright Star.
PCML Ltd.	QL + RAM cards, 64K, 128K, 256K.
Pl Computers Ltd.	Dust cover.
Powertron	''The Plug'' (spike spoiler).
Quest International	Memory Expansion, 64K, 128K, 256K, 512K. Operating System. CP/M 68K Floppy Disc. Microdrive. Disc Drives, 200K, 400K, 800K. Winchester Drives. Expansion Console. Monitor Stand.
Sigmas Research	Parallel Printer Interface.
Silicon Express Ltd.	Disc Interface 5.25''. 80 track, double sided interface. Interface and single drive. Interface and dual drive.
Simplex Data Ltd.	RAM Expansion 256K, 512K.
Strong Computers	QL Acoustic Modem. QL Computermate Disc Interface. 800K Drive. 1.5Mb Drive.
Tandata Marketing	Q-Connect. Q-Connect & Q-Mod. Q-Connect & Q-Mod & Q-Call.

Technology Research Ltd.	Delta Disc Interface with centronics interface. As above with 64K and 128K RAM.
Transform Ltd.	Centronics Interface Dust Cover. Cartridge Box.
Voltmace Ltd.	Joysticks.
4 Systems	Cartridge Box. Cartridges.

PRODUCTOS PARA EL QL DE SINCLAIR RESEARCH

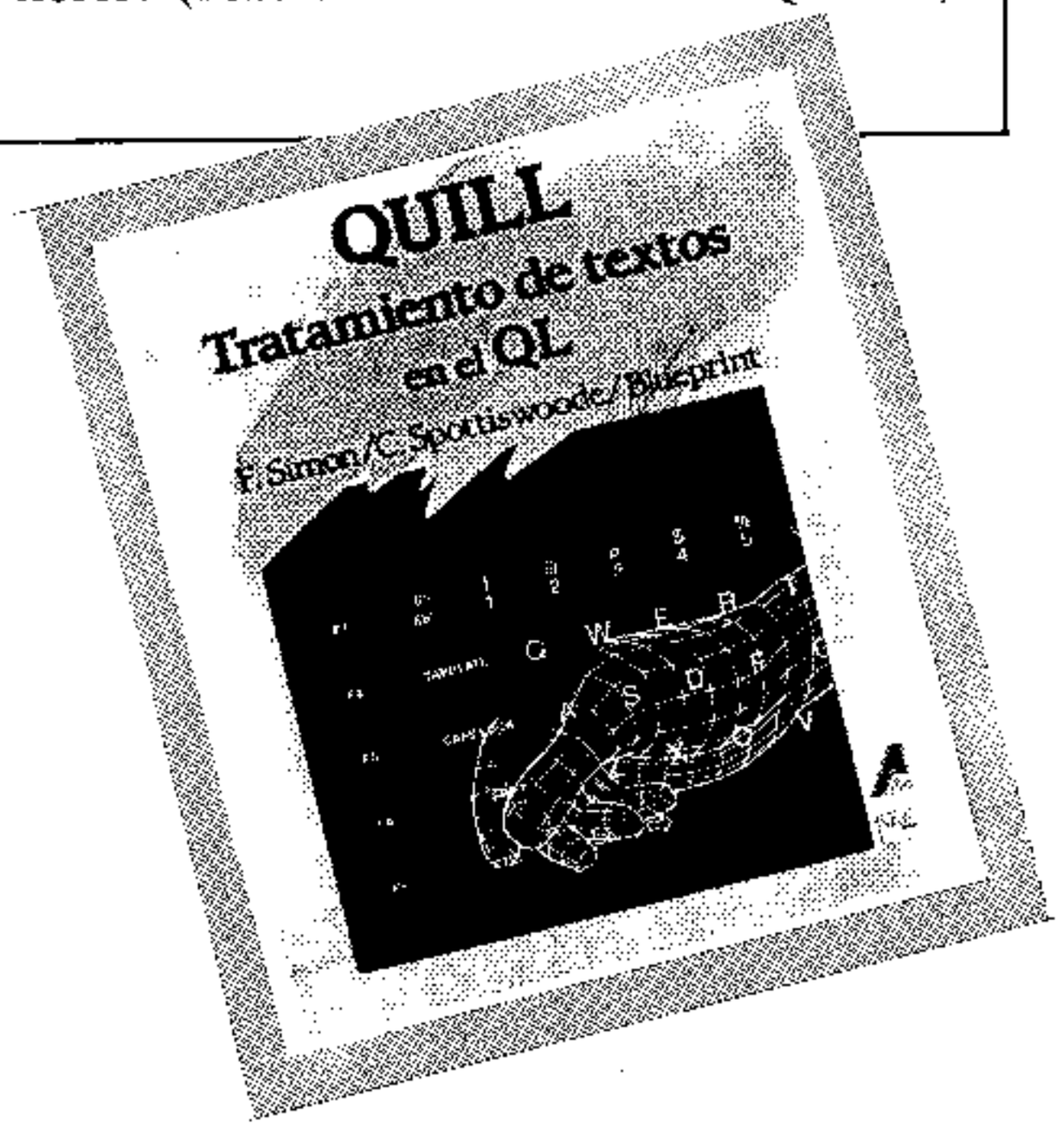
QL Chess.
QL Assembler.
QL Monitor.
QL Toolkit.
QL Touch "n" Go.
QL Cash Trader.
QL Decision Maker.
QL Entrepreneur.
QL Project Planner.
QL Integrated Accounts.
QL Home Finance.
QL Cavern.
QL Cardener.
QL Macroassembler.

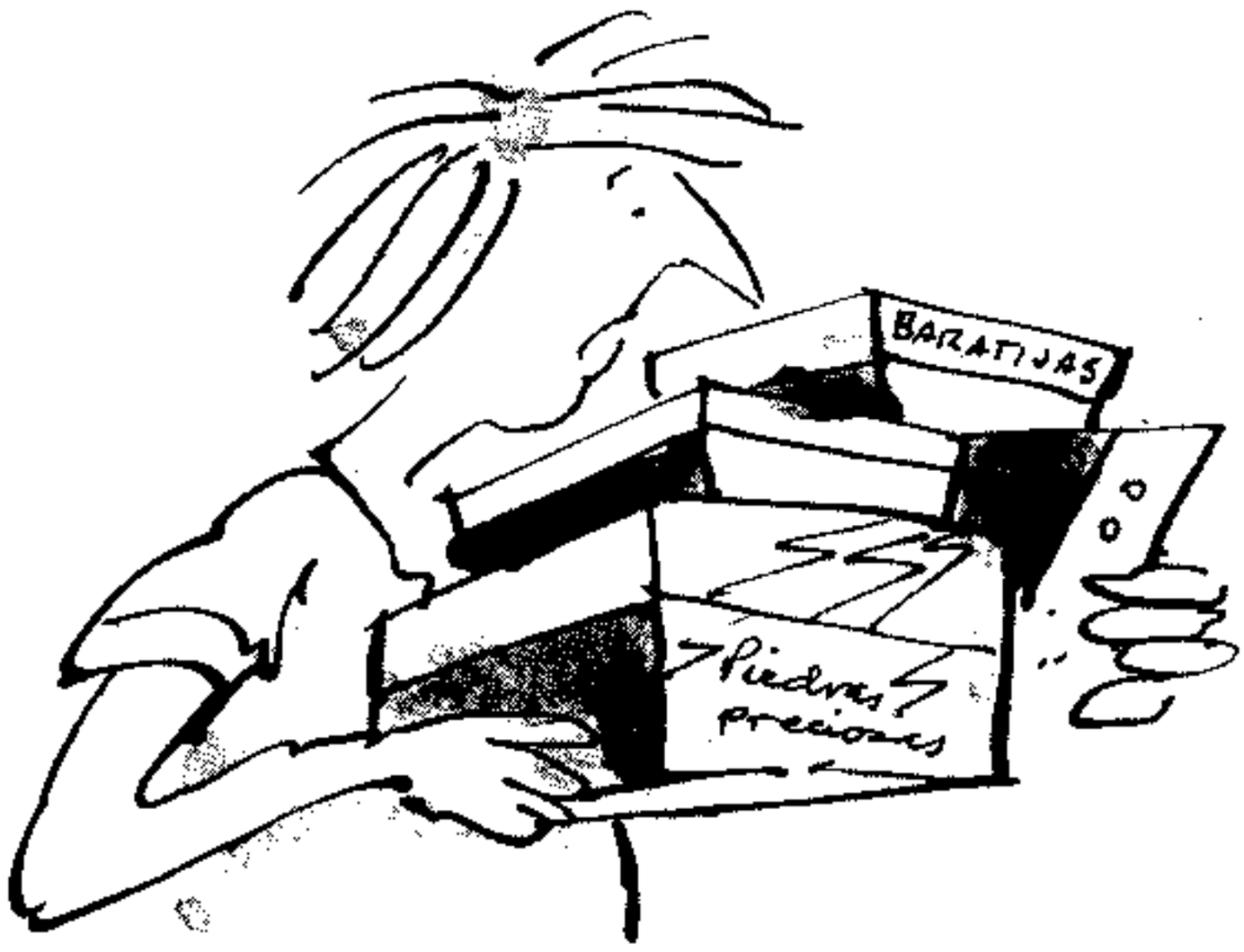
Publicaciones

QL Technical Guide.

Indice alfabético

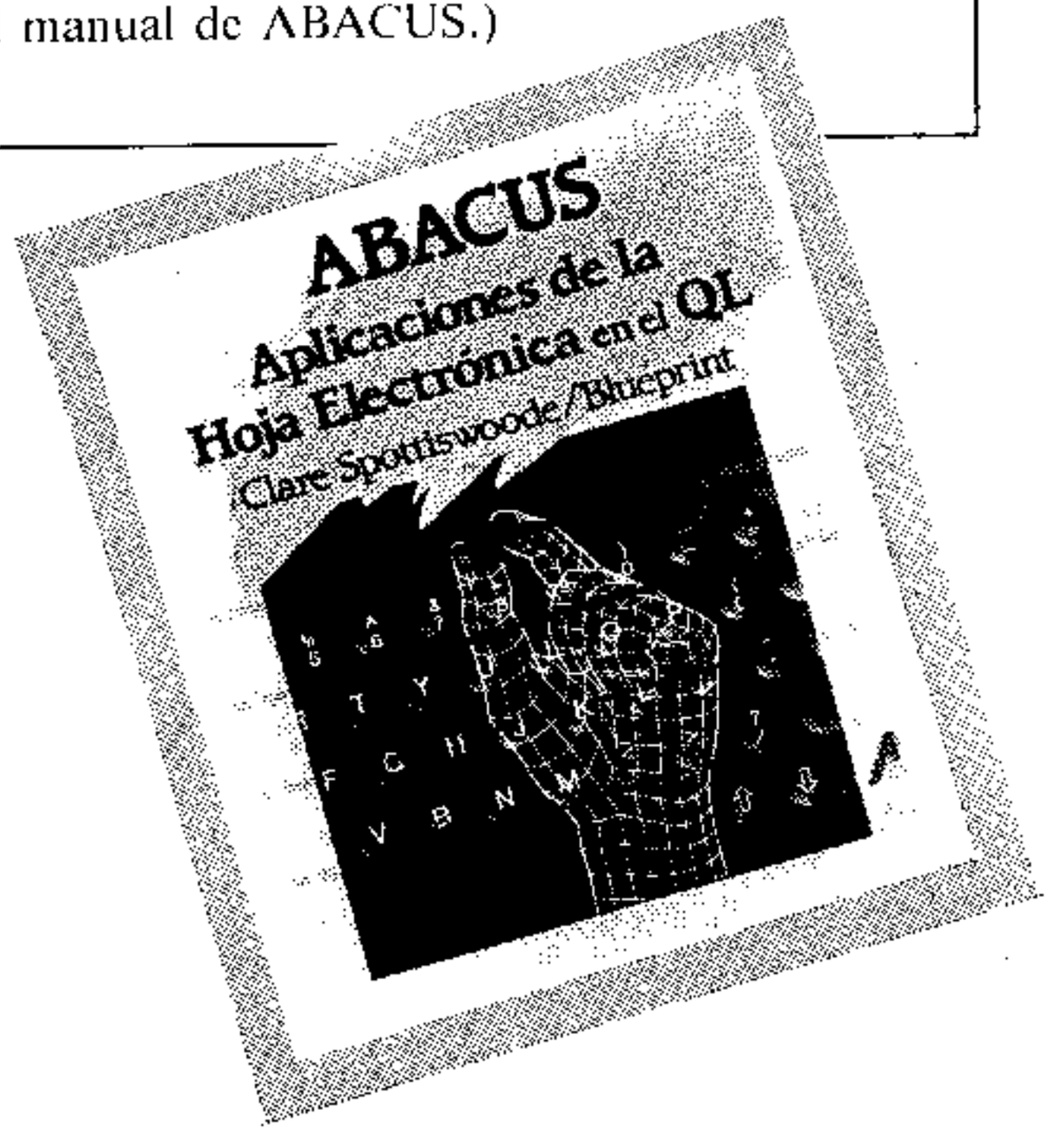
¿Habrá aceptado Maya al nuevo novio de su hija, o seguirá pensando en el médico de las flores? (Entérese en el manual de QUILL.)





¿Llegará Salvador al altar? O simplemente terminará trabajando para su suegra. (Para informarse vea el manual de EASEL.)

¿Llegarán los broches de Maya para la venta de Navidad? (Entérese en el manual de ABACUS.)



ANAYA MULTIMEDIA

INFORMATICA PERSONAL-PROFESIONAL

EL LIBRO DEL WORDSTAR. *Trucos y recursos.*—Julie Anne Arca.
EL LIBRO DEL ATASI ST. *Manejo, aplicaciones y GEM.*—Jeremy Vine.
EL LIBRO DEL IBM PC, XT, AT. *Programación, uso y aplicaciones.*—Louis E. Frenzel, Jr., Louis E. Frenzel III.
PROGRAMACION EN C. *Introducción y conceptos avanzados.*—M. Waite, S. Prata, D. Martin.
MARKETING Y VENTAS CON LOTUS 1-2-3. *Técnicas comerciales para su microordenador.*—Michael V. Laric, M. Ronald Stiff.
PROGRAMACION DEL Z80.—Rodnay Zaks.
EL LIBRO DEL APPLE IIc. *Programación, uso y aplicaciones.* Philip Lieberman.
EL LIBRO DEL RS-232.—John Campbell.
EL LIBRO DEL LOTUS 1-2-3. *Desarrollo de aplicaciones profesionales.*—Alan Simpson.
MICROINFORMATICA PARA DIRECTIVOS. *Cómo rentabilizar su ordenador personal-profesional.*—Dick Heiser.
PROGRAMACION GRAFICA EN EL IBM PC.—Dan Illowsky, Michael Abrash.
QUILL. *Tratamiento de textos en el QL.*—F. Simon, C. Spottiswoode, Blueprint.
EASEL. *Gráficos de negocios en el QL.*—Alison Spottiswoode, Blueprint.
ABACUS. *Aplicaciones de la hoja electrónica en el QL.*—Clare Spottiswoode, Blueprint.
ARCHIVE. *Manejo de la base de datos en el QL.*—Ian Murray, Blueprint.
MATEMATICAS PARA PROGRAMADORES. *Sistemas de numeración y aritmética binaria.*—William Barden, Jr.
INTRODUCCION AL UNIX SISTEMA V.—Mitchell Waite, Stephen Prata, Donald Martin.
GEOMETRIA DE TORTUGA. *El ordenador como medio de exploración de las matemáticas.*—Harold Abelson, Andrea diSessa.

ANAYA MULTIMEDIA

Colección «MICROINFORMATICA»

Angell, I. O. y Jones, B. J.: DISEÑO DE GRAFICOS Y VIDEOJUEGOS (incluye cassette).
Beechhold, Henry F.: EL LIBRO DEL HARDWARE. No destape su ordenador personal... sin leer antes este libro.
Birmingham Educational Computing Centre: INTRODUCCION A LA TECNOLOGIA DE LA INFORMACION. PREINFORMATICA.
Bishop, Peter: PROGRAMACION AVANZADA EN BASIC.
Brown, Peter: PASCAL A PARTIR DEL BASIC.
Cavalcoli, Aldo: EL ORDENADOR PERSONAL: COMO ELEGIRLO Y UTILIZARLO.
Coccione, L. y Winter, G.: LOS ORDENADORES NO MUERDEN.
Dachslager, H., Hayashi, M. y Zucker, R.: PROGRAMACION EN BASIC: UN METODO PRACTICO.
Dewhirst, J. y Tennison, R.: TU PRIMER LIBRO DEL ZX SPECTRUM.
D'Opazo, J. y Grupo GOLEM: PROGRAMACION EN LOGO.
Durst, J.: «SPRITES» Y GRAFICOS EN LENGUAJE MAQUINA (ZX SPECTRUM).
Galende Domínguez, F.; Sánchez López, A.; Alfaraz López, M. y Sánchez García, J. A.: COMETAS EN TU MICRO: EL HALLEY. Cálculos de órbitas y parámetros de cometas en BASIC.
Gavin, Maurice: ASTRONOMIA: EL UNIVERSO EN TU ORDENADOR.
Gibbons, John P.: PROGRAMACION AVANZADA DEL COMMODORE 64. Ampliación del BASIC y rutinas gráficas.
Greenwood, Gareth: CODIGOS Y CLAVES SECRETAS. Criptografía en BASIC.
Hammond, R.: EL ORDENADOR Y TUS HIJOS.
Hartnell, Tim: EL LIBRO GIGANTE DE LOS JUEGOS PARA ORDENADOR.
Hartnell, Tim: INTELIGENCIA ARTIFICIAL: CONCEPTOS Y PROGRAMAS.
Hartnell, Tim: EL LIBRO GIGANTE DE LOS JUEGOS PARA ZX SPECTRUM.
Hartnell, Tim, y otros: EL LIBRO GIGANTE DE LOS JUEGOS PARA DRAGON.
Hartnell, Tim: EL SUPERLIBRO DE LOS JUEGOS PARA ORDENADOR.
Heller, R. S. y Martin, C. D.: BITS Y BYTES: INICIACION A LA INFORMATICA.
Hollerbach, Lew: MICROINFORMATICA: CONCEPTOS BASICOS.
Hurley, R.: JUEGOS GRAFICOS DE AVENTURA PARA ZX SPECTRUM.
Johnson, David: DESCUBRE LAS MATEMATICAS CON TU MICRO.
Johnston, J.: MICROS: TAMAÑOS, FORMAS Y SABORES.
Johnston, J.: MICROS: BIPS, PITIDOS Y LUCES.
Johnston, J.: MICROS: MENUS, BUCLES Y RATONES.
Kosniowski, Czes: MATEMATICAS DIVERTIDAS EN BASIC.
Kramer, S.: PROGRAMACION AVANZADA DEL ZX SPECTRUM.
Lacey, Andrew: LIBRO GIGANTE DE LOS JUEGOS PARA MSX.
Núñez, Agustín: PROGRAMACION DEL INTERFACE 1 Y MICRODRIVE.
Otero, M. A.; Pueyo, M. A. y Cajaraville, J. A.: PRIMEROS PASOS EN LOGO. El mundo de la tortuga Fan. Libro del profesor. Libro del alumno.
O'Shea, T. y Self, J.: ENSEÑANZA Y APRENDIZAJE CON ORDENADORES. Inteligencia artificial en educación.
Pentiraro, Egidio: EL ORDENADOR EN EL AULA.
Pritchard, Joe: DESCUBRE TU MSX. Programación y aplicaciones.
Pritchard, Joe: LENGUAJE MAQUINA MSX. Introducción y conceptos avanzados.
Rosso, Vincenzo de: COMO SE PROGRAMAN LOS ORDENADORES.
Sato, T.; Mapstone, P. y Muriel, I: MSX: GUIA DEL PROGRAMADOR Y MANUAL DE REFERENCIA.
Servello, Fausto: ¿QUE ES LA TELEMATICA?
Snover, S. L. y Spikell, M. A.: JUEGOS MATEMATICOS DE INGENIO EN BASIC.
Thomasson, Don: PROGRAMACION AVANZADA DEL AMSTRAD. Entradas y salidas de la ROM.
Webb, David: LENGUAJE MAQUINA AVANZADO PARA ZX SPECTRUM.
Zaks, Rodnay: EL LIBRO DEL BASIC.

ARCHIVE
Manejo de la Base de Datos en el QL

ARCHIVE es el potente programa y lenguaje de ***base de datos*** *del QL.*

El propósito principal de ARCHIVE es almacenar y recuperar información de forma ágil y flexible, convirtiendo al QL en una potentísima máquina de gestión.

ARCHIVE: Manejo de la base de datos en el QL *es una introducción directa y práctica al uso profesional de ARCHIVE que muestra paso a paso, con numerosos ejemplos y ejercicios, cómo usar cada función del programa para aprovechar toda su potencia.*

En el libro se presta especial atención al ***lenguaje de procedimientos*** *de ARCHIVE que permite desarrollar aplicaciones especializadas que resuelvan tareas complejas y facilita todas las operaciones de ordenación, selección y mantenimiento de registros, así como la personalización de pantallas y la preparación de documentos.*

En ***ARCHIVE: Manejo de la base de datos en el QL*** *encontrará, además:*

- *Explicación de todas las funciones y comandos paso a paso.*
- *Trucos e ideas para optimizar el uso del lenguaje de ARCHIVE.*
- *Guía de referencia rápida de ARCHIVE.*
- *Solucionario de problemas: cómo reaccionar ante los mensajes de error.*
- *Glosario de la terminología empleada en ARCHIVE.*

ARCHIVE: Manejo de la base de datos en el QL *le permitirá empezar a aplicar seriamente su QL en su trabajo o afición, ¡descubra ARCHIVE y verá solucionados todos sus problemas de información!*

Los títulos de la serie *QL* de ANAYA MULTIMEDIA: "ARCHIVE: Manejo de la base de datos en el QL", "ABACUS: Aplicaciones de la hoja electrónica en el QL", "QUILL: Tratamiento de textos en el QL", "EASEL: Gráficos de negocios en el QL", han sido producidos con la cooperación y apoyo de *PSION Ltd.*, autores de los programas.

ANAYA MULTIMEDIA