SONY

MSX

Introducción al MSX-BASIC

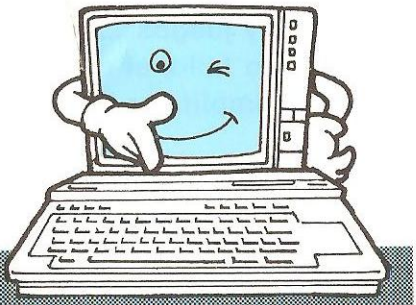
HIT BIT

Introducción al
MSX-BASIC

ÍNDICE

COMENCEMOS

- Finalidad de este libro
- Ron, el perro inteligente
- Su ordenador Sony



FINALIDAD DE ESTE LIBRO

Este libro le introducirá en el BASIC.

¿Qué es el BASIC? El BASIC es un lenguaje para "charlar" con ordenadores. El nombre del lenguaje es BASIC porque es la forma más sencilla, más **básica** de charlar con un ordenador.

Pero, ¿por qué tenemos que charlar con un ordenador? Porque este ordenador Sony nos ayudará a realizar muchas cosas: con él podremos jugar, resolver problemas, practicar matemáticas, almacenar información, y mucho más. Este libro le indicará, en solamente una o dos horas, cómo poder divertirse con su ordenador Sony.

Primeramente tendrá que aprender cómo su ordenador puede escuchar y hablarle (en lenguaje BASIC). Después utilizaremos este lenguaje especial para que el ordenador realice algunas cosas interesantes y sencillas al principio, y complicadas más adelante. Antes de finalizar el libro conseguirá que su ordenador Sony trabaje como una máquina tragaperras igual que las utilizadas en los casinos.

Después de haber usado este libro para familiarizarse con el BASIC, usted podrá leer y utilizar programas publicados en revistas sobre ordenadores, y compartir sus experiencias con amigos que también posean ordenadores. Si desea aprender más sobre el BASIC, puede adquirir libros más avanzados sobre la materia.

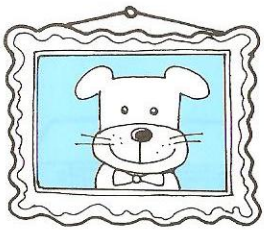
Al hablar sobre ordenadores y sus programas, se utilizan muchas palabras especiales. Éstas se enumeran en el **Índice alfabético** que encontrará al final de este libro. Cuando desee conocer el significado de una palabra o un mandato, este índice le dirá dónde poder encontrar la explicación.

For último, hay un punto muy importante que debe tener presente: los ordenadores no tienen mal carácter. Son sencillos y "**divertidos**", y no les importan las equivocaciones que se puedan cometer al utilizarlos. Con su ordenador Sony nada "saldrá erróneo" aunque pulse una tecla "equivocada". Por lo tanto, no tenga miedo. Como en cualquier otra nueva actividad, es natural que tenga que corregir cosas y volver atrás para repetir ciertos pasos a fin de conseguir el éxito. Su ordenador tendrá toda la paciencia del mundo para ayudarle a aprender.

De hecho, la única cosa realmente importante que tiene que recordar al comenzar este libro es que con su ordenador Sony habrá conseguido una nueva amistad que simplemente estará esperando a que usted la utilice.

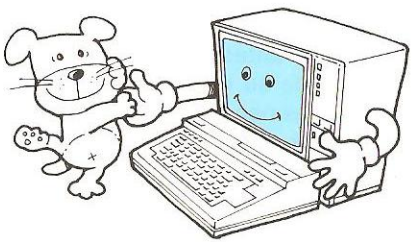
RON, EL PERRO INTELIGENTE

A lo largo de nuestro aprendizaje del BASIC nos acompañará un amigo. Éste es el perro de la familia, Ron. Al igual que la mayoría de los perros, es cariñoso y fiel. Ron es un perro de raza corriente, pero bastante inteligente. Si se lo pedimos, realizará infinidad de cosas.



Ron

¿Por qué se encuentra Ron en un libro sobre el BASIC y ordenadores? La razón es que los perros son muy buenos obedeciendo **mandatos** y realizando juegos. Y como su ordenador Sony es tan fiel como Ron, enseguida verá que obedece **mandatos** y realiza juegos también, pero no exactamente en la misma forma. Si pensamos en cómo hace las cosas Ron, y cómo las realiza su ordenador, será mucho más fácil comprender la forma que tiene el ordenador de escuchar, hablar y pensar en BASIC.

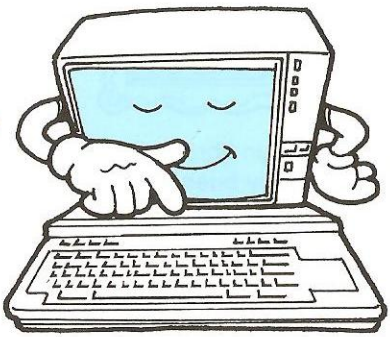


Pareja fiel

SU ORDENADOR SONY

Ron, naturalmente, obedece al pie de la letra lo que se le manda. "¡Siéntate!", "¡Revuélcate!", "¡Búscalo!", "¡Tráelo!" ... esto es muy fácil para Ron. ¿Y el ordenador? Intente "¡Dame la pata!".

Ron entendió y le dió la pata. Pero el ordenador no tiene patas. Es más, tampoco tiene oídos.
Entonces, ¿cómo decirle que haga cosas? Los "oídos" del ordenador, es decir, su forma de oír los mandatos, son el **teclado**.



Hable con sus dedos

Antes de aprender BASIC, y antes de pasar a la página siguiente, lea el **Manual de Instrucciones** de su ordenador Sony para aprender los detalles sobre el teclado. No es necesario memorizar todas las teclas, pero es muy importante entender el **cursor**: qué es, y cómo se mueve. Escriba también algunas **letras**, **números** y **gráficos** para que aparezcan en la pantalla.

PONGA SU ORDENADOR A TRABAJAR

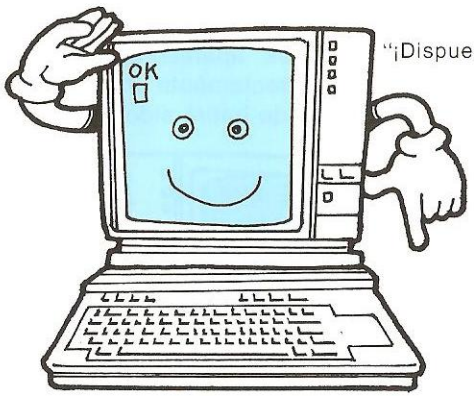
Cómo...

- Parar su ordenador
- Dar mandatos
- Escribir en color
- Utilizar la tecla de cambio de caja
- Hacer que su ordenador realice operaciones aritméticas

Ahora que ha leído el Manual de Instrucciones y que conoce el teclado, veamos lo que puede hacer su ordenador Sony.

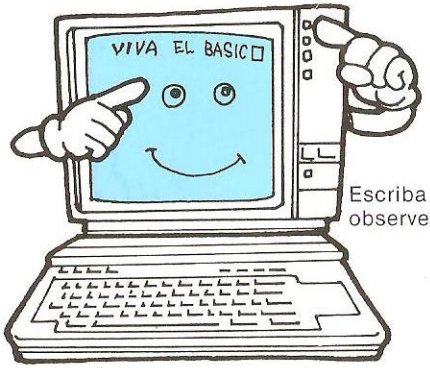
MANDATOS Y ENTRADAS

Antes hemos dicho que el teclado son los "oídos" que el ordenador utiliza para oír los mensajes. Pero, ¿cuándo sabremos que el ordenador nos está escuchando? Él nos indicará que está dispuesto con un amable "Ok".



"¡Dispuesto y a sus órdenes!"

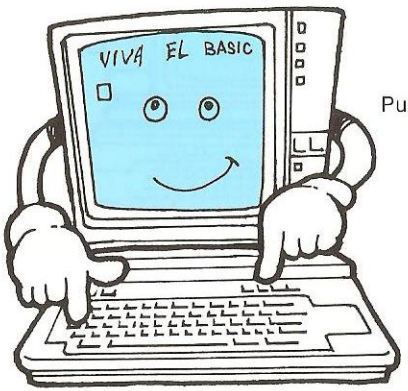
Cuando usted teclee, la pantalla le mostrará todas las letras y números que haya escrito. El cursor estará en medio de la pantalla y el mensaje "Ok" habrá quedado atrás.



Escriba algunas palabras y observe el movimiento del cursor.

Después de esto, llamemos la atención del ordenador. Pulse la tecla CTRL con un dedo, y la tecla STOP con otro. Cuando pulse CTRL y STOP **al mismo tiempo**, el ordenador dejará de escuchar los mandatos anteriores y prestará atención a los nuevos.

Los mandatos difieren de las **entradas**. Una entrada son las letras y números que componen la información introducida en el ordenador, como las letras empleadas para escribir en la pantalla "Viva el BASIC". La entrada no dice directamente al ordenador que haga nada. Para indicar al ordenador lo que tiene que hacer con lo que hayamos introducido, necesitaremos **mandatos**, que se emiten con ciertas teclas o palabras escritas. En una calculadora de bolsillo, por ejemplo, piense que en la suma 2+2=4, "2", "+" y "2" son la entrada. La tecla "=" es un **mandato** dado a la calculadora para decirle que sume 2 más 2 y que nos dé el resultado, o salida, 4. Exactamente así es como trabaja un ordenador. Un punto muy importante: las letras "Ok" aparecerán **siempre** sobre el cursor cuando se haya introducido correctamente un **mandato**; esto significará que lo que usted haya realizado habrá sido correcto.

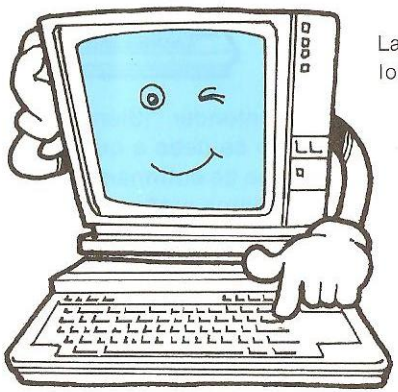


Pulse CTRL y STOP.

Ahora el ordenador estará listo para aceptar sus mandatos. Usted podrá introducir los mandatos utilizando minúsculas **o** mayúsculas, no hay diferencia ninguna. Ordenémosle que nos dé la pata, escribiendo "DAME LA PATA".

```
DAME LA PATA
```

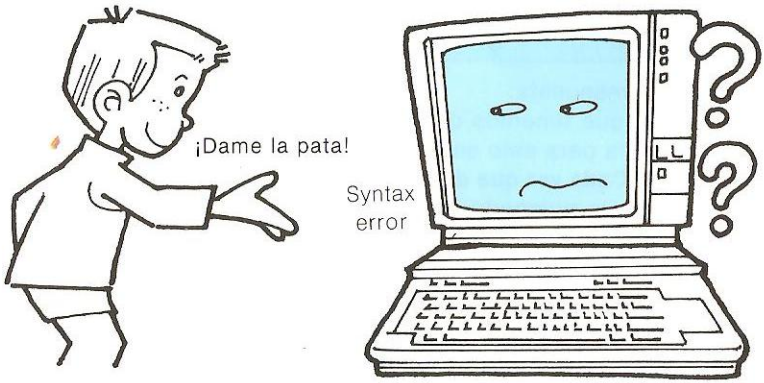
Todavía no hay respuesta.
Esto se debe a que tenemos que decirle cuándo tiene que ejecutar el mandato. La tecla para esto se encuentra a la derecha, y está marcada con "RETURN". **Cada vez que dé un mandato al ordenador**, deberá hacer que éste lo ejecute pulsando la tecla RETURN. Esta tecla es un poco más grande que el resto porque se utiliza muy a menudo.



La tecla RETURN da los mandatos.

Pulse RETURN, y obtendrá una respuesta. El ordenador no nos dará la pata, sino que emitirá un pitido y nos ofrecerá una respuesta como ésta:

Éste es un **mensaje de error**. Significa que el ordenador ha oído el mandato pero no lo ha entendido. Esto as muy natural en este caso, porque los ordenadores no saben dar la pata a la gente. Este mensaje, "Syntax error", significa que hay un error sintáctico en el lenguaje, BASIC, naturalmente.



Nuestro buen perro Ron puede entender "¡Siéntate! y ¡Dame la pata!, y muchas otras palabras, pero ello se debe a que alguien se las enseñó. Nuestro ordenador tiene otra clase de entrenamiento, por lo que los mandatos son diferentes. Estos mandatos están en el lenguaje denominado BASIC. Y cuando usted los aprenda verá que es muy fácil charlar con el ordenador. Probemos ahora algunos sencillos.

ALGUNAS COSAS QUE RON NO PUEDE HACER

Aquí tenemos, para empezar, un mandato fácil aunque Ron no lo crea así.

```
WIDTH 10
```

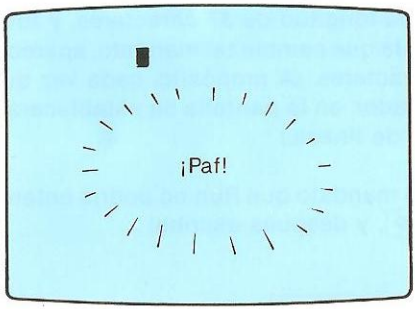
Esto no es difícil de escribir, pero asegúrese de pulsar la barra espaciadora una vez para dejar un **espacio** entre la palabra WIDTH y la cifra 10.

W I D T H [ ] 1 0

No se preocupe si comete alguna equivocación. Simplemente efectúe lo que indica el **Manual de Instrucciones** para mover el cursor y corregir errores. El ordenador Sony olvidará todas las equivocaciones tan pronto como usted las corrija. Cuando la pantalla muestre esto:

```
WIDTH 10■
```

usted podrá emitir el mandato. Pulse la tecla RETURN y, ¿qué sucede?



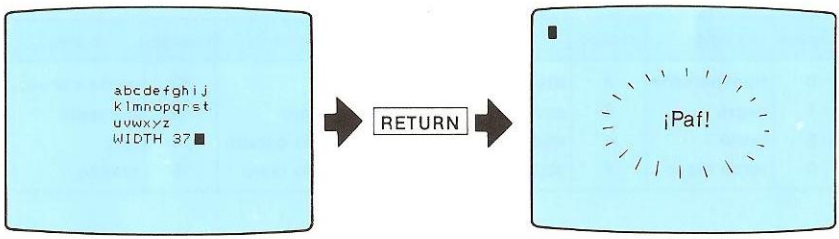
Las letras han desaparecido, y el cursor se encuentra próximo al centro de la pantalla. Entonces, ¿qué significa el mandato WIDTH 10? Escribamos algo, y enseguida lo comprenderemos. Efectuemos una **entrada**: escriba el alfabeto sin espacios ni RETURN.

¿Cuántas letras hay en cada línea? Diez. Sin ningún otro mandato, nuestro ordenador ha entendido nuestro deseo de que cada línea tenga una anchura de **diez caracteres** solamente. WIDTH 10 es uno de los mandatos que el ordenador entiende, y cuando lo oiga, hará que cada línea de la pantalla tenga una anchura exacta de diez caracteres.

Probemos otro mandato, pero primeramente pulse juntas las teclas CTRL y STOP para "borrar" las últimas entradas y preparar el ordenador para el **siguiente** mandato:

```
WIDTH 37
```

Ahora pulse la tecla RETURN.



¿Qué ha pasado ahora? La pantalla vuelve a estar vacía, con el cursor otra vez en la esquina superior izquierda. Usted mandó al ordenador que diese a las líneas una longitud de 37 caracteres, y todo lo que escriba a partir de ahora, hasta que cambie tal mandato, aparecerá en la pantalla con líneas de 37 caracteres. (A propósito, cada vez que conecte la alimentación del ordenador, en la pantalla se establecerá automáticamente 37 como longitud de líneas.)

Ahora probemos otro mandato que Ron no podría entender. Primeramente pulse CTRL y STOP, y después escriba:

```
COLOR 8
```

(No se olvide de pulsar RETURN.) ¡De repente todas las letras de la pantalla se han vuelto rojas! Escriba algunas palabras más. Todas serán del mismo color.

¿Qué tal si probamos otro color? En primer lugar, devuelva el cursor a la izquierda de la pantalla pulsando CTRL y STOP. Después escriba COLOR 15 y pulse la tecla RETURN, y volverán a estar escritas en blanco.

Es muy fácil ver que para el ordenador 15 significa blanco y 8 rojo. Pero, ¿no sería más fácil escribir simplemente COLOR ROJO o COLOR BLANCO? Para las personas, e incluso para el fiel Ron, las palabras son más fáciles de entender que los números. Los ordenadores, sin embargo, se las entienden mejor con números que con ninguna otra cosa, motivo por el que utilizamos números para cosas que deseamos que el ordenador comprenda. El BASIC es un lenguaje que, a menudo, utiliza números para reemplazar palabras.

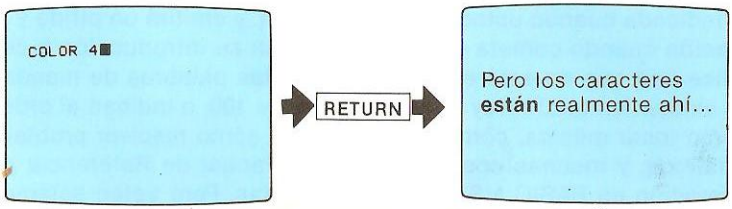
Cuando se acostumbre a dar mandatos a su ordenador, utilizará números para indicar los nombres, lugares, colores o sentencias que desee que el ordenador recuerde. Esto no es muy diferente de la utilización de números para señalar automóviles o casas, y para el ordenador es la forma más natural de charlar. Pronto le será muy fácil utilizar números y letras en sus programas BASIC.

Ahora sabemos que 8 significa rojo y 15 blanco. ¿Cuántos colores hay?

código	color	código	color	código	color	código	color
0	transparente	4	azul oscuro	8	rojo	12	verde oscuro
1	negro	5	azul claro	9	rojo claro	13	magenta
2	verde	6	rojo oscuro	10	amarillo oscuro	14	gris
3	verde claro	7	azul celeste	11	amarillo claro	15	blanco

En total hay dieciséis colores, y puede ser muy interesante jugar con ellos. Simplemente por diversión, pruebe esto:

```
COLOR 4
```



Caracteres azul oscuro sobre fondo azul oscuro. Hmm, la evidencia parece haberse desvanecido.

Utilice la tabla de códigos de colores para probar diferentes colores en la pantalla. Cambie libremente los colores de la pantalla de acuerdo con sus gustos personales.

Ahora, ¿qué le parece algo más complicado?

```
COLOR 1000
```

(Recuerde RETURN.) En la lista de colores no existe 1000, y el ordenador lo sabe. Un pitido rápido le indicará que el ordenador tiene una respuesta que darle. La respuesta es "Illegal function call", que es la forma que tiene el ordenador para decirle "Usted no puede engañarme".

De hecho, su ordenador Sony no puede engañarse nunca con mandatos. Suponga que desea escribir

```
COLOR 6
```

pero comete una equivocación y escribe

```
CILOR 6
```

Si usted pulsa, sin darse cuenta de la equivocación, la tecla RETURN, el ordenador volverá a emitir otro pitido. Esta vez el mensaje es "Syntax error".

Naturalmente, todo el mundo comete errores de mecanografía, ¡incluso los profesionales de Sony! Ésta es la razón por la que existen las teclas BS (retroceso), DEL (supresión), INS (inserción) y las de movimiento del cursor (consulte el Manual de Instrucciones). Usted aprenderá rápidamente cómo corregir equivocaciones.

¿Cuántas palabras BASIC hay?
Hasta ahora usted ha aprendido sus dos primeros mandatos BASIC: WIDTH y COLOR. Y ha visto lo fielmente que su ordenador Sony reacciona ante los mandatos correctamente introducidos. Él actuará en la forma indicada cuando usted los deletree bien, y emitirá un pitido y una explicación cuando cometa equivocaciones en su introducción o cuando utilice uno que no esté en la lista. ¿Cuántas palabras de mandatos BASIC existen en total? Hay aproximadamente 100, e indican al ordenador cómo tocar música, cómo trazar dibujos, cómo resolver problemas matemáticos, y muchas cosas más. En el Manual de Referencia para Programación en BASIC MSX se explican todas. Pero usted solamente necesita conocer unas pocas para comenzar a programar, por lo tanto no se preocupe si 100 le parece al principio una cantidad enorme.

Su ordenador Sony **le** enseñará también algunas cosas durante su operación: conoce aproximadamente 35 diferentes **mensajes de error**, como "Syntax error" e "Illegal function call" que acabamos de ver. El ordenador normalmente le dirá cuál es el error cuando usted le dé un mandato que él no entienda.

IMPRESIÓN DE RESPUESTAS

Ron tiene sus juegos favoritos, y dar la pata es uno de ellos. A su ordenador Sony le gustan otro tipo de juegos. Las matemáticas es una de las cosas que más le encantan (y que puede hacer muy bien).

Escribamos

```
PRINT 3+5
```

¿Ha encontrado el signo más (+)? Está sobre el signo de puntuación "=" y aparecerá en la pantalla cuando pulse las teclas [SHIFT] y "=" **al mismo tiempo.**

Ahora, pulse [RETURN] para introducir este mandato en el ordenador.

```
PRINT 3+5
 8
Ok
■
```

¡Muy bien! Usted ha utilizado el ordenador para resolver un problema e imprimir en la pantalla (visualizar) la respuesta. Intentemos otro ejemplo:

```
PRINT 100-10
```

Para el signo menos, como puede ver, no se utiliza la tecla SHIFT.

Ahora, utilice RETURN para introducir el mandato.

```
PRINT 3+5
 8
Ok
PRINT 100-10
 90
Ok
■
```

Naturalmente el ordenador da 90 como respuesta. Éstos son problemas sencillos, pero si usted desea sumar o restar billones y trillones, su ordenador lo hará instantáneamente.

¿Y con respecto a las multiplicaciones? No hay problema alguno. Lo único que tenemos que recordar es que el ordenador tiene un **signo especial** para ellas. En lugar de **7 × 9**, habrá que utilizar **7 * 9**.

Escriba PRINT 7 * 9, y pulse RETURN.

```
PRINT 7*9
 63
Ok
■
```

Para dividir, deberemos utilizar la tecla /. En lugar de PRINT 120 ÷ 40, escribiremos

```
PRINT 120/40
```

Y, naturalmente, el ordenador ofrecerá la respuesta correcta cuando pulse la tecla RETURN.

```
PRINT 120/40
 3
Ok
■
```

¿Desea utilizar números más grandes (7654321 × 18)?
¿Problemas complicados (2 + 9) × (6 − 8)?

Siga adelante e intente lo que guste. Algunas veces el ordenador puede ofrecer una respuesta que usted no entienda, pero no se preocupe. Efectúe las cosas paso por paso, piense en el resultado, y en pocos minutos podrá realizar maravillas.

NÚMEROS, LETRAS, Y VARIABLES

Cómo...

- Escribir palabras y frases
- Crear una variable (carácter con valor cambiante)
- Utilizar variables en matemáticas
- Asignar diferentes nombres a las variables

Hasta ahora hemos utilizado el mandato PRINT para que el ordenador resuelva problemas matemáticos, es decir, para que actúe como calculadora. Ahora vamos a ver otros tipos de cálculos que esta máquina puede realizar. Primeramente vamos a aprender algo más sobre PRINT y, al mismo tiempo, adquirir cierta práctica mecanográfica escribiendo

```
PRINT "JUAN Y MERCEDES"
```

Las **comillas (")** se encuentran sobre el signo, por lo que tendrá que pulsar al mismo tiempo la tecla SHIFT. Después, cuando haya escrito todo, pulse la tecla RETURN.

```
PRINT "JUAN Y MERCEDES"
JUAN Y MERCEDES
Ok
■
```

Dicho de otra forma, el ordenador "lee" las comillas, y entiende el mandato PRINT de la forma siguiente: debe imprimir lo que está dentro de las comillas, pero no las propias comillas. He aquí algunos ejemplos:

```
PRINT "ABC TO XYZ"
ABC TO XYZ

PRINT "How are you?"
How are you?
```

Naturalmente usted tiene que pulsar cada vez la tecla RETURN. Tenga siempre presente, y a partir de ahora puede ser que no se lo recordemos, que deberá pulsar la tecla RETURN cada vez que desee introducir un mandato.

NÚMEROS O LETRAS: ES IGUAL

Intentemos ahora algo ligeramente distinto.

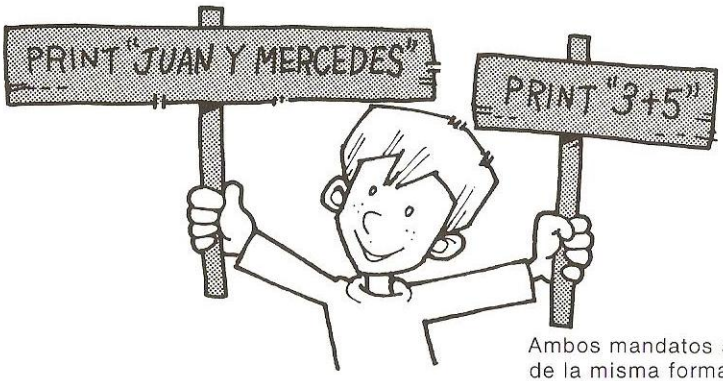
```
PRINT "3+5"
```

Esto es un poco diferente a lo que hemos hecho algunas páginas antes. En aquella ocasión, pedimos al ordenador que imprimiese (**PRINT** en inglés) **3+5**, y él amablemente imprimió 8 porque no había comillas.

Con comillas obtendremos lo siguiente:

```
PRINT "3+5"
3+5
Ok
■
```

En este caso, el mandato **PRINT**, utilizado junto **con comillas**, indica al ordenador que lea lo que está dentro de las comillas, y que lo visualice en la pantalla tal como está. Debido a las comillas, el ordenador leerá **3+5** como tres caracteres que tiene que imprimir, no como un problema matemático que debe resolver.

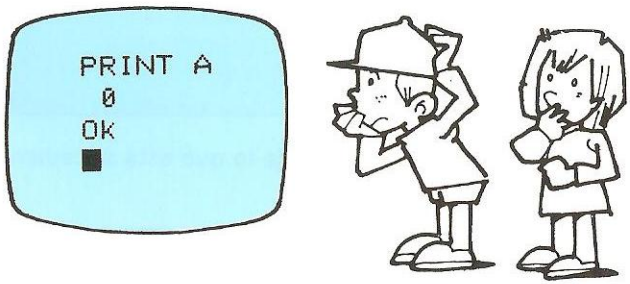


Ambos mandatos se ejecutan de la misma forma.

Esto es lo que sucede cuando hay un número dentro de las comillas.

CONVERSIÓN DE LETRAS EN VARIABLES

Ahora, vamos a aprender otra cosa interesante sobre el mandato PRINT. ¿Qué sucederá cuando después de PRINT haya una letra **sin** comillas?



¿Qué ha pasado? ¡No hay pitido! ¡No hay "Syntax error"! En vez de ello tenemos solamente un 0, y después Ok, que nos indica que el ordenador ha ejecutado nuestro mandato, y que está esperando la siguiente orden. Pero, ¿cuál **fue** nuestro mandato?

Si no hay mensaje de error significará que PRINT A es un mandato perfectamente aceptable. ¿Qué es lo que este mandato indica al ordenador que haga? Para descubrirlo, efectúe lo siguiente:

```
LET A=3
```

Al pulsar la tecla RETURN no hay mensaje de error, ni demasiada acción, solamente el amable Ok. El ordenador aceptó nuestro mandato, por lo que en su interior debe estar sucediendo algo.

¿Qué es lo que está sucediendo dentro del ordenador?

Ahora pongamos juntos estos dos mandatos:

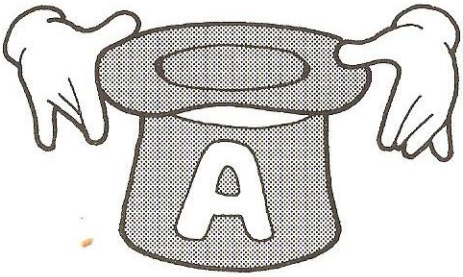
```
LET A=3
Ok
PRINT A
 3
Ok
■
```

Seguramente se habrá hecho una idea de lo que está sucediendo, pero probemos un ejemplo más:

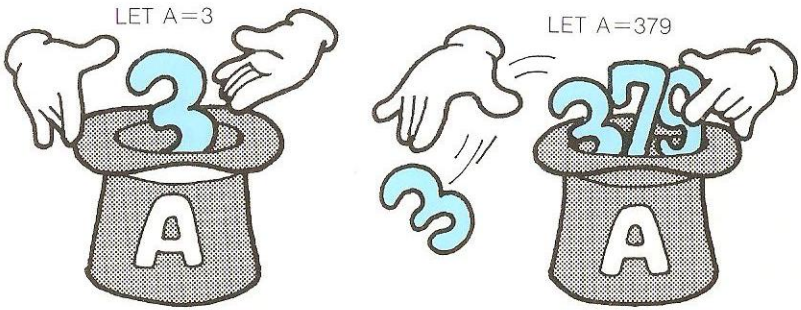
```
LET A=379
Ok
PRINT A
 379
Ok
■
```

A era **3** hace un momento, pero ahora es **379**. Si lo deseamos, podemos volver a cambiar el valor escribiendo simplemente **LET A =** y poniendo cualquier otro número. Ésta es la razón por la que denominamos **variable** a esta letra.

Por lo tanto, este tipo de **A** no es realmente una letra del alfabeto sino que, cuando aparezca después de PRINT sin comillas, será como una especie de recipiente que puede contener cualquier número que le permitamos (LET en inglés).



La chistera de un ilusionista es un buen ejemplo para las variables, porque podemos meter valores dentro de la misma, y sacarlos cuando lo deseemos.



Cuando escribamos LET A=3, habremos puesto un 3 en la chistera, y podremos sacarlo escribiendo simplemente PRINT A. También podremos cambiar el valor que esté dentro de la chistera por 379 con un solo y sencillo mandato (LET A=379), y éste es el valor que saldrá cuando lo solicitemos.

LET, por lo tanto, es un mandato que asigna un valor específico a nuestra variable. Las variables tienen muchas aplicaciones en el BASIC, y se utilizarán muy a menudo en este libro. Por razones de comodidad podemos escribir este mandato de forma más rápida omitiendo la palabra LET:

```
A=3
```

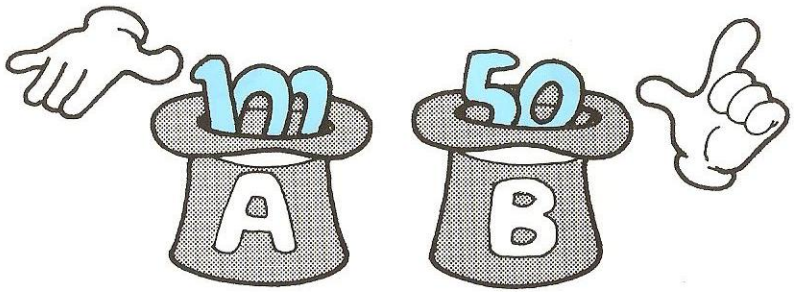
tiene el mismo significado que LET A=3.

Ahora, para asegurarse de que recuerda todo, hagamos un pequeño repaso:

```
A=100          Asignación del valor 100 to A
Ok
PRINT A        Visualización de A
 100           Aquí lo tenemos
Ok
PRINT "A"      Visualización de la letra "A" (no la variable A)
A              Aquí la tenemos
Ok
B=50           Asignación de un valor a B
Ok
PRINT B        Visualización de B
 50            Aquí lo tenemos
Ok
```

Anteriormente utilizamos el mandato PRINT A y el ordenador nos respondió con 0. Esto significa que todas las variables tendrán valor cero mientras no les asignemos valores diferentes.

Una variable puede representarse en la pantalla mediante cualquier letra. Hasta ahora hemos utilizado A y B, y les hemos asignado valores diferentes.



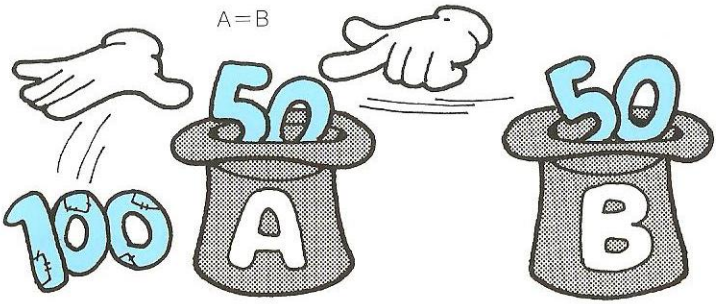
Veamos ahora lo que podemos hacer con estas variables.

Ya le hemos dicho al ordenador que A=100 y B=50. Ahora utilicemos tres mandatos: A=B, PRINT A, y PRINT B. Usted probablemente habrá adivinado el resultado, pero probémoslo para practicar.

```
A=B
Ok
PRINT A
 50
Ok
PRINT B
 50
Ok
■
```

Observe la primera línea. Hemos escrito A=B, y con ello hemos indicado al ordenador que asigne el valor de B a A. Anteriormente cambiamos el valor de A de 3 a 379, y el valor antiguo desapareció.

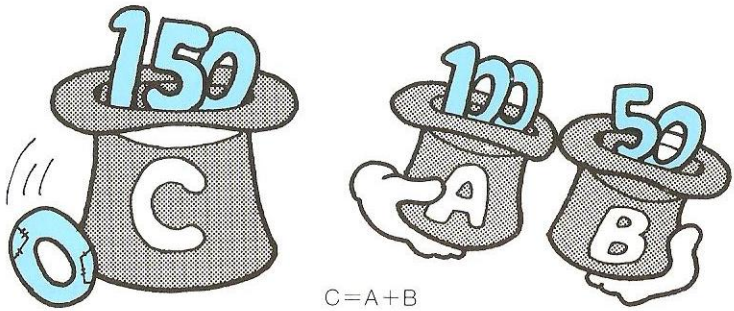
Aquí hacemos de nuevo la misma cosa, cambiar A de 100 a B, es decir, a 50.



Ahora vamos a efectuar algunas operaciones matemáticas. Escriba estos cuatro mandatos:

```
A=100
Ok
B=50
Ok
C=A+B
Ok
PRINT C
 150
Ok
■
```

Aquí tenemos tres recipientes, A, B y C, y al ordenador efectuará las operaciones matemáticas necesarias para encontrar lo que debe poner en el recipiente C.

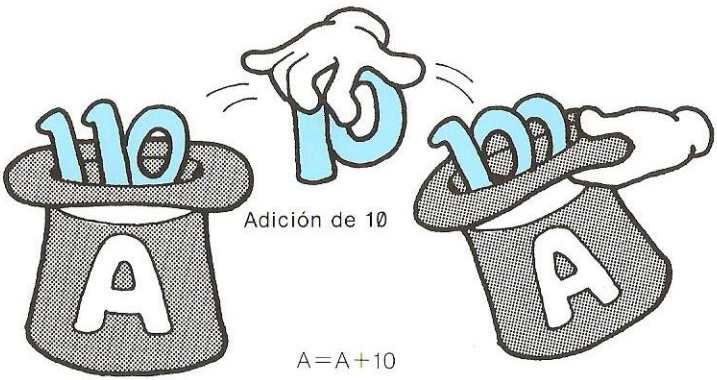


C=A+B

Aquí tenemos un ejemplo más interesante:

```
A=A+10
Ok
PRINT A
 110
Ok
■
```

¿Resulta confuso? A tenía un "valor asignado" de 100. Pero, al leer este mandato, el ordenador entendió nuestro deseo de "LET (permitir que) A tenga ahora un valor que sea 10 más el anterior". Por lo tanto A será ahora 110.



A=A+10

Si quiere, puede volver a probar una y otra vez el mismo mandato, y quizás imagine fácilmente el resultado. Sin embargo, su ordenador **recordará siempre** la cantidad asignada a cada variable.

Algo más acerca de las variables

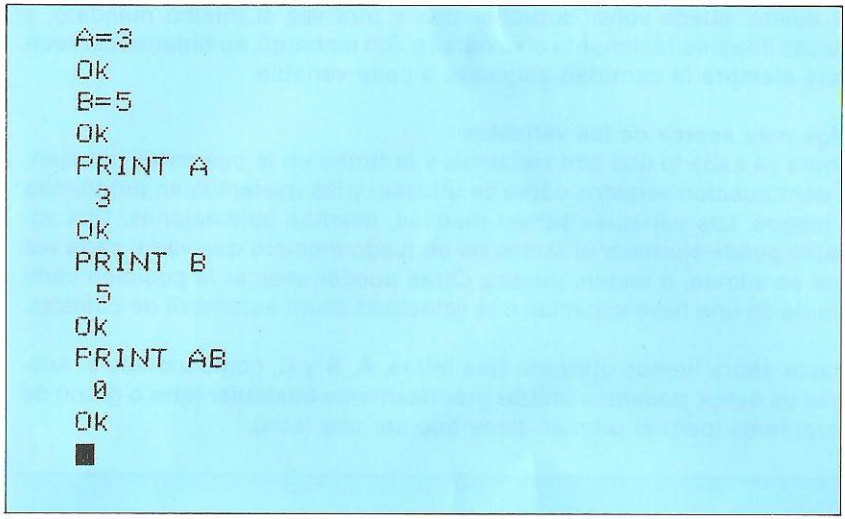
Ahora ya sabe lo que son variables, y la forma en la que éstas trabajan. A continuación veremos cómo se utilizan, y las usaremos en programas y juegos. Las variables tienen muchas, muchas aplicaciones. Una variable puede contener el tanteo de un juego (número que varía) cada vez que se sumen, o resten, puntos. Otras pueden indicar la posición cambiante de una nave espacial, o la velocidad de un automóvil de carreras.

Hasta ahora hemos utilizado tres letras, A, B y C, como variables. Además de éstas, podemos utilizar prácticamente **cualquier** letra o grupo de caracteres (pero el primero tiene que ser una letra).

```
UFO=5
Ok
PRINT UFO
 5
Ok
■
```

Como puede ver, un grupo de letras sin espacios entre ellas, en este caso OVNI, puede representar un **único** número.

Además, es muy importante recordar que cada letra o grupo diferente, como A, B, o AB, tiene distinto significado.

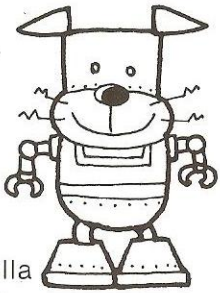


Aquí hemos asignado los valores 3 a A, y 5 a B, pero no hemos dado ningún valor a la variable AB, por lo que el ordenador visualizará AB con el valor 0 (¡**no** 35!). Una variable tendrá siempre valor 0 mientras no se le asigne otro valor diferente.

El nombre de su variable puede tener dos, tres, diez, o incluso 100 **caracteres**, y tales caracteres pueden ser tanto **números** como **letras**. Sin embargo, **el ordenador leerá solamente los dos primeros caracteres**, y el primer carácter tendrá que ser siempre una letra. Esto significa que para el ordenador POA, POB, PO1 y PO2 son iguales, es decir, estas variables tendrán para él un solo y único valor. Por lo tanto, tenga mucho cuidado cuando elija los nombres para sus variables.

Cualquier palabra que forme parte del lenguaje BASIC **no podrá** utilizarse como nombre de variable. Cuando en su ordenador introduzca palabras de lenguaje BASIC, él las leerá como **mandatos**, no como variables. Esto significa, por ejemplo, que LET, GOTO, y PRINT no podrán utilizarse como nombres de variables; tampoco GOTOA, BLET, ni cualquier otro de los mandatos que pronto aprenderemos.

HAGAMOS NUESTRO PRIMER PROGRAMA



Cómo...

- Planear un programa
- Escribir un programa sencillo
- Introducir un programa en su ordenador
- Corregir un programa
- Ejecutar su programa
- Leer sus programas completados en la pantalla
- Hacer que su ordenador memorice programas
- Borrar un programa de la memoria de su ordenador Sony

Hasta ahora hemos practicado lo básico utilizando un ordenador: poniendo letras y números en la pantalla, utilizando algunos símbolos que el ordenador entiende como parte de su propio "lenguaje", y dando instrucciones (mandatos) sencillos. En otras palabras, le hemos estado diciendo que **haga** cosas en la pantalla en la forma que nosotros queremos. Es decir, hemos aprendido cómo se comunican las personas y los ordenadores.

Pero esto no es mucho más de lo que una buena calculadora de bolsillo puede hacer. Una calculadora suma, resta, ejecuta operaciones matemáticas, y nos muestra, en su visualizador electrónico, todos los pasos y resultados que va realizando y obteniendo.

Lo que convierte a un **ordenador** en algo realmente excitante para usted (y en algo verdaderamente valioso para científicos y muchas personas más) es que puede ir más allá. Su ordenador Sony puede:

- Almacenar series largas de **instrucciones especiales** en su **memoria**.
- Utilizar las instrucciones para realizar diferentes tareas, juegos y trucos siempre que usted lo desee.
- Entender el resultado de cada función, y acarrear el resultado hasta la función siguiente.
- Combinar los resultados de todos los pasos anteriores a medida que continúa ejecutando los siguientes.

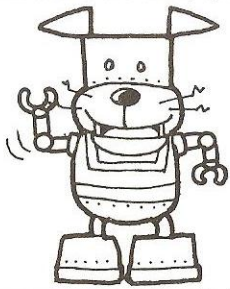
En otras palabras, puede seguir automáticamente un **programa** completo de funciones, y de forma mucho más rápida que cualquier calculadora.

Programación es el proceso de poner instrucciones en el ordenador para decirle qué funciones tiene que realizar y en qué orden debe hacerlo. Esto es lo que nos permitirá utilizar un ordenador para crear nuestros propios videojuegos, dibujar nuestras propias videoimágenes, y componer nuestra propia música electrónica. Y esto es lo que vamos a aprender en este capítulo: vamos a hacer nuestro primer programa.

Hacer un programa para un ordenador no es difícil. Usted ya sabe cómo dar algunos mandatos, y un programa no es más que una serie de mandatos. Lo importante en un programa es la **secuencia**: el **número** de los mandatos y el **orden** en que se dan.

Vamos a explicar los programas con un poco más de claridad utilizando otro ejemplo del comportamiento de un perro. Como Ron está descansando en estos momentos, vamos a intentar algunas cosas con **¡Superperro!**

Superperro es un tipo diferente de mascota: ¡es un robot! (¿Por qué no?, actualmente casi todo está automatizado, ¿no?)



Soy superperro,
y ¡puedo seguir programas!

Ron puede realizar muchas cosas, como usted bien sabe, pero realmente lo que ejecuta son secuencias muy sencillas. Si usted lanza una pelota, él la buscará, la recogerá con sus dientes, volverá, y la soltará. Ron entenderá nuestro mandato "¡dame la pata!", levantará una pata y nos la dará. Nos encanta Ron por su inteligencia y muchas otras razones, pero solamente puede ejecutar secuencias de tres o cuatro mandatos a la vez. Para que continúe haciendo cosas, usted tendrá que pasar a otro juego, es decir, un nuevo "programa".

PLANEAMIENTO DE PROGRAMAS

Superperro es diferente. Él tiene un microprocesador que recuerda cosas y toma decisiones, motivo por el que puede realizar muchos, muchos pasos más que Ron en un solo juego, de la misma forma que una calculadora puede resolver más problemas matemáticos que usted en el mismo intervalo de tiempo. Por ejemplo, podremos ordenar a Superperro:

1. ¡Busca y tráeme la pelota!
2. ¡Si la pelota es blanca, gira tres veces y ladra!
3. ¡Si la pelota es negra, échate al suelo durante 10 segundos!

El primer paso será muy fácil para Ron. Pero los otros dos son más complicados: el perro tendría que ser capaz de identificar si la pelota es blanca o negra, y después debería poder utilizar tal información para saber lo que hacer, y hacerlo correctamente.

Además, cada juego requiere un cómputo cuidadoso: Ron tendría que saber cuántas vueltas se le pidió que girase, o cuánto tiempo extactamente se le mandó que estuviese echado en el suelo. Y para efectuar correctamente todo esto, tendría que saber exactamente y en todo momento cuántas vueltas ha dado o cuántos segundos han transcurrido, y cuántos vueltas o segundos quedan antes de ladrar o traer la pelota. Sin embargo Superperro estará juzgando, tomando decisiones, y efectuando cómputos mientras esté ejecutando lo que se le haya mandado.

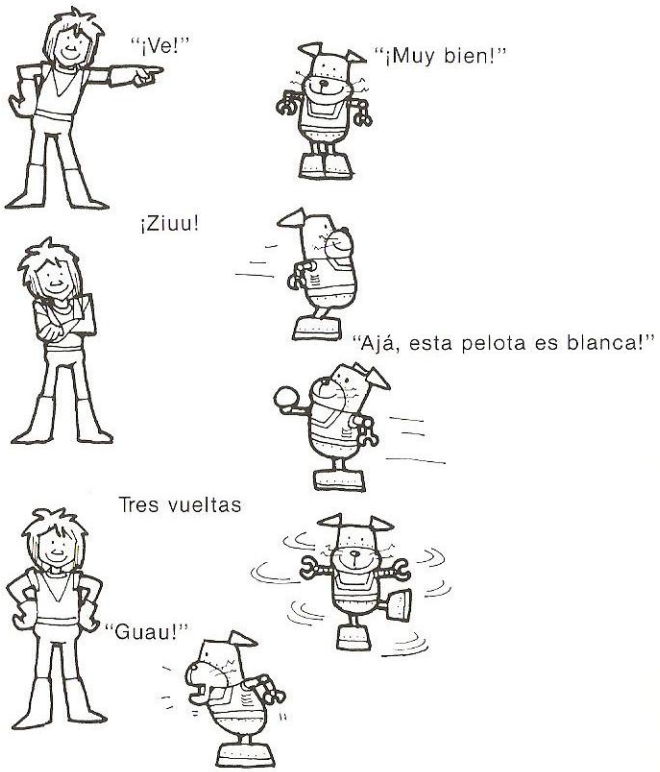
Entre Ron y Superperro existe una diferencia más. Nuestro Ron es un animal vivo, con un cerebro en cierta forma parecido al humano, por lo que puede hacer cosas por sí mismo, sin necesidad de nuestros mandatos. Si Ron ve un gato, ladrará; si ve fuego, escapará corriendo. Sin embargo, Superperro es una máquina. Puede realizar cosas muy complicadas, pero no puede "pensar" sobre ellas ni recordar lo que tiene que hacer. Superperro hará **solamente** lo que se le haya **programado** que haga, y no realizará nunca nada hasta que una persona accione, en una secuencia particular, sus motores, sensores, componentes, y el microprocesador (su cerebro).

Por lo tanto, cuando usted desee escribir un programa para que Superperro haga algo, primeramente tendrá que pensar en todo lo que desea que efectúe la máquina: **cada** paso, por pequeño que sea, y dónde debe estar éste situado en la secuencia. Los pasos del ejemplo anterior podrían ser algo así como:

1. Recibir e identificar el mandato "¡Busca y tráeme la pelota!"
2. Utilizar los sensores de imágenes para observar el movimiento de la pelota.
3. Accionar las patas mecánicas para moverse hasta la pelota.
4. Utilizar los sensores de imágenes para identificar la pelota como "blanca" o "negra".
5. Accionar las partes del cuerpo para tomar y devolver la pelota.
6. Elegir entre "girar" o "echarse".
7. Accionar las partes necesarias del cuerpo para realizar la acción requerida.

8. Calcular el número de vueltas dadas o el de segundos transcurridos durante la ejecución.
9. Ladrar/no ladrar.
10. Dejar de moverse cuando finalice el juego.
11. Prepararse para recibir el siguiente mandato del programa.

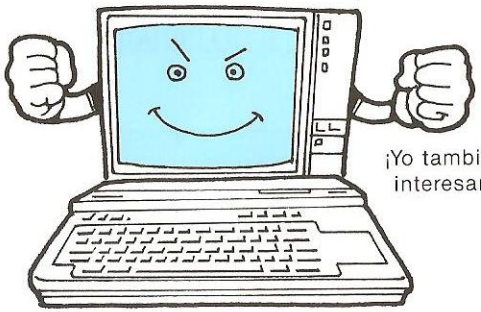
Usted deberá comprobar todos estos pasos para asegurarse de que no se ha olvidado de nada, o de que no ha cometido ningún error, y después introducirlos en el microprocesador de Superperro. Si usted piensa en todos los pasos, y los enumera en la secuencia correcta, Superperro ejecutará todo correctamente cada vez que realice el juego. Usted puede "ejecutar el programa" una, diez, o cientos de veces, y el resultado será siempre el mismo.



Por lo tanto, hay dos reglas importantes a la hora de escribir un programa: pensar en **todo** lo que el ordenador tiene que hacer para realizar exactamente lo que se le pide, y poner todos los pasos en el **orden correcto**. La única "técnica" que usted necesitará será el razonamiento normal y lógico porque los ordenadores, al igual que cualquier otra máquina, trabajan como la hace su cerebro: lógicamente.

Ahora vamos a escribir el primer programa en su ordenador Sony y veremos lo sencillo que realmente es.

ESCRITURA DE PROGRAMAS: TAN FÁCIL COMO CONTAR UNO, DOS, TRES



¡Yo también sé algunos trucos interesantes!

"Secuencia" en un programa de ordenador significa solamente que los mandatos deben escribirse en el orden de uno, dos, y tres. Recuerde nuestro juego para Superperro: había tres partes separadas, y escribimos un número delante de cada una de ellas.

1. ¡Busca y tráeme la pelota!
2. ¡Si la pelota es blanca, gira tres veces y ladra!
3. ¡Si la pelota es negra, échate al suelo durante 10 segundos!

Estos pasos están en orden lógico: nosotros deseamos ante todo que nos traiga la pelota, por lo tanto el mandato 1 está en primer lugar. (El orden de los mandatos 2 y 3 puede invertirse sin que el juego cambie.)

Nosotros tendremos que introducir nuestros mandatos en el ordenador de la misma forma: los escribiremos en la pantalla en el mismo orden en el que habrán de ejecutarse, y pondremos un número a cada mandato. Nuestra primera "tarea" para el ordenador, nuestro programa, será la visualización en la pantalla de las palabras "JUAN Y MERCEDES".

```
10 PRINT "JUAN"
20 PRINT "Y"
30 PRINT "MERCEDES"
```

Usted ya conoce el mandato PRINT, y aquí lo utilizamos tres veces, es decir, una para cada una de las tres palabras que deseamos imprimir. Naturalmente, nuestra secuencia tiene el mismo orden que el de las palabras. La única diferencia, a como escribimos anteriormente los mandatos PRINT, es que cada uno comienza con un número: 10, 20 y 30. Esto es muy importante: estos números indican al ordenador **qué mandato ejecutar a continuación** a medida que realice cada uno de los pasos del programa. En otras palabras, qué secuencia seguir.

Ahora, mecanografíe el primer mandato: 10 PRINT "JUAN". No se olvide: al final, pulse una vez RETURN.

1 0 [ ] P R I N T [ ] " J U A N " RETURN

¿Qué tal?

```
10 PRINT "JUAN"
■
```

Muy bien. Ya ha introducido su primer mandato, el cursor se ha movido hasta la línea siguiente, y ahora podrá introducir los dos siguientes exactamente de la misma forma:

```
10 PRINT "JUAN"
20 PRINT "Y"
30 PRINT "MERCEDES"
■
```

Aquí tiene su programa listo para utilizar.

CORRECCIÓN DE PROGRAMAS: COMPROBACIÓN DE FALLOS

Antes de "ejecutar" nuestro programa, tendremos que realizar un último y muy importante paso: **asegurarnos** de que hemos introducido correctamente cada parte del programa. Recuerde que su ordenador Sony es una máquina, y que solamente podrá leer mandatos escritos en lenguaje BASIC. Si hay una equivocación, aunque solamente sea un carácter o un espacio erróneamente introducido, el mandato no se parecerá a ninguna palabra del BASIC. Esto parará todo el programa, o hará que el ordenador ejecute algo erróneo. Por ejemplo:

```
10 PRIMT "JUAN"
     ↑
     └──── Equivocación
```

Las equivocaciones como ésta se denominan **fallos** en el programa. El mandato PRINT deberá deletrearse correctamente, o el ordenador no entenderá que lo que usted le está dando es un mandato.

Alégrese de haber encontrado ahora la equivocación: es mejor encontrar y corregirla ahora que más adelante.

Usted ya sabe cómo mover el cursor, utilizando las cuatro teclas del mismo con flechas sobre ellas. El cursor se encuentra ahora en la parte inferior del programa, por lo que tendremos que moverlo hacia arriba hasta "10" con la tecla ▽. Después habrá que moverlo seis espacios hacia la derecha con la tecla ◁, para que quede sobre la letra equivocada M.

```
10 PRIMT "JUAN"
```

Coloque el cursor aquí.

Para corregir, pulse simplemente la tecla correspondiente a la letra correcta.

```
10 PRINT "JUAN"
```

Pulse N.

Ahora pulse RETURN una vez ... y el fallo habrá quedado corregido.

```
10 PRINT "JUAN"
20 PRINT "Y"
```

El cursor se moverá hasta aquí después de haber pulsado RETURN.

Si hay otras equivocaciones, corríjalas moviendo el cursor de la misma forma. (Si es necesario, consulte el **Manual de Instrucciones** para ver cómo añadir un carácter o un espacio olvidado, o cómo borrar los inecesarios.) Cuando haya corregido todas las equivocaciones, utilice las teclas de las flechas para mover el cursor hasta la parte inferior del programa para que la pantalla muestre lo siguiente:

```
10 PRINT "JUAN"
20 PRINT "Y"
30 PRINT "MERCEDES"
■
```

Quizás no haya cometido equivocación alguna en este sencillo programa. Pero es muy importante **comprobar** cada vez que se escriba uno. Muy pronto, sus programas contendrán más mandatos, y muchos de ellos serán más largos. Cualquier fallo, por grande o pequeño que sea, puede parar un programa, y **cualquiera**, incluso los expertos de Sony, cometen a veces equivocaciones de deletreo.

Una cosa es cierta: si hay una equivocación, su ordenador Sony tarde o temprano se lo dirá. En la pantalla aparecerá un **mensaje de error**, el programa se parará antes de lo deseado, o el ordenador hará algo diferente a lo que usted esperaba. En tal caso, usted tendrá que comprobar toda la lista de mandatos, localizar el fallo, y corregirlo. Por lo tanto, lo mejor es comprobar y "depurar los fallos" de su programa al comienzo, cuando esté mecanografinado los mandatos.

Reglas de programación

- Piense en **cada uno de los pasos que necesitará el programa** para hacer que el ordenador realice lo que usted desee.
- Asegúrese de haber enumerado todos los mandatos en la **secuencia lógica** que dé el resultado correcto.
- Utilice siempre un **número de secuencia** al **comienzo** de cada línea de mandato.
- Después de haber introducido cuidadosamente sus mandatos **compruebe cada uno de ellos** para ver si hay "fallos", y corrija los que encuentre.

EJECUCIÓN DE PROGRAMAS

Ahora, después de haber introducido los mandatos y haber efectuado las correcciones necesarias, podrá **ejecutar** (RUN en inglés) el programa. Para ello tendremos que introducir el mandato siguiente en la pantalla:

```
RUN
```

Y esto es lo que el ordenador hará con su programa cuando lo haya introducido, y pulse RETURN. Su pantalla deberá visualizar lo siguiente:

```
10 PRINT "JUAN"
20 PRINT "Y"
30 PRINT "MERCEDES"
RUN
JUAN
Y
MERCEDES
Ok
■
```

Su ordenador Sony empleará solamente unos segundos para "leer" los mandatos e imprimir.

Mecanografíe de nuevo el mandato RUN, inmediatamente después del cursor, y pulse otra vez [RETURN]. El resultado es el mismo ¿no? El resultado para **este** programa será siempre el mismo, hasta que usted cambie o borre los mandatos. Ejecute varias veces el programa, y lo comprobará. Siempre aparecerá JUAN Y MERCEDES. Esto significa que su ordenador Sony **recuerda** el programa.

Usted podrá comprobar en cualquier momento el contenido de la memoria introduciendo **LIST** (y pulsando después [RETURN]). Nosotros utilizaremos el **mandato LIST** cada vez que escribamos un programa.

Lo más útil de su ordenador Sony es que puede recordar mandatos. Todo lo que usted tendrá que hacer es decirle que los recuerde, introduciendo **un número antes de cada mandato.**

```
10 PRINT "JUAN"
```

Si introduce un número de línea, se tratará de un programa.

Este mandato tiene un número, por lo que el ordenador sabrá que es parte de un programa, y memorizará el mandato. 10 es el **número de línea**, y puede ser cualquier entero de 0 a 65529.

¿Cuánto tiempo recordará el ordenador los mandatos de su programa? ¡Siempre!, o hasta que usted le diga que los olvide. Para hacer que olvide mandatos:

```
NEW
```

Comprobemos si el ordenador recuerda introduciendo algunos mandatos numerados, y utilizando después LIST. ¿El ordenador los ha listado todos? Muy bien. Ahora introduzca NEW.

```
LIST
10 PRINT "JUAN"
20 PRINT "Y"
30 PRINT "MERCEDES"
Ok
NEW
Ok
■
```

La pantalla muestra sus mandatos, pero el ordenador los ha olvidado. Introduzca de nuevo LIST. El programa se ha borrado de la memoria.

Existen otras dos formas de hacer que el ordenador olvide mandatos. Una es pulsar el botón RESET, y la otra es desconectar la alimentación del ordenador. **No haga ninguna de estas dos cosas** sin tener antes la seguridad de que desea que el ordenador olvide.

Más adelante aprenderemos cómo **guardar** programas, utilizando un magnetófono, para poder utilizarlos de nuevo semanas o años más tarde. Hasta entonces, su ordenador olvidará todo cuando usted desconecte la alimentación.

GRÁFICOS: ESTRELLAS CENTELLEANTES

Cómo...

- Utilizar más palabras del BASIC
- Borrar la pantalla
- Trazar puntos y líneas
- Hacer que el ordenador repita una acción
- Utilizar una variable
- Utilizar un número aleatorio
- Utilizar nuevos mandatos COLOR

UN PROGRAMA CON GRÁFICOS

Utilicemos algunas palabras nuevas del BASIC y aprendamos lo que hacen. Escriba estos mandatos:

```
10 SCREEN 2
20 PSET (100,100)
30 PSET (150,100)
40 PSET (100,150)
50 PSET (150,150)
60 GOTO 20
```

Éste es un **programa**, un juego de pasos que ordenador puede utilizar para hacer algo. Ahora que el programa está en la memoria, introduzca el mandato RUN, y vea lo que este programa hace.

Se han borrado todas las letras, y en la pantalla aparecen cuatro puntos blancos. El cursor ha desaparecido, y si usted pulsa una tecla, no aparece nada en la pantalla ¿no? El ordenador está ocupado ejecutando un programa. ¿Por qué emplea tanto tiempo?

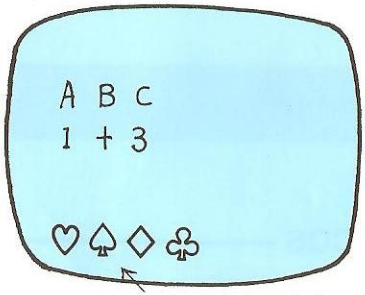
Veamos lo que significa este programa. Primeramente pulsemos CTRL y STOP para detener el programa. Los puntos desaperecerán y reaparecerá el cursor. Ahora introduzca LIST, y podremos ver el programa línea por línea. Observe en primer lugar la línea 10.

SCREEN 2

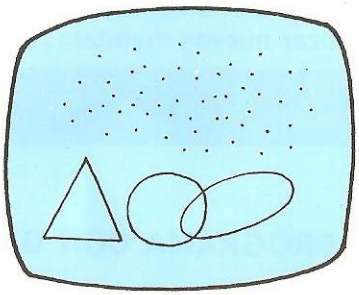
es el mandato que le permite comenzar a trazar **gráficos** (dibujos) en la pantalla. Los gráficos son puntos, líneas, círculos, y otras figuras o diseños visualizados en la pantalla. (Lo que no sean gráficos, es decir, los números, las letras y los símbolos, se denominan **caracteres**.)

Los mandatos SCREEN indican al ordenador la forma en que debe visualizar la información en la pantalla. Hay dos tipos de mandatos SCREEN.

SCREEN 0 or SCREEN 1 es para **caracteres**.



SCREEN 2 es para **gráficos**.



(Los símbolos del teclado son caracteres.)

SCREEN 2, por consiguiente, hace que el ordenador visualice gráficos. Cuando usted introdujo el mandato RUN, los caracteres desaparecieron de la pantalla, porque SCREEN 2 mandó al ordenador que visualizase gráficos. Cuando pulsó CTRL y STOP, canceló el mandato de gráficos, y los caracteres reaparecieron. Ahora que usted entiende la primera línea del programa, podremos continuar.

```
20 PSET (100,100)
30 PSET (150,100)
40 PSET (100,150)
50 PSET (150,150)
```

PSET es el mandato para poner **puntos** (gráficos, no ortográficos) en la pantalla, y los números entre paréntesis ( ) indican al ordenador dónde ponerlos.

Repetición, repetición, repetición, repetición ...
La última línea del programa contiene un mandato muy importante:

```
60 GOTO 20
```

GOTO significa (en inglés) "ir a", es decir, este mandato hace al ordenador **ir a** la línea 20 (regresar) y volver a empezar. Cuando regrese a la línea 20,

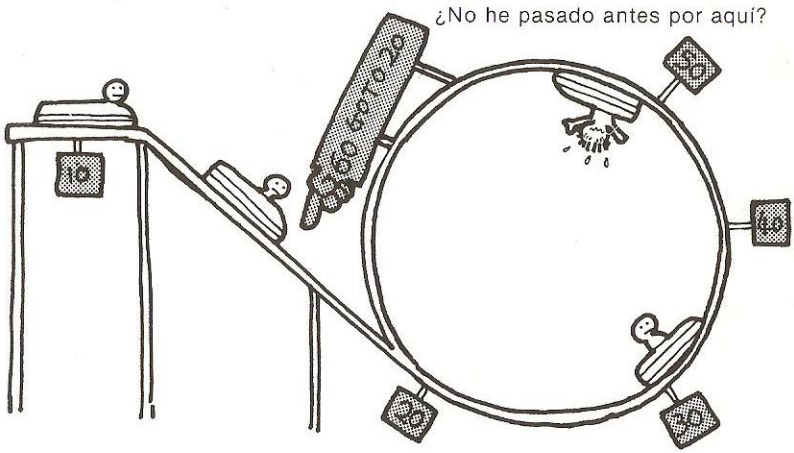
```
20 PSET (100,100)
```

el ordenador repetirá, naturalmente, el programa. En la línea 60, irá de nuevo a la línea 20, repetirá el programa, volverá a 20, repetirá el programa ...

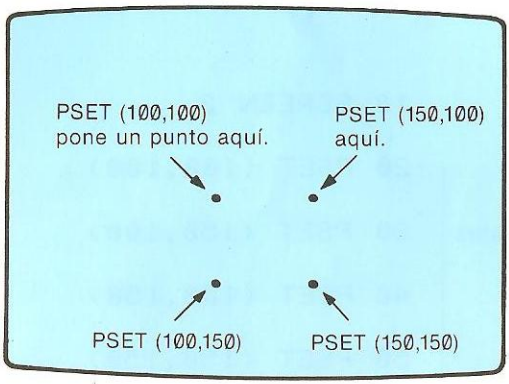
Esta parte se repetirá indefinidamente.

```
10 SCREEN 2
↓
20 PSET (100,100)
↓
30 PSET (150,100)
↓
40 PSET (100,150)
↓
50 PSET (150,150)
↓
60 GOTO 20
```

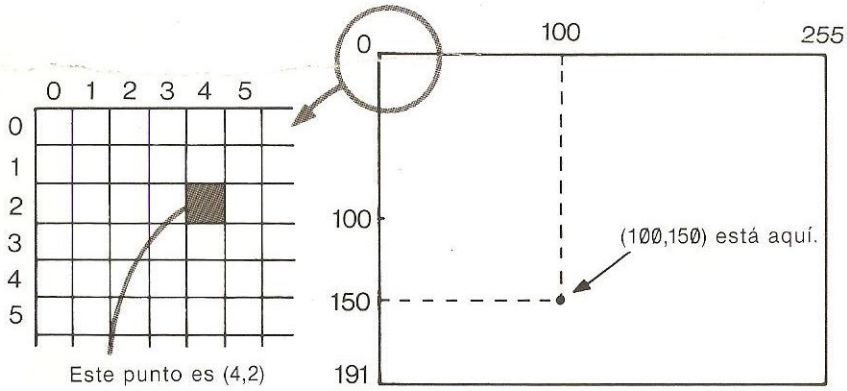
La sección de repetición de un programa se denomina **bucle**, y es una herramienta común y muy útil para la programación de ordenadores. (Afortunadamente, no todos los bucles se repiten indefinidamente.)



Veamos ahora cómo los mandatos PSET ponen puntos en la pantalla.



¿Se ha dado cuenta de cómo trabajan los números? Para ambos puntos de la izquierda, el primer número es 100, y para ambos de la derecha es 150. Aquí tenemos una imagen ampliada de la pantalla.



Para decirle a su ordenador Sony que ponga un punto, tendrá que darle dos números de posición, el primero para la posición izquierda-derecha, y el segundo para arriba-abajo. Hay 256 posiciones diferentes de izquierda a derecha (0 a 255), y 192 de arriba abajo (0 a 191). Por lo tanto, podremos poner un punto en cualquier lugar de la pantalla, entre (0,0) y (255,191).

Gráficos y variables

Como ya sabe, cualquier número puede reemplazarse por una variable. Esto es cierto también para los números de PSET. Si reemplazamos el número izquierda-derecha por la variable X, y el de arriba-abajo por la variable Y, nuestro mandato pasará a ser

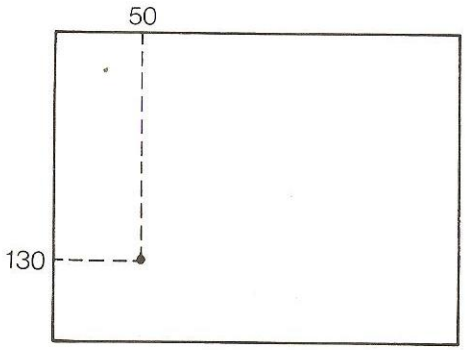
PSET (X,Y)

y podremos utilizar cualquier número que deseemos para X e Y, a fin de hacer que los puntos vayan a donde querramos.

Aquí tenemos la forma de utilizar las variables para un solo punto. Primeramente utilice el mandato NEW para hacer que el ordenador olvide el programa anterior, y después introduzca este nuevo programa:

```
10 SCREEN 2
20 X=50:Y=130
30 PSET (X,Y)
40 GOTO 30
```

Introduzca el programa en la memoria pulsando RETURN, y después utilice el mandato RUN. Su ordenador comenzará en la línea 10, que borrará la pantalla y la dejará dispuesta para **gráficos**. En la línea 20 aprenderá dos variables, y en la línea 30 las utilizará para poner un punto en la pantalla.



Como X es 50 e Y es 130, en la posición (50,130) de la pantalla aparecerá un punto blanco. Después, el ordenador leerá la línea 40, que le manda volver a la 30. Esto significa que la única forma de detener este programa bucle será pulsando CTRL y STOP.

¿Por qué hemos utilizado variables en vez de escribir simplemente PSET (50,130)? Y ¿por qué hemos hecho que el programa se repita indefinidamente con la línea 40? Estos pasos no serán necesarios cuando deseemos poner un solo punto blanco en la pantalla durante muy poco tiempo, pero son una forma muy buena de visualizar muchos puntos con un programa muy corto. Y eso es lo que vamos a hacer a continuación, después de la nota siguiente sobre puntuación y deletreo.

Nota: Tenga mucho cuidado con la puntuación.
El programa que estamos utilizando ahora tiene tres tipos diferentes de **puntuación** , (coma), : (dos puntos) y ( ) (paréntesis). Estos signos de puntuación son **caracteres**, y **tienen que ponerse en los lugares apropiados**, al igual que las letras de las palabras de los mandatos. Recuerde que su ordenador Sony no entenderá un mandato si hay algún error de deletreo, y la puntuación es parte de dicho deletreo.

Cada signo de puntuación tiene su propio significado en el BASIC. La coma indica que ha finalizado un número y que comienza otro. Los dos puntos significan que ha finalizado un mandato, y que en la misma línea hay otro. Los paréntesis son necesarios cuando se utilizan juntos dos o más números, o variables, y los paréntesis **siempre van emparejados**, primero el izquierdo "(" y después el derecho ")". Más adelante se indicarán algunos otros signos de puntuación utilizados en los mandatos del BASIC.

Como hemos dicho anteriormente, todo el mundo comete alguna vez errores de deletreo. Cuando suceda esto, su ordenador Sony no entenderá, por lo general, el mandato, y visualizará un mensaje de error. Por ejemplo,

```
Syntax error in 30
```

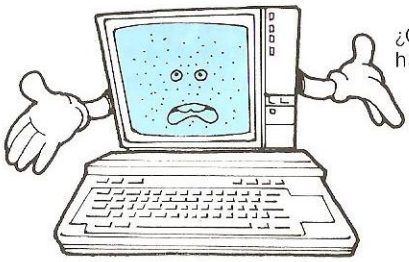
significa que puede haber un error de deletreo en la línea 30.

NÚMEROS ALEATORIOS

Utilice el mandato NEW para borrar la memoria del ordenador, y después introduzca en la misma el programa siguiente.

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)
50 GOTO 20
```

Cuando lo ejecute (RUN), la pantalla se llenará de puntos y más puntos, hasta que lo detenga, es decir, hasta que pulse CTRL y STOP.



¿Qué es lo que se propone hacer con mi cara?

Vamos a explicar las dos nuevas líneas del programa.

```
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
```

Como bien puede verse, estas líneas asignan valores a las variables X e Y. Pero ¿qué significan estos valores? Primeramente, observemos RND(1). RND significa **aleatorio** (en inglés RaNDom), y es una de las **funciones** del BASIC. Para entender esta **función de número aleatorio**, pruebe este mandato.

```
X=RND(1):PRINT X
```

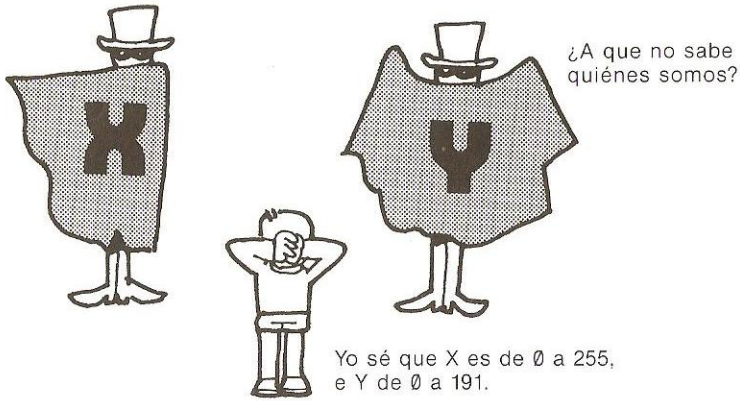
Después de haber introducido el mandato, pulse RETURN: aparecerá una fracción de 14 dígitos, un número inferior a 1 pero mayor que 0, y que comienza por un **punto decimal** (en notación inglesa se utiliza el punto en vez de la coma decimal). Por ejemplo:

```
X=RND(1):PRINT X
 .59521943994623
Ok
■
```

Pruebe este mandato de nuevo. ¿Ha cambiado el número? Pruébelo de nuevo. Su ordenador tiene una larga lista de números aleatorios, elegirá uno tras otro, y los utilizará para X.

Volviendo a la línea 20 del programa, tenemos (RND(1) * 256), que significa que el número aleatorio se multiplica por 256. Pero si multiplicamos estas fracciones por 256, los resultados serán normalmente fracciones también, mientras que nuestras posiciones de "puntos" deben ser números **enteros**, como 1, 30, o 192, ya que los números de ubicación en la pantalla son solamente números enteros fijos. Ésta es la razón por la que hemos utilizado el mandato **INT** al comienzo. INT significa **entero** (INTeger en inglés), y corta el punto decimal y los números que haya tras él, para que cada resultado sea un número entero. Ahora entendemos el valor de X. Este valor cambiará cada vez que el programa repita el bucle.

Cada vez que su ordenador lea X=INT(RND(1) * 256), redondeará X a un número entero entre 0 y 255, pero usted nunca sabrá qué número es. Y cada vez que lea la línea 40, redondeará, de la misma forma, Y a un número entero entre 0 y 191.



La línea siguiente ya la entendemos,

```
40 PSET (X,Y)
```

aunque no sepamos los valores de X e Y. El ordenador no nos dirá los valores porque no hay mandato PRINT. En vez de ello, seguirá visualizando más y más puntos en la pantalla, en los lugares que indiquen los números enteros de las posiciones izquierda-derecha y arriba-abajo.

Hasta ahora hemos explicado solamente cómo aparece un punto en la pantalla. ¿De dónde continúan viniendo los otros? Aquí es donde trabaja el bucle. La línea siguiente,

```
50 GOTO 20
```

devuelve el programa a

```
20 X=INT(RND(1)*256)
```

Entonces, X e Y adquirirán cada vez nuevos valores, por lo que

```
40 PSET (X,Y)
```

pondrá un punto en un nuevo lugar, y veremos aparecer punto tras punto por toda la pantalla.

Programas: dos tipos de mandatos

Un programa, como hemos dicho, es **una serie de mandatos en orden numérico fijo** introducidos en la memoria del ordenador. Un programa puede tener cientos o miles de líneas, es decir, mandatos, pero en la mayoría de los casos no se necesitarán tantas; con el programa anterior acabamos de dibujar miles de puntos con solamente cinco líneas.

Una de las cinco líneas de mandatos, GOTO 20, no hace en realidad más que decirle al ordenador **dónde tiene que ir a continuación**. Ésta fue la línea que hizo que nuestro programa tan corto realizase tanto, formando un bucle que se repitió indefinidamente.

Ahora podemos ver que existen dos tipos de mandatos:

1. **Mandatos que indican al ordenador lo que tiene que hacer, tales como**:

   PRINT PSET WIDTH COLOR, etc.,

y

2. **Mandatos que cambian el orden del programa**

   GOTO IF—THEN FOR—NEXT, etc.

En este segundo tipo hay solamente unos pocos mandatos, y más adelante aprenderemos más sobre ellos. Estos mandatos son quizás las herramientas más interesantes para computar: al igual que GOTO, que utilizamos para formar un bucle, todos ellos se utilizan para hacer **pequeños** pero **muy potentes** programas.

CÓMO PONER COLOR EN NUESTROS PROGRAMAS

Vamos a añadir algo de viveza al programa de puntos: ¡el color! Para empezar, visualicemos el programa que está en la memoria del ordenador, con el mandato LIST. (Si ha desconectado la alimentación, tendrá que volver a escribir el programa.)

```
LIST
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)
50 GOTO 20
```

Ahora cambie la línea 40 a: 40 PSET (X,Y),2

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)■
50 GOTO 20
```

Mueva el cursor hasta esta posición, e introduzca **,2**. (No se olvide de la coma porque es muy importante.) Pulse una vez RETURN para finalizar la introducción, y mueva el cursor hasta debajo del programa. Ahora introduzca el mandato RUN.

Los puntos han cambiado de blanco a verde, porque 2 es el código para tal color. Utilizando la tabla de colores de la página 12, podremos elegir diferentes colores. También, podemos hacer del color una variable y dejar que el ordenador cambie de color a color.

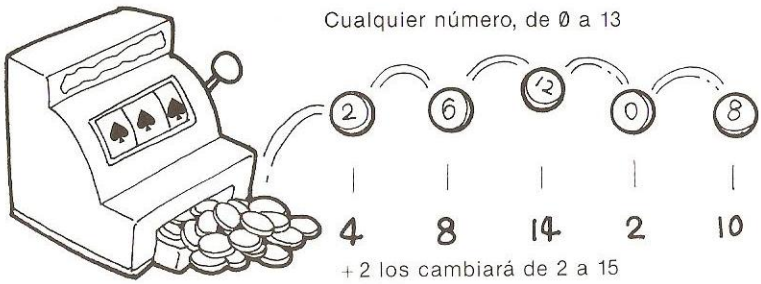
Vuelva a la línea 40 y cambie el código de color por la variable C.

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y),C
50 GOTO 20
```

Ahora tendremos que asignar un valor a la variable C, que es nuestro mandato para el color que deseemos. Pero será mucho más interesante si, de nuevo, dejamos que el ordenador asigne aleatoriamente el valor. Aquí está la línea de mandato que hará tal cosa, pero no la introduzca todavía.

```
35 C=INT(RND(1)*14)+2
```

¿Qué significa esto? Éste es un mandato de número aleatorio como los de X e Y, con un paso más al final. Con este mandato, el programa elegirá un número aleatorio entre 0 y 13, y después le añadirá 2 (+2). Esto significa que el número estará ente 2 y 15. Sin embargo, de 0 a 15 hay 16 colores. ¿Por qué este mandato descarta los colores 0 y 1? Porque 0 indica transparencia y usted no podrá ver puntos transparentes, y 1 indica negro por lo que usted tampoco podrá ver los puntos si el fondo se vuelve negro.



¿En qué lugar del programa deberemos poner esta línea? Debe estar antes de que se visualicen los puntos de la línea 40, motivo por el que nuestra nueva línea pasará a tener el número 35. (Y ésta es la razón por la que nuestras líneas de mandatos están numeradas 10, 20, 30, etc., para dejar espacio a fin de poder introducir los nuevos mandatos que deseemos añadir más tarde.) Ahora, introduzcamos la línea en el programa.

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y),C
50 GOTO 20
35 C=INT(RND(1)*14)+2
```

30, 40, 50, 35. ¿Usted cree que el ordenador entenderá? Naturalmente que entenderá, porque sabe contar, y dispondrá automáticamente las líneas por orden numérico. Esto es fácil de comprobar: simplemente introduzca de nuevo el mandato LIST.

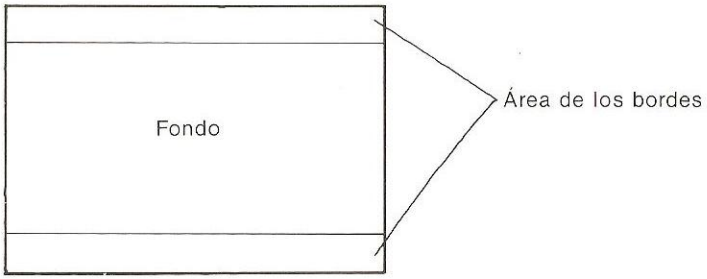
```
LIST
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
35 C=INT(RND(1)*14)+2
40 PSET (X,Y),C
50 GOTO 20
```

Ahora podemos convertir nuestros puntos en **estrellas**. Como las estrellas solamente pueden verse de noche, hagamos que la pantalla se vuelva negra. Introduzca esta nueva línea:

```
5 COLOR 15,1,1
```

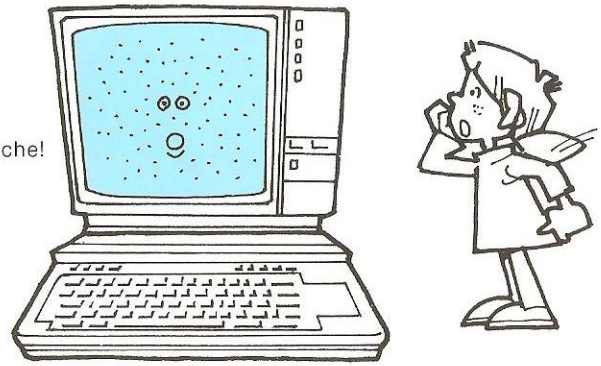
Nuevos mandatos para color y líneas

La línea anterior es una nueva forma de utilizar el mandato COLOR. Esta línea tiene tres códigos: 15 establece el color de los **caracteres** (o de primer plano), el primer 1 establece el de **fondo**, y el último 1 establece el del área de los **bordes**. (En este ejemplo, el fondo y los bordes tienen el mismo color. Cualquier número valdrá, pero probemos con éstos.)

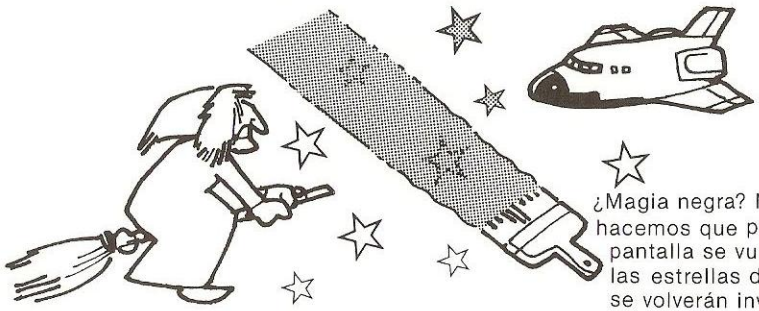


Ahora estamos preparados: RUN.

¡Estrellas en la noche!



Pero todavía no hemos finalizado. Podemos hacer que las estrellas parezcan todavía más naturales. Primeramente controlemos el número de estrellas que deseemos, antes de terminar con toda una galaxia. Utilice, como de costumbre, CTRL y STOP.



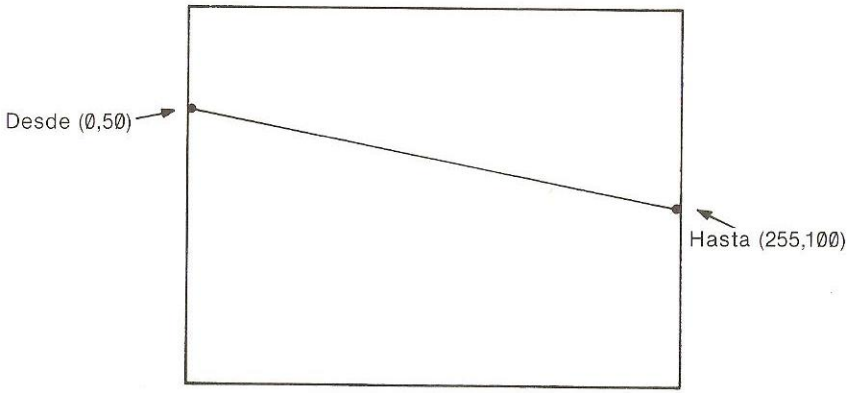
¿Magia negra? No, pero si hacemos que parte de la pantalla se vuelva negra, las estrellas de tal parte se volverán invisibles.

Trazar **líneas negras** en la pantalla es, naturalmente, una forma de hacer que parte de la pantalla se vuelva negra. El mandato PSET es solamente para puntos: para líneas, utilizaremos **LINE**. Aquí tenemos un mandato que indica al ordenador dónde tiene que trazar una línea, y qué color utilizar para ella.

```
LINE(0,50)-(255,100),2
```

A propósito, para el " - " (guión) central, utilice el signo menos.

El guión significa "desde/hasta", y este mandato trazará una línea **desde** la posición (0,50) **hasta** la posición (255,100). Usted sabe, naturalmente, que la última parte de este mandato es el color verde. Si utiliza este mandato LINE, el programa trazará una línea verde como ésta:

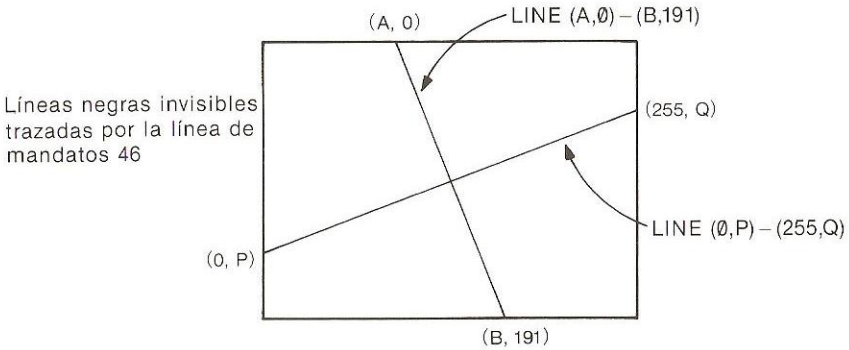


Ahora que ya comprende el mandato LINE, podrá utilizarlo para mejorar el programa de estrellas. ¿Desea utilizar una línea negra para hacer algunas estrellas invisibles? Esto significa que tendremos que cambiar el código de color a 1. Además, tendremos que utilizar variables de nuevo para que aparezca una serie aleatoria de líneas, junto con otra serie, también aleatoria, de puntos. Introduzca estas tres nuevas líneas en su ordenador:

```
42 A=INT(RND(1)*256):B=INT(RND(1)*256)
44 P=INT(RND(1)*192):Q=INT(RND(1)*192)
46 LINE (A,0)-(B,191),1:LINE (0,P)-(255,Q),1
```

Estas tres líneas ocuparán más de una línea en la pantalla, pero esto no es problema alguno. El ordenador la escribirá automáticamente en dos renglones, pero seguirá siendo **una sola línea**, porque solamente habrá **un número de línea** (42, 44 y 46).

El número de línea 46 contiene dos mandatos LINE, con cuatro nuevas variables. Las líneas 42 y 44 son mandatos de números aleatorios que ya conocemos, y que asignan valores a las nuevas variables A, B, P, y Q. Ésta es la forma en la que puede aparecer un par de líneas negras aleatorias:



Cada vez que introduzca las nuevas líneas de mandatos, pulse RETURN, y después introduzca LIST para ver todo el programa.

```
5 COLOR 15,1,1
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
35 C=INT(RND(1)*14)+2
40 PSET (X,Y),C
42 A=INT(RND(1)*256):B=INT(RND(1)*256
)
44 P=INT(RND(1)*192):Q=INT(RND(1)*192
)
46 LINE (A,0)-(B,191),1:LINE (0,P)-(2
55,Q),1
50 GOTO 20
```

Observe que las nuevas líneas de mandatos aparecen antes del bucle GOTO de la línea 50. Esto significa que se repetirán indefinidamente, al igual que las estrellas. Cada vez que el ordenador ejecute el programa desde la línea de mandatos 20 a la 46, visualizará nuevos puntos en lugares diferentes y después nuevas líneas en lugares también diferentes. Y si un punto se encuentra situado en el lugar donde se trace una línea, el punto se volverá invisible. Ahora introduzca el mandato RUN ... y tendremos estrellas que **¡centellearán!**

Si desea ver realmente estas líneas, a fin de comprenderlas mejor, cambie el código de color, del mandato 46, de 1 a 2. De esta forma podrá ver líneas verdes apareciendo en la pantalla.

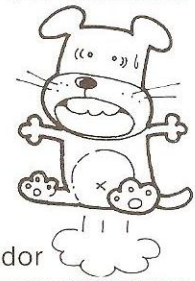
NOTA: Si paramos este programa, la pantalla seguirá siendo negra. Usted podrá volver al fondo y borde azul oscuro originales con este mandato:

```
COLOR 15,4
```

15 significa letras blancas y 4 indica azul oscuro.

Por otra parte, y aunque no se ha explicado anteriormente, cada vez que inicialice el BASIC, la pantalla se establecerá automáticamente en el modo SCREEN 0.
En este modo, el color del borde será siempre igual que el de fondo. Si usted desea colores diferentes, utilice el mandato SCREEN 1 y escriba, por ejemplo, COLOR 15, 4, 7.
Para volver a la pantalla original, introduzca el mandato SCREEN 0.

CÓMO GUARDAR NUESTROS PROGRAMAS



Cómo...

- Conectar su ordenador a un magnetófono
- Transferir un programa al magnetófono
- Comprobar si el programa se ha grabado
- Cargar un programa del magnetófono al ordenador

¿Ha visto alguna vez una fotografía de un gran ordenador como los utilizados para guiar naves espaciales, controlar fábricas, o para investigación médica? Seguramante habrá visto partes que parecen magnetófonos grandes. En realidad eso es exactamente lo que son. En máquinas grandes se denominan "impulsores de cinta" (o unidades de cinta), y graban y almacenan la información que utilizan los ordenadores.

Algo de lo que almacenan son programas, es decir, las instrucciones para la máquina, cosas como PSET, SCREEN, PRINT y otros mandatos que usted ya conoce. También almacenan **datos**, que son hechos como puntuaciones de juegos, respuestas a problemas, o fórmulas científicas.

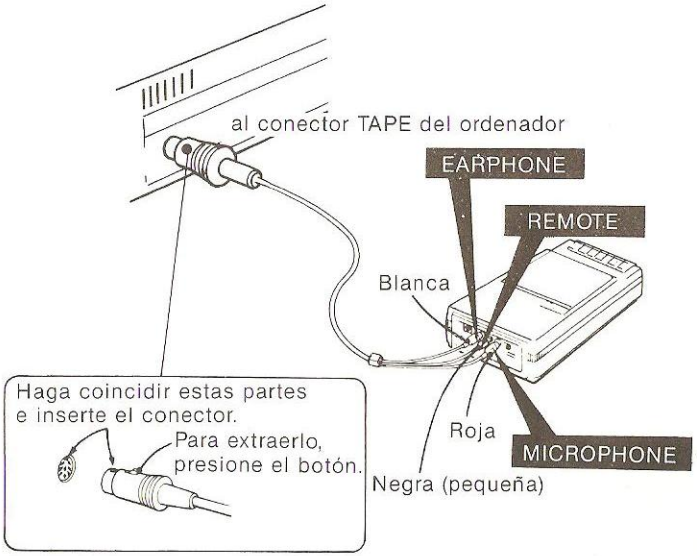
Los ordenadores se han desarrollado muy rápidamente, y lo que antes tenía que hacerse con un ordenador "central" puede realizarse ahora con máquinas pequeñas, como su ordenador Sony. Su ordenador Sony es tan pequeño que podrá sostenerlo un niño, pero es más rápido e inteligente que los antiguos ordenadores que costaban muchísimo dinero y ocupaban salas enteras.

Por lo tanto no es nada extraño que su ordenador Sony pueda también almacenar programas e información en cinta. Como es más pequeño que un ordenador central, utiliza una unidad de cinta más pequeña. Su ordenador Sony puede utilizar también un magnetófono de casetes normal, como los utilizados para escuchar música o para dictado.

CONEXIÓN DEL MAGNETÓFONO

Utilizando el cable especial suministrado con su ordenador Sony, será muy fácil conectar un magnetófono. Un extremo del cable es un cilindro de metal con patillas en el centro. Este conector deberá insertarse en la toma marcada TAPE de la parte posterior del teclado.

El otro extremo del cable tiene tres clavijas, roja, blanca, y negra. La roja deberá conectarse a la toma MICROPHONE de su magnetófono, y la blanca a la toma EARPHONE. Si su magnetófono tiene una toma REMOTE, conéctele la clavija negra. (Si su magnetófono carece de esta toma, deje simplemente esta clavija sin conectar.)



Ahora, el ordenador y el magnetófono habrán quedado listos para "hablar" entre sí.

CÓMO HACER HABLAR AL ORDENADOR

Cuando introduzca este mandato, el ordenador le dirá al magnetófono lo que hay en su memoria:

CSAVE "nombre de archivo"

C de CSAVE representa casete, y SAVE (guardar en inglés) significa simplemente: guardar el programa que esté en la memoria del ordenador, grabándolo en el casete. El "nombre de archivo" (no se olvide de las dos comillas) será el nombre que dé a su programa para que usted, el magnetófono, y el ordenador puedan distinguirlo de otros.

Un **nombre de archivo** podrá tener seis caracteres como máximo. Estos caracteres podrán ser letras, números, o signos gráficos, pero el **primer** carácter deberá ser una **letra** del alfabeto.

STAR (estrella en inglés) es un buen nombre para el programa que hemos hecho en el capítulo anterior. Tiene cuatro caracteres, comienza por una letra y, lo que es más importante: nos recordará el contenido del programa.

Vamos a guardar este programa:

```
CSAVE "STAR"
```

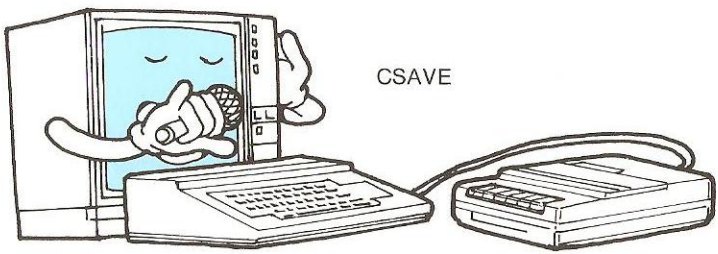
Pero no pulse todavía RETURN. Cuando pulsemos RETURN, el ordenador comenzará a enviar la información que deseemos guardar. Primeramente tendremos que asegurarnos de que el magnetófono esté listo.

Si su magnetófono tiene una toma REMOTE y la clavija negra está conectada, ponga el magnetófono en el modo de grabación.

Sin embargo, y como podrá observar, la cinta no comenzará a moverse todavía. Esto es porque, ahora, su ordenador tiene el "control" del magnetófono. Pulse RETURN, y la cinta comenzará a moverse, el programa se grabará, y la cinta se parará. Cuando haya finalizado la grabación, la pantalla mostrará:

```
CSAVE "STAR"
Ok
■
```

Si su magnetófono no tiene toma REMOTE, **presione el botón RECORD** y la cinta comenzará a moverse inmediatamente. **Espere** algunos segundos para que se estabilice la velocidad de la cinta, y pulse RETURN. Cuando en la pantalla aparezca Ok, **presione el botón STOP** del magnetófono, y habrá finalizado la operación.



CSAVE

Ahora tiene el programa en dos lugares. Todavía estará en la memoria del ordenador y, si todo ha ido bien, se habrá grabado también en la cinta.

¿SE HABRÁ GRABADO LA CINTA?

Antes de hacer ninguna otra cosa, **compruebe siempre** si el **magnetófono** ha guardado su programa. Su ordenador Sony efectuará esta comprobación comparando el contenido de la memoria con la grabación. Primeramente deberá **desconectar** la clavija de la toma REMOTE. Después, **rebobine** la cinta hasta el punto inmediatamente anterior al de comienzo de la grabación. **Ajuste el volumen** del magnetófono a aproximadamente el punto medio. Ahora **vuelva a conectar** la clavija negra a la toma REMOTE, y mecanografíe este mandato:

```
CLOAD? "STAR"
```

Si el magnetófono carece de toma "REMOTE", pulse RETURN **antes** de poner el magnetófono en el modo de reproducción. Si tiene toma "REMOTE", ponga el magnetófono en el modo de reproducción y la cinta comenzará automáticamente a moverse cuando pulse RETURN. Cuando el ordenador "oiga" el comienzo del programa, la pantalla mostrará:

```
Found:STAR
```

A medida que transcurra la reproducción, el ordenador comparará, punto por punto, el programa grabado con el contenido de su memoria y, si todo es correcto, visualizará Ok. (No se olvide de parar manualmente la cinta si el magnetófono carece de control remoto.)

Éste es un proceso muy importante (usted no deseará perder todo el trabajo realizado para escribir su programa) y algunas veces puede presentarse un problema. Si en la pantalla no aparece el mensaje "Found" (encontrado en inglés), quizás no haya rebobinado suficientemente la cinta.

Si **no** aparece Ok, aumente el volumen del magnetófono y vuelva a intentarlo. Si falla esto, puede ser que haya alguna interferencia electromagnética, o que el programa no se haya grabado adecuadamente. Compruebe las conexiones, y vuelva a empezar desde CSAVE "STAR".

Si la comprobación resulta satisfactoria y en la pantalla aparece Ok, sabrá que su programa se ha guardado. Usted puede anotar el nombre del programa en el casete, guardarlo, y volver a utilizarlo cuando lo necesite en el futuro.

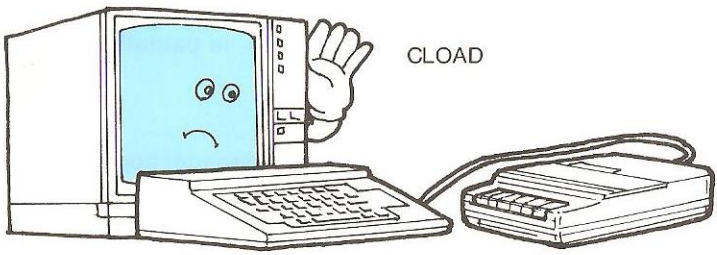
Cuando esté **seguro** de que el programa se haya grabado en la cinta, puede introducir NEW, o pulsar el botón RESET. De esta forma se borrará la **memoria del ordenador**, pero no la **memoria externa**, es decir, la grabación en la cinta.

CÓMO CARGAR NUESTROS PROGRAMAS

La posterior utilización de una cinta grabada significa "cargar el programa" en el ordenador, y ésta es una buena ocasión para practicar. El procedimento es prácticamente igual al de comprobación que acabamos de realizar. Asegúrese de que el magnetófono esté ajustado para **reproducir** en el ordenador, y que la cinta esté en el punto correcto, es decir, inmediatamente antes del comienzo del programa grabado. Después introduzca este mandato:

```
CLOAD "STAR"
```

Observe que esta vez no hay signo de interrogación (?). Naturalmente, si usted está utilizando un nombre de archivo diferente, escríbalo entre las comillas en vez de STAR.



Cuando el ordenador haya terminado de "escuchar" la cinta, la pantalla mostrará:

```
CLOAD "STAR"
Found:STAR
Ok
■
```

Asegúrese de que la cinta esté parada, y entonces podrá hacer que el ordenador ejecute (RUN) su programa.

En ocasiones, el ordenador visualizará:

```
Device I/O error
```

I/O significa **entrada/salida** (en inglés Input/Output), y en este caso se refiere a la conexión con el magnetófono. Cambie ligeramente el volumen y vuelva a intentar la carga.

DECISIONES Y JUEGOS

Cómo...

- Utilizar su ordenador para tomar decisiones
- Programar una fórmula condicional
- Jugar con su ordenador
- Hacer más sencillo su programa
- Mejorar su programa

¿Recuerda a Ron, nuestro simpático perro? Es muy bueno obedeciendo cosas sencillas como "¡Siéntate!", "¡Tráemelo!", y "¡Dame la pata!". Hasta ahora, su ordenador Sony ha estado realizando el mismo tipo de cosas sencillas que Ron obedeciendo nuestros mandatos "PRINT", "GOTO", "PSET", etc.

Después nos encontramos con Superperro, que hizo algo más difícil basado en "si la pelota es negra ..." o "si la pelota es blanca ...". Superperro pudo **tomar decisiones**. Usted ya sabe que su ordenador Sony es suficientemente inteligente como para realizar estas cosas, y a partir de ahora vamos a empezar a escribir "superprogramas" que tomen decisiones.

Comencemos por borrar nuestro último programa con el mandato NEW, y después introduzcamos:

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN GOTO 50
40 GOTO 10
50 PRINT "Acierto"
60 GOTO 10
```

Después de mecanografiar correctamente, introduzca el mandato RUN y pulse RETURN.

```
RUN
? ■
```

El ordenador nos ofreció antes un signo de interrogación, pero esta vez el cursor está en la misma línea, no en la siguiente. Esto significa que está esperando a que usted **introduzca** algo, debido al mandato INPUT (introducir en inglés) de la línea 20. En este programa, usted deberá introducir cualquier número de 1 a 5, pulsando simplemente una tecla, como ya sabe. Introduzcamos, por ejemplo, el número 3. (¿Qué sucederá si introducimos una letra o un gráfico?)

Pulse [3] y [RETURN]. ¿Cuál de las dos visualizaciones aparece en la pantalla?

```
? 3
Acierto
? ■
```

```
? 3
? ■
```

Ambas tienen un signo de interrogación al final, motivo por el que usted sabe que el ordenador está esperando a que nosotros introduzcamos otro número. Después otro, otro, otro, y otro. Éste es el juego: ¿Ha acertado el número, un "Acierto", o lo ha fallado? ¿Con qué frecuencia obtiene un "Acierto"? (La ley de probabilidades dice que usted obtendrá aproximadamente un acierto cada cinco intentos, dos aciertos cada diez, o unos 20 cada 100). Como seguramente se cansará antes que el ordenador (que **nunca** se cansa), pulse [CTRL] y [STOP].

¿Qué pasa aquí? Observemos el programa. La primera línea es nuestro viejo amigo, **el mandato de número aleatorio**. Cada vez que el programa vuelva a ejecutar el bucle (línea 60), elegirá un número entre 1 y 5, y lo asignará a la variable A.

En la línea 20 tenemos un nuevo amigo:

```
20 INPUT B
```

B es una segunda variable, y está esperando a que le asignemos un valor a través del teclado. Después de que usted introduzca un número, ambas variables adquirirán valores, y el ordenador pasará a la línea siguiente.

A es un número entre 1 y 5.

B, en nuestro primer intento, adquiere el valor 3.

ACIERTO O FALLO: EL ORDENADOR DECIDE

La línea 30 nos recuerda a Superperro, porque la primera palabra es IF.

```
30 IF A=B THEN GOTO 50
```

IF **siempre** va con THEN. Por ejemplo, "IF (si) usted tiene hambre THEN (entonces) deberá comer" o "IF usted tiene un ordenador Sony THEN podrá disfrutar con él".

IF [condición] THEN [algo]

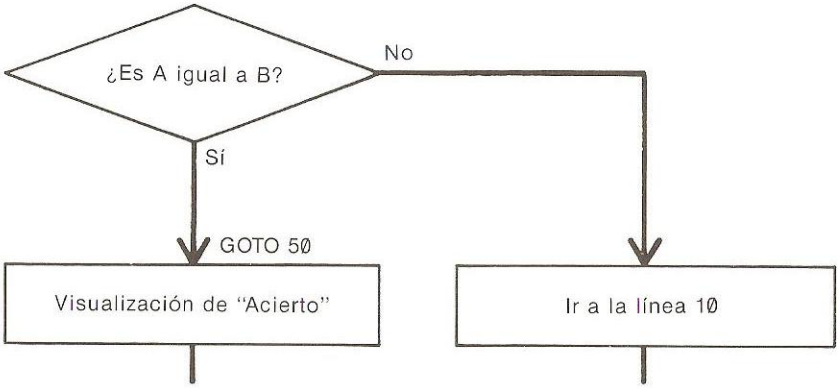
¿Cuál puede ser la **condición** de las variables A y B? Estas variables pueden ser iguales (A=B), A puede ser mayor (A>B), o B puede ser mayor (A<B).

"algo" puede ser cualquier mandato. En este caso es GOTO 50.

Esto se denomina **fórmula condicional**. En BASIC hay seis, y algunas pueden escribirse de dos formas diferentes.

Símbolo	Significado	Ejemplo	
=	Igual a	IF A=B	Si A es igual a B
>	Mayor que	IF A>B	Si A es mayor que B
<	Menor que	IF A<B	Si A es menor que B
>=	Igual o mayor que	IF A>=B	Si A es igual o mayor que B
=>		IF A=>B	
<=	Igual o menor que	IF A<=B	Si A es igual o menor que B
=<		IF A=<B	
<>	Diferente a	IF A<>B	Si A es diferente a B
><		IF A><B	

Por lo tanto, la línea 30 es una fórmula condicional. Cuando la parte de IF sea verdadera (A=B), el ordenador ejecutará el mandato (GOTO 50). Pero cuando A y B no sean iguales, pasará a la línea siguiente del programa, línea 40, sin leer el mandato THEN. La línea 40, naturalmente, hará que el ordenador vuelva al comienzo para que genere un nuevo número aleatorio que usted deberá tratar de adivinar.



Ahora usted sabe que "Acierto" solamente se visualizará cuando A y B (B será el número que usted introduzca) sean iguales. "Acierto" significa que el número que usted ha introducido para la variable B es igual al valor que el ordenador asignó a la variable A.

Después de visualizar "Acierto" el ordenador pasará a la línea 60, y el juego volverá a empezar. El juego continuará en cualquiera de los dos casos, es decir, A y B pueden ser iguales, o diferentes.

CÓMO SIMPLIFICAR EL PROGRAMA

Los programas para ordenadores deberán escribirse siempre lo más sencillamente posible, a fin de que puedan entenderse fácilmente y para que haya menos riesgo de cometer errores de mecanografía. Observemos otra vez nuestro programa acertijo.

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN GOTO 50
40 GOTO 10
50 PRINT "Acierto"
60 GOTO 10
```

Ahora sabemos que si A=B, el ordenador visualizará "Acierto". Ésta parece una sola acción, pero en el programa hay realmente dos mandatos. El primero, que viene después de THEN de la línea 30, es GOTO 50. El segundo, en la línea 50, es PRINT "Acierto". Una sola acción debería tener un solo mandato, por lo tanto cambiemos la línea 30 a

```
IF A=B THEN PRINT "Acierto"
```

Y ahora ya no necesitamos los mandatos de las líneas 50 y 60, ¿no?

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN PRINT "Acierto"
40 GOTO 10
```

¡Muy sencillo! Si A y B son iguales, después de imprimir Acierto", el ordenador pasará a la línea 40 y luego GOTO (irá a) 10. Si A y B no son iguales, hará lo mismo, pero sin imprimir "Acierto". Nuestro juego trabajará ahora con sólo cuatro mandatos.

Ahora vamos a hacer algunos cambios en la pantalla. Primeramente cambiemos el mandato de la línea 30 de GOTO 50 a PRINT "Acierto". Después, borremos las líneas 50 y 60, de la forma siguiente: mueva el cursor hasta debajo de la línea 60 e introduzca 50. Pulse RETURN, y la línea 50 se borrará de la memoria del ordenador. Haga lo mismo con 60 para borrar la línea 60.

¿Han desaparecido las líneas 50 y 60 de la pantalla? No, ¡no han desaparedido! Sin embargo se habrán borrado de la memoria. Si usted introduce el mandato LIST, el ordenador visualizará lo que quede ahora en su memoria:

```
LIST
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN PRINT "Acierto"
40 GOTO 10
```

Esto es mejor, ¿no le parece?

Algunos perfeccionamientos

Hemos simplificado todo lo posible nuestro programa acertijo, y ahora podemos pensar en perfeccionarlo añadiéndole cosas que sean convenientes y divertidas. Naturalmente, cuando le añadamos algo tendremos que mantener cada mejora lo más sencilla que podamos.

Cuando A sea igual a B, el ordenador nos lo dirá visualizando "Acierto". pero cuando no sean iguales, no visualizará mensaje alguno antes de comenzar de nuevo. El programa será mucho más fácil para sus amigos si cambia la línea 30 de la forma siguiente:

```
30 IF A=B THEN PRINT "Acierto" ELSE PRINT "Fallo
```

ELSE significa "si no". Ésta es una palabra muy útil para poner después de una fórmula condicional porque aclara tal paso del programa.

IF [fórmula condicional] THEN [mandato 1] ELSE [mandato 2]

Como este nuevo mandato tiene más de 37 caracteres, "pasará" a la línea siguiente, haciendo que la línea 40 desaparezca temporalmente. Pero no se preocupe, todavía está en la memoria del ordenador: utilice simplemente el mandato LIST para que vuelva a la pantalla. Todavía hay una mejora que hará el programa más fácil de entender para sus amigos cuando lo vean. Cambiemos la línea 20 a:

```
20 INPUT "Adivine (1-5) ";B
```

Usted entenderá el significado de este nuevo mandato si lo introduce en la pantalla y utiliza después el mandato RUN.

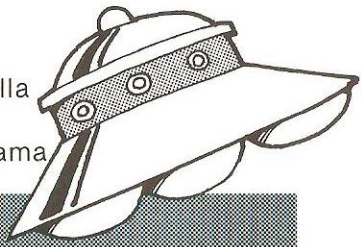
```
RUN
Adivine (1-5) ? ■
```

Ahora es muy fácil de entender ¿no? Es mucho más comprensible que el mensaje "?" que estábamos viendo hasta hace unos minutos. Ahora podrá jugar con sus amigos, y todos entenderán rápidamente el programa.

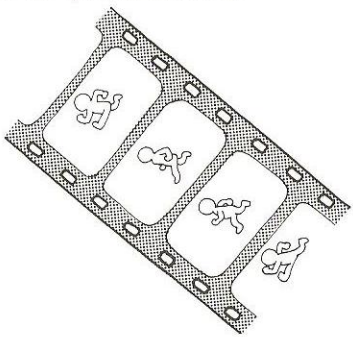
GRÁFICOS MÓVILES

Cómo...

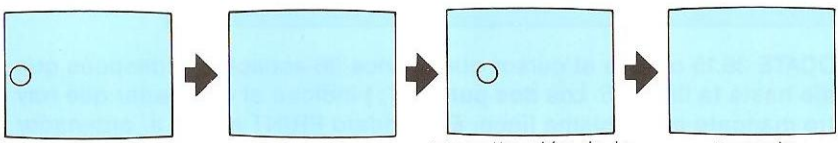
- Mover caracteres alrededor de la pantalla
- Hacer una película sobre OVNIS
- Utilizar variables para acortar un programa

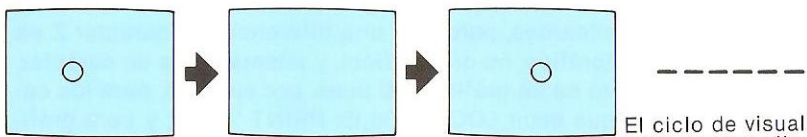


Todo el mundo sabe que las imágenes de las películas en realidad no se mueven. Una película es simplemente una serie rápida de imágenes, una tras otra, fijas. Como las imágenes son ligeramente diferentes entre sí, percibimos la ilusión de que se mueven.



La televisión es igual. Las imágenes de la pantalla de su TV cambian muchas veces por segundo, y como resultado nos parece que se mueven. La pantalla de su ordenador Sony es prácticamente igual que la de un TV, por lo que, naturalmente, podremos utilizarla para conseguir imágenes móviles. Ésta es la forma de hacerlo:





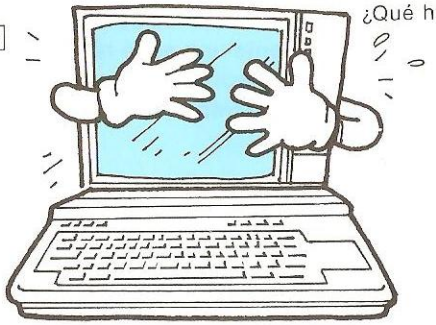
El ciclo de visualización/ borrado se repite continuamente.

Para mover la marca O através de la pantalla, la mostramos cerca de la izquierda, después la borramos, luego la mostramos un poco más allá, la borramos ... etc. Para hacer un paso de toda esta serie, tendremos que decirle al ordenador que borre, después que localice la posición correcta, y después que imprima (visualice) algo.

```
CLS
LOCATE 36,15:PRINT "Z"
```

Aquí tenemos dos nuevos mandatos, y después PRINT. CLS significa "borrar la pantalla" (en inglés CLear Screen). Si solamente se introduce este mandato, desaparecerá todo lo que haya en la pantalla.

CLS+ RETURN



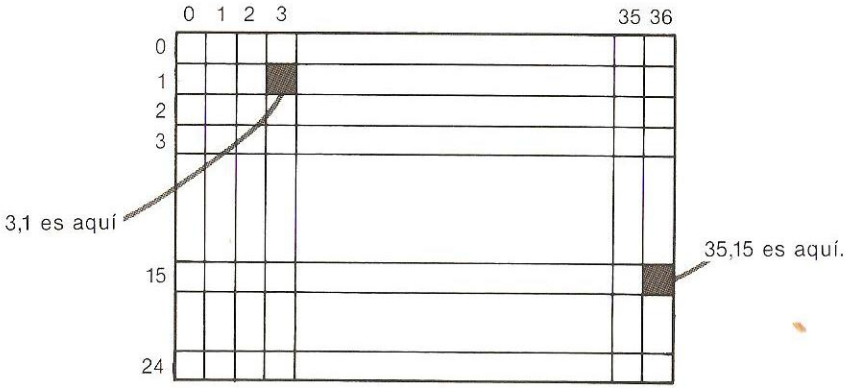
Para ver lo que significa el mandato LOCATE, introduzcamos juntos los tres mandatos anteriores.

```
LOCATE 36,15:PRINT "Z"


                                    Z
Ok
■
```

LOCATE 36,15 ordena al cursor que avance 36 espacios, y después que baje hasta la línea 15. Los **dos puntos (:)** indican al ordenador que hay otro mandato en la misma línea. El mandato PRINT indica al ordenador que visualice Z en el lugar donde esté el cursor.

Esto es muy parecido al mandato PSET que hemos utilizado antes para hacer estrellas centelleantes, pero con una diferencia: el carácter Z es mayor que un punto (gráfico, no ortográfico), y además Z es un carácter, mientras que un punto es un gráfico. Así pues, por ejemplo, para los caracteres tendremos que decir LOCATE 36,15: PRINT "algo"; y para gráficos diremos PSET (36,15): el resultado es el mismo. En ambos casos, podemos pensar que 36,15 es una **dirección**.



El mandato PRINT, que va después del mandato LOCATE y los dos puntos (:), no está limitado a un solo carácter como "Z". Con este mandato LOCATE podrá utilizarse **cualquier cantidad** de caracteres o palabras. LOCATE simplemente indica al cursor dónde tiene que ir antes de comenzar PRINT. Usted puede utilizar LOCATE para iniciar un párrafo en mitad de una página, para comenzar un juego en la parte inferior de la pantalla, o para colocar algo donde desee.

Hagamos trabajar un poco la imaginación. ¿Ha visto alguna vez un OVNI? (Los OVNIS, como usted sabe, son "objetos voladores no identificados", naves espaciales procedentes de otros planetas, y probablemente utilizan ordenadores.) Nosotros podemos componer un OVNI con solamente cinco caracteres:

```
LOCATE 13,10:PRINT "_=O=_"
```

```
_=O=_
Ok                    13,10 es aquí.
■
                                    Z
LOCATE 13,10:PRINT "_=O=_"
```

¿No le parece un OVNI? También puede dibujar otras formas, utilizando muchos caracteres y signos del teclado. Este OVNI se hace con el signo de subrayado (_), el signo igual a (=), y la letra O.

Ahora hagamos algo más:

```
LOCATE 13,10:PRINT "     "
```

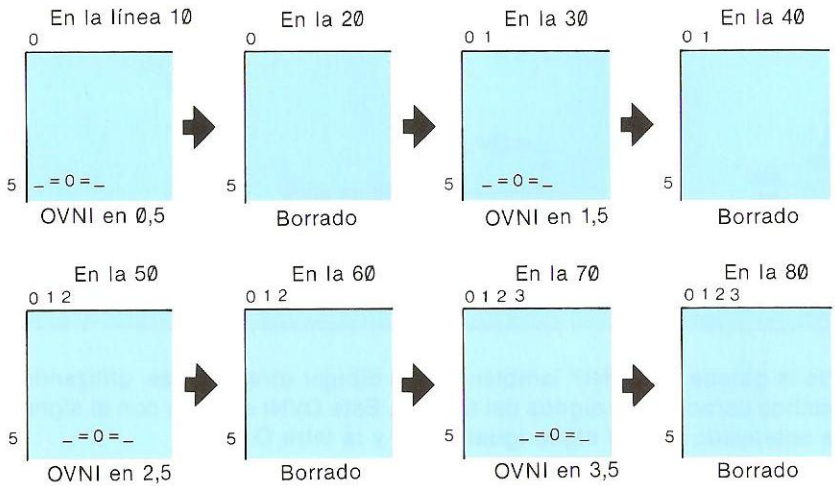
Pulse RETURN, y el OVNI desaparecerá. Los cinco espacios reemplazaron sus cinco caracteres. El "espacio" es una forma muy útil de borrar letras o palabras cuando no se desee utilizar CLS para borrar toda la pantalla.

VISUALIZACIÓN, BORRADO, VISUALIZACIÓN, BORRADO...

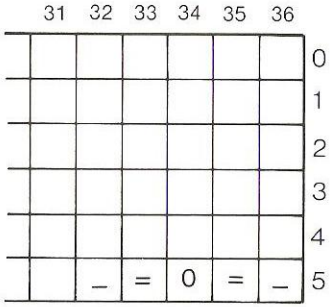
Los mandatos LOCATE y CLS, por lo tanto, suponen la forma de hacer imágenes móviles. Aquí tenemos un programa para hacer que nuestro OVNI se mueva. (No lo escriba todavía. Antes de ello, vamos a ver cómo trabaja.)

```
5 CLS
10 LOCATE 0,5:PRINT "_=0=_"
20 LOCATE 0,5:PRINT "     "
30 LOCATE 1,5:PRINT "_=0=_"
40 LOCATE 1,5:PRINT "     "
50 LOCATE 2,5:PRINT "_=0=_"
60 LOCATE 2,5:PRINT "     "
70 LOCATE 3,5:PRINT "_=0=_"
80 LOCATE 3,5:PRINT "     "
```

Esto es lo que tales mandatos harán con la visualización de la pantalla:



La primera línea, CLS, borrará la pantalla, y después cada par de líneas, 10 y 20, 30 y 40, 50 y 60, 70 y 80, mandará el OVNI a una dirección diferente. Si continuamos hacia el borde derecho, de la forma siguiente:



el último mandato será

```
650 LOCATE 32,5:PRINT "_=0=_"
660 LOCATE 32,5:PRINT "     "
```

El programa completo tiene 67 líneas. ¡Demasiado! ¡Es más interesante ver la televisión!

Pero no se rinda. Hay una forma mucho más fácil. Como habrá observado, los mandatos después de la línea 5 se parecen mucho y, aquí está la clave. En realidad están en pares prácticamente idénticos.

```
10 LOCATE 0,5:PRINT "_=0=_"
20 LOCATE 0,5:PRINT "     "
```

Esta parte no cambia nunca.

Esto es siempre 5.

Solamente cambia esta posición: 0,0,1,1,2,2,.....32,32

Solamente cambia el primer número de dirección, y éste varía sistemáticamente. Esto significa (¿lo ha adivinado?) que puede cambiarse por una **variable**. Ahora podemos ver lo útiles que son las variables: pueden acortar el programa hasta dejarlo en unas pocas líneas.

```
100 CLS
110 I=0
120 LOCATE I,5:PRINT "_=0=_"
130 LOCATE I,5:PRINT "     "
140 I=I+1
150 IF I<33 THEN GOTO 120
160 END
```

Borre la memoria del ordenador con el mandato NEW, y después introduzca este mandato. (Esta vez hemos empezado los números de las líneas con 100 para poder añadir posteriormente más líneas.) Observe ahora el OVNI: ¡vuela!

Examinemos el programa línea por línea y veamos **por qué** vuela. La línea 100 borrará la pantalla (no deseamos que nuestro OVNI choque con algo). La línea 110 definirá las variables y les asignará un valor. Como "I" significa posición izquierda-derecha, y deseamos comenzar a la izquierda, a "I" se le asigna el **valor inicial** de cero. Las líneas 120 y 130 son los pares de visualización-borrado que hemos visto antes, pero ahora utilizan la variable "I".

La línea 140 cambiará la variable: es decir I=I+1. Recuerde que esto no es aritmética, es BASIC. Por lo tanto, "=" no significa "igual a", sino "adquiere el valor de". "I" adquiere el valor de "I+1" lo que hace que "I" sea una unidad mayor que antes de ir a la línea 150.

La línea 150 significa que si "I" es menor que (<) 33, entonces el programa volverá a la línea 120. Ahora "I" será una unidad mayor, y el OVNI se visualizará un espacio más a la izquierda. Cuando "I" ya no sea menor que 33, el programa ya no tendrá que "GOTO 120" (ir a 160), pero se moverá hasta la línea 160. En realidad, incluso sin END, el programa se parará porque no hay mandato después de la línea 150.

Pero este programa contiene un **bucle** donde el ordenador pasa una y otra vez de la línea 120 a la 150. Este bucle se controla mediante la condición de "I", es decir, se repetirá mientras sea menor que 33. Por lo tanto este bucle se denomina **bucle condicional**.

Una forma todavía mejor
Observemos de nuevo el programa.

```
100 CLS
110 I=0
120 LOCATE I,5:PRINT "_=0=_"
130 LOCATE I,5:PRINT "     "
140 I=I+1
150 IF I<33 THEN GOTO 120
160 END
```

El valor inicial es cero.

El valor aumenta una unidad cada bucle, y éste se para en 33.

Las partes más importantes de este programa son las líneas 110, 140, y 150. Estas líneas indican al programa dónde tiene que empezar (en 0), y que tiene que repetir el bucle 33 veces. Este tipo de repetición se utiliza a menudo en muchos programas, y el BASIC tiene una forma especial de hacerlo, con dos mandatos denominados **FOR** y **NEXT**.

```
100 CLS
110 FOR I=0 TO 32 <-------- Repetición de 0 a 32
120 LOCATE I,5:PRINT "_=O=_"
130 LOCATE I,5:PRINT "      "
140 NEXT I <-------- Incremento en 1 del valor de I,
                     y retorno a la línea 120
150 END
```

Estos dos mandatos, FOR de la línea 110 y NEXT de la línea 140, son los "especialistas en bucles". Utilizan solamente dos líneas donde antes necesitábamos tres.

Es muy importante recordar esta forma:

FOR **variable** = **valor inicial** TO **valor final**

y

NEXT **variable**

Para nuestro programa, es

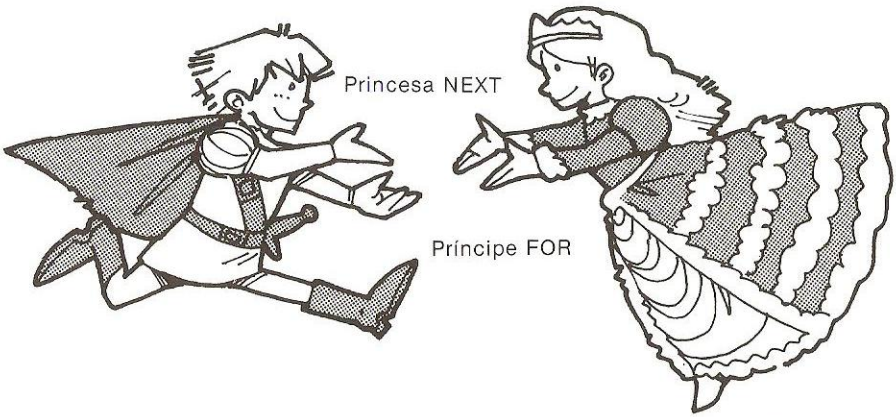
```
110 FOR I=0 TO 32
```

y

```
NEXT I
```

FOR indica al programa cuántos bucles tiene que hacer. NEXT es un mandato muy "inteligente", que cuenta los pasos y envía el programa a la línea siguiente cuando la variable alcanza el valor final.

FOR y NEXT van **siempre** emparejados.



Si el Príncipe FOR no puede encontrar a la Princesa NEXT, el ordenador se sentirá tan infeliz que visualizará un mensaje de error y se detendrá hasta que usted lo corrija.

CÓMO JUNTAR TODO

Cómo...

- Añadir color y sonido a sus programas
- Unir dos programas
- Confeccionar un diagrama de flujo
- Mejorar sus programas

CÓMO AÑADIR COLOR Y SONIDO

Ahora estamos listos para añadir algunas de las funciones que hemos aprendido a fin de conseguir un OVNI en color, sonoro, y móvil. Introduzca este programa en su ordenador:

```
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=0=_"
130 LOCATE I,Y:PRINT "      "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 100
```

Repetición debida a FOR—NEXT

Como podrá ver, esto se repetirá indefinidamente debido a que todo el programa es un bucle repetitivo, pero pulsemos CTRL y STOP para ver cómo trabajan los mandatos.

La línea 100 hace dos cosas: establece el color (fondo negro y caracteres blancos), y borra la pantalla. 103 introduce una nueva variable, Y, y le asigna un **valor aleatorio** entre 0 y 21. 106 es similar: asigna un valor aleatorio, entre 2 y 15, a una nueva variable, C, y le asigna el número de código del color. (No utilizamos el código 1 porque los caracteres serían invisibles.)

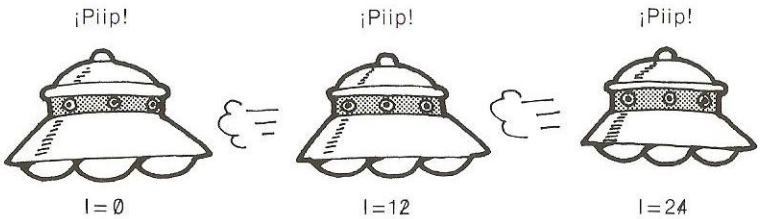
La secuencia FOR-NEXT comienza en la línea 110, y esta vez la variable Y es parte de la "dirección" para cada mandato LOCATE. La localización horizontal (izquierda-derecha) es I, que cambia **regularmente** de 0 a 32, pero, en la línea 103, la localización vertical (arriba-abajo) es **aleatoria** (RND): Y puede adquirir cualquier valor entre 0 y 21. Ésta es la razón por la que aparecen OVNIS de diferentes colores en distintas altitudes.

Ahora observemos las líneas 133 y 136:

```
A=I MOD 12
IF A=0 THEN BEEP
```

Hay dos nuevas palabras. BEEP, que significa pitido (en inglés beep), es decir, el sonido que emite el OVNI. Este pitido sonará solamente cuando A sea cero, y A tiene el valor de I MOD 12, pero ¿qué significa MOD? MOD es la abreviatura de **módulo** (en latín modulus), que es el resto de la división de un número por otro. I MOD 12 es el resto de la división de I por 10. Si I es 15, I MOD 12 será 3. Pero si I es 20, I MOD 12 será 0.

Como I aumenta de 0 a 32, I MOD 12 será 0 tres veces para cada OVNI: en 0, 12 y 24. Esto significa que cada OVNI emitirá tres pitidos en su viaje a través de la pantalla.



JUEGO ACERTIJO+SONIDO+VIDEO

Una de las cosas más interesantes de la programación de ordenadores es unir dos programas para formar uno. La combinación de dos o más programas, como pronto verá, puede ser muy útil y entretenida.

Aquí tenemos de nuevo los programas anteriores.

Programa acertijo

```
10 A=INT(RND(1)*5)+1
20 INPUT "Adivine";B
30 IF A=B THEN PRINT "Acierto" ELSE P
RINT "Fallo"
40 GOTO 10
```

Programa de OVNIS

```
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=O=_"
130 LOCATE I,Y:PRINT "      "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 100
```

Si ponemos simplemente los dos programas en el ordenador de la forma en que están, veremos que no se unen. El programa se ejecutará hasta la línea 40, volverá a la línea 10, y los OVNIS no aparecerán nunca. Por ello, el primer cambio que tendremos que hacer será conectar los OVNIS al primer programa, y el mejor lugar para hecerlo es la línea 30 (A=B). Escribamos una nueva línea 30:

```
30 IF A=B THEN GOTO 100 ELSE PRINT "Fallo"
```

Ahora nos encontramos con otro problema: cómo parar los OVNIS después de que alguien acierte el número. No deseamos que aparezcan indefinidamente OVNIS, ni tampoco parar el programa, lo que nos interesa es volver de nuevo al acertijo. Por lo tanto tendremos que cambiar la línea 150 a:

```
150 GOTO 10
```

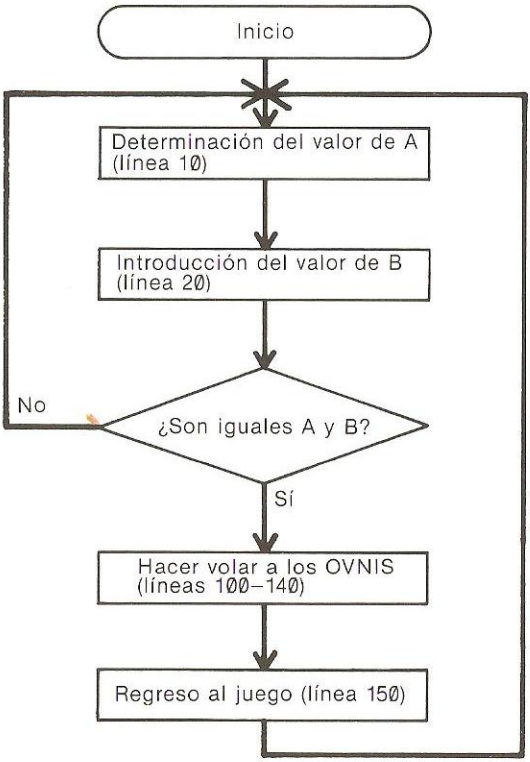
Ahora el programa se verá en esta manera:

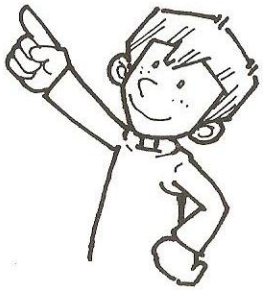
```
10 A=INT(RND(1)*5)+1
20 INPUT "Adivine";B
30 IF A=B THEN GOTO 100 ELSE PRINT "F
allo"                    <-- En caso de Acierto, paso a OVNIS
40 GOTO 10
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=0=_"
130 LOCATE I,Y:PRINT "      "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 10 <-- Retorno al acertijo
```

El diagrama de flujo: una forma de planear programas

Nuestros programas se están volviendo cada vez más largos y complicados, con bucles y múltiples funciones. Cuando los profesionales hacen programas complicados, los planean cuidadosamente antes de comenzar a escribir los mandatos, y la mejor forma de hacer esto es con un **diagrama de flujo**.

En este libro no se utilizan diagramas de flujo, y usted no necesitará entenderlos para utilizar su ordenador Sony. Si lo desea, puede saltar a la sección siguiente. Sin embargo, para ver simplemente lo que es un diagrama de flujo, aquí tenemos uno para nuestro programa acertijo/OVNIS.





El recuadro superior es donde comienza el programa, y cada uno de los demás muestra la operación que necesita un mandato (o más de uno). La línea muestra el flujo de la acción, o la lógica del programa. Como puede observar, el recuadro en forma de rombo tiene dos líneas diferentes que salen de él; esto indica que el programa comprueba una condición (IF) para ver a dónde tiene que ir. Observe también que la línea "No" y la línea inferior forman bucles para repetir el juego.

Mejorando el programa acertijo

La diferencia entre un programa aceptable (ejecutable) y otro **excelente** son las mejoras introducidas después de haberlo escrito. Aquí tenemos tres cosas que podríamos haber hecho mejor en el programa acertijo/OVNIS.

- Después de cada OVNI, el número siguiente es del mismo color que el OVNI. ¿No sería más fácil leer los números si todos estuviesen escritos en blanco?

● El juego acertijo mueve hacia arriba y hacia abajo la pantalla, empezando siempre en la línea situada debajo del último OVNI. ¿No podríamos hacer que se mantuviese fija?

● Cuando vuela el primer OVNI, el color de fondo cambia de azul oscuro a negro y permanece negro. ¿No sería mejor tener siempre un solo color de fondo?

Estos problemas podremos resolverlos con algunas adiciones y cambios sencillos.

```
5 COLOR 15,1                    <-- Adición
7 CLS:LOCATE 0,0                <-- Adición
10 A=INT(RND(1)*5)+1
20 INPUT "Adivine";B
30 IF A=B THEN GOTO 100 ELSE PRINT "F
allo"
40 GOTO 10
100 CLS                         <-- Cambio
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=0=_"
130 LOCATE I,Y:PRINT "      "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 5                      <-- Cambio
```

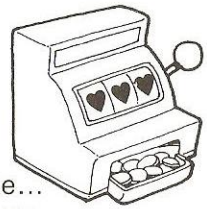
Antes de proseguir, vea si ha entendido estos cambios ...

¿Los ha entendido? Las nuevas líneas 5 y 7 resuelven bastante bien todos los problemas. La línea 5 establece inicialmente un fondo negro y letras blancas. La línea 7 mantiene el juego en la parte superior de la pantalla.

Pero la cosa más importante que tiene que recordar sobre el cambio de un programa es ésta: **un cambio en un lugar puede suponer a veces cambios en otros lugares**. En este caso, el mandato COLOR de la línea 5 viene de la línea 100, por lo tanto habrá que cambiar esta línea 100 a, simplemente, CLS. Y, como el programa comienza ahora en la línea 5 en vez de la línea 10, habrá que cambiar la línea 150 para que puedan ejecutarse las dos nuevas líneas.

Ahora tenemos un buen programa. ¿Se encuentra plenamente satisfecho con él? ¿Le gustaría hacerlo más interesante? ... ¿Por qué los OVNIS vuelan siempre en línea recta? ... Su ordenador Sony puede hacer muchas, muchas cosas, siempre y cuando usted se lo mande.

JUEGO DE MÁQUINA TRAGAPERRAS



Cómo...

- Programar un juego de máquina tragaperras que...
  - —Cambie la visualización hasta que usted la pare
  - —Compare las visualizaciones de las ventanillas
  - —Mantenga la puntuación de acuerdo con sus apuestas
- Utilizar una variable de matriz
- Utilizar el mandato ON-GOTO
- Utilizar una variable alfanumérica
- Parar un programa pulsando una tecla
- Establecer un formato para impresión
- Utilizar más de una condición en el mandato IF—THEN
- Listar por secciones un programa largo

Cuando usted acertó el número en nuestro último programa, el premio fue on OVNI volador. Ahora su ordenador Sony se convertirá en una "máquina tragaperras" estilo Las Vegas. Cada vez que juegue, ganará o perderá. El ordenador no puede manejar dinero como las máquinas tragaperras de los casinos de Las Vegas, pero mantiene el resultado con puntos.

He aquí las reglas: primeramente, el ordenador le preguntará cuántos puntos desea apostar. Después cambiará rápidamente los símbolos visualizados en sus tres "ventanillas", hasta que usted lo pare. A continuación le dirá su nueva puntuación: si los tres símbolos son iguales, usted ganará el triple del número que haya apostado. Si sólo hay dos iguales, ganará el número que haya apostado. Pero si los tres son diferentes, perderá el doble de su apuesta. Usted comenzará con un **haber** de 100 puntos, y si llega a 300, ganará el juego. Pero si su puntuación disminuye hasta cero, perderá y finalizará el juego.

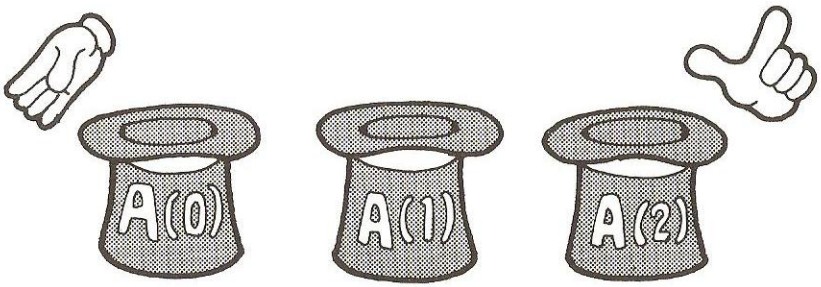
¿Le parece bien? Pues, ¡programémoslo!

La parte principal de este juego son las tres ventanillas. En cada una aparecerán y cambiarán continuamente cuatro símbolos (♥, ♠, ♦, y ♣) tan rápidamente que usted no podrá verlos, por lo que solamente podrá tratar de adivinarlos. Naturalmente, nuestro programa utilizará variables para cambiar los símbolos de cada ventanilla.

¿QUÉ ES UNA VARIABLE DE MATRIZ

Como ya hemos programado variables más de una vez, sabemos cómo una letra o un nombre puede adquirir un valor cambiante. Hemos utilizado nombres como A, B y Q, y les hemos asignado valores que variaban con el número de veces que se repetía un bucle, con la posición de un OVNI, o aleatoriamente.

Esta vez necesitamos una variable que pueda convertirse en cualquiera de nuestros cuatro símbolos. Cada una de las tres ventanillas muestra el mismo juego de símbolos, lo que significa que podemos utilizar la misma variable para todas ellas. Pero cada ventanilla funciona por separado: algunas veces coincidirán, pero otras no. Para esta situación, utilizaremos nuestra variable en tres lugares diferentes, tres de la misma variable. Las tres variables se denominan A, pero tienen además un número (entre paréntesis), por lo que sabremos qué variable es para cada ventanilla.



A(0), A(1), y A(2) son **variables de matriz.** ¿Le parece confuso? ¿Piensa que sería más fácil utilizar tres nombres diferentes? Pronto verá que las variables de matriz simplifican mucho la programación, porque son muy flexibles.

Nuestro conjunto de variables tiene una **anchura** de tres, y será tan **largo** como el número de valores diferentes que le demos. Uste puede pensar que estas variables son **multidimensionales**, y esto le ayudará a recordar el mandato del BASIC para formar una variable de matriz: **DIM**. En este programa utilizaremos la línea de mandato

```
10 DIM A(2)
```

para hacer tres variables de matriz A(0), A(1), y A(2). (Otros ejemplos de mandatos de variables de matriz son: **DIM A(10)**, que hace 11 variables de A(0) a A(10); y **DIM A(2), B(2)**, que hace seis variables, A(2) y B(0) a B(2).)

Hay una cosa más que aprender acerca de las variables de matriz: el número entre paréntesis puede convertirse también en una variable, como A(I), haciendo que estas variables sean todavía más fáciles de programar.

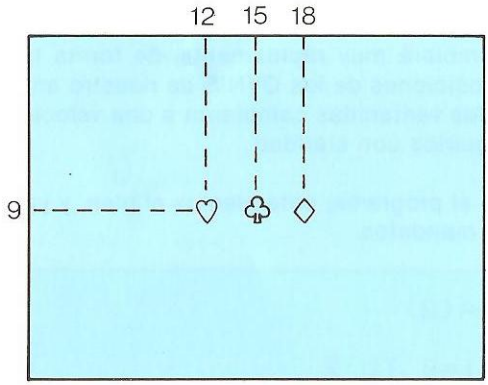
CÓMO HACER UNA MÁQUINA TRAGAPERRAS

Nosotros deseamos que nuestras variables de matriz representen símbolos diferentes, corazones (♥), picas (♠), diamantes (♦), y tréboles (♣). Lo primero que necesitamos son los números de código de los símbolos. Como usted recordará, utilizamos números de código para representar colores, y su ordenador Sony ya sabe los códigos de los diferentes colores. Esta vez, deberemos decidir nuestros propios códigos para los símbolos.

♥—1 ♠—2
♦—3 ♣—4

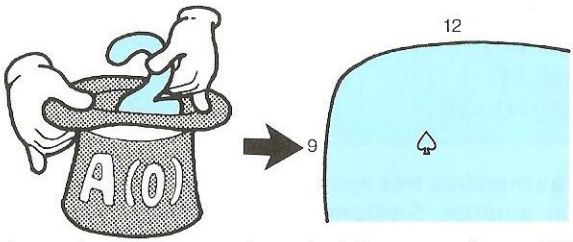
Ahora sabemos que cuando la variable A(1) tome el valor de 3, por ejemplo, en tal ventanilla habrá un diamante.

A continuación deberemos decidir dónde estarán situadas las ventanillas en la pantalla. Pongámoslas en el centro.

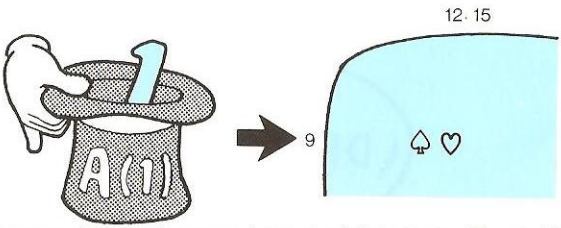


Hemos situado la ventanilla A(0) en 12,9. A(1) está en 15,9. Y A(2) se encuentra en 18,9.

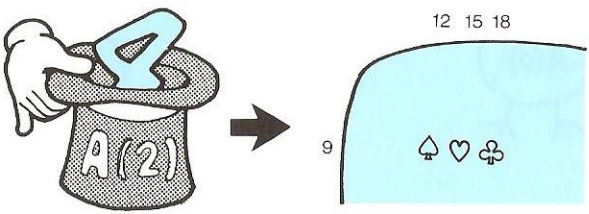
De esta forma habremos terminado de planear el programa de la "máquina tragaperras". Revisemos ahora el sistema que hemos diseñado.



Cuando A (Ø) tome el valor 2, como 2 es el código para ♠, en 12,9 aparecerá ♠.



Cuando A (1) tome el valor 1, como 1 es el código para ♥, en 15,9 aparecerá ♥.



Cuando A (2) tome el valor 4, como 4 es el código para ♣, en 18,9 aparecerá ♣.

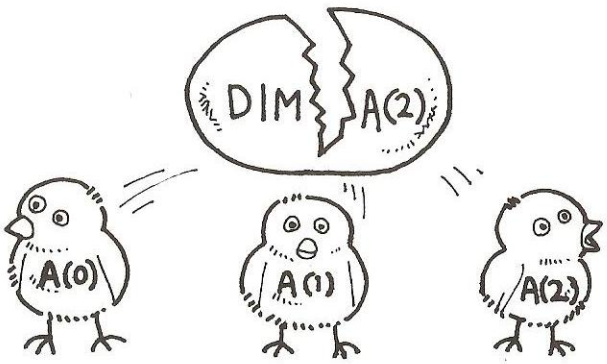
Este sistema hará que los símbolos aparezcan en las ventanillas. Y como cada variable cambiará muy rápidamente, de forma tan rápida como cambiaban las posiciones de los OVNIS de nuestro anterior programa, los símbolos de las ventanillas cambiarán a una velocidad tal que será imposible distinguirlos con claridad.

Hemos planeado el programa, entendemos el plan, y ya estamos listos para pasar a los mandatos.

```
10 DIM A(2)
20 CLS
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 110
80 PRINT "♠":GOTO 110
90 PRINT "♦":GOTO 110
100 PRINT "♣"
110 NEXT I
120 GOTO 30
```

La línea 10 crea nuestras tres variables de matriz, como henos explicado en la sección anterior. Siempre que utilicemos variables de matriz, tendremos que decírselo al ordenador al **comienzo** del programa.

```
10 DIM A(2)
```



La línea 20, naturalmente, borra la pantalla. La línea 30 comienza por FOR, y ya sabemos que cuando haya un mandato FOR tiene que haber otro NEXT en cualquier parte del programa. La sección FOR-NEXT va de la línea 30 a la 110, y los mandatos de esta sección se repetirán en bucle. Después de FOR, la línea 30 introduce una nueva variable, I, que utilizaremos entre los paréntesis de nuestras variables de matriz.

¿No le suena familiar la línea 40? Es el mismo tipo de mandato de **número aleatorio** que el que utilizamos en el juego acertijo. Cuando I tome el **valor inicial** de 0, la línea 40 será

```
A(0)=INT(RND(1) * 4)+1
```

Esto significa que A(0) podrá ser cualquier número entre 1 y 4, ¿no? A(0) es la ventanilla de la izquierda, y los números 1-2-3-4 son los códigos para los símbolos ♥, ♠, ♦, y ♣. En otras palabras, cuando I adquiera el valor de 0, la línea 40 dirá al ordenador que elija uno de los tres símbolos para la ventanilla izquierda.

A continuación tendremos que decirle al ordenador que visualice las tres ventanillas en la pantalla.

```
50 LOCATE I * 3+12,9
```

Cuando I adquiera valor 0 para la ventanilla izquierda, el cursor estará en 12,9. Cuando I tenga el valor de 1 para la ventanilla central, la ubicación será 15,9. Para la ventanilla derecha, I será 2, y el lugar será $2 \times 3 \times 12,9$, o 18,9.

Ahora el ordenador ha elegido un símbolo, y el cursor se ha movido hasta la ventanilla. Pero el ordenador no visualizará el símbolo hasta que nosotros se lo ordenemos. Ésta es la razón de la importancia de la línea 60.

```
60 ON A(I) GOTO 70,80,90,100
```

Aquí podemos ver el mandato GOTO para cuatro líneas diferentes al mismo tiempo, 70, 80, 90, y 100. ¿Cómo sabrá el programa a qué línea tiene que ir? La parte ON A(I) de este mandato toma la decisión. Ésta es la forma que siempre se utiliza para un mandato ON-GOTO:

```
ON [algo] GOTO [línea 1], [línea 2], [línea 3] ...
```

Esto significa, por ejemplo, que si "algo" es 1, el programa irá a la línea 1; si es 2, irá a la 2; si es 3, irá a la 3, etc., En otras palabras, el valor de "algo" decide automáticamente a dónde tiene que ir el programa.

ON—GOTO es un controlador de tráfico.

Como ejemplo, supongamos que A(0) tiene valor 3 (el código para ♦). La línea 60 ordena al programa que vaya a la línea 90 cuando la variable adquiera este valor.

```
90 PRINT "♦":GOTO 110
```

¿Qué sucedería si 2 fuese el valor para la variable A(0)? En este caso, 60 enviaría el programa a

```
80 PRINT "♠":GOTO 110
```

Ahora podemos entender cómo el programa elige uno de los cuatro símbolos y lo visualiza en la pantalla.

Después de cada uno de los mandatos PRINT, el programa pasa a la línea 110. (En la línea 100 no hay mandato GOTO 110 porque el programa pasará automáticamente a la línea siguiente.)

La línea 110 ordena al programa que cambie el valor de I de 0 a 1. Después volverá a la línea 40. ¿Sabe lo que sucederá a continuación? Trate de entenderlo antes de que lo expliquemos.

Muy bien. Cuando I es 1, la línea 50 pasa a ser

```
LOCATE 1 * 3+12,9
```

lo que significa localizar (LOCATE) 15,9. En otras palabras, la posición donde se visualizará el símbolo se ha movido de 12,9 a 15,9, es decir, de la ventanilla izquierda a la central. En 15,9 de la pantalla aparecerá uno de los símbolos. Después, el programa cambiará I a 2, y volverá a la línea 40. Y esta vez, la posición será 18,9, es decir, la ventanilla derecha. El ordenador habrá ejecutado el programa tres veces (¡en menos de un segundo!).

Esto es lo que usted debería ver en la pantalla, pero nunca podremos ver los símbolos que el ordenador elige aleatoriamente.

Las líneas 30 a 120 son un bucle. Su ordenador dará vueltas y más vueltas, utilizando variables de matriz para elegir y visualizar símbolos en las ventanillas izquierda, central, derecha, izquierda, central, derecha ... Como el programa es un bucle sin fin, parecerá que los símbolos "giran" constantemente en la pantalla, al igual que lo hacen en una máquina tragaperras real.

Si no ha probado nunca el programa, introdúzcalo ahora en su ordenador. Compruebe que no contenga equivocaciones. Cuando introduzca el mandato RUN, verá que hemos tenido éxito en la utilización de variables de matriz. (Sin embargo, en el modo SCREEN 0, no se visualizará el borde derecho de los símbolos.)

Pero, ¿cómo podremos parar este bucle infinito para que nos diga la puntuación? Para poner en juego nuestra máquina tragaperras, tendremos que poner algunos mandatos más en el programa. Para hacer esto utilizaremos algunas variables, que adquieren caracteres en vez de valores numéricos, denominadas **variables alfanuméricas**. Estas variables requieren una explicación especial.

VARIABLES ALFANUMÉRICAS

Hemos aprendido que una variable puede adquirir el valor de cualquier **número**, como 1, 2, 3, 191, o 252. Las variables **también** pueden adquirir el "valor" de cualquier **carácter**, o juego de caracteres que formen una **cadena** (es decir, una serie de caracteres alfanuméricos).

Hagamos algunos ejercicios sencillos para ver cómo se utiliza una variable para representar un carácter o una cadena. Primeramente escriba **PRINT A$** y después pulse RETURN. ($ está en la tecla del 4, por lo que habrá que utilizar SHIFT.)

```
PRINT A$

Ok
■
```

No ha sucedido nada. No ha sonado pitido alguno ni ha aparecido ningún mensaje de error para indicar una equivocación, y el ordenador no ha impreso nada. Introduzca ahora este mandato:

```
A$="ABC"
PRINT A$
```

Pulse RETURN, y en su pantalla aparecerá lo siguiente:

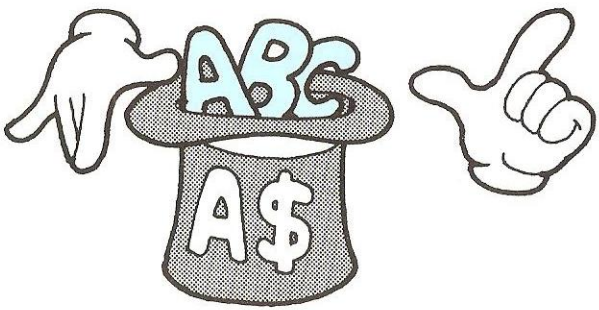
```
A$="ABC"
Ok
PRINT A$
ABC
Ok
■
```

Como puede ver, A$ se está utilizando como una variable, y su valor es "ABC". Compare esto con otro mandato:

```
A=345
Ok
PRINT A
 345
Ok
■
```

Esto le suena familiar, ¿no? A la variable A le hemos asignado el valor numérico 345, y no hay signo $ (dólares) al final del nombre de dicha variable A.

¿Ha adivinado la regla? **Cuando el nombre de una variable finaliza con un signo $, dicha variable tendrá asignado el valor de una letra o un juego de caracteres (denominado "cadena"). Sin el signo $ al final, la variable tendrá asignado un valor numérico.**



Aquí hay algo que tenemos que recordar: el valor deberá estar **entre comillas**, de la forma siguiente:

A$="ABC"

Pruebe estos ejemplos, y entenderá claramente la diferencia entre las variables numéricas y las alfanuméricas.

```
A=123                    A toma el valor de 123.
Ok
B=234                    B toma el valor de 234.
Ok
PRINT A+B                Visualización A+B
 357                     La suma es 357
Ok
A$="123"                 A$ toma el valor de "123"
Ok
B$="234"                 B$ toma el valor de "234"
Ok
PRINT A$+B$              Visualización A$+B$
123234                   La suma es "123234"
Ok
A$=987                   A$ toma el valor de 987
Type mismatch            ¡Piip! Es imposible.
Ok
■
```

Cuando introduzca A$="123", el valor de la variable A$ será 123, no ciento veintitrés sino los **caracteres** uno, dos, tres. En la línea siguiente, B$ tomará el valor de 234 (caracteres dos, tres, cuatro). ¿Por qué? Porque hemos utilizado comillas y el signo $. Esto indicó al ordenador que estábamos utilizando una variable alfanumérica.

Cuando el ordenador sume A$+B$, unirá (concatenará) estas variables en un solo juego de caracteres (cadena). Un ejemplo similar podría ser "ABC"+"BCD"="ABCBCD".

Finalmente hemos introducido A$=987. El ordenador respondió con un mensaje de error:

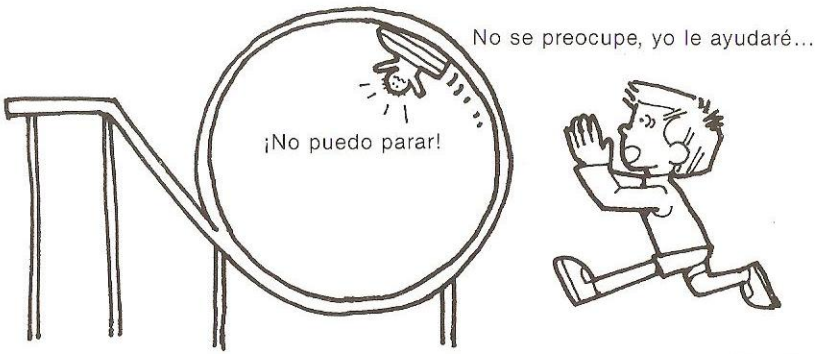
Type mismatch (discordancia de escritura en inglés)
(Usted escribió dos cosas que no concuerdan.)

Sin las comillas, 987 es un número (novecientos ochenta y siete) que **no concuerda** con la variable con signo $.

Ahora sabemos que cuando el nombre de una variable termina con un signo $, es una variable alfanumérica, a la que se le puede asignar una letra o un juego de caracteres, y el valor deberá estar "entre comillas". Ahora, volvamos a nuestro programa para el juego de la máquina tragaperras.

CÓMO PARAR UN BUCLE CON UNA SENTENCIA INKEY

El bucle FOR—NEXT que hemos utilizado en el juego acertijo se paraba solamente cuando su variable alcanzaba el valor final. Pero la línea de mandato 120, mandato GOTO 30, indica que el bucle de este juego de máquina tragaperras no se parará nunca.



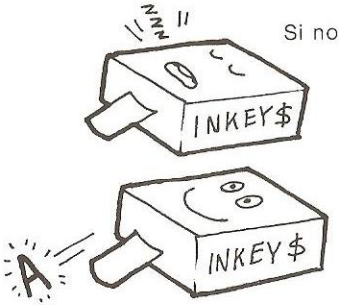
Naturalmente, usted podrá parar siempre su programa pulsando CTRL y STOP. Pero esto parará todo el programa y, si introduce el mandato RUN para comenzar de nuevo, todavía tendremos el bucle sin fin. Nosotros deseamos parar la máquina tragaperras y ver los símbolos para comprobar cuántos puntos hemos ganado o perdido. Después queremos que el programa comience de nuevo para que continúe el juego.

Una buena forma de parar la máquina tragaperras es añadir estas dos líneas al programa:

```
103 K$=INKEY$
106 IF K$="A" THEN END
```

K$ es, naturalmente, una variable alfanumérica y, como pedemos ver en la línea 106, cuando su valor sea A, la máquina tragaperras finalizará (END) la visualización giratoria.

¿Qué significa **INKEY$**? Ésta es una función del lenguaje BASIC. INKEY significa introducción desde el teclado (en inglés INput from the KEYboard) de cualquier tecla que pulse. Por consiguiente, la línea 103 significa que **K$ tomará el valor de cualquier carácter cuya tecla pulse**. Si usted no pulsa tecla alguna, el ordenador continuará sin asignar ningún valor a K$. Si pulsa [B], [C], o [X], el ordenador continuará haciendo girar la máquina tragaperras. Pero si pulsa [A] (mayúscula), la línea 106 mandará al ordenador que finalice (END).



Si no pulsa ninguna tecla, no haré nada.

Si pulsa [A], asignaré una A

Ahora nuestro bucle FOR—NEXT y GOTO 30, que va de la línea 30 a la 120, tiene un mandato END en el medio. Pero el ordenador no oirá el mandato (no parará la máquina tragaperras) a menos que se lo digamos pulsando la tecla [A]. En otras palabras, la máquina tragaperras continuará cambiando los símbolos hasta que la paremos.

```
30 FOR I=0 TO 2
      |
      |
      |
      |
      |
      |
103 K$=INKEY$
106 IF K$="A" THEN END
110 NEXT I
120 GOTO 30
```

Bucle FOR—NEXT que visualiza 3 símbolos en las ventanillas.

El programa finalizará cuando pulsemos la tecla [A].

¿CUÁL ES LA PUNTUACIÓN?

Cuando la máquina tragaperras pare de cambiar los símbolos, deseare-mos saber cuántos puntos hemos ganado (o perdido). Esto significa que que tendremos que poner reglas en el programa. ¿Recuerda las reglas? Nosotros comenzamos con un **haber** de 100 puntos. Después **apostamos** algunos de estos puntos, es decir, cualquier número entre 1 y 100. Añadamos esta información al programa.

```
23 P=100:LOCATE 2,18:PRINT USING "HAB
ER:####";P
26 LOCATE 0,20:INPUT "APUESTA";B
```

Observemos en primer lugar la línea 23. Nuestro haber cambiará cada vez que apostemos, por lo que utilizamos una variable, P. Esta P tiene un valor inicial de 100. Para visualizar la puntuación, utilizamos un mandato LOCATE, y después otro PRINT.

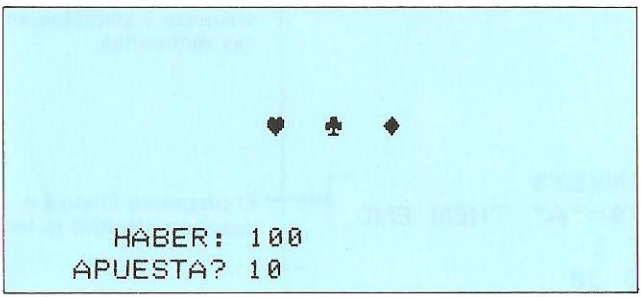
```
PRINT USING "HABER:# # # #";P
```

Este nuevo mandato PRINT (USING significa utilizando en inglés) indica al programa el **formato** en el que tiene que visualizar la información. El signo # significa "dígitos". Este mandato ordena al programa que visualice P utilizando los cuatro espacios vacíos de la pantalla siguientes a "HABER:. El juego comenzará con un haber de 100, por lo que la primera visualización en 2,18 será:

HABER: ␣ 100

4 dígitos

La línea 26 ordena al programa que introduzca la apuesta en la pantalla, justamente debajo del haber. Su apuesta cambiará cada vez que usted lo desee, por lo que esta línea utiliza la variable B para tal apuesta. El mandato INPUT ordena al programa que espere por el valor de B, es decir, por la apuesta, introducido desde el teclado. Si su apuesta son 10 puntos, se visualizará de la forma siguiente:



Después de las líneas 23 y 26, el programa avanzará hasta la sección FOR—NEXT. Introduzcamos ahora en nuestro programa las nuevas líneas de mandatos 23, 26, 103, y 106.

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING"HAB
ER:####";P
26 LOCATE 0,20:INPUT "APUESTA";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$="A" THEN END
110 NEXT I
120 GOTO 30
```

Adición

Cambio

Adición

Tenga en cuenta que deberá cambiar el mandato GOTO 110 de las líneas 70, 80, y 90 a GOTO 103. ¿Sabe lo que sucederá si se olvida de hacerlo?

Ahora la máquina tragaperras estará lista para que usted haga su primera apuesta, pero todavía no hemos indicado al ordenador cómo cambiar el haber, es decir, cómo mantener la puntuación. Después añadiremos algunos mandatos nuevos al final del programa comenzando por la línea 130. Pero regresemos primeramente a la línea 106.

```
106 IF K$="A" THEN END
```

Cuando, pulsando la tecla [A], hacemos que la máquina tragaperras deje de cambiar los símbolos, en realidad no deseamos que finalice el programa. Lo que nos interesa es que el ordenador nos diga la puntuación. Por lo tanto, cambie la línea 106 de la forma siguiente:

```
106 IF K$="A" THEN GOTO 130
```

Nosotros sabremos si hemos ganado o perdido cuando veamos los símbolos en la pantalla. Si los tres son iguales, habremos ganado el triple de nuestra apuesta. (Si nuestro haber era 100, apostamos 10, y nos salen tres símbolos iguales, nuestra nueva puntuación o haber será 130.) Si conseguimos dos símbolos iguales, habremos ganado la misma cantidad que la apostada (la nueva puntuación será 110). Si lo que vemos son

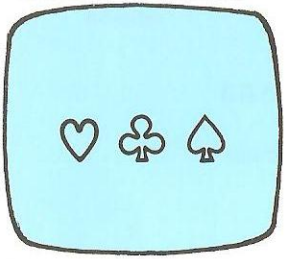
tres símbolos diferentes, habremos perdido el doble de lo apostado (la nueva puntuación será 80). Por ejemplo, cuando veamos

♥ ♣ ♠

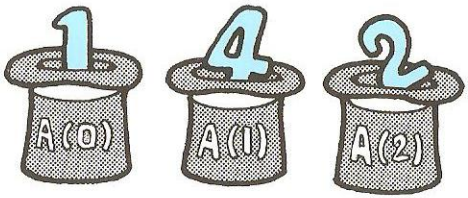
sabremos que habremos perdido algunos puntos.

Pero nuestro ordenador no observa los símbolos de la pantalla. En su lugar utiliza las variables que tiene en su memoria. El ordenador sabrá rápidamente si A(0), A(1), y A(2) son iguales comprobando los valores de estas variables de matriz.

Nosotros vemos esto.



La memoria del ordenador contiene esto.



Si la pantalla muestra:

♥ ♥ ♥

¿sabe lo que habrá en la memoria del ordenador?

Como puede ver, no es muy difícil para el ordenador decidir si hay dos o tres símbolos símbolos iguales, o si todos son diferentes. Y esto significa que los mandatos de nuestro programa para la puntuación tampoco son muy difíciles.

```
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 8,18:PRINT USING "####";P
160 GOTO 26
```

La primera línea de estos nuevos mandatos de puntuación es la 130. Esto significa que la parte de puntuación del programa se utilizará solamente después de haber pulsado la tecla [A] a fin de parar la máquina tragaperras.

```
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
```

Observe que el mandato IF—THEN tiene **dos** condiciones conectadas con **AND** (y en inglés). Si **ambas** son verdaderas, es decir, A(0)=A(1), y A(0)=A(2), entonces las tres variables de matriz serán iguales, y también lo serán los tres símbolos de la pantalla. Si (IF) esto es cierto

```
entonces THEN P=P+B * 3:GOTO 150
```

Esto significará que su apuesta se multiplicará por 3, y se añadirá al haber, P, y el ordenador irá a la línea 150. Si usted apuesta 10 con un haber de 100, la mueva puntuación será 130.

Pero si las tres variables no son iguales, el ordenador no seguirá el mandato THEN. En su lugar, pasará a la línea siguiente del programa.

```
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
```

Aquí tenemos un mandato IF con tres condiciones, y están conectadas mediante **OR** (o en inglés). Esto significa que si **un** par cualquiera de las variables es igual, (o dos símbolos son iguales),

```
THEN P=P+B
```

añadirá su apuesta al haber, con lo que usted obtendrá una nueva puntuación de 110. Pero si ninguna de las condiciones es cierta (todos los símbolos son diferentes), el programa irá a

```
ELSE P=P-B * 2
```

¡Usted habrá perdido! Su apuesta se multiplicará por dos y se restará del haber, por lo que la nueva puntuación será solamente 80.

Ahora el ordenador conocerá la puntuación, y el paso siguiente será visualizarla.

```
150 LOCATE 8,18:PRINT USING "####";P
```

Esto es fácil de entender porque es casi igual que el mandato PRINT con formato de la línea 23: primeramente LOCATE, después PRINT USING el formato de cuatro dígitos, después el nombre de la variable que adquiere un nuevo valor. Cada vez que pare la máquina tragaperras, el ordenador calculará la nueva puntuación, y la visualizará en la posición 8,18.

¿Qué sucederá a continuación? El juego continuará, lo que significa que usted deberá volver a apostar. Por lo tanto, la línea 160 hará que el programa vuelva a la línea 26, donde esperará hasta que usted introduzca desde el teclado su nueva apuesta.

Introduzca ahora en el programa los nuevos mandatos de puntuación. El programa deberá ser como sigue:

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "HAB
ER :####";P
26 LOCATE 0,20:INPUT APUESTA";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$="A" THEN GOTO 130      <-- Cambio
110 NEXT I
120 GOTO 30
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150                  <-- Adición
140 IF A(0)=A(1) OR A(1)=A(2)OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 8,18:PRINT USING"####";P
160 GOTO 26
```

¡Espere un momento! Este programa es demasiado largo para caber en la pantalla. Si introduce el mandato LIST, la primera parte del programa desaperecerá de la pantalla.

Para resolver este sencillo problema, deberemos decirle al ordenador **lo que** tiene que listar, de la forma siguiente:

```
LIST 10-100
```

Con este mandato, se visualizarán las líneas 10 a 100. Para ver la última parte, introduzca

```
LIST 110-
```

Ahora utilice mandatos LIST para comprobar cuidadosamente que no haya errores. Cuando esté seguro de que el programa es correcto, introduzca el mandato RUN, ¡y disfrute de su nuevo juego de máquina tragaperras!

ÚLTIMOS RETOQUES

Su ordenador Sony se ha convertido en una máquina tragaperras que trabaja perfectamente. Este programa realiza operaciones muy complicadas: solicita apuestas, cambia los símbolos de las tres ventanillas, y calcula la puntuación cuando usted para los símbolos. Nuestro programa para todas estas cosas emplea solamente 20 líneas de mandato, no muchas si piensa que el ordenador tiene que decirnos **cada** vez que visualice, calcule, se pare, espere por una apuesta, etc.

Pero nos hemos olvidado de una cosa. El plan inicial era que usted ganaría cuando su puntuación llegase a 300, o perdería cuando disminuyese hasta 0. Posiblemente usted habrá sobrepasado 300 puntos, o llegado a 0, o quizás ambas cosas.

Convirtamos nuestro buen programa en otro mejor, añadiendo mandatos para pérdida o ganancia. Además, hay otras mejoras que podemos efectuar para convertir el programa en **perfecto**.

- Podemos ampliar un poco el juego para hacer posible introducir otra apuesta diferente, solamente después de pulsar la barra espaciadora del teclado.
- La barra espaciadora puede ser más conveniente que la tecla [A] a fin de parar el juego, porque esta barra es más fácil de pulsar.
- Cuando hacemos una nueva apuesta, la anterior todavía permanece en la pantalla. Esto puede resultar confuso.

¡Muy pronto el programa será perfecto!

Aquí tenemos los cambios que podremos introducir en nuestro programa.

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "HAB
ER:####";P
24 LOCATE 7,20:PRINT "          "
26 LOCATE 0,20:INPUT "APUESTA";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$=" " THEN GOTO 130
110 NEXT I
120 GOTO 30
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 8,18:PRINT USING "####";P
152 GOTO 170
154 K$=INKEY$
156 IF K$=" " THEN GOTO 24
160 GOTO 154
170 IF P<300 AND P>0 THEN GOTO 154
180 IF P>=300 THEN LOCATE 3,5:PRINT "
HA GANADO"
190 IF P<=0 THEN LOCATE 3,5:PRINT "HA
 PERDIDO"
200 END
```

Adición

Cambio

Adición

Cambio

Adición

¿Comprende estos cambios? Antes de leer la explicación, trate entenderlos.

La línea comienza por un nuevo juego de mandatos sobre pérdida o ganancia. Si la puntuación es inferior a 300 y (AND) mayor que 0, entonces (THEN) el juego continuará, y el ordenador volverá al programa (GOTO 24). Si la puntuación es 300 o más, el juego finalizará, y cerca de la parte superior de la pantalla se visualizará HA GANADO. Cuando la puntuación sea igual o inferior a 0, se visualizará HA PERDIDO. Y en cualquiera de los dos casos, el programa pasará a la línea 200 y el juego habrá llegado a su fin (END).

La línea 24 borrará la apuesta anterior al comienzo de cada tirada. Moverá el cursor hasta la visualización de la apuesta, y en su lugar introducirá seis espacios en blanco.

La línea 106 tiene un cambio muy sencillo. Ahora tendremos que parar la máquina tragaperras con la barra espaciadora, no con la tecla [A]. Recuerde que su ordenador leerá un **espacio (" ")** como un **carácter**.

Las líneas 160, 154, y 156 indican que el juego comenzará de nuevo cuando pulsemos la barra espaciadora.

Éstos son los cuatro cambios que queríamos hacer: pérdida/ganancia, y las tres pequeñas mejoras.

La línea 152 no cambiará realmente el juego, pero es necesaria debido a los otros cambios del programa. Esta línea viene inmediatamente después de la visualización de la nueva puntuación, y enviará el programa hasta la nueva sección de la línea 170 ganancia/pérdida. Si usted no ha ganado ni perdido, la línea 170 devolverá el programa a la línea 154, y el juego continuará cuando usted pulse la barra espaciadora.

PARA PODER LEER MAS FÁCILMENTE LOS PROGRAMAS

Cuando comenzamos a escribir el programa de la máquina tragaperras, utilizamos números de líneas separados de diez en diez: 10, 20, 30, 40 ... Después, a medida que fuimos haciendo más completo el programa, utilizamos algunas líneas "vacías" para añadir nuevos mandatos. Como resultado, nuestros números de líneas quedaron en grupos poco usuales como 90, 100, 103, 106, 110, 120, etc. Para facilitar la lectura del programa, podemos **renumerar** las líneas.

Introduzca este mandato:

```
RENUM
```

Las líneas seguirán teniendo el mismo orden, pero sus números estarán separados de diez en diez, como podremos ver si introducimos el mandato LIST:

Ahora los números de línea se pueden leer mucho más fácilmente. ¿Pero qué ha sucedido con nuestros mandatos GOTO que contenían números de línea? ¡No hay problema alguno! Por ejemplo, observe la línea 100 del nuevo programa:

```
100 PRINT "♥":GOTO 140
```

Este mandato GOTO (GOTO 140) era GOTO 103 antes de utilizar el mandato RENUM. Ahora la línea 103 ha pasado a ser la 140, y GOTO 103 es GOTO 140. Esto significa que su ordenador Sony es bastante inteligente, y cuando ejecute de nuevo el programa de máquina tragaperras, el juego no cambiará en absoluto.

TITULACIÓN DE PROGRAMAS

Suponga que desea mostrar un programa a un amigo. O que quiere leer un programa después de varios meses de haberlo escrito. Si el lector no está muy familiarizado con el BASIC, cuando en la pantalla aparezcan 20 o 30 líneas, será muy difícil que entienda sus funciones. Para resolver este problema, el BASIC le permite poner "observaciones" como ésta en medio de sus programas:

```
5 REM * JUEGO DE MÁQUINA TRAGAPERRAS *
7 REM
10 DIM A(2)
20 CLS
30 P=100:LOCATE 2,18:PRINT USING "HABER
:####";P
40 LOCATE 7,20:PRINT "       "
50 LOCATE 0,20:INPUT "APUESTA";B
55 '
60 ' *** COMIENZO DE LA MÁQUINA TRAGA
PERRAS ***
62 '
65 FOR I=0 TO 2          — Línea 60 anterior
70 A(I)=INT(RND(1)*4)+1
80 LOCATE I*3+12,9
```

```
 90 ON A(I) GOTO 100,110,120,130
100 PRINT "♥":GOTO 140
110 PRINT "♠":GOTO 140
120 PRINT "♦":GOTO 140
130 PRINT "♣"
140 K$=INKEY$
150 IF K$=" " THEN GOTO 180
160 NEXT I
170 GOTO 60
175 '
180 ' *** CALCULO DEL HABER ***
182 '
185 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 200 — Línea 180 anterior
190 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
200 LOCATE 8,18:PRINT USING "####";P
210 GOTO 250
220 K$=INKEY$
230 IF K$=" " THEN GOTO 40
240 GOTO 220
245 '
250 ' *** GANANCIA O PÉRDIDA ***
252 '                    Línea 250 anterior
255 IF P<300 AND P>0 THEN GOTO 220 ↵
260 IF P>=300 THEN LOCATE 3,5:PRINT "
HA GANADO"
270 IF P<=0 THEN LOCATE 3,5:PRINT "HA
 PERDIDO"
280 END
```

Nuestras "observaciones" aquí son el **título** del programa y los **subtítulos** para sus diferentes partes. Estas observaciones alargan el programa, pero hacen que sea mucho más fácil de leer y entender, y no cambian en absoluto el juego, porque su ordenador no leerá las nuevas líneas ya que éstas solamente son para comodidad de la gente que utilice el programa.

Observemos algunas de las nuevas líneas de observaciones:

```
5 REM * JUEGO DE MÁQUINA TRAGAPERRAS*
7 REM
```

El mandato REM (REMarks, que en inglés significa observaciones) indica al ordenador que **ignore** los caracteres que vengan después del mandato. Usted podrá introducir este mandato siempre que desee, y el programa no cambiará. Después de REM de la línea 7 no hay nada. De esta forma quedará un espacio vacío alrededor del título, al igual que la página del título de un libro.

El espacio vacío, y las estrellas (asteriscos) después del título facilitan la búsqueda de dicho título entre todas las líneas de mandatos del BASIC.

Ahora observemos las líneas 55, 60, y 62. Como podrá ver, existe una forma más corta de utilizar el mandato REM: con una ' (comilla simple). Naturalmente, estas líneas no forman en realidad parte del programa.

El ordenador no las leerá como mandatos, porque comienzan por '. El subtítulo, las estrellas, y los espacios vacíos son para las personas, no para el ordenador.

El ordenador ve los números de línea, porque están antes de ' (o REM). Por ejemplo, en la línea 150, tenemos:

```
150 IF K$=" " THEN GOTO 180
```

Pero 180 es una línea de observaciones:

```
180 ' *** CÁLCULO DEL HABER ***
```

El ordenador no encontrará mandato alguno en la línea 180, por lo que pasará a la siguiente. La línea 182 tampoco tiene mandatos, motivo por el cual el programa irá a 185. Pero el **lector**, usted o algún amigo, al leer de la línea 150 a la 180 sabrá inmediatamente que ésta es la sección donde se calcula el haber.

HA RECORRIDO UN LARGO CAMINO

¡Enhorabuena! Después de haber realizado los ejercicios y juegos de este libro, habrá adquirido el conocimiento práctico de un nuevo lenguaje: el BASIC. Quizás fue hace solamente algunas horas, o tal vez uno o dos días, cuando usted empezó con mandatos muy sencillos como PRINT 3+5. Pero ahora puede entender y **utilizar** líneas complicadas como A(I)=INT(RND(1) * 4)+1, y PRINT USING" # # # #";P. Su ordenador Sony, ahora que usted sabe cómo "hablar en su lenguaje", le brindará muchos años de servicio, entretenimiento, y educación.

A medida que practique escribiendo y utilizando programas, adquirirá cada vez mayor habilidad en el uso de números, gráficos, mandatos, funciones, y juegos. El convertirse en un **experto** es sólo cuestión de tiempo, y el tiempo pasará volando a medida que comparta la diversión que le ofrecerá este ordenador Sony con su familia y sus amistades.

En la sección siguiente de este libro podrá aprender y practicar algunos nuevos mandatos. Después de ello, podrá utilizar el **Manual de Referencia para Programación en BASIC MSX** a fin de probar nuevos mandatos del BASIC, y descubrir lo que pueden hacer. Y después podrá experimentar con su ordenador para crear nuevos colores, gráficos, sonidos, y juegos.

Cuando desee crear **sus propios programas**, emplee primeramente algunos minutos para planearlos. Quizás le resulte muy útil hacer diagramas de flujo. Cuando sepa qué **funciones** y qué tipo de **mandatos** va a necesitar, utilice el Índice alfabético que se encuentra al final de este libro para buscarlos y revisar las explicaciones. Planee cuidadosamente los mandatos, escriba, y ejecútelos. Su ordenador Sony le ayudará de dos formas: hará siempre y **exactamente** lo que usted le mande, y olvidará los errores que usted cometa cuando introduzca el mandato NEW.

Hable de sus programas con sus amigos. Enséñeles lo que pueden hacer con su ordenadores. Ellos quizás tengan algunas ideas diferentes, y juntos podrán distrutar más. Si sus amigos tienen sus propios ordenadores, puede intercambiar programas con ellos, utilizando un magnetófono o escribiendo los programas en un papel.

Recuerde que la mejor forma de divertirse con su ordenador Sony es intentando nuevas cosas, y viendo lo que sucede. No hay límites en lo que usted pueda decirle, en lenguaje BASIC, a su ordenador. ¿Está preparado para iniciar el maravilloso viaje a través de su imaginación? ¡Buena suerte!

PRACTIQUEMOS MANDATOS DEL BASIC

PRINT

```
10 PRINT "JUAN"
20 PRINT "Y"
30 PRINT "MERCEDES"
RUN
JUAN
Y
MERCEDES
```

¿Recuerda este sencillo programa? Cambiémoslo un poco.

```
10 PRINT "JUAN";
20 PRINT "Y";
30 PRINT "MERCEDES"
RUN
JUANYMERCEDES
```

Adición

Con el ; (punto y coma) después de "JUAN" e "Y", el ordenador visualizará las tres palabras juntas, en una línea. En realidad, los mandatos pueden escribirse juntos en una sola línea de programa:

```
10 PRINT "JUAN";"Y";"MERCEDES"
RUN
JUANYMERCEDES
```

Importante

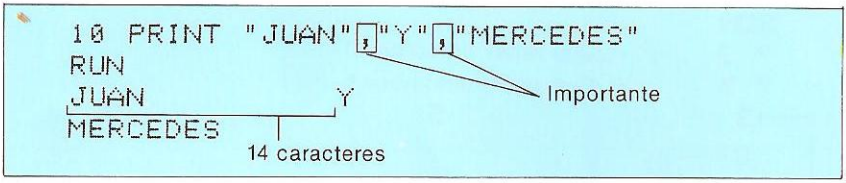
Pero es difícil leer estas tres palabras sin espacios entre ellas, por lo tanto intentemos otra cosa.

```
10 PRINT "JUAN",
20 PRINT "Y",
30 PRINT "MERCEDES"
RUN
JUAN          Y
MERCEDES
```

Importante

14 caracteres

Cuando utilicemos una , (coma) en vez de un ; (punto y coma), las primeras letras de cada palabra se imprimirán separadas 14 espacios. Lo mismo sucederá si introduce el mandato en una línea.



Ahora practique este mandato con un solo espacio entre cada palabra y las comillas de cierre.

```
10 PRINT "JUAN ";"Y ";"MERCEDES"
RUN
JUAN Y MERCEDES
```

Recuerde: el **espacio** es un carácter muy importante.

Revisemos ahora las diferentes formas de impresión de los caracteres numéricos.

```
10 PRINT "3+5=";   ← Visualización de caracteres
20 PRINT 3+5       ← Cálculo y visualización de la respuesta
RUN
3+5=  8
```

La línea 10 es un mandato para visualizar caracteres, y la línea 20 es un mandato para calcular. Pero estas dos diferentes funciones están **conectadas** por el ; al final de la línea 10. Esto indica al ordenador que las imprima juntas en una sola línea, al igual que lo hacemos normalmente al escribir en un papel. (En esta respuesta hay un espacio vacío antes del 8. Este espacio se utilizará para visualizar el signo menos (–) cuando la respuesta sea un **número negativo**, por ejemplo, 3–2=2.)

Utilicemos los mandatos INPUT y PRINT en este mismo sencillo programa.

```
10 INPUT A
20 INPUT B
30 PRINT A,B,A+B
RUN
? 3          ←¿Qué valor? Introduzca 3
? 5          ←¿Qué valor? Introduzca 5
 3                    5
  8
```

Valor de A — Valor de A+B — Valor de B

El mandato INPUT pide al ordenador que **solicite** el valor que usted desee asignar a la variable, y que espere a que usted ponga (PUT) un valor en (IN) en el ordenador a través del teclado. Después de pulsar RETURN, el ordenador pasará al mandato siguiente. La línea 30 visualizará los tres valores en la pantalla.

Aquí tenemos una forma diferente y más clara de realizar las mismas funciones.

```
10 INPUT "A=";A
20 INPUT "B=";B
30 C=A+B
40 PRINT "A+B=";C
RUN
A=? 3
B=? 5
A+B= 8    ←Resultado de la línea 40
```

Las líneas 10 y 20 indican la pregunta que tienen que visualizar, en vez de utilizar solamente el **?** (signo de interrogación). En la línea 30 tenemos una nueva variable, C, y da el valor de A + B. La línea 40 indica al ordenador que imprima la respuesta y el problema, comenzando por A + B=. Las tres líneas después de RUN son mucho más fáciles de entender que los tres números solos al final del último ejemplo.

Algunas veces es mucho más fácil leer visualizaciones cuando hay **líneas vacías** entre palabras, o entre juegos de palabras.

```
10 PRINT "JUAN"
20 PRINT
30 PRINT "Y"
40 PRINT
50 PRINT "MERCEDES"
RUN
JUAN

Y

MERCEDES
```

Una línea vacía entre medio

Una línea vacía entre medio

Cuando en la línea se introduzca solamente el mandato PRINT (como en las líneas de mandatos 20 y 40), se visualizará una línea vacía.

INPUT

```
10 INPUT "A es ";A
20 INPUT "B es ";B
30 PRINT "A+B=";A+B
40 PRINT "A-B=";A-B
RUN
A es ? 15
B es ? 3
A+B= 18
A-B= 12
```

Aquí tenemos otro programa que combina los mandatos INPUT y PRINT. Es muy fácil ver que este programa indica al ordenador que **solicite** valores para A y B, y que espués **visualice** los cálculos. Pongamos ahora juntos los dos mandatos INPUT, y añadamos algunos cálculos más.

```
10 INPUT "A y B son ";A,B
20 PRINT "A=";A,"B=";B
30 PRINT
40 PRINT "A*B=";A*B
50 PRINT "A/B=";A/B
RUN
A y B son ? 15,3
A= 15          B= 3

A*B= 45
A/B= 5
```

En la línea 10, usted puede introducir dos valores para dos variables diferentes, pero la , (coma) entre los dos valores (en este caso 15 y 3) es **muy** importante. Si el ordenador visualizase 153 en vez de 15,3, sería muy difícial entender. (El mandato PRINT de la línea 20 tiene además signos de puntuación muy importantes. ¿Sabe lo que significan la , y el ;?)

Ahora añadamos una **variable alfanumérica** a nuestro mandato INPUT.

```
10 INPUT "Nombre";N$
20 PRINT N$;" es grande."
RUN
Nombre? Juan
Juan es grande.
```

Cuando el nombre de la variable finalice con el signo $, la variable "tomará el valor de" una letra o una cadena. Aquí tenemos un programa que utiliza una variable alfanumérica y otra de número.

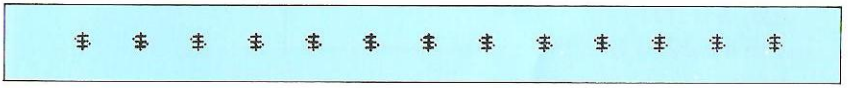
```
10 INPUT "Nombre";N$
20 INPUT "Edad";Y
30 PRINT N$;" tiene ";Y;" años."
RUN
Nombre? Juan
Edad? 10
Juan tiene 10 años.
```

FOR—NEXT

```
10 CLS
20 FOR I=0 TO 36
30 LOCATE I,10
40 PRINT "$"
50 NEXT I
```

En este programa, la función de repetición (**bucle**) del mandato FOR—NEXT hace que el ordenador visualice signos $ en la pantalla desde la posición 0,10 a la 28,10.

```
10 CLS
20 FOR I=0 TO 36 STEP 3
30 LOCATE I,10
40 PRINT "$"
50 NEXT I
```



¿Cuál es el significado de **STEP 3** de la línea 20? Como puede ver, este mandato ha movido el cursor **tres pasos** después de cada signo $. En otras palabras, el valor de I cambiará ahora en pasos de 3, de forma que el signo $ aparecerá en las posiciones 0,10, 3,10, 6,10 ..., no en 0,10, 1,10, 2,10, ... Los signos $ cubrirán casi todo el área de la pantalla, como en el programa anterior, desde 0,10, hasta 36,10.

Por lo tanto, con **STEP y un número** añadidos al mandato FOR, **podrá hacer que la variable cambie en pasos**. En el último ejemplo, la variable era la posición, por lo que ésta cambió en pasos. El programa siguiente utiliza STEP para cambiar la forma de contar del ordenador.

```
10 FOR I=50 TO 0 STEP -5
20 PRINT I,
30 NEXT I
RUN
 50             45
 40             35
 30             25
 20             15
 10             5
 0
```

I=50 TO 0 hace que el ordenador cuente cada vez que ejecute el bucle FOR—NEXT. Entre 50 y 0 hay 51 valores para I. Pero el mandato **STEP—5** hace que los valores disminuyan en pasos de -5, motivo por el que solamente habrá 11 valores para I.

FOR—FOR—NEXT—NEXT: Un bucle dentro de otro

```
10 FOR I=0 TO 2
20 PRINT "I=";I
30 FOR J=0 TO 4
40 PRINT "J=";J;
50 NEXT J
60 PRINT
70 NEXT I
RUN
I= 0
J= 0  J= 1  J= 2  J= 3  J= 4
I= 1
J= 0  J= 1  J= 2  J= 3  J= 4
I= 2
J= 0  J= 1  J= 2  J= 3  J= 4
```

FOR … NEXT

FOR … NEXT

Este programa muestra cómo poner un bucle FOR—NEXT dentro de otro también FOR—NEXT, más ancho. La variable I se controla mediante la primera y última líneas del programa, y la variable J mediante las líneas 30 y 50. El resultado es que, para cada valor de I, el ordenador hará cuatro pequeños bucles para buscar todos los valores de J. I cambiará de 0 a 2, y con cada, I, J cambiará de 0 a 4.

Aquí tenemos otra forma de utilizar un bucle.

```
10 INPUT N,S
20 CLS
30 FOR I=0 TO N STEP S
40 LOCATE 5,10:PRINT "I=";
50 PRINT USING "####";I
60 FOR J=0 TO 500
70 NEXT J
80 NEXT I
```

Este programa utiliza los mandatos INPUT, FOR—NEXT y STEP para hacer que la variable I cambie en bucles desde 0 hasta N, en pasos de S. Naturalmente, tendremos que asignar valores a N y S desde el teclado.

¿Pero cuál es la función de la variable J y su corto bucle FOR—NEXT de las líneas 60 y 70? J no se imprimirá nunca. Esta variable no se utilizará en absoluto, pero el ordenador encontrará cada vez 501 valores para J, antes de continuar calculando e imprimiendo I. Éste es un bucle **ficticio**. Para realizar 501 bucles FOR—NEXT, el ordenador emplea cierto tiempo, no demasiado, pero no hace nada más.

La función del bucle J, por lo tanto, es poner un **intervalo de tiempo** dentro de otro mayor, I. Cuando ejecute este programa, verá que el ordenador hace una pausa después de cada visualización.
Ahora ya sabe lo rápidamente que su ordenador Sony puede ejecutar 501 cosas sencillas, y cómo utilizar un bucle ficticio para poner algún tiempo extra en su programa. ¿Podría escribir un bucle ficticio que mantenga ocupado el ordenador durante más tiempo?

IF—THEN e IF—THEN—ELSE

Aquí tenemos otro juego acertijo, en el que usted tendrá que intentar acertar el número que elija el ordenador. Pero este juego es ligeramente diferente al juego acertijo y al juego de máquina tragaperras que hemos programado anteriormente. Aquellos eran juegos puramente de suerte, sin embargo, esta vez podemos utilizar cierta **maña** para adivinar más rápidamente el número correcto. Si usted juega inteligentemente, será capaz de encontrar el número en seis o siete intentos. Pero si juega al azar, tendrá que realizar muchos más intentos.

```
10 X=INT(RND(1)*100)+1
20 INPUT "Adivine";A
30 IF A=X THEN GOTO 70
40 IF A>X THEN PRINT "MENOR"
50 IF A<X THEN PRINT "MAYOR"
60 GOTO 20
70 PRINT "Acierto"
```

Las líneas 30, 40, y 50 mandan al ordenador que compare el número que usted introduzca con el valor de X, y que le dé una pista para ayudarle a encontrar el número correcto.

¿Puede encontrar el cambio en este otro programa?

```
10 X=INT(RND(1)*100)+1
20 INPUT "Adivine";A
30 IF A=X THEN 70
40 IF A>X THEN PRINT "MENOR"
50 IF A<X THEN PRINT "MAYOR"
60 GOTO 20
70 PRINT "Acierto"
```

Este programa es realmente igual que el anterior, incluso aunque se haya cambiado la línea 30. IF—THEN tiene el mismo significado que IF—THEN—GOTO. Además, el programa sería iqual con: 30 IF A=X GOTO 70. En otras palabras, IF—GOTO tiene el mismo significado que IF—THEN—GOTO. Es posible eliminar THEN o GOTO (¡pero no ambas palabras!).

Combinemos ahora dos líneas de este programa en una sola:

```
10 X=INT(RND(1)*100)+1
20 INPUT "Adivine";A
30 IF A=X THEN 70 ELSE
   IF A>X THEN PRINT "MENOR"
50 IF A<X THEN PRINT "MAYOR"
60 GOTO 20
70 PRINT "Acierto"
```

Los dos mandatos IF—THEN de las líneas 30 y 40 se han convertido en el mandato IF—THEN—ELSE. Ahora no hay línea 40, pero el significado es el mismo. Si esto es posible, tiene que serlo también hacer de las líneas 30 y 50 de este programa una sola.

```
10 X=INT(RND(1)*100)+1
20 INPUT "Adivine";A
30 IF A=X THEN 70 ELSE
   IF A>X THEN PRINT "MENOR"
   ELSE PRINT "MAYOR"
60 GOTO 20
70 PRINT "Acierto"
```

Usted puede comprobar para ver si estos cuatro programas son realmente iguales, ejecutando cada uno de ellos en su ordenador.

DIM (Variables de matriz)

Cuando utilizamos variables de matriz en el programa del juego de máquina tragaperras, empleamos el mandato

DIM A(2)

para hacer tres variables:

A(0) A(1) A(2)

Después cambiamos los números entre paréntesis por una variable, (I), y dijimos al ordenador que I=0 TO 2. Todavía teníamos las mismas tres variables, y el ordenador las cambió por un bucle FOR—NEXT.

Las variables de matriz pueden tener más de un carácter entre paréntesis. Por ejemplo, si introducimos

DIM A(3,4)

obtendremos 20 variables:

A (Ø, Ø)	A (1, Ø)	A (2, Ø)	A (3, Ø)
A (Ø, 1)	A (1, 1)	A (2, 1)	A (3, 1)
A (Ø, 2)	A (1, 2)	A (2, 2)	A (3, 2)
A (Ø, 3)	A (1, 3)	A (2, 3)	A (3, 3)
A (Ø, 4)	A (1, 4)	A (2, 4)	A (3, 4)

¿Comprende las 20 diferentes variables? El primer número entre paréntesis puede tener cuatro valores, y el segundo cinco: 4 por cinco es 20. (¿Cuántas habrá si escribimos DIM A(9,9)?)

Hemos mostrado las 20 variables no en una simple lista, sino en forma de **tabla**. (Usted puede imaginarse esto como una matriz biDIMensional, en la que el primer valor cambia de izquierda a derecha, y el segundo de la parte superior a la inferior.) Como puede ver, las variables de matriz son muy útiles para programas utilizados para confeccionar gráficos y tablas. Para utilizar estas variables de matriz bidimensionales en un programa, tendremos qe reemplazar los números entre paréntesis por variables, como por ejemplo, A(I,J).
Aquí tenemos un ejemplo de tabla real:

	Lectura	Escritura	Matemáticas	Total
1.er trimestre	68	88	70	226
2.º trimestre	73	53	91	217
3.er trimestre	92	98	82	272
Total	233	239	243	715
Media	77	79	81	238

Esta tabla muestra las calificaciones de tres materias de un alumno durante tres trimestres. Con los totales y las medias tenemos en la tabla exactamente 20 números diferentes. Esto significa que con el mismo mandato que antes, DIM (3,4), tenemos suficientes variables para confeccionar esta tabla. Pensemos ahora en las formas en las que podemos utilizar programas para asignar valores reales a nuestras variables de matriz, A(I,J). Aquí tenemos una:

```
100 FOR J=0 TO 2
110 INPUT A(0,J)
120 NEXT J
```

Estas tres líneas son para las calificaciones de lectura, que pasan a ser las variables A(0,0), A(0,1), y A(0,2). Cada vez que el programa vaya a la línea 110 en el bucle FOR—NEXT, solicitará la introducción de los valores (68, 73, y 92).

A continuación, calculará el total de las tres calificaciones de lectura que introdujimos:

```
200 T=0                    ····· T es la variable para el total de lectura.
210 FOR J=0 TO 2
220 T=T+A(0,J)
230 NEXT J
240 A(0,3)=T
```

De la misma forma, podemos introducir la calificación de escritura del 1.er trimestre en la variable A(1,0), y la de matemáticas del mismo trimestre en A(2,0). El programa siguiente introducirá el total de las tres materias de este trimestre en A(3,0).

```
300 T1=0
310 FOR I=0 TO 2
320 T1=T1+A(I,0)
330 NEXT I
340 A(3,0)=T1
```

T1 es la variable para el total del 1.er trimestre.

Para programar las otras calificaciones, y sus totales y medias, podemos utilizar bucles FOR—NEXT similares con variables de matriz en la forma de A(I,J). Éste es el programa completo:

```
 10 ' * PROGRAMA DE TABLA DE CALIFICAC
 IONES *
 20 '
 30 DIM A(3,4)
 40 CLS
 50 '
 60 ' *INTRODUCCIÓN DE CALIFICACIONES*
 70 FOR I=0 TO 2
 80 FOR J=0 TO 2
 90 ON I+1 GOTO 100,120,140
100 LOCATE 0,J:PRINT "Lectura, Trimest
re ";J+1;
110 INPUT A(0,J):GOTO 160
120 LOCATE 0,J+3:PRINT "Escritura, Tri
mestre ";J+1;
130 INPUT A(1,J):GOTO 160
140 LOCATE 0,J+6:PRINT "Matemáticas, T
rimestre ";J+1
150 INPUT A(2,J)
160 NEXT J
170 NEXT I
180 '
190 ' * CÁLCULO DE TOTALES Y MEDIAS *
200 FOR J=0 TO 2
210 T=0
220 FOR I=0 TO 2
230 T=T+A(I,J)
240 NEXT I
250 A(3,J)=T
260 NEXT J
```

```
270 FOR I=0 TO 3
280 T=0
290 FOR J=0 TO 2
300 T=T+A(I,J)
310 NEXT J
320 A(I,3)=T
330 A(I,4)=INT(A(I,3)/3)
340 NEXT I
350 '
360 ' *** FORMACIÓN DE LA TABLA ***
370 CLS
380 LOCATE 5,0
390 PRINT "LECT ESCR  MAT   TOTAL"
400 FOR S=1 TO 3
410 LOCATE 2,S+1:PRINT S
420 NEXT S
430 LOCATE 1,5:PRINT "TTL"
440 LOCATE 1,6:PRINT "MED"
450 FOR I=0 TO 3
460 FOR J=0 TO 4
470 LOCATE I*6+4,J+2
480 PRINT A(I,J)
490 NEXT J
500 NEXT I
```

Para facilitar nuestro planeamiento, dividimos el programa en tres secciones: introducción de calificaciones, cálculo de totales y medias, y confección de la tabla. Además, si usted escribe en un papel el lugar en el que se introduce cada variable, será más fácil recordar cómo trabaja el programa.

Naturalmente, las variables de matriz pueden utilizarse para otros fines que no sean "máquinas tragaperras" o tablas. Y, como puede haber adivinado, el número de variables dentro de los paréntesis puede ser más de dos: A(P,Q,R) o B(X,Y,Z,XY,XZ,YX) ... o cualquier otra combinación, ¡hasta 255 variables diferentes! Los profesionales de ordenadores utilizan a menudo variables para diseñar videojuegos, o para calcular las finanzas de grandes compañías, o para otros trabajos complicados. Su ordenador Sony puede utilizar variables para muchos tipos de cosas, como aprenderá cuando continúe practicando programación con sus amigos, o utilizando otros libros. Usted ha tenido un buen comienzo, porque ya ha utilizado variables de matriz en dos programas diferentes.

¡Buena suerte, y continúe practicando!

ÍNDICE ALFABÉTICO

Sony Corporation Printed in Japan

3-795-898-41 (1)