Basic *avanzado* *para* niños

con notas didácticas
para padres y educadores

```
10 LET A$="BASIC"
20 LET C=LEN A$
30 FOR Q=1 TO C
40 PRINT A$(Q)
50 NEXT Q
```

Sofía Watt
Miguel Mangada

PARANINFO S.A.

Basic *avanzado* para niños

Sofía Watt
Miguel Mangada

Basic *avanzado* *para* niños

QUINTA EDICION

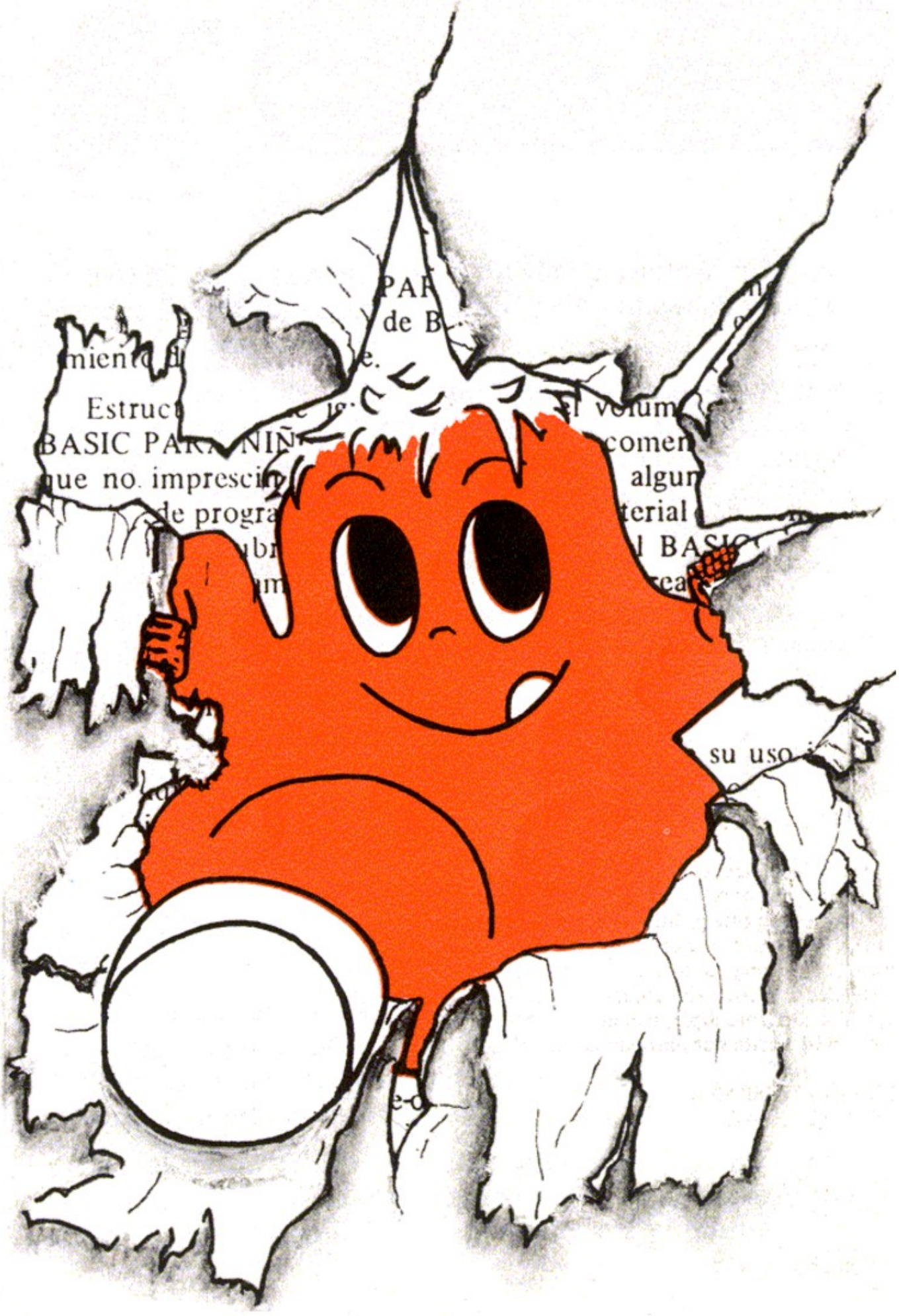
PARANINFO S.A.
1987
MADRID

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 84-283-1365-2
Depósito Legal: M - 19.413 - 1987

PARANINFO S.A. Magallanes, 25 - 28015 MADRID (04312/38/24)

ALCO, artes gráficas. Jaspe, 34. 28026 MADRID



Prólogo

BASIC AVANZADO PARA NIÑOS permite al niño que ya tiene ligeras nociones de BASIC profundizar en el conocimiento de este lenguaje.

Estructurado de igual forma que el volumen anterior, BASIC PARA NIÑOS –cuya lectura es recomendable, aunque no imprescindible si el niño ya tiene algunos conocimientos de programación–, presenta un material que facilita al niño el descubrimiento de los secretos del BASIC, avanzando progresivamente a través de su propia creatividad.

Sin olvidar la sencillez y claridad expositiva necesarias para el niño, se abordan temas y conceptos de cierto nivel en programación BASIC.

BASIC AVANZADO PARA NIÑOS permite su uso incluso a aquellos padres o educadores que tengan conocimientos mínimos de este lenguaje.

Las Notas para el Adulto que se incluyen en cada capítulo configuran una práctica y conveniente guía metodológica y didáctica para enfocar adecuadamente la comprensión y enseñanza del lenguaje BASIC.

Aquellos educadores que deseen ampliar sus conocimientos sobre el aprendizaje y la didáctica de este lenguaje, pueden consultar la interesante obra BASIC PARA MAESTROS, publicada en esta misma editorial.

No obstante el protagonista fundamental de esta obra es el niño que, gracias al enfoque dado a cada tema, estará perfectamente capacitado para aprender por sí mismo.

Indice

Unas notas previas

En cada uno de los capítulos de BASIC AVANZADO 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.

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 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.

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.

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

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*

Unas notas previas

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.

- 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.

Unas notas previas

- 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.

- Si el niño lo desea, puede intentar trabajar sin ayuda del adulto. Supervise los resultados.

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.

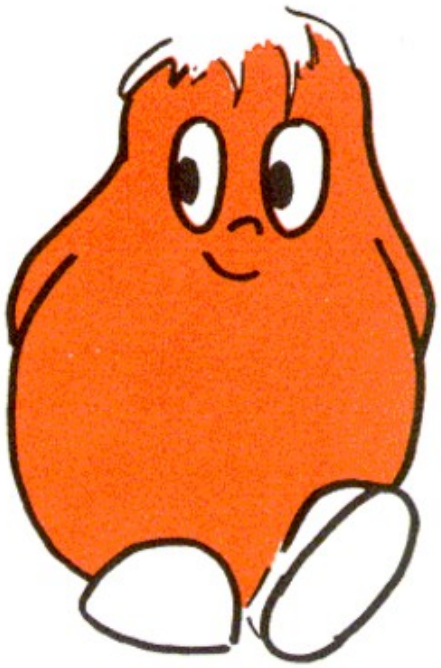
No obstante, puede ocurrir que algunos ordenadores utilicen expresiones diferentes, o distintas combinaciones de teclas para una misma función. Consulte el manual de programación de su propio ordenador cuando tenga alguna duda. Las diferencias serán mínimas y la mayor parte de ellas se comentan en las Notas para el Adulto.

HOLA. A LO MEJOR YA ME CONOCES. SOY ARTURO Y EN EL LIBRO BASIC PARA NIÑOS APRENDIMOS JUNTOS BASTANTES COSAS SOBRE EL LENGUAJE DE PROGRAMACIÓN BASIC. AHORA QUE YA ERES TODO UN PROGRAMADOR, VAMOS A APRENDER ALGUNAS COSAS MÁS QUE TE PERMITIRÁN SER CASI UN EXPERTO.

CUANDO TERMINEMOS, SABRÁS HACER PROGRAMAS MUY COMPLETOS Y ÚTILES.

SI NO APRENDISTE CONMIGO EN EL PRIMER LIBRO PORQUE YA SABÍAS ALGO, VUELVE LA PÁGINA, DONDE HAGO UN RESUMEN DE LO QUE YA SABEMOS.

¿SEGUIMOS?



Seguimos

Aquí tienes el resumen de lo que ya sabemos, para que te sirva de recordatorio:

PRINT Manda imprimir o escribir en la pantalla todo lo que le indiques.

LET Te permite dar a una variable el valor que tú quieras.

INPUT Hace que el ordenador espere un dato que tú le puedes dar a través de tu teclado mientras se está ejecutando o corriendo el programa.

GOTO Con ella le das una orden tajante al ordenador para que se dirija obligatoriamente a la línea de programa que tú le indiques.

IF ... THEN Permite que le des unas instrucciones al ordenador para que, dependiendo de si se cumplen o no unas condiciones que tú mismo le marcas, haga una cosa u otra.

Seguimos

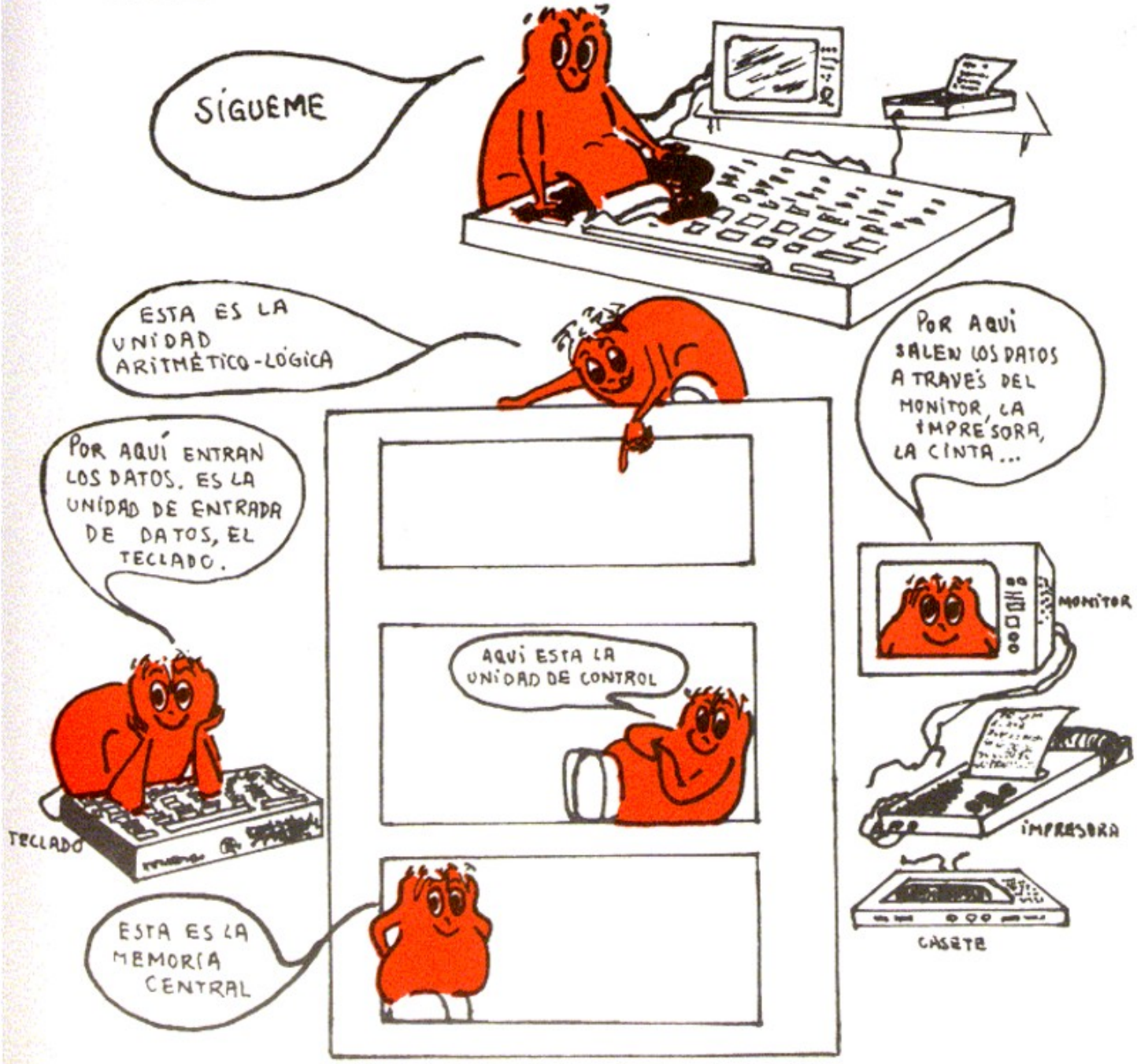
FOR/NEXT	Con esta instrucción preparamos al ordenador para efectuar un bucle y repetir una misma acción un número determinado de veces.
GOSUB	Manda al ordenador a una determinada subrutina. Con la instrucción RETURN el ordenador vuelve al programa principal.
READ/DATA	Con la instrucción DATA almacenamos en el programa unos datos que serán leídos por una instrucción READ.
REM	Te permite introducir comentarios en un programa, sin alterarlo.
INT	Quita los decimales de un número, dejando sólo la parte entera.
RND	Con ella podemos introducir el azar en un programa.
DIAGRAMAS	Nos permiten ver con un esquema cómo va a estar construido un programa.

A lo largo de este libro aprenderemos más cosas sobre estas instrucciones y también otras nuevas con las que podrás hacer programas ... ¡increíbles!

Mirando dentro

Ahora que ya sabes hacer tus propios programas vamos a analizar cómo y por qué interpreta el ordenador tus instrucciones. Es decir, como trabaja un ordenador por dentro.

Ya sabes que un ordenador memoriza todas las líneas de programa que tú le das. Entremos en el ordenador para ver dónde las guarda y qué hace con los datos que tú introduces.



Mirando dentro

ARTURO tardó algún tiempo en aprenderse estos nombres que ves en el dibujo, pero lo importante no es saberse los nombres sino entender cómo funciona cada una de estas partes del ordenador.

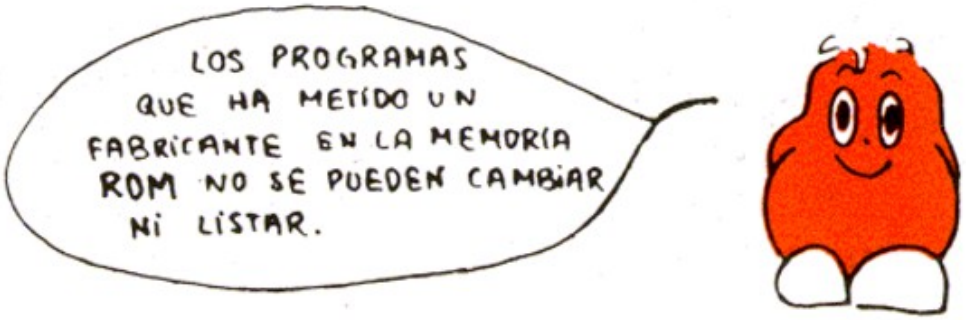
Cuando tú, a través del teclado, metes unos datos en el ordenador, éstos van directamente a la MEMORIA CENTRAL. ¿Recuerdas que al ordenador hay que decírselo casi todo y que hay que enseñarle primero lo que debe hacer para que al final el programa dé el resultado que tú querías?

Bien. Hay que decírselo casi todo, pero algunas cosas ya las sabía. Fijate bien en esto. Tú le dabas las instrucciones con unas líneas numeradas para que él las leyera por orden. Incluso podías darle las instrucciones con la numeración desordenada y él las ordenaba automáticamente. También has hecho programas para que el ordenador sume, multiplique... Pero tú no le enseñaste ni a sumar ni a poner las líneas en orden.

Dentro de la memoria central está la memoria ROM. En ella el ordenador tiene memorizados todos los datos y programas que necesita para saber contar, ordenar.... Todos esos datos los introdujo el fabricante en la memoria ROM

Mirando dentro

para que siempre estén ahí guardados. Cuando tú haces un programa en el que hay una suma, estás diciendo al ordenador "Mira en tu memoria ROM cómo hacer esta operación, y hazla".



Pero dentro de la memoria CENTRAL también existe la memoria RAM. Ella es la que se encarga de memorizar todas las instrucciones y datos que tú metes en el ordenador a través del dispositivo de entrada de datos, a través del teclado. Por eso tú puedes introducir datos y leerlos después. Porque se guardan en la memoria RAM, a la que tú puedes llegar y modificar. Cuando listas un programa, estás preguntando al ordenador: ¿qué tienes en la memoria RAM?

¡Ah! Y cuando desenchufas el ordenador, se pierde todo lo que esté almacenado en la memoria RAM, pero se sigue guardando el contenido de la memoria ROM, pues de lo contrario el ordenador no sabría como trabajar para tí cuando le pidas que haga operaciones o que interprete tus instrucciones.

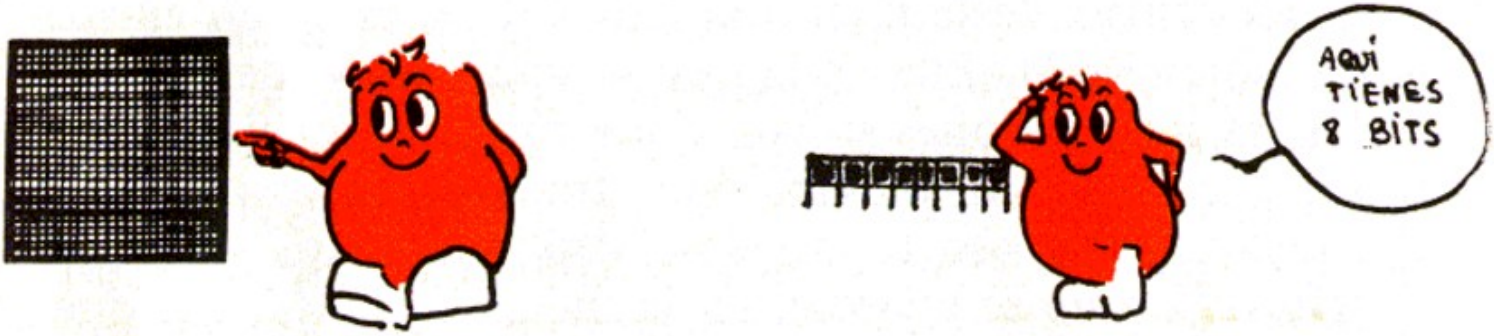
Mirando dentro

Cada modelo de ordenador tiene una capacidad de memoria diferente. Algunos ordenadores son capaces de guardar en su memoria central programas larguiiiiisimos o muchos programas y otros ordenadores no tanto.

Para saber si un ordenador tiene poca o mucha memoria, nosotros hablamos de K. Seguro que has oído decir "este ordenador tiene 32 K de memoria".

¿Qué es una K? Verás:

La memoria central por dentro podríamos verla como unas celdillas. Cada celdilla es un BIT



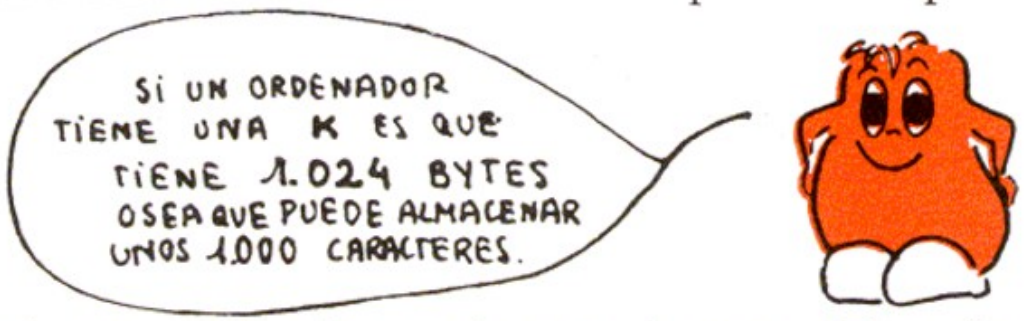
A cada conjunto de 8 bits le llamamos BYTE

En cada BYTE se almacena un carácter. Es decir, que el ordenador utiliza un BYTE, o sea, 8 bits, para cada carácter.

Mirando dentro

Así, el ordenador utilizaría 8 celdillas (8 bits) para la letra A, o la B, o para el número 2 ó para el número 7, etc.

1024 BYTES forman un KBYTE que es lo que llamamos K.



★ Pero sigamos viendo qué pasa dentro del ordenador. Los datos e instrucciones van primero a la UNIDAD CENTRAL DE MEMORIA. De ahí pasan a la UNIDAD DE CONTROL. La UNIDAD DE CONTROL se encarga de interpretar las instrucciones y órdenes que le has dado en el programa. La UNIDAD ARITMETICO LOGICA es la que hace los cálculos, las comparaciones, etc. El resultado de estos cálculos o comparaciones pasa otra vez por la UNIDAD DE CONTROL y va a parar a la MEMORIA CENTRAL, de donde va por la UNIDAD DE SALIDA al exterior. Es entonces cuando tú puedes ver en la pantalla el resultado de tu programa.

Mirando dentro

No te preocupes. Con un ejemplo lo vamos a ver en seguida. Fíjate en el programa:

```
10  LET A=5
20  LET B=3
30  LET C=A+B
40  PRINT C
50  STOP
```

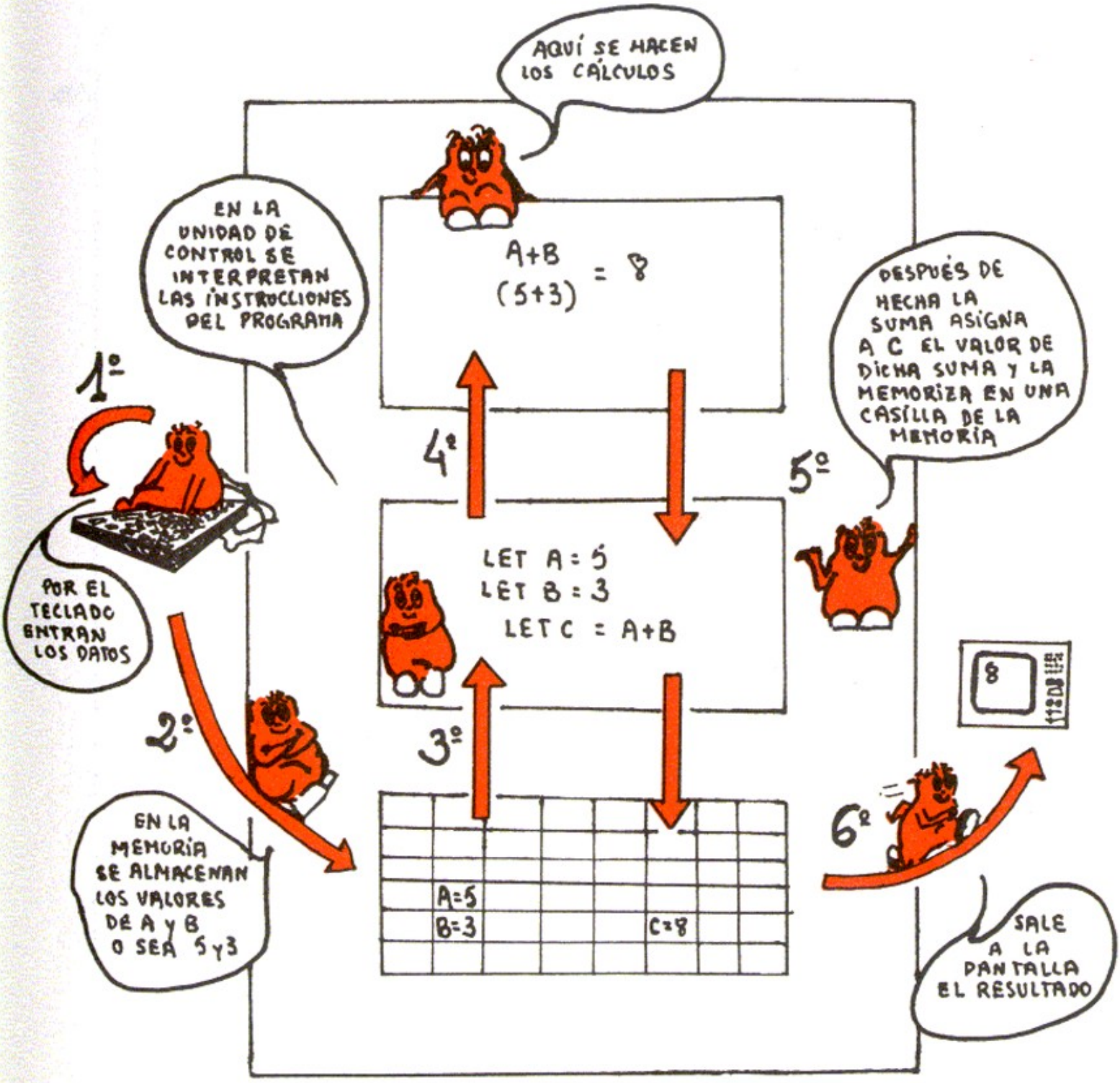
Y ahora vas a ver en el dibujo de la página siguiente, cómo este programa es interpretado y ejecutado en las distintas partes del ordenador.

No te preocupes si no entiendes muy bien todo esto. En realidad, lo importante es que tengas una idea de cómo funciona un ordenador y que cuando oigas alguna de estas palabras sepas aproximadamente de qué se está hablando.

Todo lo que puedes ver por dentro y por fuera del ordenador (cables, circuitos, teclado, pantalla, impresora, etc.) es lo que se llama HARDWARE (o equipo físico).

Los programas, las ideas... es lo que se llama SOFTWARE (o logical).

Mirando dentro



Mirando dentro

NOTAS PARA EL ADULTO

- No es importante que el niño comprenda todos los conceptos expuestos en el capítulo precedente. Se trata simplemente de que le resulten familiares.

- Siga con el niño el recorrido que marcan las flechas del dibujo de la página 22. Dibuje en un papel o en la pizarra el esquema del ordenador y pida al niño que vaya dibujando las flechas siguiendo el orden secuencial del proceso.

- Los ordenadores funcionan exclusivamente con un código binario. Es decir, sólo admiten combinaciones de dos elementos (0 y 1). Así, en un BYTE, los 8 bits pueden combinarse de 256 maneras distintas. Efectivamente $2^8 = 256$. Por ejemplo:

  el número 30 vendría representado por 00011110

  el número 211 vendría representado por 11010011

- Existen ya en el mercado microprocesadores que son capaces de procesar 16 e incluso 32 BITS a la vez, lo que les confiere una extraordinaria capacidad operativa.

La Quinta operación

ARTURO se quedó muy pensativo cuando escuchó por primera vez eso de la quinta operación. El creía que las operaciones fundamentales eran cuatro: Sumar, restar, multiplicar y dividir. Pero hay una quinta operación, que es interesantísima y que permite hacer de manera muy fácil cálculos muy complicados.

Te vamos a poner unos ejemplos sencillos. Imagínate el número

1 000 000 000 000 000 000 000

Es equivalente al número de kilómetros que nos separan de algunas estrellas. Cuando los astrónomos tienen que hacer cálculos para mandar naves espaciales, tienen que operar muchas veces con estas cifras. Esto es muy pesado y puedes cometer muchos errores. No se puede trabajar bien con tantos ceros.

La Quinta operación

Una forma más simple para escribir el mismo número sería

$$10^{21}$$

que se lee "diez elevado a 21". Es decir, que el número 10 se multiplica por sí mismo 21 veces. Al número que nos indica las veces que multiplicamos un determinado número por sí mismo (en nuestro caso, el 21) se le llama exponente. Por eso esta operación se llama *exponenciación.*

Otro ejemplo más sencillo: El número 8 también se puede escribir así: 2^3. Esto quiere decir que hay que multiplicar el número 2 tres veces por sí mismo. Es decir:

$$2 * 2 * 2 = 8$$

★ Te vamos a contar ahora una historia muy curiosa:

Cuenta la leyenda que había en la INDIA un Rey que se encontraba espantosamente aburrido. Así que mandó llamar a todos los sabios de la corte para que inventaran un juego que le divirtiera. Muchos lo intentaron sin conseguirlo.

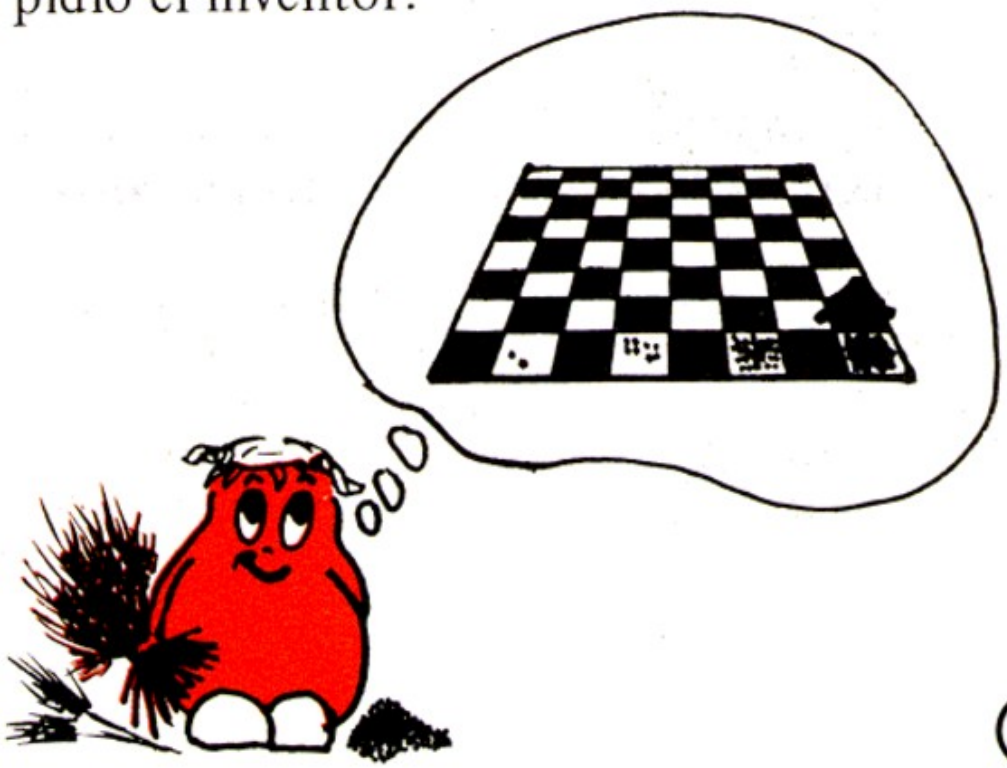
Hasta que un día apareció un personaje muy extraño que presentó al Rey un juego tan maravilloso, inteligente e interesante que el Rey quedó entusiasmado. Se trataba del juego que hoy conocemos como el juego del ajedrez.

La Quinta operación

El Rey agradecido, decidió premiar al inventor del ajedrez. Y le dijo: "Como premio a tu sabiduría, puedes pedirme lo que desees". El extraño personaje contestó: "Majestad, tan sólo quiero dos granos de trigo por la primera casilla del tablero, 4 granos por la segunda, 8 granos por la tercera, 16 granos por la cuarta.... y así sucesivamente hasta completar las 64 casillas del tablero de ajedrez".

El Rey se quedó muy contento, porque pensó que el inventor del ajedrez se conformaba con muy poco a cambio de un juego tan interesante, así que mandó llamar a su Primer Ministro y le ordenó que entregara al inventor lo que éste pedía.

Pero cuál fue su sorpresa cuando el Ministro vino a comunicarle que, aunque todo su pueblo se dedicase a cultivar trigo ... ¡no habría trigo en el mundo para reunir la cantidad que pidió el inventor!



La Quinta operación

Para que tú te hagas una idea, harían falta más o menos

18 446 744 500 000 000 000 granos de trigo

Esta cifra es tan grande, que casi ni la podemos imaginar. Si toda la tierra se dedicara a cultivar trigo, sin hacer otra cosa, se tardarían 45.000 años en reunir todo este trigo. Aunque corriéramos mucho, para contar todos los granos de trigo harían falta por lo menos 1.170 millones de siglos, o sea, 117.000 millones de años ¡INCREIBLE!

Eso es porque, como ya te habrás dado cuenta, por cada casilla el inventor pidió el doble de granos que le daban por la casilla anterior. Es decir, que multiplicaba por 2 el número de granos de la casilla anterior y así durante 64 casillas.

El enorme número anterior lo podríamos haber escrito 2^{64}. Parece mentira ¿eh?

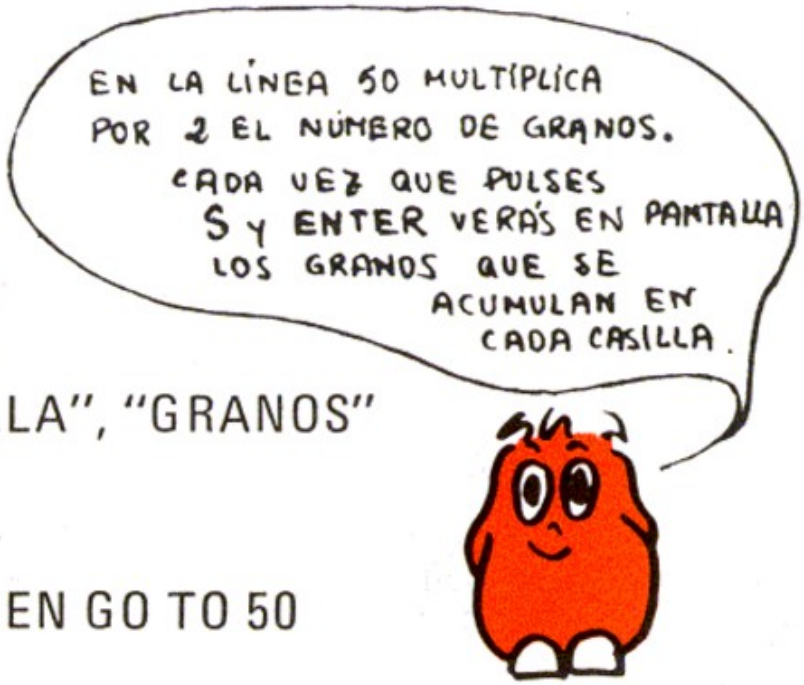
En un ordenador, 2^{64} se escribe así: 2 ↑ 64, porque en el ordenador el signo de la exponenciación es ↑.

★ Vamos a hacer un programa que reproduzca esta historia.

```
 5  REM "AJEDREZ"
10  PRINT " Pulsa S para otra casilla"
20  PRINT
```

La Quinta operación

```
30  LET C=0
40  LET A=1
50  LET A=A*2
60  LET C=C+1
70  PRINT "CASILLA", "GRANOS"
80  PRINT C, A
90  INPUT S$
100 IF S$="S" THEN GO TO 50
```



No hará falta que te expliquemos cómo funciona este programa pues todas las instrucciones son muy sencillas y ya las aprendiste en el libro BASIC PARA NIÑOS.

También podríamos haber escrito este programa utilizando un bucle FOR/NEXT.

```
5   REM "AJEDREZ"
10  PRINT "PULSA S PARA OTRA CASILLA"
20  PRINT
30  LET A=1
40  FOR C=1 TO 64
50  LET A=2↑C
60  PRINT "CASILLA", " GRANOS"
70  PRINT C, A
80  INPUT S$
90  IF S$="S" THEN GO TO 110
```

La Quinta operación

```
100  STOP
110  NEXT C
```

Ejecuta el programa. Verás como la cantidad de granos es cada vez mayor y, cuando llegues más o menos a la casilla 30, el ordenador escribe unas cifras un poco más "raras".

En la línea 30, por ejemplo, te aparecerá en pantalla:

1.07374182E + 9

Eso significa que el ordenador nos está diciendo *en su lenguaje* que también utiliza una forma simplificada para escribir un número grande.

El símbolo E + 9 es igual que decir que tienes que correr la coma 9 números hacia la derecha. Es decir, que si multiplicas 9 veces por 10 el número 1,07374182 obtendrás el número de granos que hay en la casilla 30 del tablero de ajedrez.

Casilla	*Expresión Matemática*	*Expresión Informática*	*Número*
30	2^{30}	2 ↑ 30 (Como tú lo escribirías) 1.07374182E +9 (Como te lo dice el computador)	1.073.741.820

La Quinta operación

Cuando llegues al final del programa, verás que el ordenador te da (utilizando su forma de expresarse) el número total de granos que corresponden a las 64 casillas.

El mismo resultado, pero sin verlo casilla a casilla, puedes verlo tecleando simplemente:

PRINT 2 ↑ 64

Como ves la exponenciación es muy interesante para manejar números grandes.

NOTAS PARA EL ADULTO

- Es muy probable que el niño no se sienta especialmente motivado hacia este tema. De ahí la importancia de explicárselo utilizando la leyenda del juego del ajedrez. Es probable que algunos niños no sean capaces de asimilar el concepto de magnitud numérica que se ha considerado en el ejemplo. No tiene importancia, como tampoco la tiene el que no comprendan del todo el concepto de exponencial.

La Quinta operación

- Lo hemos incluido en este capítulo para que el niño lo conozca y no le resulte nuevo cuando mencionemos la operacion de exponenciación dentro del capítulo dedicado a los operadores aritméticos.

- El desconocimiento del concepto exponencial no afectará en absoluto la comprensión de los temas que se tratarán a lo largo de este libro.

- Haga ver al niño que la expresión gráfica que utilizará el ordenador para indicar una exponenciación es E + ... mientras que la que el niño debe utilizar a través del teclado es ↑.

- No se han comentado los programas pues las instrucciones que aparecen en ellos son sobradamente conocidas por el niño. Si ha notado alguna dificultad de comprensión, repasen las instrucciones que se suponen ya conocidas.

- Si a pesar de la utilización de los exponentes un número resulta demasiado grande para la capacidad del ordenador, éste nos dará un mensaje del tipo

  ?OV ERROR o NUMBER TOO BIG o similar.

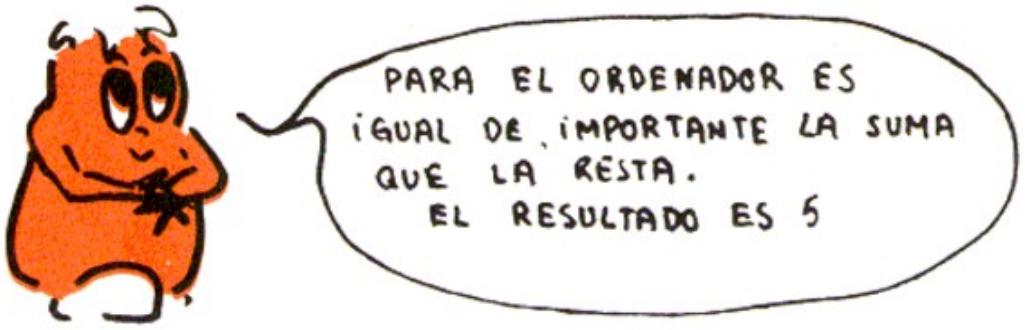
Operadores aritméticos

Ya sabes que la máquina es capaz de operar si tú se lo pides en una instrucción de tu programa. Cuando tú haces un programa, tienes mucho cuidado con el orden de las líneas, pues ése es el orden que leerá el ordenador.

Bien. Lo mismo tienes que hacer cuando quieras que el ordenador efectúe una operación. No puedes escribir las sumas, restas, multiplicaciones .... todo mezclado y sin orden. El ordenador es muy metódico y siempre sigue un mismo orden a la hora de operar.

★ Vamos a ver como operaría con la siguiente expresión:

```
PRINT 3 – 2 +8 – 4
```



El ordenador sumará y restará los términos uno tras otro en el orden en que tú los has escrito.

Operadores aritméticos

Pero vamos a hacer un cambio:

PRINT 3 – 2 +8/4

Para el ordenador es mucho más importante la división que la suma o la resta, así que lo primero que hará es dividir los números a cada lado de la barra de dividir y luego sumará y restará.

★ Como has visto, existen unas reglas que el ordenador sigue. Son muy fáciles: Supongamos que tenemos un problema matemático en el que aparecen todas las operaciones (Suma, resta, multiplicación, división y exponenciación).

El ordenador

PRIMERO hace todas las exponenciaciones, de izquierda a derecha

SEGUNDO hace todas las multiplicaciones y divisiones, empezando por la izquierda

TERCERO hace todas las sumas y restas, empezando por la izquierda.

Operadores aritméticos

Pero cuando se encuentre con algo escrito entre paréntesis, hará primero todas las operaciones que estén dentro del paréntesis (y por el orden que él conoce) y luego seguirá actuando con las restantes operaciones siguiendo la regla que te hemos dicho antes.

★ Suponte que tú no quieres dividir 8 entre 4 y al resultado sumarle 3 y restarle 2, sino que quieres saber el resultado de 3 menos 2 mas 8 y todo eso dividirlo por 4.

Como tienes que indicarle cuál es el orden en que tú quieres que opere, escribirías:

```
PRINT (3 – 2 +8)/4
```

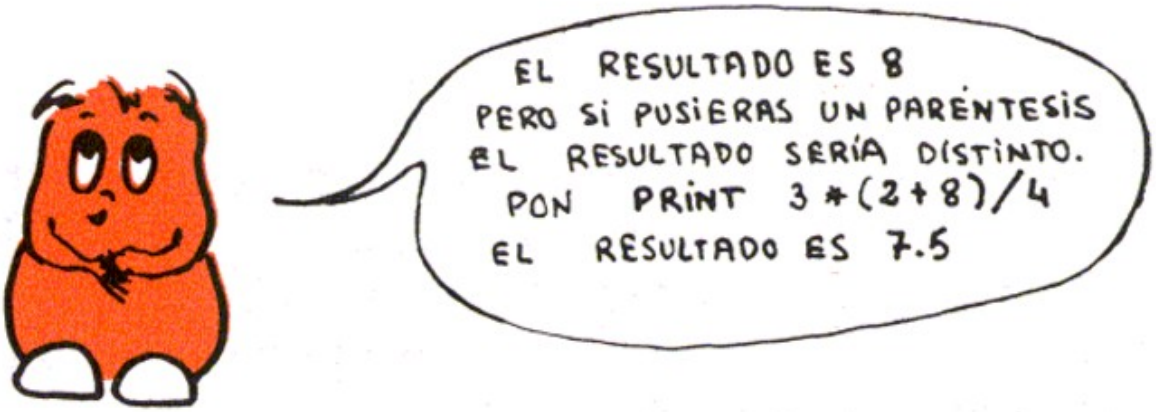
El ordenador hará primero siempre lo que se encuentre entre paréntesis ( ), así que el resultado ahora es 2.25.

★ Hagamos de nuevo otro cambio:

```
PRINT 3 * 2 +8 /4
```

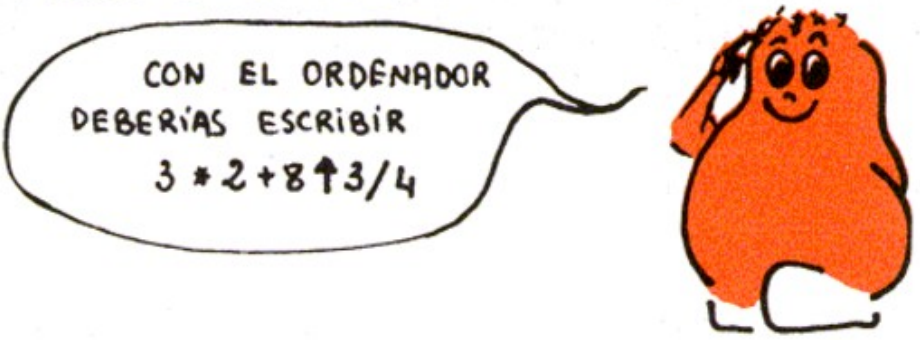
El ordenador multiplicará 3 por 2 y luego dividirá 8 entre 4. Al resultado de la multiplicación le sumará el resultado de la división. En este caso, el ordenador multiplica antes de dividir porque, en principio, el ordenador lee de izquierda a derecha.

Operadores aritméticos



★ Hagamos un nuevo cálculo:

PRINT $3 * 2 + 8^3/4$



Para el ordenador es más importante la exponenciación que la suma, multiplicación o división, por lo que primero calcula 8^3 (que es 512). Después, siguiendo el orden de izquierda a derecha, actúa así:

$\underset{\sim\sim}{3 * 2} + \underset{\sim\sim}{512/4}$	Primero multiplica 3 por 2 y luego divide 512 entre 4.
$6 + 128$	Finalmente sumará los resultados obtenidos en las operaciones anteriores.

El resultado final es 134.

Operadores aritméticos

★ Otro ejemplo, ahora con paréntesis:

```
PRINT 3 * (2 +8^3)/4
```

Primero eleva 8 a 3
Al resultado le suma 2
El resultado lo multiplica por 3
El resultado lo divide entre 4

Ten en cuenta que al escribir esto en tu ordenador lo harías así:

```
PRINT 3 * (2 +8 ↑ 3)/4
```

Sigue paso a paso el orden de las operaciones y verás como esta vez el ordenador calcula primero lo que está entre paréntesis y luego sigue las reglas que ya sabemos.

★ ¿Qué pasaría si cambiamos el paréntesis de sitio? Escribe:

```
PRINT 3 * (2 +8)^3/4
```

Primero suma 2 más 8
El resultado lo eleva a 3
El resultado lo multiplica por 3
El resultado lo divide entre 4

Operadores aritméticos

Como te habrás dado cuenta, dependiendo de los números y operaciones que encerremos dentro de los paréntesis, el resultado será distinto. Comprenderás la importancia que tiene que tengas muy claro las operaciones que quieres hacer antes de escribirlas de forma "informática". De lo contrario el ordenador te dará un resultado falso.

★ Fíjate en un ejemplo. ARTURO tenía 15 caramelos pero ha perdido 3 y los que le quedan tiene que repartirlos entre sus 6 amigos. Si escribe en el ordenador:

```
PRINT 15 – 3/6
```

el ordenador, siguiendo las reglas que conoces, daría como resultado 14.5. Como ves esto no puede ser. ARTURO debería escribir:

```
PRINT (15 – 3)/6
```

Veámoslo con un programa y utilizando además variables en lugar de números.

```
10 PRINT "CUANTOS CARAMELOS TIENE ARTURO?"
20 INPUT A
30 PRINT "CUANTOS PERDIO?"
40 INPUT B
```

Operadores aritméticos

```
50  PRINT "CUANTOS AMIGOS TIENE?"
60  INPUT C
70  LET X =( A– B)/C
80  PRINT " ARTURO DA "; X;"CARAMELOS A CA-
    DA UNO DE SUS"; C; "AMIGOS"
```

Fíjate en la importancia de escribir correctamente la línea 70. Cámbiala por:

```
70 LET X = A – B/C
```

y verás cómo el ordenador te da una respuesta absurda.

NOTAS PARA EL ADULTO

- La jerarquía de los operadores aritméticos es en realidad muy simple y fácil de comprender. Quizás resulte más difícil de explicar que de entender.

- El niño no debe retraerse ante la aparente profusión de operaciones. Son todas extraordinariamente sencillas y basta con practicar un poco para dominar el tema.

  Realicen conjuntamente distintas operaciones y cálculos en orden creciente de dificultad.

Operadores aritméticos

- Haga ver al niño la importancia de conocer claramente a priori el cálculo que se desea realizar, a fin de que pueda transcribirlo correctamente en su expresión informática. Lo ideal es escribirlo previamente en un papel.

- Comente con el niño la ventaja de escribir un programa como el que se incluye en este capítulo en lugar de teclear directamente PRINT (15 – 3)/6, lo cual evidentemente nos daría el mismo resultado. La utilización de programas con variables y la introducción de los valores a través de instrucciones INPUT permite la utilización de un mismo programa para diferentes casos.

- Si lo considera oportuno o necesario, tenga en cuenta que es posible utilizar expresiones como ésta:

$$(3 * (2 + 8) \uparrow 3)/4$$

en las que aparecen paréntesis externos e internos. El ordenador en estos casos dará siempre prioridad a los paréntesis internos.

CLEAR y CLS

Ya sabes que cada vez que empiezas un nuevo programa debes borrar todos los datos que el ordenador tiene en su memoria.

Generalmente habrás utilizado el comando **NEW**. Con ello el ordenador quedaba listo para volver a escribir un nuevo programa. Es decir, borrábamos de la memoria todos los datos y programas que el ordenador tenía memorizados.

Vamos a probar algunas cosas nuevas. Teclea este programa:

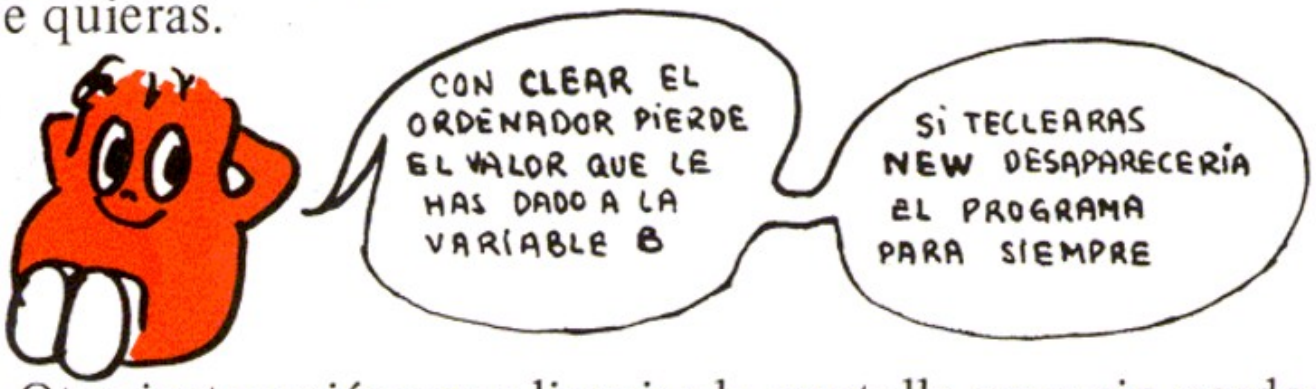
```
10  LET A=5
20  PRINT "DIME UN NUMERO"
30  INPUT B
40  LET M=A*B
50  PRINT "EL RESULTADO ES"; M
```

Ahora tienes en la pantalla este pequeño programa. Teclea el comando **CLEAR** y verás qué pasa. El programa ha desaparecido de la pantalla.

CLEAR y CLS

Ahora teclea **LIST**. El programa vuelve a estar en pantalla. Esto ocurre porque el comando **CLEAR** no borra el programa.

Haz correr el programa y dale a B el valor 6, por ejemplo. En la pantalla podrás ver el resultado. Teclea **CLEAR** y luego lístalo. Fíjate en el programa. Está igual que al principio. A sigue valiendo 5 pero B no vale 6, sino que cuando ejecutes el programa otra vez, podrás darle a B el valor que quieras.



★ Otra instrucción para limpiar la pantalla pero sin perder el programa es **CLS**. Pero mientras que **NEW** y **CLEAR** eran comandos, es decir, órdenes directas sin línea de programa, **CLS** es una sentencia y debe ir incluida en una línea de programa. Cuando en la ejecución de un programa el ordenador llega a una línea con la instrucción **CLS**, borrará todo lo que en ese momento esté escrito en la pantalla. Añade al programa la línea:

```
45  CLS
```

Corre el programa y compara las diferencias.

CLEAR y CLS

Por eso es conveniente que al principio de cada programa la instrucción de tu primera línea sea **CLS.** Así tendrás de manera automática la pantalla limpia en cuanto lo ejecutes. Añade a nuestro programa: 5 CLS

NOTAS PARA EL ADULTO

- Practique con el niño las sentencias y comandos explicados hasta conseguir su dominio.
- En nuestro programa, B es una variable y A es una constante pues su valor lo hemos determinado nosotros mismos en la línea 10 del programa.
- Tenga en cuenta que:

  El comando **RUN** pone las variables a cero y borra o no la pantalla según el tipo de ordenador.

  El comando **NEW** borra el programa de la memoria RAM.

  El comando **CLEAR** borra la pantalla pero no la memoria, aunque pone las variables a cero.

  La instrucción **CLS** borra la pantalla pero no altera las variables ni borra la memoria.
- Conviene que a partir de ahora el niño se acostumbre a utilizar de forma sistemática la instrucción **CLS.**

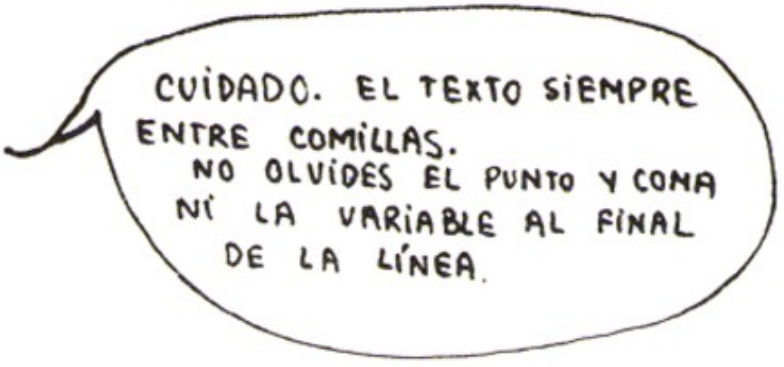
INPUT (Algo más)

Seguro que ya conoces para qué y cómo empleamos la instrucción **INPUT**. Hasta ahora habíamos combinado **PRINT** e **INPUT** para poder "hablar" con el ordenador. Vamos a ver ahora cómo **INPUT** nos permite ahorrarnos líneas de programa. Teclea de nuevo el programa de la página 41. Ahora cambia la línea 20 y escribe:

```
20  INPUT "ESTOY ESPERANDO UN NUMERO"; B
```

Borra la línea 30. Al ejecutar el programa te sale en pantalla exactamente lo mismo que antes. Has combinado en una sola línea las funciones que harían **PRINT** e **INPUT**. El programa completo te quedaría así:

```
 5  CLS
10  LET A=5
20  INPUT "ESTOY ESPERANDO UN NUMERO"; B
30  LET M=A * B
40  PRINT "EL RESULTADO ES"; M
```



INPUT (Algo más)

Vamos a hacer un programa para practicar esta forma de utilizar **INPUT**. Ayudemos a ARTURO a completar su ficha de curso:

```
 5  CLS
10  INPUT " NOMBRE"; N$
20  INPUT "APELLIDO"; A$
25  INPUT "CURSO"; C
30  INPUT " CALLE"; C$
40  CLS
50  PRINT N$; " "; A$
60  PRINT "DE"; C; "CURSO"
70  PRINT "VIVE EN";  " "; C$
80  END
```

Pero todavía podemos hacer el programa más corto, pues **INPUT** nos permite que el ordenador nos pida más de una variable en una misma línea de programa. El programa anterior lo podríamos escribir:

```
 5  CLS
10  INPUT "DIME TU NOMBRE, APELLIDO, CURSO y
    CALLE"; N$, A$, C, C$
```

INPUT (Algo más)

```
15  CLS
20  PRINT "DATOS PERSONALES"
30  PRINT
40  PRINT N$;" ";A$;" QUE ESTUDIA";C;"CURSO
    VIVE EN ";C$
```

Como ves, **INPUT** puede resultar muy cómodo. Observa una cosa curiosa de la línea 40. Entre N$ y A$ hemos dejado un espacio en blanco para que los textos asignados a cada una de estas variables queden bien separados.

Cuando en lugar de separar dos textos vayan a separar una variable numérica y un texto entre comillas, simplemente deja un espacio en blanco dentro de las comillas, como lo has venido haciendo hasta ahora.

INPUT (Algo más)

NOTAS PARA EL ADULTO

- Practique con el niño esta forma de utilizar **INPUT** y hágale ver que, dentro de una misma línea, puede hacer que el ordenador espere datos numéricos o alfanuméricos. Las variables deben ir separadas por comas.

- Practiquen igualmente la colocación de espacios dentro de las sentencias **PRINT,** a fin de evitar que se junten los textos. No todos los ordenadores actúan igual en este sentido pues algunos de ellos dejan automáticamente espacios en blanco a cada lado de un valor o dato numérico.

FOR/NEXT (Algo más)

Trabajaremos un poco sobre esta instrucción, que tú ya conoces. Usar **FOR/NEXT** en un programa es tan útil y tan cómodo, que conviene que te acostumbres a ella y la practiques mucho.

A partir de ahora, vamos a utilizarla con frecuencia, así que antes vamos a repasar un poco cómo funciona esta instrucción y de paso aprenderemos algunos trucos para aprovecharla mejor.

Teclea el programa:

```
10  FOR X=1 TO 3
20  PRINT X
30  NEXT X
```

Este programa es tan sencillo, que no hace falta que te lo expliquemos. Puedes escribirlo también así:

```
10  FOR X=1 TO 3 : PRINT X : NEXT X
```

FOR/NEXT (Algo más)

Por si no lo sabías, te diremos que dentro de una misma línea de programa se le pueden dar varias instrucciones al ordenador. Lo único que hay que hacer es separar una instrucción de otra por dos puntos (:). Esto lo puedes hacer con cualquier instrucción.

★ Una forma más completa de escribir **FOR/NEXT** sería:

```
10  FOR X=1 TO10 STEP 2
```

Esto quiere decir que X tomará valores de 1 hasta 10 pero de dos en dos. Si pusieras **STEP** 3 los tomaría de tres en tres y así sucesivamente.

★ Como otras instrucciones en BASIC, **FOR/NEXT** permite muchas combinaciones con otras instrucciones para facilitar la programación. Por ejemplo, escribe:

```
10  INPUT A
20  INPUT B
30  INPUT C
40  FOR X=A TO B STEP C
50  PRINT X
60  NEXT X
```

FOR/NEXT (Algo más)

Corre el programa y dale a las variables A, B y C el valor que tú quieras; por ejemplo, 0, 20 y 3. Tendrás una lista de números que empezará en 0 y escribirá valores de tres en tres. Fíjate que el último valor que escribe es 18. De acuerdo a la instrucción **STEP,** el próximo valor a escribir sería 21, pero tú le has dicho al ordenador que el último valor de X será 20, así que después de 18 ya no toma más valores.

★ Otra cosa que hay que tener en cuenta es que, si lo necesitas, puedes decirle al ordenador que se salga del bucle **FOR/NEXT**. Añade al programa anterior:

```
45  IF X=9 THEN GOTO 100
100 PRINT "ME HE SALIDO DEL BUCLE PORQUE
          X HA TOMADO EL VALOR 9"
```

Es decir, que con instrucciones del tipo **GO TO** o **IF .... THEN** puedes salir del bucle. Con ello el bucle queda interrumpido. Lo que no debes hacer es intentar que el ordenador entre en un bucle desde fuera del bucle, porque te surgirá algún error.

★ Sigamos viendo formas de utilizar **FOR/NEXT**. Generalmente, dentro de un bucle tú metes una serie de instrucciones que quieres que el ordenador te repita un número de veces determinado.

FOR/NEXT (Algo más)

Pues .... no pasa nada. Es decir, que tendríamos un bucle vacío. ARTURO no entendía muy bien para qué podíamos querer nosotros un bucle vacío, que no hiciera nada. La verdad es que tiene su utilidad. Aunque un bucle esté vacío, mientras lo está ejecutando el ordenador no hace otra cosa. Así que podemos utilizar un bucle así para retardar una acción del programa. Por ejemplo, teclea:

```
 5 CLS
10 PRINT "ESTA LINEA SE BORRARA DENTRO DE
   UN MOMENTO"
20 CLS
30 PRINT "YA SE HA BORRADO"
```

Cuando ejecutes el programa, no te dará tiempo a ver en la pantalla el primer texto porque el ordenador, después de escribirlo en pantalla, sigue leyendo la línea 20 y te lo borra. Conviene que introduzcas en este programa un bucle vacío para retrasar la lectura de la línea 20 por el ordenador.

FOR/NEXT (Algo más)

Añade:

```
12  FOR X=1 TO 3000
13  NEXT X
```

Corre el programa. ¡Perfecto! Estos bucles se llaman también *bucles de retardo.* Puedes controlar el tiempo de retardo cambiando la instrucción **FOR**. Por ejemplo FOR X = 1 TO 1000, o lo que tú quieras.

Te recuerdo que en lugar de las líneas 12 y 13 podrías haber escrito una sola línea:

```
12  FOR X=1 TO 3000 : NEXT X
```

★ Vamos a ver otro ejemplo:

```
10  FOR X=1 TO 10
20  PRINT "PROBANDO BUCLES"
30  NEXT X
```

Al correr el programa, en pantalla te aparecerá 10 veces el texto de la línea 20.

FOR/NEXT (Algo más)

Añade ahora:

```
25  FOR Y=1 TO 2000
26  NEXT Y
```

Te aparecerá el mismo texto en pantalla, pero poco a poco, porque después de ejecutar la línea 20, el ordenador ejecuta el bucle de retardo antes de leer la línea 30 y tomar el siguiente valor de X.

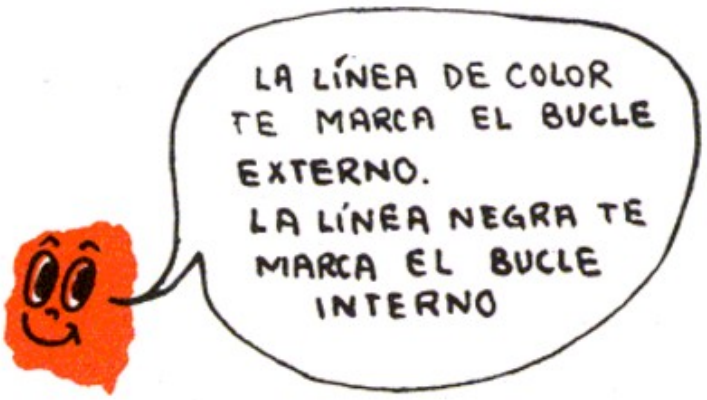
★ Casi sin darnos cuenta, hemos visto algo muy importante: Hemos metido un bucle dentro de otro bucle. Estos bucles se llaman bucles internos o, mejor, *bucles anidados.* ¿Por qué anidados? Porque el bucle de fuera es como un nido *dentro* del cuál está metido el bucle interno.

Y otra cosa muy importante: En nuestro ejemplo, el ordenador ejecuta primero todo el bucle de retardo (el interno) *antes* de tomar el siguiente valor del bucle de fuera o externo.

FOR/NEXT (Algo más)

★ ¿Qué pasaría si el bucle anidado no fuera un bucle vacío sino un bucle normal? Vamos a verlo con un ejemplo. Teclea el programa:

```
10  FOR A=1 TO 2
20  FOR B=1 TO 4
30  PRINT A, B
40  NEXT B
50  NEXT A
```



El bucle de fuera (externo) es el de las líneas 10 y 50. El bucle anidado (interno) es el de las líneas 20, 30 y 40. Fíjate en el importante detalle de que primero pones **NEXT** B (para cerrar el bucle interno) y luego **NEXT** A (para cerrar el bucle externo).

No te equivoques haciéndolo al revés, pues el programa no saldría. Veamos como funciona el programa. Sigue despacio y atentamente las explicaciones.

En la línea 10 le da el primer valor a A (A = 1).

En la línea 20 le da el primer valor a B (B = 1) y empieza a ejecutar el bucle anidado. A vale 1.

En la línea 30 escribe estos valores.

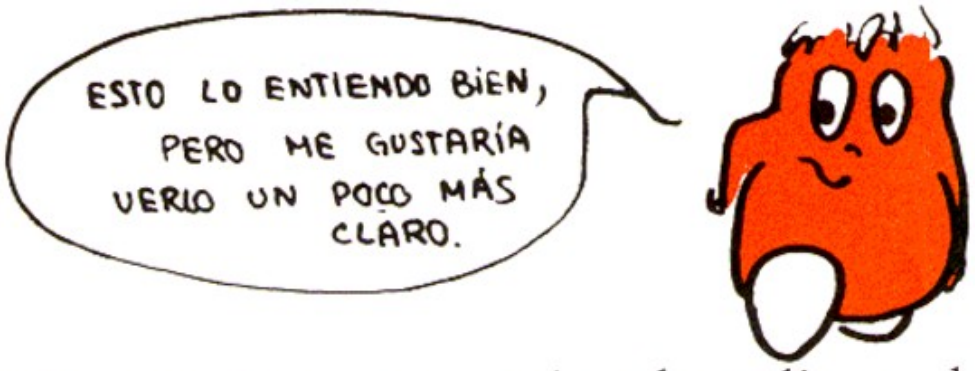
En la línea 40 le dices al ordenador que siga ejecutando el bucle interno y tome el siguiente valor de B (B = 2). En este momento A todavía vale 1.

FOR/NEXT (Algo más)

Como se está ejecutando el bucle interno, el ordenador vuelve a la línea 30 e imprime ahora los valores que tiene: (A = 1) y (B = 2).

De esta forma el ordenador sigue ejecutando el bucle interno hasta dar a la variable B todos los valores. Cuando B = 4, el ordenador ha terminado el bucle interno y llega a la línea 50, donde toma el siguiente valor de A. Ahora A vale 2.

Entonces, con este valor, vuelve a repetir todo el proceso del bucle interno. Así una y otra vez hasta terminar con todos los valores de A.



Para verlo más claro, ¿por qué no le pedimos al ordenador que nos ayude? Cambia la línea 30:

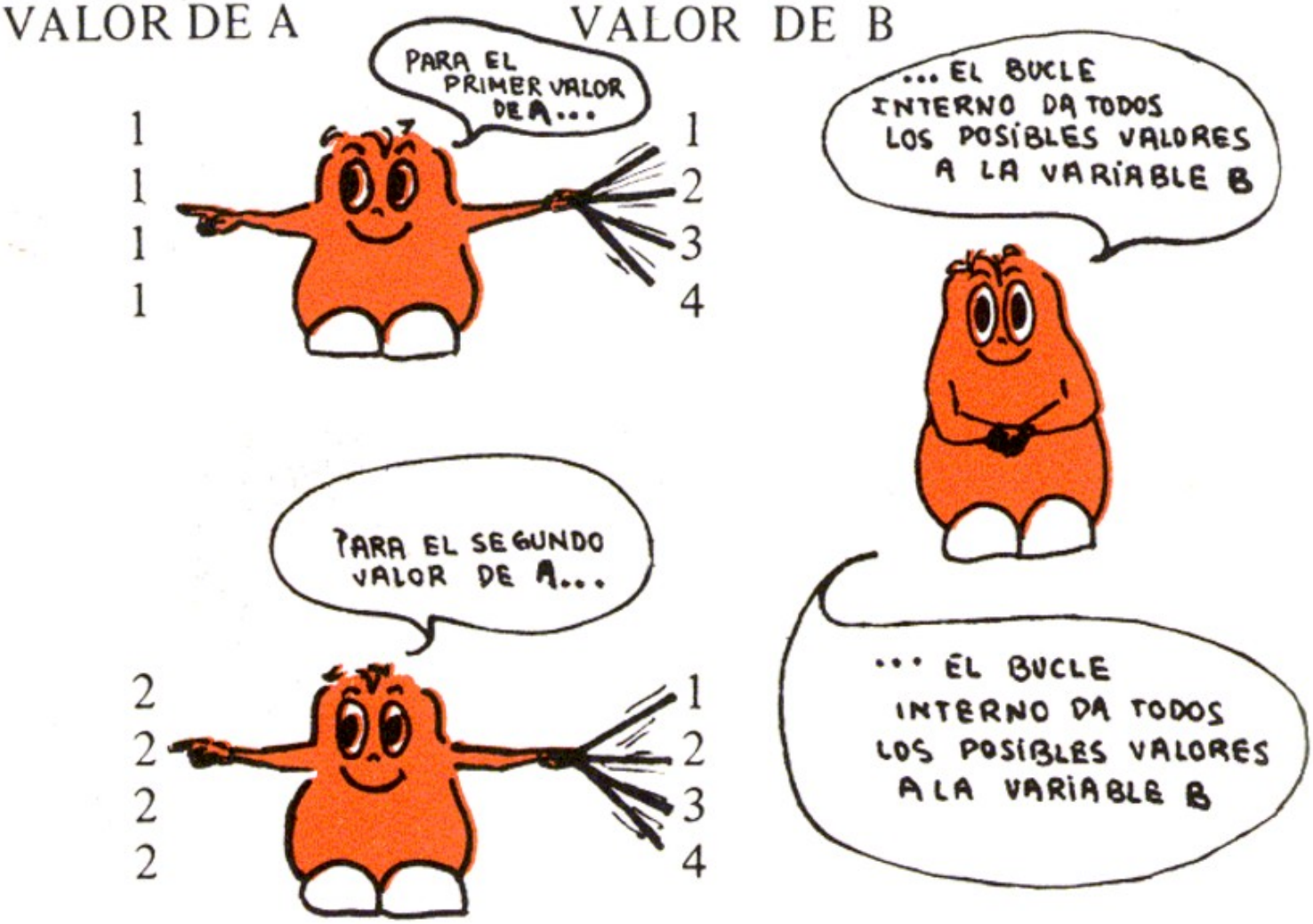
```
30  PRINT " A VALE"; A; "... AHORA B VALE"; B
```

Añade la línea:

```
35  IF B =4 THEN PRINT "TERMINADO EL BUCLE
    INTERNO PARA EL VALOR"; A; "DE LA VARIA-
    BLE A"
```

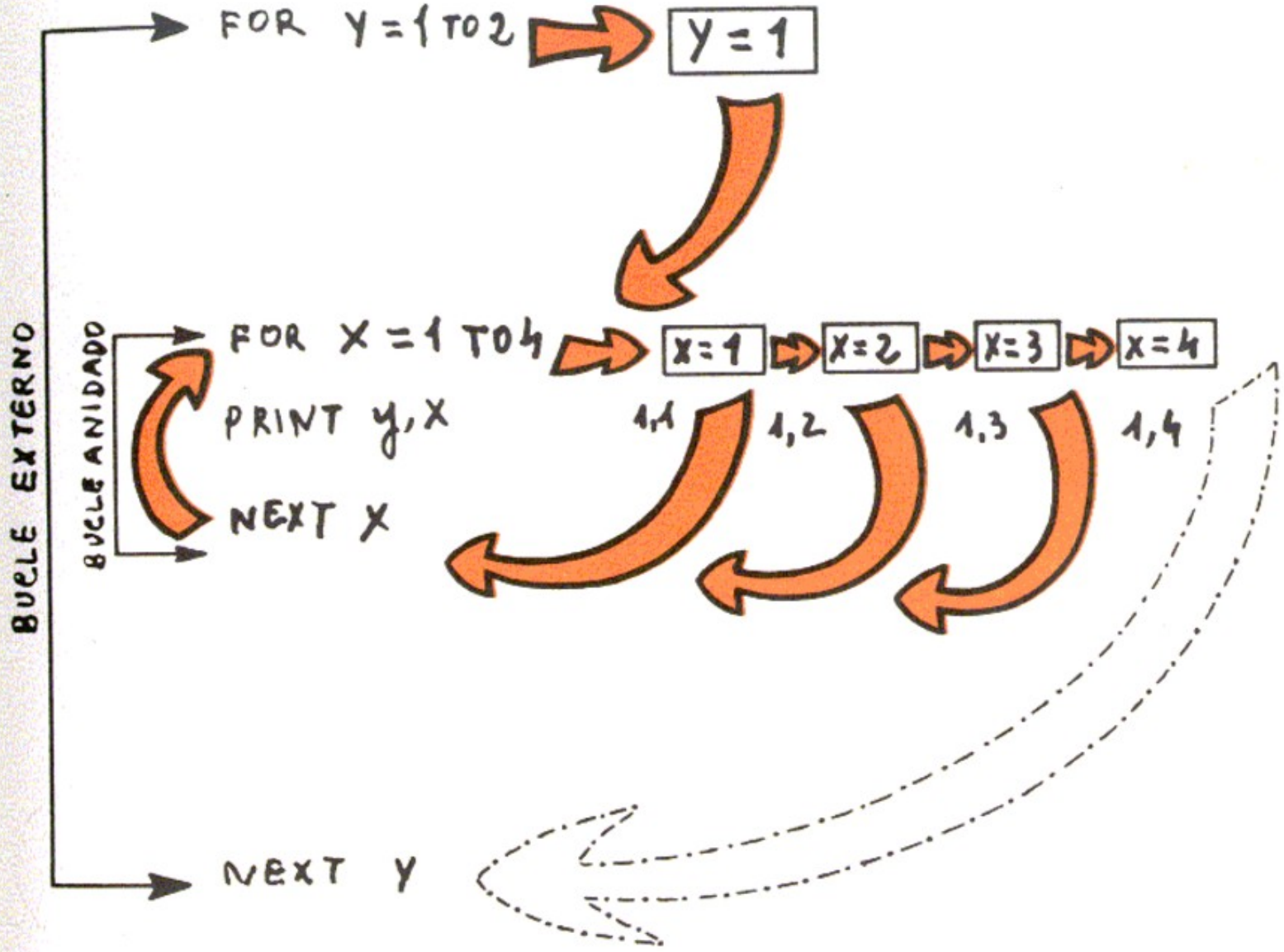
FOR/NEXT (Algo más)

Corre el programa. En pantalla tienes los distintos valores que van tomando uno detrás de otro las variables A y B.



Fíjate ahora en el dibujo de la página siguiente. Verás el recorrido que hace el ordenador cuando encuentra un bucle anidado.

FOR/NEXT (Algo más)



Vuelve a leer las explicaciones mirando la pantalla y el programa. Comprobarás que es fácil de entender.

★ Veamos algunos programas prácticos donde aplicar los bucles anidados. Por ejemplo, para obtener una lista de las distintas formas de leer las fichas de dominó, teclea:

FOR/NEXT (Algo más)

```
10  FOR X=0 TO 6
20  FOR Y=0 TO 6
30  PRINT X;" ! ";Y
40  NEXT Y
50  NEXT X
```

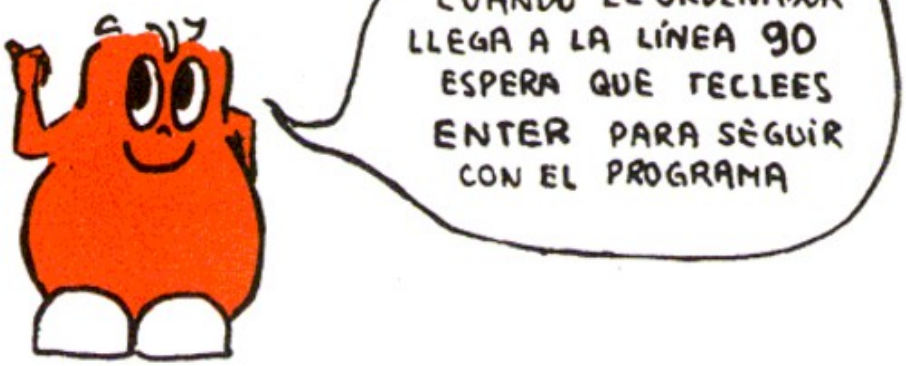
Para evitar que te aparezcan los números demasiado deprisa en la pantalla, añade el bucle de retardo:

```
35  FOR A=1 TO 500 : NEXT A
```

y los verás aparecer uno a uno.

★ Una forma de tener la tabla de multiplicar sería:

```
10  CLS
20  PRINT "TABLA DE MULTIPLICAR"
30  PRINT "PULSA ENTER DESPUES DE CADA MUL-
    TIPLICACION"
40  FOR A=1 TO 2500 :NEXT A : CLS
50  FOR X=1 TO 10
60  FOR Y=1 TO 10
70  LET C=X * Y
80  PRINT X;"POR";Y;"ES";C
90  INPUT E$
100 CLS
110 NEXT Y
120 NEXT X
```



FOR/NEXT (Algo más)

Fíjate como en la línea 40 introducimos un bucle de retardo para retrasar el borrado de la pantalla. El bucle externo se indica en las líneas 50 y 120. El bucle anidado se indica en las líneas 60 y 110.

Atención al truco de la línea 90. Con ella obligamos al ordenador a pararse esperando un dato. Así evitamos que la pantalla se nos llene con la tabla de multiplicar. En cuanto le introducimos el dato (da igual el que sea, basta con pulsar **ENTER**), el ordenador sigue con el programa. En el capítulo de **"CADENAS..."** veremos una forma más correcta de conseguir el mismo efecto utilizando **INKEY$**.

Ejecuta el programa.

Bueno, por ahora es suficiente. Como ves, se pueden hacer muchas cosas con los bucles. Incluso, organizar una liga de baloncesto con los amigos de ARTURO. ¿Te atreverías a ayudarle? Si quieres ver cómo se hace, mira el programa de la página 146.

FOR/NEXT (Algo más)

NOTAS PARA EL ADULTO

- Es conveniente que el niño practique esta instrucción, por su extraordinaria versatilidad y su constante utilización en BASIC.
- Hágale ver que se pueden introducir infinitos bucles anidados dentro de un programa, pero que es imprescindible una adecuada colocación de las sentencias **NEXT,** pues de lo contrario los bucles se cruzarían, lo cuál daría error.

CORRECTO

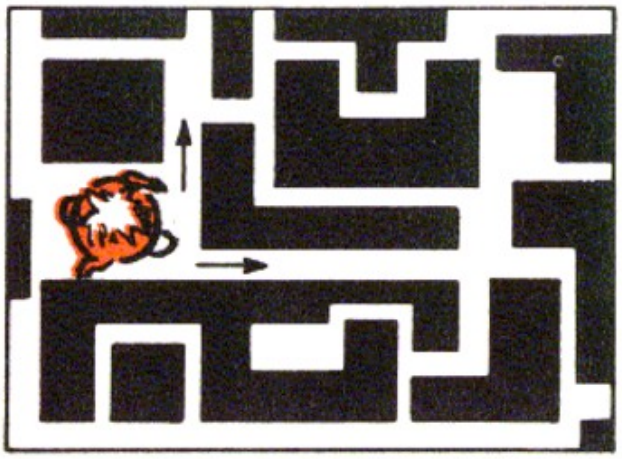
```
10  FOR A=0 TO 10
20  - - - - - - - -
30  - - - - - - - -
40  FOR B =0 TO 5
50  - - - - - - - -
60  - - - - - - - -
70  FOR C =0 TO
    100 STEP 5
80  - - - - - - - -
90  - - - - - - - -
100 NEXT C
110 NEXT B
120 NEXT A
```

INCORRECTO

```
10  FOR A=0 TO 10
20  - - - - - - - -
30  - - - - - - - -
40  FOR B =0 TO 5
50  - - - - - - - -
60  - - - - - - - -
70  FOR C =0 TO
    100 STEP 5
80  - - - - - - - -
90  - - - - - - - -
100 NEXT A
110 NEXT B
120 NEXT C
```

- En los bucles de retardo hay que tener en cuenta que no todos los ordenadores tienen la misma velocidad de ejecución, por lo que un bucle del tipo FOR X = 1 TO 1000, puede retrasar la ejecución del programa más o menos tiempo en función del tipo de ordenador.

Operadores lógicos



Tú recuerdas que utilizando **IF THEN** le poníamos unas condiciones al ordenador. Por ejemplo, decíamos cosas como éstas:

```
IF A=B THEN GO TO 1
IF M>5 THEN PRINT " CORRECTO"
IF A$<>"ARTURO" THEN STOP
```

Te habrás dado cuenta que en cada una de estas líneas de programa sólo le poníamos una condición al ordenador. El BASIC, sin embargo, te permite ponerle más de una condición, con lo que se aumentan las posibilidades de programación. Vamos a ver como se hace utilizando los operadores lógicos. Los más utilizados son:

AND	que significa	Y
OR	que significa	O

Operadores lógicos

★ Te vamos a poner algunos ejemplos para que veas cómo operan.

ARTURO tiene ganas de jugar en el jardín y su mamá le dice:

"SI TIENES UN BALON Y TIENES TIEMPO PUEDES JUGAR"

La mamá de ARTURO le ha puesto dos condiciones.

La 1ª condición es que tenga un balón ... ¿Se cumple?

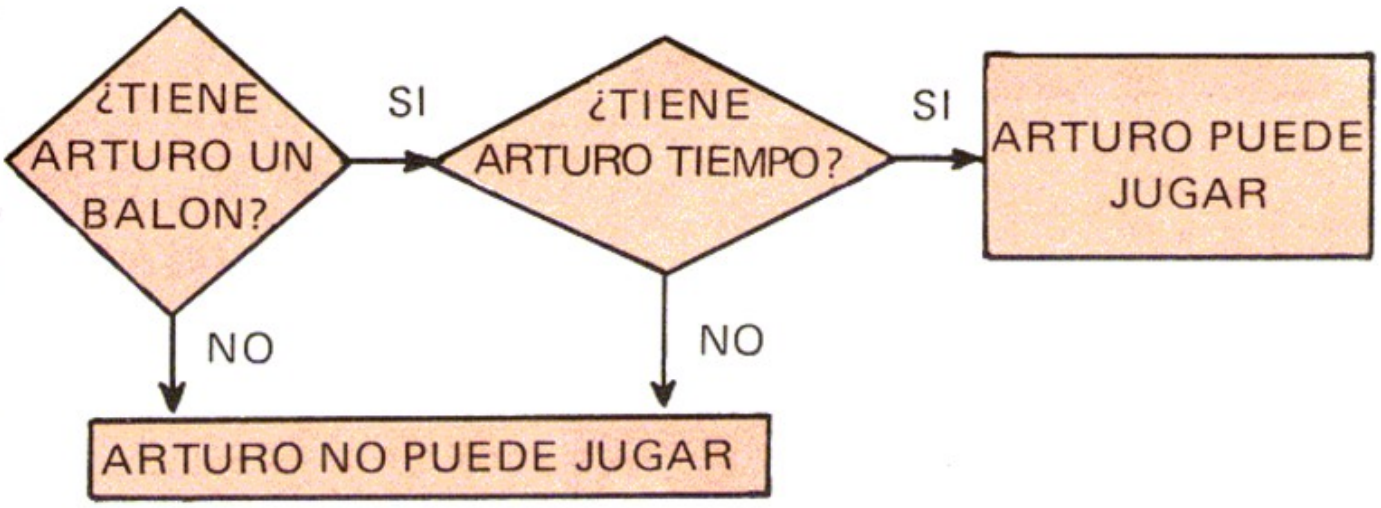
La 2ª condición es que tenga tiempo ... ¿Se cumple?

ARTURO tiene un balón y tiene tiempo para jugar. Como se cumplen las dos condiciones, ARTURO podrá salir a jugar. Pero como puede ocurrir que no siempre se cumplan las dos condiciones vamos a hacer una tabla para ver lo que puede pasar.

Operadores lógicos

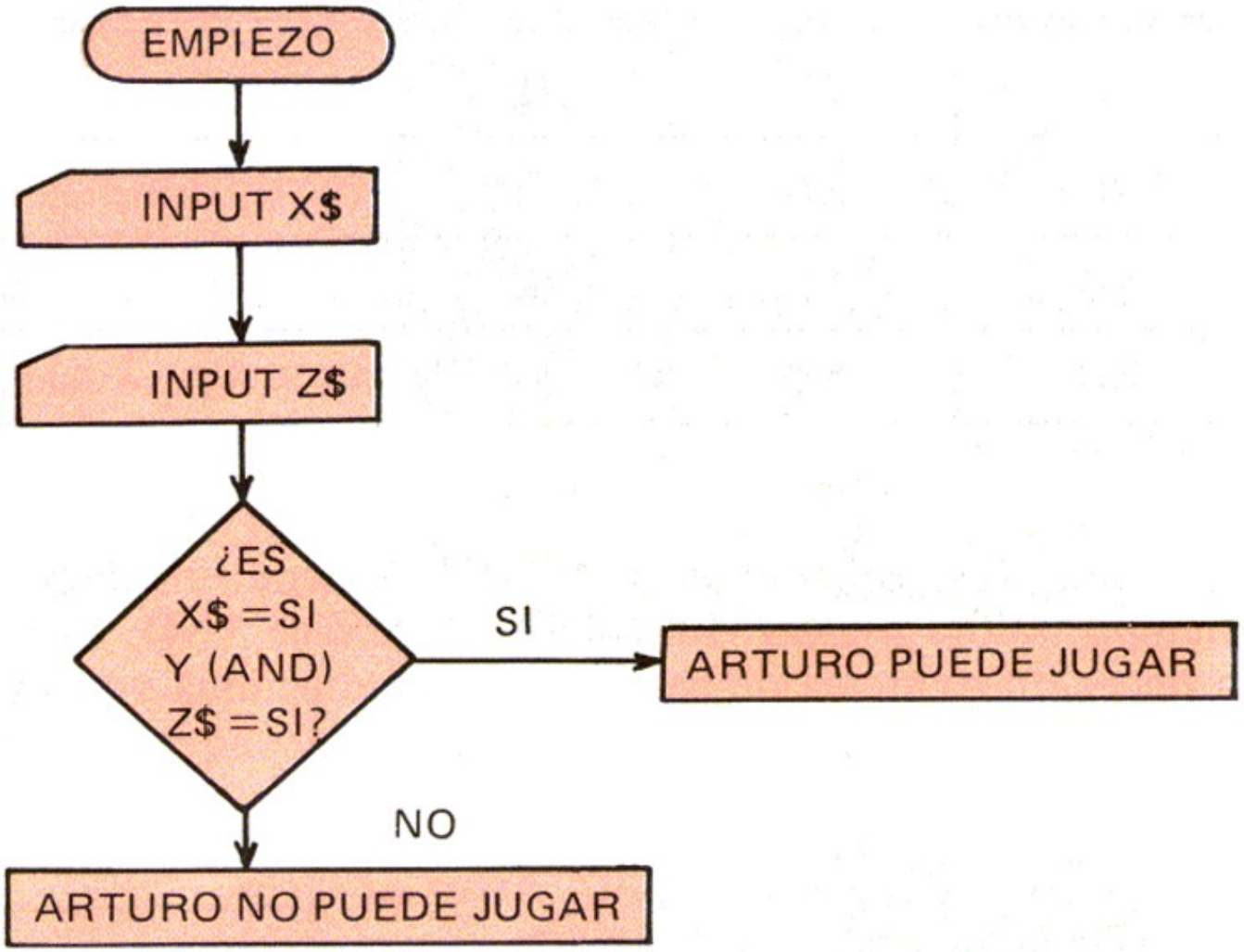
1ª condición Tener balón	*2ª condición Tener tiempo*	*Se cumple 1ª Y (AND) la 2ª?*	*Resultado*
SI	SI	SI	ARTURO saldrá
SI	NO	NO	ARTURO no saldrá
NO	SI	NO	ARTURO no saldrá
NO	NO	NO	ARTURO no saldrá

Como ves, solamente si ARTURO tiene un balón Y tiempo, saldrá a jugar. En BASIC, **Y** se dice **AND**. Vamos a hacer un esquema de esto y luego un diagrama y su programa para reproducir esta situación.



Operadores lógicos

El diagrama de flujo del programa sería:



Operadores lógicos

Y el programa sería:

```
 10  CLS
 20  PRINT "CONTESTA SI O NO A LAS PRE-
     GUNTAS"
 30  PRINT
 40  INPUT "ARTURO TIENE BALON?"; X$
 50  INPUT "ARTURO TIENE TIEMPO?"; Z$
 60  IF X$ = "SI" AND Z$ = "SI" THEN 120
 70  PRINT
 80  PRINT "COMO NO SE CUMPLEN"
 90  PRINT "LAS DOS CONDICIONES"
100  PRINT "ARTURO NO PUEDE JUGAR"
110  GOTO 30
120  PRINT
130  PRINT "COMO SE CUMPLEN"
140  PRINT "  LAS DOS CONDICIONES"
150  PRINT "  ARTURO PUEDE JUGAR"
155  PRINT "- - - - - - - - - - - - - - - - - -"
160  GOTO 30
```

Fíjate bien en la línea 60. Le estás diciendo al ordenador que sólo en el caso de que X$ y Z$ sean iguales a "SI"

Operadores lógicos

vaya a la línea 120, a partir de la cual el ordenador nos reconfirma que por cumplirse las dos condiciones ARTURO puede ir a jugar. Ejecuta el programa y contesta como quieras a las preguntas que te hará el ordenador. Compara los resultados con la tabla de la página 63.

★ ¡Ah! Hay algo que te va a ayudar mucho cuando ejecutes un programa. Puede suceder que cuando teclees la respuesta a una pregunta que te haga el ordenador cometas un error. Por ejemplo, puede ocurrir que en el programa anterior teclees "SE" o "SU" u otra cosa en lugar de "SI".

Estos errores pueden afectar de manera importante al resultado del programa. Comprúebalo contestando, por ejemplo "SI" a la primera pregunta y "SU" a la segunda. Verás como el ordenador te da una respuesta equivocada para tí, pues el ordenador solamente entiende que no has tecleado "SI" y lo entiende como "NO". Esto lo conoces tú ya perfectamente por la forma en que está escrito el programa en la línea 60 y siguientes: Si las dos respuestas no son exactamente "SI" y "SI", el ordenador sigue leyendo el programa en lugar de saltar a la línea 120.

Operadores lógicos

Así que es muy conveniente introducir en el programa una línea "chivata" que nos descubra cuándo hay un error de este tipo.

En nuestro último programa podemos utilizar precisamente una línea "chivata" que contiene el operador lógico **AND**, aunque en este caso no lo usamos para nada en la solución del problema de ARTURO sobre jugar o no en el jardín, sino únicamente para descubrir un posible fallo al pulsar las teclas. Añade:

```
 55  IF X$ <> "SI" AND X$ <> "NO" OR Z$ <> "SI"
     AND Z$ <> "NO" THEN 165
165  PRINT
170  PRINT "TECLEASTE MAL TU RESPUESTA. CON-
     TESTA OTRA VEZ"
180  GOTO 30
```

La línea 55 significa que si X$ es distinta de "SI" **y** distinta de "NO" **o** Z$ es distinta de "SI" **y** distinta de "NO" tu respuesta es incorrecta pues el programa te pedía contestar solamente "SI" o "NO". Fíjate que en esta línea utilizamos un nuevo operador lógico: **OR** (en español significa o).

Vamos a conocerlo mejor con un ejemplo, como hicimos antes. La mamá de ARTURO le dice ahora:

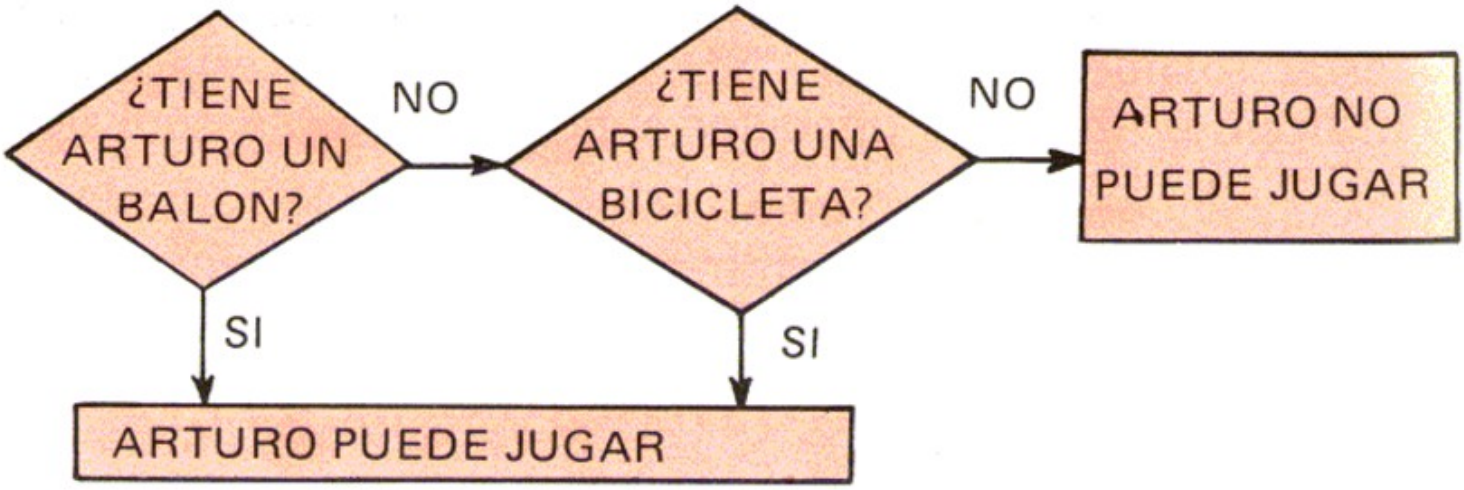
"SI TIENES UN BALON O TIENES UNA BICICLETA PUEDES JUGAR"

Operadores lógicos

La mamá de ARTURO ha puesto dos condiciones, pero como ves, basta que se cumpla una de las dos para que ARTURO pueda jugar. Hagamos nuestra tabla para ver las posibilidades que tiene ARTURO de salir a jugar.

1ª Condición Tener balón	*2ª Condición Tener bicicleta*	*Se cumple 1 o (OR) la 2ª*	*Resultado*
SI	SI	SI	ARTURO jugará
SI	NO	SI	ARTURO jugará
NO	SI	SI	ARTURO jugará
NO	NO	NO	ARTURO no jugará

Como ves, basta que ARTURO tenga una de las dos cosas para que salga a jugar. En BASIC, O se escribe **OR**. El esquema de lo que puede suceder sería así:



Operadores lógicos

Puedes hacer también un programa para ver las posibilidades de ARTURO. Sobre la base del programa anterior, cambia las líneas siguientes:

```
 50  INPUT "ARTURO TIENE BIBICLETA?"; Z$
 60  IF X$ = "SI" OR Z$ = "SI" THEN 120
 85  PRINT "NINGUNA DE"
130  PRINT "COMO SE CUMPLE UNA DE "
```

Ahora en la línea 60 le estás diciendo al ordenador que si se cumple que X$ o Z$ son iguales a "SI" vaya a la línea 120, a partir de la cual el ordenador nos reconfirma que por cumplirse una de las dos condiciones ARTURO puede ir a jugar. Ejecuta el programa y compara los resultados con los de la tabla de la página 68. Fíjate que aquí seguimos utilizando la línea 55 (la "chivata") igual que estaba en el programa anterior.

Estos ejemplos te habrán servido para entender mejor las tablas y el funcionamiento de los operadores lógicos. Estos se utilizan con cierta frecuencia y, cuando te veas en la necesidad de usarlos, te parecerán muy sencillos.

★ Te vamos a proponer una aplicación muy sencilla y muy .... misteriosa.

Operadores lógicos

ARTURO cree que es un espía y tiene un montón de programas con datos secretos que no quiere que nadie vea. Así que ha decidido poner una clave secreta a cada programa para que nadie que no conozca la clave pueda ejecutar el programa. Tú puedes hacer lo mismo, poniendo delante de cada programa que hagas unas líneas como éstas:

```
1   LET C=0
2   CLS: PRINT "ESTE PROGRAMA ES SECRETO.
    CONTESTE CORRECTAMENTE A LAS CLAVES":
    PRINT
3   INPUT "DIGA EL NUMERO Y DESPUES LA PALA-
    BRA SECRETA"; M, L$
4   LET C=C+1: IF C=3 THEN 7
5   IF M=25 AND L$="ARBOL" THEN 10
6   GOTO 2
7   PRINT "  USTED NO CONOCE LA CLAVE. DEBE
          SER UN ESPIA. TUVO TRES OPORTUNI-
          DADES. ESTE PROGRAMA SE AUTODES-
          TRUIRA EN 5 SEGUNDOS"
8   FOR Y=1 TO 3000: NEXT Y
9   CLS: STOP
10  ..... (aquí seguirá tu programa)
```

Operadores lógicos

El contador C de la línea 4 controlará el número de intentos. Cuando C toma el valor 3, el ordenador salta a la línea 7, donde, después de imprimir un texto de aviso, borra el programa de la pantalla. Fíjate en el bucle de retardo de la línea 8. En la línea 9, puedes sustituir la instrucción **STOP** por **NEW**, pero en ese caso el programa entero se borrará de la memoria.

Haz tu propia clave secreta cambiando a tu gusto los valores de M y L$ en la línea 5.

Hagamos otro cambio. Sustituye en la línea 5 **AND** por **OR** y corre el programa. Comprobarás que en este caso basta con que aciertes una de las dos claves para que el ordenador te permita seguir la ejecución del programa principal.

¡Ah! En la línea 5 hemos puesto al final **THEN** 10 porque suponemos que tu programa principal empieza en la línea 10. Haz los cambios necesarios si tu programa empieza en otra línea.

Operadores lógicos

NOTAS PARA EL ADULTO

- Los operadores lógicos sólo actúan entre dos posibilidades: verdadero (representado por 1) y falso (representado por $\emptyset$). En el análisis de las posibilidades que nos ofrecen los operadores lógicos se utilizan las llamadas TABLAS DE VERDAD. La tabla correspondiente al operador lógico **AND** sería:

X	Y	X AND Y
1	1	1
0	0	0
0	1	0
1	0	0

y la tabla del operador lógico **OR** sería:

X	Y	X OR Y
1	1	1
0	0	0
0	1	1
1	0	1

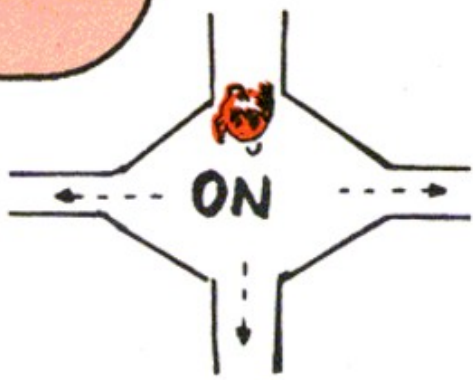
Operadores lógicos

- Estos conceptos tienen frecuentes aplicaciones no sólo en Informática, sino también en Lógica y en Lógica Matemática. Conviene que el niño los asimile, lo cual es muy sencillo si se ejemplifica la teoría con planteamientos como los expuestos en este capítulo.

- Acomode los ejemplos a la capacidad e interés del niño.

- Existen otros operadores lógicos (**NOT, XOR, NOR...**) cuya función es más compleja y que no han sido analizados por considerarlos de un nivel conceptual superior a la capacidad del niño al que está destinado este libro.

IF/THEN (Algo más)



Vamos a mostrarte lo que es un MENU y las distintas formas que puede adoptar la instrucción **IF/THEN**.

Vamos a ver cómo funciona con un ejemplo. Teclea el programa:

```
10  CLS
20  INPUT "DI UN NUMERO"; A
30  INPUT "DI UN NUMERO"; B
40  PRINT "PULSA 1 PARA MULTIPLICAR"
50  PRINT "PULSA 2 PARA DIVIDIR"
60  PRINT "PULSA 3 PARA SUMAR"
70  PRINT "PULSA 4 PARA OPERAR CON OTROS NU-
    MEROS"
80  PRINT "PULSA 5 PARA TERMINAR"
90  INPUT "QUE OPCION QUIERES?"; X
100 IF X <> 1 AND X <> 2 AND X <> 3 AND X <> 4
    AND X <> 5 THEN 220
110 IF X = 1 THEN 160
120 IF X = 2 THEN 180
130 IF X = 3 THEN 200
140 IF X = 4 THEN 10
```

IF/THEN (Algo más)

```
150  IF X = 5 THEN 230
160  PRINT "EL RESULTADO DE MULTIPLICAR "; A;
     "POR"; B; "ES"; A * B
170  GO TO 90
180  PRINT "EL RESULTADO DE DIVIDIR"; A; "EN-
     TRE"; B; "ES"; A/B
190  GO TO 90
200  PRINT "EL RESULTADO DE SUMAR"; A; "MAS";
     B; "ES"; A + B
210  GO TO 90
220  "TECLEASTE MAL TU OPCION. TECLEALA DE
     NUEVO" : GO TO 90
230  STOP
```

Como ves es un programa muy sencillo. En las líneas 40 hasta la 90 estás escribiendo lo que se llama un MENU.

Bueno, no vamos a comer. Pero al igual que en un restaurante te dan un menú para que elijas la comida que te

IF/THEN (Algo más)

gusta, en un programa puedes poner al principio un menú para que al ejecutarlo puedas elegir la opción que más te interesa.

De esta manera tienes un programa más claro y más fácil de interpretar. Pon un menú en todos los programas que hagas y que permitan elegir entre varias opciones. ¡Ah! No te olvides de poner la línea "chivata" para evitar errores.

Fíjate en la línea 160. El resultado de multiplicar A por B no lo hemos asignado a ninguna variable, sino que le hemos dicho al ordenador que escriba directamente A * B. Igual hacemos con las líneas 180 y 200.

★ Bueno, ahora vamos a utilizar **ON GOTO**. No todos los ordenadores te permiten utilizar esta instrucción, así que consulta primero en tu manual si puedes emplearla o no.

Borra las líneas 110, 120, 130, 140 y 150 y escribe:

```
110 ON X GO TO 160, 180, 200, 10, 230
```

Verás que el resultado es el mismo al correr el programa. Con **ON GOTO** es como si le dijeras al ordenador "dependiendo de lo que valga X te irás a las líneas...."

X valdrá siempre 1, 2, 3... y así hasta el número de opciones posibles que queramos incluir en el programa.

Otra posibilidad que te ofrece **ON** es combinarlo con subrutinas. Es decir, utilizando la forma **ON X GOSUB**...

IF/THEN (Algo más)

NOTAS PARA EL ADULTO

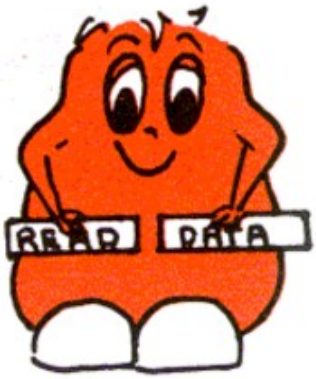
- La instrucción **ON GOTO** o **ON GOSUB** no es fundamental, sino simplemente un auxiliar de la programación. El niño puede practicar con ella para adquirir soltura y velocidad de programación.
- Tenga en cuenta que no todos los ordenadores disponen de esta instrucción. Si el suyo no la tiene, no deje de comentar con el niño otros conceptos importantes desarrollados en este capítulo, como son la presentación de un programa con un menú (líneas 40 a 90), el empleo de líneas de detección de errores al introducir datos en la ejecución del programa (línea 100) y la impresión directa del resultado de operaciones aritméticas sin asignación previa a una determinada variable (líneas 160, 180 y 200).
- Si comprueba que el niño asimila con relativa facilidad estos conceptos puede ampliar sus conocimientos relativos a la instrucción **IF...THEN** explicándosela en su forma más completa:

  IF (Condición) **THEN** (Instrucción) **ELSE** (Instrucción)

  Es decir, si la condición se cumple, se ejecutará la instrucción que sigue a **THEN.** Si no se cumple, se ejecutará la instrucción que sigue a **ELSE.** Por ejemplo:

```
IF A = 5 THEN PRINT "ACERTASTE" ELSE
PRINT "FALLASTE"
```

READ y DATA



ARTURO va a organizar unas MINI OLIMPIADAS, y quiere saber con qué amigos cuenta para cada prueba. Para tenerlo todo muy bien organizado quiere llevar estos datos con ayuda del ordenador.

Para poder hacerlo cómodamente, va a escribir un programa utilizando la instrucción **READ/DATA**. ¿Le ayudamos? Vamos a seguir paso a paso la manera de hacerlo.

ARTURO tiene amigos especialistas en tres pruebas: CARRERA, BALONCESTO y NATACION. Así que lo primero que hace es preparar unas sentencias **DATA** con el nombre de sus amigos y la especialidad que practican:

```
1  DATA LUIS, CARRERA
2  DATA JUAN, BALONCESTO
3  DATA CARLOS, NATACION
4  DATA JOSE, BALONCESTO
5  DATA PEDRO, BALONCESTO
6  DATA JORGE, CARRERA
7  DATA MIGUEL, NATACION
8  DATA ARTURO, BALONCESTO
```

READ y DATA

Estos datos serán leídos después por una instrucción **READ**. Pero eso ya lo veremos más adelante. Ahora vamos a hacer un Menú, como tú ya conoces:

```
10  CLS
20  PRINT "PULSA 1 PARA CORREDORES"
30  PRINT "PULSA 2 PARA BALONCESTISTAS"
40  PRINT "PULSA 3 PARA NADADORES"
50  PRINT "PULSA 4 PARA FIN DE PROGRAMA"
60  PRINT: INPUT "TECLEA TU OPCION"; X
70  IF X>4 OR X<1 THEN PRINT "TECLEASTE
    MAL TU OPCION": GOTO 60
```

Fíjate la forma tan sencilla y original de detectar errores de tecleado en la línea 70 (la "chivata").

En las líneas 80, 90 y 100 le diremos al ordenador que se vaya a las diferentes subrutinas del programa según la opción que elijas:

```
 80  IF X=1 THEN GOSUB 1000
 90  IF X=2 THEN GOSUB 2000
100  IF X=3 THEN GOSUB 3000
110  IF X=4 THEN PRINT "FIN DE PROGRAMA":
     GOTO 200
200  END
```

READ y DATA

Escribamos la subrutina para buscar a los corredores:

```
1000  READ A$, B$
1010  IF B$="CARRERA" THEN 1030
1020  GO TO 1000
1030  PRINT A$
1040  GOTO 1000
```

Analicemos como funciona esta subrutina. En la línea 1000 se leen los datos almacenados en las líneas **DATA** del principio del programa. En la línea 1010 le decimos al ordenador que cuando B$ sea igual a "CARRERA" vaya a la línea 1030 donde escribirá A$ (el nombre del amigo que tiene esa especialidad). Vuelve después a la línea 1000 donde continuará leyendo las sentencias **DATA** que le faltan por leer.

Si B$ es distinto de "CARRERA", el ordenador continúa leyendo el programa y en la línea 1020 recibirá la orden de volver a la línea 1000 para seguir buscando datos.

Añade también:

```
150  GOTO 20
```

Así te evitas tener que teclear RUN cada vez que quieres organizar otra búsqueda.

READ y DATA

Ejecuta el programa como está. Pero ¡cuidado! de momento sólo puedes elegir la opción 1, porque todavía no hemos escrito las subrutinas correspondientes a las otras opciones.

En la pantalla te aparecerán los amigos de ARTURO que son especialistas en CARRERA. También te aparecerá un mensaje indicándote que el ordenador no tiene más datos que leer desde la línea 1000. Por eso conviene que añadas un dato "falso":

```
9  DATA FIN, FIN
```

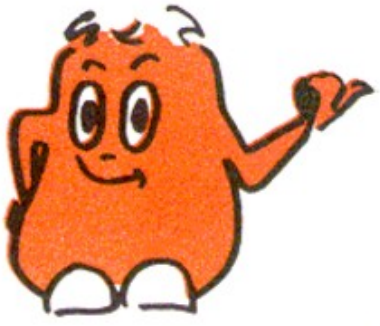
y en la subrutina añade la línea:

```
1015  IF B$ = "FIN" THEN PRINT "LOCALIZADOS LOS
      CORREDORES": RETURN
```

con lo cual damos la subrutina por terminada.

Con **RETURN** la primera línea que ejecutará el ordenador será la 150.

Hagamos ahora la subrutina que nos dará la lista de baloncestistas:



READ y DATA

```
2000  READ A$, B$
2010  IF B$="BALONCESTO" THEN 2030
2015  IF B$="FIN" THEN PRINT "LOCALIZADOS LOS
      BALONCESTISTAS": RETURN
2020  GOTO 2000
2030  PRINT A$
2040  GOTO 2000
```

No hace falta que te la expliquemos, porque funciona exactamente igual que la subrutina anterior.

Pero ¡ATENCION! Ejecuta ahora el programa. Pide la opción que tú quieras (la 1 o la 2). El ordenador te ofrece en pantalla la lista de los amigos cuya especialidad es la que acabas de pedir.

Ahora pide la otra opción. El ordenador no te imprime ningún nombre y te da un mensaje de error: No tiene datos para leer.

Pues, muy sencillo. Cuando una sentencia **READ** está leyendo valores almacenados en las sentencias **DATA**, es como si en cada dato colocara un punto o señal imaginario para saber que ese dato ya ha sido leído.

RESTORE

Añade finalmente la subrutina que nos dará los nadadores:

```
2900  RESTORE: CLS
3000  READ A$, B$
3010  IF B$ = "NATACION" THEN 3030
3015  IF B$ = "FIN" THEN PRINT "LOCALIZADOS LOS
      NADADORES" : RETURN
3020  GOTO 3000
3030  PRINT A$
3040  GOTO 3000
```

NOTAS PARA EL ADULTO

- Las instrucciones **READ, DATA** y **RESTORE** son muy útiles pues simplifican mucho el trabajo de asignación de valores a las variables. Conviene que el niño, ya en esta fase, utilice normalmente las mismas como un auxiliar más de la programación.

- Sigan el programa precedente paso a paso con el ordenador. No continúen con cada nueva fase sin haber entendido perfectamente el paso anterior.

RESTORE

- Algunos ordenadores exigen que los datos incluidos en una sentencia **DATA** se escriban entre comillas. En otros no es necesario.
- Presten atención a las líneas 1010, 2010 y 3010. No cometan ningún error ni incluyan espacios en blanco al escribir el texto entrecomillado que estamos asignando a las variables. De hacerlo así, al pedir una especialidad, el ordenador buscará en **DATA** un valor que no existe, pues el ordenador interpreta un espacio en blanco como un carácter más.
- Siguiendo los mismos criterios ya expuestos y haciendo las modificaciones necesarias puede ampliar los campos añadiendo más especialidades, o más opciones (por ejemplo, la edad de los participantes). Si tiene dificultades, vea el programa tipo de la página 150.
- Escribiendo **RESTORE** seguido de un número de línea que contenga una sentencia **DATA,** el ordenador leerá los datos existentes a partir de ese número de línea. Por ejemplo, en nuestro programa, un **RESTORE** 4 ignoraría a LUIS, JUAN y CARLOS en la segunda y sucesivas lecturas.
- Los datos del programa se han incluido en las líneas 1 a 9 para una mayor claridad, pero igualmente pueden colocarse al final del mismo o, incluso, en la forma: **DATA** LUIS, CARRERA, JUAN, BALONCESTO, CARLOS, NATACION, JOSE ... etc.

DIM

Vamos a conocer una instrucción que evita mucho trabajo. Desde que aprendió a utilizarla, ARTURO aprovecha todas las ocasiones que puede y la emplea en sus programas. La verdad es que ahorra mucho tiempo.

¿Cómo descubrió ARTURO la intrucción **DIM**? Todo surgió un día que nuestro amigo quiso clasificar y ordenar algunas de sus cosas. Primero se hizo una relación:

LIBROS	**JUGUETES**	**ROPA**
PITUFOS	BALON	PANTALON
SUPERMAN	BICICLETA	CAMISA
GALAXIAS	RAQUETA	GORRA
INDIANA JONES	PATINES	ZAPATOS
PATO DONALD		PIJAMA
PEQUEÑA LULU		

Después decidió darle a cada objeto una variable. Así que empezó a escribir:

DIM

```
LET A$ = "PITUFOS"
LET B$ = "SUPERMAN"
.....
.....
```

ARTURO tiene razón. Va a tardar mucho tiempo en definir todas las variables. Además, como luego tenga que añadir más objetos, se le van a acabar las variables. Y si luego quiere que el ordenador le dé una relación de todos los objetos, clasificados por temas, tendrá que poner en su programa instrucciones del tipo **PRINT** A$, **PRINT** B$, **PRINT** C$, etc.

Así que decidió utilizar lo que se llama *variables con subíndice.* Una variable con subíndice se escribiría, por ejemplo:

A (1), A (2), A (3) ... etc.

Como ves, todas tienen la misma letra pero son variables distintas.

Trabajar con ellas tiene muchas ventajas. En primer lugar, la asignación de estas variables se puede hacer más rápidamente utilizando un bucle **FOR/NEXT**. Vamos a trabajar con los libros de ARTURO.

DIM

Escribe primero:

```
 5  CLS
10  DIM L$ (6)
```

Esto quiere decir que el ordenador va a reservar en su memoria 6 casillas para 6 variables que tendrán todas el mismo prefijo o la misma letra. En nuestro caso, L$. Así que las variables serán L$ (1), L$ (2), L$ (3), L$ (4), L$ (5) y L$ (6). Vamos a darles valores a estas variables. Como todas tienen el mismo prefijo, no es necesario que escribamos 6 líneas de programa con **LET**, sino que podemos utilizar un bucle **FOR/NEXT**. Escribe:

```
20  FOR N = 1 TO 6
25  PRINT "Escribe el libro"
30  INPUT L$ (N)
35  CLS
40  NEXT N
```

El bucle que empieza en la línea 20 va a hacer que N tome uno detrás de otro valores desde 1 hasta 6. Por 6 veces, el ordenador nos preguntará que valor le queremos asignar a L$ (N). Si sigues el orden que puso ARTURO al principio de este capítulo, las variables quedarán así:

```
L$ (1) = PITUFOS
L$ (2) = SUPERMAN
L$ (3) = GALAXIAS
L$ (4) = INDIANA JONES
L$ (5) = PATO DONALD
L$ (6) = PEQUEÑA LULU
```

El ordenador los tiene ya todos memorizados. Para probarlo, teclea, por ejemplo, **PRINT** L$ (3) o **PRINT** L$ (5). ¡El ordenador te contesta!

Así que podemos hacer que el ordenador nos imprima –diciéndoselo con una sola instrucción– el valor de todas las variables que tienen como prefijo común L$. Añade al programa las líneas:

```
50  FOR N = 1 TO 6
60  PRINT L$ (N)
70  NEXT N
```

Con este bucle el ordenador te imprimirá en pantalla todos los libros de ARTURO. Fíjate todo el trabajo y el tiempo que te has ahorrado.

Si queremos repetir el mismo proceso para los juguetes y para la ropa de ARTURO, lo mejor es utilizar un menú.

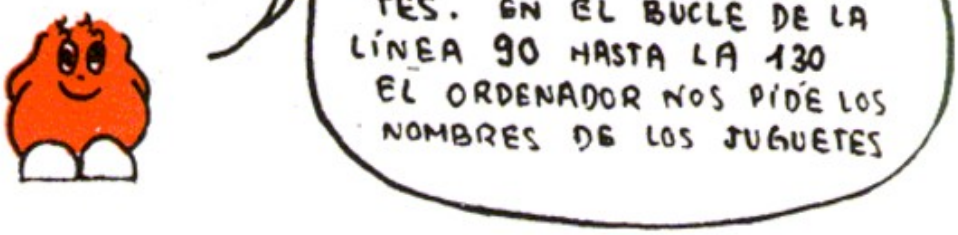
DIM

Te sugerimos que utilices el prefijo J$ para todas las variables de los juguetes y R$ para las variables de la ropa:

★ Vamos a hacer un programa para conocer los distintos objetos que tiene ARTURO clasificados por grupos. Empecemos desde el principio. Borra antes el programa anterior.

Primero prepararemos al ordenador para que reciba todos los datos:

```
 10  CLS
 20  DIM L$ (6)
 30  FOR N = 1 TO 6
 40  PRINT " ESCRIBE EL LIBRO"
 50  INPUT L$ (N)
 60  CLS
 70  NEXT N
 80  DIM J$ (4)
 90  FOR N = 1 TO 4
100  PRINT "ESCRIBE EL JUGUETE"
110  INPUT J$ (N)
120  CLS
130  NEXT N
```



DIM


```
140  DIM R$ (5)
150  FOR N = 1 TO 5
160  PRINT "ESCRIBE LA ROPA"
170  INPUT R$ (N)
180  CLS
190  NEXT N
```

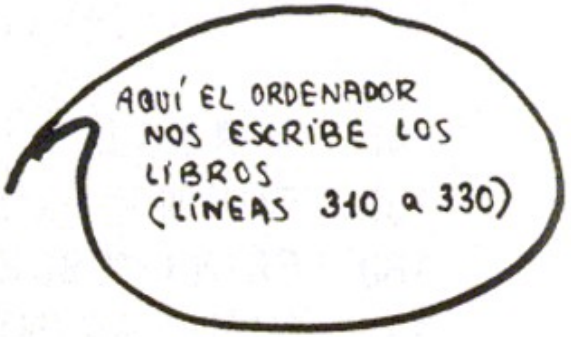
Ahora escribiremos un menú para que en su momento podamos elegir cualquiera de las tres listas de objetos que tenemos.

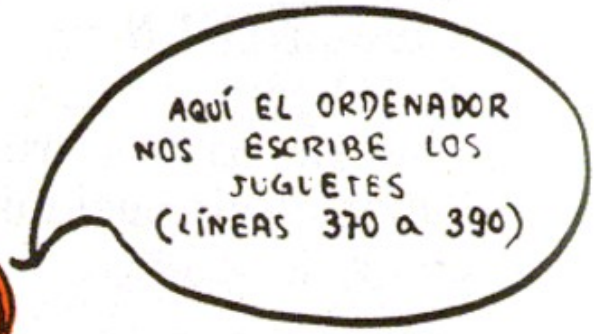
```
200  CLS
210  PRINT "OPCIONES:"
220  PRINT "PARA LIBROS PULSA 1"
230  PRINT " PARA JUGUETES PULSA 2"
240  PRINT " PARA ROPA PULSA 3"
250  INPUT " PULSA LA OPCION ELEGIDA "; X
260  IF X = 1 THEN 300
270  IF X = 2 THEN 360
280  IF X = 3 THEN 410
290  IF X <> 1 AND X <> 2 AND X <> 3 THEN PRINT
     " TECLEASTE MAL TU OPCION": GO TO 250
```

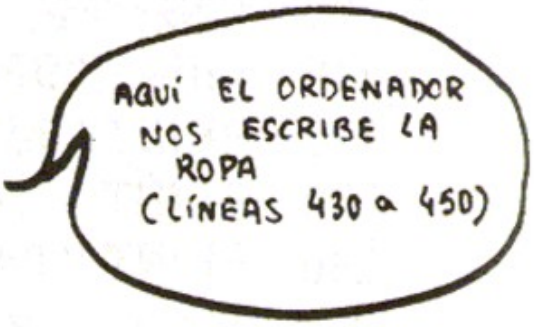
DIM

Finalmente, hagamos la parte del programa que nos escribirá la lista escogida.

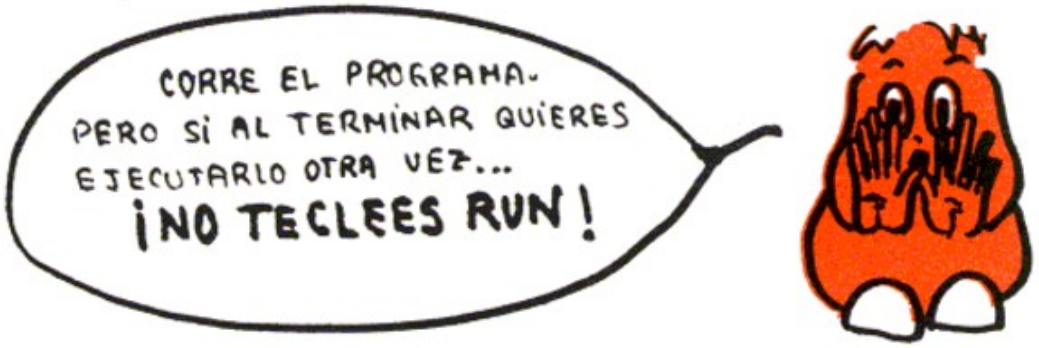
```
300  CLS
310  FOR N=1 TO 6
320  PRINT L$(N)
330  NEXT N
340  GO SUB 500
350  CLS: STOP
360  CLS
370  FOR N=1 TO 4
380  PRINT J$(N)
390  NEXT N
400  GO SUB 500
410  CLS : STOP
420  CLS
430  FOR N=1 TO 5
440  PRINT R$(N)
450  NEXT N
460  GO SUB 500
470  END
500  INPUT "PULSA K PARA OTRA LISTA. PULSA
     OTRA TECLA PARA TERMINAR"; K$
510  IF K$="K" THEN 200
520  RETURN
```







DIM

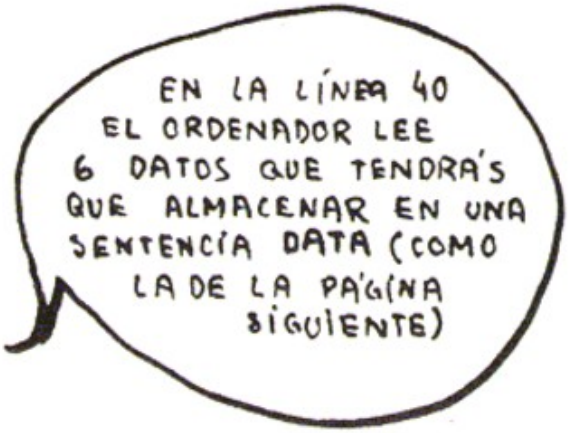


Este programa tiene un inconveniente: Cada vez que teclees **RUN** y corras el programa tendrás que introducir nuevamente los datos, pues como ya sabes, **RUN** pone todas las variables a cero.

Si tuvieras el programa guardado en un cassette con los datos ya introducidos, la única forma de evitar que se borraran sería tecleando **GOTO** 200 en lugar de **RUN**. En la línea 200 el programa empieza a pedirte opciones. Esto no es muy práctico.

★ Se puede hacer el programa de manera que el ordenador conserve en memoria todos los datos. Podemos usar las instrucciones **READ/DATA**. La estructura general del programa no cambia, pero, para introducir los datos en la memoria del ordenador escribiríamos:

```
10  CLS
20  DIM L$(6)
30  FOR N=1 TO 6
40  READ L$(N)
50  NEXT N
```



DIM

Y en alguna parte del programa (preferentemente al final) pondrías una sentencia **DATA**:

```
800 DATA PITUFOS, SUPERMAN, GALAXIAS, INDIANA JONES, PATO DONALD, PEQUEÑA LULU
```

Siguiendo el mismo método tendrías que hacer cambios parecidos para los juguetes (líneas 80 hasta la 130) y para la ropa (líneas 140 hasta la 190). También tendrás que escribir dos líneas más de **DATA**, con los juguetes y la ropa.

NOTAS PARA EL ADULTO

- La instrucción **DIM** es muy utilizada en la realización de listas y tablas. Conviene que el niño la comprenda perfectamente antes de abordar el estudio de capítulos posteriores, en los que se aplicará de forma habitual.

- El ejemplo utilizado en este capítulo no resulta –por su estructura– verdaderamente práctico en la vida real, pero ayuda al niño a comprender y practicar insistentemente la instrucción **DIM** a lo largo del programa.

DIM

- En la mayoría de los ordenadores, sólo es necesaria la utilización de **DIM** cuando se van a utilizar más de 10 variables indexadas.

  En caso de utilizar más variables que las previamente dimensionadas se producirá un error, que su ordenador detectará inmediatamente.

- En nuestro programa se han empleado variables de caracteres a fin de aplicar un ejemplo más motivador para el niño. **DIM** puede utilizarse igualmente con variables numéricas.

  Cuando emplee variables alfanuméricas, debe tener en cuenta que algunos ordenadores exigen que al establecer la dimensión se especifique también el número máximo de caracteres que puede tener cualquiera de los valores asignados a las variables. Así, en nuestro ejemplo anterior la línea 20 se escribiría:

  ```
  20  DIM L$ (6, 13)
  ```

  con lo que le indicamos al ordenador que reserve en su memoria espacio para 6 variables con prefijo común **L$**, pudiendo tener cualquiera de ellas un máximo de 13 caracteres.

- Cuando el niño esté completamente familiarizado con la instrucción **DIM** y la emplee con soltura, puede ampliar la versatilidad de sus programas combinando esta instrucción con otras, por ejemplo **INPUT.**

DIM

Así, unas líneas de programa para introducir a través del teclado el número de elementos de la lista serían:

```
10  INPUT "CUANTOS LIBROS VAS A
    LISTAR?"; A
20  DIM L$ (A)
30  FOR N = 1 TO A
40  INPUT L$ (N)
50  NEXT N
```

- La instruccion **DIM** se utiliza también frecuentemente para la realización de tablas. Aunque su explicación escapa a los objetivos de este libro, se expone a continuación un breve ejemplo. El BASIC permite la utilización de variables con dos o más subíndices.

  Así, el programa

```
10  DIM A (2, 4)
20  FOR X = 1 TO 2
30  FOR X = 1 TO 4
40  PRINT "VALOR"; X; ","; Y; "DE LA
    MATRIZ A"; "  ";
50  INPUT A (X, Y)
60  PRINT A (X, Y)
70  NEXT Y
80  NEXT X
```

  definiría los valores de una tabla similar a esta:

DIM

	Columna 1	*Columna 2*	*Columna 3*	*Columna 4*
Fila 1	A (1, 1)	A (1, 2)	A (1, 3)	A (1, 4)
Fila 2	A (2, 1)	A (2, 2)	A (2, 3)	A (2, 4)

De esta forma, el elemento A (2, 3) será el situado en la intersección entre la fila 2 y la columna 3.

Ordenando cosas

Una de las ventajas que tienen los ordenadores es que pueden ayudarte mucho a hacer trabajos en los que hay que repetir una misma cosa muchas veces. Como ya sabes, el ordenador es muy rápido. Así que, un trabajo sencillo que a tí te puede llevar mucho tiempo, el ordenador lo hace en segundos.

Nosotros a lo largo del día siempre estamos ordenando algo: Ordenamos las pegatinas, ordenamos las fichas de clase, etc. También podríamos ordenar los nombres de nuestros amigos por orden alfabético en nuestro listín. Y tu ordenador puede también ordenar todas estas cosas por tí. Sólo tienes que enseñarle cómo hacerlo.

★ Vamos a ir viendo poco a poco cómo se hace. Entrenemos primero al ordenador para que vaya comparando números. Escribe este programa para que el ordenador compare dos números dados.

Ordenando cosas

```
10  CLS
20  INPUT "DI UN NUMERO"; K
30  INPUT "DI OTRO NUMERO"; A
40  IF A > K THEN LET K = A
50  PRINT "EL MAYOR ES"; K
```

En las líneas 20 y 30 el ordenador espera los valores. En la línea 40 el ordenador los compara. Le decimos que si el valor que hemos dado a la variable A es mayor que el valor de la variable K, entonces que la variable K tome el valor de A (que es mayor) y lo imprima.

Por ejemplo K = 5 y A = 9.

Como A es mayor que K (es decir, 9 es mayor que 5) K = A (K toma el valor 9) y se imprime.

Pueba tú cambiando los valores de A y K.

Si queremos que el ordenador compare varios números y después nos escriba el mayor, podemos utilizar un bucle. Añade las líneas:

```
24  INPUT "CON CUANTOS NUMEROS LO QUIERES
    COMPARAR?"; N
25  FOR I = 1 TO N
45  NEXT I
```

Así puedes darle hasta N números para comparar. El ordenador te escribirá el mayor.

Ordenando cosas

★ Supongamos que queremos que el ordenador compare y escriba todos los números que le demos en orden (de menor a mayor).

En el programa anterior el ordenador sólo guardaba en memoria el valor mayor de los dos que comparaba. Es como si tú tuvieras una ficha con un número en tu mano. Te van dando otras fichas, y tú las comparas y te guardas solamente la ficha mayor. La menor, la tiras.

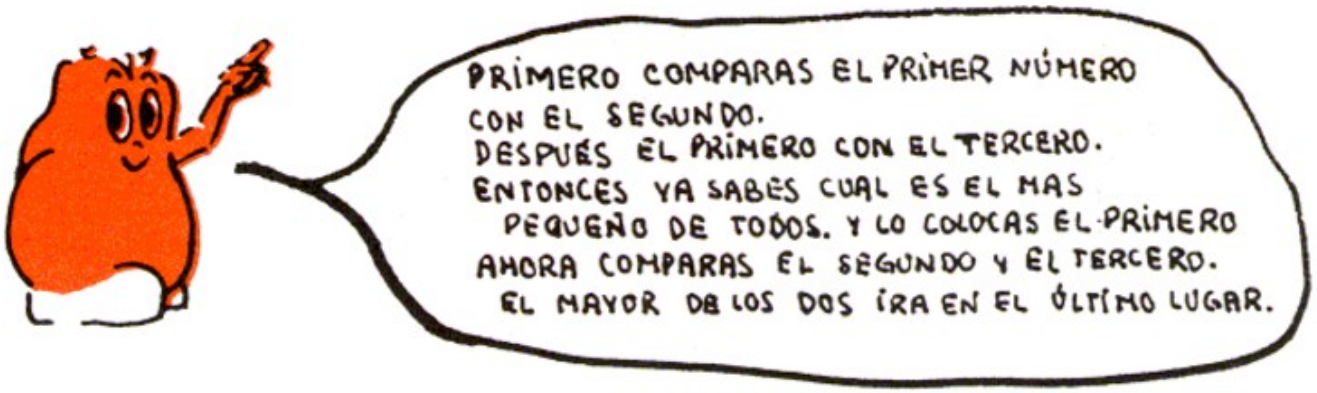
Pero si lo que quieres es ordenarlas desde el número menor al número mayor, tendrás que guardártelas todas para luego clasificarlas y ordenarlas. Así que no puedes tirar ninguna ficha o te faltarán números.

Lo mismo tiene que hacer el ordenador. Vamos a ver un ejemplo. Supongamos que queremos ordenar de menor a mayor los números que nos dan en este orden: 9, 3 y 5.

Tú con tus fichas harías así: Compararías la primera con la segunda y las ordenarías, después tomarías la que ahora ocupa el primer lugar, la compararías con la tercera y las ordenarías. Finalmente compararías las que ahora ocupen los lugares segundo y tercero y las colocarías en su lugar.

Ordenando cosas

PASOS	ORDEN
Al empezar, tenemos	9 - 3 - 5
1° Compara 9 y 3. Como 3 es menor, lo colocamos delante.	3 - 9 - 5
2° Compara 3 y 5. Como 3 es menor, lo deja donde está.	3 - 9 - 5
3° Compara 9 y 5. Como 5 es menor, lo colocamos delante.	3 - 5 - 9



La ordenación está hecha. El proceso es el mismo para 3 ó 1000 números, o los que quieras.

Pero veamos como lo haría el ordenador. Antes tendremos que asignar estos valores a unas variables.

A	B	C
9	3	5

Ordenando cosas

Cuando trabajemos con el ordenador, para obtener el resultado final tendremos que darle una orden del tipo **PRINT** A, B, C. Pero ¡OJO! Si el ordenador escribiera ahora los valores de estas variables, escribiría 9 - 3 - 5. Así que tenemos que decirle que asigne a la variable A el número más pequeño, a la variable B el número intermedio y a la varible C el número mayor.

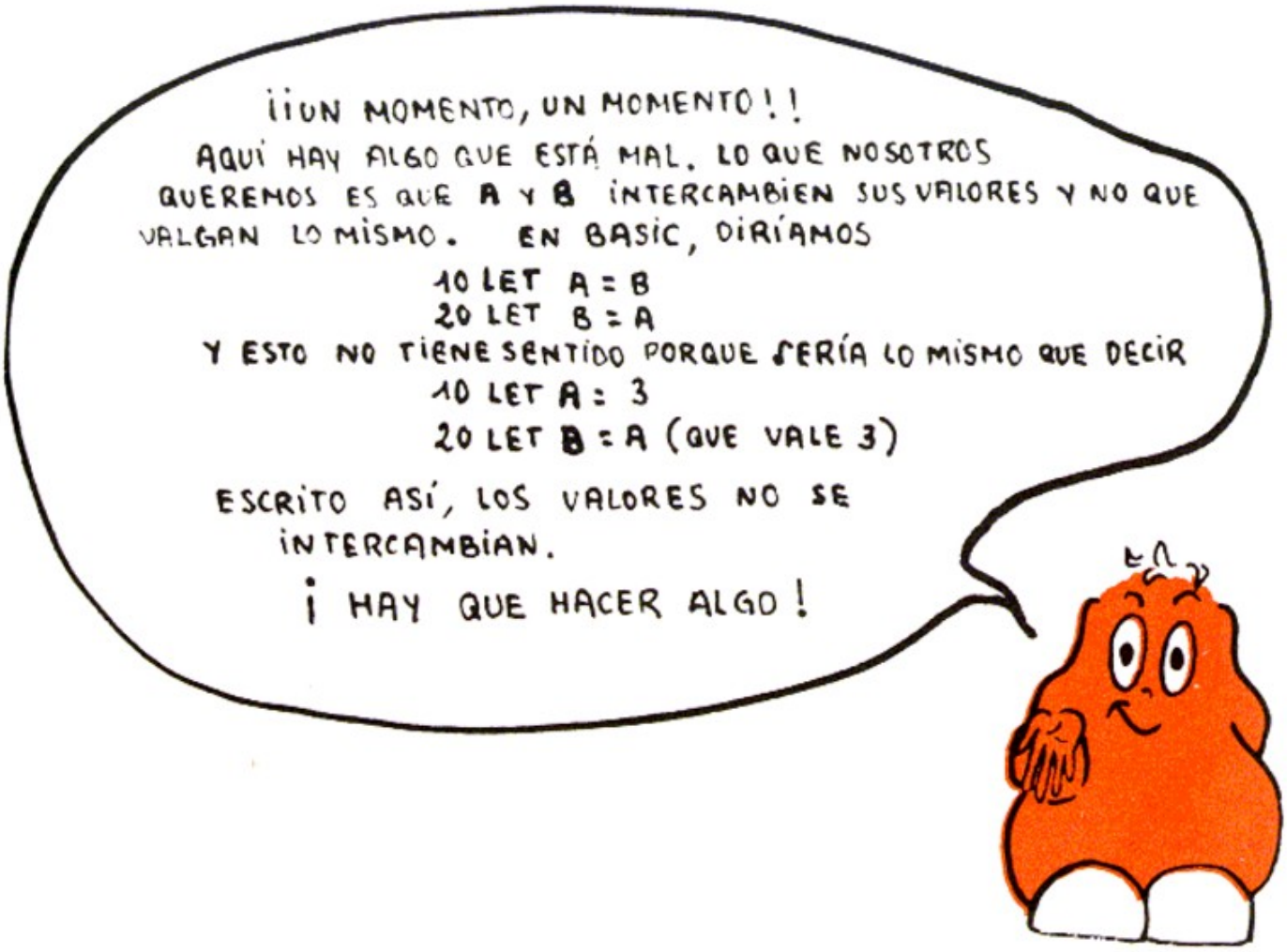
Es decir, que el ordenador tendrá que intercambiar los valores de las variables entre sí. Los pasos a seguir serían los siguientes:

A	B	C
9	3	5

1° compara A y B. Como 3 es menor que 9, lo asignamos a A. Es decir, A = 3, que es lo mismo que decir que A toma el valor de B (A = B).

A	B	C
3	3	5

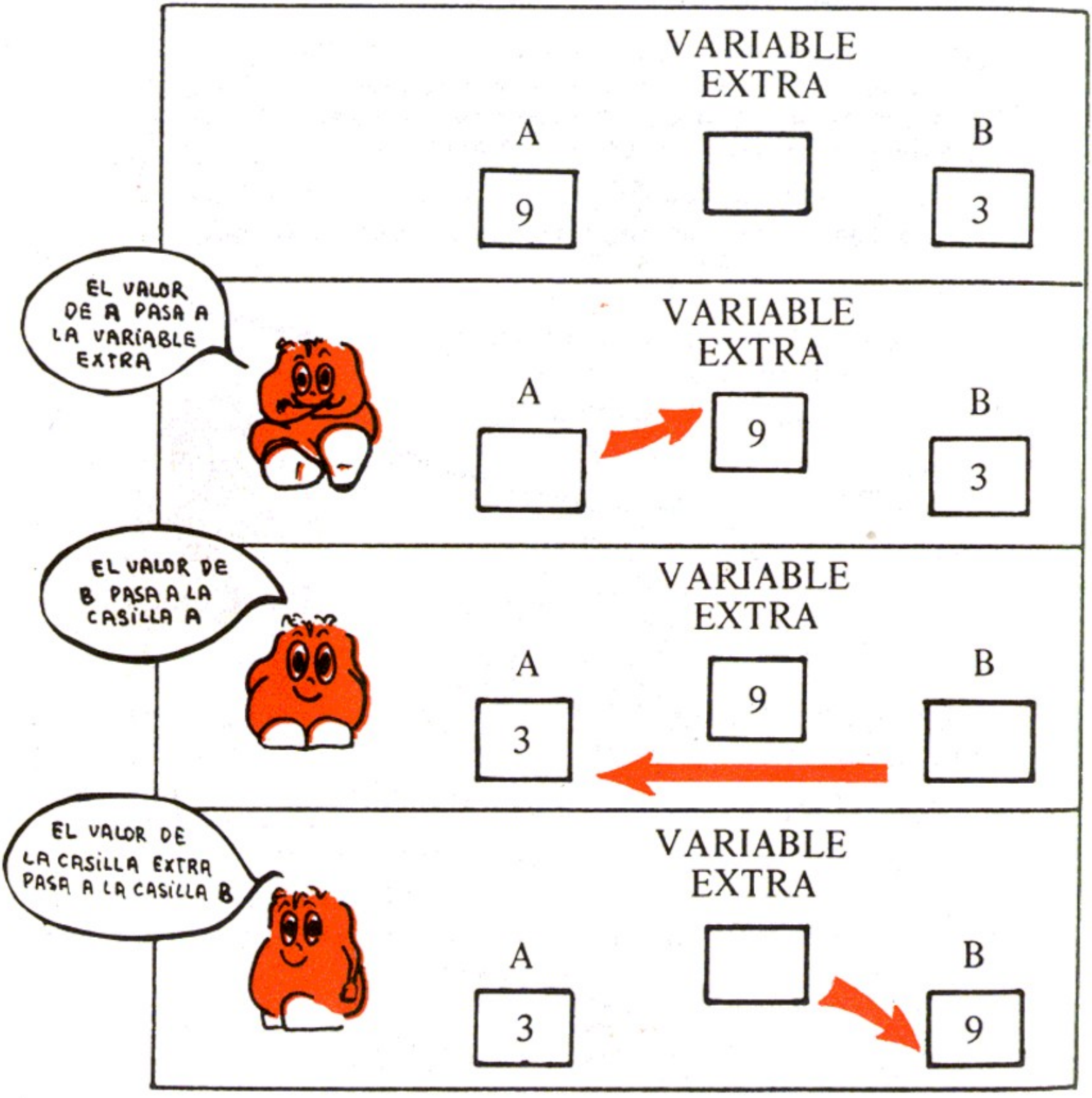
Ordenando cosas



Tiene razón ARTURO. Tenemos que decirle al ordenador que la variable A no pierda su valor cuando tome el valor de B. Hay una forma de hacerlo: Utilizar una variable *extra* donde momentáneamente se pueda guardar ese valor.

El proceso que seguiría el ordenador sería:

Ordenando cosas



Ordenando cosas

Escrito en BASIC, el anterior proceso sería así:

```
10  LET A = 9
20  LET B = 3
30  PRINT A; B
40  LET E = A
50  LET A = B
60  LET B = E
70  PRINT A; B
```

De esta forma se intercambian los valores de A y B sin perder ninguno de ellos.

Puedes comprobarlo fácilmente. Si ejecutas el programa verás como el ordenador imprime:

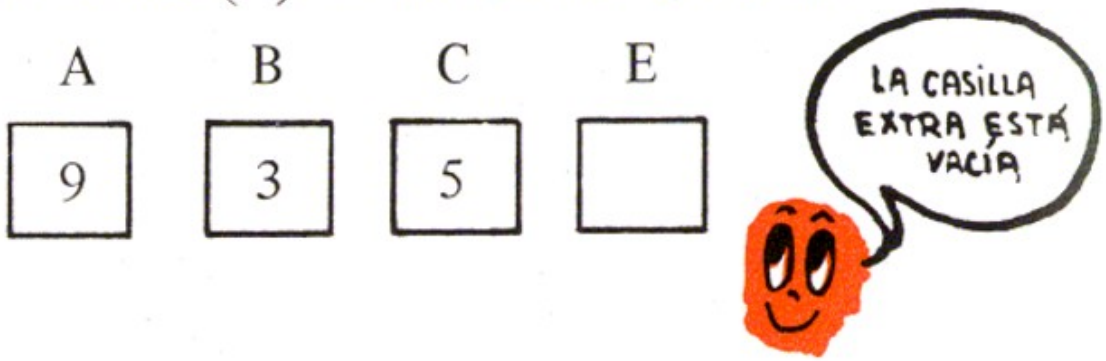

```
9    3
3    9
```

Con las líneas 40, 50 y 60 has intercambiado los valores de las variables A y B. Este truco es muy importante pues es la clave para poder desarrollar todos los programas que te explicamos a continuación. También podías haber escrito estas líneas utilizando una sola línea:

```
40  LET E = A: LET A = B : LET B = E
```

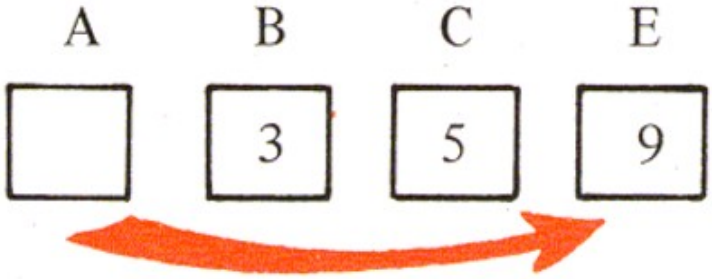
Ordenando cosas

Sigamos pues analizando el proceso que interrumpimos en la página 103 cuando ARTURO nos avisó que lo estábamos haciendo mal. Vamos a empezar otra vez pero utilizando una casilla extra (E). El ordenador actúa así:

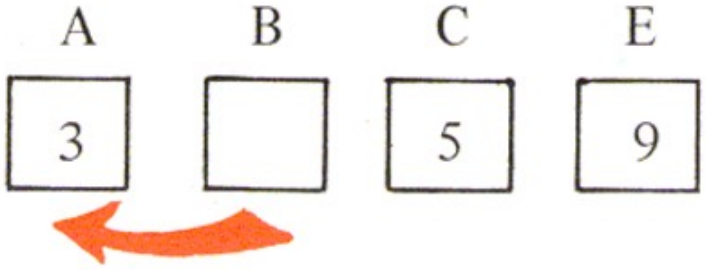


1er PASO

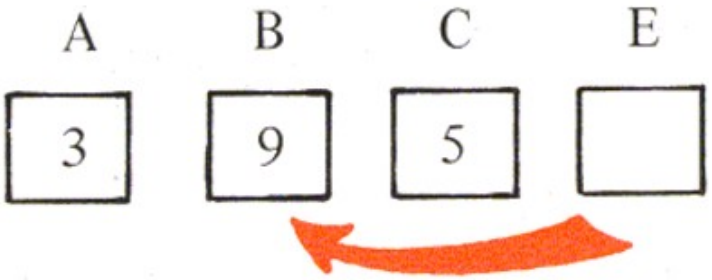
Toma A (que vale 9) y lo compara con B (que vale 3). Como B es menor tiene que intercambiarlo con A. Siguiendo el proceso que vimos antes, pasa el valor de A (que vale 9) a la casilla extra E.



Ahora coge el valor de B (que vale 3) y lo pasa a la casilla A (que ahora está vacía).



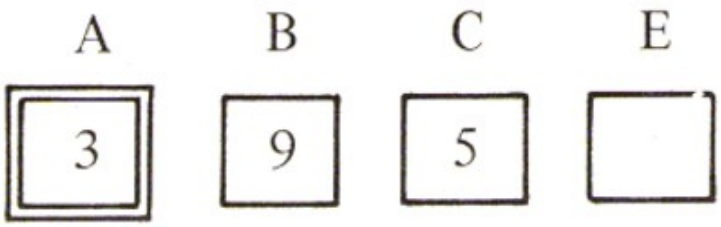
Luego pasa el valor de la casilla extra (que ahora es 9) a la casilla B.



Al terminar este paso el ordenador ha comparado el primero y segundo elementos y ha colocado en la variable A el menor de ellos.

2° PASO

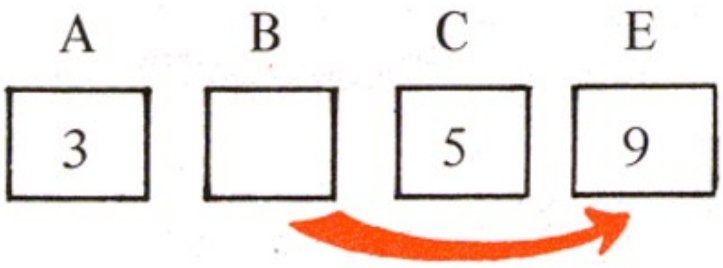
Toma A (que ahora es 3) y lo compara con C (que vale 5). Como A es menor que C, no tiene que intercambiarlos. Ya hemos fijado que el elemento menor es 3, pues ya lo hemos comparado con los otros dos. La variable A tiene ahora el menor valor.



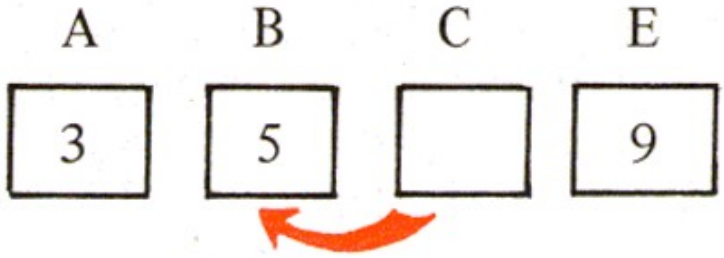
Ordenando cosas

3er PASO

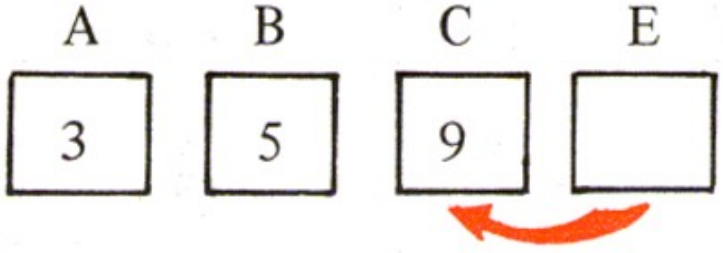
Toma B (que ahora es 9) y lo compara con C (que vale 5). Como C es menor tiene que intercambiarlo con B. Pasa el valor de B (que vale 9) a la casilla extra.



Ahora coge el valor C (que es 5) y lo pasa a la casilla B (que ahora está vacía).



Luego pasa el valor de la casilla extra (que ahora es 9) a la casilla C.



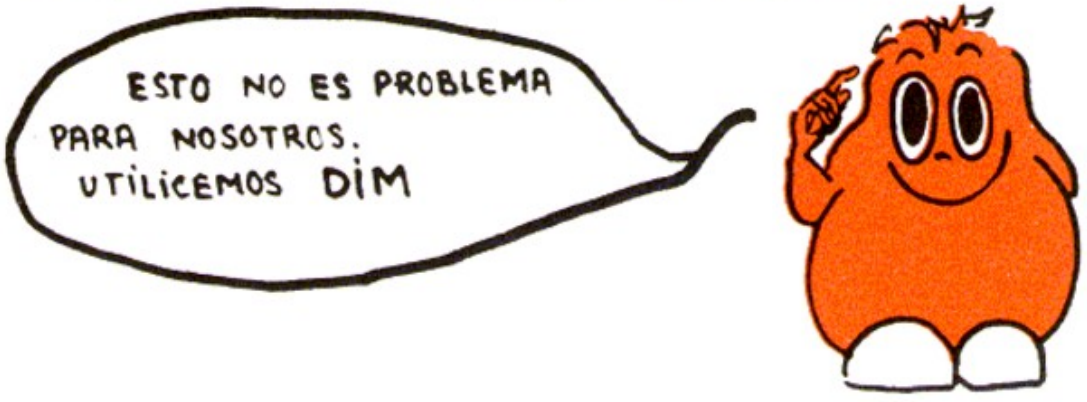
Ya hemos colocado los tres números de menor a mayor.

Ordenando cosas

Todo este proceso escrito en forma de programa, podría ser parecido a éste:

```
10  INPUT A, B, C
20  IF A > B THEN LET E = A: LET A = B: LET B = E
30  PRINT A ;" ";B;" ";C
40  IF A > C THEN LET E = A : LET A = C : LET C = E
50  PRINT A;" ";B;" ";C
60  IF B > C THEN LET E = B: LET B = C: LET C = E
70  PRINT A: " ";B;" ";C
80  END
```

Estarás pensando que el programa resulta un poquito largo para ordenar solamente 3 números. Si tuviéramos que ordenar 100 números, nos resultaría más cómodo ordenarlos a mano. Así que tendremos que pensar algo...



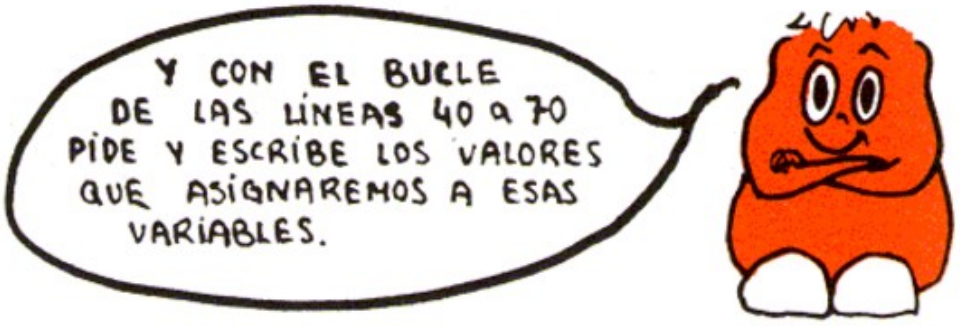
★ ¡Es cierto! Con **DIM** podemos hacerlo mucho más fácil. Tú ya sabes cómo utilizarlo,así que vamos a analizar el programa poco a poco.

Ordenando cosas

Primero escribe:

```
10  CLS
20  INPUT "CUANTOS NUMEROS QUIERES ORDE-
    NAR?"; N
30  DIM A (N)
40  FOR I = 1 TO N
50  INPUT "DIME UN NUMERO Y PULSA ENTER"; A (I)
60  PRINT A (I); " ";
70  NEXT I
75  PRINT : PRINT
```

Te recuerdo que con la instrucción **DIM** de la línea 30 preparas al ordenador para que reserve en su memoria N casillas, que es el número de variables que vamos a utilizar.



Por ejemplo, corre el programa y dale a N el valor 5. El ordenador va a comparar 5 números. Con el bucle iniciado en la línea 40 el ordenador recibirá los 5 valores que tú metas a través del teclado y los irá asignando a las variables. Por ejemplo, introduce esta serie de números: 5 - 9 - 2 - 3 - 1.

Ordenando cosas

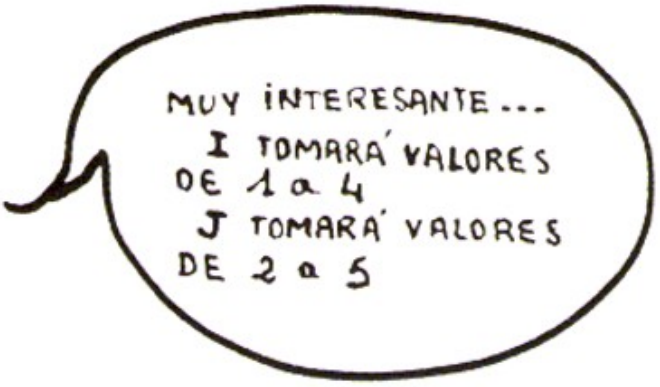
El ordenador los asignará de esta forma:

```
A (1) = 5
A (2) = 9
A (3) = 2
A (4) = 3
A (5) = 1
```

El ordenador te los escribirá en pantalla.

Seguimos. Presta mucha atención, porque ahora vienen unas líneas que son muy bonitas:

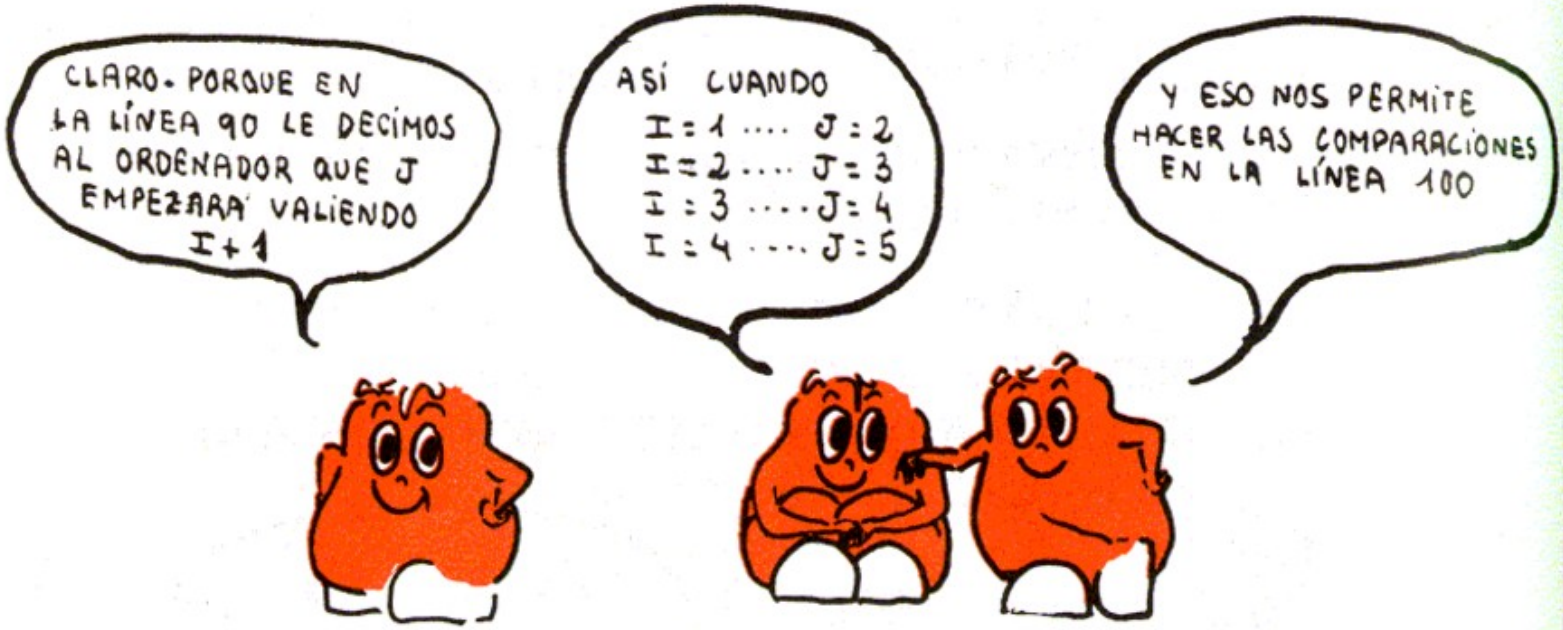
```
 80  FOR I = 1 TO N - 1
 90  FOR J = I + 1 TO N
100  IF A (I) > A (J) THEN LET E = A (I): LET A (I) =
     = A (J): LET A(J) = E
110  NEXT J
120  NEXT I
```



Ya conoces cómo funciona la línea 100 porque es donde el ordenador compara e intercambia valores ayudándose de la casilla extra.

Ordenando cosas

Fíjate que los valores que compara son A (I) y A (J). Con las líneas 80 y 90 iniciamos unos bucles que darán respectivamente valores a A (I) y A (J). Estos bucles están muy bien pensados. Verás. El bucle que empieza en la línea 90 está anidado dentro del bucle que empieza en la línea 80. Así que mientras I vale 1, J irá tomando sucesivamente los valores 2, 3, 4 y 5.



Fíjate otra vez en la línea 100. Primero compara A (1) con A (2) o sea, 5 y 9, e intercambiará sus valores si es necesario. Sigue a la línea 110 y toma el siguiente valor de J. Ahora compara A (1) con A (3), y así sucesivamente...

Cuando J ya vale 5, llega a la línea 120 y toma el siguiente valor de I (que es 2). Ahora comparará A (2) con A (3) y así sucesivamente...

Ordenando cosas

¡Ojo! Cuando el ordenador compara A (1), A (2), etc... lo que compara en realidad son los valores que A (1), A (2), etc... tienen en ese momento.

Utilizamos I y J únicamente para marcar el subíndice y así poder comparar el primer elemento de la serie con el segundo y siguientes y luego el segundo con los siguientes, etc.

Solamente nos falta añadir unas líneas para que nos imprima los resultados. Escribe:

```
130  FOR I=1 TO N
140  PRINT A (I);" ";
150  NEXT I
160  END
```

Ordenando cosas

NOTAS PARA EL ADULTO

- Los conceptos vertidos en este capítulo alcanzan ya un cierto grado de complejidad. Este capítulo no debe abordarse si no se dominan completamente las técnicas de utilización de la instrucción **DIM** explicadas en el capítulo anterior.

- Si lo considera oportuno, dedíquele más de un día o numerosas sesiones a este tema, asegurándose de que el niño va asimilando paulatinamente las distintas fases del proceso de comparación y ordenación de elementos.

- Es importante que vea la diferencia entre el proceso normal de ordenación y el proceso del computador, que requiere la utilización de una casilla o variable extra. Asegúrese de que el niño ha captado la forma de utilización de la variable extra.

- Haga ver al niño que el ordenador siempre comparará el primer elemento de una serie con todos y cada uno de los restantes, para fijar la posición definitiva de ese elemento (en nuestro caso, el menor). Una vez fijado el menor de todos, comenzará de nuevo el proceso con el siguiente elemento de la serie.

Ordenando cosas

- La clave del programa radica en la estructura de las líneas 80 a 120. Dediquen todo el tiempo que sea necesario a la comprensión de los dos bucles. Repasen si es preciso los capítulos anteriores sobre **FOR/NEXT** y **DIM.**

- Recalque que las variables I y J de dichos bucles sirven únicamente para definir A (1), A (2), A (3)... y para que el ordenador pueda comparar sus valores. No obstante, es muy importante significar que A (1), A (2), etc... tienen cada una su valor, que ha sido asignado a través del teclado al ejecutar el programa.

- En nuestro ejemplo hemos ordenado los números de menor a mayor. Para ordenar de mayor a menor, basta con cambiar el operador de relación $>$ por el operador de relación $<$ en la línea 100.

INT (Algo más)

Recordarás que con **INT** obteníamos la parte entera de un número. Vamos a recordarlo con un programa:

```
10  FOR A = 10 TO 100 STEP 10
20  LET B = A/2.34
30  PRINT B
40  NEXT A
```

Este programa toma números de diez en diez, los divide entre 2,34 e imprime el resultado. Como podrás ver al correr el programa, los resultados son números decimales.

Si utilizamos **INT**, podemos quitar la parte decimal. Añade la línea:

```
25  LET C = INT (B)
```

Y cambia la línea 30 así:

```
30  PRINT B;" ";C
```

INT (Algo más)

Ejecuta el programa. Te queda solamente la parte entera de los resultados. Sin embargo, muchas veces es mejor redondear hacia arriba o hacia abajo en lugar de quitar la parte entera.

★ ¡Claro! Suponte que los datos que estemos utilizando sean importantes en algún cálculo. El número 29,9145299 es casi 30, pero con **INT** lo dejamos en 29.

Es muy fácil. Solamente tenemos que sumar 0,5 al número antes de quitarle la parte decimal. Por ejemplo:

```
  42.7350427    Como el número está cerca de 43,
+  0.5          al sumarle 0.5
--------------
  43.2350427    se pasa de 43
INT (43.2350427) y con INT queda sólo 43
```

```
  34.1880342    Como el número está cerca de 34
+  0.5          al sumarle 0.5
--------------
  34.6880342    no llega a 35
INT (34.6880342) y con INT se queda en 34.
```

INT (Algo más)

Modificando el programa de acuerdo a esta regla, añadiríamos:

```
27  LET R = INT (B + 0.5)
 5  PRINT "DECIMAL"; "  "; "ENTERO"; " "; "RE-
    DONDEADO"
```

y cambiaríamos la línea 30 por:

```
30  PRINT B; " "; C; "  "; R
```

Al correr el programa tendrás una pantalla con tres columnas:

Número Decimal	Parte Entera	Número Redondeado
4.27350427	4	4
8.54700855	8	9
.....	....	.....
.....	....	.....

Observa las diferencias entre una columna y otra.

INT (Algo más)

NOTAS PARA EL ADULTO

- Este capítulo no se ha limitado a comentar la función **INT** (ya conocida) sino una forma de utilizarla. Mucho más importante que su dominio o eventual aplicación posterior es el hecho de que el niño comprenda que utilizando nuestra imaginación hemos podido transformar la forma de trabajar de una función conocida en otra acorde a nuestros intereses.

- En efecto, la fórmula general **LET** A = **INT** (X + 0.5) es equivalente a una supuesta nueva función en BASIC que automáticamente redondearía cualquier número.

- De esta forma, podemos continuar combinando determinadas funciones del BASIC con expresiones matemáticas sencillas para buscar nuevas aplicaciones o soluciones a problemas concretos que pudieran surgir en el transcurso de nuestros programas.

- Así, si tuviéramos que determinar dentro de un programa cuándo un número entero es par o impar, utilizaríamos la fórmula:

```
10  IF INT (X/2) * 2 = X THEN PRINT " NUME-
    RO PAR"
```

la cuál se explica por sí sola.

INT (Algo más)

- Aparte de **INT** y **RND** existen otras funciones numéricas que no se mencionan pues rara vez serán utilizadas por el niño a este nivel. Resumimos a continuación la aplicación de dichas funciones:

SGN (N)	Indica el signo de N	SGN (3) hará aparecer en pantalla 1 SGN (–2) hará aparecer en pantalla –1 SGN (0) hará aparecer en pantalla 0
ABS (X)	Indica el valor absoluto de un número	ABS (–7) hará aparecer en pantalla 7 ABS (7) hará aparecer en pantalla 7
SQR (N)	Halla la raiz cuadrada de N	SQR (9) hará aparecer en pantalla 3

- El BASIC ofrece también diversas funciones logarítmicas, exponenciales y trigonométricas que facilitan considerablemente trabajos de cálculo. No se citarán en este libro.

Cadenas...

Aparte de las instrucciones (**PRINT**, **IF**, **GOTO**, **LET**, **FOR/NEXT**, **READ** ...) hasta ahora hemos conocido también funciones numéricas como **INT** o **RND**.

Vamos a conocer ahora algunas funciones de cadena o funciones que trabajan con textos. Son muy útiles, pues con ellas el BASIC te permite trabajar con palabras, frases, etc.

No las vamos a explicar todas. Solamente las que ahora te pueden resultar más interesantes. Además, no profundizaremos mucho en ellas. Lo principal es que sepas cómo funcionan y para qué pueden servir.

LEN nos dice el número de caracteres que tiene una cadena. Por ejemplo:

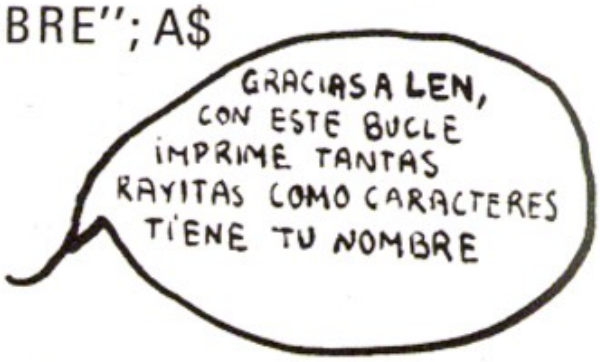
```
10  LET A$ = "BASIC PARA NIÑOS"
20  PRINT LEN (A$)
```

Cadenas...

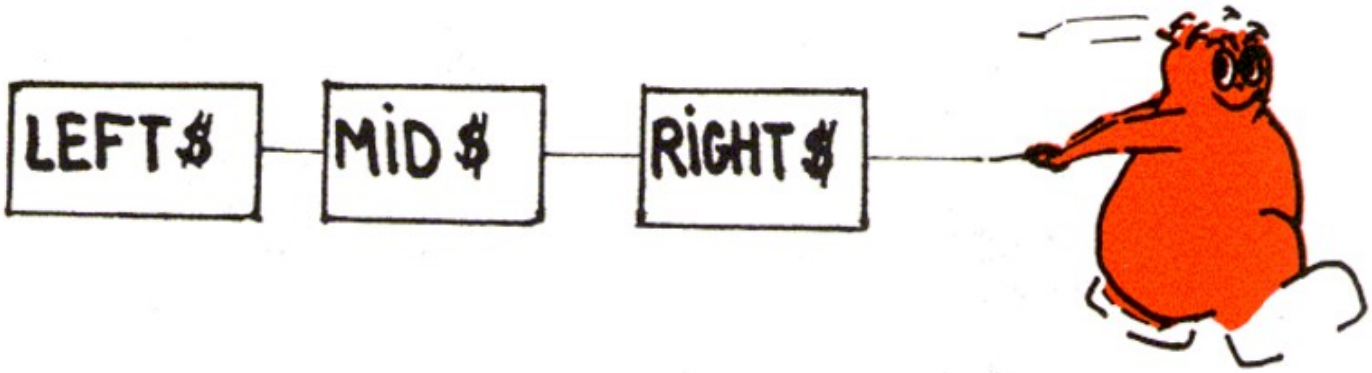
Al correr el programa, en tu pantalla aparecerá el número 16. **LEN** ha contado todos los espacios de la cadena alfanumérica A$, incluidos los espacios en blanco.

LEN puede ser muy útil cuando quieras subrayar una palabra que previamente no conoces. Por ejemplo, teclea:

```
10  INPUT "ESCRIBE TU NOMBRE"; A$
20  PRINT A$
30  FOR N = 1 TO LEN (A$)
40  PRINT " – ";
50  NEXT N
```



Toda la cadena aparecerá subrayada.



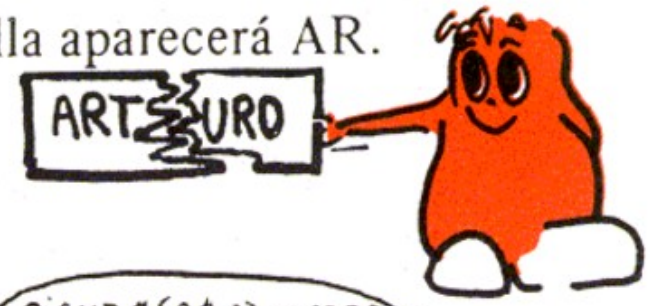
Estas funciones nos permiten obtener unos caracteres concretos sacándolos de una cadena. Por ejemplo **LEFT$** extrae los caracteres que tú le indiques, empezando a contar por la izquierda de la cadena. **RIGHT$** hace lo mismo, pero empezando por la derecha. Veamos un ejemplo: Teclea:

Cadenas...

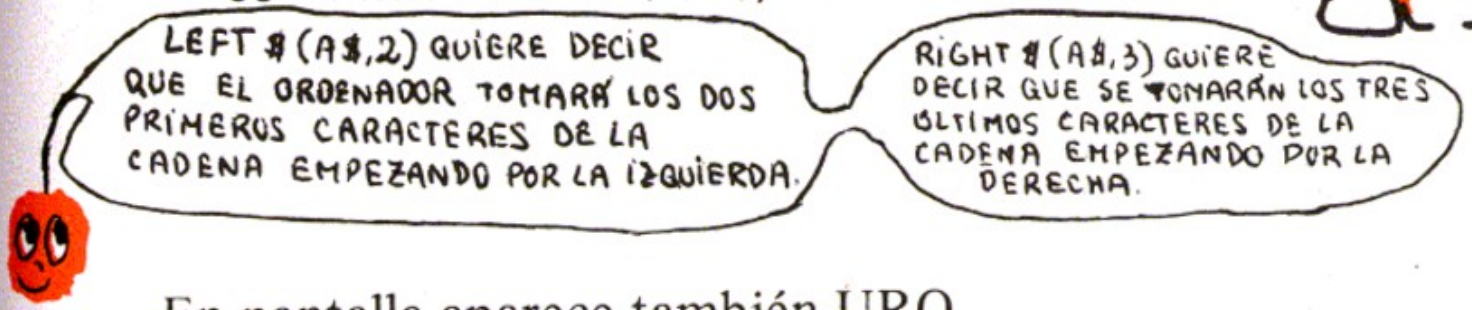

```
10  LET A$ = "ARTURO"
20  PRINT LEFT$ (A$, 2)
```

Corre el programa y en la pantalla aparecerá AR.

Añade la línea 30:




```
30  PRINT RIGHT$ (A$, 3)
```



En pantalla aparece también URO.

Hagamos un ejemplo de aplicación más útil:

```
10  INPUT "  ESCRIBE UN VERBO EN INFINITIVO.
    YO TE DIRE A QUE CONJUGACION PERTENECE";
    A$
20  LET B$ = RIGHT$ (A$, 2)
30  IF B$ = "AR" THEN PRINT A$; " "; "ES DE LA
    PRIMERA"
40  IF B$ = "ER" THEN PRINT A$; " "; "ES DE LA
    SEGUNDA"
50  IF B$ = "IR" THEN PRINT A$; " "; "ES DE LA
    TERCERA"
```

Cadenas...

Fíjate que en la línea 20 asignamos a la variable B$ el valor de los dos últimos caracteres de la variable A$. En informática se dice que B$ es una *subcadena* de A$.

MID$ es una función más perfeccionada que las anteriores. Fíjate en el ejemplo:

```
10  LET A$="ARTURO"
20  PRINT MID$ (A$, 2, 3)
```

Al ejecutar el programa en tu ordenador aparecerá RTU.

Ya conoces como funcionan **LEN**, **LEFT$**, **RIGHT$**, y **MID$**. Vamos a hacer un programa fenomenal combinando dos de estas funciones. Teclea:

```
10  INPUT "ESCRIBE UN TEXTO LARGO"; A$
20  CLS
30  LET L=LEN (A$)
40  FOR N=1 TO L
50  PRINT MID$ (A$, N, 1);
60  FOR X=1 TO 50: NEXT X
70  NEXT N
```

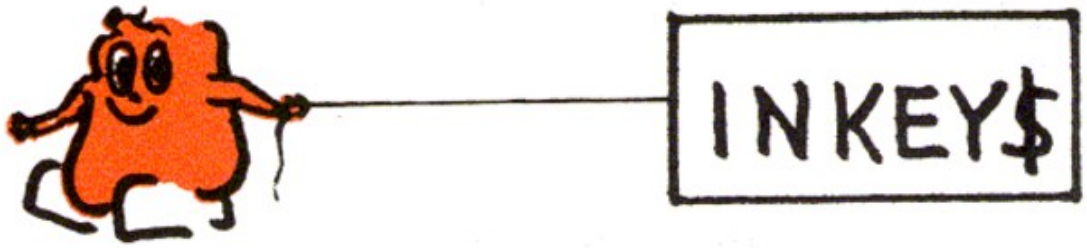
Cadenas...

En la línea 30 la variable L va a tomar como valor el número de caracteres de tu texto, incluidos los espacios en blanco.

En la línea 40 se inicia un bucle en el que N valdrá desde 1 hasta el número total de caracteres de tu texto.

En la línea 50 el ordenador toma de uno en uno los sucesivos caracteres de tu texto, empezando por el primero (puesto que N empieza valiendo 1) y los imprime uno al lado del otro.

El bucle de retardo de la línea 60 marcará la velocidad con que aparecerá tu texto en la pantalla.



Esta función nos permite asignar un valor a una variable alfanumérica a través de nuestro teclado sin necesidad de pulsar **ENTER** o **RETURN** después de hacerlo.

Por ejemplo, escribe:

```
 5 PRINT "PULSA CUALQUIER TECLA"
10 LET A$=INKEY$: IF A$="" THEN 10
```

Con esta línea el ordenador espera un valor que tú darás a través del teclado y que será asignado a la variable A$.

Cadenas...

En la segunda sentencia de la línea 10 le decimos al ordenador que mientras A$ no valga nada (es decir, mientras no pulses una tecla del teclado) continúe esperando en esa misma línea. Sigue:

```
20  PRINT " PULSASTE LA TECLA ..."; A$
30  GOTO 5
```

Corre el programa y prueba pulsando varias teclas. Verás que no es necesario pulsar **ENTER** para que el dato quede introducido en el programa.

El truco de la línea 10 es además muy práctico para parar un programa mientras se está ejecutando y que no continúe hasta que pulsemos una tecla.

★ ¿Recuerdas el programa del AJEDREZ de la página 29?

Con las líneas 10, 90 y 100 hacíamos que el ordenador imprimiera en pantalla los granos de cada nueva casilla siempre que tú teclearas S y después **ENTER** o **RETURN**.

Con **INKEY$** podemos hacer el programa más cómodo. Cambia las líneas 10 y 80 por:

```
10  PRINT " PULSA CUALQUIER TECLA PARA
    OTRA CASILLA"
80  LET A$ = INKEY$: IF A$ = " " THEN 80
```

Cadenas...

Quita las línea 90 y 100 del programa antiguo. El resultado es el mismo y la ejecución del programa más cómoda.

Puedes hacer lo mismo con el programa de la página 57, que nos daba la tabla de multiplicar (cambia la línea 90).

NOTAS PARA EL ADULTO

- Siga los ejemplos con el niño. En principio, éste no debe encontar ninguna dificultad especial en la comprensión de los programas expuestos y en el empleo de las funciones mencionadas.

- Algunos ordenadores (SINCLAIR) utilizan una expresión simplificada de las funciones **LEFT$, RIGHT$** y **MID$**. Así, considerando A$ = "ARTURO":

  PRINT A$ (2 TO 4) escribirá RTUR
  PRINT A$ (3) escribirá T
  PRINT A$ (TO 4) escribirá ARTU
  PRINT A$ (3 TO) escribirá TURO

  Así, la línea 20 del programa de la página 123 se escribirá:

  20 LET B$ = A$ (LEN A$ – 1 TO)

Cadenas...

la cual asigna igualmente a B$ los dos últimos caracteres de la variable A$.

Igualmente, la línea 50 del programa de la página 124 se escribirá:

```
50 PRINT A$ (N);
```

Tenga en cuenta también que deberá variar la colocación de los paréntesis en función del ordenador que utilice.

- Existen otras funciones de cadena, –cuya explicación escapa a los objetivos de este libro– como serían:

VAL (A$) Convierte la cadena numérica A$ en su valor numérico.

Ejemplo:

```
10  LET A$ = "234"
20  LET A = VAL (A$)
30  PRINT A
```

En la pantalla aparecerá 234, pero no como cadena, sino como número,

STR$ (A$) Convierte un número en una cadena de caracteres numéricos. Permite así aprovechar en un número las ventajas del tratamiento de textos a través de las funciones de cadena.

Ejemplo:

```
10  LET A =234
20  LET A$ =STR$ (A)
30  PRINT A$
```

En la pantalla aparecerá 234, pero no como número, sino como cadena. Prueba de ello es que añadiendo la línea:

```
40  PRINT RIGHT$ (A$, 2)
```

El ordenador escribirá:

```
234
34
```

- En algunos ordenadores, la función **INKEY$** se suple con la función equivalente **GET.** Aunque en sí, esta función no es una función de cadena, se emplea fundamentalmente en el tratamiento de cadenas. Así, **GET** permite introducir un carácter numérico y **GET$** un carácter alfanumérico. Obviamente, se suele utilizar más frecuentemente esta última, ya que de este modo el ordenador acepta cualquier tecla que se pulse.

  Esta función adopta la forma:

```
10  GET A$
```

CHR$ y ASC

Todos los números, letras, sígnos, etc, que aparecen en el teclado de tu ordenador se corresponden con unos números que tiene el ordenador memorizados.

La relación entre números y caracteres se llama *código*. Este código puede cambiar de un ordenador a otro.

Las funciones **CHR$** y **ASC** nos permiten ver la relación que existe entre esos números y los símbolos o caracteres del teclado.

Por ejemplo PRINT ASC ("A") nos dará el número de código que tiene el carácter "A" en tu ordenador.

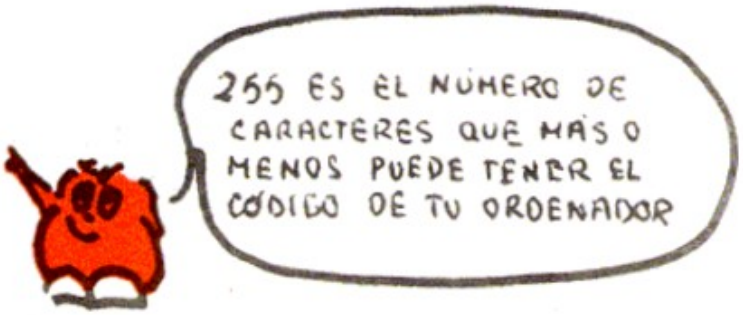
PRINT CHR$ (65) nos dará el carácter que corresponde al número 65 en el código de tu ordenador.

CHR$ y ASC

A veces es conveniente conocer la relación entre el carácter y el número de código. Vamos a ver un ejemplo, haciendo un programa para jugar.

★ Antes que nada, vamos a ver cuál es el código de tu ordenador. Teclea:

```
10  FOR N=1 TO 255
20  PRINT CHR$(N), N
30  NEXT N
```

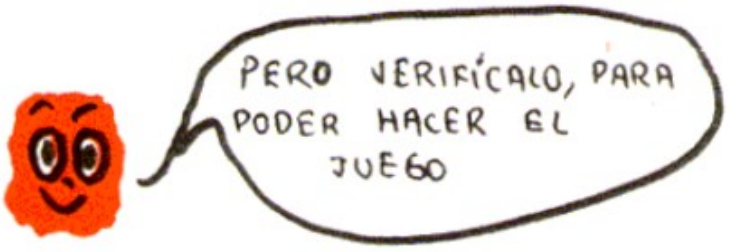


Si no te da tiempo a verlos todos, añade un bucle de retardo:

```
25  FOR I=1 TO 200: NEXT I
```

Corre el programa. En la columna de la izquierda te aparecerán los símbolos, números, caracteres y letras de tu teclado. En la columna de la derecha, sus correspondientes números de código.

Fíjate, por ejemplo, en el número de código que tienen las letras del abecedario. Posiblemente en tu ordenador las letras del abecedario se correspondan con los números 65 hasta el 90.



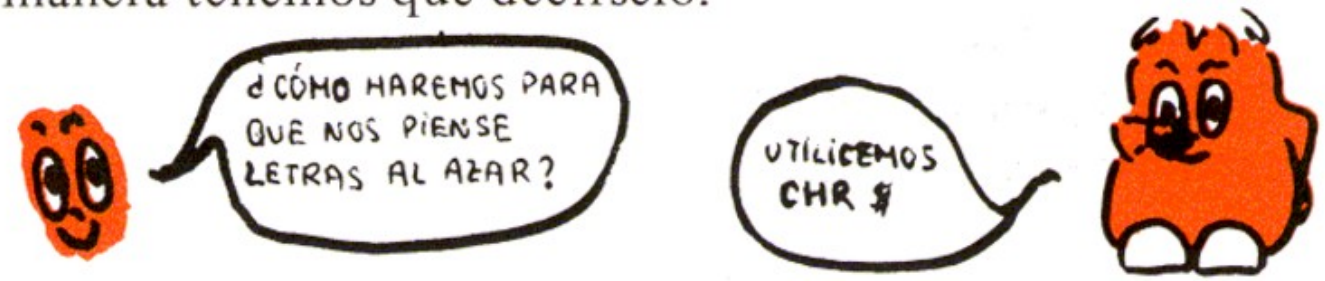
CHR$ y ASC

★ Ahora vamos con el juego. Primero lo haremos con las instrucciones que ya conocemos. Teclea:

```
10  LET A = INT ( RND (X) * 10)
20  INPUT "  HE PENSADO UN NUMERO DEL 1 AL 10.
    ADIVINALO. ESCRIBE TU NUMERO"; Z
30  IF A <> Z THEN 20
40  PRINT "ACERTASTE"
```

Este es un juego muy sencillo que sin duda ya entiendes.

★ Pero suponte que en lugar de jugar a adivinar números queremos que el ordenador piense en una letra. De alguna manera tenemos que decírselo.



¡Claro! **CHR$** transformará los números de código que le demos en sus correspondientes caracteres. Como sabemos que, en nuestro ejemplo, las letras del alfabeto van dentro del código desde el número 65 al 90, escribiremos:

```
10  LET A = INT (RND (X) * 26) + 65
15  LET A$ = CHR$ (A)
```

CHR$ y ASC

Fíjate en estas líneas. En la línea 10, hacemos que **RND** (X) * 26 genere números al azar entre 1 y 25 (que es el número de letras que tiene el alfabeto).

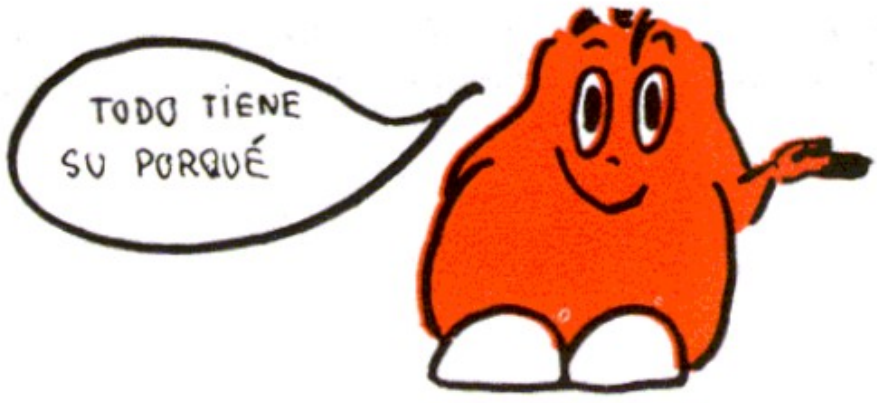
Después a ese número le sumamos 65 porque la letra A se corresponde con el número 65 del código. Así, al sumar 65 al número obtenido con **RND**, obtendremos al azar números comprendidos entre 65 y 90.

Con **CHR$** hacemos que el ordenador transforme dicho número de código a su letra correspondiente.

Añade:

```
20  INPUT "HE PENSADO UNA LETRA DEL ABECE-
    DARIO. INTENTA ADIVINARLA. ESCRIBE TU
    LETRA"; Z$
30  IF A$ <> Z$ THEN 20
40  PRINT "ACERTASTE"
```

Al correr el programa, el ordenador tomará al azar una letra cualquiera del abecedario. Como ves, **CHR$** puede ser útil.



CHR$ y ASC

Puedes completar el programa añadiendo contadores que te digan cuánto tardas en adivinarlo, dando puntuaciones, etc. Esto ya sabes como hacerlo.

NOTAS PARA EL ADULTO

- Estas instrucciones no serán frecuentemente utilizadas por el niño, pero las incluímos a fin de que, al menos, le resulten familiares. Al mismo tiempo se ha efectuado una introducción al código ASCII, que es un código internacional utilizado para representar informaciones alfanuméricas.

- El código ASCII está estandarizado y las informaciones alfanuméricas fundamentales suelen tener siempre el mismo código en todas las computadoras. No obstante, algunos fabricantes introducen ligeras variaciones por lo que le sugerimos que antes de abordar los programas de este capítulo consulte el manual de instrucciones de su computador, corra el programa de la página 131 o teclee **PRINT ASC** ("A") y **PRINT ASC** ("Z") para determinar el código del alfabeto de su ordenador.

CHR$ y ASC

- No olvide que no todos los ordenadores expresan de igual forma la instrucción **RND.** Posiblemente su ordenador precise escribir la línea 10 así:

```
10   LET A = RND (25) + 65
```

o bien:

```
10  LET A = INT (RND * 25) + 65
```

- Algunos ordenadores utilizan la función **CODE** en lugar de **ASC.** En cualquier caso no olvide que esta función nos indicará el número de código correspondiente al primer carácter de la expresión alfanumérica.

Así:

```
ASC ("ANTONIO") = 65
ASC ("A") = 65
```

- Por otra parte, algunos ordenadores escribirán:

```
ASC "ANTONIO"  en lugar de ASC ("ANTONIO")
CHR$ 65        en lugar de CHR$ (65)
```

Para programar bien

Después de leer este libro tú sabes ya muchas cosas sobre el BASIC. Serás capaz de hacer programas bastante amplios. Te gustará que tus programas sean buenos, funcionen, sean cómodos de utilizar, estén escritos con lógica...

Cuando ARTURO empezó a programar, tuvo algunas dificultades. No todos sus programas funcionaban a la primera. En la pantalla le aparecían avisos de error, etc.

En realidad, sólo con la práctica se consigue poco a poco programar con mayor seguridad. De todas formas, vamos a darte algunos consejos para que tus programas sean realmente buenos.

- Antes que nada conviene que tengas muy claro qué es lo que quieres hacer. Cuanto más domines un tema, más fácil te será hacer un programa relativo a ese tema.

Para programar bien

● Después te resultará útil hacer un diagrama de flujo. Con ello tendrás una visión general de cómo va a ser tu programa.

● Puede resultarte cómodo escribir las partes más complicadas del programa en un papel antes de pasarlo al ordenador.

● Al teclear tu programa debes tener cuidado cuando escribas los números de línea, para no repetirlos y evitar así que sin querer borres una de ellas.

● Ten mucho cuidado para no confundir las variables entre sí, y tampoco las numéricas y las alfanuméricas. Utiliza variables que te recuerden a lo que representan. Por ejemplo, L$ para Libros, R$ para Ropa, N para números, etc.

Para programar bien

- Recuerda que todas las sentencias **FOR** tienen que tener una sentencia **NEXT**.

- Vigila las instrucciones que le das al ordenador para que en un **IF ... THEN** vaya a la línea de programa adecuada.

- No te olvides de poner líneas "chivatas" para detectar los posibles errores de entrada de datos a través del teclado. Con ellas se detectan los errores justo después de haberlos cometido.

- Cuando introduzcas cambios, lista el programa para ver si el ordenador te los ha admitido correctamente, de la forma y en el lugar que querías.

- También es muy importante que tu programa quede muy claro para que pueda ser entendido por alguien que no lo haya escrito. Pon muchas sentencias **REM** para aclarar las cosas. Tú mismo las agradecerás cuando vuelvas a mirar el programa después de un tiempo.

Para programar bien

● Finalmente debes hacer que tu programa pueda ser utilizado muy fácilmente por cualquiera. Mete en el prógrama un menú que indique muy claramente las posibles opciones y los pasos a seguir.

● Tus sentencias **PRINT** deben ser muy claras. Por ejemplo, es mejor poner ¿CUANTOS CARAMELOS TIENE ARTURO? que poner ESCRIBA CARAMELOS.

Es mejor poner "EL RESULTADO DE LA MULTIPLICACION ES ..." que poner "RESULTADO ..."

En fin, tú mismo irás aprendiendo con el tiempo cómo hacer programas perfectos.

Programas

PRUEBA TUS REFLEJOS

Con este juego podrás probar tus reflejos. Intenta ganar a nuestro amigo ARTURO (que está muy entrenado). Y no te pongas nervioso...

```
 10  CLS
 20  LET Q$ = "PRUEBA TUS REFLEJOS. TE MOSTRA-
     RE UNA LETRA UN INSTANTE. EL TIEMPO EM-
     PEZARA A CORRER, Y SE PARARA CUANDO
     PULSES ESA LETRA. COMPITE CON TUS AMIGOS.
     EL QUE MENOS PUNTOS TENGA, GANA".
 30  LET L = LEN (Q$)
 40  FOR N = 1 TO L
 50  PRINT MID$ (Q$, N, 1);
 60  FOR W = 1 TO 20: NEXT W
 70  NEXT N
 80  FOR Z = 1 TO 1000: NEXT Z
 90  RANDOMIZE
100  LET X = 0
110  LET N = INT (RND (0) * 25) + 65
120  LET A$ = CHR$ (N)
130  PRINT: PRINT
140  PRINT A$
```

```
150  FOR Z=1 TO 100: NEXT Z
160  LET B$=INKEY$ : LET X=X+1: PRINT X :
     CLS : IF B$=" " THEN 160
170  IF B$=A$ THEN 200
180  IF B$<>A$ THEN PRINT " FALLASTE. INTEN-
     TALO DE NUEVO"
190  FOR Z=1 TO 200 : NEXT Z: GO TO 90
200  PRINT X : PRINT "TU PUNTUACION ES ";X
210  PRINT: PRINT: PRINT: PRINT: PRINT "¿OTRA
     VEZ? (S/N)"
220  LET R$=INKEY$ : IF R$=" " THEN 220
230  IF R$="S" THEN 90
240  IF R$="N" THEN STOP
250  IF R$<>"S" AND R$<>"N" THEN PRINT
     "CONTESTA BIEN"
260  GO TO 210
```

Programas

NOTAS PARA EL ADULTO

- De la línea 10 a la línea 70 se escribe "artísticamente" el texto de presentación en la pantalla. De la línea 90 a la línea 140 se generan aleatoriamente las letras del alfabeto, y se pone el contador a cero.

  De la línea 160 a la 180 el ordenador analiza la tecla pulsada y hace correr el marcador. De la línea 210 en adelante el ordenador permite volver a empezar.

- Conviene que añadan a este programa sentencias REM para aclarar los distintos pasos que sigue el ordenador.

- En este programa se practican de manera especial los bucles de retardo, las funciones MID$, INKEY$ y CHR$ y LEN, así como INT y RND. En algunos ordenadores, la línea 50 deberá escribirse así:

```
50  PRINT Q$ (N);
```

  y la línea 110 –que variará en función de su ordenador– se escribiría:

```
110  LET N = INT (RND * 25) + 65
```

  o también

```
110  LET N = RND (25) + 65
```

EL JUEGO DE LOS ESPIAS

Como ya has visto en las páginas anteriores, a nuestro amigo ARTURO le gusta mucho hacer de espía. Incluso ha desarrollado un programa para comunicarse en clave con su Cuartel General.

El sistema es muy sencillo. Cuando manda un mensaje, lo hace en un *código* especial. A cada letra del alfabeto le suma 5, y la letra que resulta es la que escribe. Así, en lugar de escribir una A, escribiría una F.

A	B	C	D	E	F	G	H	....
1	2	3	4	5	6	7	8	....

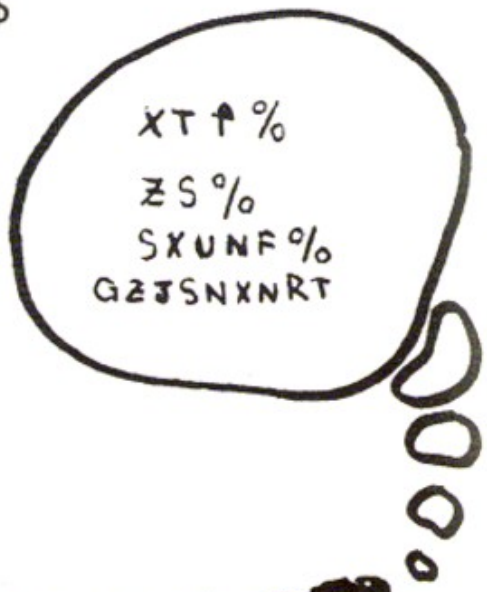
Cuando en el Cuartel General reciben un mensaje de ARTURO, lo que hacen es restarle 5 a cada letra del mensaje para poder entenderlo.

Logicamente, lo hacen con un ordenador, por ser más rápido. Tú mismo puedes establecer con tus amigos un sistema de mensajes en clave. Lo único que tienes que tener en cuenta es que hay que utilizar el mismo modelo de ordenador para luego poder descifrarlos.

Programas

El programa para codificar automáticamente los mensajes sería:

```
10  INPUT "ESCRIBE UNA FRASE"; F$
20  INPUT "INDICA LA CLAVE"; C
30  FOR X = 1 TO LEN F$
35  LET A$ = MID $ (F$, X, 1)
40  LET Z = ASC (A$) + C
50  PRINT CHR$ (Z);
60  NEXT X
70  PRINT
```



El programa para descifrarlos sería:

```
10  INPUT "ESCRIBIR FRASE CODIFICADA"; F$
20  INPUT "CLAVE SECRETA"; C
30  FOR X = 1 TO LEN F$
35  LET A$ = MID $ (F$, X, 1)
40  LET Z = ASC (A$) – C
50  PRINT CHR$ (Z);
60  NEXT X
```

En los ordenadores de SINCLAIR tendrás que cambiar la línea 35 por:

```
35  LET A$ = F$ (X TO)
```

y utilizar CODE en lugar de ASC.

Establece con tus amigos tu propio número de clave secreto y nadie podrá descubrir vuestros mensajes...

Claro que hay que tener mucho cuidado ... Si los Servicios de Contraespionaje conocen vuestro sistema de códigos, pueden averiguar incluso la clave con un programa como el que acabamos de ver, cambiando la línea 20 por

```
20  FOR C = 1 TO 24
```

y añadiendo las líneas

```
70  PRINT
80  NEXT C
90  STOP
```

Con ello el ordenador escribe en pantalla todas las posibles frases en clave. De ellas, una tendrá sentido. Intenta descubrir lo que dice ARTURO en la página anterior, utilizando este sistema de los servicios de contraespionaje.

Programas

UNA LIGA DE BALONCESTO

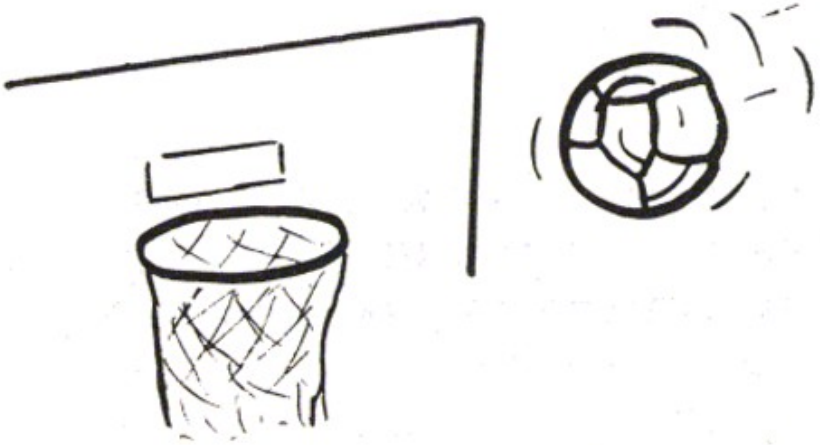
Como te habíamos prometido antes, haremos un programa muy sencillo para organizar una liga de baloncesto. ARTURO organizó su campeonato y este programa le fue muy útil pues con él tuvo la seguridad de que no se olvidaba de incluir ningún partido. Participaban solamente tres equipos.

La liga de ARTURO quedó así:

PARTIDOS DE LA LIGA

ROMPETECHOS contra LARGUIRUCHOS
ROMPETECHOS contra TROTAMUNDOS
LARGUIRUCHOS contra ROMPETECHOS
LARGUIRUCHOS contra TROTAMUNDOS
TROTAMUNDOS contra ROMPETECHOS
TROTAMUNDOS contra LARGUIRUCHOS

Programas



El programa sería:

```
 10   FOR A = 1 TO 3
 20   FOR B = 1 TO 3
 30   IF B = A THEN GOTO 100
 40   LET E = A
 50   GO SUB 200
 60   PRINT " contra ";
 70   LET E = B
 80   GO SUB 200
 90   PRINT
100   NEXT B
```

```
110  NEXT A
120  STOP
200  IF E=1 THEN PRINT " ROMPETECHOS";
210  IF E=2 THEN PRINT " LARGUIRUCHOS";
220  IF E=3 THEN PRINT " TROTAMUNDOS";
230  RETURN
```

Este programa solamente te serviría para organizar una liga como la de ARTURO. Si quieres organizar tu propia liga, puedes incluir el número de equipos que desees, utilizando la instrucción **DIM**, que conocimos en la página 86.

El programa sería:

```
  5  INPUT "CUANTOS EQUIPOS"; N
 10  DIM A$ (N)
 20  FOR J=1 TO N
 30  PRINT "ESCRIBE EL EQUIPO NO."; J
 40  INPUT A$ (J)
 50  CLS
 60  NEXT J
 70  CLS
 80  PRINT "PARTIDOS DE LA LIGA"
 90  PRINT
```

Programas

```
100  FOR X=1 TO N
110  FOR Y=1 TO N
120  IF X=Y THEN GOTO 150
130  PRINT A$(X); " contra ";
140  PRINT A$(Y)
150  NEXT Y
160  NEXT X
```

NOTAS PARA EL ADULTO

- El segundo programa permite la inclusión del número de equipos que se deseen, los cuales deberán ser introducidos a través del teclado durante la ejecución del programa.

- Algunos ordenadores (por ejemplo, SINCLAIR) exigirán una pequeña modificación en la línea 10 del programa, para definir el número máximo de caracteres que puede tener la variable alfanumérica. Así, se escribiría

```
10  DIM A$(N, 12)
```

Programas

LISTIN TELEFONICO

Puedes hacerte tu propio listín de teléfonos y direcciones. Hay varios sistemas, pero el que te ofrecemos a continuación es muy simple:

```
  5  CLS
 10  INPUT "CUANTOS NOMBRES TIENE TU
     LISTIN?"; M
 20  PRINT "LISTIN": PRINT
 30  PRINT "BUSQUEDA POR :"
 40  PRINT "NOMBRES . . . . 1"
 50  PRINT "DOMICILIO"  . . 2"
 60  PRINT "FIN  . . . . . .3"
 70  INPUT "QUE OPCION DESEAS ?"; A
 80  CLS
 90  IF A = 1 THEN GOTO 2000
100  IF A = 2 THEN GOTO 3000
110  IF A = 3 THEN STOP
120  PRINT "TECLEASTE MAL TU OPCION ":
     GOTO 30
2000  INPUT "ESCRIBE EL NOMBRE"; E$
2010  RESTORE
2020  FOR X = 1 TO M
2030  READ N$, C$, T$
```

```
2040  IF E$=N$ THEN GOTO 2070
2050  NEXT X
2060  IF E$<>N$ THEN PRINT " NO CONOZCO A ";
      E$: GOTO 2000
2070  PRINT N$: PRINT C$: PRINT T$
2080  INPUT "OTRO NOMBRE? (S/N) "; Z$
2090  IF Z$="S" THEN GOTO 2000
2100  INPUT " VUELTA AL MENU? (S/N) "; J$
2110  IF J$="S" THEN GOTO 30
2120  STOP
3000  INPUT " DOMICILIO"; E$
3010  RESTORE
3020  FOR X=1 TO M
3030  READ N$, C$, T$
3040  IF E$=C$ THEN GOTO 3070
3050  NEXT X
3060  IF E$<>C$ THEN PRINT "NO CONOZCO LA
      CALLE"; E$: GOTO 3000
3070  PRINT N$: PRINT C$: PRINT T$
3080  INPUT "OTRA CALLE? (S/N)"; Z$
3090  IF Z$="S" THEN GOTO 3000
3100  INPUT " VUELTA AL MENU? (S/N)"; J$
3110  IF J$="S" THEN GOTO 30
3120  STOP
```

Programas

```
5000  DATA LUIS, INDEPENDENCIA 345,  2-34-56-83
5010  DATA FERNANDO, CENTRAL 23,  2-67-98-99
5020  DATA ANA, PLAZA COLON 3,  2-55-54-42
5030  DATA ARTURO, AV. DE LA INFORMATICA 2,
      0-11-01-10
```

Los datos de tus amigos los tienes que introducir a través de sentencias DATA. Recuerda los consejos que te dábamos en la página 85 sobre las sentencias DATA. Si añades más nombres, no te olvides de cambiar la línea 10.

ORDENACION ALFABETICA

```
10  CLS
20  INPUT "CUANTOS NOMBRES QUIERES
    ORDENAR?"; N
30  DIM A$ (N)
40  FOR I = 1 TO N
50  INPUT A$ (I)
60  PRINT A$ (I)
70  NEXT I
```

```
 80  PRINT: PRINT
 90  FOR I = 1 TO N - 1
100  FOR J = I + 1 TO N
110  IF A$ (I) > A$ (J) THEN LET E$ = A$ (I): LET
     A$ (I) = A$ (J): LET A$ (J) = E$
120  NEXT J
130  NEXT I
140  FOR I = 1 TO N
150  PRINT A$ (I)
160  NEXT I
170  END
```

Este programa te permite ordenar alfabéticamente los nombres de todos tus amigos. Como ves, el proceso es el mismo que para la ordenación numérica, sólo que en esta ocasión utilizamos variables alfanuméricas.

Como ya has visto en capítulos anteriores, el ordenador también clasifica las letras y puede analizar cuál es mayor. Con las palabras hace exactamente igual. Primero compara la primera letra y si ésta es igual, compara la segunda, etc...

Por ejemplo,

ANTONIO < CARLOS
ARTURO > ANTONIO

Programas

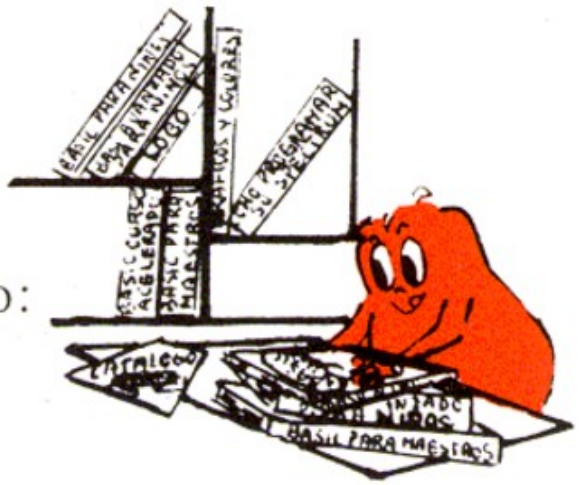
CONTROL DE BIBLIOTECA

Es muy conveniente que te acostumbres a tener tus libros ordenados y clasificados de forma que puedas localizarlos con facilidad cuando los necesites.

El ordenador te puede ayudar mucho en esto. El programa que te ofrecemos a continuación es muy eficaz y muy sencillo, aunque sea un poco largo.

Puedes utilizar hasta dos dígitos para las materias. Por ejemplo,

MEDICINA 14
LITERATURA 21



y uno para las submaterias. Por ejemplo:

NOVELAS 2

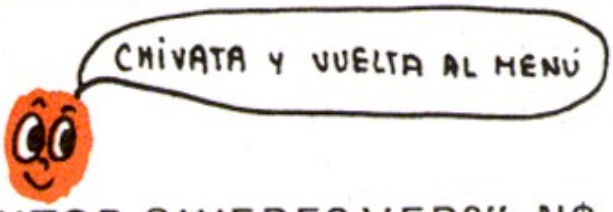
Puedes utilizar tu propia clasificación o, mejor, utilizar un sistema de clasificación que utilizan muchas bibliotecas. Pídeselo a tu maestro.

Utilizando la clave para la materia y la submateria podremos ordenar nuestros libros de manera lógica. Para clasificar una *novela* de un *autor español o hispanoamericano* utilizaríamos el código 212. También podemos poner el título y el autor. Por ejemplo:

Programas

```
 80  FOR I=1 TO N
 90  READ M$(I), S$(I), A$(I), T$(I)
100  LET A(I)=LEN(A$(I))
110  LET T(I)=LEN(T$(I))
120  FOR J=A(I)+1 TO 20
130  LET A$(I)=A$(I)+" "
140  NEXT J
150  FOR K=T(I)+1 TO 20
160  LET T$(I)=T$(I)+" "
170  NEXT K
180  LET L$(I)=M$(I)+S$(I)+A$(I)+T$(I)
190  NEXT I
1000 FOR J=1 TO N-1
1010 FOR K=J+1 TO N
1020 IF L$(J)<L$(K) THEN GOTO 1060
1030 LET E$=L$(J)
1040 LET L$(J)=L$(K)
1050 LET L$(K)=E$
1060 NEXT K
1070 NEXT J
1080 PRINT: PRINT: PRINT
1090 PRINT "BUSQUEDA POR MATERIAS PULSA 1"
1100 PRINT "BUSQUEDA POR AUTOR PULSA 2"
1110 PRINT "BUSQUEDA POR TITULO PULSA 3"
1120 INPUT "QUE OPCION DESEAS?"; O
```


```
1130  IF O = 1 THEN GOTO 1200
1140  IF O = 2 THEN GOTO 1400
1150  IF O = 3 THEN GOTO 1600
1160  IF O <> 1 AND O <> 2 AND O <> 3 THEN GOTO
      1120
1170  GOTO 1090
1200  CLS
1210  INPUT "PON LOS DOS DIGITOS DE LA MATERIA
      QUE QUIERES VER"; X$
1220  FOR I = 1 TO N
1230  IF X$ = LEFT$ (L$ (I), 2) THEN GOTO 1250
1240  GOTO 1260
1250  PRINT L$ (I)
1260  NEXT I
1270  INPUT "QUIERES VOLVER AL MENU? (S/N)";
      Y$
1280  IF Y$ = "N" THEN STOP
1290  IF Y$ <> "S" AND Y$ <> "N" THEN GOTO 1270
1300  CLS
1310  GO TO 1080
1400  CLS
1410  INPUT " QUE AUTOR QUIERES VER?"; N$
1420  LET Q = LEN (N$)
1430  FOR W = Q + 1 TO 20
```



Programas

SE COMPLETA EL NOMBRE DEL AUTOR HASTA 20 CARACTERES

```
1440  LET N$ = N$ + " "
1450  NEXT W
1460  FOR I = 1 TO N
1470  IF MID$ (L$ (I), 4, 20) = N$ THEN GOTO 1490
1480  GO TO 1500
1490  PRINT L$ (I)
1500  NEXT I
```

SE BUSCAN E IMPRIMEN TODOS LOS LIBROS DE ESE AUTOR

```
1510  INPUT "QUIERES VOLVER AL MENU? (S/N)"; Y$
1520  IF Y$ = "N" THEN STOP
1530  IF Y$ <> "S" AND Y$ <> "N" THEN GOTO 1510
1540  CLS
1550  GOTO 1090
1600  CLS
1610  INPUT "  ESCRIBE EL TITULO"; Z$
1620  LET P = LEN (Z$)
1630  FOR R = P + 1 TO 20
1640  LET Z$ = Z$ + " "
1650  NEXT R
1660  FOR I = 1 TO N
1670  IF Z$ = RIGHT$ (L$ (I), 20) THEN GOTO 1690
1680  GOTO 1700
1690  PRINT L$ (I)
1700  NEXT I
```

SE COMPLEMENTA EL TÍTULO DEL LIBRO HASTA 20 CARACTERES

SE BUSCAN E IMPRIMEN TODOS LOS DATOS RELATIVOS A ESE TÍTULO

Programas

```
1710  INPUT "QUIERES VOLVER AL MENU? (S/N)"; Y$
1720  IF Y$ = "N" THEN STOP
1730  IF Y$ <> "S" AND Y$ <> "N" THEN GOTO 1710
1740  CLS
1750  GOTO 1090
5000  DATA 3
5010  DATA 21,2,JUAN RAMON JIMENEZ, PLATERO Y
      YO
5020  DATA 21,2,G. GARCIA MARQUEZ, LA HOJARAS-
      CA
5030  DATA 08,5,MARTINEZ, MATEMATICAS E.G.B.
```

NOTAS PARA EL ADULTO

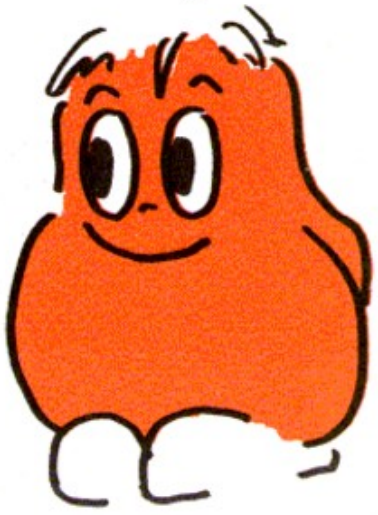
- Este es un programa-tipo, escrito en un BASIC convencional. Como ya conocen, será necesario introducir algunos cambios mínimos para poder ejecutarlo en determinados ordenadores. Deben tener en cuenta todas las observaciones efectuadas a lo largo de los capítulos anteriores a este respecto.

- En especial, reiteramos la importancia de escribir adecuadamente las sentencias DATA, efectuar correctamente los dimensionados y modificar donde sea preciso aquellas instrucciones como LEFT$, MID$ y RIGHT$ que no existen en determinados ordenadores.

AHORA ME VOY. JUNTOS HEMOS APRENDIDO MUCHO SOBRE BASIC. SI TE HA GUSTADO, PUEDES ESTUDIAR MAS. LA VERDAD ES QUE NUNCA SE TERMINA DE APRENDER, PERO CON LO QUE YA SABES ES SUFICIENTE PARA DESARROLLAR PROGRAMAS MUY AVANZADOS.

LO PRINCIPAL ES QUE HAGAS TUS PROPIOS PROGRAMAS EMPLEANDO TU IMAGINACION, Y QUE COMPRENDAS QUE TAN IMPORTANTE COMO PROGRAMAR ES SABER UTILIZAR EL ORDENADOR COMO UNA HERRAMIENTA DE TRABAJO MAS.

DE TODAS FORMAS, SI QUIERES LEER MAS LIBROS SOBRE EL TEMA, TE DIRE LOS QUE MAS ME HAN GUSTADO:



BELLIDO y SANCHEZ. *BASIC para maestros.* Ed. PARANINFO, 1985. Madrid.

BELLIDO. *Cómo programar su SPECTRUM:* 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.

Basic avanzado para Niños

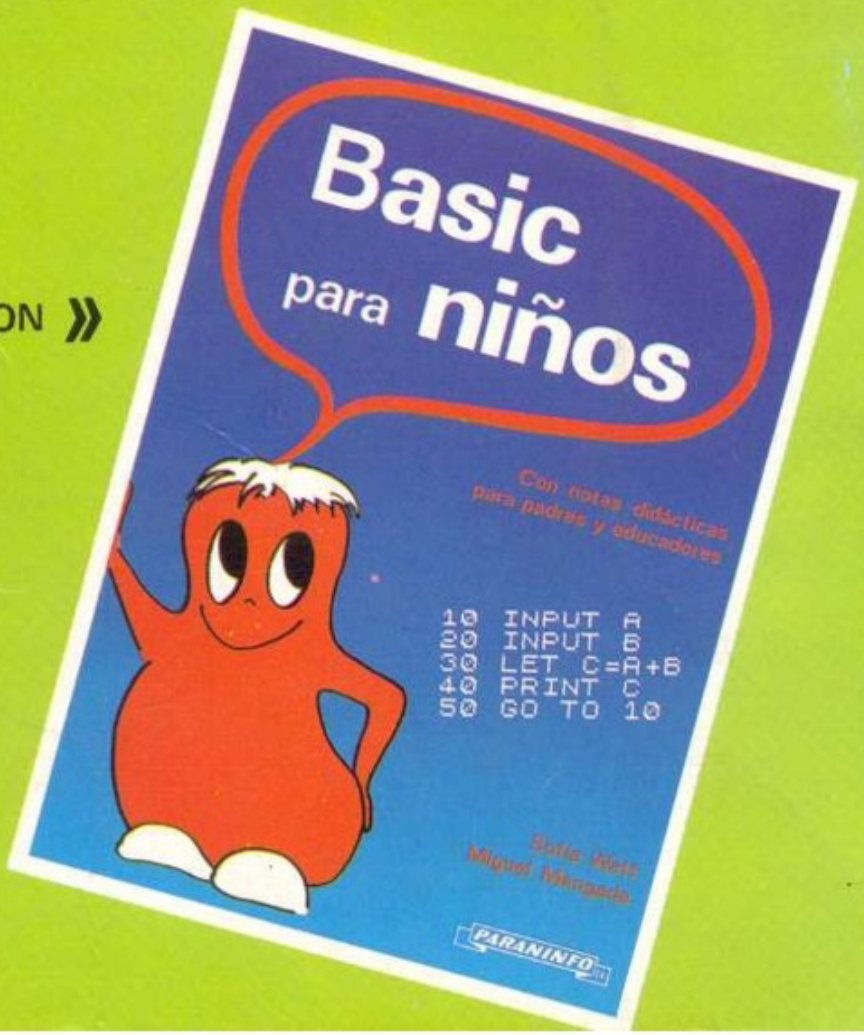
- **¿Qué es el código ASCII?**
- **¿Cómo y cuándo utilizar CHR$?**
- **¿Para qué sirven LEN, MID$, LEFT$, RIGHT$?**
- **¿Qué es eso de los operadores lógicos AND y OR?**

Este libro responde a éstas y otras muchas cuestiones de la programación en BASIC. Y lo hace de una manera clara y sencilla, de forma que cualquier niño podrá —utilizando su ordenador— desde organizarse su propio fichero de direcciones hasta listar alfabéticamente una relación nominal, pasando por autoevaluar sus conocimientos escolares u organizar su propia liga de baloncesto ...

Apenas son necesarios unos mínimos conocimientos de programación. A lo largo del texto, numerosas notas aclaratorias, comentarios y guías didácticas explican las dudas que puedan surgirle al niño y amplían sus conocimientos.

Utilizando ejemplos sencillos y directamente relacionados con su entorno, este libro confiere al niño un sólido conocimiento de la programación en BASIC.

EN ESTA MISMA COLECCION »



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

ISBN: 84-283-1365-2