ENTERPRISE

Empezamos usando la máquina. De esta manera nos habituamos a ella y podremos comprobar sus posibilidades. Los detalles técnicos los estudiaremos más adelante.

Es mejor seguir esta parte del manual por el orden en que ha sido escrita, porque su misión es la de ayudarnos a conocer nuestro nuevo ordenador. Siempre que se pueda revisar un tema antes de estudiar otros, se proporcionan referencias cruzadas, a fin de poder aprender el funcionamiento de la máquina en cualquier orden y con la rapidez que se desee. La segunda parte trata de cada 'sección' de la programación (todas ellas están en realidad íntimamente relacionadas como veremos) con detalle y a un nivel ligeramente más alto que en la primera parte. La Sección de Referencia al final nos ayudará a descubrir más sobre el Enterprise una vez que hemos aprendido los fundamentos del control del ordenador.

Antes de estudiar algunos programas, experimentamos un poco con el teclado. Podemos escribir lo que se nos antoje sin perjudicar en absoluto al ordenador. Si el ordenador deja de presentar letras en respuesta a lo que se escribe, pulsemos simplemente el botón 'reset' de la parte posterior. Este mando es realmente útil para juguetear con el ordenador ... y más tarde veremos lo manejable que es. Mientras tanto, dediquémosnos a conocer el ordenador; tiene mucho que ofrecernos.

Nota: Cuando estamos usando el ordenador para proceso de textos (ver páginas 38-42 y las tablas de las páginas 36 y 37), no es preciso insertar el cartucho IS-BASIC. Sin embargo, para escribir programas BASIC, es preciso introducir este cartucho en el compartimiento ROM BAY del lado izquierdo de la máquina.

PRIMEROS PROGRAMAS

Intentar escribir el contenido del recuadro siguiente. Los ordenadores son bastante divertidos en el caso de pequeños errores, por lo que conviene comprobar lo que se escribe antes de terminar. Recordemos que hay que pulsar la tecla marcada 'enter' al fin de cada línea. No olvidar los números que empiezan las líneas porque son también importantes. Sin embargo, no hay que preocuparse por los espacios en blanco que aparecen después de los números. El Enterprise pone automáticamente los espacios, para que los programas aparezcan más claros. Obsérvese que los ordenadores usan un símbolo especial para el cero (Ø), para distinguirlo de la letra mayúscula O.

```
100    GRAPHICS
110    PLOT 640, 360,
120    !
130    DO
140        FOR RADIUS=250 TO 1 STEP-16
150            SET INK RND(3)+1
160            PLOT ELLIPSE RADIUS, RADIUS,
170            PLOT PAINT
180        NEXT RADIUS
190        PING
200    LOOP
210    !
220    END
```

CORRECCION DE ERRORES

Si se realiza un error de mecanografía, es bastante fácil corregirlo. Si estamos en la misma línea del error, al pulsar la tecla marcada 'erase' se desplazará a la izquierda el cursor rojo que parpadea, retirando letras, etc., a medida que se mueve. Si tenemos que volver a una línea anterior, utilicemos la palanca de mando para colocar el cursor al final de esa línea, borrando después hacia atrás hasta que se llegue al error. Ahora podemos escribir la corrección, acabar la línea pulsando 'enter'y desplazar de nuevo el cursor (usando la palanca de mando) al fondo de la pantalla, o a cualquier otro lugar que queramos. Recuérdese que siempre que se pulsa una tecla, el lugar en donde aparecerá la letra o el número es el lugar en donde esté situado el cursor en ese momento.

Más adelante, en el capítulo sobre 'Edición de Programas' se expondrán métodos más versátiles para efectuar cambios.

PUESTA EN MARCHA DEL PROGRAMA

Una vez introducido el programa, escribir la palabra RUN y pulsar de nuevo 'enter'. Como variante, puede pulsarse también la tecla marcada 'function 1', encima de las teclas de número.

Si se utiliza esta tecla, no será necesario pulsar 'enter', y es más rápido que escribir RUN. (En la página 43 se dan más datos sobre las teclas de 'función'). A continuación observar durante un momento. Si se ha hecho un error, el ordenador imprimirá 'Not understood' (no se entiende) o un mensaje similar en la pantalla de TV. No preocuparse en absoluto si sucede esto. Pulsar la tecla de función 5 y a continuación la tecla de función 2, para echar un nuevo vistazo al programa, y a continuación corregir el problema y probar de nuevo.

ESCRITURA DE COMANDOS

Los comandos que este manual sugiere que se escriban en el ordenador, aparecerán siempre en la página en letras mayúsculas. Esto se hace especialmente para ponerlas en relieve, pero también, en parte, porque el mismo ordenador frecuentemente visualiza los comandos del BASIC en mayúsculas. Sin embargo, se pueden escribir palabras como mayúscula RUN (y otros comandos del BASIC) en minúsculas; el ordenador nos entenderá perfectamente. No es necesario tomarse la molestia de pulsar la tecla 'shift' para pasar de minúsculas a mayúsculas.

Cualquier palabra que no entendamos podemos hallarla en el Glosario, páginas 209-221.

Para detener el programa, pulsar 'stop'. Observará la respuesta

```
STOP AT LINE ---      (--- is a number)
ok
_
```

(--- es un número)

¿QUE ES UN PROGRAMA?

Un programa es un conjunto de instrucciones que dicen al ordenador lo que debe hacer. No hay nada especial ni mágico en esto, pero un ordenador no puede hacer nada en absoluto sin un programa. Los programas son muy exactos y detallados, pero también lo es un bordado. Y como ocurre con cualquier otro pasatiempo complejo, la programación puede hacerse con seriedad o simplemente para divertirse.

Si se quiere, se puede reanudar el programa escribiendo CONTINUE y pulsando 'enter'. Aún mejor, pulsar la tecla marcada 'shift' y al mismo tiempo la tecla de función número 1 (que hemos utilizado ya para ejecutar el programa). El programa se reanudará a partir del punto en el que se detuvo.

Por otra parte, si nos aburrimos con esto, podemos añadirle algo. Escribir en la línea siguiente.

```
135  SET PALETTE RND (256),
     RND (256), RND (256), RND (256)
```

BORRAR LA PANTALLA

Una vez ejecutado este programa y detenido, nos queda ahora una imagen que llena la pantalla. Esta imagen no nos permite leer fácilmente lo que vamos escribiendo. Para limpiar la pantalla, escribir TEXT, pulsar 'enter' y la pantalla volverá de nuevo a la representación de todo el texto. Si queremos ahorrar tiempo (esfuerzo), podemos también pulsar la tecla de función 5.

LIST

Esta vez, cuando hemos terminado de escribir la nueva línea, escribimos LIST. La tecla de función 2 puede también hacer esto. Por ahora nos habremos habituado ya a pulsar 'enter' siempre que hemos terminado de escribir una instrucción o una línea del programa. 'Enter' es la tecla que ordena al ordenador que ejecute lo que hemos escrito. Normalmente, mientras no se pulse, no sucede nada. Sin embargo, no es necesario pulsar 'enter' al utilizar las teclas de función, como habremos comprobado ya.

LIST es una palabra que muestra en pantalla la totalidad de un programa (o la parte del mismo que cabe al mismo tiempo en la pantalla de TV). Ahora podemos ver que la nueva línea se ha unido a las anteriores. Todo el programa se encuentra ahora en orden numérico. Este es el orden en el que el ordenador ejecutará nuestras instrucciones cuando se inicie (RUN) el programa.

BORRADO DE UN PROGRAMA

Cuando nos hemos cansado de un programa, escribimos simplemente NEW, pulsamos 'enter', y el programa desaparecerá.

Usando este ordenador podremos preparar todo tipo de sonidos y muchos colores con gran facilidad, pero no está simplemente ahí para hacer ruidos y representar arco iris. El ordenador puede dibujar imágenes muy finas, tomar decisiones, realizar rápidamente cálculos complicados, clasificar cosas en cualquier orden que se quiera y repetir las cosas siempre que se quiera.

PROGRAMAS PARA PRACTICAR

Los programas de las páginas siguientes son simples ejemplos de las posibilidades del Enterprise.

¡Probémoslos todos!. No son más que unas pocas de las cosas que podemos hacer con esta máquina. En este libro encontraremos la manera de hacerlos todos personalmente ... y muchas más cosas. No olvidar el uso de TEXT para obtener una pantalla completa en la que poder escribir cuando se desea. Puede usarse también CLEAR SCREEN para dejar vacía la pantalla de TV cuando se llena. Estos comandos no retiran en realidad ninguna línea del programa de la memoria del ordenador; siempre que se quiera puede verse de nuevo el programa entero escribiendo LIST.

PROGRAMA "Túnel de fuego"

Este programa dibuja un túnel de varios colores con bolas de fuego que explotan.

```
  10    PROGRAM "Fire-tunnel"
  20    !
  30    !     This program draws a
  40    !     multi-coloured tunnel with
  50    !     exploding fireballs.
  60    !
 100    GRAPHICS HIRES 256
 110    LET X = 640: LET Y = 360
 120    FOR R = 1 TO 255
 130          SET INK R
 140          LET A = X-R-220: LET A1 = Y-R-50
 150          LET C = X + R + 220: LET C1 = Y + R + 50
 160          PLOT A, A1; A, C1; C, C1; C, A1; A, A1
 170          PRINT R
 180    NEXT
 190    FOR BALL = 1 TO 100
 200          CALL FIREBALL (256, X, Y)
 210    NEXT
 220    !
 230    END
 240    !
 250    !
10000   DEF FIREBALL (COLOURS, A, B)
10010          SET LINE MODE 3
10020          SET INK RND (COLOURS)
10030          FOR GO = 1 TO 2
10040                FOR AROUND = 1 TO 650 STEP 30
10050                  PLOT A, B, ELLIPSE AROUND,
                       AROUND
10060                NEXT
10070          NEXT
10080          SET LINE MODE 0
10090    END DEF
```

```
100  !  Este programa sirve para dibujar cuadros.
110  !
120  !  100-140 son líneas de comentario.
130  !  No es preciso escribirlas.
140  !  ------------------------------
150  CLEAR SCREEN
160  PRINT AT 5,11: "ESTE PROGRAMA"
170  PRINT AT 6,10: "DIBUJA CUADROS"
180  PRINT AT 8,1: "El programa le pedirá que escriba"
190  PRINT AT 9,1: "algunos números, en pares. El primer"
200  PRINT AT 10,1: "número de cada par no debe ser superior"
210  PRINT AT 11,1: "a 1279, y el segundo no ser"
220  PRINT AT 12,1: "superior a 719. Pulsar 'enter'"
230  PRINT AT 13,1: "después de escribir cada número"
240  FOR A=1 TO 5000
250  NEXT A
260  !   ------------------------------
270  ! Las líneas 240-250 hacen que el ordenador
280  ! espere durante unos 10 segundos.
290  !   ------------------------------
300  DO
310  CLEAR SCREEN
320  INPUT AT 5,5, PROMPT "Números de una esquina: ":X
330  INPUT AT 6,29, PROMPT " ":Y
340  INPUT AT 8,5, PROMPT "Números de la esquina opuesta:" :V
350  INPUT AT 9,34, PROMPT" ":W
360  PRINT AT 11,5 "Durante cuánto tiempo estará el cuadro"
370  PRINT AT 12,5 "representado en pantalla"
380  INPUT AT 14,5, PROMPT "Segundos" : TIME
390  !   ------------------------------
400  ! Las líneas 450-480 son las instrucciones
410  ! para dibujar el cuadro y mantenerlo en
420  ! pantalla todo el tiempo
430  ! que se desee.
440  !   ------------------------------
450  GRAPHICS
```

```
46Ø   PLOT X,Y;X,W;V,W,Y;X,Y
47Ø   FOR B=1 TO 5ØØ  * TIME
48Ø   NEXT B
49Ø   TEXT
5ØØ   PRINT AT 15,18: "MAS ?"
51Ø   DO
52Ø   INPUT AT 17,17, PROMPT " Y o N: ": ANS$
53Ø   LOOP WHILE ANS$<>"Y" AND ANS$<> "N"
54Ø   LOOP WHILE ANS$ = "Y"
55Ø   !
56Ø   END ! Este es el fin del programa.
```

```
100  !  Este programa clasifica 10 números
106  !  en orden numérico
108  !  ----------------------------
110  NUMERIC ARRAY(1 TO 10)
120  NUMERIC VAR,NUM,BIG
130  CLEAR SCREEN
140  !
150  PRINT AT 10,10:"CLASIFICACION DE LOS NUMEROS"
160  FOR N=1 TO 10
170      PRINT AT 14,10: "ESCRIBIR EL NUMERO";N;
180      INPUT PROMPT ":":ARRAY(N)
190      PRINT AT 14,25: "
200  NEXT N
210  CLEAR SCREEN
220  PRINT AT 20,20: "CLASIFICANDO...."
240  LET FIN=10
250  FOR X=1 TO 10
255      LET BIG=0
260      FOR Y=1 TO FIN
270          IF ARRAY(Y)  BIG THEN LET BIG=ARRAY(Y)
280          IF ARRAY(Y)=BIG THEN LET NUM=Y
290      NEXT Y
300      LET VAR=ARRAY(FIN)
310      LET ARRAY(FIN)=BIG
320      LET ARRAY(NUM)=VAR
340      LET FIN=FIN-1
350  NEXT X
360  CLEAR SCREEN
365  FOR X=1 TO 10
370      PRINT ARRAY(X)
380  NEXT X
390  END
```

```
100  !  Este programa nos dá el
110  !  área/longitud de círculos
120  !  -------------------------------
130  LET A$=" del círculo es"
140  LET B$="Escribir el radio del círculo"
150  NUMERIC RADIUS,AREA,CIRCUM
160  DO
170     CLEAR SCREEN
180     PRINT AT 10,10:"1) AREA"
190     PRINT AT 11,10:"2) CIRCUNFERENCIA"
200     PRINT AT 12,10:"3) ABANDONAR"
210     DO
220        PRINT AT 15,10: "Escriba el número"
230        INPUT AT 16,10, PROMPT"elegido":NUM
240     LOOP WHILE NUM<1 OR NUM>3 OR NUM<>INT(NUM)
250     CLEAR SCREEN
260     IF NUM=1 THEN
270        INPUT AT 10,1, PROMPT B$:RADIUS
280        LET AREA=PI* RADIUS^2
290        PRINT AT 15,5:"El área";A$;
300        PRINT AT 16,4: AREA
310        FOR X=1 TO 5000
320        NEXT X
330     ELSE IF NUM=2 THEN
340        INPUT AT 10,1, PROMPT B$:RADIUS
350        LET CIRCUM=2*PI*RADIUS
360        PRINT AT 15,1: "La longitud de la circunferencia";A$
370        PRINT AT 16,1:STR$(CIRCUM)
380        FOR X=1 TO 5000
390        NEXT X
400     END IF
410  LOOP WHILE NUM<>3
420  END
```

MODIFICACION DE PROGRAMAS

Probemos a modificar los programas. No es buena idea cambiar la escritura de las instrucciones de los programas, porque el ordenador no nos entenderá si lo hacemos. Pero cuando aparecen palabras entre apóstrofes podemos cambiarlas sin complicar el programa. También pueden cambiarse los números. Al hacerlo, podremos descubrir lo que está haciendo el programa: el cambio de un número afectará frecuentemente al número de veces que se hace algo, o a la posición de los caracteres en la pantalla. Algunos números de los programas son códigos: es decir, que representan alguna otra cosa; por ejemplo, un color. En este manual aprenderemos todo lo que necesitamos saber sobre este tema (por ejemplo, en la página 95 se habla de los colores y en la 1Ø4 de otros códigos).

Recordemos siempre que cualquier cosa que se escriba no hará ningún daño al ordenador. Lo peor que puede suceder es escribir algo que el ordenador no entienda. Por ejemplo, en el programa de clasificación de números (página 11), puede haberse escrito:

```
25Ø  FOR X = 1 TOO 1Ø
```

En ese caso, después de ejecutar la primera parte del programa el ordenador se detendría y en pantalla aparecería:

```
*** Not understood.
25Ø FOR X = 1 TOO 1Ø
```

(*** No comprendido)

Los mensajes de error se explican con detalle en las páginas 2Ø4-2Ø8.

PARADA Y REANUDACION

Si nos hemos quedado totalmente bloqueados y queremos empezar de nuevo, pulsamos el botón 'reset' en la parte posterior de la máquina. Con esto volveremos a donde nos encontrábamos al encender la máquina, excepto que el ordenador recordará cualquier programa que hayamos escrito. Sin embargo, es mejor conservar esto para los casos de emergencia. Normalmente se utilizan las teclas 'stop' o 'hold' para suspender un programa que se está ejecutando. Cuando se utiliza 'reset', habrá que escribir de nuevo RUN para que el programa vuelva a ponerse en marcha.

La tecla 'hold' se utiliza simplemente para mantener en suspenso el programa y congelar la acción en cualquier punto al que se haya llegado. Es muy cómoda siqueremos contemplar una imágen de movimiento en una pantalla gráfica, o también si se necesita simplemente descansar un poco para tomar un café en medio de un juego difícil

de los invasores del espacio. Para seguir con el programa, simplemente pulsamos de nuevo la misma tecla. Obsérvese que 'hold' es muy útil si se está listando un programa largo que desaparece de la pantalla, y queremos suspender el movimiento a fin de examinar una determinada línea.

La tecla 'stop' se utiliza si se está ejecutando un programa y se desea suspenderlo a fin de modificarlo (o borrarlo) o contemplar un listado y examinarlo antes de escribir CONTINUE para que se reanude el programa.

EN CUANTO A LA PROGRAMACION

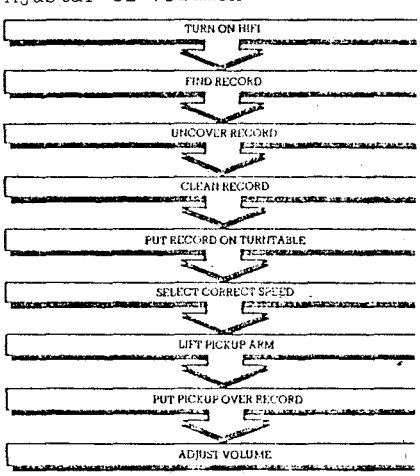
Nos hemos divertido ya un poco con la máquina y, hemos empezado a conocernos. Es probable que no se haya captado bien cómo trabajan los programas, por lo que en este punto es donde empezamos a añadir algunos conocimientos a la distracción.

Hemos aprendido ya brevemente qué es un programa, pero para escribirlo hay que hacer algo más que simplemente dar instrucciones. Podemos pensar que un programa es la manera de resolver un problema usando el ordenador. El ordenador es una herramienta que puede ampliar y dar mayor rapidez a la potencia de nuestro cerebro, o algo agradable de poseer con lo que podemos jugar o utilizarlo para inventar algo.

Todos los ordenadores entienden las instrucciones, normalmente en forma de palabras o listas de números. Nosotros utilizaremos palabras para comunicarnos con el Enterprise. Cada palabra es una pequeña instrucción; podemos reunirlas como en un rompecabezas o en una pequeña historia a fin de formar instrucciones mayores y, llegado el caso, tareas completas. De esto es de lo que trata fundamentalmente la programación; dar al ordenador instrucciones en el orden en que queremos que sucedan.

El esquema siguiente muestra (utilizando un ejemplo diario) la manera de subdividir una tarea en varias actividades más pequeñas a fin de formar un programa.

Encender el tocadiscos.
Encontrar el disco.
Sacarlo de la funda.
Limpiarlo.
Poner el disco en el plato.
Seleccionar la velocidad correcta.
Levantar el brazo.
Poner el brazo sobre el disco.
Ajustar el volúmen.



LENGUAJE DEL ORDENADOR

Ahora que sabemos lo que es un programa, demos un paso más. Al igual que hay muchas y diferentes maneras de hablar a otra persona, hay también muchas formas de programar un ordenador. Y de la misma manera que hay lenguajes humanos hay también lenguajes de ordenador.

Los lenguajes se preparan teniendo presente unos determinados límites y tipos de tarea. Algunos sirven especialmente para programas en que intervienen listas muy largas de cosas, otros son particularmente buenos para efectuar dibujos con el ordenador. Otros, por su parte, sirven para enseñar a la gente a programar.

BASIC

El lenguaje que se aprende en este manual se denomina BASIC. Utiliza palabras con una escritura y un significado similares a los de las palabras inglesas. Por tanto, es muy fácil de aprender y entender, aunque no se tenga experiencia con ordenadores. Todos los programas de este ordenador están en BASIC, que es el lenguaje que entiende el ordenador mientras esté metido el cartucho IS-BASIC. No olvidemos que, a partir de este momento, todas las instrucciones y toda la información de este manual se refieren siempre al BASIC. Algunos otros lenguajes son totalmente diferentes tanto en su filosofía como en su planteamiento, y por lo general suelen aparecer totalmente distintos del BASIC.

Veámos esto:

```
1Ø PRINT  "Hola"
```

Sabremos ya que es una línea de programa: una pequeña tarea que podemos conseguir que forme parte de otra mayor.

Igual que, en un cuento, cada frase nos enseña algo, cada línea de un programa indica al ordenador que haga algo. O si queremos decir al ordenador que realice una tarea, necesitaremos varias líneas para conseguirlo.

HACER LAS COSAS ORDENADAMENTE

El número que hay al principio de una línea de programa BASIC representa una diferencia importante.

Si escribimos PRINT "Hello!". La pantalla que aparece al pulsar 'enter' nos dirá lo que sucede. El ordenador hace simplemente lo que se le dice: coloca 'Hello!' en la pantalla, justamente debajo de las palabras que acabamos de escribir. Cuando se han introducido programas antes, el ordenador esperaba a que se le diera la orden antes de ejecutar el programa. Es el número de la línea lo que representa esta diferencia. Sin este número, el ordenador ejecuta las instrucciones en cuanto se pulsa 'enter'. Si se añade un número de línea, se debe escribir RUN (o pulsar la tecla de función 1) para que trabaje el ordenador.

Ahora sabemos ya que todos los programas del BASIC están divididos en líneas, y que cada línea debe tener un número.

Si introducimos una línea con el mismo número introducido antes, la línea anterior quedará automáticamente borrada de la memoria del ordenador. Si queremos librarnos de una línea, simplemente escribimos su número seguido por 'enter'. La línea desaparecerá porque realmente hemos introducido una línea vacía. Podemos eliminar también un grupo completo de líneas; el comando DELETE 1Ø TO 1ØØ retira todas las líneas de la 1Ø a la 1ØØ ambas inclusive.

El orden en el que se colocan las líneas en el ordenador no importa. Hemos visto ya que la máquina las clasifica por orden numérico. Es el orden numérico lo que importa.

Los números de las líneas señalan la manera como el ordenador decide ejecutar las instrucciones. Empieza por el número más bajo y va subiendo - a menos que con el programa le digamos que actúe de otro modo - hasta que llega al final.

La frase "a menos que con el programa le digamos que actúe de otro modo" es muy importante como veremos más adelante. Hay varias maneras de montar un programa. En algunas de ellas el ordenador no sigue los números de las líneas por orden, porque algunas palabras BASIC le indican que utilice otras líneas.

Esto significa que debemos conocer exactamente lo que queremos que haga el ordenador y en qué orden: que no es más que estar seguros de que sabemos lo que queremos antes de empezar a escribir un programa.

MODO INMEDIATO Y PALABRAS CLAVE

MODO INMEDIATO

Si se escriben comandos sin darles un número de línea, esto significa modo inmediato. De este modo puede utilizarse el ordenador como una simple calculadora. He aquí como.

En el teclado aparecen estos símbolos: +,*,=,- y /. De éstos, los símbolos * y / pueden sernos desconocidos. Representan 'multiplicar' (* en lugar de x) y 'dividir' (/en lugar de ÷). Si se escribe

```
PRINT 2+2
```

se recibirá la respuesta, 4, tan pronto como se pulsa 'enter'. Con todos los operadores matemáticos, ya conocidos, podemos realizar sumas en el ordenador de este modo. También podemos sacar raíces cuadradas. Probemos

```
PRINT SQR(100)
```

y la respuesta será la raíz cuadrada de 1ØØ. SQR es una palabra especial que el ordenador entiende como 'la raiz cuadrada del número entre paréntesis'. En BASIC hay algunas palabras de este tipo. En la Sección de Referencia, bajo el título 'Funciones y Variables incorporadas' se da una lista de ellas. A medida que se vayan utilizando en el texto, se explicarán otras nuevas.

El modo inmediato es muy útil cuando se necesita encontrar los resultados de cálculos para utilizar en programas. El uso del modo inmediato no afecta a ninguna línea del programa que se haya podido escribir ya, y no hay que hacer nada especial para empezar a utilizarla. Simplemente no poner el número de línea.

Muchos comandos del BASIC trabajan tanto en modo inmediato como dentro del programa. Veremos lo que significa esto experimentando. Busquemos en el manual algunas palabras del BASIC y entonces, usando los ejemplos incluídos en cada parte, probemos las palabras en modo inmediato. La lista de referencia de palabras clave (página 151) nos indica también qué mandos pueden utilizarse en modo inmediato.

PALABRAS CLAVE

Ahora que sabemos algo sobre la programación en BASIC y el modo inmediato, expliquemos un poco mejor las palabras clave. PRINT es una de ellas y tiene un significado especial en BASIC. Las palabras clave son las instrucciones a las que el manual ha hecho referencia antes. Hay muchas de ellas, y cada una indica al ordenador que haga una pequeña cosa (o tal vez

más de una). PRINT, por ejemplo, indica al ordenador que represente algo en la pantalla.

Para que el ordenador ejecute trabajos complicados, utilizamos muchas palabras clave y otros elementos de información reunidos en un programa. Es algo así como contar al ordenador una historia o enseñar a un niño el alfabeto: debe ser detallado, exacto y en el orden correcto.

SERIES

Como hemos visto ya, se puede conseguir que el ordenador imprima un mensaje (!Hola!) en la pantalla escribiendo el mensaje entre comillas (" "). El ordenador considera esto como una serie (en la página 52 se dan más datos sobre la serie). Esto puede parecer ahora complicado, por lo que hay dos puntos que pueden ayudarnos a entenderlo.

En primer lugar, 'string' (serie) no es más que una palabra que indica un determinado tipo de información que manejará el ordenador. En segundo lugar, dado que las series aparecen en un programa entre comillas, podemos considerarlas algo así como las citas entre comillas de una frase. En la sentencia PRINT "!Hola!", el ordenador cita literalmente lo que habíamos puesto entre comillas.

CARACTERES

La palabra 'caracteres' ha aparecido ya alguna que otra vez. Un caracter es una letra, un número o cualquier otro símbolo que se proporciona al ordenador para fines de visualización/ comunicación. Todos los caracteres que se proporcionan al ordenador se denominan verbalmente el conjunto de caracteres.

VARIABLES

Ahora que sabemos ya algo sobre las líneas de los programas, la modalidad inmediata y las palabras clave, añadamos algo más al cuadro.

Imaginemos dos cuadros, llamados X e Y. Dentro de cada uno podemos poner un número. Más tarde, este número puede ser retirado y sustituído por otro nuevo.

Vamos a decir al ordenador que cada vez que ejecutemos (RUN) el programa siguiente y pongamos un par de números en los dos cuadros, queremos que el ordenador sume el contenido de X e Y.

```
10  !
20  !       60 and 70 ask you to type in
25  !       numbers. They wait for your
30  !       answer. Press 'enter' when you
40  !       have typed each number.
50  !
60  INPUT PROMPT "Type in number X: ":X
70  INPUT PROMPT "Type in number Y: ":Y
80  !
90  !       Line 120 adds up numbers X and
95  !       Y and puts the sum
100 !       into a variable called Z.
110 !
120 LET Z=X+Y
130 !
140 !       Line 160 displays the answer.
150 !
160 PRINT X;"+";Y;"=";Z
170 END
180 !
190 !       Line 170 tells you and the
200 !       computer where the end of the
210 !       program is.
220 !
```

60 y 70 nos pide que escribamos números. Quedan a la espera de la respuesta. Pulsar 'enter' después de escribir cada número.

"Escribir el número X"
"Escribir el número Y"

La línea 120 suma los números X e Y y pone la suma en una variable llamada Z.

La línea 160 nos muestra la respuesta.

La línea 170 nos indica a nosotros y al ordenador en donde está el fin del programa.

Ni siquiera es necesario ejecutar (RUN) esto, para ver lo que está sucediendo. Es algo que habremos aprendido en la escuela (o, si somos ya padre, en el libro de matemáticas de nuestro hijo). ¿Comprende cómo debe decirle al ordenador todo lo que debe hacer?.

COMENTARIO EN PROGRAMAS

Las líneas de programa que empiezan con el signo de admiración son para beneficio del operador: el ordenador las recuerda pero no actúa respecto a ellas. Son comentarios que pueden utilizarse para indicar al operador lo que está haciendo cada parte

de un programa. El signo de admiración puede utilizarse también para apartar el resto de una línea de programa para comentarios después de una o dos instrucciones en la misma línea. También podría utilizarse la palabra REM (REMARK, Observación), en lugar del signo de admiración, pero esta palabra debe colocarse inmediatamente después del número de línea. Es mejor disponer los comentarios como el ejemplo anterior, a fin de que se puedan recibir rápidamente.

Observemos que el uso de los comentarios facilita la comprensión del programa. No son esenciales, pero ayuda a que el hombre y el ordenador entiendan mejor un programa.

INPUT PROMPT

INPUT PROMPT (solicitud de entrada) (ver el programa de la página 20) es el conjunto de palabras clave que pide al ordenador que nos haga una pregunta y espere acto seguido a que introduzcamos la respuesta.

Si utilizamos INPUT solamente, sin añadir PROMPT, seguido por algunas palabras entre comillas (que están ahí para beneficio del operador), el ordenador dirigirá la pregunta simplemente mostrando en pantalla el signo de interrogación y el cursor rojo. Las palabras después de PROMPT pueden ser cualquiera, y sirven únicamente para recordarnos lo que se supone que debemos escribir. Deben introducirse siempre entre comillas (aunque las comillas no aparezcan en la pantalla). Otra manera de pasar un 'input prompt' es ésta:

```
10 PRINT "Type in number X ";          "Escribir el número X"
20 INPUT X
30 PRINT "Type in number Y ";
40 INPUT Y                             "Escribir el número Y"
```

Como se puede ver, este procedimiento es algo más complicado que el de INPUT PROMPT, pero el resultado es el mismo. Ese breve programa hace exactamente lo mismo que las líneas 6Ø y 7Ø del anterior, salvo que éste permite al ordenador imprimir su propia 'solicitud de entrada', que es el signo de interrogación. La sentencia INPUT PROMPT evita imprimir signo de interrogación.

OTRA VEZ LAS VARIABLES

Las variables representan valores numéricos, cuyo contenido puede cambiar. Para X e Y podríamos escribir cualquier número: tanto Ø.ØØØØ4 como 4ØØØØØ. Si X representa el número 4ØØØØØ diríamos que el valor de X era 400.000.

Con frecuencia necesitaremos el uso de variables porque no sabemos de antemano qué valor tendrá un número.

Regresemos a las primeras páginas del manual. Todos los programas probados tenían variables, a veces porque el ordenador sacaba sus propios números de otros y otras veces porque decidimos el valor del número mientras se estaba ejecutando el programa, escribiéndolos.

En ocasiones, es conveniente utilizar nombres de variables para números muy largos que se utilizarán varias veces. Esto significa que no es preciso escribirlos una y otra vez siempre que se desean utilizar en un programa.

He aquí otro uso de las variables

```
10 LET A=0 !          A begins as 0.
20 DO !               DO/LOOP begins.
30     LET A=A+1 !    Add 1 to A.
40     PRINT A, !     Display A's value.
50 LOOP UNTIL A=20 !  Go back, repeat loop.
60 !                  Loop ends when A=20.
70 END
```

```
1Ø  A empieza como cero.
2Ø  Empieza DO/LOOP.
3Ø  Sumar 1 a A.
4Ø  Visualizar el valor de A.
5Ø  Volver, repetir el bucle.
6Ø  El bucle termina cuando A=2Ø.
```

Se trata de un contador, que utiliza algo llamado DO/LOOP. Cada vez que el Enterprise recorre el bucle, aumenta en 1 el número de la variable A. Acto seguido, al final del bucle, comprueba si el valor de A no es 2Ø. Cuando A es 2Ø, termina el programa. La coma que hay junto a 'A' en la línea 4Ø hace que en la pantalla aparezcan más claros los números, al salir espaciados.

Los DO/LOOP no son la única manera de pedir al ordenador que repita. En la página 59 y en la Sección sobre Referencia aprenderemos algo más sobre ésto.

NOMBRANDO VARIABLES ... X,Y y Z no son más que tres de los muchos nombres que podemos dar a una variable. No es necesario que sean una sola letra, podemos utilizar más de treinta si queremos.

Por tanto, podemos dar a nuestras variables nombres apropiados: NAME$ para el nombre de alguien (se trata de una variable de serie y se explica en la página 53), SUM para el resultado de la suma de dos números.

Obsérvese que da igual que llamemos a una variable PRICE, Price, price, o pRice. El ordenador no distingue entre mayúsculas y minúsculas en los nombres de variables ni en las palabras clave.

Cuando escribimos programas propios veremos rápidamente que los nombres de variables pueden ayudarnos a leer más tarde el programa. Esto es muy importante si es necesario encontrar errores, introducir cambios o sacar parte de un programa para utilizarlo en otro: cosas que en algún momento dado pueden ser necesarias.

Por cierto, se pueden utilizar dos variables con nombres muy similares, por ejemplo:

```
VAT_SUBTOTAL_JUNE_ACCOUNT_NO1
and
VAT_SUBTOTAL_JUNE_ACCOUNT_NO2
```

y el Enterprise podrá incluso descubrir inmediatamente la diferencia. El problema es: ¿la podemos descubrir nosotros?. No es muy inteligente utilizar todo el tiempo nombres enormes de variables como estos. A veces pueden ser convenientes, pero por lo general de ocho a diez caracteres suelen ser suficientes para descubrir la diferencia y saber rápidamente qué variables se utilizan para qué números.

Por tanto, las variables son una manera de calcular con números y cuyo valor real desconocemos. El nombre que damos a cada una nos indica a qué se refiere ese número. El ordenador descubrirá la diferencia entre nombres de variables de hasta 31 caracteres.

Un 'caracter' significa aquí: Una letra o un número (mejor un dígito para ser absolutamente correcto). El ordenador no nos entiende si ponemos espacios en los nombres de variables, ni entiende los operadores (+, = etc.) o los signos de puntuación (excepto el punto y el signo de subrayado, '_', que es un buen sustituto de un espacio) Debemos empezar siempre los nombres de variables por una letra, no por un número ni un signo de puntuación. He aquí algunas variables 'legales', es decir, las que aceptará el ordenador:

```
number      A2$       TOTAL$      Hello1
SUB_TOTAL_3           Name        A1234
```

Observemos que se puede utilizar el signo del dólar. Este tiene un significado especial, ya que deja aparte esa variable para utilizarla como variable de serie. Las series se han explicado ya brevemente en la página 19 y se hablará de ellas con más detalle en las páginas 52-58.

He aquí algo que el ordenador no entiende

my variable	2A	!!!A$
<>3NOW	3*THIS NUMBER	

DECLARACION DE LAS VARIABLES

En elprograma contador, la primera línea decía LET A = Ø. Probemos de nuevo el programa sin ella. No funciona.

Sin embargo, si se utiliza una sentencia INPUT para indicar al ordenador que nos deje escribir el valor de una variable, (como en el programa que sumaba 2 números libremente elegidos), no necesitamos declarar esta variable usando LET.

USO DE LAS VARIABLES

Probamos a escribir LET A = 3 en vez de LET A = Ø al comienzo de ese programa. Con esto aprenderemos algo más sobre cómo funcionan las variables.

El ordenador espera que la variable que se está cambiando o está siendo declarada vaya inmediatamente después de LET. Por tanto, si la variable S=Ø y la variable G=1Ø, podemos escribir LET S=G para hacer que S sea 1Ø. G seguiría conteniendo 1Ø, pero el ordenador supondría que tenía el mismo tamaño que la variable S.

LET presenta una variable al ordenador; decimos: 'ordenador, he aquí una variable llamada FIRSTNUM, que por el momento es igual a Ø', o cualquier número que queramos darle. No siempre es necesario usar LET, pero normalmente es mejor hacerlo. Con esto se facilita el poder encontrarlas ya desde el principio .

Si se quiere, se puede utilizar una palabra del BASIC como nombre de variable. Pero si se hace, es necesario utilizar LET para decir al ordenador que busque una variable con el mismo nombre que una de las palabras que él entiende. Hay algunas palabras del BASIC que no pueden utilizarse como nombres de variables: con un poco de práctica descubriremos muy pronto cuáles son. (En la página 73, en el capítulo sobre almacenamiento de información, se expone un sistema más versátil para declaración de variables).

"Overpage" es otro programa que ilustra todos los principios que se acaban de exponer. Esta vez cada línea se explica a la luz de nuestros nuevos conocimientos sobre estas variables.

```
 10    LET A$=" The sum of the two numbers
       is: "
 20    !
 30    !          10 declares string variable A$.
 40    !
 50    LET SUM=0
 60    !
 70    !          SUM is the variable which will
 75    !          contain the result of the
 80    !          addition below. It begins as 0.
 90    !
100    INPUT PROMPT "Please type in your
       first number: ": FIRSTNUM
110    INPUT PROMPT "Type in your second
       number: ": SECNUM
120    !
130    !          100 and 110 ask you to type in
135    !          numbers to be added. You do not
140    !          need to use LET, because the two
150    !          new variables are being 'input'.
160    !
170    LET SUM=FIRSTNUM+SECNUM
180    !
190    !          SUM (which was 0) now becomes
195    !          the result of the addition of
200    !          FIRSTNUM and SECNUM.
210    !
220    PRINT A$;SUM
230    !
240    !          Line 220 tells the computer to
245    !          display the sentence from line
250    !          10 and the value of SUM all on
260    !          the same screen line.
290    !
300    END
```

1Ø "La suma de dos números es:"

3Ø 1Ø declara la variable serie A$.

7Ø 75 8Ø SUM es la variable que contendrá el resultado de la suma siguiente. Empieza como Ø.

1ØØ "Por favor escriba su primer número"

11Ø "Escriba su segundo número"

13Ø 135 14Ø 15Ø 1ØØ y 11Ø le pide que escriba los números que hay que sumar. No es necesario utilizar LET, porque las dos nuevas variables están siendo 'input'.

19Ø 195 2ØØ SUM (que era Ø) se convierte ahora en el resultado de la suma de FIRSTNUM y SECNUM.

24Ø 245 25Ø 26Ø La línea 22Ø ordena al ordenador que muestre en pantalla la sentencia a partir de la línea 1Ø y el valor de SUM, todo ello en la misma línea de la pantalla.

Esto es una expresión: 4+2*3-5.

Esto son operadores: ^,+,-,*,/,=,<,>,<>, > = y < =

Todos estos operadores pueden usarse en el Enterprise. Algunos de ellos no necesitan explicación, pero otros tal vez sí. Los símbolos +,-,*,/,=, son claros; el ordenador utiliza * (asterisco) en lugar de x, y / (barra) en lugar de ÷ .

He aquí el resto, para asegurarnos:

^ significa 'a la potencia de' o 'involución'. 2^3 es 2 elevado al cubo.
< significa 'menor que', por ejemplo, 2< 3.
> significa 'mayor que', por ejemplo, 3 >2.
<> significa 'distinto que'.
< = significa 'menor o igual que'.
> = significa 'mayor o igual que'.

Los cinco últimos operadores se denominan 'operadores relacionales'. Su uso principal es para el manejo de variables, por ejemplo:

```
10    INPUT PROMPT "Please type in a
      number: ": A
20    !-------------------------------
30    !     Line 10 asks for a number which is
35    !     to be evaluated.
40    !-------------------------------
50    IF A < 0 THEN PRINT "This number is
      negative."
55    IF A = 0 THEN PRINT "This is zero!"
60    !-------------------------------
70    !     50 makes a simple decision. If
75    !     A is a negative number (e.g. -9),
80    !     say so. IF is the crucial point.
90    !     If number A does not fit
100   !     that line, the computer looks
110   !     for a line which A does fit.
120   !     A < 0 in line 50 literally
130   !     means 'A is less than 0'.
140   !-------------------------------
150   IF A > 0 AND A < 50 THEN PRINT "This
      number is more than 0 and less than 50."
160   !-------------------------------
165   !     150 is another 'IF/THEN'. It's
```

10 "Escriba un número"

30 35 La línea 10 pide el número que va a evaluar".

50 "Este número es negativo"

55 "!Es cero!"

70 50 toma una simple decisión. Si A es un número negativo

75 (por ejemplo, menos 9), indicarlo. IF es el punto crucial.

80 Si el número A no entra en esa línea, el ordenador busca

90 una línea en la que entre A. A < 0 en la línea 50

100 significa literalmente 'A es menor que 0'.

120

130

150 "Este número es mayor que 0 y menor que 50".

165 150 es otro 'IF/THEN'. Es

```
170     !          like English--IF you don't want
180     !          to come THEN go home. Now the
190     !          computer looks to see if A is more
195     !          than Ø but less than 5Ø. If not, it
200     !          looks at the line below.
210     !          ------------------------------
220     IF A > 5Ø AND A < 1ØØ THEN PRINT "This
        number is more than 5Ø and less than 1ØØ."
230     !          ------------------------------
240     !          Line 3ØØ is the last decision line.
245     !          It looks to see if A is more
250     !          than 1ØØ.
290     !          ------------------------------
300     IF A > 1ØØ THEN PRINT "This number is
        bigger than 1ØØ."
310     END
```

17Ø 18Ø 19Ø 195 2ØØ
Como en inglés: Si(IF) no quiere ir, entonces (THEN) váyase a casa. Ahora el ordenador parece que busca si A es mayor que Ø pero menor que 5Ø. Si no lo es, busca en la línea siguiente.

22Ø "Este número es mayor que 5Ø y menor que 1ØØ".

24Ø 245 25Ø La línea 3ØØ es la última línea de decisión. Parece que busca si A es mayor que 1ØØ.

3ØØ "Este número es mayor que 1ØØ".

el programa decidirá en qué categoría entra su número, de acuerdo con las instrucciones que reciba. Esta es una de las maneras que tiene el ordenador para tomar decisiones. Hay algunos otros sistemas para hacer esto, que se explican en la página 64.

Los operadores relacionales son el modo de comparar números. No cambia los números en absoluto. Los operadores a que hemos hecho referencia antes cambian todos los números de alguna manera.

PRIORIDAD DE OPERACION.

Veamos de nuevo la expresión: 4 + 2*3 - 5.

¿Cuál debería ser el resultado? ¿Trece, porque se opera con los tres primeros dígitos, restando 5?. Pues bien, es cinco, y he aquí por qué. Primero multiplica (y en su caso divide), y luego suma y resta.

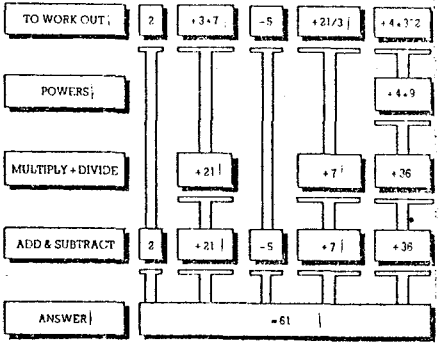
Las potencias se resuelven primero. Si la expresión hubiese tenido en ella 4^3, debería haberse calculado primero. A continuación viene la multiplicación y la división. 2*3 o 6/2 se calcularían antes de que el ordenador realizara cualquier suma o resta.

La multiplicación y la división tiene la misma prioridad y si, por ejemplo, una expresión contuviera dos divisiones y una multiplicación, la máquina las calcularía por orden de la izquierda a la derecha.

Una vez hecho esto, el ordenador pasa al siguiente nivel de prioridad y empieza por la suma y la resta. Estas se calculan también de izquierda a derecha y, para entonces, se tendría el resultado: en este caso 5.

Pero estudiemos esta expresión más a fondo, y dividámosla en partes,

como haría el ordenador.



2+3*7+21/3+4*3^2 es nuestra expresión.

Primero hallamos las potencias.

3^2 = 9

Ahora aparecería así:

2+3*7-5+21/3+4*9

A continuación realizamos las multiplicaciones y divisiones: el segundo nivel.

3*7=21 21/3=7 4*9=36

En este momento tenemos ya:

2+21-5+7+36

Y si se calcula esto de izquierda a derecha, sería:

2+21=23 23-5=18 18+7=25 25+36=61

el resultado es 61.

Como puede verse, esto afecta a la manera en que se realizarán los cálculos. Podría esperarse un resultado cuando el ordenador da otro. Si se desea, se puede cambiar la prioridad y hacer que diferentes secciones de una expresión reciba un 'tratamiento expreso' por parte del ordenador. Todo lo que hay que hacer es colocar paréntesis en las partes evaluadas en primer lugar. Veamos la diferencia que esto representa en la expresión anterior:

(2+3)*7-5+21/3+4*3^2.

El ordenador trata (2+3) como una unidad independiente y por lo tanto la calcula primero. Así:

5*7-5+21/3+4*3^2.

A continuación calcula de manera normal, y saca la potencia 3^2=9

por lo tanto:

5*7-5+21/3+4*9

Acto seguido, la multiplicación y división: de izquierda a derecha:

5*7=35 21/3=7 4*9=36

dejando

35-5+7+36

hallando:

35-5=3Ø 3Ø+7=37 37+36=73

El resultado es ahora 73 y no 61 como antes. Por ello, es fácil ver que los números se pueden entremezclar de muchas maneras usando el ordenador. No se olvide esto: dentro de cada par de paréntesis, el ordenador tratará las expresiones usando el mismo sistema de prioridad. Por tanto, si tenemos (2+3*4) la multiplicación vendrá antes que la suma.

Probemos cualquiera de los ejemplos anteriores. La manera más simple es utilizar PRINT en modalidad inmediata: por ejemplo, escribir PRINT 5*7-5+21/3+4*9.

Por último, antes de continuar, se pueden utilizar números con puntos decimales (por ejemplo, Ø.23345 ó Ø.ØØ98) y números negativos (por ejemplo, -2) siempre que se desee.

Es posible que se considere que el tratamiento de estos números sea un poco complicado. Pero una vez habituados a usar los números como lo hace el ordenador, no nos daremos cuenta de ello. No olvidemos que, aunque nunca será necesario ser matemático para programar bien un ordenador, siempre será necesario usar la aritmética. Incluso los comandos gráficos necesitan números que han de ser hallados, aunque casi siempre son muy fáciles.

Un punto y coma indica al ordenador que todo lo que hay delante y detrás debe imprimirse en pantalla uno junto a otro. Un programa sin cuatro sentencias PRINT separadas pero sin punto y coma aparecería así:

```
Hola
soy
tu
ordenador
```

en lugar de:

```
Hola, soy tu ordenador.
```

Hay otras maneras de colocar las cosas en pantalla con una mejor disposición.

Probemos a utilizar una coma, como a continuación (esto modifica el programa del contador de la página 22):

```
10  LET A=0 !                  Declare A as 0.
20  DO !                       Begin DO/LOOP.
30  LET A=A+1 !                Add 1 to A.
40      PRINT A,
50      !
60      !   Print A. The comma tells the
65      !   computer to put A eight character
70      !   positions along from the last A.
80      !
90  LOOP UNTIL A=20 ! Loop again if A < 20.
100 END
```

```
Declarar A como Ø.
Empezar DO/LOOP.
Sumar 1 a A.

6Ø 65 7Ø  Imprimir A. La coma dice al ordenador
          que ponga A en ocho posiciones de caracteres
          a partir de la última A.
9Ø  Loop de nuevo si A  2Ø.
```

¿¿Ve como la coma lo dispone todo en columnas?. La pantalla la divide normalmente el ordenador en 4Ø 'posiciones de caracteres' horizontalmente y 24 verticalmente. La coma separa normalmente las distintas entradas en 8 posiciones de caracteres.

El uso de una coma para ajustar a formato un texto es algo así como usar el punto y coma pero añade un espacio entre una serie o número y el siguiente. El punto y coma simplemente imprime las cosas una junto a otra en una línea.

Probemos este corto programa:

```
10  PRINT "There was an old man from St. Bees"
20  PRINT
30  PRINT "Who was stung on the head by
    a wasp."
```

```
1Ø   "Había un viejo de St. Bees"

3Ø   "Que fue picado en la cabeza por una avispa".
```

```
40 PRINT
50 PRINT "When asked, 'Does it hurt?' "
60 PRINT
70 PRINT "He replied," "No, it doesn't "
80 PRINT
90 PRINT " 'It can do it again if it likes!' "
100 END
```

```
5Ø  "Al preguntársele '¿Le duele?'"
7Ø  "Contestó" "No, no duele"
9Ø  '!Puede hacerlo de nuevo si quiere!'
```

Las líneas 2Ø, 4Ø, 6Ø y 8Ø imprimirán todas una línea vacía en pantalla: como el retorno del carro en una máquina de escribir. La palabra PRINT en una línea significa por sí misma 'imprimir una línea vacía' o 'saltar una línea y utilizar en su lugar la siguiente'.

No se puede invertir las comas dobles invertidas en pantalla de la misma manera que otros caracteres (por ejemplo, PRINT " " "), porque esto confunde al ordenador. Por el contrario, ha de escribirse las dobles comas invertidas dos veces a fin de conseguir su impresión inmediata. Probemos la línea 7Ø anterior como sigue:

```
7Ø PRINT "He replied," "No, it doesn't"
```

```
7Ø  "Contestó" "No, no duele".
```

PRINT AT

Otra manera de preparar el texto es con PRINT AT. Ya sabemos lo que hace PRINT. Sabemos igualmente el significado de la palabra AT: indica al ordenador en qué lugar de la pantalla ha de poner algunos caracteres.

Probemos:

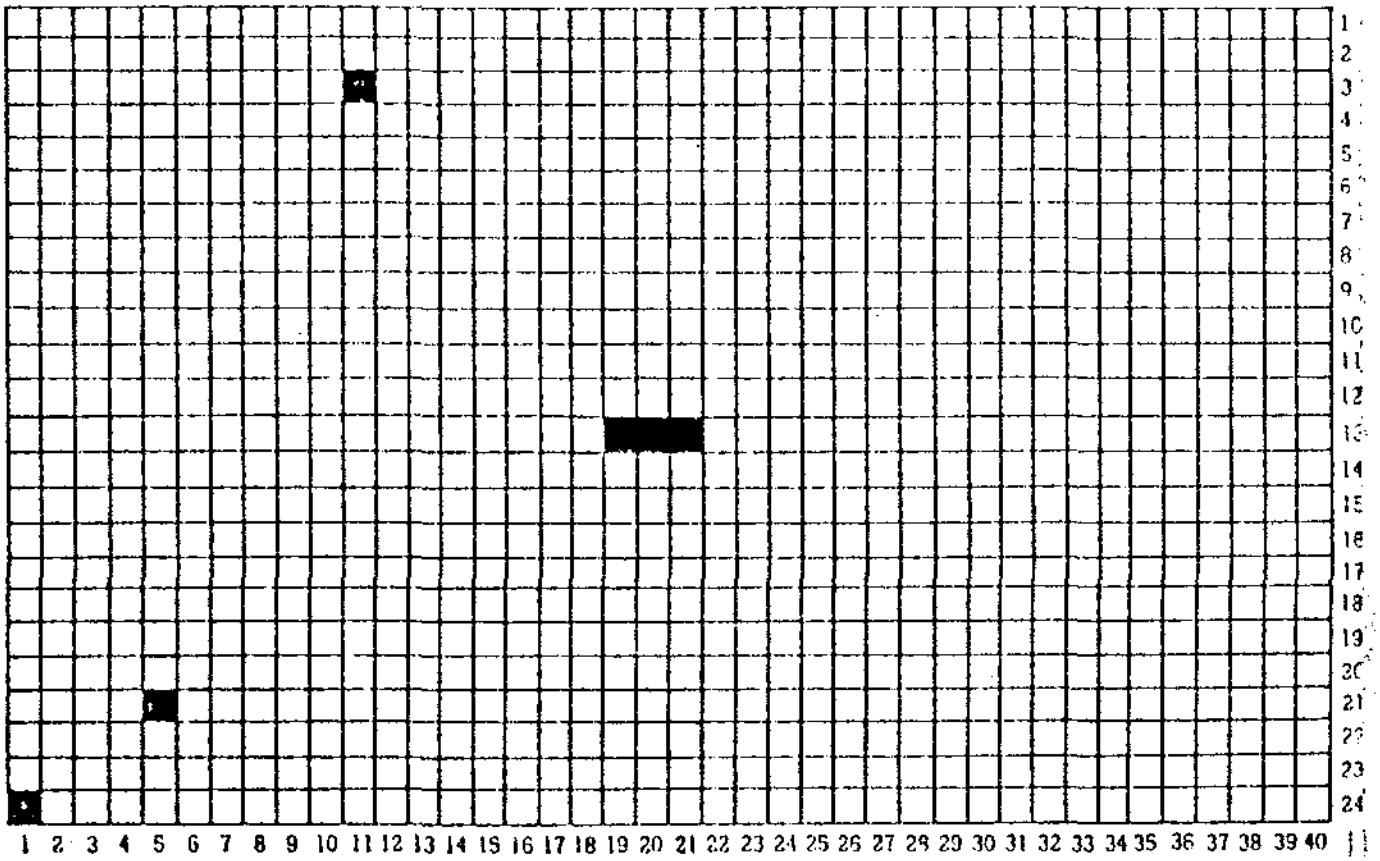
```
PRINT AT 2Ø, 2Ø: "I'M OVER HERE"
```

```
"ESTOY AQUI"
```

Imaginemos la pantalla dividida en posiciones: 4Ø horizontales y 24 verticalmente. Cualquiera de estas posiciones puede indicarse con dos números. La serie anterior se imprime 2Ø líneas abajo y 2Ø posiciones (o columnas) a la derecha.

PRINT AT 1Ø, 2Ø repite la serie de arriba en la pantalla.

El esquema siguiente indica cómo la pantalla está formada normalmente por 96Ø (24x4Ø)posiciones de caracteres; nos sirve para descubrir en donde se encuentra en pantalla cualquier posición. Se selecciona la posición PRINT AT indicando al ordenador hasta qué lugar de la pantalla debe bajar (número de la fila) y a qué distancia (número de columna) se desea colocar la serie. Una posición de carácter es un cuadrado 'imaginario' en pantalla, en el que se adaptará un único



carácter. Con los gráficos se utiliza un sistema similar pero diferente. Ver las páginas 90-91.

En la página 89 se explican algunos sistemas más avanzados para ajustar a formato un texto.

PANTALLA DE
8Ø COLUMNAS

El comando TEXT 8Ø nos permite escribir hasta 8Ø caracteres en cada línea, en lugar de sólo 4Ø; los caracteres serán ahora naturalmente más estrechos y tal vez no aparezcan claramente en la pantalla de TV. (El comando borra simultáneamente la pantalla).

TEXT 4Ø nos devuelve a la escritura en 4Ø columnas.

EDICION DE PROGRAMAS

Ya sabemos la manera de corregir los errores de mecanografía pulsando la palanca de mando y la tecla 'erase'. Ahora estudiaremos un sistema más complicado de modificar programas: insertando nuevas líneas, cambiando los números de líneas, modificando partes de ellas, etc. A veces es una tarea bastante complicada. Para facilitarla, el ordenador proporciona facilidades de 'proceso de textos' que se introducirán en líneas generales en este capítulo y que en el siguiente se explicarán con más detalle.

Estudiemos uno a uno los comandos de edición de programas.

RENUMERACION (RENUMBER)

Cuando se han escrito programas anteriormente en este mismo manual, se habrá comprobado que los números de línea empezaban frecuentemente en 1ØØ y seguían así: 11Ø, 12Ø, 13Ø ... En su lugar podríamos haber usado 1, 2, 3, 4.

La razón por la que no utilizamos normalmente 1, 2, 3, 4 ... para los números de línea, es simplemente porque queremos añadir posteriormente más líneas: al igual que cuando escribimos un cuento y cuando estamos en la mitad nos damos cuenta de que hemos dejado algo al principio. No se puede tener el número de línea 2, 5, pero sí escribir RENUMBER (si se encuentra uno en esa situación), y el ordenador cambiará los números de todas las líneas. Si se escribe RENUMBER STEP 1ØØ, las líneas aumentarán en 1ØØ al mismo tiempo. Si sólo se escribe RENUMBER, sin ningún número detrás, el ordenador aumentará en 1Ø los números de las líneas (a partir de 1ØØ). Esta versión del comando se obtiene también pulsando con 'shift' la tecla de función 3.

STEP

STEP es una palabra clave que aparece de vez en cuando en BASIC, junto con alguna otra (como RENUMBER en el caso anterior). STEP significa 'en pasos de ...'. STEP 1ØØ significa en pasos de 1ØØ: es decir, 1ØØ, 2ØØ, 3ØØ ...

AUTO

Otra palabrita muy útil hace que el ordenador coloque automáticamente las líneas del programa. Esta palabra es AUTO. Si se escribe, en la nueva línea aparecerá el número 1ØØ.

A continuación se puede escribir la línea del programa y pulsar 'enter'. Aparecerá entonces 11Ø, etc. Para detener esta numeración automática, basta con pulsar la tecla 'stop'.

Podemos especificar el punto de un programa en el que deseamos que empiece la numeración automática. AUTO AT 2ØØ empezará la numeración automática a partir de la línea 2ØØ.

```
AUTO AT 2ØØ
```

STEP 1ØØ empezará en 2ØØ y aumentará cada línea siguiente en 1ØØ. Podemos utilizar cualquier número de 1 a 9999, que es el mayor número de línea que se permite. (Si se escribe AUTO AT 9999 STEP 1Ø, o algo similar, la máquina quedará confundida). No se olvide que si (por ejemplo) se introdujera la línea 25Ø por numeración automática, sustituiría cualquier línea numerada 25Ø que se hubiese escrito antes.

DELETE AND LIST
(SUPRIMIR Y LISTAR)

Se ha utilizado ya la palabra NEW: que elimina de la memoria del ordenador el programa que se está utilizando. La palabra DELETE (seguida por 'enter') realiza también esta tarea.

Al escribirse DELETE 1ØØ se elimina la línea 1ØØ del programa. DELETE 1ØØ TO 14Ø elimina la línea 1ØØ a 14Ø, ambas inclusive.

Si sólo queremos suprimir una línea, es más rápido escribir su número sólo y pulsar 'enter'.

Junto con DELETE se pueden usar las palabras FIRST y LAST. Así, DELETE FIRST TO LAST borraría todo un programa: !naturalmente, NEW (o sólo DELETE) es mejor!. Pero DELETE FIRST TO 1ØØ es bastante mejor si queremos suprimir las líneas hasta la 1ØØ, inclusive, de un programa de gran tamaño.

Podemos igualmente suprimir trozos menores de un programa, por ejemplo, determinadas palabras o caracteres de una línea. La tabla I al final del capítulo explica la manera de hacer esto. Para hallar una línea a fin de modificarla, escribir LIST. Esta palabra visualizará por sí misma (como ya sabemos) todo el programa. LIST seguido por un determinado número de líneas hará que esa línea aparezca por su cuenta justamente encima del cursor. Al escribir LIST ... TO... veremos un grupo de líneas del número primero al segundo, ambos inclusive. FIRST y LAST pueden usarse con LIST de la misma manera que con DELETE.

Acto seguido, es necesario colocar el cursor al principio o fin de la parte de la línea que deseamos retirar. Naturalmente, lo hacemos usando la palanca de mando. Una vez introducidos los cambios en la línea, pulsar 'enter' mientras el cursor sigue en esa misma línea. Con esto se volverá a introducir la línea modificada en lugar de la original. (Obsérvese que mientras se hace esto, el ordenador sigue 'recordando' la línea en su forma original).

ESCRIBIR ENCIMA
O INSERTAR
(OVERWRITE OR
INSERT)

Si queremos modificar algo en la mitad de una línea de programa, no es necesario utilizar las teclas 'erase' o 'del'. Mientras la máquina esté trabajando

en lo que se conoce como modalidad de sobre-escritura, cualquier carácter que esté debajo del cursor queda automáticamente suprimido cuando escribimos otro nuevo: lo que hacemos es 'escribir encima' del material antiguo y sustituirlo.

Así, para cambiar el número 234 por 567, colocamos el cursor sobre el 2 y escribimos los tres nuevos caracteres.

Si queremos introducir caracteres nuevos sin suprimir al mismo tiempo los antiguos, podemos poner el ordenador en modalidad de inserción. Para cambiar de modalidad de 'sobre-escritura' a modalidad de 'inserción', o viceversa, mantener pulsada la tecla 'ctrl' y apretar 'ins'. Para indicar que se está en 'modalidad de inserción', el cursor cambia de aspecto (y toma la forma de una flecha que señala a la izquierda).

En modalidad de inserción, cualquier cosa que se escriba en la mitad de la línea se coloca entre los caracteres que estaban allí antes: los de la derecha se desplazan para dejarle sitio. Esto significará a veces que desaparezcan algunas palabras en el margen derecho de la pantalla. Pero el ordenador no olvida estas palabras: podemos hacerlas reaparecer manteniendo apretada 'ctrl' y pulsando la tecla de función 1, como se explica en el capítulo siguiente. Mientras haya caracteres 'fuera del margen' de la pantalla, al final de la línea se verá el símbolo '>'.

Por tanto, otra manera de cambiar 234 por 567 es suprimir el número antiguo y seleccionar después 'modalidad de inserción' para poner el nuevo.

INSERCION DE LINEAS

Para insertar una línea de programas nueva completa, basta con hallar el lugar en la secuencia en donde se desea colocarla, y escribirla donde debe entrar la línea, como hemos visto ya al principio del manual.

MOVIMIENTO DEL CURSOR

Sabemos ya que el mantenimiento de la palanca de mando en una posición hará que el cursor se mueva repetidamente en esa dirección mientras no se suelte. Y hemos comprobado que cuando el cursor llega a los extremos superior e inferior de la pantalla, el texto 'se desliza' hacia abajo o hacia arriba, para representar las líneas que actualmente no aparecían en pantalla.

Si el cursor se está moviendo arriba o abajo, la pulsación de 'shift' hará que se mueva una página (o una pantalla completa) cada vez. Esto es útil para pasar rápidamente de una parte a otra de un programa largo. (Al utilizar 'ctrl' en lugar de 'shift' el cursor se moverá por párrafos, algo útil en el proceso de palabras).
La pulsación de 'ctrl' mientras

se mueve lateralmente el cursor lo desplaza en unidades de palabras en vez de letras. La pulsación de 'shift' y el movimiento lateral de la palanca de mando, hacen que el cursor se dirija directamente al final de la línea en la dirección que se desee.

Las dos tablas siguientes muestran las distintas maneras de suprimir e insertar caracteres. La colocación del cursor se hace siempre con la palanca de mando.

TABLA 1 - SUPRESION

FUNCION	TECLA (COMBINACION)	EXCEPCIONES/CONDICIONES
Borrar carácter a la izquierda: Borra el carácter situado a la izquierda del cursor y desplaza a la izquierda el resto de la línea	ERASE	Mueve el cursor un espacio a la izquierda. Si está ya al fondo de la izquierda, la línea se une a la anterior.
Borrar línea izquierda: Borra desde el comienzo de la línea hasta el cursor. Desplaza a la izquierda la parte derecha de la línea para cerrarla.	SHIFT + ERASE	Deja el cursor en la posición del carácter situado más a la izquierda.
Suprimir carácter derecha: Suprime el carácter debajo del cursor. Mueve a la izquierda la parte derecha de la línea para cerrarla.	DEL	Deja el cursor en la misma posición. !Son los caracteres los que se mueven!. Si el cursor está al final de la línea, pasa a la línea siguiente para unirla a ella.
Suprimir línea derecha: Suprime desde el cursor al final de la línea.	SHIFT + DEL	Deja el cursor en la misma posición.
Borrar palabra izquierda: Borra a la izquierda hasta que ha borrado el carácter situado más a la izquierda de una palabra.	CTRL + ERASE	Empieza con la posición del carácter a la izquierda del cursor. Sólo se consideran partes de la palabra las letras o números. Los caracteres especiales intermedios se borran.
Suprimir palabra derecha: Suprime a la derecha hasta que ha suprimido el carácter situado más a la derecha de una palabra.	CTRL + DEL	Empieza la posición de carácter debajo del cursor.

TABLA 2 - INSERCION

FUNCION	TECLA (COMBINACION)	EXCEPCIONES/CONDICIONES
Insertar carácter: Mueve caracteres del cursor al fin de la línea, una posición a la derecha. Coloca espacio debajo del cursor.	INS	En modalidad de inserción, equivale a pulsar la barra espaciadora, salvo que el cursor no se mueve.
Insertar línea: Baja una línea los caracteres que hay desde el cursor al fin de la línea.	SHIFT + INS	Si el cursor está al comienzo de la línea, con esto se crea una línea en blanco.
Pasar a inserción: Cambia entre modalidad de inserción y de sobre-escritura.	CTRL + INS	Aparece un cursor diferente según que el ordenador esté en modalidad de inserción o de sobre-escritura.
MODALIDAD DE INSERCION: Permite pasar el cursor a cualquier posición del programa o texto y escribir caracteres que entran por desplazamiento a la derecha de los que están después de ellos.		
MODALIDAD DE SOBRE-ESCRITURA: Los nuevos caracteres escritos sustituyen los que ya hay. Moviendo los caracteres a la derecha, no se deja espacio. Literalmente se escribe encima de caracteres escritos previamente.		

PROCESADOR DE TEXTOS

Algo muy interesante que podemos hacer con el ordenador, es utilizarlo como una máquina de escribir muy compleja. Podemos escribir un pasaje de un texto (una carta, un ensayo o cualquier otra cosa) e imprimirlo después en una impresora en línea si la tenemos. La ventaja del ordenador sobre la máquina de escribir normal, es que los errores se pueden corregir más simplemente, y que se proporcionan diversas funciones especiales para ayudarnos a ajustar a formato los párrafos y colocarlos en el sitio deseado.

EL PROCESADOR DE TEXTOS

El procesador de textos incorporado al Enterprise puede utilizarse con o sin el cartucho BASIC insertado. Si el cartucho está introducido y queremos cambiar de programación a proceso de palabras, escribimos la palabra TYPE y pulsamos 'enter'. En ese momento cambiará la pantalla; se verán unas palabras especiales en las partes superior e inferior, que indican los usos de las teclas de función para el proceso de palabras.

Es la misma pantalla que aparecerá si se enciende el ordenador sin que haya ningún cartucho introducido.

Obsérvese que el comando TYPE borrará de la memoria del ordenador cualquier programa BASIC. Otra manera de pasar este comando es pulsando 'shift' con la tecla de función 8.

Escribamos ahora algunas letras sin sentido, y pulsamos 'enter' cuando pensemos que hemos escrito bastante. Se verá entonces simplemente que el cursor pasa a la siguiente línea de abajo; no habrá ningún mensaje de error que indique que hemos introducido algo que el ordenador no comprende. Cuando se utiliza para proceso de textos, el ordenador se comporta simplemente como una máquina de escribir con algunas facilidades extra: muestra en pantalla lo escrito, pero no hace nada por intentar comprenderlo.

La tecla 'enter' la trata el procesador de textos como un 'retorno del carro': pero sólo es necesario pulsarlo al final del párrafo, no al término de cada línea. Recuérdese que el ordenador utiliza algo llamado "word wrap" (reiniciación de palabras), que mantiene automáticamente las líneas dentro de la longitud requerida.

Obsérvese que todo el texto que existe entre dos retornos del carro (la tecla 'enter') se cuenta como un párrafo. Las funciones que se aplican a párrafos completos (véase más adelante) trabajan todas en el párrafo que contiene actualmente el cursor.

Se pueden realizar modificaciones en el texto que aparece en pantalla exactamente igual que se modificaría un programa : ver las dos tablas que hay al final.

Las funciones especiales que vamos a examinar ahora están diseñadas fundamentalmente para utilizar cuando se escribe un documento, aunque sirven también si se introducen líneas del BASIC.

FUNCIONES ESPECIALES

Para utilizar cualquiera de estas funciones, pulsar la tecla de función correspondiente, mientras mantiene preparadas las teclas 'ctrl' o 'alt'.

'CTRL' + FUNCTION 1 - RE-FORM

Esta función ajusta la longitud de línea del párrafo que contiene el cursor. El útil si se han insertado o retirado varias palabras, y las líneas aparecen desiguales.

'ALT' + FUNCTION 1 - JUSTIFY AND RE-FORM

'Justificación' significa dejar rectos los márgenes de un párrafo de manera que el comienzo y el final de las líneas formen una columna perfectamente vertical. Es como el texto de la mayoría de los periódicos.

'CTRL' + FUNCTION 2 - CENTRE

Pone una línea de párrafo en la mitad horizontal de la 'página'. Es útil para títulos y encabezamientos.

'ALT' + FUNCTION 2 - CLEAR ALL TABS

Como una máquina de escribir, el ordenador permite colocar topes de tabulador y saltar después la línea hasta el punto elegido una vez pulsada la tecla 'tab'. Antes de reajustar un número de tabuladores, se pueden eliminar los anteriores con la pulsación de una sola tecla.

'CTRL' + FUNCTION 3 - TAB SET/CLEAR

Tab es la abreviatura de 'tabulación'. Significa dividir la pantalla en varias 'columnas' horizontales, a efectos de ajuste a formato. Si queremos escribir una tabla de información (como la tabl de teclas de función de la página 45, por ejemplo), podemos colocar tantos tabuladores en pantalla como columnas haya en la tabla. La tecla 'tab' puede utilizarse para pasar directamente de una columna a la siguiente. De este modo se asegura la rapidez y la precisión en la alineación de las columnas.

El ajuste del tabulador se hace simplemente pasando el cursor (con la palanca de mando) al lugar en donde se desea que esté el tabulador y pulsando después 'ctrl' con la tabla de función 3. Si hay ya un tabulador en esta posición, este mismo comando lo eliminar

'ALT' + FUNCTION 3 - RULER LINE

La 'regla' en la parte superior de la pantalla muestra en dónde están los márgenes actuales de la izquierda y la derecha, y el lugar en el que se han ajustado los tabuladores (indicado por unas líneas verticales). Este comando hace que aparezca o desaparezca la regla.

'CTRL' + FUNCTION 4 - LEFT MARGIN

Naturalmente, a veces se desea cambiar las posiciones de los márgenes. Para ajustar el margen de la izquierda (donde empiezan las líneas) mover el cursor por la pantalla al lugar en donde se desea que esté el margen, pulsándose acto seguido esta tecla de función con 'ctrl'.

'ALT' + FUNCTION 4 - RIGHT MARGIN

Como antes, pero para el margen de la derecha.

'CTRL' + FUNCTION 5 - RELEASE MARGINS

Este comando permite la introducción de palabras o caracteres fuera de los márgenes actuales. El símbolo (*) en la línea de estado indica que se ha pasado el comando; repetir la pulsación de la mismatecla para anularlo.

'ALT' + FUNCTION 5 - RESET MARGINS

Sirve para ajustar los márgenes a las posiciones más alejadas de la izquierda y derecha de la 'página'. También repone los tabuladores. Si se ha estado escribiendo una columna estrecha de texto en la mitad de la pantalla y se quiere desplazar los márgenes hacia afuera para el párrafo siguiente, pulsar 'alt' con la tecla de función 5 y acto seguido 'enter'. Los márgenes se pueden colocar en donde se desee.

'CTRL' + FUNCTION 6 - MOVE UP

Con este comando, se hace subir una línea al párrafo que contiene el cursor. La línea que estaba arriba reaparece debajo de ella. La pulsación de la tecla se puede repetir hasta que el párrafo esté en la posición deseada.

'ALT' + FUNCTION 6 - MOVE DOWN

Es lo opuesto de lo anterior.

'CTRL' + FUNCTION 7 - CHANGE LINE COLOUR

Se selecciona el nuevo par de colores del texto y del fondo en la línea en donde está el cursor. En una pantalla de 8Ø columnas, se pueden elegir cuatro pares de colores; la pulsación repetida de la tecla hará que aparezcan todos

ellos uno tras otro. En la pantalla de 4Ø columnas hay solamente dos pares de colores.

'ALT' + FUNCTION 7 - CHANGE PARAGRAPH COLOUR

Como antes, pero aplicado a un párrafo completo.

Los usos de las teclas de función independientemente (sin 'ctrl' o 'alt') varían según que se esté en programación o en proceso de palabras. Obsérvese que, en algunos casos, hay algunas características 'a prueba de fallo': por ejemplo, para cambiar a texto de 8Ø columnas mientras se usa el procesador de palabras, se pulsa la tecla de función 5 y se pide después la confirmación del comando pulsándose 'enter' (o su cancelación 'esc'). Esto impide que se borre un documento con la pulsación por accidente de la tecla de función.

En la página 45 se dan tablas completas de las operaciones de las teclas de función.

IMPRESION DEL TEXTO

Ahora que hemos escrito el documento, podemos desear su impresión en papel (si se tiene impresora) o su conservación en cassette para su uso posterior.

Si se quiere usar la impresora, debe estar conectada naturalmente al ordenador en la toma que hay en la parte posterior de la máquina (marcada 'printer'). Debe estar conectada y 'en línea'. Pulsar acto seguido la tecla de función 3. Un mensaje en la pantalla nos recordará que debemos asegurarnos que la impresora esté correctamente ajustada. Pulsar 'enter' y se imprimirá el documento.

CONSERVACION DE UN TEXTO EN CASSETTE

La conservación de un texto en cassette es similar a la conservación de un programa; en las páginas 46-47 se indica la manera de conectar el ordenador a la grabadora de cassette. Como el programa, el documento que queremos preservar debe recibir un nombre de acuerdo con las reglas de la página 46. La única diferencia es que el nombre del documento -el 'filename' o nombre de fichero- no se escribe entre comillas.

Pulsar la tecla de función 2. El ordenador nos pedirá entonces que introduzcamos el filename. Después de escribir (por ejemplo) LETTER, pulsar el botón 'record' de la grabadora de cassette y acto seguido 'enter'. El texto quedará conservado bajo control remoto si se han efectuado las conexiones adecuadas.

De igual manera, el procesador de textos nos permite cargar un documento al igual que se cargaría un programa (pero será necesario sacar el enchufe de la toma REM mientras se

rebobina la cinta). Pulsar simplemente la tecla de función 1, escribir LETTER - o el nombre que se le haya dado al documento - y pulsar 'enter'.

SALIDA DEL PROCESADOR DE TEXTOS

Cuando se ha terminado de utilizar el procesador de textos y se quiere volver a programación BASIC, pulsar la tecla de función 8. El ordenador nos preguntará entonces a qué programa queremos cambiar; escribir BASIC y pulsar 'enter'. Naturalmente, si se ha introducido un cartucho distinto al IS-BASIC, habrá que escribir en su lugar el nombre adecuado (por ejemplo LISP) para poderlo utilizar. Al escribir WP se vuelve al procesador de palabras mientras se borra el texto anterior. Si se ha pulsado por error la tecla de función 8, la pulsación de 'esc' nos devuelve al lugar en que estábamos anteriormente.

LAS TECLAS DE FUNCION

Hemos probado ya la pulsación de algunas o todas estas teclas. Están situadas encima de las teclas de los números y están también numeradas del 1 al 8.

El operador puede redefinirlas para que realicen cualquier función que se desee. Pero mientras no sean redefinidas, el ordenador proporciona a cada tecla funciones que es probable que se encuentren útiles en un cierto momento.

La redefinición de las teclas de función se hace escribiendo (por ejemplo):

```
SET FKEY 1 "PRINT"
```

La función que deseamos que ejecute la tecla, se debe indicar entre comillas. el número después de FKEY es el número de la tecla que se redefine.

Probemos a escribir el comando anterior y pulsar después la primera tecla de función. En pantalla aparecerá la palabra PRINT. Esto significa que las teclas se pueden usar para poner en pantalla palabras clave utilizadas frecuentemente, pulsando una sola tecla en vez de tener que escribir toda la palabra. Las palabras más útiles son RUN, LIST y RENUMBER, porque no siempre es necesario añadir algo detrás de ellas. La pulsación de una tecla de función para listar un programa es mucho más rápido que escribir la palabra LIST y pulsar 'enter'.

A la definición de una tecla de función se le puede añadir un 'retorno' del carro (lo equivalente a pulsar 'enter'), añadiendo

```
&CHR$(13)
```

al fin de dicha definición, por ejemplo:

```
SET FKEY 1 "PRINT" &CHR$(13)
```

La tabla grande de la página 45 muestra el uso de cada tecla de función tal como se encuentran configuradas cuando se enciende el ordenador, con el cartucho BASIC introducido. Obsérvese que con todas las teclas de función se pueden utilizar 'shift', 'ctrl' y 'alt', lo que significa que cada tecla realiza en realidad cuatro funciones diferentes. Cuando se quiere reponer las teclas a estas funciones después de redefinirlas, se ha de escribir CLEAR FKEYS.

Cuando se pulsan solas, como ya se sabe, las teclas de funciones tienen usos diferentes cuando estamos

utilizando el procesador de textos. Estos usos se indican en la tabla pequeña.

OPERACIONES DE LAS TECLAS DE FUNCION

BASIC solamente — BASIC o procesador de palabras

Tecla	NORMAL	+ SHIFT	+ CTRL	+ ALT
1	START. Ejecuta el programa. Si no hay programa, carga en la memoria el programa a partir del disco y lo ejecuta. Si no hay disco, lo carga desde la cassette.	CONTINUE Continúa el programa después del comando 'stop'.	REFORM Ajusta las longitudes de las líneas para mantenerlas dentro de los márgenes (hace que el párrafo aparezca más claro después de edición)	JUSTIFY AND REFORM Iguala los espacios en las líneas y alinea los márgenes.
2	LIST Lista el programa completo en pantalla.	LLIST Lista el programa completo en la impresora.	CENTRE Coloca el texto en el centro de la pantalla.	CLEAR ALL TABS
3	AUTO Numeración automática en pasos de 1Ø. para desconectarlo, pulsar 'stop'.	RENUMBER Renumera todo el programa en pasos de 1Ø.	TAB SET/CLEAR Fija o retira los topes de tabulador en la posición actual del cursor.	RULER LINE Se conecta o desconecta una regla que muestra los márgenes y los tabuladores.
4	REMOTE 2 Conecta y desconecta el interruptor de control Rem 1.	REMOTE 2 Como el anterior, pero para la toma 2.	LEFT MARGIN Ajusta el margen izquierdo a la posición actual del cursor.	RIGHT MARGIN Lo mismo que con el margen izquierdo, pero para el derecho.
5	TEXT Pone toda la pantalla en modalidad de texto y borra las páginas de gráfico y texto.	DISPLAY TEXT Pasa de gráfico a página de texto, sin borrar ninguno de los dos.	RELEASE MARGINS Permite la introducción de texto fuera de los márgenes ajustados.	RESET MARGINS Ajusta márgenes para permitir que se escriba a todo lo ancho de la pantalla.
6	GRAPHICS Prepara las 2Ø primeras líneas como gráficos y las últimas 4 como texto; borra las páginas tanto de texto como de gráficos.	DISPLAY GRAPHICS Pasa de representación de texto a gráfico sin borrar ninguno de los dos.	MOVE UP Coloca el párrafo encima de la línea que lo precedía.	MOVE DOWN Traslada el párrafo por debajo de la línea.
7	CLICK Conecta o desconecta el ruido de "clic" del teclado (se oye cada vez que se pulse una tecla).	SPEAKER Desconecta toda la capacidad sonora (o la conecta de nuevo).	CHANGE LINE COLOUR (Cambia el color de la línea).	CHANGE PARAGRAPH COLOUR (Cambia el color del párrafo).
8	INFO Da información sobre los programas que hay en la memoria.	TYPE Nos coloca en modalidad de procesador de textos.		

FUNCIONES 'NORMALES' PARA PROCESADOR DE TEXTOS SOLAMENTE

1. LOAD Permite la carga del texto desde cassette o disco.	2. SAVE Permite la preservación del texto en cassette o disco.	3. PRINT Envía un texto a la impresora.	4. HELP Visualiza información sobre diversas funciones del procesador de textos.
5.TEXT 8Ø Borra el texto y dispone la pantalla en modalidad de 8Ø columnas.	6. TEXT 4Ø Borra el texto y dispone la pantalla en modalidad de 4Ø columnas.	7. CLICK Conecta y desconecta el "clic" del teclado.	8. EXIT Retira el procesador de textos y permite la vuelta al BASIC, etc.

MANEJO DE PROGRAMAS EN CASSETTE

En la Guía para la Preparación, hemos aprendido a controlar una grabadora de cassette para cargar simplemente un programa en la memoria del ordenador. Sin embargo, se puede utilizar también una grabadora de cassette para almacenar los propios programas.

Supongamos que se quiere conservar un programa que fue copiado de una parte anterior de esta sección de manual, tal vez con algunas adiciones propias al mismo, supongamos que tenemos ya un programa en la memoria del ordenador cuando hemos llegado a este punto.

En primer lugar, asegurarse de que el ordenador y la grabadora de cassette están correctamente conectados. La toma que hay en la parte posterior de la máquina, marcada OUT, es la que sirve para poner los programas en cassette. Conectar en esta forma uno de los cuatro conectores del cable del cassette.

Acto seguido conectar el extremo opuesto del cable en la toma marcada MIC (o algo similar) de la grabadora.

A continuación enchufar uno de los enchufes pequeños en la toma marcada REM2, y su extremo opuesto en la pequeña toma para control a distancia de la grabadora (marcada probablemente REM), si la tiene el aparato.

!Cuando se hace esto no importa que el ordenador esté conectado o desconectado!.

A continuación hay que dar un nombre al programa. Para mayor claridad supongamos que lo llamamos 'miprog'. El nombre puede tener hasta 28 caracteres de longitud y contener letras, números y los signos de puntuación siguientes: '.', '-', '_', '' y '/'.

Podemos llamar al programa 'Mi-programa-número-1' si lo deseamos.

Escribir:

```
SAVE "miprog"
```

Pulsar el botón 'record' y a continuación 'enter'. Aparecerá el mensaje.

```
SAVING miprog
```

- y a continuación:

```
OK_
```

una vez que se ha terminado de grabarlo. Dado que está conectada la toma a distancia, el Enterprise detendrá y pondrá en marcha automáticamente

la cinta en el momento adecuado, a condición de que la grabadora tenga esta facilidad.

Mientras se está cargando el programa en cinta, el sonido de esta operación saldrá suavemente por el altavoz incorporado al ordenador; esto es una indicación útil de que se está realizando adecuadamente la operación de conservación.

ASEGURARSE

Naturalmente, conviene comprobar que el programa ha quedado conservado adecuadamente en la cinta. Esto se hace con el comando VERIFY.

En primer lugar, rebobinamos la cinta al lugar en que estaba al empezar a cargar el programa. (Si se utiliza el control a distanci el ordenador impedirá probablemente que se rebobine la grabadora. Si así ocurre, escribir TOGGLE REM 2 o pulsar la tecla 'shift' junto con la tecla de función 4). A continuación escribir:

```
VERIFY "miprog"
```

y poner en marcha la cinta pulsando después 'enter'. También aquí el Interprise controlará la grabadora.

Si el programa ha sido grabado sin error, el ordenador responderá con el mensaje:

```
ok
```

En caso contrario, aparecerá un mensaje de error. Si así ocurre, hay que asegurarse de que las conexiones están bien hechas, comprobar si el control de volúmen está ajustado al nivel correcto de grabación, y repetir todo el proceso de nuevo.

MEZCLA DE PROGRAMAS

Si se tiene ya un programa en el ordenador y se le quiere añadir otro a partir de cinta, se puede hacer con el comando MERGE.

MERGE es bastante útil cuando nos encontramos a la mitad de la grabación de un programa largo, y lo hemos salvado como una serie de subprogramas (o programas menores que forman parte de otro mayor). Si queremos reunir todos los programas a partir de todas sus partes, grabados ya en cinta, podemos hacerlo usando MERGE.

Si se quiere reunir en el ordenador las líneas de ambos programas, hay que asegurarse de que todos los números de las líneas sean diferentes. Esta es la razón:

Imaginemos que tenemos en la memoria un pequeño programa.

Los números de sus líneas son 1ØØ, 11Ø, 14Ø y 16Ø.

Imaginemos ahora que queremos unirlo a otro pequeño programa en cinta. Sus números de líneas son 14Ø, 16Ø, 18Ø, y 2ØØ.

Si intentamos MERGE estos dos programas sin RENUMBER (cambiar la numeración) del que está en la memoria, las líneas 14Ø y 16Ø del programa en cinta sustituirían a las 14Ø y 16Ø de la memoria. Por tanto, hubiéramos perdido realmente dos líneas del programa original.

Así, cuando se fusionan dos programas, las líneas de cada programa se "ensamblan" entre sí para formar un programa mayor.

He aquí la manera de fusionar un programa a partir de cinta.

Con la grabadora, el método es exactamente igual que para la carga (MERGE es igual que cargar, salvo que la carga de un programa sustituye cualquier cosa que haya en la memoria en ese momento: "elimina" el programa actual y lo sustituye por el nuevo). Si no estamos seguros de cómo cargar un programa, ver la página 11 de la Guía para la Preparación.

Por tanto, conviene que nos aseguremos que la cinta está al principio del programa que deseamos MERGE y que estamos seguros de que son correctos todos los números de línea. A continuación escribimos

```
MERGE "nuevo prog"
```

('nuevo prog' es el nombre del programa: el nombre con el que fue originalmente preservado).

Poner en marcha la cinta y pulsar 'enter'. Los programas se unirán. !Probémoslo!.

FICHEROS

Los programas almacenados en cinta se denominan también ficheros. Esto se aplica igualmente al otro medio de almacenamiento de programas, los discos. Un 'fichero' puede ser una colección de información para que la utilice el ordenador. Se denomina fichero de datos.

DISCOS

Si hay una unidad de discos acoplada al ordenador, se pueden utilizar igualmente todos los comandos ya conocidos, con la ventaja de que todas las operaciones de conservación, carga, etc. se realizarán con mucha mayor rapidez.

LA SECCION TUTORIAL

SERIES

En la parte primera del manual tratamos de las series con muy poco detalle. Ahora que hemos aprendido los conceptos fundamentales, unas palabras sobre las series servirán para poner en perspectiva nuestros conocimientos.

Las series son uno de los dos tipos principales de información - siendo el otro los números - que puede manejar el ordenador. En efecto, lo que en realidad hace un ordenador es manejar información. Probablemente se habrá caido ya en la cuenta de que un ordenador no tiene la capacidad de pensar como lo hace una persona. Hace exactamente lo que le decimos que haga, con la información que nosotros le damos. Por eso, en última instancia, la programación no es más que una manera de clasificar, recoger y manipular información.

Antes las series aparecían como frases entre comillas. Las comillas se utilizan cuando se repite exactamente lo que se ha dicho, pero no hay necesidad de entender las palabras citadas, pueden carecer de sentido.

Este es el principio que se esconde tras la idea de las series. El ordenador no entiende lo que son ni lo que significan. Simplemente las trata como símbolos sin sentido.

Cualquier cosa que hay en un programa, colocada entre comillas, será considerada por el ordenador como una serie. He aquí una serie:

"¿Que edad tienes?"

Estos mismos caracteres pueden aparecer en un libro como cita:

"¿Que edad tiene?" , dijo Jim.

Observemos también que se pueden colocar números entre comillas:

"tengo 22", respondió Sally.

Probemos estos ejemplos:

```
PRINT 2*2
```

Este ejemplo pide al ordenador que realice un pequeño producto; pone en pantalla el valor de 2 multiplicado por el valor de 2 ; es decir, hace el producto y visualiza el resultado

```
PRINT "2*2"
```

Este ejemplo parece exactamente igual al anterior, pero las comillas convierten 2*2 en una serie. Ahora estamos diciendo al ordenador que ponga 2, y después *, y más tarde 2 en pantalla.

El ordenador puede decirnos cual es la longitud de una serie de caracteres si se lo pedimos. Probemos este programa:

```
100     LET A$="COMPUTER"
110     LET B$="MICROCOMPUTER"
120     !
130     !     Lines 100 and 110 declare two string
140     !     variables, A$ and B$. String variables
150     !     are just like numeric ones, except that
160     !     they are names for groups of symbols,
170     !     not for the values of numbers. The name
180     !     given to a string variable must always
190     !     have a dollar sign ($) at the end of it.
210     !
220     LET A=LEN(A$)
230     LET B=LEN(B$)
240     !
250     !     A and B are numeric variables which
260     !     contain the lengths of the two strings.
270     !     LEN tells you the number of characters
280     !     in a string (i.e. its length).
300     !
310     PRINT "The string called A$ is ";A;
        " characters long."
320     PRINT "The string called B$ is ";B;
        " characters long."
330     END
```

130 140 150 160 170 180 190

Las líneas 100 y 110 declaran dos variables de series. A$ y B$. Las variables de series son como las numericas, salvo que son nombres de grupos de símbolos, no valores de números. El nombre que se da a una variable de serie debe tener siempre el signo del dólar ($) al final.

250 260 270 280

A y B son variables numéricas que contienen las longitudes de las dos series. LEN nos indica el número de caracteres de una serie (es decir, su longitud).

310 "La serie llamada A$ tiene"; A;
"caracteres de longitud".
320 "La serie llamada B$ tiene"; B;
"caracteres de longitud".

Probemos a cambiar el conteido de A$ y B$: el ordenador podrá decirnos siempre cuantos símbolos contiene cada uno

ESPACIOS

El espacio puede que no represente nada para nosotros, y es comprensible que pensemos que no cuenta. Pero el ordenador lo trata como si fuera un símbolo. Probemos a cambiar A$ y B$ (en el programa anterior) por:

" " y " "

Veremos que el ordenador nos dice cuántos espacios contienen, aunque parece que ambos están

vacios. Una serie realmente vacia aparecería más o menos de esta forma:

""

Como puede verse, ni siquiera contiene espacios. Se llama serie nula.

RIES
NTRO DE
RIES

El Enterprise puede hacer con las series algunas cosas interesantes. Por ejemplo, podemos formar una serie de otra. He aquí un programa que hace esto:

```
100   LET BIGSTRING$="quite a few words"
110   !
120   !   Line 100 declares a string variable.
130   !
140   LET SMALLSTRING$=BIGSTRING$(9:17)
150   !
160   !   Line 140 declares another string
170   !   variable—with a difference. It starts at
180   !   the 9th character of BIGSTRING$ and
190   !   ends at the 17th character.
210   !
220   PRINT BIGSTRING$
230   PRINT SMALLSTRING$
240   END
```

1ØØ "unas pocas palabras"

12Ø La línea 1ØØ declara una variable de serie.

16Ø 17Ø 18Ø 19Ø La línea 14Ø declara otra variable de serie, con una diferencia: comienza por el 9º carácter de BIGSTRING$ y termina en el carácter número 17.

El ordenador copia literalmente algunos caracteres de una variable en otra distinta. A esta variable separada se la llama subserie. Para sacar una serie de otra, hemos dado a la subserie un nombre - SMALLSTRING$ - y especificar a continuación, entre paréntesis después del nombre de la serie mayor, el carácter en el que queremos que empiece la subserie y el carácter en el que debe terminar.

Por tanto, una sentencia como LET INITIAL$ = NAME$(1:1) dice 'llama a la subserie INITIAL$ y haga que contenga la primera letra de NAME$'.

En cada uno de estos ejemplos hemos declarado nuestra subserie como variable por sí misma, dándole un nombre (un 'identificador') propio. Esto no es estrictamente necesario.

En el programa anterior, podríamos suprimir la línea 14Ø y cambiar la línea 23Ø por:

```
230   PRINT BIGSTRING$(9:17)
```

que quiere decir 'imprime los caracteres noveno a decimoséptimo de la serie llamada BIGSTRING$'. El método que

elijamos dependerá de la razón por la que usamos subseries. Si queremos usar la misma subserie varias veces, es casi siempre mejor declararla como variable separada. En caso contrario, bastará con el método anterior.

INKEY$

INKEY$ es una función de serie muy útil e importante. (Cualquier palabra del BASIC que haga algo en una serie se denomina función de serie; la mayoría de ellas termina con el signo $, indicando que el resultado que produce es también una serie).

INKEY$ nos permite pulsar una tecla mientras se está ejecutando el programa, a fin de afectar a la manera en que continuará el programa. En cierto modo es como poner un INPUT, pero las diferencias las veremos muy pronto con claridad. Probemos esto:

```
100     PRINT "Have you understood this chapter?"
110     PRINT
120     PRINT "Answer with y or n"
130     DO
140         LET A$=INKEY$
150     LOOP UNTIL A$<>""
160     !  ------------------------------------------
170     !  Line 140 puts the result of a key-
180     !  press into a string variable. You'll
190     !  see why this is important.
200     !  ------------------------------------------
210     IF A$="y" THEN
220         PRINT "Let's hope you're right."
230     ELSE IF A$="n" THEN
240         PRINT "Well, read it again."
250     ELSE
260         PRINT "That was a wrong key."
270     END IF
280     !  ------------------------------------------
290     !  You have seen IF/THEN statements
300     !  before. The purpose of ELSE in lines 230
310     !  and 250 is fairly obvious from the
320     !  ordinary meaning of the word. 'Con-
330     !  ditional' statements like these are dealt
340     !  with at length in the chapter on decisions.
350     !  ------------------------------------------
360     !  END
```

```
100  "¿Ha entendido este capítulo?"

120  "Responda con Y o N"

170  180  190   La línea 140 pone el
resultado de una pulsación de tecla
en una variable de serie. Comprobaremos
por qué es importante esto.

220  "Esperemos que tenga razón"

240  "Bueno, leálo de nuevo"

260  "Fue una tecla errónea"

290  300  310  320  330  340
```

Hemos visto antes las sentencias IF/THEN. La finalidad de ELSE en las líneas 230 y 250 es bastante clara por el sentido ordinario de la palabra. Las sentencias 'condicionales' como éstas se tratan con mayor extensión en el capítulo sobre decisiones.

Cuando el programa dirige su pregunta, la siguiente pulsación de la tecla da la respuesta y hace que el ordenador

pase a la línea 21Ø y continue.

Contrariamente a la sentencia INPUT, la función INKEY$ sólo pide que se pulse una tecla, y no nos exige el uso de 'enter'. (Por lo tanto, es muy útil en los juegos: para respuestas de una letra, o para continuar una acción suspendida en la mitad por un bucle). Obsérvese que 'shift' más otra tecla se cuenta como una sola pulsación.

Otra diferencia entre INKEY$ y INPUT es que INKEY$ no hace automáticamente que el ordenador espere a que se de la respuesta. Esta es la razón de que se necesiten las líneas 13Ø y 15Ø en el programa anterior. Probemos a suprimirlas. Veremos que el programa se ejecuta perfectamente hasta el final en una fracción de segundo, sin darnos tiempo a escribir la respuesta. Cuando el ordenador llega a la línea 14Ø, decide que al no haberse pulsado ninguna tecla, IN KEY$ equivale a una serie nula (""); por lo tanto, hace que A$ sea también una serie nula.

Por otra parte, las líneas 13Ø y 15Ø dicen 'manten el bucle hasta que A$ se convierta en algo diferente de una serie nula... acto seguido continua con el resto del programa'.

Téngase presente que INKEY$ es como una variable cuyo valor cambia rapidísimamente. El caso es que el ordenador 'observa' el teclado aproximadamente una vez cada quinta parte de segundo, para registrar cualquier pulsación realizada en ese intervalo. Si pulsamos (por ejemplo) la tecla 'a', entonces INKEY$ se hace equivalente a "a", pero sólo durante un período de tiempo muy corto. Normalmente, una quinta parte de segundo más tarde, INKEY$ volverá a serie 'nula' y permanecerá con este valor hasta la siguiente pulsación de una tecla.

Esto explica por qué se necesita la línea 14Ø en el programa. Esta pone inmediatamente el resultado de la pulsación en una variable de serie separada que no cambiará mientras el ordenador la está examinando (probándola para comprobar si cumple determinadas condiciones: ver las líneas 21Ø a 27Ø).

Podríamos intentar suprimir la línea 14Ø y modificar las líneas 15Ø, 21Ø y 23Ø por:

```
15Ø    LOOP UNTIL INKEY$<>""
21Ø    IF INKEY$="y" THEN
23Ø    ELSE IF INKEY$="n" THEN
```

La razón por la que el programa no funciona ya adecuadamente es porque el valor de INKEY$ cambia (vuelve a "") entre las líneas 15Ø y 21Ø.

Naturalmente, no es necesario poner el valor de INKEY$ en una variable separada si no se va a utilizar en líneas posteriores del programa. Por ejemplo, si queremos simplemente suspender la acción de un programa hasta que el usuario le indica que continúe, podemos escribir algo así:

```
100   PRINT "Press any key to continue."
110   DO
120   LOOP UNTIL INKEY$<>""
```

```
10Ø  "Pulsar cualquier tecla para
     continuar"
```

UCASE$ Y LCASE$

Otra cosa interesante que puede hacerse con las series es cambiarlas en letras mayúsculas o minúsculas.

Las dos palabras que hacen esto son UCASE$ y LCASE$. Este programa enseñará cómo se hace:

```
100   !
110   !  This program will convert a string that
120   !  you have typed, first into capitals and
130   !  then into small letters.
140   !
150   INPUT PROMPT "Please type in some lette
      ": A$
160   PRINT A$
170   PRINT UCASE$(A$)
180   PRINT LCASE$(A$)
190   END
```

```
11Ø  12Ø  13Ø  Este programa convertir
     una serie escrita primero en
     mayúsculas y después en minúsculas.

15Ø  "Por favor, escriba unas letras"
```

VAL

Otra cosa que podemos hacer es convertir una serie que contiene símbolos de números en el número que sería si no se tratara de una serie. La palabra BASIC que realiza esto es VAL (abreviatura de 'valor'). Probemos esto:

```
100   INPUT PROMPT "Please type in some
      characters: ":A$
110   PRINT A$
120   PRINT VAL(A$)
130   END
```

```
10Ø  "Por favor, escriba unos caracteres:"
```

Para determinar VAL(A$), el ordenador sólo encontrará los números de A$ que aparecen antes de la primera letra. Así, VAL ("123AB45") equivale a 123, mientras que VAL ("AB1234") equivale a Ø.

Podemos hacer también lo contrario - convertir un número en una serie - usando la palabra STR$.

MAR SERIES
TRE SI

Hemos visto ya cómo podemos convertir parte de una serie en una subserie. Otra manera de formar una nueva serie es enlazando series (o partes de series) entre sí. Al acto de unión entre sí de varias series se denomina concatenación; todo lo que hay que hacer es poner un signo ('&') entre las series, como si dijéramos 'una serie y otra serie'.

El ejemplo siguiente utiliza la concatenación además de las subseries y la función LEN.

```
100     INPUT PROMPT "Please type a word:
        ":STRING$
120     CLEAR SCREEN
130     LET ABBREV$=STRING$(1:1) & STRING$
        (LEN(STRING$): LEN (STRING$))&"."
140     !
150     !     In line 130, STRING$(1:1),makes the first
160     !     character of STRING$ into a substring.
170     !     LEN(STRING$) gives the number of
180     !     characters that STRING$ contains, so
190     !     STRING$ (LEN(STRING$): LEN
200     !     STRING$)) makes another substring out
210     !     of the final character. Then, using the '&'
220     !     symbol, the two substrings are linked to
230     !     each other and to a full stop, thus
240     !     defining a new string variable,
245     !     ABBREV$.
250     !
260     PRINT STRING$;" abbreviated is "; ABBREV$
270     END
```

```
1ØØ   "Por favor, escriba una palabra:"

15Ø 16Ø 17Ø 18Ø 19Ø 2ØØ 21Ø
22Ø 23Ø 24Ø 245

En la línea 13Ø, STRING$ (1:1)
convierte en una subserie el primer
carácter de STRING$. LEN (STRING$)
da el número de caracteres que con
tiene STRING$, por lo que STRING$
(LEN(STRING$):LEN (STRING$)) forma
otra subserie del carácter final.
A continuación, usando el símbolo
&, las dos subseries se unen entre
sí y con un punto, definiendo así
una nueva variable de serie, ABBREV$
26Ø   "abreviado es"
```

Podemos hacer esto si queremos usar repetidas veces la abreviatura. Si fuéramos a utilizarla sólo una vez, podríamos naturalmente suprimir la línea 13Ø y cambiar la línea 26Ø por:

```
260     PRINT STRING$ (1:1) &STRING$
        (LEN(STRING$): LEN(STRING$))&"."
```

En este caso, el signo (&) podría ser sistituído por punto y coma.

BUCLES

Hemos visto ya algunos programas cortos que utilizan bucles para que el ordenador repita algunas sentencias. Para que la máquina realice una operación varias veces seguidas, el bucle representa una manera mucho más fácil de hacerlo que escribir el mismo elemento del BASIC una y otra vez.

Un bucle es también una manera fácil de controlar todo un programa como veremos.

Con una sola excepción (GOTO, que puede tratarse como bucle), cada bucle contiene una sentencia especial para marcar su comienzo, y termina con una línea que indica al ordenador que vuelva de nuevo al principio.

DO/LOOPS

Veamos primero la manera más fácil, los DO/LOOP. Ya lo hemos visto anteriormente. Aparecen más o menos así:

```
 80    INPUT PROMPT "Please type in a number
       and I'll print the 'times table' for it. ":A
 90    LET B=A
100    DO !          Beginning of DO/LOOP
120      LET B=B+A
130      PRINT B,
140      !  ------------------------------------
150      !  Line 90 puts your number into
155      !  another variable, B. Then, in
160      !  lines 120 and 130, the value of
170      !  A (which has not changed) is
175      !  added to B, and the new value
180      !  for B is put on the screen. This
185      !  happens each time round the
190      !  loop, so B increases by the value
200      !  of A each time.
210      !  ------------------------------------
220    LOOP UNTIL B > 150 ! End of DO/LOOP
230    !  ------------------------------------
240    !  UNTIL B > 150 stops the program
245    !  going on forever without any
250    !  interference from you (e.g. 'stop' or
260    !  'hold' keys). UNTIL means exactly
265    !  what you might think it does—i.e.
270    !  'until B is bigger than 150'.
280    !  ------------------------------------
290    END
```

80 "Por favor, escribe un número y yo imprimiré el "horario" para él"

100 Comienzo de DO/LOOP

150 155 160 170 175 180 185 190 200

La línea 90 pone el número en otra variable, B. A continuación, en las líne 120 y 130, se añade a B el valor de A (que no ha cambiado) y aparece en pantal el nuevo valor de B. Esto sucede cada ve que se repite el bucle, por lo que B aumenta cada vez en el valor de A.

240 245 250 260 265 270 UNTIL B 150 impide que el programa continúe indefinidamente sin interferencia nuestr (por ejemplo, las teclas 'stop' o 'hold' UNTIL significa exactamente lo que la misma palabra indica: es decir, 'hasta que B sea mayor que 150'.

Un DO/LLOP, como podemos ver, hace que el ordenador 'de vuelta en círculo' hasta que se cumple alguna condición

ág. 60 que le ordena que se detenga. Esta condición puede especificarse con el uso de las palabras UNTIL o WHILE.

Sea cual fuere la que se utilice, podemos elegir entre DO WHILE (o DO UNTIL) y LOOP WHILE (o LOOP UNTIL). Normalmente, puede usarse cualquiera de ambas alternativas, sobre todo en un programa como el anterior, que no depende de una disposición especialmente delicada de condiciones.

La diferencia esencial entre poner una condición al comienzo y ponerla al final de un buble, es que afecta, con una diferencia de uno, al número de veces que podría suceder un bucle. Si la condición está al principio, se probará antes de que se ejecute el bucle, lo que significa que si se cumple la condición, el bucle nunca se ejecutará. Si se pone al final, la condición no puede ser realmente leída por el ordenador hasta entonces, lo que significa que el bucle se ejecutará siempre al menos una vez. WHILE y UNTIL son tipos opuestos de condiciones. UNTIL B= 15Ø y WHILE B= 15Ø significan cosas totalmente diferentes (UNTIL significa 'mientras no sea' y WHILE significa 'mientras sea').

Intentemos cambiar el bucle anterior usando una condición WHILE en lugar de UNTIL, con el uso de números mayores y menores en respuesta a la solicitud INPUT PROMPT del principio, y poniendo la condición después de DO en vez de después de LOOP. Las pruebas nos mostrarán muy pronto lo que podemos hacer con un DO/LOOP.

ALIDA Si se desea, puede hacerse que la máquina salga en cualquier momento de un bucle usando la palabra EXIT. Ahora bien, EXIT se aplica igualmente a los bucles FOR/NEXT (se describen en la siguiente página). Por tanto, es necesario especificar de que tipo de bucle se quiere salir: EXIT FOR o EXIT DO es la manera de hacerlo. Se debe salir usando algún tipo de condición, por ejemplo:

```
1000  IF X > 25 THEN EXIT DO
```

Un rápido repaso al capítulo sobre decisiones nos enseñará cómo hacer esto usando también SELECT CASE.

Recuérdese que la palabra EXIT es la única manera adecuada de abandonar un bucle antes de que termine. Otras cosas como GOTO no debe utilizarse para esto, ya que pueden confundir tanto al operador como al ordenador. (GOTO y su comando asociado GOSUB se tratan en las páginas 127-128). EXIT hará simplemente que el

ordenador salte a la primera línea después del final del bucle.

BUCLES FOR/NEXT

Los bucles FOR/NEXT son bastante distintos a los DO/LOOP. Es posible que sean un poco menos sencillos, pero tienen unos usos muy claros y definidos.

He aquí un ejemplo corto de bucles FOR/NEXT:

```
100     INPUT PROMPT "How many times would you
        like me to loop? ":X
110     ! ------------------------------------
120     !       Notice that in this and other INPUT
125     !       PROMPTs, a space is put at the end
130     !       of the prompt itself. This makes the
140     !       screen look tidier.
150     ! ------------------------------------
160     FOR P=1 TO X
170       PRINT "LOOP THE LOOP!"
180     NEXT P
190     END
```

100 "¿Cuántas veces quieres que repita el bucle?"

120 125 130 140 Obsérvese que en ésta línea y en otras solicitudes de entrada INPUT PROMPT, hay un espacio al término del mismo mensaje de solicitud. Esto hace que la pantalla aparezca más clara y limpia

---

¿Vemos cómo el número que hemos escrito es el número de veces que el ordenador ejecutará el bucle? Esta es la finalidad esencial de un FOR/NEXT: podemos especificar rápidamente y con concisión cuántas veces debe ejecutarse el bucle. No es preciso utilizar condiciones salvo en circunstancias especiales.

FOR/NEXT puede contar también en sucesiones de tres, veinte, Ø.2, 1ØØØ, 466.666 o incluso volver atrás. STEP es la clave para esta pequeña habilidad. He aquí un programa que cuenta hacia atrás de dos en dos.

```
10      FOR P=40 TO Ø STEP -2
20        PRINT P,
30      NEXT P
40      END
```

Simplemente no podemos pedir a la máquina que cuente de 4Ø a Ø sin pasarle un comando STEP. Debe saber restar (contar en números negativos) del número dado en la línea FOR antes de que haga esto. Salvo que especifiquemos el uso de STEP, el bucle FOR/NEXT contará siempre adelante en pasos de 1. Obsérvese también la sentencia PRINT P. La primera línea del bucle crea una variable, en este caso P, cuyo valor cambia cada vez que se ejecuta el bucle. El programa 'horario', utilizado como ejemplo del DO/LOOP, será

más corto si usamos FOR/NEXT.

```
100    INPUT PROMPT "Please type in a number
       and I will give you a 'times table' for it. ":A
110    FOR P=0 TO 150 STEP A
120      PRINT P,
130    NEXT P
140    END
```

```
100  "Por favor, escriba un número y yo
     te daré un 'horario' para él.":
```

Este programa hace prácticamente lo mismo que el DO/LOOP (página 59) pero de manera diferente. Los DO/LOOP son algo más fáciles de leer y entender cuando no estamos familiarizados con programas de ordenador.

Observemos que el bucle FOR/NEXT se prueba siempre al comienzo y no se ejecutará si ha pasado más allá del límite especificado en la sentencia FOR. Probemos a introducir número mayores de 150 para el programa con 'horario' anterior.

Si se quiere, pueden probarse diferentes maneras de presentar este programa usando PRINT AT (página 31). Combinando los comandos PRINT y GRAPHICS (página 89), podríamos diseñar una tabla formada por casillas para introducir en ella el 'horario'. ¡Y ésta vez, el manual no nos va a decir nada más! La programación es una aventura y significa que debemos encontrar cosas nosotros mismos. Podríamos también tener mejores ideas. Si somos un padre con niños pequeños, o tenemos un hermano o hermana menor, este es el tipo de programa que podríamos usar para enseñarles a divertirse un poco.

UCLES
ANIDADOS

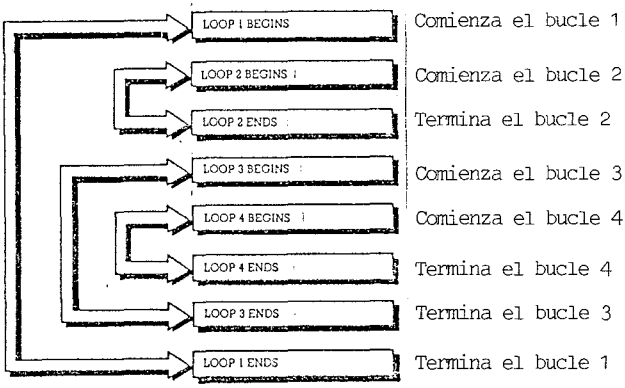
Estas palabras pueden sonar algo extrañas. Sin embargo, los bucles anidados no son más que bucles colocados dentro de otros bucles. Lo que conviene recordar es no terminar nunca el segundo bucle fuera del primero. He aquí un ejemplo:

```
100   FOR P=1 TO 20 !          First loop begins.
110     PRINT P,
120     FOR A=1 TO 4 !         Second loop begins.
130       PRINT P+10; P-10;
140     NEXT A !               Second loop ends.
145     PRINT
150   NEXT P !                 First loop ends.
```

```
100  Comienza el primer bucle.
120  Comienza el segundo bucle.
140  Termina el segundo bucle.
150  Termina el primer bucle.
```

Podemos anidar otro bucle dentro del segundo y otro en su interior, y así sucesivamente. Si observamos una serie de bucles anidados, las líneas de extremo deben estar en

orden descendente cuando miramos el programa. Los comienzos estarán en orden ascendente. Veámos el esquema.



Como puede verse en el esquema, si pensamos realmente que los bucles eran la respuesta a todos nuestros problemas, podríamos también "emparedar" bucles entre dos comienzos o dos finales de otros bucles.

Sin embargo, una disposición muy compleja de bucles de este estilo necesita una gran presencia de ánimo si queremos no perder el control de ellos. El Enterprise ayuda sangrando los bucles como se muestra en el último programa, lo que facilita su lectura. Las líneas de comentario ayudan también a entender mejor las cosas. Normalmente, si tenemos varias capas de bucles anidados, es más fácil definir las 'capas' interiores como funciones (ver página 79). Esto hace que se mantenga clara la finalidad y el desarrollo del programa.

¡DECISIONES, DECISIONES!

La ciencia-ficción otorga frecuentemente a los ordenadores el poder de razonamiento de la mente humana. Es posible que, como la mayoría de las personas, pensáramos antes que los ordenadores son más 'inteligentes' de lo que son en realidad.

Pero si bien no pueden pensar como las personas, sí que pueden tomar decisiones y comparaciones muy simples.

MPARACIONES
ANDO IF/THEN

Podríamos empezar por las comparaciones. ¿Recordamos los operadores relacionales mencionados en la parte 1 del manual? Son la clave de la manera en que los ordenadores realizan comparaciones y, a veces, toman decisiones basadas en las mismas.

Probemos este programa. Nos indicará el modo de usar estos operadores relacionales para decidir si dos series son o no diferentes.

```
100   LET THAT$="kettle"
110   INPUT PROMPT "Please type in some letters:
      ": THIS$
120   ! ------------------------------------------
130   !        100 and 110 should be familiar.
140   !        The two strings are compared in
150   !        220 and 230, using <> which
155   !        means 'different from'. If the two
160   !        are the same, both will be printed.
170   !        If they are different, "kettle" is
180   !        printed once.
190   ! ------------------------------------------
200   PRINT
210   PRINT
220   IF THIS$=THAT$ THEN PRINT    THIS$;" = ";
      THAT$
230   IF THIS$<>THAT$ THEN PRINT THAT$
240   ! ------------------------------------------
250   !        220 and 230 use IF/THEN to make a
260   !        decision. IF THIS$ is the same as
265   !        THAT$ THEN put both of them on
270   !        the screen. If it is not the same,
290   !        print THAT$.
295   ! ------------------------------------------
300   END
```

100 "Cafetera"
110 "Por favor, escriba algunas letras:"

130 140 150 155 160 170 180 Ya estamos familiarizados con el 100 y el 110. Las dos series se comparan en 220 y 230 usando lo que significa 'diferente de'. Si las dos son iguales, se imprimirán ambas. Si son diferentes, se imprime "cafetera" una vez.

250 260 265 270 290 220 y 230 utilizan IF/THEN para tomar una decisión. THIS$ es igual que (THAT$) THEN (entonces) se ponen ambas en pantalla. Si no es igual, se imprime THAT$.

IF/THEN es una de las dos maneras principales con las que el Enterprise puede tomar una decisión. Es una de las sentencias más similares a la notación inglesa del BASIC. Se utiliza con operado-- res lógicos

para observar las variables y números y comprobar si cumplen determinadas condiciones. IF/THEN no se usa para realizar cálculos sobre variables, sino para redirigir un programa SI (IF) es verdad algo respecto un número o una serie.

Por ejemplo, podemos escribir un programa para jugar a un juego en el que a cada jugador sólo le toca una serie de veces por vuelta. Por lo tanto, dejemos aparte una variable que sube en uno cada vez que un jugador ha completado una vuelta.

A continuación, cuando la variable es igual al número de vueltas que se concede al jugador, se puede usar una sentencia IF/THEN para indicar al programa que devuelva la variable a Ø para el jugador siguiente, informe al jugador actual que ha terminado su vuelta, registre su puntuación y pase al jugador siguiente, etc.

BLOQUES IF

Una sentencia IF/THEN no tiene que tener sólo una línea de longitud. Se puede utilizar también lo que se denomina bloque IF. Son simplemente varias líneas que permiten al ordenador efectuar varias comparaciones o decisiones. Significa también que se pueden utilizar varias líneas para tratar cada condición. Esto nos da un gran campo para la toma de decisiones.

Las líneas 22Ø y 23Ø del programa anterior podrían haberse escrito de manera diferente. Las líneas siguientes se pueden usar para sustituir las 22Ø y 23Ø del programa original. Tendrían el mismo efecto.

```
220   IF THIS$=THAT$ THEN
224         PRINT THIS$;"=";THAT$
230   ELSE
232         PRINT THAT$
236   END IF
```

La razón es que el ordenador sólo tenía dos alternativas. O las dos series eran iguales o no lo eran. Por tanto, las líneas anteriores dicen 'Si THIS y THAT$ son iguales, imprime ambos. Si es verdadera cualquier otra cosa, imprime sólo THAT$'. Se trata de un ejemplo de un pequeño bloque IF que sólo contiene una sentencia IF, una sentencia THEN, otra ELSE (por razones evidentes sólo puede colocarse una sentencia ELSE en un bloque IF), y las palabras END IF, que comunican al ordenador que no tiene que hacer más comparaciones.

ELSE significa exactamente lo que dice: Cualquier otra cosa. Esto

significa que se puede utilizar IF/THEN para decir al ordenador que realice determinadas cosas si esto y aquello es 'verdad' y usar entonces ELSE para tener en cuenta las otras posibilidades. Debe estar siempre en una línea independiente, nunca en la misma línea, como una sentencia IF/THEN.

El ejemplo siguiente es un bloque IF. Usa también una función numérica que hemos visto antes (página 5-6).

```
100   RANDOMIZE
110   LET A=RND(5)
120   !  ------------------------------------
130   !  100 and 110 make numbers out of the
140   !  blue! These can be used to make
150   !  patterns on the screen, to play games
160   !  like roulette or Simon (the flashing
170   !  sequences game) or many other things.
180   !  Here we'll use the random numbers to
190   !  draw shapes on the screen. The
200   !  commands are explained in the chapter
210   !  on graphics. The word RND is the one
220   !  which makes the computer produce a
230   !  random number. RANDOMIZE makes
240   !  sure the number is different each time
250   !  the program is run.
260   !  ------------------------------------
270   GRAPHICS
280   PLOT 600, 300;
290   OPTION ANGLE DEGREES
300   PLOT ANGLE 0;
310   IF A=1 THEN
320     PLOT FORWARD 100;
330     PLOT LEFT 90;
340     PLOT FORWARD 100;
350     PLOT LEFT 90;
360     PLOT FORWARD 100;
370     PLOT LEFT 90;
380     PLOT FORWARD 100
390     !  ------------------------------------
400     !  270 to 300 prepare the computer for line
410     !  drawing. In this example, some very
420     !  simple graphics commands are used.
430     !  LEFT gives turns; FORWARD is followed
440     !  by a number of screen positions
450     !  (measured according to the graphics
460     !  conventions). Lines 320 to 380 draw a
```

130 ..... al 250 ¡100 y 110 significan números como llovidos del cielo! Se pueden usar para hacer dibujos en la pantalla, para juegos como la ruleta o el Simon (el juego de las luces que se encienden alternativamente) u otras muchas cosas. Aquí utilizamos los números aleatorios para dibujar algunas formas en la pantalla. Los comandos se explican en el capítulo sobre gráficos. La palabra RND es la que hace que el ordenador produzca un número aleatorio. RANDOMIZE asegura que el número sea diferente cada vez que se ejecuta el programa.

400 ..... al 465 270 a 300 preparan el ordenador para dibujar líneas. En este ejemplo se usan algunos comandos gráficos muy simples. LEFT permite girar; FORWARD va seguido por una serie de posiciones de la pantalla (medidas según las convencione de los gráficos). Las líneas 320 a 380 dibujan un cuadrado.

```
465     !  square.
470     !  ------------------------------------------
480   ELSE IF A=2 THEN
490     PLOT FORWARD 50;
500     PLOT LEFT 90;
510     PLOT FORWARD 100;
520     PLOT LEFT 90;
530     PLOT FORWARD 50;
540     PLOT LEFT 90;
550     PLOT FORWARD 100
560     !  ------------------------------------------
570     !  490-550 draw a rectangle.                    57Ø  49Ø-55Ø dibujan un rectángulo
580     !  ------------------------------------------
590   ELSE IF A=3 THEN
600     PLOT FORWARD 100;
610     PLOT LEFT 120;
620     PLOT FORWARD 100;
630     PLOT LEFT 120;
640     PLOT FORWARD 100
650     !  ------------------------------------------
660     !  600-640 draw a triangle.                     66Ø  6ØØ-64Ø dibujan un triángulo
670     !  ------------------------------------------
680   ELSE
690     TEXT                                          7ØØ  "No tengo instrucciones para números"
700     PRINT "I have no instructions for number"; A
710   END IF
720   END
```

Ahora que hemos tenido la posibilidad de jugar con los gráficos del Enterprise, volvamos a IF/THEN. Como podemos ver, se pueden usar todos estos números y las palabras del BASIC para ayudarnos a dibujar gráficos... y para conseguir también efectos musicales y de sonido. Todo lo que necesitamos hacer es combinar el BASIC 'ordinario' con los comandos gráficos.

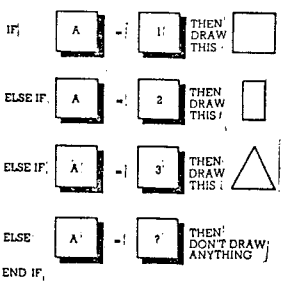
El programa anterior realiza su propio número aleatorio usando una fórmula (nunca tendremos que preocuparnos de la manera como lo hace, salvo que estemos realmente interesados en el funcionamiento interno del ordenador), para producir números 'de la nada'. Por cierto, si lo deseamos, podemos imprimir una secuencia de estos números aleatorios usando DO/LOOP, y será imposible ver el modelo en que sigue la secuencia.

A continuación, según el número producido por RND, el ordenador dibujará una de las tres formas posibles. Si el número no es uno de los mencionados en la línea IF, ejecutará por el contrario la

sentencia ELSE solitaria (línea 68Ø) y visualizará en la línea 7ØØ, que es en realidad un poco pobre si se compara con el resto del programa. Esto es lo que se intenta con el uso de IF/THEN/ELSE para reunir condiciones más complejas poniéndolas en varias líneas -un bloque - en vez de en una o dos.

El programa anterior muestra exactamente el uso de un bloque IF. IF y ELSE separan cada una el posible camino a seguir.

Un bloque IF debe terminar siempre con END IF - ya que en caso contrario el ordenador no ejecutará el resto del programa si ha fallado la última prueba IF.



SI ENTONCES DIBUJA ESTO

O SI ENTONCES DIBUJA ESTO

O SI ENTONCES DIBUJA ESTO

O ENTONCES NO DIBUJES NADA

IF/THEN es, por tanto, interesante como línea para recoger las excepciones o tratar de las decisiones cuando como resultado sólo se necesita hacer algo simple. Como bloque, se puede usar para ejecutar un programa complejo completo, según la decisión tomada al principio a través del bloque IF.

ELECT CASE Con SELECT CASE se pueden efectuar comparaciones y decisiones similares a las ofrecidas por IF/THEN. CASE significaría 'en caso de...', en vez de 'si...entonces..., o bien...':

En el ejemplo que sigue, observamos la falta del nombre de variables en las sentencias CASE. CASE 1,2,3 es correcto y CASE X= 1,2,3 es erróneo: la variable que ha de probarse ha sido dada ya por la línea SELECT. El uso de CASE para tomar una decisión entre varias alternativas, es, como es fácil comprobar, más breve y claro que el uso de IF/THEN.

Como ocurre con END IF, se debe marcar el final del bloque SELECT, en este caso escribiendo END SELECT.

```
100   CLEAR SCREEN
105   !  ------------------------------------------
110   !     Line 100 clears the screen. Then
111   !     a menu is printed in the middle
115   !     of the screen, using PRINT AT to
118   !     position the letters. After that,
120   !     you are asked to type in your
122   !     choice, and the number you choose
125   !     is put into a variable called A.
127   !     If you type in a number bigger
130   !     than 3 or less than 1, the program
133   !     just asks you to do it again.
135   !     410 deals with that using IF/THEN.
136   !     It also refuses to allow numbers
137   !     with a decimal point. A <> INT(A)
138   !     makes this possible.
140   !  ------------------------------------------
150   PRINT AT 9,18:"MENU"
200   PRINT AT 11,10:"1) Print my name"
250   PRINT AT 12,10:"2) Print your name"
300   PRINT AT 13,10:"3) Print both of our names"
360   DO
400         INPUT AT 16, 10, PROMPT "Please enter
            your choice: ":A
410   LOOP WHILE A < 1 OR A > 3 OR A <> INT(A)
450   CLEAR SCREEN
460   !  ------------------------------------------
465   !     The program from Lines 500 to
466   !     1000 makes the decision about
467   !     what to do with your choice, and
468   !     does it. The numbers after the
470   !     word CASE each time are the
472   !     numbers which A can be equal to.
473   !     For instance, 550 and 600 say,
474   !     'In the case of 'A' being 1,
475   !     print "ENTERPRISE!!!" in the
476   !     middle of the screen.' 1000 is
477   !     essential to tell the machine no
479   !     more cases can be expected.
480   !  ------------------------------------------
500   SELECT CASE A
550   CASE 1
600    PRINT AT 9,18: "ENTERPRISE!!!"
```

100 ..... a 138 La línea 100 borra la pantalla. Entonces se imprime el menú en la mitad de ella, usando PRINT AT para colocar las letras. A continuación, el ordenador nos pide que escribamos nuestra elección, y el número escogido se pone en una variable llamada A. Si escribimos un número mayor de 3 o menor de 1, el programa simplemente nos pide que repitamos. 410 sirve para esto, usando IF/THEN. También rechaza los números con coma decimal. A <> INT(A) hace esto posible.

200 "1) Imprime mi nombre"
250 "2) Imprime tu nombre"
300 "3) Imprime nuestros nombres"
400 "Por favor, introduce tu elección:"

465 ..... a 479 El programa de las líneas 500 a 1000 toma la decisión sobre lo que hay que hacer con la elección y la ejecuta. Los números que van después de la palabra CASE en cada ocasión, son los números a los que puede ser igual A. Por ejemplo, 550 y 600 dicen, 'En el caso de que 'A' sea 1, imprime "ENTERPRISE!!!" en medio de la pantalla.' 1000 es esencial para indicar a la máquina que no pueden esperarse más casos.

```
650   CASE 2
700     INPUT PROMPT "Then please tell me your
        name. ":NAME$
750     PRINT AT 9,18:NAME$
800   CASE 3
850     INPUT PROMPT "Then please tell me your
        name. ":NAME$
900     PRINT AT 9,18:NAME$
950     PRINT AT 11,18: "Enterprise"
1000  END SELECT
1010  !  ------------------------------------
1015  !    Lines 1200 to 1350 are very similar
1016  !    to the check on 'A' at the
1020  !    beginning of the program. They
1022  !    make sure that the program will
1025  !    only end or go back to the
1027  !    beginning if the first letter of
1030  !    A$, converted to a capital, is
1035  !    either "Y" or "N". GOTO 100 tells
1037  !    the computer to jump back to the
1040  !    beginning of the program only if
1041  !    the first letter of A$ (converted
1043  !    to a capital) is "Y". Otherwise
1045  !    the program ends—which, as you
1046  !    can see, it can only do if the
1047  !    first letter of A$ is "N".
1048  !  ------------------------------------
1050  PRINT
1100  PRINT
1150  PRINT
1200  DO
1250        INPUT PROMPT "Would you like to
          do that again? ":A$
1300  LOOP UNTIL UCASE$(A$(1:1))="Y" OR
      UCASE$(A$(1:1))="N"
1350  IF UCASE$(A$(1:1))="Y" THEN GOTO 100
1400  PRINT
1450  PRINT
1500  PRINT
1550  END
```

700 "A continuación dime tu nombre"

850 "A continuación dime tu nombre"

1015 ..... a 1047 Las líneas 1200 a 1350 son muy similares a la verificación de 'A' al comienzo del programa. Se aseguran de que el programa termine o regrese al comienzo únicamente si la primera letra de A$, convertida a mayúscula, es "Y" o "N". GOTO 100 ordena al ordenador que vuelva al comienzo del programa únicamente si la primera letra de A$ (convertida en mayúscula) es "Y". En caso contrario, el programa termina, lo cual, como es fácil comprobar, sólo puede hacerse si la primera letra de A$ es "N".

1250 "¿Quiere repetirlo de nuevo?".

Intentemos escribir un programa que simule un dado. Puede hacerse con el uso de IF/THEN o SELECT CASE. Utilizar un número aleatorio como en el programa de la página 66. Natualmente, tendría que estar entre 1 y 6. Así

```
RND(6)+1
```

se encargaría de esta tarea. Sabiendo que tenemos 6 resultados posible no debe resultarnos difícil. Podríamos también usar los gráficos para poner en pantalla una imagen del dado.

Recuérdese que el uso de IF/THEN como bloque IF, no significa que tengamos que usar ELSE. Las líneas ELSE son opcionales, al igual que el número de posibilidades mencionadas dentro del bloque o el número de selecciones CASE.

CASE puede tener también 'claúsula ELSE'. Es CASE ELSE, al igual que CASE 1 o CASE "HELLO". Observamos que se puede utilizar también CASE con series, y que una ventaja sobre IF/THEN es su capacidad para reunir varias 'verdades' con el mismo resultado, algo así:

```
CASE 1,3,5
PRINT "These are odd numbers."                    "Estos son números impares".
```

Por último, experimentemos con los códigos ASCII (ver 'El Conjunto de Caracteres' página 104). Pueden utilizarse para que el ordenador ponga las series en orden alfabético. Los codigos ASCII sólo son números que representan caracteres. El uso de matrices (pronto hablaremos de ellas) es también útil en este tipo de tarea de programación.

ALMACENAMIENTO DE CANTIDADES MAYORES DE INFORMACION

Como hemos podido descubrir, la programación es fundamentalmente una manera de manejar información. Hasta ahora, hemos usado únicamente pequeñas cantidades de información en programas, frases o números que hemos escrito en respuesta a las sentencias INPUT, o frases que el ordenador ha presentado en pantalla para que las leyéramos.

Es probable que nos hayamos pregundado qué ocurriría con cantidades mayores de información: una lista larga de palabras o números, tal vez, o varios párrafos de instrucciones de un juego o de un programa para manejar nuestra economía privada. El Enterprise proporciona varios sistemas eficientes para manipular listas de números o grupos grandes de palabras.

La manera de conservar con programa una lista de nombres, por ejemplo, sería utilizanto una matriz. La matriz es como una gran variable con cajas cuyo contenido puede cambiar. La matriz puede ser considerada como una caja grande en la que se guardan una serie de cajas pequeñas. Podemos también contemplarla como una página de un cuaderno en la que ponemos una lista.

ATRICES
UMERICAS

Como ocurre con las variables, tenemos matrices de serie y otras numéricas. Por ello, para declarar una matriz numérica utilizaríamos (por ejemplo):

```
NUMERIC STORE(1 TO 1Ø)
```

Escribimos esto en el ordenador como línea 1ØØ. Indica a la máquina que debe dejar aparte un 'recipiente' llamado STORE con espacio para 1Ø variables menores en él. Estas variables se conocen como elemento de la matriz, y podemos registrar en estos pequeños espacios todos los números que queramos. La séptima variable (o elemento) dentro de STORE se llamaría STORE(7).

Escribamos ahora las líneas siguientes:

```
11Ø  FOR S=1 TO 1Ø
12Ø      INPUT PROMPT "Enter a number ": STORE(S)  12Ø  "Introduzca un número"
13Ø  NEXT S
```

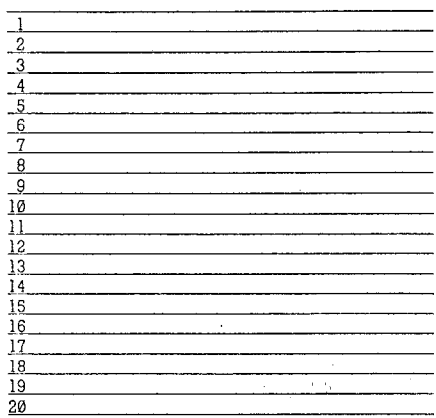
Esto nos permite poner números en la matriz. Al ejecutarlo, escribir PRINT STORE(9) o (8) o cualquier otro número de 1 a 1Ø. Como puede verse, podemos llamar a los números de la matriz. En el ordenador hemos puesto

una lista de números. Si quisiéramos registrar las temperaturas diarias del mes de Diciembre, lo haríamos en una matriz llamada DECEMBER con elementos numerados de 1 a 31. La temperatura de cada día formaría el contenido de un elemento, y los números de elementos significarían las fechas.

Una matriz podría también consistir en elementos numerados de 56 a 76, de 123 a 171, e incluso de 12345 a 12445. Si se quiere, puede usarse un número negativo para los extremos superior e inferior de la serie de elementos: por ejemplo, NUMERIC TABLE(-1Ø TO 1Ø), o NUMERIC TABLE(-2Ø to -1Ø). Los números de elementos sólo sirven para referencia

MATRICES BI-DIMENSIONALES

Una matriz puede ser más compleja que una simple lista. El esquema de abajo y el de la página siguiente muestran la diferencia entre las matrices unidimensionales y bidimensionales.



g. 74 La matriz bidimensional puede representarse en forma de cuadrícula. O (si se quiere), puede imaginarse que esta vez guardamos las cajas en un armario en el que cada estatería tiene un número y hay sitio para el mismo número de cajas en cada una.
A(1 TO 4, 1 TO 4) produciría una matriz así:

1.1	1.2	1.3	1.4
2.1	2.2	2.3	2.4
3.1	3.2	3.3	3.4
4.1	4.2	4.3	4.4

Un ejemplo del uso de este tipo de matriz es el tablero de algunos juegos. La matriz podría usarse para almacenar información sobre las piezas que hay en cada cuadrícula del tablero. Podría usarse también una matriz similar como tabla para una serie de palabras: un tipo más versátil de lista.

ATRICES DE
ERIES Las matrices de serie se declaran de la misma manera que las numéricas, salvo que se usa la palabra STRING. Con ella se puede también modificar la longitud de cada elemento: algo que no puede ocurrir con la numérica.

Modificar la longitud de los elementos de una matriz sólo significa modificar la mayor cantidad que pueden contener estos elementos. Al igual que no es imprescindible llenar un cubo hasta los bordes cada vez que cogemos agua con él, tampoco hay que rellenar totalmente los elementos de una matriz.

Como ya se sabe, el Enterprise tiene en su interior una cierta

cantidad de espacio, en la que podemos almacenar programas y la información que utilizará con ellos.

Cuando se declara una matriz de serie, el ordenador conservará para cada elemento una cierta cantidad de espacio. A esto se le llama MAXLEN. Salvo que se especifique otra cosa, MAXLEN tiene 132 caracteres. Esto no afecta a los números sino que sólo se refiere a las series.

Si quisiéramos conservar espacio en la memoria de nuestro ordenador - o aceptar elementos de más de 132 caracteres - habría que decírselo al ordenador. Se haría así:

```
STRING ARRAY$(30 TO 50)*10
```

Esta sentencia BASIC apartaría una matriz con 20 elementos en ella (numerados del 30 al 50), con 10 caracteres de longitud cada elemento. MAXLEN(ARRAY$(45)) sería entonces 10. Cuanto más largos sean los elementos, más puede ponerse en ellos, pero ocuparán más espacio de memoria y dejarán menos para otras cosas. Por tanto sólo deben hacerse mayores de lo normal si es realmente necesario, o si se tiene la seguridad de que se cuenta con espacio suficiente para ellos. Si se amplia la memoria del Enterprise, podrá naturalmente almacenarse programas mucho mayores y colocar mucha más información.

ECLARACION
E VARIABLES

Las variables simples pueden también declararse usando STRING o NUMERIC. NUMERIC A declara una variable numérica. NUMERIC A,M,N,H,D o algo similar podría usarse para declarar todas las variables numéricas al mismo tiempo: al comienzo del programa.
STRING A$,B$,HELLO$*8... haría lo mismo con las variables de serie.

La única diferencia en declarar variables como ésta y declarar una matriz es que, en este último caso, se especifican los elementos.

EAD/ DATA

Muchas palabras del BASIC van siempre unidas a otras del mismo lenguaje. READ y DATA (como también la palabra RESTORE) son ejemplos.

Se mencionan en esta parte del manual porque representan otra manera más de conservar cantidades importantes de información dentro de un programa. Representan igualmente otra manera de poner los números o series en una matriz. READ es la palabra que realiza todo el trabajo duro. He aquí un breve ejemplo.

```
 90    DO UNTIL P=0
100        READ P
110        PRINT P,
120    LOOP
130    DATA 1,2,3,4,5,6,7,8,9,10
150    DATA 11,12,13,14,15,16,17,18
160    DATA 19,20,21,22,23,24,25,0
170    END
```

Las sentecias DATA mantienen simplemente la información. Cada elemento de información debe ir separado del siguiente por una coma que indica que ha terminado un elemento de información y continua el siguiente. READ indica al ordenador que busque un elemento DATA y haga con él lo que nosotros le digamos. En este caso, el elemento se coloca en la variable P y a continuación se visualiza en pantalla el contenido de P.

Observemos que para todos los elementos DATA se utiliza el mismo nombre de variable. Esto no importa; el mismo se utiliza una y otra vez. Por tanto, el programa READx (lee) un elemento DATA, lo pone en la variable llamada P, lo pasa a pantalla, lee el siguiente, lo pone en P en lugar del último elemento DATA, y así sucesivamente. El 0 al final de la última sentencia DATA se utiliza para indicar al ordenador que no siguen más datos.

El ordenador sólo leerá cada elemento de datos una vez. Después de leído un elemento, la máquina 'recuerda' su posición y pasa al siguiente (leyendo de la izquierda a la derecha y de arriba a abajo como se lee un libro). Una vez que los ha leido todos, considera que allí ya no hay más ... y queda confundido si le decimos que continue buscando.

El programa siguiente demuestra el uso de READ/DATA con series.

```
150    CLEAR SCREEN
200    PRINT "I'm going to tell you a story."
250    PRINT
300    PRINT "Here goes!"
350    PRINT
400    PRINT
410    FOR X=1 TO 5000
415    NEXT X
420    !
430    !    Lines 410 and 415 merely specify an
```

```
200    "Voy a contarte un cuento"
300    "¡Ahí vá!"
430    Las líneas 410 y 415 especifican
       simplemente un
```

```
440   !     interval. The DO/LOOP (550-810 is the
450   !     important part. Let's go round it and se
460   !     what happens. First it READs A$—whi
470   !     means it looks for a DATA statement
480   !     (which can be anywhere in the prograr
490   !     reads it and checks to see if
500   !     it's "END". If it is, the computer
515   !     stops looping and goes to
520   !     the rest of the program. If not, 700
525   !     tells the computer to print the
530   !     string from the DATA statement
535   !     on the screen (and put a space after it).
540   !     ------------------------------------------
550   DO
600         READ A$
650         IF A$="END" THEN EXIT DO
700         PRINT A$;" ";
750         FOR Y=1 TO 500
760         NEXT Y
810   LOOP
820   !     ------------------------------------------
830   !     The rest of this program may look
840   !     confusing. If you remember that the
850   !     DATA statements are read by an
855   !     earlier part of the program, and that
870   !     900-1100 are actually performed
873   !     after the DATA has been read and
875   !     put on the screen, the logic of
880   !     the program will fall into place.
890   !     ------------------------------------------
900   PRINT
950   PRINT
1000  PRINT "                    THE END"
1050  PRINT
1100  PRINT
1125  END
1150  DATA Once,upon,a,time,there,was,a,
      little,computer,
1200  DATA called,the,Enterprise.,It,was,
      a,very,
1250  DATA happy,computer.,All,the,best,
      programmers,in,
1300  DATA a,land,called,England,had,worked,
      all,day,
1350  DATA and,all,night,for,months,to,make,
      the,Enterprise,
```

440 ..... a 535 intervalo. El DO/LOOP (550-810) es la parte importante. Veamos qué sucede. Primero lee A$, lo que significa que busca una sentencia DATA (que puede estar en cualquier lugar del programa), la lee y comprueba si es "END". Si lo es, el ordenador suspende el bucle y pasa al resto del programa. Si no lo es, 700 indica al ordenador que imprima en pantalla la serie de la sentencia DATA (y ponga un espacio después de ella).

830 ..... a 880 El resto de este programa puede parecer complicado. Si se recuerda que las sentencias DATA son leídas por una parte anterior del programa, y que 900-1100 son ejecutados realmente después de leídos y puestos en pantalla los datos, se verá con claridad la lógica del programa.

1150 Erase, una, vez, había, un, pequeño, ordenador

1200 llamado, el, Enterprise, Era, un

1250 ordenador, muy, feliz, Los, mejores, programadores, del

1300 país, llamado, Inglaterra, habían, trabajado, de, día

1350 y, de, noche, durante, meses, para, fabricar, el, Enterprise

```
1400  DATA the,best,computer,ever.,Today,
      you're,learning,
1450  DATA to,write,BASIC,on,the,Enterprise.,
      Aren't,you,
1500  DATA lucky?,END
```

```
14ØØ  el, mejor, ordenador, de, todos, los,
      tiempos, Hoy, estás, aprendiendo,
145Ø  a, escribir, el, BASIC, en, el,
      Enterprise, ¿No, eres,
15ØØ  afortunado?, END
```

En este programa podrían introducirse dos posibles modificaciones. Una es sustituir la línea 65Ø, que impide que el ordenador continue y ejecute el DO/LOOP (Líneas 55Ø a 81Ø) cuando no hay más datos que leer. Si cambiamos la línea 65Ø por IF MISSING THEN EXIT DO, el ordenador no necesitará el END al final de los datos de la línea 15ØØ. Ahora podemos retirar esto. IF MISSING significa simplemente 'si no hay más datos...'. Se usa para decir al ordenador que continue con el resto del programa cuando llegue al final de los datos.

A continuación, se puede intentar añadir a este programa 1ØØ DO y 112Ø LOOP. A continuación podemos ejecutarlo de nuevo. Como se verá ahora, trabaja muy bien una vez y acto seguido pasa un mensaje para decir que no hay más datos. Por tanto, añadimos la línea 111Ø RESTORE. Esto indica al ordenador que debe represar y utilizar de nuevo los datos. Si se desea, puede ponerse RESTORE 14ØØ, que ordena a la máquina el uso de los datos que vienen en la línea 14ØØ y siguientes. Esto nos permite elegir una parte de los datos y utilizarla varias veces si lo deseamos.

No hay que confundirse por el hecho de que la serie DATA no necesite comillas. El hecho de que haya que indicar al ordenador que lea A$ o X$ le indica que busque unas series: la clave aquí es el signo del dólar. En efecto, sería muy aburrido si tuviéramos una larga lista de palabras para incluir como datos, y tuviéramos que poner en todas ellas las comillas: considérelo como una suerte.

Lo mismo se aplica a las sentencias INPUT: puede escribirse 'si' o 'no' en respuesta a una pregunta planteada por una SOLICITUD DE ENTRADA (INPUT PROMPT), pero no es necesario escribir las comillas. El 'dólar' al final del nombre de la variable dice al ordenador que debe aceptar una serie.

Sin embargo, es necesario colocar las comillas antes y después de un elemento DATA o INPUT si queremos incluir una coma en la serie, ya que, de lo contrario, el ordenador pensaría que la coma marca el final de la serie.

DEFINICION DE FUNCIONES

Una función es una especie de 'programa dentro de un programa', destinada a realizar alguna tarea específica: una secuencia de instrucciones que se reserva para utilizar cuando se necesite y que puede ser utilizada una y otra vez.

como ejemplo simple, supongamos que se desea que aparezca en pantalla un determinado mensaje en diversas etapas de un programa. Esta acción podría ser definida como función, escribiendo algo así:

```
100     DEF WARNING
110         CLEAR SCREEN
120         PRINT AT 10,7: "NOW PAY CLOSE
            ATTENTION..."
130         !
140         SOUND !          Line 140 adds a
150         !                sound signal.
160         !
170     END DEF
```

14Ø La línea 14Ø añade una señal sonora.

A la función hay que darle siempre un nombre (en este caso WARNING), y se introduce con la palabra clave DEF. La definición que sigue entonces podría tener una o mil líneas de longitud, pero sea cual fuere su longitud debe tener las palabras END DEF como la última línea.

LLAMADA A LAS FUNCIONES

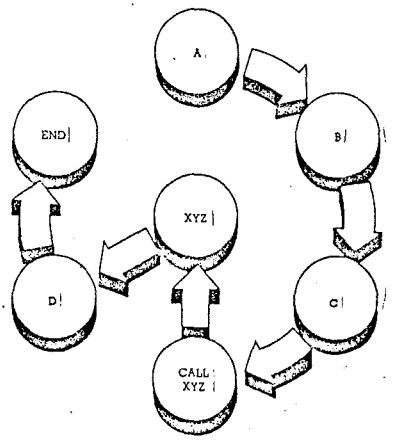
La función no puede trabajar por sí misma. Si se escriben las líneas anteriores y se intenta ejecutarlas, por el momento no sucederá nada. La función debe ser activada con la sentencia CALL WARNING. Escribámosla como línea 18Ø y ejecutemos el programa escribiendo después la misma instrucción en modalidad inmediata. Obsérvese que la definición de la función puede venir después de la sentencia CALL o incluso después de la sentencia END del programa. Si, en lugar de la línea 18Ø, se escribe:

```
80      CALL WARNING
90      END
```

la función seguirá sirviendo.

Cuando el ordenador llega a una sentencia CALL, detiene cualquier cosa que esté haciendo, encuentra la función que es llamada, la ejecuta y vuelve después al punto del programa que está inmediatamente después de

CALL. En el diagrama se verá esto con mayor claridad



Recuérdese que una función permanece inactiva mientras no le digamos realmente al ordenador que la ejecute. Si el ordenador está simplemente siguiendo la secuencia de números de línea y llega a la parte de un programa en la que está colocada una función, la salta y realiza todo lo que haya después. No se puede conseguir que trabaje una función sin utilizar su nombre en otro lugar del programa.

ARIABLES OCALES Y OBALES

Normalmente las funciones manejan variables. Para conseguir que lo hagan correctamente, es preciso respetar algunas reglas importantes.

Probemos a escribir esto:

```
100     DEF CUBE
110         INPUT PROMPT "Number to be
            cubed: ":Z                    110  "Número que ha de ser elevado al
120         PRINT Z;" cubed is ";Z*Z*Z            cubo:"
130     END DEF
140     CALL CUBE
```

y después de ejecutarlo, añadamos:

```
150    PRINT Z
```

y volvamos a ejecutarlo. Veremos entonces que aunque sigue trabajando la función CUBE, el ordenador presenta un mensaje de error al llegar a la línea 15Ø. ¿Por qué ocurre esto?

La respuesta es que, en este caso, Z es lo que se conoce como variable local. Pertenece exclusivamente a la función CUBE, y la parte del programa que está fuera de esa función no sabe nada de ella. Dado que el ordenador trata a la función como un pequeño programa separado, podría utilizar sus propias variables privadas para ayudarle a realizar su tarea. Pero estas variables privadas no significan nada para el resto del programa; una vez completada la tarea, sus valores quedan eliminados. Así, en la línea 15Ø anterior, el ordenador no sabe lo que debe imprimir.

A continuación, cambiemos el número de la línea 11Ø por el 9Ø: para ponerla fuera de la función. Veremos ahora que el programa nos permite escribir un número para Z, nos indica el cubo de este número y a continuación vuelve a imprimir el mismo número. Es decir, la línea 15Ø ya no lo confunde.

La razón es que al introducir Z antes de que se llame a la función, se ha convertido la Z en una variable global. Una variable 'global' es la que está a disposición del 'mundo' general del programa.

Añadamos ahora:

```
125    LET Z=2Ø
```

y ejecutemos otra vez el programa. Lo que sucede ahora es que la función saca el número de la 'caja' (marcada Z) de la parte 'principal del programa, realiza con él un cálculo e imprime resultado, a continuación modifica el mismo número y lo devuelve a la misma caja que antes. El 'programa principal' imprime entonces este nuevo número.

Lo que conviene recordar es que si una función contiene una línea que menciona una variable, y esta variable no ha sido presentada antes de llamar a la función, ésta la tratará como variable local o privada. Si, por otro lado, la variable ha sido ya declarada, la función la considerará como 'global'; cualquier valor nuevo que de la función se pasará al resto del programa.

Hemos leído ya algo sobre la declaración de valores en

partes anteriores del manual. (Si hay que revisarlo, está en la página 24 y 75). Se dijo que, aunque no siempre es esencial, es mejor declarar cada variable que se utilice. Naturalmente, la declaración de las variables es especialmente importante si utilizamos frecuentemente funciones.

Como se sabe, una variable puede declararse con una sentencia NUMERIC o STRING (NUMERICA O DE SERIES) o con la palabra LET (por ejemplo, LET A=Ø). En los ejemplos anteriores, no importaba gran cosa cuál de estas formas de declaración se usaba; pero sus efectos deben ser bien entendidos cuando trabajamos con funciones. Dentro de una función, una sentencia NUMERIC (o STRING) tiene siempre el efecto de crear una variable local. En el programa anterior (después de cambiar el número de la línea original 11Ø), probemos a añadir:

```
11Ø    NUMERIC Z
115    LET Z=3
```

Se verá entonces que el programa actúa con dos Z totalmente separadas, una dentro de la función (una Z 'local') y otra fuera (una 'global'). Por otra parte, si suprimimos la línea 11Ø, sólo habrá una Z. La secuencia LET de la línea 115 no creará una variable nueva (local) pero modificará la variable global introducida por la línea 9Ø.

El programa siguiente contiene algunos ejemplos bastante complicados de funciones. Es una versión reestructurada de un programa anterior que apareció en el capítulo sobre decisiones. Aparte de mostrar cómo aparecen las funciones dentro de un programa, mostrará también que hay siempre varias maneras de montar un programa. Algunas pueden parecer bonitas, otras horribles, unas incomprensibles, otras muy eficientes o todo lo contrario. Si observamos el programa globalmente, es probable que estemos de acuerdo en que esta versión es mucho más clara.

Mientras queramos que el ordenador imprima más nombres en la pantalla, no terminará el programa. Para terminarlo, habrá que escribir 'N' cuando se nos pregunte.

```
1Ø0    DO
11Ø        LET A=Ø
12Ø        LET A$=""
13Ø        !
14Ø        !    Lines 11Ø and 12Ø declare two
```

14Ø Las líneas 11Ø y 12Ø declaran dos

```
150         !   'global' variables, which the func-
160         !   tions will use (and alter) and then
170         !   hand back to the main program.
180         !   ------------------------------------
190         CALL MENU
200         FOR X=1 TO 1500
210         NEXT X
220         !   ------------------------------------
230         !   Then comes the main program,
240         !   which begins by calling the menu,
250         !   and, after the menu has finished,
260         !   delays for 3 seconds to give you
270         !   time to read the screen. After you
280         !   have made your choice from the
290         !   menu, the main program goes
300         !   through the CASE block, does what
310         !   you have chosen, and then calls the
320         !   ANSWER function which decides
324         !   whether or not the program will run
327         !   again.
330         !   ------------------------------------
340         CLEAR SCREEN
350         SELECT CASE A
360         CASE 1
370             PRINT AT 9,18:"ENTERPRISE!"
380         CASE 2
390             INPUT PROMPT "Then please tell
                me your name. ":NAME$
400             PRINT AT 9,18:NAME$
410         CASE 3
420             INPUT PROMPT "Then please tell
                me your name. ":NAME$
430             PRINT AT 9,18:NAME$
440             PRINT AT 11,18: "ENTERPRISE!"
450         END SELECT
460         FOR X=1 TO 3000
470         NEXT X
480         CLEAR SCREEN
490         CALL ANSWER
500     LOOP WHILE A$="Y"
510     END
520     !   ------------------------------------
530     !       500 concludes the main loop. If A$
540     !       is "Y", the program goes back to
550     !       the beginning. In effect, the
560     !       program does not end until you
```

15Ø Variables 'globales' que utilizará (y modificarán) las
16Ø funciones y a continuación volver al programa principal
17Ø

23Ø ..... a 327 A continuación viene el programa principal, que comienza llamando al menú y, una vez terminado, espera tres segundos para dar tiempo a leer la pantalla. Una vez que se ha elegido en el menú, el programa principal continua con el bloque CASE, ejecuta el escogido y acto seguido llama a la función ANSWER que decide si se debe o no repetir el programa.

39Ø "Por favor, dime tu nombre"

42Ø "Por favor, dime tu nombre"

53Ø 54Ø 55Ø 56Ø 5ØØ pone fin al bucle principal. Si A$ es "Y", el programa vuelve al principio. En efecto, el programa no termina mientras no

```
570   !          reply with "N" (or no, or nope, etc.)
575   !          in the ANSWER function.
580   !
590   DEF MENU
600         CLEAR SCREEN
610         PRINT AT 9,18:"Menu"
620         PRINT AT 11,9:"1) Print my name."
630         PRINT AT 12,9: "2) Print your name."
640         PRINT AT 13,9:"3) Print both of our
            names."
650         PRINT AT 16,1: "Please enter the
            number of your choice: "
660         DO
670               INPUT A
680         LOOP WHILE A<1 OR A>3 OR
            A<>INT(A)
690   END DEF
700   DEF ANSWER
710         PRINT
720         PRINT
730         DO
740               INPUT PROMPT "Would you like
                  to do that again? ":A$
750               LET A$=UCASE$(A$(1:1))
760         LOOP UNTIL A$="Y" OR A$="N"
770   END DEF
```

570 575 Contestemos con "N" (o no, etc.) en la función ANSWER.

610 "Menú"
620 "1) Imprime mi nombre"
630 "2) Imprime tu nombre"
640 "3) Imprime los dos nombres"

650 "Por favor, introduce el número que has elegido"

740 "¿Te gustaría hacerlo de nuevo?"

Si retiramos las líneas 11Ø y 12Ø, el programa no servirá ya porque las dos variables cuyas declaraciones no hemos efectuado se han convertido en locales para las funciones que las contienen.

Hasta ahora hemos estado usando funciones que son activadas con sentencia CALL y pueden producir una serie de efectos, como invitarnos a escribir más datos o imprimir mensajes en pantalla. Veamos ahora una clase muy diferente de función: una que tiene simplemente como objeto el devolver un único número a la parte principal del programa.

Algunas funciones de este tipo son suministradas ya preparadas por el ordenador. Tomamos por ejemplo SQR. Una línea del programa puede contener la sentencia PRINT SQR(121), o PRINT SQR(P), o LET M = 2 * SQR(N)+1. La función SQR calcula la raíz cuadrada del número o variable entre paréntesis, y a continuación nos permite usar esa raíz cuadrada como parte de una 'expresión' o hacer cualquier

otra cosa que queramos con ella. Estamos también familiarizados con la función INT. Estas palabras del BASIC nos proporcionan los medios para realizar, rápida y fácilmente, cálculos que podemos necesitar con frecuencia.

Supongamos que estamos escribiendo un programa que utiliza varios números 'factoriales' (el factorial de 4 significa 4*3*2*1; el factorial de 6 es 6*5*4*3*2*1; etc.). No hay ninguna función preparada para calcular factoriales. Pero si quisiéramos, podríamos idear una con los métodos aprendidos hasta ahora. Escribiríamos:

```
100    DEF FACT
110    !
120    !                    This function will take a
130    !                    global variable, F, from
140    !                    the 'main' program, alter
150    !                    it and hand it back again.
160    !
170        FOR Y=F-1 TO 1 STEP -1
180          LET F=F*Y
190        NEXT Y
195    IF F=0 THEN LET F=1
200    END DEF
```

120 130 140 150 Esta función tomará una variable global, F, del programa 'principal', la modificará y la entregará de nuevo.

Y entonces, para poder imprimir (por ejemplo) el factorial de 13, o usar el 7 en una 'expresión', añadiríamos:

```
210    LET F=13
220    CALL FACT
230    PRINT F
240    LET F=7
250    CALL FACT
260    LET NUMBER=F*1.5+3
270    PRINT NUMBER
```

Pero esto, como es fácil comprobar, es bastante más complicado que usar una palabra BASIC como SQR, porque cada vez que se llama a la función FACT, el número sobre el que queremos que opere tiene que ser colocado primero en la variable F. No obstante, el ordenador nos ofrece la manera de superar esta limitación. Suprimamos todo lo anterior y escribamos en su lugar:

```
100    DEF FACT(X)
110        FOR Y=X-1 TO 1 STEP -1
120          LET X=X*Y
```

```
130         NEXT Y
135     IF F=0 THEN LET F=1   |
140         LET FACT=X
150     END DEF
160     PRINT FACT(13)
170     LET NUMBER=FACT(7)*1.5+3
180     PRINT NUMBER
```

Haciendo algo así, convertimos FACT en una especie de nueva palabra BASIC nuestra, que puede utilizarse de manera similar (y con la misma comodidad) que utilizaríamos INT o SQR.

Las dos cosas nuevas que aparecen en el programa anterior, son las líneas 1ØØ y 14Ø. La X entre paréntesis de la línea DEF dice a la función que busque un número entre paréntesis seguido por la palabra FACT en una línea del programa principal y ponga automáticamente ese número en su propia variable local X. En efecto, la línea 14Ø asigna un valor a una variable que tiene el mismo nombre que la función, y permite así que este valor pueda ser entregado a la línea del programa 'principal' en donde se menciona la función (ver líneas 16Ø y 17Ø. De este modo, se trabaja sin una sentencia CALL.

RIABLES
CTICIAS

En términos técnicos, la X entre paréntesis de la línea 1ØØ anterior se conoce como variable ficticia. Dice a la función que espere un número que se le entregará para su proceso. (El ejemplo final de este capítulo muestra una función con dos variables ficticias, que le dice que espere dos números). Sin embargo, ese número podría ser suministrado por una variable global del programa principal. Suprimir las líneas 16Ø-18Ø, y sustituirlas por:

```
160     LET A=11
170     PRINT FACT(A)
```

lo que sucede ahora es que la función busca en la casilla A y toma una nota (saca una copia) del número que ve allí. A continuación coloca un número idéntico en su propia casilla X, que usa para sus cálculos, durante los cuales cambia el número en la casilla X pero el de la casilla A permanece igual.

La manera particular como trabajan las variables ficticias se comprueba si modificamos las líneas 16Ø y 17Ø por:

```
160     LET X=11
```

```
170    PRINT FACT(X)
```

Hemos definido ahora una variable global con el mismo nombre que la variable ficticia de la función. Pero encontraremos que el programa sigue tratando estas dos 'casillas' como separadas, aunque al principio de la función, se 'copie' un número de una casilla a la otra. Podríamos añadir a la función una línea extra:

```
145    LET X=100
```

y una línea extra al programa 'principal';

```
180    PRINT X
```

pero encontraríamos que la línea 18Ø imprimía 11, no 1ØØ. La línea 145 modifica únicamente la X 'local'.

REFERENCIA DE PARAMETROS

Acabamos de ver a una función, utilizando una variable ficticia, realizar un cálculo con un número 'copiado' de una variable global. Pero aunque la función devolviera un número al programa principal, permanecería sin cambios la variable real de la que se sacó la copia.

Es una cosa muy distinta si ponemos REF (abreviatura de 'referencia') frente a la variable ficticia, como indicará el sencillo ejemplo siguiente.

Este programa permite escribir dos números - para A y B - y a continuación los cambia elevando A a la potencia de B, y B a la potencia de A:

```
100    INPUT A
110    INPUT B
120    CALL POWERS(A,B)
130    PRINT A,B
140    END
150    DEF POWERS(REF X, REF Y)
160        LET Z=X
170        LET X=X^Y
180        LET Y=Y^Z
190    END DEF
```

La línea 15Ø introduce dos variables ficticias. Cuando se llama a la función con CALL, el valor de la primera variable (de la línea 12Ø) se transfiere a la variable X, y el valor de la segunda se transfiere a Y. Es lo mismo que hemos visto anteriormente, excepto

 que en los ejemplos anteriores sólo había una variable ficiticia; por otra parte, dado que esta función entregará dos números (no simplemente uno) al programa principal, debe ser activada con una sentencia CALL.

La diferencia que representa introducir REF en la línea 15Ø es simplemente que cuando la función ha terminado sus cálculos, el nuevo valor que ha dado a X se devuelve a la variable global A (e Y se devuelve a B).

El movimiento de vaivén de valores entre funciones y otras partes del programa se denomina paso de parámetros. Si la función tiene el efecto de modificar las variables globales (o matrices, etc.) de las que ha tomado sus números para proceso, lo denominamos referencia de parámetros. En el ejemplo anterior, A y B son parámetros de referencia

GRAFICOS

Las potentes facilidades gráficas de la Enterprise pueden utilizarse para obtener algunas imágenes y efectos visuales impresionantes. Con la cassette de demostración, se habrán probado ya algunos, y varios de los programas de las partes anteriores del manual muestran un atisbo de las posibilidades de gráficos con elevada resolución.

En la primera parte del manual, se explicó el uso de PRINT AT. Este comando utilizaba un sistema que dividía la pantalla en una serie de 'posiciones', o cuadrados imaginarios, a fin de poder especificar en dónde se quería imprimir algo. PRINT AT 1,1 pondría una serie (o un número) en la esquina superior izquierda de la pantalla.

Los comandos gráficos utilizan un sistema similar para poner en pantalla líneas y puntos y permitirnos efectuar diagramas e imágenes. En este caso, sin embargo, las 'posiciones de pantalla' son mucho menores. He aquí un breve programa que serviría para trazar una línea.

```
100   GRAPHICS
110   PLOT 640, 360; 1000, 700
120   END
```

En los ejemplos anteriores hemos visto ya como funcionaba la primera sentencia. La palabra GRAPHICS es una manera rápida y simple de seleccionar una 'página' en blanco, en la que se pueden efectuar ilustraciones e imágenes.

El comando PLOT se usa para dibujar puntos o trazar líneas. Los cuatro números que siguen a PLOT en la línea 11Ø corresponden a dos posiciones de la página de gráficos; este programa dibuja una línea desde el centro de la pantalla (64Ø, 36Ø) a una posición (1ØØØ-7ØØ) situada en la dirección de la esquina superior derecha.

Observemos que estos números de 'coordenada' son mucho mayores que los números de filas y columnas utilizados con PRINT AT. Esto no significa que hagan referencia a un área superior; sino que la 'página de gráficos' se divide en un número mucho mayor de posiciones de lo que podía hacerse en una página de 'texto'. Es decir, la resolución cuando trabajamos con los gráficos es muy superior.

Otra diferencia más entre 'gráficos' y 'textos' es que, en la página de gráficos, el 'original' (la posición Ø,Ø) está en la esquina inferior izquierda, y el primer número que se da en un par de coordenadas es la posición horizontal.

ág. 90 Esto sigue las convenciones (x,y) que se utilizan normalmente para dibujar gráficos.

Al ejecutar el programa, habrá comprobado que en la parte inferior de la pantalla han quedado cuatro líneas de la página de texto normal. Esto nos permite continuar escribiendo en el ordenador mientras mantenemos visible el dibujo.

Esta división entre una página de gráficos standard y cuatro líneas de la página de texto se proporciona para facilitar las operaciones y se obtiene siempre cuando se pasa el comando simple GRAPHICS. Este nos deja área de pantalla para dibujar que mide 128Ø posiciones horizontales por 72Ø verticales, por lo que las coordenadas de la esquina superior derecha son (1279,719).

Las dos partes de la pantalla pueden borrarse separadamente con los comandos CLEAR GRAPHICS y CLEAR TEXT, o ambas a la vez con CLEAR SCREEN.

Si escribimos DISPLAY TEXT, la pantalla volverá a la 'página' de texto de tamaño completo. De igual manera, DISPLAY GRAPHICS devuelve a la página gráficos sin modificar nada de lo que había en ella antes. Observe la diferencia entre estos comandos y las simples palabras TEXT y GRAPHICS, que tienen el objeto de borrar las páginas de texto y de gráficos.

Cuando aprendamos qué son los 'canales' y las características más sofisticadas de los gráficos, podremos especificar libremente el tamaño de nuestras 'páginas' y visualizarlas en cualquier parte de la pantalla que escojamos.

UNTOS O
INEAS

Intentemos cambiar la sentencia PLOT del programa de manera que diga simplemente:

```
11Ø     PLOT 1ØØ, 1ØØ
```

- y ejecutemos de nuevo el programa. En pantalla aparece un punto.

Añadamos ahora un punto y coma después de 1ØØ,1ØØ. Y a continuación:

```
115     PLOT 1ØØØ,7ØØ
```

y ejecutemos una vez más el programa. Este trazará de nuevo una línea. A continuación retiremos el punto y coma de la línea 11Ø. Ejecutemos el programa. Aparecen dos puntos. ¿Por qué?.

La respuesta es que el punto y coma controla la intervención del 'rayo' video. Cuando el rayo está 'conectado', deja una línea visible cuando dibuja entre dos puntos.

Para mantenerlo conectado, se necesita el punto y coma después de una sentencia PLOT.

Podemos considerar el rayo como un lápiz de dibujo. El comando PLOT, con coordenadas, pondrá el lápiz en el papel y dibujará al menos un punto. Para trazar una línea entre dos posiciones de la pantalla, escribir los dos pares de números de coordenadas y separar los con punto y coma. Con esto se cambia el comando a fin de que diga: 'dibuja (PLOT) un punto en la posición (1ØØ,1ØØ) y a continuació mantiene el lápiz sobre el papel, desplazándose en línea recta hasta (1ØØØ, 7ØØ)'. Si no ponemos el punto y coma, nuestro lápiz imaginario seguirá moviéndose, pero sin tocar el papel.

Podemos usar también los comandos SET BEAM ON y SET BEAM OFF para poner el 'lapiz' en el papel o levantarlo.

He aquí un programa de puntos dispersos:

```
100   RANDOMIZE
110   INPUT PROMPT "How many measles? ": B        11Ø  "¿Cuántos puntos dispersos?"
120   GRAPHICS
130   LET Z=Ø
140   DO
150       LET X=RND(1279)
160       LET Y=RND(719)
170       PLOT X,Y
180       LET Z=Z+1
190   LOOP UNTIL Z=B
200   END
```

Este programa dibujará puntos en posiciones aleatorias de la pantalla. Cambiando 17Ø por:

```
170 PLOT X,Y;
```

podemos cambiar los puntos dispersos por líneas.

Naturalmente, en la definición de una fución pueden incluirse cualquiera de los comandos gráficos. El ejemplo siguiente, muestra por cierto, que podemos poner varias posiciones de pantalla en un comando PLOT y trazar líneas entre todas ellas:

```
100   DEF DIAGRAM
110       GRAPHICS
120       PLOT 504,544;564,464;516,448;504,544;
          460,464;516,448
```

```
130     END DEF
```

Ejecutar esto y escribir después CALL DIAGRAM en modalidad inmediata.

Si ponemos una coma después de un par de coordenadas, no se insertará ningún punto en esa posición; el ordenador simplemente pasará allí el rayo, y lo apagará. Se comprobará que esto es necesario si queremos pasar la instrucción PLOT PAINT.

OMANDOS
ORTUGA

Veamos otro conjunto de comandos que nos permiten dibujar líneas. Se denominan comandos 'tortuga', porque originalmente se utilizaban para controlar un animal robot que se movía lentamente. Esta vez no es preciso dar las coordenadas de toda una serie de posiciones de pantalla.

```
100     OPTION ANGLE DEGREES
110     GRAPHICS
120     PLOT 300,150;
130     PLOT ANGLE 80;
140     PLOT FORWARD 500;
150     PLOT BACK 320;
160     PLOT RIGHT 35;
170     PLOT FORWARD 420;
180     PLOT BACK 285;
190     PLOT RIGHT 100;
200     PLOT FORWARD 340
210     END
```

Se puede ver, lógicamente, que es algo así como guiar un animal por toda la pantalla. Pero algunas de las líneas del programa necesitan un poco de explicación.

La línea 1ØØ indica al ordenador que queremos medir los ángulos en grados, no en radianes. (Un radían tiene aproximadamente 57 grados; hay algunas operaciones matemáticas en las que son más cómodos los radianes).

La sentencia GRAPHICS (línea 11Ø) tiene como efecto colocar el rayo en (Ø,Ø) y apagarlo. (CLEAR GRAPHICS haría también esto). Por tanto, se necesita la línea 12Ø para dar la posición de partida desde la que queremos que se mueva el rayo (nuestro 'animal').

A continuación, tenemos que apuntar al animal hacia la derecha. PLOT ANGLE Ø lo dejaría virando horizontalmente hacia la parte derecha de la pantalla. PLOT

pág. 93 ANGLE 9Ø lo apuntaría recto hacia arriba. El comando PLOT ANGLE dice: 'consideremos primero el que A mira a la derecha y lo giraremos en sentido contrario a las agujas del reloj en el número de grados especificado'.

Un comando para PLOT FORWARD o PLOT BACK va seguido por el número requerido de posiciones en la pantalla de gráficos. PLOT RIGHT o PLOT LEFT hacen que el animal cambie de dirección, y van seguidos por el número de grados que se quiere que gire el rayo, respecto a su dirección anterior. Ya se vió esto anteriormente, en el programa de la página 66.

Observemos que con los comandos 'tortuga' tenemos que seguir utilizando punto y coma para mantener el rayo encendido. (Probemos a retirar algunos de los puntos y coma del programa y sustituirlos por comas, para ver lo que sucede).

ELIPSES Y CIRCULOS

En el siguiente programa dibuja una elipse:

```
100    GRAPHICS
110    PLOT 640,250,
120    PLOT ELLIPSE 100,200,
130    END
```

La línea 11Ø da el centro de la elipse. El primer número después de PLOT ELLIPSE es la distancia horizontal (en posición de pantalla) entre el centro y la circunferencia, y el siguiente es la distancia vertical. Si estos números fueran iguales, el programa trazaría un círculo. Observemos las comas al final de las líneas 11Ø y 12Ø. Si dejamos de poner una de ellas, el centro de la elipse estaría marcado en la pantalla con un punto. Tal como está, el programa deja el rayo en esta posición central, pero lo apaga.

COLORES

Probablemente sabemos ya que el Enterprise puede visualizar 256 colores. Hasta ahora, no hemos tenido la posibilidad de utilizarlos Aquí es donde tenemos que aprender los muchos matices que el Enterprise puede visualizar en pantalla.

El programa siguiente sirve para visualizar inmediatamente los 256 colores:

```
100    !
110    GRAPHICS 256 ! Note the number which thi
120    !              time has to follow
```

11Ø observe el número que debe seguir esta vez.

```
125   !              GRAPHICS.
130   !
140   LET Z=Ø
150   FOR Y=Ø TO 560 STEP 8Ø
160       FOR X=32 TO 1Ø52 STEP 32
170         SET INK Z
180         PLOT X, Y; X, Y+7Ø
190         LET Z=Z+1
200       NEXT X
210   NEXT Y
220   END
```

El Enterprise identifica cada color con un número de código en la gama Ø a 255. Para mayor simplicidad, se proporciona una función especial 'RGB' que permite seleccionar colores mezclando ciertas cantidades de rojo, verde y azul. Esto se explica más tarde en este mismo capítulo.

DALIDADES
COLOR

En el programa anterior, tuvimos que escribir el número 256 después de la palabra GRAPHICS, para decir al ordenador que ponga todos sus colores inmediatamente a nuestra disposición. Es decir, tuvimos que seleccionar la modalidad de color apropiada.

Observemos que, en este programa, las líneas trazadas en la pantalla son más gruesas que en los anteriores programas gráficos. El caso es que cuanto mayor es el número de colores de que disponemos, menos finos podrán ser los dibujos. Esto es necesario para conservar espacio de memoria en el ordenador.

Hemos dicho que la página 'gráficos' da una mayor 'resolución' (grado de precisión) que la página 'texto'. Pero debemos considerar ahora otras diferencias en la resolución que dependen de la modalidad de color que se utilice. Hay cuatro de estas modalidades de resolución (HIRES) y cada una de ellas da una cierta compensación entre el número de colores que podemos visualizar y el número de 'puntos' por línea horizontal que tenemos disponibles para el dibujo. Son estas:

GRAPHICS HIRES 2 - Cuando se pasa este comando, sólo pueden visualizarse al mismo tiempo dos colores, pero tenemos 64Ø puntos separados a lo ancho de la pantalla.

GRAPHICS HIRES 4 - Cuando se dispone de cuatro colores, tenemos 32Ø puntos por línea horizontal.

GRAPHICS HIRES 16 - Esta vez hay 16Ø puntos por línea.

GRAPHICS HIRES 256 - Con la posibilidad de

visualizar hasta 256 colores al mismo tiempo, se nos dan 8Ø puntos separados por línea.

Si escribimos simplemente GRAPHICS, el ordenador utilizará la misma modalidad que usó la última vez que pasamos el comando. Cuando se escribe por primera vez después de encender el ordenador o a la reposición, GRAPHICS equivale a GRAPHICS HIRES 4.

El número de puntos verticales no queda afectado por la modalidad de color; en la página standard de gráficos (con las líneas de texto en la parte inferior), es siempre de 18Ø.

A pesar de la diferencia de resolución, todas las modalida-- des de color utilizan el mismo sistema de coordenadas. Dicho de otro modo, PLOT Ø,Ø; 64Ø,36Ø dibujará siempre una línea desde la esquina inferior izquierda hasta la mitad de la 'página' aunque la finura del dibujo variará según la modalidad. (Este mismo esquema de coordenadas podría utilizarse realmente para una resolución el doble de lo que sería posible incluso en el Enterprise).

La Sección de Referencias explica como puede reducirse a la mitad la resolución (y ahorrarse memoria) pasando el comando GRAPHICS LORES. También se dán allí detalles de la modalidad de gráficos denominada de 'atributos'.

MODALIDADES DE GRAFICOS

Además de HIRES, hay otras dos modalidades de gráficos.

LORES es idéntica a HIRES, pero utiliza menos memoria del ordenador y proporciona la mitad de la resolución horizontal. Las modalidades de color se especifican de igual manera que HIRES, por ejemplo:

GRAPHICS LORES 16

daría una página de gráfico con baja resolución y 16 colores.

La modalidad ATTRIBUTE es una forma especial de representación en video, que se encuentra entre las modalidades de texto y gráficos. Se puede usar para imprimir caracteres, o para los comandos de dibujo, y proporciona 16 colores, pero es preciso manejar cuidadosamente el 'señalizador' ATTRIBUTE para su uso más efectivo. Ver la sección de referencia, página 188.

Si no se especificara ninguna modalidad de color, el comando sería:

GRAPHICS ATTRIBUTE

SELECCION DE COLORES

Volvamos de nuevo a la modalidad de 256 colores. En esta modalidad, podemos dibujar formas en cualquier color que queramos, haciendo que los comandos de dibujo vayan precedidos por la sentencia SET INK y el número de código del color. Igualmente, antes de usar CLEAR GRAPHICS, podemos escribir SET

 PAPER ..., a fin de elegir el color del 'fondo'.

La experiencia puede enseñarnos que color corresponde a cada número: el 18 es un verde claro, el 91 un amarillo fuerte, etc. Pero si queremos utilizar un determinado color y sucede que no sabemos cual es su número de código, existe también otro sistema para especificarlo.

Cada color posible puede crearse con una combinación de rojo, verde y azul. Por ejemplo, el blanco se obtiene de la mezcla de los tres colores. El amarillo mezclando simplemente rojo y verde. (Esto tal vez nos sorprenda, pero tengamos presente que la mezcla de las fuentes reales de luz proporciona resultados bastante diferentes que las mezclas de pinturas). El negro no significa ningún color. Se crean colores complejos mezclando el rojo, el verde y el azul en cantidades diversas.

Este es el principio que siguen los colores del Enterprise; podemos definir cualquier color como una mezcla, escribiendo la palabra RGB, seguida (entre paréntesis) por tres números con comas que los separen. Estos números, que deben encontrarse en la gama Ø-1, definen las proporciones de rojo, verde y azul (respectivamente) que queremos mezclar.

Por tanto, SET INK RGB (1,.5,.5) nos daría el rosa como color. RGB (.4,.4,Ø) es un amalillo apagado; RGB (.6,.6,.4) es una tonalidad de gris; y así sucesivamente.

Los 8 colores siguientes pueden seleccionarse sin complicación, simplemente escribiendo sus nombres (por ejemplo, SET INK GREEN). He aquí las 'mezclas' a que corresponden:

BLACK	= RGB(Ø,Ø,Ø)	NEGRO
RED	= RGB(1,Ø,Ø)	ROJO
GREEN	= RGB(Ø,1,Ø)	VERDE
YELLOW	= RGB(1,1,Ø)	AMARILLO
BLUE	= RGB(Ø,Ø,1)	AZUL
MAGENTA	= RGB(1,Ø,1)	MAGENTA
CYAN	= RGB(Ø,1,1)	CYAN
WHITE	= RGB(1,1,1)	BLANCO

PALETA
ALETTE)

Si pasamos el comando GRAPHICS HIRES 16, limitándonos a 16 colores en pantalla en un momento dado, seguimos teniendo una notable libertad para elegir cuáles serán estos colores. Esto se hace especificando una 'paleta': una lista de colores seleccionados que están disponibles para dibujar.

En primer lugar, escribimos SET PALETTE, y enumerar a continuación ocho de los

colores que queremos usar. Estos pueden elegirse libremente de toda la gama de 256, y podemos especificarlos con su número de código standard, con sus nombres (si están en la lista anterior), o definiéndolos como 'mezcla'. Por ejemplo, podríamos escribir:

```
100   SET PALETTE 67,31,WHITE,
      4,RGB(0,.3,.8),RGB(.7,.7,.1),
      187,190
```

- y estos colores tendrían en nuestra 'paleta' los números 0-7.

Para los ocho colores restantes, tenemos menos libertad de elección. Los colores numerados 8-15 en la paleta tienen que pertenecer todos a un único grupo de colores afines. Los seleccionamos con el comando SET BIAS, seguido por cualquier número perteneciente al grupo que deseamos. Por ejemplo, si se escribe SET BIAS 67 (o cualquier otro número en la gama 64-71), entonces el color con el número de código standard 64 (el primero de este grupo) se convertiría en el número 8 en nuestra paleta; el número standard 65 sería el número 9, etc. Es fácil imaginar que, con SET BIAS, se especifica el 'filtro' o el 'lavado' que deben colocarse sobre los colores 'primarios' del Telextext.

Podemos ahora seleccionar cualquier color de la paleta para preparar la 'tinta' para el dibujo de la siguiente línea o dibujo. Un punto importante es que, si no estamos utilizando la modalidad de 256 colores, cualquier comando como SET INK 6 o SET PAPER 6 hará referencia al color enumerado como número 6 en la paleta, no al que tiene el 6 como número de código standard. No olvidemos tampoco que los números de la paleta se cuentan hacia arriba desde 0, no desde 1.

En la modalidad de cuatro colores, sólo pueden usarse los colores numerados 0-3 en la paleta, por lo que no hace falta listar más de cuatro colores en un comando SET PALETTE. De igual manera, en la modalidad de dos colores, sólo queremos especificar los colores de la paleta número 0 y 1.

A veces podremos desear modificar sólo un color de la paleta, dejando el resto como está. Este ejemplo modifica el color 3:

```
SET COLOUR 3, 110
```

Como es lógico, se pueden utilizar cualquiera de las modalidade sin tener que preocuparse de seleccionar la paleta. Si

 no se especifica nada, el ordenador utilizará siempre determinados colores pre-programados 'por omisión'.

JSO DE LA PALETA

Recordemos que si se modifica la serie de colores de la paleta, esto afectará no sólo a las líneas y dibujos que han de dibujarse después, sino también a las que están ya en pantalla. Esta capacidad para cambiar todos los colores de la pantalla con un sólo comando, es la base de algunos de los efectos gráficos más atractivos y rápidos.

En el programa siguiente, que crea elipses de tamaños y colores aleatorios, empezaremos haciendo que todos los colores de la paleta sean iguales; por tanto, una línea dibujada en 'rosa' con el número de color 2, no aparecerá diferente de la dibujada en el color 3, y ambas podrán distinguirse del 'papel'; en efecto, el dibujo será invisible. Acto seguido, al pulsarse una tecla, se suspenderá el dibujo y variando repetidas veces el contenido de la paleta, el programa hará que las diferentes 'tintas' cambien de color y se diferencien entre sí y respecto del fondo.

El programa termina con un bucle infinito: nunca termina a menos que sea interrumpido. Para suspender la ejecución, pulsar la tecla 'stop'.

```
100   RANDOMIZE
140   GRAPHICS HIRES 4
150   SET PALETTE BLUE,BLUE,BLUE,BLUE
160   !
170   !              Invisible display.
180   !
190   DO
200       SET INK RND*3+1
210       PLOT 625,330,
220       PLOT ELLIPSE RND*500,RND*300,
230   LOOP WHILE INKEY$=" "
240   !
250   !     When key is pressed, show display.
260   !
270   DO
280       SET PALETTE BLUE,BLUE,RED,GREEN
290       !
300       FOR X=1 TO 500 !   Delay for
310       NEXT X !           nearly 1 second.
320       !
330       SET PALETTE BLUE,RED,GREEN,BLUE
340       FOR X=1 TO 500
```

250 Cuando se pulsa la tecla, mostrar el display.

300 310 Retraso durante aprox. 1 segundo.

```
350         NEXT X
360         SET PALETTE BLUE,GREEN,BLUE,RED
370         FOR X=1 TO 500
380         NEXT X
390     LOOP
400     !-------------------------------
410     !   Use 'stop' key to halt program.
420     !-------------------------------
```

41Ø Utilizar la tecla 'stop' para detener el programa.

Digamos, de pasada, que los comandos PALETTE, INK y PAPER pueden usarse también si se trabaja con una página de 'texto'. El capítulo sobre 'Opciones de Video' de la Sección de Referencias da detalles sobre esto. Probemos los experimentos siguientes:

```
SET &1Ø2: COLOUR 1, MAGENTA

SET &1Ø2: INK 3
```

PLOT PAINT
(PINTURA DE
RELLENO)

Esta instrucción rellena una forma sólida con el color de la 'tinta' actual. He aquí un programa que dibujará un círculo dentro de otro y los pintará de dos colores diferentes:

```
100     GRAPHICS HIRES 4
110     SET PALETTE WHITE, YELLOW, BLUE
120     PLOT 400,400,
130     PLOT ELLIPSE 200,200,
140     PLOT ELLIPSE 80,80,
150     SET INK 3
160     PLOT PAINT
170     PLOT 400,250,
180     SET INK 2
190     PLOT PAINT
200     END
```

Así, el área rellenada por PLOT PAINT es la que contiene actualmente el 'rayo' y está encerrada por una línea continua de color diferente desde la posición del rayo. Cualquier hueco en la línea significa que la pintura se saldrá de la forma e intentará llenar toda la pantalla. Obsérvense las comas al final de las líneas de programas 13Ø, 14Ø y 17Ø; tenemos que impedir que apareciera un punto en la posición del rayo, ya que de otro modo PLOT PAINT habría pintado únicamente este punto.

LINE STYLE AND
LINE MODE

ESTILO Y
MODALIDAD DE
LINEA)

LINE STYLE nos permite dibujar con diversos tipos de líneas interrumpidas. Por ejemplo, si se escribe SET LINE STYLE

1Ø, todas las líneas dibujadas (hasta que se reponga en otro estilo) están formadas por guiones largos y muy unidos. SET LINE STYLE 9 nos proporciona líneas de puntos y rayas alternados. Hay 14 estilos de líneas (numerados a partir del 1) con los que pueden experimentarse.

Con el comando SET LINE MODE, podemos determinar como se relacionarán las líneas trazadas en la pantalla con las formas que hay ya en ella. En la modalidad de línea Ø (el valor 'por omisión'), cualquier línea nueva simplemente 'escribirá encima' de las líneas o formas anteriormente dibujadas. En otras modalidades (numeradas del 1 al 3), las 'tintas' antiguas y nuevas de la página se combinarán de diversas maneras para determinar el color del trazado; la Sección de Referencias nos proporciona más detalles.

GINAS Y NALES

Hasta ahora, hemos estado utilizando el comando GRAPHICS (o TEXT) para crear una 'página' en blanco en la que poder dibujar (o escribir). La mayoría de las veces esta disposición es adecuada, pero a medida que se vaya dominando mejor el ordenador, podrá desearse utilizar una mayor flexibilidad que se obtiene con la especificación de números de 'canal' (normalmente, en la gama de 1 a 1ØØ) para las diversas 'páginas' de texto y gráficos. La exposición que sigue debe leerse junto con el capítulo sobre 'canales' y las partes correspondientes de la Sección de Referencias (ver, por ejemplo, 'Opciones de Video' y la palabra clave OPEN).

Al abrir nuevos 'canales' para las páginas, se tienen dos ventajas principales:

1) Se pueden conservar al mismo tiempo en la memoria del ordenador varias páginas diferentes que contengan dibujos completos o texto; cualquiera de ellas podrá entonces pasarse a la pantalla con un sólo comando.

2) Se puede especificar el tamaño de la página. Así, puede hacerse que un dibujo gráfico llene (prácticamente) la totalidad de la pantalla (pero obsérvese que no puede utilizarse la 'línea de estado' de la parte superior); o se puede ahorrar memoria haciendo que la página sea pequeña. Se puede elegir la posición vertical en la pantalla para visualizar la página o una parte de ella.

El ejemplo siguiente crea una pequeña página de texto y la muestra en medio de la pantalla:

```
50    SET BORDER CYAN
100   SET VIDEO MODE Ø
110   SET VIDEO COLOUR Ø
120   SET VIDEO X 20
130   SET VIDEO Y 10
140   OPEN £1: "VIDEO:"
```

```
150    DISPLAY £1:AT 7 FROM 1 TO 10
160    PRINT £1: "A small text page..."
170    END
```

16Ø "Una pequeña página de texto..."

La línea 14Ø asigna el número de canal 1 a nuestra nueva página video. Pero antes de poder hacer esto, es preciso especificar la 'modalidad video', 'modalidad de color' y las dimensiones de la página.

La modalidad Video Ø es una página de texto de 4Ø columnas; la 1 es una página de gráficos y la 2 un texto de 8Ø columnas.

VIDEO COLOUR selecciona la modalidad de color, según la convención siguiente:

VIDEO COLOUR Ø - modalidad de 2 colores.

VIDEO COLOUR 1 - modalidad de 4 colores.

VIDEO COLOUR 2 - modalidad de 16 colores

VIDEO COLOUR 3 - modalidad de 256 colores

Una página de texto debe estar siempre en modalidad de color Ø; esto nos sigue permitiendo una cierta elección de los colores.

Las líneas 12Ø y 13Ø dan la anchura y altura de la página en 'posiciones de caracteres'.

La línea 15Ø es una instrucción para colocar la parte superior de la página (medida como 1Ø filas de caracteres) en el display, comenzando por la fila 7 de la pantalla. En este caso, como es natural, muestra la página como un todo.

Obsérvese que el comando PRINT de la línea 16Ø debe incluir el número de canal.

La altura especificada para la página, puede ser, si se desea, mayor que la altura de la pantalla . (El máximo es 255 filas de caracteres). El programa siguiente define una página que mide 8 columnas horizontalmente y 30 verticalmente, y dibuja en ella una elipse; a continuación se viasualizan uno tras otro dos segmentos, que contienen las porciones superior e inferior del dibujo.

```
100    SET VIDEO MODE 1
110    SET VIDEO COLOUR 2
120    SET VIDEO X 8
130    SET VIDEO Y 30
140    OPEN £1: "VIDEO:"
150    PLOT £1:128,540,
160    PLOT £1:ELLIPSE 115,500,
170    DISPLAY £1:AT 10 FROM 21 TO 30
180    FOR X=1 TO 1500
190    NEXT X
200    DISPLAY £1:AT 10 FROM 1 TO 10
```

```
210    END
```

El ejemplo siguiente define 3 'páginas' y las visualiza simultáneamente en diferentes partes de la pantalla:

```
100    SET VIDEO MODE 0
110    SET VIDEO COLOUR 0
120    SET VIDEO X 42
130    SET VIDEO Y 8
140    !
150    OPEN £1: "VIDEO:" !                Text page.      150  página de texto
160    !
170    SET VIDEO MODE 1
180    SET VIDEO COLOUR 3
190    !
200    OPEN £2: "VIDEO:" !   256-colour graphics.   200  Gráficos de 256 colores
210    !
220    SET VIDEO COLOUR 1
230    !
240    OPEN £3: "VIDEO:" !    4-colour graphics.    240  Gráficos de 4 colores.
250    !
260    DISPLAY £1: AT 9 FROM 1 TO 8
270    DISPLAY £2: AT 1 FROM 1 TO 8
280    DISPLAY £3: AT 17 FROM 1 TO 8
290    PRINT £1: "Text..."
300    SET £2: BEAM ON
310    PLOT £2: 100,100;
320    SET £3: BEAM ON
330    PLOT £3: 100,100;
340    END
```

Después de ejecutar esto, no se podrá ver lo que se está escribiendo. Escribir TEXT o pulsar la tecla de función 7 para volver a la normalidad.

Obsérvese que cada 'página' tiene su propia 'paleta', aunque a todas las páginas se aplicará un comando SET BIAS.

El uso de los números de canal nos permite dibujar o escribir en una página aunque no esté actualmente visualizada. Permite también que se impriman caracteres en una página de gráficos, en la posición actual del 'rayo'. Por ejemplo, si el canal 3 se ha abierto como una página de gráficos, puede escribirse algo así:

```
PLOT £3: 640, 50
PRINT £3: "Hello"
```

y la serie se añadirá a la página, esté o no presente en pantalla.

SET BORDER
(INTRODUCIR
RECUADRO)

Tal como se utilizó en el primer ejemplo de esta sección, puede seleccionarse un color para el recuadro que rodea todo el display visible: la 'mesa de despacho' en la que se colocan las diversas 'páginas' de texto o de gráficos. El comando SET BORDER es independiente de todos los comandos de 'paleta' (que sólo se aplican a determinadas páginas), por lo que este color debe introducirse especificándose un número de código standard, un nombre de color o una 'mezcla'. Por ejemplo, SET BORDER 255, SET BORDER WHITE o SET BORDER COLOUR (1,1,1) darán un recuadro blanco a todo el display.

Si no está abierto el canal 1Ø1 (el canal de la página de gráficos standard) cuando se introduce el color del recuadro, el comando debe pasarse por un número de canal adecuado (normalmente 1Ø2, el número de la página de 'texto' standard) por ejemplo, SET £1Ø2: BORDER 116.

EL JUEGO DE CARACTERES

El Enterprise lleva un grupo predefinido de caracteres que forman el juego de caracteres standard. Siguen las definiciones de caracteres de la Organización Internacional de Normas (ISO), pero normalmente se denomina como ASCII (abreviatura de American Standard Code for Information Interchange), (Código Standard Americano para Intercambio de Información).

Cada carácter del juego tiene un número de código entre 32 y 159. Puede hacerse referencia a un carácter standard por su aspecto, como el PRINT "N", o por su código. PRINT CHR$(78) es lo mismo que PRINT "N": como se comprueba probándolos ambos.

El ordenador utiliza estos códigos para hacer referencia a los caracteres que hay en su interior, pero no es preciso comprobar cómo lo hacen mientras no se haya avanzado realmente. Como simple explicación: si, por ejemplo, se pulsa una tecla del teclado -digamos que la 'a' - el ordenador recibirá una señal para indicarle que esta tecla ha sido pulsada, registrará instantáneamente el código del carácter (no la forma), enviará dicho código a la parte del ordenador que controla la pantalla y traducirá el código en los detalles para visualizar el carácter; que aparecerá entonces en el display.

Por lo que al operador se refiere, todo lo que sucede es que se pulsa una tecla y el carácter aparece al mismo tiempo. Esto es un ejemplo de la rapidez con que trabaja un ordenador, sobre todo si tenemos en cuenta que la operación arriba descrita es realmente mucho más compleja; cada fase de la tarea está subdividida en operaciones mucho más pequeñas.

He aquí un programa que imprimirá todo el juego de caracteres y nos permitirá entonces escribir un carácter para el que se imprimirá el código ASCII:

```
100   FOR N=33 TO 159
110   !
125   !    There are 128 pre-set characters.
130   !    This FOR/NEXT loop allows you to
140   !    miss out the 'control' characters,
150   !    which are Ø to 32 (32 is 'space').
160   !
170         PRINT CHR$(N),
180   NEXT N
190   DO
195         DO
200           INPUT PROMPT "Type a character
              and its code will appear: ":C$
```

125 13Ø 14Ø 15Ø Hay 128 caracteres pre-establecidos. Este bucle FOR/NEXT nos permite dejar a un lado los caracteres de 'control', que son del Ø al 32, (32 es el 'espacio').

2ØØ "Escribe un carácter y aparecerá un código:"

```
210          IF LEN(C$) > 1 THEN
220            PRINT "Only one character at a
               time, please!"
221            ! ---------------------------
222            !  The small IF block (210-235)
223            !  makes sure you do not type in
224            !  more than one character. Notice
225            !  that throughout this program, DO
226            !  LOOPS control it. LEN gives the
227            !  length of a string (line 210).
228            ! ---------------------------
235          END IF
240        LOOP WHILE LEN(C$) > 1
250        PRINT"The ASCII code for ";C$
260        PRINT "is:";ORD(C$)
261        ! ---------------------------
262        !  ORD gives the ASCII code for a
263        !  character.
264        ! ---------------------------
265        DO
270          INPUT PROMPT "More characters,
             y/n? ":A$
280        LOOP WHILE A$ < > "y" AND A$ < >
           "n"
290      LOOP WHILE A$="y"
300      END
```

```
220  "¡Sólo un carácter cada vez, por
     favor!"

222 ..... a 227  El pequeño bloque
     IF (210-235) se asegura de que no
     escribamos más de un carácter.
     Obsérvese que en todo este
     programa, el control lo tiene
     DO LOOPS.  LEN  da la longitud de
     una serie (línea 210).

250  "El código ASCII para"
260  "es:"

270  "¿Más caracteres, Y/N?"
```

Como puede verse, ORD y CHR$ son contrarios. ORD da un código ASCII para el carácter y CHR$ imprime el carácter para un código ASCII.

IDEAR CARACTERES PROPIOS

Ya que sabemos algo sobre los códigos ASCII, unas pocas líneas para la realización de formas propias de caracteres. Para cada código ASCII, el ordenador recuerda una forma que tiene en su memoria, pero nosotros podemos redefinir esta forma.

Imaginemos un carácter que está formado por nueve filas de ocho pequeñas linternas. Para conseguir una forma, algunas linternas estarán encendidas y otras apagadas. Esto forma un dibujo de pequeños puntos que son tan pequeños y tan cercanos que aparecen unidos. Así, para diseñar o rediseñar un carácter, conviene dibujar una matriz cuadrada de 8 x 9 espacios. A continuación podemos formar el carácter colocando puntos en los cuadros adecuados.

Ahora bien, para programar esta información en el ordenador, debemos imaginar que todos los puntos son uno y que

todos los blancos son Ø. Observando cada fila podemos reunir una secuencia de Ø y 1. El programa siguiente designa un pequeño carácter y a continuación lo imprime.

```
100   NUMERIC N (1 TO 9)
110   !
120   !   A standard character is 8 dots wide and
130   !   9 rows deep. So each row is defined by
140   !   an eight-digit number. A series of nine
150   !   such numbers makes up the definition of
160   !   the character. Lines 280-300 store these
170   !   numbers in the array N.
180   !
190   DATA 00111110
200   DATA 01000001
210   DATA 01010101
220   DATA 01000001
230   DATA 00100010
240   DATA 00010100
250   DATA 00001000
260   DATA 00000000
270   DATA 00000000
280   FOR ROW = 1 TO 9
290         READ N (ROW)
300   NEXT ROW
310   SET CHARACTER 63,BIN (N(1)), BIN(N(2)),
      BIN(N(3)), BIN (N(4)), BIN(N(5)), BIN(N(6)),
      BIN(N(7)), BIN(N(8)), BIN(N(9))
320   !
330   !   BIN tells the computer to treat the 1's
340   !   and Ø's as binary digits.
350   !
360   PRINT "?"
370   PRINT CHR$ (63)
380   END
```

12Ø ..... a 17Ø Un carácter standard tiene 8 puntos de ancho y 9 filas de altura. Así, cada fila se define por un número de ocho dígitos. Una serie de nueve de estos números forma la definición del carácter. Las líneas 28Ø–3ØØ almacenan estos números en la matriz N.

33Ø 34Ø BIN indica al ordenador que trate los 1 y Ø como dígitos binarios.

La línea 31Ø es la sentencia que almacena la información del carácter. ¡Lo que hemos hecho es redefinir el signo de interrogación! El número 63 es el código ASCII, y todos los Ø y 1 son los puntos encendidos y oscuros (o puntos y espacios) de los que está formado el carácter. El carácter se imprime entonces con el comando PRINT "?" o PRINT CHR$(63). Intentemos también escribir el signo de interrogación en modalidad inmediata.

Una vez ejecutado un programa que redefine caracteres, podemos también imprimirlos hasta que

escribamos CLEAR FONT, pulsemos rápidamente dos veces seguidas el botón 'reset' o apaguemos el ordenador.

Es fácil comprobar lo divertido que pueden ser estos caracteres definidos por el usuario, tal como se conocen. Se puede rediseñar por ejemplo todo el alfabeto. La programación en letra gótica o en cursiva podría introducir una dimensión totalmente nueva a nuestras actividades de informática. Los egos especiales serán más vividos si se tienen unos alienígenas hechos por el mismo usuario (se puede formar una criatura con varios caracteres, imprimiéndolos juntos). La cassette demostración contiene un programa que nos proporciona varios caracteres ya hechos y otros que nos permite realizar los nuestros y preservarlos en la cassette.

SONIDO Y RITMO

Los efectos sonoros representan un punto muy interesante en la mayoría de los programas, sobre todo con los juegos. Los programas 'serios' pueden utilizar ruidos como señales. Los juegos son mucho más interesantes si se añaden algunos pocos sonidos a tiempo y apropiados.

El Enterprise nos ofrece la posibilidad de escuchar los sonidos en estéreo; con auriculares o a través del aparato de alta fidelidad. Si queremos hacer esto, y no queremos que se escuche al mismo tiempo el altavoz mono incorporado, puede ser silenciado escribiéndose SET SPEAKER OFF o pulsando 'shift' con la tecla de función 5.

El control de los sonidos se obtiene con dos palabras clave del BASIC. Son SOUND y ENVELOPE. La sentencia SOUND pasa al ordenador información general sobre el tiempo que debe durar un sonido, a partir de qué tono debe empezar, cuál será su volúmen máximo y algunos otros puntos. Una sentencia ENVELOPE es una serie de instrucciones que especifican detalladamente las variaciones de tono y volum que sufrirá el sonido durante el tiempo que se esté reproduciendo.

LA SENTENCIA SOUND

He aquí un ejemplo de una sentencia SOUND:

```
100  SOUND PITCH 40, LEFT 127, RIGHT 191,
     DURATION 200, ENVELOPE 10
```

Consideremos uno a uno los diversos puntos (separados entre sí por comas) que especifica la línea del programa.

En primer lugar, el número que hay después de PITCH indica cuál será la altura de la nota cuando empiece a reproducirse. En teoría, este número puede ser cualquiera, de Ø (que en realidad hace que el sonido sea prácticamente inaudible) a 127. La gama en la que se obtiene normalmente buenos resultados llega hasta 83 aproximadamente. Dentro de esta gama, cada aumento de 1 subirá el sonido en un semitono. El valor del tono 37 equivale al 'DO medio'. Si no se indica ningún valor del tono, se utiliza 37 como valor 'por omisión'.

El siguiente elemento de la serie determina el volúmen del sonido que se enviará al altavoz izquierdo. El número que hay después de LEFT debe estar en la gama de Ø a 255. Si se especifica LEFT Ø, el altavoz no sonará. LEFT 255 (que es el valor 'por omisión') permite que el sonido suba al más alto volúmen que puede producir la máquina; sujeto a las instrucciones posteriores que se contengan en una sentencia 'envelope'. En el ejemplo anterior, LEFT 127 significa que el sonido enviado al altavoz izquierdo nunca subirá por encima de

un volúmen medio, sea cual fuere el valor estipulado por ENVELOPE.

RIGHT dá de igual modo el volúmen máximo para el altavoz derecho. En nuestro ejemplo, este altavoz ha sido predispuesto para tres cuartas partes del volúmen.

El número que hay después de DURATION mide la longitud del sonido en 'tics': un tic es una 1/50 parte de segundo, por lo que en este caso el sonido durará 4 segundos. (El valor por omisión es 50 tics).

ENVELOPE hace referencia al número de la sentencia ENVELOPE que debe utilizarse en unión con esta sentencia SOUND. Hay una ENVELOPE incorporada, con el número 255, que se utiliza por omisión; los números válidos de nuestras envolventes están en la gama de Ø a 254.

Señalemos que todos estos elementos que siguen a la palabra clave pueden disponerse en cualquier orden. Hay algunas cosas más que pueden añadirse a la lista; volveremos a ellas más tarde. Dado que en todos los casos hay valores 'por omisión', la palabra SOUND producirá naturalmente un efecto si se utiliza independientemente.

ENVELOPES

He aquí un ejemplo de envelope (envolvente):

```
90  ENVELOPE NUMBER 10;2, 8, 63, 50; 0, 24,
    -16, 100; -5, 47, -39, 50
```

Una vez más, la palabra clave va seguida por una lista bastante larga de datos.

Para empezar, NUMBER 1Ø identifica la envolvente de manera que las sentencias SOUND pueden referirse a ellas (ver arriba).

A continuación viene un punto y coma, seguido por una serie de cuatro números (separados entre sí por comas).

Estos números definen los cambios que sufrirán el volúmen y el tono durante la primera 'fase': la primera parte del tiempo que ha sido asignado para el sonido (en este caso, la fase dura un segundo).

El primer número del lote significa que, durante este período, el tono subirá 2 semitonos respecto a su valor inicial dado por la sentencia SOUND. Por el contrario, -1,5 bajaría el tono en tres cuartos de tono; Ø daría un tono constante; etc.

El siguiente número (8) especifica el cambio de volúmen del altavoz izquierdo. En una sentencia ENVELOPE, una unidad de volúmen equivale a 1/64 del máximo permitido para el altavoz: es decir, 1/64 del volúmen establecido en la sentencia SOUND.

El número 63 subirá siempre el volúmen al máximo, mientras que -63 lo reducirá al silencio (se ignora cualquier fracción).

Al comienzo de su duración, el sonido no tiene volúmen en absoluto. Por tanto, en nuestro ejemplo, el volúmen de sonido enviado al altavoz izquierdo aumentará, durante la primera fase de la envolvente, de cero a un octavo del máximo.

El tercer número del lote (47) tiene una finalidad similar al segundo, pero para el altavoz derecho.

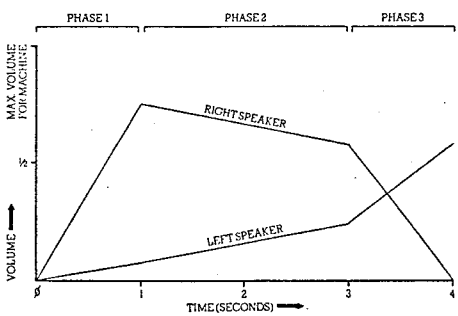
Las sentencias SOUND y ENVELOPE trabajan juntas, por tanto produciendo dos valores para el nivel de volúmen en cualquier momento concreto: un valor para el altavoz izquierdo y otro para el derecho. Obsérvese que si no se utiliza equipo estéreo, el volúmen se decidirá en cualquier momento simplemente sumando los dos valores separados.

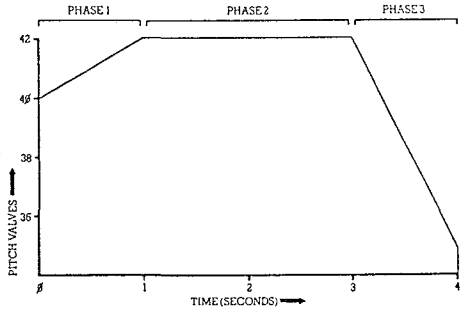
El cuarto número del lote (5Ø) especifica cuanto tiempo tardará esta primera fase en la envolvente. También aquí el tiempo se mide en 'tics'; cincuenta tics equivalen a un segundo.

A continuación llegamos a otra serie de cuatro números, seguidos por otro punto y coma; estos trabajan igual que el primer lote, salvo que definen lo que sucederá en la segunda fase de la envolvente.

A continuación cuatro números más (también con punto y coma al final) definen la tercera fase.

Si escribimos y ejecutamos las dos líneas de programa que hemos estado estudiando, se combinarán produciendo un sonido que podría representarse con este par de gráficos:





Los sonidos pueden ser mucho más complejos que éste, naturalmente. Más tarde veremos como puede definirse una envolvente con cualquier número de fases hasta 255.

Recuérdese que todas las instrucciones que se contienen en una envolvente son relativas: sus resultados dependen de la gama de volúmen y del tono inicial que especifica la sentencia SOUND.

Por tanto, la misma envolvente podría utilizarse con una serie de sentencias SOUND diferentes.

Recuérdese también que la envolvente debe ser definida en una línea de programa que se ejecute antes de cualquier sentencia de sonido que haga referencia a la misma.

COLAS DE SONIDO

La reproducción de sonido no suspende cualquier otra actividad que se haya dicho al ordenador que realice. En el programa siguiente, los comandos de impresión y gráficos se realizan con sonidos como acompañamiento:

```
100  ENVELOPE NUMBER 20; -2, 63, 63, 100; -3,
     0, 0, 100; 3, -36, -36, 100; 2, -12, -12, 100; 0,
     0, 0, 100
110  SOUND PITCH 61, LEFT 255, RIGHT 0,
     DURATION 500, ENVELOPE 20
120  CLEAR SCREEN
130  PRINT AT 10,5: "A graph will presently be
     drawn"                                          130  "Se dibujará actualmente un gráfico"
140  PRINT AT 11,5: "to represent the envelope
     that"                                           140  "Para que represente la envolvente que"
150  PRINT AT 12,5: "these sounds are using.
     Volume"                                         150  "está utilizando estos sonidos. El
                                                          volúmen"
160  PRINT AT 13,5: "will be indicated by the blue"  160  "se indicará con el azul"
```

```
170   PRINT AT 14,5: "area, pitch by the yellow
      line."
180   SOUND PITCH 43, LEFT 127, RIGHT 127,
      DURATION 200, ENVELOPE 20
190   !
200   !   The sound in line 180 will stop after
210   !   the first two phases of the envelope.
220   !
230   FOR X=1 TO 2000
240   NEXT X
250   SOUND PITCH 25, LEFT 0, RIGHT 255,
      DURATION 500, ENVELOPE 20
260   GRAPHICS
270   PLOT 0, 0; 250, 540; 500, 540; 750, 270; 1000,
      180; 1250, 180; 1250, 0; 0, 0
280   PLOT 100, 20,
290   PLOT PAINT
300   SET INK 3
310   PLOT 0, 450; 250, 350; 500, 100; 750, 350; 1000,
      450; 1250, 450
320   END
```

```
170  "el área, el tono por la línea
     amarilla".

200 210  El sonido de la línea 180 se
     detendrá después de las dos
     primeras fases de la envolvente.
```

La línea 110, el ordenador registra las instrucciones y empieza a reproducir el sonido; pero no espera a que este sonido termine antes de pasar a la línea siguiente del programa. El sonido de la línea 110 continúa mientras el ordenador está imprimiendo en pantalla su mensaje (líneas 130-170). En la línea 180, ocurre otra sentencia SOUND. Lo que sucede ahora es que el ordenador registrará esta sentencia e intentará reproducir el nuevo sonido tan pronto como termine el anterior: en otras palabras, el sonido especificado en la línea 120 se coloca en una 'cola' detrás del de la línea 110. De igual modo, en la línea 250, el tercer sonido se coloca a continuación del segundo. El gráfico (líneas 260-310) se realiza ahora mientras un sonido sigue al otro.

RELEASE

El Enterprise permite que una o más fases al término de una envolvente de sonido se traten de manera ligeramente diferente del resto. Estas fases van precedidas por la palabra RELEASE. Convencionalmente, la etapa 'release' de una envolvente significa un período en el que se permite cortar un sonido; cuando ha terminado efectivamente y sólo se desean sus efectos residuales.

Tomemos este ejemplo:

```
100   ENVELOPE NUMBER 4;. 1, 63, 40, 5; -. 3,
      -32, -20, 20; 2, 0, 0, 25; RELEASE; 0, -16,
      -10, 10; 0, -15, -10, 15;
```

En este caso, las tres fases que hay antes de la etapa RELEASE duran un total de un segundo, mientras que las dos fases RELEASE duran medio segundo.

La diferencia entre las fases no-RELEASE y RELEASE es ésta: si la 'duración' especificada en la sentencia SOUND termina antes de que hayan acabado las fases no-RELEASE, se corta; pero la fase RELEASE se ejecuta aunque haya terminado la 'duración', a condición de que no haya ninguno de los sonidos esperando a continuación.

Por tanto, si se especifica DURATION 25 en una sentencia SOUND que utilice la envolvente anterior, no se ejecutará la tercera fase; el ordenador saltará directamente de la segunda a la fase RELEASE, o al siguiente sonido en espera, si lo hay.

En el caso de DURATION 60, por otra parte, el ordenador ejecutará las tres fases no-RELEASE y la primera RELEASE, y pasará entonces al sonido siguiente si hay alguno a la espera.

En el caso de DURATION 100, se ejecutarán todas las fases y habrá una pausa de medio segundo antes de que comience el sonido siguiente.

INTERRUPT
(INTERRUPCION)

Hemos aprendido ya los fundamentos del control de los sonidos, pero aún quedan varias características interesantes por explorar. Por ejemplo, podemos hacer que un sonido interrumpa a otro. Volvemos al programa que trazó un gráfico con efectos sonidos como fondo. Retiremos las líneas 130-170, 230-240 y 260-310, e insertemos en su lugar:

```
260   ENVELOPE NUMBER 30; 0, 63, 63, 1; 0, 0, 0, 24
270   PRINT "Press i when you want to interrupt this
      sound."
280   DO
290   LOOP UNTIL INKEY$="i"
300   SOUND PITCH 37, LEFT 255, RIGHT 255,
      DURATION 25, ENVELOPE 30, INTERRUPT
```

```
270   "Pulse la tecla i cuando quiera
      interrumpir este sonido".
```

La inclusión de INTERRUPT en la línea 300 significa que

cuando se llegue a esta línea, el sonido que se está reproduciendo quedará cortado por el nuevo, y se olvidarán todos los que están esperando.

FUENTES SONORAS

Se pueden reproducir más de un sonido al mismo tiempo. El Enterprise incorpora cuatro fuentes sonoras, cada una de las cuales puede tener su propia "lista de espera" independiente de sonidos.

Para poner un sonido en "lista de espera", (por ejemplo) el generador de tono 2, todo lo que hay que hacer es incluir SOURCE 2 en la sentencia SOUND. Los generadores de tono están numerados Ø-3. El número 3 es el denominado 'generador de ruido' (que ignora los valores del tono). Hasta ahora, sólo hemos utilizado el generador de tono Ø, que es la fuente 'por omisión'.

Cuando se reproducen varios sonidos juntos. se querrá naturalmente asegurar una sincronización exacta entre las diferentes fuentes sonoras. Incluyendo la instrucción SYNC en una sentencia SOUND, podemos hacer que un sonido comience exactamente en el mismo momento que otro: o dos o tres más, según el número que se inserte después de SYNC.

Escribir un número de sentencia SOUND, distribuyéndolas entre SOURCE Ø, SOURCE 1 y SOURCE 2, es decir, efectuando tres 'esperas', separadas. A la primera línea del programa de cada espera, añadir la instrucción SYNC 2, de manera que aparezca algo así:

```
120   SOUND PITCH 40, LEFT 255, RIGHT 127,
      DURATION 200, ENVELOPE 6, SOURCE 1,
      SYNC 2
```

Cuando se han ejecutado las líneas (suponiendo que se han introducido también envolventes adecuadas), las tres "esperas" sonoras se pondrán en marcha al mismo tiempo.

Para ser exactos, la instrucción SYNC 2 dice: 'Cuando éste sonido llegue al comienzo de su "espera", mantenlo así hasta que estén preparados pra empezar otros dos nuevos sonidos de otras "esperas".

Esto significa que cuando hay preparado otro sonido, se mantendrá también automáticamente hasta que venga un nuevo sonido del comienzo de la tercera "espera", en cuyo momento las tres "esperas" empezarán a actuar a la vez.

El comando CLEAR QUEUE, seguido por el número de un generador de tono, silenciará cualquier sonido que proceda actualmente de esta fuente y borrará todo lo que esté esperando en cola. Obsérvese que una sentencia SOUND con una instrucción INTERRUPT no interrumpirá cualquier sonido que proceda de otras

'fuentes'.

SONIDOS MAS COMPLEJOS

Salvo que se le de instrucciones en otro sentido, el ordenador supone que todas las envolventes sonoras contendrán entre 1 y 2Ø 'fases'. No obstante, si (por ejemplo) queremos definir envolventes con un máximo de 25 fases, podemos escribir:

```
1ØØ    CLOSE £1Ø3
11Ø    SET SOUND BUFFER 25
12Ø    OPEN £1Ø3: "SOUND:"
```

- y el ordenador pondrá a disposición el espacio de memoria extra para esta finalidad. Las líneas anteriores cierran y vuelven a abrir un 'canal'; la explicación de cómo los canales se da en el capítulo separado sobre el tema y también en la Sección de Referencia (bajo la palabra clave OPEN). El canal 1Ø3 es el canal normal 'por omisión' para la producción de sonidos.

El número de fases especificado por el comando SET SOUND BUFFER puede ser cualquiera entre 1 y 255, aunque, en la práctica, la complejidad de una envolvente queda limitada por la longitud de una línea de programas BASIC (25Ø caracteres). SET SOUND BUFFER sólo afecta a un canal que se abre posteriormente, no al que está ya abierto: lo que explica las líneas 1ØØ y 12Ø anteriores.

Por último, una sentencia SOUND puede contener la palabra STYLE seguida por un número de Ø a 255. La experimentación con los efectos de los diferentes estilos sonoros es particularmente interesante cuando se utiliza el generador de 'ruido' (fuente sonora 3). Para ello, ver la Sección de Referencia bajo el título 'Opciones Sonoras'.

CONVERTIR PROBLEMAS EN PROGRAMAS

Hemos empleado mucho tiempo dividiendo la programación en diferentes secciones y tratándolas una a una.

Sin embargo, aún no hemos estudiado cómo debería ser considerada unatarea en términos de la manera en que trabaja el ordenador, o cómo introducir tareas en un programa a fin de que se pueda entender fácilmente mucho tiempo después de haberlo escrito. Veamos este aspecto ahora.

El problema que se utiliza aquí es fácil de entender por ser simple. Una vez captado el principio de la planificación de programas se comprobará que es cada vez más fácil resolver problemas complejos.

EL PROBLEMA

Trataremos el problema de descubrir qué cantidad nos ahorraríamos al comprar un artículo que tiene un porcentaje de reducción. Para resolver este problema, sólo necesitamos saber dos cosas: el precio original y el porcentaje en el que el comercio ofrece reducirlo. A partir de estas dos cifras se puede hallar fácilmente la cantidad de reducción del precio y el precio que realmente habrá que pagar.

Lo primero que hay que hacer cuando se escribe un programa es decir exactamente lo que queremos que haga. En este caso, el programa:
(1) tomará dos números que escribimos, el precio antiguo y el porcentaje;
(2) determinará qué suma (en libras) equivale al porcentaje del precio;
(3) restará el descuento del precio antiguo para hallar el precio real, y (4) nos indicará cuales son los resultados, y nos pedirá si queremos que hagamos lo mismo con cualquier otro precio. Es bastante sencillo. La etapa siguiente es decir cómo debe hacer el programa el trabajo. Hay siempre diversas maneras de agrupar un programa, pero existen algunos principios generales útiles.

PROGRAMAS MODULARES

Un programa limpio y bien escrito debe ser modular: es decir, con partes claramente estructuradas e interconectadas. La razón de esto es que necesitaremos definitivamente comprender el programa una vez escrito. No sirve para nada escribir línea tras línea del BASIC si queremos cambiar algo después y no podemos comprender el programa. La mejor manera de considerarlo será la de plantearse como objetivo, al escribir un programa, que funcione correctamente y con el menor esfuerzo posible, y poder entenderlo.

rápidamente una vez que hemos terminado. Cuanta mayor sea la claridad con que escribimos el programa, más fácil será determinar tanto los problemas iniciales como los inconvenientes del programa que se hacen evidentes una vez utilizado durante algún tiempo. El orden de un programa modular es algo similar a esto:

(1) declaraciones de variables globales;
(2) el programa principal (controlador);
(3) una sentencia END que marca el punto en el que el ordenador dejará de ejecutar el programa; no es necesario que sea la línea final en orden numérico;
(4) funciones que manejan parte de la tarea principal;
(5) sentencias DATA si se utilizan (pueden aparecer al principio).

Siguiendo una estructura como esta será mucho más fácil escribir un programa complejo y eficiente. Hará que consuma menos tiempo (y a veces sea menos frustrante) si, por ejemplo, se sabe exactamente en dónde colocar una función y buscarla más tarde.

LOS ELEMENTOS DEL PROGRAMA

En primer lugar deben colocarse las variables globales (es decir, las usadas en todo el programa); las razón es que si el ordenador llega a una línea que contiene una variable que no se ha declarado, puede dejar de trabajar por no entender el programa. Por tanto, si las declaramos todas juntas inmediatamente y al comienzo, no tenemos que preocuparnos ya más de ellas. Tenemos también reunidas todas las variables importantes, por lo que sabemos cual es cada una.

Otro tipo de variables es la local, que se utiliza en una función y sólo dentro de esa función. No es necesario declararlas hasta que la función haya sido escrita ... y entonces deben comenzar al principio de ella.

El programa principal viene entonces en segundo lugar, por dos razones. En primer lugar, un programa es más fácil de escribir si la parte que controla todas las demás partes aparece hacia el principio. En segundo lugar, se puede buscar cerca del comienzo del programa, incluso algún tiempo después de escribirlo, y leer sólo el programa principal para recordar lo que hace. El programa principal utilizará también algunas de las variables globales, si no todas, que han de declararse antes de que se utilicen. La parte principal del programa llama a sus programas y valores desde funciones. Una función puede ser considerada como una caja negra. El programa principal pone números o

series en la caja negra, que acto seguido devuelve otros nuevos. Cada función puede tratarse también como un programa separado. Puede contener sus propias funciones a nivel inferior. Esto permite que se diseñen programas con una jerarquía claramente entendida. La sentencia DATA no necesita necesariamente aparecer al final de un programa, pero todas ellas deberían estar evidentemente en el mismo lugar. Puede ser muy pesado buscar en una gran cantidad de ellas o contar cuántas hay. Al reunirlas se ahorra el tiempo empleado buscando si una de ellas es errónea.

La sentencia data puede usarse también dentro de una función, a condición de que sean locales de esa función.

Naturalmente, la sentencia END marca el final del programa controlador principal. Como se ha dicho antes, en este capítulo, no siempre es la última línea de un programa. Cuando se utilizan funciones o subrutinas, el ordenador 'salta' fuera del programa principal para usarlas, en vez de trabajar con ellas en orden secuencial. La sentencia END debe pues colocarse en donde termine realmente el programa controlador.

Por último, no se olvide incluir en todos los programas muchas líneas de comentario, colocadas de manera que se vean con claridad.

Algunos de los programas de este manual contienen muchos más comentarios que líneas del BASIC. Los comentarios sirven para que sepamos exactamente lo que está haciendo el programa y nos permite explicar en nuestro idioma lo que podría ser difícil de entender cuando volvemos a estudiar el programa en una fecha posterior.

No pensemos que la programación tiene que buscar obsesivamente la claridad. No es así. Pero al igual que al aprender a leer y escribir hemos tenido que seguir una cierta disciplina, también la necesitamos en la programación. Nos ayudará mucho a largo plazo si no olvidamos aplicar este capítulo siempre que escribimos en un papel un programa -por pequeño que sea- y a continuación lo pasamos al teclado.

EL EJEMPLO

Volvamos ahora al problema del precio.

En primer lugar, tenemos que decidir qué hará el programa principal. En este caso, simplemente nos pide que escribamos el precio de los artículos y el porcentaje de reducción, pasa estos valores a la parte de función del programa y nos da el resultado, terminando por preguntar si queremos hacer algo más.

A continuación, descubramos cuántas funciones necesitaremos. En este caso sólo necesitamos una, para manejar 'cuanto es el descuento' y 'cuánto costará', que son cálculos muy simples. Llamemos a la función DISCOUNT.

El paso siguiente es descubrir cómo el ordenador realizaría los cálculos.

En este programa, damos al ordenador un número -por ejemplo 1ØØ-, y un porcentaje -por ejemplo 2Ø %-. En primer lugar, el ordenador debe encontrar cual es el 2Ø % de 1ØØ.

Empieza encontrando el 1% -que es una centésima- y a continuación lo multiplica por el porcentaje. Por tanto el cálculo en cuestión sería simplemente:

(precio/1ØØ) * porcentaje.

Acto seguido la máquina necesita hallar qué costarían los artículos. Lo hace simplemente restando el descuento del precio original. Ahora tenemos ya alguna idea de las variables que necesitaremos. Son (junto con algunos nombres):

- El precio original - PRICE
- El porcentaje de descuento - DISC
- El descuento en libras - TOTDISC (descuento total)
- El precio con descuento - NEWPRICE

Las dos primeras variables se introducirán en respuesta a una sentencia INPUT. Por lo tanto no necesitamos declararlas. Ahora sabemos que lo único que se necesita es declarar dos variables globales al principio.

La función se puede escribir inmediatamente. No hay que decir al ordenador que estamos trabajando con porcentaje, no nos entendería.

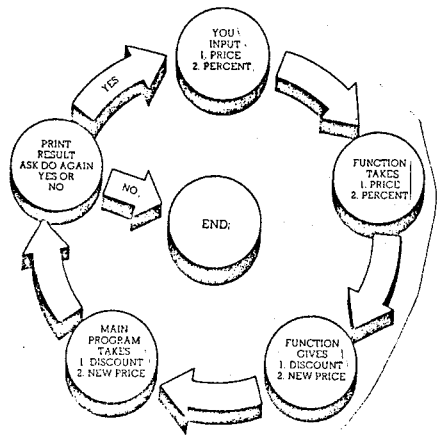
Un porcentaje es sólo un cierto número de centésimas.

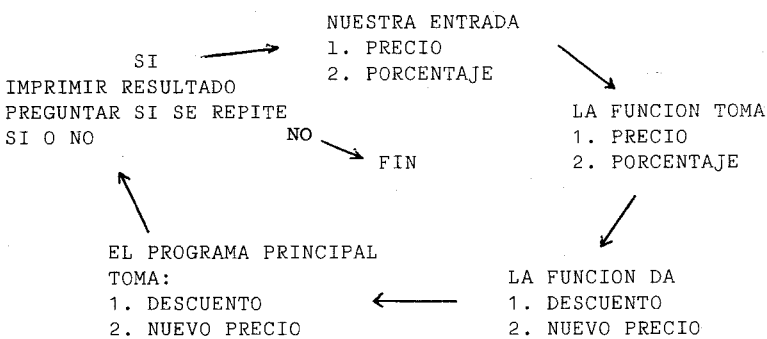
```
DEF DISCOUNT
  LET TOTDISC=(PRICE/100)*DISC
  LET NEWPRICE=PRICE-TOTDISC
END DEF
```

Ahora sabemos aproximadamente qué partes vienen, en qué orden... y exactamente qué estamos intentando conseguir, por lo que podemos empezar a poner el programa en BASIC. Con un programa más complejo tendríamos mucho más trabajo

que realizar, y probablemente escribiríamos muchas sentencias del BASIC en un cuaderno antes de empezar.

Antes de escribir nada en el ordenador, observemos este esquema





El diagrama anterior muestra el orden en el que aparecerá el BASIC real, mientras que las flechas muestran el orden en que actuará el programa si las seguimos desde la casilla superior. En primer lugar escribamos las declaraciones de las variables

```
100     NUMERIC TOTDISC, NEWPRICE
110     !     Then the main program:
120     DO
130           CLEAR SCREEN
140           INPUT PROMPT "Please type in the
              price ":PRICE
150           INPUT PROMPT "Please type in the
              percentage discount offered ":DISC
160           CALL DISCOUNT
170           CLEAR SCREEN
180           PRINT "A discount of ";DISC;" percent
              on goods priced at £";PRICE
190           PRINT "Would be £";TOTDISC;"."
```

```
11Ø A continuación el programa principal

14Ø "Por favor, escribe el precio"

15Ø "Por favor, escribe el porcentaje de
    descuento ofrecido"

18Ø "Un descuento de";"porcentaje de las
    mercancías con precio en libras".
19Ø "Sería libras".
```

```
200         PRINT
210         PRINT "The new price would be £"; 21Ø "El nuevo precio sería libras".
            NEWPRICE; "."
215         PRINT
218         PRINT
220         INPUT PROMPT "Would you like to 22Ø "Quieres saber más descuentos?".
            know more discounts? ": ANS$
230     LOOP WHILE UCASE$(ANS$(1:1)) = "Y"
240     END
```

Este es el programa principal. Ahora todo lo que se necesita hacer es añadir la función:

```
250     DEF DISCOUNT
260         LET TOTDISC = (PRICE/100) * DISC
270         LET NEWPRICE = PRICE - TOTDISC
280     END DEF
```

Si ejecutamos ahora (RUN) este programa, encontraremos que sirve, pero no es más que la estructura de un programa que necesita algo de contenido. El programa siguiente es prácticamente igual, pero mucho más claro que el que acabamos de escribir. Todo lo que no entendamos debe aparecer en las líneas ! (!Hay muchas!).

```
100     NUMERIC TOTDISC, NEWPRICE
120     CLEAR SCREEN
160     PRINT AT 9,5:"A PROGRAM TO WORK OUT
        DISCOUNTS"
170     PRINT AT 11,9:"ON MARKED RETAIL
        PRICES"
180     !  ------------------------------------
190     !  160 and 170 print a title in the centre
200     !  of the screen, which is neater
205     !  than the top. 220 and 230 ensure that the
207     !  title remains displayed for about 5
208     !  seconds.
210     !  ------------------------------------
220     FOR X = 1 TO 2500
230     NEXT X
240     CLEAR SCREEN
310     !  ------------------------------------
320     !  360 to 440 are the main program.
325     !  They control the function called
330     !  DISCOUNT. It is through this part
335     !  that you exit the program when
```

190 - 208. 160 y 170 imprimen un título en el centro de la pantalla, que es más claro que la parte superior. 220 y 230 aseguran que el título permanezca en pantalla durante unos 5 segundos.

320 - 335. 360 a 440 son el programa principal. Controlan la función llamada DISCOUNT. A través de esta parte es como se sale del programa cuando

```
340     !    you need to.
350     !
360     DO
365          CLEAR SCREEN
370          INPUT PROMPT "Please type in the
             price of the goods ":PRICE
375          PRINT
380          INPUT PROMPT "Please type in the
             percentage discount offered ":DISC
390          CALL DISCOUNT
400          PRINT "A";DISC;"percent discount o
             goods priced at £";PRICE
405          PRINT
410          PRINT "would be £";TOTDISC;"."
415          PRINT
420          PRINT "The new price would be
             £";NEWPRICE; "."
425          PRINT
430          INPUT PROMPT "Would you like to
             know any other discounts? ":ANS$
440     LOOP WHILE UCASE$(ANS$(1:1))="Y"
445     END
450     DEF DISCOUNT
460          LET TOTDISC=(PRICE/100)*DISC
470          LET NEWPRICE=PRICE-TOTDISC
480     END DEF
```

```
340  Es necesario
370  "Por favor, escribe el precio de las
     mercancías"
380  "Por favor, escribe el porcentaje de
     descuentos ofrecidos"
400  "Porcentaje descuento de mercancías con
     precio a libras".
410  "Sería libras".
420  "El nuevo precio sería libras"
430  "¿Quieres saber algún otro descuento?".
```

El programa anterior tiene un pequeño defecto. Cuando nos pide si queremos saber un descuento más, no puede hacer frente a una entrada errónea. Sería muy fácil escribir 't' en lugar de 'y', que tendría el resultado de poner fin al programa. Un buen programa debe tener posibilidad de descubrir los errores del usuario.

Por el momento, es posible que esto no represente una diferencia crucial para nuestros programas, pero sí lo será cuando se hagan más largos y más complicados. Es muy frustrante hacer un error de escritura que hace que el programa termine antes de que lo deseemos o simplemente se derrumbe. Derrumbarse es una palabra muy expresiva. Cuando un programa se derrumba significa que se hace algo tan equivocado que el ordenador deja de ejecutarlo y salta a modalidad inmediata, de manera que debemos empezar de nuevo la ejecución del programa. Esto puede deberse a toda clase de acontecimientos, pero sucede normalmente porque se escribe un programa que intenta hacer algo que es ilegal en BASIC. Esto se le conoce como error fatal.

El programa siguiente no proporciona todas las respuestas, pero nos muestra cómo asegurarnos que un programa no terminará o dejará de trabajar por haberse escrito una entrada errónea.

```
100   DO
110       !  ---------------------------------------
120       !  The whole program is a large DO/
125       !  LOOP. In 480 to 500 you will
130       !  notice another DO/LOOP inside
140       !  the large one. This is fine—as long
145       !  as the one loop is inside the
150       !  other. A loop inside a loop is
160       !  a nested loop. 175 to 290 are
165       !  also a nested loop.
170       !  ---------------------------------------
175       DO
180           INPUT PROMPT "Type in a whole
              number from 1 to 5: ": NUM
190           !  ---------------------------------------
200           !  180 asks you to give input as
205           !  shown. 290 makes sure it is an
210           !  acceptable number by check-
220           !  ing first that it is not bigger
225           !  than 5 or less than 1, and
230           !  secondly that it is a whole
235           !  number—this is done by
240           !  making sure the variable
245           !  NUM is not bigger or smaller
250           !  than its integer part. If NUM is
260           !  acceptable, the LOOP is not
270           !  repeated.
280           !  ---------------------------------------
290       LOOP WHILE NUM > 5 OR NUM < 1
          OR NUM < > INT(NUM)
300       SELECT CASE NUM
310       CASE 1
320           PRINT "Number 1"
330       CASE 2
340           PRINT "Number 2"
350       CASE 3
360           PRINT "Number 3"
370       CASE 4
380           PRINT "Number 4"
390       CASE 5
400           PRINT "Number 5"
```

120 –
165 La totalidad del programa es un gran DO/LOOP. En 48Ø a 5ØØ se observa otro DO/LOOP dentro del grande. Esto es correcto, a condición de que uno de los bucles esté dentro del otro. Un bucle en otro es un bucle encajado. De 175 a 290 hay también un bucle encajado.

2ØØ –
28Ø El 18Ø nos pide que demos las entrada indicada. 29Ø se asegura de que sea un número aceptable comprobando primero si no es mayor que 5 o menor que 1, y después si es un número entero: esto se hace haciendo que la variable sea segura. NUM no es mayor ni menor que su parte entera. Si NUM es aceptable no se repite el LOOP.

32Ø Número 1
33Ø Número 2
36Ø Número 3
38Ø Número 4
4ØØ Número 5

```
410         END SELECT
420         !  ------------------------------
430         !  Lines 300 to 410 select the
435         !  right response to your number.
440         !  If the number is not 1 the computer
445         !  drops down to the next line and
450         !  checks that, and so on. SELECT
460         !  CASE is dealt with on page 64.
470         !  ------------------------------
480         DO
490             INPUT PROMPT "Do you want to
                try another? ":ANS$
500         LOOP UNTIL UCASE$(ANS$(1:1))="Y"
            OR UCASE$(ANS$(1:1))="N"
510         !  ------------------------------
520         !  480 to 500 are the nested loop.
530         !  It will keep looping until the
540         !  first letter of your response,
545         !  converted to a capital letter, is
550         !  "Y" or "N". This is the same in
555         !  principle as the check used on the
560         !  variable NUM above, but the
565         !  method is different. 640 tells the
590         !  computer to go back to the
600         !  beginning as long as the first
610         !  letter of ANS$ is "Y". That means
615         !  that when it is "N" the machine
620         !  leaves the loop.
630         !  ------------------------------
640     LOOP WHILE UCASE$(ANS$(1:1))="Y"
650     END
```

430 - 460 Las líneas 300 a 410 seleccionan la respuesta adecuada a nuestro número. Si el número no es 1, el ordenador lo baja a la línea siguiente, lo comprueba y así sucesivamente. SELECT CASE se explica en la página 64.

490 "¿Quieres probar otro?"

520 - 620 480 a 500 son el bucle encajado. Continuará girando hasta que la primera letra de la respuesta, convertida en mayúsculas, sea "Y" o "N". Este es el mismo principio que la comprobación usada en la variable NUM anterior, pero el método es diferente. 640 dice al ordenador que vuelve al comienzo mientras que la primera letra de NASS sea "Y". Esto significa que cuando es "N" la máquina abandona el bucle.

Recordemos que el método aquí utilizado no es más que una manera de comprobar la entrada. Muchas de las fuentes que proporciona el ordenador pueden utilizarse para hacer esto de una u otra forma. El secreto de escribir programas eficientes reside en parte en inventar pequeños trucos como los anteriores -entre otras cosas- y eso forma parte también de la diversión.

Por último, los programas serán mucho más agradables de utilizar si los hacemos claros y limpios para la persona que los utiliza. Hacer que un programa tenga un aspecto 'limpio' incluye borrar la pantalla siempre que se llena o cuando el programa pasa de una etapa a otra, por ejemplo, de la entrada a la salida de los resultados.

Se pueden imprimir también las palabras limpiamente en la pantalla usando PRINT AT para señalar los márgenes y UCASE$ o LCASE$ para ordenar bien las series que escribe la persona que utiliza el programa. Como señales pueden usarse también los efectos sonoros y los colores.

Cuando se ha escrito algo realmente bien estructurado, unos programas útiles y atractivos, no sólo nos sentiremos satisfechos por lo realizado sino al mismo tiempo con la sensación del trabajo bien hecho y de que han aumentado nuestros conocimientos.

CARACTERISTICAS MINIMAS DEL BASIC

Como se sabe, el BASIC viene en muchos dialectos, como ocurre con los idiomas hablados. En el pasado, las compañías de informática inventaron sus propios BASIC con sus propias palabras e ideas. Pero el núcleo del lenguaje y su planteamiento han permanecido sin cambios.

El Enterprise sigue el dialecto del BASIC, propuesto por la asociación europea de fabricantes de ordenadores y el instituto nacional americano de normas (ANSI).

El BASIC standard está diseñado para eliminar los problemas asociados al uso de un lenguaje no estándar. Permite pues la transferencia de programas entre el Enterprise y otros ordenadores que utilicen el BASIC Standar. Este BASIC incluye también algunas características potentes que no se encuentran disponibles en los BASIC no Standar.

Se puede desear la ejecución de programas escritos en BASIC no Standar sin tener que revisarlos enteros y cambiarlos de manera detallada y complicada. Por suerte, la mayoría de los BASIC siguen el denominado a veces 'BASIC Mínimo', que fué ideado hace algunos años y adoptado por varios fabricantes. Es probable que hayamos oído hablar del BASIC Microsoft, que es uno de los que utilizan las características del BASIC Mínimo.

Para ayudar a la compatibilidad entre los distintos BASIC, el Enterprise proporciona las principales características del BASIC Mínimo así como las de BASIC Standar. Este capítulo detalla tales características.

RAMAS (BIFURCACIONES)

No, no se trata de una lección sobre poda de árboles. Pero es muy conveniente si pensamos por un momento que la estructura de un programa es algo similar a un árbol: con muchas ramas gruesas y otras más finas.

Algunos árboles tienen muchas ramas y otros sólo unas pocas. De igual manera se pueden estructurar los problemas de manera que pasen de una tarea a otra secuencialmente o (mejor) organizarlos de manera que tengan un 'tronco' y varias 'ramas' que brotan directamente del tronco. Sobre todo conviene evitar el salto de una rama a otra. Estas cosas deben ser tratadas con gran cuidado. Dos maneras de bufurcar un programa son GOTO y GOSUB. El otro método principal de bifurcación que se proporciona en este ordenador es el de las funciones, que se han tratado ya en la página 79.

GOTO y GOSUB hacen que el programa salte.

a otra parte del mismo desde la parte que ha alcanzado cuando llega a una de estas palabras.

GOTO

GOTO utiliza un número de línea para indicar al ordenador a dónde debe dirigirse a continuación. Algo así:

```
100   CLEAR SCREEN
110   INPUT PROMPT "Do you like computers? ":SAY$
120   IF UCASE$(SAY$(1:1))="Y" THEN
130       GOTO 220
135       !          GOTO 220 directs the
136       !          program to line 220.
140   ELSE IF UCASE$(SAY$(1:1))="N" THEN
150       GOTO 240
155       !          GOTO 240 directs the
156       !          program to respond to
157       !          a 'NO' answer.
190   ELSE
200       GOTO 110
205       !          This makes the program
206       !          begin again if you do
207       !          not answer yes or no.
210   END IF
220   PRINT "Oh good I'm okay here!"
230   END
240   PRINT "But I'm different. You'll see!"
250   END
```

```
110 "¿Te gustan los ordenadores?"
135 -
136 GOTO 22Ø dirige el programa a la línea 22Ø.
155 -
157 GOTO 24Ø dirige el programa para que se
    responda a una respuesta 'no'.
2Ø5 -
2Ø7 Esto hace que el programa empiece de nuevo
    si no se responde sí o no
22Ø "!Bien, estoy perfectamente aquí!"
24Ø "Pero soy diferente. !Ya lo verás!".
```

Así, GOTO dirige al ordenador a un número de líneas. Puede también utilizarse como un bucle. Si se cambia la línea 25Ø por

```
250   INPUT PROMPT"Would you like to do that again? ":A$
```

```
25Ø "¿Te gustaría hacer eso de nuevo?"
```

Y se añadiera la línea 26Ø:

```
260   IF UCASE$ (A$(1:1))="Y" THEN GOTO 100
```

(Y la línea 27Ø-END), entonces el programa volvería en bucle al comienzo si se respondiera con 'sí'. Esto es igual que realizar un DO/LOOP fuera de este programa, terminando con:

```
260   LOOP WHILE UCASE$(A$(1:1))="Y"
```

Pág. 128 Aunque el ordenador no descubriría lo que debe hacer exactamente de la misma manera con ambos métodos, el resultado sería el mismo. GOTO es una manera mucho más extraña de dividir los programas grandes en componentes separados que el uso de las funciones que describimos en la página 79.

La complicación de GOTO se debe especialmente a que rompe el orden normal de desarrollo del programa sin proporcionar una estructura alternativa.

Esta es la razón de que se pierdan programas!.

GOSUB GOSUB es similar a GOTO porque envía a el ordenador a un número de línea de un bit diferente del programa, pero con GOSUB esperamos un retorno.

Cuando se usa GOSUB se necesita dejar a un lado parte de un programa como una 'unidad': un bit que significa hacer sólo una cosa. A continuación, al término de ese bit de programa, se debe añadir la palabra RETURN para indicar al ordenador qué línea que sigue inmediatamente es la que contiene la palabra GOSUB.

El bit del programa entre la línea indicada por un comando GOSUB y el comando RETURN se denomina subrutina. Una función puede ser también una subrutina con el uso de la palabra CALL.

COMO EVITAR ERRORES Si no queremos equivocarnos y confundirnos, debemos recordar dos puntos siempre que se utilice GOSUB/RETURN:

Primero: un GOSUB no debe usarse nunca sin un RETURN. Si sucede esto, el resultado es que el ordenador nos dice que nos hemos equivocado. En segundo lugar, aunque hayamos dejado aparte la subrutina como unidad separada, el ordenador no lo ha hecho. No existe ningún comando especial para marcar el comienzo de una subrutina. Por esta razón, el ordenador puede pasar a la subrutina por error. Si sucede esto, y el ordenador lee la palabra RETURN antes que GOSUB, detendrá el programa porque no lo entiende.

La mejor manera de evitar este problema con el GOSUB es colocar todas las subrutinas al final físico de los programas. Entonces, inmediatamente antes de comenzar, se introduce la sentencia END o el comando STOP. Con esto se dice al ordenador que termine el programa antes de que tenga la probabilidad de leer de nuevo las subrutinas.

Los programas siguientes muestran un ejemplo de esta

técnica:

```
 80   CLEAR SCREEN
100   PRINT AT 1,5:"POPULATION TO LIVING
      SPACE"
110   PRINT AT 2,9:"RATIO PROGRAM"
120   PRINT AT 4,5:"This program works out the"
125   PRINT AT 5,5:"average ground space"
130   PRINT AT 6,5:"available to each person in"
140   PRINT AT 7,5:"a city, in square yards."
150   PRINT AT 8,5:"Remember—it's only an"
160   PRINT AT 9,5:"average, but if you compare"
170   PRINT AT 10,5:"lots of cities you can soon"
180   PRINT AT 11,5:"see which are overcrowded!"
183   FOR X=1 TO 5000
184   NEXT X
187   CLEAR SCREEN
190   PRINT AT 14,5:"Please type in the following
      information:"
200   PRINT AT 16,5:"A) Name of city,"
210   PRINT AT 17,5:"B) Population,"
220   PRINT AT 18,5:"C) Size of city in square
      miles."
230   FOR X=1 TO 2500
232   NEXT X
235   CLEAR SCREEN
240   INPUT PROMPT "CITY? ":CITY$
250   INPUT PROMPT "POPULATION? ":POP
260   INPUT PROMPT "SIZE? ":SIZE
270   GOSUB 1000
350   CLEAR SCREEN
360   PRINT AT 10,5:CITY$;", at a size of";
370   PRINT AT 11,5:SIZE;" square miles,"
380   PRINT AT 12,5:"and a population of ";POP
390   PRINT
400   PRINT AT 13,5:"would give everyone living
      there"
410   PRINT AT 15,12: SPA
420   PRINT AT 16,5:"square yards of ground space
      to live in - "
440   PRINT AT 17,12:"on average!"
450   PRINT
455   PRINT
460   INPUT PROMPT "Do you want to try any
      more? ":ANS$
470   IF UCASE$(ANS$(1:1))="Y" THEN GOTO 187
```

```
120 "Este programa halla el"
125 "Espacio medio de terreno"
130 "Disponible para cada persona en"
140 "Una ciudad, en yardas cuadradas"
150 "recuerda que no es más que una"
160 "Media, pero si comparamos"
170 "un montón de ciudades, muy pronto podremos
180 "ver que están superpobladas!"

190 "por favor, escribe la información
    siguiente"
200 "A) Nombre de la ciudad"
210 "B) Población"
220 "C) Tamaño de la ciudad en millas
        cuadradas"

360 "a un tamaño de"
370 "millas cuadradas"
380 "y una población de"
400 "daría cada persona que vive en ellas"
420 "yardas cuadradas de espacio de terreno
    en el que vivir"
440 "!como media!"

460 "¿Quieres probar algo más?"
```

```
480  !  ------------------------------
490  !  470 uses GOTO to send you back to
500  !  type more info if you answer yes.
510  !  ------------------------------
520  END
530  !  ------------------------------
540  !  The END is line 520 so that the
550  !  subroutine is not run into again.
560  !  This prevents errors.
600  !  Line 1000 begins the subroutine.
610  !  ------------------------------
1000 LET SQY=SIZE*1760^2
1010 LET SPA=SQY/POP
1080 RETURN
```

490 -

500 La línea 470 utiliza GOTO para que escribamos más información si la respuesta ha sido sí.

540 -

600 El END es la línea 520 por lo que la subrutina no se ejecuta de nuevo. Esto impide errores. La línea 1000 empieza la sub-rutina.

Así, el programa anterior nos enseña a utilizar GOSUB. Pero como podemos ver, es mucho menos claro que un programa que utiliza una función con CALL; observemos, por ejemplo, que a parte de la línea GOSUB no hay ninguna indicación del lugar en donde empieza la subrutina. Los programas que utilizan GOTO o GOSUB tienen más probabilidad de error que los que utilizan bucles y funciones estructurados.

LA SENTENCIA 'DIM'

Un repaso al capítulo sobre el almacenamiento de cantidades importantes de información nos permitirá recordar cómo reservar áreas de memoria para utilizarlas como matrices. Hay otra manera, más versátil, de hacer esto.

DIM A(10) reservará una matriz unidimensional con elementos numerados del 0 al 10. DIM A(4,4) reservará una matriz bidimensional usando el mismo sistema de enumeración de elementos. Se puede utilizar (X TO Y) para especificar una serie de números que se han de dar a los elementos de la matriz, como haríamos usando STRING o NUMERIC. Lo que falta, sin embargo, es la capacidad de decidir qué longitud tendrá cada elemento de la serie. Cuando escribimos programas muy largos que utilizan varias matrices de serie, comprendemos tal vez la utilidad que puede tener esto, porque podemos usarlos para conservar espacio de memoria.

El DIM sólo tiene compatibilidad con otros BASIC. He aquí un breve programa para demostrarlo. Observemos que DIM ARRAY$ (9) es igual que STRING ARRAY$ (9): el elemento final es siempre 0 salvo que se especifique un número diferente.

```
100    DIM ARRAY$(9)  !  a 10 element array
110    FOR N=0 TO 9
120      READ ARRAY$(N)
130    NEXT N
140    FOR N=0 TO 9
150      PRINT ARRAY$(N);" ";
160    NEXT N
170    DATA This,is,an,array,declared,
180    DATA using,the,DIM,statement,
190    DATA "—simple, eh?"
```

```
1ØØ  una matriz de 1Ø elementos

17Ø  esta, es, una, matriz, declarada,
18Ø  usando, la, sentencia, DIM,
19Ø  "Sencillo - ¿no?-".
```

CANALES

Para dar más flexibilidad al uso del ordenador y 'dispositivos' acoplados al mismo, el Enterprise utiliza un concepto llamado canales. Un canal es un paso especial abierto entre dos partes del ordenador. Una vez abierto el paso, puede darse la comunicación entre las dos partes del ordenador simplemente especificando el número de canal precedido, en las instrucciones, por el símbolo £(o '#' en algunos juegos de caracteres).

El BASIC del Enterprise no intenta fundarse en muchos supuestos sobre cómo utilizaremos el ordenador.

Después de todo, la potencia del ordenador se debe a su flexibilidad para actuar de acuerdo con nuestros deseos. Por esta razón, todas las instrucciones 'entrada-salida' nos permiten especificar canales si lo deseamos.

No obstante, dado que no queremos complicarnos con comandos innecesarios, el Enterprise los usa 'por omisión'siempre que puede. Normalmente cuando ordenamos por ejemplo PRINT, queremos que las palabras o números aparezcan en la pantalla de TV. Por tanto la impresión de "Hola" coloca el mensaje en la pantalla. Si por otra parte, queremos que el mensaje aparezca en la impresora (y si contamos con ella), podemos pasar el comando PRINT£1Ø4 : "Hola".

Cuando se enciende el ordenador, conecta automáticamente el **canal 104 a la impresora. Cualquier mensaje enviado a dicho canal es desviado a la impresora.**

Podemos hacer también referencia a los números de canal usando nombres de variables de manera que el programa pueda él mismo elegir el lugar al que transmitir un mensaje.

```
12Ø  INPUT PROMPT "Please enter a message: ":A$
13Ø  PRINT
14Ø  PRINT "Where would you like the message
     repeated?"
15Ø  DO
16Ø      INPUT PROMPT "Please enter Ø for screen
         or 1Ø4 for printer: ":CHANNEL
17Ø  LOOP UNTIL CHANNEL=Ø OR CHANNEL=1Ø4
19Ø  PRINT £CHANNEL ! Blank line to screen or
     printer
2ØØ  PRINT £CHANNEL:A$ ! Message sent to screen
     or printer
22Ø  END
```

12Ø "Por favor, introduce un mensaje:"

14Ø "¿En dónde te gustaría que se repitiera el mensaje?"

16Ø "Por favor, introduce Ø para la pantalla o 1Ø4 para la impresora:"

19Ø Línea en blanco a la pantalla o a la impresora

2ØØ Mensaje enviado a la pantalla o a la impresora.

Naturalmente, este programa sólo servirá si tenemos una impresora conectada y encendida. Pero podemos probar el mismo experimento con el canal 1Ø1 en vez de el 1Ø4; éste nos conectará con la pantalla habitual de gráficos. Incluyendo el comando GRAPHICS en el programa para comprobar lo que sucede.

Algo que conviene recordar cuando se utilizan canales es que todas las entradas-salidas trabajan de esta manera, incluídas cosas tales como el generador de sonido y la conexión con las grabadoras de cinta. Si somos descuidados en el uso de los números de canal podrían suceder cosas muy extrañas al ordenador. No es probable que causen un daño permanente, pero es posible que tuviéramos que empezar de nuevo el programa o RESET el ordenador. Para más detalles sobre los canales, consultar el capítulo sobre comandos de la Sección de Referencias, particularmente en el capítulo OPEN (página 168).

MANEJO DE LAS EXCEPCIONES

Para poder tratar los errores, manejar la red o algunos otros dispositivos, y también para hacer frente a determinadas condiciones especiales que son independientes del desarrollo normal de los programas, el BASIC del Enterprise dispone de algo llamado manipuladores de excepciones.

Las operaciones de manejo de excepciones se parecen un poco a las funciones. Pero al contrario de éstas, estos manipuladores no son normalmente activados por una referencia específica en un programa tal como un CALL.

En primer lugar, expliquemos algo esto. Una excepción es algo que ocurre independientemente del programa que se esté ejecutando en ese momento, pero que puede afectar al programa o que se le haga que lo afecte de alguna manera. La tecla 'stop' es una excepción porque el programa no necesita comprobar si hemos pulsado 'stop' pero queda afectado por ella. Los errores de programas son también excepciones.

Si ejecutamos un programa con algún tipo de error en él, el ordenador, si el error es de un tipo que detiene el programa, responderá a él con un breve mensaje y un número. En este caso, el número es lo que puede utilizarse como 'tipo de excepción' para dar al ordenador una clave de lo que debería hacer.

El BASIC nos permite también efectuar nuestras propias excepciones, que deben numerarse del 1 al 999 y utilizarse para hacer frente a cualquier entrada mal escrita (por ejemplo, un número demasiado grande o pequeño) u otra condición excepcional reconocida por un programa.

Empecemos el estudio del manejo de excepciones provocando una en un programa y tratándola después con el uso de las declaraciones del manipulador de excepciones.

```
50    WHEN EXCEPTION USE INPUT_ERROR
60    !
65    !   50 tells the computer to use the
70    !   handler (see below) if an exception
75    !   occurs. It is valid until the computer
80    !   reaches END WHEN in 145.
90    !
100   INPUT PROMPT "Please type a word: ":
      STRING$
110   IF VAL(STRING$)<>0 THEN
115       CAUSE EXCEPTION 10
120   ELSE
130       PRINT "Your word has been
          accepted."
140   END IF
```

65-80 50 indica al ordenador que utilice el manipulador (ver más adelante) si ocurre una excepción. Es válida hasta que el ordenador alcanza END WHEN en 145.

100 "por favor, escriba una palabra:"

130 "tu palabra ha sido aceptada"

```
145     END WHEN
150     END
160     !
170     !     260 to 310 is the part of the
180     !     program which copes with an
190     !     exception; in this case exception
200     !     number 10. This is caused if you
210     !     type in a number, not a word.
220     !     You may notice some parallels
230     !     between the exception handler and
240     !     calling functions.
250     !
260     HANDLER INPUT_ERROR
270         IF EXTYPE = 10 THEN
280           PRINT "That was not a word."
283           RETRY
285         ELSE IF EXTYPE < > 10 THEN
290           EXIT HANDLER
300         END IF
310     END HANDLER
```

170 - 240 Las líneas 260 a 310 forman parte del programa que maneja las excepciones; en este caso se trata de la excepción número 10. La razón es que se ha escrito un número no una palabra. Podemos ver algunos paralelos entre el manipulador de excepciones y las funciones de llamada.

280 "no era una palabra".

Es fácil comprobar que esto podría hacerse con otros métodos. La línea 110 hace en realidad que el ordenador registre un 'error' de acuerdo con nuestras instrucciones.

CAUSE EXCEPTION está ahí para enviar el programa a un manipulador en caso de error que el ordenador no reconocería normalmente, por ejemplo, una palabra que empezara con un número, como ha ocurrido arriba. Normalmente, el ordenador aceptaría números como una serie, como ya sabemos, pero el uso del VAL convierte en error una serie que empieza con un número, junto con la sentencia CAUSE EXCEPTION.

El VAL de una serie será Ø si no hay ningún número válido en la (serie excepto Ø) que preceda a la primera letra o a otro carácter no numérico.

RETRY significa "retroceda a la línea en la que empezó y repita". EXIT HANDLER es igual que EXIT DO o EXIT FOR pero sólo se refiere al bloque manipulador.

EXTYPE es el número de tipo de la excepción: en el programa tiene el número 10. EL EXTYPE varía según el error y, en el caso de la sentencia CAUSE EXCEPTION, según el número dado a la excepción provocada. Ver en la página 180, que trata de los mensajes de error. Otra palabra, EXLINE, da el número de línea en donde ha sucedido la excepción.

El programa anterior ilustraba un tipo de manejo de excepciones. En principio es similar a CALLing (llamar),una función (usándola como subrutina o módulo). Observemos que este método utilizaba las palabras WHEN EXCEPTION USE, que podemos considerar que significan 'en caso de que haya un error en algún sitio, utiliza estos manipuladores'. Este comando debe hacerse coincidir con una sentencia END WHEN.

LA RED

El Enterprise puede comunicar con otro ordenador con una simple conexión por hilo. Naturalmente, los otros ordenadores deben tener la misma facilidad (conocida como Red Inteligente) si se quiere que se encarguen de su parte en el diálogo, pero la conexión con otros Enterprises no representa ningún problema.

La ventaja de una red es que se pueden unir entre sí a muchos ordenadores, pero cuando un ordenador quiere hablar con otro, los ordenadores restantes permanecen fuera del diálogo, como ocurre con un sistema telefónico.

El Enterprise no tiene un número preestablecido como ocurre con el teléfono; cuando nos conectamos a la red debemos seleccionar un número para ella, escribiendo, por ejemplo:

```
SET NET NUMBER 5
```

El número de red puede ser cualquiera de 1 a 32. Se puede utilizar ASK NET NUMBER para recordar el número seleccionado.

TRANSFERENCIA DE PROGRAMAS

Una vez que se ha dado al ordenador un número de 'dirección' de red, es muy sencillo transferir programas entre las máquinas. En el ordenador que recibirá el programa, escribir LOAD "NET-Ø". Esto nos permite aceptar un programa enviado desde cualquier otro ordenador de la red.

Si por el contrario sólo se quiere recibir un programa enviado por otro ordenador, escribir LOAD "NET-n", en donde 'n' es el número de red del otro ordenador. Por ejemplo:

```
LOAD "NET-17:"
```

se intentará cargar un programa desde el ordenador que tiene el número de red 17.

Como es lógico, para que el otro ordenador transmita el programa deberá pasársele instrucciones. Habría que dar al ordenador número 17 la instrucción.

```
SAVE "NET-5:"
```

Con lo que se enviaría el programa actual desde el ordenador 17 al ordenador 5.

DIFUSION

El número de red Ø, es un caso especial. Este número no puede darse a ningún ordenador, pero se utiliza para operaciones generales de la red.

Para transmitir por la red, el número Ø se utiliza para indicar los mensajes 'difundidos'. Son los mensajes que no se dirigen a un determinado ordenador, sino que se difunden (como una señal de radio) a cualquier ordenador que esté conectado y escuchando. Esta facilidad es muy útil si tenemos un mensaje breve para todos los demás ordenadores, o si desconocemos los números de otros ordenadores de la red.

Un problema que tienen estas transmisiones difundidas es que no son un medio muy seguro de comunicación. Con una señal dirigida, enviada a un ordenador determinado, el mensaje se retransmite hasta que el ordenador receptor acusa recibo de la llegada segura del mensaje. Con las señales difundidas esto no es posible. Tampoco se puede reducir automáticamente la velocidad de transmisión a fin de hacerla coincidir con la capacidad del ordenador receptor para manejar la información.

Así pues, la instrucción

```
SAVE "NET-Ø:"
```

enviará un programa para que lo escuchen todos los ordenadores, pero es muy posible que el programa no sea correctamente recibido. Por otra parte, el ordenador transmisor no sabrá si la transferencia ha sido o no satisfactoria.

OPEN (ABIERTO)
A TODOS LOS
MENSAJES

El número Ø puede usarse también para recepción indiscriminada de todos los mensajes de la red. Cuando se abre un canal del ordenador (ver en la página 132, y el comando OPEN en la sección de referencia, página 168) a "NET-Ø:" éste se convierte entonces en un canal 'general' para operaciones de la red.

Los mensajes que se difunden desde cualquier otro ordenador por la red serán recibidos si hay un canal general abierto. Por otra parte, un mensaje que ha sido dirigido específicamente a nuestro ordenador, será recibido por nosotros, aunque no nos hayamos preparado para este mensaje abriendo un canal especial para comunicar con el otro aparato.

Esto no significa que recibamos mensajes privados entre otros dos ordenadores de la red. En un mensaje dirigido entre dos ordenadores, ningún otro ordenador puede escuchar.

Observemos que si contamos con un canal especial abierto a otra máquina, recibiremos mensajes a través de este canal (no a través del canal general) aunque estén en difusión.

CANALES DE COMUNICACION

Para un uso detallado de la red, un canal se abre específicamente para comunicación con una determinada máquina. Esto se hace usando las convenciones normales para apertura de canales

```
OPEN £11Ø:"NET-17:"
```

abrirá el número de canal 11Ø para comunicación bilateral con el ordenador número 17.

Este canal puede posteriormente usarse con las instrucciones de entrada-salida, por ejemplo

```
PRINT £11Ø: "This is message for computer 17"
LINE INPUT £11Ø:A$ ! A$ will receive line from computer 17
```

"Este es un mensaje para el ordenador 17". ... recibirá la línea del ordenador 17.

Dado que los mensajes enviados entre ordenadores de la red se mantienen en memoria intermedia (se mantienen en la memoria antes de la transmisión o después de su recepción) a menudo es necesario utilizar un par de instrucciones especiales.

FLUSH £chan forzará la transmisión de cualquier dato que se encuentre en una memoria intermedia a la espera de su transmisión. Los mensajes no se transmitirán normalmente hasta que no tengan 256 caracteres de longitud, o se haya cerrado el canal, por lo que debe utilizarse el comando FLUSH cada vez que se debe transmitir inmediatamente un mensaje breve.

CLEAR £chan: NET : borra las memorias intermedias de entrada y salida. El ordenador no aceptará ningún mensaje de otro ordenador por un determinado canal mientras que esté borrada su memoria intermedia de recepción. Esto sirve para impedir la corrupción de datos antes de que hayan sido utilizados por el ordenador receptor.

Si existen datos que aún no han sido retirados de la memoria intermedia de recepción (con la instrucción INPUT, por ejemplo) y se pueden eliminar estos datos, se debe utilizar la instrucción CLEAR. Si aún no se han transmitido datos, utilizar FLUSH antes que CLEAR.

```
PRINT £11Ø:"Message for computer 17"
FLUSH £11Ø
CLEAR £11Ø:NET
```

"Mensaje para el ordenador 17"

En la selección de los números de canal para las operaciones de red, es conveniente utilizar un número de canal superior a 1ØØ (pero evitando los canales por omisión, ver OPEN página 168). Esto se debe a que el BASIC cerrará todos los canales 1-99 cuando borra las variables, lo que es probable que ocurra a menudo cuando se está en modalidad inmediata.

MANEJO DE LA RED SUBORDINADA

El uso más interesante de la red ocurre cuando un programa está escrito de manera que la comunicación por la red puede desarrollarse efectivamente como tarea subordinada durante el funcionamiento principal de un programa.

El sistema operativo del Enterprise utiliza un método conocido como 'interrupciones' cuando controla la red. Esto significa que los ordenadores pueden hablarse entre sí en su propio tiempo, de manera invisible para los usuarios de los ordenadores.

El BASIC proporciona un método para hacer frente a esta operación, a través de los manipuladores de excepción. Las excepciones son errores u otros acontecimientos, que interrumpen el desarrollo normal de un programa (ver el capítulo sobre manipulación de excepciones página 134).

Cuando se está ejecutando un programa del BASIC, y se han habilitado unas interrupciones desde la red con la instrucción SER INTERRUPT NET ON, en este caso, la recepción de un mensaje por un canal de la red provocará una excepción.

Al manipulador de excepciones destinado a llevar a cabo las operaciones de red, se le debe dar siempre la instrucción SET INTERRUPT NET OFF como primera línea del bloque del manipulador. Esto impedirá que el ordenador quede confundido al recibir una nueva interrupción mientras está dentro del manipulador de excepciones.

Una vez en el manipulador, las memorias intermedias de entrada de red son 'sondeadas' (comprobadas una a una) por la instrucción ASK NET CHANNEL. En un determinado momento, varios canales pueden estar manteniendo mensajes (y pueden venir más mensajes mientras el manipulador está funcionando). Después de la transferencia de datos desde un canal, se debe utilizar de nuevo la instrucción ASK NET CHANNEL hasta que devuelva el valor 255 ,indicando que no hay ningún mensaje más a la espera de su recogida.

SET INTERRUPT NET ON debe ser la última línea de instrucciones antes de abandonar el manipulador.

Observemos que los manipuladores de excepciones son puestos en marcha por 'software interrupts' (interrupciones del software). Estas no ocurren en la

modalidad inmediata del BASIC, y por tanto, para utilizar la técnica del manipulador de excepciones en las operaciones de red se debe estar ejecutando en BASIC un programa preferente o prioritario.

USO DEL CODIGO DE MAQUINA

El 'cerebro' del Enterprise es un microprocesador Z80. El Z80 puede ejecutar unas 500 operaciones específicas, cada una de las cuales lleva un número de código: un 'código de máquina'. Si se programa un procesador en código de máquina, lo estamos direccionando directamente, en su propio lenguaje, no a través del intérprete BASIC.

Hay dos razones principales por las que podríamos desear la inclusión de rutinas de código de máquina en nuestros programas BASIC. O podemos necesitar un poco de velocidad extra (especialmente cuando manejamos gráficos y sonido), o queremos utilizar una característica del hardware del Enterprise que no está soportada por el BASIC.

La programación en código de máquina es un tema muy amplio, y no se podría tratar en un sólo capítulo de este manual. Si el lector está interesado en el tema, hay publicados muchos manuales de programación del Z80.

El BASIC del Enterprise tiene varios comandos que nos permite construir rutinas en código de máquina (hexadecimal) y ejecutarlas desde el interior de un programa BASIC, aunque estos comandos no forman parte de la norma ANSI.

ALLOCATE (ASIGNAR)

En primer lugar, debemos reservar parte de la memoria para almacenar nuestro código; decidamos cuántos bytes de longitud va a tener nuestro código, y usemos a continuación: ALLOCATE number-of-bytes (número de bytes).

Pero observemos que ALLOCATE destruye todos los valores de variables almacenados, por lo que es mejor utilizarlo únicamente al comienzo de un programa.

CODE Y HEX$

La rutina en código de máquina se almacena -con un determinado nombre- con el uso del comando CODE:
CODE name (nombre de código) = rutina en códigos hexadecimales.
El nombre tiene el mismo formato que un nombre de variable.
CODE sólo puede almacenar una serie ,nuestros códigos hexadecimales deben ser convertidos en el formato adecuado usando HEX$, como sigue:
HEX$ ("hex, hex ...")
(!No olvidemos las comillas ni las comas que separan los valores hexadecimales!); o

HEX$ (cualquier expresión de serie)
Por ejemplo, una rutina que duplique un determinado número
TEST: 29 ADD HL, HL; sumar el número a él mismo C9 RET
- puede insertarse en el programa BASIC de esta manera:

```
100    ALLOCATE 2
110    CODE TEST=HEX$("29,C9")
```

Una vez ejecutada esta parte del programa, la rutina queda almacenada en la memoria que hemos reservado.

EJECUCION DE LA RUTINA

Hablando en líneas generales, nuestro código creará una nueva 'función'. En el capítulo sobre definición de funciones, las hemos dividido en dos tipos. Un tipo está destinado a calcular un resultado ,en esta categoría entran las funciones 'built-in' (incorporadas). Un buen ejemplo es SIN que calcula el seno de un ángulo determinado cuando escribimos algo así:

```
PRINT SIN (63)
```

El valor que debe procesarse (y que hemos puesto entre paréntesis) se denomina argumento de la función. Algunas funciones incorporadas no tienen argumento -por ejemplo RND- pero todas devuelven un valor, una 'respuesta'.

El otro tipo de función se parece más a un 'comando'. Un comando es una serie de operaciones que realizan algo, como borrar la pantalla o introducir la modalidad gráfica. Un comando no devuelve ningún valor.

La manera de ejecutarse la rutina depende del tipo que hayamos usado. Si devuelve un valor, como en nuestro ejemplo de duplicación de un número, utilicemos USR (nombre, argumento)
Por ejemplo:

```
PRINT USR (TEST, 2)
```

imprimirá 4 en la pantalla; mientras que

```
LET A=3*USR(TEST, 2)
```

asignará el valor de 12 a la variable A.
Observemos que el argumento de USR se pasa a HL en el comienzo de la rutina; y el valor devuelto por USR es el contenido del registro HL al termino de la rutina.
Si la rutina es una operación tipo comando, debemos utilizar CALL USR - **que** no devuelve un valor; como por ejemplo:

```
CALL USR(NAME,Ø)
```

Tenemos que dar también a USR un argumento, pero el valor no importa.
Los comandos pueden fusionarse como en el ejemplo siguiente:

```
CALL USR(CLEAR,Ø)+USR (GRAPH, Ø)
+USR(PICTURE,Ø)
```

WORD$ (PALABRA)

WORD$ convierte su argumento en una serie de dos bytes -LSB MSB- por lo que es útil para realizar saltos hacia atrás desde etiquetas. Por ejemplo,

```
WORD$(TEST)
```

devolverá la dirección inicial de la rutina TEST al formato correcto para saltos y llamadas en código de máquina.

SECCION DE REFERENCIAS

REGLAS DE BASIC

La sección de Referencias proporciona una guía de todas las palabras en BASIC disponibles en el Enterprise, junto con su finalidad y el método de uso. Algunas de ellas se han mencionado brevemente en la sección Tutorial, y otras ni se mencionan. Conviene que experimentemos con todas estas palabras y descubramos por nosotros mismos la potencia que tiene realmente el Enterprise. Si conocemos ya el BASIC, encontraremos que, en líneas generales, esta sección es la mejor vía para el BASIC Standard Intelligent (derechos reservados Intelligent Software Ltd, 1984) que se proporciona en el Enterprise.

REGLAS GENERALES

Las letras en mayúsculas y en minúsculas son intercambiables en las palabras clave e identificadores del BASIC, por ejemplo, FOR, For, for y fOr son todas la misma palabra.

Una línea de programa puede tener hasta 25Ø caracteres de longitud, con un número de línea de 1-9999. Puede incluir varias sentencias, separadas por comas; cualquier cosa que se permita después de THEN en una sentencia IF/THEN puede incorporarse a una línea de varias sentencias.

Un identificador puede tener hasta 31 caracteres de longitud, y todos los caracteres son significativos. El identificador puede contener letras, números, puntos y caracteres de subrayado; el primer carácter debe ser una letra.

El símbolo ! se utiliza para separar el resto de la línea como comentario. En modalidad inmediata, una coma al principio de la línea significa que el resto se pasa a través del sistema operativo sin interpretación.

El intérprete elimina los espacios que hay antes y después del número de línea y de la primera palabra clave así como al término de la línea. Entonces sangra (corre el margen) el programa para cada nuevo bloque. FOR, DEF, DO, HANDLER, SELECT y WHEN sangrarán la línea siguiente en dos espacios. ELSE CASE dentro de un bloque sangrado están colocadas dos caracteres a la izquierda. LOOP, END y NEXT terminan el sangrado. Los números de línea se imprimen corriéndose a la izquierda de manera que a la derecha estén alineados.

```
1    LET A=Ø
1Ø   DO WHILE A<1Ø
1ØØ    LET A=A+1
11Ø    SELECT CASE A
12Ø    CASE 1
13Ø      PRINT "first time"
14Ø    CASE ELSE
15Ø      PRINT "not first time"
```

```
13Ø  "primera vez"

15Ø  "no la primera vez"
```

```
160     END SELECT
170     PRINT A
180   LOOP
190   GOTO 1
1000  END
```

Para más datos sobre la sintaxis y las convenciones del BASIC, ver el borrador de propuesta para el BASIC Standard del comité X3J2/82-17 del ANSI.

Las palabras clave se dan en mayúsculas en negrilla en el margen izquierdo de la página. Los formatos de los comandos que utilizan la palabra clave se indican en letra normal y los ejemplos en cursiva.

PROGRAMAS MULTIPLES

El IS-BASIC del Enterprise permite que varios programas se encuentren al mismo tiempo en el ordenador. Cada programa tiene sus propios números de línea y sus propias variables.

Se puede hacer referencia a un programa por un número o por un nombre dado en una línea PROGRAM. Ver en particular los comandos CHAIN, EDIT, y PROGRAM.

En un determinado momento, uno de los programas (por omisión el programa Ø) es el 'actual' en el que operarán los comandos tales como LIST y RENUMBER. El número de este programa aparece en la 'línea de estado' de la parte superior de la pantalla.

El programa Ø puede utilizar unos 42 K de memoria. Otros programas se limitan a 32 K cada uno.

EXTENSIONES

Existe la posibilidad de proporcionar extensiones del BASIC, que pueden cargarse desde cassette o disco o incluirse en una unidad de almacenamiento añadida para el ordenador. La explicación de los comandos o funciones extra se encuentran en las instrucciones que acompañan a estos productos.

TIPOS DE DATOS

Se proporcionan dos tipos explícitos de datos: numéricos y de serie. Las variables numéricas tienen nombres que siguen las reglas de los identificadores (ver reglas generales). Los identificadores de las variables de serie deben terminar con el símbolo $.

Los valores numéricos se calculan en aritmética decimal codificada en binario, y se imprimen hasta con 1Ø dígitos. Los números se manejan en la gama de $1e^{-64}$ a $9.9999999e^{62}$.

Las series tienen una longitud máxima de 254 caracteres si son declaradas con esta longitud (ver STRING). Las subseries pueden

ser referenciadas en la forma string-id (x:y), que especifica una serie que comienza por el carácter número X y termina por el carácter número Y. Si se omiten X o Y, se encuentran por omisión al comienzo o final de la serie, respectivamente.

OPERADORES

Operadores aritméticos:

* - multiplicar
/ - dividir
^ - elevar a la potencia de
+ - más
- - menos

Operadores de serie:

& - concatenar

Operadores relacionales:

> - mayor que
< - menor que
= - igual a
>= - mayor que o igual que
<= - menor que o igual que
<> - no igual
AND - Y lógica (verdadero/falso)
OR - O lógica (verdadero/falso)
BAND - Y lógica binaria
BOR - O lógica binaria.

ABREVIATURAS

En estas referencias se utilizan las abreviaturas siguientes:

chan - número de canal
id - identificador (por ej., el nombre de la variable)
str - serie
var - variable
expr - operador relacional (por ej. >, >=, etc.)
para - parámetro.

COMANDOS Y SENTENCIAS

line-number
(número de línea)

líne-number texto
line-number espacio
line-number

Añade o sustituye una línea de programa. Si el número de línea va seguido únicamente por un espacio, se inserta entonces una línea que contiene un '!'. Si el número de línea no va seguido por nada, se suprime la línea. Sólo se ejecuta en modalidad inmediata. Borra las variables.

Obsérvese que todos los comandos o sentencias que borran variables, cierran también los canales abiertos en la gama de 1 a 99, ambos inclusive (ver OPEN).

ALLOCATE
(Asignar) ALLOCATE expr

Se utiliza en relación con subrutinas en código de máquina. Traslada hacia arriba la fuente del programa para crear un espacio con el número especificados de bytes, en donde se colocará el código de máquina del usuario. Coloca el contador de posición en el primer byte libre del hueco. Obsérvese que esto destruye todas las variables, por lo que sólo se debe utilizar al comienzo del programa.

ASK
(preguntar) ASK opción de máquina variable.
Consulta sobre alguna opción (por ejemplo KEY RATE); ver las secciones sobre 'Opciones de Máquina' 'Opciones de video' y 'Opciones de sonido'. Comparar también SET y TOGGLE. La variable tomará el valor actual de la opción de máquina.
Ejemplo ASK KEY RATE A
asigna a la variable A la velocidad de repetición actual del teclado.

AUTO Un comando especial de edición que imprime automáticamente números de línea. Sólo trabaja en modalidad inmediata.

El número de línea inicial de arranque es el 1ØØ.
El tamaño de paso por omisión es 1Ø. Las nuevas líneas sustituyen a las antiguas con los mismos números de línea.

AUTO puede cancelarse pulsando 'stop'

AUTO
AUTO AT 1ØØ STEP 1Ø
AUTO STEP 1ØØ

CALL
(llamar)

```
CALL función
CALL función (para-lista)
```

Se utiliza para llamar una función (tanto incorporada como definida por DEF), cuando no se requiere ningún resultado de la función.

Se evaluará cualquier expresión que siga a CALL y se ignorará el resultado. CALL USR(A,B) + USR(C,D) llamará pues dos programas USR en código de máquina.

Se puede ejecutar en modalidad inmediata.

CAPTURE
(capturar)

```
CAPTURE FROM #canal TO #canal
```

Captura la entrada procedente del primer canal y la sustituye por la entrada esperada del segundo. La entrada del segundo canal queda bloqueada hasta que se pulsa 'stop', surge una condición de fin de fichero en el primer canal, u ocurre un error. CAPTURE FROM un determinado canal puede terminarse también dando 255 (normalmente inválido) como canal TO (a) en una sentencia posterior.

CASE
(mayúscula o
minúscula) Ver el bloque SELECT

CAUSE EXCEPTION
(causar excepción)

```
CAUSE EXCEPTION expr
```

Provoca un error y lo asigna a la categoría dada por la expresión; los valores de usuario deben encontrarse entre 1 y 999, ya que éstos no serán nunca utilizados por el BASIC. La palabra EXCEPTION es opcional.

CHAIN
(encadenar)

```
CHAIN número de programa
CHAIN "nombre" (par -lista)
```

Se utiliza para ejecutar programas BASIC a partir del programa actual.

Los parámetros pueden pasarse por su valor de un programa a otro, por ejemplo:

```
CHAIN "Mi-Programa" (1, "Fred")
```

Ver también PROGRAM.

CLEAR
(Borrar)

```
CLEAR #canal
CLEAR ENVELOPE
CLEAR FKEYS
CLEAR FONT
CLEAR GRAPHICS
```

```
CLEAR £ canal:NET
CLEAR QUEUE número-fuente-sonido
CLEAR SCREEN
CLEAR SOUND
CLEAR TEXT
```

Borra diversas opciones. Se puede ejecutar en modalidad inmediata.

CLOSE
(cerrar) `CLOSE£canal`

Borra cualquier dato en las memorias intermedias (buffers) de salida, cierra el canal y deja libre las memorias intermedias.

CODE
(codificar) `CODE = serie`
`CODE nombre de variable=serie`

Se utiliza en relación con las subrutinas en código de máquina. Copia una serie en la posición indicada por el contador de posición actual. Si se da una variable, toma el valor del contador de posición. Este contador se deja señalando el byte que sigue a la serie, que se supone que contiene el código de máquina. El nombre de variable puede utilizarse posteriormente para llamar a la rutina o para formar la dirección de destino de saltos, etc.

CONTINUE
(continuar) Como comando en modalidad inmediata, repone el programa a la línea siguiente después de un comando STOP o de pulsarse la tecla 'stop'.

Se utiliza en un programa como salida de un manipulador de excepciones, y continúa en la sentencia que sigue a la que ha provocado la excepción.

COPY
(copiar) `COPY FROM £canal TO £canal`

Copia el contenido de un canal en un segundo canal (ambos deben estar abiertos). La copia termina en fin de fichero, en error o con la tecla 'stop'. La entrada por omisión es el canal Ø. La salida por omisión es el canal 1Ø4.

```
COPY
copia de £Ø  a £1Ø4

COPY FROM £5
```

copia de £5 a £1Ø4 . Exige que esté abierto el canal 5.

DATA
(datos) DATA lista de datos

Permite la inclusión de una lista de constantes, números y/o series, para posterior lectura (reading). Ver READ.

DATE
(fecha) DATE serie de fecha

Especifica la fecha actual que mantiene el ordenador. La fecha se incrementa automáticamente cuando la hora actual que mantiene el ordenador alcanza la medianoche.

La fecha se especifica en el formato internacional YYYYMMDD (año, mes, día).

DATE "19850727"
equivale al 27 de Julio de 1985.
Puede utilizarse en modalidad inmediata. Ver la función DATE$.

DEF DEF id-numérica = expresión
DEF id-numérica (lista-parámetros) = expresión
DEF id-serie = expresión-serie
DEF id-serie (lista-parámetros) = expresión-serie

Definición de función en una línea.

DEF AVERAGE (X,Y)=(X+Y)/2

El bloque DEF: se trata de un grupo de sentencias que puede llamarse como función que devuelve un valor en la expresión, o como sentencia de procedimiento. Hay varios cambios pequeños respecto a la definición del ANSI. Ver también CALL, EXIT DEF, NUMERIC y STRING.

Línea-def

cualquier número de sentencias de bloques
fin-línea

línea-def

DEF id-numérico
DEF id-numérico (lista-parámetros)
DEF id-serie
DEF id-serie (lista-parámetros)

fin-línea-def
END DEF

Si se pretende que la función devuelva un valor, este valor debe asignarse al nombre de función dentro del bloque DEF.

```
DEF ANSWER (A$)
  IF UCASE$(A$(1:1)) = "Y" THEN
    ANSWER = 1
  ELSE IF UCASE$(A$(1:1)) = "N" THEN
    ANSWER = Ø
  ELSE
    ANSWER = -1
  END IF
END DEF
```

El ámbito de variables en cualquier punto de un programa es dinámico, es decir, depende de qué líneas han sido ejecutadas, y no de la disposición estática del programa.

```
100    NUMERIC FRED
110    LET FRED = 1
120    CALL Q
130    PRINT FRED
140    END
200    DEF P
210      LET FRED = 123! This FRED is a global.   21Ø  Esta FRED es una variable global
220    END DEF
300    DEF Q
310      NUMERIC FRED! A local FRED.              31Ø  Una FRED local
320      LET FRED = Ø
325      CALL P
330      PRINT FRED
350    END DEF
```

En este ejemplo, FRED se utiliza como global y como local al mismo tiempo. Cuando se ejecuta la línea 21Ø, la FRED de 31Ø cambia a 123 y no a 1ØØ. El problema 123 en 1. En un lenguaje de ámbito estático, el programa imprimirá Ø y 123, esto podría suceder si se ejecuta el mismo programa bajo un complicador BASIC.

Todo lo declarado dentro de un bloque DEF es local respecto a dicho bloque, y se asigna a cada primera ejecución de la declaración después de la llamada. Cualquier cosa no declarada podría

ser local o global según la historia.

Es mejor declarar todas las variables al comienzo de cada programa o función a fin de evitar resultados inesperados.

```
100   CALL P ! This call of P has I as local to P.
110   LET I=9
120   CALL P ! This call of P changes the global I.
130   END
200   DEF P
210     LET I=6
220   END DEF
```

```
1ØØ  Esta llamada de P tiene I como local
     de P
12Ø  Esta llamada de P cambia el I global.
```

A fin de dar resultados consistentes, se debe añadir una línea

```
9Ø   NUMERIC!
```

al programa; esto hará que I sea global en ambas llamadas de P.

La memoria utilizada para el almacenamiento de variables locales queda libre cuando se llama a una función. Esta característica se puede aprovechar para el uso eficiente de la memoria del ordenador; por ejemplo una matriz de datos temporales puede encontrarse dentro de una función.

Como parámetro de referencia puede pasarse por prácticamente cualquier cosa. Normalmente, los parámetros se pasan por valor, lo que significa que las copias se pasan a la función y que cualquier operación dentro de la función no cambia las variables exteriores. Los parámetros de referencia toman su tipo del parámetro real, y cualquier cambio dentro de la función cambia también las variables exteriores.

```
100   DEF SWAP (REF A,REF B)
110     NUMERIC T
120     LET T=A
130     LET A=B
140     LET B=T
150   END DEF
200   LET X=99
210   LET Y=23
220   CALL SWAP (X,Y)
230   PRINT X, Y
```

imprime 23 99

Las matrices y funciones deben ser llamadas siempre por

```
100    NUMERIC A(10)
110    OPTION ANGLE DEGREES
120    DEF P(REF FN, X)
130      PRINT FN(X),
140    END DEF
150    LET A(2)=66
160    CALL P(A,2)
170    CALL P(SIN,30)
```

imprime 66 .5

El paso de funciones incorporadas y de usuario puede ser muy útil para el software de biblioteca. Una función de dibujo de gráficos puede tener pasada como parámetro la función que hay que trazar, una función de clasificación puede pasar como funciones la rutina de intercambio y comparación.

Las funciones pueden llamarse ellas mismas de manera recursiva.

DELETE
(suprimir)

DELETE descripción-línea To descripción-línea,...

DELETE descripción-línea - descripción-línea,...

DELETE nombre-bloque

Suprime líneas del programa. Sólo se ejecuta en modalidad inmediata. Borra las variables.

```
DELETE LAST
DELETE FIRST TO 100
DELETE 1 TO 199, 300, 500 TO 9999
```

La sintaxis aceptable es utilizada '-' en lugar de TO. Si se omite el primero (o el último) número de una serie, el valor por omisión es la primera (o última) línea del programa.

por ejemplo DELETE FIRST-1ØØ,5ØØ-LAST

o DELETE TO 1ØØ,5ØØ-

para DELETE FIRST TO 1ØØ,5ØØ TO LAST

Se pueden suprimir las líneas que definen una función P con DELETE P. DELETE retirará por sí misma todas las líneas del programa; se puede suspendar con la tecla 'stop'.

DIM DIM lista-matriz

Declara matrices numéricas o de serie; el límite inferior

señala por omisión a Ø si no se especifica. Se permiten una o dos dimensiones. No se puede especificar la longitud máxima para una serie usando DIM, por lo que se utiliza el valor por omisión de 132 caracteres (comparar STRING).

```
DIM A(1 TO 1Ø),FRED$(9),B(-7899 TO-789Ø)
```

Nota: todo lo anterior tiene 1Ø elementos.

DISPLAY

DISPLAY £ canal:AT a FROM b TO c

Define una ventana para visualizar un segmento de un texto o una página video de gráficos. La fila de pantalla 'a' es la posición en la que estará situada la línea superior del segmento. Los párametros 'b' y 'c' son filas de caracteres en la página que debe visualizarse, y definen las líneas superior e inferior del segmento. La numeración de filas de caracteres sigue las convenciones del texto, tanto si la página visualizada es de texto como si es gráfica. Ver PRINT.

DISPLAY GRAPHICS

Introduce 2Ø líneas como gráficos y visualiza la página de gráficos anterior si había una abierta. (£ 1Ø1). No borra la página de texto.

DISPLAY TEXT

Coloca toda la pantalla en modalidad de texto y visualiza la página completa de texto si estaba previamente abierta (£ 1Ø2). No borra la pantalla de gráficos.

Si sólo estaba abierta antes una pequeña página de texto, se borra y se abre una nueva página de texto de tamaño completo.

```
do-line
cualquier número de sentencias o bloques
loop-line

do-line
DO
DO WHILE expresión-relacional
DO UNTIL expresión-relacional

loop-line
LOOP
```

pág. 159 LOOP WHILE expresión-relacional
LOOP UNTIL expresión-relacional

La estructura de un bucle se define como un bloque, con una línea DO, el cuerpo del bucle y una línea LOOP. No se pueden colocar ni DO ni LOOP en una línea condicional.

```
DO WHILE A>3 AND A<10
  LET A=A+1
  PRINT A
LOOP
```

No se puede transferir el control del exterior al interior de un bucle. Ver también EXIT DO.

EDIT EDIT número-programa
EDIT "nombre"

Convierte el programa especificado en el actual, de manera que actúen con él LIST, RENUMBER, RUN, etc. Sólo trabaja en modalidad inmediata. Ver CHAIN, INFO y PROGRAM.

ELSE Ver IF

END
(fin) Suspende la ejecución, marca el fin del programa. END DEF, END HANDLER, END IF, END SELECT y END WHEN marcan también el fin de sus bloques correspondientes.

ENVELOPE
(envolvente)

ENVELOPE canal: NUMBER
a,b,c,d,e,f,g,h,i;...; RELEASE; j.k.l.m;...

Define una envolvente de sonido que se utilizará junto con una sentencia SOUND de control. El número 'a' que identifica la envolvente debe encontrarse en la gama de Ø a 254.

Los parámetros 'b','c','d', y 'e' definen la primera fase de la envolvente; 'b' da el cambio de tono en semitonos (se permiten decimales), 'c' y 'd' especifican el cambio de volumen de los altavoces de la izquierda y de la derecha, respectivamente, y 'e' da la duración de la fase en 'tics' (tic tiene 1/5Ø de segundo).

Los valores de 'c' y 'd' se encuentran en la gama de Ø a 63; especifican el cambio de volumen como proporción del volumen global máximo permitido por la sentencia SOUND. Un valor de -63 desconectará el sonido.

(se ignora cualquier sobremodulación); se supone que el sonido está 'off' al comienzo de la envolvente. Si no se utiliza equipo estéreo, el volumen en cualquier momento dependerá de la suma de los valores (de acuerdo con las sentencias SOUND y ENVELOPE) para los altavoces de la izquierda y de la derecha.

La fase siguiente se define por 'f', 'g', 'h', e 'i'... Para el número de fases posibles, ver SOUND BUFFER; en 'Opciones de sonido'.

RELEASE es opcional; puede ir seguida por cualquier número de fases con sus parámetros separados. Las fases 'release' se ejecutan después de concluidas las fases anteriores, o al terminar la duración de SOUND si no hay ningún sonido que siga en ese mismo canal.

EXIT DO
EXIT FOR
EXIT DEF
(salida)

Interrumpe los bloques FOR, DO o DEF. no es válido salvo que esté dentro del tipo adecuado de bloque.

EXIT HANDLER
(Manipulador
de salida)

Interrumpe un manipulador de excepciones, que propaga la excepción al entorno circundante. Esto provocará la activación de otro manipulador de excepciones, que puede ser un manipulador de usuario o un manipulador por omisión.

EXT

EXT serie-parámetros

Pasa una serie por el sistema operativo, la cual se hace después pasar a los programas exteriores válidos de la memoria (que puede ser memoria ROM o RAM). La serie es acto seguido interpretada por estos programas según sea apropiado.

Normalmente la primera palabra de la serie de parámetros especifica un comando que ha de operarse o un nuevo programa al que ha de saltarse.

Por ejemplo:

```
EXT "WP"
```

salta al procesador de palabra incorporado del ordenador.

La palabra "HELP" tiene un significado especial ya que exige que todos los programas exteriores respondan con sus nombres. Las respuestas de los programas exteriores se envían al canal del sistema por omisión. Ver la opción de máquina DEFAULT CHANNEL (canal de omisión).

Los programas adicionales de aplicación y servicio definirán sus propios nombres de comandos y requisitos de parámetros.

Con frecuencia, estos programas responderán a la instrucción

EXT "HELP NAME"
(en la que "NAME" es el nombre principal del programa) dando una lista de los comandos de serie disponibles.

El mismo efecto que con EXT se puede obtener en modalidad inmediata comenzando la línea con una coma. En este caso no se necesitan comillas alrededor de la serie de parámetros.

HELP NAME

FLUSH (limpieza)

FLUSH canal

Obliga a los datos que quedan en una memoria intermedia de canal aunque sean transmitidos, sin cerrar el canal ni señalar fin de fichero. Esta operación sólo es adecuada para ciertos dispositivos (por ejemplo NET). Puede utilizarse en modalidad inmediata.

FOR

for-line

cualquier número de sentencias o bloques

linea-siguiente

for-line:
FOR variable-simple= expresión TO expresión STEP expresión.

STEP puede omitirse -el valor por omisión de STEP es un 1.

linea-siguiente
NEXT
NEXT variable

La estructura de un bucle FOR se define como bloque, con una línea FOR, el cuerpo del bucle, y una línea NEXT. FOR y NEXT no pueden colocarse en una línea condicional. Se permite en BASIC Mínimo

```
FOR Y=0 TO 10 STEP 2
  PRINT Y
NEXT Y
```

El valor de la variable de control una vez que ha definido el bucle es el valor de terminación más la expresión STEP,

es decir: Y tendrá el valor de 12 en el ejemplo anterior.

Los bucles encajados FOR no pueden utilizar la misma variable de control. Las expresiones de límite e incremento se copian en una memoria local oculta al ejecutarse la línea FOR; estos valores no pueden ser cambiados por el cuerpo del bucle. El control no puede ser transferido del exterior al interior de un bucle. Ver también EXIT FOR.

GET

GET canal:-id-serie

Recibe un único carácter de un canal y devuelve una serie nula ("") si no hay ningún carácter disponible.

Señala por omisión el canal 1Ø5 (KEYBOARD), y en uso simple es similar a la función INKEY$.

GOSUB

GOSUB número-línea

Llama a la subrutina que comienza en el número de línea especificado.

GOTO

GOTO número-línea

La ejecución del programa continúa en el número de línea especificado. Puede utilizarse para sacar los bloques FOR, DO, HANDLER o DEF, pero esto no se recomienda.

GRAPHICS (gráficos)

GRAPHICS

GRAPHICS HIRES/LORES número-cantidad-color

GRAPHICS ATRIBUTE

El comando GRAPHICS tiene el efecto de cerrar y abrir de nuevo las páginas de gráficos por omisión y de texto (1Ø1 y 1Ø2); visualiza en la mayor parte de pantalla la página de gráficos por omisión, pero con cuatro líneas de texto en la parte inferior.

GRAPHICS establece también el canal por omisión (1Ø1) para opciones de máquina de video talescomo PALETTE.

Los números válidos de cantidad-color son 2,4,16 y 256. Si no se especifica nada para la cantidad de color ni la opción HIRES/LORES, se volverán a utilizar los valores usados para el anterior comando GRAPHICS. Para el significado de estos valores, ver 'Modalidad Video' en la sección 'Opciones Video'. Al principio, GRAPHICS selecciona una página de gráficos de alta resolución con cuatro colores.

GRAPHICS ATRIBUTE selecciona una modalidad de 'atributos'.

de gráficos en la que cada célula de color (8 puntos de anchura por un punto de profundidad) puede contener un color 'tinta' y un color de 'papel'. Esta modalidad combina una paleta de 16 colores con la misma resolución que los gráficos HIRES de 4 colores (la resolución y la cantidad de color no pueden ser especificadas por el usuario). Pueden darse tanto los comandos de impresión como los de dibujo aunque puede haber efectos interactivos entre los colores. Para un uso flexible de esta modalidad, ver la opción de video ATTRIBUTES. Ver DISPLAY GRAPHICS.

HANDLER

HANDLER nombre-manipulador

sentencias del manipulador de excepciones.

END HANDLER

El bloque HANDLER se utiliza para hacer frente a las excepciones del programa causadas por errores, el comando CAUSE EXCEPTION o las interrupciones de máquina.

El manipulador que ha de utilizarse se especifica con el nombre-manipulador que se da en el bloque WHEN actual.

Ver CONTINUE RETRY, EXIT HANDLER y las funciones EXLINE y EXTYPE.

El control se puede transferir a un manipulador de excepciones únicamente como resultado de una excepción (no con un GOTO ni GOSUB).

Si ocurre una excepción dentro del manipulador de excepciones, el efecto es similar a EXIT HANDLER; ya que el control pasa al siguiente nivel exterior de manipulador (tal como especifique el siguiente nivel exterior del bloque WHEN). Sin embargo, los valores anteriores de EXTYPE y EXLINE habrán sido sustituidos por los nuevos.

IF
(Si...entonces...)

IF expresión-relacional THEN número-línea

IF expresión-relacional THEN sentencia-simple

Las sentencias que no se permiten en una línea IF son DATA, DEF, END, DIM; NUMERIC, STRING, otro IF, o cualquier sentencia que introduzca un bloque.

```
IF A>=3 AND A<=9 THEN 100
IF A>=3 THEN GOTO 100
```

linea-if

cualquier número de sentencias o bloques.

opción o bien-líneas- si cualquier número de sentencias o bloques

opción o bien-línea
    cualquier número de sentencias o bloques.
fin-línea-if

línea-if
IF expresión relacional THEN

o bien-línea-if:
ELSE IF expresión relacional THEN

Puede haber cualquier número de líneas ELSE IF

línea-o bien
ELSE

fin-línea-if
END IF

Los bloques IF pueden contener cualquier sentencia que no esté limitada a modalidad inmediata.

```
IF A<10 THEN
  PRINT A
ELSE IF A>30 AND A<=40 OR A>50 THEN
  PRINT A+100
ELSE
  PRINT B
END IF
```

Las líneas ELSE y ELSE IF pueden utilizarse para dividir los bloques en subbloques con los significados habituales. ELSE sólo puede utilizarse una vez, pero ELSE IF puede usarse siempre que se necesite. No se puede transferir el control del exterior al interior de un bloque IF.

IMAGE
(imagen)

IMAGE: especificación-formato

Se utiliza junto con el formato PRINT, para controlar el formato de la salida. La especificación de formato es una serie que contiene caracteres que, en este contexto, tienen los significados siguientes:

caracteres de formato numérico:
, - imprime una coma
$ - imprime el signo del dólar flotante antes del signo

- - imprime un espacio flotante o el signo '-'
+ - imprime un signo flotante '+' o '-'
% - imprime un dígito, incluidos los ceros a la izquierda
£ - imprime un dígito o espacio, los ceros a la izquierda de una coma decimal.
* - imprime un dígito o un '*' a la izquierda.
. - imprime una coma decimal
^ - imprime la parte del exponente; mínimo cuatro caracteres.

Si el número no entra en el espacio de formato, se genera un error.

Caracteres en formato de serie:

< - justificación a la izquierda de la serie, en el campo definido por '£' caracteres
£ - imprime un carácter
> - justificación a la derecha de la serie

El carácter de formato 'justificar' debe iniciar el campo; si no se utiliza este carácter, la serie se coloca centrada.

El formato de la línea IMAGE comienza inmediatamente después de ':' y termina con el último carácter impreso en la línea o con el signo de admiración que comienza un comentario de fin de línea.

INFO
(información) Imprime la cantidad de memoria que hay en el sistema y el número de bytes no utilizados. También se imprime una tabla de información sobre los programas que hay en memoria, en la forma siguiente:

número-programa	número de bytes del programa	primera línea del programa

INFO borra todas las variables. Sólo se ejecuta en modalidad inmediata.

INPUT
(entrada) INPUT canal, IF MISSING acción, ET expr-fila, expr-columna, PROMPT serie: lista-variable.

Lee los datos del canal en una lista de variables. El canal por omisión es el editor (canal Ø). Los elementos de datos leídos para que coincidan con las variables de la lista de variables deben ir separados por comas.

"¿Por favor, introduce el número siguiente?"

Las parte IF MISSING y PROMP pueden estar en cualquier orden, o no existir. El mensaje de solicitud de entrada por omisión es "?".

PROMPT sustituye la solicitud por omisión con la serie.

La opción AT (con expr-fila, expr-columna) es independiente de la opción PROMPT.

IF MISSING se utiliza si ocurre en el canal una condición de fin de fichero, o si hay una entrada nula cuando se espera una entrada numérica. La acción tomada entonces sigue la misma regla que con READ.

Ver también LINE INPUT.

LET LET lista-variable=expresión

Una simple asignación; LET es opcional a menos que el nombre de la variable equivalga a una palabra clave. El listado o preservación del programa hace que se inserte LET de manera que el programa siga la norma. Se puede ejecutar en modalidad inmediata.

Se puede asignar un valor a varias variables:

```
LET A, B(4), C=0
A_VAR=A_VAR+1
A$,FRED$="He said"&"Don't"&". "&FRED$(1:1) "él dijo" "¿no es así?"
LET INPUT=3
```

LINE INPUT (entrada de línea) Similar a INPUT, pero lee una línea completa (incluidas las comas, etc.) de cada elemento de la lista de variables, que sólo puede contener variables de serie.

LIST (listar)
LIST #canal:descripción-línea TO descripción-línea
LIST #canal: descripción- línea -descripción-línea
LIST nombre-bloque

Lista la totalidad de una parte del programa. Puede ser suspendido por la tecla 'stop' o colocarse en pausa con la tecla 'hold'. Sólo se ejecuta en modalidad inmediata. El canal por omisión es #0.

pág. 167 TO puede ser sustituido por '-'. Comparar DELETE.

por ejemplo, LIST FIRST-1ØØ,5ØØ-LAST
para LIST FIRST TO 1ØØ,5ØØ TO LAST

LLIST LLIST expresión-lista

Idéntico a LIST, pero señala por omisión a £1Ø4, el canal de listado de la impresora.

LOAD
(cargar) LOAD £canal:nombre de fichero

LOAD nombre-dispositivo

Carga un fichero desde el canal indicado o, si no se especifica ninguno, desde el canal 1Ø6 (cassette o discos si van acoplados). Si se escribe LOAD sin parámetros, señala por omisión al fichero 'boot' (en cinta, es el primer fichero que se encuentra).

```
LOAD
LOAD "My_Program"                "Mi_Programa"
LOAD "NET-Ø:"
```

Si el fichero contiene un programa BASIC, sustituye entonces el programa actual de la memoria. Si se han preservado como un fichero múltiples programas BASIC, éstos sustituirán a todos los programas de la memoria y volverán al programa Ø.

El fichero puede contener otros tipos de datos y programas (como ampliaciones u otros programas de aplicación) que serán manejados automáticamente por el sistema operativo.

Ver OPEN, donde se define un nombre-dispositivo y un nombre-fichero.

Borra las variables. Sólo se ejecuta en modalidad inmediata, pero ver RUN que puede usarse en una sentencia de programa.

LOOK
(observar) LOOK£canal ATx,y:v.

Asigna a la variable'v' el color de paleta en el punto (x.y) de la página de gráficos standard u otra página especificada por la expresión del canal. Tanto la expresión del canal como la parte AT son opcionales. Si se omite esta parte AT, se utilizará la posición del haz actual (cursor). Obsérvese que el uso del AT apagará el haz y

pág. 168 lo pasará a (x,y)

LOOP Ver DO

LPRINT LPRINT expresión-impresión
idéntica a PRINT, pero señala por omisión a &1Ø4, el canal de listado de la impresora.

MERGE
(fusionar)
MERGE £canal: nombrefichero

Fusiona el fichero de disco, cinta u otro canal con el fichero actual. Las líneas del nuevo programa sustituirán a las que tienen el mismo número en el fichero actual. Sólo se ejecuta en modalidad inmediata. Borra las variables.

NEW
(nuevo) Suprime todo programa actual. sólo se ejecuta en modalidad inmediata. Borra las variables.

NEW ALL
(todo nuevo) Borra todo los programas de la memoria del ordenador y vuelve al programa Ø.

NEXT
(próximo) Ver FOR

NUMERIC
(Numérico)
NUMERIC variable/lista-matrices

Declara variables o matrices numéricas (que son locales si se declaran dentro de una función DEF). El límite inferior por omisión será Ø. Comparar DIM.

NUMERICI,A(1Ø),B(-1Ø TO 2Ø, 2 TO 4)

ON ON expr GOTO lista-número-línea

ON expr GOSUB Lista-número-línea

Evalúa una expresión, convierte el resultado en un entero y utiliza el resultado entero N para elegir el Nº número de línea en la lista (el recuento empieza a partir de 1). La ejecución del programa se reanuda entonces a partir de esa línea. Si no existe ningún Nº número de línea, no se emprende ninguna acción. Utilizar el bloque SELECT o IF para conseguir un programa más legible.

OPEN OPEN£canal: NAME dispositivo/nombrefichero ACCESS modalidad

OPEN£canal: dispositivo/nombrefichero

En la modalidad de acceso es INPUT o OUTPUT ACCESS OUTPUT intenta crear un nuevo fichero (si está en cinta o disco); ACCESS INPUT intenta utilizar un fichero existente. Para dispositivos como video, puede utilizarse cualquiera de los dos. La presunción por omisión es INPUT.

Conecta un dispositivo, o un fichero en el caso de cinta y disco, a un canal. Los comandos pueden entonces leer, escribir o manipular de cualquier otro modo datos desde y al dispositivo (o fichero) haciendo referencia al número del canal.

OPEN#8:"DISK-0:TEST_PROGRAM"ACCESS OUTPUT

En un momento dado, sólo puede conectarse a un determinado canal un único dispositivo (o fichero), aunque se puede utilizar un único canal para acceder a varios dispositivos (ficheros), uno detrás del otro.

Para desconectar el canal de un dispositivo (o fichero), usar el comando CLOSE.

Los números de canal van de Ø a 254 (255 es un número de canal inválido que se utiliza para fines especiales).

El sistema BASIC utiliza varios canales como presunción por omisión cuando no se especifican los canales en las sentencias. Estos canales son:

Ø - se utiliza para entrada de comandos y salida de texto normal (por ejemplo para LIST y PRINT). Este canal se conecta a la reposición (o al encender el aparato) al dispositivo "EDITOR".

El dispositivo "EDITOR": utiliza por su parte los dispositivos "KEYBOARD:" y "VIDEO:", dispuestos en modalidad de video Ø con tamaño de página 24, 4Ø.

Este canal es el canal por omisión supuesto para COPY FROM y REDIRECT FROM.

El canal Ø se abre automáticamente a la reposición y permanece abierto hasta que se cierra explícitamente.

Obsérvese que el canal Ø se especifica como canal de comandos por omisión para compatibilidad con ANSI. Otros canales por omisión están numerados por encima del 1ØØ para dejar los números más simples de canal para definición por el usuario.

1Ø1 - Se utiliza para sentencias de entrada y salida de gráficos. Este canal se conecta, al utilizarse por primera vez el comando GRAPHICS, al dispositivo "VIDEO:".

que está preparado en modalidad de video 1, color de video 1, con un tamaño de página 2Ø, 4Ø. El canal 1Ø1 permanece abierto hasta que se cierre explícitamente, por ejemplo, con un comando TEXT.

1Ø2 - la página de 'texto' standard. Se abre y repone automáticamente con un tamaño de página de 24, 4Ø.

1Ø3 - Se usa para salida de sonido standar. El canal se conecta al dispositivo "SOUND".

El canal 1Ø3 es automáticamente abierto y repuesto y permanece abierto hasta que se cierre explícitamente.

1Ø4 - Se usa para las operaciones en 'soporte legible' (en papel). Este canal se conecta en la reposición al dispositivo "PRINTER:". Es el canal por omisión que se supone para COPY TO y REDIRECT TO.

El canal 1Ø4 se abre automáticamente en la reposición y permanece abierto hasta que sea explícitamente cerrado.

1Ø5 - Se utiliza para operaciones de entrada y salida (conectado en la reposición al dispositivo "KEYBOARD"). Permanece abierto hasta que se cierre explícitamente.

1Ø6 - Se utiliza para operaciones de entrada y salida basadas en fichero. Siempre que se requiere, el canal va conectado a "DISK-1:" si hay una unidad de disco acoplada; si no la hay, se conecta a "TAPE".

Entre las operaciones de fichero standard se incluye LOAD, MERGE y VERIFY.

El canal 1Ø6 sólo se abre cuando es necesario y se cierra después de terminada cada operación, salvo que se haya pasado explícitamente un comando OPEN.

1Ø7 - Se usa para operaciones de red.

El canal 1Ø7 se abre solo automáticamente por un comando que asume este canal como omisión, y se cierra después de terminada la operación.

Los canales 1ØØ-254 permanecen abiertos salvo que sean cerrados específicamente, pero los canales 1-99 se cierran siempre cuando se escribe RUN, o si ocurre cualquier otra operación que borre todas las variables. Si el BASIC descubre un

canal por omisión cerrado, entonces cerrará todos los canales (Ø-254) e intentará reabrir sus canales por omisión. Si no puede hacerlo, el BASIC supone que ha ocurrido un error irreparable y hace que parpadee intermitentemente el borde de la pantalla hasta que el ordenador es repuesto.

Los nombres de dispositivos pasados a través del sistema operativo terminan con una coma, a fin de que puedan ser reconocidos. Cuando más de un dispositivo es conocido por el mismo nombre, hay un número añadido al nombre, por ejemplo, "DISK2:" o "DISK-2:":

Los nombres válidos son:

"DISK-n:" Unidades de discos

"EDITOR:" Editor de pantalla. Utiliza a su vez los dispositivos "VIDEO:" y "KEYBORARD:"

"KEYBOARD:" Teclado transparente. Incluye las palancas de mando exteriores.

"NET-n:" Red local incorporada. El número'n' es la dirección de la red (en la gama de 1 a 32) de la máquina con la que se establece la comunicación. Si 'n' es Ø, define un canal 'general', que se utiliza para difundir a todas las máquinas y para recibir datos sin especificar la máquina de origen.

"PRINTER:" Puerto para impresora 'estilo Centronic'

"SERIAL:" E/S RS423 Serie.

"SOUND:" Generador de sonido.

"TAPE-n:" Unidades de cinta

"VIDEO:" Página de video

Cuando hay otros dispositivos acoplados al ordenador, pueden definir nombres adicionales dentro del sistema operativo.

En la mayoría de los casos, sólo se necesita un

Operación OPEN. Cuando se utiliza entrada/salida con base en fichero, se debe dar un nombre de fichero.

La especificación completa de un nombre de fichero es "device-n: name" (dispositivo-nombre)
-"device" es opcional; si se omite, se utilizará el dispositivo por omisión de almacenamiento de masa del sistema: para un sistema no ampliado, es "TAPE:".

"n" es el número de dispositivo y su valor por omisión es 1 si se omite; por ejemplo, "DISK:" pasaría a "DISK-1:"

"name" es la descripción del fichero dentro del dispositivo. Sigue las mismas reglas de formato que un identificador BASIC, excepto que sólo son significativos los 28 primeros caracteres.

Si no se incluye ninguna coma en el nombre de fivhero, se supone que se ha omitido el nombre de dispositivo. Así, por ejemplo, "SOUND" es un fichero en "TAPE", pero "SOUND" es el dispositivo generador de sonido.

Por ejemplo, "DISK1:DATAFILE""DISK-1:DATAFILE" "1:DATAFILE" "DATAFILE" se referirán todos al mismo fichero (suponiendo que hay discos montados).

La parte "name" de un nombre de fichero la ignoran todos los dispositivos actualmente definidos, excepto "TAPE:"y "DISK:". Así, por ejemplo, "PRINTER: PRETTY-LISTING" equivale a "PRINTER:".

Hay algunos comandos que permiten especificar tantos canales como nombres de fichero dentro de una sola sentencia; por ejemplo LOAD y SAVE.

La especificación completa en estos casos toma la forma
£chan:filename

Si falta £chan, se utiliza entonces un canal por omisión.

OPTION — OPTION ANGLE DEGREES/RADIANS
(Opción ángulos en grados/radianes)

Selecciona la unidad base de las operaciones siguientes que usan ángulos. La presunción por omisión es radianes.

OUT — OUT n,a
Escribe el byte 'a' en la puerta 'n' de E/S
PERMITE HASTA ~~65791~~ 255

PING — Produce el sonido 'ping'

PLOT
(trazar-dibujar)

PLOT # canal:lista-punto
PLOT # canal:ANGLE expr
PLOT # canal: FORWARD/BACK expr
PLOT # canal: LEFT/RIGHT expr
PLOT # canal: ELLIPSE expr, expr
PLOT # canal: PAINT

PLOT seguido por una lista de puntos dibuja puntos y/o líneas. Cuando un comando PLOT termina con punto y coma, el haz quedará encendido después de ejecutado el comando; en caso contrario se apagará.Así:

PLOT x,y;
dejará el haz encendido.

Las dos últimas sentencias trazan ambas un punto en (x,y). Si el par de coordenadas va seguido por una coma, el haz pasa a la posición especificada sin dibujar ningún punto allí (y queda apagado).

PLOT x1,y1;x2,y2;...
dibujará líneas con el haz encendido entre los puntos especificados y lo dejará encendido si el comando termina com punto y coma. Si el haz estaba encendido antes de ejecutar el comando, se dibujará también una línea desde la posición actual del haz hasta el punto (x1,y1).

El dibujo se realiza en el color actual de la tinta y de acuerdo con el estilo y modalidad de línea actuales (ver la Sección Opciones de Video).

Las coordenadas que se utilizan en la sentencia PLOT siguen las convenciones del dibujo de gráficas. La esquina inferior izquierda de la página de video es (Ø,Ø). En la especificación de coordenadas (x,y), x es la posición horizontal contando desde la izquierda e y es la posición vertical contando desde abajo.

ELLIPSE dibuja una elipse con su centro en la posición actual del haz. Los dos parámetros que siguen dan las distancias horizontales y verticales desde el centro hasta la circunferencia en las posiciones de la pantalla de gráficos. La

elipse debe dibujarse con el haz apagado si no debe aparecer ningún punto en el centro.

PLOT 3ØØ, 35Ø, ELLIPSE 2ØØ, 3ØØ,
evitará que se dibuje el punto.

PAINT rellena un área cerrada (que contiene la posición actual del haz) con el color actual de la tinta. El área pintada está limitada por una línea contínua de color diferente al color original de la posición del haz.

Si el haz está en una posición en la que se ha dibujado un punto del color de la tinta actual, no tendrá efecto el comando PAINT, ya que detectará una condición de límite inmediatamente. Como ocurre con ELLIPSE conviene tomar precauciones para evitar que se dibuje un punto.

PLOT 4ØØ,3ØØ, PAINT

Un comando PLOT pasará por omisión al canal 1Ø1

POKE

POKE dirección, valor
Establece el valor de la posición especificada en la memoria del procesador Z80.

PRINT

PRINT #canal, AT expresión-fila, expresión-columna: lista-salida.
PRINT #canal, USING número-línea:lista-salida
PRINT #canal, USING serie:lista-salida.

Un elemento de la lista de salida puede ser una expresión o la palabra TAB seguida por un número de columna entre paréntesis. Los distintos elementos pueden ir separados por comas o punto y comas. Un punto y coma genera una serie nula; una coma inserta espacios hasta el comienzo de la siguiente zona de impresión. TAB inserta espacios hasta la columna especificada.

Una lista de salida que termina con coma o punto y coma no genera una secuencia de fin de línea. Se puede ejecutar en modalidad inmediata.

La opción AT coloca el cursor en la fila y columna especificadas antes de imprimir la lista. El número de canal opcional redirige la salida (el canal por omisión es la página de texto standard).

Las coordenadas de fila y columna de la especificación AT siguen las convenciones para colocación del texto. La esquina superior izquierda de la página de video tiene coordenadas de texto (1,1). La 15ª columna de la segunda fila

tiene coordenadas de texto (2,15)

```
PRINT "VALUE ="=;A
PRINT AT x,y:"o";
```

La opción USING controla el formato de la salida.
El número-línea debe ser el número de una sentencia IMAGE. Ver en IMAGE los detalles de la especificación de formato.

PROGRAM — PROGRAM nombre (lista-variables)

Define el nombre del programa actual para utilizar en la sentencia CHAIN y como nombre por omisión para SAVE.

```
PROGRAM "My_Program"(A,B$)
```

La lista de variables (si se incluye) permite que los parámetros especificados se pasen por su valor desde otro programa. Ver CHAIN y EDIT.

RANDOMIZE
(Aleatorizar) Normalmente, cada ejecución de un programa empieza siempre por la misma secuencia de números aleatorios. RANDOMIZE cambia los números aleatorios a una nueva secuencia.

READ
(leer)

```
READ lista-variables
READ IF MISSING número-línea:lista-variables
READ IF MISSING EXIT DO: lista-variables
```

Lee los datos de las sentencias DATA; la acción MISSING se ejecuta en un intento por leer más allá del fin de los datos.

```
READ A,B$(i)
```

REDIRECT
(Redirigir) REDIRECT FROM #canal TO #can1

Lee la entrada del primer canal y la dirige al segundo, hasta que se alcanza el fin de un fichero, se pulsa la tecla 'stop' o hay un error en uno de los canales. La redirección puede suspenderse también utilizando el número de canal inválido 255 como canal TO (a1) en la sentencia REDIRECT posterior.

REM — REM línea-comentario.

REM debe estar al comienzo de la línea e ir seguido por al menos un espacio. Para conseguir una mayor flexibilidad, se recomienda '!'.

RENUMBER
(renumerar) RENUMBER descripción-línea TO descripción-línea AT expr STEP expr
RENUMBER nombre-bloque AT expr STEP expr.

Renumera la totalidad o una parte del programa. Sólo se ejecuta en modalidad inmediata.

```
RENUMBER FIRST TO 1ØØ
RENUMBER 1Ø TO 1ØØ AT 3ØØ STEP 1Ø
RENUMBER STEP 1ØØ
RENUMBER MY_FUNCTION AT 5ØØØ
```

STEP y AT pueden estar en cualquier orden u omitirse. Si no se especifica STEP, el valor por omisión es 1Ø. Si se omite AT, se usa el primer número de línea del segmento que hay que renumerar. Si no se da ninguna gama de números de línea, todo el programa se renumera y el valor por omisión de AT es 1ØØ.

Se puede dar el nombre de un bloque DEF o HANDLER en lugar de una gama de números de línea. Para la sintaxis de las descripciones de línea, comparar DELETE.

Todas las referencias en el programa a líneas renumeradas se cambian.

RENUMBER no puede cambiar el orden de las líneas de un programa. Por lo tanto, si las líneas renumeradas se superponen o rodean a otras líneas no renumeradas, o si se pusieran en un nuevo lugar en la secuencia, o si crearan un número de línea excesivamente alto, entonces no se ejecutaría el mandato RENUMBER, y quedaría sin cambios el texto del programa.

RESTORE
(restaurar) RESTORE
RESTORE número-línea

Repone el comienzo de DATA (para sentencias READ) al comienzo del programa o el número de líneas dado.

RETRY
(reintentar) Se utiliza como salida de un manipulador de excepciones, devolviendo el control a la línea o sentencia que ha provocado la excepción. Comparar CONTINUE.

Si se utiliza un manipulador de excepciones para bloquear la tecla 'stop' o cualquier interrupción de software, se debe utilizar RETRY para continuar el programa.

RETURN — Retorna de una subrutina llamada por GOSUB.

RUN
(ejecutar)

```
RUN (lista de parámetros)
RUN número-línea
RUN #canal:nombre-fichero (lista-parámetros)
RUN nombre-dispositivo (lista-parámetros)
```

RUN por su parte ejecuta el programa actual desde la primera línea. Si se dá un número de línea, la ejecución comienza a partir del mismo. Si se da un nombre de fichero (con canal opcional) el programa se carga y se ejecuta después.

Los parámetros se pueden pasar a los programas con RUN, pero éstos deben corresponder a los parámetros declarados para el programa. Ver PROGRAM. Borra las variables.

SAVE
(Preservar)

```
SAVE #canal:nombrefichero
SAVE nombre-dispositivo
SAVE ALL #canal:nombrefichero
```

Preserva el programa actual. Por omisión se preserva a través del canal 1Ø6. Si no se da nombre de fichero entonces se utilizará el nombre PROGRAM, si es que existe.

SAVE ALL preservará todos los programas que se encuentran actualmente en la memoria.

Los programas se preservan en formato codificado. Para preservar un programa en formato de caracteres (ASCII), utilizar LIST

#canal:nombrefichero ,estos programas pueden cargarse más tarde si es preciso. Sólo trabaja en modalidad inmediata. Ver LOAD.

SELECT
(seleccionar)

```
seleccionar-línea
línea-caso
      cualquier número de sentencias o bloques
opción línea-caso
      cualquier número de sentencias o bloques
fin-selección-línea

Seleccionar-línea:
      SELECT CASE Expresión
línea-caso:
CASE expresión
CASE expresión TO expresión
```

CASE IS expresión relacional
CASE ELSE
fin-selección-línea:
END SELECT

El bloque SELECT es un grupo de sentencias para probar la variable o expresión con una serie de condiciones alternativas.

La palabra CASE en la línea SELECT es opcional a menos que la expresión empiece con un identificador CASE.

Por ejemplo SELECT CASE CASE +23

Puede haber cualquier número de líneas CASE. Los casos se prueban en orden de números de línea. No hace falta tener líneas de casos adicionales después de un CASE ELSE, ya que no pueden normalmente alcanzarse. Se pueden combinar varios casos en una línea separándolos con comas.

Por ejemplo CASE 1,2,3 TO 6,99

```
SELECT CASE N
CASE 1
  PRINT "first case"              "primer caso"
CASE 2 TO 9,11,21
  PRINT "some more cases"         "algunos casos más"
CASE IS< =A+20
  PRINT "even more cases"         "aún más casos"
CASE ELSE
  PRINT "rest of cases"           "el resto  de los casos"
END SELECT
```

La línea CASE ELSE sólo puede utilizarse una vez, y debe seguir a todas las demás líneas CASE. Las otras líneas CASE pueden utilizarse en cualquier orden, según sea necesario, formando un bloque las líneas que hay entre dos líneas CASE. No se puede transferir el control del exterior al interior de un bloque SELECT.

También se dispone de Selecciones de serie.

SET Introduce los valores actuales de las opciones de máquina. Ver 'Opciones de máquina', 'Opciones de Video' y 'Opciones de Sonido'. Comparar con ASK y TOGGLE.

SOUND
(Sonido)

Proporciona el control general de un sonido. Los parámetros pueden listarse en cualquier orden.

El número específico por PITCH puede ser cualquiera de Ø a 127, aunque normalmente sólo se obtienen buenos resultados en la gama de Ø a 83. Dentro de esta gama, un aumento de 1 elevará la altura en un semitono; el valor 37 del tono (el valor por omisión) equivale al Do medio.

DURATION da la duración del sonido (asignado a las fases sin liberación de la envolvente), en 'tics' (un tic tiene 1/5Ø de segundo). El valor por omisión es 5Ø tics.

Los parámetros LEFT y RIGHT especifican el volumen general del sonido para los dos canales de salida estéreo. Los valores varían de Ø (ningún sonido) a 255 (volúmen máximo de la máquina: el valor por omisión). Si no se utiliza equipo estéreo, el volúmen se determinará por la suma de los valores dados para los canales de la derecha y de la izquierda.

SOURCE especifica el generador de tono utilizado; los valores son Ø-3 (valor por omisión:Ø). El generador de tono 3 es el 'generador de ruido' (que ignora los valores del tono).

El parámetro STYLE se encuentra en la gama de Ø a 255 (omisión: Ø); para sus efectos, ver las 'Opciones de Sonido'.

ENVELOPE especifica el número de la envolvente que se ha de aplicar al sonido. Ver la sentencia ENVELOPE. 255 (la omisión) es una envolvente incorporada.

SYNC permite que el comienzo del sonido se sincronice exactamente con 1,2 ó 3 sonidos distintos procedentes de 'fuentes' diferentes. Si, por ejemplo, tres sonidos deben empezar juntos, se puede dar a cada uno la instrucción SYNC 2, que hace que se sincronize con los otros dos. (El valor por omisión es Ø).

INTERRUPT, si se incluye, hace que el nuevo sonido sustituya a cualquier sonido (de la misma fuente) que pueda estar actualmente ejecutándose.

SPOKE SPOKE segmento, dirección, valor

Como POKE, pero escribe el valor en la dirección del sistema dentro del segmento especificado.

START
(arranque) Si no hay ningún programa actualmente cargado, este comando carga

y ejecuta el primer fichero del canal 1Ø6. Si hay algún programa en la memoria, entonces START actúa como RUN en el programa actual.

STOP — Suspende la ejecución (imprime mensajes STOP).

Obsérvese que después de una instrucción STOP se permite el CONTINUE.

STRING — STRING variable/lista-matriz

Declara las variables o matrices de serie con la longitud máxima. La longitud por omisión es 132. Añadiendo *n después de la palabra STRING o de la declaración de variables se establece la longitud en n. El límite inferior por omisión de una matriz es Ø.

```
STRING*8 LAST_NAME$*20,FIRST_NAME$,
```

En este ejemplo, LAST_NAMES$ tendrá una longitud máxima de 2Ø. FIRST_NAMES y MIDDLE_NAMES tienen una longitud de hasta 8 caracteres.

```
STRING NAME$
```

En este caso, NAME$ tiene una longitud máxima de 132.

```
STRING NAME$ (4 TO 99)*1Ø
```

Esta matriz tiene 96 elementos, cada uno de ellos con 1Ø caracteres.

Nota: no se puede utilizar una sentencia DIM para definir la longitud de una variable de serie.

TEXT (texto) — Abre una página de texto que cubre toda la pantalla excepto el área de la línea de estado. Cierra la página de gráficos standard si se encontraba abierta.

TEXT — TEXT 4Ø TEXT 8Ø

4Ø u 8Ø especifican el número de columnas en pantalla. Si no se especifica, se utilizará el valor anterior.

Ver DISPLAY TEXT.

THEN — Ver IF

TIME
(hora) TIME serie-hora

Especifica la hora actual que mantiene el ordenador. Se incrementa automáticamente cada segundo.

El tiempo se especifica en el formado HH:MM:SS (horas, minu_tos, segundos)

TIME "15:35:00"

Se puede utilizar en modalidad inmediata. Ver la función TIME$; ver también el comando WAIT DELAY y la opción de máquina: TIMER.

TOGGLE Actúa como opciones de la máquina que tiene sólo dos valores posibles (por ejemplo, 'conectado' y 'desconectado'), pasando del valor actual a la alternativa. Ver las secciones sobre 'Opciones de Máquina', 'Opciones de Video' y 'Opciones de Sonido'; comparar con SET y ASK.

TRACE ON TO £chan
TRACE OFF
TRACE
(rastrear) Después de TRACE ON, se informa del número de la línea que se está ejecutando actualmente. La salida se dirige al canal Ø salvo que se dirija a un número concreto de canal.

TYPE TYPE

Instrucción especial para salir del BASIC e introducir el procesador de palabras incorporado. Esta acción destruirá todos los programas y variables BASIC, por lo que el usuario recibe un mensaje de solicitud para que pulse 'enter' confirmando la instrucción.

VERIFY
(verificar) VERIFY £canal:nombrefichero

Comprueba si se ha preservado correctamente un programa; compara el fichero del programa actual con el fichero especificado y presenta un mensaje de error si ambos ficheros no son idénticos. Por omisión se utiliza el canal 1Ø6. Sólo se ejecuta en modalidad inmediata.

WAIT DELAY
(período de
espera) WAIT DELAY expr

Hace que la ejecución del programa espera durante un período especificado en segundos.

```
WAIT DELAY 6Ø
```

suspenderá la ejecución del programa durante un minuto. El retardo máximo es de 32.777 segundos. Ver en la opción de máquina TIMER la operación automática de espera mientras se continúa la ejecución del programa.

La palabra DELAY es opcional.

WHEN
(cuando)

```
WHEN EXCEPTION USE nombre-manipulador
Sentencias
END WHEN
```

Especifica el manipulador de excepciones que se utilizará cuando ocurre una excepción causada por la ejecución del programa dentro del bloque WHEN.

Las sentencias del programa pueden incluir bloques adicionales encajados WHEN.

Ver HANDLER.

 OPCIONES DE MAQUINA (GENERALIDADES)

Algunas variables del sistema y funciones de máquina que se pueden controlar directamente desde el BASIC, se denominan opciones de máquina. Para asignar un valor a una opción, se utiliza el comando SET. Cuando se indica, las opciones enumeradas a continuación pueden ser manejadas también junto con ASK o TOGGLE.

DEFAULT CHANNEL (CANAL POR OMISION)

SET DEFAULT CHANNEL expr

Especifica el canal por omisión del sistema. Este valor del canal puede ser usado por los programas de servicio que desean comunicar con el usuario pero desconocen la finalidad de los canales actualmente utilizados.

En particular, este canal lo utilizará el programa que responde al comando HELP enviado a través del sistema operativo. Antes de haber sido SET, este canal tendrá el número Ø.

EDITOR BUFFER (MEMORIA INTERMEDIA DEL EDITOR)

SET EDITOR BUFFER expr

Define el tamaño de la memoria intermedia o buffer del editor, en trozos de 256 bytes, para su utilización con los canales de editor posteriormente abiertos. Se puede usar con ASK.

EDITOR KEY (TECLA EDITOR)

SET EDITOR VIDEO número-canal

Permite que se utilice el canal especificado como entrada a través del teclado del editor, para utilizar con los canales de editor posteriormente. Se puede usar con ASK

EDITOR VIDEO (VIDEO DE EDITOR)

SET EDITOR VIDEO número-canal

Permite que se utilice el canal especificado como página de texto del editor, para usar con los canales de editor abiertos posteriormente. Se puede usar con ASK.

FAST SAVE (PRESERVACION RAPIDA)

SET FAST SAVE ON/OFF

Introduce la velocidad rápida de preservación para operaciones con cinta. Esta velocidad es de aprox. 2400 baudios, y es la velocidad por omisión. Si está desconectada la preservación rápida la velocidad se reduce a la mitad.

La carga desde cinta hace frente automáticamente a las variaciones, en la velocidad de preservación:

Se puede usar con TOGGLE.

KEY (TECLA) SET#canal: FKEY número de tecla-serie

Ajusta la tecla de función para que produzca la serie especificada cada vez que se pulsa (una serie nula provocará una excepción).

El canal por omisión es 1Ø5.

Las teclas de función están numeradas de 1 a 16. Los números 1 a 8 son las teclas de función sin desplazar; los números 9-16 son los equivalentes desplazados con series por omisión por el sistema y la redefinición de las teclas eliminará los ajustes por omisión. Para devolver todas las teclas de función a sus valores por omisión, utilizar CLEAR FKEYS.

Para crear un'enter' automático, utilizar &CHR$ (13)

INTERRUPT (INTERRUPCION)

ASK INTERRUPT CODE

Solicita el código de interrupción del software para la última interrupción.

SET INTERRUPT KEY ON/OFF

Cuando está conectado, provoca la interrupción del software con la pulsación de cualquier tecla. Se puede utilizar con TOGGLE.

SET INTERRUPT NET ON/OFF

Conecta y desconecta la interrupción del software causada por la recepción de datos de la red.

SET INTERRUPT STOP ON/OFF

Conecta y desconecta la interrupción de software con la tecla 'stop'. Se puede usar con TOGGLE.

KEY CLICK (CLIC DE LA TECLA)

SET KEY CLICK ON/OFF

Determina si se escucha un clic con cada pulsación de la tecla. Puede utilizarse con TOGGLE.

KEY DELAY (RETARDO DE LA TECLA)

SET KEY DELAY expr

Ajusta el retardo inicial del teclado antes de que comience la autorrepetición, en unidades de 1/50 segundo. Puede usarse con ASK.

KEY RATE (VELOCIDAD DE TECLA)

SET KEY RATE expr

Especifica la velocidad de autorepetición del teclado en unidades de

1/50 segundos. Puede usarse con ASK.

NET CHANNEL (CANAL DE RED)

ASK NET CHANNEL var

Devuelve el número de canal en el que deben leerse los datos de una memoria intermedia de la red que se encuentra a la espera. Cuando se lee en este canal el primer byte, NET CHANNEL continúa señalando el siguiente canal que necesita servicio. Si no hay ningún canal más que tenga datos, se devuelve el valor 255.

NET MACHINE (MAQUINA DE RED)

ASK NET MACHINE var

Se actualiza al mismo tiempo que NET CHANNEL, y devuelve el número de red de la máquina situada a distancia. Esto es particularmente importante si los datos se reciben en el canal 'general' (NET-Ø)

NET NUMBER (NUMERO DE RED)

SET NET NUMBER expr

Determina el número de dirección de red del ordenador. Debe encontrarse en la gama de 1 a 32. Comienza como Ø, que no es válido como dirección de red.

Antes de usar la red, debe especificarse un número de red, que no debe tener el mismo valor que utilice cualquier otro ordenador de la red.

REMI

SET REM1 ON/OFF

Controla el interruptor de control a distancia 1(Es controlado también por las operaciones de cintas).

REM2

SET REM2 ON/OFF

Como antes, pero para el interruptor nùmero 2.

SERIAL BAUD (BAUDIOS SERIALES)

SET SERIAL BAUD expr

El parámetro (en la gama de Ø a 15) determina la velocidad en baudios de la puerta RS232 y la red, según el código que se da a continuación.

Puede usarse con ASK.

Ø = > 5Ø	baud	6 = > 3ØØ	baud
1 = > 75	"	7 = > 6ØØ	"
2 = > 11Ø	"	8 = > 12ØØ	"
3 = > 134.5	"	9 = > 18ØØ	"
4 = > 15Ø	"	1Ø = > 24ØØ	"
5 = > 2ØØ	"		
11 = > 36ØØ	baud		
12 = > 48ØØ	"		
13 = > 72ØØ	"		
14 = > 96ØØ	"		
15 = > 96ØØ	"		

La velocidad de baudios por omisión es de 96ØØ (valor 15).

SERIAL FORMAT
(FORMATO
SERIE)

SET SERIAL FORMAT expr

Define el formato de palabra del accionador de dispositivo serial. El formato es controlado por los bits binarios del número como sigue:

BIT	VALOR	EFECTO
Ø	Ø	8 bits
	1	7 bits
1	Ø	sin paridad
2	Ø	paridad par
	1	paridad impar se ignora si el bit 1 es Ø
3	Ø	dos bits de parada
	1	un bit de parada

Los bits de 4 en adelante deben ser Ø

El formato por omisión es de 8 bits, sin paridad, 2 bits de parada. El uso de la red reinicializará siempre este valor por omisión.

STATUS
(ESTADO)

SET SERIAL FORMAT expr

Conecta o desconecta la 'línea de estado' (en la parte superior de la pantalla puede usarse con TOGGLE).

TAPE LEVEL
(NIVEL DE
CINTA)

SET TAPE LEVEL expr

Controla el nivel de volumen utilizado al preservar cualquier material en cinta. Los niveles aceptables de cinta se encuentran en la gama de 1 a 6, con duplicación del volumen para cada nivel. El nivel 1 equivale aprox. 40 mV pico a pico. El nivel por omisión es 2.

TAPE SOUND (SONIDO DE CINTA)

SET TAPE SOUND ON/OFF

Controla la transmisión de sonido de la entrada de la cinta a la salida de sonido. Permite la producción directa de música o palabras de la cinta al altavoz interior o la salida en alta fidelidad. Puede usarse con TOGGLE.

TIMER (TEMPORIZADOR)

SET TIMER expr

Pone en marcha un temporizador (o cronómetro) que provoca una interrupción del software cuando cuenta retrocediendo hasta llegar a Ø. El valor se especifica en segundos, con un máximo de 255. El ajuste de valor a Ø detiene el temporizador sin provocar interrupción.

El temporizador se detiene siempre cuando llega a Ø, y debe rearrancarse explícitamente.

La excepción de interrupción de software (EXTYPE 9235) deberá ser manejada siempre por un manipulador de excepciones, ya que de lo contrario el programa suspende la operación. Después de la excepción, ASK INTERRUPT CODE A asignará a A el valor 64 si la interrupción ha venido del temporizador.

VARIABLE

SET número variable, expr
ASK número-variable var
TOGGLE número-variable

Introduce, solicita o cambia de posición la variable especificada del sistema operativo. Para más detalles ver el Manual Técnico del Enterprise.

 OPCIONES DE VIDEO

Trabajan con el dispositivo de video incorporado, que puede contener muchas páginas de video, cada una de ellas con parámetros diferentes. Los comandos que trabajan con páginas individuales pueden recibir una especificación de canal, pero si esto se deja, algunos de ellos señalan por omisión la página de texto standard (#1Ø2), otros la página de gráficos standard (#1Ø1), como se detalla a continuación.

Obsérvese que se acepta siempre COLOR en lugar de COLOUR.

ATTRIBUTES
(ATRIBUTOS) SET ATTRIBUTES expr

Introduce un señalizador especial para controlar las operaciones en modalidad de video de atributos (número 15). Los valores de este señalizador tienen los siguientes significados:

1 - dibuja en Øs (color del papel)
2 - dibuja sin afectar a los datos del mapa de bits (pixel)
4 - dibuja sin afectar a los atributos de la tinta
8 - dibuja sin afectar a los atributos del papel
16 - dibuja en Øs
32 - dibuja sin afectar el mapa de bits.
64 - dibuja sin afectar a los atributos de tinta
128- dibuja sin afectar a los atributos del papel.

Para conseguir efectos combinados, los números se deben sumar entre sí. El valor por omisión es Ø.

BEAM (HAZ) SET #chan: BEAM ON/OFF

A la posición actual de dibujo de gráfico se le denomina la posición del'haz'. Cuando se mueve el haz, puede o no dejar una línea detras, según que esté conectado (on) o desconectado (off). El valor por omisión del número de canal es = 1Ø1.

BIAS
(POLARIZACION)
SET # canal:BIAS código-color

Establece qué grupo de colores figurará como números 8-15 dentro de la paleta. El número especificado en el comando es el número de código standard de cualquier color deseado; hay 32 valores efectivos. La polarización puede también especificarse con el uso de la función RGB.

SET BIAS RGB (Ø,.6,.4)

El número de canal por omisión es £1Ø1. Sin embargo, la polarización se aplica a cada paleta utilizada en pantalla.

BORDER
(RECUADRO) SET £canal:BORDER código-color

Cambia el recuadro al color correspondiente al número de código standar especificado. El número de canal por omisión es £1Ø1.

CHARACTER
(CARACTER) SET £ canal:CHARACTER n,a,b,c,d,e,f,g,h,i

Define el dibujo del carácter con el código ASCII 'n'. Cada uno de los parámetros a-i define una línea del dibujo, empezando por arriba.

Para crear caracteres se puede usar la función BIN a fin de especificar cada pixel de una línea como Ø ó 1.

Aunque se especifica un número de canal, el comando afectará a todas las páginas de video. El número de canal por omisión es £1Ø2. Para devolver todos los caracteres a sus ajustes por omisión utilizar CLEAR FONT.

COLOUR
(COLOR) SET £ canal:COLOUR número-paleta, código-color

Ajusta el valor de un color en la paleta (ver abajo PALETTE). Los números de paleta se encuentra en la gama de Ø a 7. El código de color pertenece a la gama standard de Ø a 255 (o se especifica con la función RGB).

CURSOR SET £ canal:CURSOR CHARACTER código
SET £ canal:CURSOR COLOUR número-paleta

Especificar el código ASCII del carácter y/o el número de paleta del color que se utilizará para el cursor. El número del canal por omisión es £1Ø2.

INK
(TINTA) SET £ canal:INK número-color

Ajusta el color del dibujo actual. El número del color es un número de paleta, excepto en la modalidad de color 3 (256 colores), en el que es un número de código de color standard. El número de canal por omisión es £1Ø1:

LINE MODE (MODALIDAD DE LINEA)

SET £ canal:LINE MODE parámetro

Determina la interacción entre los colores del display existente y las nuevas líneas que se dibujan. En la modalidad Ø (la modalidad por omisión), una nueva línea se escribe encima de cualquier cosa que se haya dibujado antes. La modalidad 1-3, el color usado para cualquier parte de la nueva línea, se determinará combinando los números de paleta de los colores de tinta nuevo y antiguo, de la manera siguiente:

Modalidad 1- 'O'
Modalidad 2- 'Y'
Modalidad 3- 'O exclusiva'

El número de canal por omisión es £1Ø1.

LINE STYLE (ESTILO DE LINEA)

SET£canal:LINE STYLE parámetro

El estilo de línea actual puede ajustarse a cualquier valor en la gama de 1-14, permitiendo que se dibujen distintos tipos de línea interrumpida. El número de canal por omisión es 1Ø1.

PALETTE (PALETA)

SET £canal:PALETTE a,b,c,d,e,f,g,h

Ajusta los valores de los 8 primeros colores de la paleta, que los utilizan, entonces, las opciones video tales como SET PAPER y SET INK. El número de canal por omisión es £1Ø1.

En la modalidad de color 1 sólo pueden usarse los cuatro primeros colores, y una página de gráficos en modalidad de color Ø puede usar únicamente los dos primeros. Si sólo se especifican los 2 ó 4 colores primeros, el resto señala por omisión el color Ø. Los colores que deben colocarse en la paleta se especifican por el código de color standard en la gama de Ø a 255, o por la función RGB (ver 'Variables y Funciones incorporadas'). Los colores de 'teletexto primario' pueden especificarse con el nombre (por ej. MAGENTA).

La paleta contiene en total 16 colores, aunque sólo pueden elegirse de manera totalmente libre los 8 primeros. Ver en la opción BIAS los detalles de los 8 colores restantes.

PAPER (PAPEL)

SET £canal:PAPER: número-color

Selecciona el color que servirá como fondo para imprimir o dibujar. En la modalidad de color 3, el color del papel se define por un número de código standard en otras modalidades, por un número de paleta. El número de canal

por omisión es £1Ø1.

Para una página video de gráficos (modalidades 1 y 5); ver opción VIDEO MODE; el comando PAPER sólo tendrá efecto cuando la página esté borrada: cuando se seleccione un nuevo fondo para la visualización de los gráficos.

Para la página de texto de 80 columnas (modalidad de video 2), los colores válidos de papel son los números de paleta Ø, 2, 4 y 6. Van apareados con los colores de tinta, 1, 3, 5 y 7, respectivamente; un carácter impreso en un color correspondiente de papel para su propio fondo individual. Un par de colores de tinta y papel se selecciona escribiendo SET PAPER o SET INK, seguido por uno de los dos números de paleta correspondientes.

La página de texto de 40 columnas (modalidad de video Ø) es similar, salvo que sólo hay dos pares de colores disponibles.

SCROLL
(REVISION) SET £canal:SCROLL ON/OFF

Conecta o desconecta la revisión automática. El número de canal por omisión es £1Ø2.

SET £canal:SCROLL UP/DOWN n,m

Revisa la pantalla subiéndola o bajándola desde la línea (n-32) a (m-32). El número de canal por omisión es £1Ø2 .

VIDEO COLOUR
(COLOR DE
VIDEO) SET VIDEO COLOUR expr

Ajusta la modalidad de color para las páginas de video que deben abrirse posteriormente (se ignora el número de canal).

Cuando se define una página de video de texto se debe seleccionar siempre la modalidad de color Ø. Para las páginas de gráficos de alta resolución, las modalidades de color tienen el significado siguiente:

modalidad Ø - 2 colores; resolución horizontal 64Ø
modalidad 1 - 4 colores; resolución horizontal 32Ø
modalidad 2 - 4 colores; resolución horizontal 16Ø
modalidad 3 - 256 colores; resolución horizontal 8Ø

En una página de gráficos LORES (utilizando la mitad de la memoria que HIRES), la cantidad de color para cada modalidad es la anterior, pero la resolución horizontal se reduce a la mitad.

VIDEO MODE
(MODALIDAD
DE VIDEO) SET VIDEO MODE expr

Ajusta las modalidades de video para las páginas que deben abrirse posteriormente. (Se ignora el número de canal).

Los valores de parámetros son los siguientes:

modalidad Ø - página de texto de 4Ø columnas (2 pares de colores)
modalidad 1 - página de gráficos de alta resolución
modalidad 2 - página de texto de 8Ø columnas (4 pares de colores)
modalidad 5 - página de gráficos de baja resolución
modalidad 15 - pantalla de gráficos de 'atributos'

VIDEO X SET VIDEO **X expr**

Define el tamaño horizontal de las páginas de video que deben abrirse posteriormente (se ignora el número de canal). El tamaño se especifica como un número de posiciones de caracteres en la gama 2 a 42, usando las convenciones de coordenadas de las páginas de texto.

VIDEO Y SET VIDEO Y expr

Como antes, pero define el tamaño vertical de la página como una serie de líneas de caracteres en la gama de 1 a 255.

OPCIONES DE SONIDO

Trabajan con el generador incorporado de sonidos.

SOUND BUFFER
(BUFFER O
MEMORIA
INTERMEDIA
DE SONIDO) SET SOUND BUFFER expr

Ajusta el tamaño del área de almacenamiento de la envolvente de sonido para la apertura posterior al dispositivo "SOUND". La expresión es el número de fases. Los valores posibles son 1-255; el valor por omisión es 2Ø. Se puede usar con ASK.

SOUND STYLE
(TIPO DE
SONIDO)

Los valores del parámetro STYLE de una sentencia SOUND 4 ver la sección sobre 'comandos y sentencias' tiene los efectos siguientes.

En el canal de tono Ø:

16- distorsión baja
32- distorsión media
48- distorsión alta
64- uso del filtro paso alto. El canal de tono 1 es el reloj
128- modulación de anillo con canal 2

En el canal de tono 1:

Como en el canal 1, pero el filtro paso alto utiliza el canal de tono 2; el modulador de anillo utiliza el canal de ruido (canal 3)

En el canal de tono 2:

Como en el 1, pero el filtro paso alto utiliza el canal de ruido (canal 3); el modulador de anillo usa el canal de tono Ø.

En el canal 3 (canal de ruido):

1,2,3- utilizan el canal de tono Ø, 1 ó 2 como frecuencia de reloj, en vez de la frecuencia de ruido de contadores polinómicos de 15, 11 ó 9 bits, en lugar del contador standard de 17 bits.

16- sustituye un contador polinómico de 7 bits en lugar del de 17.

32- utiliza un filtro paso bajo en el canal de ruido, usando el canal de tono 2 como reloj.

64- utiliza el filtro paso alto en el canal de ruido, usando el canal de tono Ø,o de reloj.

128- utiliza el modulador de anillo con el canal de tono 1.

Para seleccionar una combinación de opciones del tipo de sonido, sumar los valores de cada opción individual y espedificar el número resultante como parámetro STYLE.

SPEAKER
(ALTAVOZ) SET SPEAKER ON/OFF

Controla la producción de sonido del altavoz interno; para silenciar rápidamente la máquina se utiliza SET SPEAKER OFF.

Las funciones trigonométricas trabajan en grados o radianes (ver la sentencia OPTION). Las funciones mínimas del BASIC son: ABS, ATN, COS, EXP, INT, RND, SGN, SIN, SQR, TAB y TAN, LOG.

ABS(X) El valor absoluto de un número. Significa sólo retirarle el signo. Así, ABS(-9) sería 9.

ACOS(X) El ángulo asociado del coseno X, es decir, el opuesto a cos. Así ACOS(COS(X)) es X.

ANGLE(X,Y) El ángulo entre el eje X positivo y la línea que une el punto (ØØ) con el punto (X,Y).

ASIN(X) El ángulo del que X es el seno.

ATN(X) El ángulo del que X es la tangente.

BIN(X) Devuelve el número correspondiente a la representación binaria dada, por ejemplo BIN(11ØØ1) es 25.

BLACK El color negro, equivalente a RGB (Ø,Ø,Ø)

BLUE El color azul, equivalente a RGB (Ø,Ø,1).

CEIL(X) Da el menor número entero no inferior a X. En otras palabras, X se 'redondea' el número entero más cercano. CEL (3,45) sería 4 y CEL(-3,45) sería -3.

CHR$(X) Devuelve el carácter del que X es el número de código ASCII

COS(X) El coseno de X.

COSH(X) El coseno hiperbólico de X.

COT(X) La cotagente de X.

CSC(X) La cosecante de X.

CYAN El color morado, equivalente a RGB (Ø,1,1).

DATE$ Devuelve la fecha actual en formato standard (año-mes-día). Ver comando DATE.

DEG(X) Convierte X de radianes en grados
DEG(X) = X * 18Ø/PI.

EPS(X) La menor cantidad que puede sumarse a X o restarse de ella para que el ordenador registre un cambio en el valor de X.

EXLINE Devuelve el número de la última sentencia que provocó una excepción.

EXP(X) Devuelve el valor de e elevado a la potencia de X. El número conocido como 'e'(2.71828...) es la base de los logaritmos naturales.

EXSTRING$(N) Devuelve la serie de mensajes asociada al número de excepción N. Obsérvese que la serie comienza con un espacio.

EXTYPE Devuelve el número de categoría de la última excepción.

FP(X) FP representa la parte fraccionaria. FP(1.23) sería Ø.23 y FP(-1,23) sería -Ø,23. FP es el opuesto de IP.

FREE La cantidad de memoria libre que puede utilizar el BASIC (ver el comando INFO).

GREEN El color verde, equivalente a RGB (Ø,1,Ø).

HEX$(X$) Devuelve una serie de bytes dando los valores hexadecimales de los bytes en X$. Los bytes hexadecimales están en mayúsculas o en minúsculas, separados por comas, por ejemplo HEX$("21,E3, ff").

IN(N) Lee un byte de la puerta N I/O.

INF El mayor número positivo que puede manejar el Enterprise, su idea del infinito. Este número es 9.9999999999*1Ø^62.

INKEY$ Devuelve el carácter del teclado si se pulsa una tecla, en caso contrario devuelve una serie nula ("").

INT(X) El mayor número entero no superior a X. Por tanto INT(3,4) sería 3 e INT(-3,4) sería -4.

IP(X) La parte entera de X. Significa que todas las cifras

que siguen a la coma decimal se eliminan. IP(9.9) sería 9, e IP(-9.9) sería -9.

JOY(N) Da un valor que depende del estado de los conmutadores de la palanca de mando especificada:

1- derecha
2- izquierda
4- abajo
8- arriba
16- botón de disparo

Obsérvese que los valores podrían sumarse entre sí, si la palanca de mando apuntara diagonalmente, o si se pulsara también el botón de disparo.

Las palancas de mando (valor N) están numeradas de Ø a 2, siendo Ø la palanca de mando incorporada. Para la palanca de mando incorporada, la barra espaciadora es el botón de disparo.

LBOUND(A) Límite inferior de dimensión de una matriz unidimensional A

LBOUND(A,N) Límite inferior de dimensión N de una matriz A.

LCASE$(A$) Convierte todos los caracteres alfabéticos de mayúsculas en minúsculas.

LEN(A$) El número de caracteres (longitud) de A$.

LOG(X) El logaritmo natural (logaritmo base e) del número X.

LOG 1Ø(X) El logaritmo de X con base 1Ø.

LOG2(X) Logaritmo de X con base 2.

LTRIM$(A$) Elimina todos los espacios que están al comienzo de la serie A$. Así, LTRIM$(" Hello") sería "Hello".

MAGENTA El color magenta, equivalente a RGB(1,Ø,1)

MAX(X,Y) Devuelve el mayor número de X e Y. Así, MAX(6,99) es 99.

MAXLEN(A$) Da la longitud máxima especificada para una variable de serie o matriz.

MIN (X,Y) Como MAX (X,Y) pero devuelve el número menor.

MOD (X,Y) X módulo Y. O, en términos más simples, el entero resto de X dividido por Y. Obsérvese que MOD(-1,3)=2. Ver REM(X,Y).

ORD (A$) Da el código ASCII del carácter entre comillas, o el código ASCII del primer carácter de una variable de serie. ORD significa ordinal, y es el número asociado al carácter, en el juego de caracteres usado por el ordenador. Dado que el ENTERPRISE utiliza el ASCII, se devuelve el valor ASCII.

~~ORD (A$)~~ Da el código ASCII del carácter entre paréntesis, o el código ASCII del primer carácter de la variable de serie.

PEEK (N) Devuelve el byte en la dirección N del microprocesador Z80.

PI El número conocido como pi. En el ENTERPRISE se redondea a 3,141592654. Devuelve el valor.

POS (X$, Y$) Da la posición en X$ (contando los caracteres de izquierda a derecha) en donde ocurre por primera vez Y$. Si no pueden encontrarse Y$ en X$, el resultado es Ø. Añadiendo un número después de la segunda serie (por ejemplo POS (X$,Y$,X), se puede ordenar a la máquina que empiece a buscar Y$ - desde un lugar especificado en X$. Si X$ es "LONDON" e Y$ es "ON", entonces POS (X$, Y$) es 2. Pero POS (X$,Y$,4) diría al ordenador que empezara desde la "D" de "LONDON" cuando está buscando "ON", y daría el resultado 5.

POS (A$, B$, M) Versión alternativa de POS. Ver arriba.

RAD (X) Convierte X de grados en radianes.
RTD(X)=XxPI/18Ø .

RED El color rojo equivalente a RGB (1,Ø,Ø)

REM (X, Y) El resto de X dividido por Y. Obsérvese REM(-1,3)= - 1. Ver MOD (X,Y).

RGB Devuelve el número de color dependiente de la máquina equivalente a la mezcla especificada de los colores rojo, verde y azul. R especifica la proporción de rojo (Ø a 1), G especifica la de verde (Ø a 1) y B la de azul (Ø a 1).

Ejemplo SET INK RGB (1/2, 1/3, 1/4)

RND Genera un número aleatorio entre Ø y 1. Para

RAD

uso práctico, los números aleatorios se multiplican y se convierten en números mayores. INT(RND*1ØØ) daría un número aleatorio entero entre Ø y 99, ambos inclusive (RND nunca es 1).

RND(X) Genera un número aleatorio entero menor que X. El mayor valor permisible para X es 32767.

ROUND(X,N) Redondea X a N lugares decimales. ROUND(1,7668,2) sería 1,77. ROUND(-1.7668,2) sería -1,76.

RTRIM$(A$) Corta los del final de la serie. Como LTRIM$ pero eliminando espacios de la derecha.

SEC(X) La secante de X

SGN(X) Devuelve el signo de X. Devuelve -1 si X es negativo, Ø si X es Ø y 1 si X es mayor que Ø.

SIN(X) El seno de X.

SINH(X) El seno hiperbólico de X.

SIZE(A) El número de elementos en la matriz A.

SIZE(A,N) El número de elementos que se permiten en la dimensión N de la matriz.

SPEEK(S,N) Como PEEK, pero devuelve el byte en la dirección del sistema N dentro del segmento S.

SQR(X) La raíz cuadrada de X. X debe ser positivo.

STR$(X) Convierte el valor X en una serie de dígitos sin espacios a la izquierda ni a la derecha, pero con un signo '-' si X es negativo.

TAB(X) Sólo se permite en las sentencias PRINT. Mueve la posición del cursor a la columna X de la fila actual.

TAN(X) Tangente de X.

TANH(X) Tangente hiperbólica de X.

TIME$ Devuelve la hora actual en el formato standard (horas:minutos:segundos). Ver el comando TIME.

TRUNCATE(X,N) Elimina N lugares decimales de X

UBOUND(A) Límite superior de la dimensión de una matriz unidimensional A.

UBOUND(A,N) Límite superior de la dimensión N de una matriz A.

UCASE$(A$) Convierte en mayúsculas todas las letras de la serie A$.

USR(N,X) Llama una dirección N (que probablemente ha sido definida usando CODE), y pasa el entero X a HL a la rutina del código de máquina . El valor que quede en HL será el devuelto por USR.

VAL(A$) Convierte una serie en un número (el opuesto de STR$). VAL empieza convirtiendo en el primer dígito de la serie, y se detiene cuando llega al primer carácter que no es un dígito.

WHITE El color blanco equivalente a RGB (1,1,1)

WORD$(N) Devuelve una serie de 2 bytes que contiene los bytes superior e inferior de N, que se supone es una dirección. N será normalmente una dirección definida por una sentencia CODE, y permite que se realicen saltos hacia atrás, etc., usando etiquetas. El primer byte de la serie será LSB. C\$= WORD$(TEST):a=ord(a$(1))+256*ord(a$(2))

YELLOW El color amarillo equivalente a RGB (1,1,Ø).

VER$
VERNUM

EXOS

EXOS es la abreviatura de Enterprise Expandable Operating System (Sistema Operativo Ampliable Enterprise). Un sistema operativo es un programa que intenta que se utilice un ordenador y sus facilidades de la mejor y más fácil manera posible. Forma un interface (elemento intermedio o de unión) entre los programas de alto nivel (como el lenguaje BASIC) y el ordenador. Las facilidades principales de un ordenador son sus dispositivos y periféricos. Los periféricos son cosas tales como las pantallas, el cassette, la impresora, etc. Por ello, la parte principal de un sistema operativo maneja los dispositivos y los periféricos: el sistema de entrada/salida. Otras facilidades de las que se encarga el sistema operativo incluyen la distribución de la memoria.

SISTEMA DE ENTRADA/SALIDA

El microordenador Enterprise es sumamente complejo; para ejecutar funciones incluso sencillas como incluir una serie en la pantalla necesita millares de instrucciones a nivel de máquina y para transmitir esa misma serie en una impresora necesita otros centenares de instrucciones. El sistema operativo de Enterprise racionaliza la interface entre un programa y el microordenador, haciendo que la impresión de una serie en una impresora sea tan sencilla como la de la serie en la pantalla. Esto se consigue permitiendo que los programas traten todos los dispositivos de entrada y salida de manera idéntica. Toda la entrada y salida se ejecuta a través de 'canales' (un canal conecta simplemente un programa a un dispositivo). Los canales están enumerados del Ø al 254. El sistema operativo proporciona en los canales las funciones siguientes:

Nº de código	Función
Ø	RESET del sistema
1	Abrir un canal (conectar un dispositivo)
2	Crear y abrir un canal
3	Cerrar un canal (desconectar)
4	Cerrar y suprimir un canal
5	Leer un carácter en un canal
6	Leer un bloque
7	Escribir un carácter en un canal
8	Escribir un bloque
9	Devolver el estado del canal
1Ø	Introducir y leer el estado del canal
11	Ejecutar una función especial
16	Leer, escribir o cambiar de posición una variable del sistema.
17	Captar la entrada de un canal a otro

18	Redirigir un canal
19	Introducir nombre dispositivo por omisión
20	Devolver estado del sistema
21	Enlazar dispositivo
22	Leer límites del sistema
23	Introducir límite de usuario
24	Asignar un segmento
25	Liberar un segmento
26	Explorar extensiones
27	Asignar buffer (memoria intermedia) de canal
28	Devolver un mensaje de error
29	Cargar módulo
30	Cargar módulo reubicable
31	Ajustar la hora
32	Leer la hora
33	Ajustar la fecha
34	Leer la fecha

Estas funciones las utiliza el BASIC para proporcionar facilidades de entrada/salida. Se encuentran disponibles para todos los lenguajes que se utilizan, y proporcionan así un método uniforme de comunicación con dispositivos. Están igualmente disponibles para el programador de código de máquina, facilitándole la escritura de programas en código de máquina.

Para llamar al sistema operativo desde un programa de código de máquina, se requiere una única instrucción, seguida por el código de la función. Por ejemplo, para abrir un canal, se necesita el código siguiente:

Código de máquina	Código de ensamblador
F7	RST3ØH
Ø1	DB 1

El sistema operativo del Enterprise proporciona muchas más funciones que las enumeradas arriba. En el Manual Técnico del Enterprise puede encontrarse una lista completa de las funciones y de las convenciones para llamada.

MEMORY USAGE
(Uso de la memoria)

El sistema operativo se basa en una memoria ROM (Memoria de Lectura solamente). Esto significa que el programa está almacenar sus datos. El Enterprise puede manejar una gran cantidad de almacenamiento en RAM y en ROM.

Este almacenamiento se maneja dividiéndolo en 'páginas' (no confundirlos con las páginas de video); cada

página tiene 16K bytes de longitud, y hay un total de 256 páginas, que dan una capacidad de almacenamiento de 4M bytes. El Z80 del Enterprise sólo puede usar 4 de estas páginas al mismo tiempo (en vez de lo que ocurre al leer un libro, en el que sólo podemos ver y usar 2 páginas al mismo tiempo).

MENSAJES DE ERROR

Con frecuencia es imposible evitar caer en un extraño error en un programa. Podría ser difícil encontrar dónde está el error ... e incluso en qué consiste.

El ordenador nos ayuda pasándonos mensajes que nos indican lo máximo posible sobre lo que está equivocado. Si ejecutamos un programa que contiene un error BASIC, el ordenador se detendrá en el punto en el que ya no puede entender el programa y hará aparecer en pantalla una breve declaración indicando la causa del problema.

Recordemos: el ordenador no puede hablarnos de igual manera sobre otros tipos de error. Si, por ejemplo, olvidamos la 'prioridad del operador' o pensamos que el resultado de un cálculo debería ser diferente del que es en realidad, es posible que esto no detenga la ejecución del programa sino que, simplemente, el programa estará haciendo otra cosa distinta a la que habíamos pensado nosotros. El ordenador sólo puede detectar errores en la sintaxis u organización del BASIC, o los problemas causados porque es imposible la acción exigida por el programa.

```
*** NOT UNDERSTOOD          (No se entiende)
```

Si el programa se está ya ejecutando cuando surge un problema que hace imposible continuar, el mensaje de error contendrá el número de línea correspondiente; por ejemplo:

```
*** INVALID ARGUMENT TO SQR          (Argumento inválido para SQR)
3ØØ PRINT SQR(Y)
```

En este momento se puede subir el cursor y editar la línea 3ØØ. Las funciones EXLINE, EXTYPE y EXSTRING$ se proporciona para ayudar a manejar errores y otras excepciones. Cada error, tiene su propio número, que puede referenciarse con EXTYPE, y para la mayoría de estos errores se imprimirá un mensaje especial si no está suprimido.

Si ocurre una excepción no cubierta por los mensajes incorporados, se imprime el tipo general de error, junto con el número de excepción, por ejemplo:

```
* Error de desbordamiento tipo 1234
```

Los tipos generales de error son:

Ø	- 999	Usuario
1ØØØ	- 1999	Desbordamiento
2ØØØ	- 2999	Subíndice
3ØØØ	- 3999	Matemático
4ØØØ	- 4999	Parámetros
5ØØØ	- 5999	Memoria agotada
6ØØØ	- 6999	Matriz
7ØØØ	- 7999	Uso de fichero
8ØØØ	- 8999	Entrada-salida
9ØØØ	- 9999	Exos
1ØØØØ	- 1Ø999	Control
11ØØØ	- 11999	Gráficos
12ØØØ	- 12999	Tiempo real
2ØØØØ	- 2Ø999	Sintaxis
3ØØØØ	-	Sistema

Los mensajes específicos con:

1ØØØ - Valor inesperado dado
1ØØ1 - Desbordamiento en constante numérica
1ØØ2 - Desbordamiento en expresión numérica
1Ø51 - Desbordamiento en expresión de serie
11Ø6 - Desbordamiento en asignación de serie (es decir, serie demasiado larga)
2ØØ1 - Subíndice de la matriz fuera de sus límites
3ØØ1 - División por cero
3ØØ4 - Argumento inválido para LOG
3ØØ5 - Argumento inválido para SQR
3ØØ7 - Argumento inválido para ASIN o ACOS
4ØØØ - Error en parámetros DEF
4ØØ2 - Argumento para CHR$ fuera de gama
4ØØ3 - Argumento inválido para ORD
4ØØ4 - Indice de SIZE fuera de gama
4ØØ5 - Argumento de TAB fuera de gama
4ØØ8 - Indice de LBOUND fuera de gama
4ØØ9 - Indice de UBOUND fuera de gama
43Ø1 - Error en parámetros CHAIN

5ØØØ - Memoria insuficiente
51ØØ - Espacio de pila insuficiente
511Ø - Espacio de ampliación insuficiente
512Ø - Espacio ALLOCATE insuficiente

7ØØ1 - Número de canal inválido
7ØØ3 - Canal ya abierto
7ØØ4 - Canal no abierto
74Ø1 - Canal TRACE no abierto
8ØØ1 - Sin datos en READ/INPUT
81Ø1 - Se esperan datos numéricos
82Ø1 - Serie USING no válida
82Ø2 - No hay elemento de formato en la serie USING
82Ø3 - El elemento de formato en USING es demasiado corto

92Ø8 - Error CRC en cassette
92Ø9 - Editor-fichero de carga demasiado grande
921Ø - Editor-error en el canal de teclado
9211 - Editor-error en el canal de teclado
9212 - Editor-error en el canal de video
9213 - Existe ya un enlace con la red
9214 - No está introducida la dirección de red
9215 - No puede utilizarse al mism otiempo serial y red
9216 - Posición del haz no válida
9217 - Coordenadas del cursor no válidas
9218 - Número de línea para hojear no válido
9219 - Fichero de página de video no válido
9220 - Parámetro de visualización no válido
9221 - Modalidad de video no válida
9222 - Tamaño de página de video no válido
9223 - Está llena la cola de sonido
9224 - Está lleno el almacenamiento de la envolvente
9225 - Envolvente demasiado grande
9226 - Serie de teclas de función demasiado larga
9227 - Violación de la protección
9228 - Fin de fivhero inesperado
9229 - Se ha pulsado la tecla STOP
9230 - Secuencia de escape no válida
9231 - Este dispositivo no soporta la llamada
9232 - Número de unidad no válido
9233 - El dispositivo se está usando ya
9234 - Llamada de función no válida
9235 - Valor de la fecha o la hora no válido
9236 - Módulo de fin de fichero
9237 - Módulo reubicable no válido
9238 - Tipo de módulo desconocido
9239 - Cabecera de fichero Enterprise no válida
924Ø - Serie de comandos no reconocida
9241 - Descriptor de dispositivo no válido
9242 - Número de variable EXOS desconocido

9243 - Límite de usuario no válido
9244 - No se puede dejar libre el segmento
9245 - No hay segmento libre
9246 - Memoria de video insuficiente
9247 - Materia insuficiente
9248 - Error abierto en canal
9249 - El canal existe ya
925Ø - El dispositivo no existe
9251 - El canal no existe
9252 - Desbordamiento en la pila EXOS
9253 - serie EXOS no válida
9254 - No se permite la llamada en función EXOS
9255 - Código de función EXOS no válida

1ØØØ2 - Retorno sin GOSUB
1ØØØ4 - No hay ningún CASE seleccionado
1ØØØ5 - El programa no existe

2ØØØØ - No se entiende
2ØØØ1 - Número de línea no válido
2ØØØ2 - Gama del número de línea no válida
2ØØØ4 - El número de línea no existe
2ØØ1Ø - No se puede especificar RENUMBER
2ØØ2Ø - No es posible continuar
2ØØ3Ø - Se espera el identificador
2ØØ31 - Se espera el identificador de la serie
2ØØ32 - Se espera el identificador de la matriz
2ØØ34 - El tipo no coincide
2ØØ4Ø - Variable no inicializada
2ØØ41 - El identificador está declarado dos veces
2ØØ42 - Identificador demasiado largo
2ØØ43 - Faltan las comillas finales
2ØØ5Ø - Falta el fin del bloque
2ØØ51 - Fin de bloque no válido
2ØØ52 - Demasiados bloques encajados
2ØØ6Ø - Uso de opción de máquina no válido
2ØØ71 - Sentencia y modalidad inmediata
2ØØ72 - Comando en programa
2ØØ73 - Sentencia no permitida después de THEN
2ØØ74 - Línea de multisentencia no válida
2ØØ75 - Línea demasiado larga
2ØØ8Ø - Formato de fichero no válido
2ØØ81 - Los programas no VERIFY (no verifican)

3ØØØØ - Los datos del BASIC han sido corrompidos.

Los mensajes de error pueden ser bloqueados, si se desea, usando WHEN EXCEPTION y un bloque manipulador (página 134). Se puede utilizar un manipulador de excepciones para bloquear cualquier error, incluso los de desbordamiento de memoria o error de sintaxis (por ejemplo, una palabra clave mal escrita). Esto debe manejarse con cuidado ya que RETRY de un error permanente hará que el programa gire en bucle indefinidamente.

Los errores como la división por cero o un argumento SQR negativo se pueden solucionar sin tener que desmantelar el programa.

GLOSARIO

Este Glosario se incluye aquí para que nos familiaricemos con toda la jerga de la informática que podemos encontrar a medida que aumentan nuestros intereses. La mayoría de las palabras del Glosario aparecen en algún lugar del manual, pero no así otras, ya que el manual ha intentado, en la medida de lo posible, evitar la jerga y dar en su lugar explicaciones. Este Glosario es de gran utilidad cuando se leen libros o revistas sobre informática que no contienen glosario pero que están llenos de palabras que no se entienden.

ACCEDER (ACCESS) Significa recuperar información de un dispositivo de almacenamiento exterior, por ejemplo, una impresora o una cassette. También significa recuperar información de un programa tal como una base de datos.

ALFANUMERICO (ALPHANUMERIC) Letras o números. El nombre que se da al juego de caracteres, excluídos los caracteres especiales, los de puntuación o gráficos.

ALGORITMO (ALGORITHM) Las series de ideas y tareas que están detrás de un programa. En primer lugar hay que disponer el propio algoritmo y escribir después el programa. Un algoritmo es el sistema por el que se resuelve un problema.

ALMACENAMIENTO (STORES) Vocablo que designa la memoria en la que puede escribirse (RAM o discos, por ejemplo).

ALMACENAR (STORE) Reservar información o datos para uso posterior temporalmente en memoria RAM o permanentemente en cassette o disco. La memoria o los terminales periféricos en donde puede preservarse información se denominan a veces dispositivos de almacenamiento o dispositivos de almacenamiento en masa.

ANSI American National Standards Institute. La contrapartida americana del Instituto Británico de Normas (BSI). Un comité conjunto del ANSI y de la Asociación Europea de Fabricantes de Ordenadores creó la especificación de la norma BASIC.

ARGUMENTO (ARGUMENT) Ver OPERANDO y PARAMETRO

ASCII American Standard Code for Information Interchange (Código Standard Americano para Intercambio de Información). Sistema de caracteres de codificación que se utiliza dentro del ordenador y para transmisión de un ordenador a otro. A cada carácter se le da un número.

BASE DE DATOS (DATEBASE)	Almacén de datos. Se utiliza también a menudo como breve descripción de un programa destinado pura y simplemente a almacenar y manipular datos: por ejemplo, direcciones postales. Las bases de datos varían en complejidad, desde el tipo que nos permite solamente escribir información y recibirla a medida que se escribe hasta programas muy complejos que nos permiten diseñar todo un sistema de ficheros en el ordenador y/o realizar operaciones de clasificación o extracción.
BASIC	El lenguaje de programación que se suministra con el Enterprise (y con la mayoría de los micro-ordenadores). Hay varios tipos diferentes o dialectos del BASIC. El nombre es una sigla que significa Beginners All-purpose Symbolic Instruction Code. Fue diseñado originalmente en los ordenadores de gran tamaño como ayuda para el aprendizaje de la programación.
BASIC MICROSOFT (MICROSOFT BASIC)	Descripción genérica de los intérpretes del BASIC diseñados por Microsoft en los Estados Unidos. La mayoría de las características del BASIC Microsoft pueden utilizarse como subconjunto de las características que se proporcionan en el BASIC del Enterprise.
BAUDIO (BAUD)	La velocidad a la que pueden transmitirse datos desde o hacia el ordenador. Equivale a 'bits' por segundo. El Enterprise carga solamente un programa desde cassette a una velocidad equivalente a 24ØØ baudios.
BINARIO (BINARY)	Contar con 2 como base en vez de 10. Los números binarios están pues compuestos totalmente por unos y ceros. Así 9 sería 1ØØ1. Los números binarios pueden contemplarse en columnas, al igual que los números ordianrios. En lugar de unidades, decenas y centenas, tenemos unidades, doses, cuatros, ochos, dieciseis, treinta y doses, etc. El ordenador piensa en binario.
BIT	Dígito binario (Ø ó 1). La más pequeña unidad de información que puede reconocer el ordenador. Puede representarse también como tensión alta/baja (como dentro del ordenador) o impulsos magnéticos positivos/ negativos, como en una cassette.
BORRAR (WIPE)	Vaciar un medio de almacenamiento (disco o cassette, por ejemplo) o la memoria del ordenador.
BUCLE (LOOP)	Repetición dentro de un programa por retorno a una línea que es el comienzo del bucle. Parte de un programa cuyo final vuelve otra vez al comienzo salvo que se cumpla una condición de salida. Un bucle infinito es un bucle sin condición de salida, y es la causa normal de que un ordenador quede 'colgado' y no haga nada útil.

BUSCAR (SEARCH)	Encontrar en un fichero o listado algo en particular.
BYTE	Ocho bits, Por esta razón se le conoce también como octeto. Se considera como una unidad dentro del ordenador. La memoria se mide en bytes o kilobytes (1Ø24 bytes, llamado también K) o megabytes (1Ø24 K bytes). Para alcanzar un carácter se necesita normalmente un byte.
CABLE COAXIAL (COAXIAL CABLE)	El cable que conecta al ordenador al receptor de TV. Técnicamente es un cable cuyo núcleo central está totalmente rodeado por otra capa conductora, para impedir interferencias.
CAMPO (FIELD)	Parte del registro de un fichero. Por ejemplo, una dirección postal podría ser un registro. El nombre de la persona sería un campo, el nombre de la calle otro, etc. . Los campos permiten una manipulación más detallada de la información haciendo que sea más fácil encontrar sus distintas partes.
CANAL (CHANNEL)	'Ruta' a través de la cual puede entrar o salir la información del ordenador, o pasar entre diferentes partes del mismo. El uso de un canal puede ser normalmente redefinido por un programa.
CARACTER (CHARACTER)	Símbolo utilizado para representar información; las letras, números, signos del operador y signos de puntuación con caracteres.
CARGAR (LOAD)	Sacar algo del almacenamiento y llevarlo a la memoria del ordenador.
CARTUCHO (CARTRIDGE)	Caja de plástico que contiene uno o dos 'chips' donde se encuentra el programa. Se puede enchufar en la ranura lateral del Enterprise, proporcionando un programa que se ejecutará instantáneamente sin necesidad de cargar ni escribir. Las consolas de video utilizan cartuchos para los distintos juegos. Cuando se quiere un juego diferente se puede cambiar el cartucho.
CIRCUITO INTEGRADO (INTEGRATED CIRCUIT)	Otra palabra para indicar 'chip' de silicona.
CLASIFICAR (SORT)	Distribuir de nuevo un fichero o lista en un determinado orden, por ejemplo, por orden alfabético.

CODIGO (CODE)
Ver CODIGO DE CONTROL. 'Código' describe también el texto de un programa (código fuente) y el programa que ejecuta el ordenador (código objeto). Ver también CODIGOS DE MAQUINA.

CODIGO DE CONTROL (CONTROL CODE)
Carácter que no es visible en la pantalla pero que, sin embargo, provoca una determinada acción por parte del ordenador. Algunos ejemplos son GHR$(8), que es 'retroceso' y CHR$(13) que es 'retorno del carro'.

CODIGO DE MAQUINA (MACHINE CODE)
El código de máquina real está siempre en binario. Así es como trabaja realmente el ordenador, y lo que utilizamos -el BASIC- es un programa ejecutado por las inscripciones en código de máquina del ordenador. El lenguaje ensamblador ("Assembler") describe directamente el código de máquina, utilizando letras y símbolos para representar los números binarios.

COMANDO (COMMAND)
Palabras clave de un programa, por ejemplo, GOTO, CALL, etc., que envían al ordenador instrucciones directas. La primera palabra clave de la línea de un programa suele ser un comando.

COMPILADOR (COMPILER)
Programa especial que traduce un lenguaje del ordenador a otro comando normalmente el código fuente en alto nivel y produciendo un código objeto de máquina. Al contrario de lo que ocurre con el INTERPRETE, no se puede efectuar cambios instantáneos en un programa que utiliza compilador, pero normalmente el programa se ejecuta más rápidamente cuando ha sido compilado.

CONCATENACION (CONCATENATION)
Unir entre sí dos series usando el signo &. A$ & B$, sería una serie larga formada por el contrario de A$ seguido por el de B$.

CONCURRENTE (CONCURRENT)
Algo que ocurre al mismo tiempo que otra cosa. Nuestra respiración es concurrente con nuestro latido del corazón.

CONDICIONAL (CONDITIONAL)
Describe parte de un programa que utiliza condiciones para tomar decisiones. IF/THEN es una sentencia condicional; podría traducirse aproximadamente como: 'a condición de que A 1Ø ENTONCES ...'

CONSTANTE (CONSTANT)
Número o serie que no cambia. Por ejemplo, la palabra PI significa una constante (3,14...), utilizada en los cálculos de circunferencias y círculos. Dos es una constante; "A" es una constante. Hay casos en que lo que se conoce como 'variables' se utiliza en realidad como constante en todo el programa.

COORDENADA (CO-ORDINATE)
Número que especifica la posición de algo, especialmente en la pantalla. PRINT AT utiliza dos coordenadas, una vertical y otra horizontal, para especificar el lugar en donde debe inscribirse un carácter. PLOT utiliza también coordenadas para especificar las condiciones de los puntos al comienzo o final de una línea de gráficos.

CTRL
Abreviatura de 'control'. Se utiliza como clave para generar directamente desde el teclado códigos de control.

CHIP
Una pequeña caja de circuitos microscópicos dentro del ordenador. Un 'chip' sólo tiene 2 mm^2 (aunque puede variar de tamaño) y está hecho de silicona, que es un material conocido como semiconductor o metaloide. Va envuelta en un estuche de celuloide para proteger sus circuitos delicados y tiene puntas metálicas para conectarse con otros chips del ordenador. Los chips de silicona se conocen también como Circuitos Integrados.

DATO (DATA)
(1) Sentencia en BASIC que indica al ordenador que incluímos palabras o números constantes para utilizar en un programa; se ponen en uso con el comando READ. (2) Letras o números utilizados por un programa para proporcionar información útil o ejecutar una tarea. Los números que se utilizan en la descripción de nuestros propios caracteres son datos.

DECIMAL
El sistema de recuento a que todos estamos habituados que utiliza una base de 1Ø.

DECLARACION (DECLARATION)
Decir al ordenador que vamos a utilizar una determinada variable o matriz.

DECLARACION (STATEMENT)
Parte autónoma y significativa de un programa. Las líneas IF/THEN forman declaraciones. IF A=2 por sí sólo no tendría mucho significado. Pero lo tiene IF A=2 THEN PRINT "A=2".

DEFINIR (DEFINE)
Especificar, especialmente para uso posterior, los detalles de algo. Se aplica normalmente a un carácter diseñado por nosotros o a una función.

DEPURAR (DEBUG)
Buscar y corregir los errores de un programa. Se trata de una etapa crucial en el desarrollo de programas grandes o complejos. Los errores no siempre son evidentes y conviene probar cuidadosamente un programa de todas las maneras posibles.

DERRUMBAMIENTO (CRASH) — Fallo dramático (de un programa o del ordenador en su trabajo) por una serie de razones.

DIMENSIONES (DIMENSIONS) — Se utiliza al describir matrices. ARRAY(X) es una matriz unidimensional y ARRAY(X,Y) es bidimensional.

DISCO (DISK) — Placa redonda para grabación magnética (similar a la cinta), que se utiliza con una unidad de disco para almacenar programas y datos que más tarde pueden ser leídos de nuevo. Es un método de almacenamiento y retorno mucho más rápido que los cassettes. Los discos vienen en una funda permanente de plástico, pero si se sacan de la funda (!sólo debe hacerse con los discos que hayan quedado irreparablemente dañados de alguna manera!) parecen algo así como un disco flexible. Los discos normales que se utilizan en el Enterprise son "microfloppy" de 3,5 pulgadas.

DISPLAY DISPOSITIVO (DEVICE) — Máquina, por ejemplo, el aparato de TV o el grabador de cassette, conectada al ordenador y de algún modo utilizada por él. El Enterprise considera dispositivos tanto al generador de sonidos como a los gráficos. Cualquier unidad periférica acoplada al Enterprise es tratada como dispositivo adicional controlado por el BASIC a través del sistema operativo.

E/S (I/O) — Entrada/salida. Se utiliza para indicar cualquier parte del ordenador que trata del flujo de datos que entra y sale del mismo.

EJECUTAR (EXECUTE) — Llevar a efecto un comando o un programa.

ELEMENTO (ELEMENT) — Parte de una matriz. El número de elementos en una matriz lo especifica el ordenador cuando la declara.

ENTRADA (INPUT) — Cualquier dato o programa que pasa al ordenador, ya sea desde un dispositivo de almacenamiento (disco, etc.) ya a través del teclado.

ERROR (BUG) — Equivocación en un programa. A veces es fácil de descubrir, sobre todo cuando es una palabra BASIC mal escrita. Otras veces no es tan fácil encontrarlos podemos ver sus resultados pero no encontrar las causas de los mismos.

ESCRIBIR (WRITE) — Introducir datos o información. Podemos escribir datos en un fichero, por ejemplo.

ESTRUCTURA (STRUCTURE)
Organización de las partes de un programa; se distingue del algoritmo, que es el plan que prepara cómo un programa resolverá su problema.

EXPONENTE (EXPONENT)
La potencia a la que se eleva una base. 1Ø^3 sería 1Ø*1Ø*1Ø, que es 1Ø elevado al cubo. 3 es el exponente Ver INVOLUCION.

EXPRESION
Grupo de números o palabras que tendrán un valor cuando haya sido calculado. Una expresión numérica del BASIC podría ser algo así: X * 6. EL BASIC tiene también expresiones en serie.

FICHERO (FILE)
Colección organizada de datos. Puede ser un fichero de programa o un fichero de datos que ha de utilizar el programa. Se almacena en disco o cassette (o se lista en una impresora). El fichero está dividido en registros.

FIRMWARE
Programa que está siempre en el ordenador. En el Enterprise (y en casi todos los demás ordenadores domésticos) el BASIC se proporciona como programa llamado INTERPRETE, que está siempre en el ordenador cuando lo encendemos. Este es el firmware.

FUNCION (FUNCTION)
Parte de un programa que realiza una tarea o cálculo específico. Se considera como 'programa dentro del programa'. El ordenador sólo la utiliza como y cuando se le ordene. Una función puede estar predefinida, es decir, proporcionada como parte del BASIC (por ejemplo, INT, CHR$, LOG), o bien introducida por el operador. Las 8 teclas programables del Enterprise se denominan teclas de función porque se puede redefinir la función que ejecutan.

GRAFICOS (GRAPHICS)
La capacidad de un ordenador para realizar dibujos o utilizar líneas y símbolos especiales para presentar la información.

HARDWARE
Parte física de un sistema de ordenador. La maquinaria, ya sea parte del mismo ordenador o un dispositivo separado, pero conectado a él.

HEXADECIMAL
Sistema de recuento que utiliza una base de 16 en lugar de 1Ø. El sistema hexadecimal utiliza los dígitos Ø-9 y A-F (F es 15). Los números hexadecimales llevan a veces el prefijo & o terminan con H, para indicar que son hexadecimales (26H es el 38 decimal). El sistema hexadecimal se utiliza como una manera cómoda de representar números binarios.

IDENTIFICADOR (IDENTIFIER) — Nombre que se da a una variable, función, dispositivo o cualquier otro componente del ordenador o programa. El uso de identificadores significativos facilita la comprensión de los programas.

INFORMACION (INFORMATION) — Se considera a veces como el resultado de la elaboración de los datos. Esto significa los resultados de clasificar o buscar a través de ficheros y/o campos. La información son datos reunidos en forma significativa. Puede ser también cualquier cosa que maneja un programa, por ejemplo, números, series, variables.

INTELLIGENT SOFTWARE — Software que se comporta inteligentemente. Intelligent Software Ltd es la creadora del Enterprise, entre otras cosas.

INTERFACE — Elemento del hardware que forma un lazo de unión entre el ordenador y algo que haya en su exterior. La cassette y los conectores de TV están unidos al Interface dentro del Enterprise.

INTERPRETE (INTERPRETER) — Programa especial del ordenador que interpreta de un lenguaje de ordenador en otro. El BASIC del Enterprise es un intérprete.
Los intérpretes efectúan la traducción, instrucción por instrucción, mientras que hay programas que traducen todas las instrucciones a la vez. A estos se les conoce como COMPILADORES.

INTERRUPCION (BREAK) — Detener un programa en la mitad, normalmente con el resultado de que se debe escribir RUN para empezar de nuevo el programa o CONTINUE para reanudarlo en donde se quedó.

INVOLUCION (INVOLUTION) — Elevar un número a una potencia, por ejemplo, 2^3; ver EXPONENTE.

KILOBYTE — 1Ø24 bytes (el número binario redondeado más cercano de 1ØØØ).

LEER (READ) — Sacar algo del almacenamiento y empezar a utilizarlo Puede ser sinónimo de CARGAR, aunque tiene un significado ligeramente diferente cuando se utiliza como comando del BASIC.

LENGUAJE ENSAMBLADOR (ASSEMBLER LENGUAGE)	Lenguaje de programación que habla al ordenador en sus propios términos. El ensamblador tiene instrucciones más simples que el BASIC y, aunque más aburrido que difícil de utilizar, nos ofrece la posibilidad de programar nuestro ordenador con mucho más detalle que usando el BASIC. La programación en lenguaje Esamblador se denomina programación a bajo nivel. Utilizándola, puede comunicarse con el ordenador con palabras clave, pequeñas y muy simples, y usando códigos como el ASCII más frecuentemente de lo que podrían hacerse en el BASIC.
LOGICA (LOGIC)	La ciencia del razonamiento. Los ordenadores están diseñados para seguir las reglas de la toma de decisiones con lógica. Las palabras AND y OR son lógicas.
LLAMADA (CALL)	Bifurcación dentro de un programa a un subprograma o subrutina. Las funciones definidas por el usuario pueden utilizarse con los programas BTSIC en el Enterprise con el uso de la palabra clave CALL.
MACHACAMIENTO DE NUMEROS (NUMBER CHUNCHING)	Una expresión de la jerga de la informática que significa cálculos rapidísimos y muy complejos con números. Algunos ordenadores diseñados exclusivamente como "machacadores de números", son rápidos y seguros en la ejecución de largas secuencias de cálculos.
MAINGRAME	Ordenador realmente grande. Alguno de ellos tienen un tamaño tal que pueden llenar varias salas.
MATRIZ (ARRAY)	Variable que contiene ella misma algunas variables más. Se puede considerar como una lista (unidimensional) o una rejilla o cuadrícula (bidimensional).
MEGABYTE	Aprox. un millón de bytes equivale a 1Ø24 kilobytes.
MODULAR	Formado por partes más pequeñas interconectadas.
NULO (NULL)	Vacío o con un valor de cero. Una serie nula no tiene caracteres. Un carácter nulo tiene un valor de código cero.
OPERADOR (OPERATOR)	Palabra matemática que se utiliza para hacer referencia a los caracteres que significan operaciones, por ejemplo, *,/,+y=, que significan multiplicación, división, suma e igualdad, son todas ellas operaciones matemáticas.

OPERANDO (OPERAND)	Número o variable objeto de una operación. En la expresión 6*2 el 6 y el 2 serían operandos. Se conoce también a veces como ARGUMENTO.
PALABRA CLAVE (KEY WORD)	Cualquier palabra del BASIC, cada una tiene un significado propio.
PALABRA RESERVADA (RESERVE WORD)	Palabra del BASIC que no puede utilizarse en el BASIC para ninguna otra finalidad. La mayoría de las palabras del BASIC pueden utilizarse como nombres de variables, y las que no pueden usarse así, son palabras reservadas.
PARAMETRO (PARAMETER)	Variable que se da a una función a la que se asigna un valor. En TAN(X). X es el parámetro(O ARGUMENTO).
PARADA (PASS)	Ejecución una sola vez de un bucle.
PASAR (PASS)	Transferencia de valores o variables entre dos programas o dos partes del mismo programa.
PERIFERICA (PERIPHERICAL)	Palabra que se utiliza para describir cualquier máquina que trabaja bajo el control de un ordenador: por ejemplo, el receptor de TV, la cassette, la impresora. Dispositivo utilizado por un ordenador pero que no forma parte del mismo.
PICO (SPIKE)	Subida o bajada rápida de la tensión de red. Puede provocar problemas ocasionales en los sistemas de disco y desconectar incluso temporalmente un ordenador, haciendo que se pierda un programa que se está utilizando en ese momento (esto no sucede a menudo).
PORTABILIDAD (PORTABILITY)	La calidad (bastante rara) de un programa de poderse utilizar en más de un sistema.
PROCEDIMIENTO (PROCEDURE)	Otro nombre del subprograma. Con el BASIC del Enterprise, las funciones proporcionan todas las características normales de los procedimientos.
PROCESO	Realización de operaciones con datos. Incluye el cálculo con números, los gráficos (uso de coordenadas) y, en realidad, cualquier cosa que realiza un ordenador.
PROGRAMAS (PROGRAMS)	Serie de instrucciones ordenadas que establecen una tarea que debe ejecutar el ordenador.

PRUEBA (TEST)
Comprobación de un programa para ver si funciona. En términos de programación, es "echar una mirada a algo" para ver si cumple una condición. El ordenador prueba las variables en las sentencias IF o CASE para comprobar si coinciden con alguna de las sentencias.

PSG
Sigla de Programable Sound Generator. Chip es el interior del Enterprise que transmite señales a un altavoz, produciendo así sonidos.

PUERTA (PORT)
Toma de conexión que enlaza el ordenador (a través de una interface) con un dispositivo periférico.

RAM
Random Access Memory (Memoria de acceso aleatorio). Memoria que se utiliza para almacenar temporalmente programas o datos. Se vacía cuando se escribe la palabra NEW o cuando se apaga el ordenador.

RAMA (BRANCH)
Se conoce también como bifurcación, y es un punto del programa en el que se suspende la ejecución consecutiva del número de línea y el ordenador ejecuta otra parte del programa, por ejemplo, una función o una subrutina.

RED (NETWORK)
Grupo de ordenadores unidos todos entre sí y que pueden comunicarse mutuamente. El Enterprise utiliza un esquema conocido como RED INTELIGENTE.

REFERENCIA (REFERENCE)
Se utiliza normalmente cuando se describe un parámetro que se da a una función y contiene en realidad una variable en vez de simplemente el valor de la variable.

RESOLUCION (RESOLUTION)
Número de puntos disponibles para dibujar en la pantalla. Depende de la calidad de los gráficos disponibles en el ordenador. La resolución puede ser alta o muy baja. Cuanto más alta es, mejor es la definición de líneas y formas cuando aparecen en la pantalla. La más alta resolución del Enterprise permite 672 puntos horizontalmente y 512 verticalmente en la pantalla.

ROBUSTO (ROBUST)
Descripción que se aplica a un programa o elemento del Software que tiene muy pocos errores y que trabaja bien, sea cual fuere lo que hagan con él los usuarios.

ROM — Read Only Memory (memoria de lectura solamente). Memoria que contiene "firmware". No puede cambiarla el operador, aunque puede utilizarse o llamarse. Permanece igual cuando se apaga el ordenador.

SALIDA (OUT) — Cualquier cosa que sale del ordenador: displays, sonidos, listados, impresos, resultados.

SALTO (JUMP) — Igual que RAMA.

SERIE (STRING) — Secuencia de caracteres que no son entendidos pero que pueden ser procesados por el ordenador. En BASIC, dentro de un listado de programas, una serie aparece como una variable con $ al final o como una serie de caracteres entre comillas.

SISTEMT OPERATIVO (OPERATING SYSTEM) — Programa que controla toda la entrada y la salida. Se utiliza normalmente con los sistemas de disco para controlar la ejecución del disco y la organización de los datos en él. El CP/M es un sistema operativo. El Enterprise contiene un sistema operativo muy amplio para controlar todas las características del hardware de la máquina y de los dispositivos que pueden acoplarse a ella.

SOFTWARE — Un programa es software. Instrucciones al ordenador que pueden ser modificadas.

SUBINDICE (SUBSCRIPT) — Número de referencia que se da cuando se accede a una matriz. Por ejemplo, en ARRAY(X), X es el subíndice.

SUBRUTINA (SUBROUTINE) — Es muy similar a una función. Parte de un programa que realiza una tarea específica bajo el control del resto del programa. Una función tiene un comienzo y un fin específicos. La rutina se controla con las palabras GOSUB y RETURN, sobre la base de los números de las líneas. A la función se le da un nombre.

TIEMPO REAL (REAL TIME) — Tiempo que guarda una relación con el mundo real en lugar de hacerlo con el mundo interno del ordenador. Se utiliza frecuentemente cuando se inscribe el proceso de datos simultáneamente a su entrada. INKEY$ es casi una operación en tiempo real. Los juegos infantiles con el ordenador trabajan en tiempo real para dar la impresión de cosas reales que suceden rápidamente. El tiempo real sólo se proporciona en ciertos ordenadores o en determinados programas.

UCP (CPU)	Unidad central de proceso. Un gran chip dentro del ordenador que controla todos los demás. El lenguaje ensamblador está escrito para "conversar" directamente con la UCP y varía según el tipo de la misma. A las UCP se les llama también procesadores. El Enterprise utiliza una UCP Z80.
UNIDAD (UNIT)	Puede significar uno de alguna cosa (un carácter es una unidad de datos) o un dispositivo. El ordenador es una unidad. El dispositivo que mueve el disco es una unidad.
USUARIO (USER)	Persona que utiliza un ordenador o un programa. El término "amistad" para el usuario se utiliza para indicar la facilidad con que pueden utilizarse algunos programas.
VARIABLE	Número o serie que puede cambiar, y al que por tanto le damos un nombre por el que el ordenador lo reconoce.

INDICE