Basic para niños

Con notas didácticas
para padres y educadores

```
10 INPUT A
20 INPUT B
30 LET C=A+B
40 PRINT C
50 GO TO 10
```

Sofía Watt
Miguel Mangada

PARANINFO SA

BASIC para niños

Sofía Watt
Miguel Mangada

BASIC
Para Niños

CUARTA EDICION

PARANINFO SA

1984
MADRID

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 84-283-1327-X

Depósito Legal: 37.144-1984.

PARANINFO S.A. Magallanes, 25 - 28015 MADRID (10-3455)

ALCO, artes gráficas. Jaspe, 34 - Madrid-26

el
El
mite

ones, el
ayuda, y
adecuada-

e tal forma que
adores que carez-

A Cristina
porque, ¡... ya está bien!

Prólogo

BASIC PARA NIÑOS muestra los conocimientos básicos del uso del ordenador y los conceptos elementales de programación en lenguaje BASIC.

Presenta a los niños un material que le facilita el descubrimiento de los secretos de un ordenador a través de su propia creatividad y experiencia, avanzando progresivamente en el estudio del BASIC.

La obra está estructurada para que el niño y el adulto descubran juntos el mundo del ordenador, siendo el padre o educador un guía, una ayuda, no un profesor. La función primaria del adulto no es impartir información, sino estimular al niño. El propio libro ofrece al niño el material adecuado que le permite encontrar la respuesta a sus preguntas.

Mediante breves aclaraciones, explicaciones o indicaciones, el padre o educador encontrará en este libro una valiosa ayuda, y una útil guía metodológica y didáctica para enfocar adecuadamente la comprensión del lenguaje BASIC.

BASIC PARA NIÑOS ha sido estructurado de tal forma que facilita su uso incluso a aquellos padres o educadores que carezcan de conocimientos previos de este lenguaje.

No obstante aquellos educadores que deseen profundizar más en el aprendizaje y la didáctica del BASIC, pueden consultar la interesante obra BASIC PARA MAESTROS, publicada en esta misma editorial.

Indice

Unas notas previas

En cada uno de los capítulos de BASIC PARA NIÑOS, se estudia un nuevo concepto. Los capítulos se han estructurado en progresión de dificultad conceptual, aunque se reiteran sistemáticamente instrucciones e informaciones ya presentadas, a fin de que se desarrolle adecuadamente la capacidad de asimilación y se mecanicen los conceptos ya vistos.

Este paulatino aumento de información favorece el aprendizaje individual, garantizando la adecuada distribución de energía entre los procesos de recepción y la actividad de tipo elaborador, de forma que niños desde los 8 años y hasta los 14 pueden aprender el lenguaje BASIC permaneciendo en cualquiera de los niveles todo el tiempo que sea necesario hasta que se esté preparado para pasar a un nuevo nivel.

El programa de aprendizaje instrumentalizado en esta obra facilita el refuerzo y afianzamiento de los conceptos fundamentales del lenguaje BASIC. Aun a costa de pecar de repetitivos, se utiliza una permanente "vuelta atrás" sobre lo ya visto, con lo que al final del libro se habrá cubierto el proceso total de aprendizaje: recepción y asimilación, retención y memorización.

Deliberadamente se han excluido determinadas instrucciones en BASIC que, por su complejidad conceptual, escapan de los objetivos de este libro.

Al final de la obra se ofrecen una serie de juegos sencillos, utilizando instrucciones en BASIC ya conocidas.

Unas notas previas

Para lograr con éxito nuestros objetivos, es preciso tener en cuenta una serie de requisitos esenciales:

- El adulto debe leer el texto previamente o con el niño. Conviene también que lea previamente las *notas para el adulto* que se encuentran a lo largo del texto de cada capítulo, antes de trabajar con el niño.

- El niño debe practicar solamente cuando lo desee. En cualquier caso, no debe extenderse en períodos muy largos de tiempo a fin de evitar agotamiento y fatiga mental.

- Es conveniente emplear el número de sesiones que sean necesarias hasta conseguir profundizar en cada concepto.
  No se debe pasar al estudio de un nuevo item si no estamos seguros de haber afianzado profundamente el anterior.

- Hay que permitir al niño que sea él quien avance, quien cree nuevos ejercicios prácticos, nuevas posibilidades. Que descubra por sí mismo.

- Despierte su curiosidad, anímele para que experimente mediante breves indicaciones, si fueran necesarias, tales como "qué pasaría si...", "intenta...". Abra camino a su iniciativa. Los resultados le servirán de estímulo.

Unas notas previas

- No estudien el mismo día más de una instrucción en BASIC. Antes de trabajar hagan un breve repaso de los conceptos fundamentales aprendidos en sesiones anteriores, para comprobar si continúan afianzados.

- Es conveniente que el niño haga con su ordenador todos y cada uno de los programas que se presentan a lo largo del libro, por sencillos que parezcan.

NUESTRO BASIC

Para trabajar con nuestros programas debe utilizar las instrucciones características de su propio ordenador personal. En el libro se hace referencia a un tipo de BASIC convencional y generalizado.

No obstante, puede ocurrir que algunos ordenadores utilicen expresiones diferentes, o distintas combinaciones de teclas para una misma función. Consulte el manual de programación de su propio ordenador cuando tenga alguna duda. Las diferencias serán mínimas. Por ejemplo:

- Algunos ordenadores hacen imprescindible la instrucción **END** al final de cada programa. En otros, ésto no es necesario.

Unas notas previas

- Algunos modelos no utilizan la instrucción **LET**. Por tanto, supriman esta instrucción cuando asignen valores a las variables. Así:

  Escriban 10 A = A + 1

  en lugar de 10 LET A = A + 1

- La función **RND** puede representarse igualmente de distintas formas: es muy frecuente el formato RND (1), RND (0), o RND (X). También se usa, simplemente, RND.

- Algunos ordenadores exigen que se incluya al comienzo del programa la función **RANDOMIZE**, a fin de obtener diferentes series de números aleatorios cada vez que se ejecuta un programa.

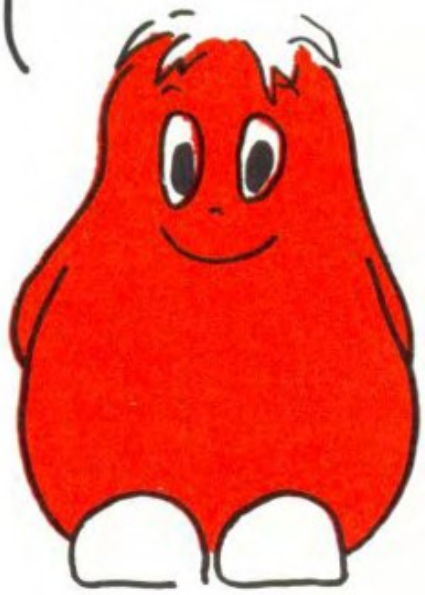
- Para interrumpir el programa durante su ejecución, pueden utilizarse distintas instrucciones, dependiendo del tipo de ordenador. Por ejemplo, BREAK o RUN STOP, etc.

- Repetimos que estas diferencias son mínimas y que la lectura del manual de instrucciones de su ordenador aclarará cualquier duda que pueda surgirle.

HOLA. ME LLAMO ARTURO.
UN ORDENADOR PUEDE HACER COSAS MUY DIVERTIDAS E INTERESANTES.

PUEDE JUGAR CONTIGO, HACER ADIVINANZAS, AYUDARTE A ESTUDIAR MATEMÁTICAS Y MUCHAS COSAS MÁS. LO ÚNICO QUE TÚ TIENES QUE APRENDER ES A HABLARLE.

ESTE LIBRO TE ENSEÑA CÓMO HACERLO. YA VERÁS COMO ES MUY FÁCIL.
ADEMÁS, SI HAY ALGO QUE NO ENTIENDES TE AYUDAREMOS ENTRE TUS PADRES, PROFESORES Y YO.
¿EMPEZAMOS?



Empezamos

Como ya te ha dicho ARTURO, un ordenador puede ayudarte en muchas cosas y puedes pasar ratos muy divertidos con él. El ordenador aprende rápido y es también muy rápido en hacer las cosas (muchísimo más rápido que tú), pero para hacerlas necesita que primero seas tú quien le explique lo que tiene que hacer. Y se lo tienes que explicar en un lenguaje que él entienda.

En realidad, el ordenador no sabe hacer casi nada que tú no le hayas enseñado antes. Este libro te explica un idioma o *lenguaje* que tu ordenador entiende, y con el que te puedes comunicar con él. Este lenguaje se llama BASIC.

Imagínate que ARTURO no sabe jugar al escondite. Tú le enseñarías más o menos así: Primero yo cuento hasta 10 y, mientras, tú te escondes. Luego te busco. Si te encuentro, he ganado yo. Si no te encuentro has ganado tú.

Al ordenador le pasa lo mismo que a ARTURO. Tienes que explicarle primero lo que quieres hacer para que él pueda hacerlo.

Pero tienes que tener en cuenta que el ordenador es muy metódico, muy "ordenado", y no le podemos hablar a lo loco. Tenemos que hablarle con mucho orden.

Empezamos

Además tienes que tener siempre en cuenta que el ordenador hará siempre las cosas en el mismo orden en que tú se las digas. Por ejemplo, si le explicaras cómo se juega al escondite, le tendrías que decir:

1. Yo cuento hasta diez
2. Tú te escondes
3. Yo te busco
4. Si te encuentro he ganado yo
5. Si no te encuentro has ganado tú.

El ordenador irá *guardando en su memoria,* es decir, recordará la primera línea, después la segunda, luego la tercera, y así hasta el final.

Todo esto lo hace en muy poco tiempo y cuando tú termines de darle las instrucciones de cómo se juega, ya sabrá jugar al escondite. El conjunto de órdenes o *instrucciones* que le has dado, se llama *programa.* Lo que hemos escrito antes sería un programa para jugar al escondite.

Como has podido ver, este programa tiene unas líneas que van numeradas y son las que va leyendo el ordenador para saber cómo se juega. Esas líneas deben ir numeradas, para que él sepa cuál tiene que leer primero, cuál después, etc. Nosotros tenemos que ordenar las líneas siempre desde el número más bajo al más alto, es decir, de menor a mayor.

Empezamos

Escribe en un papel el programa que estamos viendo y efectúa en él los cambios que vamos a ir realizando seguidamente.

Ahora suponte que quieres completar el juego con una nueva regla: Que no vale esconderse dentro de la casa. Entonces escribirías un nueva línea en ese programa que hemos visto antes:

6. Si te escondes en la casa, el juego se acaba

El programa entonces quedaría así:

1. Yo cuento hasta diez
2. Tú te escondes
3. Yo te busco
4. Si te encuentro he ganado yo
5. Si no te encuentro has ganado tú
6. Si te escondes en la casa, el juego se acaba.

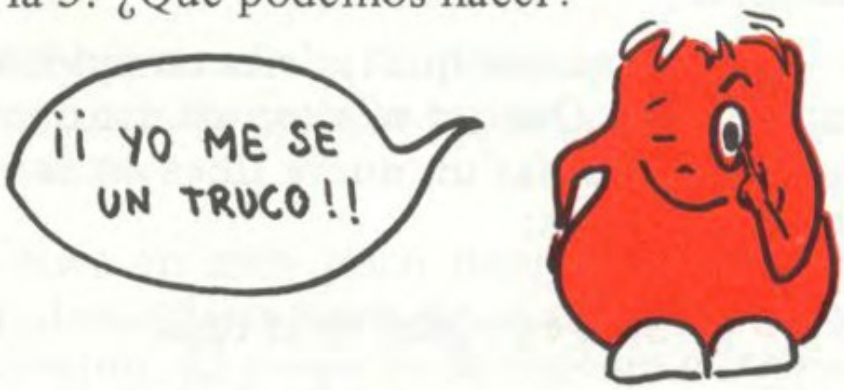
Como ves, al ordenador hay que decírselo todo.

Empezamos

Como tenemos que ser muy ordenados, lo mejor hubiera sido colocar esta regla número 6 entre la línea número 2 y la número 3. Así, el juego se parará en cuanto ARTURO haga trampa y se esconda en la casa.

De esta forma nos ahorramos mucho tiempo buscándolo. Fíjate en lo importante que es esto, y la diferencia que hay entre poner la última regla del juego al final o entre la línea 2 y la 3.

Lo que pasa es que no tenemos sitio para meter otra línea entre la 2 y la 3. ¿Qué podemos hacer?



Este truco te va a servir ya para siempre. Es muy sencillo: Numera las líneas no de una en una, sino con la numeración de 10 en 10, porque así siempre tendremos sitio para meter o intercalar, donde nos convenga, otra línea que nos sea necesaria.

Al ordenador le da igual, porque lo único que él necesita es que las líneas estén numeradas de menor a mayor, para saber en qué orden tiene que leer nuestras instrucciones.

Empezamos

De esta forma, el anterior programa quedaría así:

```
10 Yo cuento hasta diez
20 Tú te escondes
30 Yo te busco
40 Si te encuentro he ganado yo
50 Si no te encuentro has ganado tú.
```

Y la nueva línea de programa sería cualquiera entre la 20 y la 30 (por ejemplo, la número 25).

```
25 Si te escondes en la casa, el juego se acaba
```

Si estuvieras ya dando instrucciones a un ordenador, no te preocuparías por escribir esta línea después de las demás. El ordenador la coloca automáticamente en el sitio que le corresponde.

Cuando consideres que ya no tienes más órdenes que dar y que el programa ya está listo, le darás una orden especial, y el ordenador se pondrá a hacer todo lo que le has enseñado previamente. Esto se llama *ejecutar* el programa o *correr* el programa.

Más adelante veremos la forma de hacer todo esto. Pronto jugaremos al escondite con el ordenador.

Empezamos

NOTAS PARA EL ADULTO

- A través de un lenguaje coloquial y utilizando elementos cotidianos y muy fáciles de asimilar, se habitúa paulatinamente al niño a la terminología específica de la microinformática.
- Repase los conceptos del ejemplo con el niño, utilizando al principio un lenguaje familiar y posteriormente los términos marcados en cursiva en el texto.

PRINT

Como te hemos dicho antes, el ordenador necesita entender solamente unas pocas órdenes e instrucciones muy concretas. Vamos a conocer la primera orden en lenguaje BASIC. Es la orden **PRINT**.

Fundamentalmente, **PRINT** manda al ordenador imprimir todo lo que le indiques y que se encuentre entre los signos " " (comillas). El ordenador lo escribirá en la pantalla.

★ Vamos a probar a escribir el nombre de nuestro amigo ARTURO. Utilizando las teclas de tu ordenador, escribe:

```
10 PRINT "ARTURO"
```

Para meter esta línea en la memoria del ordenador, debes pulsar una tecla especial que sirve para eso. Esta tecla tiene distintos nombres según el ordenador que tengas. Puede ser ENTER, RETURN, u otra.

Consulta cuál es.

Fíjate lo importante que es. Cada vez que la aprietas, es como si le dijeras ¡Memorízalo! No olvides teclearla siempre después de escribir cada línea de programa que quieras que el ordenador memorice. Es muy importante, porque de lo contrario olvidará esta línea.

¿Y ahora qué hacemos?

PRINT

Cuando decidas que un programa ya está terminado y no quieres añadir ninguna otra orden, teclea **RUN.**

Esta es la segunda instrucción que vamos a conocer. Sirve para que el ordenador ponga en marcha el programa que tenía en su memoria.

Ya sabes que poner en marcha un programa se llama también ejecutar o correr el programa. La instrucción **RUN** la vas a utilizar siempre que termines un programa y quieras ver qué pasa.

En nuestro caso, como de momento damos por terminado el programa (aunque tenga solamente una línea) vamos a ejecutarlo. Teclea **RUN.**

Verás que en la pantalla aparece el nombre de nuestro amigo.

PRINT

NOTAS PARA EL ADULTO

- Pregunte al niño qué pasaría si hubiera olvidado poner una o las dos comillas en una sentencia **PRINT**. Que experimente los resultados (pueden variar en función del ordenador).
- No olvide recordar al niño que debe pulsar la tecla ENTER O RETURN después de cada orden o instrucción, o para introducir las líneas en el programa.
- El niño debe hacer, con su ordenador, todos y cada uno de los programas que presentamos a lo largo del libro, por sencillos que parezcan.

PROBEMOS OTRAS COSAS

★ Vamos a ampliar el programa haciendo que escriba el nombre de los amigos de ARTURO. Para ello es conveniente ver el programa original.

LIST

Para hacerlo, teclea **LIST.**

¡Ya conocemos otra orden! **LIST** sirve para que podamos ver en pantalla todas las líneas de programa que tiene el ordenador guardadas en su memoria. Es como si le preguntáramos: ¿Qué es lo que ya sabes? Entonces el ordenador *lista* o escribe en la pantalla las líneas de programa que ya teníamos.

Al listar el programa te aparecerá en la pantalla:

```
10 PRINT "ARTURO"
```

Recordarás que éste era nuestro programa inicial. Amplíalo añadiendo otras líneas con los amigos de ARTURO. Teclea:

```
20 PRINT "CARLOS"
30 PRINT "JUANITO"
40 PRINT "JORGE"
```

No te olvides de que después de escribir cada línea, tienes que meterla en la memoria del ordenador (con ENTER o RETURN). Cuando hayas terminado, teclea **RUN** y verás lo que aparece en pantalla. Tienes a toda la pandilla de ARTURO.

NEW

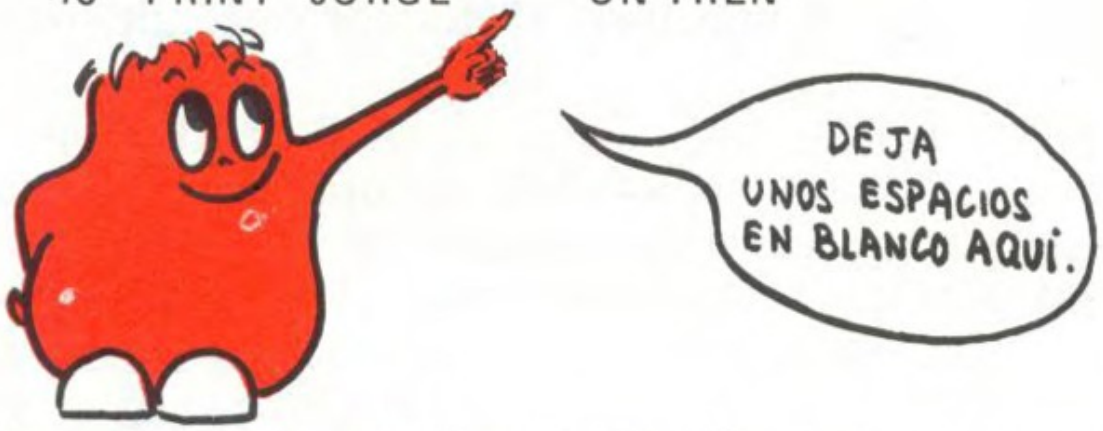
★ ¡Ya has hecho tu primer programa! Vamos a avanzar un poco más haciendo un programa más grande. Como va a ser un programa completamente nuevo, primero borraremos de la memoria del ordenador todo lo que ha aprendido hasta ahora. Para borrar, hay varios sistemas. Ya los conocerás todos. Uno de ellos (muy fácil) consiste en desenchufar el ordenador. Con ello el ordenador vuelve a estar como al principio: Tienes que enseñarle todo otra vez.

Otro sistema (más serio) es teclear **NEW**. Con **NEW** también quitas de la memoria del ordenador todo lo que tenga almacenado o guardado en ella. Ya no se acuerda de nada de lo que le enseñaste antes.

PRINT

★ Hagamos un programa con los amigos de ARTURO y los juguetes que más les gustan. Escribe:

```
10 PRINT "ARTURO    UNA BICICLETA"
20 PRINT "CARLOS    UN BALON"
30 PRINT "JUANITO   UN LIBRO"
40 PRINT "JORGE     UN TREN"
```



Ejecuta el programa (aprieta **RUN**) y verás que aparece en tu pantalla:

```
ARTURO    UNA BICICLETA
CARLOS    UN BALON
JUANITO   UN LIBRO
JORGE     UN TREN
```

★ Y ahora viene algo muy importante. Esto mismo te puede aparecer en la pantalla si cambias el programa de la siguiente manera:

PRINT

```
10 PRINT "ARTURO", "UNA BICICLETA"
20 PRINT "CARLOS", "UN BALON"
30 PRINT "JUANITO", "UN LIBRO"
40 PRINT "JORGE", "UN TREN"
```

Al correr el programa (con **RUN**...) verás que te aparece casi lo mismo de antes. Eso es porque la coma (,) la entiende el ordenador de distinta manera que nosotros. Lo que escribas después de la coma lo escribirá el ordenador más separado de lo anterior.

NOTAS PARA EL ADULTO

- Practique las instrucciones **LIST** y **RUN** hasta que se consiga la soltura necesaria.
- En una instrucción **PRINT**, todos los textos entre comillas separados de otros textos entrecomillados por una coma los imprime el ordenador en la siguiente zona de la pantalla. Esta puede estar dividida en dos o más zonas, según el tipo de ordenador. Presten atención, pues si un texto separado por la coma es demasiado largo, pasará a ocupar la siguiente zona de la pantalla.
- Muestre al niño formas menos "expeditivas" para borrar un programa. Utilicen **NEW.**
- Es imprescindible adecuar el tiempo de estudio dedicado a cada nueva instrucción en lenguaje BASIC a la aptitud y actitud del niño en cada momento.

PRINT

SIGUE PROBANDO

★ A CARLOS le han regalado una raqueta de tenis y ahora ésto es lo que más le gusta. ¿Cómo modificarías el programa?

Fijate que ahora no vas a hacer un nuevo programa, sino a modificar el que ya tienes. Tenemos que cambiar la línea en donde estaba CARLOS. Si no te acuerdas del número de línea, lista el programa y después lo que tienes que hacer es escribir de nuevo esta línea:

```
20 PRINT "CARLOS", "UNA RAQUETA"
```

Mete esta línea en el programa, ejecútalo y fíjate bien en lo que ocurre. Tendrás una pantalla como ésta:

ARTURO	UNA BICICLETA
CARLOS	UNA RAQUETA
JUANITO	UN LIBRO
JORGE	UN TREN

★ Ahora JUANITO se ha marchado de vacaciones y la pandilla no puede jugar con él. ¿Cómo lo quitarías de tu lista?

PRINT

Escribe solamente el número de línea que corresponde a JUANITO y mételo directamente en la memoria. Es como si le dijeras al ordenador que se acuerde de que esa línea ahora no tiene nada. Al correr el programa verás que JUANITO ya no está en la lista.

★ ¿Quieres formar parte de la pandilla de ARTURO? ¿Cómo incluirías tu nombre en la lista?

Tendrás que añadir una nueva línea de programa, de la misma manera que las veces anteriores.

★ Vamos a probar otra cosa. Teclea:

```
15 PRINT
```

Mételo en la memoria y ejecútalo. ¿Qué ha pasado? Prueba ahora lo que ocurre si añades al programa la línea:

```
25 PRINT
```

Corre el programa y tendrás una pantalla como ésta:

ARTURO	UNA BICICLETA
CARLOS	UNA RAQUETA
JORGE	UN TREN

PRINT

NOTAS PARA EL ADULTO

- Recuerde al niño que debe listar el programa cada vez que modifique el mismo, para cerciorarse de que lo ha hecho correctamente.
- Para añadir más líneas, utilice números de línea mayores que el de la última línea del programa.
- Para que aparezca una línea en blanco, basta con numerar esa línea, seguida de **PRINT.**
- Haga ver al niño que debe prestar atención para no numerar nuevas líneas con los números de las ya existentes, porque las borraríamos.
- Ayúdele a familiarizarse con las características de su propio teclado, borrando alguna línea o letra erróneamente escrita, utilizando el espaciador, etc.

PRINT

MAS SOBRE PRINT

Con el ordenador también puedes hacer dibujos. Una de las maneras es utilizar la instrucción **PRINT**.

Prueba a hacer un rectángulo con el siguiente programa: Emplea la tecla de los espacios en cuanto lo necesites.

```
10  PRINT "X X X X X X X X"
20  PRINT "X               X"
30  PRINT "X               X"
40  PRINT "X               X"
50  PRINT "X               X"
60  PRINT "X X X X X X X X"
```

Corre el programa y verás lo que pasa.

SIGUE PROBANDO

¿Cómo harías un rectángulo más pequeño?

Borra, por ejemplo, las líneas 40 y 50. Si no recuerdas como se hace, acuerdate de lo que hiciste para quitar a JUANITO de la pandilla de ARTURO. Corre el programa.

PRINT

Si lo has hecho bien, te aparecerá un rectángulo como éste:

```
XXXXXXXX
X      X
X      X
XXXXXXXX
```

Puedes hacer dibujos más completos, combinando distintos símbolos. Prueba a hacer algo parecido a éste:

```
10 PRINT "000$000"
20 PRINT "00$$$00"
30 PRINT "0$$$$$0"
40 PRINT "$$$$$$$"
50 PRINT "0$$$$$0"
60 PRINT "00$$$00"
70 PRINT "000$000"
```

Intenta hacer dibujos más complicados. Prueba con este barquito de vela. Para que se vea mejor, en lugar de rodear el dibujo con un símbolo (como en el caso del rombo, que estaba rodeado por ceros) cuenta los espacios correspondientes al lado izquierdo de la figura y dejalos en blanco.

PRINT

Para ello puedes utilizar la tecla del espaciado. Teclea:

```
 10  PRINT "          X "
 20  PRINT "          XX "
 30  PRINT "          XXX "
 40  PRINT "          XXXX "
 50  PRINT "          XXXXX "
 60  PRINT "          XXXXXX "
 70  PRINT "          XXXXXXX "
 80  PRINT "          XXXXXXXX "
 90  PRINT "          XXXXXXXXX "
100  PRINT "          XXXXXXXXXX "
110  PRINT "          XXXXXXXXXXX "
120  PRINT "          XXXXXXXXXXXX "
130  PRINT "          X "
140  PRINT "          X "
150  PRINT "  XXXXXXXXXXXXXXXXXXXXXXX "
160  PRINT "   XXXXXXXXXXXXXXXXXXXXX "
170  PRINT "    XXXXXXXXXXXXXXXXXXX "
180  PRINT "     XXXXXXXXXXXXXXXXX "
190  END
```

PRINT

NOTAS PARA EL ADULTO

- Para borrar una línea, basta con teclear el número de la línea y pulsar ENTER o RETURN.
- Proponga al niño nuevos dibujos esquemáticos. Sugiérale triángulos, círculos, etc., pero incentive su creatividad animándole a realizar sus propios diseños.
- No olvide borrar la pantalla tecleando **NEW** o la tecla correspondiente antes de comenzar un nuevo dibujo. Esta instrucción borra toda la información contenida en el ordenador. Este comentario es válido siempre que se vaya a comenzar un nuevo programa.
- Conviene que el niño utilice papel milimetrado o cuadriculado para diseñar previamente sus dibujos.

LET

Vamos a conocer ahora la instrucción **LET**. Es muy útil. Para que la entiendas mejor, veámosla con un ejemplo. Fíjate bien en este pequeño programa:

```
10  LET A=14
20  PRINT A
```

★ Corre el programa. Verás que en la pantalla aparece escrito el número 14. ¿Qué ha pasado? La segunda línea del programa ya la entiendes perfectamente. Le has mandado al ordenador que imprima A.

Pero en lugar de escribir A, el ordenador escribe 14. Eso ocurre porque en la primera línea tú le has dicho al ordenador que A vale 14. Eso lo has hecho con **LET**. LET A = 14 es como decir HAZ A = 14.

Fíjate que ahora ya no escribes **PRINT** y luego la letra A entre comillas (" "). Eso es porque no quieres que el ordenador te imprima la letra A, sino el valor que tú le has dado a esa letra (en este caso, 14).

LET

★ Vamos a probar otra vez. Escribe el programa:

```
10  LET A=14+1
20  PRINT A
```

Corre el programa y en la pantalla te aparecerá el número 15. Como puedes ver, la instrucción **LET** te permite dar a la letra A el valor que tú quieras. El ordenador guarda en su memoria este valor que tú le has dado. En lugar de la letra A puedes utilizar cualquier otra, o incluso grupos de letras (como AB, HZ, etc.) o combinaciones de letras y números, siempre que empiecen por una letra (como A2, P3, etc.)

Como a todas ellas les puedes dar el valor que tú quieras, se llaman *variables*, porque pueden variar o cambiar su valor.

★ Puedes variar su valor todas las veces que quieras. Por ejemplo:

```
10  LET X=6
20  LET X=8
30  PRINT X
```

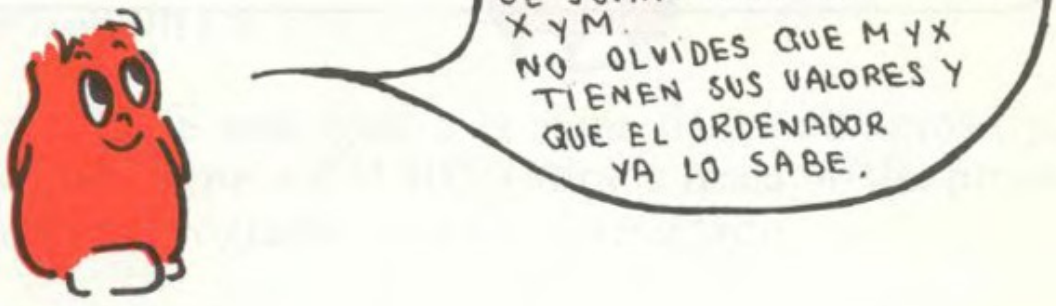
LET

Al correr el programa, el ordenador escribe 8, porque en la segunda línea le has dado a la variable X el valor 8, con lo que se anula el valor 6 que le dabas en la primera línea.

★ Fíjate en lo útil que es la instrucción **LET** para hacer cálculos.

Escribe el programa:

```
10  LET M=4
20  LET X=6
30  LET S=M+X
40  PRINT S
```



Cuando corras el programa, el ordenador escribe 10. En las dos primeras líneas has dado a M el valor 4 y a X el valor 6.

En la línea 30 le damos a la variable S el valor de la suma de M + X (que es 10). En la línea 40 ordenamos imprimir ese valor.

LET

NOTAS PARA EL ADULTO

- La instrucción **LET** asigna a una variable situada a la izquierda del signo = el valor de la expresión situada a la derecha de dicho signo.
  Las variables mencionadas en el texto son variables numéricas.

- En algunos ordenadores, no se usa **LET** para asignar valores a las variables. Directamente se escribiría, por ejemplo, 10 X=6 en lugar de 10 LET X=6.

- Haga ver al niño que PRINT A (o cualquier otra variable) hará aparecer en pantalla el valor asignado a esa variable. Comente con el niño la diferencia al escribir PRINT "A".

LET

SIGUE PROBANDO

★ Un programa muy sencillo sería el que nos dice el número total de libros que tiene ARTURO. Tiene 3 del CAPITAN TRUENO, 6 de LOS CINCO y 2 de LOS PITUFOS. Para hacerlo más simple, a los libros del CAPITAN TRUENO les asignamos la variable T. A los de LOS CINCO, la variable C y a los de los PITUFOS, la variable P. El programa sería:

```
10  LET T=3
20  LET C=6
30  LET P=2
40  LET S=T+C+P
50  PRINT S
```

La variable S será igual a la suma de los distintos tipos de libros que tiene ARTURO. (Mira la línea 40 del programa). Corre el programa para ver el resultado.

★ Si le regalan un nuevo libro de los PITUFOS ¿cómo tendrías que cambiar el programa? Cambia la línea 30 por:

```
30  LET P=3
```

Como ves, con **LET** puedes darle distintos valores a cada variable.

LET

★ La mamá de ARTURO le ha dicho que vaya a la tienda a comprar una bolsa de caramelos, que vale 10 pesetas; y dos Pasteles, que valen 25 pesetas cada uno. ¿Cuánto dinero necesitará?

El programa sería:

```
10  LET C=10
20  LET P=25
30  LET X=P*2
40  LET S=C+X
50  PRINT "ARTURO NECESITA ";
60  PRINT S;
70  PRINT "PESETAS"
```

Repasemos con qué variables estamos trabajando y qué valores tiene cada una de ellas:

C: Es la variable que representa los caramelos. Y como sabemos que los caramelos valen 10 pesetas, diremos que C = 10. Por eso escribimos: C = 10.

P: Es la variable que representa a un pastel. Como sabemos que un pastel vale 25 pesetas, diremos por tanto, que P = 25. Por eso escribimos LET P = 25.

LET

X: Es la variable que representa los dos pasteles. Como sabemos que un pastel es P, los dos pasteles serán 2 veces P. Por eso escribimos LET X = P * 2.

S: Es la variable que representa el dinero total que necesita ARTURO. O sea, la suma de lo que cuestan los caramelos más los dos pasteles. Por eso escribimos LET S = C + X.

Corre el programa y verás lo que aparece en pantalla.

★ ¡Una cosa muy importante! Fíjate lo que ha ocurrido cuando ponemos el signo ;(punto y coma) en una instrucción **PRINT**. ¿Qué ha pasado con las siguientes instrucciones **PRINT**? ¿Qué pasaría si quitaras el signo; (punto y coma)?

LET

NOTAS PARA EL ADULTO

- Comente con el niño la línea 30 y su repercusión en el programa.
- El punto y coma (;) en una instrucción **PRINT** hace que la siguiente sentencia **PRINT** se escriba inmediatamente a continuación de la primera. Hágaselo ver al niño.
- Donde corresponda, deberán dejar un espacio en blanco al lado de las comillas, para que las sucesivas sentencias **PRINT** ligadas entre sí a través del punto y coma no se escriban demasiado juntas. Sobre este punto se insiste al niño en próximos capítulos.
- Sugiera al niño que haga otro tipo de cálculos sobre temas relacionados con su entorno familiar. Utilicen sumas, restas, multiplicaciones y divisiones elementales.
- En BASIC, la multiplicación se representa con el signo *. La división, por el signo /.

LET

SIGUE PROBANDO

★ Hasta ahora hemos visto como con la instrucción **LET** le dabas a unas variables el valor que tú querías. Ahora veremos como puedes dar o asignar a una variable incluso palabras o textos.

Sólo tienes que poner detrás de la variable el signo $ y no olvidarte de poner las palabras o textos entre comillas (" ").

Teclea el programa:

```
10  LET A$="ARTURO"
20  LET J$="JORGE"
30  LET C$="CARLOS"
```

El ordenador ya tiene en su memoria los nombres que le has dado a cada variable. Compruébalo añadiendo:

```
40  PRINT A$;
50  PRINT " ES AMIGO DE ";
60  PRINT C$
```

LET

Corre el programa. Como ves, el ordenador tiene muy buena memoria. Ya has visto como escribe el ordenador cuando en las sentencias **PRINT** pones un punto y coma (;).

De todas maneras, también puedes obtener el mismo resultado si borras las líneas 40 y 60 y escribes la línea 50 así:


```
50  PRINT A$;" ES AMIGO DE ";C$
```

Corre el programa. Para que te quede bonito, pon mucha atención en la línea 50 (en cualquiera de los dos programas). Debes dejar un espacio en blanco entre las primeras comillas y la palabra ES y también entre la palabra DE y las segundas comillas. Si no lo haces, el ordenador escribirá todo junto.

★ Como ves, resulta muy interesante hacer que todo un grupo de palabras o textos quede asignado a una variable.

LET

Además, puedes trabajar con estas variables casi igual que con las otras. Recordarás por los capítulos anteriores que podías operar con las variables: Las podías sumar, restar, multiplicar, dividir...

Estas variables que acabamos de ver ahora solamente se pueden sumar. No tiene sentido que digas "ARTURO" multiplicado por "CARLOS". Sin embargo, sí puedes decir "ARTURO" y "CARLOS".

Vamos a ver un programa que nos sirva de ejemplo.

```
10  LET A$="ARTURO "
20  LET B$="ES "
30  LET C$="NO ES "
40  LET D$="ALTO"
50  LET E$="LISTO"
60  LET F$="FEO"
70  LET G$="BUENO"
80  PRINT A$+B$+D$
```

Corre el programa. Después, cambia la línea 80 poniendo distintas variables. Por ejemplo, PRINT A$ + C$ + F$. O también, PRINT A$ + B$ + G$.

Como ves, se pueden sumar estas variables y, por tanto, los grupos de textos que les corresponden.

LET

NOTAS PARA EL ADULTO

- Estas variables se denominan variables alfanuméricas, pues a ellas se asignan cadenas de caracteres numéricos, alfabéticos o alfanuméricos. Unicamente pueden ser sumadas y entre sí. No puede sumarse una variable numérica y una alfanumérica.

- En el texto anterior se ha utilizado la expresión "textos" o "grupos de palabras" al hacer referencia a las cadenas de caracteres. Utilice inicialmente estas expresiones, para sustituirlas paulatinamente por una terminología más ortodoxa, denominando preferentemente a éstas "cadenas de caracteres".

- Remarque nuevamente la atención del niño sobre la función de las comillas, y la colocación de los espacios en blanco dentro de los textos incluidos en una sentencia PRINT.

INPUT

Esta instrucción es una de las más bonitas. Con ella puedes "hablar" con el ordenador, que te pregunta y te responde.

INPUT te permite introducir datos con tu teclado mientras se está ejecutando o corriendo un programa. Fíjate en el ejemplo:

```
10  LET A=2
15  PRINT " ESTOY ESPERANDO UN NUMERO"
20  INPUT M
30  LET B = A * M
40  PRINT B
```

★ Corre el programa. El ordenador empieza a ejecutarlo. Al leer la primera línea, asigna el valor 2 a la variable A. Entonces llega a la instrucción **INPUT** y lo que hace es pararse esperando un número que tú le tienes que dar a través del teclado.

INPUT

Este dato es el valor que tú le vas a dar a la variable M. Es decir, el ordenador se dice a sí mismo: "Estoy esperando un dato, y el valor de ese dato se lo voy a asignar a la variable M". Esto es lo que hace **INPUT**.

Dale a M el valor que tú quieras. Por ejemplo, 5. Para ello, pulsa 5 (y luego ENTER o RETURN). Entonces el ordenador sigue con el programa y como ya todas las variables tienen su correspondiente valor, hace las operaciones que le has indicado en la línea 30 e imprime el resultado.

En nuestro ejemplo toma A que vale 2 y M al que le has dado el valor de 5 y los multiplica imprimiendo el resultado: 10.

★ Prueba dando distintos valores a M y luego cambiando el valor de A.

NOTAS PARA EL ADULTO

- La sentencia **INPUT** permite la interacción entre el ordenador y el usuario. Posibilita la entrada de datos en el transcurso de la ejecución de un programa. Combinando instrucciones **PRINT** e **INPUT** podremos conseguir una relación "conversacional" entre el niño y el ordenador, como veremos más adelante.

INPUT

SIGUE PROBANDO

★ Vamos a hacer un programa para que puedas hablar con tu ordenador. Escribe:

```
10  PRINT "ESCRIBE TU NOMBRE"
20  INPUT N$
30  PRINT "¿EN QUE AÑO HAS NACIDO "; N$; " ?"
40  INPUT A
50  PRINT "¿EN QUE AÑO VIVIMOS?"
60  INPUT B
70  LET X = B - A
80  PRINT
90  PRINT
100 PRINT
110 PRINT
120 PRINT
130 PRINT"BUENO "; N$; " TIENES UNOS "; X;
    " AÑOS"
```

Corre el programa y contesta las preguntas. ¡El ordenador puede hablar contigo!

INPUT

Vamos a analizar un poco el programa. En las líneas de programa 10, 30 y 50 estás enseñando al ordenador a preguntarte. En la línea 20 verás que ponemos el signo $ detrás de la variable. Eso es porque a esta variable no le vas a asignar un valor numérico sino una palabra. Si no te acuerdas muy bien de ésto, vuelve a leerte la página 45.

En las líneas 40 y 60 no ponemos el signo $ porque los valores de A y B van a ser numéricos. Puedes ver en el programa que A es el año en que has nacido y B el año en que vives.

Con las líneas 80 a 120 hemos dejado líneas sin escribir para que la respuesta quede más separada de las preguntas que has hecho antes.

Vamos a hacer una modificación muy interesante. Hagamos que nuestras respuestas queden también escritas en la pantalla. Añade las líneas:

```
25  PRINT N$
45  PRINT A
65  PRINT B
```

Corre de nuevo el programa y observa las diferencias.

INPUT

★ ARTURO está preparando sus vacaciones y quiere saber cuantos días va a pasar en su casa. Como ya sabe cuantos días va a estar en la playa, haría el siguiente programa:

```
10 PRINT "¿CUANTOS DIAS DE VACACIONES
   TENGO?"
20 INPUT T
30 PRINT "¿CUANTOS DIAS ESTARE EN LA
   PLAYA?"
40 INPUT P
50 LET D=T-P
60 PRINT "ME QUEDAN ";D;" DIAS DE VACA-
   CIONES EN CASA"
```

Ejecuta el programa. El ordenador te irá preguntando. Contesta tú por ARTURO. El ordenador te dará la respuesta final.

Al igual que hiciste en el programa anterior, modifica éste, de forma tal que aparezcan tus respuestas en la pantalla.

INPUT

★ A ARTURO le gusta mucho la playa y quiere saber cuántas horas va a poder disfrutar de la arena y del mar. Para hacer el cálculo, utiliza el siguiente programa. Tú ya sabes que un día tiene 24 horas.

```
10 PRINT "¿CUANTOS DIAS PASARA ARTURO EN
   LA PLAYA?"
20 INPUT P
30 LET H = 24
40 LET S = P * H
50 PRINT "ARTURO PASARA "; S ; " HORAS EN LA
   PLAYA"
```

Corre el programa y tendrás la respuesta. Modifícalo para que aparezca tu respuesta en pantalla.

Sin embargo, ARTURO no se da cuenta de que no se puede pasar todas las horas del día en la playa. Tiene que dedicar algunas horas a descansar. Mediante la siguiente ampliación del programa ayúdale tú a ver cuántas horas le quedan de verdad para jugar.

Contesta tú mismo a las preguntas que te haga el ordenador cuando se esté ejecutando el programa completo.

INPUT

Añade primero al programa anterior:


```
 55  PRINT
 56  PRINT
 57  PRINT
 58  PRINT
 60  PRINT "DIME LAS HORAS QUE
     DESCANSA ARTURO CADA DIA"
 70  INPUT X
 75  PRINT X
 80  LET Z = 24 - X
 90  LET Q = Z * P
100  PRINT "ARTURO JUGARA EN LA PLAYA  ";
     Q ;" HORAS DURANTE SUS VACACIONES"
```

Si además quieres ayudarle a que sepa cuántas horas jugará cada día, cambia la línea 100 ampliándola así:

```
100  PRINT "ARTURO JUGARA EN LA PLAYA  "; Q;
     "  HORAS DURANTE SUS VACACIONES, O SEA ";
     Z;" HORAS CADA DIA"
```

INPUT

NOTAS PARA EL ADULTO

- Trabaje insistentemente con este programa. Sugiera al niño la utilización de **LIST** cada vez que sea conveniente. Sean cuidadosos en la colocación de los espacios en blanco entre las sentencias **PRINT**, a fin de que las frases queden correctamente escritas.

- Reitere la importancia del punto y coma. Signifíquele al niño la diferencia de utilizar la coma a modo de instrucción, para escribir en la siguiente zona de la pantalla o de incluirla dentro de una instrucción **PRINT**, como en la línea 100. Practique algunos ejemplos de su invención situando la coma dentro y fuera de la parte entrecomillada de una sentencia **PRINT.**

- Repase varias veces el programa, analizando con el niño los distintos valores asignados paulatinamente a las diferentes variables a través de la sentencia **LET.**

- Hágale ver la forma y los números de línea en los que se han efectuado añadidos y modificaciones.

- Sobre esta base, motive al niño a inventar nuevas situaciones y desarrollen conjuntamente los programas correspondientes.

GOTO

Vamos a hacer nuestros programas más completos, utilizando **GO TO**.

Con esta instrucción le das una orden tajante al ordenador. Significa VETE A, y el ordenador tiene que ir a la línea de programa donde tú le mandes.

Esta instrucción te alterará el orden normal de ejecución del programa.

★ Por ejemplo, en este programa:


```
10  PRINT "HOLA"
20  GO TO 10
```

El ordenador lee la primera línea y escribe HOLA. Luego lee la segunda línea, que le ordena ir a la línea 10. Entonces imprime otra vez HOLA. Luego lee la segunda línea, que le ordena ir a la línea 10. Entonces imprime otra vez HOLA y así sucesivamente. ¡Este programa no termina nunca! Córrelo y verás qué pasa.

GOTO

En tu pantalla hay una columna de palabras HOLA. Aunque ya no tenga sitio para seguir escribiendo, el ordenador sigue por dentro ejecutando el programa. Para que pueda pararse, tienes que teclear BREAK, que es una instrucción que interrumpe los programas mientras se están ejecutando. (Tal vez tú ordenador utiliza otra instrucción en lugar de BREAK. Consulta cuál es.)

Cambia la línea 10 del programa por:

```
10 PRINT "HOLA",
```

y comprobarás algo que ya conoces sobre la importancia de la coma.

Lista el programa (no olvides pulsar antes BREAK).

Cambia la línea 10 otra vez, poniendo un punto y coma (;) en lugar de la coma. Recordarás así la influencia del punto y coma.

No te olvides que no es lo mismo poner una coma dentro de las comillas (" ") o fuera de ellas. Prueba poniendo la línea 10 del programa así:

```
10 PRINT "HOLA,"
```

Volviendo a la instrucción **GO TO**, vamos a ver algunos programas donde veremos la utilidad que tiene.

GOTO

★ Por ejemplo, ARTURO quiere contar números de dos en dos. Podríamos escribir el programa:

```
10  LET A=0
20  PRINT A
30  LET A=A+2
40  PRINT A
50  LET A=A+2
60  PRINT A
```

Este programa sólo cuenta hasta 4. Si queremos que siga contando, el programa será larguísimo...

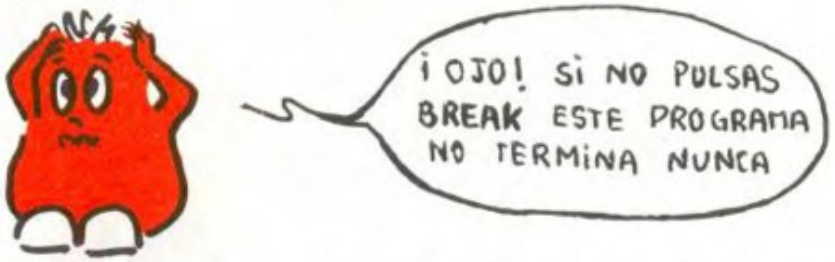
Vamos a utilizar **GO TO**. Escribe el siguiente programa:

```
10  LET A=0
20  LET A=A+2
30  PRINT A
40  GO TO 20
```

GOTO

Al correr el programa te aparecerá en la pantalla lo mismo que la vez anterior:

2
4
6



Fíjate bien en el programa. En la línea 10, le has dado un valor a la variable A. En este caso, el valor 0.

En la línea 20, le das un nuevo calor a A: Su valor inicial más las unidades en que quieres aumentar dicho valor (en nuestro ejemplo, dos unidades). Es decir, A vale ahora 2.

En la línea 30, imprimes el valor que tiene A en ese momento (es decir, 2).

En la línea 40 le ordenas que vaya a la línea 20 y entonces el ordenador toma el último valor de A (que era en este momento 2) y le suma 2 unidades. Entonces A vale ahora 4.

A continuación imprimirá este valor y, obligado otra vez por la instrucción **GO TO** de la línea 40, irá de nuevo a la línea 20, donde tomará el último valor de A (que ahora es 4) y le suma dos unidades. Imprime el nuevo valor (que ya es 6) y así sucesivamente.

Conviene que sigas estas explicaciones con el programa a la vista.

GOTO

★ Haz tú mismo un programa para contar de 5 en 5. Para ello, cambia la línea 20. Si no lo sabes, fíjate en lo que dice ARTURO.

NOTAS PARA EL ADULTO

- La instrucción **GO TO** implica un salto o transferencia incondicional. Altera el orden secuencial de las líneas de programa. Estos saltos pueden ser hacia líneas de mayor o menor número que aquella en la que se encuentra la instrucción **GO TO.**

- Haga ver al niño que en los programas vistos en este capítulo, cada vez que a través de la instrucción **GO TO** el ordenador vuelve a una línea de programa anterior, ya tiene en memoria los últimos valores que ha adoptado la variable.

- Fíjese que en matemáticas la expresión A = A + 2 no tiene sentido. En informática sí, ya que primero se calcula el valor situado a la derecha del signo = y después se asigna ese valor a la variable que está a la izquierda de dicho signo.

GOTO

SEGUIMOS PROBANDO

★ ARTURO está aprendiendo a sumar y quiere hacerse una tabla de sumas. Vamos a ayudarle haciendo un programa que le sirva para, por ejemplo, obtener y aprender a sumar añadiendo 3 unidades a cualquier número natural. Teclea:

```
10  LET X=0
20  LET S=X+3
30  PRINT X;" +3= ";S
40  LET X=X+1
50  GO TO 20
```

Corre el programa. Te aparecerá una tabla como ésta:

```
0 +3 =3
1 +3 =4
2 +3 =5
3 +3 =6
4 +3 =7
```

★ Ahora vamos a hacer una tabla que ayude a ARTURO a multiplicar por 3. Cambia las líneas 20 y 30 del programa. Intenta hacerlo tú solo. Piensa que ahora vas a multiplicar en lugar de sumar. Si no te sale, no te preocupes. Estamos empezando.

GOTO

Mira la forma de hacerlo:

```
20  LET S=3 * X
30  PRINT " 3 *  ";X;" = ";S
```

Cambia estas líneas en el programa y al correrlo tendrás una pantalla como ésta:

```
3 * 0=0
3 * 1=3
3 * 2=6
3 * 3=9
3 * 4=12
3 * 5=15
```

★ La instrucción **GO TO** no sirve únicamente para volver a líneas de programa que ya habían sido ejecutadas, sino también para saltarnos líneas de programa e ir directamente a líneas posteriores a aquélla en la que está la instrucción **GO TO**.

O sea, que de la línea 40 podemos ir directamente –por ejemplo– a la línea 100.

Esto lo veremos mucho mejor más adelante. Por ahora, puedes entrenarte un poco con el programa "loco" de la página siguiente.

GOTO

Normalmente en un programa tú no harás una ordenación de líneas de programa tan absurda como ésta, pero te servirá para practicar y ver la velocidad con la que trabaja el ordenador.

Escribe:

```
10  GO TO 40
20  PRINT "QUIEN ANDAS ";
30  GO TO 80
40  PRINT "DIME CON ";
50  GO TO 20
60  PRINT "QUIEN ERES"
70  END
80  PRINT " Y TE DIRE ";
90  GO TO 60
```

Fíjate en la línea 70. Sirve para que el ordenador se pare cuendo llegue allí. De lo contrario, el ordenador repetiría indefinidamente una serie de palabras (prueba a quitar la línea 70 y verás).

La instrucción END tiene, pues, su importancia. Algunos ordenadores utilizan STOP en lugar de END, pero su función es casi la misma.

¡Ah! En este tipo de programas tienes que poner mucha atención al colocar los espacios entre las comillas, para que te quede el texto bien escrito.

IF

La instrucción **GO TO** que acabamos de ver resulta todavía mucho más útil una vez que conozcas la instrucción **IF**. Esta instrucción permite que tu ordenador piense un poco por sí solo y tome sus decisiones.

Lógicamente, siempre lo hará sobre las órdenes o informaciones que tú le des.

★ Veamos cómo funciona esta instrucción en un programa:

```
10  PRINT "¿CUANTAS SON DOS MAS DOS?"
20  INPUT A
30  IF A =4 THEN 60
40  PRINT "TE HAS EQUIVOCADO"
50  GO TO 10
60  PRINT "HAS ACERTADO!!!"
70  END
```

IF

Este programa funciona así: En la línea 10 empieza la conversación contigo. En la línea 20 espera un dato que tú mismo le das a través del teclado.

Ahora viene la línea más interesante. En la línea 30, el ordenador se encuentra con una condición: Si el valor que le das a A es igual a 4, entonces saltará obligatoriamente a la línea 60, sin tener en cuenta las líneas 40 y 50. Y te dirá que has acertado.

Pero si A no es igual a 4, entonces el programa seguirá su curso normal. Hemos hecho que el ordenador tome una decisión siguiendo unas instrucciones que le hemos dado anteriormente.

Como ves, si A no es igual a 4, al llegar a la línea 40 te dice que te has equivocado y en la línea 50 vuelve a la línea 10 para preguntarte de nuevo.

Es decir, que al ordenador le pones unas condiciones. Se las pones con la instrucción **IF**. Si la condición se cumple, el ordenador hará una cosa (que tú le habrás indicado previamente). Si la condición no se cumple, hará otra (también indicada por tí).

En el ejemplo anterior, le poníamos la condición de que A fuera *igual* (=) a 4. Las condiciones que le quieras poner al ordenador pueden ser otras, tales como *menor que* (que se representa con el signo <), *mayor que* (que se representa con el signo >), *distinto de* (que se representa con el signo <>) y otras.

IF

★ El programa anterior podías haberlo escrito también de la siguiente manera:

```
10  PRINT " ¿CUANTAS SON DOS MAS DOS?"
20  INPUT A
30  IF A <> 4 THEN 60
40  PRINT "HAS ACERTADO!!!"
50  GO TO 80
60  PRINT "TE EQUIVOCASTE"
70  GO TO 10
80  END
```

Fíjate como ha sido necesario volver a organizar todas las líneas del programa al cambiar la condición. Pero el resultado es el mismo.

La condición que has puesto era: Si A es *distinto* de 4, vete a la línea 60.

Analiza bien cómo funciona cada uno de los programas.

IF

★ También puedes utilizar cadenas de caracteres (es decir, textos).

Vamos a escribir un programa casi igual que el primero, pero que nos pregunte como se llama ARTURO.

```
10  PRINT "¿COMO SE LLAMA NUESTRO AMIGO?"
20  INPUT A$
25  PRINT A$
30  IF A$="ARTURO" THEN 60
40  PRINT "TE HAS EQUIVOCADO"
50  GO TO 10
60  PRINT "HAS ACERTADO!!!"
70  END
```

Tienes que poner mucho cuidado al contestar al ordenador. Si no le contestas con los textos que él ya conoce, no entenderá tu respuesta. Vamos a poner a prueba al ordenador. Corre otra vez este programa y contéstale "ATURO", en lugar de "ARTURO". El ordenador desconoce la palabra ATURO, y la entenderá como un nombre distinto. Te dirá que te has equivocado, puesto que al no ser A$ exactamente igual a "ARTURO", el ordenador no salta a la línea 60 y sigue su lectura en la línea 40.

IF

★ Vamos a ver otro ejemplo, para que te acostumbres a usar esta instrucción.

El programa te parecerá un poco largo, pero es muy sencillo, porque ya conocemos todas las instrucciones que se usan en él.

Con este programa le resolvemos un problema a nuestro amigo ARTURO. El quiere comprar unos balones en una tienda. Solamente tiene 200 pesetas.

Hay dos tipos de balones. Rojos y azules. Los rojos valen 100 pesetas y los azules 90 pesetas. ARTURO tiene que decidirse entre comprar un balón rojo, uno azul o los dos. Y quiere saber cuánto dinero le va a sobrar en cada caso.

Hagamos el siguiente programa:

```
10  LET D=200
20  LET BR=100
30  LET BA=90
```

Con estas líneas llamamos D al dinero que tiene ARTURO y BR y BA a los precios de los balones rojos y azules, respectivamente.

IF

Sigue:

```
40  PRINT "DI CUAL COMPRAS: EL ROJO,
    EL AZUL O LOS DOS"
50  INPUT A$
55  PRINT A$
```

Con estas líneas le enseñamos al ordenador a preguntarnos y le indicamos que espere nuestra respuesta y la imprima. Sigue escribiendo:


```
60  IF A$ = "EL ROJO" THEN 90
70  IF A$ = "EL AZUL" THEN 110
80  IF A$ = "LOS DOS" THEN 130
85  PRINT "NO ENTIENDO. CONTESTA
    CORRECTAMENTE A MI PREGUNTA"
87  GO TO 40
```

Con estas líneas le indicamos al ordenador que actúe según sea nuestra respuesta. Como ves, según el balón elegido, al ordenador le mandas a una u otra línea del programa, para hacer todos los cálculos únicamente en base a ese balón y no a otro.

IF

¿Qué pasa si tu respuesta no es igual a las previstas en las líneas 60, 70 y 80? Por ejemplo, si pones "verde" o "rojo" o incluso "azul" (en lugar de "el azul"), el ordenador no salta a ninguna línea y sigue leyendo el programa. Leerá las líneas 85 y 87 y te volverá a preguntar.

Sigue escribiendo:

```
 90  LET X=D-BR
100  PRINT "TIENES QUE PAGAR ";BR;
     " PESETAS Y TE SOBRAN ";X;" PESETAS"
105  GO TO 150
```

El ordenador se dirige a la línea 90 solamente cuando tú has seleccionado el balón rojo. (Recuerda la línea 60.) Cuando llega a la línea 90, calcula X (que es el dinero que le sobra a ARTURO).

Para ello, el ordenador resta el precio del balón (BR) del dinero que llevaba (D). Acuérdate que estos valores los tenía el ordenador memorizados en las líneas 10 y 20.

IF

Seguimos:


```
110 LET X = D - BA
120 PRINT "TIENES QUE PAGAR "; BA;
    " PESETAS Y TE DEVOLVERAN ";
    X;" PESETAS"
125 GO TO 150
```

Como ves, el ordenador sólo llega a la línea 110 si eliges el balón azul (recuerda la línea 70). El ordenador hace todos los cálculos de igual forma que para el caso anterior, pero con los datos del precio del balón azul (BA).

No hará falta explicarte que, en el caso de que quieras los dos balones, el ordenador actuará de la misma forma, pero a partir de la línea 130.

¿Serás capaz de añadir tú las líneas 130 y siguientes?

Se pueden escribir así:

```
130 LET X = D - BR - BA
140 PRINT "TIENES QUE PAGAR "; BR + BA;
    " PESETAS Y TE DEVOLVERAN ";X;
    " PESETAS"
150 END
```

Corre el programa.

IF

★ Fijate en un detalle curioso de la línea 140. La cantidad total a pagar por los dos balones la hemos llamado BR + BA. El ordenador ha hecho el cálculo automáticamente.

Pero lo más correcto hubiera sido que este valor tuviera su correspondiente variable. Por ejemplo, podríamos haber dicho:

```
135 LET Z=BR+BA
```

Y en la línea 140 hubieramos escrito:

```
140 PRINT "TIENES QUE PAGAR ";Z;" PESETAS
    Y TE DEVOLVERAN ";X;" PESETAS"
```

Hazlo de las dos formas y verás como el resultado es el mismo.

También habrás visto que en este programa utilizamos variables con dos letras para que nos sea más fácil identificarlas.

IF

NOTAS PARA EL ADULTO

- La instrucción **IF** permite establecer determinadas condiciones dentro del programa. Generalmente se utiliza para provocar un salto condicional dentro del mismo. Adopta entonces la forma **IF ... THEN** o **IF ... THEN GO TO** (según el tipo de ordenador). Es decir, el salto se produce únicamente si la condición puesta por **IF** se cumple. En caso contrario, se sigue ejecutando la línea siguiente a la de la instrucción **IF**.
- Esta instrucción es muy importante dentro de la programación, pero requiere una cierta madurez y capacidad de análisis por parte del niño.
  Si el niño lo encuentra confuso, limítese de momento a comentarle qué hace esta instrucción dentro de un programa, pero sin profundizar en el tema.
- Los ejemplos tratados en el texto son muy sencillos. Analícelos con el niño para hacerle ver que básicamente está utilizando instrucciones ya conocidas para él.
- Esta instrucción puede combinarse también con otras instrucciones en BASIC, como **PRINT, LET, STOP, END,** etc. Algunos ejemplos se comentarán más adelante (como el programa de la página 108). Otros ejemplos pueden ser:

```
IF X = 1 THEN PRINT "...."
IF A = B THEN STOP
IF M = 5 THEN LET C = C + 1
```

IF

SIGUE PROBANDO

★ Te vamos a explicar ahora un poco más sobre la instrucción **IF**. Tienes que prestar mucha atención, pero si no lo entiendes, no te preocupes. Ya sabrás hacerlo más adelante.

Verás. Como has comprobado, la instrucción **IF** permite poner unas condiciones dentro del programa. Si se cumplen, el programa salta a otra línea que nosotros queramos. Si no se cumplen, sigue su curso.

¿Te acuerdas del programa anterior? Con las líneas 60, 70 y 80 le poníamos condiciones al ordenador. Según se cumplirán esas condiciones, el ordenador saltaría a las líneas de programa 90, 110 ó 130.

En realidad, podríamos haber suprimido la línea de programa número 60. También tienes que quitar las líneas 85 y 87. Hazlo y verás que el resultado es el mismo.

Eso sucede porque si eliges el balón rojo no se cumple ninguna de las condiciones que habías puesto en las líneas 70 y 80 y el ordenador seguirá su curso normal y automáticamente leerá la línea de programa número 90. Sigue estas explicaciones que te damos con el programa a la vista y verás como lo entiendes.

Pero ¡Atención! Si no escoges "el azul" o "los dos", el ordenador también irá a la línea 90, pongas lo que pongas. Por ejemplo, pon "el verde" (que no existe) y verás.

IF

★ A ARTURO le han dado las notas de la primera evaluación. Si le dan notas entre 0 y 4 está suspendido. Si le dan notas entre 5 y 10 está aprobado. Ahora tú vas a ser el profesor de ARTURO en la asignatura de matemáticas. Dale la nota que quieras.


```
10  PRINT "¿QUE NOTA TIENE ARTURO?"
20  INPUT A
30  IF A<=4 THEN 60
40  PRINT "¡¡ARTURO HA APROBADO!!"
50  GO TO 70
60  PRINT " ... ARTURO HA SUSPENDIDO..."
70  END
```

El signo especial que aparece en la línea 30 significa "menor o igual a". Por eso la línea 30 es como si dijeramos: Si el valor que le demos a la variable A es menor o igual que 4, entonces vete a la línea 60.

Si el valor es distinto a esa condición, entonces el programa sigue su curso normal y se ejecutará la línea 40.

Corre el programa y dale diferentes notas a ARTURO.

IF

NOTAS PARA EL ADULTO

- Existen unos símbolos específicos por medio de los que establecemos una relación entre los términos de una expresión. Estos símbolos son similares a los utilizados en matemáticas. No obstante, existen pequeñas diferencias que deben ser respetadas escrupulosamente en lenguajes informáticos.
  Relacionamos a continuación los símbolos más usuales. Se llaman operadores lógicos y son:

=	es igual a
<	es menor que
>	es mayor que
<=	es menor o igual que
>=	es mayor o igual que
<>	es distinto de

- Invente algún programa sencillo en el que el niño pueda aplicar alguno de estos operadores lógicos.
- Analice cuidadosamente con el niño los comentarios de la página 75, especialmente en lo que se refiere a la supresión de la línea 60 del programa y sus consecuencias.
- Las instrucciones **STOP** y **END** tienen funciones similares, pero no idénticas. **STOP** representaría el final lógico del programa, mientras que **END** representaría el final físico del programa. **END** debe ponerse siempre en la última línea de programa.

FOR / NEXT

Con todo lo que ya sabes, puedes hacer ya programas muy completos en lenguaje BASIC. Con un poco de imaginación y paciencia eres ya capaz de hacer cosas muy interesantes.

En este capítulo y en los siguientes vamos a explicarte algunas instrucciones que, aunque no son imprescindibles, son muy útiles para programar. Te facilitarán mucho las cosas y te ahorrarán mucho trabajo y tiempo.

Una de esas instrucciones se llama **FOR ... NEXT**. Verás la utilidad que tiene. Suponte que quieres hacer un programa que te de una lista de números del 0 hasta el 10. De acuerdo a lo que ya conocemos, tú harías el siguiente programa (intentalo tú primero):

```
10  LET A=0
20  PRINT A
30  LET A=A+1
40  IF A>10 THEN 60
50  GO TO 20
60  END
```

FOR / NEXT

En la línea 40 le hemos puesto al ordenador la condición de que cuando A tome un valor *mayor* que 10, salte a la línea 60 del programa, en donde se parará.

Mientras esta condición no se cumpla, el programa seguirá ejecutándose hasta la línea 50. Allí recibe la orden de volver a la línea 20, en donde irá imprimiendo el valor que tiene A en ese momento. Luego, en la línea 30, irá asignando nuevos valores a la variable A.

Pues bien, con la instrucción **FOR...NEXT** escribirás el mismo programa de la siguiente manera:

```
10  FOR A=0 TO 10
20  PRINT A
30  NEXT A
```

Córrelo y verás..Te has ahorrado muchas líneas de programa. Vamos a ver como funciona esta instrucción:

Las líneas 10 y 30 funcionan juntas. Es decir, que siempre que tengamos una línea FOR ... TO, tenemos que poner una línea con NEXT.

FOR / NEXT

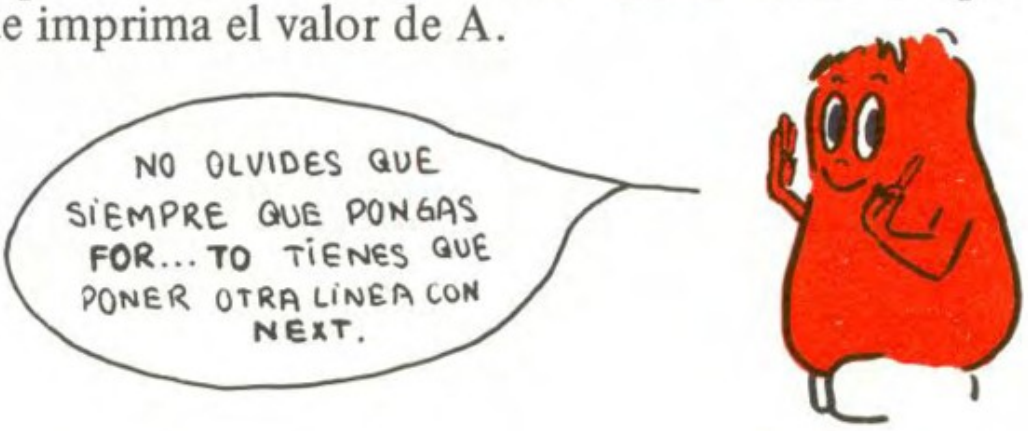
Fíjate en nuestro ejemplo. En la línea 10 le decimos al ordenador que a la variable A le vamos a dar valores entre 0 y 10. Es decir, que A va a valer primero 0, luego 1, luego 2, 3, 4, 5... hasta 10. Bien. El ordenador lo memoriza.

En la línea 20 le decimos al ordenador lo que queremos que haga con esos valores que va tomando A. En este caso, queremos que los imprima. En otros casos, podemos pedirle que haga cálculos o lo que queramos. Ya lo veremos en su momento.

En la línea 30, utilizando **NEXT**, le decimos al ordenador que tome los valores de A que tiene en su memoria. Es como si le dijéramos: "Toma uno detrás de otro todos los valores de A que te hemos dado con FOR ... TO".

Por eso esta instrucción tiene dos partes que van siempre juntas dentro del programa, aunque estén separadas por otras líneas de programa.

Entre la primera parte (FOR ... TO) y la segunda (NEXT), podemos indicarle al ordenador que haga lo que nosotros queramos. En este caso le hemos dicho simplemente que imprima el valor de A.



FOR / NEXT

Por tanto, el programa funciona así: En la línea 10 el ordenador memoriza que A va a valer 0, 1, 2, 3 ... hasta 10. En la línea 20, recibe la orden de imprimir dicho valor. En la línea 30 mandamos al ordenador tomar, uno detrás de otro, los valores de A.

O sea, que las líneas de este programa actúan así:

En la línea 10 . . . . . Se memorizan los valores que tendrá A
En la línea 20 . . . . . Imprime 0 (primer valor)
En la línea 30 . . . . . Toma el siguiente valor de A (que es 1) y vuelve a la línea 20

En la línea 20 . . . . . Imprime 1
En la línea 30 . . . . . Toma el siguiente valor de A (que es 2) y vuelve a la línea 20

En la línea 20 . . . . . Imprime 2

Y así sucesivamente hasta que A valga 10, momento en que en ordenador terminaría este proceso.

Este proceso se llama *bucle*. Dentro de un bucle se le puede pedir al ordenador que haga todos los cálculos que queramos. Los bucles son muy útiles cuando queremos repetir una operación que es siempre la misma porque, como has visto, el ordenador toma por sí solo los distintos valores con los que queramos trabajar.

FOR / NEXT

★ Veamos otra forma de utilizar esta instrucción. Haz el programa:

```
10  FOR A=0 TO 50
20  PRINT "HOLA";
30  NEXT A
```

Le estás diciendo al ordenador que imprima 50 veces la palabra HOLA.

La variable A no la estás utilizando para hacer cálculos con los valores que tome, sino simplemente como un "contador". Interesante ¿eh?

★ Fíjate en este programa de cálculo:

```
10  FOR A=1 TO 10
20  LET X=A*5
30  PRINT X
40  NEXT A
```

Este programa quiere decir que el ordenador dará sucesivamente a la variable A valores desde 0 hasta 10, los multiplicará por 5 e imprimirá el resultado.

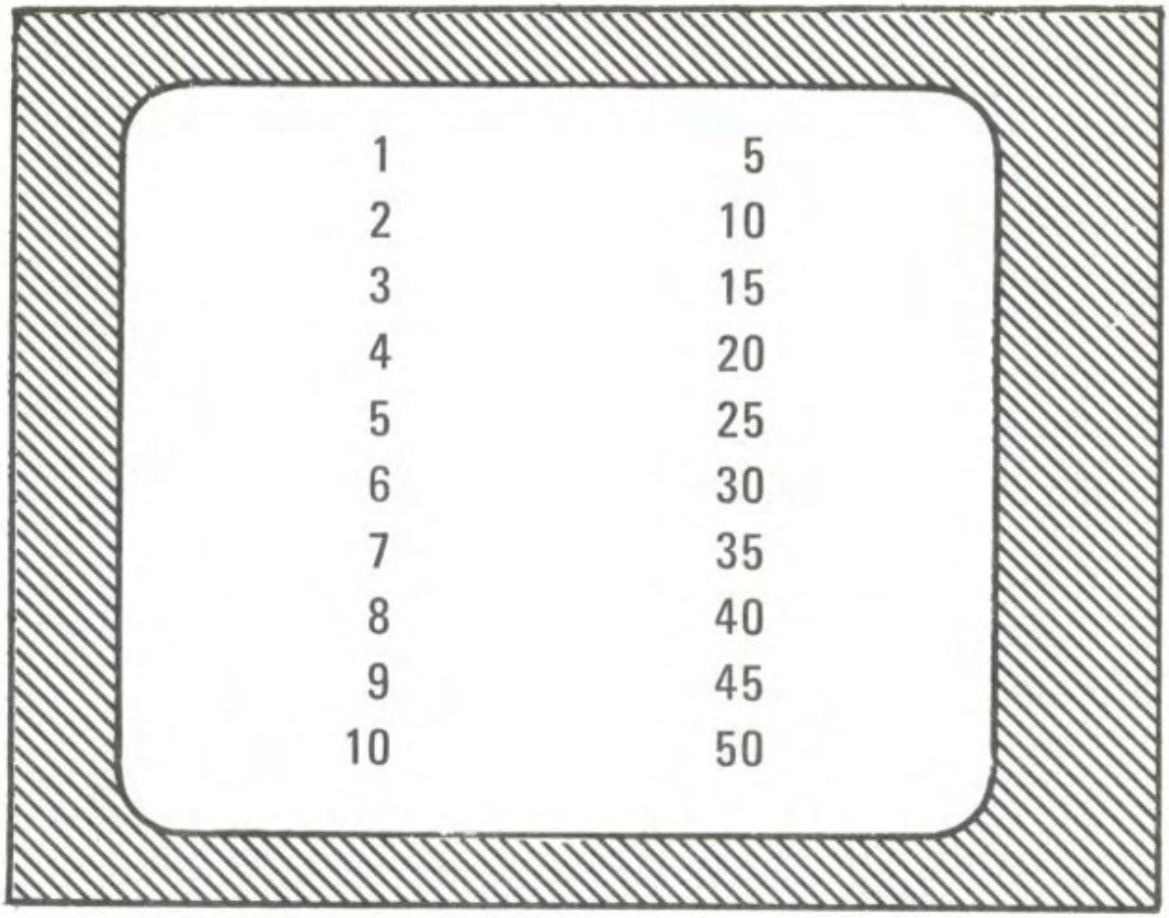
Ejecútalo.

FOR / NEXT

★ Completa el programa cambiando la línea 30 por:

```
30  PRINT A, X
```

Tendrás una pantalla como ésta:



En la primera columna aparecen los valores que va tomando A. En la segunda columna aparecen los sucesivos valores de X (resultado de multiplicar los sucesivos valores de A por 5).

FOR / NEXT

NOTAS PARA EL ADULTO

- La instrucción **FOR/NEXT** ejecuta un bucle o secuencia iterativa que se inicia cuando la variable toma el valor inicial indicado en la instrucción y termina al alcanzar el valor final.
- Esta instrucción es muy utilizada en BASIC por su rapidez y comodidad, pero debe aprenderse dosificadamente. No fuerce al niño. Por el momento es más importante que sepa utilizarla en la práctica a la comprensión de todo el proceso desde un punto de vista conceptual.
- En los juegos que se ofrecen al final de este libro, aparecen dentro de los programas algunos bucles de este tipo. Su justificación es sencilla y conviene recomentar con el niño su repercusión en los programas.
- En un bucle **FOR/NEXT** es posible predeterminar el incremento del valor de la variable. Basta con añadir la instrucción **STEP**. Así, si quisiéramos que A incrementara su valor de dos en dos, escribiríamos:

```
10 FOR A=0 TO 10 STEP 2
```

- No olvide que una vez finalizado un bucle, el ordenador continúa el programa en la línea siguiente a la de la sentencia **NEXT.**
- Dentro de un bucle **FOR/NEXT** puede incluirse una instrucción (con **GOTO**) que obligue al programa a salirse del bucle, pero no se puede acceder al bucle desde el exterior del mismo.

GO SUB

Ya eres casi un programador experto. Te vamos a mostrar ahora una instrucción que no es imprescindible, pero que es también muy útil.

Se llama **GO SUB**, y va siempre acompañada de otra que se llama **RETURN**. ¿Te acuerdas de **GO TO**? Con **GO TO** obligábamos al ordenador a saltar a otra línea de programa. Con **GO SUB** pasa lo mismo, pero luego con **RETURN** hacemos que vuelva automáticamente otra vez al programa. Concretamente, a la línea siguiente a donde se encontraba la instrucción **GO SUB**.

¿Y esto para qué sirve?

Pues verás. Imagínate que tienes un programa y que hay un conjunto de instrucciones que se repite varias veces. Para no escribirlo todo todas esas veces, lo escribes solamente una (generalmente al final del programa) y mediante **GO SUB** haces que cuando sea necesario, el ordenador vaya a esa parte del programa.

GO SUB

Vamos a verlo con un ejemplo:

★ Suponte que estás rellenando un papel en donde te piden tu nombre y apellidos. El programa sería así:

```
10  PRINT "NOMBRE"
20  INPUT A$
30  PRINT "PRIMER APELLIDO"
40  INPUT B$
50  PRINT "SEGUNDO APELLIDO"
60  INPUT C$
70  PRINT "FECHA DE NACIMIENTO"
80  INPUT D$
90  PRINT
100 PRINT
110 PRINT
120 PRINT A$ + B$ + C$
130 PRINT "NACI EL "; D$
```

Corre el programa. Cuando contestes, conviene que teclees un espacio en blanco después de dar tu nombre, apellidos, etc. Así queda más bonito cuando aparezca en pantalla y no se juntan las palabras.

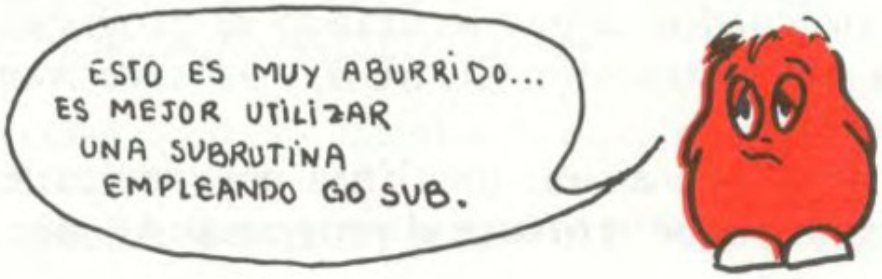
GO SUB

★ Supongamos que queremos que después de cada pregunta aparezca una línea de puntos para separar una pregunta de otra. Modificarías el programa añadiendo las líneas:

```
15 PRINT " .-.-.-.-.-.-.-.-.-.-"
35 PRINT " .-.-.-.-.-.-.-.-.-.-"
55 PRINT " .-.-.-.-.-.-.-.-.-.-"
75 PRINT " .-.-.-.-.-.-.-.-.-.-"
```

Hazlo y corre el programa. Introducir estas líneas ha sido muy aburrido, y tienes que prestar atención para no equivocarte poniendo puntos y rayas.

Si tuvieras un programa que te hiciera 20 ó 30 preguntas te pasarías el día entero escribiendo líneas como la línea 15.



Para evitarlo, utilizaremos una *subrutina.* Una subrutina es como un programa dentro del programa general. Casi siempre se ponen al final del programa general y puede tener muchas líneas.

GO SUB

★ En nuestro ejemplo, resolveríamos esta repetición de líneas con una subrutina como ésta:

```
200 PRINT " .-.-.-.-.-.-.-.-.-."
210 RETURN
```

Y ahora cambia las líneas 15, 35, 55 y 75 por líneas que digan :

```
GO SUB 200
```

Cada vez que el programa llega a una instrucción **GO SUB** salta a la línea 200, donde le mandamos imprimir la línea de puntos. En la línea 210 le obligamos a volver al programa principal. Vuelve siempre a la línea siguiente a aquella en que se encontraba **GO SUB**. En este caso, volverá sucesivamente a las líneas 20, 40, 60 y 80.

El ordenador sigue ejecutando el programa hasta que llega a otra instrucción **GO SUB**, y así sucesivamente.

★ Es necesario añadir una línea para indicarle al ordenador cuándo debe terminar el programa. Añade:

```
135 STOP
```

Tal vez te parezca que hay poca diferencia entre escribir cada línea de puntos en su sitio o indicarle al ordenador que se dirija a la subrutina. En nuestro ejemplo no hay

mucha diferencia, pero eso es porque hemos elegido una subrutina muy sencilla.

Imagínate que la subrutina fuera un complicado cálculo con muchas operaciones distintas y que tuvieras que escribirlas cada vez que, dentro del programa, fuera necesario efectuarlas.

¡Seguro que prefieres utilizar **GO SUB**!

Además, no te olvides que una subrutina puede estar formada por muchas líneas de programa.

NOTAS PARA EL ADULTO

- Haga ver al niño las ventajas de la utilización de subrutinas. No obstante, no insista en este concepto hasta que el niño adquiera la suficiente soltura en el manejo y utilización de las demás instrucciones.

- Si lo cree oportuno, intenten conjuntamente el desarrollo de un ejemplo con subrutinas complejas, tanto con cálculos amplios como con textos muy extensos.

- Observe como las cadenas alfanuméricas pueden estar constituidas por grupos de palabras o números. La fecha de nacimiento es una cadena y por tanto hay que asignarla a una variable alfanumérica.

READ y DATA

Otra instrucción útil para ganar tiempo es la formada por **READ** y **DATA**.

Cuando en un programa hay que introducir muchos datos, las sentencias **LET** e **INPUT** nos hacen el trabajo muy pesado.

Fíjate en este programa:

```
10  LET A=5
20  LET B=7
30  LET C=4
40  LET D=2
50  LET E=A+B
60  LET S=A+B+C+D+E
70  PRINT S
```

Corre el programa y te aparecerá en pantalla la suma, que es 30.

READ y DATA

★ El mismo resultado lo obtenemos utilizando la instrucción **READ** y **DATA**.

Con ellas, el programa se escribiría así:

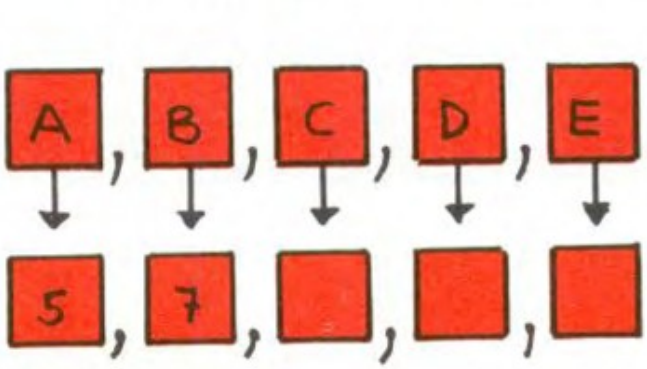
```
10  READ A, B, C, D, E
15  LET E = A + B
20  DATA 5, 7, 4, 2, E
30  LET S = A + B + C + D + E
40  PRINT S
```

El resultado es el mismo. Como ves, una sentencia **DATA** es como un almacén de datos, que están separados por comas. El ordenador no la tiene en cuenta a menos de que se encuentre una instrucción READ, que le obliga a leer los datos que están almacenados en la sentencia **DATA**.

El ordenador toma los valores que estan en **DATA** y los asigna en el mismo orden a las variables que están en **READ**.

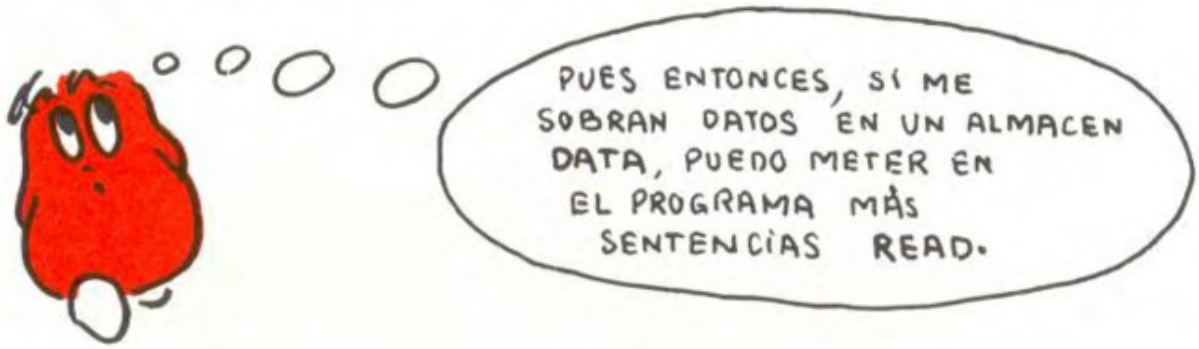
En nuestro ejemplo, A toma el valor 5, B toma el valor 7, etc.

Observa que E toma el valor A + B.



READ y DATA

En un programa, una instrucción **DATA** puede tener más datos o valores que variables contenga una instrucción **READ**. Los datos que sobran quedan "a la espera" de que otras sentencias **READ** obliguen al ordenador a tomarlos.



Sobre nuestro ejemplo anterior, hagamos un programa con el que puedes ver como distintas instrucciones **READ** a lo largo del programa van tomando los valores que estaban en una instrucción **DATA**.

★ Fíjate también que esos valores pueden ser textos, números o expresiones. Vamos con el programa:

```
10  DATA 1, 2, 3, 4, 5, "ARTURO", "CARLOS"
20  READ A, B, C
30  READ D, E
40  LET S = A + B + C + D + E
50  READ F$, G$
60  PRINT F$; " y "; G$; " TIENEN "; S;
    " CARAMELOS"
```

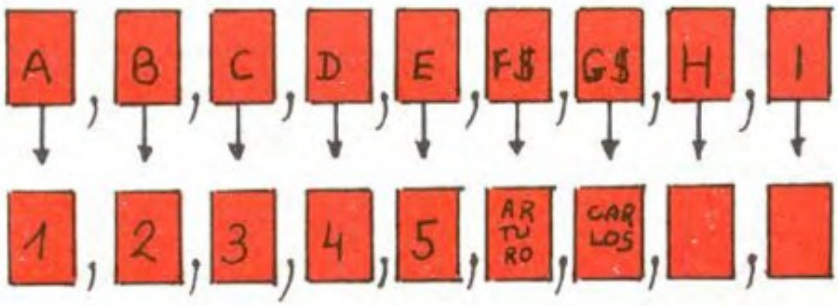
READ y DATA

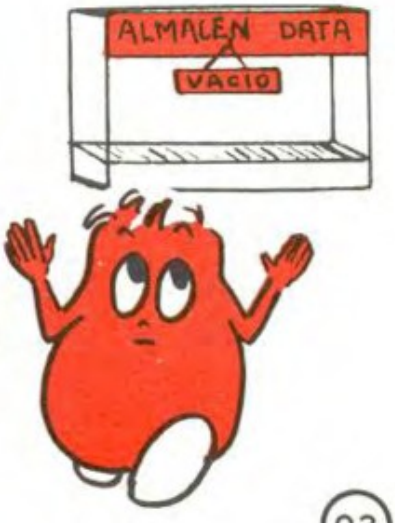
EN EL PROGRAMA ANTERIOR, LAS SENTENCIAS READ DE LAS LINEAS 20, 30 Y 50 VAN TOMANDO LOS VALORES QUE ESTÁN EN EL ALMACEN DATA DE LA LÍNEA 10.

FÍJATE COMO LAS VARIABLES DE LA LÍNEA 50 TIENEN EL SIGNO $ Y LES SERÁN ASIGNADAS LOS TEXTOS "ARTURO" Y "CARLOS".

Has visto que no hay problema si tenemos más datos que variables. Simplemente quedan a la espera de que el ordenador los lea a través de una nueva instrucción **READ.**

★ Pero si el ordenador se encuentra que tiene que leer un dato que no existe (es decir, que tenemos más variables que datos), aparecerá en la pantalla una indicación advirtiendote del error. El programa no sigue adelante. Compruébalo añadiendo a nuestro último programa la línea 55 READ H, I.





READ y DATA

Verás que el programa se interrumpe y te aparece un mensaje parecido a OUT OF DATA IN... indicándote seguidamente la línea en la que el ordenador encuentra variables a las que no puede asignar ningún valor.

★ Este mensaje puede aparecerte también cuando en un bucle, el ordenador se encuentre sin datos en una instrucción **DATA** y no pueda asignarlos a variables que esten en **READ**. Por ejemplo:

```
10  DATA 5, 7, 4, 2
20  READ A
30  PRINT A, 2 * A
40  GO TO 20
```

En este programa el ordenador toma el primer valor (5) y lo asigna a la variable A. En la línea 30, imprime ese valor junto con dicho valor multiplicado por dos. En la línea 40 vuelve a comenzar el proceso asignándole sucesivamente a la variable A los valores almacenados en **DATA**. Cuando ya todos los valores están asignados, como la línea 40 remite al ordenador a la Instrucción **DATA**, aparecerá el mensaje OUT OF DATA.

READ y DATA

★ Si no se tiene cuidado, una instrucción de este tipo puede producir inconvenientes en un programa. Suponga-mos que se desean sumar los valores de una sentencia **DATA**. El programa podría ser:

```
10  DATA 5, 7, 4, 2
20  LET S = 0
30  READ A
40  LET S = S + A
50  GO TO 30
60  PRINT S
70  END
```

Este programa no funcionará nunca, pues está mal estructurado. Hubiera sido mejor cambiar el orden de las líneas 50 y 60. Tal y como está el programa, antes de llegar a la línea 60 e imprimir el resultado de la suma, el ordenador vuelve a la sentencia **DATA** y llega un momento en que se encuentra sin valores.

READ y DATA

★ Si el programa está muy complicado y resulta difícil cambiar el orden de las líneas, se puede utilizar un truco:

Se puede añadir un valor "falso" a la sentencia **DATA**. Es decir, un valor que a nosotros no nos interesa para nada en nuestro cálculo.

Además, ponemos una sentencia **IF** para que cuando el ordenador lea ese valor "falso" el programa salte a otra sentencia que nos convenga.

En nuestro ejemplo, transformaríamos la sentencia 10 así:


```
10  DATA 5, 7, 4, 2, 1000
```

y añadiríamos:

```
35  IF A = 1000 THEN 60
```

Así, cuando el ordenador lee el valor 1000, va directamente a la línea 60, donde imprime el valor de S acumulado hasta ese momento y termina el programa.

READ y DATA

NOTAS PARA EL ADULTO

- Esta instrucción asigna a las variables que figuran en **READ** los datos que aparecen en **DATA**. Insista ante el niño que una instrucción **DATA** puede abastecer a varias instrucciones **READ.**

- Conviene situar las sentencias **DATA** al principio o al final del programa.

- Esta sentencia sólo debe practicarse cuando ya estén perfectamente asimilados los conceptos vertidos en capítulos anteriores.

- Si el niño no ha adquirido la suficiente soltura, no insistan sobre esta sentencia. Siempre pueden volver sobre ella cuando lo consideren oportuno, ya que no es sino un valioso auxiliar en la programación.

REM

Las instrucciones que has conocido últimamente eran un poco complicadas. ¡Esta es muy fácil! Además, ayuda bastante a refrescarnos la memoria.

Una instrucción **REM** no afecta para nada el programa. El ordenador no la tiene en cuenta. Nos sirve a nosotros para no olvidarnos de lo que estamos haciendo.

Puede ocurrirte que cuando mires un programa que tú mismo has hecho, no te acuerdes muy bien de por qué hiciste algunas líneas de programa. Por eso puedes intercalar sentencias **REM** donde lo consideres oportuno, para que te refresquen la memoria cuando vuelvas a ver el programa dentro de algún tiempo. Es como una "chuleta".

REM

Por ejemplo, en el programa de la página 94 podríamos añadir las líneas:

```
5   REM VALORES QUE TOMARA A
25  REM IMPRIME EL VALOR DE A Y
    EL DOBLE DE SU VALOR
```

Se pueden añadir todas las sentencias **REM** que se quiera. Ya sabes que no afectan al programa y sólo las puedes ver cuando listes el programa. Puedes poner el texto que tú quieras en una sentencia **REM.** Lo importante es que te recuerden, dónde tú lo precises, los pasos que has ido siguiendo al hacer un programa.

NOTAS PARA EL ADULTO

- Practique con el niño sobre programas anteriores añadiendo aclaraciones allí donde lo crean necesario.

INT

Esta instrucción o, mejor dicho, esta función, también será útil en programas de cálculo y en programas de juegos, como verás más adelante.

Lo que hace **INT** es quitar los decimales de un número, dejando sólo su parte entera.

De esta manera, el número 2,489 se quedaría en 2.

El número 5,246589 se quedaría en 5.

★ Vamos a poner un ejemplo. Escribe:

```
10 LET A=2.3489
20 PRINT A
```

Al correr el programa, aparece en pantalla 2.3489. Fíjate en algo muy importante: En lugar de utilizar la coma para separar la parte entera de la decimal, utilizas un punto. Esto debes hacerlo siempre en Informática.

★ Añade ahora la línea:

```
15 LET A=INT (A)
```

INT

Corre el programa.

En pantalla te aparece el número 2. Has dejado únicamente la parte entera.

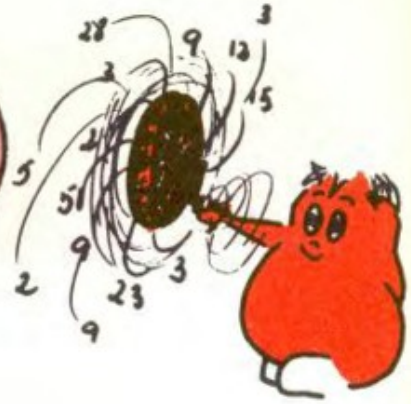
Fíjate que la función **INT** no redondea. Simplemente, deja la parte entera, o sea, que con **INT** el número 2.9 se convierte en el número 2 y no en el número 3.

Como te hemos dicho antes, **INT** es de mucha utilidad en lenguaje BASIC. Ya lo irás viendo con el tiempo. En el próximo capítulo verás algunas cosas interesantes sobre su utilización.

NOTAS PARA EL ADULTO

- Sobre el programa anterior, haga que el niño modifique la línea 10 varias veces, con números de su elección.
- Es probable que el niño no se encuentre especialmente motivado por esta función. Anticípele que tiene gran importancia en los juegos y en cálculos y programas más complejos.
- Remarque la importancia de utilizar un punto para separar la parte entera de la decimal. Además, tenga en cuenta que en Informática no se utiliza el punto para diferenciar unidades de mil. Así, dos mil cien se escribiría 2100.

RND



¿Sabes lo qué es el azar, o la casualidad?

Muchas veces tú dices: "Esto ha pasado por casualidad". Es decir, quieres decir que también podía *no haber pasado*. O podía *haber pasado otra cosa.*

Pues bien, con la función **RND** puedes introducir el azar en un programa. Y eso ¿para qué sirve? Pues para mucho. Por ejemplo, para jugar al escondite con el ordenador, como veremos más tarde. También, para simular con el ordenador sucesos que ocurren en la vida real y en los que interviene el azar.

★ Pero vayamos poco a poco. Con la instrucción **RND** se hace que el ordenador fabrique o genere números al azar (aleatorios) entre 0 y 1 (sin incluir el 1). Por ejemplo, escribe:

```
10  LET A = RND (X)
20  PRINT A
```

RND

Cada vez que corras el programa, en la pantalla te aparecerá un número igual o mayor que 0 y menor que 1. Por ejemplo:

```
0.53118986
0.83547393
0.97816467
0.24576398
```

★ Para que el ordenador nos de números al azar comprendidos entre 0 y 10 (por ejemplo), tendremos que multiplicar por 10 el valor de RND (X). Cambia la línea 10 por:

```
10  LET A = RND (X) * 10
```

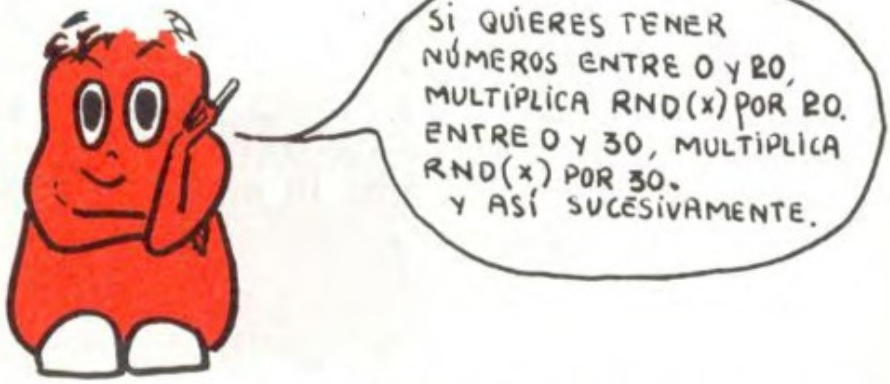
RND

Añade:

```
30  GO TO 10
```

Tendrás una lista de números al azar entre 0 y 10 (sin incluir el 10). Por ejemplo:

```
0.18976534
4.00789654
1.36962891
3.00000000
8.34698762
```

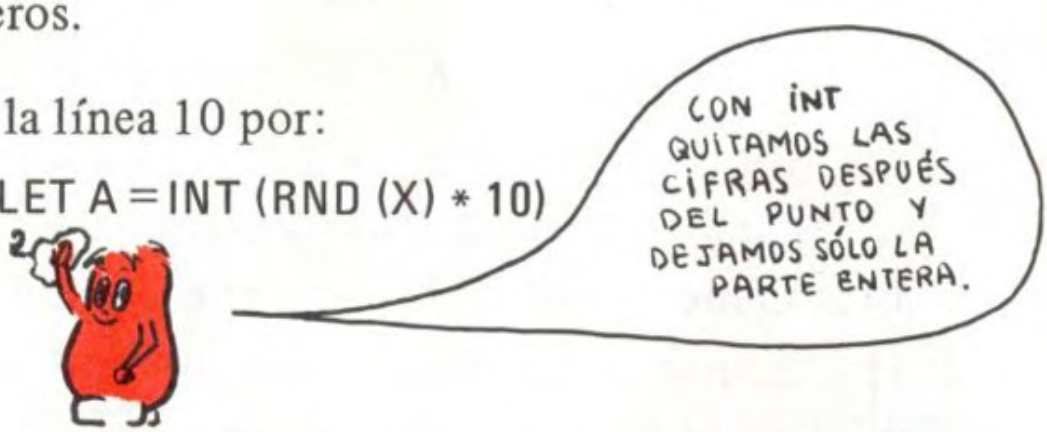


¿Te acuerdas de la instrucción **INT**? Te va a ser muy útil ahora.

Con ella puedes hacer que sólo quede la parte entera de estos números.

★ Cambia la línea 10 por:

```
10  LET A = INT (RND (X) * 10)
```



Tendrás una lista de números enteros entre 0 y 10, excluyendo el 10.

RND

★ Suponte ahora que quieres analizar los números que saldrían al lanzar un dado. Siguiendo las fórmulas del ejemplo anterior, escribirías:

```
10  LET A=INT (RND (X) * 6)
```

Pero ¡ ¡atención!! Esta sentencia sólo te da valores enteros entre 0 y 6, excluido el 6. Y además, el dado no tiene ceros. Por eso, tendrás que añadir una unidad, para que tengas efectivamente números del 1 al 6, ambos inclusive. Escribe pues:

```
10  LET A=INT (RND (X) * 6) +1
20  PRINT A ;"  ";
30  GO TO 10
```

Al correr el programa, tendrás en la pantalla una lista con los números que podían haber salido al tirar un dado.

Has reproducido con el ordenador lo mismo que te puede salir en la vida real cuando juegas a los dados. Quita la línea 30 y cada vez que corras el programa, es como si tiraras un dado.

RND

★ Suponte que queremos hacer exactamente 100 tiradas de dados para ver qué números nos pueden salir en esas 100 tiradas. Para hacerlo, modificaremos el programa añadiendo un "contador". Cuando el contador llegue a 100, el programa se parará. Añade las líneas:

```
5   LET C=0
25  LET C=C+1
26  IF C=100 THEN 40
30  GOTO 10
40  END
```

Tendrás ahora un programa como éste:

```
5   LET C=0
10  LET A=INT (RND (X) * 6) +1
20  PRINT A;"  ";
25  LET C=C+1
26  IF C=100 THEN 40
30  GO TO 10
40  END
```

RND

El programa funcionaría así: En la línea 5 pones el contador a cero. En las líneas 10, 20 y 30 el ordenador va dando al azar e imprimiendo números comprendidos entre 1 y 6. En la línea 25 hacemos que el contador vaya aumentando de uno en uno. En la línea 26 indicamos al ordenador que cuando el contador llegue a 100, salte a la línea 40 con lo que el programa se para.

Analiza bien estas aclaraciones con el programa a la vista. Luego, ejecútalo.

★ ¡Y ahora viene lo bueno! Tú sabes que hay las mismas posibilidades de que al tirar un dado te salga un número u otro.

En teoría, podríamos pensar que de 100 tiradas, un número determinado (por ejemplo, el 5) se repetiría 16 veces. (Claro. 100 tiradas dividido entre 6 números que pueden salir = 16 posibles repeticiones por número).

Pero en la práctica, el azar hace que pueda salir más o menos veces. Podemos hacer pruebas tirando 100 veces un dado, para ver qué pasa. Pero eso es muy aburrido y lento.

Con el ordenador podemos simular la tirada de los dados. De forma mucho más rápida, podemos ver que pasaría tirando 100 veces un dado y contando cuántas veces sale un número (por ejemplo, el 5).

RND

★ Para hacerlo, sólo tenemos que añadir al programa anterior unas líneas que nos cuenten, dentro del programa, cada vez que aparece el número 5.

Añade:

```
7   LET P=0
22  IF A=5 THEN LET P=P+1
35  PRINT C, P
```

y cambia la línea 26 por:

```
26 IF C=100 THEN 31
```

Añade también:

```
31  PRINT
32  PRINT
33  PRINT
34  PRINT
```

Con ello tendrás una pantalla final más clara, pues el resultado final queda separado de los otros números.

Con la línea 7 inicias el contador que marca las veces que aparece el número 5. Con la línea 22, le dices al ordenador que cada vez que el número sea 5, añada una unidad al contador.

RND

Con la línea 35 haces que, al final, el ordenador te imprima el número de tiradas y las veces que ha salido el número 5.

Como has visto, tienes que cambiar un poco la línea 26 para que el programa esté bien estructurado.

El programa final te habrá quedado así:

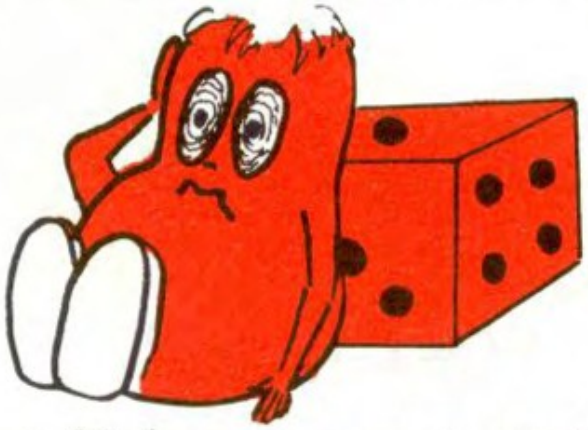
```
5   LET C=0
7   LET P=0
10  LET A=INT (RND (X) * 6) +1
20  PRINT A;"  ";
22  IF A=5 THEN LET P=P +1
25  LET C=C +1
26  IF C=100 THEN 31
30  GO TO 10
31  PRINT
32  PRINT
33  PRINT
34  PRINT
35  PRINT C, P
40  END
```

Ejecútalo y verás. Al final, el ordenador te dirá que ha probado 100 veces y el número de veces que salió el número 5.

RND

★ Fíjate la ventaja que tienes cuando trabajas con un ordenador. Suponte que quieres ver cuantas veces sale el número 5 en 1000 tiradas.

ARTURO intentó tirar 1000 veces el dado pero se cansó cuando llevaba 168. Con el ordenador es todo mucho más rápido.



Quita la línea 20 (para que no te escriba todos los números que van saliendo) y cambia la línea 26 por:

```
26  IF C=1000 THEN 35
```

Cambia también la línea 35 poniendo:

```
35  PRINT "EN ";C;" TIRADAS HAN SALIDO ";
    P;" CINCOS"
```

Corre el programa. Tendrás que esperar un poco, pero mucho menos que si esperaras a que ARTURO te diera el resultado.

RND

★ ¿Serías capaz de modificar el programa para que el ordenador te diga además cuántos cuatros o seises han salido en las 1000 tiradas? Es muy fácil. Sólo tienes que tener en cuenta que tendrás que poner más contadores para estos números. Por ejemplo:

```
6   LET Q=0
23  IF A=6 THEN LET Q=Q+1
```

y cambia la línea 35 por:

```
35 PRINT "EN "; C; " TIRADAS HAN SALIDO ";
   P; " CINCOS y "; Q; " SEISES"
```

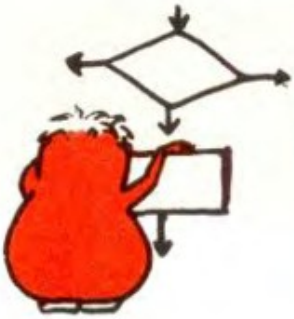
Haz tú las modificaciones para los demás números.

RND

NOTAS PARA EL ADULTO

- Al igual que en capítulos anteriores, esta instrucción debe tratarse secuencial y paulatinamente. Lo único importante es que el niño capte el concepto de aleatoriedad que se obtiene a través de **RND.**
- Motive al niño después de cada paso animándole a descubrir las ventajas que le ofrece el paso siguiente. Plantee todo como un experimento.
- La función **RND** no se expresa igual en todos los ordenadores. Generalmente, en BASIC se exige que después de **RND** se sitúe un valor entre paréntesis. Este valor puede ser, sencillamente X. No se preocupe si ya existe esta variable en el mismo programa en que utilice la instrucción **RND.**
- Consulte previamente el manual de instrucciones de su ordenador personal. Estudie con atención la forma de expresar la función **RND** y haga las modificaciones oportunas en los programas expuestos en este capítulo.
- Algunos ordenadores requieren que se incluya al principio del programa el comando RANDOMIZE, a fin de obtener diferentes series de números aleatorios cada vez que se ejecute el programa. De lo contrario, siempre darán la misma serie. Así, se escribiría:

```
10  RANDOMIZE
20  LET A = RND (X)
30  PRINT A
```



Diagramas

Conviene que sepas que, a la hora de programar, es útil utilizar unos dibujos o diagramas para tener las ideas muy claras y organizar bien el trabajo que vamos a hacer.

Además, estos diagramas te servirán para comprender, con un poco de atención, como está construido o estructurado el programa. Puedes ver enseguida cómo va a funcionar cuando lo estés ejecutando.

Lo mejor es hacer un diagrama antes de hacer el programa, para tener muy claro como vamos a explicarle al ordenador lo que tiene que hacer. Sobre todo cuando el programa es un poco complicado.

No vamos a estudiarlos en profundidad. Solamente vamos a ver unos ejemplos muy fáciles para que te acostumbres a conocerlos.

¿Te acuerdas del juego del escondite y de las reglas del juego que marcábamos al principio de este libro?

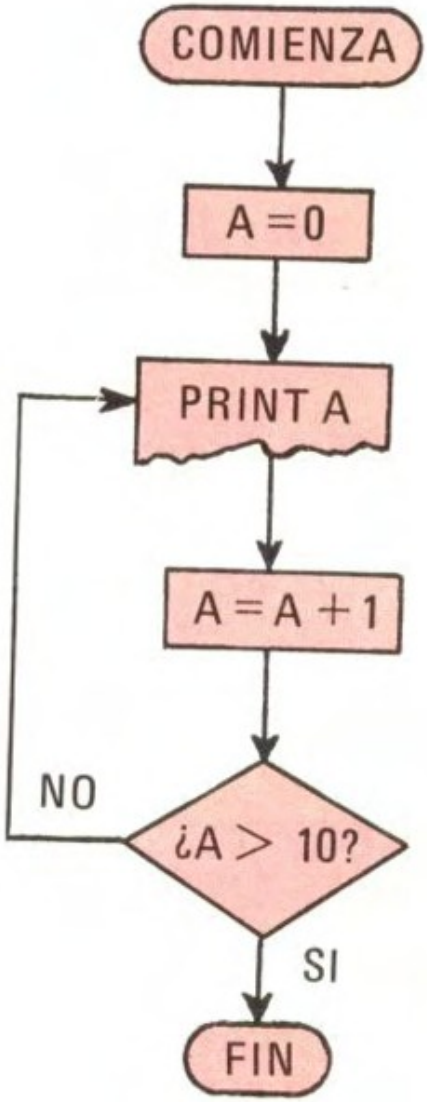
Si tuviéramos que hacer un diagrama de aquél juego, haríamos algo parecido a lo que te mostramos en la página siguiente. Míralo con atención y verás como todo el desarrollo es muy lógico.

Diagramas


EMPEZAMOS
YO CUENTO
ARTURO SE ESCONDE
¿SE ESCONDE EN LA CASA?
SI
NO VALE
NO
YO TE BUSCO
¿TE ENCUENTRO?
NO
HA GANADO ARTURO
SI
HE GANADO YO
FIN

Diagramas

★ Tomemos un programa muy sencillo que nos escribe los números naturales del 0 al 10. El diagrama sería:



Diagramas

El programa escrito quedaría así:

```
10  LET A=0
20  PRINT A
30  LET A=A + 1
40  IF A> 10 THEN 60
50  GO TO 20
60  END
```

Corre el programa y verás como aparece una lista de números del 0 al 10. Te habrá parecido muy fácil pues ya conoces muy bien todas las instrucciones.

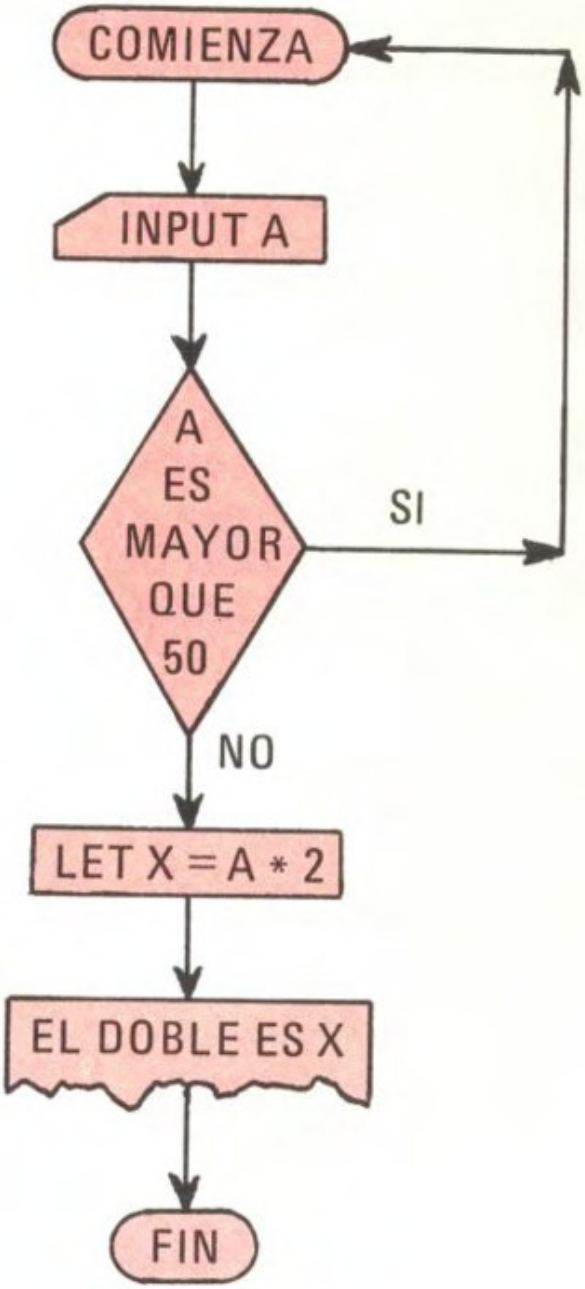
★ ¿Cómo sería el diagrama de flujo de un programa que calculara el doble de un número comprendido entre 0 y 50?

El programa sería:

```
10  PRINT "DI UN NUMERO ENTRE 0 y 50"
20  INPUT A
30  IF A>50 THEN 10
40  LET X=A * 2
50  PRINT" EL DOBLE ES ";X
60  END
```

Y el diagrama sería:

Diagramas


COMIENZA
INPUT A
A ES MAYOR QUE 50
SI
NO
LET X = A * 2
EL DOBLE ES X
FIN

Diagramas

NOTAS PARA EL ADULTO

- Los símbolos convencionales que se usan en los diagramas de flujo pueden variar ligeramente, pero los más comúnmente admitidos son:

PRINCIPIO O FIN
DE PROGRAMA

ENTRADA O SALIDA DE
DATOS (INPUT)

ASIGNACION DE VARIABLES O
REALIZACION DE CALCULOS (LET)

IMPRESION DE
RESULTADOS (PRINT)

DECISIONES LOGICAS (IF ...)

- Aunque es conveniente iniciar al niño en el desarrollo de diagramas de flujo, su explicación detallada escapa a los objetivos de este libro.

- Cuando el niño domine las instrucciones en BASIC que hemos estudiado en este libro y su repercusión dentro de un programa intenten desarrollar diagramas sobre algunos de los programas ya vistos.

Juegos

En este capítulo vamos a ver unos pocos juegos o aplicaciones que pueden resultarte divertidos, útiles e interesantes.

Ojalá te sirvan para desarrollar luego tus propios juegos.

En todos los programas se utilizan instrucciones y sentencias que ya conoces bien. Intenta analizarlos primero para que veas como funcionan.

APRENDIENDO A MULTIPLICAR

Este programa te ayudará de manera divertida a practicar la tabla de multiplicar.

```
10  PRINT "DI UN NUMERO ";
20  INPUT N
30  PRINT N
```

Juegos

```
 40  FOR I=1 TO 10
 50  PRINT N;" * ";I;" =";
 60  INPUT M
 70  PRINT M
 80  IF M=N*I THEN 110
 90  PRINT "¡FALLASTE. INTENTALO DE NUEVO!"
100  GO TO 50
110  PRINT "¡MUY BIEN!"
120  NEXT I
130  END
```

NOTAS PARA EL ADULTO

- En este programa se practican las instrucciones **PRINT, INPUT, FOR NEXT, IF** y **GOTO.**

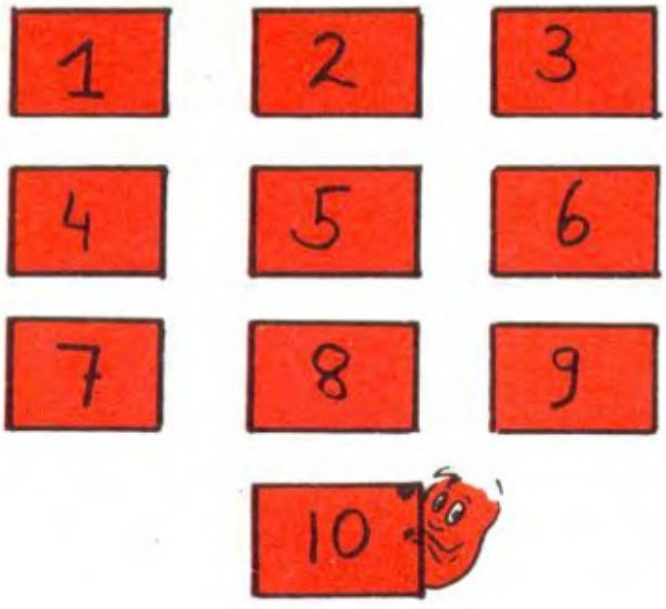
Juegos

JUGANDO AL ESCONDITE CON ARTURO Y EL ORDENADOR

Como te prometimos al principio del libro, vamos a jugar al escondite con ARTURO. El programa es muy sencillo. Si lo analizas un poco, lo comprenderás perfectamente y verás que hubieras podido hacerlo tú mismo.

El juego consiste en lo siguiente: ARTURO se va a esconder en cualquiera de los sitios que hay en el jardín. Sólo puede esconderse en 10 sitios, numerados del 1 al 10. Si se esconde en la casa el juego se acaba en cuanto tú empieces a buscarle.

Cuando lo encuentres, el ordenador te dirá cuántos intentos has hecho para encontrarle. Cada vez que juegues se esconderá en un sitio distinto. ¡A ver quien lo encuentra primero!



Juegos

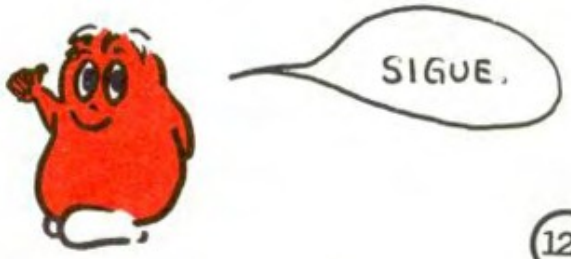
```
 5  REM ESCONDITE
10  LET A=0
20  LET R=INT (RND (X) * 10) +1
25  PRINT "ESCRIBE EL NUMERO DE ESCONDITE"
30  PRINT "¿DONDE ESTA ARTURO?";
35  LET A=A +1
40  INPUT N
50  PRINT N
60  IF R=3 THEN 200
70  IF R=N THEN 100
80  PRINT "AQUI NO ESTA ... "
90  GO TO 30
100 PRINT " ¡LO HAS ENCONTRADO!"
101 PRINT
102 PRINT
103 PRINT "EN  "; A; "  INTENTOS"
105 GO TO 220
200 PRINT "NO VALE. ARTURO SE HA ESCONDIDO
    EN LA CASA"
210 PRINT "EMPIEZA OTRA VEZ"
220 END
```

Juegos

ADIVINANDO NUMEROS

Esto es un programa parecido al escondite. El ordenador pensará un número entre 1 y 100. Tienes que adivinarlo. Al responderte, te dará pistas. ¡Descubre el número en pocos intentos!

```
   5  REM ADIVINANDO
  10  LET A=0
  20  LET R=INT (RND (X) * 100) +1
  30  PRINT "PIENSO UN NUMERO ENTRE 1 y 100"
  40  PRINT "INTENTA ADIVINARLO"
  50  PRINT "DI UN NUMERO"
  60  INPUT X
  63  PRINT
  65  PRINT X
  70  LET A=A +1
  80  IF X<R THEN 2000
  90  IF X=R THEN 3000
1000  PRINT
1005  PRINT
1010  PRINT "FALLASTE. MI NUMERO ES MENOR"
1020  GO TO 50
2000  PRINT
2010  PRINT
```



```
2020  PRINT "FALLASTE. MI NUMERO ES MAYOR"
2030  GO TO 50
3000  PRINT
3010  PRINT
3020  PRINT "¡¡ACERTASTE!!"
3030  PRINT
3040  PRINT "MI NUMERO ES EL  "; R
3050  PRINT
3060  PRINT "LO HAS AVERIGUADO EN "; A;
      " INTENTOS"
3070  PRINT
3080  IF A >= 5 THEN 3100
3090  PRINT "BUEN RESULTADO"
3093  PRINT
3095  GO TO 3105
3100  PRINT "NO ESTA MAL, PERO INTENTA
      MEJORARLO"
3105  PRINT
3110  END
```

Juegos

EL ORDENADOR SABIO

El ordenador puede adivinar tus pensamientos. Compruébalo con este programa. Toma papel y lápiz y haz lo que te diga.

```
10 PRINT "VOY A ADIVINAR TU NUMERO"
20 PRINT
30 PRINT "PIENSA UN NUMERO"
40 PRINT " PERO NO ME LO DIGAS"
50 GO SUB 300
60 PRINT "MULTIPLICALO POR 6"
70 GO SUB 300
80 PRINT "SUMA 10 AL RESULTADO"
90 GO SUB 300
100 PRINT "DIVIDE EL RESULTADO ENTRE 2"
110 GO SUB 300
120 PRINT "RESTALE 5 AL RESULTADO"
130 GO SUB 300
140 PRINT "DIME EL RESULTADO FINAL"
150 INPUT X
160 LET Z = X + 5
170 LET Y = Z * 2
180 LET H = Y - 10
```

Juegos

```
190 LET N=H/6
200 PRINT
210 PRINT
215 PRINT
216 PRINT
220 PRINT "PENSASTE EL NUMERO "; N
225 PRINT
230 STOP
300 PRINT "SI ESTAS LISTO"
310 PRINT "PULSA ENTER"
320 INPUT A$
330 PRINT
340 PRINT
350 RETURN
```

NOTAS PARA EL ADULTO

- En este programa se hace uso de la instrucción **GO SUB**, además de las anteriores.

¡Hasta pronto!

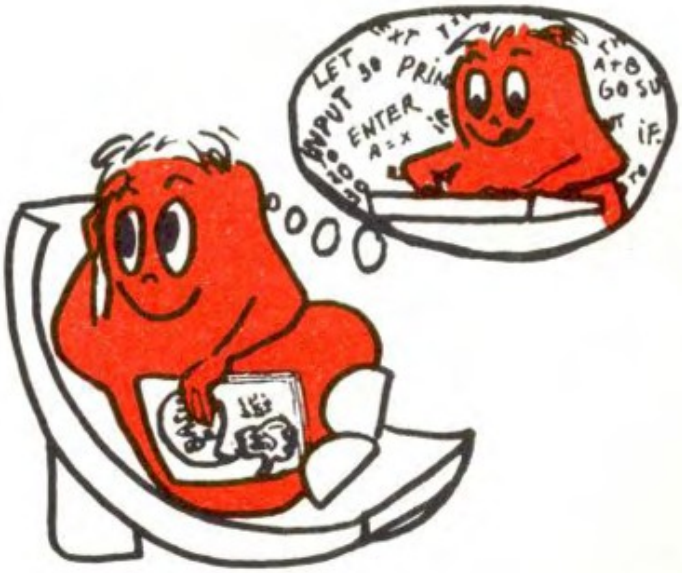
Bueno pues... ya hemos terminado. Ya eres todo un programador (en pequeño, claro). Todavía puedes aprender muchas cosas más sobre el lenguaje de programación BASIC.

Pero no te preocupes, pues con lo que has aprendido en este libro ya conoces lo fundamental y el resto te resultará mucho más fácil.

Ahora tienes que practicar y, sobre todo, experimentar con el ordenador todo lo que se te ocurra. Haz tus propios programas y no tengas miedo a probar una y otra vez diferentes maneras de desarrollarlos.

Y, si alguna vez tienes una duda muy grande, escríbenos. ARTURO intentará resolvértela.

Además, si quieres saber más cosas, ARTURO te las explica en su otro libro BASIC AVANZADO PARA NIÑOS.



BIBLIOGRAFIA CONSULTADA

BELLIDO. *Cómo programar su SPECTRUM.* Ed. PARANINFO, 1984. Madrid.

BELLIDO. *ZX81. Curso de programación BASIC.* Ed. PARANINFO, 1984. Madrid.

BELLIDO y SANCHEZ. *BASIC para maestros.* Ed. PARANINFO, 1984. Madrid.

CHECROUN. *BASIC. Programación de microordenadores.* Ed. PARANINFO, 1984. Madrid.

LARRECHE. *BASIC. Iniciación a la programación.* Ed. PARANINFO. 1984. Madrid.

ROSSI. *BASIC. Curso acelerado.* Ed. PARANINFO, 1984. Madrid.

SPENCER. *BASIC programming.* CAMELOT, 1983. New York.

VARIOS. *BASIC básico. Curso de programación.* Ed. AGUADO, 1984. Madrid.

BASIC para NIÑOS

Este es un libro para niños. Y también es un libro para aquellos padres que quieran acompañar a sus hijos en el aprendizaje del lenguaje BASIC.

No son necesarios conocimientos previos sobre programación. A lo largo del texto, numerosas notas aclaratorias, comentarios y guías didácticas permiten explicar las dudas que puedan surgirle al niño.

Todos los procesos y programas se analizan detalladamente.

De forma amena, utilizando ejemplos sencillos y relacionados directamente con el entorno infantil, se cubren los primeros pasos de programación en BASIC, facilitando una sólida base para el estudio posterior de desarrollos más complejos. Es, pues, una obra ideal para niños.

¡... y para PADRES!

EDITORIAL PARANINFO S.A.
Magallanes, 25 - 28015 Madrid

ISBN: 84-283-1327-X