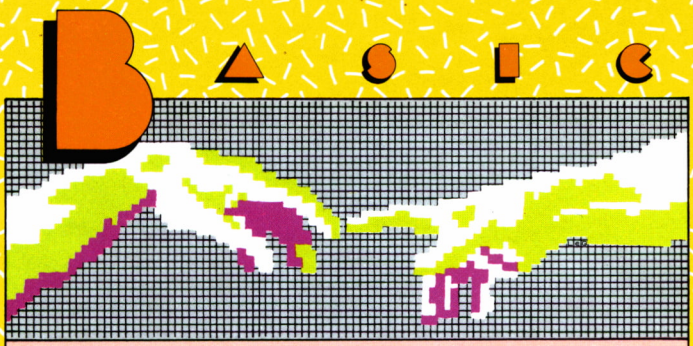


BASIC
PROGRAMACIÓN

E. Lowy Frutos – A.E. Gallego Palomero
S. Mansilla Romo

sm

EQUIPO DE AUTORES

Ernesto Lowy Frutos
Agregado de Física y Química de I.B.

A. Enrique Gallego Palomero
Agregado de Matemáticas de I.B.

Serafín Mansilla Romo
Licenciado en Matemáticas

EQUIPO EDITORIAL

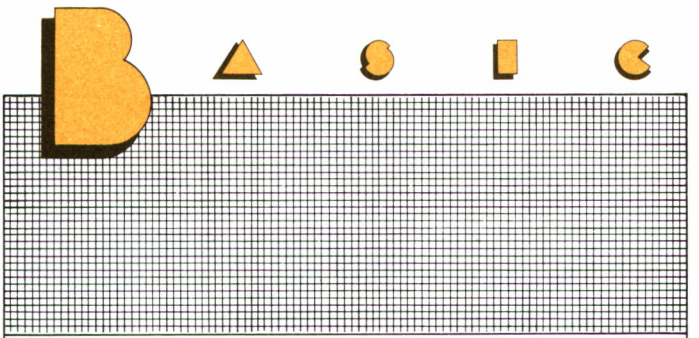
Coordinación: José Luis Robles Cid

Maqueta: José Ugarte

Dibujos: Ernesto Cobeño

Fotografías: José Manuel Navia, AISA

Portada: Alfonso Ruano y José Luis Cortés



BASIC
PROGRAMACIÓN

E. Lowy Frutos – A.E. Gallego Palomero
S. Mansilla Romo

I.S.B.N.: 84-348-1591-5 / Depósito legal: M. 15.226-1985 / Fotocomposición: Grafilia, S. L.
Imprime: Gráficas VELASCO, S. A. - Antonio de Cabezón, 13 - 28034 Madrid.
Impreso en España - Printed in Spain.

PRESENTACIÓN

La presencia cada vez más frecuente de los microordenadores en el hogar y en la escuela ha hecho posible nuevas y sugestivas formas de aprendizaje e incluso de diversión, accesibles a cualquier persona con un nivel básico de conocimientos. Pero para aprovechar estas posibilidades es necesario comunicarse con el microordenador, y esta comunicación se establece mediante un lenguaje especial que «entienden» la mayoría de los microordenadores: *el lenguaje BASIC,*

Este lenguaje se puede aprender con relativa facilidad y tiene la ventaja de permitir desde un principio una comunicación directa con el microordenador, a modo de diálogo. Esto aporta una cierta satisfacción personal, al ir descubriendo la versatilidad de la máquina con la que se está aprendiendo o jugando, y al surgir progresivamente una especie de desafío en el dominio de la misma.

En este contexto se ha escrito este libro, cuyo principal objetivo es facilitar el aprendizaje del lenguaje BASIC. Siguiéndole paso a paso se logran adquirir las técnicas fundamentales de la programación, y perseverando en su estudio se puede llegar a saber programar en breve tiempo.

En el proceso de elaboración del libro se ha seguido el *método inductivo,* que lleva a la comprensión y utilización de las instrucciones BASIC partiendo de ejemplos sencillos y concretos. También se ha procurado dar a esta obra un *enfoque práctico,* lo que contribuye a que el propio lector pueda realizar sin ayuda, pequeños programas sobre diversos asuntos vinculados a su experiencia diaria.

El enfoque práctico se ha reforzado incluyendo en cada uno de los temas la sección EJERCICIOS RESUELTOS, cuyos programas pueden servir de referencia para realizar los abundantes y variados programas propuestos en los EJERCICIOS DE RECAPITULACIÓN, si bien hay que advertir que no existen reglas fijas para programar.

LOS AUTORES

Tanto los programas con los que se explican las distintas instrucciones BASIC como los incluidos en la sección EJERCICIOS RESUELTOS se pueden ejecutar en todos los microordenadores.

Índice

1. El microordenador

1. Una nueva máquina: el microordenador

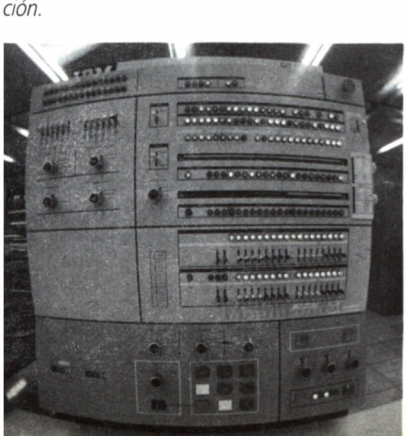
En sentido amplio, una máquina es un dispositivo o un conjunto de dispositivos que cumple una determinada función y que contribuye a satisfacer nuestras necesidades. Por ejemplo, una carretilla es una máquina que nos ayuda a transportar pesos con menor esfuerzo y mayor comodidad; los medios de transporte, como el automóvil, el tren y el avión, son máquinas que nos permiten viajar más rápidamente; los medios de comunicación, como el teléfono, la radio y la televisión, son también máquinas que nos ayudan a relacionarnos mejor.

En estos últimos años, en la sociedad moderna en que vivimos, ha surgido una nueva necesidad como consecuencia del desarrollo alcanzado por la economía y la ciencia: la de ***mecanizar*** y ***automatizar*** una parte de nuestro trabajo intelectual. Ello supone ***organizar, almacenar*** y ***procesar*** una gran cantidad de datos y, en general, todo tipo de información.

Las máquinas que nos ayudan a realizar esta función son los ***ordenadores,*** que han nacido y evolucionado paralelamente al desarrollo de la electrónica, y cuya capacidad en el tratamiento de la información supera la propia capacidad humana.

Los ***microordenadores*** son un tipo de ordenadores que han aparecido últimamente, en la década de los 70. Estas máquinas, tan pequeñas y potentes, han surgido como consecuencia del espectacular desarrollo alcanzado por la tecnología electrónica, concretamente en el campo de los ***circuitos integrados.***

*Un **ordenador** es una máquina que nos ayuda a mecanizar parte de nuestro trabajo intelectual: organizar, almacenar y procesar grandes cantidades de datos y, en general, de información.*



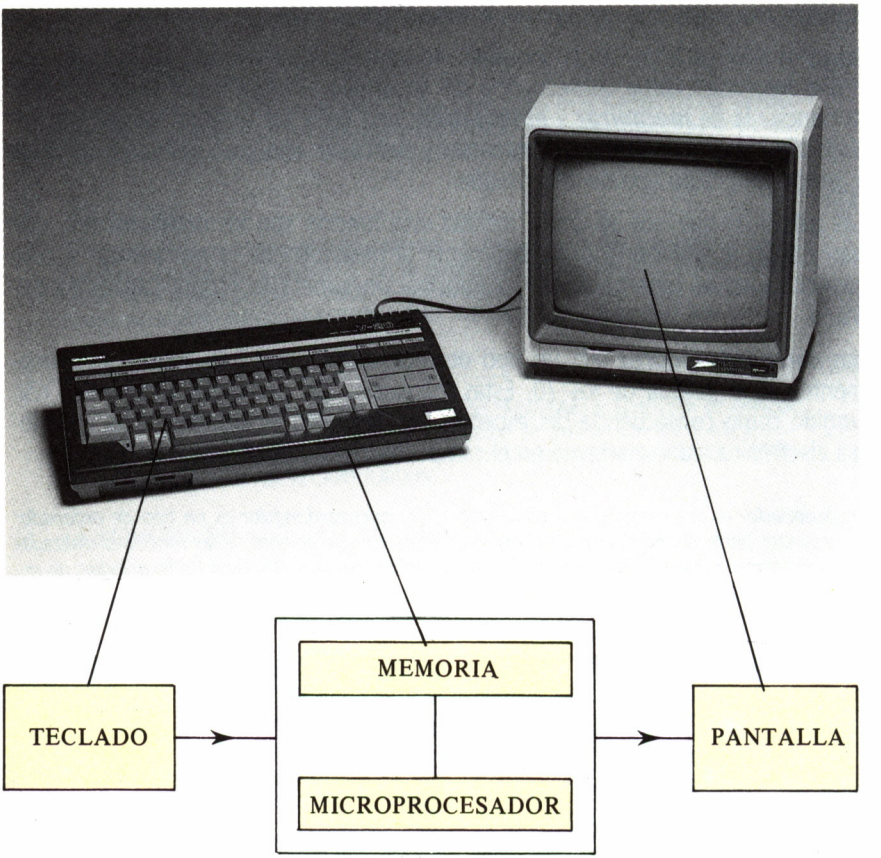
*Un **microordenador** es un tipo de ordenador que cumple muchas de las funciones realizadas por éste y que, gracias a las tecnologías de miniaturización electrónica, ocupa un reducidísimo espacio.*

Por el hecho de que los microordenadores puedan realizar complicadas funciones, como resolver cálculos larguísimos a elevada velocidad y sin cansarse ni equivocarse, mucha gente cree que son máquinas misteriosas con capacidades poco menos que sobrehumanas. Nada de eso: un microordenador, como cualquier máquina, *no piensa, hay que pensar por él;* y no sirve para nada si no se le ***programa adecuadamente,*** es decir, si no se le indica lo que tiene que hacer.

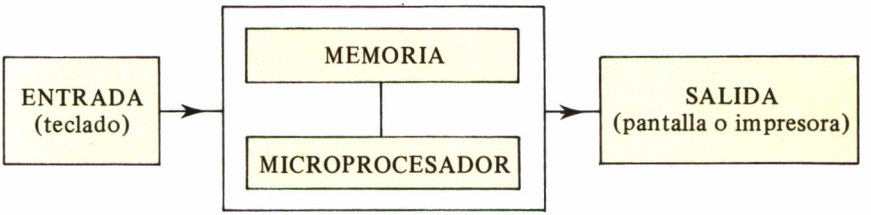
2. Partes de un microordenador

Al observar un microordenador, como el de la fotografía, distinguimos a simple vista una consola que lleva incorporado un ***teclado,*** y una ***pantalla*** como la de una televisión (en muchos microordenadores puede servir incluso un televisor).

El teclado y la pantalla son partes importantes del microordenador, pero lo esencial del mismo se encuentra en el interior de la consola: es la MEMORIA y el MICROPROCESADOR.



En resumen, podemos decir que un microordenador se compone de cuatro partes unidas entre sí: la **ENTRADA** (teclado), la **MEMORIA**, el **MICROPROCESADOR** y la **SALIDA** (pantalla).



El dispositivo de salida en lugar de una pantalla (o monitor) puede ser una impresora.

3. Cómo mecanizamos nosotros algunos procesos intelectuales y cómo lo hace el microordenador

Antes de estudiar cómo trabaja un microordenador, vamos a analizar, en un ejemplo concreto, cómo se mecaniza un cierto proceso intelectual.

Supongamos que deseamos obtener el valor de Y, para X = 8 en la expresión:

$$Y = X + 6$$

Aunque este proceso lo llevamos a cabo casi instantáneamente, en realidad lo realizamos siguiendo varios pasos sucesivos:

— ***Leemos*** la expresión Y = X + 6.

— ***Retenemos en la memoria*** la expresión anterior.
Para la variable X, que en principio no tiene asignado ningún valor, reservamos en nuestra memoria una «posición», por decirlo de alguna manera, en espera de asignarle un valor numérico. También reservamos otra «posición» de memoria para la variable Y, que al final almacenará el resultado de la operación X + 6.

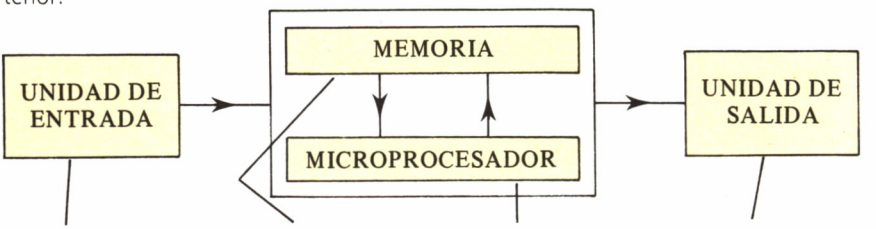
— ***Calculamos*** la suma 8 + 6 y el resultado se lo asignamos a la variable Y.

— ***Escribimos*** el resultado de la suma.

En concreto, las funciones básicas llevadas a cabo en este proceso son:

* LECTURA
* MEMORIZACIÓN
* CÁLCULO
* ESCRITURA

Pues bien, el microordenador, para realizar este proceso, cumple esencialmente las mismas funciones, mediante los dispositivos electrónicos que componen las UNIDADES que muestra la figura, y que ya hemos mencionado en el apartado anterior.



LECTURA	MEMORIZACIÓN	CÁLCULO	ESCRITURA
En la unidad de entrada (a través del teclado) se lee la expresión $Y = X + 6$.	*La expresión leída pasa a la memoria. Las variables X e Y ocupan posiciones de memoria, en las que se alojarán los valores 8 y 14, respectivamente.*	*El microprocesador calcula la suma 8 + 6 y transfiere el resultado 14 a la memoria, almacenándose en Y.*	*En la unidad de salida (en la pantalla o impresora) aparece el resultado 14.*

Para el microordenador, leer un dato, una variable, un operador o una expresión consiste en teclearla. Inmediatamente después de ser tecleada pasa a ocupar una posición en la memoria.

El microprocesador cumple dos funciones: dirige y sincroniza las distintas etapas del proceso y realiza operaciones aritméticas y lógicas.

Para que el microordenador lleve a buen término todo el proceso hay que indicarle el orden de los distintos pasos que tiene que seguir. En el ejemplo concreto que venimos analizando, el orden de las instrucciones podría ser el siguiente:

1. INTRODUCIR X
   Inmediatamente después de ser tecleada esta instrucción (leída) se almacena en la memoria.

2. CALCULAR EL VALOR DE Y EN LA EXPRESION Y = X + 6
   Como en la instrucción 1, inmediatamente después de ser tecleada esta instrucción se almacena en la memoria.

3. ESCRIBIR O IMPRIMIR EL VALOR DE Y
   Esta instrucción también se almacena en la memoria.

4. TERMINAR (FIN)
   Esta instrucción, que se almacena en la memoria, indica al microordenador el fin del proceso.

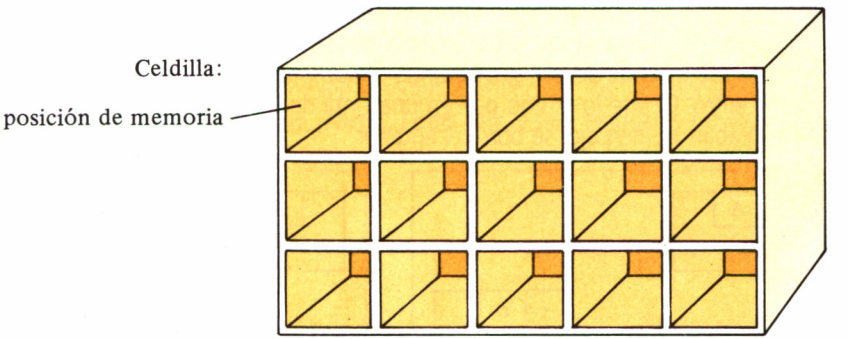
Este conjunto de órdenes o instrucciones constituye el **PROGRAMA** de trabajo, que seguirá ciegamente el microordenador.

Para obtener el resultado de la suma y su escritura, hay que ordenar al microordenador que **ejecute** el programa y, a continuación, introducir un valor para la variable X (por ejemplo, 8).

4. La memoria

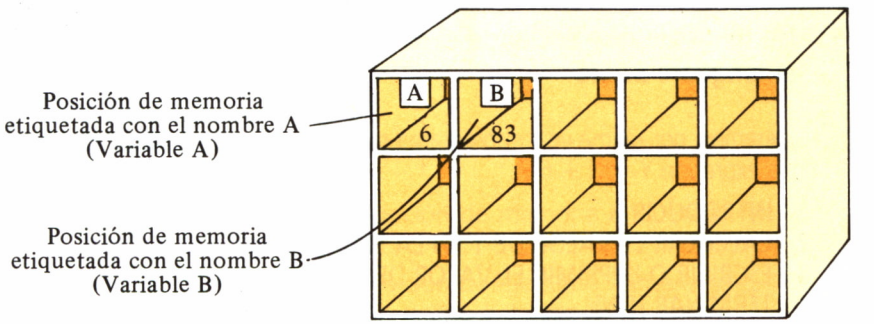
- La memoria de un microordenador es un dispositivo electrónico que cumple las funciones de ***almacenar*** datos y demás información, ***conservarla*** y ***devolverla*** cuando se la pida.

  La memoria se puede comparar con un panal con muchas celdillas, llamadas ***posiciones de memoria.***



Al introducir un dato en la memoria es necesario dar un nombre a la posición de memoria en la que se guarda dicho dato, pues de lo contrario sería imposible encontrarlo posteriormente.

Una posición de memoria etiquetada con un nombre se denomina ***variable***, ya que puede contener datos diferentes (variables) en el curso de la ejecución de un programa.



La posición de memoria etiquetada con A almacena en este instante el dato numérico 6 y la etiquetada con B contiene el dato numérico 83.

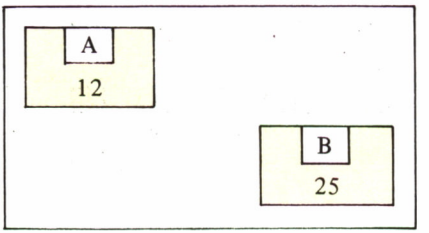
Esto suele indicarse así:

A =6
B = 83

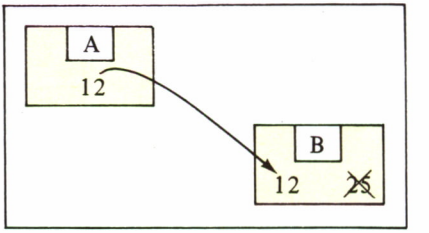
(El nombre de la variable se escribe a la izquierda y el valor que tiene en un determinado instante se escribe a la derecha, separados por el signo igual.)

- *¿Qué ocurre cuando se extrae un determinado dato de una posición de memoria?* Cuando de una posición de memoria se extrae un dato, bien para realizar un cálculo con él, bien para llevarlo a otra posición de memoria, dicho dato ***se conserva*** en la posición en que se encontraba. Pasa algo parecido a lo que ocurre cuando se reproduce un mensaje grabado en una cinta de casete: por el hecho de reproducirlo para escucharlo, no se borra la cinta.

- *¿Y qué ocurre cuando se introduce un dato en una posición de memoria?* Cuando se introduce un dato en una posición de memoria, lo que contenía hasta ese instante se borra, quedando almacenado en la misma el último dato que entró. Ocurre lo mismo que en una cinta de casete grabada: en el instante de grabar un mensaje, se borra el anterior.



La posición A contiene el dato 12 y la B, el 25.
(A = 12, B = 25.)



Cuando se transfiere el 12 de la posición A a la B se ***conserva*** *el 12; en la B, en cambio, se* ***borra*** *el 25.*
(A = 12, B = 12.)

5. Un lenguaje que entiende el microordenador: el lenguaje BASIC

Consideremos, por última vez, el programa mediante el cual se calcula el valor de Y, en la expresión Y = X + 6.

```
1 INTRODUCIR X
2 CALCULAR EL VALOR DE Y EN LA EXPRESION  Y = X + 6
3 ESCRIBIR O IMPRIMIR EL VALOR DE Y
4 TERMINAR (FIN)
```

El lenguaje en el cual está escrito este programa no lo "entiende" el microordenador. Sin embargo, si lo escribimos de la siguiente forma

```
1 INPUT X
2 LET Y = X + 6
3 PRINT Y
4 END
```

sí lo "entiende" el microordenador. El lenguaje en el cual están escritas estas instrucciones se llama BASIC; esta palabra se forma con las iniciales de las palabras de la expresión inglesa "**B**eginner's **A**ll-purpose **S**imbolic **I**nstruction **C**ode" (código de instrucción simbólica para principiantes para todos los usos). Este lenguaje utiliza algunas palabras en inglés, símbolos de puntuación habituales y funciones matemáticas corrientes.

A lo largo de este libro explicaremos detalladamente el significado de los términos que aparecen en el programa anterior y de otros más que forman parte del lenguaje BASIC.

Existen otros lenguajes que también pueden "entender" los microordenadores, como el ***Pascal, Cobol, Fortran, Forth, Logo,*** etc. Estos lenguajes no los estudiaremos en este libro.

EJERCICIOS RESUELTOS

1. Dos posiciones de memoria están etiquetadas con A y B y almacenan los datos 1 y 7, respectivamente (A = 1 y B = 7). Explicar el significado de las siguientes expresiones:

   a) A = 1
   b) B = 7
   c) B = A

   Solución:

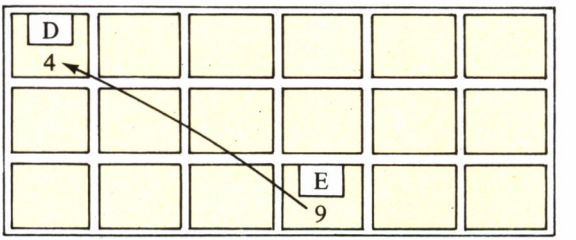
   a) A = 1, significa que la posición de memoria etiquetada con A almacena el dato 1.

   b) B = 7, significa que la posición de memoria etiquetada con B almacena el dato 7.

   c) B = A, significa que la posición de memoria etiquetada con B almacena el contenido de A, o sea, 1. Luego B =1.

2. Si D = 4 y E = 9, indicar la situación final de estas posiciones de memoria si el contenido de E se lleva a D (D = E).

Solución:



El contenido de D se borra al "entrar" el contenido de E, o sea, 9, mientras el contenido de esta última posición permanece.

Se tiene, entonces, la siguiente situación final:

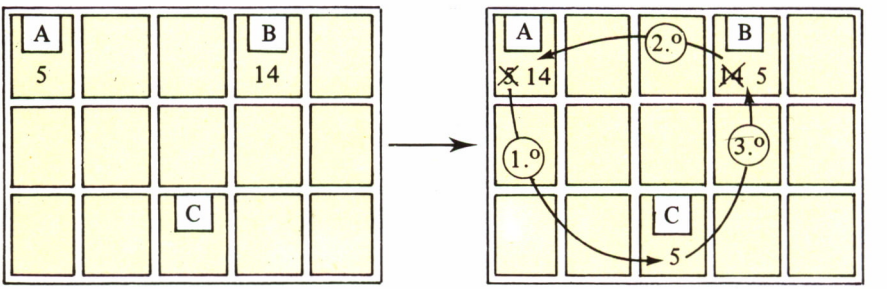
D = 9 y E = 9

3. Sea A = 5 y B = 14. Indicar el procedimiento para traspasar el contenido de A a B y el de B a A.

Solución:

Si se traspasa el contenido de A a B, el de B ya no se puede traspasar a A, porque el contenido inicial de B queda borrado.

Para poder hacer el traspaso es necesario utilizar una tercera posición de memoria; la llamaremos C. Realizamos el movimiento de datos indicado en el esquema:



A = 5
B = 14

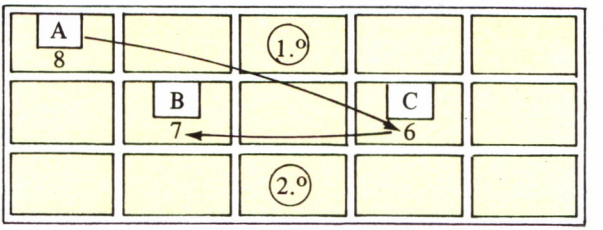
1.º C = A (C = 5)

2.º A = B (A = 14)

3.º B = C (B = 5)

EJERCICIOS DE RECAPITULACIÓN

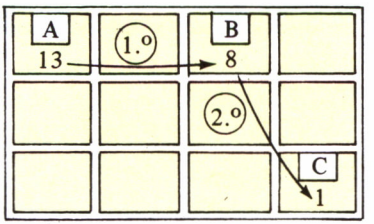
1. 1. Indica el contenido de las posiciones de memoria, después hacer los movimientos de datos indicados en la figura.

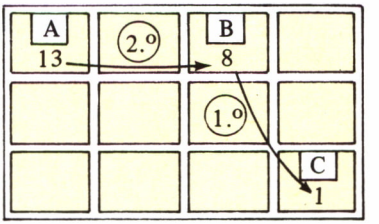


1. 2. Dos posiciones de memoria están etiquetadas con P y Q. Interpreta el significado de la expresión Q = P.

1. 3. Las posiciones de memoria A y B almacenan los datos 3 y 7, respectivamente. Indica el contenido final de A y B, si se hace el movimiento de datos B = A.

1. 4. Expresa en símbolos los movimientos de datos representados en las figuras y el contenido final en las posiciones de memoria A, B y C.





1. 5. Las posiciones A, B y C contienen los datos 5, 8 y 13, respectivamente. Indica el contenido de estas posiciones, después de hacer los siguientes movimientos.

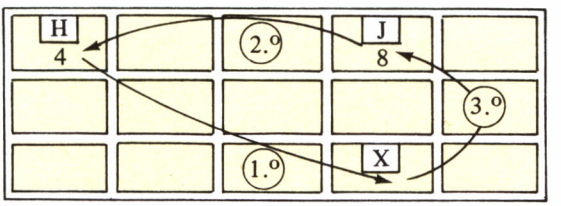
a) B = A
C = A

b) B = A
B = C

c) B = C
A = B
C = B

d) A = B
B = C
C = A

1. 6. Averigua el contenido de las posiciones de memoria H y J, después de hacer los movimientos de datos indicados en la figura.



1. 7. Escribe las expresiones que describen el procedimiento para traspasar el contenido de la posición de memoria R a la posición S y el de S a R.

1. 8. Analiza si el siguiente procedimiento logra traspasar el contenido de A a B y el de B a A. (Almacena datos en A y B para ayudarte a resolver el problema.)

1.° C = B
2.° A = C
3.° B = A

1. 9. ¿Qué es necesario hacer con un programa antes de ser ejecutado por el microordenador?

1.10. En tu propio lenguaje, escribe un programa que calcule el valor de Z, para X = 3 e Y = 17, en la expresión Z = X + Y.

1.11. Escribe las funciones que realizan las distintas unidades de un microordenador para realizar el cálculo del ejercicio anterior. (Dibuja un esquema como el incluido en el apartado 3 de este capítulo.)

1.12. En tu propio lenguaje, escribe un programa que calcule, para A = 5, P = 5 * A y Q = 2,5 * A + 7 y halle la suma P + Q de tal manera que ésta quede almacenada en X.

2. Comenzando a programar en BASIC

1. Un primer programa en BASIC

Vamos a hacer un programa que calcule el valor de P en la expresión P = 2,53X − − 20,358, para un cierto valor de X.

En nuestro propio lenguaje, el programa sería el siguiente:

```
1  INTRODUCIR X
2  HACER P = 2,53 X − 20,358
3  ESCRIBIR O IMPRIMIR EL VALOR DE P
4  TERMINAR (FIN)
```

Pero, como ya sabemos, estas instrucciones expresadas en nuestro lenguaje no las entiende el microordenador; las escribiremos, por tanto, en lenguaje **BASIC.**

```
10  INPUT X
20  LET P = 2.53 * X − 20.358
30  PRINT P
40  END
```

Observaciones sobre este programa

1.ª Este programa se compone de cuatro órdenes o instrucciones numeradas de 10 en 10. El número que precede a cada instrucción indica el orden en que el microordenador debe ejecutarlas.

Las instrucciones se pueden ordenar con los números 1, 2, 3, ..., pero se aconseja numerarlas de 10 en 10, pues ello permite añadir en cualquier momento instrucciones olvidadas con numeración intermedia.

2.ª En cada instrucción, después del número aparece una palabra o expresión inglesa, que indica el tipo de acción que debe ejecutar el microordenador:

INPUT: introduce
LET: deja, haz
PRINT: escribe, imprime
END: fin (*)

3.ª La instrucción

20 LET P = 2.53 * X − 20.358

merece una atención especial.

— El número **20** es el número de instrucción.

— **LET** es la orden escrita en BASIC.

— Las letras P y X son variables (posiciones de memoria etiquetadas).

— El símbolo igual (=), aquí no tiene el mismo significado que en aritmética. Cuando el microordenador se "encuentra" con este símbolo, interpreta que el resultado de la operación escrita a la derecha del mismo lo tiene que almacenar en la variable escrita a la izquierda.

— El signo * es el operador aritmético de multiplicar.

— Los números 2,53 y 20,358 en BASIC se escriben 2.53 y 20.358, respectivamente: observar que la coma decimal está sustituida por un punto y que el cero está tachado (Ø). Respecto a esto último señalaremos que la letra O y el cero, 0, están en teclas distintas y ambos símbolos se distinguen porque el cero está tachado, (Ø) (**).

2. Cómo se almacena un programa en memoria

Una vez escrito un programa en el papel, se teclea cada instrucción pulsando a continuación la tecla [ENTER] (o [RETURN] o [NEW LINE], etc., según el microordenador).

En este libro utilizaremos [ENTER].

(*) En algunos microordenadores no es necesario poner la instrucción **END**; incluso, el microordenador *ZX Spectrum* no dispone de esta instrucción.

(**) En este libro no tacharemos el cero porque en tipos de imprenta ambos símbolos están perfectamente diferenciados (el cero es alargado, 0, y la letra es más redonda, O).

Cuando se acciona esta tecla la instrucción pasa automáticamente a la memoria. ***Si no se acciona, el microordenador no se da por enterado de la instrucción escrita;*** es decir, no pasa a la memoria. Por ejemplo, si se teclea

10 INPUT X

y no se pulsa la tecla ENTER, la instrucción queda escrita en la pantalla pero no pasa a la memoria. En cambio, si se teclea

10 INPUT X ENTER

la instrucción aparece en la pantalla y, además, se almacena en la memoria.

Si tecleamos el programa anterior, lo veremos así en la pantalla. (¡NO OLVIDARSE DE PULSAR LA TECLA ENTER DESPUÉS DE CADA INSTRUCCIÓN!)

```
10  INPUT X
20  LET P = 2.53 * X - 20.358
30  PRINT P
40  END
```

3. Cómo se procesa un programa almacenado en memoria

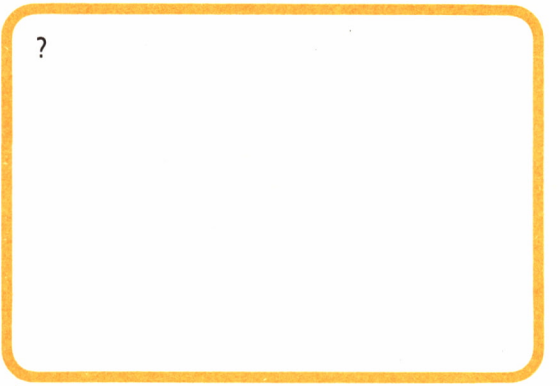
El programa lo tenemos ya almacenado en la memoria. *¿Qué tenemos que hacer para que el microordenador lo ejecute, es decir, para que interprete las instrucciones y realice los cálculos?*

Para que el microordenador ejecute el programa se escribe la palabra **RUN,** pulsando a continuación la tecla ENTER.

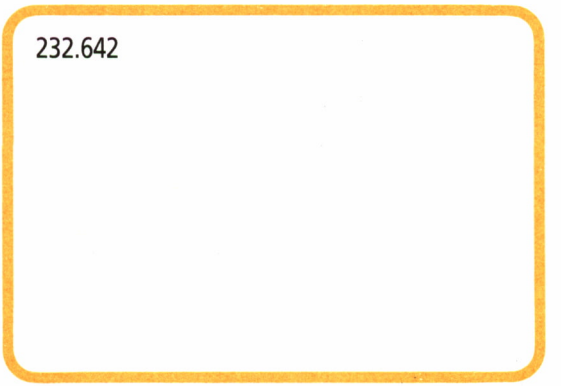
Al ejecutar la primera instrucción

10 INPUT X

aparece en pantalla un signo de interrogación (en algunos microordenadores, L), deteniéndose momentáneamente la ejecución del programa.



Con esta interrogación (o con L) el microordenador pregunta qué valor deseamos asignar a la variable X de la primera instrucción. Si, por ejemplo, tecleamos 100 (y ENTER), la ejecución continúa, apareciendo inmediatamente en la pantalla el valor correspondiente a P:



¿Cómo ha llegado el microordenador al resultado 232.642?

Con el valor 100 almacenado en la posición de memoria, representada por X, el microordenador ejecuta la instrucción

20 LET P = 2.53 * X – 20.358

Esta instrucción indica a la máquina que realice las operaciones aritméticas indicadas a la derecha del signo igual y el resultado lo almacene en la posición de memoria representada por P, situada a la izquierda del signo igual. Es decir, calcula 2.53 * 100 – 20.358 y el resultado 232.642 lo almacena en P.

A continuación se ejecuta la instrucción

30 PRINT P

que ordena al microordenador escriba o imprima en la pantalla el valor almacenado en la posición de memoria representada por P.

Finalmente, se ejecuta la instrucción

40 END

que indica al microordenador que el programa ha terminado.

Ejercicios

2.1. Escribe un programa que calcule la expresión Y = 0,23 X + 0,8 e imprima el resultado.

2.2. Explica cómo harías para ejecutar el programa anterior.
¿Se detiene la ejecución en algún momento? ¿Qué tienes que hacer para que continúe la ejecución?

2.3. Escribe un programa que calcule la suma de dos números e imprima el resultado.

2.4. Escribe un programa que calcule el producto de dos números e imprima el resultado.

4. Operadores aritméticos

En la instrucción **20** del programa del apartado anterior.

20 LET P = 2.53 * X – 20.358

se ha utilizado el operador * (correspondiente a ×) para multiplicar la constante 2.53 y la variable X. En lenguaje BASIC se utilizan los operadores aritméticos habituales, cuyos símbolos se transcriben a continuación:

Paréntesis: ( )
Elevar a una potencia: ↑ o ∧
Producto: *, y cociente: /
Suma: + , y resta: –

En una expresión aritmética, primero se calculan las potencias (*), después los productos y cocientes y, finalmente, las sumas y restas. Si se desea alterar este orden de prioridad se utilizan los paréntesis. Por último, después de haber seguido todos estos criterios, la expresión se evalúa de izquierda a derecha.

(*) El *ZX Spectrum* sólo calcula potencias de números positivo.

Por ejemplo, las dos expresiones siguientes contienen los mismos números y, sin embargo, su valor es distinto debido a la prioridad impuesta por los paréntesis:

a) $10-8/2 \uparrow 3+5-5*4=10-8/8+5-20=10-1+5-20=-6.$
b) $10-(8/2) \uparrow 3+5-5*4=10-4 \uparrow 3+5-5*4=10-64+5-20=-69.$

En la expresión a) el primer cálculo que hay que realizar es la potencia $2 \uparrow 3 = 2^3 = 8$ y en la expresión b) el paréntesis indica que lo primero que hay que calcular es el cociente encerrado entre paréntesis (8 / 2).

5. Cómo se cambian las instrucciones de un programa y cómo se borran

- Según se va escribiendo una instrucción de un programa es normal que se teclee un símbolo por otro. Si uno se da cuenta del error cometido antes de pulsar la tecla ENTER, es decir, antes de cargar en memoria la instrucción, se puede corregir el error utilizando una tecla que tienen todos los microordenadores y, aunque recibe distinto nombre cumple la misma función: ***hace retroceder el cursor, borrando, uno a uno, los símbolos escritos.***

- Si deseamos hacer algún cambio en una determinada instrucción de un programa cargado ya en la memoria procedemos de la siguiente manera:

  — Averiguamos primero el número de la instrucción o instrucciones que deseamos modificar. Para ello se hace aparecer en la pantalla el ***listado*** del programa, tecleando la instrucción **LIST** (y ENTER).

  Supongamos que en este momento está cargado en la memoria del microordenador el programa anterior. Utilizando la instrucción **LIST** se obtiene su listado en la pantalla:

```
10  INPUT X
20  LET P = 2.53 * X - 20.358
30  PRINT P
40  END
```

— Si ahora deseamos modificar la instrucción 20, por ejemplo, basta teclearla de nuevo, introduciendo las modificaciones. Si escribimos

20 LET P = 0.5 * X + 16

y pulsamos la tecla ENTER, se borra de la memoria la primitiva instrucción **20** y queda cargada la **20** modificada. Para verificarlo, listamos de nuevo el programa y obtendremos en pantalla el nuevo listado:

```
10  INPUT X
20  LET P = .5 * X + 16
30  PRINT P
40  END
```

(*)

- Cuando se conoce de antemano el número de la instrucción que se desea modificar, no es necesario listar el programa; es suficiente con teclearla de nuevo. Sin embargo, para mayor seguridad conviene listarla, para lo cual se teclea la instrucción **LIST** y, a continuación, el número de la instrucción (y ENTER). Así, tecleando **LIST 20** (y ENTER) se consigue hacer aparecer en la pantalla la instrucción 20. Para modificarla, ésta se teclea de nuevo.
- Si se desea eliminar una instrucción de un programa basta con teclear el número de la misma (y ENTER). Por ejemplo, si escribimos 30 (y ENTER) y, a continuación, listamos el programa anterior, se obtiene lo siguiente en la pantalla:

```
10  INPUT X
20  LET P = .5 * X + 16
40  END
```

(*) Los números decimales con parte entera nula, el microordenador los escribe omitiendo el cero.

En este listado comprobamos que la instrucción **30 PRINT P** no se encuentra ya cargada en la memoria.

- Para borrar todo el contenido de la memoria se escribe **NEW** (nuevo programa) y se pulsa a continuación la tecla ENTER.

  También se puede borrar todo el contenido de la memoria desconectando y conectando el microordenador.

- Para borrar solamente lo que está escrito en la pantalla, la mayoría de los microordenadores utilizan la instrucción **CLS**. Esta instrucción no borra lo almacenado en la memoria.

6. Errores de sintaxis

Tecleamos el siguiente programa que calcula el producto de un número X por 13.85 e imprime el resultado:

```
10  IMPUT X
20  LET Y = 13.85 * X
30  PRINT Y
40  END
```

Inmediatamente después de escribir **RUN** (y ENTER) para ejecutar el programa, el microordenador, en lugar de imprimir el resultado, nos informa que hemos cometido un error al teclear el programa, imprimiendo un mensaje en la pantalla:

```
? SINTAXIS ERROR IN 10
```

Este mensaje significa ***error de sintaxis en 10.*** Efectivamente, si nos fijamos bien en la instrucción 10 observamos que escribimos **IMPUT**, con M, en lugar de **INPUT**. (El mensaje de error puede ser diferente, según el microordenador; puede ser incluso un número clave.)

Para corregir este error se modifica la instrución 10, mediante el procedimiento explicado en el apartado anterior.

Ejercicios

2.5. ¿Qué aparecerá en la pantalla al ejecutar el siguiente programa?

```
10 INPUT A
20 INPUT B
30 LET C = A ↑ B
40 PRONT C
50 END
```

2.6. Explica cómo harías para corregir el programa anterior, de tal manera que al ejecutarlo llegue a imprimir el valor de C.

2.7. Escribe un programa que calcule el cuadrado de un número e imprima el resultado.

2.8. Indica qué calcula el siguiente programa:

```
10 INPUT A
20 INPUT B
30 INPUT C
40 LET X = A * B * C
50 PRINT X
60 END
```

¿Cuántas veces se detiene la ejecución de este programa? ¿Por qué?

2.9. El siguiente programa debe sumar dos números y a la suma multiplicarla por 5. ¿Logra hacer este cálculo? En caso negativo modifica el programa.

```
10 INPUT H
20 INPUT J
30 LET P = H + J * 5
40 PRINT P
50 END
```

7. Resumen de las instrucciones estudiadas en este capítulo

- LET
  - — ***Significado:*** asigna.
  - — ***Formato:*** n LET {variable} = {expresión}

— ***Función:*** calcula el valor de la expresión y lo asigna a la variable.
— ***Observación:*** en la mayoría de los microordenadores no es necesario escribir la palabra **LET**, basta escribir la variable:
Ejemplo: la instrucción
40 LET Y = X ↑ 2 + 0.5 * X
la aceptan escrita así:
40 Y = X ↑ 2 + 0.5 * X

- END

— ***Significado:*** fin.
— ***Formato:*** n **END**.
— ***Función:*** indica la terminación de un programa.
— ***Observación:*** algunos microordenadores no exigen incluir esta instrucción, por ejemplo, el *ZX Spectrum*.

- ENTER , RETURN , NEW LINE , u otra equivalente

— ***Significado:*** entra, vuelve, línea nueva...
— ***Función:*** hace que el dato o instrucción tecleados se almacene en la memoria, o que se ejecute una instrucción del tipo **NEW, LIST**, etc.
— ***Observación:*** en prácticamente todos los microordenadores esta orden está en una sola tecla, lo que permite ahorrar tiempo de tecleado, ya que para que cumpla su función no es necesario teclear todas sus letras, sólamente hay que pulsar dicha tecla.

- RUN

— ***Significado:*** corre, ejecuta.
— ***Función:*** inicia la ejecución de un programa. (En el argot informático suele decirse ***"correr"*** el programa.)

- NEW

— ***Significado:*** nuevo.
— ***Función:*** borra todo el contenido de la memoria.

- LIST

— ***Significado:*** lista.
— ***Función:*** hace aparecer en la pantalla el listado de un programa almacenado en la memoria.

— ***Observaciones:***

- — Para listar todo el programa se escribe **LIST**.
- — Para listar una sola instrucción se escribe **LIST** n, donde n es el número de dicha instrucción.
- — Para listar una serie seguida de instrucciones se escribe **LIST** n_1 - n_2, donde n_1 es la primera instrucción de la serie y n_2, la última.
- — Para listar las instrucciones comprendidas entre n y el final, se escribe **LIST** n-
- — Para listar todas las instrucciones anteriores a la n se escribe **LIST**-n.

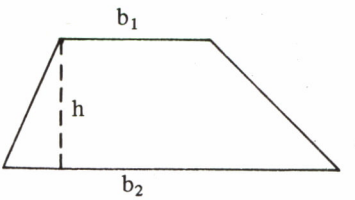
NOTA: Las instrucciones **INPUT** y **PRINT** se estudiarán con más detalle en el siguiente capítulo.

EJERCICIOS RESUELTOS

1. Hacer un programa que calcule el área de un trapecio.

Solución:

— La fórmula del área del trapecio es $A = \frac{b_1 + b_2}{2} \cdot h$



— En el programa habrá que poner tres instrucciones **INPUT**, una para introducir b_1, otra para introducir b_2, y una tercera para introducir h.

```
10 INPUT B1
20 INPUT B2
30 INPUT H
40 LET A = ((B1 + B2)/2) * H
50 PRINT A
60 END
```

— Después de teclear **RUN** y ENTER el microordenador pedirá tres valores (aparecerán en la pantalla sucesivamente tres interrogaciones).

2. Sírvete del microordenador para obtener valores de Y que te ayuden a representar gráficamente la parábola de ecuación $y = 0{,}5\,X^2 - 4{,}27\,X + 4{,}58$.

Solución:

Lo primero que conviene calcular es la abscisa del vértice, mediante la fórmula:

$X_v = -\frac{b}{2a}$. Sustituyendo a y b se obtiene 4.27 (*).

A continuación, hacemos un programa para calcular valores de Y correspondientes a los valores de X.

```
10 INPUT X
20 LET Y = 0.5 * X ↑ 2 - 4.27 * X + 4.58
30 PRINT Y
40 END
```

Cada vez que se desee obtener un valor de Y hay que teclear **RUN** y ENTER. El primer valor de Y que conviene calcular es la ordenada del vértice. Para ello tecleamos 4.27, después de que aparezca la interrogación en la pantalla; obtenemos el valor – 4.53. Así se puede obtener una tabla de valores, introduciendo valores de X a la izquierda y a la derecha de 4.27.

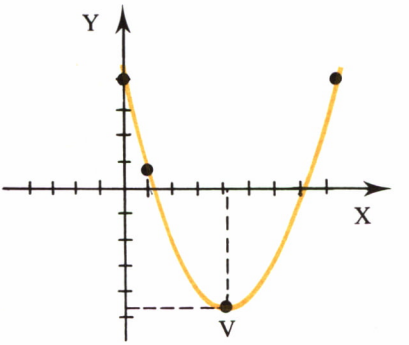
X	Y
0	4.58
1	0.81
.	.
.	.
.	.
4.27	– 4.53
.	.
.	.
.	.
8.5	4.41

(*) Este cálculo se puede hacer también con el microordenador, escribiendo

PRINT – (–4.27) / (2 * 0.5)

Después de pulsar ENTER aparece en la pantalla 4.27 (Ver capítulo siguiente, apartado **5, El microordenador como calculadora**.)

Después de obtener varios pares de valores, procedemos a hacer la representación gráfica en papel milimetrado o cuadriculado.



EJERCICIOS DE RECAPITULACIÓN

2.1. Escribe un programa que calcule el producto de dos números y lo imprima.

2.2. Explica qué hace el siguiente programa:

```
10  INPUT A
20  LET B = A + 1
30  PRINT B
40  END
```

2.3. Suponte que el programa del ejercicio anterior está cargado en la memoria del microordenador. ¿Qué tienes que hacer para ejecutarlo?

2.4. Según estás tecleando la instrucción.

```
20 LEP B = A + 1
```

te das cuenta, antes de pulsar la tecla ENTER, que has cometido un error al escribir **LEP** en lugar de **LET**. ¿Qué harías para corregir este error?

2.5. Al ejecutar el siguiente programa, aparece en la pantalla un mensaje de error. Averigua cuál es dicho error y el mensaje que emite el microordenador.

```
10  INPUT N
20  LET = C = N ↑ 3
30  PRINT C
40  END
```

2.6. Explica cómo harías para corregir el programa anterior para lograr que termine su ejecución.

2.7. Al teclear el programa siguiente tienes un lapsus y escribes una instrucción dos veces con número diferente. ¿Cómo harías para borrar la instrucción que no corresponde?

```
10  INPUT P
20  LET D = P/168.690
30  PRINT D
40  PRINT D
40  END
```

Nota: La cotización del dólar corresponde al día 10-9-84.

2.8. Después de escribir un programa en el papel deseas cargarlo en memoria, estando almacenado en ésta un programa que no te interesa conservarlo. ¿Qué harías antes de empezar a teclear el nuevo programa?

2.9. Escribe en lenguaje BASIC la siguiente expresión:

$$y = 0{,}27\left(x^3 - \frac{1}{2}\,x + \frac{1}{25}\right)^2 : 30 - 25$$

2.10. En los programas siguientes indica en qué orden se hacen las operaciones y qué resultado imprimirá el microordenador en la pantalla.

a)
```
10  LET Y = 5 - 5 * 8 ↑ 2
20  PRINT Y
30  END
```

b)
```
10  LET Y = (5 - 5) * 8 ↑ 2
20  PRINT Y
30  END
```

c)
```
10  LET Y = 5 - 2 * 4 ↑ 3
20  PRINT Y
30  END
```

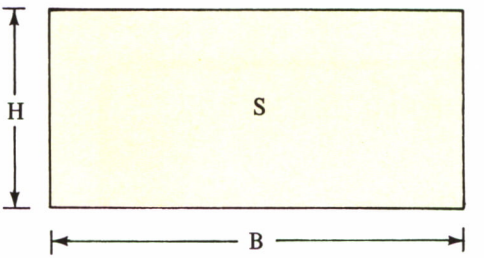
d)
```
10  LET Y = 5 - (2 * 4) ↑ 3
20  PRINT Y
30  END
```

3. Avanzando en la programación

1. La instrucción INPUT

Vamos a elaborar un programa que calcule el área de cualquier rectángulo y después lo escriba o imprima.

Llamamos B a la base del rectángulo y H a su altura. Como ya sabemos, el área S es igual al producto B por H.



- Un posible programa sería el siguiente:

```
10  INPUT B
20  INPUT H
30  LET S = B * H
40  PRINT S
50  END
```

(Introducirlo en la memoria pulsando ENTER después de cada instrucción.)

Al teclear **RUN** (y ENTER) comienza la ejecución. Ésta se interrumpe apareciendo una interrogación, pidiendo un dato (tecleamos 8, y ENTER). Nuevamente se detiene la ejecución, apareciendo otra interrogación, pidiendo un se-

gundo dato (tecleamos 5, y ENTER). A continuación aparece en la pantalla el resultado 40:

```
? 8
? 5
40
```

- El programa anterior se puede modificar a efectos de abreviar pasos y de conseguir mayor claridad en la introducción de los datos.

```
10  INPUT "INTRODUCE B Y H"; B, H
20  LET S = B * H
30  PRINT S
40  END
```

Al teclear **RUN** y ENTER, aparece en la pantalla lo siguiente:

```
INTRODUCE B Y H?
```

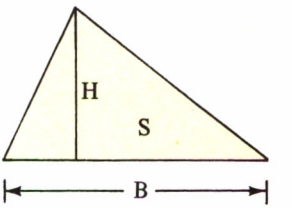
Esta frase nos indica explícitamente que introduzcamos los datos correspondientes a las variables B y H. Si tecleamos 8 (y ENTER) y, a continuación, 5 (y ENTER), aparecerá en la pantalla el resultado 40.

Vemos que una instrucción **INPUT** permite introducir un texto entrecomillado y varias variables separadas por ***comas***. El texto y las variables se separarán normalmente por un ***punto y coma.***

2. La instrucción REM

El siguiente programa calcula el área de un triángulo e imprime el resultado.
(Área del triángulo = $\frac{1}{2}$ × base × altura.)

```
10  INPUT "INTRODUCE B Y H"; B, H
20  LET S = 1/2 * B * H
30  PRINT S
40  END
```



Tecleando **RUN** y pulsando la tecla ENTER el programa se empieza a ejecutar. Si introducimos primero el valor 10 y, a continuación, el 4, aparece impreso en la pantalla el valor del área, igual a 20.

Ahora ejecutaremos el programa anterior al que se ha añadido la instrucción 5.

```
 5  REM AREA DE UN TRIANGULO
10  INPUT "INTRODUCE B Y H"; B, H
20  LET S = 1/2 * B * H
30  PRINT S
40  END
```

Introduciendo los mismos valores, 10 y 4, aparece el mismo resultado en la pantalla.

La instrucción **5** no introduce ninguna modificación en la ***ejecución*** del programa.

La instrucción REM ***únicamente sirve para poner título a un programa o a una parte del mismo.*** Sin embargo, aunque ***no es una instrucción ejecutable,*** hay que tener en cuenta que sí ocupa posiciones de memoria.

Ejercicios

3.1. Escribe un programa que calcule el cubo de un número e imprima el resultado. Utiliza la instrucción **REM** para dar título al programa.

3.2. Escribe un programa que calcule la longitud de la circunferencia (L = 2 ×3,1416 R) y el área del cuadrado ($S = l^2$), utilizando dos veces la instrucción **REM**, e imprime los resultados.

3. La instrucción PRINT

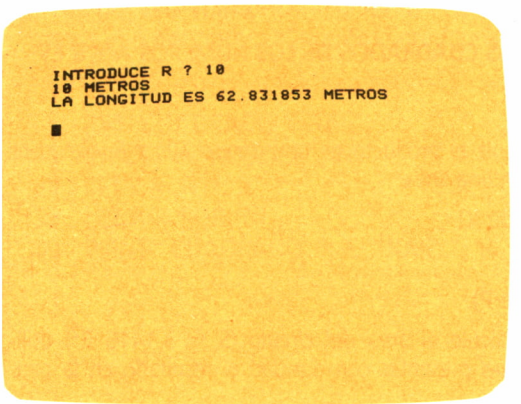
El siguiente programa calcula la longitud de la circunferencia e imprime el resultado.

```
10  INPUT "INTRODUCE R"; R
20  LET L = 2 * 3.1416 * R
30  PRINT L
40  END
```

Sin alterar esencialmente este programa, se le puede modificar para que los resultados que se impriman en la pantalla aparezcan con mayor claridad.

```
 5  REM LONGITUD DE LA CIRCUNFERENCIA
10  INPUT "INTRODUCE R"; R
15  PRINT R; "METROS"
20  LET L = 2 * 3.1416 * R
30  PRINT "LA LONGITUD ES"; L; "METROS"
40  END
```

Al teclear **RUN** (y ENTER) e introduciendo el valor 10 para R, se obtiene lo siguiente en la pantalla:



Analicemos las líneas siguientes a la instrucción **RUN**, que corresponden a la parte de los resultados impresos:

INTRODUCE R ?	← Corresponde a la línea **10**
10 METROS	← Corresponde a la línea **15**
LA LONGITUD ES 62.832 METROS	← Corresponde a la línea **30**

Observamos que la instrucción **PRINT** imprime en la pantalla:

— ***Un texto entrecomillado pero sin las comillas (instrucciones*** 15 ***y*** 30***).***

— ***El valor de una variable pero no su nombre (instrucciones*** 15 ***y*** 30***).***

Para fijar ideas veamos más en detalle la instrucción **30**

30 PRINT	"LA LONGITUD ES";	L;	"METROS"
	Imprime este texto sin comillas.	Imprime el valor de L.	Imprime este texto sin comillas.

El resultado de esta instrucción es, por tanto, el siguiente: LA LONGITUD ES 62.832 METROS

Ejercicios

3.3. Escribe lo que aparecerá impreso en la pantalla al ejecutarse los siguientes programas:

a)
```
10 LET X = 3.25
20 PRINT "X"
30 END
```

b)
```
10 LET X = 3.25
20 PRINT X
30 END
```

c)
```
10 LET X = 3.25
20 PRINT "X="; X
30 END
```

3.4. Escribe lo que imprimiría en la pantalla la instrucción 40 una vez ejecutada:

```
10  LET A = 3
20  LET B = 4
30  LET C = A ↑ 2 + B ↑ 2
40  PRINT "LA SUMA DE LOS CUADRADOS DE LOS NUMEROS A Y B ES
C ="; C
50  END
```

3.5. Escribe un programa que calcule el producto de un número X por 5, almacene el resultado en P e imprima en una línea:

EL NÚMERO ES X = (valor de X)

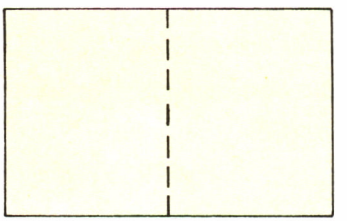
y en otra línea:

EL PRODUCTO P = (valor de P).

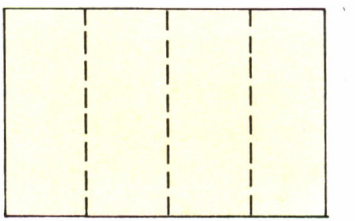
3.6. Elabora un programa que calcule el área del círculo ($S = 3{,}1416\ R^2$) utilizando instrucciones **PRINT** para presentar con claridad el resultado en la pantalla.

4. El punto y coma y la coma en la instrucción PRINT

- El número de caracteres que se puede escribir en una línea de la pantalla de un microordenador es limitado. La pantalla, además, puede estar dividida en dos o más zonas.



Pantalla dividida en dos zonas.



Pantalla dividida en cuatro zonas.

- En los ejemplos siguientes supondremos que la pantalla tiene capacidad para imprimir 32 caracteres por línea (de 0 a 31) y está dividida en dos zonas de 16 caracteres cada una por línea (de 0 a 15 y de 16 a 31).

- Si en una instrucción **PRINT** los textos y las variables están separados por ***puntos y comas,*** el microordenador interpreta que no tiene que dejar ningún espacio entre los textos y el valor de las variables. Si, en cambio, están separados por una ***coma,*** los imprime en zonas diferentes. Por ejemplo, supongamos que ejecutamos el siguiente programa para N = 25.

```
 5  REM CUADRADO Y RAIZ
10  INPUT "INTRODUCE"; N
20  LET C = N ↑ 2
30  LET R = SQR (N)
40  PRINT "CUADRADO:"; C, "RAIZ:"; R
```

(*)

Después de teclear **RUN** (y ENTER) y de introducir el valor 25, los textos y los números aparecerán distribuidos así en la pantalla:

CUADRADO: 625	RAIZ: 5

- Si al final de una instrucción **PRINT** no se pone ***punto y coma*** ni ***coma,*** el texto o los valores de las variables del siguiente **PRINT** se imprimen en la línea siguiente. Por ejemplo, al ejecutar el siguiente programa, para N = 25.

```
 5  REM CUADRADO Y RAIZ
10  INPUT "INTRODUCE N"; N
20  LET C = N ↑ 2
30  LET R = SQR (N)
40  PRINT "EL CUADRADO DE"; N; "ES"; C
50  PRINT "LA RAIZ CUADRADA DE"; N; "ES"; R
60  END
```

(*) La función SQR (N) calcula la raíz cuadrada del número N.

se obtiene este resultado en la pantalla

```
EL CUADRADO DE 25 ES 625

LA RAIZ CUADRADA DE 25 ES 5
```

- Una instrucción **PRINT** que no lleve incorporada ninguna expresión produce el efecto de dejar una línea en blanco. Por ejemplo, si en el programa anterior añadimos la instrucción **45 PRINT**

```
 5  REM CUADRADO Y RAIZ
10  INPUT "INTRODUCE N"; N
20  LET C = N ↑ 2
30  LET R = SQR (N)
40  PRINT "EL CUADRADO DE"; N; "ES"; C
45  PRINT
50  PRINT "LA RAIZ CUADRADA DE"; N; "ES"; R
60  END
```

y la ejecutamos para N = 25, se obtiene el siguiente resultado en la pantalla:

```
EL CUADRADO DE 25 ES 625

LA RAIZ CUADRADA DE 25 ES 5
```

Línea en blanco debida a la instrucción **45 PRINT**

- En algunos microordenadores se puede sustituir la palabra **PRINT** por un signo de interrogación, lo que contribuye a abreviar la escritura. Así,

  30 PRINT A, B

  se puede escribir de la siguiente forma:

  30 ? A, B

5. El microordenador como calculadora

La instrucción **PRINT** permite también realizar cálculos directamente, sin necesidad de elaborar un programa para ello. Así, escribiendo

PRINT 13 ↑ 2

y accionando la tecla **RETURN**, se obtiene inmediatamente en pantalla el cuadrado de 13, es decir,

169

Vemos que el microordenador, utilizando la instrucción **PRINT**, actúa como calculadora, y para estos efectos no es necesario escribir delante de **PRINT** ningún número.

6. Cómo escribe los números el microordenador

- Tecleamos las siguientes instrucciones **PRINT**:

```
10 PRINT 38, –205
20 PRINT –5.28, 0.0024
30 PRINT "DISTANCIA TIERRA-SOL ="; 149600000000; "M"
40 PRINT "LA VELOCIDAD DEL CARACOL ES:"
50 PRINT 0.000012; "KM/S"
```

Al ejecutar estas instrucciones, tecleando **RUN** (y ENTER), se obtienen los siguientes resultados en la pantalla:

38 –205	Corresponde a la línea **10**
–5.28 0.0024	Corresponde a la línea **20**
DISTANCIA TIERRA-SOL = 1.496 E + 11 M	Corresponde a la línea **30**
LA VELOCIDAD DEL CARACOL ES:	Corresponde a la línea **40**
1.2 E-5 KM/S	Corresponde a la línea **50**

— Los números que corresponden a la línea **10** no tienen punto decimal. Se dice que están escritos en notación ***entera.***

— Los números que corresponden a la línea **20** tienen punto decimal. Se dice que están escritos en notación ***real.***

— Los números que corresponden a las líneas **30** y **40** el microordenador los ha transformado a notación ***exponencial.***

149600000000 = 1.496 E + 11

- La notación exponencial es similar a la notación ***científica.***

¿Cómo escribiremos el número 4800000000 en notación científica y exponencial?

$4800000000 = 4.8 \times 10^{+9} = 4.8\ E + 9$

¿Y el número 0,000000000014?

$0{,}000000000014 = 1.4 \times 10^{-12} = 1.4\ E - 12$

Vemos que la notación exponencial se obtiene de la científica, sustituyendo ***x 10 elevado a,*** por E.

La notación exponencial se compone de tres partes: ***un número escrito en notación entera o real,*** la ***E,*** que significa ***x 10 elevado a,*** y ***el exponente de 10 con su signo.*** Observemos estas tres partes en el número

$$-5.876\ E - 11$$

que equivale a -5.876×10^{-11}, o sea, a

– 0.00000000005876

- ¿Por qué el ordenador emplea esta notación? Porque normalmente tiene capacidad para escribir números hasta con nueve cifras, un punto decimal y un signo (+ ó −). Si un número contiene más de nueve dígitos, automáticamente lo transforma en notación exponencial.

7. Resumen de las instrucciones estudiadas en este capítulo

- INPUT

  — ***Significado:*** introduce.

  — ***Formatos:***

  n **INPUT** {texto entrecomillado}; {nombre de variables}
  n es el número de instrucción.
  (Las llaves no se escriben. Ejemplo: se escribe **10 INPUT** "RAIZ"; R, no **10 INPUT** {"RAIZ"}; **INPUT** {R}

  — ***Función:*** permite introducir datos desde el teclado, los cuales se asignan a las variables.

- REM

  — ***Significado:*** remarca.

  — ***Formato:*** n **REM** {lista de caracteres}

  — ***Función:*** permite añadir aclaraciones o comentarios en un programa sin afectar su ejecución.

- PRINT

  — ***Significado:*** imprime.

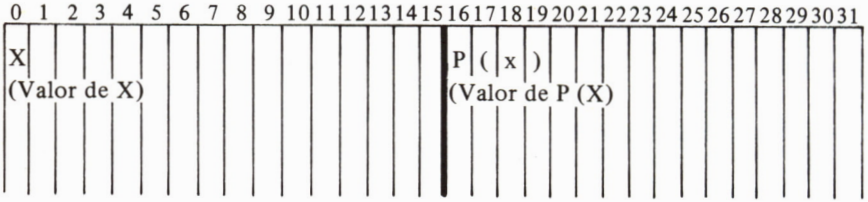
  — ***Formato:***

  $$\text{n PRINT} \left\{ \begin{array}{l} \text{nombre de variables} \\ \text{expresiones aritméticas} \\ \text{texto entre comillas} \end{array} \right\}$$

  — ***Función:***
  Escribe el valor de las variables.
  Escribe el resultado de las expresiones aritméticas.
  Escribe el texto que está entre comillas, pero no éstas.

EJERCICIOS RESUELTOS

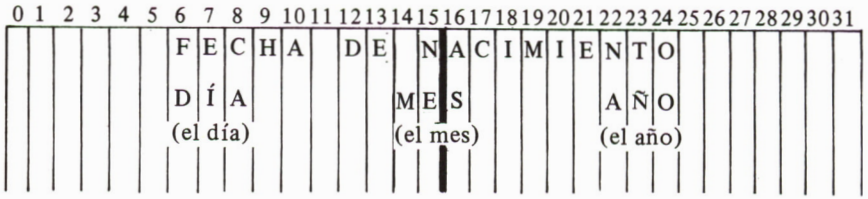
1. Escribir un programa que calcule el valor numérico del polinomio $P(X) = 0{,}5\,(6X - \frac{1}{4})^2 + X^3 : 6{,}25 - (X^2 + 1)^5$, para un determinado valor de X, e imprima en la pantalla el valor de X y P(X) en la siguiente forma:



Solución:

```
 5 REM VALOR NUMERICO DE UN POLINOMIO
10 INPUT "INTRODUCE X"; X
20 LET P = 0.5 * (6 * X - 1 / 4) ↑ 2 + X ↑ 3 / 6.25 - (X ↑ 2 + 1) ↑ 5
30 PRINT "X", "P(X)"
40 PRINT X, P
50 END
```

2. Escribir un programa que imprima el día, el mes y el año de nacimiento de una determinada persona y presente estos datos en la pantalla de la siguiente forma:



Solución:

```
 5 REM FECHA DE NACIMIENTO
10 INPUT "DIA, MES, AÑO"; D, M, A
20 PRINT "      FECHA DE NACIMIENTO"
                6 espacios
30 PRINT "      DIA      MES      AÑO"
           6 espacios  5 espacios  5 espacios
40 PRINT "      "; D; "      "; M; "      "; A
           6 espacios  4 espacios  4 espacios
50 END
```

Para conseguir que el microordenador deje espacios en blanco se abren comillas, se acciona el espaciador del teclado tantas veces como blancos se quieren incluir y, finalmente, se cierran comillas.

EJERCICIOS DE RECAPITULACIÓN

3.1. ¿Cómo aparecerán escritos los números 5 y 37 al ejecutarse los siguientes programas?

a)
```
10 PRINT 5, 37
20 END
```

b)
```
10 PRINT 5; 37
20 END
```

c)
```
10 PRINT 5
20 PRINT 37
30 END
```

d)
```
10 PRINT 5
20 PRINT
30 PRINT
40 PRINT 37
50 END
```

3.2. Escribe en un papel lo que aparecerá en la pantalla al ejecutarse el siguiente programa. (Suponte que se introduce 14.35 para H y 25 para R.)

```
 5 REM LLEGADA DE TRENES
10 INPUT H
20 INPUT R
30 PRINT "             AVISO"
              13 espacios
40 PRINT
50 PRINT "HORA DE LLEGADA", "RETRASO"
60 PRINT H, R; "MIN"
70 END
```

3.3. Escribe un programa que haga la conversión de dólares a pesetas y escriba en la pantalla lo siguiente:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
C O N V E R S I Ó N   D E   D Ó L A R E S   A   P E S E T A S
D Ó L A R E S                 P E S E T A S
(Número de dólares)           (Número de pesetas)
```

3.4. Averigua la cotización del día y escribe un programa que convierta pesetas a francos belgas. Incluye en el programa instrucciones **PRINT** que presenten los resultados en la pantalla en forma similar a la presentada en el ejercicio anterior.

3.5. Las fórmulas del área y del volumen de la esfera son $S = 4 \pi R^2$ y $V = \frac{4}{3} \pi R^3$, respectivamente. Elabora un programa que introduzca el valor del radio R, calcule S y V, y presente los resultados así:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
R A D I O , R = (Valor del radio)
(Una línea en blanco)
S = (Valor del área)                V = (Valor del volumen)
```

3.6. Dibuja en un papel la figura que hace aparecer en la pantalla el siguiente programa:

```
 5 REM DIBUJO DE FIGURAS
10 PRINT "          ----------"
             10 espacios
20 PRINT "          |        |"
             10 espacios
30 PRINT "          |        |"
             10 espacios
40 PRINT "          |        |"
             10 espacios
50 PRINT "          |        |"
             10 espacios
60 PRINT "          ----------"
             10 espacios
70 END
```

3.7. Haz un programa que dibuje en la pantalla la siguiente figura, incluidos los números que expresan la medida de los lados.

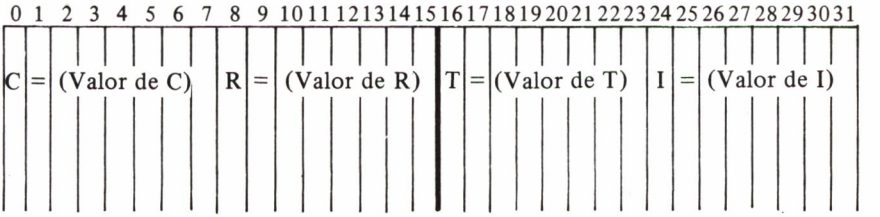
3.8. Realiza un programa que dibuje en la pantalla el encabezamiento de la siguiente tabla estadística; en la que aparezcan la variable X, la frecuencia F y la frecuencia acumulada, FA.

X	F	FA

3.9. La fórmula que calcula el interés simple I en función del capital C, el rédito R y el tiempo T, expresado en años, es:

$$I = \frac{CRT}{100}$$

Escribe un programa que calcule el interés y presente los datos y resultados en la pantalla así:



3.10. La fórmula que transforma grados centígrados a grados Fahrenheit es

$$F = \frac{9}{5} C + 32$$

Realiza un programa que haga la conversión de grados centígrados a Fahrenheit y presente en la pantalla el dato y resultado así:

GRADOS CENT. GRADOS FAHR.
(Una línea en blanco)
(Valor de gr. cent.) (Valor de gr. Fahr.)

3.11. Las fórmulas de la altura descendida y de la velocidad alcanzada por un móvil que cae libremente en el vacío son:

$$h = \frac{1}{2} g t^2 \quad y \quad v = g t$$

donde h es la altura descendida, g la aceleración de la gravedad ($9{,}8 \frac{m}{s^2}$) y t el tiempo. Escribe un programa que calcule la altura descendida y la velocidad alcanzada para un tiempo t. Escribe instrucciones **PRINT** para presentar con claridad los datos y resultados en la pantalla.

3.12.. Recuerda la ***ley de Ohm*** que relaciona la resistencia R de un conductor, la caída de potencial V que se produce entre los extremos de la resistencia al pasar la intensidad I por la misma:

$$R = \frac{V}{I} \qquad I = \frac{V}{R}$$

Haz un programa que calcule la intensidad I en función de R y V y presente los datos y resultados de la siguiente forma:

RESISTENCIA, R = (valor de R).
CAÍDA DE TENS, V = (valor de V).
INTENSIDAD, I = (valor de I).

4. Saltos incondicionales e instrucciones condicionales

1. Una instrucción con la que se puede escribir infinitas veces una frase: GOTO

En todos los programas que hemos visto hasta ahora, las instrucciones se ejecutan secuencialmente, es decir, siguiendo el orden creciente de sus números. Sin embargo, a veces es necesario que la ejecución no se ajuste a esa secuencia, sino que salte de una instrucción a otra anterior o posterior. Esto es lo que ocurre en el programa siguiente:

```
 5 REM SALTO INCONDICIONAL
10 PRINT "SOY EL MICRO: MI MISION ES OBEDECER"
20 GOTO 10
30 END
```

Si ejecutamos este programa (teclear **RUN** y ENTER), obtendremos en la pantalla lo siguiente:

Vemos que la frase "SOY EL MICRO: MI MISIÓN ES OBEDECER" se repite en la pantalla una y otra vez, no terminándose nunca la ejecución de este programa.

Estudiémoslo con más detalle:

— La instrucción **10** escribe la frase.

— La instrucción **20 GOTO 10** hace saltar al microordenador a la instrucción 10, el cual escribe otra vez la frase, y así sucesivamente, obligando al microordenador a escribir dicha frase un número infinito de veces.

— La instrucción **30 END** no se ejecuta nunca.

2. Un programa que sirve para contar

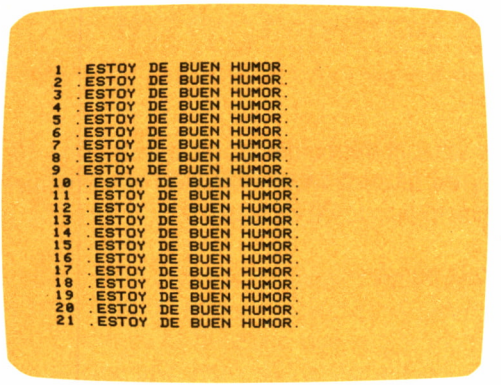
En el programa que transcribimos a continuación se ha introducido la variable C, que cuenta el número de veces que el microordenador escribe la frase: "ESTOY DE BUEN HUMOR":

```
 5  REM CONTADOR
10  LET C = 0
20  LET C = C + 1
30  PRINT C; "ESTOY DE BUEN HUMOR"
40  GOTO 20
50  END
```

(*)

(*) Para «poner a cero» una variable, algunos microordenadores no exigen que se haga explícitamente mediante una instrucción, como la **10**; la «ponen a cero» directamente, cuando es leída en una instrucción, como la **20**.

Al ejecutar este programa (**RUN** ENTER) obtenemos el siguiente resultado en la pantalla:



¿Cómo funciona este programa?

— La instrucción **10** define la variable C y almacena en ella el valor cero.

— La instrucción **20 LET C = C + 1** suma 1 al valor que tiene en este instante la variable C, y almacena en C el resultado de la suma.

— La instrucción **30** imprime el valor de C y la frase "ESTOY DE BUEN HUMOR".

→ La instrucción **40 GOTO 20** (ir a **20**) transfiere el control a **20**, y a partir de aquí empieza a repetirse otra vez el proceso.

— La instrucción **50 END** nunca se ejecuta, y por eso la ejecución de este programa no termina nunca.

La instrucción

20 LET C = C + 1

actúa como ***contador*** porque cada vez que se ejecuta incrementa en una unidad el valor de la variable C. Por ejemplo, si en este instante el valor almacenado en C es 0, va almacenando valores consecutivos en las sucesivas ejecuciones:

C
1.ª ejecución: C = 0 + 1 = 1
2.ª ejecución: C = 1 + 1 = 2
3.ª ejecución: C = 2 + 1 = 3
4.ª ejecución: C = 3 + 1 = 4

Ejercicios

4.1. Escribe en un papel lo que escribiría el microordenador en la pantalla al ejecutarse este programa:

```
10  LET A = 0
20  PRINT "YA ESTA BIEN DE REPETICIONES!"
30  LET A = A + 1
40  PRINT A
50  PRINT
60  GOTO 20
70  END
```

4.2. Indica qué instrucciones nunca se ejecutan en el siguiente programa:

```
10  LET A = 5.1
20  LET B = 8.9
30  LET S = A + B
40  GOTO 70
50  LET P = A * B
60  PRINT P
70  PRINT S
80  END
```

4.3. Elabora un programa que escriba un número ilimitado de veces la frase ESTOY PROGRAMANDO CON LA INSTRUCCIÓN GOTO e imprime el número de veces que lo escribe.

4.4. Haz un programa que escriba todos los números naturales.

4.5. Escribe en un papel lo que aparecería en la pantalla si se ejecutara el siguiente programa:

```
10  LET C = 0
20  PRINT C
30  PRINT
40  LET C = C + 2
50  GOTO 20
60  END
```

4.6. Elabora un programa que imprima los sucesivos números impares en filas.

3. Una instrucción que toma decisiones: IF...THEN

Nos dan un número N mayor que 3 y nos piden que calculemos y escribamos todos los múltiplos de 3 menores que N.

Este trabajo lo puede realizar muy rápidamente el microordenador aunque N sea muy grande.

Para elaborar el programa tendremos que seguir este razonamiento lógico:

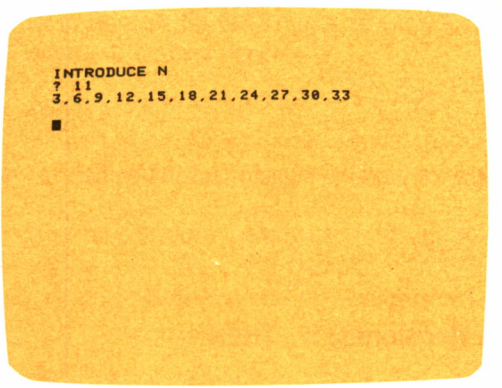
10 Introducir N
20 Hacer M = 3
30 Escribir M
40 Buscar el siguiente múltiplo (M = M + 3)
50 Si M es menor que N entonces ir a 30
60 Fin

Aquí la instrucción importante es la 50, que obliga a repetir las instrucciones 30 y 40 siempre que M < N, y, en caso contrario, a ejecutar la siguiente, es decir, la 60.

El programa en BASIC es:

```
 1  REM MULTIPLOS DE TRES
 5  PRINT "INTRODUCE N"
10  INPUT N
20  LET M = 3
30  PRINT M; ",";
40  LET M = M + 3
50  IF M < N THEN GOTO 30
60  END
```

Si ejecutamos este programa (RUN ENTER) e introducimos, por ejemplo, el valor 11 para N, obtendremos el siguiente resultado en la pantalla:



¿Cómo funciona este programa?

— La instrucción **5** escribe en la pantalla: INTRODUCE N.

— La instrucción **10** interrumpe la ejecución en espera de que introduzcamos por el teclado el valor de N.

— La instrucción **20** asigna a la variable M el valor 3.

— La instrucción **30** imprime el valor de la variable M seguido de una coma para ir separando los distintos números.

— La instrucción

50 IF M < N THEN GOTO 30 (Si M < N, entonces ir a **30**) (*)
es la que compara M con N y toma la decisión; hace lo siguiente

- Si se cumple la condición M < N entonces la ejecución continúa en la instrucción **30** ("salta" a **30**).
- Si ***no*** se cumple la condición M < N entonces la ejecución continúa en la instrucción siguiente, es decir, en la **60.**

— Cuando ***M no es menor que N*** se ejecuta la instrucción **60,** dándose por terminada la ejecución del programa.

Vamos a analizar con más detalle el funcionamiento de la instrucción **IF...THEN,** suponiendo que N = 11.

1.ª ejecución:

— Se introduce 11.
— Se hace M = 3.
— Se imprime 3.
— Se hace M = M + 3 = 3 + 3 = 6.
— Se hace la comparación: "Si 6 < 11 entonces ir a **30**".

2.ª ejecución

— Se imprime 6.
— Se hace M = M + 3 = 6 + 3 = 9.
— Se hace la comparación: "Si 9 < 11 entonces ir a **30**".

3.ª ejecución:

— Se imprime 9.
— Se hace M = M + 3 = 9 + 3 = 12.
— Se hace la comparación: "Si 12 < 11 entonces ir a **30**".

Como 12 no es menor que 11, entonces no salta a **30,** sino que pasa a ejecutar la instrucción siguiente, o sea, la **60,** dando por finalizada la ejecución del programa.

(*) En algunos microordenadores esta instrucción se puede escribir directamente sin el **GOTO**. Por ejemplo, la instrucción **50** se escribirá así: **50 IF M < N THEN 30.**

Resumiendo:

- En este apartado hemos visto que la instrucción **IF...THEN** ocasiona saltos en un programa siempre que se cumpla una determinada condición. Produce ***saltos condicionales.***
- En los apartados anteriores hemos visto que la instrucción **GOTO** ocasiona saltos sin estar sujeta a que se cumpla una determinada condición. Produce ***saltos incondicionales.***

Ejercicios

4.7. Escribe en un papel lo que aparecerá en la pantalla al ejecutarse el siguiente programa, para N = 8.

```
10 INPUT N
20 LET A = 0
30 PRINT A; ",";
40 LET A = A + 1
50 IF A < N THEN GOTO 30
60 END
```

4.8. Escribe un programa que calcule e imprima los múltiplos de 5 hasta un cierto número.

4.9. Escribe un programa que calcule e imprima los números pares naturales hasta un cierto número.

4.10. Indica qué hace el siguiente programa:

```
 10 INPUT N
 20 LET A = 0
 30 LET A = A + 7
 40 PRINT A; ",";
 50 IF A < N THEN GOTO 30
 60 PRINT
 70 PRINT
 80 LET B = 0
 90 LET B = B + 11
100 PRINT B; ",";
110 IF B < N THEN GOTO 90
120 END
```

4.11. Haz un programa que calcule e imprima primero los múltiplos de 19 menores que 216 y después los de 23.

4.12. Elabora un programa que introduzca mediante **INPUT** una variable, de tal manera que, si ésta tiene un valor mayor que cero, el programa vuelve a pedir un nuevo valor y, en caso contrario, termine la ejecución.

4. Instrucción IF...THEN... generalizada

- Hasta ahora, a continuación de la palabra **THEN** en la instrucción **IF...THEN ...**, hemos puesto la instrucción **GOTO**. Sin embargo, también admite otras instrucciones como muestra el ejemplo que sigue.

```
35 IF X > 0 THEN PRINT X
```

Esta instrucción ordena que ***si se cumple la condición X > 0*** entonces ***se imprima el valor de X.***

Incluso, admite varias instrucciones, las cuales deberán estar separadas por ***dos puntos***. Ejemplo:

```
IF Z < = X THEN LET Z = Z + 1 : PRINT Z : GOTO 20
```

- En el siguiente programa, que va escribiendo el menor de los números introducidos hasta el momento, se ve la utilidad de la instrucción **IF...THEN...** generalizada.

```
10  INPUT M
20  PRINT "EL MENOR ES:"; M
30  INPUT X
40  IF X < M THEN LET M = X: GOTO 20
50  GOTO 20
60  END
```

Al teclear **RUN** (y ENTER) comienza la ejecución del programa.

— La instrucción **10** pide un valor para M; este valor, la instrucción **20** lo imprimirá en la pantalla por ser el único introducido hasta el momento.

— La instrucción **30** pide otro valor para X.

— La instrucción **40** compara el valor de X con el menor M, y si X < M entonces se ejecutan las instrucciones **LET M = X** (el menor es ahora X) y **GOTO 20,** que envía el control de la ejecución a la instrucción **20**. Si no se cumple la condición X < M, se ejecuta la instrucción siguiente, **50 GOTO 20,** que envía el control a **20**.

Si introducimos sucesivamente los valores 24, 13, 50, −10 y 31, en la pantalla aparecerá lo siguiente:

```
EL MENOR ES 24
EL MENOR ES 13
EL MENOR ES 13
EL MENOR ES −10
EL MENOR ES −10
```

- Existen microordenadores que admiten una variante en la instrucción **IF... THEN...**; es la siguiente:

  IF ***condición*** THEN ***instrucciones 1*** ELSE ***instrucciones 2***

  Su significado es:

  SI se cumple la ***condición,*** ENTONCES ejecutar las ***instrucciones 1*** y

  SI NO se cumple, ejecutar las ***instrucciones 2.***

  Ejemplo:

  ```
  230  IF X < Y THEN LET X = X + 1 ELSE LET Y = Y + 1
  ```

5. Instrucción compuesta

Como se ha visto en el apartado anterior, en una misma línea se pueden poner varias instrucciones, siempre que vayan separadas por ***dos puntos.***

Ejemplo:

```
10 LET X = 0 : LET Y = 3 : LET A = 7
```

Un conjunto de instrucciones escritas en una misma línea (separadas por dos puntos) se llama ***instrucción compuesta.***

Si bien en algunas ocasiones conviene escribir instrucciones compuestas para abreviar la escritura de los programas, en algunos casos pueden llegar a perder claridad.

6. Un programa que cuenta el número de alumnos que miden entre 1,60 y 1,70

El programa siguiente cuenta el número de alumnos cuya estatura está comprendida entre 1,60 y 1,70.

El programa que realice este trabajo necesitará utilizar una instrucción condicional **IF...THEN,** de tal manera que si se cumple la condición aumente en una unidad un contador, cuyo valor se imprimirá posteriormente.

```
 5  REM RECUENTO DE ALUMNOS ENTRE 1.60 Y 1.70
10  LET S = -1
20  LET S = S + 1
30  PRINT "EL NUMERO TOTAL HASTA ESTE MOMENTO ES: S ="; S
40  PRINT "INTRODUCE LA TALLA"
50  INPUT X
60  IF X > 1.60 AND X < 1.70 THEN GOTO 20
70  GOTO 40
80  END
```

¿Cómo funciona este programa?

Después de teclear **RUN** y pulsar la tecla ENTER el programa inicia la ejecución:

— La instrucción **10** define la variable S y la asigna el valor –1.

— La instrucción **20 LET S = S + 1** suma 1 al valor que tiene en este instante y almacena en S el resultado de la suma: almacena el valor –1 + 1 = 0, o sea, pone a cero el contador.

— La instrucción **30** imprime: EL NÚMERO TOTAL HASTA ESTE MOMENTO ES S = 0.

— La instrucción **40** imprime: INTRODUCE LA TALLA.

— La instrucción **50** interrumpe la ejecución en espera de que se introduzca por teclado la talla de un alumno.

— La instrucción

60 IF X > 1.60 AND X < 1.70 THEN GOTO 20

(Si X > 1.60 y X < 1.70 entonces ir a 20)

es la que compara X con 1.60 y 1.70 y toma la decisión: *si se cumplen las dos condiciones simultáneamente la ejecución continúa en la instrucción* **20** *y, en caso contrario, continúa en la siguiente instrucción, es decir, en la* **70.** Por ejemplo, supongamos que se ha introducido el dato 1.67 (X = 1.67); como X > 1.60 y X < 1.70, la ejecución sigue en la **20,** incrementándose en 1 el contador. Pero si en cambio, el dato es 1.78 (X = 1.78); como X $\not<$ 1.70, la ejecución continúa en **70.**

— La instrucción **70 GOTO 40** hace saltar la ejecución a la instrucción **40** no incrementándose en este caso el contador.

— En este programa no llega a ejecutarse la instrucción **80 END.**

7. Relaciones aritméticas y lógicas

• Observamos la instrucción **60** del programa anterior:

60 IF X > 1.60 **AND** X < 1.70 **THEN GOTO 20**

Relación aritmética — Relación aritmética

Relación lógica

Esta instrucción contiene dos ***relaciones aritméticas*** (X > 1.60, X < 1.70) en las que encontramos los operadores aritméticos > (es mayor que) y < (es menor que), respectivamente. Estas dos relaciones aritméticas están a su vez vinculadas mediante el operador lógico **AND** (y), constituyendo en conjunto una ***relación lógica.***

Operadores aritméticos

X > 1.60 **AND** X < 1.70

Operador lógico

Vemos, pues, que los operadores aritméticos vinculan ***números o expresiones*** aritméticas y el operador lógico, ***relaciones aritméticas.***

Además de los operadores vistos en este ejemplo, existen otros que se utilizan frecuentemente en la instrucción **IF...THEN.** Los transcribimos a continuación:

RELACIONES ARTIMÉTICAS	
A < B	A es menor que B
A < = B	A es menor o igual que B
A > B	A es mayor que B
A > = B	A es mayor o igual que B
A = B	A es igual a B
A < > B	A no es igual a B

A y B son números o expresiones aritméticas.

RELACIONES LÓGICAS	
P **AND** Q	P y Q (se verifica la relación P y Q)
P **OR** Q	P o Q (se verifica la relación P o Q)
NOT P	no P (no se verifica la relación P)

P y Q son relaciones aritméticas.

- A su vez, las relaciones lógicas se pueden combinar entre sí, por ejemplo:

  (A < B **AND** C > A) **OR** C = A
  X = 3 **AND** (Y = 2 **OR** Z = X)

Ejercicios

4.13. Explica lo que hace este programa:

```
10 INPUT A
20 IF A < 10 AND A > 5 THEN GOTO 50
30 PRINT "NO SE CUMPLE"
40 GOTO 60
50 PRINT "SI SE CUMPLE"
60 END
```

4.14. Escribe un programa que introduzca un número y que imprima el signo + si este número es positivo y el signo – si es negativo.

4.15. Explica cada una de las instrucciones del programa siguiente:

```
10 LET C = –1
20 LET C = C + 1
30 PRINT C
40 PRINT "INTRODUCE EL PESO"
50 INPUT X
60 IF X > 50 AND X < 60 THEN GOTO 20
70 GOTO 40
80 END
```

4.16. Escribe un programa que calcule el número de alumnos cuyo peso sea mayor que 60 kilos y su estatura inferior a 1.65 metros.

4.17. Elaborar un programa que dé los múltiplos comunes de 3 y 7 hasta un cierto número.

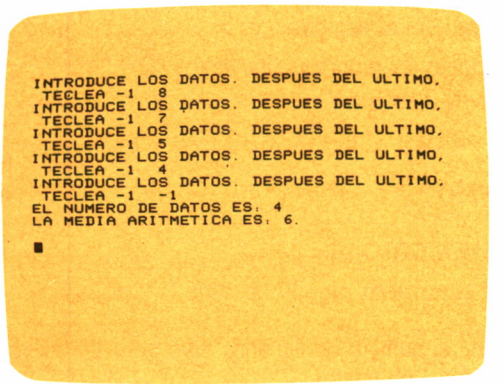
8. Un programa para calcular la media aritmética

El siguiente programa calcula el valor medio o la media aritmética de una serie de datos que se introduce por el teclado del microordenador.

En este programa se utiliza una instrucción **IF ... THEN** como indicador, para concluir el proceso de introducción de datos.

```
 5 REM CALCULO DE LA MEDIA ARITMETICA
10 LET T = 0: LET N = 0
20 INPUT "INTRODUCE LOS DATOS. DESPUES DEL ULTIMO, TECLEA -1 ";
X
30 IF X = -1 THEN GOTO 70
40 LET T = T + X
50 LET N = N + 1
60 GOTO 20
70 PRINT "EL NUMERO DE DATOS ES: "; N
80 PRINT "LA MEDIA ARITMETICA ES: "; T/N
90 END
```

Si al ejecutar el programa (**RUN** ENTER) introducimos los datos 8, 7, 5 y 4 (y, a continuación, – 1) en la pantalla aparecerá lo que muestra la fotografía:



Analicemos cómo funciona este programa

— La instrucción **10** define las variables T (suma total) y N (número de datos), a los que asigna el valor cero.

— La instrucción **20** imprime la frase que está entre comillas y detiene la ejecución del programa en espera de que se introduzca un dato.

— La instrucción

30 IF X = −1 THEN GOTO 70

actúa de indicador: verifica si X = −1 y, en caso afirmativo, hace saltar la ejecución a la instrucción **70**. Si X ≠ −1, se ejecuta la instrucción siguiente, la **40**.

— La instrucción **40** va sumando los datos que se introducen.

— La instrucción **50** es un ***contador*** del número de datos que se van introduciendo.

— La instrucción **60** produce un ***salto incondicional*** a la instrucción **20**, para pedir un nuevo dato.

— La instrucción **70** escribe en la pantalla: EL NÚMERO DE DATOS ES: 4 (en el ejemplo elegido).

— La instrucción **80** escribe en la pantalla: LA MEDIA ARITMÉTICA ES: 6 (en el ejemplo elegido).

— Finalmente, la instrucción **90** da por terminada la ejecución del programa.

9. Una instrucción para detener la ejecución de un programa: instrucción STOP

Vamos a analizar el siguiente programa:

```
 5 REM NOTAS
10 INPUT "INTRODUCE UNA NOTA"; N
20 IF N > = 5 THEN GOTO 50
30 PRINT "SUSPENSO"
40 GOTO 10
50 STOP
60 PRINT "APROBADO"
70 END
```

- Si ejecutamos este programa (**RUN** ENTER) e introducimos para N el valor 3, la instrucción **20** compara 3 con 5, y como 3 ***no*** es mayor o igual que 5, sigue la ejecución en **30**, imprimiéndose en la pantalla la palabra SUSPENSO. Sigue el proceso con la instrucción **40** que hace saltar la ejecución a la **10**, la cual pide otra nota.
- Si ejecutamos este programa introduciendo para N la nota 7, por cumplirse la condición 7 > = 5, la instrucción **20** hace saltar la ejecución a la instrucción **50 STOP** que interrumpe la ejecución justo en este punto, no imprimiéndose la palabra APROBADO.
- Una vez interrumpida la ejecución por la acción de una instrucción **STOP**, se puede continuar dicha ejecución en el momento que se desee tecleando la instrucción **CONT** (continuar) y pulsando la tecla ENTER. En este ejemplo, como la ejecución está detenida en la línea **50**, si se teclea

  CONT (y ENTER)

  la ejecución ***continuará*** con la instrucción **60**, apareciendo en la pantalla la palabra APROBADO. La instrucción **70** dará por finalizado el proceso.

10. Resumen de las instrucciones estudiadas en este capítulo

- GOTO

  — ***Significado:*** ir a
  — ***Formato:*** n GOTO n'
  — ***Función:*** hace saltar la ejecución del programa a la instrucción n'; ésta puede ser anterior o posterior a n.

- IF ... THEN ...

  — ***Significado:*** Si ... entonces ...
  — ***Formato:*** n IF ***condición*** THEN ***instrucciones.***
  — ***Función:*** Si se cumple la ***condición*** entonces se ejecutan las instrucciones que siguen a **THEN**; y si no se cumple, se ejecuta la instrucción escrita en la línea siguiente.

- STOP

  — ***Significado:*** parada.
  — ***Formato:*** n STOP
  — ***Función:*** detiene la ejecución del programa justo en la línea en que se encuentra **STOP**.

- CONT

  — ***Significado:*** continúa.
  — ***Formato:*** CONT.
  — ***Función:*** continúa la ejecución de un programa a partir de la instrucción n **STOP**, que previamente la interrumpió.

EJERCICIOS RESUELTOS

1. Elaborar un programa que permita calcular el promedio de las notas de Lengua de un curso de N alumnos, las cuales se deberán introducir a través del teclado del microordenador.

   Solución:

```
 5  REM PROMEDIO NOTAS DE LENGUA
 8  LET C = 0 : LET T = 0
10  INPUT "INTRODUCE EL NUMERO DE ALUMNOS"; N
20  INPUT "INTRODUCE LA NOTA DE CADA ALUMNO"; X
30  LET C = C + 1
40  LET T = T + X
50  IF C = N THEN GOTO 70
60  GOTO 20
70  PRINT "PROMEDIO ="; T/N
80  END
```

2. Hacer un programa que averigüe si el número que se introduce está entre 1 y 5, de tal manera que la ejecución del mismo se detenga hasta que se ordene que continúe, mediante la instrucción **CONT**.

   Solución:

```
10  INPUT "INTRODUCE UN NUMERO"; N
20  IF 1 < = N AND N < = 5 THEN PRINT "EL NUMERO ESTA ENTRE 1
y 5": GOTO 40
30  PRINT "EL NUMERO NO ESTA ENTRE 1 y 5"
40  STOP
50  GOTO 10
60  END
```

3. Dados los tres lados de un triángulo, hacer un programa que indique si es isósceles, equilátero o escaleno.

Solución:

```
 1 REM TIPO DE TRIANGULO SEGUN SUS LADOS
10 INPUT "LADOS"; A, B, C
20 IF A = B AND A = C THEN PRINT "EQUILATERO": GOTO 50
30 IF A = B OR A = C OR B = C THEN PRINT "ISOSCELES": GOTO 50
40 PRINT "ESCALENO"
50 END
```

EJERCICIOS DE RECAPITULACIÓN

4.1. ¿El siguiente programa imprimirá algo en la pantalla? Da una explicación.

```
10 LET R = 5 ↑ 3
20 GOTO 40
30 PRINT R
40 END
```

4.2. Escribe un programa que imprima, en columna, 8 veces el número 5.1.

4.3. Modifica el programa anterior para que escriba los números en fila.

4.4. Explica lo que hace el siguiente programa.

```
10 LET C = 0
20 INPUT "INTRODUCE UN NUMERO"; N
30 LET C = C + 1
40 IF C > N THEN GOTO 70
50 PRINT "SIGUE"
60 GOTO 30
70 PRINT "TERMINA"
80 END
```

4.5. ¿Qué imprimirá el siguiente programa para X = 4? ¿Y para X = 83?

```
10 INPUT "INTRODUCE EL NUMERO X"; X
20 IF X < 50 THEN GOTO 40
30 STOP
40 PRINT X
50 GOTO 10
60 END
```

4.6. Escribe en tu propio lenguaje las siguientes condiciones expresadas en lenguaje BASIC:

a) IF X < 5 THEN GOTO 70
b) IF A < > –1 THEN GOTO 20
c) IF X < 20 AND X > 10 THEN GOTO 100
d) IF X < > Y OR X = Z THEN GOTO 80

4.7. Escribe las siguientes condiciones en lenguaje BASIC.

a) Si el valor de X es menor o igual que cero pasa a la instrucción **30.**
b) Si el valor de B_1 es mayor que el valor B_2 pasa a la instrucción **350.**
c) Si el valor de X es mayor que Y y mayor que Z pasa a la instrucción **500.**
d) Si el valor de X es mayor que Y o mayor que Z pasa a la instrucción **320.**

4.8. Escribe en lenguaje BASIC:

a) Si Z es mayor o igual que A al cuadrado, entonces pasa a la línea **150.**
b) Si 3 veces A es igual a 12 entonces pasa a la línea **80.**

4.9. Escribe en una frase lo que expresan las siguientes instrucciones:

a)
```
10 IF A > B + 2 THEN GOTO 500
20 GOTO 200
```
b)
```
10 IF C > N THEN GOTO 205
20 LET C = C + 1
```

4.10. Escribe una o más instrucciones en lenguaje BASIC que expresen lo siguiente:

a) Si X es mayor que Y más 2 ir a la instrucción **220;** en caso contrario ir a la instrucción **125.**
b) Si I es igual a cero ir a la instrucción **150,** y si no es así aumenta el valor de J en una unidad.

4.11. Indica cuál de las dos instrucciones es incorrecta y explica por qué.

a) 5 IF X ↑ 2 = 100 THEN GOTO 50
b) 30 IF X < = 50 THEN X = X + 5

4.12. ¿Qué imprimirá el siguiente programa al ser ejecutado, si se introduce 8, 7 y 5 para la variable X?

```
10 LET T = 0
20 INPUT X
30 LET T = T + 5 * X
40 PRINT T
50 GOTO 20
60 END
```

4.13. Añade una o más instrucciones para que la ejecución del programa anterior termine después de imprimirse T, para los mismos valores de X.

4.14. Escribe lo que aparecería en la pantalla si se ejecutara el programa siguiente para X = 5, –3, 6, 0, 8, –1.

```
10 INPUT X
20 IF X < O THEN GOTO 10
30 PRINT X
40 GOTO 10
50 END
```

4.15. El siguiente programa ordena los números A y B.

```
10 INPUT "INTRODUCE LOS DOS NUMEROS"; A, B
20 IF A < B THEN GOTO 50
30 IF A > B THEN GOTO 60
40 IF A = B THEN GOTO 70
50 PRINT A; "ES MENOR QUE"; B
60 PRINT A; "ES MAYOR QUE"; B
70 PRINT A; "ES IGUAL QUE"; B
80 GOTO 10
90 END
```

Sin embargo, al introducir, por ejemplo, los números 5 y 7 se obtiene en la pantalla los siguientes resultados:

```
5 ES MAYOR QUE 7
5 ES MAYOR QUE 7
5 ES IGUAL A 7
```

Modifica el programa para corregir este fallo.

4.16. Escribe un programa que lea A y B y compare ambos valores de tal manera que si A > B entonces calcule e imprima A - B, y, en caso contrario, calcule e imprima B - A.

4.17. Escribe un programa que lea dos números y los compare de tal manera que si el primero es mayor que el segundo, escriba su diferencia y su cociente; y en caso contrario, escriba su suma y su producto.

4.18. Para conocer la opinión del público sobre un determinado producto se hace una encuesta, preguntando a un conjunto de personas sobre dicho producto. Las respuestas favorables se indican con 1 y las desfavorables, con 0. Elabora un programa que cuente las preguntas favorables y desfavorables.

4.19. Escribe un programa que calcule la velocidad de un movimiento uniformemente variado, si la aceleración a es positiva. (La fórmula de la velocidad en este movimiento es $v = v_0 + a\,t$, donde v_0 es la velocidad inicial, a la aceleración, t el tiempo y v la velocidad final.)

4.20. Haz un programa que calcule la media de una serie de números ordenados en forma creciente, siendo el último número de la serie el 1,523.

4.21. Explica lo que hace el siguiente programa:

```
10 INPUT "NOTA": X
20 IF X < 5 THEN GOTO 40
30 IF X > = 5 THEN GOTO 50
40 PRINT "NO APROBADO"
50 PRINT "APROBADO"
60 GOTO 10
70 END
```

4.22. Escribe un programa que lea el salario mensual de un conjunto de personas, y cuando encuentre un salario superior a 200 000 ptas pare la ejecución, pero de tal manera que permita continuarla cuando se desee.

4.23. Explica cómo funciona el siguiente programa y qué es lo que hace.

```
10 LET X = 0
20 LET X = X + 10
30 IF X > 100 THEN GOTO 60
40 PRINT X
50 GOTO 20
60 END
```

4.24. Escribe un programa similar al anterior que imprima los múltiplos de 5 comprendidos entre 1 535 y 1 640, excluidos estos números.

4.25. Hacer un programa que averigüe si un número es par o impar.

4.26. Explica cómo funciona el siguiente programa y lo que hace.

```
10 LET A = 32
20 LET A = A + 3
30 IF A < 51 AND A > 29 THEN GOTO 50
40 STOP
50 PRINT A
60 GOTO 20
70 END
```

4.27. Escribe un programa que introduzca números sucesivamente, averigüe cuál es el mayor de los introducidos hasta el momento y dé el informe correspondiente.

4.28. Modifica el programa anterior de tal manera que escriba en una columna el número introducido y en otra, el mayor introducido hasta el momento. Como cabeceras de ambas columnas deberá escribir

NÚMERO EL MAYOR

4.29. Dados cuatro números A, B, C y D, escribir un programa que haga I = 1 si los cuatro son iguales; I = 2 si de los cuatro, tres son iguales; I = 3 si dos son iguales; I = 4 si son iguales dos a dos; e I = 5 si todos son diferentes.

5. Datos de un programa

1. Instrucciones READ y DATA

Los programas que contienen una o varias instrucciones **INPUT** presentan el inconveniente de que en el momento de la ejecución ésta se interrumpe para dar lugar a que se introduzcan los datos.

Ahora bien, si el programa tiene que utilizar una gran cantidad de datos, el tiempo total de ejecución puede llegar a ser muy largo, de ahí que en muchos casos convenga que el mismo programa los contenga. Esto se consigue mediante las instrucciones **READ** y **DATA,** las cuales siempre actúan juntas, como se puede observar en el siguiente programa:

```
10  READ A, B, C, D
20  LET S = A + B + C + D
30  PRINT "LA SUMA ES:"; S
40  READ E, F
50  LET P = E * F
60  PRINT "EL PRODUCTO ES:"; P
70  DATA 1, 2, 3, 4
80  DATA 5, 6
90  END
```

Las instrucciones **10 READ A, B, C, D** y **READ E, F** leen los datos que se encuentran en las instrucciones **70 DATA** y **80 DATA.** Los valores se van asignando secuencialmente.

Así, al ejecutarse este programa, a las variables A, B, C, D, E y F contenidas en las

instrucciones **READ,** se asignan estos valores: A = 1, B = 2, C = 3, D = 4, E = = 5 y F = 6, y en la pantalla aparece lo siguiente:

```
LA SUMA ES: 10
EL PRODUCTO ES: 30
```

Observaciones relativas a READ y DATA

— La instrucción **READ** manda al ordenador que lea los datos que se encuentran en las instrucciones **DATA** y que no han sido leídos todavía.

— La instrucción **DATA** es un almacén de datos, los cuales serán leídos secuencialmente por instrucciones **READ.**

— Las instrucciones **DATA** pueden estar en cualquier lugar del programa, por ser instrucciones no ejecutables (sólo son almacenes de datos). No obstante, conviene que estén todas juntas, al principio o al final del programa.

— En las instrucciones **READ** y **DATA,** las variables y los datos tienen que ir separados por comas.

2. Falta de datos

En un programa que contenga instrucciones **READ** y **DATA** puede darse el caso de que algunos datos se queden sin leer; este exceso de datos no ocasiona la interrupción del programa. Por el contrario, si las instrucciones **READ** necesitan leer más datos de los almacenados en las instrucciones **DATA,** en el momento de intentar leer un dato inexistente la ejecución se detiene, apareciendo en la pantalla un mensaje de error, que puede ser el siguiente:

? OUT OF DATA ERROR IN... (Número de instrucción que da lugar al error.)

Consideremos el siguiente programa que calcula el cuadrado de los números naturales del 1 al 9.

```
10  READ N
20  LET C = N * N
30  PRINT "EL CUADRADO DE"; N; "ES"; C
40  GOTO 10
50  DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
60  END
```

Después de teclear **RUN** y ENTER, se inicia la ejecución, apareciendo en la pantalla el cuadrado de los números naturales del 1 al 9. Pero inmediatamente la ejecución se detiene apareciendo un mensaje de error en la instrucción **10.**

Esta interrupción se debe a que la instrucción **40 GOTO 10** obliga al microordenador a ejecutar la instrucción **10 READ N** un número ilimitado de veces, teniendo la instrucción **50 DATA** un número limitado de datos. Llega entonces un momento en que después de haberse utilizado todos los datos, la instrucción **READ** busca el siguiente, y al no encontrarlo (porque no existe), se detiene la ejecución.

Si en este programa se quisiera calcular el cuadrado de más números naturales, por ejemplo del 10 al 15, habría que añadir la instrucción
60 DATA 10, 11, 12, 13, 14, 15

Ejercicios

5.1. ¿Qué datos no se leerán al ejecutar el siguiente programa?

```
10  READ A, B, C
20  PRINT B
30  READ X, Y
40  PRINT A, Y
50  READ U, V, W, L
60  DATA 1.1, 1.2, 1.3, 1.4
70  DATA 2.3, 3.3, 3.7, 4.8, 5.9, 7.3
80  DATA −2.3, −4.5
90  END
```

5.2. Elabora un programa que compare un número con la siguiente serie de datos: 1, 3, 5, 7 y 9, dé un informe afirmativo en el caso de que el número introducido coincida con alguno de la serie y dé un mensaje de error, a la falta de datos, en caso de que no coincida.

5.3. Escribe un programa que contenga una instrucción **DATA** con los diez primeros números enteros positivos, y −1 en último lugar. Dicho programa deberá ir sumando los diez primeros números y cuando lea −1 imprimirá la suma; a continuación que detenga la ejecución.

3. Instrucción RESTORE

En algunos programas es necesario volver a leer los datos almacenados en una instrucción **DATA.** Esto se consigue mediante la instrucción **RESTORE.**

Esta instrucción cumple la función de ***regenerar*** los valores de la instrucción **DATA,** haciendo que se asigne el primer dato a la primera variable de la primera instrucción **READ** situada después de **RESTORE,** produciendo una asignación secuencial para el resto de las variables y datos.

Así, al ejecutarse este programa:

```
10  READ A, B, C
20  PRINT A; B; C
30  READ A, B, C
40  PRINT A; B; C
50  RESTORE
60  READ A, B, C
70  PRINT A; B; C
80  DATA 1, 2, 3, 4, 5, 6
90  END
```

se imprime en la pantalla

```
1  2  3
4  5  6
1  2  3
```

Con la inclusión en el programa de la instrucción **50 RESTORE** se consigue que los primeros valores de **DATA** se asignen por segunda vez a las variables A, B y C en la instrucción **60 READ A, B, C.**

En resumen: En las instrucciones **READ** de este programa se hacen las siguientes asignaciones:

— En **10 READ A, B, C** se hace A = 1, B = 2, C = 3.
— En **30 READ A, B, C** se hace A = 4, B = 5, C = 6.
— En **60 READ A, B, C** se hace A = 1, B = 2, C = 3.

Ejercicios

5.4. ¿Por qué razón en el siguiente programa las instrucciones **READ** se comienzan a leer siempre por el primer dato, sin seguir el orden secuencial?

```
10 READ A
20 RESTORE
30 READ H, I, J
40 RESTORE
50 READ X, Y
60 DATA 2, -3, 4, -5, 2.3, 7
70 END
```

5.5. Añade instrucciones **RESTORE** al siguiente programa para que no se interrumpa su ejecución por falta de datos.

```
10 READ X, Y
20 PRINT X, Y
30 READ A, B, C
40 PRINT (A * B)/C
50 READ L, M
60 PRINT L ↑ M
70 DATA 4, 7, 2
80 END
```

4. Resumen de las instrucciones estudiadas en este capítulo

- READ

— ***Significado:*** lee.
— ***Formato:*** n READ {lista de variables}
Las variables han de ir separadas por comas.
— ***Función:*** asigna a las variables incluidas en **READ** los datos almacenados en una instrucción **DATA,** en forma secuencial.

- DATA
  - — ***Significado:*** datos.
  - — ***Formato:*** n DATA {lista de datos}
    Los datos van separados por comas.
  - — ***Función:*** Contiene los datos que serán leídos por una instrucción **READ.**

- RESTORE
  - — ***Significado:*** regenera.
  - — ***Formato:*** n RESTORE.
  - — ***Función:*** permite volver a leer desde el principio los datos contenidos en las instrucciones **DATA.**

EJERCICIOS RESUELTOS

1. Escribir un programa que permita practicar las tablas de multiplicar, de tal manera que vayan apareciendo en la pantalla los distintos productos que se pueden obtener multiplicando dos números comprendidos entre 1 y 9, ambos inclusive, pregunte cuál es su producto, y si el valor introducido es incorrecto vuelva a repetir la pregunta.

   Solución:

```
  1 REM TABLAS DE MULTIPLICAR
 10 READ A
 20 RESTORE
 30 READ B
 40 PRINT "CUANTO VALE"; A; "*"; B
 50 INPUT X
 60 IF X = A * B THEN PRINT "CORRECTO": GOTO 90
 70 PRINT "INCORRECTO; ESCRIBE OTRO RESULTADO"
 80 GOTO 40
 90 IF B < 9 THEN GOTO 30
100 LET A = A + 1
110 IF A < = 9 THEN GOTO 20
120 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
130 END
```

2. Sabemos que el microordenador puede escribir los números en notación entera, real y exponencial. Hacer un programa que sume, por ejemplo, los números 275, 38.45 y 2.3 E 7 utilizando **READ** y **DATA**.

Solución:

```
10  READ A, B, C
20  LET S = A + B + C
30  PRINT S
40  DATA 275, 38.45, 2.3 E 7
50  END
```

EJERCICIOS DE RECAPITULACIÓN

5.1. Las siguientes instrucciones contienen errores. Identifícalos.

a) 30 READ X, 30
b) 10 READ X, Y; Z
c) 40 DATA 5, 6, 7.5, X, 3.4
d) 70 DATA 3, 4; 5

5.2. Un programa contiene las siguientes instrucciones.

```
30 READ A, B, C
.....
60 RESTORE
70 READ W, X, Y, Z
.....
200 DATA 1, 3, 5, 7, 9, 11, 13
```

¿Qué valores se asignan a cada variable? ¿Qué valores se les asignarían si no estuviese la instrucción **60 RESTORE**?

5.3. Escribe las instrucciones **READ** y **DATA** que asignan los valores −1.6 E − 6 −500, 0.4077, 100, 110 a las variables C1, C2, C3, C4, C5.

5.4. ¿Qué imprimiría en la pantalla el microordenador al ejecutar este programa?

```
10  READ A, B
20  PRINT A, B
30  READ C
40  PRINT A + B + C
50  DATA 2, 7
60  END
```

5.5. ¿Qué valores imprimirá el siguiente programa al ejecutarse?

```
10 READ X
20 PRINT X
30 IF X < > 3 THEN GOTO 10
40 DATA 7, 6, 5, 4, 3, 2, 1
50 END
```

5.6. Diseña un programa que contenga en una instrucción **DATA** los números primos 2, 3, 5, 7, 11, 13, 17, 19 y 23, y que al introducir un número positivo menor que 25 imprima si es primo o no.

5.7. Haz un programa que al introducir una cantidad en pesetas dé su equivalente en libras esterlinas, dólares, marcos, liras y francos.
El programa almacenará en una instrucción **DATA** las siguientes equivalencias:

1 libra = 210 ptas
1 dólar = 170 ptas
1 marco = 57 ptas
1 lira = 0.09 ptas
1 franco = 18 ptas.

5.8. En un observatorio se han registrado en una semana las siguientes temperaturas:

D	L	M	X	J	V	S
20	22	25	24	23	21	20

Escribe un programa que introduzca estos datos mediante instrucciones **READ** y **DATA**, y calcule la temperatura media.

5.9. En una encuesta las respuestas afirmativas se registran con 1 y las negativas, con 0. Se encuesta a 50 personas, estando la última respuesta seguida del valor −1. Escribe un programa que incluya las siguientes instrucciones **DATA** y que cuente el número de respuestas afirmativas y negativas.

```
900 DATA 1, 2, 2, 1, 2, 2, 1, 1, 2, 1
910 DATA 2, 1, 1, 1, 2, 1, 2, 2, 2, 1
920 DATA 1, 2, 1, 2, 1, 1, 2, 1, 2, 2
930 DATA 1, 1, 1, 2, 2, 1, 1, 2, 2, 2
940 DATA 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, −1
```

5.10. El siguiente programa muestra cómo se pueden hacer dibujos sencillos con las instrucciones BASIC conocidas hasta ahora.

```
  1  REM IMPRIME UN DIBUJO
 10  READ X
 20  IF X = -1 THEN GOTO 920
 30  IF X = 1 THEN GOTO 60
 40  IF X = 2 THEN GOTO 80
 50  IF X = 3 THEN GOTO 100
 60  PRINT "***************"
 70  GOTO 10
 80  PRINT "***"
 90  GOTO 10
100  PRINT "**********"
110  GOTO 10
900  DATA 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3
910  DATA 2, 2, 2, 2, 2, 1, 1, 1, -1
920  END
```

¿Qué letra dibuja este programa?

5.11. Basándote en el ejercicio anterior haz un programa que dibuje la letra C en la pantalla.

5.12. Diseña un programa que dibuje una L en la pantalla.

5.13. Haz un programa que dibuje la letra U en la pantalla.

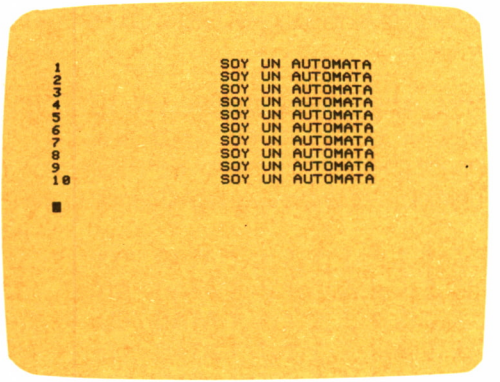
6. Repetición de procesos: Bucles

1. Cómo repite el microordenador un proceso muchas veces. Instrucción FOR-NEXT

- Una de las cualidades más importantes del microordenador es su capacidad de repetir procesos un gran número de veces, sin equivocarse y empleando muy poco tiempo.
- La repetición de una tarea la realiza el microordenador mediante dos instrucciones, cuya función puede verse en el siguiente programa que escribe diez veces la frase SOY UN AUTÓMATA.

```
10  REM
20  FOR X = 1 TO 10
30  PRINT X, "SOY UN AUTOMATA"
40  NEXT X
50  END
```

Después de teclear **RUN** y [ENTER] se ejecuta el programa apareciendo en la pantalla lo que muestra la fotografía.



Las instrucciones claves de este programa son la **20 FOR X = 1 TO 10,** que significa ***desde X = 1 hasta 10,*** y la **40 NEXT X** (siguiente X).

— Cuando se inicia la ejecución de la instrucción **20,** la variable X adquiere el valor 1 y la instrucción **30** imprime este valor y la frase SOY UN AUTÓMATA.

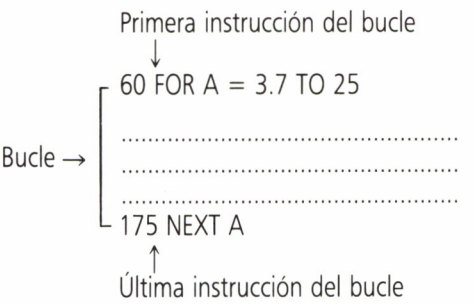
— A continuación, se ejecuta la instrucción **40 NEXT X,** que significa ***siguiente X.*** Esta instrucción hace saltar la ejecución del programa a la instrucción **20,** que al ejecutarse por segunda vez, hace adquirir a X el valor 2. Acto seguido, la instrucción **30** imprime 2 y la frase SOY UN AUTÓMATA.

— A continuación, se ejecuta nuevamente la instrucción **40 NEXT X** que hace saltar la ejecución a la instrucción **20,** repitiéndose otra vez lo mismo para X = = 3, y así sucesivamente.

El proceso se ejecuta por última vez cuando la variable X alcanza el valor **10,** y después de imprimirse este valor y la frase SOY UN AUTÓMATA, la ejecución continúa en la instrucción **50 END,** dándose por terminada la ejecución del programa.

- Vemos que, mediante las dos instrucciones **FOR** y **NEXT**, podemos ejecutar varias veces un conjunto de instrucciones; éstas, juntamente con **FOR** y **NEXT** se denominan ***ciclo*** o ***bucle***.

  La primera instrucción de un bucle es siempre **FOR** y la última, **NEXT**.

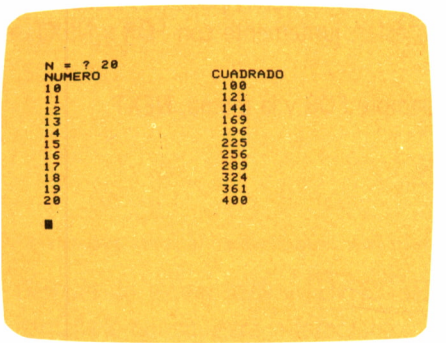


2. Un programa que confecciona una tabla de cuadrados

- El siguiente programa confecciona una tabla de cuadrados de los números naturales del 10 a N, ambos inclusive

```
 5  REM TABLA DE CUADRADOS
10  INPUT "N ="; N
20  PRINT "NUMERO", "CUADRADO"
30  FOR I = 10 TO N
40  LET C = I ↑ 2
50  PRINT I, C
60  NEXT I
70  END
```

La ejecución de este programa se interrumpe nada más comenzar, apareciendo una ? o una L, según el microordenador. Para que continúe la ejecución se introduce desde el teclado un valor para N (introducimos, por ejemplo, **20**). Una vez introducido, sigue la ejecución imprimiéndose las sucesivas filas de la tabla, tal como se observa en la fotografía. El bucle se ejecuta hasta que I alcanza el valor N inclusive, y una vez ejecutado el bucle para I = N se transfiere el control a la instrucción siguiente a la **60 NEXT I,** es decir, a la **70 END.**



- Tanto en este programa como en el anterior, vemos que después de la palabra **FOR** se encuentra una variable que adquiere un ***valor inicial*** (el número que se encuentra antes de la palabra **TO**) y un ***valor final*** o ***tope*** (el número que está después de la palabra **TO**). Ejemplo:

```
30 FOR A = 5 TO 23
```

Variable de control — *Valor inicial de la variable* — *Valor final (tope) de la variable*

En los casos considerados hasta ahora, la variable aumenta sucesivamente de valor «saltando» de 1 en 1, desde que adquiere el valor inicial hasta que llega al valor final; se dice que la instrucción tiene un ***salto*** o ***incremento*** de 1.

Ejercicios

6.1. Indica qué escribiría en la pantalla el siguiente programa:

```
10  FOR N = 1 TO 6
20  PRINT "LA RAIZ CUADRADA DE"; N; "ES"; SQR(N)
30  NEXT N
40  END
```

6.2. Escribe un programa que imprima en la pantalla

1. VEO LA VIDA CON OPTIMISMO
2. VEO LA VIDA CON OPTIMISMO

.....

.....

15. VEO LA VIDA CON OPTIMISMO

6.3. Haz un programa que confeccione una tabla de las raíces cuadradas de los números comprendidos entre 100 y 112, ambos inclusive, con la siguiente cabecera:

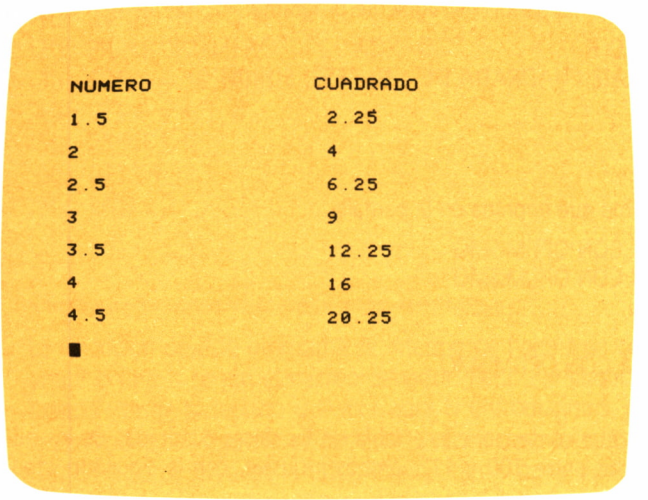
NÚMERO RAÍZ CUADRADA

3. Instrucción FOR ... TO ... con incremento distinto de 1

En su forma más simple, la instrucción FOR ... TO ... avanza incrementando en 1 la variable de control, hasta llegar a un valor final o tope. Pero añadiendo la palabra **STEP**, que significa ***paso***, el incremento puede ser distinto de 1, tal como se ve en el siguiente programa que calcula los cuadrados de los números 1.5, 2, 2.5, 3, 3.5 y 4.5 disponiéndolos en forma de tabla.

```
 5  REM CUADRADOS
10  PRINT "NUMERO", "CUADRADO"
20  FOR X = 1.5 TO 4.5 STEP 0.5
30  LET C = X ↑ 2
40  PRINT
50  PRINT X, C
60  NEXT X
70  END
```

Al ejecutar el programa (RUN ENTER) aparece la tabla que muestra la fotografía:



Observamos en la instrucción **20 FOR X = 1.5 TO 4.5 STEP 0.5**, que los valores iniciales y finales no tienen por qué ser necesariamente enteros y que la variable X se incrementa también en un valor no entero, con la utilización de la palabra **STEP**.

4. Algunos programas con la instrucción FOR ... TO ... STEP

- El incremento de la variable también puede ser un valor negativo. En este caso, en lugar de crecer los valores de la variable, decrecen, como se puede comprobar en este programa:

```
 5  REM CUADRADOS
10  PRINT "NUMERO", "CUADRADO"
20  FOR X = 4.5 TO 1 STEP -0.5
30  LET C = X ↑ 2
40  PRINT
50  PRINT X, C
60  NEXT X
70  END
```

que imprime la tabla anterior invertida

```
NUMERO          CUADRADO
3.5              12.25
3                9
2.5              6.25
2                4
1.5              2.25
1                1
■
```

- La instrucción **FOR** también acepta variables y expresiones aritméticas en lugar de valores numéricos, pero en el momento de la ejecución las evalúa previamente llegando a un valor numérico inicial y final. Por ejemplo, en la instrucción **20 FOR P = Q TO 2 * Q − 1**, del siguiente programa, el valor inicial de la variable P es el de la variable Q, es decir, ***4***, y el valor final corresponde al de la expresión 2 * Q − 1, o sea, 2 * 4 − 1 = 8 − 1 = ***7***.

```
10  LET Q = 4
20  FOR P = Q TO 2 * Q − 1
30  PRINT P
40  NEXT P
50  END
```

Según se va ejecutando este programa, va escribiendo en la pantalla los valores sucesivos que adquiere P, lo que permite comprobar la extensión del bucle:

```
 4
 5
 6
 7
■
```

- En este otro programa el valor inicial y el final de la variable de control P, también viene dado por dos expresiones aritméticas; el inicial corresponde al valor de la expresión Q/2, es decir, 4/2 = ***2,*** y, el final, al valor numérico de Q * 3, o sea, 4 * 3 = ***12***.

```
10  LET Q = 4
20  FOR P = Q/2 TO Q * 3
30  PRINT P
40  NEXT P
50  END
```

La ejecución del programa hace aparecer en la pantalla los sucesivos valores que adquiere la variable P, uno debajo de otro.

5. Entrada y salida de un bucle

- Solamente se puede entrar en un bucle por su primera instrucción FOR ... TO ... Es decir, en un programa no se puede acceder directamente a una instrucción interior del bucle sin antes haber pasado por la FOR ... TO ... Por ejemplo, si se intentara ejecutar el siguiente programa, inmediatamente se interrumpiría, apareciendo en la pantalla un mensaje de error.

```
Bucle →   10  LET N = 5
          20  GOTO 40
          30  FOR X = N TO 10
          40  PRINT
          50  PRINT X ↑ 2
          60  NEXT X
          70  END
```

- Por el contrario, se puede salir de un bucle mediante una instrucción **GOTO**, sin que por ello se detenga la ejecución. Así, en el siguiente programa la instrucción **30 IF A > 10 THEN GOTO 70** hace saltar la ejecución a la instrucción **70 PRINT A**, sin que por ello se interrumpa la ejecución del programa.

```
          10  INPUT N
          20  FOR A = 1 TO N
Bucle →   30  IF A > 10 THEN GOTO 70
          40  LET X = -A
          50  PRINT X
          60  NEXT A
          70  PRINT A
          80  END
```

Ejercicios

6.4. Elabora un programa que imprima los 25 primeros números pares.

6.5. Indica qué imprime en la pantalla el siguiente programa.

```
10  FOR L = 12 TO 30 STEP 3
20  PRINT L;
30  NEXT L
40  END
```

6.6. Escribe un programa que imprima en la pantalla los múltiplos de 5 comprendidos entre 100 y 150.

6.7. Elabora un programa que escriba en la pantalla los treinta primeros números impares negativos.

6.8. Escribe los resultados que imprimiría el siguiente programa si fuera ejecutado.

```
10 LET A = -4
20 LET B = 5
30 FOR R = A + B TO A * B STEP -2
40 PRINT R, R ↑ 3
50 NEXT R
60 END
```

6. Bucles en el interior de otros bucles: bucles anidados

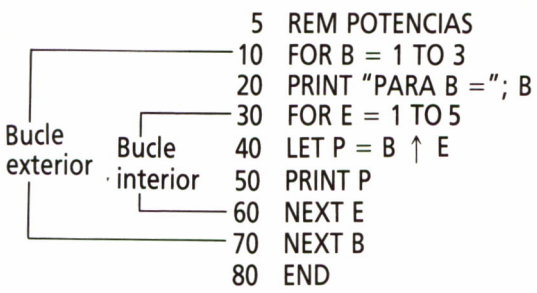
- Nos planteamos este problema: calcular las potencias 1.ª, 2.ª, 3.ª, 4.ª, y 5.ª de los números 1, 2 y 3, es decir, las potencias de base B = 1, 2 y 3 de exponente E = 1, 2, 3, 4, y 5. En concreto, se trata de hacer lo siguiente:

  — Para B = 1, calcular:
  1^1
  1^2
  1^3
  1^4
  1^5

  — Para B = 2, calcular:
  2^1
  2^2
  2^3
  2^4
  2^5

  — Para B = 3, calcular:
  3^1
  3^2
  3^3
  3^4
  3^5

  El programa que transcribimos lleva a cabo estos cálculos utilizando dos bucles, uno de variable B (el «exterior»: instrucciones **10-70**) y otro de variable E (el «interior»: instrucciones **30-60**).



¿Cómo funciona este programa?

— Primeramente, la instrucción **10 FOR B = 1 TO 3,** inicializa el bucle **10 - 70,** que en su primera ejecución almacena en la variable B el valor 1. A continuación, la instrucción **20** imprime PARA B = 1.

— La instrucción **30 FOR E = 1 TO 5** inicializa el bucle **30 - 60,** de variable E. Este bucle da "cinco vueltas", en las cuales las instrucciones **40** y **50** calculan e imprimen, respectivamente, las potencias 1^E, es decir, 1^1, 1^2, 1^3, 1^4 y 1^5. Después de la "quinta vuelta", la ejecución sigue en la instrucción **70 NEXT B,** que devuelve el control a la instrucción **10.**

— Por segunda vez se ejecuta el bucle **10 - 70,** haciéndose B = 2 e imprimiéndose PARA B = 2. A continuación, el bucle **30 - 60** da otras "cinco vueltas", en las cuales se calculan e imprimen las potencias 2^E, es decir, 2^1, 2^2, 2^3, 2^4 y 2^5. Después de la "quinta vuelta", la ejecución sigue en la instrucción **70,** que otra vez devuelve el control a la instrucción **10.**

— Por tercera y última vez se ejecuta el bucle **10 -70,** haciéndose B = 3 e imprimiéndose PARA B = 3. A continuación, el bucle **30 -60** da nuevamente otras "cinco vueltas", en las cuales se calculan e imprimen las potencias 3^E, o sea, 3^1, 3^2, 3^3, 3^4 y 3^5. Después de la "quinta vuelta", la ejecución sigue en la instrucción **70** (que ya no devuelve el control a la **10**), y después en la **80 END;** ésta da por terminada la ejecución del programa.

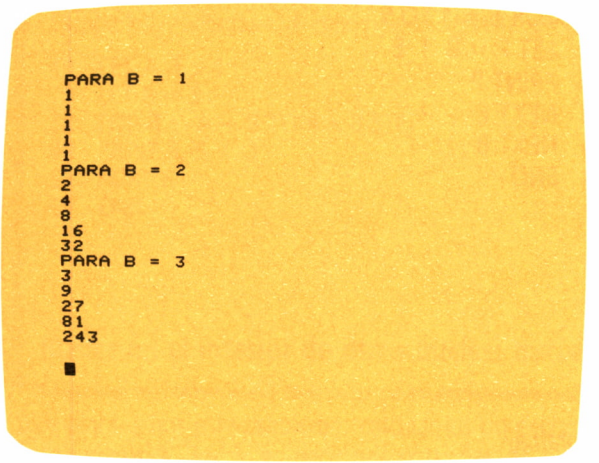
Resumiendo: Este programa contiene dos bucles:

— Uno ***exterior*** de variable B, que da «tres vueltas».

— Otro ***interior*** de variable E, que da «cinco vueltas» por cada vuelta del primero (en total, «quince vueltas»).

Se dice que ambos bucles son ***anidados*** (el bucle interior está anidado en el exterior).

Al ejecutarse este programa aparece en la pantalla lo que muestra la fotografía.



- Un bucle interno tiene que estar incluido en su totalidad en otro más externo, tal como se muestra en este ejemplo:

```
10 FOR I = ...
  20 FOR J = ...
    30 FOR K = ...
    .....
    .....
    .....
    100 NEXT K
  110 NEXT J
120 NEXT I
```

En cambio, la disposición de los bucles en el siguiente ejemplo es ***incorrecta:***

```
10 FOR I = ...
  20 FOR J = ...
    30 FOR K = ...
    .....
    .....
    .....
    70 NEXT I
  80 NEXT J
90 NEXT K
```

Vemos, entonces, que los bucles no se pueden «cruzar» y que se ejecutan «de dentro hacia afuera».

Ejercicios

6.9. Escribe lo que imprimiría el siguiente programa al ser ejecutado.

```
10 FOR J = 1 TO 4
20 FOR K = 1 TO 2
30 PRINT J; ","; K,
40 NEXT K
50 NEXT J
60 END
```

Observación: Advertir el diferente significado de las dos comas en la instrucción **30**: la coma entre comillas se imprime a continuación del valor de J y la coma no entrecomillada significa que el siguiente valor del par J, K se imprime en la segunda zona de la pantalla.

6.10. Haz un programa que escriba en la pantalla los siguientes pares de valores y en la disposición que se indica.

```
5,1
5,2
5,3
6,1
6,2
6,3
```

6.11. Elabora un programa que imprima la siguiente tabla. (Suponer una pantalla de dos zonas.)

```
1,1     1,2
2,1     2,2
3,1     3,2
4,1     4,2
5,1     5,2
```

6.12. Escribe un programa que genere el producto cartesiano de los conjuntos:

A = {1, 2, ..., 10}
B = {1, 2, ..., 5}

Es decir, el programa deberá generar los pares

```
1,1   1,2   1,3   1,4   1,5
2,1   2,2   2,3   2,4   2,5
.....
.....
.....
10,1  10,2  10,3  10,4  10,5
```

6.13. Escribe un programa que imprima los resultados posibles que se obtienen al lanzar dos dados. (Algunos resultados posibles son: 1,1; 1,6; 2,3; 5,6; 6,6; etcétera.)

6.14. Elabora un programa que genere las tablas de multiplicar del 1, 2, ..., y 10.

7. Las instrucciones FOR – NEXT (Resumen)

- n FOR... TO... STEP

  — ***Significado:*** desde ... a ... en pasos de ...
  — ***Formato:*** n FOR V = i TO t STEP s
  — ***Función:*** Ejecuta un conjunto de instrucciones llamado ***bucle,*** que empieza en FOR y termina en NEXT. El bucle se inicia cuando la variable V toma el valor inicial ***i,*** y termina después de alcanzar el valor final o tope ***t***. La variable V incrementa su valor en función del incremento ***s***.

- n' NEXT V

  — ***Significado:*** siguiente V.
  — ***Formato:*** n' **NEXT** V, donde n' es el número de la instrucción y V la variable definida en la instrucción **FOR.**
  — ***Función:*** Transfiere la ejecución a la instrucción n **FOR ... TO ... STEP** siempre que el valor de la variable V sea menor o igual que el valor final o tope ***t***.

EJERCICIOS RESUELTOS

1. Escribir lo que se obtendría en la pantalla si se ejecutara el siguiente programa.

```
10 LET M = 0
20 FOR I = 0 TO 17
30 IF I = M THEN LET M = M + 4: PRINT I; "ES MULTIPLO
DE 4"
40 NEXT I
50 END
```

   Solución:

```
0 ES MULTIPLO DE 4
4 ES MULTIPLO DE 4
8 ES MULTIPLO DE 4
12 ES MULTIPLO DE 4
16 ES MULTIPLO DE 4
```

2. Escribir un programa que imprima en la pantalla múltiplos de 3, de tal manera que si el múltiplo es par salte al siguiente, y si es impar, escriba además su consecutivo, y después salte al siguiente múltiplo.

Solución:

```
 5  REM IMPRIME 0, 3, 4, 6, 9, 10, ...
10  CLS: PRINT
15  INPUT "TOPE"; N
20  LET D = 0
30  FOR M = 0 TO N STEP 3
40  IF D = 0 THEN PRINT M; ","; : LET D = 1 : GOTO 70
50  LET D = 0
60  PRINT M; ","; M + 1; ",";
70  NEXT M
80  END
```

EJERCICIOS DE RECAPITULACIÓN

6.1. Analiza los dos programas siguientes. ¿Qué imprimen en la pantalla? Obtén una conclusión.

```
10  LET F = 1
20  IF F > 8 THEN GOTO 99
30  PRINT "F="; F
40  LET F = F + 1
50  GOTO 20
99  END
```

```
10  FOR F = 1 TO 8
20  PRINT "F="; F
30  NEXT F
99  END
```

6.2. Escribe lo que imprime en la pantalla el siguiente programa, al ser ejecutado.

```
10  LET X = 0
20  FOR K = 1 TO 4
30  LET X = X + K
40  NEXT K
50  PRINT X
60  END
```

6.3. Haz un programa que produzca los mismos resultados en la pantalla que el anterior, sin utilizar las instrucciones **FOR - NEXT.**

6.4. ¿Qué imprimirá en la pantalla el siguiente programa?

```
10 LET P = 1
20 FOR K = 1 TO 4
30 LET P = P * K
40 NEXT K
50 PRINT P
60 END
```

6.5. Analiza el siguiente programa y escribe lo que imprime en la pantalla.

```
10 LET N = 1
20 FOR K = 1 TO N
30 PRINT "*"
40 NEXT K
50 PRINT
60 LET N = N + 1
70 IF N > 10 THEN GOTO 90
80 GOTO 20
90 END
```

6.6. ¿Qué imprime el siguiente programa?

```
10 LET S = 0
20 FOR K = 1 TO STEP 2
30 LET S = S + K
40 NEXT K
50 PRINT S
60 END
```

6.7. ¿Son correctos los programas que incluyen las siguientes instrucciones?

a)
```
20 FOR I = 1 TO 100 STEP 2
.....
80 NEXT J
.....
```

b)
```
50 FOR N = N 1 TO N 2
.....
90 NEXT N
.....
120 IF N = 10 THEN GOTO 75
```

6.8. Analiza los siguientes programas e indica si son correctos.

a)
```
100 FOR K = 3 TO –3 STEP –1
.....
130 PRINT X ↑ K
.....
150 NEXT K
160 END
```

b)
```
25 FOR X = 1 TO 2 STEP 0.05
.....
50 FOR Y = 1 TO 10 STEP 0.1
.....
75 NEXT X
.....
100 NEXT Y
.....
```

c)
```
100 FOR I = 1 TO M
.....
125 FOR J = 1 TO N
.....
135 FOR K = 1 TO M + N
.....
160 NEXT K
.....
180 NEXT J
.....
200 FOR J = 1 TO N
.....
225 NEXT J
235 FOR K = 1 TO M + N
.....
240 NEXT K
.....
250 NEXT I
.....
```

d)
```
10 FOR A = 5 TO 10
20 FOR B = 1 TO 3
.....
60 NEXT B
70 NEXT A
.....
100 FOR B = 1 TO 4
110 FOR A = 4 TO 10
.....
140 NEXT B
.....
180 NEXT A
.....
```

6.9. Escribe un programa que calcule e imprima la suma de los números enteros comprendidos entre 1 y N, ambos inclusive, siendo N un valor que se introduce mediante una instrucción **INPUT.**

6.10. Escribe un programa que confeccione una tabla de las raíces quintas de los números comprendidos entre 1 y 2, en intervalos de 0.1.

6.11. Haz un programa que calcule la suma

$$1^2 + 2^2 + 3^2 + \ldots + 99^2$$

6.12. Escribe un programa que calcule la media aritmética de una lista de N números utilizando una instrucción **INPUT** para introducir N y después un bucle **FOR - NEXT** en el que esté incluida otra instrucción **INPUT** o una **READ** para introducir cada número.

6.13. Elabora un programa que calcule el producto de los N primeros números naturales, sin contar el cero. (***Observación:*** El producto aludido en este enunciado se llama factorial de N y suele simbolizarse así: N!. Por ejemplo, el producto 1*2*3*4*5*6*7 se llama factorial de 7 y se indica por 7!)

7. Cadenas

1. Cadenas de caracteres

- ***Una cadena de caracteres es una sucesión de letras, números y cualquier otro símbolo gráfico, los cuales se escriben entre comillas para indicar el comienzo y el final de la cadena.***

  Los espacios en blanco también se consideran como caracteres.

  Ejemplos de cadenas:

  "PEDRO"
  "328"
  "A43"
  "42.A?"
  "CARMEN LOPEZ"

- Así como los números se guardan en variables numéricas, las cadenas de caracteres se guardan en ***variables de cadenas,*** también llamadas variables alfanuméricas. Éstas, se distinguen de las primeras, porque su nombre siempre termina con el símbolo $ (símbolo de dólar).

  Ejemplos de variables de cadenas:

  A$
  X$
  Z3$
  LL27A$

  Las variables de cadenas, como las numéricas, comienzan por una letra seguida de números o letras y terminando con $. Algunos microordenadores admiten sólo una letra seguida de $.

- En el siguiente programa se utiliza una variable de cadena para dar los buenos días a la persona que lo ejecuta.

```
 1  REM SALUDO
10  PRINT "CUAL ES SU NOMBRE"
20  INPUT N$
30  PRINT "BUENOS DIAS "; N$
40  END
```

Si se ejecuta el programa (teclear **RUN** y ENTER) aparecerá en la pantalla:

```
CUAL ES SU NOMBRE
?
```

Al introducir "PEDRO GARCÍA" (y ENTER) se obtiene este resultado en la pantalla:

```
BUENOS DIAS PEDRO GARCIA
```

Ejercicios

7.1. Si al ejecutar este programa introduces tu nombre, ¿qué obtendrás en la pantalla?

```
10 INPUT "NOMBRE": A$
20 FOR I = 1 TO 10
30 PRINT "EL ALUMNO"; A$; "ES UN GRAN PROGRAMADOR"
40 NEXT I
50 END
```

7.2. Haz un programa que lea el nombre y los apellidos de una persona y los imprima infinitas veces.

7.3. Escribe un programa que lea los días de la semana en una instrucción **DATA** y los imprima seguidos, dejando un espacio en blanco entre cada dos nombres.

2. Suma o concatenación de cadenas

- La suma o concatenación de cadenas es una operación que une dos cadenas, poniendo la segunda a continuación de la primera sin separación entre ellas.

  El símbolo con que se representa esta operación es +.

  Por ejemplo, si
  A$ = "RÍO" y B$ = "EBRO".
  A$ + B$ da como resultado "RIOEBRO", sin separación entre las dos palabras, pues no se dejó ningún espacio en blanco al definirlas.

- En el programa que transcribimos a continuación se concatenan dos cadenas:

```
 1  REM CONCATENACION
10  INPUT "QUE PIDO"; A$
20  LET B$ = "QUIERO " + A$
30  PRINT B$
40  END
```

La operación de concatenación se realiza en la instrucción **20**, uniendo la cadena "QUIERO " con el contenido de la variable A$. Si se ejecuta el programa (teclear **RUN** y ENTER) aparece en pantalla este mensaje:

```
QUE PIDO?.
```

Al introducir el dato AGUA (pulsar ENTER) se obtiene:

```
QUIERO AGUA
```

3. Longitud de una cadena

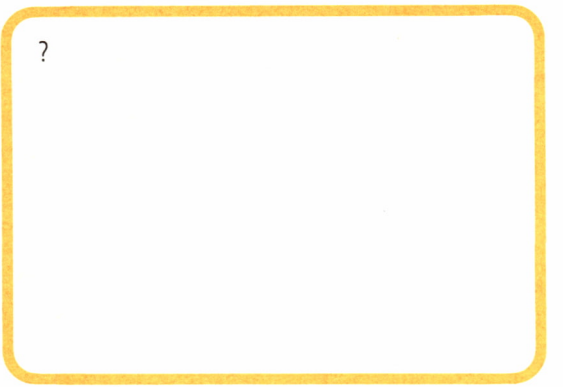
Se llama longitud de una cadena al número de caracteres que tiene.

Existe una función que permite calcular la longitud de una cadena: es la función **LEN**. Así, **LEN** (A$) determina la longitud de la cadena almacenada en la variable A$.

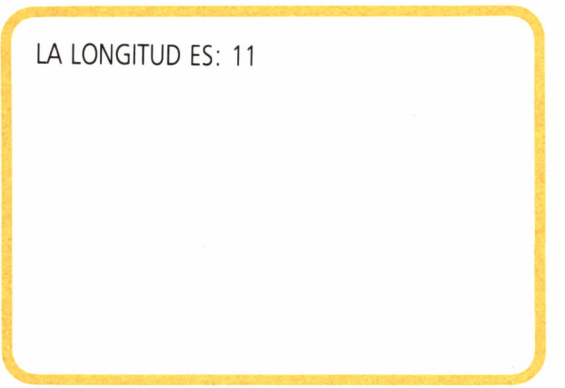
El siguiente programa aplica la función **LEN** para calcular la longitud de cualquier cadena que se introduzca desde el teclado.

```
 1  REM LONGITUD DE UNA CADENA
10  INPUT A$
20  LET N = LEN (A$)
30  PRINT "LA LONGITUD ES:"; N
40  END
```

Al ejecutar el programa (teclear **RUN** y ENTER) el microordenador pedirá una cadena para la variable A$.



Si se introduce la cadena "HOLA MIGUEL" (pulsar ENTER) se obtiene en la pantalla lo siguiente:



Observar que el espacio que separa HOLA de MIGUEL también se ha computado como un carácter.

4. Transformación de un número en una cadena numérica y recíprocamente

- Una sucesión de dígitos puede considerarse como número o como cadena. Así, 358, escrito sin comillas, hace referencia al número 358 y como tal, se le pueden aplicar las operaciones de adición, sustracción, multiplicación, etc.; en cambio, "358" es una cadena de dígitos, a la cual se le puede aplicar la operación de concatenación y la función **LEN**.
- Existen dos funciones que permiten transformar un número en cadena de dígitos, y recíprocamente: son las funciones **STR$** y **VAL**, respectivamente.

— La función **STR$** (N) transforma el valor numérico almacenado en la variable N en una cadena de dígitos. Por ejemplo, si N = 3250, la función STR$(N) transforma el número 3250 en la cadena "3250".

— La función **VAL** (B$), produce la transformación inversa, es decir, transforma una cadena de dígitos en un número. Por ejemplo, si B$ = "02315", la función **VAL**(B$) transforma la cadena "02315" en el número 2315.

- A continuación se aplican ambas funciones. El programa pide dos números, los transforma en cadenas numéricas por medio de la función **STR$**, concatena estas cadenas y el resultado lo pasa a valor numérico por medio de la función **VAL**.

```
 1  REM CONCATENACION DE CADENAS NUMERICAS
10  INPUT N
20  INPUT M
30  LET A$ = STR$(N)
40  LET B$ = STR$(M)
50  LET C$ = A$ + B$
60  LET P = VAL(C$)
70  PRINT "EL NUMERO ES:"; P
80  END
```

Si se ejecuta el programa (teclear **RUN** y ENTER) y se introducen los números 4567 y 123 el resultado que se observará en la pantalla será el siguiente:

```
EL NUMERO ES: 4567123
```

Observar que en la ejecución de este programa se han seguido los pasos descritos en su enunciado:

— Las instrucciones **10** y **20** permiten introducir dos números (en este ejemplo, 4567 y 123).

— Estos números son transformados en cadenas numéricas mediante las instrucciones **30 LET A$ = STR$(N)** y **40 LET B$ = STR$(M),** quedando almacenados en las variables A$ y B$, respectivamente.

— En la instrucción **50** se concatenan ambas cadenas, poniéndose la segunda a continuación de la primera, y guardando el resultado "4567123" en variable C$.

— La instrucción **60 LET P = VAL (C$)** transforma la cadena numérica almacenada en C$ en un número que se guarda en P.

— Por último, la instrucción **70** imprime el número resultante.

Ejercicios

7.4. ¿Qué imprime en la pantalla el siguiente programa, si cuando se ejecuta se introduce una palabra?

```
10 INPUT A$
20 LET X$ = "***" + A$ + "***"
30 PRINT X$
40 END
```

7.5. Haz un programa que permita introducir dos palabras, las concatene intercalando cinco asteriscos entre ellas e imprima el resultado.

7.6. Diseña un programa que dé las longitudes de las cadenas que se van introduciendo y que detenga su ejecución al introducir una de longitud 1.

7.7. Realiza un programa que permita introducir como cadena el nombre de un abonado y como número, su número de teléfono; transforme éste en cadena, concatene ambas cadenas, y, finalmente, imprima el resultado.

5. Extracción de subcadenas

Dada una cadena, puede interesar extraer unos cuantos caracteres consecutivos, es decir, una ***subcadena;*** por ejemplo, de la cadena CASTELLÓN, sus tres primeras letras: CAS. Esto se puede realizar utilizando una de estas funciones, **LEFT$, RIGHT$, MID$,** según que los caracteres que se quieran extraer estén al principio, al final o en medio de la cadena. Así:

**LEFT$** (A$, N) extrae los N primeros caracteres de la cadena guardada en A$.
**RIGHT$** (A$, N) extrae los N últimos caracteres de la cadena A$.
**MID$** (A$, P, N) extrae en la cadena A$, N caracteres a partir del que está en la posición P.

Por ejemplo, si

A$ = "CASTELLON",
X$ = **LEFT$** (A$, 3) guarda en X$, "CAS"
Y$ = **RIGHT$** (A$, 4) guarda en Y$, "LLON"
Z$ = **MID$** (A$, 4, 5) guarda en Z$, "TELLO"

En el programa que sigue al introducir el nombre de una ciudad A$, se almacena en X$ sus dos primeras letras, en Y$ las tres últimas y en Z$, cuatro, a partir de la tercera letra inclusive (*).

```
 1  REM EXTRACCION DE SUBCADENAS
10  INPUT "CIUDAD"; A$
20  LET X$ = LEFT$ (A$, 2)
30  LET Y$ = RIGHT$ (A$, 3)
40  LET Z$ = MID$ (A$, 3, 4)
50  PRINT X$
60  PRINT Y$
70  PRINT Z$
80  END
```

Al ejecutar el programa (teclear **RUN** y ENTER) aparece en la pantalla:

```
CIUDAD?
```

(*) Si el nombre no tiene suficientes caracteres, al intentar extraer una subcadena se interrumpiría el programa, dando un mensaje de error.

Si introducimos el nombre SANTANDER seguido de ENTER, se obtiene lo siguiente:

```
SA
DER
NTAN
```

En este programa:

— La instrucción **10** pide el nombre de la ciudad.

— Las instrucciones **20, 30** y **40** extraen las subcadenas, almacenando sus valores en X$, Y$ y Z$, respectivamente.

— Las instrucciones **50, 60** y **70** imprimen las subcadenas en tres líneas consecutivas, y la instrucción **80** da por finalizado el programa.

6. Otra forma de extraer subcadenas

Algunos microordenadores no tienen las funciones que se han definido anteriormente para extraer subcadenas. Pero esto no significa que no las puedan extraer, sino que lo hacen de otra forma, por ejemplo, utilizando la palabra **TO**, que actúa de la siguiente manera:

A$ (N TO P)	extrae los caracteres que se encuentran entre la posición N y la posición P, ambas inclusive, de la cadena A$.
A$ (N TO)	extrae desde el carácter de posición N al último carácter, de la cadena A$.
A$ (TO P)	extrae todos los caracteres desde el principio hasta el de posición P, de la cadena A$.

Si, por ejemplo, A$ = "LEVANTE", se tiene que:

X$ = A$ (3 TO 6)	guarda en X$, "VANT"
Y$ = A$ (4 TO)	guarda en Y$, "ANTE"
Z$ = A$ (TO 3)	guarda en Z$, "LEV"

Las equivalencias entre estas dos formas de extraer subcadenas se indican a continuación:

LEFT$** (A$, N) ~ A$ (TO** N)
RIGTH$** (A$, N) ~ A$ (LEN**(A$) – N **TO**)
MID$** (A$, P, N) ~ A$ (P **TO P + N – 1)

Ejercicios

7.8. Haz un programa que forme una subcadena con la primera y última letra de cualquier cadena que se introduzca a través del teclado.

7.9. Elabora un programa que lea una cadena, calcule su longitud L e imprima en la primera fila el primer carácter L veces, en la segunda fila, el segundo carácter L veces, y así sucesivamente.

7. Comparación de cadenas

- En la vida diaria observamos que no sólo los números están ordenados, sino que también lo están las palabras; basta abrir las páginas de un diccionario para ver que las palabras siguen el orden lexicográfico o alfabético.

  Las cadenas también se ajustan a un orden que se ha establecido siguiendo estos criterios:

  — La comparación entre dos cadenas se realiza carácter a carácter, empezando siempre por la ***izquierda.***

  — Cualquier cifra es siempre anterior a cualquier letra.

  — El orden entre letras se ajusta al alfabético.

  Ejemplos:

  "123" < "42" La cadena "123" ***es anterior*** a la cadena "42", pues 1 < 4 (léase 1 ***es anterior a*** 4).
  Observamos que el orden entre cadenas no coincide con el orden entre números. Así:
  123 > 42 (orden entre números)
  "123" < "42" (orden entre cadenas)

  "MARÍA" < "MARTA" La cadena "MARÍA" ***es anterior a*** la cadena "MARTA", pues, aunque las tres primeras letras de ambas cadenas son iguales (M, A y R), la cuarta letra de "MARÍA" (la I) ***es anterior a*** la cuarta de "MARTA" (la T).

  "X32" > "427" La cadena "X32" ***es posterior a*** la cadena "427" por ser X posterior a 4.

Además de los comparadores < ***(es anterior a)*** y > ***(es posterior a)*** se pueden utilizar estos comparadores: =, < =, > = y <>. También se pueden intercalar entre cadenas los operadores lógicos **AND**, **OR** y **NOT**.

- En el siguiente programa aplicamos los conceptos anteriores a la comparación de cadenas.

```
 1  REM ORDENADOR DE CADENAS
10  INPUT A$, B$
20  IF A$ < B$ THEN PRINT A$, B$: GOTO 40
30  PRINT B$, A$
40  END
```

Si al ejecutar el programa (teclear **RUN** y [ENTER]) se introducen como datos "DAVID" y "CARLOS" (seguidos de [ENTER]), en la pantalla aparecerá:

```
CARLOS      DAVID
```

En la instrucción **10** se introducen las dos cadenas A$ y B$.

La instrucción **20** compara las cadenas; si están en orden las imprime y envía la ejecución al final del programa. Como en este ejemplo no están en orden, pues no se cumple que "DAVID" < "CARLOS", se ejecuta la instrucción siguiente (la **30**), que imprime "CARLOS" y después "DAVID".

Ejercicios

7.10. Realiza un programa que lea cadenas y las imprima junto con su longitud hasta que se introduzca la cadena "BASTA".

7.11. Construye un programa que imprima el primer carácter de la cadena que se introduzca si ésta es anterior a "ÚLTIMA", y termine su ejecución cuando se introduzca "FIN".

8. Instrucciones de espera y de borrado de pantalla

- Así como la instrucción **STOP** interrumpe la ejecución del programa un tiempo indefinido (hasta que se teclea **CONT** y [ENTER]), la instrucción **WAIT** (n) o **PAUSE** n, según el microordenador, detiene la ejecución del programa y mantiene la imagen en la pantalla un tiempo limitado, que depende de n. Para conseguir una espera de 1 segundo n debe ser igual a 50, aproximadamente.
- La instrucción que borra la pantalla es **CLS**, aunque suele variar según los microordenadores.
- En el siguiente programa se utilizan ambas instrucciones para dar alternativamente los buenos días y buenas tardes, diez veces consecutivas.

```
10  FOR I = 1 TO 10
20  PRINT "BUENOS DIAS"
30  WAIT (100)
40  CLS
50  PRINT "BUENAS TARDES"
60  WAIT (100)
70  CLS
80  NEXT I
90  END
```

o 30 PAUSE 100

o 60 PAUSE 100

Al ejecutar el programa (teclear **RUN** y [ENTER]) la frase "BUENOS DIAS" permanece en la pantalla 2 segundos aproximadamente. A continuación se borra la pantalla e inmediatamente aparece la frase "BUENAS TARDES", permaneciendo otros 2 segundos. La instrucción **70 CLS** borra otra vez la pantalla.

Este proceso se repite diez veces en total.

9. Un programa para realizar el cambio de moneda

El programa siguiente, que transforma un número de pesetas a dólares, libras o marcos, utiliza la comparación entre cadenas y las instrucciones de espera y borrado de pantalla.

```
 1 REM CAMBIO DE MONEDAS
10 PRINT "INTRODUZCA EL CAMBIO ACTUAL"
20 INPUT "VALOR DEL DOLAR EN PESETAS"; D
30 INPUT "VALOR DE LA LIBRA EN PESETAS"; L
40 INPUT "VALOR DEL MARCO EN PESETAS"; M
50 CLS
60 STOP
```

Las instrucciones **10** a **40** sirven para introducir el valor en pesetas de cada moneda.

La instrucción **50 CLS** borra el contenido de la pantalla.

La instrucción **60 STOP** detiene la ejecución del programa, justo en ese lugar. Escribiendo la palabra **CONT** (y pulsando ENTER) el programa prosigue su ejecución.

```
 70 LET A$ = "DOLARES"
 80 LET B$ = "LIBRAS"
 90 LET C$ = "MARCOS"
100 INPUT "CUANTAS PESETAS VA A CAMBIAR"; P
110 INPUT "A QUE MONEDA QUIERE CAMBIAR"; M$
120 IF M$ = A$ THEN LET C = P/D: GOTO 190
130 IF M$ = B$ THEN LET C = P/L: GOTO 190
140 IF M$ = C$ THEN LET C = P/M: GOTO 190
150 PRINT "NO SE CAMBIA ESTA MONEDA"
160 WAIT (200)                          o 160 PAUSE 200
170 CLS
180 GOTO 100
```

Las instrucciones **70, 80** y **90** asignan las cadenas "DOLARES", "LIBRAS" y "MARCOS" a las variables A$, B$ y C$, respectivamente.

La **100** y la **110** piden la cantidad de pesetas y la moneda a la que se quiere cambiar estas pesetas.

Las instrucciones comprendidas entre la **120** y la **140** comparan M$ con A$, B$ y C$, y como resultado de esta comparación se calcula el número P de pesetas, pasando el control a la instrucción **190** para imprimir el resultado. Si la moneda que se introduce no es ninguna de las tres, la instrucción **150** imprime un informe al respecto. La **160 WAIT (200)** detiene la imagen 4 segundos aproximadamente, la **170 CLS** borra la pantalla y la **180** devuelve el control a la **100**.

```
190  PRINT "LA CANTIDAD DE"; M$; "ES"; C
200  WAIT (500)                          o 200 PAUSE 500
210  CLS
220  INPUT "QUIERE SEGUIR SI/NO"; X$
230  IF X$ = "SI" THEN GOTO 100
240  END
```

La instrucción **200 WAIT (500)** o **200 PAUSE 500** hace permanecer en la pantalla el valor del cambio durante 10 segundos aproximadamente, después de borrar la pantalla **(210 CLS),** y, por último, se pregunta si se quiere seguir (instrucción **220**). En caso afirmativo la instrucción **230** envía el control a la **100.** Si se introduce "NO" se detiene la ejecución.

10. Resumen de las instrucciones y funciones estudiadas en este capítulo

- OPERACIÓN +
  - — ***Significado:*** concatenación o suma.
  - — ***Formato:*** A$ + B$
  - — ***Función:*** la operación + de cadenas forma una nueva cadena con el contenido de A$ seguido del de B$.

- LEN
  - — ***Significado:*** longitud.
  - — ***Formato:*** LEN (A$)
  - — ***Función:*** da el número de caracteres de la cadena almacenada en A$.

- VAL
  - — ***Significado:*** valor numérico.
  - — ***Formato:*** VAL (A$).
  - — ***Función:*** convierte la cadena numérica almacenada en A$ en su valor numérico.

- STR$
  - — ***Significado:*** cadena numérica.
  - — ***Formato:*** STR$ (N)
  - — ***Función:*** convierte el número almacenado en N en una cadena numérica.

- LEFT$
  - — ***Significado:*** subcadena izquierda.
  - — ***Formato:*** LEFT$ (A$, N)
  - — ***Función:*** extrae los N primeros caracteres de la cadena almacenada en A$.

- RIGHT$
  - — ***Significado:*** subcadena derecha.
  - — ***Formato:*** RIGHT$ (A$, N)
  - — ***Función:*** extrae los N últimos caracteres de la cadena almacenada en A$.

- MID$
  - — ***Significado:*** subcadena interior.
  - — ***Formato:*** MID$ (A$, P, N)
  - — ***Función:*** extrae N caracteres de la cadena almacenada en A$ desde la posición P, siguiendo hacia la derecha.

- TO
  - — ***Significado:*** subcadena.
  - — ***Formato:*** A$ (N TO M)
  - — ***Función:*** extrae una subcadena de A$ desde la posición N hasta la posición M.
  - — ***Observación:*** este método de extraer subcadenas se utiliza en algunos microordenadores en sustitución de LEFT$, RIGHT$ y MID$.

- <
  - — ***Significado:*** anterior a.
  - — ***Formato:*** A$ < B$.
  - — ***Función:*** la cadena almacenada en A$ es anterior a la almacenada en B$.

- CLS
  - — ***Significado:*** borra.
  - — ***Formato:*** n CLS.
  - — ***Función:*** borra la pantalla.
  - — ***Observación:*** no todos los microordenadores utilizan CLS para borrar la pantalla (consultar el manual del correspondiente microordenador).

- WAIT
  PAUSE
  - — ***Significado:*** espera, pausa, etc.

— ***Formato:*** WAIT (N) o bien PAUSE N.
— ***Función:*** detiene la ejecución del programa y retiene la imagen de la pantalla un tiempo, que depende de N.
— ***Observaciones:***

1) Utilizar **WAIT** o **PAUSE** según el microordenador (consultar el manual).
2) Algunos microordenadores no poseen esta instrucción. Para conseguir el mismo efecto es necesario escribir un bucle, por ejemplo:

```
30  FOR I = 1 TO N
31  NEXT N
```

dependiendo el tiempo de espera del valor de N.

EJERCICIOS RESUELTOS

1. Hacer un programa que lea una cadena y le añada tantos asteriscos como longitud tenga.

Solución:

```
10  INPUT "CADENA"; A$
20  LET L = LEN (A$)
30  FOR I = 1 TO L
40  LET A$ = A$ + "*"
50  NEXT I
60  PRINT A$
70  END
```

2. Diseñar un programa que lea una cadena y sustituya el primero y el último carácter por asteriscos.

Solución:

```
10  INPUT "CADENA"; A$
20  LET L = LEN (A$)
30  LET A$ = MID$ (A$, 2, L - 2)
40  LET A$ = "*" + A$ + "*"
50  PRINT A$
60  END
```

En los microordenadores que utilicen **TO** para extraer subcadenas, sustituir la instrucción 30 por:

```
30  LET A$ = A$ (2 TO L - 1)
```

EJERCICIOS DE RECAPITULACIÓN

7.1. ¿Qué caracteres imprime el siguiente programa?

```
10 LET A$ = "FERNANDEZ"
20 PRINT LEFT$ (A$, 2)
30 PRINT RIGHT$ (A$, 1)
40 PRINT MID$ (A$, 2, 3)
50 END
```

7.2. Escribe las instrucciones necesarias para imprimir el número de caracteres de la palabra NABUCODONOSOR.

7.3. ¿Qué imprimirá el siguiente programa?

```
10 LET A$ = "PATA"
20 LET B$ = "TA"
30 PRINT A$ + B$
40 END
```

7.4. ¿Qué instrucciones son incorrectas? ¿Por qué?

a) 100 IF X = "FECHA" THEN GOTO 500
b) 200 IF N$ < > A + B THEN PRINT N$
c) 300 IF P$ = "1234" THEN LET S = S + 1
d) 400 IF G$ = "MAYOR" THEN GOTO 600
e) 500 IF A$ = "B" AND X > 3 THEN GOTO 800

7.5. ¿Qué tiene que ocurrir para que se cumpla la condición P$ > Q$?

7.6. Indica si son correctas las siguientes relaciones entre cadenas.

a) "PEDRO" > = "JUAN"
b) "M33" < "M4"
c) "127" > "1235"
d) "4PERRO" < = "37GATO"

7.7. Escribe una o varias instrucciones que expresen la siguiente condición:

Si N$ = "SI" transferir el control a la línea **70,** y si N$ = "NO" transferirlo a la **90.**

7.8. Sea una cadena A$ en la que están almacenados el primer apellido de un alumno, a continuación el segundo apellido y después el nombre, cada uno con diez caracteres (si algunos no se ocupan se dejan con blancos); además está almacenada la edad, que ocupa dos caracteres. Escribe un programa que imprima por separado cada una de estas partes de la cadena.

7.9. Escribe un programa tal que para tres personas cuyos apellidos, nombres y edad se conocen y están incluidos en tres cadenas A$, B$ y C$, imprima en pantalla su edad media (ver el ejercicio anterior y usar la instrucción **VAL** (X$)).

7.10. Haz un programa que introduzca por separado el día, el mes y el año de nacimiento de una persona, forme una única cadena para estos datos e imprima el resultado.

7.11. Escribe un programa que permita introducir una palabra de cinco letras y después imprima sus caracteres de derecha a izquierda.

7.12. Elabora un programa que invierta una palabra con cualquier número de letras.

7.13. Realiza un programa que lea una serie de palabras e imprima solamente las que empiecen por la letra A.

7.14. Diseña un programa que lea una serie de palabras e imprima solamente las que empiecen por la letra A y terminen con la S.

7.15. Dadas las variables:

A$ = "EL"
B$ = "DIA"
C$ = "COMIENZA"
D$ = "AL"
E$ = "AMANECER"

hacer un programa que realice la concatenación de las cadenas, que almacene las variables e imprima el resultado.

7.16. Rectifica el programa anterior para que al imprimir deje un espacio en blanco entre cada dos cadenas.

7.17. Realiza un programa que lea tres cadenas y dé la longitud total de las tres.

7.18. Construye un programa que lea una cadena, calcule su longitud y vaya extrayendo subcadenas desde la izquierda, de longitud 1, de longitud 2, ..., y de longitud máxima.

7.19. Haz un programa que lea una cadena y añade blancos (espacios en blanco) hasta que alcance la longitud 20.

7.20. Realiza un programa que dé el informe "CORRECTO" si la cadena leída tiene longitud inferior a 3 y que imprima "MAL" en caso contrario.

7.21. Diseña un programa que lea cadenas e imprima el primer carácter y el último hasta que se introduzca la cadena "FIN".

7.22. Escribe un programa que compruebe si una cadena contiene la letra A.

7.23. Diseña un programa tal que al introducir un texto calcule el número de veces que aparece la letra E.

7.24. Escribe un programa que cuente el número total de vocales que hay en un texto cualquiera.

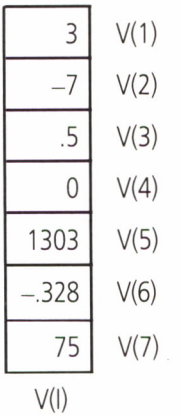
7.25. Haz un programa que lea un texto y dé un informe con el número de veces que han aparecido cada una de las vocales: A, E, I, O y U.

8. Listas numéricas

1. Listas numéricas

- ***Una* lista numérica *es un conjunto de números dispuestos uno a continuación de otro,*** tal como muestra el ejemplo.

  El nombre de esta lista es V(I), donde la letra I, llamada ***índice,*** identifica a cada una de las posiciones de la lista. Concretamente, en el ejemplo representado en la figura, el índice recorre del 1 al 7, identificando V(1) a la primera posición, V(2) a la segunda, y así sucesivamente, hasta llegar a V(7) que identifica a la última posición.



- Los valores de una lista se almacenan en las llamadas ***variables de un índice,*** distinguiéndose así de las ***variables sin índice*** estudiadas anteriormente.

Ejemplos:

— Las variables de nombre A(I), N(X) y Z(R) son ***variables de un índice,*** donde I, X y R son sus correspondientes índices.

— Las variables de nombre P, H y N son ***variables sin índice.***

2. Cómo se almacenan en la memoria los datos de una lista

- *Se desea almacenar en la memoria del microordenador las notas de la asignatura de lengua correspondientes a cuarenta alumnos de una clase.*

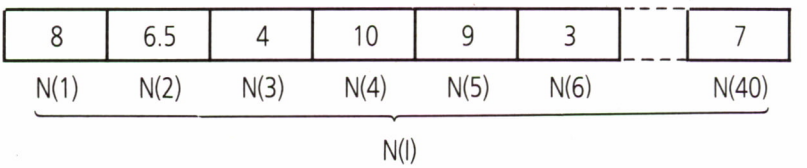
  El conjunto de estas cuarenta notas se almacena en una ***variable con un índice,*** y se introducen en la memoria del microordenador mediante el siguiente programa:

```
10 DIM N (40)
20 FOR I = 1 TO 40
30 INPUT "NOTA:"; N(I)
40 NEXT I
50 END
```

— La instrucción **10 DIM N (40)** sirve para especificar la ***dimensión*** de la lista que se va a introducir en la memoria. Esta instrucción (cuyo nombre proviene de la palabra inglesa DIMENSION), reserva 40 posiciones seguidas de memoria para la ***variable dimensionada*** N(I) (*).

— Con la instrucción **20** se inicializa un bucle de variable I (bucle **20-40**).

— Mediante la instrucción **30 INPUT "NOTA:"; N(I)** se introduce por el teclado las cuarenta notas, que se almacenarán en la variable dimensionada N(I); ésta abarcará entonces 40 posiciones seguidas de memoria.



La ejecución de este programa se interrumpe cuarenta veces, que corresponden a las cuarenta ejecuciones de la instrucción **30 INPUT "NOTA:"; N(I)** incluida en el bucle **20-30.**

Cuando se utilizan listas con un número de datos menor o igual que 10, algunos microordenadores no exigen que se ponga al principio del programa la instrucción **DIM N(I).** Si, en cambio, el número de datos es mayor que 10, el microordenador imprimiría un mensaje de error en el momento de la ejecución.

Se puede comprobar que los datos están en la memoria mediante la instrucción **PRINT**. Si, por ejemplo, tecleamos **PRINT** N(3) y ENTER, inmediatamente aparecerá en la pantalla la nota 4, que corresponde al tercer valor de N(I).

(*) Algunos microordenadores reservan las posiciones de memoria empezando a contar desde cero. En este ejemplo reservarían, en consecuencia, 41 posiciones.

- Ya tenemos almacenados en la memoria las 40 notas en la variable N(I). ¿Cómo haremos para conseguir que aparezcan todas en la pantalla? Para ello, primero borramos la instrucción **50 END** del programa anterior y, a continuación, lo ampliamos con el siguiente conjunto de instrucciones:

```
50 FOR I = 1 TO 40
60 PRINT N(I)
70 NEXT I
80 END
```

Si tecleamos **RUN** y ENTER aparecerán en la pantalla los datos introducidos en la memoria uno debajo de otro, o sea, en columna.

Para que las notas aparecieran en fila habría que añadir un punto y coma a la instrucción **60**; es decir, ésta tendría que escribirse así: **60 PRINT N (I);**

3. Cómo se almacenan en la memoria los datos de una lista con READ y DATA

El programa anterior se puede sustituir por el siguiente, obteniéndose el mismo resultado.

```
  5 DIM N(40)
 10 FOR I = 1 TO 40
 20 READ N(I)
 30 NEXT I
 40 DATA 5, 8, 7, 6, 3, 2, 9, 10, 10, 5
 50 DATA 3, 3, 9, 9, 8, 5, 6, 5, 6, 6
 60 DATA 4, 3, 4, 4, 8, 9, 10, 7, 7, 2
 70 DATA 10, 3, 5, 4, 5, 10, 9, 10, 7, 7
 80 FOR I = 1 TO 40
 90 PRINT N(I)
100 NEXT I
110 END
```

Con el bloque de instrucciones **5-40** se almacenan en la memoria las cuarenta notas en la variable N(I), y con el bloque **80-100** se imprimen en la pantalla. La instrucción **110 END** finaliza el programa. Para ejecutarlo teclear **RUN** y ENTER

Las ventajas de este programa respecto al anterior, son las siguientes:

— En primer lugar, permite más autonomía al usuario, pues ahora puede introducir todas las notas de una sola vez y antes de la ejecución del programa, mediante las instrucciones **20, 40, 50, 60 y 70.**

— En segundo lugar, la ejecución de este programa es mucho más rápida, en cuanto que no se interrumpe ni una sola vez. (En el anterior se interrumpe cuarenta veces.)

Ejercicios

8.1. Indica cuántas posiciones de memoria abarca la variable dimensionada A (K).

```
10 DIM A(18)
20 FOR K = 1 TO 15
30 INPUT A(K)
40 NEXT K
50 END
```

8.2. ¿Qué ocurriría al ejecutar el programa anterior si la instrucción **10 DIM A(18)** se cambiara por **10 DIM A(13)?**

8.3. Haz un programa que produzca los mismos efectos que el transcrito en el ejercicio 8.1., utilizando las instrucciones **READ** y **DATA.**

8.4. Escribe un programa que lea una lista de 12 datos numéricos, es decir, los almacene en la memoria, y los imprima en la pantalla uno a continuación de otro, en fila.

4. Ordenación de una lista de números

- Consideremos la siguiente lista numérica:

4	6	3	1

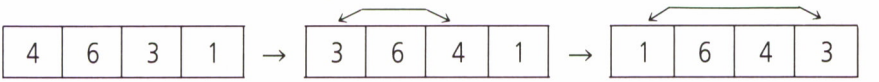
Para ordenar sus valores de menor a mayor, o sea, para disponerlos en forma creciente se pueden utilizar varios métodos; uno de ellos lo esbozamos a continuación:

— Comparamos el primer elemento con cada uno de los siguientes. Si en cada par comparado el primer elemento es menor o igual, los dejamos como están y si es mayor, los intercambiamos. De esta manera se consigue llevar a la primera posición de la lista el número más pequeño.

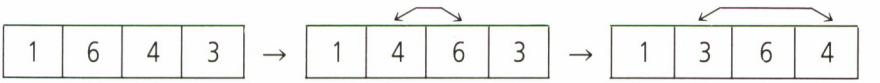
— A continuación se hace lo mismo, comparando el segundo elemento con los siguiente.

— Este proceso se repite hasta llegar al penúltimo elemento de la lista. A título de ejemplo, vamos a ordenar la lista arriba transcrita:

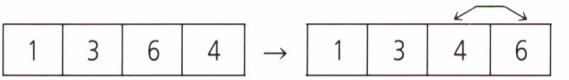
Primera fase: llevar a la primera posición el número menor



Segunda fase: llevar a la segunda posición el menor de los tres siguientes



Tercera fase: llevar a la tercera posición al menor de los dos siguientes



- Si la lista tiene muchos elementos (50, 100, 1 000 ó más), el método aplicado sería muy laborioso, de ahí que sea necesario realizarlo con el microordenador, mediante el siguiente programa:

```
 5  REM ORDENACION DE NUMEROS
10  INPUT "NUMERO:"; N
20  DIM V(N)
30  FOR J = 1 TO N
40  INPUT "DATO:"; V(J)
50  NEXT J
```

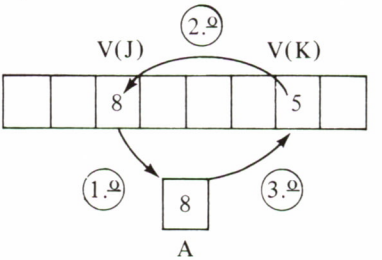
Con la instrucción **20** se introduce el número N que da el tamaño de la tabla, y mediante el bucle **30 – 50** se introducen en la memoria los N datos que componen la lista, utilizando la instrucción **40 INPUT "DATO:"; V(J).**

Una vez almacenados los elementos de la lista en la memoria se procede a su ordenación. Esto es lo que hace el siguiente bloque de instrucciones, siguiendo el método esbozado en el ejemplo anterior, utilizando dos ***bucles anidados,*** el exterior de variable J y el interior de variable K.

```
 60  FOR J = 1 TO N - 1
 70  FOR K = J + 1 TO N
 80  IF V(J) < = V(K) THEN GOTO 120
 90  LET A = V(J)
100  LET V(J) = V(K)
110  LET V(K) = A
120  NEXT K
130  NEXT J
```

Las tres instrucciones **90, 100** y **110** hacen el intercambio de una pareja de datos.

Para visualizarlo, lo esquematizamos así:



El bucle exterior **(60-130)** recorre la lista desde el primer elemento hasta el penúltimo (de 1 a N-1), y el bucle interior **(70-120)** la recorre desde el siguiente elemento a J, hasta el último (de J + 1 a N); por ejemplo, en la «primera vuelta» del bucle exterior (J = 1), al bucle interior recorre la lista desde el segundo elemento (K = J + 1 = 1 + 1 = 2) hasta el último; en la «segunda vuelta» (I = 2), desde el tercero al último, etc. La instrucción **80** es la que hace la comparación de cada pareja de números: si cumplen la condición ***(menor o igual),*** la ejecución salta a la instrucción **120,** no realizándose entonces el intercambio; si el primer elemento del par ***es mayor que*** el segundo, entonces la ejecución sigue con las instrucciones **90, 100** y **110** que son las que realizan el intercambio, tal como se indica en el esquema de arriba.

Una vez ordenados los números que componen la lista, se hacen aparecer en pantalla mediante el bucle siguiente:

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

Observando la instrucción **150 PRINT V(I); " ";** deducimos que los números de la lista se imprimirán en fila en la pantalla.

Ejecución de este programa

Si la lista que deseamos ordenar es

8	0	1	3	3	10	15	–3	6	1.8	–2	4	2	2	2	6	7	30	1

al teclear **RUN** y ENTER aparecerá en la pantalla

NUMERO: ?

Se introduce entonces desde el teclado 20, que corresponde a N.

Inmediatamente aparecerá

DATO: ?

Aquí se introducen sucesivamente los veinte elementos de la lista: 8, 0, ..., 1.

Inmediatamente después de introducir el último, obtendremos en la pantalla los veinte elementos ordenados de menor a mayor.

–3 –2 0 1 1 1.8 2 2 2 3 3 4 6 6 7 8 9 10 15 30

Ejercicios

8.5. Escribe una lista de diez números y ordénalos siguiendo las instrucciones del programa anterior.

8.6. Haz un programa que ordene una lista de números de mayor a menor, o sea, en forma decreciente, y los escriba en la pantalla en forma vertical.

5. La instrucción DIM (resumen)

En este capítulo hemos utilizado esta nueva instrucción cuyas características son las siguientes:

— ***Significado:*** dimensión.
— ***Formato:***
n DIM A (N).
— ***Función:*** reserva posiciones o lugares de la memoria para que en ellos se almacenen los datos de las variables.

EJERCICIOS RESUELTOS

1. Escribir un programa que lea un conjunto de números naturales, almacene en una lista los que sean pares y por último imprima esta lista.

Solución:

```
 10  REM GUARDA EN UNA LISTA LOS NUMEROS QUE SEAN PARES E IM-
     PRIME LA LISTA
 20  READ N
 30  DIM A(N)
 40  LET K = 1
 50  FOR I = 1 TO N
 60  READ X
 70  FOR C = X TO –1 STEP –2
 80  IF C = 0 THEN LET A(K) = X : LET K = K + 1
 90  NEXT C
100  NEXT I
110  FORL = 1 TO K – 1
120  PRINT A(L)
130  NEXT L
140  DATA 11
150  DATA 37, 8, 1, 4, 5, 6, 7, 2, 100, 14, 99
160  END
```

— El bucle **70-90** comprueba si el número leído X es par, en cuyo caso lo guarda en la lista A(K) y corre el índice de la lista un lugar (k = k + 1) para el siguiente número par.

— Para comprobar si X es par, la instrucción **70 FOR C = X TO –1 STEP –2** resta 2 sucesivamente al número X. Si C llega a adquirir el valor 0 significa que el número es par y, en caso contrario, que es impar.

— El tope de la instrucción **110** es K – 1 en lugar de K porque al terminarse de ejecutar por última vez la instrucción **80** la variable K adquiere un valor en una unidad mayor que el número de elementos de la lista A(K).

2. Elaborar un programa que lea tres conjuntos numéricos, sume sus respectivos datos y con las sumas obtenidas forme una lista. A continuación ordenará en forma decreciente esta última lista y la imprimirá en la izquierda de la pantalla sin ordenar y a la derecha, ordenada.

Solución:

```
 10  REM FORMA UNA LISTA DE SUMAS Y LA ORDENA
 20  DIM S(3)
 30  REM LEE LOS TRES CONJUNTOS Y SUMA
 40  FOR I = 1 TO 3
 50  READ N
 60  FOR J = 1 TO N
 70  READ X
 80  LET S(I) = S(I) + X
 90  NEXT J
100  NEXT I
110  DATA 4
120  DATA 5.1, 6.3, – 8, 12.48
130  DATA 4
140  DATA –3, 1.8, – 40, 53
150  DATA 6
160  DATA –1, 0, 5.37, 80, – 4.27, 14
170  REM HACE UNA COPIA DE LA LISTA S(I) Y LA ORDENA
180  DIM C (3)
190  FOR I = 1 TO 3
200  LET C (I) = S(I)
210  NEXT I
220  FOR I = 1 TO 2
230  FOR K = I + 1 TO 3
240  IF C(I) > = C(K) THEN GOTO 280
250  LET D = C(I)
```

```
260  LET C(I) = C(K)
270  LET C(K) = D
280  NEXT K
290  NEXT I
300  CLS : PRINT
310  REM IMPRIME LA LISTA S(I) Y LA C(I)
320  FOR I = 1 TO 3
330  PRINT S(I), C(I)
340  NEXT I
350  END
```

— El bloque de instrucciones **30-100** lee los conjuntos numéricos y suma sus respectivos valores, almacenándolos en la lista S(I).

— El bloque de instrucciones **180-210** forma la lista C(I) cuyos valores coinciden con los de la lista S(I).

— El bloque de instrucciones **220-290** ordena la lista C(I).

— Finalmente, el bloque **300-340** da la lista S(I), no ordenada, a la izquierda de la pantalla, y la lista C(I), ordenada, a la derecha de la pantalla.

EJERCICIOS DE RECAPITULACIÓN

8.1. En la memoria de un microordenador se tienen estas variables con sus correspondientes valores almacenados:

Q	2	A(1)	37
A	3	A(2)	4
A1	1	A(3)	23
		A(4)	19

Escribe el valor que almacenan las siguientes variables: A(A), A(A1), A(A(Q)), A(A(2)).

8.2. ¿Qué aparecerá en pantalla al ejecutar el siguiente programa?

```
 10  READ N
 20  FOR K = 1 TO N
 30  READ X(K)
 40  NEXT K
 50  FOR K = 1 TO N
 60  IF X (K) < 0 THEN GOTO 80
 70  PRINT X (K)
 80  NEXT K
 90  DATA 7
100  DATA 23, – 44, 37, 0, – 14, – 38, 14
200  END
```

8.3. Si se añadiesen al programa anterior las siguientes instrucciones:

110 DATA 8,4, –3,2, 2,4, 3,8, 5,1
120 DATA 5,0, –2,4, 3,9, – 8, 4,3, 8,4

¿qué problemas se presentarían? ¿Cómo se solucionarían?

8.4. Haz un programa que calcule la suma de los 100 primeros elementos de una lista T(I).

8.5. Dado un conjunto de valores, elabora un programa que escriba aquéllos que se encuentren entre dos dados.

8.6. Se tiene una lista de 20 personas entre los 30 y 69 años. Realiza un programa que contabilice el número de personas que se encuentran entre:

30-39
40-49
50-59
60-69
e imprime los resultados.

8.7. Sea una lista de veinte elementos, los cuales pueden ser valores positivos, negativos o nulos. Haz un programa que imprima el total de elementos positivos, el total de negativos y el total de nulos.

9. Tablas numéricas

1. Tablas numéricas

Un conjunto de números dispuestos en filas y en columnas se llama tabla o matriz.

Consideremos las notas de 40 alumnos, de las 7 asignaturas que cursan. Estas notas se pueden registrar en una tabla como la siguiente:

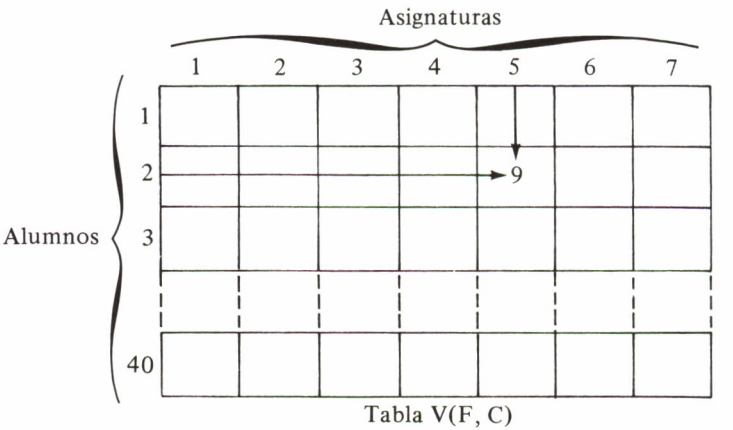


Tabla V(F, C)

En la tabla, cada ***fila*** corresponde a un alumno, y en ella se registran sus notas en las siete asignaturas, y cada ***columna*** corresponde a una asignatura, y en ella se registran las notas de esta asignatura de los cuarenta alumnos.

La posición de cada casilla de la tabla está identificada por un ***par de números,*** el primero corresponde a la fila a que pertenece, y el segundo, a la columna. Así, la posición en la que está registrada la nota 9 se identifica por el par (2, 5); esto significa que el alumno 2 en la asignatura identificada por el número 5 ha obtenido un 9.

El conjunto de las posiciones de esta tabla se denomina ***variable de dos índices.*** El nombre de esta variable es V(F, C), donde F hace referencia a las filas y C a las columnas. En este ejemplo, F puede tomar los valores 1, 2, 3, ..., y 40 y C, los valores 1, 2, 3, 4, 5, 6 y 7; luego, la variable V(F, C) comprende $40 \times 7 = 280$ elementos.

2. Almacenamiento de una tabla en la memoria del microordenador

Los elementos de la tabla anterior se consigue almacenarlos ordenadamente en la memoria, mediante un programa que comprende dos bucles, uno exterior de variable F y otro interior de variable C.

```
10  DIM V(40, 7)
20  FOR F = 1 TO 40
30  FOR C = 1 TO 7
40  INPUT "NOTA"; V (F, C)
50  NEXT C
60  NEXT F
70  END
```

— La instrucción **10 DIM V(40, 7)** especifica la dimensión de la tabla que se va a almacenar en la memoria, y reserva 40 × 7 = 280 posiciones de memoria para la variable de dos índices, de nombre V(F, C).

— Después de tomar el índice F el valor 1 en la instrucción **20 FOR F = 1 TO 40,** en el bucle interior **30 – 50,** el índice C va tomando sucesivamente los valores 1, 2, 3, ..., y 7; en consecuencia, según se va ejecutando este bucle, se almacenan en la memoria las notas en las posiciones V(1,1), V(1,2), V(1,3), ..., y V(1,7), es decir, las correspondientes a una ***fila,*** o en otras palabras, las siete notas del primer alumno.

— Después de dar el bucle "siete vueltas", la ejecución pasa nuevamente a la instrucción **20,** en la que el índice F toma el valor 2; después sigue en el bucle **30 – 60,** mediante el cual introduce en la memoria las notas de la segunda fila, es decir, las del segundo alumno.

— La ejecución continúa hasta que F toma el valor 40, la que permitirá introducir las notas del alumno número 40.

Vemos que este programa almacena las notas ordenadamente en 280 posiciones distribuidas en 40 filas y en 7 columnas, lo cual obliga al bucle interior a dar "siete vueltas" por cada una de las "cuarenta vueltas" del exterior.

- Esta forma de almacenar notas en la memoria tiene el inconveniente de que la ejecución se interrumpe 280 veces, al llegar a la instrucción **40 INPUT "NOTA:"; V(F, C).** Y si suponemos que para introducir una nota empleamos en promedio 2 segundos, para introducir las 280, emplearíamos 280 × 2 = 560 segundos, o sea, 9 min y 20 segundos, lo cual da idea de la lentitud de la ejecución. Si este programa hubiera que ejecutarlo varias veces conviene modificarlo introduciendo instrucciones **READ** y **DATA.**

```
 10  DIM V(40, 7)
 20  FOR F = 1 TO 40
 30  FOR C = 1 TO 7
 40  READ V (F, C)
 50  NEXT C
 60  NEXT F
 70  DATA –, –, –, ..., –
 80  DATA –, –, –, ..., –
 90  DATA –, –, –, ..., –
100  DATA –, –, –, ..., –
.....
150  END
```

Aquí se pondrán las suficientes instrucciones DATA como para que contengan en total 40 × 7 = 280 notas.

El resultado sería similar siempre que se dispongan adecuadamente los datos de las instrucciones **DATA.** Para comprobarlo se obliga al programa a imprimir la tabla, borrando primero la instrucción **150 END** y añadiendo después las siguientes:

```
200  FOR F = 1 TO 40
210  FOR C = 1 TO 7
220  PRINT V(F, C)
230  NEXT C
240  NEXT F
250  END
```

Ejercicios

9.1. Escribe la tabla que imprimiría en la pantalla el siguiente programa

```
  5 REM TABLA
 10 DIM A (2,3)
 20 FOR I = 1 TO 2
 30 FOR J = 1 TO 3
 40 READ A (I, J)
 50 NEXT J
 60 NEXT I
 70 DATA –1, 0, 0, 3, 0.5, 8
 80 FOR I = 1 TO 2
 90 FOR J = 1 TO 3
100 PRINT A (I, J)
110 NEXT I
120 NEXT J
130 END
```

9.2. Haz un programa que almacene en la memoria una tabla de 4 filas y 10 columnas y a continuación la escriba en la pantalla.

3. Programa: promedio de notas por asignatura

Un conjunto de A alumnos ha cursado X asignaturas, y sus notas quedaron registradas en una tabla como la siguiente:

	Asignatura 1	*Asignatura 2*		*Asignatura X*
Alumno 1				
Alumno 2				
Alumno 3		7		
Alumno A				

(El número 7, en la posición en que se encuentra, significa que el alumno 3 en la asignatura 2 ha obtenido un 7.

Se desea obtener el promedio de las notas de todos los alumnos en cada asignatura, es decir, el promedio correspondiente a la ***Asignatura 1,*** a la ***Asignatura 2,*** ..., y a la ***Asignatura X.***

Se trata, entonces, de diseñar un programa que calcule el promedio de las notas situadas en cada una de las ***columnas*** de la tabla.

Para ello primeramente las almacenamos en la memoria y, a continuación, calcularemos el promedio de la ***Asignatura 1*** (1.ª columna), de la ***Asignatura 2*** (2.ª columna), ..., y de la ***Asignatura X*** (X columna).

```
 10  REM PROMEDIO POR ASIGNATURA
 20  READ A
 30  READ X
 40  DIM N(A, X)
 50  REM ALMACENAR EN MEMORIA LAS NOTAS
 60  FOR J = 1 TO X
 70  FOR I = TO A
 80  READ N (I, J)
 90  NEXT I
100  NEXT J
110  REM CALCULO DEL PROMEDIO DE CADA ASIGNATURA
120  FOR J = 1 TO X
130  LET T = 0
140  FOR I = 1 TO A
150  LET T = T + N (I, J)
160  NEXT I
170  PRINT "EL PROMEDIO DE LA ASIGNATURA"; J; "ES"; T/A
180  PRINT
190  NEXT J
200  DATA 5, 3
210  DATA 4, 4, 5, 6, 2, 8, 3, 7, 5, 4, 4, 10, 2, 9, 8
300  END
```

Ejecutando el programa se obtiene el resultado que muestra la pantalla

```
EL PROMEDIO DE LA ASIGNATURA 1 ES 4.2
EL PROMEDIO DE LA ASIGNATURA 2 ES 5.4
EL PROMEDIO DE LA ASIGNATURA 3 ES 6.6
■
```

— Las instrucciones **20 READ A** y **30 READ X** se corresponden con la **200 DATA 5,3;** de modo que la variable A adquiere el valor 5 (número de alumnos) y la X, el valor 3 (número de asignaturas).

— Con los bucles anidados **60-100** y **70-90,** se leen por columnas las notas de los 5 × 3 = 15 alumnos, es decir, se almacenan en memoria. Dentro de estos bucles, la instrucción que lee las notas es la **80 READ N(I, J)** la cual se corresponde con la **210 DATA;** en ésta se encuentran las 15 notas.

— En la instrucción **120** del bucle **120-190** se inicializa la variable J, y mientras contiene el valor 1, la variable I del bucle **140-160,** va adquiriendo los valores del 1 al 5; esto hace que la instrucción **150 LET T = T + N(I, J)** sume las notas de las casillas [1,1] , [2,1] , [3,1] , [4,1] y [5,1] , es decir, las de la ***Asignatura 1.***

— Una vez cumplido el bucle interior **140-160,** la instrucción **170** calcula e imprime el promedio de las notas correspondientes a la **Asignatura 1.**

— Acto seguido, la instrucción **190** devuelve el control a la **120,** adquiriendo J el valor 2. A continuación se repite el proceso para la ***Asginatura 2,*** y en la siguiente vuelta para la ***Asignatura 3.***

Para colectivos de alumnos más numerosos y con mayor número de asignaturas se puede utilizar este mismo programa, cambiando las instrucciones **200 DATA** y **210 DATA,** y añadiendo otras más si fuera necesario.

Ejercicios

9.3. Haz un programa similar al anterior que calcule el promedio de las notas por alumno en todas sus asignaturas.

9.4. Los precios de cinco productos en tres ciudades españolas están registrados en la tabla. Hacer un programa que dé los precios medios de los distintos productos.

	Valencia	*La Palmas*	*Soria*
Naranjas	100	140	120
Plátanos	100	90	130
Fresones	90	150	110
Manzanas	80	100	90
Peras	60	90	90

EJERCICIOS RESUELTOS

1. Hacer un programa que lea e imprima una tabla de 4 filas y 4 columnas, forme una lista con los elementos de la diagonal principal de la tabla y a continuación la imprima.

Solución:

Antes de empezar a escribir el programa hay que tener en cuenta que los elementos de la diagonal principal tienen sus dos índices iguales. Son los siguientes:

A(1,1), A(2,2), A(3,3) y A(4,4).

```
  1 REM FORMA UNA LISTA A PARTIR DE UNA TABLA
  5 CLS : PRINT
 10 DIM A(4,4) : DIM D(4)
 15 PRINT "TABLA"
 20 FOR I = 1 TO 4
 30 FOR J = 1 TO 4
 40 READ A(I,J) : PRINT A(I,J); "   ";
 50 IF I = J THEN LET D(I) = A(I,J)
 60 NEXT J
 70 PRINT : PRINT
 80 NEXT I
100 DATA 1,1,2,2
110 DATA 3,3,4,4
120 DATA 5,5,6,6
130 DATA 7,7,8,8
140 REM IMPRIME LA LISTA
150 PRINT "LISTA DIAGONAL"
160 FOR I = 1 TO 4
170 PRINT D(I); "   ";
180 NEXT I
190 END
```

La instrucción **50** es la que extrae los datos de la diagonal principal de la tabla para formar la lista; estos datos son 1, 3, 6 y 8.

2. La distribución de la población y de los ingresos totales en las tres provincias de Aragón en 1981 fueron las registradas en la tabla siguiente:

	Población	*Ingresos totales*
Huesca	221 761	72 610 000 000
Teruel	173 861	44 423 000 000
Zaragoza	757 543	362 135 000 000

Hacer un programa que introduzca estos datos en forma de tabla y presente en la pantalla los ingresos per cápita (ingresos totales/población) de cada una de las provincias de Aragón y de toda la región.

Solución:

```
  5 REM INGRESOS PER CAPITA DE LAS PROVINCIAS DE ARAGON Y
    DE TODA LA REGION
 10 DIM T(3,2)
 20 FOR I = 1 TO 3
 30 FOR J = 1 TO 2
 40 READ T(I, J)
 50 NEXT J
 60 NEXT I
 70 DATA 221761, 72610000000
 80 DATA 173861, 44423000000
 90 DATA 757453, 362135000000
100 REM CALCULA LOS INGRESOS PER CAPITA Y LOS IMPRIME
110 CLS : PRINT
120 PRINT "INGRESOS PER CAPITA DE HUESCA:"; T(1, 2)/T(1, 1)
130 PRINT "INGRESOS PER CAPITA DE TERUEL:"; T(2, 2)/T(2, 1)
140 PRINT "INGRESOS PER CAPITA DE ZARAGOZA:"; T(3, 2)/T(3, 1)
150 PRINT "INGRESOS PER CAPITA DE ARAGON:"; (T(1, 2) + T(2, 2) +
+ T(3, 2))/(T(1, 1) + T(2, 1) + T(3, 1))
160 END
```

EJERCICIOS DE RECAPITULACIÓN

9.1. Señala las variables dimensionadas de dos índices correctamente escritas.

a) X(2 + 2)
b) X(5,5)
c) X(A + B, C)
d) X(X(1, 2), X(2, 1))
e) X(A, A)

9.2. A la siguiente tabla se asigna el nombre A.

1	2
3	4
5	6

a) ¿Qué clase de variable es A?

b) ¿Qué valores tienen A(1, 1) y A(3, 1)?

c) Si X = 2, e Y = 3, ¿cuál es el valor de A(X, Y), A(X + 1, Y − 1) y A(A(1, 2), A(2, 1) − 1)?

9.3. a) Escribe un programa que almacene ceros en una variable que represente a una tabla de 10 filas y 5 columnas.

b) Lo mismo que en el apartado a), pero en lugar de almacenar ceros, que almacene 1.

9.4. Al hacer una encuesta para anticipar los posibles resultados de unas próximas elecciones se han hecho las siguientes preguntas:

a) ¿A qué candidato votaría?

1. NOMBRE 1
2. NOMBRE 2
3. EN BLANCO

b) ¿En qué grupo de edad está usted?

1. Menos de 30 años
2. 30 años o más

Escribe un programa que recuente los votos conseguidos por cada candidato y los clasifique por edades. Los datos de los votos se introducirán con instrucciones **READ** y **DATA**.

Sugerencia: Se puede definir una variable de dos índices, A(X, Y), representando X el candidato votado e Y la edad del votante. Así, A(1, 1) es un voto conseguido por el candidato 1 emitido por un votante con edad menor de 30 años.

9.5. Dadas las calificaciones de 10 alumnos en 5 asignaturas, escribe un programa que obtenga la nota media de cada alumno.

9.6. Amplía el programa correspondiente al ejercicio anterior, para que dé también la media por asignatura.

9.7. Una red comercial tiene 10 tiendas y cada tienda 4 departamentos. Escribe un programa que lea los valores de las ventas por tienda y departamento e imprima:

1.º El total de ventas por tienda.
2.º El total de ventas por departamento en todas las tiendas.

9.8. Haz un programa que lea una tabla y ponga a cero los elementos que estén repetidos.

9.9. Escribe un programa que lea dos tablas de igual dimensión y forme una tercera sumando los elementos que ocupan la misma posición.

9.10. Diseña un programa que lea una tabla y obtenga el elemento menor de cada fila y el elemento mayor de cada columna.

9.11. Escribe un programa que lea una tabla e imprima aquellos elementos que sean los menores de su fila y a la vez los mayores de su columna.

10. Listas y tablas de cadenas

1. Listas y tablas de cadenas

- Así como las listas y tablas numéricas guardan un conjunto de números, las ***listas y tablas de cadenas*** almacenan un conjunto de ***cadenas.***

Ejemplos:

1203
– 438
– 8
.3
–1.85

Lista numérica

"PITÁGORAS"
"EUCLIDES"
"NEWTON"
"GALILEO GALILEI"
"ARQUÍMEDES"

Lista de cadenas

	Columnas			
F	328	0	–1050	43.214
i	1	0	37843	25
l	14	–.33	–1000	3008
a	4385	27	–.0008	1.001
s	33	– 40	–27	–1

Tabla numérica

	Columnas			
F	"VALLES"	"2328"	"ALBOR"	"DIEZ"
i	"1413"	"GOMEZ"	"– 431"	"URRUTI"
l	"RIVEIRO"	"..."	" "	"280584"
a	"***"	"+300"	"PUIG"	"OLARRA"
s				

Tabla de cadenas

- Analizaremos a continuación un programa que almacena en la memoria una ***lista*** de 40 apellidos.

```
10  DIM A$ (40)
20  FOR = 1 TO 40
30  INPUT A$ (I)
40  NEXT I
50  END
```

Para el *ZX Spectrum,* cambiar por **10 DIM A$ (40, 20)**

— En este programa, la instrucción **10 DIM A$ (40)** reserva en la memoria 40 lugares capaces de almacenar 40 cadenas. (En algunos microordenadores, como el *ZX Spectrum,* hay que especificar la longitud máxima de las cadenas. Así, la instrucción **10 DIM A$ (40, 20)** reserva 40 lugares de memoria, cada uno de ellos capaces de almacenar como máximo 20 caracteres.)

— Con el bucle **20-40** se van cargando en la memoria los cuarenta apellidos en la variable A$ (I).

También aquí se pueden utilizar las instrucciones **READ** y **DATA** como alternativa a la instrucción **INPUT.**

Para que el programa consiga escribir los apellidos en la pantalla habría que borrar la instrucción **50 END** y, a continuación, completarlo con el siguiente bloque de instrucciones:

```
50  FOR I = 1 TO 40
60  PRINT A$ (I)
70  NEXT I
80  END
```

Ejercicios

10.1. Explica qué función cumple el siguiente programa:

```
 10  DIM M$ (12)
 20  FOR I = 1 TO 12
 30  READ M$ (I)
 40  NEXT I
 50  DATA ENERO, FEBRERO, MARZO, ABRIL
 60  DATA MAYO, JUNIO, JULIO, AGOSTO
 70  DATA SEPTIEMBRE, OCTUBRE, NOVIEMBRE, DICIEMBRE
100  END
```

Para el *ZX Spectrum* cambiar por **10 DIM M$ (12, 10)**

10.2. Escribe un programa que almacene en la memoria los días de la semana y los imprima en la pantalla.

2. Ordenación de cadenas

Las cadenas se pueden ordenar mediante un procedimiento similar al utilizado en las listas numéricas. (Repasar el apartado 4 de este mismo capítulo.)

- El siguiente programa ordena alfabéticamente los N apellidos de una lista.

```
  5  REM ORDENADOR DE APELLIDOS
 10  READ N
 15  DIM A$ (N)
 20  FOR J = 1 TO N
 30  READ A$ (J)
 40  NEXT J
 50  DATA 5
 60  DATA "ORTEGA", "ALBENIZ", "PUIG", "URRUTI",
"FERREIRA"
100  FOR J = 1 TO N - 1
110  FOR K = J + 1 TO N
120  IF A$ (J) < A$ (K) THEN GOTO 160
130  LET X$ = A$ (J)
140  LET A$ (J) = A$ (K)
150  LET A$ (K) = X$
160  NEXT K
170  NEXT J
200  FOR I = 1 TO N
210  PRINT A$ (I)
220  NEXT I
230  END
```

Para el *ZX Spectrum* cambiar por **15 DIM A$ (N, 30)**

— Después de leer el valor de N (instrucción **10**), la instrucción **15** reserva N lugares de memoria para los N apellidos.

— El bucle **20-40** almacena en la memoria los apellidos de la lista.

— La instrucción **60 DATA** contiene apellidos.

— Mediante los dos bucles anidados **100-170** y **110-160** se ordenan alfabéticamente los apellidos. Estos bucles funcionan así:

- Cuando J = 1, el bucle interior hace tomar a K los valores 2, ..., N y compara el primer apellido con los siguientes, permutándolos de lugar cuando no se cumpla la condición expresada en la instrucción **120.**
- Cuando J = 2, K adquiere sucesivamente los valores 3, ..., N comparándose entonces el apellido que está en segundo lugar con todos los siguientes.
- Este proceso continúa hasta que J adquiere el valor N – 1; en esta situación, K adquiere el valor N, comparándose entonces el apellido que está en penúltimo lugar con el que se encuentra en el último.

— Finalmente, el bucle **200-220** escribe en la pantalla los N apellidos ordenados alfabéticamente.

3. Elección de asignaturas

En un centro de enseñanza un grupo de alumnos tiene que elegir tres asignaturas entre las seis siguientes:

Historia.

Música.

Inglés.

Francés.

Alemán.

Literatura.

Para concretar ideas, supongamos que el número de alumnos que hace la elección es 10, y el resultado de la misma es el registrado en la tabla.

PREFERENCIA / ALUMNO	1.ª	2.ª	3.ª
1	Inglés	Música	Alemán
2	Literatura	Música	Francés
3	Historia	Inglés	Literatura
4	Alemán	Inglés	Francés
5	Historia	Literatura	Música
6	Francés	Alemán	Inglés
7	Música	Historia	Literatura
8	Francés	Literatura	Inglés
9	Inglés	Francés	Historia
10	Historia	Literatura	Música

Cada casilla de esta tabla contiene el nombre de una asignatura, luego es una tabla de cadenas de caracteres, que representaremos por T$(A,P), donde el primer índice A indica el alumno y el segundo P, la preferencia. Así, el elemento T$(7,2) de la tabla indica que el alumno 7 ha elegido en 2.º lugar de preferencia, historia. (En el *ZX Spectrum* es necesario señalar, además, el número máximo de caracteres de las cadenas; como ***literatura*** es la palabra más larga y tiene 10 caracteres, la tabla se tendría que indicar así: T$(A,P, 10)).

Después de registrar las asignaturas elegidas en la tabla, lo que se pretende es averiguar cuántos alumnos han elegido cada una de las seis asignaturas, sin tener en cuenta el orden de preferencia de cada elección individual.

El siguiente programa resuelve el problema planteado.

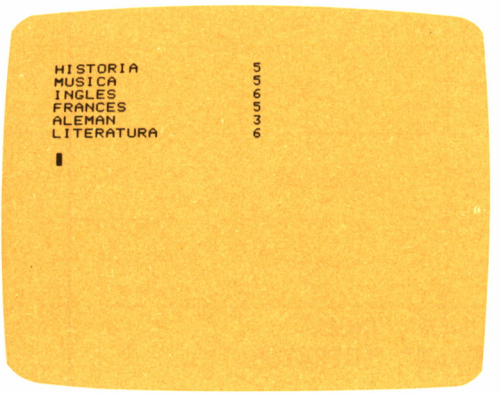
```
 10  DIM T$(10,3)
 20  DIM X$(6)
 30  DIM C(6)
 40  FOR A = 1 TO 10
 50  FOR P = 1 TO 3
 60  INPUT T$(A,P)
 70  NEXT P
 80  NEXT A
 90  FOR I = 1 TO 6
100  READ X$(I)
110  NEXT I
120  DATA "HISTORIA", "MUSICA", "INGLES"
130  DATA "FRANCES", "ALEMAN", "LITERATURA"
140  FOR A = 1 TO 10
150  FOR P = 1 TO 3
160  FOR I = 1 TO 6
170  IF T$(A,P) = X$(I) THEN LET C(I) = C(I) + 1
180  NEXT I
190  NEXT P
200  NEXT A
210  FOR I = 1 TO 6
220  PRINT X$(I), C(I)
230  NEXT I
240  END
```

En el *ZX Spectrum* hay que sustituir las instrucciones **10** y **20** por

```
10  DIM T$(10, 3, 10)
20  DIM X$(6, 10)
```

para especificar el número máximo de caracteres de las cadenas.

Ejecutando este programa se obtiene en la pantalla el siguiente resultado:



Para comprender bien el programa centraremos la atención en los distintos bloques de instrucciones que contiene:

— Las instrucciones **10, 20** y **30** dimensionan una tabla, una lista de caracteres y una lista numérica.

— Los dos bucles anidados, comprendidos entre las instrucciones **40** y **80** introducen en la memoria la tabla T$(A,P).

— El bucle **90-110** asigna a la lista X$(I) los nombres de las asignaturas; éstos se encuentran en las instrucciones **DATA, 120** y **130.**

— Los bucles anidados, comprendidos entre las líneas **140** y **200** cuentan el número de alumnos que ha elegido cada asignatura, por medio de los contadores C(I).

— Por último, el bucle **210-230** imprime el nombre de cada asignatura X$(I) y el número C(I) de alumnos que la eligieron.

Este programa se puede generalizar para N alumnos, añadiendo la instrucción

```
5 INPUT "NUMERO DE ALUMNOS"; N
```

y cambiando las instrucciones **10, 40** y **140** por:

```
 10  DIM T$(N, 3)
 40  FOR A = 1 TO N
140  FOR A = 1 TO N
```

EJERCICIOS RESUELTOS

1. Hacer un programa que lea los apellidos de doce personas y su estatura, y presente estos datos en la pantalla según el siguiente formato:

APELLIDO	ESTATURA (M)
CAJAL	1.75
.	.
.	.
.	.
.	.

Solución:

```
 5  REM APELLIDO, ESTATURA
 8  CLS
10  DIM A$ (12): DIM E (12)
20  PRINT "APELLIDO", "ESTATURA (M)"
30  PRINT " ....."
40  FOR I = 1 TO 12
50  READ A$ (I)
60  READ E (I)
70  PRINT A$ (I), E (I)
80  NEXT I
90  DATA "CAJAL", 1.75, "RUIZ", 1.62, "ORTEGA", 1.80, "CALLEJA",
1.65, "SANCHEZ", 1.75, "COLON", 1.55.
100 DATA "PEREZ", 1.88, "NIETO", 1.59, "AMOROS", 1.72, "HI-
DALGO", 1.80, "JUAREZ", 1.81, "PORTILLO", 1.73.
110 END
```

2. Elaborar un programa que lea un conjunto de palabras, las ordene alfabéticamente y las imprima en la pantalla de la siguiente forma:

```
PALABRAS ORDENADAS ALFABETICAMENTE
1. ARMAS
2. BELLEZA
3. ESTRELLA
   .
   .
   .
```

Solución:

```
  1  CLS: PRINT
  5  REM ORDENA ALFABETICAMENTE PALABRAS
 10  READ N
 20  DIM P$ (N)
 30  FOR J = 1 TO N
 40  READ P$ (J)
 50  NEXT J
 60  DATA 7
 70  DATA "ESTRELLA", "ARMAS", "BELLEZA", "MESA", "YEMA",
"CORTEZA", "CORAL"
100  PRINT "PALABRAS ORDENADAS ALFABETICAMENTE"
110  PRINT "....."
115  PRINT
120  FOR J = 1 TO N - 1
130  FOR K = J + 1 TO N
140  IF P$ (J) < P$ (K) THEN GOTO 180
150  LET X$ = P$ (J)
160  LET P$ (J) = P$ (K)
170  LET P$ (K) = X$
180  NEXT K
190  NEXT J
200  FOR J = 1 TO N
210  PRINT "          "; J; ". "; P$ (J)
220  NEXT J
230  END
```

EJERCICIOS DE REC.\PITULACIÓN

10.1. Escribe un programa que almacene en una variable A$ (I) los meses del año.

10.2. Escribe un programa que almacene en una variable A$ (I) los días de la semana.

10.3. Haz un programa que lea varios nombres, forme una lista con todos los que empiezan por A y la imprima.

10.4. Escribe un programa que lea una lista de nombres y elimine los que están duplicados.

10.5. Diseñar un programa que lea dos listas de nombres y forme una tercera con los nombres comunes a ambas.

10.6. Las siete empresas españolas que registraron mayores ventas en 1982 fueron las siguientes:

Empresa	*Ventas (millones de pesetas)*	*Beneficios (millones de pesetas)*	*Empleados*
Campsa	1 094 685	4 890	9 319
Empetrol	522 615	2 734	5 949
Cepsa	381 809	2 136	4 520
Telefónica	266 392	26 546	65 629
Fasa Renault	184 432	13 877	21 810
Iberia	183 347	– 8 124	24 825
Petronor	182 616	2 070	581

Escribe un programa que almacene estos datos en una tabla de caracteres, transforme las ventas, beneficios y número de empleados a valores numéricos y los almacene en una tabla numérica. Por último, deberá imprimir dos columnas, la primera con la relación ventas por empleado (V/E) y la segunda con la relación beneficios por empleado (B/E).

10.7. Basándote en el ejercicio anterior diseña un programa que determine en qué empresa la relación ventas por empleado (V/E) es mayor y también la empresa en la que la relación beneficios por empleado es mayor (B/E).

11. Funciones

1. Algunas funciones del lenguaje Basic

Además de sumar, restar, multiplicar, dividir, calcular potencias y extraer la raíz cuadrada de un número, el microordenador puede realizar otros cálculos utilizando ciertas ***funciones,*** algunas de las cuales estudiaremos en este capítulo.

- Las instrucciones del siguiente programa contienen varias funciones.

```
10  INPUT "INTRODUCE X"; X
20  PRINT "X", X
30  PRINT "ABS(X)", ABS(X)
40  PRINT "INT(X)", INT(X)
50  PRINT "SGN(X)", SGN(X)
60  END
```

Si ejecutamos este programa (tecleamos **RUN** y ENTER), después de introducir el valor –7.25 para X, en la pantalla aparecen los siguientes resultados:

```
X          –7.25
ABS(X)      7.25
INT(X)     –8
SGN(X)     –1
```

Corresponde a la instrucción **20**
Corresponde a la instrucción **30**
Corresponde a la instrucción **40**
Corresponde a la instrucción **50**

— La instrucción **20** imprime en la pantalla el número introducido (X = −7.25).
— La instrucción **30** imprime el ***valor absoluto X*** (**ABS** (–7.25) = 7.25).
— La instrucción **40** imprime la ***parte entera de X*** (**INT**(−7.25) = −8).
— La instrucción **50** imprime el ***signo de X*** (**SGN**(−7.25) = −1).

- ***Función valor absoluto.***

Esta función se escribe **ABS(X),** siendo X su *argumento.* Calcula el valor absoluto de X, es decir, del número o del valor de la expresión que está dentro del paréntesis.

Ejemplos:

1) **ABS** (5) = 5; **ABS** (–0.845) = 0.845; **ABS** (0) = 0

2)

```
10  LET X = – 4
20  ABS (2*X/1000)
30  END
```

Al ejecutar este programa, la instrucción 20 calcula primero el valor del argumento y, a continuación, su valor absoluto:

ABS (2 * (– 4)/1000) = **ABS** (– 8/1000) = **ABS** (– 0.008) = 0.008

- ***Función parte entera***

Esta función se escribe **INT (X).** Calcula el valor entero más próximo a X, y menor o igual que X.

Ejemplos:

INT (5.8) = 5 **INT** (–10.2) = –11 **INT** (7) = 7

- ***Función signo***

Esta función se escribe **SGN (X),** y da los siguientes valores:

\+ 1 si X es positivo
0 si X es nulo
– 1 si X es negativo

Ejemplos:

SGN (–50.23) = –1 SGN (0) = 0 SGN (0.0028) =1

2. Dos aplicaciones de la función INT (X)

- ***Cambio de unidades***

El siguiente programa transforma una cantidad expresada en metros en otra equivalente expresada en kilómetros y metros.

```
 5  REM PASA DE METROS A KILOMETROS
10  INPUT "INTRODUCE LA CANTIDAD EN METROS"; A
20  PRINT A; "METROS"
30  LET B = INT (A/1000)
40  LET C = A – B * 1000
50  PRINT B: "KM"; C; "M"
60  END
```

Si al ejecutar el programa introducimos el valor 13728 para A, en la pantalla aparecen los siguientes resultados:

```
13728 METROS
13 KM 728 M
```

— La instrucción **30** almacena en B el cociente entero de la división A/1000. Para 13728, se tiene:

$B = INT (13728/1000) = INT (13.728) = 13$

— La instrucción **40** almacena en C el resto de la división A/1000. Recordemos que $D = d \cdot c + r$, de donde $r = D - d \cdot c$. En consecuencia, la instrucción **40** produce el siguiente resultado:

$C = 13728 - 13 * 1000 = 13728 - 13000 = 728$

— Finalmente, la instrucción **50** imprime en la pantalla

13 KM 728 M

- ***Redondeo de números***

— Redondear un número que contiene varias cifras decimales significa transformarle en otro con menor cantidad de cifras decimales y cuyo valor sea el más próximo posible al número dado.

Ejemplos:

A) Al redondear el número 32.283654 a dos cifras decimales se transforma en 32.28.

B) Al redondear el número 0.2873222 a dos cifras decimales se transforma en 0.29.

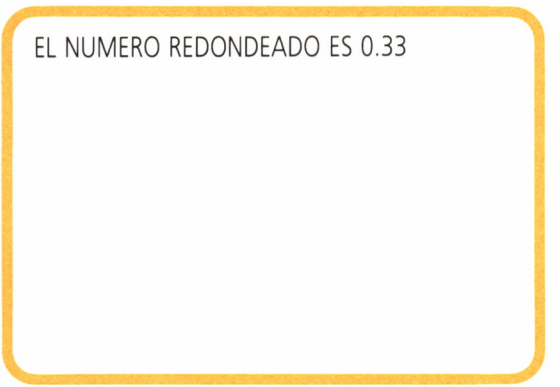
— Para redondear números decimales se puede utilizar este programa:

```
 5  REM REDONDEO A DOS CIFRAS DECIMALES
10  INPUT "INTRODUCE UN NUMERO DECIMAL"; A
20  LET A1 = INT (A * 100 + 0.5)/100
30  PRINT "EL NUMERO REDONDEADO ES"; A1
40  PRINT
50  GOTO 10
60  END
```

— Al ejecutar el programa (teclear **RUN** y ENTER), si introducimos 0.33333 para A, la instrucción **20** efectúa los siguientes cálculos:

A1 = **INT** (0.33333 * 100 + 0.5)/100 = **INT** (33.333 + 0.5)/100 =
= **INT** (33.833)/100 = 33/100 = 0.33

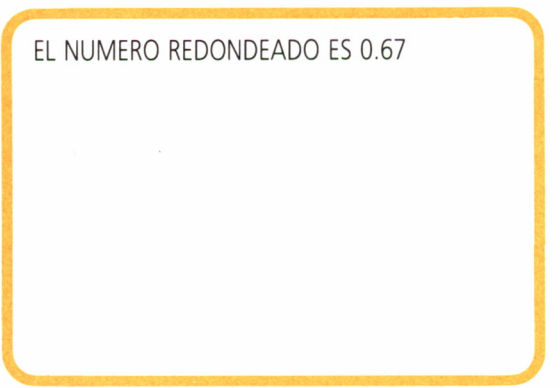
Después de realizar estos cálculos, la instrucción **30** imprimirá en la pantalla lo siguiente:



— Como la instrucción **50** devuelve el control a la **10,** podemos introducir otro número, por ejemplo, 0.6666. La instrucción **20** realizará entonces los siguientes cálculos:

A1 = **INT** (0.6666 * 100 + 0.5)/100 = **INT** (66.66 + 0.5)/100 =
= **INT** (67.36)/100 = 67/100 = 0.67

En la pantalla aparecerá:



Ejercicios

11.1. Haz un programa que exprese una cantidad de segundos, en horas, minutos y segundos.

11.2. Haz un programa que exprese en grados, minutos y segundos la amplitud de un ángulo dado en segundos.

11.3. Indica qué escribirá en la pantalla el siguiente programa, si en el momento de su ejecución se introduce 0.451966.

```
10  INPUT "INTRODUCE X"; X
20  LET Y = INT (X * 1000 + 0.5)/1000
30  PRINT Y
30  END
```

11.4. Realiza los cálculos indicados en las siguientes expresiones para X = –3.1482.

a) INT (X * 10 + 0.5)/10

b) INT (X * 100 + 0.5)/100

c) INT (X * 1000 + 0.5)/1000

11.5. Escribe los programas que permitan redondear:

a) A una cifra decimal.

b) A dos cifras decimales.

c) A n cifras decimales.

3. La función azar

La atracción que suelen ejercer los *juegos de azar* se debe en parte a que nunca se tiene la seguridad de ganar o perder, cuando se participa en ellos. Esto es lo que ocurre, por ejemplo, en el juego del dado.

¿Cuándo se dice que un juego es de *azar o aleatorio*?

Un fenómeno, proceso o experimento se dice que es *aleatorio* si antes de producirse no se puede predecir su resultado. Por ejemplo, el juego del lanzamiento de un dado es un experimento aleatorio, porque antes de lanzarlo no se puede afirmar si se obtendrá un punto, dos puntos, ..., o seis puntos. También son procesos aleatorios el lanzamiento de una o varias monedas, los juegos que se realizan con cartas, el juego de la ruleta, etc.

Cada vez que se lanza un dado se obtiene un número que se llama aleatorio. De forma similar, cada vez que se detiene la ruleta, marca un número, que también se denomina aleatorio.

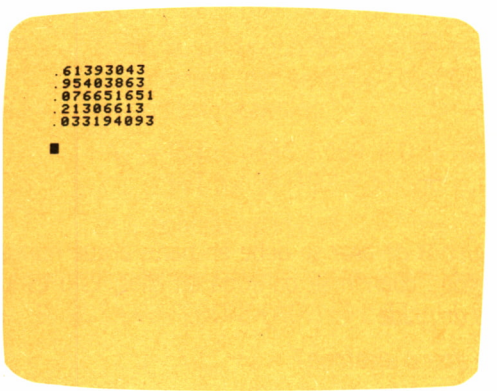
El microordenador tiene también la capacidad de generar números aleatorios o al azar. Esto lo consigue ejecutando la ***función azar,*** que en BASIC se escribe **RND (A),** nombre que deriva de la palabra inglesa RANDOM (azar). La letra A es un número determinado, frecuentemente 1.

El siguiente programa genera cinco números aleatorios comprendidos entre 0 y 1, excluido este último.

```
 5  REM NUMEROS ALEATORIOS
10  FOR I = 1 TO 5
20  LET X = RND (1)
30  PRINT X
40  NEXT I
50  END
```

(*)

Después de teclear **RUN** y [ENTER] obtuvimos los números que se transcriben a continuación:



Conviene advertir que si se ejecuta por segunda vez este programa es muy probable que no se obtengan los mismos números, justamente porque el microordenador genera números aleatorios. Pasa lo mismo que con el juego de la ruleta: lo normal es que en jugadas sucesivas no aparezcan los mismos resultados.

(*) Algunos microordenadores, para conseguir un azar de las características del que se consigue con **RND (A)** necesitan dos instrucciones: **RANDOMIZE** y **RND**, ésta sin argumento. **RANDOMIZE** debe estar al principio del programa y sólo una vez; en cambio, **RND** se puede utilizar todas las veces que sea necesario, en lugar de **RND (A)**.

```
 5  REM NUMEROS ALEATORIOS
 8  RANDOMIZE
10  FOR I = 1 TO 5
20  LET X = RND
30  PRINT X
40  NEXT I
50  END
```

4. Simulación de procesos aleatorios

Con la instrucción **RND (A)** se pueden hacer programas que simulen fenómenos, experimentos y, en general, procesos aleatorios, como por ejemplo el lanzamiento de una moneda, el juego de los dados, etc.

- ***Jugar a «cara y cruz»***

  En el lanzamiento de una moneda caben dos resultados posibles: CARA, al que asignaremos el número 1, y CRUZ, al que asignaremos el número 0. Estos dos resultados se pueden simular a partir de la función RND (1), de la siguiente manera:

  RND (1) da un número aleatorio entre 0 y 1, excluido éste.
  RND (1) + 0,5 da un número aleatorio entre 0.5 y 1.5.
  INT (**RND** (1) + 0.5) genera cada vez que se ejecuta dos valores aleatorios: el 0 o el 1.

  Por ejemplo, si el número aleatorio generado por **RND** (1) fuera 0.342701, se tendría:

  $$\text{INT}\,(\overbrace{0.342701}^{\text{RND}\,(1)} + 0.5) = \text{INT}\,(0.8342701) = 0$$

  Y si el número aleatorio dado por RND (1) fuera 0.732211, se tendría:

  $$\text{INT}\,(\overbrace{0.732211}^{\text{RND}\,(1)} + 0.5) = \text{INT}\,(1.232211) = 1$$

  Vemos, pues, que los dos números aleatorios posibles que puede generar la función **INT (RND** (1) + 0.5) son 0 y 1. La aplicaremos, entonces, para jugar a ***cara y cruz.***

```
 5  REM JUGAR A CARA Y CRUZ
10  PRINT
20  PRINT "LANZA OTRA VEZ LA MONEDA; PARA ELLO TECLEA
CONT Y PULSA ENTER"
30  STOP
40  LET R = INT (RND (1) + 0.5)
50  IF R = 1 THEN PRINT: PRINT "CARA": GOTO 10
60  PRINT
70  PRINT "CRUZ"
80  GOTO 10
90  END
```

El funcionamiento de este programa es el siguiente:

— La instrucción **20** escribe la frase escrita entre comillas, y, a continuación, la instrucción **30 STOP** detiene la ejecución del programa. Para proseguir con la ejecución hay que teclear **CONT** y ENTER, tal como indica la frase.

— Se ejecuta entonces la instrucción **40** que genera aleatoriamente el 0 o el 1.

— La instrucción **50** escribe en la pantalla CARA si R = 1 y transfiere el control a la instrucción 10, ejecutándose otra vez el programa.

— Si R $\neq$ 1 la ejecución salta a la instrucción **60** y después a la **70,** que imprime CRUZ.

— La instrucción **80** transfiere el control a **10,** ejecutándose otra vez el proceso.

Vemos que cada ejecución equivale al lanzamiento de una moneda, de ahí que este programa permita jugar a «cara y cruz». (En lugar de lanzar al aire una moneda, se teclea **CONT** y ENTER.)

- ***Simulación del lanzamiento de un dado***

Cuando se lanza un dado 60 veces, por ejemplo, una cualquiera de sus caras puede salir cero, una, dos, ..., e incluso las sesenta veces. Sin embargo, las experiencias realizadas en multitud de ocasiones y por distintos experimentadores confirman que prácticamente nunca sale cero veces, y tampoco sesenta. Lo que se ha comprobado es que sale un número de veces próximo a 10.

Si en lugar de 60, se hicieran 600 lanzamientos, lo que se observaría es que cada cara saldría un número de veces próximo a 100. Y si se tuviera paciencia para repetir el lanzamiento del dado 6 000 veces, se comprobaría que cada cara saldría un número de veces próximo a 1 000.

Para comprobar estos hechos, en lugar del dado podemos utilizar el microordenador, el cual, mediante un programa muy sencillo, se comporta como un dado, pero con la ventaja de que los «lanzamientos» y cómputos los realiza con mucha rapidez (*).

La simulación del lanzamiento del dado la lleva a cabo el microordenador mediante el siguiente programa:

(*) Si suponemos que en lanzar una vez el dado y en registrar el resultado se tardan 6 segundos, en 6 000 lanzamientos se tardarían 36 000 segundos, o sea, 10 horas. Un microordenador, en cambio, lo puede realizar en menos de 5 minutos.

```
 1  REM SIMULACION DEL LANZAMIENTO DE UN DADO
 5  LET U = 0 : LET D = 0 : LET T = 0 : LET C = 0 : LET CI = 0 :
LET S = 0
10  INPUT "INTRODUCE N (NUMERO DE LANZAMIENTOS)"; N
20  FOR I = 1 TO N
30  LET X = INT (RND (1) * 6 + 1)
40  IF X = 1 THEN LET U = U + 1
50  IF X = 2 THEN LET D = D + 1
60  IF X = 3 THEN LET T = T + 1
70  IF X = 4 THEN LET C = C + 1
80  IF X = 5 THEN LET CI = CI + 1
90  IF X = 6 THEN LET S = S + 1
100  NEXT I
110  CLS
120  PRINT : PRINT "EN"; N; "LANZAMIENTOS HA SALIDO:"
130  PRINT : PRINT "*"; U; "VECES EL 1"
140  PRINT : PRINT "*"; D; "VECES EL 2"
150  PRINT : PRINT "*"; T; "VECES EL 3"
160  PRINT : PRINT "*"; C; "VECES EL 4"
170  PRINT : PRINT "*"; CI; "VECES EL 5"
180  PRINT : PRINT "*"; S; "VECES EL 6"
190  END
```

— La instrucción que permite al microordenador hacer la simulación es **30 LET X = INT (RND (1) * 6 + 1)**

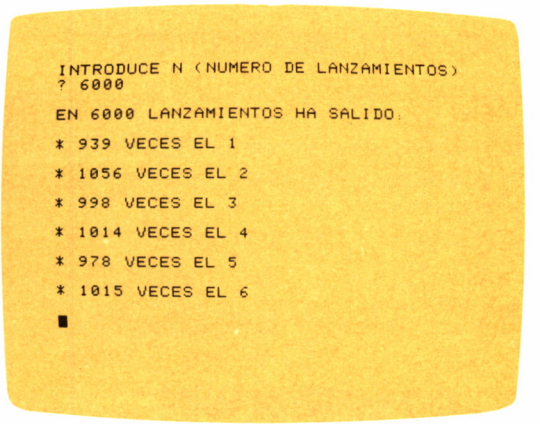
Esta instrucción genera números al azar entre 1 y 6, ambos inclusive. Efectivamente:

RND (1) genera un número aleatorio entre 0 y 1, excluido este último.
RND (1) * 6 genera un número aleatorio entre 0 y 6, excluido este último.
RND (1) * 6 + 1 genera un número aleatorio entre 1 y 7, excluido este último.
INT (**RND** (1) * 6 + 1) toma la parte entera de los números generados entre 1 y 7, excluido éste; es decir, genera los números naturales 1, 2, 3, 4, 5 y 6, que corresponden a las caras del dado.

— Las instrucciones **40** a **90** contabilizan las veces que sale el 1, 2, 3, 4, 5 y 6, respectivamente.

— Las instrucciones **130** a **180** escriben en la pantalla las veces que han salido cada una de las caras.

Hemos ejecutado este programa y en el momento de la ejecución introducimos el valor 6 000 para N. Los resultados obtenidos aparecen en la fotografía.



Observar la proximidad de cada uno de los resultados al número 1 000. Conviene hacer presente que al ejecutar otra vez este programa los resultados no coincidirán exactamente con los que aparecen en la fotografía, debido a que la función **RND** (1) genera números aleatorios.

Ejercicios

8.8. Haz un programa, similar al anterior, que simule el lanzamiento de una moneda.

8.9. Indica qué números al azar generan las siguientes instrucciones:

a) X = **INT** (**RND** (1) * 9)
b) X = **INT** (**RND** (1) * 8 + 1)
c) X = **INT** (**RND** (1) * 10 + 11)

8.10. Escribe un programa que genere al azar los números 10, 11, 12, 13 y 14.

5. Otras funciones matemáticas

El lenguaje BASIC posee un conjunto de funciones matemáticas, llamadas de ***librería,*** cuya definición y características se estudian en distintos cursos de matemáticas.

Transcribimos a continuación el nombre, la notación algebraica usual, la notación en BASIC y el valor de dichas funciones.

Nombre	*Notación algebraica*	*Notación en BASIC*	*Valor*
Signo	sgn x	SGN (X)	+ 1 si X > 0 0 si X = 0 − 1 si X < 0
Valor absoluto	|x|	ABS (X)	X si X ⩾ 0 − X si X < 0
Raíz cuadrada	$\sqrt{x}$	SQR (X)	X si X ⩾ 0 error si X < 0
Parte entera	E (x)	INT (X)	parte entera de X
Logarítmo natural o neperiano	Ln x	LN (X)	Ln X si X > 0 error si X ⩽ 0 error si X < 0
Logaritmo decimal	Log x	LOG (X)	log X si X > 0 error si X ⩽ 0
Exponencial	e^x	EXP (X)	potencias del número e, de exponente X (e = 2,71828)
Seno	sen x	SIN (X)	seno de X (X, ángulo de radianes)
Coseno	cos x	COS (X)	coseno de X (X, ángulo de radianes)
Tangente	tg x	TAG (X)	tangente de X (X, ángulo de radianes)
Arco tangente	arc tg x	ATN (X)	arco cuya tangente es X
Función azar		RDN (A) (RANDOMIZE y RND)	un número aleatorio entre 0 y 1, excluido este último

6. Definición de funciones

- Además de las funciones anteriores, el usuario tiene la posibilidad de definir otras funciones mediante la instrucción **DEF FNA (X)**. Así, en el siguiente pro-

grama se define una función polinómica y se calcula una tabla de valores referida a la misma.

```
10  DEF FNA (X) = 3 * X ↑ 5 − X ↑ 2 + 10
20  FOR X = −2 TO 3
30  PRINT X, FNA (X)
40  NEXT X
50  END
```

Al ejecutar el programa (teclear **RUN** y ENTER) se obtiene el siguiente resultado:

```
-2            -90
-1             6
 0             10
 1             12
 2             38
 3             730
■
```

La instrucción 10 define la función $f(x) = 3x^5 - x^2 + 10$, que en BASIC se expresa así: **FNA (X)** = 3 * X ↑ 5 − X ↑ 2 + 10.

Una vez definida **FNA (X)**, los valores que va adquiriendo esta función corresponden a los que toma el segundo miembro para los distintos valores de X.

La expresión que se encuentra a la derecha del signo igual puede contener cualquier fórmula, inclusive las funciones **SIN (X)**, **COS (X)**, **LOG (X)**, etc.

- Cada **DEF** define una sola función, de ahí que en un mismo programa se puedan definir varias funciones, repitiendo la palabra **DEF** y haciendo variar la tercera letra de **FN...** desde A hasta Z. Así, el programa siguiente define tres funciones y calcula una tabla de valores para X = 1, 2, 3 y 4.

```
10  REM DEFINICION DE FUNCIONES
20  DEF FNA (X) = (1 - X)/(2 + X)
30  DEF FNB (X) = 2 ↑ X
40  DEF FNC (X) = -1/X
50  PRINT "X", "FNA (X)", "FNB (X)",
"FNC (X)"
55  PRINT
60  FOR X = 1 TO 4
70  PRINT X, FNA (X), FNB (X), FNC (X)
80  NEXT X
90  END
```

Si se ejecuta este programa (teclear **RUN** y ENTER), se obtiene el siguiente resultado (en una pantalla de cuatro zonas):

X	FNA (X)	FNB (X)	FNC (X)
1	0	2	-1
2	-.25	4	-.5
3	-.4	8	-.333333
4	-.5	16	-.25

Las instrucciones **20**, **30** y **40** definen tres funciones diferentes, y la instrucción **70** imprime sus valores para X = 1, 2, 3 y 4.

- Algunos microordenadores pueden definir funciones de varias variables, como la siguiente

  DEF FNA (X, Y, Z) = X + Y + Z

  Esta función es de tres variables, y da el valor de la suma de tres números cualesquiera X, Y y Z.

- De la misma forma que se definen funciones numéricas, también se pueden definir funciones de cadenas, desde **FNA$ (X$)** hasta **FNZ$ (X$)**.

  El siguiente programa extrae el primero y el último carácter de una cadena y los concatena.

```
 1  REM CONCATENACION DE INICIAL
Y FINAL
10  DEF FNA$ (X$) = LEFT $ (X$, 1) +
RIGHT$ (X$, 1)
20  INPUT "INTRODUCE LA PALABRA"; X$
30  PRINT X$, FNA$ (X$)
40  GOTO 20
50  END
```

En la instrucción **10** se define la función **FNA$ (X$)** que concatena el primero y el último carácter de X$, la cual se introduce por teclado en la instrucción **20.**

La instrucción **30** imprime la cadena y el resultado de la función.

- Por último, conviene tener en cuenta que en algunos microordenadores es necesario definir **DEF FNA (X)** antes de que sea utilizada **FNA (X),** por eso interesa definir las funciones al principio del programa.

Ejercicios

11.11. Indica qué función define el siguiente programa y qué imprime en la pantalla.

```
10  DEF FNR (X) = X ↑ 4
20  FOR X = 1 TO 10
30  PRINT X, FNR (X)
40  NEXT X
50  END
```

11.12. Escribe un programa que defina una función para calcular los cubos de los veinte primeros números naturales, y después los imprima.

11.13. Haz un programa que defina la función polinómica $3x^2 + 2x + 1$ e imprima una tabla para valores de X comprendidos entre –2 y 2, ambos inclusive, incrementando X en 0.1.

11.14. Explica qué hace el siguiente programa:

```
10  DEF FNA (X) = X ↑ 2
20  DEF FNB (X) = X ↑ 3
30  DEF FNS (X) = X ↑ 2 + X ↑ 3
40  FOR X = 1 TO 10
50  PRINT X, FNS (X)
60  NEXT X
70  END
```

EJERCICIOS RESUELTOS

1. Hacer un programa que defina la función $f(x) = \frac{1}{(1 - x^2)}$ y calcule e imprima una tabla de valores comprendidos entre – 4 y 4, dando a x incrementos de 0.5. (Con los valores obtenidos se sugiere representar gráficamente la función en papel cuadriculado o milimetrado, para observar su forma.)

Solución:

Esta función presenta una dificultad en el cálculo de la tabla de valores, ya que cuando x adquiere el valor 1, el denominador $(1 - x)^2$ se anula ($(1 - 1)^2 = 0^2 = 0$). En el momento de presentarse esta situación, el microordenador interrumpe la ejecución imprimiendo un mensaje de error, referido a la imposibilidad de dividir por cero. En el programa, por tanto, hay que prever esta circunstancia para evitar que la ejecución quede bloqueada.

```
  7 REM TABLA DE VALORES DE F(X) = 1/(1 - X)) ↑ 2
 10 DEF FNA (X) = 1/(1 - X) ↑ 2
 20 PRINT "TABLA DE VAL. DE F(X) = 1/(1 - X) ↑ 2"
 30 PRINT " ____________________________"
 40 PRINT "    (4 espacios)X", "    (4 espacios)1/(1 - X) ↑ 2"
 50 PRINT «____________________________"
 60 FOR X = - 4 TO 4 STEP 0.5
 70 IF X = 0 THEN GOTO 90
 80 PRINT "   (3 espacios)"; X, "    (4 espacios)"; FNA (X)
 90 NEXT X
100 PRINT "____________________________"
110 END
```

Ejecutando este programa (teclear **RUN** y ENTER) se obtiene el siguiente resultado en la pantalla:

```
TABLA DE VAL. DE F(X) = 1/(1-X)↑2
 X             1/(1-X)↑2
-4              .04
-3.5            .049382716
-3              .0625
-2.5            .081632653
-2              .11111111
-1.5            .16
-1              .25
-.5             .44444444
 0              1
 .5             4
 1.5            4
 2              1
 2.5            .44444444
 3              .25
 3.5            .16
 4              .11111111
```

Observar que la instrucción **70 IF X = 0 THEN GOTO 90** evita que el microordenador calcule el valor de la función para X = 0.

2. ***Jugar a los dados con el microordenador***

Éste es un juego muy simple; consiste en lanzar dos dados y sumar los puntos de las caras que salen. Gana el que acumule más puntos en N tiradas.

Un jugador es el «micro», que es el que lanza los dados, o sea, genera dos números aleatorios entre 1 y 6, ambos inclusive, y los suma. El otro jugador eres tú.

Solución:

```
  1  CLS : PRINT
  5  REM JUEGO CON DOS DADOS
 10  LET M = 0 : LET T = 0
 15  INPUT "NUMERO DE LANZAMIENTOS"; N
 20  PRINT "EMPIEZO YO (MICRO). (PARA CONTINUAR TECLEA
CONT)"
 30  STOP
 40  FOR I = 1 TO N
 50  LET X = INT (RND (1) * 6 + 1)
 60  LET Y = INT (RND (1) * 6 + 1)
 70  LET S = X + Y
 80  IF I = INT (I/2) * 2 THEN LET T = T + S : GOTO 140
 90  LET M = M + S
100  CLS
105  PRINT "HE OBTENIDO"; S; "PUNTOS"
110  PRINT "TE TOCA A TI. (TECLEA CONT)"
120  STOP
130  GOTO 170
140  CLS
145  PRINT "HAS OBTENIDO"; S; "PUNTOS"
150  PRINT "ME TOCA A MI (TECLEA CONT)"
160  STOP
170  NEXT I
180  CLS
190  PRINT "HE AQUI EL GANADOR:"
200  PRINT "_____ "
210  PRINT "YO HE SACADO"; M; "PUNTOS"
220  PRINT "Y TU HAS SACADO"; T; "PUNTOS"
230  END
```

— Las instrucciones **50** y **60** generan dos números aleatorios entre 1 y 6, ambos inclusive (simulan el lanzamiento de los dos dados). La instrucción **70** suma los dos números generados (suma los puntos de las caras de los dados).

— La instrucción **80** averigua si el lanzamiento lo ha hecho el micro o tú. Como empieza a jugar el micro, los valores impares de I corresponden a jugadas de aquél, y los pares, a tus jugadas. Por ejemplo, si I = 1, la instrucción **80** actúa así:

Si 1 = **INT** (1/2) * 2 .....
Si 1 = **INT** (0.5) * 2 .....
Si 1 = 0 * 2 .....
Si 1 = 0 .....

Como no se cumple la condición, la ejecución sigue en la instrucción **90 LET M = M + S,** que acumule los puntos que va obteniendo el micro, y, a continuación, te cede la vez (instrucción **110**).

Si, en cambio, n = 2, se tiene:

Si 2 = **INT** (2/2) * 2 .....
Si 2 = **INT** (1) * 2 .....
Si 2 = 1 * 2 .....
Si 2 = 2 .....

Como se cumple la condición, en esta misma instrucción se acumulan los puntos que has obtenido, y se transfiere el control a la **140,** para ceder la voz al micro (instrucción **150**).

- Después de llegar I a obtener el valor N, las instrucciones **210** y **220** imprimen el total de puntos obtenidos por el micro y por ti.

EJERCICIOS DE RECAPITULACIÓN

11.1. Define una función para cada una de las siguientes fórmulas

a) $y = \frac{x + 5}{2}$

b) $y = 3x^2 - \frac{1}{2}x + 5$

c) $y = x \cdot 5^x$

d) $y = x + E(x)$

e) $y = x + |x| + E(x)$

f) $y = \frac{1}{x^2} + x^2 - \frac{x^3 - 1}{5}$

11.2. Calcula el valor de **FNA** (6.285), si la función **FNA(*)** se define así:

```
10  DEF FNA (*) = x – INT (x)
```

11.3. Indica qué precauciones hay que tener a la hora de utilizar las siguientes funciones en un programa:

a) `5 DEF FNB (X) = 1/(2x – 1)`

b) `20 DEF FNC (X) = 5/(x² – 1)`

11.4. Escribe un programa que dé el valor absoluto de un número sin utilizar la función **ABS (X).**

11.5. Cuando se lanzan dos monedas los resultados posibles son:

+, + que se puede expresar así: 0, 0
c, + que se puede expresar así: 1, 0
+, c que se puede expresar así: 0, 1
c, c que se puede expresar así: 1, 1

Haz un programa que simule el lanzamiento de las dos monedas, e imprima el resultado que se produce.

11.6. Haz un programa que defina la función polinómica de tercer grado:

$$f(x) = Ax^3 + Bx^2 + Cx + D$$

e introduzca los valores A = 3, B = 2, C = 1 y D = 0.
Evalúa esta función desde I = 1 hasta 10, incrementando a x de 1 en 1.

11.7. Elabora un programa que defina la función ***parte decimal*** y escriba su valor para cualquier X que se introduzca.

11.8. ¿Cómo se tiene que definir la función parte decimal si el número X es negativo?

11.9. Haz un programa que dé la media de dos números aleatorios, mediante la definición de una función.

11.10. Escribe un programa que lea un número y averigüe si es primo o compuesto.

11.11. Haz un programa que obtenga la lista de los números primos menores que N.

11.12. Elabora un programa que descomponga un número en sus factores primos.

11.13. Escribe un programa que lea dos números y averigüe si son primos entre sí.

11.14. Realiza un programa que lea dos números y averigüe si tienen factores primos comunes.

11.15. Elabora un programa que defina la función $f(x) = -0.5x^2 - x + 6$, calcule una tabla de valores incrementando a x de 0.5 en 0.5 e imprima la tabla.

11.16. Escribe un programa que genere e imprima una tabla de valores para la función f(x) = sen X, entre X = 0 y X = 6.28, e incrementando a X de 0.2 en 0.2. (Con la tabla obtenida representar esta función en papel milimetrado.)

11.17. Haz lo mismo que en el ejercicio anterior con la función f(x) = cos x.

12. Subrutinas

1. Subrutinas: instrucciones GOSUB y RETURN

En algunos programas puede ocurrir que un determinado bloque de instrucciones se debe ejecutar varias veces. En estos casos, no es necesario que aparezca repetido en el programa; basta con escribirlo una sola vez. Para ello se utilizan las instrucciones **GOSUB** y **RETURN.**

Por ejemplo, en el siguiente programa, al bloque de instrucciones **200, 210, 220** y **230** se ejecuta tres veces y, sin embargo, sólo aparece escrito una vez.

```
 10  REM CALCULAR LA SUPERFICIE DEL CIRCULO PARA DISTINTOS
RADIOS
 20  PRINT "RADIO", "SUPERFICIE"
 30  LET R = 1
 40  GOSUB 200
 50  LET R = 5.3
 60  GOSUB 200
 70  LET R = 8.57
 80  GOSUB 200
 90  GOTO 240
200  REM SUBRUTINA
210  LET S = 3.1415 * R ↑ 2
220  PRINT R, S
230  RETURN
240  END
```

La instrucción **40 GOSUB 200** transfiere el control del programa a la instrucción **200,** dando lugar a la ejecución del bloque de instrucciones **200** a **230,** llamado ***subrutina.*** Ésta, termina siempre con la instrucción **RETURN,** que cumple la función de devolver el control a la instrucción siguiente a la **GOSUB.**

Las instrucciones **60 GOSUB 200** y **80 GOSUB 200** obligan a ejecutar la subrutina otras dos veces más.

El orden de ejecución del programa es, por tanto, el siguiente:

```
 10  REM CALCULAR LA SUPERFICIE DEL
CIRCULO PARA DISTINTOS RADIOS
 20  PRINT "RADIO", "SUPERFICIE"
 30  LET R = 1
 40  GOSUB 200
                                  200  REM SUBRUTINA
                                  210  LET S = 3.1415 * R ↑ 2
                                  220  PRINT R, S
                                  230  RETURN
 50  LET R = 2
 60  GOSUB 200
                                  200  REM SUBRUTINA
                                  210  LET S = 3.1415 * R ↑ 2
                                  220  PRINT R, S
                                  230  RETURN
 70  LET R = 3
 80  GOSUB 200
                                  200  REM SUBRUTINA
                                  210  LET S = 3.1415 * R ↑ 2
                                  220  PRINT R, S
                                  230  RETURN
 90  GOTO 240
240  END
```

Hay que tener cuidado en colocar al final de cada subrutina la instrucción **RETURN,** pues en caso contrario se interrumpe la ejecución, apareciendo a continuación un mensaje de error en la pantalla.

2. Cálculo de la raíz cuadrada de varios números mediante una subrutina

Este programa lee una serie de valores almacenados en una instrucción **DATA,** descarta los negativos y calcula, mediante una subrutina, la raíz cuadrada de los positivos y nulos.

```
  5  REM PROGRAMA CON SUBRUTINA
 10  READ A
 15  IF A = -100 THEN GOTO 340
 20  IF SGN (A) < 0 THEN GOTO 10
 30  GOSUB 300
 40  GOTO 10
300  REM SUBRUTINA
310  LET R = SQR (A)
320  PRINT "LA RAIZ CUADRADA DE"; A; "ES"; R
330  RETURN
340  DATA -1, 6.7, 0, 8, -4.6, 5, 6.2, 2.87 E 10, -100
350  END
```

— La instrucción **15** compara el número leído con –100; si coincide, el control se transfiere a **350 END,** dándose por terminada la ejecución del programa. En caso contrario, la instrucción **20** compara el signo de A con 0; si es –1 (número negativo) se pasa a leer otro número, y si es 0 ó 1 se ejecuta la instrucción **30 GOSUB 300,** que transfiere el control a la subrutina.

— Una vez terminada de ejecutarse la subrutina, su última instrucción **330 RETURN** envía el control a la **40,** y ésta a su vez, a la **10,** para volver a leer otro dato.

Ejercicios

12.1. Explica cómo funciona el siguiente programa al ser ejecutado.

```
 10 INPUT "A ="; A
 20 INPUT "B ="; B
 30 GOSUB 200
 40 GOSUB 300
 50 PRINT "P ="; P; "C ="; C
 60 GOTO 10
200 REM PRODUCTO
210 LET P = A * B
220 RETURN
330 REM COCIENTE
310 LET C = A/B
320 RETURN
400 END
```

12.2. Indica lo que ocurrirá al ejecutarse el siguiente programa.

```
 10 READ A
 20 GOSUB 100
 30 PRINT A, B, C
100 REM SUBRUTINA
110 LET B = A ↑ 2
120 LET B = A ↑ 3
200 DATA –3, 2, 1, 8, 4
210 GOTO 10
220 END
```

12.3. Escribe un programa con dos subrutinas, una para sumar dos números introducidos por el teclado y la otra, para restarlos.

12.4. Haz un programa que genere un número aleatorio entero comprendido entre 0 y 10, éste excluido, y que permita introducir un número comprendido también entre 0 y 10.

Además, el programa averiguará mediante una subrutina si el número generado coincide con el número introducido.

3. Un programa que permite elegir entre diversas opciones

- A veces, un programa se puede dividir en varias partes o módulos (subrutinas), en función de tareas más simples que tiene que cumplir.

  Cada módulo es un cierto proceso o tarea con entidad propia. Esto supone que el programa en conjunto sea más fácil de entender. Se consigue aún mayor claridad utilizando al principio de cada módulo (subrutina) una instrucción **REM** que describe la tarea a realizar.

- El programa siguiente, que actúa como una sencilla calculadora, aplica lo que acabamos de comentar:

```
 10  REM CALCULADORA
 20  PRINT "OPCIONES:"
 30  PRINT "ESCRIBE:"
 40  PRINT "1 PARA SUMAR"
 50  PRINT "2 PARA RESTAR"
 60  PRINT "3 PARA MULTIPLICAR"
 70  PRINT "4 PARA DIVIDIR"
 80  PAUSE 200
 90  CLS
100  INPUT "INTRODUCE LOS NUMEROS"; A, B
110  PRINT : PRINT
120  INPUT "INTRODUCE LA OPCION"; V
130  IF V = 1 THEN GOSUB 1000
140  IF V = 2 THEN GOSUB 2000
150  IF V = 3 THEN GOSUB 3000
160  IF V = 4 THEN GOSUB 4000
170  GOTO 120
1000 REM SUMA
1010 CLS
1020 PRINT A; "+"; B; "="; A + B
1030 RETURN
2000 REM RESTA
2010 CLS
2020 PRINT A; "-"; B; "="; A - B
2030 RETURN
3000 REM PRODUCTO
3010 CLS
3020 PRINT A; "*"; B; "="; A * B
3030 RETURN
4000 REM COCIENTE
4010 CLS
4020 PRINT A; "/"; B; "="; A/B
4030 RETURN
4100 END
```

Las instrucciones **20** a **70** presentan el ***menú*** de opciones.

Las instrucciones **100** a **160** envían a las subrutinas que correspondan, las cuales se inician en las instrucciones **1000**, **2000**, **3000** y **4000**.

4. La instrucción ON ... GOSUB

- En muchos microordenadores las instrucciones **130** a **160** del programa anterior se puede sustituir por

  130 ON V GOSUB 1000, 2000, 3000, 4000

  que envía el control a las subrutinas **1000, 2000, 3000** ó **4000,** según que V valga 1, 2, 3 ó 4, respectivamente.

- Una instrucción que cumple una función parecida a la anterior, pero no idéntica, es la instrucción **ON ... GOTO.** Así,

  130 ON V GOTO 1000, 2000, 3000, 4000

  transfiere el control a la instrucción **1000, 2000, 3000** ó **4000,** según que V valga 1, 2, 3 ó 4, respectivamente. Sin embargo, utilizando esta instrucción ya no es posible volver al lugar de la bifurcación; además, el microordenador daría un mensaje de error en el momento de ejecutar la instrucción **RETURN.**

5. Procedimientos

- Ciertos lenguajes de alto nivel, como el Pascal y el Logo, permiten definir un ***proceso*** (procedure) dentro de un programa y, una vez definido, utilizarlo como si fuera una instrucción más.

  Aunque el BASIC no tiene esta posibilidad, puede simularla ***definiendo una variable y asignándole un número correspondiente a una subrutina.*** Una vez hecho esto, en lugar de escribir **GOSUB** {número} se puede escribir **GOSUB** {nombre de la variable}.

  Este recurso se utiliza en el siguiente programa que dibuja varias rectas horizontales paralelas, mediante una subrutina llamada RECTA.

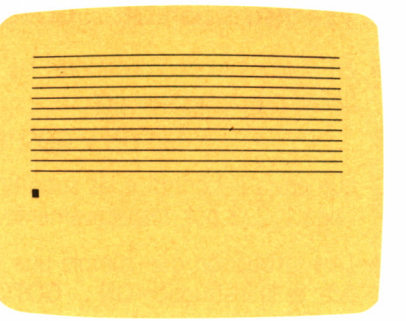
```
10  LET RECTA = 1000
20  INPUT "NUMERO DE RECTAS"»; N
25  CLS
30  FOR I = 1 TO N
40  GOSUB RECTA
50  NEXT I
60  GOTO 1050
1000  REM RECTA
1010  PRINT "_________________"
1020  RETURN
1050  END
```

(*)

(*) En el *ZX Spectrum*, en lugar de 1050 END hay que poner 1050 STOP, dado que este microordenador no posee la instrucción END.

La instrucción **10** define la variable RECTA y almacena en ella el valor 1000. Esto hace que la instrucción **50 GOSUB RECTA** dibuje una recta horizontal cada vez que se ejecuta, debido a que transfiere el control a la subrutina RECTA (instrucciones **1000** a **1020**).

Cuando se ejecuta este programa se obtiene una familia de rectas paralelas, como muestra la fotografía.



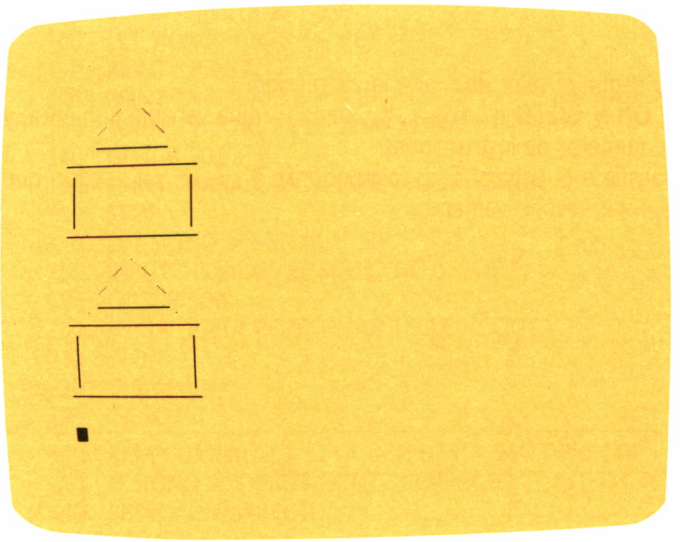
- En el siguiente programa, combinando dos subrutinas llamadas TRIÁNGULO y CUADRADO, se logra dibujar una casa. Ésta se repite varias veces en función del valor que asignemos a N.

```
  5  REM DIBUJA CASAS
 10  LET TRIANGULO = 500
 20  LET CUADRADO = 600
 30  INPUT "NUMERO DE CASAS"; N
 40  CLS
 50  FOR I = 1 TO N
 60  GOSUB TRIANGULO
 70  GOSUB CUADRADO
 80  NEXT I
 90  GOTO 660

500  REM TRIANGULO
510  PRINT "          ."
520  PRINT "         / \"
530  PRINT "        /   \"
540  PRINT "       /_____\"
550  RETURN

600  REM CUADRADO
610  PRINT "      |     |"
620  PRINT "      |     |"
630  PRINT "      |     |"
640  PRINT "      |_____|"
650  RETURN
660  END
```

Este programa contiene dos subrutinas, la subrutina TRIÁNGULO (instrucciones **500** a **550**) y la subrutina cuadrado (instrucciones **600** a **650**).

Combinando estas dos subrutinas (instrucciones **60** y **70**) se consigue «armar» una casa, la cual se puede dibujar tantas veces como se desee, mediante el bucle **40-70**.

Si se ejecuta este programa y se asigna a N el valor 2, se obtiene en la pantalla lo que muestra la fotografía.



6. Resumen de las instrucciones estudiadas en este capítulo

- GOSUB n
  - — ***Significado:*** ir a la subrutina.
  - — ***Formato:*** n **GOSUB n'**, siendo n' el número de la instrucción en la que comienza la subrutina.
  - — ***Función:*** transfiere el control de la ejecución a la instrucción n'.

- RETURN
  - — ***Significado:*** volver.
  - — ***Formato:*** n' **RETURN**
  - — ***Función:*** devuelve el control a la instrucción siguiente a la n GOSUB n'.

- ON ... GOSUB

  — ***Significado:*** según el valor de ... ir a la subrutina.
  — ***Formato:*** n ON V GOSUB n_1, n_2, ... n_k, siendo V una variable numérica, y n_1, n_2, ..., n_k los números de las instrucciones en las cuales comienzan las subrutinas.
  — ***Función:*** envía el control a las subrutinas que se inician con las instrucciones n_1, n_2, ... n_k, según que V valga 1, 2, ..., k, respectivamente.

- ON ... GOTO

  — ***Significado:*** según el valor de ... ir a la instrucción.
  — ***Formato:*** n ON V GOTO n_1, n_2, ..., n_k, siendo V una variable numérica, y n_1, n_2, ..., n_k números de instrucciones.
  — ***Función:*** transfiere el control a las instrucciones n_1, n_2, ..., n_k, según que V valga 1, 2, ..., k, respectivamente.

EJERCICIOS RESUELTOS

1. Hacer un programa que genere dos números aleatorios, el primero comprendido entre –5 y 5, y el segundo, que deberá ser entero, comprendido entre 0 y 20. Una vez generados estos números, el programa preguntará por el signo de su producto y por el de la potencia del primero elevado al segundo. Dará opción a introducir la respuesta y comprobará, mediante una subrutina, si es correcta.

Solución:

```
 10  REM SIGNO DEL PRODUCTO Y POTENCIA
 20  PRINT "RECUERDA:"
 30  PRINT "        * SIGNO MENOS ="; -1
 40  PRINT "        * NO TIENE SIGNO ="; 0
 50  PRINT "        * SIGNO MAS ="; 1
 60  PRINT : PRINT : PRINT
 70  RANDOMIZE
 80  LET A = RND * (-10) + 5
 90  LET B = INT (RND * 20)
100  LET SIGNO = SGN (A * B)
110  PRINT : PRINT
120  PRINT "A ="; A; "B ="; B
130  INPUT "? CUAL ES EL SIGNO DE A * B?"; S
140  GOSUB 500
150  PRINT "PARA CONTINUAR TECLA CONT"
160  STOP
170  LET SIGNO = SGN (A ↑ B)
180  INPUT "CUAL ES EL SIGNO DE A ↑ B?"; S
190  GOSUB 500
200  PRINT "PARA CONTINUAR TECLEA CONT"
210  STOP
220  CLS : PRINT
230  GOTO 70
500  REM COMPRUEBA SI LA RESPUESTA ES CORRECTA
510  IF SIGNO = S THEN PRINT "CORRECTO" : GOTO 530
520  PRINT "INCORRECTO"
530  RETURN
600  END
```

— Las instrucciones **70**, **80** y **90** generan los números aleatorios A y B y la instrucción **100** calcula el signo del producto A * B.

— Mediante la instrucción **130** se introduce la respuesta (–1, 0, ó 1) y la **140 GOSUB 500** transfiere el control a la subrutina, donde se verifica si la respuesta es correcta o incorrecta.

— Análogamente, las instrucciones **170**, **180** y **190** cumplen la misma función para la potencia A ↑ B.

— La instrucción **230** devuelve el control a la 70, para repetir otra vez todo el proceso.

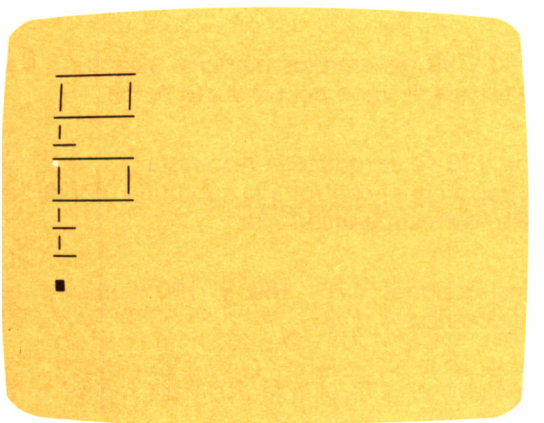
2. a) Dibujar lo que aparece en la pantalla al ejecutarse el siguiente programa para N = 0, 1, 2 y 3.

   b) ¿Por qué da un mensaje de error cuando se introduce el valor 3 para N?

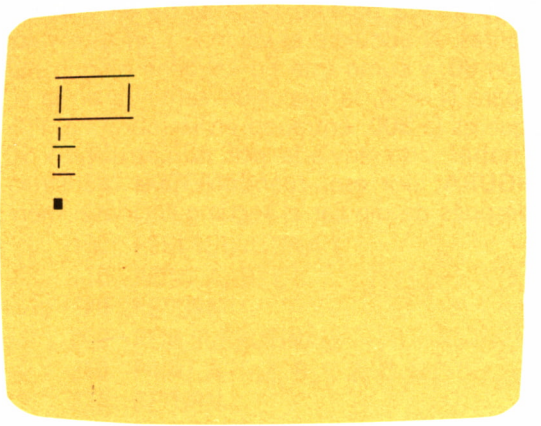
```
  5 REM DIBUJA FIGURAS
 10 INPUT "VALOR DE N"; N
 20 CLS
 30 LET PRIMERA FIGURA =300
 40 LET SEGUNDA FIGURA = 400
 50 GOSUB PRIMERA FIGURA
 60 GOSUB SEGUNDA FIGURA
 70 LET N = N + 1
 80 ON N GOTO 50, 60, 500
300 REM PRIMERA FIGURA
310 PRINT "  ¯¯¯¯¯|"
320 PRINT "  |    |"
330 PRINT "  |    |"
340 PRINT "  |____|"
350 RETURN
400 REM SEGUNDA FIGURA
410 PRINT "  |"
420 PRINT "  --"
430 RETURN
500 END
```

Solución:

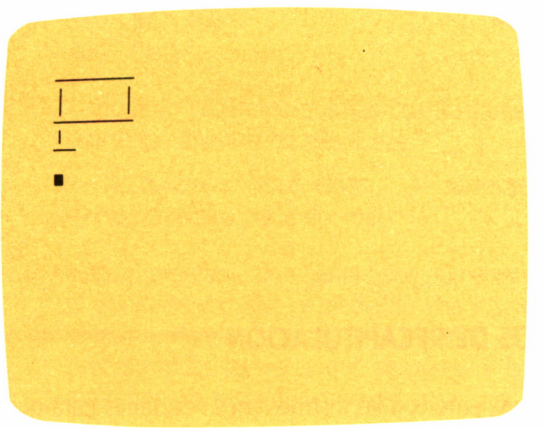
a) Para N = 0



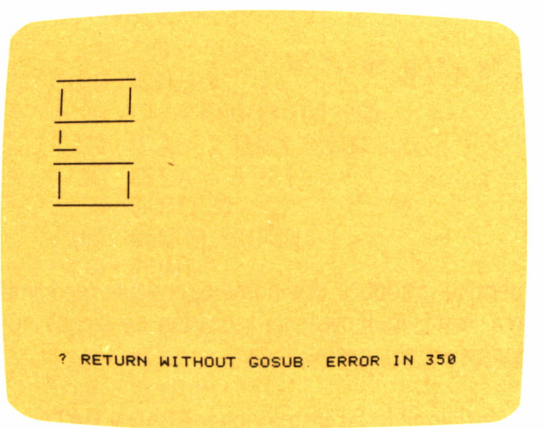
Para N = 1



Para N = 2



Para N = 3



b) Cuando se introduce el valor 3 para N, después de dibujarse la primera y segunda figuras, N adquiere el valor 4. Con este valor se ejecuta la instrucción **80,** y como ésta no contiene ningún número de instrucción para N = 4, la ejecución continúa en la siguiente instrucción, que es la **300,** entrando en la subrutina, PRIMERA FIGURA, sin haber sido enviada por la instrucción **50 GOSUB PRIMERA FIGURA;** por eso, cuando llega a la instrucción **350 RETURN,** después de dibujar el rectángulo, da el mensaje de error.

EJERCICIOS DE RECAPITULACIÓN

12.1. Completa el programa siguiente con las instrucciones necesarias para que al ejecutarlo no dé mensaje de error.

```
 10  INPUT "A ="; A
 20  INPUT "B ="; B
 30  PRINT "A ="; A; "B ="; B, "P ="; P
 40  GOTO 10
100  REM PRODUCTO
110  LET P = A * B
120  RETURN
200  END
```

12.2. Haz un programa que permita introducir dos números, calcule mediante cuatro subrutinas la suma, resta, multiplicación y división de dichos números, y después los imprima en una misma línea.

12.3. Repite el programa anterior utilizando las instrucciones **READ** y **DATA.**

12.4. Escribe lo que imprime el siguiente programa al ser ejecutado.

```
 10  READ A, B, C
 20  GOSUB 100
 30  GOSUB 200
 40  GOSUB 300
 50  GOTO 400
 60  DATA 1, 2, 3
100  REM SUBRUTINA 1
110  PRINT A; B; C
120  RETURN
200  REM SUBRUTINA 2
210  PRINT A ↑ 2; B ↑ 2; C ↑ 2
220  RETURN
300  REM SUBRUTINA 3
310  PRINT A ↑ 3; B ↑ 3; C ↑ 3
320  RETURN
400  END
```

12.5. Elabora un programa que lea cuatro números positivos y mediante tres subrutinas imprima en tres líneas,

— los cuatro números leídos, — su respectiva parte entera,
— su respectiva parte decimal.

12.6. Haz un programa que genere dos números aleatorios enteros A y B del 1 al 6, ambos inclusive, y mediante una subrutina averigüe cuál de ellos es el mayor, dando un mensaje al respecto. El programa permitirá repetir el proceso todas las veces que se desee, y la ejecución del mismo deberá interrumpirse entre dos procesos sucesivos.

12.7. Dibuja lo que imprimiría el siguiente programa al ser ejecutado, si se introduce el valor 1 para A.

```
  1  CLS
 10  INPUT A
 20  ON A GOSUB 100, 200
 30  IF A > 2 THEN GOTO 300
 40  LET A = A + 1
 50  GOTO 20
100  REM SUBRUTINA 1
110  PRINT "------------------------------"
120  RETURN
200  REM SUBRUTINA 2
210  PRINT "**********************************"
220  RETURN
300  END
```

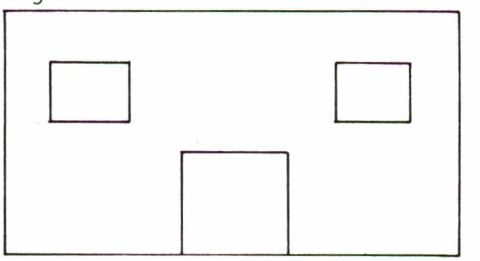
12.8. ¿Qué imprimirá el programa anterior si se introduce el valor 2 para A? ¿Y si se introduce el valor 3?

12.9. Cambia la instrucción **20** del programa escrito en el ejercicio 12.7 por **20 ON A GOTO 100, 200** y analiza lo que aparecería en la pantalla al ser ejecutado, con los valores 1, 2 y 3 para A.

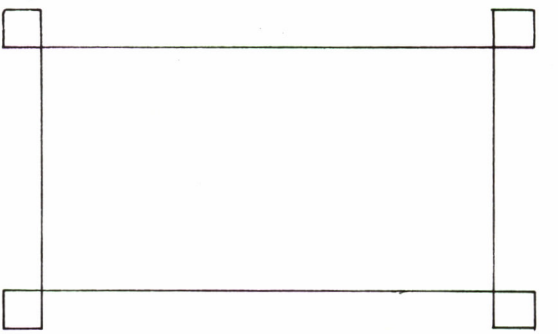
2.10. El siguiente programa contiene una subrutina. Dibuja lo que imprime en la pantalla al ser ejecutado.

```
 10  LET PROCEDIMIENTO = 200
 20  FOR I = 1 TO 3
 30  GOSUB PROCEDIMIENTO
 40  NEXT I
550  GOTO 300
200  REM PROCEDIMIENTO
210  PRINT "*"
220  PRINT "**"
230  PRINT "***"
240  PRINT "****"
300  END
```

12.11. Haz un programa con una subrutina que dibuje dos veces en la pantalla la siguiente figura:



12.12. Elabora un programa con tres subrutinas que dibuje tres veces la siguiente figura:

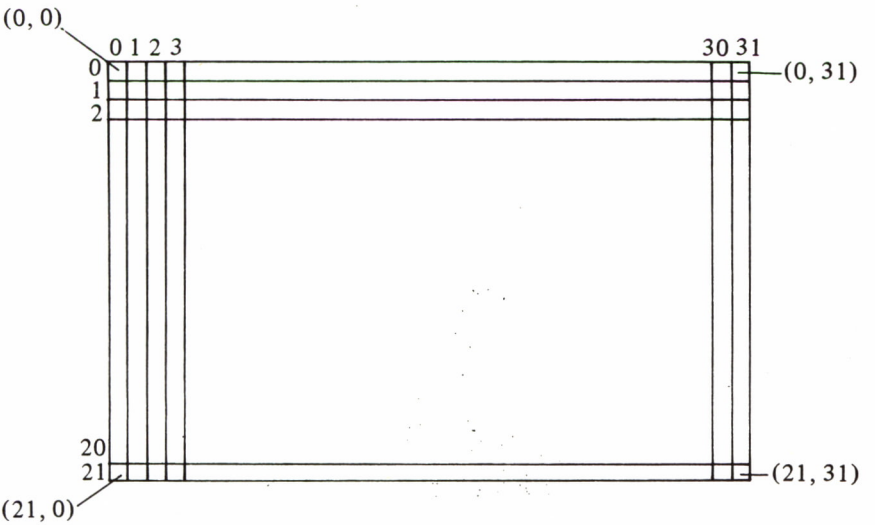


13. Impresión ordenada

1. La pantalla de un microordenador

La pantalla de un microordenador está dividida en filas y columnas. En este libro consideraremos un microordenador con posibilidades de escribir en 22 filas (de la 0 a la 21) y en 32 columnas (de la 0 a la 31).

Como el cursor o punto luminoso que sigue la escritura de los caracteres suele encontrarse inicialmente en la posición superior izquierda, a esta posición se asignan las coordenadas (0, 0), a la superior derecha (0, 31), a la inferior izquierda (21, 0) y, por último, a la inferior derecha (21, 31).



Cualquier gráfico que se dibuje en este tipo de pantalla se dice que está en ***baja resolución.***

2. Instrucción PRINT TAB

- La instrucción **PRINT TAB** (imprimir tabulando) sirve para indicar al microordenador en qué columna ha de comenzar a imprimir.

Así, por ejemplo,

```
PRINT TAB (8); "MAYO"
```

imprime la palabra MAYO a partir de la columna 8.

- En el siguiente programa se aplica la instrucción **PRINT TAB** para obtener una tabla que imprime a partir de la columna 3.ª, N números naturales, a partir de la 8.ª , sus cuadrados, y a partir de la 19.ª, sus raíces cuadradas.

```
 1  REM CUADRADO Y RAIZ DE UNA SERIE DE NUMEROS
10  INPUT "CUANTOS NUMEROS"; N
20  PRINT TAB (4); "X"; TAB (8); "X ↑ 2"; TAB (19); "RAIZ (X)"
30  FOR I = 1 TO N
40  INPUT X
50  PRINT TAB (3); X; TAB (8); X ↑ 2; TAB (19); SQR (X)
60  NEXT I
70  END
```

Si al ejecutar el programa (teclear **RUN** y ENTER), se introducen los datos

$$N = 4,\quad X = 3,\quad X = 7,\quad X = 8 \quad y \quad X = 11,$$

en la pantalla aparecerá:

Columnas: 3.ª 8.ª 19.ª

X	X ↑ 2	RAIZ (X)
3	9	1.7320508
7	49	2.6457513
8	64	2.8284271
11	121	3.3166248

Este programa funciona de la siguiente manera:

— La instrucción **10** pide cuántos números se van a introducir.

— La instrucción **20** imprime la cabecera X, X ↑ 2 y RAÍZ (X) a partir de las columnas 3.ª, 8.ª y 19.ª respectivamente, de la fila 0.

— El bucle **30-60** repite N veces las instrucciones **40** y **50.** La **40** pide un valor X y la **50** imprime ese valor X, su cuadrado X ↑ 2 y su raíz cuadrada SQR (X), en las columnas 3.ª, 8.ª y 19.ª, respectivamente.

Ejercicios

13.1. Haz un programa que escriba como cabecera «NOMBRE» a partir de la columna 10, y «APELLIDO», a partir de la 20.ª. Además, permitirá introducir nombres y apellidos de personas, imprimiéndolos a partir de la columna 10.ª y 20.ª, respectivamente.

13.2. ¿Qué aparecerá en la pantalla al ejecutar el siguiente programa? ¿Cada vez que se ejecute aparecerá el mismo contenido en la pantalla?

```
10 RANDOMIZE
20 FOR I = 1 TO 10
30 LET N = INT (10 * RND)
40 PRINT TAB (N); "BUENOS DIAS"; TAB (25); N
50 NEXT I
60 END
```

13.3. Haz un programa que imprima una A en cada una de las 22 filas, de tal manera que en cada fila la posición de la columna se obtenga de forma aleatoria, entre las 32 posibles.

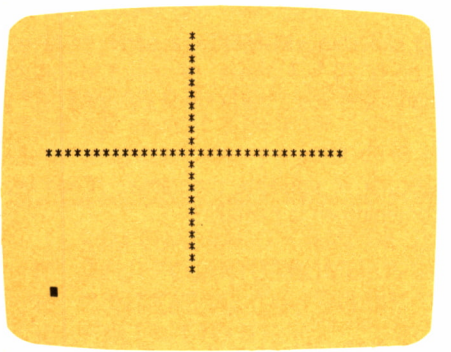
3. Obtención de gráficos con PRINT TAB

La instrucción **PRINT TAB** también se puede utilizar para dibujar gráficos, aunque su manejo no siempre es sencillo, debido a que esta instrucción no permite saltar filas (para ello es necesario elaborar un programa).

Aplicaremos la instrucción **PRINT TAB** para dibujar dos ejes coordenados perpendiculares.

```
  1 REM EJES COORDENADOS
 10 FOR I = 0 TO 10
 20 PRINT TAB (16); "*"
 30 NEXT I
 40 FOR J = 0 TO 31
 50 PRINT "*";
 60 NEXT J
 70 FOR K = 12 TO 21
 80 PRINT TAB (16); "*"
 90 NEXT K
100 END
```

Si se ejecuta el programa (**RUN** y ENTER), se obtiene lo siguiente:



Este programa funciona así:

— El bucle **10-30,** de variable I, imprime en la columna 16 asteriscos desde la fila 0 a la 10.

— El bucle **40-60** dibuja el eje horizontal, poniendo un asterisco en cada columna de la fila 11. Esto se consigue gracias a que la instrucción 50 termina en **punto y coma** y, por tanto, cada carácter se escribe a continuación del anterior.

— Por último, el bucle **70-90,** de variable K, continúa imprimiendo asteriscos en la columna 16, de la fila 12 a la 21.

Ejercicios

13.4. Haz un programa que dibuje un asterisco en la fila 11, columna 16. (Para llegar a la fila 11, introducir un bucle.)

13.5. Realiza un programa que llene la columna 12 de asteriscos.

13.6. Haz un programa que imprima A en toda la fila 13.

13.7. Escribe un programa que dibuje una recta que vaya de (0, 0) a (21, 21).

13.8. Haz un programa que dibuje una recta de (0, 31) a (21, 10).

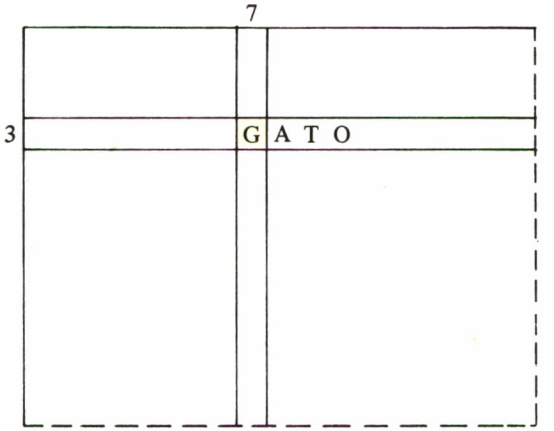
13.9. ¿Qué figura geométrica dibuja este programa?

```
10  PRINT TAB (16); "*"
20  FOR I = 1 TO 10
30  PRINT TAB (16 – I); "*"; TAB (16 + I); "*"
40  NEXT I
50  FOR I = 9 TO 1 SETP –1
60  PRINT TAB (16 – I); "*"; TAB (16 + I); "*"
70  NEXT I
80  PRINT TAB (16); "*"
90  END
```

4. Instrucción PRINT AT

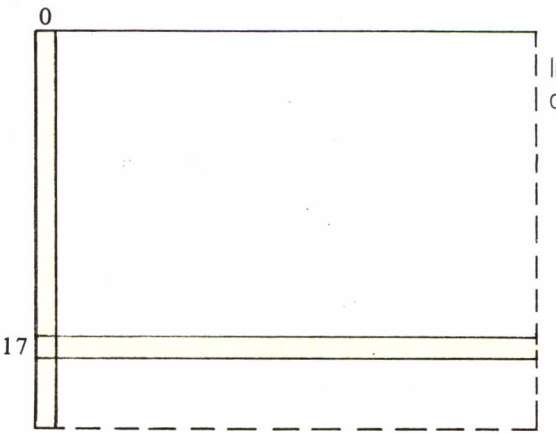
- La instrucción **PRINT AT** sirve para imprimir un número o una cadena a partir de una fila y columna determinada. Así, por ejemplo:

PRINT AT 3, 7; "GATO"



Imprime la palabra GATO a partir de la fila 3 y de la columna 7.

PRINT AT 17, 0; N



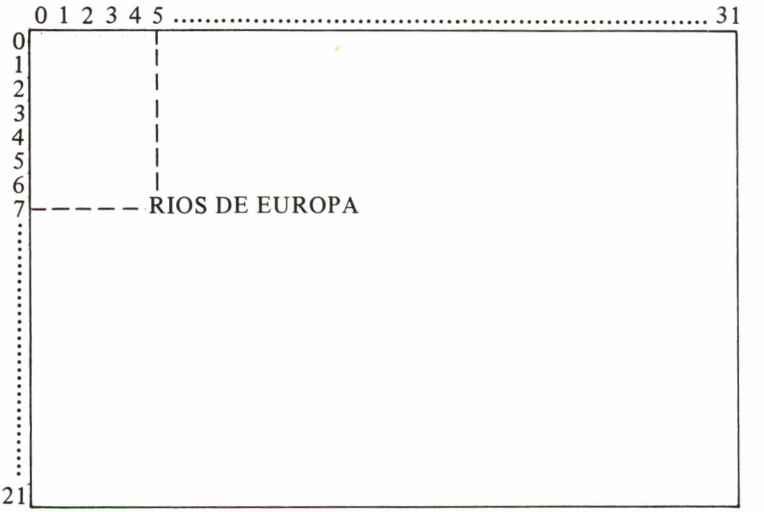
Imprime el valor de N a partir de la fila 17 y de la columna 0.

Conviene tener presente que el formato de esta instrucción puede variar de un microordenador a otro (consultar manual).

- Aplicaremos esta instrucción en un programa que escribe la frase que se introduce por teclado, a partir de la fila y columna deseada; además subraya dicha frase.

```
  1  REM SUBRAYAR
 10  INPUT "TEXTO"; T$
 20  INPUT "FILA"; F
 30  INPUT "COLUMNA"; C
 40  PRINT AT F, C; T$
 50  LET A$ = "  "
 60  FOR I = 1 TO LEN(T$)
 70  LET A$ = A$ + "--"
 80  NEXT I
 90  PRINT AT F + 1, C; A$
100  END
```

Si al ejecutar el programa (teclear **RUN** y ENTER) se introduce "RIOS DE EUROPA" como texto, 7 como valor de fila y 5 como valor de columna, se obtiene como resultado lo siguiente:



El programa actúa de la siguiente manera:

— Por medio de las instrucciones **10, 20** y **30** se introduce el texto, la fila y la columna a partir de las cuales se desea imprimir dicho texto.

— La instrucción **40** imprime a partir de la fila F y la columna C el texto almacenado en T$.

— En la instrucción **50** se define la variable A$, en la cual se va guardando el subrayado, a medida que se ejecuta el bucle 60-80.

— Por último, la instrucción **90** subraya el texto almacenado en T$, imprimiendo en la fila F + 1 y a partir de la columna C el subrayado almacenado en A$.

Ejercicios

13.10 Indica qué escribe el siguiente programa, y en qué posición de la pantalla.

```
 5 REM ESCRIBE EN LA PANTALLA
10 PRINT
20 PRINT AT 10,12; "*********"
30 PRINT AT 11,14; "BASIC"
40 PRINT AT 12,12; "*********"
50 PRINT
60 END
```

13.11. Haz un programa que lea un nombre y lo imprima 10 veces; la primera vez a partir de la fila 1 y columna 1, la segunda a partir de la fila 2 y columna 2, y así sucesivamente.

13.12. Escribe un programa tal que al introducir una frase calcule su longitud L y la imprima a partir de la fila L y columna L.

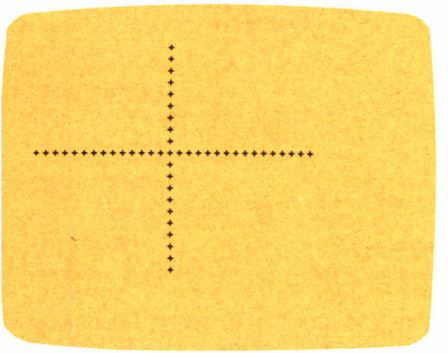
5. Obtención de gráficos con PRINT AT

La instrucción **PRINT AT** permite dibujar gráficos en la pantalla con mayor facilidad que la **PRINT TAB**, debido a que la primera determina ***la fila y la columna*** en la cual se desea imprimir un carácter (asterisco, punto, etc.), mientras que la segunda sólo determina la columna.

Veamos cómo se dibujan dos ejes coordenados utilizando la instrucción PRINT AT.

```
 1 REM EJES DE COORDENADAS
10 FOR J = 0 TO 31
20 PRINT AT 11, J; "+"
30 NEXT J
40 FOR I = 0 TO 21
50 PRINT AT I, 16; "+"
60 NEXT I
70 END
```

Al ejecutar este programa (teclear **RUN** y [ENTER]) aparece en la pantalla lo siguiente:



En este programa las instrucciones **10, 20** y **30** dibujan el eje horizontal en la fila 11 de la pantalla y las instrucciones **40, 50** y **60** dibujan el eje vertical en la columna 16.

Ejercicios

13.13. Haz un programa que dibuje una línea recta de asteriscos en la fila 3.

13.14. Haz un programa que dibuje un cuadrado de 21 × 21 (no considerar las columnas 22 a 31).

13.15. Escribe un programa que dibuje las diagonales de un cuadrado de 21 × 21.

13.16. ¿Qué hace el siguiente programa?

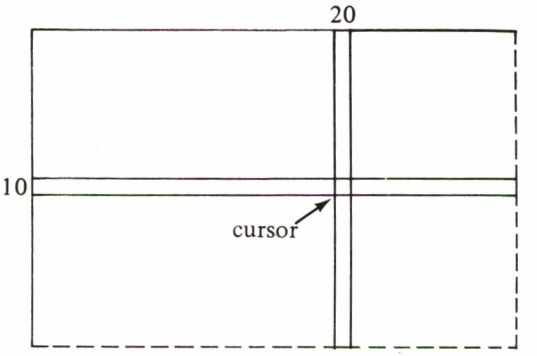
```
10  FOR I = 0 TO 21
20  PRINT AT I, 0; "*"
30  PRINT AT I, 31; "*"
40  NEXT I
50  FOR J = 0 TO 31
60  PRINT AT 0, J; "*"
70  PRINT AT 21, J; "*"
80  NEXT J
90  END
```

6. Instrucción LOCATE

- La instrucción **LOCATE** permite situar el cursor en una posición determinada de la pantalla. Por ejemplo,

```
10  LOCATE 20, 10
```

sitúa el cursor en la posición correspondiente a la columna 20 y a la fila 10.



Una vez situado el cursor, se puede imprimir cualquier texto a partir de la columna especificada por el primer número. Así, ejecutando estas dos instrucciones

```
10 LOCATE 20, 10
20 PRINT "ANTONIO"
```

se imprime en la pantalla la palabra ANTONIO a partir de la columna 20 y de la fila 10.

A veces, para ganar claridad, conviene poner estas dos instrucciones juntas en una misma línea, separadas por dos puntos:

```
10 LOCATE 20, 10 : PRINT "ANTONIO"
```

- Cuando la palabra **LOCATE** va seguida de un solo número, el cursor se posiciona en la columna especificada por dicho número. Por ejemplo, ejecutando

```
10 LOCATE 15 : PRINT "ARBOL"
```

se imprime la palabra ÁRBOL a partir de la columna 15.

Si en cambio, la palabra **LOCATE** va seguida de una coma y de un número, el cursor se posiciona en la fila especificada por dicho número. Así.

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

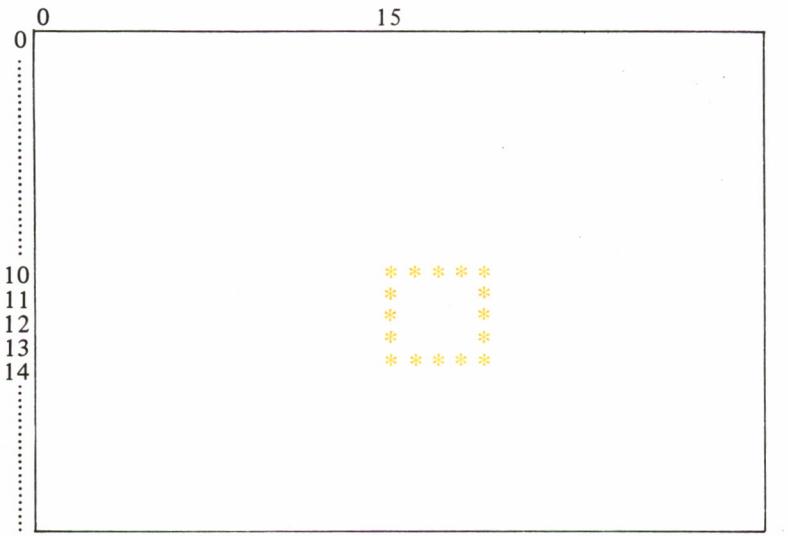
imprime la palabra JARDIN en la fila 13.

En relación a este último caso, conviene advertir que si la anterior instrucción **PRINT** termina en blanco, la palabra JARDÍN se imprime a partir de la columna 0; si termina en ***punto y coma*** (;), se imprime a partir de la siguiente columna en la que se imprimió el último carácter; finalmente, si termina en ***coma,*** se imprime en el siguiente campo.

- Aplicamos la instrucción **LOCATE** en el siguiente programa para dibujar un cuadrado de lado 5

```
10  REM CUADRADO
20  CLS
30  LOCATE 15, 10 : PRINT "*****"
40  LOCATE 15, 11 : PRINT "*    *"
50  LOCATE 15, 12 : PRINT "*    *"
60  LOCATE 15, 13 : PRINT "*    *"
70  LOCATE 15, 14 : PRINT "*****"
```

Tecleando **RUN** y ENTER se obtiene como resultado el siguiente cuadrado de asteriscos:



7. Resumen de las instrucciones estudiadas en este capítulo

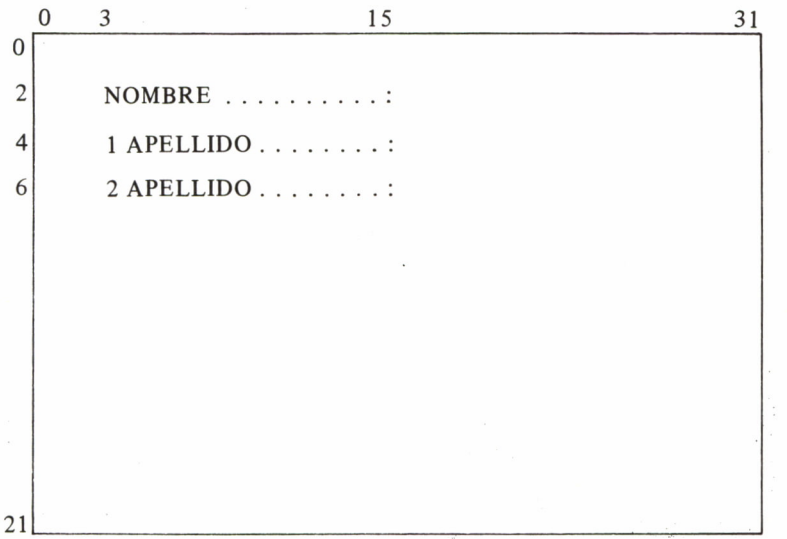
- PRINT TAB
  - — ***Significado:*** tabula e imprime.
  - — ***Formato:*** n PRINT TAB (columna); { variables / expresiones aritméticas / texto entre comillas }
  - — ***Función:*** escribe a partir de la columna indicada.

- PRINT AT
  - — ***Significado:*** imprime en un punto determinado.
  - — ***Formato:*** n PRINT AT fila, columna; { variables / expresiones aritméticas / texto entre comillas }
  - — ***Función:*** escribe a partir de la fila y columna que se indica.
  - — ***Observación:*** el formato de esta instrucción varía mucho según el microordenador (consultar manual).

- LOCATE
  - — ***Significado:*** localizar, situar.
  - — ***Formato:*** LOCATE C, F.
  - — ***Función:*** sitúa el cursor en la posición correspondiente a la columna C y a la fila F.
  - — ***Observación:*** Si sólo interesa especificar la columna, se puede poner **LOCATE** C
    Si sólo interesa especificar la fila se puede poner **LOCATE,** F.

EJERCICIOS RESUELTOS

1. Escribir un programa que imprima en la pantalla lo indicado en esta figura:



Además, permitirá introducir el nombre y los dos apellidos de una persona, los cuales imprimirá a partir de la columna 17, en las filas 2, 4 y 6, respectivamente.

Solución:

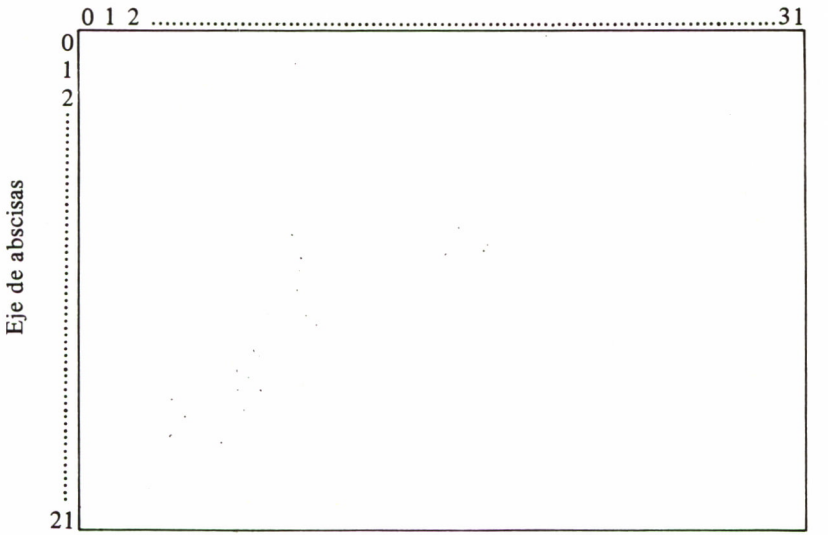
```
 10  PRINT AT 2, 3; "NOMBRE.....:"
 20  PRINT AT 4, 3; "1 APELLIDO.....:"
 30  PRINT AT 6, 3; "2 APELLIDO.....:"
 40  INPUT "NOMBRE"; N$
 50  INPUT "1 APELLIDO"; A$
 60  INPUT "2 APELLIDO"; B$
 70  PRINT AT 2, 17; N$
 80  PRINT AT 4, 17; A$
 90  PRINT AT 6, 17; B$
100  END
```

2. Diseñar un programa que recuadre un texto que se introdujo previamente.

Solución:

```
 10  INPUT "NOMBRE"; N$
 20  LET L = LEN (N$)
 30  FOR I = 0 TO L + 3
 40  PRINT AT 0, I; "*"
 50  PRINT AT 4, I; "*"
 60  NEXT I
 70  FOR J = 1 TO 3
 80  PRINT AT J, 0; "*"
 90  PRINT AT J, L + 3; "*"
100  NEXT J
110  PRINT AT 2, 2; N$
120  END
```

3. Como la instrucción **PRINT TAB** imprime fila a fila sin poder saltarse ninguna, para representar gráficas de funciones es conveniente imaginarse la pantalla girada 90°. De este modo, la primera fila (fila 0) sería el eje de ordenadas y la primera columna (columna 0), el eje de abscisas.



Teniendo esto en cuenta hacer un programa que dibuje la gráfica de la función seno en posición vertical.

Solución:

Para completar un período, los valores del seno tendrán que ir desde X = 0 a X = 2π, y como hay 22 filas, los pasos serán de 2 π/22.

Como los valores extremos de la función seno son –1 y 1, y la pantalla tiene 32 columnas, multiplicaremos la función por 15 (la mitad aproximada de 32). Además, para que la gráfica quede centrada en la pantalla tendremos que sumar 15 a cada valor del seno. Luego el valor de Y, que determinará en qué columna se deberá imprimir un punto o un asterisco, será:

Y = 15 + 15 * SIN (X)

```
10  FOR X = 0 TO 2 * PI STEP 2 * PI/22
20  LET Y = 15 + 15 * SIN (X)
30  PRINT TAB (Y); "*"
40  NEXT X
50  END
```

Veamos cómo actúan las instrucciones **20** y **30** para algunos valores posibles del seno:

— Si SIN (X) = –1, se tiene:

Y = 15 + 15 * (–1) = 15 – 15 = 0

Luego la instrucción **30** imprimirá un asterisco en la columna 0 (extremo izquierdo de la pantalla).

— Si SIN (X) = 0, se tiene:

$$Y = 15 + 15 * 0 = 15 + 0 = 15$$

Luego la instrucción **30** imprimirá un asterisco en la columna 15 (centro aproximado de la pantalla).

— Si SIN (X) = 1, se tiene:

$$Y = 15 + 15 * 1 = 15 + 15 = 30$$

Luego la instrucción **30** imprimirá un asterisco en la columna 30 (extremo derecho de la pantalla).

Los demás valores del seno se imprimirán entre la columna 0 y 30.

Modificando las instrucciones **10** y **20** del programa anterior se consigue resultados similares:

```
10  FOR X = 0 TO 21
20  LET Y = 15 + 15 * SIN (X * 2 * PI/22)
30  PRINT TAB (Y); "*"
40  NEXT X
50  END
```

EJERCICIOS DE RECAPITULACIÓN

13.1. ¿Qué instrucciones son erróneas? ¿Por qué?

a) 100 PRINT TAB (23), "HOLA"
b) 200 PRINT TAB (10); 1; TAB (25); "N"
c) 300 PRINT N; TAB (13); "ADIOS"
d) 400 PRINT TAB (7); X : TAB (9); A$

13.2. Utiliza la instrucción **PRINT TAB** para resolver lo indicado en los apartados siguientes:

a) Imprimir la palabra MENÚ a partir de la columna 10.
b) Imprimir el número 3.1416 y el texto NÚMERO PI a partir de las columnas 5 y 15 respectivamente.

13.3. Identifica los errores contenidos en el siguiente programa:

```
10 INPUT A$
20 LET L = LEN (A$)
30 PRINT AT L, L, A$
40 IF L = 1 THEN GOTO 10
50 PRINT AT L; "FIN"
60 END
```

13.4. Haz un programa que vaya admitiendo números y tabule la pantalla para que en la columna 10 aparezcan los números que se van introduciendo y en la columna 20 el número más pequeño introducido hasta este momento.

13.5. Elabora un programa que imprima a partir de la columna 5 el nombre de una persona; a partir de la 25, su edad y a partir de la 30, su peso. Particularizar el programa para que lea los nombres, edades y pesos de seis personas, mediante **READ** y **DATA.**

13.6. Escribe un programa que lea un texto y lo imprima en diagonal, es decir, el primer carácter en la fila 0, columna 0; el segundo carácter en la fila 1, columna 1, y así sucesivamente.

13.7. Haz un programa que dibuje con asteriscos, a partir de la columna 6, la letra I, con una altura de diez puntos y con una anchura de tres.

13.8. Haz un programa que dibuje la letra A con las mismas condiciones que en el programa anterior.

13.9. Por medio de la instrucción **PRINT TAB** haz un programa que dibuje una V que ocupe toda la pantalla, imprimiendo uves.

13.10. Diseña un programa que rellene la parte inferior de la uve de la pantalla del ejercicio anterior con uves.

13.11. Realiza un programa que dibuje una M que ocupe toda la pantalla.

13.12. El siguiente programa dibuja la función seno en una pantalla con baja resolución.

```
10 FOR I = 0 TO 31
20 PRINT AT 10 – INT (10 * SIN (I * PI/15); "*"
30 NEXT I
```

Modifica este programa para que dibuje el coseno.

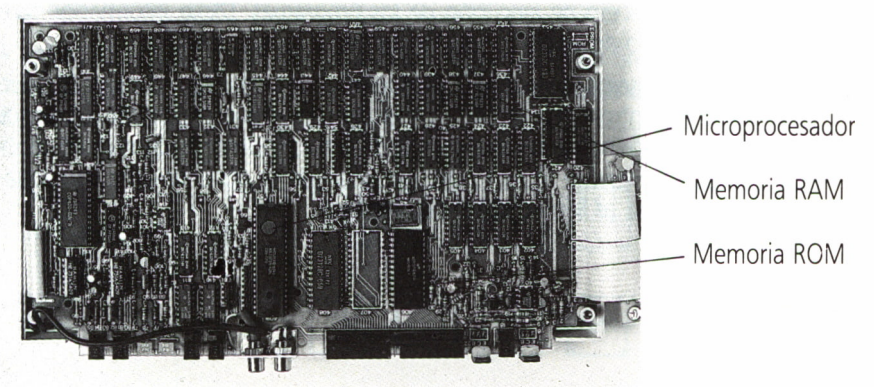
13.13. Haz un programa que dibuje la función $F(x) = x^2$ utilizando la instrucción **PRINT AT.**

13.14. Diseña un programa que por medio de la instrucción **PRINT AT** dibuje la función $F(x) = e^x$.

14. La memoria del microordenador

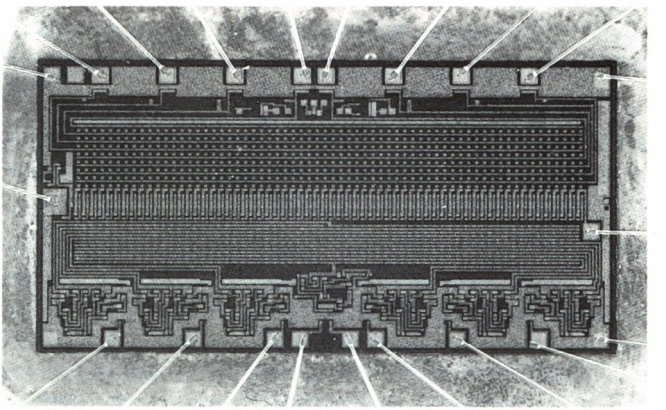
1. El microordenador por dentro

Cuando se abre la caja de un microordenador llama la atención, entre otros elementos, varias cápsulas de plástico negro con muchas patas de metal.



El microordenador por dentro

En el interior de dichas cápsulas, llamadas ***dips,*** se encuentra un trozo de silicio (semiconductor) de muy pequeño tamaño, denominado ***chip*** en el que están grabados muchos circuitos eléctricos, unidos mediante cables a las patas metálicas.



2. El microprocesador y las memorias ROM y RAM

- ***El microprocesador***

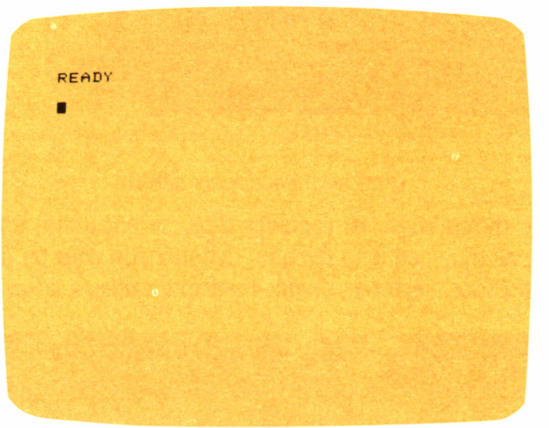
  Uno de estos chips es el ***microprocesador,*** el cual controla todas las acciones del microordenador: realiza los cálculos aritméticos y las comparaciones lógicas.

  El lenguaje que «entiende» directamente el microprocesador está constituido por impulsos eléctricos y por ausencia de los mismos, en diversas secuencias. Este lenguaje se llama ***lenguaje de máquina.***

- ***La memoria ROM***

  Otro chip importante es la memoria ROM (READ ONLY MEMORY), o ***memoria de sólo lectura.*** Esta memoria almacena un tipo de información que no se modifica en el proceso de ejecución de los programas. Dicha información sólo se puede leer, y no desaparece al desenchufar el microordenador.

  Esta porción de memoria contiene el programa que verifica si se han pulsado las teclas, que controla la impresión en la pantalla y que realiza otros trabajos vinculados al funcionamiento del microordenador; incluso, hace que aparezca en la pantalla el mensaje READY, OK, etc., cuando se conecta el microordenador. Este programa se llama ***sistema operativo.***



Muy frecuentemente, y sobre todo en el caso de los pequeños microordenadores personales, la ROM contiene el programa que traduce las instrucciones del lenguaje BASIC al lenguaje de máquina.

En el caso del BASIC, al escribir RUN y pulsar ENTER, este programa traductor (intérprete) empieza a examinar el programa escrito por el usuario. Compara las palabras que encuentra con las de su «diccionario», y si encuentra un carácter que no comprende (hecho que puede suceder al teclear), deja de interpretar el programa y da un mensaje de error. Si la palabra está incluida en el diccionario, el programa pasa a interpretar la instrucción tecleada.

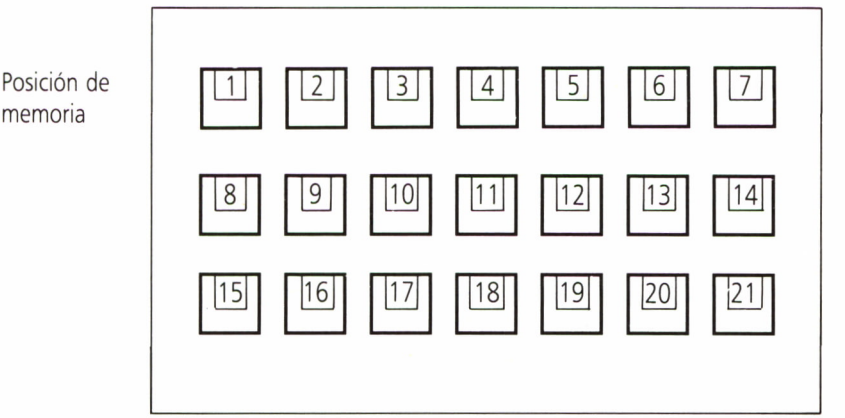
- ***La memoria RAM***

  El chip de la memoria RAM (RANDOM ACCESS MEMORY) o ***memoria de acceso aleatorio*** permite leer y almacenar información, y ésta desaparece cuando se desenchufa el microordenador.

  La RAM está organizada en zonas, diseñadas para almacenar una determinada información: el programa BASIC, la imagen que aparecerá en la pantalla, etc. Esta distribución en zonas es distinta según el microordenador, y constituye lo que se llama ***mapa de memoria*** del microordenador.

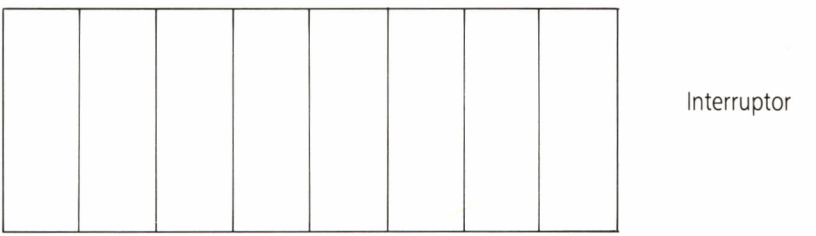
3. Cómo está organizada la memoria de un microordenador

- Las memorias ROM y RAM podemos imaginarlas como una larga lista de casillas numeradas; a cada una de ellas la llamamos ***posición de memoria.***



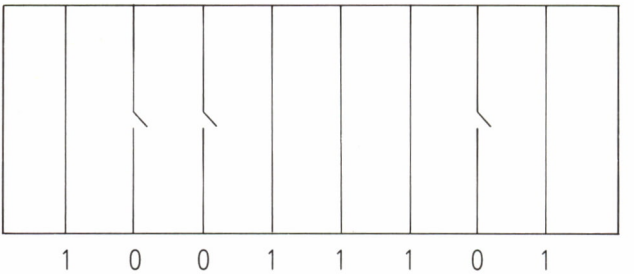
Representación de la memoria

A su vez, cada posición de memoria podemos considerarla como un conjunto de 8 interruptores colocados en fila, pudiendo estar cada uno de ellos abierto o cerrado.



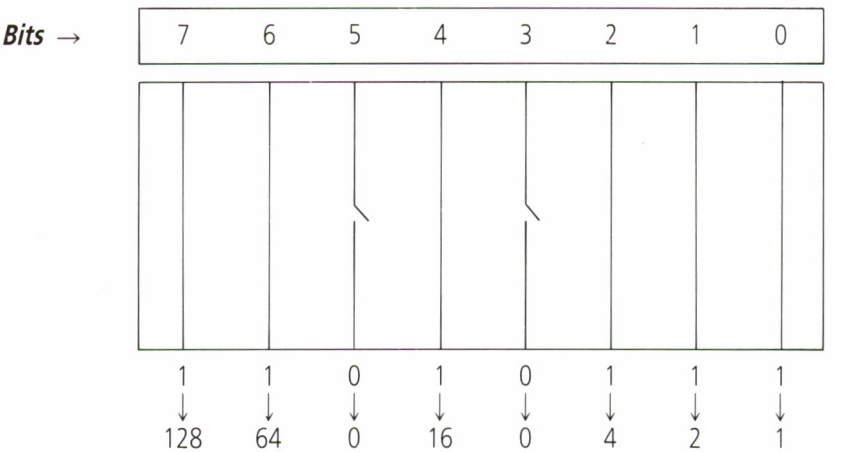
Cada posición de memoria (una casilla) se puede imaginar como un conjunto de 8 ***interruptores*** colocados en fila.

Si cada interruptor ***cerrado*** representa un 1 y cada interruptor ***abierto,*** un 0, una posición de memoria queda representada por una sucesión de 8 ceros y unos. Cada cero o uno se llama ***bit*** (dígito binario) y la sucesión de los 8 bits se denomina ***byte*** (1 byte comprende 8 bits).



*Un **byte** define una posición de memoria.*

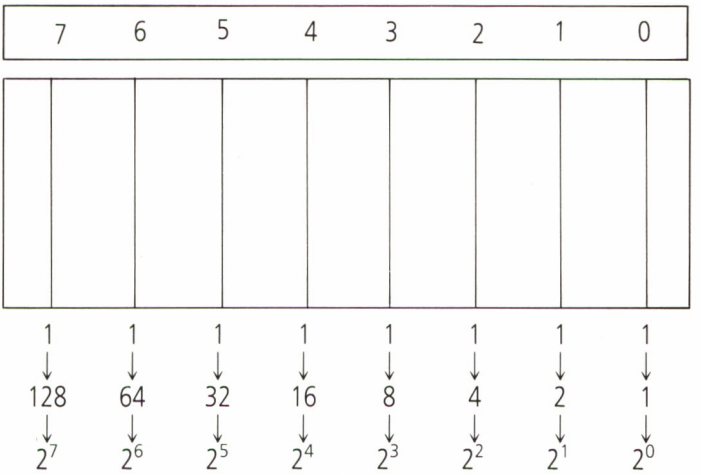
- En un byte, los bits se numeran del 0 al 7, de derecha a izquierda, y cada uno de ellos, si es 1, representa un valor doble del que corresponde al que se encuentra inmediatamente a su derecha. Así, en el byte representado en la figura siguiente,
  - el ***bit 0*** representa el valor 1
  - el ***bit 1*** representa el valor 2
  - el ***bit 2*** representa el valor 4
  - el ***bit 3*** representa el valor 0 (por estar el interruptor abierto)
  - el ***bit 4*** representa el valor 16
  - el ***bit 5*** representa el valor 0 (por estar el interruptor cerrado)
  - el ***bit 6*** representa el valor 64
  - el ***bit 7*** representa el valor 128



El valor correspondiente a un ***byte*** es el que resulta de la suma de los valores de cada uno de los bits. En este ejemplo el valor es 215:

$$128 + 64 + 0 + 16 + 4 + 2 + 1 = 215$$

- Es evidente que un byte cuyos bits son todos 1, representa el máximo valor posible.

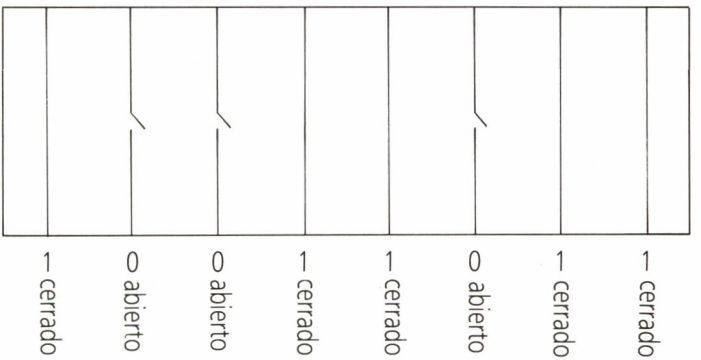


Este valor máximo es 255 y se obtiene, como en el caso anterior, sumando los valores correspondientes a cada bit:

$$128 + 61 + 32 + 16 + 8 + 1 + 2 + 1 = 255$$

Vemos que el número expresado en sistema binario por 11111111, en sistema decimal se representa por 255.

- Recíprocamente, ¿en qué estado se encuentran los interruptores correspondientes a un byte que representa el número binario 10011011? ¿Cómo se representa este valor en el sistema decimal?

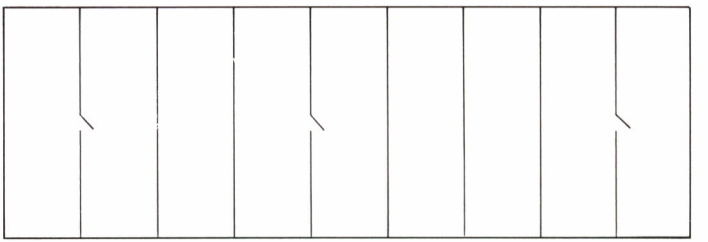


El valor de este byte es 155:

$$128 + 0 + 0 + 16 + 8 + 0 + 2 + 1 = 155$$

Ejercicios

14.1. Indica el valor del byte correspondiente a una posición de memoria cuyos interruptores se encuentran en el estado indicado en la figura.



14.2. Representa en un esquema similar al anterior el valor del byte representado por esta secuencia binaria: 10001001.

14.3. Ayudándote de un esquema con circuitos como el que venimos utilizando, representa mediante una secuencia de 8 bits el número 34.

14.4. Escribe una secuencia binaria cuyo valor corresponda a 254. En esta secuencia, ¿qué bits es cero, el último de la derecha o el último de la izquierda?

4. Capacidad de la memoria

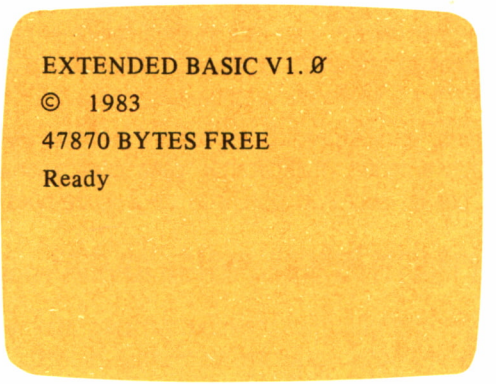
- ***La capacidad de la memoria de un microordenador viene dada por el número de bytes que contienen la ROM y la RAM.***

  La capacidad de la memoria suele expresarse en K bytes, siendo 1 K byte = 2^{10} = 1024 bytes (aproximadamente, 1000 bytes). Por ejemplo, cuando se afirma que «un microordenador es de 64 K bytes (abreviadamente, 64 K)», se quiere decir que su memoria contiene exactamente 64 × 1024 bytes = 65536 bytes (aproximadamente 64 000 bytes).

- ¿Cuáles son los bytes libres o disponibles y qué significado tiene esto desde el punto de vista del usuario?

  Los bytes libres o disponibles son parte de los de la memoria RAM, dado que los de la ROM están ocupados por el programa llamado sistema operativo y por el intérprete BASIC; unos y otros se encuentran en las especificaciones técnicas del microordenador. El número de bytes disponibles suele aparecer escrito

en la pantalla nada más conectar el microordenador, tal como muestra la fotografía.



¿Qué significa 47870 BYTES FREE?

En este microordenador de 64 K bytes (65536 bytes), ***los 47870 bytes libres*** pertenecen a la memoria **RAM;** los RESTANTES (17666 bytes) pertenecen a la parte de la RAM que utiliza el sistema operativo, y a la **ROM.**

Sabemos que, en un instante dado, un byte solo puede representar un carácter (una letra, un dígito u otro símbolo especial). Luego, ***disponer de 47870 bytes libres*** significa que el número de caracteres que componen las instrucciones y datos de un programa no puede superar dicho número, pues una vez alcanzado éste, la información que se intentara introducir en la memoria no la aceptaría el microordenador.

5. Paso del sistema binario a decimal

- El sistema de numeración que habitualmente usamos en la vida diaria es el decimal o de base 10. En este sistema se pueden representar todos los números, combinando adecuadamente diez símbolos o cifras: 0, 1, 2, ..., 9 (dígitos decimales). Según la posición que ocupen estas cifras en un número, representan distinto valor. Por ejemplo, en el número 38 035
  - — 5 vale 5 unidades = $5 * 10^0$
  - — 3 vale 30 unidades = $3 * 10^1$
  - — 0 vale 0 unidades = $0 * 10^2$
  - — 8 vale 8 000 unidades = $8 * 10^3$
  - — 3 vale 30 000 unidades = $3 * 10^4$

Luego este número se puede expresar en función de las potencias de la base 10:

$$38\,035 = 3 * 10^4 + 8 * 10^3 + 0 * 10^2 + 3 * 10^1 + 5 * 10^0$$

El sistema binario, en cambio, tiene por base 2, y en este sistema cualquier número se puede representar combinando adecuadamente solo dos símbolos o cifras: el 0 y el 1 (dígitos binarios).

Dado un número en sistema binario, para expresarlo en sistema decimal hay que proceder como en el apartado anterior, multiplicando cada cifra binaria por su valor posicional, es decir, por las sucesivas potencias de 2 y, después, sumar los valores obtenidos.

Así, para pasar el número binario 11011111 a sistema decimal hacemos estos cálculos:

$$\begin{aligned} 11011111 &= 1 * 2^7 + 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + \\ &+ 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = \\ &= 1 * 128 + 1 * 64 + 0 * 32 + 1 * 16 + \\ &+ 1 * 8 + 1 * 4 + 1 * 2 + 1 * 1 = \\ &= 128 + 64 + 0 + 16 + 8 + 4 + 2 + 1 = 223 \end{aligned}$$

- Este proceso lo puede llevar a cabo el microordenador ejecutando el siguiente programa (*).

```
 1  REM PASO DE BINARIO A DECIMAL
 5  LET N = 0 : LET P = 0
10  INPUT "INTRODUCE UNA SECUENCIA
DE UNOS Y CEROS"; A$
20  FOR I = LEN (A$) TO 1 STEP –1
30  LET B$ = MID$ (A$, LEN(A$) – P, 1)
40  LET N = N + VAL (B$) * 2 ↑ P
50  LET P = P + 1
60  NEXT I
70  PRINT A$; "EN BASE DECIMAL SE
EXPRESA ASI:"; N
80  END
```

En el *ZX Spectrum*, cambiar por:
`30 LET B$ = A$ (I TO I)`

(*) El microordenador *ZX Spectrum* posee una instrucción que pasa directamente un número binario a base decimal. Dicha instrucción se escribe **BIN**, Así

PRINT BIN 11011111

imprime en la pantalla 223.

— En el momento de la ejecución, la instrucción **10** permite introducir la ***secuencia de unos y ceros*** como cadena.

— La instrucción **30** extrae cada uno de estos dígitos binarios y los almacena temporalmente en la variable de cadena B$.

— La instrucción **40** pasa a valor numérico lo que almacena B$, multiplica por la correspondiente potencia de 2 y almacena el resultado en N. Esto lo hace para cada dígito binario, y al concluirse la ejecución del bucle **20-60,** N llega a almacenar el número en base decimal.

6. Paso del sistema decimal a binario

- Para pasar un número expresado en base decimal a base binaria se descompone el número en las sucesivas potencias de 2. Esto se consigue procediendo en forma inversa que en el apartado anterior, es decir, dividiendo por las sucesivas potencias de 2, empezando por la inmediata inferior al número dado.

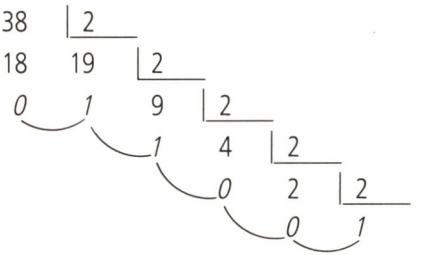
  Ejemplo: Expresar en binario el número 38.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
		1			1	1	0

Número y restos sucesivos	Potencias de 2	Cocientes
38	32	1
6	4	1
2	2	1
0	1	0

Luego, la expresión binaria del número 38 es 00100110.

El paso del sistema decimal a binario también se puede hacer aplicando la siguiente regla práctica, que consiste en dividir sucesivamente por 2.



Ahora escribimos de izquierda a derecha el último cociente, seguido de los sucesivos restos

100110

Finalmente, para obtener una secuencia de 8 dígitos, añadimos dos ceros a la izquierda

00100110

- El procedimiento seguido en la regla anterior, lo lleva a cabo el microordenador mediante el siguiente programa:

```
  5  REM PASO DE DECIMAL A BINARIO
 10  LET B$ = "   "
 20  INPUT "INTRODUCE EL NUMERO EN BASE DECIMAL"; N
 30  LET I = INT (N/2)
 40  LET B = N - 2 * I
 50  IF B = 0 THEN LET B$ = "0" + B$ : GOTO 70
 60  LET B$ = "1" + B$
 70  LET N = I
 80  IF N = 0 THEN GOTO 100
 90  GOTO 30
100  PRINT "EL NUMERO INTRODUCIDO, EN BINARIO ES: "; B$
110  END
```

Veamos cómo funciona este programa:

— Al iniciarse la ejecución, en la variable B$ se almacena un espacio en blanco.

— La instrucción **20** pide el número N, en base decimal.

— Las instrucciones **30** y **40** calculan los restos sucesivos de la división de N por 2. Ejemplo, si N = 39, en la primera división se obtiene el siguiente resto:

```
30 LET I = INT (39/2) = INT (19.5) = 19
40 LET B = 39 - 2 * 19 = 39 - 38 = 1
```

— La instrucción **50** añade por la izquierda un 0 a la cadena almacenada en B$, si se cumple la condición B = 0, y la **60** añade un 1, en caso de no cumplirse B = 0.

— La instrucción **70** guarda en N el siguiente cociente parcial (en el ejemplo es 19) con el cual se realizará otra división por 2, y así hasta que N sea igual a 0, condición que comprueba la instrucción **80.** En el caso de que la condición se cumpla se transfiere el control a la instrucción **100,** que imprime el número en binario.

Ejercicios

14.5. Halla la representación en base decimal de los siguientes bytes:

00000001

00000010

00000011

00000100

00000101

14.6. Encuentra todas las secuencias binarias comprendidas entre 11111001 y 11111111.

14.7. Haz un programa que imprima las representaciones en base decimal de los números binarios 11110001, 01101101, 01010101, 10101010, 01111110 y 10000001. (Este programa puede obtenerse mediante una sencilla modificación del transcrito en el apartado 4 de este capítulo.)

14.8. Halla la representación en binario de los números naturales 0, 1, 2, ..., 9.

14.9. Halla la representación en binario del número 248.

14.10. Modificando el programa estudiado en el apartado 5 de este capítulo, imprimir en la pantalla la representación en binario de los números 245 a 255.

7. Instrucciones PEEK y POKE

En la memoria de un microordenador todo está muy ordenado; cada una de sus posiciones («casillas») tiene asignado un número llamado ***dirección.***

El microordenador, para localizar la información almacenada en una posición de memoria busca su dirección.

- El acceso a una determinada posición de la memoria RAM o ROM se consigue mediante la instrucción **PEEK (N),** siendo N la dirección de la posición de la memoria cuyo contenido se desea averiguar.

En concreto, la instrucción **PEEK (N)** suministra lo almacenado en la posición de la memoria de dirección N.

Aplicamos esta instrucción en el siguiente programa:

```
10  INPUT "INTRODUCE UNA DIRECCION"; N
20  LET P = PEEK (N)
30  PRINT "EN LA POSICION DE DIRECCION"; N; "SE GUARDA"; P
40  GOTO 10
50  END
```

Al ejecutar este programa en un microordenador y en un momento determinado, al introducir el número 48728 obtuvimos el siguiente resultado en la pantalla:

EN LA POSICIÓN DE DIRECCIÓN 48728 SE GUARDA 32

- La instrucción **POKE** permite modificar el contenido de una determinada posición de la memoria **RAM,** almacenando en la misma el valor de un byte, o sea, un número comprendido entre 0 y 255.

Su formato es:

n **POKE** {dirección}, {número contenido}

Por ejemplo, la instrucción

35 POKE 31000, 56

almacena en la posición de memoria de dirección 31000, el número 56. Esto se puede comprobar escribiendo

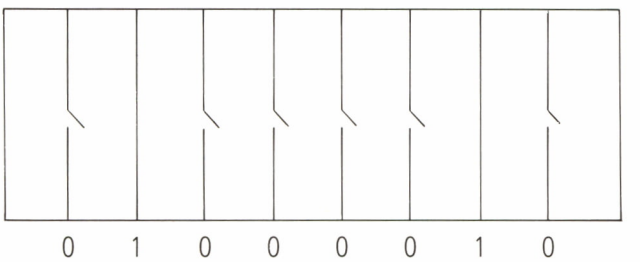
PRINT PEEK (31000)

Nada más pulsar la tecla ENTER aparecerá en la pantalla 56.

8. El conjunto de caracteres que maneja el microordenador

- Las letras, los dígitos y los signos de puntuación y de las operaciones aritméticas que pueden aparecer en una cadena, forman parte del conjunto de caracteres de que dispone el microordenador.

  Estos caracteres se pueden representar con los 8 bits que forman un byte. El procedimiento que se utiliza para ello es parecido al del código Morse, solamente que en lugar de combinarse ***puntos*** y ***rayas,*** se combinan ***unos*** y ***ceros.*** Cada ***combinación de unos y ceros*** de un byte representará un carácter. Por ejemplo, en un sistema de codificación que enseguida mencionaremos, la letra B está representada por el estado del byte correspondiente a la siguiente combinación binaria:



Un byte puede representar 256 caracteres diferentes debido a que con sus 8 bits se pueden hacer 256 combinaciones de ceros y unos (de 00000000 = 0 a 11111111 = 255).

- El sistema de codificación que utilizan la mayoría de los microordenadores es el código **ASCII** (American Standard Code for Information Interchange).

CÓDIGOS A.S.C.I.I. (decimal)

Código	Caracter	Código	Carácter
32	Space	79	O
33	!	80	P
34	"	81	Q
35	#	82	R
36	$	83	S
37	%	84	T
38	&	85	U
39	'	86	V
40	(	87	W
41	)	88	X
42	*	89	Y
43	+	90	Z
44	,	91	[
45	-	92	\
46	.	93	]
47	/	94	↑
48	0	95	£
49	1	96	©
50	2	97	a
51	3	98	b
52	4	99	c
53	5	100	d
54	6	101	e
55	7	102	f

Código	Carácter	Código	Carácter
56	8	103	g
57	9	104	h
58	:	105	i
59	;	106	j
60	<	107	k
61	=	108	l
62	>	109	m
63	?	110	n
64	@	111	o
65	A	112	p
66	B	113	q
67	C	114	r
68	D	115	s
69	E	116	t
70	F	117	u
71	G	118	v
72	H	119	w
73	I	120	x
74	J	121	y
75	K	122	z
76	L	123	{
77	M	124	|
78	N	125	}

De las 256 combinaciones diferentes de 8 bits, aquí solamente aparecen las comprendidas entre 00100000 (en base diez, 32) y 01111101 (en base diez, 125), que son comunes a todos los microordenadores.

9. Instrucciones ASC (o CODE) y CHR$

- La instrucción BASIC ASC (o CODE) imprime el número del código ASCII correspondiente al primer carácter de una cadena. Así, el programa

```
10  INPUT K$
20  PRINT "EL NUMERO ASCII DE SU PRIMER CARACTER ES"; ASC (K$)
30  GOTO 10
40  END
```

imprime en la pantalla el númedo del código **ASCII** correspondiente al primer carácter de la cadena K$. Si, por ejemplo, introducimos la cadena "BELLA", aparece en la pantalla lo siguiente:

EL NÚMERO ASCII DE SU PRIMER CARÁCTER ES 66

porque 66 es el número **ASCII** correspondiente a la letra B.

- La instrucción **CHR$** produce un efecto opuesto a **ASC** (o **CODE**). Así, el programa.

```
10  FOR N = 65 TO 90
20  PRINT CHR$ (N); "   ";
30  NEXT N
40  END
```

imprime las letras mayúsculas del alfabeto, dado que los números **ASCII** 65 a 90, corresponden a las letras A a Z, respectivamente.

Los números **ASCII** de las letras minúsculas se obtienen sumando 32 a sus correspondientes mayúsculas. Así, el número **ASCII** de a es 97 y el de z, 122.

- La correspondencia entre cada símbolo gráfico y un número del código **ASCII** permite ordenar cadenas e interpretar los signos < y > intercalados entre ellas. Así, el microordenador interpreta que

  A < B (A ***es anterior a*** B)

  porque sus correspondientes números del código **ASCII** están en esa misma relación de orden:

  65 < 66
  ↓ ↓
  A < B

  Por la misma razón se cumplen las siguientes relaciones:

  A < a y A > 9
  ↓ ↓ ↓ ↓
  65 < 97 65 > 57

Ejercicios

14.11. Explica lo que hace el siguiente programa:

```
10 FOR I = 0 TO 9
20 POKE 34230 + I, I
30 NEXT I
40 END
```

14.12. Haz un programa que almacene en posiciones de la memoria de dirección consecutiva las letras mayúsculas A, B, C, D, E y F.

14.13. Indica qué imprimiría en la pantalla el siguiente programa:

```
10 FOR I = 1 TO 5
20 READ N
30 LET P = PEEK (N)
40 PRINT P
50 NEXT I
60 DATA 42627, 43700, 43701, 43702, 44800
70 END
```

14.14. Escribe un programa que imprima en la pantalla los números del código **ASCII** correspondientes a las letras A, a, B, b, c y C.

14.15. Fijándote en los valores obtenidos en el ejercicio anterior ordena de ***anterior a posterior*** (<), las palabras siguientes: Bernardo, azucena, Antonio, abeto, canica, Claudio.

14.16. Haz un programa que imprima en la pantalla los caracteres cuyos números **ASCII** son 56, 57, 58, 59 y 60.

10. Resumen de las instrucciones estudiadas en este capítulo

- PEEK
  - — ***Significado:*** extrae ("pica").
  - — ***Formato:*** PEEK (N), siendo N una dirección.
  - — ***Función:*** extrae el contenido de la posición de memoria de dirección N.

- POKE
  - — ***Significado:*** almacena.
  - — ***Formato:*** n POKE {dirección}, {nuevo contenido}
  - — ***Función:*** almacena en la posición de memoria de la ***dirección*** indicada, el valor de un byte (un número comprendido entre 0 y 255).

- ASC (o CODE)
  - — ***Significado:*** código ASCII.
  - — ***Formato:*** ASC (A$).
  - — ***Función:*** da el número del código **ASCII** correspondiente al primer carácter de la cadena A$.

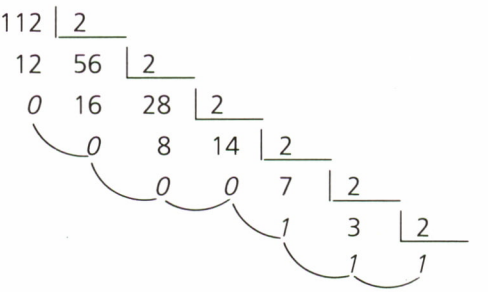
- CHR$
  - — ***Significado:*** carácter.
  - — ***Formato:*** CHR$ (N).
  - — ***Función:*** es opuesta a la que cumple la instrucción **ASC** (A$): da el símbolo que corresponde al número N del código ***ASCII.***

EJERCICIOS RESUELTOS

1. Obtener la secuencia binaria que representa a la letra p minúscula, que en el código **ASCII** le corresponde el número 112.

   Solución:

   Pasamos el número 112, escrito en base decimal, a base binaria.



Ahora escribimos de izquierda a derecha el último cociente, seguido de los sucesivos restos:

1110000

Finalmente, para obtener una secuencia de 8 bits, añadimos un cero a la izquierda:

01110000

2. Hacer un programa que imprima los números del código **ASCII** correspondientes a los números naturales, 0, 1, 2, ..., 9.

Solución:

```
10 REM NUMEROS ASCII DE 0, 1, 2, ..., 9
20 FOR I = 1 TO 10
30 READ N$
40 PRINT ASC (N$); "  ";
50 NEXT I
60 DATA "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"
70 END
```

EJERCICIOS DE RECAPITULACIÓN

14.1. Escribe lo que imprimiría el siguiente programa al ser ejecutado:

```
10 READ A$
20 FOR I = LEN (A$) TO 1 STEP –1
30 LET B$ = MID$ (A$, LEN (A$) – P, 1)
40 LET N = N + VAL (B$) * 2 ↑ P
50 LET P = P + 1
60 NEXT I
70 PRINT A$; "EQUIVALE A"; N
80 GOTO 10
90 DATA "10101010", "01010101", "11111000", "00011111"
```

14.2. El siguiente programa pasa a sistema binario, números expresados en base decimal. Explica, instrucción a instrucción, su funcionamiento.

```
10  READ N
20  FOR I = 1 TO N
30  LET B$ = "  "
40  READ A
50  LET K = INT (A/2)
60  LET B = A – 2 * K
70  IF B = 0 THEN LET B$ = "0" + B$
80  LET B$ = "1" + B$
90  LET A = K
100 IF A = 0 THEN GOTO
110 GOTO 50
120 PRINT A; "EN BINARIO ES:" B$
130 NEXT I
135 DATA 5
140 DATA 25, 125, 50, 200, 10
150 END
```

14.3. Explica qué hace la siguiente instrucción:

17 POKE 25328, 29

14.4. Introduce en la memoria los diez primeros números naturales impares en posiciones de memoria, tal que sus direcciones sean consecutivas, empezando por la de dirección 45000.

14.5. Explica qué se obtiene en la pantalla si se escribe la siguiente expresión

PRINT PEEK (45005)

14.6. Haz un programa que verifique si en las posiciones de memoria indicadas en el ejercicio 4 se encuentran almacenados los diez primeros números naturales impares.

14.7. Explica la función que cumple la instrucción **ASC** (o **CODE**).

14.8. Indica qué se obtiene en la pantalla al ejecutar el siguiente programa:

```
10  READ C$
20  LET A = ASC (C$)
30  PRINT A
40  GOTO 10
50  DATA "  ", "<", "=", ">"
60  END
```

14.9. Escribe un programa que halle el número del código **ASCII** correspondiente a las letras l y L, y según el resultado obtenido después de ejecutar el programa, indica qué símbolo pondrías entre las siguientes palabras:

Luis ... laberinto

14.10. Indica qué se obtiene en la pantalla cuando se ejecuta el siguiente programa. (Consulta el código **ASCII** en el apartado 7 de este capítulo).

```
10 READ N
20 PRINT "EL CARACTER ASCII QUE CORRESPONDE
AL NUMERO"; N; "ES"; CHR$ (N)
30 DATA 100
40 END
```

14.11. Realiza un programa que haga aparecer en la pantalla los caracteres del código **ASCII** correspondientes a los números 33 a 125, dispuestos en la siguiente forma:

NÚMERO	CARÁCTER
33	.
.	.
.	.
.	.

Apéndice A

GRABACIÓN Y REPRODUCCIÓN DE PROGRAMAS EN CASETE

Un programa que se ha introducido en la memoria RAM (Random Access Memory) de un microordenador se pierde cuando éste se desconecta. Sin embargo, este programa puede conservarse utilizando un dispositivo de memoria externa.

El sistema de almacenamiento de programas más barato es el que se realiza en cintas magnéticas con ayuda de un ***grabador-reproductor*** de casetes. Este sistema, aunque no es tan fiable y rápido como el basado en discos, resulta satisfactorio en muchas ocasiones.

Grabación de programas

Para grabar un programa en una casete hay que llevar a cabo las siguientes acciones:

1.ª Conectar correctamente el microordenador a las salidas MIC y EAR* del aparato grabador-reproductor de casetes.

2.ª Rebobinar la cinta hasta la zona de la casete, a partir de la cual se desea iniciar la grabación del programa. Si la cinta está totalmente rebobinada, hay que tener la precaución de no empezar la grabación hasta que no se haya sobrepasado la parte de cinta transparente; esto se consigue dejándola correr unos segundos.

3.ª Teclear la palabra **SAVE** (**C SAVE** en algunos microodenadores) y a continuación el nombre entre comillas que se desea asignar al programa. El número de caracteres del nombre del programa no debe sobrepasar un valor máximo; éste depende del microordenador (consultar el manual).

4.ª Pulsar la tecla RECORD y PLAY del aparato grabador-reproductor de casetes.

5.ª Pulsar en el teclado del microordenador [ENTER] o [RETURN] u otra equivalente, según el microordenador.

Una vez cumplido este último paso se inicia la grabación del programa. Ésta termina cuando aparece en la pantalla la palabra READY, OK, u otra equivalente según el microordenador. En este instante se acciona la tecla **STOP** del grabador-reproductor para que la cinta no siga avanzando.

Si todo ha funcionado correctamente, en la casete se obtiene una copia del programa con el nombre asignado.

(*) En el microordenador ZX Spectrum, antes de proceder a grabar hay que desconectar el terminal que lleva incorporado.

Ejemplo:

En la memoria del microordenador se encuentra almacenado un programa. Guardar este programa en una casete con el nombre PROGRAMA 1.

Suponiendo que el cable que une el microordenador y el aparato grabador-reproductor de casetes está correctamente conectado, el proceso a seguir es éste:

- Se coloca la casete en el aparato grabador-reproductor.
- Se teclea en el microordenador

  SAVE "PROGRAMA 1"

  No pulsar todavía la tecla ENTER
- Accionar las teclas RECORD y PLAY del aparato grabador-reproductor.
- Pulsar ahora la tecla ENTER. Se inicia, entonces, la grabación.

No olvidarse de accionar la tecla STOP del grabador-reproductor después que aparezca en la pantalla READY, OK, etc, para que no siga avanzando la cinta.

Reproducción de programas grabados en casete

Para pasar a la memoria del microordenador un programa que se encuentra grabado en una casete hay que realizar las siguientes acciones:

1.ª Colocar la cinta en el aparato grabador-reproductor en una zona anterior y próxima a la que se encuentra grabado el programa. (Si el aparato tiene un ***cuentavueltas*** la localización del programa en la casete evita cualquier tipo de tanteos en su búsqueda.)

2.ª Teclear LOAD (C LOAD en algunos microordenadores) y a continuación, entre comillas, el nombre del programa que se desea reproducir. Escribiendo el nombre, se consigue que el microordenador ignore otros programas que pudieran estar grabados en la casete.

3.ª Pulsar la tecla ENTER u otra palabra equivalente, según el microordenador.

4.ª Pulsar la tecla PLAY en el aparato grabador-reproductor.

Cumplida esta última acción, la cinta empieza a avanzar, pero el microordenador no introducirá nada en su memoria hasta que no encuentre el nombre del programa tecleado en la acción 2.ª.

Concluida la «carga» del programa en la memoria del microordenador, aparece en la pantalla la palabra READY, OK u otra equivalente. En este momento, accionar la tecla STOP del aparato grabador-reproductor para que la cinta no siga avanzando.

Ejemplo:

Cargar en la memoria del microordenador el programa grabado anteriormente en la casete.

- Colocar la cinta en el aparato grabador-reproductor.

- Teclear en el microordenador

  LOAD "PROGRAMA 1" [ENTER]
- Pulsar la tecla PLAY del aparato grabador-reproductor. Se inicia, entonces, la "carga" del programa.

No olvidarse de accionar la tecla **STOP** del grabador-reproductor después que aparezca en la pantalla READ, OK, etc. para que no siga avanzado la cinta.

Si en la acción 2.ª se omite el nombre del programa, la instrucción LOAD " " carga el primer programa que encuentre.

Otras posibilidades

Todos los microordenadores poseen las instrucciones **SAVE (o C SAVE)** y **LOAD (o C LOAD).** Sin embargo, la mayoría de los microordenadores poseen otras posibilidades en el proceso de grabación-reproducción; algunas de ellas pasamos a considerar.

- ***VERIFICAR un programa que se acaba de grabar.*** La verificación consiste en comparar el programa grabado en la casete y el almacenado en la memoria del microordenador. Este proceso permite comprobar si ambos son iguales.
- ***UNIR un programa grabado en una casete con otro que se encuentra en la memoria del microordenador.*** *En este proceso hay que tener cuidado de que los números de instrucciones de ambos programas no coincidan, ya que en caso contrario las instrucciones del primer programa con idéntico número se perderían.*
- ***HACER que un programa que se acaba de cargar en la memoria del microordenador se ejecute automáticamente, sin necesidad de escribir la instrucción*** RUN.
- ***GRABAR datos numéricos o de cadenas alfanuméricas que posteriormente se utilizarán en la ejecución de un programa*** **(tratamiento secuencial de archivos).**

Apéndice B

PALABRAS BASIC ESTUDIADAS EN ESTE LIBRO

Palabras BASIC	*Función que realiza*	*Página del libro*
ABS (X)	Da el valor absoluto del número X	145
ASC (X$)	Da el número de código ASCII correspondiente al primer carácter de la cadena X$.	210
AT F, C	Sitúa el cursor en la posición correspondiente a la fila F y a la columna C.	183
ATN (X)	Calcula el arco cuya tangente es X. (El arco calculado se expresa en radianes.)	155
CHR$ (X)	Da el carácter del código ASCII correspondiente al número X. (X es un entero comprendido entre 32 y 128.)	210
CLS	Borra lo escrito en la pantalla, pero no lo almacenado en la memoria del microordenador.	26
C LOAD "XXX" (o LOAD "XXX")	Carga en la memoria del microordenador un programa grabado en una casete con el nombre XXX	216
CONT	Continúa la ejecución de un programa interrumpida por la instrucción **STOP.**	62
CODE (X$)	Da el número del código ASCII correspondiente al primer carácter de la cadena X$.	210
COS (X)	Calcula el coseno del ángulo X, expresado en radianes.	155
C SAVE "XXX" (o SAVE "XXX")	Graba en una casete con el nombre XXX un programa almacenado en la memoria del microordenador.	216

DATA	Almacena los datos que serán leídos por una instrucción **READ.**	69
DEF FNA (X)	Define una función numérica de nombre FNA y de variable X. (Para definir otras funciones cambiar la letra A por otras del abecedario.)	155
DIM L(N) DIM T(N,M)	Dimensiona una variable de un índice. Dimensiona una variable de dos índices.	114 125
END	Da por finalizado un programa.	20
ENTER	Hace que el dato o instrucción tecleados se almacene en la memoria.	20
EXP (X)	Calcula la potencia e^x, siendo e = 2.71828	155
FNA (X)	Da el valor de la función de nombre FNA previamente definida, para el valor X.	155
FOR V=i TO t STEP s	Ejecuta un conjunto de instrucciones, llamado ***bucle,*** que empieza con la instrucción FOR y termina con una instrucción NEXT. (Variable: V; valor inicial de V, i; tope t; incremento, s.)	78
GOSUB n	Transfiere el control de la ejecución de una subrutina que empieza en una instrucción numerada con n.	164
GOTO n	Transfiere el control de la ejecución a la instrucción del programa numerado con n. Esta instrucción puede ser anterior o posterior a n.	48
IF (condición) THEN (instrucciones)	Si se cumple la ***condición*** entonces se ejecutan las instrucciones que siguen a THEN; y si no se cumple se ejecuta la instrucción escrita en la línea siguiente.	51
INPUT A INPUT A$	Permite introducir datos numéricos o alfanuméricos, los cuales se asignan a la variable A o A$, respectivamente.	33 94

INT (X)	Calcula la parte entera del número X.	145
LEFT $ (A$,N)	Extrae los N primeros caracteres de la cadena almacenada en A$.	100
LEN (A$)	Da el número de caracteres de la cadena almacenada en A$.	97
LET variable = = expresión	Calcula el valor de la expresión y lo asigna a la variable.	20
LIST LIST n LIST n_1-n_2 LIST n –	Hace aparecer en la pantalla: — Todo el listado del programa almacenado en la memoria. — La instrucción n. — Las instrucciones comprendidas entre n_1 y n_2, ambas inclusive. — Las instrucciones comprendidas entre n y la última, ambas inclusive.	24
LN (X)	Calcula el logaritmo natural o neperiano de X.	155
LOCATE C, F	Sitúa el cursor en la posición correspondiente a la columna C y a la fila F.	186
LOG (X)	Calcula el logaritmo decimal de X.	155
MID $ (A$,P,N)	Extrae N caracteres de la cadena almacenada en A$ desde la posición P, siguiendo hacia la derecha.	100
NEW LINE	Hace que el dato o instrucción tecleados se almacenen en la memoria.	20
NEW	Borra todo el contenido de la memoria.	26
NEXT V	Transfiere la ejecución del programa a la instrucción FOR del bucle siempre que el valor de V sea menor o igual que el tope del bucle.	79
ON V GOSUB n_1... n_k	Transfiere el control de la ejecución a las subrutinas que se inician con las instrucciones n_1, n_2, ..., n_k, según V valga 1, 2, ..., k.	169

ON V GOTO n_1... n_k	Transfiere el control de la ejecución a las instrucciones n_1, n_2, ..., n_k del programa según V valga 1, 2, ..., k, respectivamente.	169
PAUSE N	Detiene la ejecución del programa y retiene la imagen un tiempo que depende de N.	105
PEEK (X)	Da el contenido de la posición de la memoria de dirección X.	206
POKE N, V	Almacena el valor de V en la posición de la memoria de dirección N, (V es un entero comprendido entre 0 y 255).	206
PRINT	Escribe o imprime el valor de la variable, el resultado de operaciones aritméticas y texto entrecomillado.	36
RANDOMIZE y RND	Da un número aleatorio entre 0 y 1.	150
READ A, B, ..., X	Asigna a las variables A, B, ..., X los datos almacenados en una instrucción DATA, en forma secuencial.	69
REM	Permite añadir aclaraciones o comentarios en un programa sin afectar su ejecución.	35
RESTORE	Permite volver a leer desde el principio de los datos contenidos en las instrucciones DATA.	72
RETURN	Hace que el dato o instrucción tecleado se almacene en la memoria.	20
RETURN	Última instrucción de una subrutina, cuya función es transferir el control de la ejecución siguiente a la GOSUB del programa principal.	164
RIGHT$ (A$,N)	Extrae los N últimos caracteres de la cadena almacenada en A$.	100

RND (A)	Calcula un número aleatorio entre 0 y 1 (A es un argumento, al que habitualmente se asigna el valor 1.)	149
RUN	Inicia la ejecución del programa. ("Correr el programa".)	21
SGN (X)	Calcula el signo de X (+1, si X > 0; 0 si X = 0; –1 si X < 0).	145
SIN (X)	Calcula el seno del ángulo X, expresado en radianes.	155
SQR (X)	Calcula la raíz cuadrada de X.	155
STOP	Detiene la ejecución del programa justo en la línea en que se encuentra STOP.	61
STR$(N)	Convierte el número almacenado en N en una cadena numérica.	98
TAB (N)	Mueve la posición del cursor a la columna N.	179
TAN (X)	Calcula la tangente de X, expresado en radianes.	155
TO	A$ (N TO M): estrae una subcadena de A$ desde la posición N a la M.	102
VAL (N$)	Convierte la cadena numérica almacenada en N$ en su valor numérico.	98
WAIT (N)	Detiene la ejecución del programa y retiene la imagen de la pantalla un tiempo, que depende de N.	105

COLECCIÓN BASIC

Basic Programación

Gráficos, Colores y Música en el ZX Spectrum

MSX. Programación con Gráficos, Colores y Música

Programas resueltos en BASIC

Aplicaciones en MSX

distribuidor exclusivo
cesma sa
C/ Aguacate, 25
28044 MADRID