Introducción a la programación en BASIC utilizando el microcomputador DRAGON

---

por Richard Wadman

Introducción a la programación en BASIC utilizando el microcomputador DRAGON

---

por Richard Wadman

I.S.B.N. 84-398-0599-3

D. L.: M. 40.796-1983

Editado por I. C. S. (Informática, Cálculo y Sistemas, S. A.)

INSTRUCCIONES DE MANEJO

Tras abrir la caja, junto con este manual encontrará:

1. Su computador DRAGON.
2. Un cable de conexión para la red.
3. Un cable de conexión para el televisor.

Además de estos elementos, necesitará un televisor provisto de UHF. Su computador DRAGON, funciona indistintamente con un televisor en color o uno en blanco y negro. Sin embargo, para obtener las facilidades gráficas en color del DRAGON, deberá conectarlo a un televisor en color. Estos son todos los elementos necesarios para poner en funcionamiento su DRAGON.

Puede además mejorar las posibilidades de su máquina agregando los siguientes elementos opcionales:

1. Un magnetófono de cassete para grabar datos y programas.
2. Una impresora.
3. Joysticks para juegos.
4. Drives para disquetes que le permitirán gran capacidad de almacenamiento de datos y programas.

Ninguno de estos elementos son indispensables, pero un cassete le evitará el tener que introducir repetidamente lo mismo.

CONECTORES

1. **TOMA PARA TELEVISION**
   Conexión para toma standard de antena de televisión.

2. **BOTON DE RESET**
   Se utiliza para inicializar el sistema. Detiene inmediatamente la ejecución del programa, u operaciones de entrada/salida. Cualquier programa que resida en memoria permanecerá alli después de pulsar reset.

3. **TOMA PARA EL JOYSTICK IZQUIERDO**

4. **TOMA PARA EL JOYSTICK DERECHO**

   Para el 3. y 4. Tomas DIN de 5 pines utilizadas para conectar joysticks, accesorio opcional.

5. **TOMA PARA ENTRADA/SALIDA DE CASETE**
   Toma DIN de 5 pines para conectar un casete.

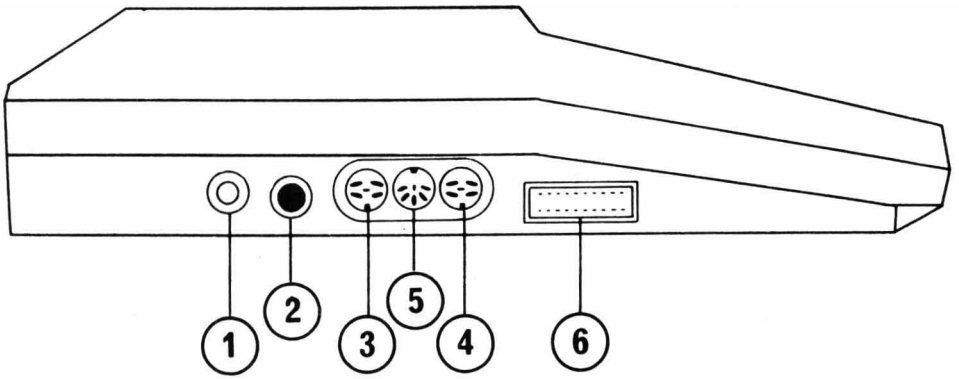
6. **PORT PARA IMPRESORA PARALELO**
   Conexión para una impresora de tipo centronics por medio de un cable centronic standard.

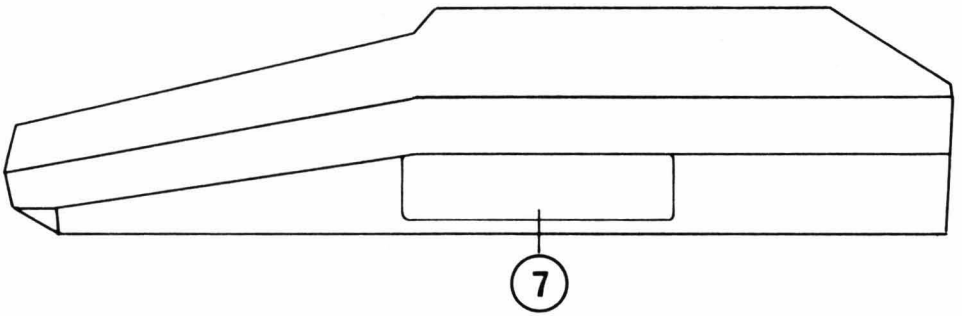
7. **TOMA PARA CARTUCHOS DE PROGRAMAS**
   Utilizado para conectar cartuchos de programas. Deben conectarse cuando el computador esté desconectado.

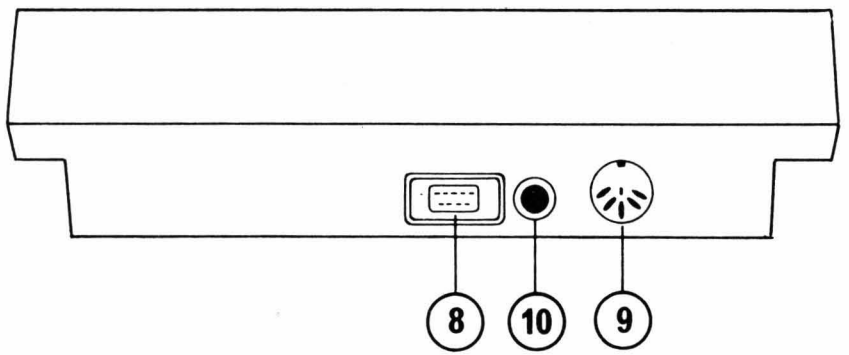
8. **TOMA DE ALIMENTACION**
   Para conectar el cable de alimentación.

9. **TOMA PARA EL MONITOR**
   Para conectar un monitor en color RGB.

10. **INTERRUPTOR DE ENCENDIDO.**


1
2
3
5
4
6


7


8
10
9

COMO CONECTAR EL DRAGON

1. Conecte el cable de conexión del televisor a la toma de antena del televisor y a la toma **1** en el Dragón.
2. Conecte el cable de alimentación a la toma **8** en el Dragón.
3. Encienda el televisor y el computador.
4. Seleccione un canal de televisión y ajuste su frecuencia hasta que aparezca un cuadro verde con un borde negro en la pantalla, (o un cuadro gris claro con un borde negro si el televisor es en blanco y negro).

   En el cuadro verde aparecerá el siguiente mensaje:

   (C) 1982 DRAGON DATA LTD
   16K BASIC INTERPRETER 1.Ø
   (C) 1982 BY MICROSOFT

   OK

   Su computador ya está listo para su uso.

UTILIZANDO CARTUCHOS DE PROGRAMAS

Conecte su Dragón al televisor como se indica arriba. Antes de encender el computador, introduzca, por la toma **7**, el cartucho con la etiqueta hacia arriba.

Antes de introducir o sacar un cartucho de programa, asegúrese de que el computador está apagado. Si no lo hace, puede deteriorar el computador y el cartucho.

CONEXION DE JOYSTICKS

Se pueden conseguir varios tipos de joysticks como accesorio opcional. Los joysticks a utilizar con su computador Dragón deben ser del tipo potenciómetro. Para conectar los joysticks simplemente enchúfelos en las tomas **3** y **4.**

CONEXION DE UN CASETE

Se puede utilizar cualquier aparato de casete, de calidad razonable, para almacenar datos y programas desde su Dragón. El aparato de casete debe tener tomas para control remoto, auriculares y entrada auxiliar.

Los cables de conexión, para la mayoría de los casetes, se pueden conseguir como accesorio opcional. La toma en el computador es la **5**. Las conexiones en el casete dependerán del tipo de éste. Vea el capítulo 4, del manual de programación, para instrucciones sobre como manejar el casete.

COMO CUIDAR SU COMPUTADOR DRAGON

1. Mantenga todo tipo de líquidos lejos de su computador. Al Dragón no le sientan tan bien como a Vd. ni el café ni el té.
2. Asegúrese de que no deja cables colgando en sitios peligrosos. Un tropiezo accidental puede costar caro.
3. Asegúrese de que todas las conexiones están bien hechas antes de encender el ordenador.
4. Desconéctelo todo y apague el ordenador cuando no lo esté utilizando.
5. Para limpiar la carcasa y el teclado, primero desconecte totalmente la unidad de la red. Con un trapo humedecido ligeramente, frote la carcasa y el teclado. No utilice ningún producto con alcohol.

CONTENIDO

INTRODUCCION

Este manual está pensado para aquellos que desean aprender a programar su Dragón utilizando el lenguaje BASIC.

El lenguaje BASIC es un lenguaje de programación muy potente, pero al mismo tiempo muy fácil de aprender. Se compone de menos de 100 sentencias, lo que supone menos de un 1% de la media del vocabulario individual.

Aunque la programación pueda parecer difícil inicialmente, como cualquier cosa nueva, realmente soluciona un problema en un número lógico de pasos. El secreto está en tomárselo con calma y estar seguro de que entiende un paso antes de continuar con el siguiente. No se preocupe si comete errores, esto forma parte del proceso de aprendizaje. No se estropeará el ordenador por cometer un error de programación, encuentre el error, corríjalo y siga adelante. Pruebe sus propias ideas — el "¿Qué ocurrirá si hago esto?" puede ser una forma muy rápida de descubrir de lo que es capaz su ordenador. Introduzca y ejecute cada ejemplo, no solo para ver lo que hace, si no porque lo hace.

Con tiempo y paciencia descubrirá que su computador no es solo una máquina para cartuchos de juegos. La programación, en si misma, puede convertirse en una actividad absorbente y divertida, aparte de los resultados tan divertidos que con ella se pueden conseguir.

CAPITULO UNO

COMENZANDO

EL TECLADO

Suponemos que ya habrá conectado el computador de acuerdo con las instrucciones anteriores, y estará preparado para comenzar. Asi que enciéndalo, su televisor deberá mostrar un cuadro verde con un mensaje. (Si no es asi, apáguelo todo y revise otra vez las conexiones). El mensaje dependerá del tipo de máquina, según tengan conectados drives para disquetes o no, etc. De todos modos, la última línea es siempre la misma.

OK

OK es el mensaje que emite el ordenador, para indicarle que está listo para recibir instrucciones. Deberá esperar a que aparezca el mensaje antes de introducir nada. Debajo del **OK** hay un cuadrado parpadeante — se llama cursor. Le indica la posición de la línea donde se encuentra.

Fíjese en el teclado: se parece a una máquina de escribir con algunas teclas de más. Además se comporta como una máquina de escribir, pulse algunas teclas y verá como aparecen en la pantalla. Observe como se mueve el cursor detrás de cada carácter para indicarle donde se encuentra. Si pulsa la tecla [SHIFT] y Ø (cero) simultáneamente, y sigue escribiendo, verá que los caracteres de la pantalla se han convertido en verdes sobre fondo negro. **A lo largo del manual utilizaremos Ø para representar el cero, para distinguirlo de la letra O.** El ordenador es bastante quisquilloso en cuanto a esta diferencia. Hay letras minúsculas (es decir a, b, c, d); pero solo aparecerán como tales en una IMPRESORA. **Todos los comandos o instrucciones al computador deben de estar en MAYUSCULAS—.** Asi que pulse [SHIFT] y Ø simultáneamente otra vez y escriba algo más. Deberá volver a ver letras negras sobre fondo verde.

Pruebe a continuación la tecla [←]. Es la tecla de retroceso, cuando la pulse el cursor retrocerá a lo largo de la línea y el carácter inmediatamente a su izquierda desaparecerá. Esto es útil para corregir errores, solo deberá retroceder hasta el carácter erróneo y volver a escribir.

Para limpiar la pantalla por completo pulse la tecla [**CLEAR**]; desaparecerá todo lo que hubiera en la pantalla y el cursor se posicionará en el ángulo superior izquierdo. [**CLEAR**] **solo borra la pantalla, la información que esté almacenada en la memoria del ordenador no se verá afectada.**

Practique un rato con el teclado para acostumbrarse a la posición de las teclas y como corregir errores. Después borre la pantalla y asegúrese de que está en el modo de mayúsculas.

LA CALCULADORA DRAGON

Su computador trabaja en dos modalidades distintas—inmediata, obedecerá un comando inmediatamente, y en diferido, almacenará un conjunto de instrucciones y después las ejecutará como un programa. En el modo inmediato el computador se comporta como una calculadora.

El computador entiende un lenguaje llamado BASIC. En BASIC hay un cierto número de palabras especiales que indican al computador que haga ciertas cosas: Por ejemplo, **PRINT, indica que se escriba en pantalla lo que aparezca a continuación.**

Compruébelo, escriba PRINT 12+7 y pulse la tecla [**ENTER**]; la pantalla mostrará

19
OK

Pruebe otro caso, teclee PRINT 12+8/2 después pulse [**ENTER**]

16
OK

Si comete un error al escribir, probablemente recibirá el mensaje

? SN ERROR

Significa "error de sintaxis", que quiere indicar que el ordenador no ha reconocido algo, generalmente porque no ha sido deletreado correctamente. El ordenador dará mensajes de error cuando no entienda un comando, y a veces, cuando entiende el comando, pero lo que se le ha pedido que haga es ilógico o imposible.

Escriba PRINT 3/Ø después pulse [**ENTER**]

El mensaje será

?/Ø ERROR

que indica que se ha intentado dividir por cero, lo cual es imposible.

Los mensajes de error están resumidos para ahorrar espacio, pero una lista completa con las causas posibles se da en el Apéndice C.

Para recuperarse de un error de síntaxis: si detecta el error antes de pulsar la tecla [**ENTER**], puede utilizar la tecla de retroceso [←] para volver y corregirlo. Si no, no hay más alternativa, por ahora, que volver a escribir la línea entera correctamente.

REGLAS ARITMETICAS. OK?

Hasta ahora, nuestros ejemplos solo han pedido al ordenador que realice dos operaciones, estas son sumar (+) y dividir (/). La palabra "operación" significa algo que queremos que haga el ordenador. Hay seis operaciones aritméticas simples que el ordenador realiza y existen reglas extrictas de como se llevan a cabo estas operaciones. En el ejemplo anterior

PRINT 12 + 8/2

no está muy claro si la respuesta debería ser 16 ó 10.

12 más 8 dividido por 2 es 1Ø, pero 8 dividido por 2 es 4, más 12 son 16.

La respuesta dada por el ordenador es 16 debido al orden que elige para realizar las operaciones. Las operaciones y su prioridad son:

1. *Signo menos*

Esto se refiere a cuando un signo menos se utiliza para indicar un número negativo.

Asi, —3+2 se evalúa como —1. Si el ordenador hiciera la suma primero, —3+2 daría —5, pero no es asi.

2. *Potenciación*

La potenciación significa elevar un número a una potencia. 5 elevado a la potencia 4, es 5×5×5×5.

Después de aplicar el signo menos se realizan las potenciaciones.

PRINT 4 + 3 ↑ 2

se evalúa elevando 3 al cuadrado (3 × 3 = 9) y sumando después 4, para dar un total de 13. Si hay más de una potenciación se evalúan de izquierda a derecha. Pruebe un ejemplo:

NOTA: A lo largo del texto aparece el signo ↑ en lugar de la flecha hacia arriba que es el símbolo que deberá pulsar en su ordenador.

PRINT 2 ↑ 3 ↑ 2 ↑ 3 (la tecla con una flecha hacia arriba, a la izquierda del teclado, se utiliza para la potenciación).

esto se evalúa asi: se multiplica 2 por si mismo tres veces (2×2×2=8) después se multiplica el resultado consigo mismo (8×8=64) después se multiplica el resultado consigo mismo 3 veces (64×64×64=262144).

3. *Multiplicación*

El signo que utiliza el ordenador para multiplicar es *, para no provocar confusiones con la letra x. Se obtiene con las teclas [SHIFT] y [:].

PRINT 5 * 2 + 3

da como resultado 13 (5×2=10+3=13).

4. *División*

El símbolo que utiliza el ordenador para dividir es /.

PRINT 5/2+3

da como resultado 5.5 (5/2=2.5 más 3=5.5).

La multiplicación y división tienen la misma precendencia, es decir la misma prioridad. Cuando los operadores aritméticos tienen la misma precedencia, se evalúan de izquierda a derecha.

PRINT 5 + 2 * 3 + 4/2

da como resultado 13 (2 × 3 = 6, 4/2 = 2, 5 + 6 + 2 = 13).

5. *Suma*

El signo para la suma es +

6. *Resta*

El signo para la resta es —

La resta y la suma tienen igual precedencia, asi pues, siempre se evalúan de izquierda a derecha después de efectuar las operaciones con mayor prioridad.

Resumiendo, el orden de precedencia al efectuar operaciones aritméticas es:

PRIMERO — (El signo menos se utiliza para números negativos)
SEGUNDO ∧ (potenciación, de izquierda a derecha)
TERCERO * / (multiplicación y división, de izquierda a derecha)
CUARTO + —(suma y resta, de izquierda a derecha)

Debajo aparecen algunas operaciones aritméticas a evaluar. Calcule primero cada una mentalmente (o con lápiz y papel) y después inténtelo con su ordenador. Si su respuesta es distinta a la del ordenador, intente averiguar porqué. A menos que tenga mucha experiencia en como evalúan expresiones los ordenadores, debería realizar estos ejemplos. La mayoría de los errores achacados al ordenador son provocados por el programador, al seguir un conjunto de reglas distintas de las del ordenador. No se dan las soluciones. Si se introducen tal como se presentan, el computador le dará la respuesta correcta.

PRINT 3 + 2
PRINT 4 + 6 — 2 + 1
PRINT 8 * 4
PRINT 4 ∧ 2 + 1
PRINT 5/4 — 1
PRINT 5 — 4/2
PRINT 6 * — 2 + 6/3 + 8
PRINT 4 + — 2
PRINT 2 * 2 + 3 * 4
PRINT 8/2/2/4
PRINT 2Ø/2 * 5
PRINT 8 * 2/2 + 5 * 3 * 2 ∧ 2

No es necesario que pulse PRINT cada vez, hay un símbolo más corto [?]. Si escribe

? 3 + 2 es lo mismo que PRINT 3 + 2

Ya deberá estar acostumbrado a pulsar la tecla [ENTER] después de cada línea.

El mensaje OK le indica que el ordenador está preparado.
La tecla [ENTER] le indica al ordenador que Vd. está listo.

Después de todas las reglas sobre precedencias, le diremos que es posible modificar las prioridades. Esto se realiza mediante la utilización de paréntesis. Suponga que quiere dividir 14 entre 4 más 3, si escribe: 14/4+3 la respuesta será 6.5, ya que obtendrá 14 dividido entre 4 más 3. Pero esto no es lo que quería. Para evitarlo puede escribir

14/(4+3) la respuesta será 2.

Los paréntesis modifican la precedencia, la regla es sencilla, primero se efectúa lo que se encuentra entre paréntesis. Si hay más de un par, entonces se efectúa de dentro hacia fuera.

12/ (3 + (1 + 2) ∧ 2) se evalúa como sigue

a) 12/ (3 + 3 ∧ 2) (1 + 2) se calcula primero
b) 12/ (3 + 9) 3 ∧ 2 después
c) 12/12 (3 + 9) después
d) 1 por último la división.

He aquí algunas otras expresiones para evaluar. De nuevo, si no está familiarizado en la forma en que trabajan los ordenadores, los pocos minutos que utilice, le permitirán usar su ordenador de un modo más eficiente.

? 44/ (2 + 2)
? (44/2) + 2
? 4 + (—5 * 2)
? 100/ (200/ (2 * 9 (9 — 5)))
? 42/ ((9/3) + 1.75 + (5/4))

IMPRIMIENDO PALABRAS

Hasta ahora, solo hemos utilizado números con el comando PRINT. El profano, a menudo ve al ordenador como un devorador de números, pero ésta no es la única función que tiene. Un ordenador se puede utilizar también para manejar caracteres. Por caracteres nos referimos a las letras de la A a la Z, los dígitos 0 al 9, signos de puntuación y otros caracteres especiales que veremos después. El BASIC nos permite manejar grupos de caracteres llamados **cadenas (strings).**

Una cadena puede ser cualquier grupo de caracteres—incluso un espacio puede ser un carácter importante en una cadena. Para indicarle al ordenador que está manejando una cadena, y no un número, esta se encierra entre comillas (""). Algunos ejemplos de cadenas son:

"EL TITULO" **"Z+*?!"** **"SR. J. PEREZ"**

"M—0000—ZZ" **"01—479—6172"**

Las dos últimas cadenas podrían corresponder a la matrícula de un coche y a un número telefónico, y por ello ambos contendrán caracteres numéricos, no utilizaremos estas colecciones de dígitos para realizar una operación aritmética. El BASIC no considera una colección de dígitos entre comillas como un número, la cadena "12345" es algo totalmente distinto del número 12345. De hecho, las cadenas y los números se almacenan en el ordenador en lugares de la memoria completamente separados.

PRINT "2 + 5 = "; 2 + 5

La primera parte de la sentencia PRINT aparecerá en la pantalla tal y como se ha escrito en la cadena. (Las comillas se usan para encerrar la cadena, no forman parte de ella). La segunda parte de la sentencia se evalúa como una expresión numérica, luego la pantalla mostrará:

2 + 5 = 7

Como hemos visto antes, el espacio puede ser un carácter importante en una cadena. En las sentencias BASIC, los espacios no se tienen en cuenta, solo se utilizan para hacer que la línea sea más legible. Pero en las cadenas si tienen sentido, ya que cuando se imprime una cadena se hace tal y como aparece entre las comillas.

Deberá ser ya capaz de utilizar su computador para resolver problemas sencillos, como los que hay al final de este capítulo. Una vez más, si no está familiarizado con ordenadores, inténtelos

1) Juan mide 168 cm. ¿Cuál es su altura en pulgadas?
(1 pulgada = 2.54 cms).

2) Una receta requiere 0.75 kgs. de harina. ¿Cuántas libras de harina necesita?
(1 kg = 2.2 lbs).

3) Su coche consume 22.8 galones de gasolina en un viaje. Al arrancar, el cuentakilómetros marcaba 10346 y al final del trayecto marcaba 11193. ¿Cuál es el número de millas por galón?

4) Si pone en una cuenta la cantidad de 15 libras a un interés anual del $11^0/_0$. ¿Cuántas libras tendrá en 5 períodos? (C=P * 1 + I/100) ^N; P = Ingreso I = Interés anual N = número de períodos C = cantidad después de N períodos).

5) ¿Cuántas vueltas dará una rueda de bicicleta de 26 pulgadas en una milla? (Longitud de la circunferencia $L=2\times\pi\times$radio $\pi=3{,}14159$).

RESPUESTAS

1) 66.14 pulgadas
2) 1.65 libras
3) 37.149 millas por galón
4) 25.27 libras
5) 775.69

CAPITULO DOS

QUE CONTIENE UN NOMBRE

CONSTANTES

Todos los ejemplos que hemos realizado hasta ahora solo han utilizado constantes. Una constante es exactamente eso — algo que no cambia. 3.145 es una constante, cambiándola por 3.146 la convierte en otra constante distinta. Las constantes son útiles en los programas, pero no tanto como las variables.

VARIABLES

Una variable es algo que puede cambiar de valor. En la ecuación

X + 5 = Y

X e Y son variables, y como tales, podrán tomar muchos valores que harán que la igualdad se cumpla. Las variables se conservan en una zona de la memoria del ordenador — como en casilleros. Para identificarlas hay que etiquetarlas (o darles un nombre como X e Y). Las variables, en su ordenador, pueden ser de dos tipos — numéricas o de cadena, y a su vez de otros dos, simples o matrices. Ya conocemos la diferencia entre constantes numéricas y de cadena — las variables numéricas mantienen constantes numéricas (números) y las variables de cadena mantienen constantes de cadena (caracteres). Por ahora solo consideraremos variables simples, posteriormente manejaremos matrices.

NOMBRANDO VARIABLES

Para nombrar una variable numérica, puede utilizar cualquier combinación de letras, o de letras y números.

N, AA, X, TI, Y, Z9, L5, PQ, K9

son todos ejemplos de variables numéricas válidas.

Su ordenador admite cualquier longitud para el nombre de una variable, pero solo diferenciará con los dos primeros; asi pues, aunque acepte nombres como

PAJARO PASO PATRON

serán tomados como la misma variable **PA.** Lo mismo ocurre con variables de cadena.

NOMBRES DE VARIABLES NUMERICAS

Una variable numérica solo puede almacenar números.

El nombre de una variable númerica puede constar de cualquier combinación de letras y números, pero debe comenzar por una letra.

Ya que el ordenador solo reconoce los dos primeros caracteres de un nombre, nombres como:

ESTRELLA, ESTADO, ES

serán consideradas como la misma (ES).

Las variables con un nombre de más de dos caracteres, son útiles como medio para recordar su contenido,

NUMERO, CONTADOR, SUMA

serán aceptadas, pero ocuparán más memoria que NU, CO, SU.

PEPE$ PESO$ PESCADO$

se considerarán todas como **PE.**

Para nombrar una variable de cadena, se pueden utilizar las mismas combinaciones de caracteres, **pero el nombre debe llevar un signo $ al final.**

A$, P7$, MN$, Z0$, FP$

son todos ejemplos de variables de cadena válidas.

ASIGNANDO VALORES A VARIABLES

¿Cómo utilizamos estas variables? Introduzca el siguiente ejemplo:

```
A = 5                    Recuerde pulsar la tecla [ENTER] después
B = 2                    de cada línea
C = A + B
D = D + 3
PRINT A,B,C,D            (asegúrese de poner las comas).
```

Su pantalla mostrará

```
5          2
7          3
```

La primera línea significa, guardar el valor 5 en la variable llamada A, la siguiente línea almacena el valor 2 en la variable B. (El ordenador decide exactamente donde están estos lugares en su memoria, Vd. solo tiene que suministrar el nombre). La tercera línea indica busca los valores almacenados en las variables A y B, súmalos y pon el resultado en la variable llamada C. Después de que el ordenador haya hecho esto, las variables A y B siguen teniendo sus valores originales (5 y 2 en este caso) y C contiene la suma (7). La cuarta línea puede resultar un poco confusa para todos aquellos que sepan algebra. Esto es debido a que el signo igual (=) en el lenguaje BASIC no significa lo mismo que en matemáticas. **El = en BASIC significa asignar a, o toma la expresión a la derecha del signo igual (evalúala si es necesario), y pon su valor en la variable a la izquierda del signo igual.**

Este tipo de sentencia se llama **sentencia de asignación.** A la izquierda del signo igual, deberá haber siempre una variable. Algo como 2 = B + C, puede tener sentido en algebra, pero no lo tiene en lenguaje BASIC.

Volviendo a la sentencia D = D + 3, significa: toma el contenido actual de la variable D (que resulta ser Ø al no haber puesto nada), súmale 3, y pon el resultado en la variable D (o de forma equivalente — incrementa el valor

de D en 3). Esto puede parecer un poco confuso, pero es una sentencia muy útil (y muy común) en programas de ordenador. También demuestra otra característica de las variables—solo pueden tener un valor en cada momento. Si asigna un valor a una variable (es decir: aparece a la izquierda de una sentencia de asignación), este se escribe sobre el antiguo valor que tuviera, perdiéndose el anterior. Vd. puede, sin embargo, copiar el valor en otra variable tantas veces como quiera sin cambiarlo.

Si teclea ahora

```
A = B
B = 17
D = D + 2
PRINT A,B,C,D
```

recuerde pulsar [**ENTER**] después de cada línea

La respuesta será

```
2          17
7           5
```

La variable A tiene ahora una copia del valor que tenía B, en el ejemplo anterior, la variable B contiene un nuevo valor (17), y los valores previos de A y B se han perdido. La variable C permanece inalterable. La variable D tiene ahora 5, ya que contenía 3 del ejemplo anterior, y se le han agregado otros 2.

Las variables de cadena se comportan exactamente de la misma forma (excepto en que debe recordar que los nombres acaban en un signo $). Pruebe este ejemplo

```
A$ = "ESTA ES UNA"
B$ = "CADENA"
C$ = "MUY"
D$ = "LARGA"
E$ = A$ + B$ + C$ + C$ + C$ + C$ + C$
E$ = E$ + D$
PRINT D$
```

Su pantalla mostrará

```
ESTA ES UNA CADENA MUY MUY MUY
MUY MUY LARGA
```

En las líneas 1, 2, 3 y 4 hemos asignado valores a las variables de cadena A$, B$, C$ y D$. En la línea 5 agregamos 5 copias de C$ a A$ y B$. Con las variables de cadena el signo (+) no significa

lo mismo que con las variables numéricas. Significa agregar al final de la primera cadena. La operación se llama **concatenación.** En la línea 6 agregamos D$ al final de la recién construída E$. Esta es una forma de utilizar un ordenador para construir sentencias.

LOS NUMEROS Y LAS CADENAS NO CASAN

Recuerde mantener las variables numéricas y de cadena separadas, solo se pueden almacenar números en las variables numéricas y solo se pueden guardar cadenas en las variables de cadena; sentencias del tipo:

D = "CADENA"
A$ = 6
B = A$ * 2

provocarán el mensaje de error **?TM** ERROR (Desigual entre tipos).

El signo (+) es la única operación aritmética que se puede utilizar con cadenas y variables de cadena, todas las demás (—,*,/^): provocarán un mensaje de error.

RESUMEN DE COMANDOS

A lo largo de lo que resta de manual, aparecerán páginas enmarcadas. Darán detalles sobre cada comando a medida que los vayamos introduciendo. Al final de cada una de estas páginas, hay un programa ejemplo que demuestra el uso del comando. Estudiélos cuidadosamente, intente adivinar el resultado que producirán, antes de ejecutarlos. Estos programas están pensados para mostrar como funciona una sentencia en particular, pero frecuentemente, incluyen detalles útiles que deseará incluir en sus programas posteriormente.

NOMBRES DE VARIABLES DE CADENA

Una variable de cadena solo puede contener cadenas.

Un nombre de variable de cadena puede constar de cualquier combinación de letras y números, pero debe comenzar con una letra, y acabar con el signo $.

Como con los nombres de las variables numéricas el ordenador solo reconocerá los dos primeros caracteres de modo que,

RESPUESTA$, RES1$, RES2$

serán consideradas como la misma (RE$).

CAPITULO TRES
UN PROGRAMA

Hasta ahora su ordenador ha hecho poco más que provocar el eco de la línea que Vd. ha introducido. Empezaremos ahora a crear un **programa** de ordenador.

Un **programa** es un conjunto de instrucciones que dicen al ordenador lo que debe de hacer. Un programa BASIC consiste en un número de **líneas.** Una **línea** tiene dos partes: primero, **un número de línea** y segundo, una o más **sentencias.** Si hay más de una sentencia en una línea, cada sentencia deberá ir separada por dos puntos (:). Una **sentencia** es una línea de comandos como los que hemos venido utilizando hasta ahora.

PRINT A$: A = 47

He aquí un programa BASIC:

```
10 CLS
20 PRINT "CUAL ES SU NOMBRE? "
30 INPUT NOMBRE$
40 I = RND (255): J = RND (9) —1
50 CLS (J)
60 PRINT @ 200 + J, NOMBRE$;
70 SOUND I, 2
80 GOTO 40
```

Como puede ver, este programa contiene algunos comandos BASIC nuevos. No se preocupe de ellos por ahora, los veremos más tarde. Fíjese en la forma del programa BASIC—una secuencia de **líneas,** cada una consiste en un **número de línea** y al menos una **sentencia** (la línea 40 tiene 2).

INTRODUCIENDO UN PROGRAMA

Para introducir un programa en la memoria de un ordenador, primero deberemos limpiar la memoria de lo que pudiera tener. **Para hacer esto teclee NEW y después pulse** [**ENTER**]. Después introduzca cada línea como arriba, pulsando la tecla [**ENTER**] al final de cada línea. Observará que no ocurre nada después de pulsar la tecla [**ENTER**]. Una línea que comienza por un número no se ejecuta inmediatamente, tan solo se almacena. Cuando se ejecuta un programa, comienza por la línea que tiene el número menor, se ejecuta, y se busca la siguiente superior y asi sucesivamente. Debido a que la secuencia del programa sigue los números de línea, puede

introducir las líneas en el orden que quiera, el computador las ordenará en la secuencia correcta.

Trate de introducir el programa en el ordenador. Si comete un error antes de pulsar la tecla [**ENTER**], use la tecla con la flecha hacia la izquierda para retroceder. Si el error se hace evidente después de pulsar [**ENTER**], reescriba la línea. El ordenador solo almacena la última versión de una línea.

Después de haber introducido todo el programa, para ver como ha almacenado el ordenador las líneas que acaba de introducir, teclee **LIST** y pulse [**ENTER**]. **Fíjese en que no hay número delante de LIST.** Al no tener un número delante, el ordenador ejecutará inmediatamente el comando. Revise el programa para cerciorarse de que está bien (si no lo está, reescriba las líneas incorrectas).

Por fin podemos ejecutar un programa. Para hacerlo escriba **RUN** y pulse [**ENTER**].

La pantalla se limpiará y aparecerá un mensaje en la parte superior pidiéndole su nombre. Introduzca su nombre y pulse [**ENTER**].

El ordenador volverá a la acción.

La pantalla presentará diversos colores, y su nombre aparecerá saltando en mitad de la pantalla. Además de toda esta actividad se escucharán curiosos sonidos (si se acordó de subir el control de volumen de su televisor).

Asi continuará indefinidamente, a menos que lo detenga. Una forma de detener el programa es apagar el ordenador, pero no es una solución muy satisfactoria—perderá el programa. La mejor forma de detener este programa es pulsando la tecla BREAK.

PASO A PASO

Ya ha visto lo que hace un programa, le explicaremos ahora como lo hace, línea a línea.

10 CLS

Puesto que es la línea con menor número, será la primera en ejecutarse. El comando CLS significa limpiar la pantalla, y ajustar el fondo a su color normal (verde).

20 PRINT "CUAL ES SU NOMBRE"

Esta es la sentencia PRINT que vimos anteriormente. Esta línea imprime el mensaje en el extremo superior de la pantalla.

30 INPUT NOMBRE$

El comando INPUT indica al ordenador que se detenga y espere a que se le introduzca algo por el teclado, que será asignado a la variable que sigue al comando.

LIST

El comando LIST presenta en pantalla, el programa que se encuentra en memoria. No va precedido por un número de línea.

Si el programa es demasiado largo para entrar en la pantalla, se puede detener el listado pulsando la tecla [SHIFT] y la tecla [@] simultáneamente, (pero debe ser rápido). Se puede continuar el listado pulsando cualquier otra tecla.

Para listar solo una parte del programa puede usar

LIST N_1—N_2

donde N_1 y N_2 son dos números de línea (N_2 deberá ser mayor que N_1).

LIST 40—100

presentará las líneas de programa comprendidas entre las líneas 40 y 100.

LIST—80

mostrará todas las líneas de programa desde el comienzo hasta la línea número 80.

LIST 120—

mostrará todas las líneas de programa comprendidas entre la línea número 120 y el final de este.

RUN

El comando RUN se utiliza para comenzar a ejecutar un programa.

No lleva número delante.

Si desea comenzar un programa desde un punto distinto del principio, puede hacerlo escribiendo.

RUN número de línea

Donde "número de línea" es el punto donde desea que comience la ejecución.

RUN 25Ø

NEW

El comando NEW borrá la memoria, y ajusta todas las variables a cero.

Fíjese que no lleva número de línea.

Es una buena costumbre ejecutar NEW antes de introducir un programa, para asegurarse de que no quedan restos del programa anterior, que puedan interferir en el programa actual.

SENTENCIA DE ASIGNACION

La sentencia de asignación, se utiliza para meter un valor en una variable.

El formato de la sentencia de asignación es:

LET variable = expresión

La palabra LET de la sentencia forma parte del BASIC standard, pero no es necesaria en su ordenador, por ello no aparecerá en ninguno de los listados de programa en este manual.

La variable, puede ser cualquier nombre de variable.

La expresión, puede ser: una constante, otra variable o una combinación de ambas conectadas mediante operadores (+, —*, etc). Ya que no se pueden mezclar cadenas y variables numéricas, la variable y la expresión deberán ser del mismo tipo.

El signo igual (=), no tiene el mismo significado que se le da en algebra, es mejor interpretarlo como "asigna a". Esto quiere decir que la misma variable puede aparecer a ambos lados de la asignación, como en

```
40 X = X + 1
```

lo que indica al ordenador que sume 1 al valor actual de X y ponga el resultado en X.

```
10 S = 0: N = 0: CLS 5
20 PRINT @ 64, "INTRODUZCA UN NUMERO";
30 INPUT X: CLS 5
40 S = S + X: N = N + 1
50 PRINT @ 194, "NUMEROS INTRODUCIDOS :"; N;
60 PRINT @ 260, "LA MEDIA ES"; S/N;
70 GOTO 20
```

(Recuerde que el ordenador llamará a la variable, en este caso, NO$ ignorando el resto de las letras. De cualquier modo, a menudo es útil usar nombres de variables más largos para indicar mejor su contenido y como recordatorio).

40 I = RND (255): J = RND (9) —1

esta línea muestra como puede aparecer más de una sentencia en la misma línea. (Fíjese en los dos puntos que las separan). Esta línea, introduce además un nuevo comando, RND. Este comando genera un número aleatorio. Un número aleatorio es como extraer un número del interior de un sombrero. El número entre paréntesis después de RND, indica el intervalo de donde se extraerá el número. En la primera sentencia I = RND (255), RND (255) significa: dame un número aleatorio comprendido entre 1 y 255 y después introdúcelo en la variable que se llama I. En la segunda sentencia el rango para el número aleatorio va entre 1 y 9, después de seleccionar el número se le resta 1 y se introduce en J. Esto quiere decir que J estará comprendido entre 0 y 8.

50 CLS (J)

Esta línea limpia la pantalla. Esta vez sin embargo, el color de fondo dependerá del valor de J. Hay nueve colores posibles, numerados entre 0 y 8. Esta es la línea que provoca los distintos colores en la pantalla.

60 PRINT @ 200 + J, NOMBRE$;

Esta es una versión más sofisticada de la sentencia PRINT. Pide al ordenador que presente el valor de NOMBRE$ (en este caso su nombre), en una posición específica de la pantalla. La posición en este caso es 200 + J, que es un lugar entre 200 y 208. La posición 200 está en la línea 7 y en la 8 columna. (Vea el cuadro PRINT @ donde se muestra como está dividida la pantalla). Ya que el valor de J varía, su nombre aparecerá saltando por la línea.

70 SOUND 1,2

Esta es la línea que provoca los sonidos. El comando SOUND indica al ordenador que utilice su generador de tonos para producir un sonido; el sonido, y su duración vienen determinados por los dos números que siguen al comando.

80 GOTO 40

El comando GOTO significa simplemente: vete a la línea indicada (40 en este programa).

PRINT

El comando PRINT se utiliza para sacar algo por pantalla.

Se puede utilizar para mostrar constantes, el valor de una variable, cadenas y también evalúa expresiones.

Si se incluye más de un elemento en una sentencia PRINT, estos deberán ir separados por una coma (,) o un punto y coma (;).

Una coma provocará que se imprima la salida en dos columnas, cada quince caracteres. (Si la longitud del primer elemento es mayor de quince caracteres se imprimirá completo. El siguiente elemento aparecerá en la línea siguiente).

El punto y coma hace que se comprima la salida. Las cadenas se imprimirán una junto a otra, y los números tendrán un espacio a cada lado. El punto y coma mantiene la posición después de la última impresión, para continuar en la próxima sentencia PRINT.

Una sentencia PRINT sin elementos imprime una línea en blanco.

```
10 CLS
20 PRINT
30 PRINT"          UNA DEMOSTRACION"
40 PRINT"          DEL COMANDO PRINT"
50 PRINT
60 PRINT"COLUMNA UNO" , "COLUMNA DOS"
70 PRINT 14.2, 13.7
80 PRINT 1,2,7,11
90 PRINT
100 A$ = "COMPRIMIDA": B = 3
110 PRINT, "SEGUNDA LINEA"
130 PRINT A$; "SALIDA EN LA LINEA"; B
140 PRINT
150 PRINT "ESTO APARECERA";
160 PRINT "COMO UNA LINEA",
170 PRINT "DEMOSTRACION",
180 PRINT "TERMINADA"
```

INPUT

Cuando un programa llega a una sentencia INPUT, se detiene su ejecución, y espera a que se introduzca algo por el teclado. Después del comando INPUT debe haber uno o más nombres de variable separados por comas.

25 INPUT A,B,F$,H7

La sentencia anterior requiere que se introduzcan 4 elementos. Esto se puede realizar uno a uno pulsando la tecla [ENTER] después de cada uno, o como una lista separada por comas, es decir

146.2, 78.1, CADENA, 3 [ENTER]

Es siempre una buena idea presentar un mensaje antes de una sentencia INPUT, para recodarle lo que debe introducir. Esto se puede realizar con una sentencia PRINT, o incluirlo en una sentencia INPUT como a continuación:

35 INPUT "DOS NUMEROS POR FAVOR"; N1, N2

Fíjese en el punto y coma que separa la cadena de la lista de elementos.

Debe asegurarse de que introduce el tipo de dato correcto (cadenas para variables de cadena, números para variables numéricas), o de lo contrario el programa se detendrá dando el mensaje ?REDO, entonces tendrá que volver a introducir los datos.
No es necesario incluir las comillas en una cadena cuando se introduce, tanto "CADENA" como CADENA son válidas.

En el siguiente programa fíjese que:

a) Las líneas 1Ø0 y 17Ø. Utilizan el comando INPUT para detener el programa hasta que esté preparado. A$ no tendrá nada.

b) El uso de las variables de cadena (C$, P$ y T$) para evitar el tener que escribir el mismo mensaje varias veces.

```
1Ø CLS: T$ = "DEMOSTRACION DE ENTRADAS"
2Ø P$ = "PULSE [ENTER] PARA CONTINUAR".
3Ø C$ = "INTRODUZCA 4 NUMEROS",
4Ø PRINT: PRINT T$: PRINT
5Ø PRINT C$; "PULSANDO"
6Ø PRINT "[ENTER] DESPUES DE CADA UNO".
7Ø INPUT A,B,C,D: PRINT
8Ø PRINT: PRINT "HA INTRODUCIDO ESTOS VALORES"
```

```
90  PRINT A;B;C;D: PRINT
100 PRINT P$: INPUT A$: CLS
110 PRINT: PRINT T$: PRINT
120 PRINT "AHORA,  "; C$; "  SEPARADOS"
130 PRINT "POR COMAS Y PULSE LA TECLA [ENTER]".
140 INPUT A,B,C,D
150 PRINT: PRINT "ESTA VEZ LOS NUMEROS HAN SIDO:"
160 PRINT A;B;C;D
170 PRINT: PRINT P$: INPUT A$: CLS
180 PRINT: PRINT T$: PRINT
190 INPUT "ESCRIBA UNA CADENA Y UN NUMERO, PULSANDO [ENTER]
    DESPUES DE CADA UNO";B$, N
200 PRINT: PRINT "LA CADENA QUE HA INTRODUCIDO ES:"
210 PRINT B$
220 PRINT: PRINT "Y EL NUMERO ES:"; N
230 PRINT: PRINT "FIN DE LA DEMOSTRACION."
```

Asi pues, el programa retorna a la línea 4Ø donde selecciona un nuevo número aleatorio tanto para I, como para J. Ya que el color del fondo lo decide J, este cambiará cuando el programa llegue a la línea 5Ø. Como I también es distinto, el sonido cambia en la línea 7Ø. Cuando el programa llega de nuevo a la línea 8Ø vuelve a la línea 4Ø donde selecciona... (asi indefinidamente, o hasta que pulse la tecla BREAK).

HACIENDO CAMBIOS

Ese fue el primer programa, no hacía demasiadas cosas, pero es solo el comienzo. Continuemos introduciendo algunos nuevos comandos. Una explicación más detallada de los comandos se puede encontrar en las "cajas" a lo largo del texto.

A lo mejor no le gustó alguna cosa del programa anterior y cree que lo puede mejorar. Es fácil modificar un programa que ya está introducido. Ya que la secuencia en un programa BASIC, viene determinada por el número de línea, se puede cambiar una línea escribiendo simplemente otra línea con el mismo número que la del programa. La nueva línea sustituye a la antigua. Pruebe introducir una nueva línea 7Ø.

7Ø SOUND I,K

Se puede insertar una nueva línea dándole a esta un número que la sitúe en el lugar correcto.

Pruebe escribiendo

45 K = RND (2Ø)

Puesto que esta línea está numerada como 45, se situará entre las líneas 4Ø y 5Ø.

(Cuando escriba sus propios programas podrá numerar las líneas con cualquier número comprendido entre Ø y 63999. Es buena práctica numerar las líneas de 1Ø en 1Ø, es decir 1Ø, 2Ø, 3Ø..., de forma que tenga espacio suficiente en el caso de que tenga que introducir nuevas líneas).

Ejecute el programa modificado (escriba RUN), la nueva línea 45 selecciona otro número aleatorio, esta vez entre 1 y 2Ø. La línea 7Ø modificada, utiliza este número, (en la variable K), para cambiar la duración del sonido.

CONSTRUCCION DE PROGRAMAS

Aunque es fácil sentarse delante del teclado e introducir líneas de programa, las dificultades tienden a aparecer después, especialmente cuando los programas son un poco largos. La introducción del programa debe de ser la última fase de la creación de un nuevo programa.

RND

El comando RND genera números aleatorios.

RND es una función. Una función, en BASIC, es algo que toma uno o más números y realiza una operación sobre ellos devolviendo un valor simple. Los números utilizados por la función se llaman argumentos y se ponen siempre entre paréntesis después del nombre de la función. El resultado de una función se devuelve al programa.

La función RND devuelve un número aleatorio, que depende del valor del argumento de la función.

Si el valor del argumento es Ø (RND(Ø)) la función devuelve un valor comprendido entre Ø y 1.

Si el valor del argumento es mayor de Ø (RND(6)) la función devuelve un valor entero comprendido entre 1 y el valor del argumento. (RND(6) devolverá uno de los valores: 1, ó 2, ó 3, ó 4, ó 5, ó 6, no sabemos cuál ya que la selección es aleatoria!).

```
1Ø CLS
2Ø PRINT @ Ø, "INTRODUZCA UN NUMERO";: INPUT N
3Ø CLS: PRINT @ 96, "TRES NUMEROS ALEATORIOS PARA N = "; N
4Ø PRINT @ 27Ø, RND(N)
5Ø PRINT @ 3Ø2, RND(N)
6Ø PRINT @ 334, RND(N)
7Ø GOTO 2Ø
```

CLS

El comando CLS se utiliza para limpiar la pantalla, y ajustar el color del fondo. El color normal del fondo es el verde; si utiliza CLS a secas, esta será el color utilizado.

Para cambiar el color del fondo agregue un número entre Ø y 8 después del comando CLS, (CLS 2).

Los colores disponibles son:

Ø — Negro	1 — Verde	2 — Amarillo
3 — Azul	4 — Rojo	5 — Beige
6 — Turquesa	7 — Magenta	8 — Naranja

El matiz de estos colores dependerá de su televisor.

Notará de todos modos, que independientemente del color del fondo, el ordenador presentará el texto en negro o verde.

```
1Ø CLS
2Ø PRINT @ Ø, "DEMOSTRACION DE COLORES DE FONDO";
3Ø PRINT @ 192, "INTRODUZCA UN NUMERO ENTRE Ø Y 8";
4Ø INPUT C
5Ø CLS (C)
6Ø PRINT @ 288, "ESTE ES EL COLOR DE FONDO:"; C
7Ø GOTO 2Ø
```

Comience con un lápiz y papel, y escriba lo que intenta hacer. Divida el problema en distintas secciones. La mayoría de los problemas a resolver con un ordenador, pueden dividirse fácilmente en tres partes.

1) Preparación, títulos, instrucciones y entrada de datos.
2) Cálculos
3) Presentación de resultados.

Ahora tome cada sección por separado y divídala en subsecciones, hasta que le queden acciones simples (ejemplo sumar 1 a un contador). Ordene estas acciones en secuencia, escríbalas y comience con otra sección. Cuando haya terminado, tendrá una serie de pasos (algunos muy sencillos) dentro de cada sección. Cualquiera que lea el resultado deberá ser capaz de resolver el problema utilizando aritmética simple. Lo único que le queda ahora por hacer es transformar su plan en un lenguaje que el ordenador pueda entender. Idealmente, cada paso en su plan, deberá convertirse en una línea de programa en BASIC.

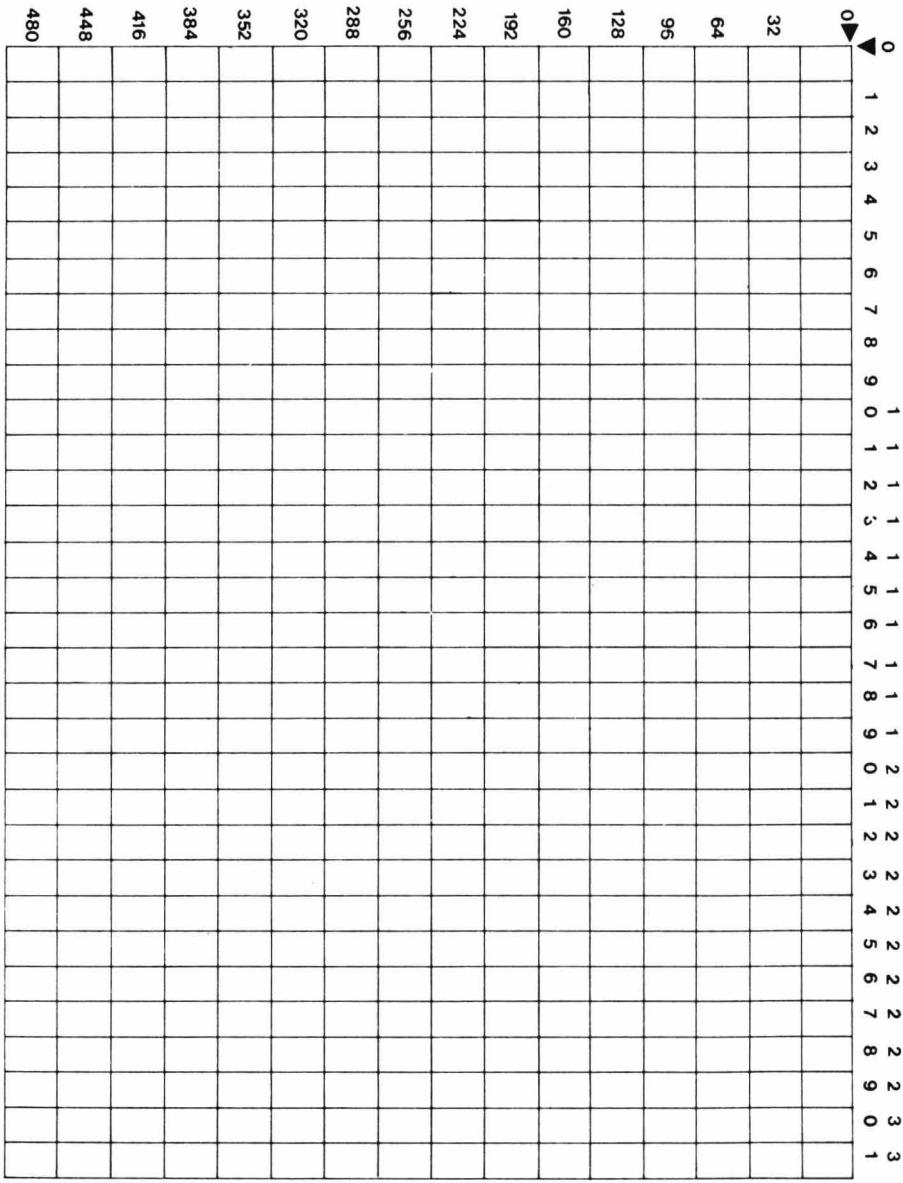
Después de la traducción y antes de montar el programa completo, deberá chequear cada sección separadamente y asegurarse de que hace lo que debe. Cuando el programa funcione satisfactoriamente, no lo celebre haciendo una hoguera con todos los papeles que utilizó. Guarde el plan final, puede que haya partes que le sean útiles en otros programas; además puede que aparezcan errores bastante después de haber acabado el programa. También resulta útil introducir algunos comentarios en el propio programa para recordar lo que hace. Para realizar esto, el BASIC posee la sentencia REM. Esta sentencia no hace nada. El BASIC ignora todo lo que venga después de una sentencia REM dentro de la misma línea, por lo tanto, asegúrese de no poner ninguna sentencia REM antes de otras sentencias en la misma línea. Hay otra versión más abreviada para la sentencia REM, que utiliza la comilla simple ('), (se obtiene con [SHIFT] y [7]).

```
10 REM PROGRAMA PARA CALCULAR LA MEDIA
140 A = B * C: GOTO 15: REM VOLVER AL COMIENZO
48 K = K + 1:' INCREMENTAR EL CONTADOR
```

Aunque las sentencias REM roban algo de memoria, no es aconsejable eliminarlas completamente. Tratar de entender sus propios programas meses después de haberlos escrito puede resultar una experiencia frustrante si no introdujo comentarios en el.

Toda esta sección puede parecer innecesaria, pero es un hecho bien conocido en círculos informáticos, que se pierde más tiempo depurando programas (localizando errores en un programa) que escribiéndolo. Si trabaja como le sugerimos, hay menos probabilidades de que introduzca errores en sus programas, y si los tiene, serán más fáciles de detectar.

Retícula Print @



PRINT @

El comando PRINT @ se utiliza para dirigir la salida a un punto determinado de la pantalla.

Con este propósito, la pantalla está dividida en una tabla de 16×32 casillas, lo que proporciona 512 posiciones. Vea el diagrama de la otra página para ver como se numera la tabla.

El formato del comando PRINT @ es

PRINT @ expresión, lista de elementos

La expresión puede ser un número, una variable o una expresión aritmética, mientras el valor se mantenga entre Ø y 511.

La lista de elementos es la misma que la utilizada en el comando PRINT, pueden ser números, variables, cadenas o expresiones, separadas por comas o punto y comas.

Si considera que su pantalla tiene 16 líneas, entonces la sentencia

PRINT @ 32 * (LINEA—1), A

imprimirá el valor de A al comienzo de la LINEA imaginaria de su pantalla. Cada línea dependerá del valor de la variable LINEA, (un número entre 1 y 16).

El siguiente ejemplo es una forma deliberadamente confusa de producir un resultado bastante sencillo. Trate de averiguar lo que hace, ejecute el programa, y después escriba una versión más directa de producir el mismo resultado.

Fíjese en la utilización del punto y coma al final de las sentencias, para evitar que se borre el resto de la línea. (Si no se lo cree, trate de introducir la línea 11Ø sin el punto y coma final).

```
1Ø CLS: P$ = "PRINT @": N = 1
2Ø FILA = 12: A$ = "LA PANTALLA"
3Ø PRINT @ 26 * (FILA—1) + 15, "UTILIZANDO"; P$

4Ø PRINT @ 230 + 3, "EN CUALQUIER PUNTO DE";
5Ø PRINT @ 16Ø, "ESTE"; : PRINT @ 174, "MUESTRA";
6Ø PRINT @ 16 * (FILA—1) + 11, "PUEDE"; : PRINT @ 12, "PAGINA";
7Ø PRINT @ 263 + 25, A$;
8Ø PRINT @ 182, "COMO  ";
```

```
90  PRINT @ 31 * (FILA—5) + 7, "ESCRIBIR";
100 PRINT @ 68, P$; "  DEMOSTRACION";
110 PRINT @ 165, "PROGRAMA";
120 PRINT @ 18, N
130 GOTO 130
```

La última línea (130) pone al computador en un bucle indefinido, que no hace nada. Esto impide que aparezca el mensaje OK que sale cuando termina el programa. Pulse la tecla BREAK para detener el programa.

SOUND

El comando SOUND genera un sonido con un tono y una duración determinados. Requiere dos argumentos:

5Ø SOUND P,D

P es un número entre 1 y 255. El tono más bajo es 1, el más alto 255. El Do medio del piano es P = 89.

D puede ser cualquier número entre 1 y 255. D = 16 da un tono durante aproximadamente 1 segundo.

```
1Ø CLS
2Ø PRINT @ 6, "DEMOSTRACION DE SONIDO"
3Ø PRINT @ 64, "INTRODUZCA UN NUMERO COMPRENDIDO ENTRE
   1 Y 255,";
4Ø PRINT @ 111, "PARA ESPECIFICAR EL TONO SONORO": INPUT P
5Ø PRINT @ 192, "DE UN NUMERO PARA LA DURACION DE LA NOTA";
6Ø INPUT D: CLS (RND (9)  —  1)
7Ø SOUND P, D
8Ø GOTO 1Ø
```

GOTO

El comando GOTO tiene el formato

GOTO número de línea

El número de línea debe de ser un número (no una variable), y debe existir en algún lugar del programa. Si no se encuentra el número de línea se detendrá el programa con un mensaje ?UL ERROR (línea indefinida).

La sentencia GOTO se ejecuta inmediatamente, por ello, no se deberá poner ninguna otra sentencia en la misma línea después de GOTO, el programa nunca pasará por ellas.

```
2Ø CLS
3Ø GOTO 6Ø
4Ø PRINT "EN LA LINEA 4Ø"
5Ø GOTO 8Ø
6Ø PRINT "EN LA LINEA 6Ø"
7Ø GOTO 4Ø
8Ø PRINT "FIN DEL PROGRAMA"
```

UN EJEMPLO DE PROGRAMACION

PROBLEMA:— Usar el ordenador para simular el lanzamiento de dos dados:

SECCION 1—Presentar título e instrucciones.

a) Limpiar la pantalla
b) Presentar el título
c) Presentar instrucciones
d) Presentar las cabeceras: dado 1, dado 2.

SECCION 2—Encontrar los valores para ambos dados.

(Cuando se tira un dado, puede salir cualquiera de las seis caras aleatoriamente)

a) El primer dado es un número aleatorio entre 1 y 6.
b) El segundo dado es un número aleatorio entre 1 y 6.

SECCION 3—Presentar los resultados

a) Presentar los valores del dado 1 y dado 2.

SECCION 4—Detener el programa y repetir si es necesario.

a) Detener el programa
b) Preguntar si quiere repetir
c) Si es necesario, repetir secciones 1, 2 y 3.

Traducido a BASIC (con algunos comentarios), resulta:

```
10 'SIMULACION DEL LANZAMIENTO DE DADOS'
20 '
30 'SECCION PRIMERA
40 CLS: REM LIMPIAR LA PANTALLA
50 PRINT "SIMULACION DE DADOS": 'TITULO
60 PRINT
70 PRINT "UTILICE LA TECLA [BREAK] PARA ACABAR LA EJECUCION
   DEL PROGRAMA": 'INSTRUCCIONES
80 PRINT
90 PRINT "DADO 1", "DADO 2": 'CABECERAS
100 PRINT
110 REM FIN DE LA PRIMERA SECCION
```

REM

El comando REM se utiliza para introducir comentarios en un programa. El ordenador ignora cualquier cosa que aparezca en la misma línea después de REM (o su abreviatura').

```
10 REM ESTA ES UNA LINEA DE COMENTARIO
35 D = B * B—4 * A * C: 'CALCULO DEL DISCRIMINANTE
```

```
12Ø REM
13Ø REM SECCION SEGUNDA
14Ø D1 = RND (6): 'LANZAMIENTO DEL PRIMER DADO
15Ø D2 = RND (6): 'LANZAMIENTO DEL SEGUNDO DADO
16Ø 'FIN DE LA SECCION SEGUNDA
17Ø '
18Ø 'SECCION TERCERA
19Ø PRINT @ 229, D1, D2: 'PRESENTAR EL RESULTADO
2ØØ 'FIN DE LA TERCERA SECCION
21Ø '
22Ø 'SECCION CUARTA
23Ø INPUT "PULSE [ENTER] PARA LANZAR LOS DADOS"; A$
24Ø GOTO 4Ø: 'VOLVER AL COMIENZO
25Ø 'FIN DE LA SECCION CUARTA
```

Este programa no necesita todos los comentarios, escriba su propia versión del programa.

No se puede decir que haya una versión 'correcta' de un programa, una versión correcta es una que funciona. Hay formas más elegantes o eficientes para resolver el mismo problema. Generalmente, un programa más corto es más rápido y utiliza menos memoria.

Terminamos esta sección con otra versión del mismo programa, fíjese en como al dividir el problema en secciones, nos permite a veces montar el programa en distinto orden pero con resultados semejantes.

```
1Ø 'SIMULACION DEL LANZAMIENTO DE DADOS
2Ø '
3Ø 'SECCION PRIMERA
4Ø CLSØ: PRINT @ 8, "SIMULACION DE DADOS";
5Ø PRINT @ 167, "PRIMER ";: PRINT @ 178, "SEGUNDO";
6Ø PRINT @ 2ØØ, "DADO";: PRINT @ 211, "DADO";
7Ø PRINT @ 452, "UTILICE BREAK PARA FINALIZAR";
8Ø 'SECCION CUARTA
9Ø PRINT @ 358, "PULSE ENTER PARA TIRAR";
1ØØ INPUT A$
11Ø 'SECCION SEGUNDA
12Ø D1 = RND (6): D2 = RND (6)
13Ø 'SECCION TERCERA
14Ø PRINT @ 265, D1;: PRINT @ 276, D2;
15Ø GOTO 9Ø
```

CAPITULO CUARTO

UN BUEN ALMACENAMIENTO

CONECTANDO EL CASETE

Algunos de los programas están empezando a ser un poco largos, y es una lata tener que reescribirlos cada vez que los queremos ejecutar. Para evitar esto, resulta muy sencillo almacenar los programas en cinta y cargarlos en memoria cuando los necesitemos. Para hacer esto, necesitará un casete que permita grabar y un cable de conexión. El cable de conexión se suministra con su ordenador y tiene una toma DIN (de 5 pines) en un extremo, y tres tomas jack en el otro.

Se puede utilizar cualquier casete que tenga una calidad razonable, suponiendo que permita

a) Grabar de una fuente exterior (una toma para jack marcada habitualmente con AUX o LINE IN)
b) reproducir a través de un altavoz exterior o auriculares (una toma para jack marcada con EAR, o MONIT, o L/S o SPKR)
c) la parada y continuación por control remoto (una toma para jack marcada normalmente con REM y próxima a la toma de micrófono.
d) funcionar conectado a la red. Esto no es necesario, pero unas pilas descargadas pueden afectar seriamente el éxito en la grabación y recuperación de programas.

Para conectar el casete al ordenador, conecte el enchufe DIN del cable en la toma del ordenador que pone TAPE.

Los tres conectores del otro extremo del cable, se conectan en el casete como sigue:

1) El jack más pequeño va en la toma marcada REM.
2) El jack grande con el cable gris, va en la toma marcada AUX.
3) El otro jack grande con el cable negro va en la toma marcada EAR.

Encienda el casete, introduzca una cinta y rebobínela hasta el comienzo. Ajuste el control de volumen a 6 (aproximadamente a mitad de recorrido). Ya está listo para grabar sus programas.

GRABANDO UN PROGRAMA EN CINTA

Escriba un programa y ejecútelo para asegurarse de que funciona correctamente. Después proceda como sigue:

1) Pulse los botones **PLAY** y **RECORD** simultáneamente.
2) Teclee el comando

CSAVE "PROGRAM1" pulse [ENTER]

Puede sustituir el nombre "PROGRAM1" por cualquiera que desee, (deberá comenzar por una letra y no tener más de 8 caracteres). Cuando pulse [ENTER], el motor del casete arrancará y se grabará el programa. Después de un rato, volverá a aparecer el mensaje OK, y se detendrá el motor del casete.

El programa estará todavía en la memoria del ordenador, solo se ha copiado en la cinta. Ha grabado un programa llamado PROGRAM1, (o como quiera que lo llame) en la cinta.

CARGANDO PROGRAMAS EN MEMORIA

Para cargar en memoria un programa previamente grabado, primero teclee NEW, para borrar cualquier programa que existiera en memoria . El proceso es como sigue:

1) Rebobine la cinta hasta el principio.
2) Pulse el botón **PLAY.**
3) Teclee el comando.

CLOAD "PROGRAM1" PULSE [ENTER]

El motor del casete arrancará y aparecerá una letra S en la esquina superior izquierda de la pantalla. Esto indica que el ordenador está buscando el programa. Cuando lo encuentre la S se convertirá en

F PROGRAM 1

Cuando aparezca el mensaje OK, y se detenga el motor del casete, el programa ya estará cargado en la memoria del ordenador. Para asegurarse de que asi es, teclee **LIST.**

Si no está el programa, o aparece **I/O ERROR** en la pantalla, puede que su casete necesite otro ajuste. Asegúrese de que las conexiones están bien, después repita las secuencias de grabación y carga con distinto ajuste en el control de volumen, hasta que obtenga un resultado satisfactorio.

CSAVE
CLOAD
SKIPF

El comando CSAVE graba un programa en una cinta de casete. El nombre del programa puede tener hasta 8 caracteres.

CSAVE "PROGRAMA"

Para grabar datos en el casete utilice el parámetro opcional A, la información entonces se grabará en formato ASCII. Después se podrá leer estos datos mediante el comando INPUT# —1.

CSAVE "DATOS", A

El comando CLOAD carga un programa determinado en memoria desde el cassete.

CLOAD "PROGRAMA"

El comando SKIPF salta hasta el siguiente fichero de programa después del programa especificado.

SKIPF "PROGRAMA"

GRABANDO MAS DE UN PROGRAMA

Para grabar más de un programa en una cinta, deseará evitar el grabar encima de los programas ya existentes. Esto requiere el posicionar la cinta después del último programa. Esto se hace como sigue:

1) Rebobine la cinta.
2) Pulse **PLAY**
3) Teclee el comando:

SKIPF "PROGRAM1"

El motor arrancará, el ordenador buscará (S) el programa, lo encontrará (F) y se lo saltará, después se detendrá el motor y aparecerá el mensaje OK.

4) Pulse el botón **STOP**
5) Pulse simultáneamente **PLAY** y **RECORD,** dé nombre al nuevo programa y después grábelo (CSAVE).

DETALLES PARA GRABACIONES FIABLES

1) Trabajar con el casete enchufado a la red para obtener velocidad constante.
2) Utilice casetes nuevos de alta calidad. Aunque parezcan más convenientes los casetes de larga duración (C120), es mejor utilizarlos de (C30 ó C12).
3) Empiece siempre la búsqueda al comienzo de la cinta, no se fíe del contador del casete.
4) No deje pulsados los botones PLAY y RECORD, pulse STOP después de terminar de grabar o cargar.
5) Rebobine los casetes antes de retirarlos.
6) Ponga etiquetas en los casetes, nada más grabar. Para programas importantes, proteja el casete contra escritura.

Aunque sea muy cuidadoso, pueden ocurrir accidentes! Algunos programas pueden requerir una gran inversión de esfuerzo y tiempo —asi que haga otra copia en otro casete.

EL EDITOR

A medida que crezcan las líneas de programa, hay más posibilidades de cometer errores. Hasta ahora, la única solución era repetir la línea completa. Esto no es necesario, ya que vamos a introducir el EDITOR.

El EDITOR le permite moverse dentro de una l ínea hacia delante y hacia atrás, cambiando o insertando caracteres . Para llamar al EDITOR teclee,

EDIT número de línea [ENTER]

donde "número de línea" es el número de línea del programa que queremos modificar. La línea se presentará totalmente en pantalla. Entonces aparecerá el número de la línea con el cursor parpadeando a su lado.

MOVIENDONOS SOBRE LA LINEA

El cursor se encuentra ahora al comienzo de la línea. **Para moverse hacia delante sobre la línea, pulse la barra espaciadora.** El cursor se moverá presentando los caracteres sobre los que ha pasado. **Para retroceder podemos utilizar la tecla de retroceso** [←]. Puede indicar el incremento del movimiento tecleando un número seguido de la tecla, (por ejemplo, [5] [SPACE] moverá el cursor 5 posiciones hacia delante, y [3] [←] lo moverá 3 hacia atrás). Utilizando estas dos teclas podrá posicionar el cursor en cualquier lugar de la línea. No verá el carácter ya que el cursor estará parpadeando sobre el. Otros dos comandos le permitirán saltar directamente a una posición determinada de la línea. **Para agregar caracteres al final de la línea, pulse [X], de extensión.** El cursor saltará al final de la línea, y solo tendrá que teclear los caracteres que desee agregar. Utilizando la facilidad de búsqueda puede posicionarse sobre el carácter buscado. **Pulse [S] seguido del carácter sobre el que desea situarse.** ([S] [A] moverá el cursor a la primera ocurrencia de la letra A en la línea). Si hay más de una A en la línea, y quiere colocarse sobre la tercera A, entonces utilice [3] [S] [A].

TODOS LOS CAMBIOS

Una vez posicionado el cursor mediante la barra espaciadora, retroceso o tecla [S], puede:

a) **Borrar un carácter pulsando [D].** Esto borrará el carácter que se encuentra debajo del cursor. Para borrar más de un carácter indique el número; [5] [D] borrará los cinco caracteres siguientes a partir de la posición del cursor.

b) **Cambiar un carácter pulsando [C],** seguido por el nuevo carácter. ([C] [F] sustituirá el carácter que se encuentra debajo del cursor por F). [3] [C] cambiará los tres caracteres siguientes por los tres introducidos a continuación.

c) **Insertar caracteres pulsando [I]** seguido por los caracteres que desee insertar. Una vez pulsada [I], el editor pasa al modo

EDIT

El comando EDIT se usa para modificar el contenido de una línea determinada.

EDIT número de línea

Una vez en el modo de edición se puede utilizar cualquiera de los siguientes comandos.

L	Lista el contenido actual de la línea.
C carácter	Cambia el carácter actual.
n C carácter	Cambia los siguientes n caracteres por los nuevos.
I	Inserta caracteres.
D	Borra el carácter actual.
n D	Borra los próximos n caracteres.
H	Borra el resto de la línea, desde la posición actual.
X	Extiende la línea, se posiciona al final y se sitúa en modo de inserción.
S carácter	Busca la primera aparición del carácter.
n S carácter	Busca la n — sima aparición del carácter.
K	Borra el resto de la línea desde el cursor.
n K carácter	Borra la línea hasta la n — sima aparición del carácter.
n [**ESPACIO**]	Avanza el cursor n espacios. Si se omite n, avanza 1.
n [←]	Retrocede el cursor n posiciones. Si se omite n se retrocede 1.
[**SHIFT**] [^]	Deja el modo de inserción y vuelve al modo de edición.
[**ENTER**]	Deja el editor y almacena la línea.

de inserción. En esta modalidad, todo lo que se teclee se introduce en la línea. Para dejar el modo de inserción y volver al modo normal del editor, deberá pulsar [SHIFT] y [^] simultáneamente.

d) **Listar el estado actual de la línea tecleando [L].** Se mostrará la línea con todos los cambios realizados hasta el momento. Después de presentar la línea, el cursor volverá al comienzo.

e) **Dejar el editor pulsando la tecla [ENTER]. Esto volverá a situar la línea alterada en la memoria del programa y presentará el mensaje OK. Puede dejar el editor en cualquier momento pulsando la tecla [ENTER].**

He aquí un ejemplo de edición. Al principio, puede parecer un poco complicado, pero se sorprenderá de lo rápido que puede mejorar con muy poca práctica. Las teclas que pulsa están entre corchetes.

[**E**] [**D**] [**I**] [**T**] [**1**] [Ø] [**ENTER**]

10 PRINT "THE IR ARE MANY MISTOOK IN LINE

10 ■	**(■ indica la posición del cursor)**
[1] [Ø] [SPACE]	**moverse 10 espacios hacia delante**
[C] [R] [C] [E]	**cambiar I por R y R por E**
[S] [M]	**situarse al comienzo de MANY**
[2] [D]	**borrar M y A**
[ESPACE] [C] [O]	**saltar sobre N y cambiar Y por O**
[S] [O]	**situarse en la primera O de Mistook**
[D] [C] [A]	**borra la primera O y sustituye la segunda por A**
[SPACE] [I] [E] [S]	**se salta K e inserta ES**
[SHIFT] [^]	**deja el modo de inserción**
[S] [L]	**se sitúa en la L de LINE**
[I] [T] [H] [I] [S] [SPACE]	**inserta THIS y un espacio antes de LINE**
[SHIFT] [^]	**deja el modo de inserción**
[X]	**se sitúa al final de la línea (ahora está en el** modo de inserción).
[**SPACE**] [**N**] [**O**] [**W**] ["]	agrega NOW y comillas para terminar la cadena
[**SHIFT**] [^]	deja la modalidad de inserción
[**L**]	lista el estado actual de la línea

10 PRINT "THERE ARE NO MISTAKES IN THIS LINE NOW"

10 ■

[**ENTER**]	almacena la línea y deja el EDITOR

Hay otros dos comandos de edición, que deben ser utilizados con cuidado:

a) **El carácter kill. Pulsando [K] se borrará lo que haya a partir de la posición del cursor.** [K] seguido de un carácter, borrará todo lo que haya desde la posición del cursor, hasta la primera aparición de ese carácter. [3] [K] [A] borrará hasta la tercera aparición de A.

b) **El carácter hack. Pulsando [H] se borrará el resto de la línea desde la posición del cursor y después se situará en el modo de edición.** Es útil cuando se quiere reescribir el final de una línea.

Si no comete nunca errores al escribir los programas. Puede ignorar esta sección. Para el resto, el EDITOR nos facilitará mucho el trabajo de ahora en adelante. Introduzca uno de los ejemplos propuestos, grábelo en una cinta si quiere, después conviértalo en otro programa mediante el EDITOR.

MAS COMANDOS DEL SISTEMA

Comandos como LIST y RUN, son comandos del sistema. No son parte del programa sino comandos directos que indican al ordenador que haga algo inmediatamente. He aquí algunos otros que facilitan el trabajo de programar.

Borrando líneas de programa

Para eliminar una línea de programa, hemos estado escribiendo el número de línea seguido de [ENTER]. Esto va bien para una o dos líneas, pero, y si son 3Ø ó 4Ø?

DEL número de línea—número de línea

borrará un bloque completo de líneas, desde la primera línea indicada hasta la segunda inclusive.

DEL 1ØØ—25Ø

eliminará todas las líneas desde la 1ØØ a la 25Ø inclusive, del programa que se encuentra actualmente en memoria.

El comando DEL se puede utilizar también de otras formas,

DEL 2Ø	borrará solo la línea 2Ø
DEL 3Ø—	borrará todas las líneas desde la 3Ø hasta el final del programa.
DEL—2ØØ	**borrará todas las líneas desde el comienzo del programa hasta la línea 2ØØ inclusive**
DEL—	**borrará todo el programa.**

DEL

El comando DEL se utiliza para borrar líneas determinadas del programa que está actualmente en memoria.

DEL número de línea 1 — número de línea 2

El comando borrará todas las líneas desde la lineal hasta la línea 2 inclusive. Ambos números de línea son opcionales y el comando se puede utilizar en cualquiera de las siguientes formas:

DEL —	**Borra el programa completo**
DEL —100	**Borra desde el principio hasta la línea 100**
DEL 300 —	Borra desde la línea 300 hasta el final
DEL 40	Borra la línea 40 exclusivamente
DEL 100—200	Borra todas las líneas entre la 100 y 200.

RENUM

El comando RENUM permite cambiar parte o todos los números de línea del programa. El comando RENUM también cambia los números de línea en sentencias de bifurcación (GOTO, etc) para asegurar que el flujo del programa discurre de la misma forma.

RENUM nueva línea, línea de comienzo, incremento

El comando renumerará todas las líneas desde la línea de comienzo, empezando por el número "nueva línea" y el incremento indicado. Todos los parámetros son opcionales y la sentencia se puede usar en cualquiera de las siguientes formas:

RENUM	Renumera todo el programa. Las líneas se numerarán 10, 20, 30...
RENUM 100	Renumera todo el programa. Las líneas se numerarán 100, 110, 120...
RENUM 100, 50, 5	Renumera comenzando en la antigua línea 50. Las líneas se numerarán 100, 105, 110...
RENUM,,20	Renumera el programa completo. Las líneas se numerarán 10, 30, 50...

Fíjese en que si se omite un parámetro y el siguiente debe ser utilizado, deberá de escribirse la coma. No se puede utilizar RENUM para alterar la secuencia de las líneas.

RENUMerando líneas de programa

El comando RENUM renumerará todas, o algunas de las líneas del programa. También cambiará los números de línea en las sentencias GOTO, GOSUB, IF THEN, ON GOTO y ON GOSUB para asegurar que siguen bifurcando al mismo lugar, más tarde veremos las sentencias anteriores.

RENUM nueva línea, línea de comienzo, incremento

"Nueva línea" es el nuevo número de línea para la primera línea a renumerar. La "línea de comienzo" es la línea desde la que se desea comenzar a numerar, e "incremento" es el incremento utilizado entre cada nuevo número de línea. Se puede omitir cualquier o todos los parámetros. Si omite "nuevalínea" se supondrá 1Ø. Si omite la "línea de comienzo" provocará que se renumere todo el programa, y si se omite "incremento", hará que las líneas se numeren de 1Ø en 1Ø.

RENUM	Numera todo el programa como 1Ø, 2Ø, 3Ø...
RENUM 1ØØ, 5Ø, 5	Numerará las líneas desde 5Ø como 1ØØ, 1Ø5, 11Ø... Las líneas anteriores a la 5Ø permanecerán igual.
RENUM 11Ø,,2	Renumerará todo el programa como 11Ø, 112, 114...
RENUM,,5	Renumerará todo el programa como 1Ø, 15, 20...

Fíjese que si omite un parámetro, pero desea utilizar otro que viene después, deberá de incluir la coma.

Siguiendo la pista al programa

A veces, cuando tiene dificultades con un programa, es útil saber donde va a parar. Las facilidades de "trace" nos permiten hacerlo. Escribiendo **TRON** antes de ejecutar un programa, activa la función de seguimiento (trace). Ahora aparecerán en la pantalla los números de línea del programa a medida que se van ejecutando. Esto le permite ver si el programa pasa por las líneas adecuadas. Para desactivar esta facilidad, escriba **TROFF.**

Deteniéndose y comenzando

Se puede detener la ejecución de un programa, incluyendo una línea de programa con la sentencia STOP.

185 STOP

Esta línea provocará la detención del programa cuando pase por la línea 185. Aparecerá un mensaje en la pantalla indicándole que el

programa se ha detenido y en que línea ha ocurrido. Podrá ahora mirar el contenido de cualquier variable utilizando PRINT o ? Para continuar la ejecución del programa escriba **CONT** y el programa continuará su ejecución a partir de STOP.

Ya puede renumerar programas, borrar líneas no deseadas, cambiar el contenido de cualquier línea, y grabar el resultado en un casete. Como ya nos encontramos en un punto en el que podemos conservar los programas, empezaremos a hacer alguno más interesante.

TRACE

Esta función permite seguir el flujo de un programa. A medida que se alcanza una línea se imprime su número en pantalla. La función trace se debe activar antes de ejecutar un programa.

TRON activa el seguimiento (trace)
TROFF desactiva el seguimiento.

Ambos son comandos directos que no necesitan número de línea.

END
STOP
CONT

El comando END termina la ejecución de un programa y devuelve el control al teclado.

El comando STOP detiene la ejecución de un programa en la línea que contiene dicha sentencia. Aparece en la pantalla el mensaje BREAK AT N para indicar que la ejecución se ha detenido en la línea N. Para continuar la ejecución del programa utilice CONT, sin indicar número de línea. La ejecución del programa continuará a partir de la línea después de STOP. No podrá continuarse un programa después de la sentencia END, deberá ejecutarlo de nuevo.

CAPITULO CINCO
POSICIONANDOSE

Al final del capítulo 3, analizamos como construir un programa considerándolo como un conjunto de secciones separadas. A continuación, veremos como dirigir la ejecución del programa para unir estas secciones. Esto se llama bifurcar. Ya vimos la sentencia de bifurcación GOTO. GOTO provoca un salto (bifurcación) incondicional, ya que en cuanto el programa alcanza la sentencia GOTO salta a la línea especificada y continua la ejecución a partir de alli. El programa en este caso no tiene opción, GOTO significa ir de inmediato. Hasta ahora hemos utilizado GOTO, principalmente, para volver al comienzo del programa. Aunque es de gran utilidad el repetir una y otra vez una parte de un programa, no es corriente que queramos que se repita indefinidamente. (Tampoco es una buena costumbre el tener que utilizar la tecla BREAK para detener un programa). Afortunadamente, el lenguaje BASIC nos provee con un número de sentencias que nos permiten controlar el flujo del programa.

OPERACIONES DE SELECCION

La primera de estas sentencias es una extensión de nuestra vieja amiga GOTO. Es la sentencia ON... GOTO que tiene el siguiente formato:

ON expresión numérica GOTO lista de números de línea

Se evalúa la expresión numérica, y si es necesario, se trunca a un valor entero (es decir, se rechazan los decimales). Después se transfiere el control a uno de los números de línea de la lista. Si la expresión vale 1, irá al primer número de línea, si 2, al segundo, y asi sucesivamente. Si el valor de la expresión es menor que 1, o mayor que el número de líneas en la lista, el ordenador ignorará por completo la sentencia y continuará con la siguiente. Un valor negativo de la expresión, sin embargo, provocará la detención del programa con un mensaje de error. Normalmente es una buena idea comprobar que el valor está dentro del rango pretendido, antes de ejecutar la sentencia ON... GOTO. He aquí algunos ejemplos de la sentencia.

140 ON P GOTO 200, 30, 40

210 ON X—4 GOTO 20, 40, 70, 10, 690

185 ON B * C/D—E GOTO 115, 285, 900, 40

La sentencia ON... GOTO es un modo útil de seleccionar de entre un número de opciones y se puede considerar como un salto (bifurcación) condicional. Lo utilizaremos en este sentido en el ejemplo de la próxima sección.

ON... GOTO

El comando ON GOTO provee un método de ramificación múltiple a los números de línea especificados.

ON expresión GOTO lista de números de línea

Se evalúa la expresión (y se trunca si no da como resultado un entero). El programa después continua ejecutándose a partir del número de línea correspondiente a la posición en la lista con un valor igual al valor de la expresión. Si el valor de la expresión es 4, el comando ON GOTO seleccionará el cuarto elemento de la lista de números de línea. El valor de la expresión no debe ser negativo, ya que se producirá un error.
Si el valor de la expresión es cero o mayor que el número de líneas en la lista, se ignorará la sentencia y el programa continuará en la línea siguiente.

```
10 CLS: PRINT "SOLUCION DE LA ECUACION CUADRATICA"
20 INPUT "A,B,C"; A,B,C: IF A=0 THEN 20
30 R = -B/ (2 * A): D = B * B-4 * A * C: S = SGN (D)
40 P = SQR (D * S) / (2 * A)
50 ON S + 2 GOTO 80, 60, 70
60 PRINT "RAICES PERFECTAS": PRINT R, R: END
70 PRINT "RAICES REALES": PRINT R + P, R-P: END
80 PRINT "RAICES COMPLEJAS": PRINT R, R: PRINT P,-P: END
```

DECISIONES

Una forma mucho más útil de bifurcación condicional la provee la sentencia IF... THEN. Esta sentencia, es posiblemente la más potente del lenguaje BASIC, y como tal puede variar desde una forma muy simple, hasta una muy compleja. En su forma más simple:

IF condición THEN número de línea

y tiene el siguiente significado: "Si la condición se cumple (es cierta), ENTONCES ir al número de línea indicado, si no, continuar con la siguiente".

```
120 IF D >9 THEN 250
180 IF A$="SI" THEN 600
```

En la línea 120, anterior, el programa continuará ejecutándose en la línea 250 si, y solo si, el valor de D es mayor que 9. Si D es menor o igual que 9 el programa continuará en la línea siguiente. En la línea 180 el salto a la línea 600 se producirá, solo si la variable de cadena contiene los caracteres SI. La igualdad en este caso deberá de ser total. SIS, SA, S, (o incluso 'si' en minúsculas), hará que el programa ignore el salto.
En su forma más completa la sentencia IF... THEN aparecerá como:

IF condición THEN acción 1 ELSE acción 2

que indica: "SI la condición se cumple ENTONCES realizar la acción 1, SI NO se cumple realizar la acción 2".

```
210 IF P=3 THEN PRINT "CIERTO" ELSE PRINT "FALSO"
```

En este ejemplo, CIERTO se imprimirá solo si P es igual a 3, si P tiene cualquier otro valor, se imprimirá FALSO. En ambos casos el programa continuará ejecutando la línea siguiente. Tanto acción 1, como acción 2, puede constar de varias sentencias.

```
210 IF P=3 THEN PRINT "CIERTO": R=R+1: GOTO 560
    ELSE PRINT "FALSO": L=L+1
```

Si P es igual a 3, el ordenador imprimirá CIERTO, sumará 1 a la variable R y continuará el programa en la línea 560. Para cualquier otro valor, imprime FALSO, suma 1 a L y continúa en la línea siguiente.

El ordenador aceptará cualquier sentencia BASIC válida para acción 1 y acción 2, incluyendo otras sentencias IF... THEN. La única restricción es que la sentencia completa IF... THEN debe de entrar en una línea. (Una línea puede contener hasta un máximo de 256 caracteres incluyendo el número de línea).

Hasta ahora solo hemos considerado lo que ocurre después de evaluar la condición. La decisión del cálculo depende de la condición. Una condición solo puede ser CIERTA o FALSA. (El ordenador da a CIERTO el valor 1, y a FALSO el valor Ø). Una condición simple tiene la forma:

expresión 1 relación expresión 2

expresión 1 y expresión 2 son las expresiones usuales en BASIC, tal como aparecen en las sentencias de asignación. Ambas expresiones deben de ser del mismo tipo, (ambas numéricas, o ambas de cadena). Una relación puede ser cualquiera de estas:

SIGNIFICADO	SIMBOLO	EJEMPLO
Igual a	=	**6Ø IF X = Y + 2 THEN 1ØØ**
Menor que	<	**11Ø IF A * B + 2 < C/2 THEN 2Ø**
Mayor que	>	**185 IF A$ > B$ THEN PRINT A$**
Menor o igual que	<=	**22Ø IF 4 * W9 <= B/Z9 THEN A = A — 1**
Mayor o igual que	>=	**415 IF B7 >= Ø THEN X = Ø**
Distinto de	<>	**8Ø IF A$ <> "SI" THEN 999**

Fíjese en que la relación se puede aplicar tanto a cadenas como a expresiones numéricas. Cuando se comparan cadenas, se miran los caracteres por su orden, por ello se pueden utilizar las sentencias IF... THEN para comparar el orden alfabético.
Ejemplo: La condición "A" < "B" es cierta, ya que la letra A precede alfabéticamente a la letra B. La condición "AAA" <= "A" es falsa, ya que AAA aparecería después de AA en un diccionario.

Las condiciones se pueden extender combinando dos o más condiciones mediante los operadores OR y AND.

27Ø IF A = 4 AND B = 7 THEN 5ØØ

El operador **AND** indica que ambas condiciones deben ser ciertas simultáneamente para que lo sea la expresión condicional completa. En el ejemplo, si A es igual a 4 y B es distinta de 7, toda la expresión será falsa y el programa continuará en la línea siguiente.

El operador OR significa que si cualquiera de las condiciones es cierta, lo será también toda la expresión.

32Ø IF D < 3 OR F + G > 25 THEN 1ØØ

El programa saltará a la línea 1ØØ si D es menor que 3, sin importar que valor tenga F + G. Alternativamente, saltará a la línea 1ØØ si F + G es mayor de 25, independientemente del valor que tenga D.

IF... THEN... ELSE

El formato completo de la sentencia IF es,

IF condición THEN acción 1 ELSE acción 2

La sentencia evalúa la condición que puede ser CIERTA o FALSA. Si es CIERTA se realizará la acción 1, si es FALSA se realizará la acción 2.

Una condición está formada por una expresión, una relación y otra expresión. Las expresiones pueden ser cualquier expresión en BASIC del mismo tipo (es decir, ambas numéricas o de cadena). El operador de relación puede ser cualquiera de los siguientes:

=	Igual a	<>	Distinto de
>	Mayor que	<	Menor que
>=	Mayor o igual que	<=	Menor o igual que

Las condiciones también se pueden combinar mediante los operadores lógicos AND, OR, NOT.

"Condición" AND "condición" será solo cierta, si ambas condiciones lo son.
"Condición" OR "condición" es cierta si lo son cualquiera de las dos condiciones.
NOT "condición" es cierta si la condición es falsa.

Acción 1 y acción 2 pueden ser cualquier sentencia BASIC, incluyendo otra sentencia IF.

La parte ELSE del comando es opcional y se puede omitir, en cuyo caso el programa continuará en la línea siguiente si no se cumple la condición.

```
10 CLS: PRINT @ 9, "JUEGO DE ADIVINANZAS": N = RND (100): T = 0
20 INPUT "ADIVINA MI NUMERO"; G: T = T + 1
30 IF G = N THEN PRINT "LO ADIVINASTE EN"; T; "INTENTOS": END
40 IF G > N THEN PRINT "NO, ES MENOR" ELSE PRINT "NO, ES MAYOR"
50 GOTO 20
```

INKEY$

El comando INKEY$ es una función. Muestrea el teclado para ver si se ha pulsado alguna tecla, si es asi, devuelve el carácter pulsado.

Se puede utilizar para meter un carácter en una variable de cadena, y no necesita que se pulse [ENTER] después del carácter.

```
10 CLS0: PRINT  , "PULSE CUALQUIER TECLA Y YO LE";
20 PRINT@32, "DIRE CUAL HA SIDO";
30 B$="NO ESTA PULSANDO NINGUNA TECLA"
40 C$="LA TECLA QUE PULSO FUE:"
50 A$=INKEY$
60 IF A$=""THEN PRINT@193, B$; ELSE PRINT@193, C$; A$;
70 FOR D=1 TO 600: NEXT D: GOTO 50
```

El programa que viene ahora, es una forma simplificada de un programa educativo. Los comentarios del programa indican lo que hace cada sección. Fíjese en la utilización de ON... GOTO para seleccionar la opción, y en las sentencias IF... THEN para revisar las operaciones. En la línea 8ØØ hemos introducido una nueva palabra, INKEY$. Este comando muestrea el teclado para ver si se ha pulsado algo, si ha sido asi, el carácter pulsado se almacenará en A$. No necesita que se pulse [ENTER] después de la entrada.

El programa parece mucho más largo de lo que es, debido a que la mitad de las líneas son de comentarios. En un futuro, no usaremos tantos comentarios con objeto de ahorrar memoria.

```
1Ø  REM PROGRAMA PARA PRACTICAR ARITMETICA
2Ø  REM
3Ø  REM PONER A CERO LOS CONTADORES Y PEDIR EL NIVEL DE
    DIFICULTAD
4Ø  REM
5Ø  R = Ø: W = Ø: CLS
6Ø  T$ = "PRACTICAS ARITMETICAS"
7Ø  PRINT @ 6, T$
8Ø  PRINT @ 64, "INTRODUZCA EL NIVEL DE DIFICULTAD"
9Ø  PRINT @ 128, "     UN NUMERO ENTRE 1 Y 1Ø": INPUT L: L1=1Ø * L—1
1ØØ REM
11Ø REM PRESENTAR OPCIONES
12Ø REM
13Ø CLS: PRINT @ 6, T$
14Ø PRINT @ 71, "1.  SUMA"
15Ø PRINT @ 1Ø3, "2.  RESTA"
16Ø PRINT @ 135, "3.  MULTIPLICACION"
17Ø PRINT @ 167, "4.  DIVISION"
18Ø REM
19Ø REM FIJAR LA POSICION DE ESCRITURA
2ØØ P = 224: Q = 352
22Ø REM
23Ø REM SELECCIONAR OPCIONES
24Ø REM
25Ø PRINT @ P, "     CUAL QUIERE INTENTAR  "; :INPUT A
26Ø REM
27Ø REM SELECCION DE DOS NUMEROS PARA EL PROBLEMA
28Ø REM
```

```
290 N1=RND (L1): N2=RND (L1)
300 REM
310 REM SALTAR A LA OPCION
320 REM
330 ON A GOTO 390, 460, 530, 600
340 REM
350 REM OPCION ERRONEA - NUEVO INTENTO
360 REM
370 CLS: SOUND 160, 3: GOTO 130
380 REM
390 REM SECCION PARA SUMAR
400 REM
410 PRINT @ P, "                                    SUMA"
420 PRINT @ Q, "     CUANTO ES"; N1; "MAS  "; N2; : INPUT N4
430 N3 = N1 + N2
440 IF N4 = N3 THEN 690 ELSE 730
450 REM
460 REM SECCION PARA RESTAR
470 REM
480 PRINT @ P, "                              RESTA"
490 PRINT @ Q, "CUANTO ES"; N1; "MENOS  "; N2; : INPUT N4
500 N3 = N1 — N2
510 IF N4 = N3 THEN 690 ELSE 730
520 REM
530 REM SECCION PARA MULTIPLICAR
540 REM
550 PRINT @ P, "                        MULTIPLICACION"
560 PRINT @ Q, "CUANTO ES"; N1; "POR"; N2; : INPUT N4
570 N3 = N1 * N2
580 IF N4 = N3 THEN 690 ELSE 730
590 REM
600 REM SECCION PARA DIVIDIR
610 REM
620 PRINT @ P, "                              DIVISION"
630 PRINT @ Q, "CUANTO ES"; N1; "DIVIDIDO POR"; N2; : INPUT N4
640 N3 = N1/N2
650 IF N3 = N4 THEN 690 ELSE 730
660 REM
670 REM RESPUESTA CORRECTA
680 REM
```

```
69Ø R = R + 1: PRINT @ P, "                    CORRECTO": GOTO 77Ø
7ØØ REM
71Ø REM RESPUESTA ERRONEA
72Ø REM
73Ø W = W1: PRINT @ P, "                    ERRONEO": PRINT
     @ P + 32, "          LA RESPUESTA ES"; N3: GOTO 77Ø
74Ø REM
75Ø REM PETICION DE REPETICION Y BORRAR LINEAS
76Ø REM
77Ø FOR D = 1 TO 1ØØØØ: NEXT D
78Ø PRINT @ P, "      ": PRINT @ P + 32, "      ": PRINT @ Q, "      "
79Ø PRINT @ P, "QUIERE INTENTARLO OTRA VEZ? (S/N)"
8ØØ A$ = INKEY$: IF A$ = " "THEN 8ØØ
81Ø IF A$ = "S" THEN 13Ø
82Ø REM
83Ø REM DAR RESULTADOS Y TERMINAR
84Ø REM
85Ø CLS: PRINT @ 128, "HA OBTENIDO"; R; "RESPUESTAS CORRECTAS Y",
     W; "ERRONEAS"
86Ø END
```

REPITIENDOLO UNA Y OTRA VEZ

Un programa a menudo necesita repetir un número de veces una secuencia de líneas. El próximo tipo de sentencia de bifurcación nos permite hacer esto. Introduzca este programa y ejecútelo

```
1Ø CLS
2Ø FOR I = 1 TO 5
3Ø PRINT @ 198, "CONTADOR I ="
4Ø PRINT @ 2Ø9, I
5Ø NEXT I
6Ø PRINT @ 396, "FIN DEL BUCLE"
```

Es demasiado rápido para ver lo que ocurre, asi que agregue la línea:

```
45 FOR J = 1 TO 1ØØ: NEXT J
```

Como puede ver, el programa cuenta desde 1 a 5Ø. Las líneas que se repiten son la 3Ø, 4Ø y 45. Una secuencia que se repite como esta se llama un "bucle", el programa vuelve a la línea 2Ø. Las sentencias que

controlan el bucle están en la línea 2Ø (el principio del bucle) y en la línea 5Ø (el final del bucle). La línea 2Ø indica que 'PARA todos los valores entre 1 y 5Ø en incrementos de 1, ejecutar todas las sentencias que hay a continuación hasta que se alcance la siguiente sentencia NEXT'. La variable I actúa como contador y toma los valores 1,2,3... 5Ø. Hemos introducido la línea 45, para que no contara tan deprisa. Este bucle no tiene sentencias que realizar, asi que el contador (esta vez la variable J) solo cuenta de 1 a 1ØØ. Esta pausa es suficiente para retardar el otro bucle, y permitirnos leer los números de la pantalla. Si cambiamos ahora la línea 2Ø por:

2Ø FOR I = 1 TO 5Ø STEP 2

El contador contará ahora desde 1 en incrementos de 2 (1,3,5,7...49). El agregar STEP podemos decidir el incremento del contador. Si se omite la palabra STEP se toma por defecto un incremento de 1. Modifique el ejemplo introduciendo las siguientes líneas,

```
15 INPUT "COMIENZO, FIN, INCREMENTO"; A,B,C
2Ø FOR I = A TO B STEP C
7Ø PRINT @ 428, "Y I="; I
```

Ejecute el programa con varios valores para el comienzo, fin y incremento. Pruebe alguno de los siguientes valores

COMIENZO	FIN	INCREMENTO
50	1	1
—50	5Ø	5
1	1Ø	Ø.5
2.6	3.9	Ø.1
1	—2	1

Notará que puede contar hacia arriba o hacia abajo con cualquier incremento que desee. La única regla es que si el incremento es positivo el comienzo debe de ser menor que el valor final, o si el incremento es negativo el comienzo debe de ser mayor que el final. Del último ejemplo anterior (1,2,—1) puede ver que si rompe esta regla, el bucle se continuará realizando una vez.

Los bucles se pueden "anidar" dentro de otros (el bucle de la línea 45 está anidado dentro de otro que comienza en la línea 2Ø). Deberá asegurarse de cerrar los bucles en el orden adecuado.

```
20 FOR I = 1 TO 10
30 FOR J = —2 TO 4 STEP 0.6
40 FOR K = 1 TO 0 STEP —0.1
100 NEXT K
110 NEXT J
120 NEXT I
```

Cada sentencia FOR deberá tener su correspondiente sentencia NEXT, la variable después de la sentencia NEXT indica a que FOR pertenece. Los bucles se deben cerrar en el orden inverso al que fueron abiertos. Si antes, las tres últimas líneas fueran.

```
100 NEXT J
110 NEXT K
120 NEXT I
```

el programa se detendrá dando un mensaje de error, ya que los bucles J y K se solapan.

Si todos los bucles terminan en el mismo punto, la sentencia NEXT se puede poner como:

```
100 NEXT K, J, I
```

pero el orden de las variables deberá conservarse como antes.

Las variables utilizadas para fijar un bucle, (en el ejemplo I, A, B y C), se pueden utilizar dentro del bucle. Pero no podrá alterar los valores de comienzo, fin o incremento. Se puede modificar el contador poniéndolo a la izquierda en una sentencia de asignación, pero no es aconsejable hacerlo.

Se pueden utilizar otras sentencias de bifurcación, como GOTO o IF... THEN, dentro de un bucle para dejarlo antes de que termine. Sin embargo, no se puede realizar un salto al interior de un bucle desde el exterior. Un bucle debe comenzar por la sentencia FOR.

El siguiente programa demuestra la utilización de bucles anidados. Simula un reloj digital. Para ajustarlo deberá variar el bucle de retardo en la línea 220. Fíjese en la utilización de INKEY$ para detener y arrancar el reloj en las líneas 90, 150 y 170.

```
10 CLS: PRINT @ 10, "RELOJ DIGITAL";
20 'IMPRIMIR POSICIONES
```

FOR... NEXT... STEP

El comando FOR... NEXT consta de dos sentencias,

FOR variable numérica = expresión TO expresión
STEP expresión

y,

NEXT variable numérica

Las sentencias FOR... NEXT trabajan conjuntamente para controlar el número de veces que es ejecutada una sección de programa.

Las expresiones se evalúan y el bucle va desde el valor de la primera expresión hasta el valor de la segunda en incrementos definidos por la tercera. El valor actual del contador se mantiene en la variable numérica. Un bucle se realizará por lo menos una vez, aunque el rango y el incremento sean imposibles. Si se omite la parte STEP de la sentencia, se toma 1 por defecto.

Los bucles se pueden anidar dentro de otros, pero debe hacerse en orden correcto.

Puede saltar fuera de un bucle utilizando una sentencia GOTO... pero no puede saltar desde fuera al interior de un bucle.

```
10 CLS: CLEAR: DIM L (1000)
20 INPUT "INTRODUZCA UN NUMERO"; N: IF N. 1000 OR N<2 THEN 20
30 CLS: PRINT "NUMEROS PRIMOS ENTRE 2 Y"; N
40 PRINT: FOR I = 2 TO N: IF L (I)<0 THEN 80
50 PRINT I;
60 IF I>SQR (N) THEN 80
70 FOR J = I TO N STEP I: L (J) = 1: NEXT J
80 NEXT I
```

```
30  P = 261: Q = 196
40  ' BLOQUE DE INSTRUCCIONES
50  PRINT @ 384, "PULSE<ESPACIO>PARA PARAR Y COMENZAR";
60  PRINT @ 425, "Y 'R' PARA COMENZAR";
70  ' PRESENTAR CABECERAS Y ESPERAR EL ARRANQUE
80  PRINT @ Q, "HORAS"; :PRINT @ Q + 10, "MINS"; :PRINT @ Q + 20,
    "SEGS";
90  A$ = INKEY$: IF A$<>" " THEN 90
100 'COMIENZO DEL BUCLE PARA LAS HORAS, MINS Y SEGS
110 FOR H = 0 TO 23: FOR M = 0 TO 59: FOR S = 0 TO 59
120 SOUND 220, 1
130 'BUCLE PARA LAS DECIMAS DE SEGUNDO
140 FOR T = 0 TO 9
150 A$ = INKEY$: IF A$<>" " THEN 200
160 ' CHEQUEO DE UNA PULSACION
170 A$ = INKEY$: IF A$ = "R" THEN 110
180 IF A$<>" " THEN 170
190 ' PRESENTAR EL TIEMPO
200 PRINT @ P, H; :PRINT @ P + 10, M; :PRINT @ P + 19, S; ". ; T;
210 'BUCLE PARA EL AJUSTE DE TIEMPO
220 FOR D = 1 TO 13: NEXT D
230 NEXT T
240 'FIN DEL BUCLE PARA LAS DECIMAS
250 NEXT S, M, H
260 ' FIN DEL BUCLE DE LOS SEGS, MINS Y HORAS
270 PRINT @ 448, "DETENIDO"
280 END
```

SUBRUTINAS

Normalmente los programas están divididos en bloques de instrucciones. Algunos de estos bloques son necesarios varias veces en distintas partes del programa, por ello, la estructura del programa se podrá a menudo simplificar tratando estas situaciones como subrutinas. Una subrutina, como el nombre indica, es una parte subsidiaria de un programa, o un programa dentro de otro. La característica principal de una subrutina es que cuando se la llama, se ejecuta su conjunto de líneas y al terminar se continúa la ejecución a partir del punto de llamada. Para llamar a una subrutina se utiliza la sentencia:

GOSUB número de línea

Donde número de línea es el número de línea de programa donde comienza la subrutina. Para volver de una subrutina, cada subrutina deberá terminar con la sentencia,

RETURN

GOSUB se comporta de forma parecida a GOTO. La diferencia estriba en que con GOTO la ejecución del programa continúa a partir de la línea indicada, y no volverá al punto de llamada a menos que se encuentre con otra sentencia GOTO que se lo indique. Introduzca el siguiente ejemplo y ejecútelo (RUN)

```
10 CLS: PRINT "EN EL PROGRAMA PRINCIPAL"
20 GOSUB 50
30 PRINT "DE VUELTA EN EL PROGRAMA PRINCIPAL"
40 END
50 PRINT "EN LA PRIMERA SUBRUTINA"
60 GOSUB 90
70 PRINT "DE VUELTA EN LA PRIMERA SUBRUTINA"
80 RETURN
90 PRINT "EN LA SEGUNDA SUBRUTINA"
100 RETURN
```

La secuencia en que se ejecutarán las líneas de programa es:

10, 20, 50, 60, 90, 100, 70, 80, 30, 40

Fíjese en como la primera subrutina (líneas 50-80) llama a la segunda subrutina (líneas 90, 100) La sentencia END simplemente provoca el fin de la ejecución cuando esta alcanza ese punto. Quite esta instrucción del programa y ejecútelo de nuevo. Aparecerá el error ?RG ERROR IN 80. Esto se debe a que el programa encontró una sentencia RETURN sin haberle indicado que fuera a una subrutina. El programa continúa su ejecución después del final introduciéndose en una subrutina. Deberá por tanto proteger las subrutinas para asegurarse de que el único modo de acceder a ellas es mediante una sentencia GOSUB, y la única forma de salir es mediante RETURN. Puede utilizar sentencias GOTO, IF... THEN y cualquier otro tipo de bifurcaciones dentro de una subrutina, pero no deberán provocar un salto fuera de esta.

Como ocurría con la sentencia ON... GOTO, se puede también provocar una ramificación condicional a subrutinas, como un formato parecido,

ON expresión GOSUB lista de números de línea

Muchos programadores experimentados mantienen una librería de subrutinas, de forma que muchos programas se pueden hacer a partir de lo almacenado. Luego, es aconsejable numerar las subrutinas con números altos (10000) de forma que se puedan introducir directamente en un programa sin necesidad de renumerarlo.

Aparecerán bastantes ejemplos de subrutinas en los capítulos posteriores, asi que no damos ahora ningún ejemplo.

GOSUB
RETURN
ON... GOSUB

El comando GOSUB transfiere el control del programa al comienzo de una subrutina. RETURN devuelve de nuevo el control a la línea que viene después de la sentencia GOSUB.

GOSUB viene seguido por un número de línea, que es la primera de la subrutina.

GOSUB 16ØØ

Una subrutina deberá contener por lo menos una sentencia RETURN.

El comando ON... GOSUB permite una ramificación condicional a subrutinas de forma parecida al comando ON... GOTO.

ON expresión GOSUB lista de números de línea

Si el valor de la expresión es negativa el programa se detendrá dando un mensaje de error, si es cero o mayor que el número de elementos en la lista de números de línea, se ignorará la sentencia y el programa continuará en la siguiente línea.

```
1Ø  CLS: INPUT "INTRODUZCA UN PAR DE NUMEROS"; A, B
2Ø  INPUT "AHORA INTRODUZCA UN NUMERO ENTRE 1 Y 4"; C
3Ø  ON C GOSUB 1ØØ, 2ØØ, 3ØØ, 4ØØ
4Ø  IF C>=1 AND C<=4 THEN END
5Ø  PRINT C; "NO ESTA ENTRE 1 Y 4": GOTO 2Ø
1ØØ PRINT "LA SUMA DE"; A; "MAS"; B; "ES"; A + B
11Ø RETURN
2ØØ PRINT "LA DIFERENCIA DE"; A; "MENOS"; B; "ES"; A * B
21Ø RETURN
3ØØ PRINT "EL PRODUCTO DE"; A; "POR"; B; "ES"; A + B
31Ø RETURN
4ØØ PRINT "EL COCIENTE ENTRE"; A; "Y" ; B; "ES"; A/B
41Ø RETURN
```

CAPITULO SEIS

NUEVAS DIMENSIONES

En el capítulo 2 cuando hablamos de los tipos de variables, dijimos que las variables podían ser de dos tipos, simples y matrices. Hasta ahora solo hemos utilizado las simples.

Si quisiera utilizar el ordenador para mantener un índice de todos sus libros, o discos, se le acabarían terminando las variables donde almacenarlos, además sería un programa muy difícil de escribir, para poder seguir la pista a tantos nombres de variable! Su ordenador le ayuda en este tipo de problemas mediante las matrices.

LISTAS Y TABLAS

Las variables de tipo matriz son especialmente útiles para manejar listas de elementos, asi podremos tener un conjunto de libros ordenados de este modo:

1. Título 1
2. Título 2
3. Título 3

Para referirnos ahora a los libros, podemos preguntar por el número 8 de la lista. En nuestro programa daremos un nombre a esta secuencia de variables (los títulos), y nos referimos a un valor de la lista dando su número de índice. Asi que lo primero que debemos hacer es dar un nombre a la lista.

Los nombres para matrices siguen las mismas reglas que los de variables simples. El ordenador sabe la diferencia que existe entre ambos, ya que las variables de tipo matriz van seguidas por paréntesis conteniendo el número de índice.

A (5) se refiere al quinto elemento de una matriz numérica llamada A.

D7$ (28) se refiere al 28avo elemento de una matriz de cadena llamada D7$.

Para definir una matriz hay que dar al ordenador el nombre y su dimensión. Esto se realiza mediante la sentencia DIM.

DIM nombre (número), nombre (número, número)

DIM indica dimensión, la sentencia no solo nombra las matrices, además define su dimensión máxima. El número puede ser cualquier número positivo, o una variable simple, suponiendo que esta ya tenga un valor asignado. No dimensione las matrices con más espacio del necesario, las matrices grandes quitan mucho espacio.

1Ø DIM A (22), NA$ (4Ø)

La línea anterior indica al ordenador que fije una matriz llamada A de dimensión 22 y una matriz de cadenas llamada NA$ de dimensión 4Ø. (Realmente las dimensiones son 23 y 41, ya que la numeración de los índices comienza por Ø).

Para hacer referencia a un elemento de una matriz en una expresión, simplemente incluya el número de índice, entre paréntesis después del nombre.

25 A (4) = 7.Ø
32 A (M) = B * C/A

En la línea 25 se asigna el valor 7.Ø al elemento de índice 4 de la matriz A. La línea 32 calculará la expresión y pondrá el resultado en el elemento M de la matriz A, (fíjese en que la A a la derecha de la sentencia de la línea 32 se refiere a una variable simple llamada A, que no tiene nada que ver con la matriz A del lado izquierdo).

Las matrices pueden tener dos dimensiones, la línea:

1Ø DIM T (1Ø,5), TB$ (12,4)

creará una matriz numérica T, que será una tabla con 10 filas y 5 columnas. La matriz de cadena, TB$, tendrá 12 filas y 4 columnas. Por ejemplo, un profesor puede querer almacenar las notas de 25 alumnos en seis asignaturas distintas. La matriz EXAM (25,6) nos servirá para este propósito. Para referirnos a un elemento de esta matriz necesitaremos dos índices.

25 PRINT EXAM (1Ø,3)

presentará la nota del estudiante 1Ø en la asignatura 3. Por supuesto los índices deberán referirse a elementos existentes. Si intenta utilizar EXAM (3Ø,7) se producirá un error ya que la matriz es de menor dimensión.

A continuación veremos un programa reducido sobre el uso de matrices, necesitará estudiarlo un poco para ver como trabaja. El programa mezcla un paquete de cartas. Cada carta tiene un número entre 1 y 52 en la matriz X. Los elementos se toman de X aleatoriamente y se introducen en la matriz Y. Cuando acaba el programa, y contiene los números 1 al 52 situados aleatoriamente. No se puede utilizar el generador de números aleatorios directamente para producir el paquete de cartas mezcladas, ya que cada carta tiene un número distinto, y la función RND podría generar el mismo número más de una vez. La línea 5Ø imprime la baraja mezclada, fíjese en como utiliza una expresión como índice de la matriz.

DIM

El comando DIM dimensiona matrices. Las matrices pueden ser de una o dos dimensiones y pueden ser numéricas o de cadena. Las reglas para nombrar matrices son las mismas que para variables.

DIM nombre de matriz (n), nombre de matriz (n,n)

Si la dimensión máxima de la matriz no excede de 1Ø, no es necesario dimensionarla con DIM.

Para referirse a un elemento de la matriz, se debe poner el número del elemento entre paréntesis después del nombre.

A (14, K) N9 (B) L$ (14,4)

```
10 DIM X (52), Y (52): CLS
20 FOR I = 1 TO 52: X (I) = I: NEXT I
30 FOR I = 52 TO 1 STEP — 1
40 J = RND (I): Y (I) = X (J): X (J) = X (I): NEXT I
50 FOR I = 1 TO 13: FOR J = 1 TO 4: PRINT Y (4 * (I — 1) + J); :NEXT J: PRINT:
   NEXT I: END
```

Los elementos de una matriz de cadena se pueden utilizar de forma similar. Una de las aplicaciones más corrientes con cadenas es ordenar una lista por orden alfabético. En el siguiente ejemplo la subrutina que empieza en la línea 200 ordena una lista de palabras. Se ha escrito mucho sobre ordenación mediante computador, y el método aqui utilizado se llama ordenación por intercambio (método de la burbuja). No es necesariamente el mejor método pero si el más sencillo. Si no conoce el método, trabaje a mano con la lista D, B, A, E, C.

```
10  CLS: DIM W$ (50)
20  INPUT "CUANTAS PALABRAS": N
30  CLS: PRINT "ORIGINALMENTE"
40  FOR I = 1 TO N: PRINT I; ".      ";
50 INPUT W$ (I): NEXT I
60 GOSUB 200
70  PRINT @ 18, "ORDENADAS"
80  FOR I = 1 TO N
90  PRINT @ 18 + 32 * I, I; ".      "; W$ (I)
100 NEXT I: END
200 M = N
210 F = 0: FOR I = 1 TO M — 1
220 IF W$ (I) < = W$ (I + 1) THEN 240
230 T$ = W$ (I): W$ (I) = W$ (I + 1): W$ (I + 1) = T$: F = 1
240 NEXT I: IF F = 1 THEN M = M — 1: GOTO 210
250 RETURN
```

QUE ES UNA FUNCION

Recuerda RND, y que la llamábamos una función? Pues bien, no es la única existente. Una función, en el sentido informático, es un subprograma especial al que se le pasan un conjunto de argumentos y devuelve un valor simple. Una función en BASIC tiene el siguiente formato:

Nombre de función (argumentos)

y puede ser utilizada en cualquier expresión del mismo modo que otros operadores aritméticos (* ,/,+,—,^). Las funciones tienen prioridad sobre cualquier otro operador, excepto los paréntesis.

Los argumentos de una función son los valores que se le pasan a esta para que con ellos calcule el resultado. Los argumentos pueden ser constantes, variables o expresiones.

RND (1Ø), RND (X) o RND (A * 2 + F)

tienen argumentos válidos. Fíjese que los argumentos van siempre entre paréntesis.

Su ordenador tiene ya definidas algunas funciones como parte del lenguaje BASIC. Las funciones ya definidas se pueden clasificar en cinco grupos distintos. Veremos cada grupo por turno, daremos una lista de las funciones, una breve explicación y una línea ejemplo. Estas funciones aparecerán de ahora en adelante en los programas.

Grupo I

Estas son funciones numéricas, principalmente para usos matemáticos. Tienen un argumento numérico y devuelven un valor numérico. Las funciones del grupo I solo se pueden usar en expresiones numéricas. Aquellos que no estén familiarizados con las funciones trigonométricas vean el apéndice D.

NOMBRE DE FUNCION	OPERACION	EJEMPLO
ABS (X)	Valor absoluto de X	**1ØØ A = ABS (D * 2—C)**
ATN (X)	Arcotangente de X, en radianes. Inversa de TAN (X)	**11Ø PRINT "ANG ="; ATN (R3)**
COS (X)	Coseno de X, siendo X un ángulo en radianes.	**51Ø F7 = COS (X + 4)**
EXP (X)	Eleva el número e a la potencia X. Inversa del LOG (X).	**215 Q = EXP (—A * A)**
FIX (X)	Devuelve la parte entera de X (Trunca la parte decimal).	**172 N = FIX (Z * .Ø5)**
INT (X)	Trunca X si es positivo como FIX. Si X es negativo, redondea por debajo. ej. INT de —12.ØØ1 es 13	**28Ø P = INT (1ØØ * X)**

NOMBRE DE FUNCION	OPERACION	EJEMPLO
JOYSTK (X)	Devuelve la posición actual del joystick izq. o derecho, como sigue: X = Ø, joystick izq. horizontal X = 1, joystick izq. vertical X = 2, joystick der. horizontal X = 3, joystick der. vertical	**1ØØØ A = JOYSTK (Ø)** **B = JOYSTK (1)**
LOG (X)	Logaritmo natural de X. El valor del argumento debe ser mayor que Ø. Inversa de EXP (X).	**617 L1 = 5.2 * LOG (W4)**
PEEK (X)	Devuelve el contenido de la posición de memoria cuya dirección es X.	**55 P = PEEK (65280)**
POINT (X, Y)	Mira a ver si una celda en baja resolución está off. X debe estar entre Ø — 63 (horizontal), y entre Ø — 31 (vertical). Devuelve el valor Ø si la celda está off, —1 en modo texto, si está activa devuelve de 1 a 8 según el color.	**5Ø IF POINT (5, A) = C THEN 21Ø**
POS (X)	Devuelve la posición actual de impresión. Los únicos argumentos son: Ø para pantalla —2 para impresora.	**168 IF POS (Ø) >3Ø THEN PRINT A$**
PPOINT (X, Y)	Testea una celda en alta resolución de gráficos. Devuelve Ø si la celda está off, si no, devuelve el código del color de la celda. (X comprendido entre Ø—255, y entre Ø—191)	**115 C = PPOINT (A1, A2)**

NOMBRE DE FUNCION	OPERACION	EJEMPLO
RND (X)	Devuelve un número aleatorio entero entre 1 y X. Si X es Ø, devuelve un número aleatorio entre Ø y 1.	**22Ø PRINT @ RND (51Ø); " ★ "**
SGN (X)	Devuelve el signo del argumento. Si X es negativo da —1 Si es cero devuelve Ø Si X es positivo da + 1	**412 Y = RND (ABS (N)) ★ SGN (N)**
SIN (X)	Calcula el seno de X, estando X en radianes	**2Ø5 S = SIN (K ★ PI/180)**
SQR (X)	Raiz cuadrada del argumento, X no debe ser negativo, si lo fuera se devuelve SQR (ABS (X)).	**33Ø C = SQR (A ★ A + B ★ B)**
TAN (X)	Tangente de X, donde X es un ángulo en radianes. Es la inversa de ATN (X).	**84Ø R5 = B/TAN (EQ—5)**

Grupo II

Las funciones del grupo II tienen un argumento numérico, pero devuelven un valor de cadena. Solo se pueden utilizar en expresiones de cadena.

NOMBRE DE FUNCION	OPERACION	EJEMPLO
CHR$ (X)	Devuelve el carácter para el código dado en X. X debe estar entre Ø y 255 Ver el apéndice A.	**2Ø M$ = CHR$ (143) + CHR$ (128)**

NOMBRE DE FUNCION	OPERACION	EJEMPLO
HEX$ (X)	Calcula el valor hexadecimal del valor X decimal.	**42 PRINT HEX$ (3Ø)**
STR$ (X)	Convierte un valor numérico en su carácter equivalente.	**175 A$ = STR$ (12.49)**

Grupo III

Las funciones del Grupo III son funciones de cadena. Los argumentos (son dos normalmente), son una cadena y un número. Todas devuelven un valor de cadena y por ello deberán ser parte de una expresión de cadena.

NOMBRE DE FUNCION	OPERACION	EJEMPLO
LEF$ (X$, N)	Devuelve los primeros N caracteres de la cadena X$	**114 A$ = LEFT$ (B$, 7)**
MID$ (X$, M, N)	Devuelve N caracteres de la cadena X$, a partir de la posición M. Si se omite N se devuelve toda la cadena a la derecha de M. M debe ser mayor de Ø.	**76Ø K$ = MID$ (W$, I, 4)**
RIGHT$ (X$, N)	Devuelve los últimos N caracteres de la cadena X$	**34Ø T$ = RIGHT (Q$, B + 7)**
STRING$ (N, C)	Devuelve una cadena de longitud N con el carácter definido por C. C puede ser un número (el código ASCII del carácter) o el propio carácter entre comillas.	**4ØØ A$ = STRING$ (5,67)** **41Ø PRINT STRING$ (32, "*")**

Grupo IV

Estas son unas funciones mezcladas parecidas a las del grupo II. Tienen un argumento de cadena y devuelven un valor numérico, y por lo tanto solo pueden aparecer en expresiones numéricas.

NOMBRE FUNCION	OPERACION	EJEMPLO
ASC (X$)	Devuelve el código ASCII del primer carácter de la cadena.	**715 P = ASC (F$)—64**
INSTR (P, S$, T$)	Explora la cadena S$ buscando la cadena T$, a partir de la posición P. Devuelve Ø si no la encuentra, si no su posición.	**212 F = INSTR (N, X$, "AB")**
LEN (X$)	Devuelve la longitud de la cadena X$. Se cuentan también los espacios. Si la cadena está vacía da Ø.	**845 N = LEN (N$)**
VAL (X$)	Convierte una cadena numérica en un número. Si esta empieza por un carácter alfabético, devuelve Ø.	**92 Z = VAL ("123")**

Grupo V

Las funciones del Grupo V son funciones del sistema. No tienen argumentos.

NOMBRE DE FUNCION	OPERACION	EJEMPLO
INKEY$	Chequea el teclado y devuelve la tecla pulsada (si se ha hecho). Devuelve una cadena, luego debe usarse en una expresión de cadena.	**146 P3$ = INKEY$**

NOMBRE DE FUNCION	OPERACION	EJEMPLO
MEM	Devuelve el espacio libre de memoria.	**PRINT MEM**
TIMER	Devuelve el contenido del timer, un valor entre Ø y 65535. Para ponerlo a Ø use TIMER = Ø.	**62 T1 = TIMER—T** **63 TIMER = Ø**

FUNCIONES D.I.Y.

Además de las funciones suministradas por el sistema, Vd. puede definir hasta 26 funciones numéricas. El formato de esta sentencia es:

DEF FN letra (variable objeto) = fórmula

La letra puede ser cualquiera de la A a la Z. La variable objeto es una letra, que será sustituída por el argumento cuando se llame a la función. La fórmula es una expresión en BASIC escrita en términos de la variable objeto y/o otras variables. Pueden aparecer otras funciones dentro de una función (ya sean del sistema o definidas por el usuario), pero una función no se puede llamar a si misma.

25 DEF FNY (X) = ((X—3)∧2 + (X—4)∧4)/X∧3

X es la variable objeto, no un nombre de variable. Cuando posteriormente se llame a la función desde el programa, el argumento sustituirá a la variable objeto. Para utilizar la función, simplemente inclúyala en una expresión como lo haría con una función del sistema.

15Ø Y (I) = FNY (X) + FNY (W)

Se pueden usar también las funciones para proveer 'rutinas de servicio', (operaciones utilizadas frecuentemente). Puede que haya notado que los números obtenidos como resultado y presentados en pantalla aparecen un poco desordenados. El ordenador trata de ayudar presentando el número con la máxima precisión. Algunas veces no se necesita tanta precisión e incluso puede ser molesto, por ejemplo, al presentar cantidades monetarias.

La siguiente función se puede utilizar para presentar el número de cifras decimales deseadas (D).

1Ø DEF FND (X) = INT (X * 1Ø^D + Ø.5)/1Ø^D

Fíjese que X es una variable objeto, pero D no lo es, el valor para D debe suministrarse fuera de la función, posiblemente mediante una sentencia INPUT. Para usar la función:

2Ø5 PRINT FND (A) etc.

Puesto que todas las funciones trigonométricas necesitan que el argumento esté en radianes, sería útil tener una función para pasar de grados a radianes:

1Ø DEF FNR (X) = X/57.295779

Se puede obtener un resultado más exacto utilizando:

1Ø DEF FNR (X) = X * ATN (1.Ø)/45

Se puede generar la constante PI mediante:

2Ø DEF FNP (X) = 4.Ø * ATN (1.Ø)

Fíjese que en los casos anteriores la variable objeto no tiene ninguna utilidad, solo se pone porque lo requiere el formato.

Como no podrá utilizar una función antes de haberla definido, es aconsejable definirlas todas al comienzo del programa.

ALTERNATIVAS PARA INPUT

La única forma que hemos visto hasta ahora para asignar valores a las variables, desde fuera del programa, ha sido la sentencia INPUT. Es muy útil, pero habrá notado que la sentencia INPUT no acepta ciertos caracteres. Si empieza una cadena con espacios, estos se pierden, y si escribe una coma perderá todo lo que venga detrás. Hay una sentencia alternativa, LINE INPUT.

LINE INPUT "mensaje"; variable de cadena

La sentencia LINE INPUT se comporta de forma parecida a INPUT, salvo en que lo acepta todo, incluyendo comas y espacios. El mensaje es el mismo que en INPUT. Solo puede haber una variable de cadena en cada sentencia LINE INPUT.

25 LINE INPUT "INTRODUZCA UNA LINEA DE TEXTO"; L$

Frecuentemente, es necesario fijar un conjunto de constantes antes de que un programa comience a trabajar. Podrá, por

DEF FN

El comando DEF FN se utiliza para definir una función numérica

DEF FN nombre (variable objeto) = fórmula

El nombre puede ser cualquier letra de la A a la Z.

La variable objeto puede ser cualquier letra, y será reemplazada por el argumento cuando se utilice la función.

La fórmula describe la operación en términos de la variable objeto y/o otras variables.

Las funciones definidas por el usuario no pueden ocupar más de una línea. Una función puede utilizar otras funciones, (definidas o del sistema) en la fórmula, pero no deberá llamarse a si misma.

Una función deberá haberse definido antes de utilizarla, por ello convendrá definirlas al comienzo del programa.

Se pueden definir otras funciones matemáticas como:

```
10 DEF FNS (X) = 1/SIN (X): 'SECANTE'
20 DEF FNI (X) = —ATN (X/SQR (—X * X + 1)) + 1.5708:
   REM INVERSA DEL COSENO
30 DEF FNH (X) = —EXP (—X)/(EXP (X) + EXP (—X))* 2 + 1:
   REM TANGENTE HIPERBOLICA
40 DEF FNM (A) = INT (A/B—INT (A/B))* B + 0.5) SGN (A/B):
   REM A MODULO B
```

LINE INPUT

El comando LINE INPUT introduce una línea completa en una variable de cadena, incluyendo comas y espacios en blanco, los cuales no son aceptados por la sentencia INPUT.

LINE INPUT "mensaje"; variable de cadena

El mensaje puede ser cualquiera que vaya entre paréntesis. Es opcional, y si se incluye deberá separarse de la variable de cadena por un punto y coma. Solo puede aparecer una variable de cadena en la sentencia LINE INPUT. La longitud máxima de la línea almacenada por esta sentencia es de 255 caracteres.

```
10 CLEAR 500: CLS
20 LINE INPUT "INTRODUZCA SU NOMBRE COMPLETO"; N$
30 LINE INPUT "Y SU DIRECCION"; D$
```

supuesto, introducir estas cada vez que se ejecute el programa, mediante una sentencia INPUT. Hay sin embargo una forma mucho más útil, mediante las sentencias READ y DATA. Se utilizan siempre juntas y tienen el siguiente formato:

READ lista de variables
DATA lista de valores

La sentencia READ se comporta del mismo modo que la sentencia INPUT, salvo que el programa en lugar de detenerse y esperar a que se le introduzcan datos por el teclado, busca los valores en las sentencias DATA que forman parte del programa. Las sentencias DATA pueden incluirse en cualquier lugar del programa. Si hay más de una sentencia DATA, la sentencia READ comienza leyendo por la sentencia DATA que tiene menor número de línea y continúa después ordenadamente.

```
10 DATA 1,2,3,4,5
20 FOR I = 1 TO 3
30 READ A: PRINT A: NEXT I
40 READ D, G: PRINT D: G
```

El ejemplo anterior leerá el primer elemento de la lista DATA (1), lo presentará, leerá el segundo (2), etc. A medida que se van leyendo los elementos el puntero se mueve al siguiente. La línea 40 leerá los dos últimos elementos. Agregue la línea:

```
50 READ X: PRINT X
```

y vuelva a ejecutar el programa, obtendrá el mensaje **?OD ERROR IN 50**. Esto indica que READ no tiene más elementos para leer en la lista DATA. Agregue ahora la línea:

```
45 RESTORE
```

Cuando ejecute de nuevo el programa funcionará, y X tomará el valor 1. La sentencia RESTORE vuelve a situar el puntero al comienzo de la primera sentencia DATA. Se pueden incluir también cadenas en las sentencias DATA. Deberá tener cuidado de que no se mezclen variables en sentencias READ con datos de otro tipo en sentencias DATA.

Si ha escrito un programa que utiliza muchas cadenas, habrá posiblemente recibido el mensaje **?OM ERROR.** Esto quiere decir que ha sobrepasado el espacio reservado para almacenar cadenas. Para reservar espacio para cadenas utilice la sentencia CLEAR.

```
10 CLEAR 1000
```

Esta sentencia reserva 1000 bytes de memoria para almacenar cadenas. Al mismo tiempo pone las variables a 0. Utilícela solo al comienzo de un programa.

READ
DATA
RESTORE

El comando READ lee el siguiente elemento de una línea DATA y se lo asigna a la variable correspondiente.

READ lista de nombres de variable

La línea DATA almacena datos dentro de un programa y puede estar situada en cualquier lugar del programa.

DATA lista de valores

En las sentencias DATA y READ se pueden utilizar tanto variables numéricas como de cadena. El tipo de variable debe ajustarse con el tipo del dato que le corresponde; es decir, una cadena debe asignarse a una variable de cadena, etc.

El comando RESTORE posiciona el puntero de datos apuntando al primer elemento en la línea DATA con número inferior.

RESTORE

```
10 CLS: PRINT: PRINT: PRINT
20 READ A, B: IF A = -9999 THEN RESTORE: GOTO 20
30 PRINT A; "  +  5  ES  "; : INPUT C
40 IF C = B THEN PRINT "CORRECTO" ELSE PRINT "ERRONEO"
50 FOR D = 1 TO 600: NEXT D: GOTO 10
60 DATA 8,13,12,17,5,10,27,32,14,19,3,8
70 DATA 7,12,6,11,1,6, -9999, -9999
```

CLEAR

El comando CLEAR borra todas las variables y reserva espacio para el almacenamiento de cadenas.

CLEAR 500

Reservará 500 bytes para el almacenamiento de variables de cadena.

El comando CLEAR se puede usar también para fijar la dirección mayor de memoria BASIC, con objeto de reservar espacio para rutinas en lenguaje máquina.

CLEAR 200, 14000

reservará 200 bytes para el almacenamiento de cadenas y fijará la dirección superior para BASIC en 14000. Ahora se podrán almacenar rutinas en lenguaje máquina a partir de la dirección 14001.

Si no se utiliza CLEAR, se reservan automáticamente 200 bytes de espacio para cadenas.

UN MOMENTO PARA REFLEXIONAR

El contenido de este capítulo, junto con los capítulos, 1, 2, 3 y 5, constituyen el núcleo del lenguaje BASIC. Aunque quedan todavía por explicar algunas sentencias BASIC, son una mínima parte del total.

Todo el material visto hasta ahora será utilizado frecuentemente en los próximos capítulos, ya que son parte fundamental de cualquier programa. Aunque estará impaciente por empezar a dibujar gráficos con su ordenador, bastante del tiempo empleado hasta ahora era necesario para que pudiera entender como funcionan y facilitarle su utilización.

Revise los programas, trate de adaptarlos a sus propias ideas.

Acabaremos esta sección con dos ejemplos más. El primero es una extensión del programa que mezclaba cartas. El programa reparte una mano de cartas. Fíjese en los siguientes detalles:

a) la mezcla aparece ahora como una subrutina en la línea 9Ø.
b) las sentencias READ y DATA fijan los valores de las matrices inicialmente.
c) las líneas 13Ø y 14Ø encuentran el palo y la carta dentro de este.

```
1Ø  DIM X (52), BARAJA (52), CARTA$ (13), PALO$ (3)
2Ø  FOR I = Ø TO 3: READ PALO$ (I): NEXT I
3Ø  DATA ESPADAS, DIAMANTES, TREBOLES, CORAZONES
4Ø  FOR I = 1 TO 13: READ CARTA$ (I): NEXT I
5Ø  DATA AS, DOS, TRES, CUATRO, CINCO, SEIS, SIETE
6Ø  DATA OCHO, NUEVE, DIEZ, JOTA, REINA, REY
7Ø  CLS: INPUT "CUANTAS CARTAS HAY QUE REPARTIR"; N
8Ø  GOSUB 19Ø
9Ø  ST = 1
1ØØ EN = ST + N—1: IF EN > 52 THEN GOTO 8Ø
11Ø CLS: PRINT @ 1Ø, "SU MANO": PRINT: PRINT
12Ø FOR I = ST TO EN
13Ø S = INT ((BARAJA (I)—1)/13)
14Ø C = PACK (I) —S * 13
15Ø PRINT @ 8, CARTA$ (C); "  DE  "; PALO$ (S)
16Ø NEXT I: ST = ST + N
17Ø PRINT @ 448, "OTRA MANO. SI O NO  "; : INPUT A$
18Ø IF A$ = "SI" THEN 1ØØ ELSE END
19Ø FOR 19 =1 TO 52: X (19) = 19: NEXT 19
2ØØ FOR 19 = 52 TO 1 STEP —1
21Ø J9 = RND (19): BARAJA (19) = X (J9): X (J9) = X (19)
22Ø NEXT 19: RETURN
```

El segundo ejemplo utiliza casi todas las funciones disponibles para cadenas. Este programa busca a lo largo de todo un texto e indica el número de ocurrencias de cada letra. Este tipo de programas se utilizan frecuentemente para descifrar mensajes en código. Se puede adaptar fácilmente para que busque palabras o secuencias de caracteres.

```
10  CLEAR 1000: CLS: READ A$
20  DATA ABCDEFGHIJKLMNOPQRSTUVWXYZ
30  PRINT "INTRODUZCA UNA LINEA DE TEXTO": PRINT
40  LINE INPUT L$
50  FOR I = 1 TO LEN (A$): CLS
60  T$ = MID$ (A$, I, 1): C = 0: P = 1: P$ = L$
70  F = INSTR (P, L$, T$)
80  IF F >0 THEN C = C + 1 ELSE 140
90  P$ = LEFT$ (P$, F—1) + STRING$ (LEN (T$), CHR$ (128))
100 IF F >LEN (L$) THEN 120
110 P$ = P$ + RIGHT$ (L$, LEN (L$) —F)
120 P = F + LEN (T$ )
130 IF P< = LEN (L$) —LEN (T$) + 1 THEN 70
140 PRINT P$
150 PRINT @ 354, "ENCONTRADAS"; C; "OCURRENCIAS DE    "; T$
160 PRINT @ 416, "PULSE «ESPACIO» PARA SEGUIR, «N» PARA PARAR"
170 Z$ = INKEY$: IF Z$ = ""THEN 170
180 IF Z$ = "N" THEN 200
190 NEXT I
200 CLS: END
```

CAPITULO SIETE

GRAFICOS

Cuando su ordenador presenta algo en la pantalla de la televisión, lo que está haciendo es activar o desactivar puntos en el tubo del televisor para construir la imagen. Si el punto está activado aparecerá como un punto coloreado, si está desactivado aparecerá en negro. Todas las letras que hemos estado imprimiendo están formadas por estos puntos de luz. La dimensión del punto que se puede controlar determina el nivel de resolución en que se trabaja. Una gran dimensión representa una baja resolución, menores dimensiones suponen mayor resolución, (ya que a medida que el punto disminuye aumentan el número de puntos disponibles en pantalla).

Su ordenador puede trabajar en cinco niveles de resolución. desde 512 puntos hasta 49152. Esto le permite una gran flexibilidad en cuanto al nivel de detalle que puede dar a sus figuras. Comenzaremos trabajando con el nivel de resolución más bajo e iremos subiendo. Los métodos utilizados para dibujar y conseguir movimiento en la pantalla son muy parecidos independientemente del nivel de resolución.

DIBUJANDO

Recuerda, en el capítulo 3, cuando vimos la sentencia PRINT @, describimos como la pantalla estaba dividida en 16 × 32 celdas. Esto nos permitía imprimir un carácter en cualquier lugar de la pantalla dando simplemente su posición. Utilizando la función CHR$ (ver el capítulo 6), podemos generar caracteres gráficos especiales. El siguiente programa presenta todos los caracteres posibles.

```
10 FOR I = 1 TO 255: CLS0
20 PRINT @ 100, "CHR$ (";I")";
30 PRINT @ 120, CHR$ (I);
40 FOR D = 1 TO 600: NEXT D, I: CLS
```

Los números del 1 al 31 se utilizan como caracteres de control. Del 32 al 127 aparecen los caracteres del teclado. Los códigos del 128 al 255 representan caracteres gráficos especiales. (Se da una lista completa de los caracteres en el apéndice A). Estos caracteres gráficos son moldes de bloques en color que se pueden unir para formar figuras. El modelo más simple es un rectángulo en color. Por ejemplo, CHR$ (143) da un rectángulo verde, que es el color 1. Sume 16, y CHR$ (159) dará un rectángulo amarillo, color 2, y asi sucesivamente. Hay 16 modelos

distintos, desde CHR$ (128) a CHR$ (143), estos están formados de verde y negro.

Para obtener el mismo modelo pero en otro color, simplemente sume el número apropiado de veces 16.

+ 16 amarillo	+ 32 azul	+ 48 rojo
+ 64 beige	+ 80 turquesa	+ 96 magenta
	+ 112 naranja	

El siguiente programa muestra los efectos de incrementar el código de 16 en 16 (Se puede también utilizar para ajustar el balance del color en su televisor, use C = 143).

```
10 CLS0: INPUT "INTRODUZCA EL CODIGO ENTRE 128 Y 143"; C
20 FOR I = 1 TO 14: FOR J = C TO 255 STEP 16
30 FOR K=1 TO 14: PRINT CHR$ (J); : NEXT K, J, I
40 GOTO 40
```

Como puede observar de lo anterior, los caracteres CHR$ se pueden imprimir directamente en pantalla. Pero como son también caracteres, se pueden además introducir en variables de cadena. Esto es muy conveniente, ya que de este modo pueden manejarse mucho mejor.

Empezaremos dibujando un castillo, primero por sus formas simples y segundo, porque muestra como empezar desde la parte más elemental para ir luego agregando detalles. Introduzca y ejecute cada sección en el orden en que están para que vea los distintos pasos del dibujo.

```
10 CLEAR 500: CLSØ
20 FOR I = 1 TO 6: FOR J = 1 TO 32
30 MURO$ = MURO$ + CHR$ (207): NEXT J, I
40 PRINT @ 256, MURO$;
200 GOTO 200
```

La primera línea reserva espacio para las cadenas que vamos a utilizar. Las líneas 20 y 30 construyen el muro con bloques beiges, CHR$ (207), y se almacenan en una cadena llamada MURO$. La línea 40 imprime MURO$ que aparece en la pantalla como un bloque sólido. La última línea (200) sirve únicamente para mantener el dibujo en pantalla.

A continuación ponemos las almenas con bloques alternativos en negro y beige. Solo necesitamos una línea esta vez.

```
50 FOR I = 1 TO 16: B$ = B$ + CHR$ (128) + CHR4 (207): NEXT I
60 PRINT @ 224, B$;
```

La línea 50 construye las almenas y la 60 las imprime sobre el muro.

Ahora necesitamos una torre. La torre se construye con el mismo material que el muro, asi que tomemos algunos ladrillos de MURO$.

```
7Ø P = 11
8Ø FOR I = 1 TO 3: PRINT @ 128 + 32 * I + P , LEFT$ (MURO$, 1Ø); :NEXT I
```

La línea 8Ø toma 1Ø ladrillos de MURO$ y construye 3 filas en mitad del muro. El valor de P sitúa la torre en el medio. Si quiere poner la torre en otra posición, cambie el valor de P. La torre tiene un ancho de 1Ø bloques, por lo tanto P puede tomar cualquier valor entre Ø y 22.

Construimos ahora las almenas de la torre.

```
9Ø PRINT @ 128 + P, LEFT$ (B$, 1Ø);
```

Esto completa nuestro castillo en BASIC. Podemos poner hendiduras utilizando otro carácter y sobreimprimiendo.

```
1ØØ FOR I = 2 TO 32 STEP 4: PRINT @ 288 + I, CHR$ (2Ø6); : NEXT I
```

Necesitamos además una puerta, asi que imprimimos bloques negros en los lugares apropiados.

```
11Ø P$ = CHR$ (128) + CHR$ (128) + CHR$ (128)
12Ø FOR I = 353 TO 417 STEP 32: PRINT @ I + P + 5, P$;:NEXT I
```

La puerta se almacena en P$ y se imprime en la línea 12Ø (utilizamos la P en la expresión PRINT @ para mantener la puerta alineada con la torre)

Un castillo con la puerta siempre abierta no es de mucha utilidad, asi que necesitaremos unas rejas, usando otro carácter (142 + 112), podemos dibujar algo parecido.

```
13Ø R$ = CHR$ (254) + CHR$ (254) + CHR$ (254)
14Ø FOR I = 353 TO 417 STEP 32: PRINT @ I + P + 5, R$;
15Ø FOR K = 1 TO 3ØØ: NEXT K, I
```

La línea 13Ø construye una reja (!), que se imprime de arriba a abajo en la línea 14Ø. El bucle de retardo K hace que descienda lentamente.

Dejaremos ahora el castillo, puede ponerle más detalles, como un foso azul y una bandera en la torre.

MOVIENDO FIGURAS

En el ejemplo del castillo, las líneas 14Ø y 15Ø nos muestran como obtener movimiento en un dibujo, mediante la impresión sucesiva de trozos. Este tipo de movimientos está limitado principalmente al acto de abrir y cerrar puertas. Un método mucho mejor es imprimir la imagen por completo, borrarla y volverla a imprimir ligeramente desplazada. Como está continuamente dibujando la figura; la parte que dibuja deberá ser una subrutina. Haremos ahora una figura, la construiremos en un bloque de 3×4. La línea superior será la cabeza, la siguiente el cuerpo, y la última las piernas.

```
1Ø CLEAR 5ØØ: CLSØ
2Ø M1$ = CHR$ (128) + CHR$ (193) + CHR$ (194) + CHR$ (128)
3Ø M2$ = CHR$ (196) + CHR$ (2Ø7) + CHR$ (2Ø7) + CHR$ (2ØØ)
4Ø M3$ = CHR$ (128) + CHR$ (2Ø2) + CHR$ (197) + CHR$ (128)
```

La figura está almacenada en tres variables de cadena M1$, M2$, M3$. Ahora haremos la subrutina que imprime las cadenas en el orden correcto:

```
5ØØ P = 32 * Y + X
51Ø PRINT @ P, M1$;:PRINT @ P + 32, M2$;
52Ø PRINT @ P + 64, M3$;:RETURN
```

Esto imprimirá la figura como tres líneas, una debajo de otra, comenzando en un el punto indicado por X e Y. (X de Ø a 31 horizontalmente e Y de Ø a 15 verticalmente). Antes de que funcione el programa, deberá introducir el resto de las líneas hasta la 16Ø.

Para mover un carácter necesitamos cambiar la posición de impresión, o sea X e Y. Esto se puede hacer desde el teclado. Utilizaremos INKEY$ para leer del teclado, y los caracteres lógicos para provocar el movimiento son las teclas con flechas. Estas teclas tienen también códigos:

[←] CHR$ (8)
[→] CHR$ (9)
[∨] CHR$ (1Ø)
[∧] CHR$ (94)

Utilizaremos X para mantener la posición horizontal de la figura, e Y para mantener su posición vertical. Si pulsamos la tecla [←] queremos movernos hacia la izquierda, luego decrementaremos en una unidad el valor de X. Pero deberemos tener cuidado de no salirnos del extremo de la pantalla.

```
90 GOSUB 500
100 A$ = INKEY$: IF A$ = ""THEN 100
120 IF A$ = CHR$ (8) THEN X = X—1: IF X < 0 THEN X = 0
```

La línea 100 muestrea el teclado hasta que se pulse una tecla. Si es [←], entonces la línea 120 decrementa X, mira a ver si nos hemos salido de la pantalla, si es asi, detiene el movimiento en el extremo izquierdo. Para movernos hacia la derecha, arriba y abajo el método es parecido.

```
130 IF A$ = CHR$ (9) THEN X = X + 1: IF X >28 THEN X = 28
140 IF A$ = CHR$ (94) THEN Y = Y—1: IF Y< 0 THEN Y = 0
150 IF A$ = CHR$ (10) THEN Y = Y + 1: IF Y >13 THEN Y = 13
160 GOSUB 500: GOTO 100
```

En la línea 130 fijamos que el máximo valor posible para X sea 28. De todos modos no podría ser mayor de 31, y recuerde que nuestra figura tiene 4 bloques de ancho. Lo mismo ocurre con Y, debemos dejar espacio para imprimir las tres líneas. La línea 160 salta a la subrutina de impresión, después vuelve a mirar si se ha pulsado una tecla. Si ejecuta ahora el programa, podrá mover la figura alrededor de la pantalla, pero se organiza un buen jaleo al no haber eliminado la figura de su antigua posición. Para hacer esto, necesitamos una cadena que borre y una subrutina que la imprima.

```
50 BL$ = CHR$ (128) + CHR$ (128) + CHR$ (128) + CHR$ (128)

600 P = 32*Y + X: PRINT @ P, BL$;
610 PRINT @ P + 32, BL$;:PRINT @ P + 64, BL$;
620 RETURN
```

La nueva subrutina (600) hace exactamente lo mismo que la otra, salvo que ahora imprime un bloque de cuadros negros. Lo único que nos queda por hacer es borrar la figura antes de moverla.

```
110 GOSUB 600
```

Ahora ya debe poder mover la figura a cualquier punto de la pantalla. En su forma actual este programa no es más que un ejemplo; pero el movimiento de figuras se puede incorporar a muchos juegos y programas educativos infantiles.

OTRA RESOLUCION

Veremos a continuación el siguiente nivel de resolución. Este tiene una definición de 32×64 puntos, lo cual supone 2048 puntos en la pantalla. Este nivel y el anterior de 16×32 son los de menor resolución, y se pueden utilizar juntos. Para activar y desactivar los puntos en esta pantalla existen dos comandos:

SET (X, Y, C) y **RESET (X, Y)**

El comando **SET activa el punto X, Y en el color C.** La X (de Ø a 63) y la Y (de Ø a 31), son las posiciones horizontal y vertical. El número C indica el color y debe estar entre Ø y 8.

El comando RESET desactiva el punto X, Y. Utilizando este comando se puede conseguir la sensación de movimiento activando y desactivando por orden. Pruebe el siguiente programa.

```
1Ø CLS: X1 = Ø: Y1 = Ø: XI = 2: YI = 2
2Ø X2 = X1 + XI: IF X2 > 63 OR X2 < Ø THEN XI = —XI: SOUND 18Ø, 1: GOTO 2Ø
3Ø Y2 = Y1 + YI: IF Y2 > 31 OR Y2 < Ø THEN YI = —YI: SOUND 18Ø, 1: GOTO 3Ø
4Ø SET (X2, Y2, 8): RESET (X1, Y1): X1 = X2: Y1 = Y2: GOTO 2Ø
```

Puede ver lo que hace; cómo lo hace? Comenzando por un punto X1, Y1 el programa incrementa X1 en una cantidad XI e Y1 en YI, para crear un nuevo punto X2, Y2. En la línea 4Ø se activa el nuevo punto y se desactiva el antiguo (X1, Y1). El punto X2, Y2 se convierte en el antiguo y el programa vuelve a la línea 2Ø para crear otro X2, Y2. Esto hace que se mueva la bola a lo largo de la pantalla; cuando esta alcanza el extremo de la pantalla, se cambia el signo del incremento. Esto significa que X (por ejemplo) comienza ahora a decrementarse, haciendo que cambie la dirección. Esto provoca el rebote en los extremos. Si cambia el incremento (XI y YI en la línea 1Ø), puede hacer que la bola se mueva a diferentes velocidades. Este tipo de programa forma la base de la mayoría de los juegos de pelota con ordenador, pero normalmente se escriben en lenguaje máquina no en BASIC.

Otra utilidad del movimiento de puntos está en los juegos de tiro. Este tipo de juegos requiere moverse a lo largo de la pantalla, y la habilidad de poder disparar un arma. Podemos utilizar las teclas con flechas como antes, pero una forma mucho más útil es mediante el **Joystick.**

Los joysticks se conectan a un lado del ordenador, y permiten un control del movimiento mucho mejor que las flechas. La posición del joystick se lee mediante la función JOYSTK. **JOYSTK (Ø) devuelve la posición horizontal del joystick izquierdo, y JOYSTK (1) su posición vertical. JOYSTK (2) y JOYSTK (3) hacen lo mismo con el joystick derecho.** Puesto que el valor devuelto por la función en cada caso está entre Ø y 63, será necesario hacer su escalado para ajustarlo al nivel de resolución con el que esté trabajando.

SET

El comando SET se utiliza para activar con un color determinado un punto de la pantalla, cuando se trabaja en baja resolución.

SET (X, Y, C)

X e Y son las cordenadas del punto. X deberá estar comprendido entre Ø y 63, e Y entre Ø y 31.

C es el código del color deseado. Deberá ser un número entre Ø y 8.

```
1Ø CLSØ: SET (5, 27, 8): SET (6, 27, 8)
2Ø FOR X = Ø TO 6: FOR Y = 28 TO 3Ø
3Ø SET (X, Y, 8): NEXT Y, X
4Ø FOR X = 7 TO 63: FOR D = 1 TO 2ØØ: NEXT D
5Ø FOR Y = 27 TO 3Ø: IF Y = 27 THN RESET (X—2, Y)
6Ø SET (X, Y, 8): RESET (X—7, Y): NEXT Y, X
7Ø GOTO 7Ø
```

RESET

El comando RESET se utiliza para borrar un punto activado con el comando SET. Se utiliza en baja resolución de gráficos.

RESET (X, Y)

X, Y son las cordenadas del punto a desactivar. X deberá estar entre Ø y 63, e Y entre Ø y 31.

El punto se pone del color del fondo, provocando el borrado de lo que hubiera. Para ver un ejemplo vea SET.

```
10 CLS0: FOR I = 0 TO 3
20 PRINT @ 74 + 3*I, "JOYSTK (";I;")      ";JOYSTK (I);
30 NEXTI: FOR D = 1 TO 400: NEXT D: GOTO 10
```

Ejecute el programa anterior y mueva los joysticks. Verá como cambian los valores dependiendo de la posición de estos. Puede también utilizar el botón de los joystick. Agregue la línea:

```
25 P = PEEK (65280): PRINT @ 202, "VALOR DEL BOTON      "; P
```

La función PEEK dice el ordenador que mire en una posición determinada de su memoria. La dirección de memoria 65280 contiene el resultado de chequear el botón. Por ahora tendrá 127 ó 255. Si pulsa el botón izquierdo cambiará a 125 ó 253. si pulsa el derecho cambiará a 126 ó 254. (Si se pulsan simultáneamente, el número será 124 ó 252).

Comencemos pues a trabajar en un programa de juegos — una batalla entre dos naves espaciales. Utilizaremos los joysticks para mover las naves y los botones para disparar.

Empecemos por las naves; utilizaremos un método similar al usado para construir la figura en el último ejemplo. Cada nave será un bloque de 2×3, una amarilla y la otra azul, almacenadas en las matrices S$ y S2$.

```
10 CLEAR 500: FOR I = 0 TO 5: READ S (I): NEXT I
20 DATA 128, 131, 128, 134, 140, 137
30 FOR Y = 0 TO 1: C = (Y + 1)*16
40 S$ (Y) = CHR$ (S (0) + C) + CHR$ (S (1) + C) + CHR$ (S (2) + C)
50 S2$ (Y) = CHR$ (S (3) + C) + CHR$ (S (4) + C) + CHR$ (S (5) + C)
60 NEXT Y
```

Si no le gusta la forma de las naves, haga sus propias naves cambiando los datos de la sentencia DATA en la línea 20.

A continuación deberemos leer la posición de los joysticks, comprobar que se encuentran en los límites de la pantalla y mirar donde posicionar las naves.

```
70  FOR Y = 0 TO 1: A (Y) = JOYSTK (Y*2):
80  B (Y) = INT (JOYSTK (1 + Y*2)/2)
85  IF A (Y) > 58 THEN A (Y) = 58
90  IF A (Y) < 2 THEN A (Y) = 2
100 IF B (Y) > 27 THEN B (Y) = 27
110 L (Y) = INT (B (Y)/2)*32 + INT (A (Y)/2): NEXT Y
```

Las posiciones de los joysticks son leídas por turno en la línea 7Ø. Los límites se fijan en las líneas 8Ø—1ØØ (recuerde que hay que incluir las naves). El resultado final se convierte después en un valor para el comando PRINT @. (Este es un ejemplo de combinación de los dos niveles menores de resolución—los joysticks trabajan en uno, los comandos PRINT @ en el otro).

Ahora necesitamos imprimir las naves y volver a mirar si se han movido los joysticks.

```
12Ø CLSØ: FOR Y = Ø TO 1: PRINT @ Ø, Z (Ø); :PRINT @ 26, Z (1);
13Ø PRINT @ L (Y), S$ (Y); :PRINT @ L (Y) + 32, S2$ (Y); : NEXTY

17Ø A$ = INKEY$: IF A$ = "" THEN 7Ø
18Ø CLS: END
```

La línea 12Ø también imprime la puntuación (todavía no lo hemos generado). Para terminar el programa, pulse cualquier tecla si no irá a la línea 7Ø y leerá de nuevo la posición de los joysticks.

Ejecute lo que va de programa y asegúrese que las naves pueden moverse a cualquier lugar de la pantalla.

El siguiente paso es disparar las armas y mostrar el rastro del disparo. Esta es la parte más difícil, ya que debe leer los botones de los joysticks y decidir quien está disparando. Además como las naves se pueden mover a cualquier punto de la pantalla debemos averiguar la dirección del disparo. Para hacerlo más simple, solo permitiremos que el disparo siga la línea horizontal desde la nave que disparó hacia la nave contraria.

```
14Ø P = PEEK (6528Ø)
15Ø IF P = 125 OR P = 253 THEN F = Ø: T = 1: GOSUB 2ØØ
16Ø IF P = 126 OR P = 254 THEN F = 1: T = Ø: GOSUB 2ØØ
```

Estas líneas leen los botones y deciden cuál de las naves está disparando. El disparo lo presentará la subrutina 2ØØ.

```
2Ø V1 = B (F): H1 = A (F): H2 = A (T): ST = 1
21Ø IF H1 > H2 THEN ST = —1
22Ø FOR H = H1 ST * 5 TO H2 + 2 STEP ST

24Ø SET (H, V1, 4): SOUND 2ØØ, 1: RESET (H—2 * ST, V1)
25Ø NEXT H: RETURN
```

Fíjese en como la instrucción 21Ø conmuta cuando se invierten las posiciones izquierda y derecha. El movimiento del disparo se presenta en la línea 24Ø.

Lo único que nos falta ahora es comprobar si el disparo ha dado en el blanco, si es asi se generarán los sonidos adecuados y se anotará la puntuación.

La función POINT se utiliza para comprobar un acierto. El formato es POINT (X, Y), donde X, Y es el punto que desea examinar. La función devuelve Ø si el punto está desactivo si no, devuelve el código del color.

Ya que la pantalla está en negro, y estamos disparando en la dirección adecuada (eso esperamos), solo tenemos que comprobar si hay algún punto que esté activo en la línea de disparo. Si metemos la siguiente línea en nuestra subrutina,

23Ø IF POINT (H, V1) > Ø THEN GOSUB 3ØØ: RETURN

si se produce un acierto saltará a la subrutina 3ØØ, (donde mantenemos la puntuación, etc.), cuando retorna no tiene objeto seguir el disparo, asi que dejamos la subrutina y volvemos a comenzar.

3ØØ Z (F) = Z (F) + 1
31Ø FOR K = 1 TO 15: I = RND (5) —2: J = RND (4) —2
32Ø SET (H + I* ST, V1 + J, 8): SOUND (RND (95)), 1
33Ø NEXT K: RETURN

Esta subrutina mantiene la puntuación, dibuja y simula el sonido de la explosión.

Aún con solo 28 líneas, este programa es suficiente para hacer un juego que tiene bastante movimiento. Le dejamos como ejercicio el desarrollar nuevos refinamientos al programa.

Este ha sido un capítulo largo y denso, pero que contiene la mayoría de los elementos necesarios para desarrollar gráficos con su ordenador en cualquiera de los niveles de resolución.

CAPITULO OCHO

MOVIENDONOS A UN NIVEL SUPERIOR

Pasamos a continuación a ver figuras en alta resolución, completamente aparte de la baja resolución. Los dos niveles de baja resolución se pueden utilizar juntos, y se muestran en lo que se conoce como la "pantalla para texto". Los niveles altos de resolución no se pueden mezclar con texto de pantalla. Podrá conmutar entre uno y otro, pero no podrá escribir texto cuando está en alto nivel de resolución y viceversa, no podrá dibujar alta resolución con la pantalla para textos.
Cuando se dibuja cualquier cosa en alta resolución, el ordenador escribe las instrucciones de como presentar la información en una parte especial de su memoria llamada 'RAM para video'. Esta memoria se lee y se pasa al televisor donde se transforma en imágenes. Unas cuantas páginas de la memoria RAM están reservadas para este propósito, normalmente cuatro. A medida que incremente el nivel de detalle utilizado, aumentarán el número de instrucciones necesarias para presentar el resultado. Cuantas más instrucciones haya, hará falta reservar más memoria para almacenarlas, es decir más páginas. Esto se hace mediante el comando PCLEAR, seguido del número de páginas que quiere reservar (hasta un máximo de 8).

PCLEAR 8

Puesto que cada página ocupa 1536 posiciones de memoria, reserve solo el espacio estrictamente necesario. La cantidad de memoria disponible es fija, luego cuanto más espacio asigne para páginas de gráficos, menos espacio le quedará para líneas de programa. PCLEAR se comporta de forma parecida a CLEAR y debe ser utilizada al principio del programa.

MODOS

La cantidad de espacio que necesita reservar depende del nivel de resolución que vaya a utilizar. Una desventaja de la alta resolución es que no puede utilizar todos los colores que tenía disponibles en baja resolución. Los colores disponibles y la resolución vienen determinados por el modo en el que esté trabajando. El modo se fija mediante el comando PMODE,

PMODE modo, página de comienzo

donde "modo" es un número entre Ø y 4, y "página de comienzo" es la página, en la RAM para video en la que desea comenzar a escribir. Como antes, la pantalla está dividida en celdas. Esta vez, sin embargo, solo es necesario acordarse de una dimensión (256×192). Aunque el nivel de resolución varía con el modo.

PCLEAR

PCLEAR se utiliza para reservar páginas para gráficos en los modos de alta resolución.

PCLEAR n

n debe ser un número entre 1 y 8. Si se omite la sentencia PCLEAR se toma 4 por defecto.

Puesto que cada página para gráficos utiliza 1536 bytes de memoria, reserve solo el espacio imprescindible.

NUMERO DE MODO	DIMENSION DEL PANEL	DIMENSION DEL PUNTO	PAGINAS USADAS	JUEGO DE COLORES DISPONIBLES	
				PANTALLA 1.Ø	PANTALLA 1.1
Ø	128 × 96	■■ ■■	1	Negro (Ø), Verde (1)	Negro (Ø), beige (5)
1	128 × 96	■■ ■■	2	Verde (1), Amarillo (2) Azul (3), Rojo (4)	Beige (5), Turquesa (6) Magenta (7), Naranja (8)
2	192 × 128	■ ■	2	Negro (Ø), Verde (1)	Negro (Ø), Beige (5)
3	192 × 128	■ ■	4	Verde (1), Amarillo (2) Azul (3), Rojo (4)	Beige (5), Turquesa (6) Magenta (7), Naranja (8)
4	256 × 192	■	4	Negro (Ø), Verde (1)	Negro (Ø), Beige (5)

SCREEN

El comando SCREEN se utiliza para conmutar la pantalla entre los modos gráfico y de texto.

SCREEN tipo, juego de colores

El tipo puede ser: Ø para texto y gráficos en baja resolución. ó 1 para gráficos en alta resolución.

El juego de colores es Ø ó 1. El juego de colores para la pantalla de textos es Ø, (negro sobre verde) ó 1 (negro sobre naranja). Para gráficos en alta resolución el juego de colores depende del modo en que estemos trabajando:

PMODE	PANTALLA 1,Ø	PANTALLA 1,1
Ø	Negro, Verde	Negro, Beige
1	Verde, Amarillo	Beige, Verde
	Azul, Rojo	Magenta, Naranja
2	Negro, Verde	Negro, Beige
3	Verde, Amarillo	Beige, Turquesa
	Azul, Rojo	Magenta, Naranja
4	Negro, Verde	Negro, Beige

seguirá refiriéndose a los puntos de la pantalla utilizando el panel de 256×192. La diferencia estriba en la dimensión del punto dibujado. El modo seleccionado, también decide los colores que se pueden usar. Cada modo tiene dos juegos posibles de colores. El conjunto de colores se define mediante el comando SCREEN, que también selecciona el tipo de imagen.

SCREEN tipo, juego de colores

"tipo" es Ø para pantalla de textos y 1 para alta resolución. El "juego de colores" también puede ser Ø ó 1. El valor tomado por defecto, que es el que hemos estado utilizando hasta ahora es SCREEN Ø,Ø. Este fija la pantalla en modo texto con color negro sobre fondo verde. (También se puede utilizar SCREEN Ø, 1, que nos da el texto en negro sobre un fondo naranja, pero cada vez que el ordenador imprima cambiará a negro sobre verde). Para fijar la imagen en alta resolución deberá utilizar el tipo 1. La tabla de la otra página muestra los colores y modos disponibles.

Como puede ver en la tabla, los colores y la resolución están íntimamente relacionados. También notará que a medida que aumenta el modo, también lo hace el número de páginas necesarias.

Asi que para presentar una pantalla de gráficos con PMODE Ø, solo necesita una página de memoria, mientras que PMODE 3 y PMODE 4 necesitan 4 páginas. Cuando se ha seleccionado el juego de colores, el ordenador selecciona como color de fondo aquél con el número inferior. El color con mayor número se tomará como color en primer término. Por ejemplo, con PMODE 3 y SCREEN 1,Ø, el ordenador dibujará en rojo sobre fondo verde. Puede cambiar el color en primer término y de fondo con el comando COLOR.

COLOR primer término, fondo

ANTIGUOS CONOCIDOS

Recordará de las imágenes en baja resolución los comandos **CLS, SET, RESET** y **POINT.** Sus equivalentes en alta resolución son **PCLS, PSET, PRESET** y **PPOINT,** y tienen la misma función que antes, **PCLS** limpia la pantalla en alta resolución, y si está seguido por un código de color lo utilizará como color de fondo. **PSET** activa un punto, y **PRESET** lo desactiva. **PPOINT** comprueba si un punto está o no activo. El siguiente

ejemplo trabaja, por orden, en todos los modos y colores posibles. Fija puntos de la pantalla con colores aleatorios y estarán sobre un panel rectangular. Los lugares de la pantalla que están en blanco son debidos a que su color coincide con el del fondo, o no están disponibles en el juego de colores.

```
10 FOR P = 0 TO 4: PMODE P, 1
20 FOR S = 0 TO 1: SCREEN 1, S
30 PCLS: FOR I = 50 TO 150 STEP 20
40 FOR J = 50 TO 150 STEP 20
50 C = RND (8): PSET (I, J, C): NEXT J, I
60 FOR D = 1 TO 1000: NEXT D, S, P
```

Mire detenidamente la dimensión del punto, esta es la resolución disponible en ese modo.

DIBUJE UNA LINEA

Asi que podemos dibujar puntos en la pantalla, y ahora qué? Lo más evidente que podemos hacer con dos puntos es unirlos mediante una línea. Afortunadamente, hay un comando que lo hace, LINE. Borre las líneas 40 y 50 del programa anterior y modifique la línea 30 para que ponga,

```
30 PCLS: LINE (10, 180)—(245, 10), PSET
```

y ejecute el programa. Se dibujará una línea a través de la pantalla desde el extremo superior izquierdo al inferior derecho. La sentencia significa: dibuja una línea desde el punto de comienzo (10, 180) hasta el punto final (245, 10) con el color (PSET). Si cambia PSET por PRESET la línea se dibujará en el color de fondo. Dibujar con el color de fondo supone que no se podrá ver, se puede también utilizar para borrar la línea previamente dibujada. PSET y PRESET forman parte esencial del comando LINE, y aqui no tienen nada que ver con el hecho de activar o desactivar puntos.

No es siempre necesario especificar el comienzo de una línea. Si no se especifica el punto de comienzo la línea comenzará en el punto terminal más reciente. (Si no se ha utilizado todavía una sentencia LINE en el programa, se toma como último punto el 128,96 que es el centro de la pantalla). Agregue otra línea al ejemplo anterior.

```
40 LINE — (130, 180), PSET
```

Se dibujará una línea desde el punto final anterior (245,10) hasta un punto en la parte inferior de la pantalla (130, 180).

COLOR

El comando COLOR se usa para cambiar los colores utilizados como fondo y en primer término cuando se trabaja en el modo de gráficos en alta resolución.

COLOR en primer término, fondo

Tanto "en primer término" como "fondo" son números entre Ø y 8, que representan el código del color. Ambos colores deben pertenecer al conjunto de colores posibles dentro del modo actual.

PCLS

El comando PCLS se utiliza en alta resolución, para borrar la pantalla y dejarla con el color de fondo especificado.

PCLS c

c es el código del color de fondo deseado. Deberá ser uno de los colores disponibles dentro del conjunto del modo de trabajo; si no es asi o se omite c se tomará el color de fondo por defecto.

Vea CLS para los códigos de color.

PSET

Es la versión de SET para alta resolución.

PSET (x, y, c)

activa el punto (x, y) con el color c. x deberá estar comprendido entre Ø y 255, e y entre Ø y 191.

c es el código de color entre Ø y 8, y deberá estar dentro del conjunto de colores disponibles.

PRESET

Es la versión en alta resolución del comando RESET.

PRESET (x, y)

desactiva el punto (x, y), (lo vuelve del color del fondo). x deberá estar comprendido entre Ø y 255 e y entre Ø y 191.

Para dibujar un cuadro o un rectángulo, puede utilizar cuatro líneas, pero existe una extensión del comando LINE que lo hace. Utilice el EDITOR para agregar B a la línea 3Ø, que ahora dirá:

3Ø PCLS: LINE (1Ø, 18Ø)—(245, 1Ø), PSET, B

En lugar de una diagonal obtendrá ahora un rectángulo. Para dibujar un rectángulo todo lo que necesita es especificar la posición de dos vértices opuestos y agregar B a la sentencia LINE. Vuelva a editar y agregue F al final de la línea 3Ø.

3Ø PCLS: LINE (1Ø, 18Ø)—(245, 1Ø), PSET, BF

La F indica que se rellene el rectángulo, con el color principal. Un comando tan flexible puede ser de gran utilidad. Probémoslo.

Empezaremos como antes, en el negocio de la construcción, pero esta vez dibujaremos una casa. Ejecute el programa después de cada sección para que vea como se va construyendo.

Primero fijamos el nivel de resolución y dibujamos el cuerpo de la casa.

1Ø PMODE 3, 1: SCREEN 1, Ø: PCLS
2Ø LINE (6Ø, 48)—(2ØØ, 144), PSET, B
26Ø GOTO 26Ø

A continuación ponemos el tejado.

4Ø LINE (6Ø, 48)—(13Ø, 2Ø), PSET
5Ø LINE—(2ØØ, 48), PSET

ahora, un garaje con una puerta.

7Ø LINE (2ØØ, 144)—(255, 94), PSET, B
9Ø LINE (21Ø, 144)—(245, 1Ø4), PSET, BF

Utilizaremos la misma técnica para poner una puerta en la casa

1ØØ LINE (16Ø, 144) — (188, 1Ø5), PSET, BF

Para dibujar ventanas necesitamos un rectángulo con dos líneas que lo crucen.

11Ø LINE (85, 132) — (135, 1Ø8), PSET, B
12Ø LINE (11Ø, 1Ø8) — (11Ø, 132), PSET
13Ø LINE (85, 12Ø) — (135, 12Ø), PSET

Las ventanas del segundo piso se hacen igual,

14Ø LINE (9Ø, 84) — (125, 64), PSET, B
15Ø LINE (9Ø, 74) — (125, 74), PSET
16Ø LINE (11Ø, 84) — (11Ø, 64), PSET

LINE

El comando LINE se utiliza para dibujar líneas y rectángulos en el modo de alta resolución de gráficos.

LINE (x1, y1) — (x2, y2), a, b

x1 e y1 son las coordenadas del punto de comienzo de la línea.
x2 e y2 son las coordenadas del extremo final de la línea.
"a" puede ser PSET o PRESET. Si se utiliza PSET se dibujará una línea en el color utilizado en primer término. Si es PRESET la línea se dibujará en el color utilizado para el fondo. "b" es un parámetro opcional; si se utiliza, puede ser B o BF. Si se pone B se dibujará un rectángulo en lugar de la línea, la esquina superior del rectángulo será x1, y1 y la esquina inferior izquierda será x2, y2. Si se pone BF, se dibujará el rectángulo y además se pintará con el color utilizado en primer término.

```
10 PMODE 4, 1: SCREEN 1, 1: PCLS 5: COLOR 0,5
20 FOR I = 1 TO 1000
30 X = X + L * SIN (R): Y = Y + L * COS (R)
40 IF X < —128 OR X > 128 THEN 90
50 IF Y < —96 OR Y > 95 THEN 90
60 LINE — (X + 128, Y + 96), PSET
70 R1 = R1 + 60: R = R1 /57.29578: L = L + 0.5
80 NEXT I
90 GOTO 90
```

17Ø LINE (155, 64) — (175, 84), PSET, B
18Ø LINE (165, 84) — (165, 64), PSET
19Ø LINE (155, 74) — (175, 74), PSET

y para completar la estructura, pongamos una chimenea,

2ØØ LINE (15Ø, 4Ø) — (16Ø, 15), PSET, BF

Este programa muestra lo rápido que se puede dibujar una figura, y con un solo comando. Hay por supuesto que conocer los puntos desde donde se van a trazar las líneas: la forma más fácil para conocerlos es sacar una copia de la hoja para gráficos del apéndice B, y hacer en ella un esquema del dibujo.

UNA PINCELADA DE COLOR

Nuestra casa está un poco apagada, lo que necesita es una mano de pintura que la haga resaltar. Asi que le diremos al DRAGON que saque las brochas y se ponga a trabajar. El comando PAINT le permite pintar cualquier forma con cualquier color disponible. Todo lo que tiene que hacer es indicar donde hay que comenzar, con que color pintar y el color del borde donde hay que parar.

PAINT (x, y), a, b

donde x, y son las cordenadas del comienzo, y a y b son los códigos de color de la pintura y del borde. Ponga la siguiente línea al ejemplo de la casa,

3Ø PAINT (9Ø, 9Ø), 2, 4

Esto indica que empezando en el punto (9Ø, 9Ø), pinta de amarillo (color 2), hasta que te encuentres un borde rojo (4). Ejecute el programa para ver que pasa. Ahora borre la línea e introdúzcala como línea 195 y ejecute otra vez.

195 PAINT (9Ø, 9Ø), 2, 4

Fíjese como ahora se detiene en los bordes de las ventanas que antes no estaban. Pinte el garaje del mismo modo,

8Ø PAINT (21Ø, 14Ø), 2, 4

Ahora vamos con el tejado. Si omite poner el color o el borde en una sentencia PAINT, se tomará para ambos el color actual utilizado en primer término.

6Ø PAINT (13Ø , 25)

Terminaremos dibujando la línea del horizonte y el cielo.

21Ø LINE (6Ø, 64) — (6Ø, 64), PSET
22Ø LINE (2ØØ, 64) — (255 , 64), PSET
23Ø PAINT (Ø, 54), 3, 4

PAINT

El comando PAINT se utiliza en el modo de alta resolución de gráficos, para rellenar una forma con un color determinado.

PAINT (x, y), c, b

x, y son las coordenadas donde se comienza a pintar.

c es el código del color que se va a utilizar para pintar. Deberá estar entre Ø y 8 y ser uno de los colores disponibles en el modo de trabajo actual. Si se omite, se utilizará el color actual utilizado en primer término.

b es el código del color del borde donde se debe detener la pintada. También deberá estar comprendido entre 1 y 8, la pintada continuará sobre los bordes de cualquier otro color. Si se omite se tomará el color actualmente utilizado en primer término.

Vea CIRCLE para un ejemplo de su utilización.

Ahora ya está todo un poco más brillante. Puede seguir pintando más cosas, por ejemplo una verja.

CIRCULOS

Ya tenemos líneas, cuadrados y rectángulos y ahora círculos. La sentencia CIRCLE dibujará círculos, elipses y arcos.

CIRCLE (x, y), radio, color, haradio, comienzo, fin

El punto x, y es el centro del círculo, el radio es el radio del círculo medido en puntos de pantalla. El "color" es uno de los disponibles en el modo en que se está trabajando, (si se omite, se utilizará el color utilizado actualmente). Los otros parámetros son para dibujar elipses y arcos, trataremos de ellos más tarde. Primero veamos que pasa con los círculos.

```
10 FOR P = 0 TO 4: PMODE P, 1
20 SCREEN 1, 1: PCLS
30 FOR R = 120 TO 10 STEP —10
40 CIRCLE (128, 96), R: NEXT R
50 FOR D = 1 TO 500: NEXT D, P
```

Esto dibujará círculos alrededor del centro de la pantalla. Los círculos son difíciles de dibujar y para obtener uno muy preciso, deberá trabajar con PMODE 4.

Habrá observado que no hay problemas si el círculo se sale de la pantalla. Si intenta dibujar una línea hasta un punto que se encuentra fuera de la pantalla, puede que no se dibuje, especialmente en los modos de alta resolución. Pruebe insertando

```
42 LINE —(300, 40), PSET
```

y observe el resultado.

El comando PAINT se puede también utilizar para rellenar círculos

```
45 PAINT (128, 96)
```

rellena el centro de la diana.

Mediante el parámetro "haradio" puede cambiar un círculo por una elipse. Haradio significa, relación entre la altura y la anchura del radio. El ancho en el comando CIRCLE siempre se mantiene igual, el doble del radio. La altura se puede variar mediante haradio, si es mayor de 1 entonces el círculo será más alto que ancho. Un valor menor de 1 achatará el círculo en el otro sentido, más ancho que alto.

CIRCLE

El comando CIRCLE dibujará círculos, elipses y arcos. Solo se puede utilizar en los modos de alta resolución.

CIRCLE (x, y), r, c, ha, comienzo, fin

x	es la coordenada X del centro del círculo (de Ø a 255).
y	es la coordenada Y del centro del círculo (de Ø a 191).
r	es el radio del círculo medido en puntos de pantalla.
c	es un código de color (de Ø a 8), deberá ser uno de los colores disponibles actualmente.
ha	es la relación entre la altura y la anchura (entre Ø y 255). Se utiliza para dibujar elipses. Si se omite se toma 1.
comienzo	es el comienzo del arco del círculo (de Ø a 1). La posición Ø representa las 3 horas. Si se omite se usa Ø.
fin	indica el final del arco (de Ø a 1). El dibujo avanza en el sentido de las agujas del reloj desde el comienzo. La posición Ø.5 representa las 9 horas. Si se omite se toma 1.

```
1Ø PMODE 3, 1: SCREEN 1, Ø: PCLS
2Ø CIRCLE (18Ø, 156), 28, 3: PAINT (18Ø, 156), 3, 3
3Ø CIRCLE (11Ø, 156), 28, 3: PAINT (11Ø, 156), 3, 3
4Ø CIRCLE (144, 8Ø), 68, 4, 1, Ø,.5
5Ø LINE (212, 8Ø) — (76, 8Ø), PSET: LINE —(48, 32), PSET
6Ø PAINT (144, 82): CIRCLE (144, 8Ø), 70, 4, .8, .79, 1
7Ø LINE (16Ø, 8Ø) — (16Ø, 28), PSET: PAINT (21Ø, 75)
8Ø GOTO 8Ø
```

Asi pues el ancho sobre el eje X (horizontal) es siempre el mismo, solo cambia la altura del eje Y (vertical). Cuando haradio es Ø el círculo se convierte en una línea horizontal, y con un haradio muy grande se transforma en una línea vertical (realmente es un rectángulo muy delgado). El valor máximo permitido es 255. Cambie las líneas 3Ø y 4Ø del programa para que pongan:

```
3Ø FOR H = Ø.5 TO 3 STEP Ø.5
4Ø CIRCLE (128, 96), 4Ø,,H: NEXT H
```

Fíjese en las comas de más que hay en la línea 4Ø, están porque hemos omitido el parámetro del color.

La última extensión del comando CIRCLE es la habilidad para dibujar arcos (parte de un círculo). Para utilizar esta opción deberá especificar el comienzo y el final del arco. Ambos deberán ser un número entre Ø y 1. El punto de comienzo del círculo es equivalente a las 3 horas en un reloj. El dibujo discurre en el sentido de las agujas del reloj a partir de la posición de comienzo. Por ejemplo, comenzando en Ø.25 y acabando en Ø.75 dibujará desde las 6 hasta las 12 en un reloj. Comenzando en Ø.5 y acabando en 1.Ø dibujará el semicírculo superior. El siguiente programa utiliza arcos para dibujar un modelo.

```
1Ø PMODE 4, 1: SCREEN 1, 1: COLOR Ø,5: PCLS
2Ø FOR R = 15 TO 6Ø STEP 5
3Ø CIRCLE (128, 96 + R), R,, 1, .5, 1
4Ø CIRCLE (128, 96—R),, 1, Ø, .5
5Ø CIRCLE (128—R, 96), R,, 1, .75, .25
6Ø CIRCLE (128 + R, 96), R,, .25, .75
7Ø FOR D = 1 TO 5ØØ: NEXT D
8Ø NEXT R
9Ø GOTO 9Ø
```

CAMBIANDO DE PAGINA

Una manera de dar animación a las figuras es colocar un dibujo ligeramente distinto en cada página y después cambiar las páginas. Recuerde que el número de páginas se fijaba con el comando PCLEAR y el número de la página en la que se escribe viene dado por el segundo parámetro de PMODE. Por supuesto deberá tener en cuenta el nivel de resolución con el que está trabajando. Con PMODE3 y PMODE4 cada pantalla de gráficos requiere 4 páginas asi pues solo tiene sentido cambiar entre la página 1 y la 5. En PMODE1 y PMODE2, que necesitan 2 páginas, podrá cambiar entre 1, 3, 5 y 7. El siguiente ejemplo muestra como se hace esto, pruebe introduciendo todos los valores de PMODE.

```
10 PCLEAR 8: CLS
20 INPUT "MODO"; M: ON M GOTO 40, 40, 50, 50
30 S = 1: GOTO 60
40 S = 2: GOTO 60
50 S = 4
60 FOR P = 1 TO 8 STEP S: PMODE M, : PCLS
70 LINE (128,0) — (128, (P—1) * 15), PSET
80 CIRCLE (128, P * 15), 20: PAINT (128, P * 15): NEXT P
90 FOR P = 1 TO 8 STEP S: GOSUB 150: NEXT P
100 IF M > 2 THEN D = 4: S1 = 3 ELSE D = 7: S1 = S
110 FOR P = D TO 1 STEP —S1: GOSUB 150: NEXT P
120 GOTO 90
150 PMODE M, P: SCREEN 1, 1
160 FOR T + 1 TO 20: NEXT T: RETURN
```

Las líneas 60 a 80 dibujan la figura cambiante en las diferentes páginas. Todo este dibujo ha tenido lugar sin presentarse, ya que no se ha puesto todavía ningún comando SCREEN. El resto del programa muestra cada página por turno, cambiando las páginas primero hacia delante y después hacia atrás para dar la sensación de movimiento. Puede ver que cuantas más páginas utilice mayor uniformidad conseguirá en el movimiento.

Otra forma de construir imágenes es usando el comando PCOPY.

PCOPY página fuente TO página destino

Podrá copiar el contenido de una página en otra cualquiera, suponiendo que la página ha sido anteriormente reservada con PCLEAR. También se puede utilizar PCOPY para empaquetar duplicados en una página en PMODE3 o PMODE4. El siguiente programa muestra como se utiliza PCOPY con este propósito, fíjese en que tiene que tener cuidado de donde sitúa la figura.

```
5 PCLEAR 8
10 PMODE 3, 4: PCLS
20 LINE (100, 20) — (140, 40), PSET, BF
30 CIRCLE (50, 25), 20
40 CIRCLE (200, 50), 20
50 FOR D = 3 TO 1 STEP —1
60 PCOPY 4 TO D: NEXT D
70 FOR P = 4 TO 1 STEP —1: PMODE3, P
80 SCREEN 1, 1: FOR I = 1 TO 1000: NEXT I, P
90 GOTO 90
```

En PMODE3 (y 4), la imagen consta de 4 páginas, la página 1 representa el cuadrante inferior de la pantalla, la página 2 el siguiente y asi sucesivamente. Asi que copiando el contenido de la página 4 en la página 1 habrá duplicado el cuadrante superior en el inferior. Se puede conseguir el mismo efecto para PMODE1 y PMODE2, pero esta vez la pantalla estará dividida en mitades no en cuartos.

Los que estén insatisfechos y quieran continuar con gráficos, pueden saltar al capítulo 1Ø, el resto de nosotros tendremos ahora un interludio musical.

PCOPY

PCOPY es un comando para gráficos en alta resolución, que copia el contenido de una página de gráficos en otra.

PCOPY fuente TO destino

Tanto "fuente" como "destino" deben ser números entre 1 y 8, y deben referirse a páginas previamente reservadas con el comando PCLEAR. El espacio necesario para mantener una figura depende del modo utilizado y deberá tenerse en cuenta al utilizar PCOPY.

PCOPY 3 TO 5

CAPITULO NUEVE

SONIDOS ELECTRONICOS

INTRODUCCION DE UN CAMINO SONORO

Los gráficos y otros programas, pueden resultar más interesantes si les ponemos sonido. Ya hemos utilizado el comando SOUND anteriormente. Una forma más fácil de poner sonido es que usted lo proporcione. No queremos decir que se ponga a cantar con sus programas; su computador utiliza un casete para grabar los programas pero también para hacer funcionar una cinta a petición. Los comandos **MOTOR ON** y **MOTOR OFF** hacen exactamente eso. Estos comandos junto con **AUDIO ON** y **AUDIO OFF** conectarán y desconectarán la salida del casete al altavoz de la televisión. Esto significa que utilizando estas sentencias en su programa podrá poner música de fondo a sus gráficos. O podrá preparar una cinta que presente las preguntas y respuestas en un programa educacional. El siguiente ejemplo muestra lo fácil que es. Si no tiene una cinta a mano, utilice una de las cintas para sus programas. Los ruidos que oirá es como se comunican los ordenadores.

```
10 CLS PRINT @ 128, "PULSE LA BARRA ESPACIADORA PARA"
20 PRINT @ 160, "PARAR O ARRANCAR EL CASETE"
30 A$ = INKEY$: IF A$<> " " THEN 30
40 IF F = 0 THEN MOTOR ON: AUDIO ON: F = 1 ELSE MOTOR OFF:
   AUDIO OFF: F = 0
50 GOTO 30
```

Rebobine la cinta hasta el comienzo y pulse el botón PLAY, después ejecute el programa. Pulsando la barra espaciadora podrá detener o arrancar la reproducción de la cinta.

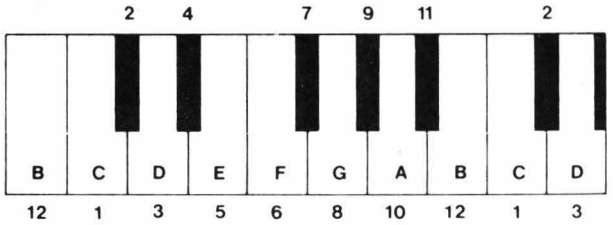
Utilizando este método se puede escribir un programa educacional con preguntas y respuestas que traten cierto número de materias, simplemente cambiando la cinta de preguntas. O podrá poner música o efectos sonoros apropiados a sus animaciones.

TOCA ESA PIEZA

Alternativamente puede hacer que el ordenador toque música. El comando PLAY transforma en sonidos el contenido de una cadena.

PLAY cadena

donde "cadena" puede ser una constante o una cadena variable. No es una cadena cualquiera sino una formada por notas, octavas, duración de las notas, silencios y tiempo. Las notas son evidentemente las notas musicales que desea oir. La forma más sencilla de hacer esto es introducir la letra que representa cada nota musical—A, B, C, D, E, F, G. (Símbolos anglosajones equivalentes a LA, SI, DO, RE, MI, FA y SOL respectivamente). Para indicar un sostenido se utiliza el símbolo # o + (F# o F+ representan el FA sostenido); para los bemoles se utiliza el signo —. El ordenador no reconocerá B# (SI sostenido) o C— (DO bemol) ya que no existen en la escala. Otra forma de introducir una nota es utilizando el número que representa su posición en la escala de 12 tonos.



Las notas y sus números equivalentes están marcados en el teclado superior.

El comando PLAY se puede utilizar como un comando directo, lo cual es útil para comprobar una cadena musical antes de incorporarla a un programa. Como todo buen músico, comenzaremos practicando las escalas.

PLAY "CDEFGABCCBAGFEDC" escala en DO.
PLAY "GABCDEF#GGF#EDCBAG" escala en SOL.

La escala en DO está casi bien, pero la que está en SOL es un jaleo. Esto es debido a que las escalas cambian de octava, y debemos decírselo al ordenador. Para seleccionar la octava, use O seguido por un número entre 1 y 5; O2 (que incluye el DO central) se selecciona automáticamente cuando se enciende el ordenador. Se utilizará la octava actualmente seleccionada mientras no se haga un cambio, asi que es preferible especificar siempre la octava que quiere utilizar al comenzar. Intentemos de nuevo las escalas.

PLAY "O3CDEFGABO4CCO3BAGFEDC"
PLAY "O3GABO4CDEF#GG#EDCO3BAG"

Para tocar la escala de DO usando números en lugar de letras

PLAY "O3; 1; 3; 5; 6; 8; 1Ø; 12; O4; 1; 1; 12; 1Ø; 8; 6; 5; 3; 1"

Fíjese en el separador (;) utilizado en la cadena. Se puede utilizar el punto y coma donde quiera, pero con los números es necesario para evitar confusiones.

AUDIO

El comando AUDIO controla la salida de sonido del casete por el altavoz del televisor. **AUDIO ON** dirige la salida de casete hacia la televisión. **AUDIO OFF** desconecta la comunicación.

MOTOR

El comando MOTOR controla el funcionamiento del motor del casete. **MOTOR ON** arranca el motor, **MOTOR OFF** detiene el motor.

El botón play del casete deberá estar pulsado para que el comando funcione.

Puesto que la cadena de música no deja de ser una cadena, se puede manipular con todas las operaciones para estas. El siguiente ejemplo toca la escala de DO sobre todo el rango del comando PLAY.

```
10 A$ = "CDEFGAB"; FOR I = 1 TO 5
20 B$ = "O" + STR$ (I) + A$
30 PRINT B$: PLAY B$: NEXT I
```

Utilizando la misma técnica y números en lugar de letras podemos tocar la escala cromática completa:

```
10 FOR I = 1 TO 5: A$ = "O" + STR$ (I) + ";"
20 FOR J = 1 TO 12: PLAY A$ + STR$ (J): NEXT J, I
```

En la mayoría de las melodías, las notas raramente tienen la misma duración, luego necesitaremos fijar la duración de cada nota. Esto se realiza con el parámetro de duración de una nota (L). La letra L viene seguida por un número entre 1 y 255. A medida que el número crece la duración decrece, L1 es una nota completa, L2 media nota, L4 un cuarto de nota, y asi sucesivamente. Es posible tener hasta 1/255avo de nota, pero no hay muchos compositores que lo utilicen. Los que sepan algo de música habrán oído hablar de las notas con puntillo. El puntillo indica que incremente la duración de una nota en la mitad de su duración. Para conseguir este efecto con el comando PLAY, ponga un punto (o varios) después del número en el parámetro L.

L4. = 1/4 + 1/8 = 3/8 de una nota

Ya sabemos lo suficiente para interpretar una melodía sencilla. Introduzca lo siguiente con cuidado:

```
5 CLEAR 500
10 A$ = "O2L4GG; L2GDL4BB; L2BGL4GB;
         O3L2DDL4CO2B; L1AL4AB;
         O3L2CCO2L4BA; L2BGL4GB;
         L2ADL4F#A; L1G;"
20 B$ = A$ + A$: PLAY B$
```

Los separadores (;) se utilizan para indicar las líneas divisorias del pentagrama, no son realmente necesarias. Deberá ser capaz de reconocer la melodía "Clementina", aunque está interpretada demasiado despacio. El parámetro del Tempo se encarga de esto, la letra T seguida por un número entre 1 y 255. Cuanto mayor sea el número más rápido se interpretará la melodía. Cambie la línea 20 por.

```
20 B$ = A$ +A$: PLAY "T6" + B$
```

PLAY

El comando PLAY se utiliza para generar una secuencia musical. El argumento es una expresión de cadena, o una constante, o una variable de cadena. Su formato es,

PLAY música

Donde "música" se forma con los siguientes elementos:

Notas Una letra de la "A" a la "G" o un número entre 1 y 12.

Octava "O" seguida de un número entre 1 y 5. Por defecto se toma O2. Los valores por defecto los fija el ordenador al encenderlo.

Duración de la nota "L" seguida de un número entre 1 y 255. Valor por defecto, L4.

Tempo "T" seguida de un número entre 1 y 255. Por defecto T2.

Intensidad "V" seguida por un número entre 1 y 31. Por defecto V15.

Duración de un silencio "P" seguida por un número entre 1 y 255.

Ejecución de subcadenas "X" seguida de una variable de cadena y un punto y coma.

Una nota sostenida se puede indicar por "+" o "#" y un bemol mediante "—".

El parámetro sobre la duración de una nota se puede modificar mediante un punto (.) adicional después del número; (L2.) representa una nota con puntillo.

La octava, intensidad, tempo y duración de una nota se pueden modificar utilizando uno de los siguientes sufijos:

+ Suma 1 al valor actual.
— Resta 1 al valor actual.
> Multiplica por 2 el valor actual.
< Divide por dos el valor actual.

```
10 X$ = "O3L4EF#L4.EL8AAG#ABL4O+C#O—B"
20 A$ = "XX$;O4C#O—AF#O+DC#O—BL2AXX$;
   O+C#DEL8DO—BL< AG#L< AL4.BL8Ø+C#L4
   DO—BL4.O+C#L8DL< EC#L4.EL8EEEEEL1
   EL4.EL8DC#EDO—BL< AG#L< A"
30 PLAY "T2V2Ø" +A$
```

Pruebe cambiando el 6 por otro número que se ajuste mejor a su idea de la velocidad que debe llevar la melodía.

Muchas piezas musicales requieren además colocar silencios en los lugares adecuados y también variar la intensidad de ciertos pasajes. El parámetro para los silencios es P seguida de un número. Los silencios siguen el patrón de la duración de la nota (L), salvo que no puede poner el puntillo después del número. Para introducir un silencio equivalente a L4., se utilizará P4P8. El parámetro de intensidad le permite variar el volumen mediante la inserción de la letra V, seguida por un número entre Ø y 31, a medida que aumenta el número la pieza suena más alta. El siguiente ejemplo utiliza el parámetro de intensidad (volumen) para provocar un crescendo.

```
1Ø A$ = "V1ØO2L4GG; L1GP4V14L4GGG;
        L1GP4V18L4GGG;L2BL4BBBV22L2BL4BBB;
        V26O3L2DL4DDD3L2DL4DDD;
        V3ØL1GL2.F#L4C#;L2EDCO2A
        L1GL2AL4.DL8A;L2B"
2Ø PLAY "T5" +A$
```

A menudo una pieza musical contiene un pasaje que se repite en varios lugares dentro de la pieza, en lugar de escribir el pasaje más de una vez, es preferible introducirlo en una variable separada. El comando X de ejecución de subcadenas, permitirá que esta subcadena aparezca como parte de una secuencia normal de comandos play. La X deberá ir seguida por el nombre de la variable de cadena y un punto y coma, como en:

```
1Ø X$ = "O3L2GBO4C;DL4CO3BAG;"
2Ø Y$ = "L2ADD;L1.A"
3Ø Z$ = "L2ADD;L1.G"
4Ø A$ = "XX$;XY$;XX$;XZ$;L2BGG;O4CO3L4
        BAGF#;XY$;XX$;XZ$;"
5Ø PLAY "T8" + A$
```

Podíamos haber utilizado una subcadena en "Clementina", simplemente cambie la línea 2Ø por:

```
2Ø PLAY "T6XA$; XA$;"
```

El punto y coma deberá seguir al signo $. La utilización del punto y coma solo es indispensable después de subcadenas y al utilizar números (en lugar de letras) como notas.

Hay una opción más que se puede utilizar con los parámetros de intensidad (V), octava (O), tempo (T) y duración de la nota (L). En lugar de poner un número después de la letra, puede utilizar uno de los siguientes sufijos:

+ Suma 1 al valor actual.
— Resta uno al valor actual.
> Multiplica el valor actual por 2.
< Divide por 2 el valor actual.

Nuestro ejemplo sobre escalas lo podemos escribir definitivamente utilizando esta opción,

```
10 PLAY "O1C": FOR I = 1 TO 4: PLAY "DEFGBAO +C": NEXT I
```

De donde podemos obtener la música? Puede desde luego componer sus propias obras, pero para el resto de los mortales duros de oído será mejor conseguir partituras para instrumentos no polifónicos, como la flauta, trompeta etc.

Para aquellos que no sientan inclinación por interpretar melodías con el ordenador no conviene que ignoren por completo el comando PLAY. Los fans de los juegos podrán conseguir con el interesantes efectos sonoros; intente cualquiera de los ejemplos de este capítulo con el parámetro de tempo fijado en 255.

El ejemplo final utiliza toda la moderna tecnología del comando PLAY en una canción con 4 siglos. Enrique quedaría impresionado!

```
10 A$ = "O3L2E;L1GL2AL2.BL4O+C#L20—B;
   L1AL2F#L2,DL4EL2F#;L1GL2EL2.EL4DL2E;
   L1F#V10$L2DVBL.10—BV6L20+E;L1GL2A?2.B
   L40+C#L20—B;L1AL2F#L2.DL4EL2F#;
   L2.GL4F#L2EV8L2.D#V10L4C#V15L2D#;
   L1.EL1EP1;"
20 B$ = "O4L1.DL2.DL4C#O—L2B;L1AL2F#L2.D
   L4EL2F#;L1GL2EL2.EL4DL2E;L1F#L2D
   O—L1BO+L2B;O+L1DL2DL2.DL4C#O—L2B;
   L1AL2F#L2.DL4EL2F#;L2.GV10L4F#L2
   EV6L2.D#L4C#V4L2D#;V15L1.EL2EP1;"
30 PLAY "T10XA$;XB$;XA$;XB$"
```

CAPITULO DIEZ

MAS DETALLES SOBRE GRAFICOS

En el capítulo ocho, vimos como se podían utilizar los comandos LINE y CIRCLE para producir formas regulares como rectángulos, círculos, elipses y arcos. Aunque estos comandos son muy útiles, no podemos realizar fácilmente con ellos formas irregulares. La forma más sencilla de crear estas formas es dibujándolas.

Cuando dibuja una figura en un papel, comienza en un punto determinado y mueve el lápiz un poco hacia arriba, luego hacia la derecha, etc. El comando DRAW le permite realizar este proceso sobre la pantalla.
El formato es,

DRAW cadena

Donde "cadena" es una variable o una constante de cadena, conteniendo un conjunto de los subcomandos de DRAW.

Normalmente la primera acción al dibujar es situarse en el punto de comienzo.

M x, y significa ir al punto dado por las coordenadas x e y; por ejemplo, M128,96 se colocará en el centro de la pantalla. Cuando nos movemos a un punto, a menudo queremos hacer un movimiento en blanco, es decir sin pintar o con el lápiz levantado. Un movimiento en blanco se realizará utilizando la letra B, cualquier instroducción de dibujar que siga a la letra B producirá una línea en blanco. BM128,96 significa: muévete al centro de la pantalla sin dibujar.

Una vez decidido el punto de comienzo, podrá moverse hacia arriba (U), abajo (D), derecha (R), izquierda (L) tantos puntos como quiera. La secuencia U20R20D20L20 hará que se dibuje una línea 20 puntos hacia arriba, después 20 a la derecha, 20 hacia abajo y 20 hacia la izquierda, dibujando un cuadrado. Es hora de ver un ejemplo,

```
10 PMODE 3, 1: PCLS: SCREEN 1, 1

30 DRAW "BM120, 96; U26; R13; D26; L13"
80 GOTO 80
```

El punto y coma de una cadena se utiliza como separador. Realmente no es necesario, los hemos puesto para facilitar la lectura de la cadena. El ejemplo dibuja un rectángulo cerca del centro de la pantalla.

Además de las líneas horizontales y verticales también se pueden dibujar líneas diagonales. Para ello se utilizan los subcomandos E,F,G, y H; E12,

por ejemplo, dibujará una línea diagonal de 12 puntos a 45 grados de la vertical. Todos los ángulos se miden desde la vertical como sigue:

E 45 grados	F 135 grados
G 225 grados	H 315 grados

Esto permite dibujar diagonales en las cuatro direcciones. Agregue la línea,

40 DRAW "L6;U6;E6;BR13;F6;D6;L6;BU26;H6;G6"

a nuestro programa actual y el rectángulo se convertirá en un cohete. El ordenador recuerda su última posición de forma que la l ínea 40 continuará dibujando a partir de ese punto. Estudie la línea 40 para ver como está hecha. La última posición dibujada es el extremo inferior izquierdo del rectángulo y BR13 significa muévete a la derecha 13 puntos sin dibujar.

El cohete se ha dibujado con el color tomado por defecto, pero podemos cambiarlo utilizando C. La letra C seguida de un número entre 0 y 8 representa el código del color entre aquellos disponibles. Utilizando el editor cambie la línea 40 para que ponga,

40 DRAW "C7;L6;U6;E6;BR13;F6;D6;L6;BU26;H6;G6"

y obtendrá un cohete en dos colores. El dibujo se puede pintar exactamente igual que otras formas, pero el comando C cambia el color tomado por defecto en primer término, asi que habrá que tener cuidado para no pintarlo todo.

El dibujo es un poco pequeño, lo agrandaremos con el parámetro S. La S significa escala, y permite escalar un dibujo o parte de el para agrandarlo o reducirlo en 1 /4 de unidad. Asi pues, S1 reduce el dibujo en 1/4 de escala, S2 en 2/4 de escala, S8 en 8 /4, y asi sucesivamente. El valor tomado por defecto para la escala es de 4/4 (es decir, la dimensión original). La S puede ir seguida por cualquier número desde 1 a 62. Introduzca la línea

15 DRAW "S12"

y el cohete tendrá tres veces su dimensión original.

Otra opción posible es el parámetro de ángulo A. Este nos permite girar todo o parte de un dibujo, ya que todas las líneas después de la A se dibujarán giradas según el valor dado por An. n es un número entre 0 y 3, como sigue,

0 0 grados	1 90 grados
2 180 grados	3 270 grados

Altere el programa actual con las siguientes líneas

```
20 FOR I = 0 TO 3: DRAW "A" + STR$ (I): PCLS
50 FOR D = 1 TO 100: NEXT D, I
```

El cohete ahora gira y además cambia de color! Esto es debido a que el ordenador no solo recuerda la última posición sino además los últimos ajustes de C y A. Este problema se puede resolver poniendo C8 al comienzo de la cadena de la línea 30.

La línea 20 muestra que, al igual que con el comando PLAY, las cadenas utilizadas por DRAW pueden utilizarse con funciones de cadena. También, de forma parecida al comando PLAY, puede ejecutar subcadenas con el comando X seguido por una variable de cadena, XA$, y de un punto y coma (;).

```
10 PMODE 3, 1: SCREEN 1, 1: PCLS
20 S$ = "L8E4F4"
30 D$ = "A0; XS$; A1; XS$; A2; XS$; A3; XS$;"
40 DRAW SS "24" + D$
50 GOTO 50
```

En este programa se almacena un triángulo como una subcadena en S$, y después se utiliza en la línea 30 para construir un modelo. Fíjese en que este es el único lugar en donde es esencial el punto y coma, después del signo $.

El último parámetro es N, que indica no actualizar la posición de dibujo. Esto quiere decir que se dibuja una línea pero el final de esta no queda como nueva posición ; NU10L5, dibujará una línea hacia arriba de 10 puntos, volverá al comienzo de la línea y dibujará 5 puntos hacia la derecha (una L).

```
10 PMODE 3, 1: SCREEN 1, 1: PCLS
20 DRAW "BM128 , 96; NU25ND25NR25NL25; NE17NF17NG17NH17"
30 GOTO 30
```

El ejemplo anterior dibujará líneas hacia afuera desde el centro, volviendo siempre a este para dibujar la línea siguiente.

A menudo querrá poner otro dibujo junto al que acaba de hacer. Vd. sabe donde debe de ir el nuevo dibujo respecto del anterior, pero no desea estudiar sus coordenadas. Esto se puede hacer con movimientos relativos, ej. 5 puntos a la derecha y 10 hacia arriba. El comando mover (M) nos permite realizar esto fácilmente todo lo que tiene que hacer es especificar la distancia como positiva o negativa respecto del punto actual, por ej. M+5, −10. Recuerde utilizar la B para evitar líneas indeseadas.

25 DRAW "BM—25, —25; U1ØR25D1ØL25"

Introduzca la línea anterior y se dibujará un rectángulo sobre el dibujo anterior. La última posición era 128,96 debido al parámetro N. Nos hemos movido ahora 25 puntos hacia la izquierda y 25 puntos hacia arriba (recuerde que y = Ø está en la parte superior de la pantalla), y el dibujo del rectángulo comenzará a partir de esa posición (1Ø3,71).

Los resultados de un comando DRAW se pueden combinar con las formas de los comandos CIRCLE y LINE, pero recuerde que cualquier cambio posterior de la escala (S), color (C), o ángulo (A), afectarán solo a la parte de la figura creada mediante DRAW.

Se pueden utilizar los comandos PSET y PRESET junto con PAINT para introducir más detalles y colores. Tenga cuidado, particularmente con PAINT, ya que los cambios en la parte de la figura realizada con DRAW puede colorear partes equivocadas.

GUARDE EL DIBUJO

Una vez realizada su obra maestra utilizando LINE, CIRCLE y DRAW querrá ahora moverla por la pantalla. Esto se puede hacer borrando y volviendo a dibujar, como hicimos antes. Este procedimiento puede llevar algún tiempo si el dibujo es de algún modo complicado, los dos próximos comandos se ocupan de esto. Todo lo que tiene que hacer es coger (GET) una copia de su dibujo y ponerla (PUT) en otro lugar. El comando GET le permita copiar en una matriz una parte rectangular de su pantalla que luego se podrá colocar, mediante PUT, en la pantalla.

GET (x_1, y_1) — (x_2, y_2), nombre de la matriz, G

x_1, y_1 y x_2, y_2 son las coordenadas de la esquina superior izquierda y la inferior derecha del área rectangular conteniendo el dibujo que quiere almacenar. El nombre de la matriz es el de una matriz previamente dimensionada, donde se almacenará el dibujo. La dimensión de la matriz deberá ajustarse con la dimensión del rectángulo. La primera dimensión de la matriz es el ancho del rectángulo (x_2—x_1), la segunda su longitud (y_2—y_1). El último parámetro, G, es opcional y determina la cantidad de detalle almacenado, no es normalmente necesario con PMODE Ø, 1 ó 3.

Utilizaremos el cohete para ver como se hace. Primero, deberemos calcular la dimensión de la matriz que necesitaremos. El dibujo empieza en 12Ø,96, la aleta izquierda está 6 puntos a un lado, el cohete y la aleta derecha están a 13 + 6, de modo que el dibujo tiene 25 puntos de ancho. desde 114,96 hasta 139,96. La altura es de 26 más la punta del cono, que son 5 puntos, luego la altura es de 31. Agreguemos algunos puntos

de más en cada sentido y tendremos un rectángulo de 30×40 con el extremo superior izquierdo en 112,60 y el inferior derecho en 142,100.

```
10 PMODE 3, 1: SCREEN 1, 1: PCLS: DIM R (29,39)
20 R$ = "C8BM120, 96; U26R13D26L13; C7L6U6E6BR13
   F6D6L6BU26H6G6"
30 DRAW R$
40 GET (112,60) — (142, 100), R, G
100 GOTO 100
```

El ejemplo anterior dibuja nuestro cohete como antes (todo incluído en una línea) y lo almacena en la matriz R. Fíjese que solo necesitamos una matriz de 29 × 39, ya que podemos utilizar los elementos con índice 0.

Una ve z almacenado el dibujo necesitamos ponerlo en pantalla. El comando PUT tiene un formato parecido a GET.

PUT (X_1, Y_1) — (X_2, Y_2), nombre de matriz, acción

x_1, y_1 y x_2, y_2 son las coordenadas de un rectángulo como antes, pero esta vez referidas al área donde desea poner el dibujo. El "nombre de la matriz" es la matriz que contiene el dibujo almacenado. El parámetro "acción" es opcional y solo es necesario cuando el parámetro G se ha utilizado con el comando GET. La acción deberá ser una de las siguientes palabras, y decide como se presentará el resultado en su nueva posición.

PSET	Activa cada punto que se activó (PSET) en la matriz origen. En otras palabras, preséntalo como lo cogiste.
PRESET	Anula cada punto (RESET) activo en la matriz fuente. O bien borrará el dibujo o invertirá los colores, dependiendo del juego de colores fijado.
AND	Compara los puntos del original con los de destino. Si los dos están activos entonces se activará (SET), si uno u otro no lo están el punto no se activará. Esto indica que si se superpone un dibujo sobre otro, solo se mostrarán los puntos que coinciden.
OR	Compara los puntos como antes. Si cualquiera de los puntos está activo (SET) el punto de la pantalla también lo estará. Esto produce el efecto de superponer un dibujo sobre otro.
NOT	Esto invierte cada punto en el área de presentación, o sea presenta la figura sobre el color en primer término.

GET

El comando GET solo se puede utilizar en gráficos en alta resolución. GET copiará los gráficos contenidos en un área rectangular determinada de la pantalla, y los almacenará en una matriz. La matriz deberá haber sido dimensionada correctamente antes.

GET (x_1, y_1) — (x_2, y_2), nombre de matriz, G

x_1, y_1 y x_2, y_2 son las coordenadas del extremo superior izquierdo y del inferior derecho de un rectángulo de la pantalla.

"nombre de matriz" es el nombre de una matriz previamente dimensionada donde se almacenará el contenido del rectángulo.

G indica que se almacenen todos los detalles del gráfico , este comando es opcional.

Vea PUT para un ejemplo del uso de GET.

Deberá utilizar siempre PUT en el mismo modo que GET, si no, pueden aparecer resultados raros. Volvamos a nuestro ejemplo y pongamos (PUT) nuestro cohete en un lugar diferente.

```
50 Y = 150: FOR X = 10 TO 210 STEP 40
60 PUT (X, Y) — (X + 30, Y + 40), R, PSET
80 NEXT
```

Deberá aparecer ahora una fila de cohetes a lo largo del extremo inferior de la pantalla. Para hacer que se mueva el cohete por el extremo introduzca,

```
70 FOR D = 1 TO 200: NEXT D: PCLS
```

Mediante la utilización de los joysticks conjuntamente con los comandos GET y PUT, podrá mover el cohete a voluntad.

```
10 PMODE 3, 1: SCREEN 1, 1: PCLS: DIM S (39,39)
20 DRAW "BM24, 12; S8; C4; E2H2D4D8R8H8G8R2
   NR6F3R6E3"
30 GET (8,8) — (48,48), S
40 A$ = INKEY$: IF A$ = ""THEN 40
50 PCLS: A = JOYSTK (0)*3.25: B = JOYSTK (1)*2.25
60 PUT (A, B) — (A + 46, B + 46), S: GOTO 50
```

El ejemplo anterior dibuja una figura en la esquina superior izquierda de la pantalla. Cuando pulse cualquier tecla se borrará la pantalla y podrá mover la figura utilizando el joystick izquierdo.

Esto completa nuestro estudio sobre las facilidades gráficas existentes. Los ejemplos que hemos ofrecido son necesariamente limitados y no representan en absoluto todo lo que se podría hacer con un poco de imaginación y paciencia. El dibujar figuras puede resultar mucho más sencillo con una planificación previa y dibujando previamente las formas en una hoja para gráficos en lugar de hacerlo en la pantalla directamente.

DRAW

El comando DRAW dibuja una línea, o serie de ellas, de acuerdo con las instrucciones de una cadena. Solo opera en el modo de alta resolución de gráficos.

DRAW cadena

La cadena puede ser una constante de cadena o una variable de cadena y puede contener cualquiera de los siguientes subcomandos:

Mx, y	Se mueve a la posición de dibujo x, y
Un	Arriba n puntos
Dn	Abajo n puntos
Ln	Hacia la izquierda n puntos
Rn	Hacia la derecha n puntos
En	n puntos a 45 grados
Fn	n puntos a 135 grados
Gn	n puntos a 225 grados
Hn	n puntos a 315 grados
X	ejecuta una subcadena
C	Fija el color de la línea
Ak	Gira la siguiente línea un ángulo k = Ø Ø grados k = 1 9Ø grados k = 2 180 grados k = 3 270 grados
Sk	Hace el dibujo a 1/4s de escala, k entre 1 y 62 k = 1 un cuarto de escala, k = 8 dobla la escala Por defecto k = 4
N	No actualizar la posición de dibujo
B	En blanco (no dibuja, solo mueve)

Se pueden especificar movimientos relativos con el parámetro M en la forma

M x desplazado, y desplazado

Donde x desplazado e y desplazado son números que especifican la distancia que hay que mover a partir de la posición actual. Ambos números deberán estar precedidos por un signo más (+) o menos (—).

Un ejemplo de la utilización de DRAW aparece en PUT.

PUT

El comando PUT se utiliza para presentar el contenido de una matriz de gráficos almacenada con el comando GET.

PUT se debe utilizar en el mismo modo que se usó para crear la matriz en el primer lugar, de otra forma los resultados serán impredecibles.

PUT (x_1, y_1) — (x_2, y_2), nombre de matriz, acción

x_1, y_1 son las coordenadas de la esquina superior izquierda de la pantalla donde se presentará la imagen y x_2, y_2 es la esquina inferior derecha. El "nombre de matriz" se refiere a la matriz conteniendo el dibujo. El parámetro "acción" es opcional pero se deberá utilizar si se utilizó el parámetro G en el comando GET.

PSET	Activa los puntos en el destino como en la matriz fuente
PRESET	Desactiva cada punto fijado en la matriz fuente
AND	Compara las matrices fuente y destino. Si ambos puntos están activos, el punto lo seguirá estando, si no, quedará desactivado.
OR	Compara los puntos como antes, si cualquiera de los puntos está activo el punto en pantalla también.
NOT	Invierte el estado de cada punto en el área de destino, de acuerdo con la matriz fuente.

El área seleccionada para presentación deberá tener la misma dimensión que la matriz o aparecerá "basura" en la pantalla.

```
10 PCLEAR 4: PMODE 3, 1: PCLS: SCREEN 1, 1: DIM W (30, 30)
20 DRAW "BM10, 12; S8; R1U3R1D2R2U2R1D3R1D2L1
   D2R1D1L2U3L4D3L2U1R1U2L1U2R1U2BR1BD1D2R2
   U2NL2R2D2L2U2"
30 PAINT (11, 13), 6, 5: GET (0,0) — (30,30) W
40 A$ = INKEY$: IF A$ = "" THEN 40
50 PCLS: FOR C = 0 TO 100 STEP 20
60 FOR A = 0 TO 200 STEP 20
70 PUT (A, C) — (30 + A, 30 +C), W
80 PUT (A, C + 30) — (30 +A, 60 + C), W
90 PUT (A, C+ 60) — (30 + A, 90 + C), W
100 PLAY "T255; ABFGBA": PCLS: NEXT A, C
```

CAPITULO ONCE
EL TOQUE FINAL

EXTRAS DE IMPRESION

Aunque tenemos bastante control sobre como se presentan los resultados en la pantalla, mediante la utilización de PRINT y PRINT @ tendremos más facilidades. El comando PRINT USING le permite especificar exactamente como se debe presentar una línea. Es especialmente útil para sacar tablas, formularios y salidas de cuentas

PRINT USING formato; lista de salida

El formato es una constante de cadena o una variable de cadena conteniendo las instrucciones de como se debe presentar la lista de salida. La lista de salida es la lista normal de constantes y variables, tal y como aparecen en el comando PRINT normal.

Las instrucciones sobre el formato se dan mediante 'especificadores de campo'. Estos son un conjunto de caracteres que indican al ordenador cuantas posiciones de impresión hay que utilizar al presentar una cadena o número.

El especificador #

Este carácter se utiliza para indicar la posición de cada dígito dentro de un número.

PRINT USING "# # #..# #"; A

La sentencia anterior presentará el contenido de A con tres dígitos antes del punto decimal y dos después. Si el número tuviera más de dos dígitos decimales el número se redondeará para ajustarse. Las posiciones no utilizados a la izquierda del punto decimal se presentarán como espacios; si el número es muy grande para ajustarse, el ordenador lo presentará lo mejor que pueda y pondrá un signo % delante para indicar lo que ha pasado.

```
PRINT USING "# # #..# #"; 13.4695
  13.47
PRINT USING "# # #..# #"; 1492.878
%1492.88
PRINT USING "# # #..# #"; 146
146.ØØ
PRINT USING "# # #"; 18.76
  19
```

El especificador *

Muchas veces no interesa dejar espacios delante de un número, especialmente en los cheques. Esto se puede evitar utilizando el especificador *. Si coloca dos asteriscos al comienzo de su campo numérico las posiciones no utilizadas se rellenarán con asteriscos.

```
PRINT USING "**# # #. # #"; 1.492
* * * * 1.49
```

El especificador +

Cuando se coloca un + al comienzo de un campo numérico provoca la impresión del signo del número.

```
PRINT USING "+ # # #.# #"; 14.7
  + 14.70
PRINT USING "+ * * # # #.# #"; —7.4
* * * * —7.40
```

Si el signo (+) se sitúa después del campo numérico forzará que el signo aparezca después del número.

```
PRINT USING "# # #.# # +"; 27.86
  27.86 +
PRINT USING "# # #.# # +"; —1.6
    1.60—
```

Si se coloca el signo menos después de un número, provocará que los números negativos aparezcan seguidos del signo menos, los números positivos vendrán seguidos de un espacio.

```
PRINT USING "** # # #.# #—"; —12.418
* * * 12.42—
PRINT USING "# # #.# #—"; 47.25
  47.25
```

El especificador ^^^^

Este especificador hace que los números se presenten en formato exponencial. Las cuatro flechas hacia arriba deberán seguir al campo numérico.

```
PRINT USING "# #.# ^ #^^^^";123456
1.2346E + 05
```

El especificador!

Este especificador se utiliza con cadenas. Imprimirá solo el primer carácter de la cadena.

```
PRINT USING "!": "CREDIT"
C
```

El especificador %

Para imprimir cadenas es necesario especificar la longitud del campo donde va a aparecer. Esto se realiza con dos signos % separados por un número de espacios. La longitud del campo será el número de espacios más dos. Si la cadena es más larga que la longitud del campo disponible, solo se imprimirán los primeros n caracteres, donde n es la longitud del campo.

```
PRINT USING "%          %"; "DEBITO"
DEBITO
PRINT USING "%    %"; "BALANCE"
BAL
```

El especificador $

El signo ($) se utiliza para representar dinero. Si se coloca delante de un número forzará la salida de un signo $.

```
PRINT USING "$# # #.# #"; 2.87
$    2.87
```

Si se utilizan dos signos $, aparecerá el signo $ justo delante del número.

```
PRINT USING "$$# # #.# #"; 2.87
      $2.87
```

Utilizado conjuntamente con los dos asteriscos el signo dolar provocará el siguiente resultado:

```
PRINT USING "** $ #.# #"; 1.49
** $14.90
```

Los espacios y otros caracteres que aparezcan en la cadena de formato aparecerán también a la salida,

```
PRINT USING "MEDIA    # #.# #    TOTAL    # # #.# #"; 3.4,40.8
MEDIA      3.40      TOTAL      40.80
```

Si la lista de salida contiene más elementos que el número de campos en el formato, el formato se vuelve a empezar desde el comienzo.

```
PRINT USING "# # #.# #      "; 7.84, 142.5, .234
    7.84      142.50      0.23
```

Utilizando la pantalla, la longitud de la línea producida con la sentencia PRINT USING está limitada a 32 caracteres. Cualquier cosa que sobrepase esta longitud continuará al comienzo de la línea siguiente. Sin embargo, para aquellos que tengan impresora, la longitud que pueden utilizar es mucho mayor (al menos 80 caracteres en la mayoría de las impresoras). El formato para la impresora es

PRINT # —2, USING formato; lista de salida

Formato y lista de salida son lo mismo que antes, el # —2 indica que la salida se envíe a la impresora no a la pantalla. Si desea que la salida se haga por impresora y pantalla, tendrá que utilizar dos sentencias PRINT USING.

ENTRADA Y SALIDA POR CASETE

Hasta ahora nuestros programas han requerido que nosotros introdujeramos todos los datos que necesitábamos (o leerlos de una sentencia DATA), y toda la salida se ha mandado a la pantalla. Puede, sin embargo, utilizar el casete para almacenar datos, igual que programas. Estos datos almacenados pueden luego ser leídos posteriormente. El casete se conecta y ajusta exactamente igual que para grabar programas. Luego tendrá que indicar al ordenador que está trabajando con ficheros de datos Esto se realiza con el comando OPEN

OPEN a, # —1, fichero

La "a" debe ser o bien "O" o "I". "O" significa salida (output), es decir que los datos van del ordenador a la cinta. "I" significa entrada (input), los datos pasan del casete al ordenador.

El # —1 indica al ordenador que se utiliza el casete. El "fichero" es el nombre que quiere dar al fichero de datos (cualquiera que empiece con una letra y tenga un máximo de 8 caracteres).

El siguiente paso es escribir los datos en la cinta. Esto se realiza con un comando PRINT de la forma siguiente.

PRINT # —1, lista de salida

La única diferencia respecto al comando PRINT que hemos visto hasta ahora, está en el # —1. Esto indica al ordenador que saque la lista de salida hacia la cinta y no hacia la pantalla.

Cuando haya acabado de escribir los datos, deberá cerrar el fichero con el comando CLOSE.

CLOSE # —1

PRINT USING

El comando PRINT USING permite un gran control sobre la salida de resultados por pantalla o impresora.

PRINT USING formato; lista de salida

El formato es una constante o variable de cadena conteniendo los especificadores de campo que indican como se debe imprimir la lista de salida. La lista de salida es una lista de cadenas o variables numéricas (o constantes) separadas por comas .

Los especificadores de campo son:

CARACTER	ACCION	EJEMPLO	RESULTADO
#	Formato de número	"####"; **147.2**	**147**
,	Presenta una coma a la izq. cada 3 caracteres	"######.#"; **123456**	**123,456**
**	Rellena los espacios iniciales con *	"**###.###"; **1.47**	*** 1.47**
$	Coloca un signo $ delante del número	"$###.##"; **12.689**	**$ 12.69**
** $	Signo $ flotante	"**$####.##"; **12.689**	****$12.69**
+	En primera posición hace que se ponga el signo delante, en última posición lo imprime después del número	"##.##+"; **—12.689**	**12.69—**
↑↑↑↑	Imprime en formato exponencial	"##.## ↑↑↑↑ ; **12.689**	**0.13E + 02**
!	Imprime solo el primer carácter	"!"; **"CREDITO"**	**C**
%espacios%	Campo de cadena. La longitud del campo es el número de espacios más 2	"% %"; **BALANCE**	**BALANCE**

Cada especificador de campo puede ir separado por cualquier número de espacios y aparecerán como tales en la línea de salida.

```
10 CLS: INPUT "INTRODUZCA EL ULTIMO BALANCE";B: C = 0: D =0
20 CLS: T$ +"      %      %      %      %      %      %"
30 L$ ="      ####.##     ####.##     ####.##+"
40 PRINT USING T$; "DEBITO", "CREDITO", "BALANCE"
50 PRINT USING L$; D, C, B
60 OPEN "I", # -1, "CHEQ"
70 IF EOF (-1) THEN 110
80 INPUT # -1, A; D = 0; C =0
90 IF A< 0 THEN D = ABS (A) ELSE C = A
100 B = B + C-D: PRINT USING L$; D, C, B: GOTO 70
110 CLOSE# -1: END
```

SALIDA POR IMPRESORA

Para aquellos que tengan impresora conectada al port de I/O paralelo, existen variaciones en algunos comandos, que permitirán que la salida se dirija a la impresora en lugar de la pantalla.

PRINT # -2, lista de salida
PRINT # -2, USING formato; lista de salida

El formato y la lista de salida son los mismos que los utilizados para la pantalla.

POS (-2) devolverá la posición actual de la cabeza impresora. **LLIST** listará un directorio del programa directamente por la impresora. Se utiliza como el comando LIST.

Utilizando la combinación [SHIFT] [0] permite que salgan las letras minúsculas por la impresora. La opción de minúsculas solo se puede utilizar con cadenas o sentencias REM, ya que todos los comandos en BASIC deberán estar en mayúsculas.

Para volver a leer los datos, se siguen los mismos pasos salvo que esta vez el fichero se abre para entrada (INPUT) y en lugar de PRINT utilizará,

INPUT # — 1, lista de entrada

El comando CLOSE es el mismo para ambos.

Los ejemplos que vienen muestran como se realiza todo el proceso. Primero ajuste el casete y avance la cinta hasta el lugar donde quiere poner el fichero, (utilice SKIPF). Ahora pulse los botones PLAY y RECORD simultáneamente.

```
10 CLS: PRINT "CREACION DE UN LISTIN TELEFONICO"
20 OPEN "O", # —1, "TELEFONO": PRINT "INTRODUZCA XXX PARA
   TERMINAR"
30 PRINT @ 128, "";:INPUT "NOMBRE    ", N$; PRINT @ 128, "";
40 INPUT "NO. TELEFONO"; T$: IF N$ = "XXX" OR T$ = "XXX" THEN 60
50 INPUT #  —1, N$, T$: PRINT @ 128, "": GOTO 30
60 CLOSE # — 1: END
```

Cuando ejecute el programa la cinta arrancará y empezará a grabarse el fichero en la cinta. Cada vez que introduzca un nombre y un número se escribirá en el fichero. (La sentencia PRINT @ 128 en la línea 30 simplemente borra esa línea de la pantalla). Esto continuará hasta que introduzca XXX, XXX, entonces se cerrará el fichero y acabará el programa.

Una vez escritos, todo lo que tenemos que hacer es volverlos a leer. La principal diferencia entre salida y entrada, es que con INPUT no debe intentar leer después del final del fichero. De esto se encarga otra sentencia que necesitará para la entrada, el comando EOF. Este comando comprueba si se ha alcanzado el final de un fichero cuando lee.

Rebobine la cinta hasta el comienzo y esta vez pulse solo el botón PLAY.

```
10 CLS: PRINT "LECTURA DEL LISTIN TELEFONICO"
20 OPEN "I", #  —1, "TELEFONO"
30 PRINT "NOMBRE", "NUMERO"
40 IF EOF (—1) THEN 60
50 INPUT #  —1, A$, B$: PRINT A$, B$: GOTO 40
60 CLOSE #  —1: END
```

Cuando ejecute el programa esta vez, la cinta arrancará y buscará el fichero "TELEFONO". (Deberá esperar un poco si este se encuentra hacia el final de la cinta). Después leerá los nombres y los números y los presentará en la pantalla. Fíjese que no tiene porque utilizar el mismo

nombre de variable que utilizó para escribir los datos; deberá sin embargo utilizar el mismo tipo de variable. Cuando se llega al final del fichero este se cierra y el programa termina.

El comando EOF deberá aparecer antes del comando INPUT # —1, si no aparecerá un error IE, por intentar leer después del final de un fichero.

No se olvide cerrar (CLOSE) un fichero ya que puede causar problemas, especialmente cuando se escribe en el.

UN POCO MAS

Ya está a punto de convertirse en un experto programador de BASIC, y querrá ver el siguiente paso — el lenguaje máquina. Este es el lenguaje nativo del ordenador; hasta ahora nos hemos estado comunicando con el a través de un intérprete que es el BASIC.

Para que nos vamos a molestar? Pues verá, las instrucciones en lenguaje máquina trabajan bastante más deprisa, utilizan menos memoria, e incluso le permiten hacer cosas que con el BASIC no puede.

La mejor forma de conocer el lenguaje máquina es obteniendo un manual que haga referencia a la serie de microprocesadores 6800. Uno de estos manuales es

Basic Microprocesors and the 6800 por Ron Bishop publicado por Hayden Book Co. Inc.

Una vez que tenga la base, su ordenador tiene un número de rutinas que le permiten utilizar rutinas en lenguaje máquina. Se dan breves detalles sobre esto a continuación.

USRn le permite llamar hasta 10 (0 a 9) rutinas en lenguaje máquina. El formato es,

USRn (argumento)

Donde el "argumento" puede ser una cadena o una expresión numérica. Cuando se encuentra una llamada de usuario en un programa, se transfiere control a la dirección dada en la sentencia DEF USRn. La dirección específica el punto de entrada de la rutina en lenguaje máquina.

DEF USRn se utiliza para definir la dirección de una función USRn. Su formato es

DEF USRn = dirección

n está entre 0 y 9 y debe corresponderse con la n en USR. La dirección deberá estar entre 0 y 65535 y contener la dirección de entrada para USRn.

CLEAR s, h. La sentencia CLEAR se deberá utilizar para reservar memoria para las funciones USR. La "s" indica el espacio reservado para cadenas. Y la "h" es la dirección mayor de memoria que puede utilizar el BASIC. Desde h + 1 en adelante estará reservado para rutinas en lenguaje máquina.

El comando POKE se utiliza para poner un valor en una dirección específica de la memoria.

POKE dirección, valor

La dirección es como antes y valor debe estar entre Ø y 255.

VARPTR. Un puntero a una variable BASIC se puede utilizar como argumento por una función USR. Esto permitirá que una función USR acceda al contenido de una matriz.

VARPTR (nombre de variable)

Donde "nombre de variable" es la variable BASIC a la que quiere acceder. VARPTR se utiliza como parte del argumento USR como en,

USRØ (VARPTR (X))

Las rutinas en lenguaje máquina pueden grabarse y cargarse de un casete utilizando los comandos CSAVEM y CLOADM.

CSAVEM nombre, comienzo, fin, difer
CLOADM nombre, desplazamiento

"Nombre" es el nombre para el fichero en cinta, "comienzo" es la dirección de comienzo de la rutina en memoria, "fin" es la última dirección ocupada por la rutina y "difer" es la diferencia entre el comienzo y fin. El "desplazamiento" en el comando CLOADM le permite volver a cargar la rutina en memoria en la dirección dada por comienzo + desplazamiento.

Una vez cargada, se puede transferir el control a la rutina mediante el comando EXEC,

EXEC dirección

La dirección indica el comienzo de la rutina, si se omite, el ordenador utilizará el comienzo del último comando CLOAD.

APENDICE A

CODIGOS A.S.C.I.I.
(Decimales)

TECLA	SIN PULSAR SHIFT	PULSANDO SHIFT
[BREAK]	3	3
[CLEAR]	12	92
[ENTER]	13	13
[ESPACIO]	32	32
!	33	—
"	34	—
#	35	—
$	36	—
%	37	—
&	38	—
'	39	—
(	40	—
)	41	—
*	42	—
+	43	—
,	44	—
—	45	—
.	46	—
/	47	—
Ø	48	18
1	49	—
2	5Ø	—
3	51	—
4	52	—
5	53	—
6	54	—
7	55	—
8	56	—
9	57	—
:	58	—
;	59	—
<	6Ø	—
=	61	—
>	62	—

TECLA	SIN PULSAR SHIFT	PULSANDO SHIFT
?	63	—
@	64	19
A	97	65
B	98	66
C	99	67
D	100	68
E	101	69
F	102	70
G	103	71
H	104	72
I	105	73
J	106	74
K	107	75
L	108	76
M	109	77
N	110	78
O	111	79
P	112	80
Q	113	81
R	114	82
S	115	83
T	116	84
U	117	85
V	118	86
W	119	87
X	120	88
Y	121	89
Z	122	90
↑	94	95
↓	10	91
←	8	21
→	9	93

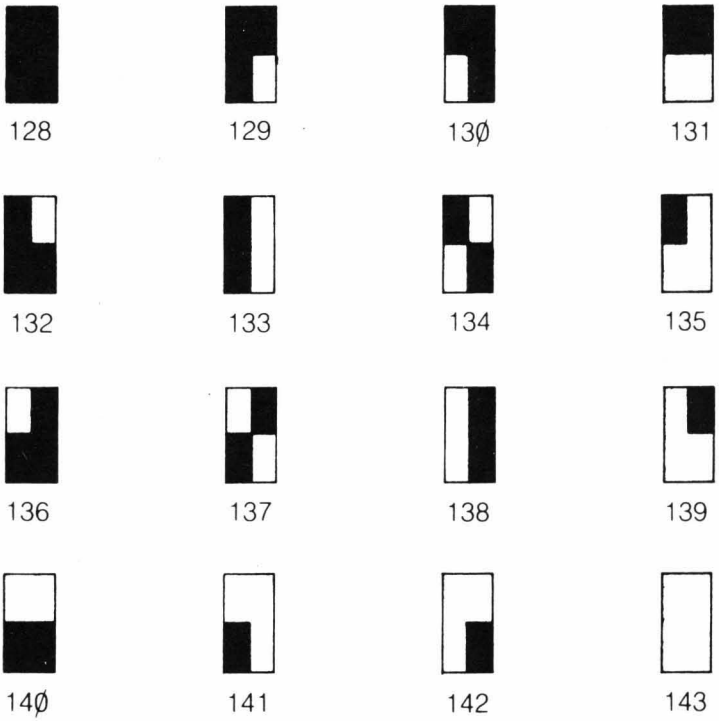
Los caracteres sin [SHIFT] se obtienen pulsando [SHIFT] [0]; combinación que introduce las minúsculas.

Los siguientes caracteres en minúsculas están disponibles con la función CHR$:

[CHR$(123)
/ CHR$(124)
] CHR$(125)
↑ CHR$(126)
← CHR$(127)

Los caracteres del 127 al 255 son los caracteres gráficos siguientes:

CARACTERES GRAFICOS



Para generar los caracteres anteriores utilice CHR$ con el código correspondiente. Para obtener los demás colores, sume al código el número apropiado. Por ejemplo, PRINT CHR$ (142 + 112) produce el carácter 142 pero el área verde se convierte en naranja.

+16	amarillo	+32	azul	+48	rojo
+64	beige	+8Ø	turquesa	+96	magenta
		+112	naranja		

APENDICE B

PANTALLAS DE IMPRESION Y GRAFICOS

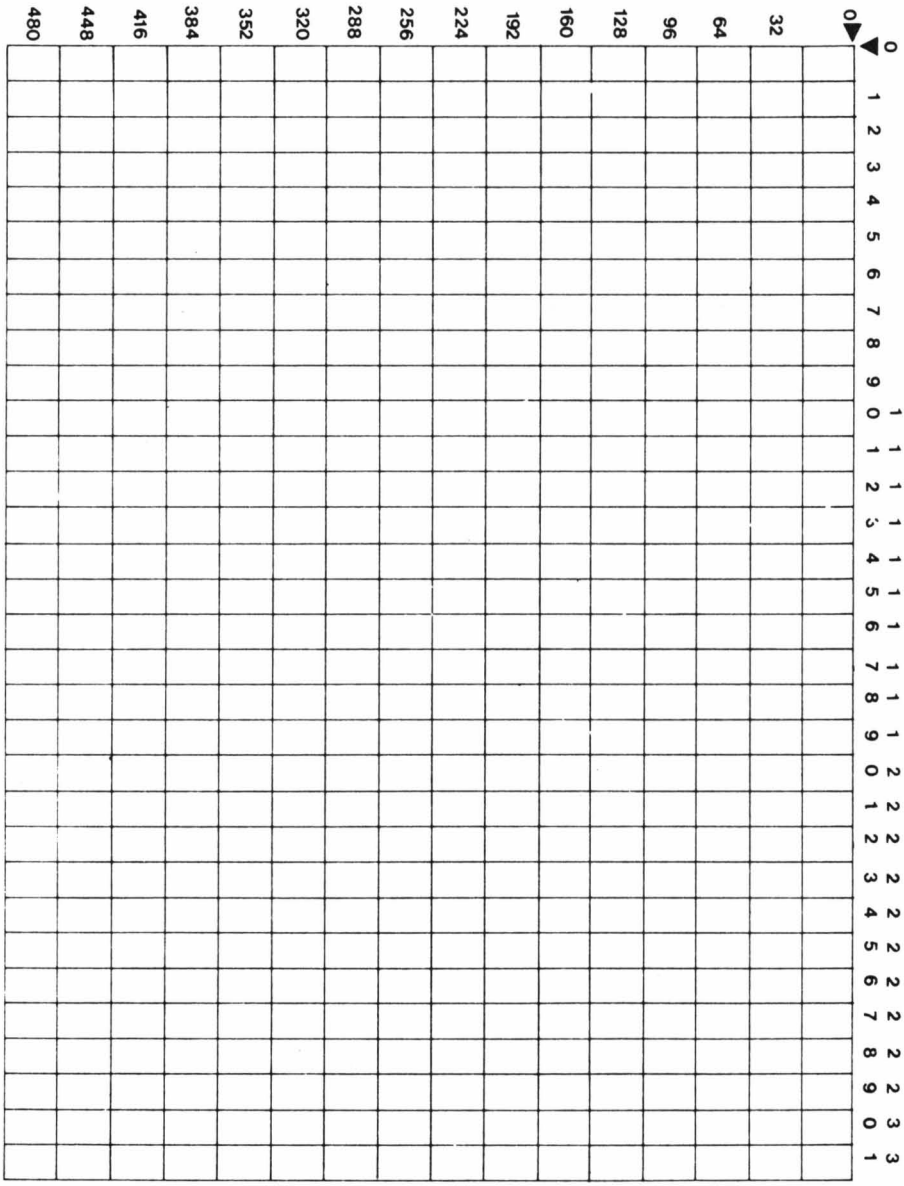
Las siguientes hojas de trabajo son muy útiles para hacer gráficos y preparar impresiones de salida.

La primera se utiliza para el comando PRINT @

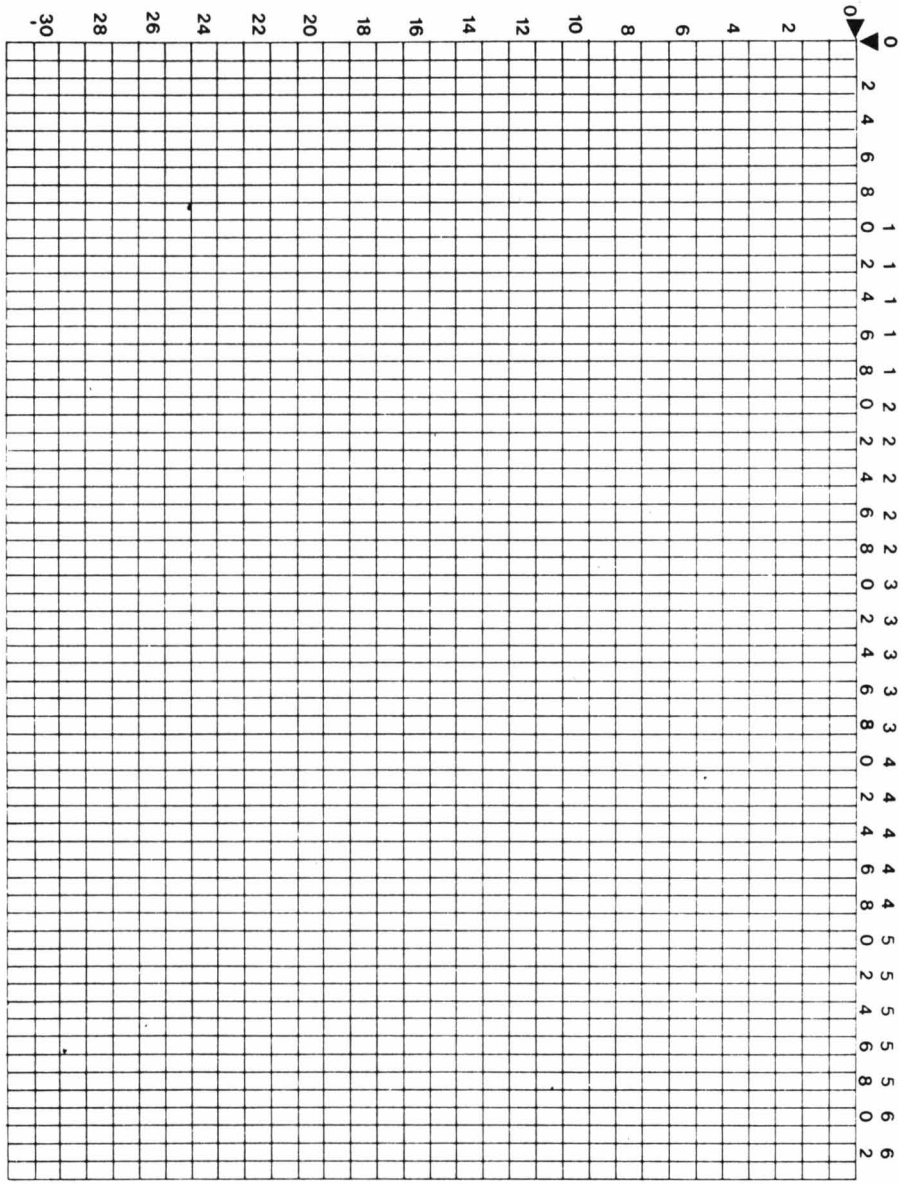
La segunda sirve para gráficos en baja resolución en la pantalla para texto, utilizando los comandos SET y RESET.

La tercera es para gráficos en alta resolución.

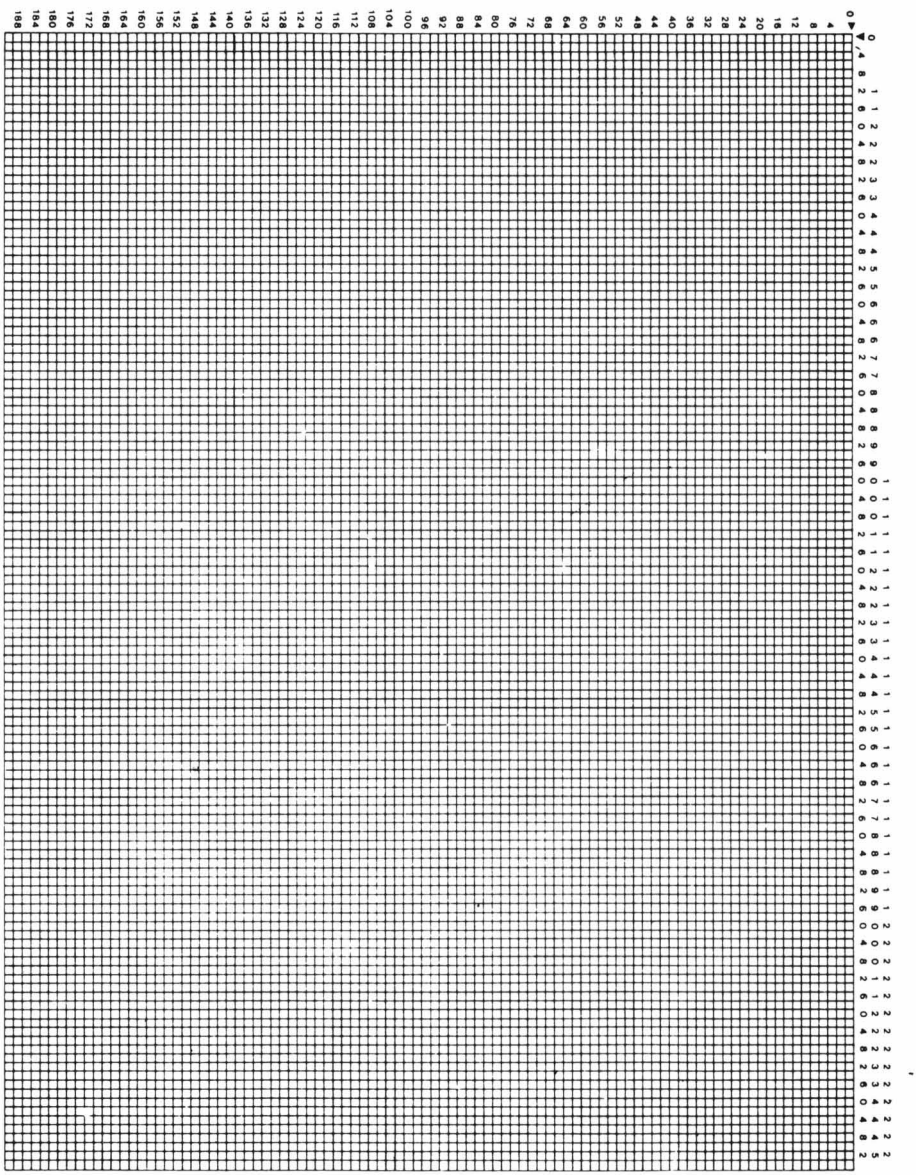
Retícula Print @



Retícula de Baja Resolución



Retícula de Alta Resolución



APENDICE C

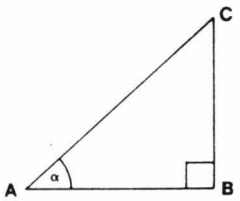
CODIGOS DE ERROR

CODIGO	EXPLICACION
/Ø	División por Ø. Imposible.
AO	Intento de abrir un fichero que ya lo está. Normalmente aparece después de pulsar RESET para detener un programa que utiliza ficheros. Apague y vuelva a encender.
BS	Indice erróneo. Normalmente porque el valor del índice es mayor que la dimensión declarada para la matriz.
CN	No se puede continuar (CONT) después de un END.
DD	Intento para redimensionar una matriz: solo se pueden dimensionar una vez.
DS	Sentencia directa. Normalmente aparece si intenta hacer CLOAD con un fichero de datos.
FC	Llamada ilegal de Función. Normalmente es porque el parámetro está fuera de rango o no corresponde el tipo.
FD	Fichero erróneo. Producido al insertar un dato de tipo cadena en una variable de cadena utilizando un fichero de datos en casete.
FM	Fichero en modo erróneo. Tratando de leer de un fichero abierto para salida (O), o imprimiendo datos sobre un fichero abierto para lectura (I).
ID	Sentencia directa errónea. Intento de utilización de una sentencia que solo se puede usar desde programa.
IE	Intento de lectura pasado el final del fichero. Utilice IF EOF (—1) para detectarlo.
IO	Error de entrada/salida. El casete no está bien ajustado o la cinta es defectuosa.
LS	Cadena demasiado larga. El máximo son 255 caracteres.
NF	NEXT sin FOR. Ocurre normalmente si están invertidos los NEXT de dos bucles.
NO	Fichero sin abrir. Solo se puede leer o escribir en un fichero después de haberlo abierto.
OD	Sin datos. Una sentencia READ terminó con los datos.
OM	Sin memoria. Se ha utilizado toda la memoria o está reservada.
OS	Sin memoria para cadenas. Utilice CLEAR para obtener más espacio.

CODIGO	EXPLICACION
OV	Desbordamiento. El número es demasiado largo para que el ordenador lo pueda manejar.
RG	RETURN sin GOSUB. El programa posiblemente se ha caído en una subrutina, o ha saltado a su interior.
SN	Error de síntaxis. Provocado normalmente por un mal deletreo o un mal uso de los signos de puntuación.
ST	Fórmula para cadena demasiado compleja. Divida la operación.
TM	Error de tipos. Intento de asignar un dato de cadena a una variable numérica o a la inversa.
UL	Línea indefinida. Se ha saltado a una línea inexistente.

APENDICE D

FUNCIONES TRIGONOMETRICAS



En un triángulo rectángulo ABC,

AB es el lado adyacente al ángulo
BC es el lado opuesto al ángulo
AC es la hipotenusa.

El seno, coseno y tangente del ángulo se definen como:

SIN = lado opuesto/hipotenusa

COS = lado adyacente/hipotenusa

TAN = lado opuesto/lado adyacente

Todas las funciones trigronométricas en BASIC se suponen medidas en radianes. Un radián es la medida de un ángulo en unidades de círculo. Un círculo tiene 360 grados o 2 × PI radianes. (PI es una constante que vale 3.1415926). Luego para pasar de uno a otro:

grados/ (180/PI) = radianes

radianes * (180/PI) = grados

La inversa de una función es el recíproco de aplicar esa función. Por ejemplo, la tangente de un ángulo de 1.5 radianes es,

TAN (1.5) = 14.101419

La inversa de TAN (tangente) es ATN (arcotangente). Esta se utiliza para hallar el ángulo si se conoce la tangente,

ATN (14.101419) = 1.5

La inversa de las funciones SIN y COS no están disponibles en BASIC, pero se pueden calcular utilizando la función ATN en las siguientes fórmulas:

Inversa del seno = (ATN (X/SQR (—X * X* + 1))

Inversa del coseno = (ATN (X) SQR (—X* X + 1)) + 1.5708

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

INDICE

DRAGON 32 - INFORMACION ADICIONAL

Estas hojas contienen algunos detalles sobre su DRAGON 32. Tiene una especial importancia el párrafo referente al cable de conexión del cassete, ya que difiere de la información dada en el manual "Introducción a la programación en BASIC".

También se incluye información sobre la conexión y el ajuste de su televisor, detalles sobre como conectar la impresora y una copia del mapa de memoria del Dragon 32.

TELEVISION

Una vez conectado el ordenador al televisor, y de haberlos encendido, seleccione un canal que no utilice y ajústelo como se indica en su manual. Si la imagen del televisor no se mantiene estacionaria, será necesario mover el potenciómetro de ajuste vertical; si su televisor no tiene este botón, llame a un técnico .

CASSETE

Para conectar el cassete al Dragon, utilizando un cable de conexión de Dragon, conecte la toma DIN en el hueco marcado TAPE en el lado izquierdo de su ordenador. Los tres conectores del otro lado del cable se conectan al cassete como se indica a continuación:

(i) El jack menor con un cable azul va en el orificio marcado normalmente como REM y próximo a la toma de micrófono.

(ii) El jack con el cable rojo va en la toma marcada AUX o MIC o LINE IN. Si hay posibilidad de elegir entre AUX y MIC, utilice siempre AUX.

(iii) El Jack con el cable blanco va en la toma marcada EAR o MONIT, o L/S o SPKR .

Es posible que para utilizar los controles de avance y rebobinado rápido del cassete, necesite desconectar el cable de control remoto. Asegúrese después de volverlo a conectar.

Alternativamente, teclee MOTOR ON y pulse ENTER antes de utilizar los controles de avance y rebobinado rápido. Después escriba MOTOR OFF y pulse ENTER para continuar utilizando el cassete con el ordenador.

Si va a reutilizar una cinta le recomendamos que antes la borre por completo.

IMPRESORA

La toma para impresora que tiene el Dragon 32 en el lado izquierdo es para conectar una impresora que utilice una interfaz paralela de tipo centronics (toma 6 en la ilustración del manual). La conexión de los pines es:

PIN 1 Inhibidor de impresión (strobe)	PIN 11 Bit 4 de datos	PIN 2 +5 voltios	PIN 12 0 voltios
PIN 3 Bit 0 de datos	PIN 13 Bit 5 de datos	PIN 4 +5 voltios	PIN 14 0 voltios
PIN 5 Bit 1 de datos	PIN 15 Bit 6 de datos	PIN 6 0 voltios	PIN 16 0 voltios
PIN 7 Bit 2 de datos	PIN 17 Bit 7 de datos	PIN 8 0 voltios	PIN 18 0 voltios
PIN 9 Bit 3 de datos	PIN 19 NACK	PIN 10 0 voltios	PIN 20 NOT BUSY

La posición de los pines con números impares están en la parte superior del conector, con el PIN 1 situado a la derecha. Los pines pares están en la línea inferior con el PIN 2 a la derecha.

CARTUCHOS

Deberá asegurarse de que ha apagado el ordenador antes de introducir o cambiar un cartucho en el lado derecho del ordenador.

MAPA DE MEMORIA DEL DRAGON 32

DIRECCION DECIMAL	CONTENIDO	DIRECCION HEXADECIMAL
0-1023	Sistema	0-3FF
255	Página RAM direccionada	0FF
1023	Página RAM extendida	3FF
1024-1535	Memoria de pantalla para texto	400-5FF
	Memoria para gráficos en pantalla	
1536-3071	Página 1	600-BFF
3072-4607	Página 2	C00-11FF
4608-6143	Página 3	1800-17FF
6144-7679	Página 4	1800-1DFF
7680-9215	Página 5	1E00-23FF
9216-2559	Página 6	2400-29FF
2560-12287	Página 7	2A00-2FFF
12288-13823	Página 8	3000-35FF
13824-32767	Programas y variables	3600-7FFF
32768-49151	Intérprete BASIC	8000-BFFF
49152-65279	Cartucho de memoria	C000-FEFF
65280-65375	Entrada/salida	FF00-FFDF
65376-65503	Bits de control SAM	FF60-FFDF
65504-65535	Vectores MPU	FFE0-FFFF

Eurohard

Espąñoleto, 25
Teléfs. (91) 410 30 64 - 410 31 96 - 410 34 98
Télex 45845 ICSG E
28010 MADRID