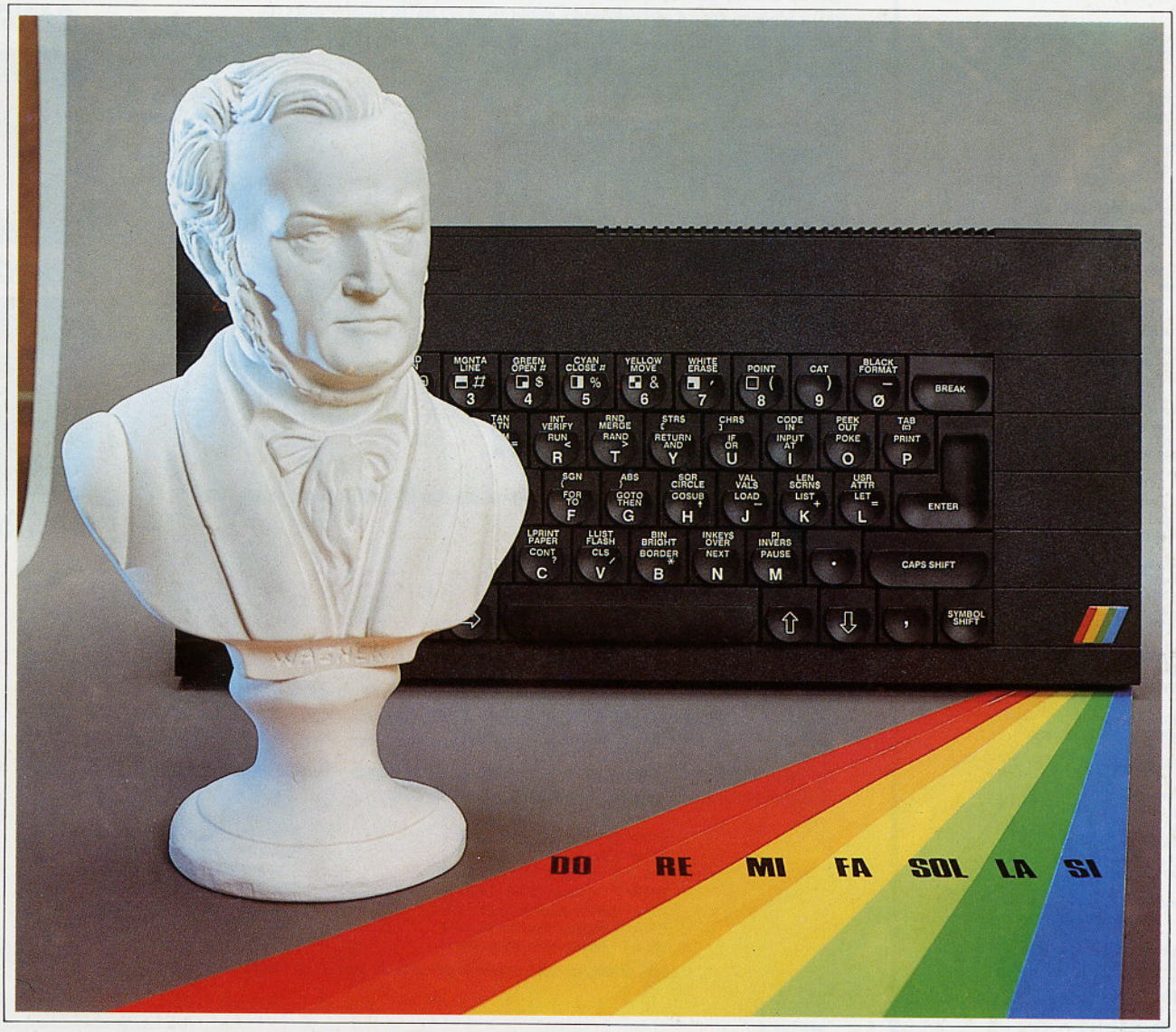
16
150pts.

RUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

VARIABLES SUSCRITAS

RECORDEMOS que hasta ahora hemos hablado de dos tipos de variables, las numéricas y las de cadena o alfanuméricas; siendo el empleo de un tipo de variable u otro discrecional, en función de las características del dato que deba albergar. Las diferencias entre ambos tipos de variables son considerables, y afectan tanto a la forma como al fondo.

El nombre de una variable numérica es una serie de caracteres, y su contenido cualquier número, natural, entero o real, pudiendo efectuarse con ellas cálculos matemáticos, y establecer comparaciones; el nombre de una variable de cadena está formado por un solo carácter seguido del símbolo dólar **($)**, pudiendo ser su contenido cualquier serie de caracteres, numéricos o alfabéticos, y permitiendo operaciones de unión y troceado, además de comparaciones.

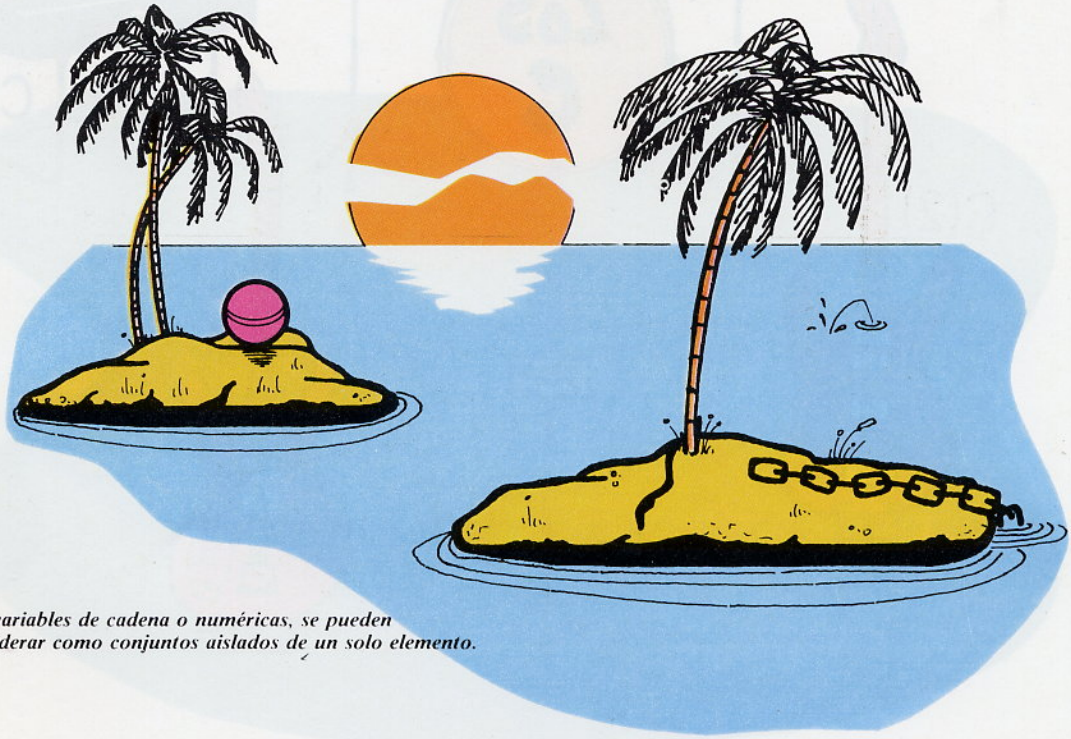
Por la misma forma en que definimos las variables (diferente nombre identificador), cada una de ellas es independiente de las otras, pudiéndolas considerar como conjuntos aislados de un único elemento. Sin embargo, se plantea el problema de definir grupos de variables que nos permitan una doble posibilidad de tratamiento: una individual y otra de «conjunto».

Normalmente, esta situación se produce cuando manejamos un cierto volumen de datos relacionados entre sí. Supongamos, por ejemplo, que queremos averiguar la media aritmética de las notas obtenidas por un alumno, a lo largo de los 10 meses de curso, imprimiendo al final un informe con cada una de las notas y la media del curso.

Para ello, debemos almacenar primero las calificaciones en variables por medio de la sentencia **INPUT.** Podríamos hacerlo pidiendo línea por línea cada una de las calificaciones, y almacenándolas en variables del tipo **N1, N2, ... N10,** pero esta estructura no nos permitiría hacer uso del bucle **FOR NEXT,** y nuestro programa requeriría 10 líneas de **INPUT** y una considerable pérdida de tiempo en la codificación:

```
10 INPUT "NOTA 1";N1
20 INPUT "NOTA 2";N2
   . . .
100 INPUT "NOTA 10";N10
```

Ya hemos visto algunas de las ventajas que se desprenden del uso de los bucles **FOR NEXT.** Sin embargo, nos hemos dejado quizás la más importante: la posibilidad de tratamiento de los elementos de un conjunto.



Las variables de cadena o numéricas, se pueden considerar como conjuntos aislados de un solo elemento.

¡!

Si se intenta asignar a un elemento de una variable suscrita de cadena, un valor de longitud mayor a la preestablecida, éste se trunca, tomándose los *n* primeros caracteres (siendo *n* la longitud máxima), pero no se produce ninguna detención por error.

*

Las matrices de dos dimensiones se suelen conocer como TABLAS.

*

Si se asigna a un elemento de una variable suscrita de cadena, un valor de longitud menor a la preestablecida, éste es completado con espacios en blanco por la derecha, hasta alcanzar dicha longitud.

¡!

Cuando se efectúa un **DIM**, desaparece cualquier variable suscrita de igual nombre que pudiera existir en el momento de la definición, pasando a adquirir la matriz las nuevas características señaladas, como si nunca hubiera existido la variable suscrita anterior.

*

Al definir una variable suscrita numérica, todos sus elementos toman inicialmente el valor cero.

*

Al definir una variable suscrita de cadena, todos sus elementos toman el valor inicial de cadenas de espacios en blanco de la longitud preestablecida.

*

Las variables suscritas, son generalmente conocidas por el nombre de MATRICES.

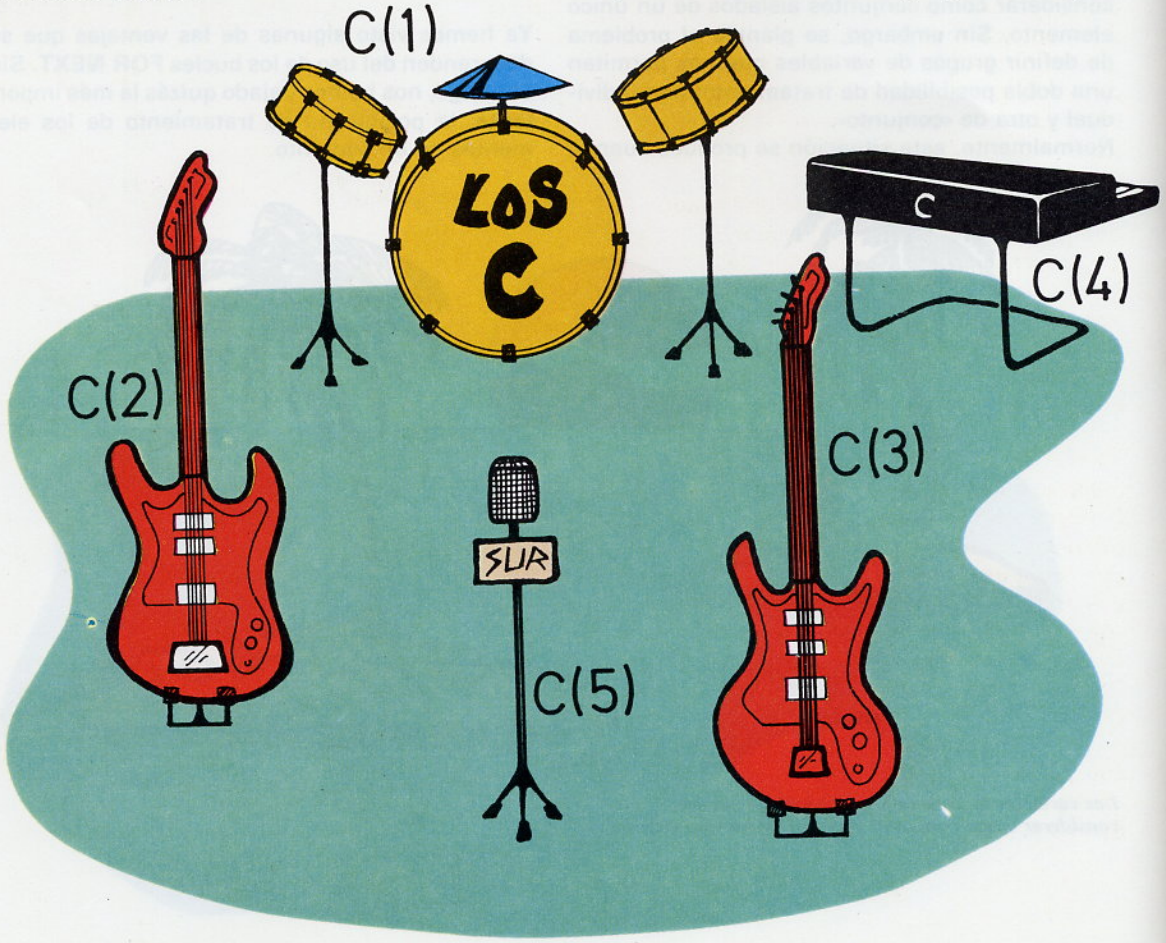
CONJUNTOS DE VARIABLES

No cabe duda que la solución ideal a nuestros problemas es definir un conjunto de variables que se denomina «matriz». Esta matriz va a ser de tipo numérico y su nombre genérico será **N( )**, refiriéndose a cada elemento en particular a través de un «índice», que colocaremos entre paréntesis a continuación del nombre de la matriz.

A modo general, podemos decir que una matriz es un conjunto de elementos de un mismo tipo, a los que nos referimos por medio del nombre del conjunto, seguido de un índice que identifica a cada elemento en particular. Este índice designa a un elemento determinado por su posición relativa frente a los demás, de forma que todos ellos guardan dentro del conjunto un orden determinado que impone el programador.

En el caso concreto de nuestro ejemplo, podríamos llamar N(1) a la nota del primer mes, N(2) a la del segundo, etc... hasta N(10), para el último de los meses.

El BASIC permite tratar determinadas variables como parte de un conjunto.



DEFINICION DE MATRICES

Para que el BASIC pueda manejar una matriz, es necesario que la hayamos definido previamente. Esta definición consiste en determinar, a través de la sentencia BASIC **DIM**, el tipo y número de elementos de que se compone, así como su estructura interna. Se trata pues, de una definición algo más compleja que la que debemos hacer cuando nos referimos por primera vez a un nombre de variable, dentro de un programa, a través de la sentencia **LET**.

La citada operación de definición se conoce como DIMENSIONADO, y produce en el ordenador una reserva del espacio suficiente de memoria para contener la variable; de ahí que en algunas ocasiones, la ejecución de estas sentencias conduzca a un error del tipo **4 Out of memory.**

Respecto a la denominación que pueden adoptar

En el programa de ejemplo, el conjunto N( ) agrupa las calificaciones medias obtenidas por un alumno durante los diez meses del curso.

estos conjuntos de variables, son evidentemente diferentes según tengan un contenido numérico o de cadena. En el primer caso, el nombre estará formado por una soia letra, que siempre irá seguida de los índices de referencia del elemento, encerrados entre paréntesis.

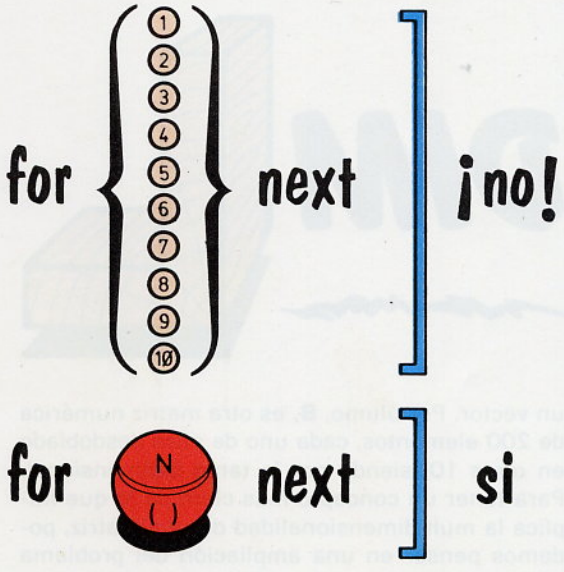
Tratándose de conjuntos de cadena, habremos de añadir a la norma anterior el consabido símbolo dólar **($)**, antes del índice entre paréntesis. En lo referente a este tipo de variables suscritas, existe una restricción muy importante: nunca podemos tener en un mismo momento una variable de cadena y de conjunto de cadenas, que se llamen igual. La razón está bien clara; como ya sabemos, nuestro Spectrum, y en general cualquier artefacto informático, aborrece las ambigüedades, y puesto que el troceado de cadenas se indica encerrando sus parámetros entre paréntesis, podría darse la confusión entre la referencia a un troceado de cadena, o a un índice de variable suscrita. Veamos un ejemplo.

Si suponemos la existencia de una cadena **A$**, y una matriz **A$( )**, puede que queramos conocer el carácter tercero de la cadena **A$**, para lo cual ejecutaríamos **PRINT A$(3)**; sin embargo, idéntica instrucción emplearíamos para conocer el elemento tercero del conjunto **A$( )** (como variable suscrita). Así pues; el ordenador no puede permitir tal confusión en sus electrónicas «neuronas», y se ve obligado a prohibirnos expresamente la utilización de variables de cadena y suscritas que tengan un mismo nombre.

Por otra parte, dado que como denominación de matrices, ya sean numéricas o alfanuméricas, se puede emplear una sola letra, y puesto que, como es habitual en las variables, no existe diferencia alguna entre mayúsculas o minúsculas, el número máximo de variables suscritas de cada tipo es 26 (de la A a la Z); aunque esto no debe resultar un inconveniente insalvable en ningún caso, dado que la posibilidad de manejar múltiples dimensiones con cada matriz, garantiza prácticamente que nunca echaremos en falta variables de este tipo.

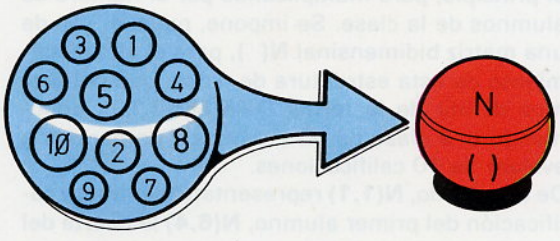
Las matrices alfanuméricas o de cadena, alcanzan usualmente una dimensión. Su definición consiste en indicar, a través de la sentencia BASIC **DIM**, su nombre y, entre paréntesis, su número de elementos y longitud del mayor de ellos, separados por comas. A este tipo de matrices unidimensionales, ya sean numéricas o alfanuméricas, se las denomina también VECTORES o LISTAS, puesto que su desarrollo es lineal. Un vivo ejemplo de matriz unidimensional o vector, es el propuesto anteriormente para la obtención de la nota media en una asignatura. La matriz **N( )** es unidimensional porque, para localizar a un elemento determinado dentro de ella, precisamos de un solo índice, en nuestro caso, la posición relativa de la nota buscada frente a las demás.

Las matrices numéricas admiten más de una dimensión, pero sin embargo no es necesario referenciar la longitud de sus elementos, puesto que



El tratamiento por* FOR-NEXT *de variables diferentes, no es posible; sin embargo, sí es válido el de una misma variable suscrita.

Un conjunto de variables con algún denominador común, pueden reunirse en una sola variable suscrita.



¡!

Las matrices de una sola dimensión se denominan comúnmente LISTAS o VECTORES.

*

Las variables suscritas son conjuntos de variables con un mismo nombre genérico, distinguidas por el ordenador mediante un número entre paréntesis, denominado índice.

*

Las variables suscritas pueden ser numéricas o alfanuméricas.

*

La denominación general de una variable suscrita numérica, debe ser una sola letra; como en el caso de las variables individuales, no existe distinción entre mayúsculas y minúsculas, por lo cual el máximo número de variables suscritas diferentes es 26 (de la A a la Z).

todas las variables numéricas BASIC ocupan lo mismo, con independencia de su contenido. Esto que en un principio nos puede parecer algo difícil de comprender, no nos debe preocupar ahora: es simplemente un hecho, que bastante más adelante estudiaremos.
De esta forma, una matriz numérica queda definida por una sentencia **DIM**, al indicar su nombre y, a continuación y entre paréntesis, su número de elementos y dimensión, separados por una coma. A continuación se muestran algunos ejemplos de definición de diversas matrices:

10 DIM A$(10,20): DIM A(200): DIM B(200,10)

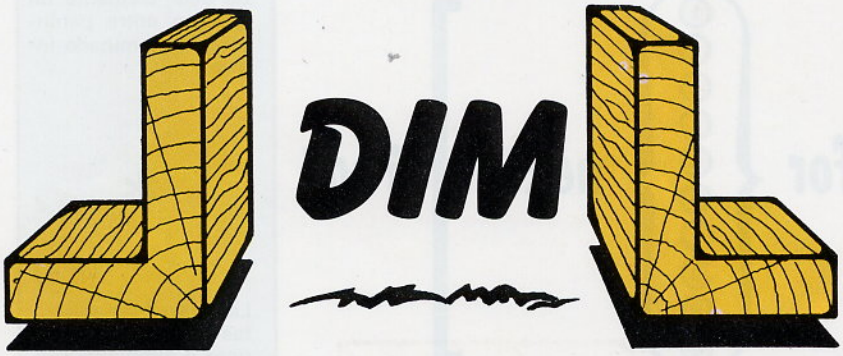
En esta línea de programa se han dimensionado tres matrices. La primera de ellas, **A$**, es de cadena y consta de 10 elementos de longitud máxima 20. La segunda, **A**, es numérica y consta de 200 elementos, en una dimensión, es decir, es un vector. Por último, **B**, es otra matriz numérica de 200 elementos, cada uno de ellos desdoblado en otros 10, siendo por lo tanto bidimensional.
Para tener un concepto más claro de lo que implica la multidimensionalidad de una matriz, podemos pensar en una ampliación del problema propuesto anteriormente. Supongamos que, en vez de calcular la nota media de las calificaciones de un alumno, deseamos obtener además las de toda la clase.
En este caso, es evidente que podríamos aprovechar gran parte del programa anterior. De hecho, nuestro problema es el mismo que el propuesto al principio, pero multiplicando por el número de alumnos de la clase. Se impone, pues, el uso de una matriz bidimensinal **N( )**, para el almacenamiento de esta estructura de datos, que deberemos definir de la forma **DIM N(50,10)**, suponiendo una clase de 50 alumnos y un período a evaluar de 10 calificaciones.
De este modo, **N(1,1)** representará la primera calificación del primer alumno, **N(6,4)** la cuarta del sexto alumno y, por último, **N(50,10)** la décima del que hace el número cincuenta.
Si ampliamos aún más nuestro ejemplo sobre las notas de los alumnos, haciéndolo extensivo a todas las clases que componen el curso, nos encontramos ante el caso de una matriz tridimensional, que podríamos representar como:

DIM N(8,50,10)

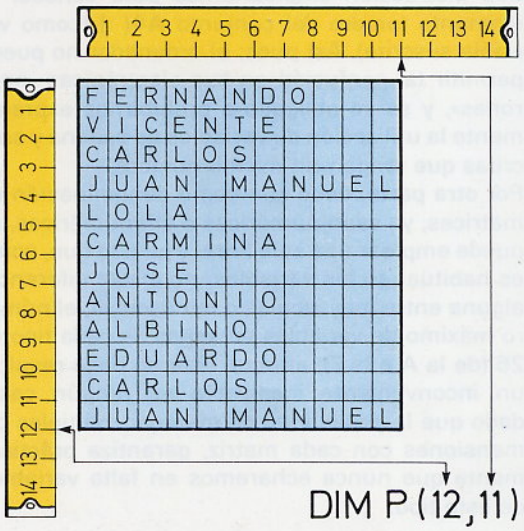
En este caso, sería preciso el empleo de 3 índices para calificar a un elemento determinado, es decir, el número de aula, el número de alumno y de qué calificación se trata.

Cada uno de los elementos del conjunto, se distingue de los demás por su índice entre paréntesis.

La sentencia **DIM** *se utiliza para reservar espacio de memoria a una variable suscrita.*

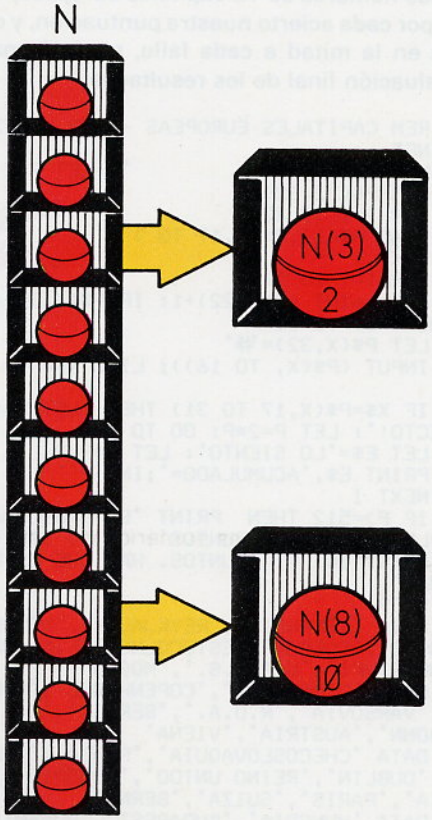


Las variables suscritas alfanuméricas, se declaran indicando el número de elementos que las componen y su longitud máxima.



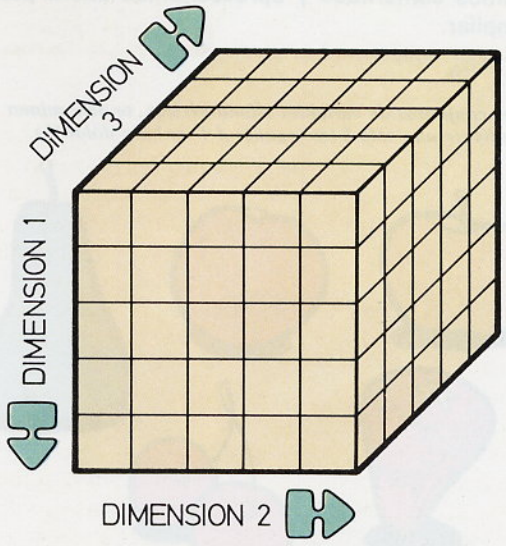
CONTENIDO DE LAS MATRICES

Las matrices, teóricamente, deben definirse una sola vez en el programa. Han de colocarse, por ello, en un lugar concreto del programa por el que sólo se pase una vez. Expliquemos el porqué de ese «teóricamente» del comienzo de párrafo. Cuando una matriz ha sido definida y contiene datos, si se efectúa una nueva definición de la misma matriz a través de otra sentencia **DIM,** se pierde su contenido, adoptando la variable suscrita la nueva configuración indicada, y su contenido pasa a ser ceros si es numérica, y blancos si es de cadena.
Este efecto que «teóricamente» no es deseable, puede ser utilizado por nosotros mismos, a nuestra conveniencia, puesto que el «redimensionado» no producirá ninguna detención por error, sino simplemente el efecto descrito. En todo caso, lo habitual es no desear el borrado súbito del contenido de una matriz, debido a lo cual, deberemos tener cuidado con el lugar en que situamos las definiciones.



En las matrices numéricas, todos los elementos ocupan la misma memoria, independientemente de su contenido.

Las variables suscritas pueden alcanzar más de dos dimensiones, en cuyo caso se las suele conocer como MATRICES DE n DIMENSIONES.

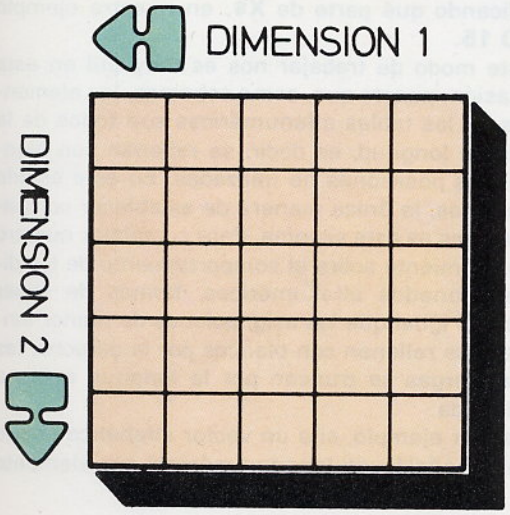


Las matrices bidimensionales, se conocen como TABLAS.

Las variables suscritas unidimensionales, se denominan LISTAS o VECTORES.

UN EJEMPLO

Para aclarar ideas, proponemos el siguiente programa de ejemplo, que nos pregunta aleatoria-



¡!

La denominación general de las variables suscritas alfanuméricas (de cadena), es una sola letra seguida del símbolo dólar **($)**; lógicamente, el máximo número de variables de este tipo es también 26.

*

Para evitar confusiones entre índices y parámetros de *slicing* (troceado de cadenas), no es posible hacer coexistir en la memoria una variable suscrita de cadena y una cadena individual, con el mismo nombre.

*

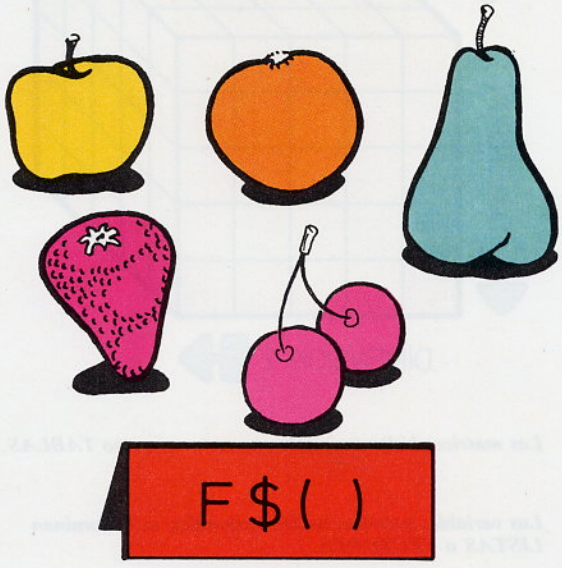
La definición inicial de las variables suscritas, se lleva a cabo mediante la sentencia **DIM**, en la cual se indican las características de la misma: nombre, y por tanto, tipo (numérica o alfanumérica), longitud y número de dimensiones.

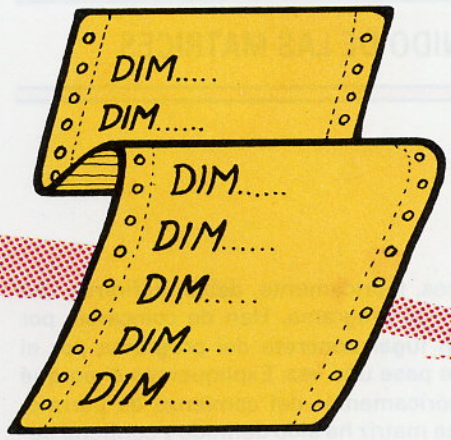
mente los nombres de 10 capitales europeas, doblando por cada acierto nuestra puntuación, y dejándola en la mitad a cada fallo, para alcanzar una evaluación final de los resultados:

```
  10 REM CAPITALES EUROPEAS - J.M.LOPEZ
MARTINEZ
  20 LET X$="                    ": LET P=1
  30 DIM P$(22,32)
  40 FOR I=1 TO 22
  50 READ P$(I),P$(I,17 TO )
  60 NEXT I
  70 FOR I=1 TO 10
  80 LET X=INT (RND*22)+1: IF P$(X,32)="
#" THEN  GO TO 80
  90 LET P$(X,32)="#"
 100 INPUT (P$(X, TO 16)); LINE X$( TO 1
5)
 110 IF X$=P$(X,17 TO 31) THEN  LET E$="
CORRECTO!": LET P=2*P: GO TO 130
 120 LET E$="LO SIENTO": LET P=P/2
 130 PRINT E$,"ACUMULADO=";INT P''
 140 NEXT I
 150 IF P>=512 THEN  PRINT "ENHORABUENA!
 POR LOS ";INT P;" PUNTOS": GO TO 170
 160 PRINT INT P;" PUNTOS. HAY QUE ESTUD
IAR MAS"
 170 STOP
 180 DATA "ISLANDIA","REYKJAVIK","NORUEG
A","OSLO","SUECIA","ESTOCOLMO","FINLANDI
A","HELSINKI","U.R.S.S.","MOSCU"
 190 DATA "DINAMARCA","COPENHAGUE","POLO
NIA","VARSOVIA","R.D.A.","BERLIN","R.F.A
.","BONN","AUSTRIA","VIENA"
 200 DATA "CHECOSLOVAQUIA","PRAGA","IRLA
NDA","DUBLIN","REINO UNIDO","LONDRES","F
RANCIA","PARIS","SUIZA","BERNA"
 210 DATA "HUNGRIA","BUDAPEST","RUMANIA"
,"BUCAREST","YUGOSLAVIA","BELGRADO","POR
TUGAL","LISBOA","ESPANA","MADRID"
 220 DATA "ITALIA","ROMA","GRECIA","ATEN
AS"
```

Este programa tiene de todo. Entre otras cosas, algunos conceptos nuevos sobre temas que ya hemos comentado y aprovechamos ahora para ampliar.

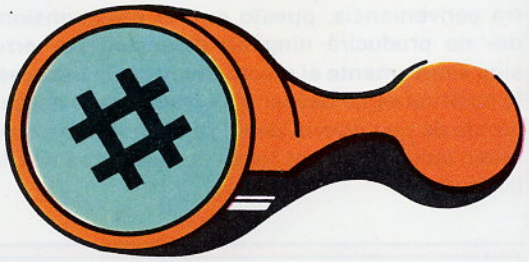
Los conjuntos de variables alfanuméricas, se denominan mediante una sola letra seguida del símbolo dólar ($).





El número de* DIM*s en un programa, así como su situación en el mismo, no tienen limitación alguna.

En el programa CAPITALES DE EUROPA, hemos utilizado el símbolo almohada (#) como «marca» que evita la repetición de preguntas.



Para la primera novedad, debemos fijarnos en las líneas 20 y 100. En la línea 20, asignamos el valor de 15 espacios a la variable **X$**. Esto es debido a que **X$** es la variable que empleamos para el **INPUT** de la capital en la línea 100, de la forma **X$ (TO 15)**. Generalizando, podemos conseguir variables de longitud fija utilizando este procedimiento: primero definir la variable con tantos blancos como deseemos, 15 en nuestro caso, y después hacer las posteriores asignaciones de valor por medio de **LET, READ** o **INPUT,** especificando qué parte de **X$,** en nuestro ejemplo **TO 15.**

Este modo de trabajar nos es muy útil en esta ocasión, puesto que, como sabemos, los elementos de las tablas alfanuméricas son todos de la misma longitud, es decir, se rellenan con blancos las posiciones no utilizadas. En este estado de cosas, la única manera de establecer comparaciones es este sistema. Para completar nuestro conocimiento sobre el comportamiento de los dimensionados alfanuméricos, hemos de saber que, al igual que las asignaciones de menor longitud se rellenan con blancos por la derecha, las más largas se truncan por la longitud máxima admitida.

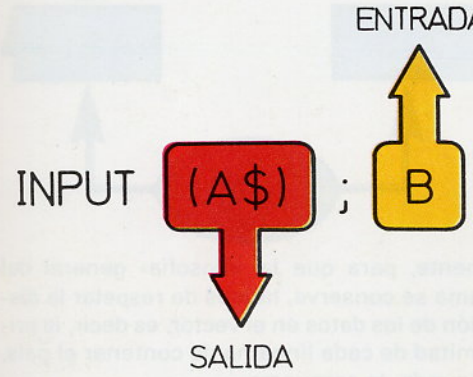
Así por ejemplo, si a un vector alfabético al que se le señaló una longitud máxima por elemento

de cinco caracteres, se le intenta asignar el valor "ANA", pasará a tener el contenido "ANA ". Por el contrario, si se le asigna "GLORIA", tomará el valor "GLORI".

En la línea 30, definimos el vector alfanumérico **P$(22,32)**, es decir, 22 elementos de 32 posiciones.

En las líneas 40 a 60, se establece la carga de la matriz, a partir de los datos contenidos en las líneas de data, 180 a 220. Como vemos, cada elemento de la matriz está dividido en dos mitades, de las cuales la primera (posiciones 1 a 16) contiene el país y la segunda (posiciones 17 a 32) contiene la capital.

En la línea 70 comienza el bucle de diez preguntas. Es evidente que precisamos una elección de preguntas al azar, pero sin que éstas puedan



Cuando en un **INPUT** *una variable aparece entre paréntesis, es considerada de salida (título).*

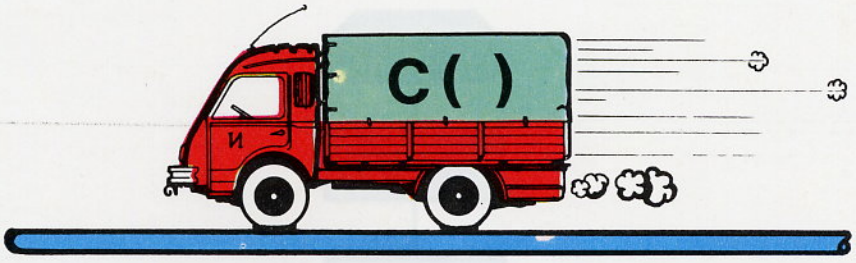
repetirse. Esto se logra con las instrucciones 80 y 90. En 80, seleccionamos primero un número aleatorio entre 1 y 22 y comprobamos si tiene la marca de haber aparecido ya. Si la tiene, repetimos la generación aleatoria; si no, pasamos a la instrucción 90, donde marcamos el elemento de la tabla como utilizado, con un símbolo almohada (#) en la posición 32.

En la línea 100 utilizamos una nueva versión de **INPUT.** Es la modalidad de petición con texto variable, que consiste en indicar a continuación del **INPUT,** y entre paréntesis, el nombre de la variable que deseamos aparezca en la petición de los datos. Es muy importante que esta variable aparezca entre paréntesis, para distinguirla de la que se va a asignar en el **INPUT.**

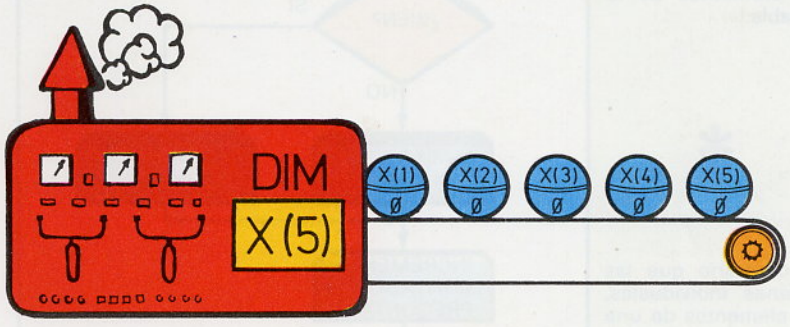
En la línea 110, comparamos la respuesta tecleada (de longitud fija), con las posiciones 17 a 31 del elemento **(X)** seleccionado. En caso de acierto, se asigna a **E$** su valor, y se duplica el total acumulado de puntos, desviando la ejecución a la instrucción 130. A la línea 120 sólo se accede en caso de error, para colocar **E$** en su valor, y dejar el acumulado de puntos en la mitad.

En la línea 130 se imprime el valor de **E$,** evaluación de la respuesta, y el acumulado de puntos obtenidos hasta el momento, y en la línea 140 se cierra el bucle de preguntas abierto en la instrucción 70. Por último, las líneas 150 a 160, que sólo se alcanzan al final del ciclo de preguntas, evalúan los resultados obtenidos en base a la puntuación acumulada.

El programa en su estado actual, memoriza hasta 22 países europeos con sus respectivas capitales. Para ampliar sus conocimientos geográficos, sólo es necesario añadir las **DATA** correspondientes, teniendo en cuenta que también hemos de aumentar el número de elementos en el dimensionado de la matriz, y por supuesto, el bucle de lectura de las **DATA,** para su asignación a la variable suscrita. Del mismo modo, la extracción aleatoria de la pregunta también debe ser alterada, puesto que se mueve en el rango antes citado.

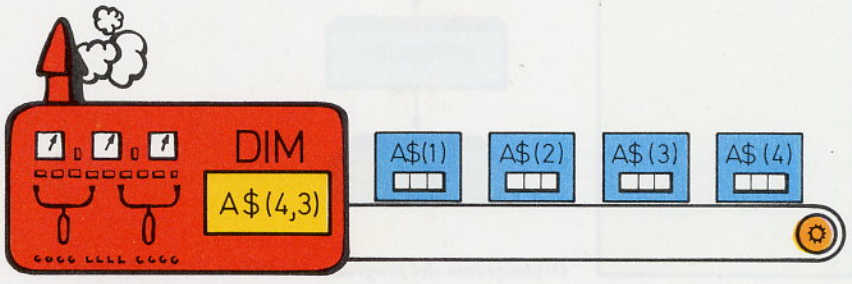


Una matriz numérica debe tener un nombre compuesto por una sola letra.



Cuando una variable numérica se dimensiona, sus elementos adoptan automáticamente el valor cero.

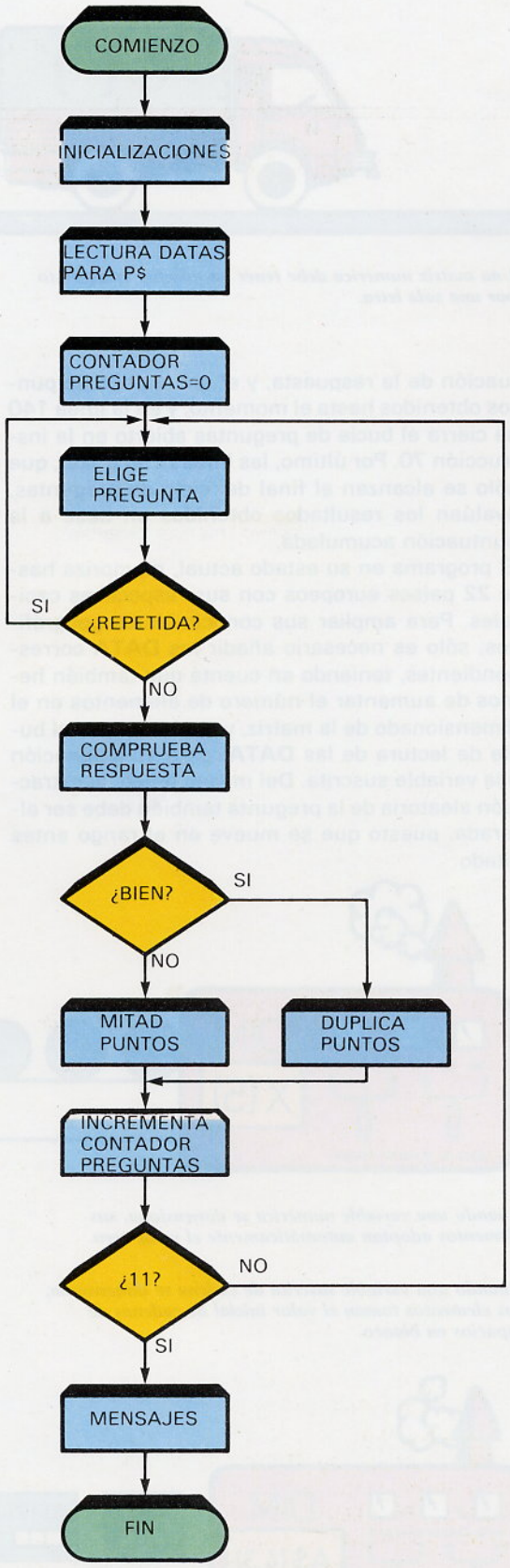
Cuando una variable suscrita de cadena se dimensiona, sus elementos toman el valor inicial de cadenas de espacios en blanco.



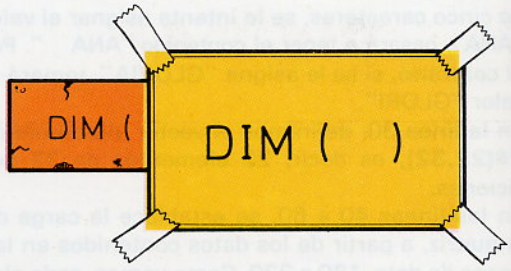
En las variables suscritas alfanuméricas, el parámetro que corresponde a la segunda dimensión en la sentencia **DIM**, se emplea para expresar la longitud máxima de los elementos de la variable.

*

Al contrario que las cadenas individuales, los elementos de una variable suscrita tienen una longitud fija, señalada en su correspondiente **DIM**.

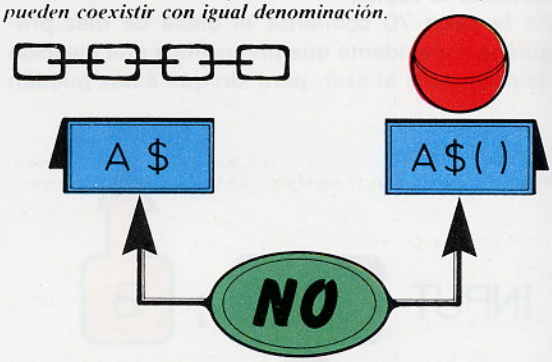


Organigrama del programa CAPITALES DE EUROPA.



Cualquier variable suscrita desaparece completamente al efectuarse un nuevo **DIM** *de la misma variable.*

Una variable de cadena y una suscrita del mismo tipo, no pueden coexistir con igual denominación.



Finalmente, para que la «filosofía» general del programa se conserve, hemos de respetar la distribución de los datos en el vector, es decir, la primera mitad de cada línea ha de contener el país, y la segunda, la respuesta.
Como demostración de lo fácil que es modificar este programa, añadiremos cinco preguntas más a nuestro test geográfico: las capitales de BULGARIA (SOFIA), ALBANIA (TIRANA), BELGICA (BRUSELAS), PAISES BAJOS (LA HAYA) y LUXEMBURGO (LUXEMBURGO). Siguiendo los pasos que hemos mencionado anteriormente, deberemos completar las líneas **DATA** que finalizan el programa con los nuevos datos:

```
230 DATA ''BULGARIA'',''SOFIA'',''ALBA
NIA'',''TIRANA'',''PAISES BAJOS'',''LA HA
YA'',''BELGICA'',''BRUSELAS'',''LUXEM
BURGO'',''LUXEMBURGO''
```

Además, es necesario adaptar las dimensiones de la matriz, para que pueda albergar estos nuevos elementos, y por supuesto, prolongar el bucle de lecturas **(READ)**, puesto que el simple hecho de añadirlos a la **DATA** no implica que sean leídos.

```
30 DIM P$(27,32)
40 FOR I=1 TO 27
```

Por último, es preciso ampliar el rango de selección aleatoria de las preguntas, para que también sea posible escoger las recién llegadas:

```
80 LET X=INT (RND*27)+1:IF P$(X,32)=''#''
THEN GO TO 80
```

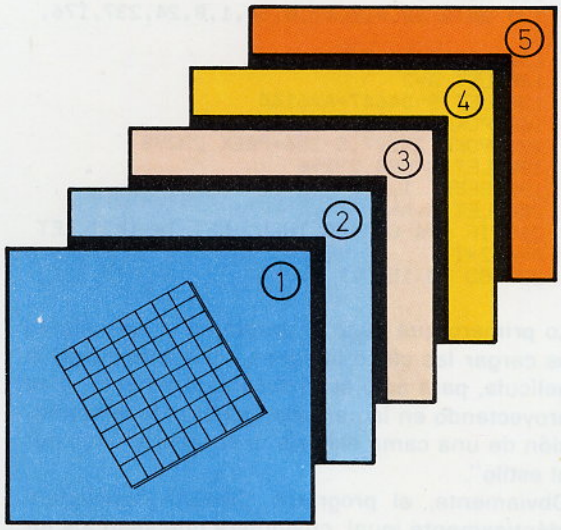
¡SE RUEDA!

EN este capítulo daremos ya por cerrado el tema del dinamismo en la pantalla completa, con una técnica basada en la subrutina de transferencia de memoria que vimos anteriormente. Lamentablemente, debido a la gran cantidad de memoria que se precisa, la técnica que a continuación exponemos está vedada para los usuarios del modelo de 16 K.

Realmente, su fundamento es muy sencillo. Se trata de almacenar varias pantallas en la memoria, por supuesto no en la memoria de pantalla, para posteriormente situarlas en esta zona una después de otra, de forma que consigan una animación similar a la que se obtiene en el cine con el paso de los fotogramas. La restricción en nuestro caso está bien clara: el máximo número de pantallas que se pueden almacenar son cinco, y con sólo cinco fotogramas mal vamos a hacer un largometraje.

El sistema de animación que ponemos en práctica en este capítulo, es similar al utilizado para la proyección de películas de cine.

No obstante, el efecto que se consigue con esta técnica es muy digno de tener en cuenta. El primer paso será obtener los necesarios cinco fotogramas; a tal fin, podemos introducir el programa de gráficos en tres dimensiones, aparecido en el capítulo segundo de la sección de programas. Una vez hecho esto, iremos creando pantallas de acuerdo con las fórmulas que a continuación se facilitan, y por el mismo orden. Cada pantalla deberá ser grabada una detrás de otra en una misma cinta, para que luego puedan ser utilizadas por el programa de demostración "CINEMA".



Gracias a la subrutina de transferencia en código máquina, la animación se consigue por superposición de imágenes a alta velocidad.

La grabación de las imágenes es muy fácil. Sólo hemos de esperar tranquilamente a que la figura correspondiente se dibuje, y cuando aparezca el mensaje **9 STOP statement, 470:1** podremos grabar la pantalla con el comando **SAVE "<nombre>"SCREEN$**. Lógicamente, el nombre para la grabación no tiene la menor importancia, y puede ser por ejemplo, el número de "fotograma" ("1", "2", "3", etc...).

Para que aparezca el mensaje de interrupción del programa, que nos dará opción a la grabación de la pantalla, es necesario, previamente a la ejecución del programa **3D**, introducir una pequeña modificación: cambiar la línea 470 para que quede como **470 STOP**.

Bien, ahora hemos de hacer el reparto de papeles para nuestra película: la función para la primera pantalla será **(X*X+Z*Z)/1000**, para la segunda pantalla utilizaremos **(X*X+Z*Z)/3000**,

BITS

Habitualmente, el color que el Spectrum utiliza como tinta para el marco de la pantalla, es el de contraste; si queremos utilizar otro color, debemos de indicarlo mediante un **POKE** a la Variable del Sistema **BORDCR** (23624). El valor a introducir será el código de color del marco multiplicado por ocho, más el código de tinta a emplear. Así por ejemplo, **POKE 23624,52** producirá la impresión en tinta verde sobre marco amarillo, puesto que 62 = 8 * 6 (código amarillo) + 4 (código verde). Este efecto desaparecerá al efectuar un nuevo **POKE** a esta dirección, o al ejecutar cualquier comando **BORDER**.

BITS

La lectura de información del casete no está sólo reservada al código máquina. Nosotros mismos, desde el BASIC, podemos averiguar mediante una sencilla operación, si el Spectrum está "escuchando" algún sonido. Tal y como se hace en el siguiente ejemplo, esta técnica puede emplearse para amenizar un programa hasta la carga del próximo, con la seguridad de que, sea cual sea el espacio entre ambos, no perderemos ni un bit de información.

```
10 BORDER 1: PAPER 1: CLS
20 FOR I=0 TO 6
30 BEEP .01,30+RND*2*I: BORDER 6-I
40 LET J=IN 254: IF J>127 THEN LET J=J-128
50 IF J>63 THEN LOAD ""
60 NEXT I: GO TO 20
```

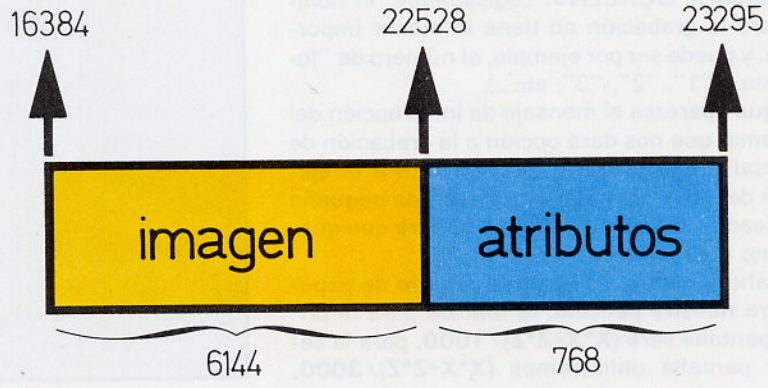
para la tercera **0** (dibujo plano), para el cuarto "fotograma" **SIN (((60-X*X-Z*Z) * (ABS SIN (60-X*X-Z*Z)-1))/5000)**, y para el último, **SIN (((60-X*X-Z*Z)*(ABS SIN (60-X*X-Z*Z)+1))/2000)**.

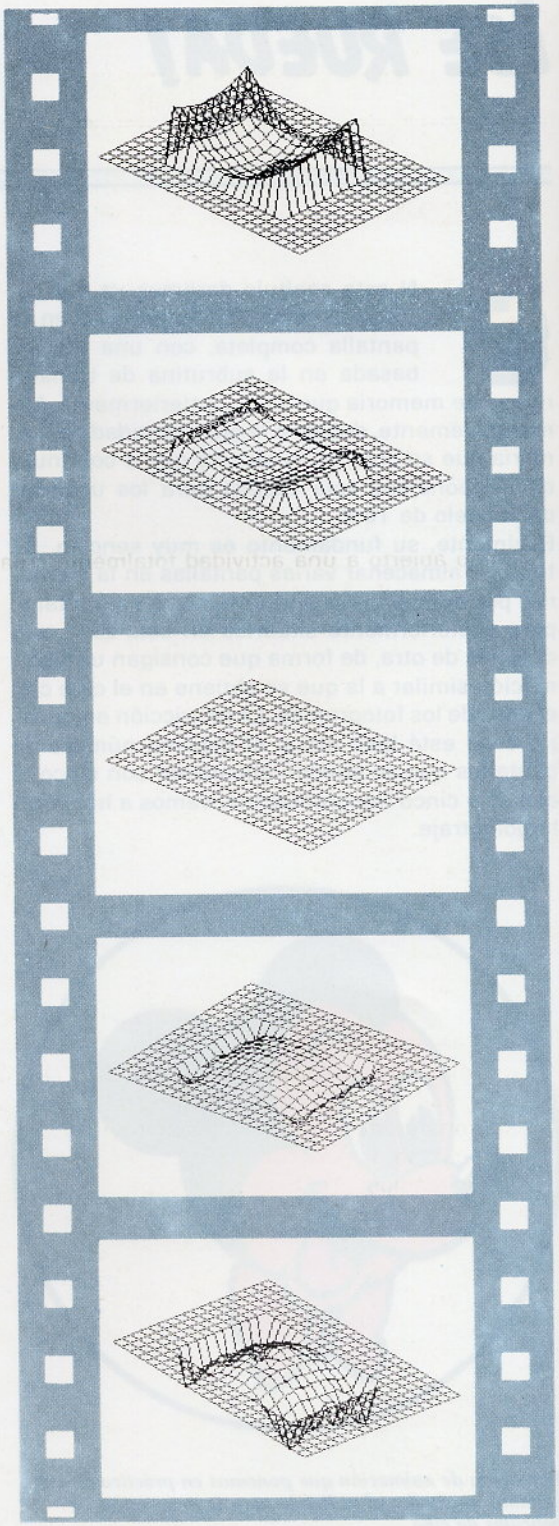
Una vez preparada la película, ya sólo queda proyectarla, para lo cual emplearemos el programa CINEMA.

```
 10 REM CINEMA - C.DE LA OSSA & F.LOPEZ MARTINEZ
 20 CLEAR 34646
 30 LOAD "1"CODE 34647
 40 LOAD "2"CODE 40791
 50 LOAD "3"CODE 46935
 60 LOAD "4"CODE 53079
 70 LOAD "5"CODE 59223
 80 FOR I=23296 TO 23307
 90 READ A
100 POKE I,A
110 NEXT I
120 DATA 33,0,0,17,0,64,1,0,24,237,176,201
130 LET I=1
140 LET A=2
150 LET S=34647+A*6144
160 POKE 23298,INT (S/256)
170 POKE 23297,S-256*PEEK 23298
180 LET S=USR 23296
190 BEEP .1,-20+RND*60
200 LET A=A+I
210 IF A>4 OR A<0 THEN  LET I=-1*I: LET A=A+2*I
220 GO TO 150
```

Lo primero que hace el programa al ejecutarse, es cargar las cinco imágenes que configuran la película, para acto seguido, y cíclicamente, irlas proyectando en la pantalla, generando la animación de una cama elástica, o al menos "algo por el estilo".

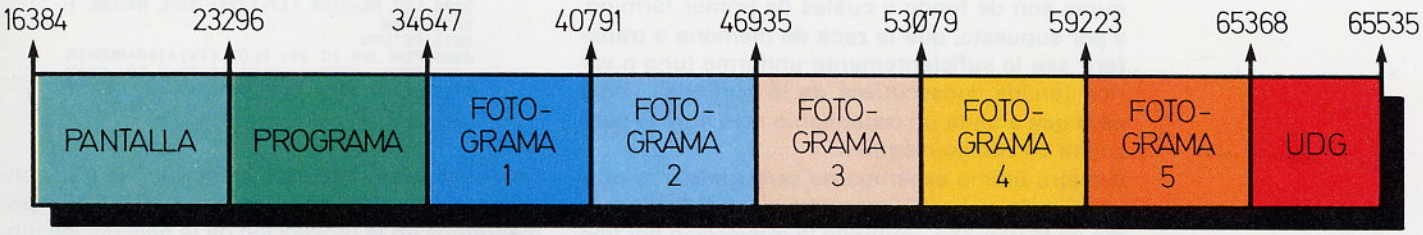
Obviamente, el programa CINEMA funcionará idénticamente igual, cualquiera que sean los fotogramas que se sitúen en la memoria; por tanto, podemos convertirnos en mini-George Lucas, y protagonizar la dirección de apasionantes secuencias a nuestro gusto: el estallido de una nave espacial, el movimiento de un brazo, el pestañeo de unos ojos, etc... En nuestro caso hemos empleado el programa **3D**, pues con un corto esfuerzo de introducción, nos permite disponer de los cinco fotogramas necesarios. Indudablemen-





Para hacer nuestra película, disponemos tan solo de cinco fotogramas.

Para ahorrar memoria, sólo se han almacenado los archivos de imagen, pero no las áreas de atributos; por tanto, nuestras películas son en blanco y negro.



En el gráfico se señala la distribución de memoria en el programa CINEMA.

te, otras secuencias considerablemente más artísticas pudieron ser obtenidas, sin embargo, su espectacularidad no justificaba la gran ocupación de espacio, y lo aburrido de su introducción. Como hemos visto, el programa CINEMA deja un camino abierto a una actividad totalmente creativa y artística, gracias a un ordenador. El punto más importante en este aspecto, es que los fotogramas estén lo suficientemente próximos en el tiempo, como para que no se produzcan saltos en la imagen, sino una armoniosa sensación de continuidad.

Sobre la subrutina de transferencia, sólo hay que añadir que es completamente reubicable, debido a lo cual, funcionará perfectamente cualquiera que sea la zona de memoria en que se almacene. Para ahorrar espacio, en el ejemplo ha sido depositada en el *buffer* (memoria intermedia) de la impresora. Por otra parte, el sistema de utilización de la rutina es muy simple: el primer código es siempre un 33, al cual le sigue la dirección en que se encuentra la pantalla a transferir, expresada en peso bajo-peso alto.

Los próximos tres códigos, no se cambian (17, 0 y 64), puesto que indican la dirección de destino de la transferencia, la cual, lógicamente, es siempre la misma: la pantalla (16384). Por último, al próximo código (1), le debe seguir, expresado nuevamente en bajo-alto, el número de bytes a transferir; en nuestro caso son 6144, es decir, el archivo de imagen, y por tanto, los bytes codificados son 0 y 24 (24*256+0=6144). Si quisieramos trasladar la pantalla completa, incluyendo el área de atributos, los bytes a codificar serían 0 y 27, correspondientes a los 6912 bytes que integran la pantalla completa (27*256+0=6912).

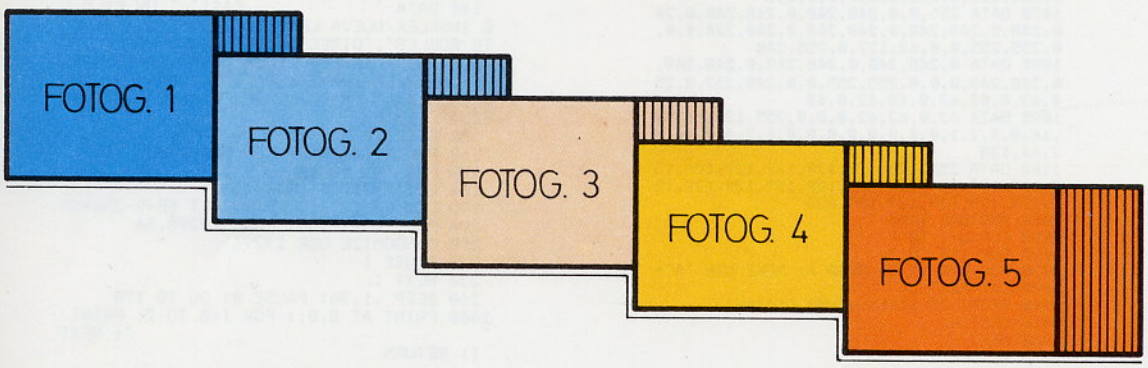
CINEMASCOPE

En la introducción al dinamismo de pantallas completas, comentamos la posibilidad de, utilizando las mismas técnicas, poder animar una determinada zona de la pantalla, de fácil tratamiento por la subrutina de transferencia. Ya entonces, sugerimos para la creación de estos efectos la utilización de tercios de pantalla, que debido a la peculiar organización del archivo de imagen, podían ser tratados prácticamente como una pantalla independiente.

Como vimos entonces, para la consecución de esta técnica de dinamismo, no son importantes las dimensiones del área a tratar, sino el hecho de que las imágenes sean transferidas completas, es decir, sin tomar en consideración qué fi-

Por motivos de memoria, la técnica descrita en este capítulo, está vedada a los usuarios del modelo de 16 K.

Debido a las direcciones de almacenamiento, aunque las pantallas se cargan completas, sus respectivos archivos de imagen solapan el área de atributos de la anterior.



guras son de fondo y cuales de primer término, y por supuesto, que la zona de memoria a transferir sea lo suficientemente uniforme (uno o varios tercios consecutivos de la pantalla), como para que admita un tratamiento mediante la subrutina en código máquina.

Nuestro último experimento será realizar la proyección de un "cortísimometraje" en el tercio medio de la pantalla, imitando la proyección del tipo CINEMASCOPE. Como en el caso anterior, la confección del "film", consta de dos fases. En primer lugar, un árduo rodaje, cuyo protagonista será el conocido actor TORPEDO RUN, y que será llevado a cabo mediante el programa ACCION!

Ha nacido una estrella: TORPEDO RUN.

```
  10 REM ACCION! - C.DE LA OSSA & F.LOPE
Z MARTINEZ
  20 FOR I=0 TO 7: READ A: POKE USR "D"+
I,A: NEXT I: DATA 56,24,24,48,96,103,126
,56
  30 PRINT AT 12,15;"D": PLOT 123,80: DR
AW 0,32
  40 LET I=32: GO SUB 3000
  50 FOR I=30 TO 18 STEP -2
  60 LET F=31: GO SUB 2000
  70 GO SUB 3000
  80 NEXT I: CLS
  90 FOR I=16 TO 2 STEP -2
 100 LET F=I+13: GO SUB 2000
 110 PRINT AT 12,I-1;"D": PLOT (I-1)*8+4
,80: DRAW -(I-16)*8,32
 120 GO SUB 4000
 130 GO SUB 3000
 140 CLS
 150 NEXT I
 160 LET I=0: LET F=13: GO SUB 2000
 170 PRINT AT 9,10;"D": PLOT 84,104: DRA
W 48,7
 180 GO SUB 4000: GO SUB 3000
 190 CLS
 200 GO SUB 2000
 210 PRINT AT 12,19;"4337 3374  7"
 220 PRINT AT 13,19;"537   5 56 3"
 230 PRINT AT 14,19;"5     5 5 65"
 240 PRINT AT 15,19;"1    3321  2"
 250 PLOT 56,88: DRAW 7,7: DRAW 0,-7: PL
OT 56,63: DRAW 7,-7: DRAW 0,7
 260 LET I=-2: GO SUB 3000
 270 GO TO 10000
1000 DATA 0,0,0,0,1,2,4,4,24,40,72,72,72
,40,24,4,4,2,1,0,0,0,0,0,31,48,64,128,0,
0,0,0
1010 DATA 0,0,0,0,0,0,0,0,0,0,0,128,64,3
2,24,15,255,17,33,65,178,244,248,112,0,0
,0,0,0,0,0,1
1020 DATA 1,1,6,9,9,50,194,255,255,0,0,0
,0,0,0,0,0,0,0,128,128,128,128,128,128,1
28,128,0,0,0,0,0,255
1030 DATA 255,15,15,15,240,240,240,240,1
5,15,15,15,240,240,240,240,15,15,15,15,2
40,240,240,255
1040 DATA 255,0,0,15,15,0,15,15,0,15,15,
0,15,15,0,15,15,0,15,15,0,0,0,255
1050 DATA 255,0,0,255,255,0,255,131,0,13
1,131,0,131,255,0,255,255,0,131,131,0,0,
0,255,255,0,0,131,195,0,227,227,0,227,22
7,0,227,195,0,131
1060 DATA 195,0,225,224,0,0,0,255,255,0,
0,224,224,0,224,224,0,224,224,0,224,224,
0,224,255,0,255,255,0,0,0,255
1070 DATA 255,0,0,248,248,0,248,248,0,24
8,248,0,248,248,0,248,248,0,240,224,0,0,
0,255,255,0,0,63,127,0,255,248
1080 DATA 0,248,248,0,248,248,0,248,248,
0,248,248,0,0,0,255,255,0,0,248,252,0,25
4,62,0,62,62,0,62,62,0,62
1090 DATA 62,0,62,62,0,0,0,255,128,64,32
,16,8,4,2,1,0,0,0,0,0,0,0,1,2,4,8,16,3
2,64,128
1100 DATA 255,129,129,129,129,129,129,12
9,129,129,129,129,129,129,129,129,129,12
9,129,129,129,129,129,255
2000 RESTORE 1000
2010 FOR J=I TO F
2020 FOR K=0 TO 2
2030 FOR L=0 TO 7: READ A: POKE USR "A"+
L+K*8,A: NEXT L
2040 PRINT AT 11+K,J;CHR$ (144+K)
2050 NEXT K
2060 NEXT J
2070 RETURN
3000 LET N$=STR$ (17-I/2): SAVE N$CODE 1
8432,2048
3010 RETURN
4000 FOR J=0 TO 39: PLOT (I+14)*8+RND*15
,72+8*RND: NEXT J: PLOT (I+7)*8,88: DRAW
 7,7: DRAW 0,-7: PLOT (I+7)*8,63: DRAW 7
,-7: DRAW 0,7
4010 RETURN
```

Con la ejecución de este programa, se irán conformando las diferentes imágenes que luego emplearemos en la proyección de la película (segunda fase). Por cada una de ellas, aparecerá en la parte inferior de la pantalla el mensaje habitual de grabación *Start tape then press any key;* de esta forma iremos grabando en una cinta, una a continuación de otra, las dieciocho pantallas diferentes que se nos presenten. En esta ocasión, el número de fotogramas ha aumentado sensiblemente, debido a que la cantidad de memoria que utiliza cada uno es una tercera parte de la pantalla completa (2048 bytes).

Finalmente, el programa TORPEDO, cuyo listado aparece al final de este artículo, pondrá en movimiento las imágenes "rodadas" con TORPEDO RUN. Para ello, se sigue un sistema prácticamente idéntico al del programa CINEMA, con la diferencia de que en esta ocasión se realiza la carga de dieciocho fotogramas, y por lo tanto, no en las mismas posiciones que en el ejemplo a pantalla completa. Por otra parte, la rutina de transferencia ha sido ligeramente alterada, puesto que esta vez, varía tanto la zona de destino de la imagen, como el número de bytes a transferir. La dirección de destino es ahora el primer byte del segundo tercio de la pantalla (16384 + 2048 = 18432). Del mismo modo, el número de bytes a transferir es sólo un tercio (2 K), lo que suponen 2048 bytes.

```
  10 REM TORPEDO - C.DE LA OSSA & F.LOPE
Z MARTINEZ
  20 CLEAR 28499
  30 BORDER 0: PAPER 0: INK 7: CLS : PRI
NT AT 11,6;"VISITE NUESTRO BAR"
  40 FOR I=1 TO 18
  50 LOAD (STR$ I)CODE 28500+2048*(I-1),
2048
  60 IF I>14 THEN  BEEP .5,30
  70 NEXT I
  80 FOR I=23296 TO 23307
  90 READ A: POKE I,A: NEXT I
 100 DATA 33,0,0,17,0,72,1,0,8,237,176,2
01
 110 PRINT AT 21,0;"CINEMASCOPE"
 120 PAPER 2: INK 7: GO SUB 1000
 130 FOR I=8 TO 15: READ A$: PRINT AT I,
0;A$: NEXT I
 140 DATA "           CAST"," UN FILM D
E INGELEK/NUEVA LENTE","PRODUCTOR: VICEN
TE ROBLES","DIRECTOR: FERNANDO LOPEZ","R
EALIZADOR: CARLOS DE LA OSSA","AYTE.DIRE
CCION: PILAR MANZANERA","FOTOGRAFIA: EQU
IPO GALATA","MAQUILLAJE: CARLOS GLEZ.-AM
EZUA"
 150 BEEP 10,0
 160 PAPER 5: INK 0: GO SUB 1000
 170 FOR I=1 TO 18
 180 LET S=28500+2048*(I-1)
 190 LET SA=INT (S/256): LET SB=S-256*SA
 200 POKE 23297,SB: POKE 23298,SA
 210 RANDOMIZE USR 23296
 220 PAUSE 1
 230 NEXT I
 240 BEEP .1,30: PAUSE 0: GO TO 170
1000 PRINT AT 8,0;: FOR I=0 TO 7: PRINT
                                 ": NEXT
 I: RETURN
```

SIMON

ESTE nuevo programa nos ayudará a ejercitar nuestra capacidad de retención, tanto visual como auditiva, al mismo tiempo que nos divertiremos y pasaremos un rato muy agradable.

El juego del SIMON es una competencia contra nosotros mismos, es decir, un solitario, en el cual la máquina tiene un papel pasivo, de forma similar al MASTER MIND (MASTER CUB, para los lectores de esta Obra).

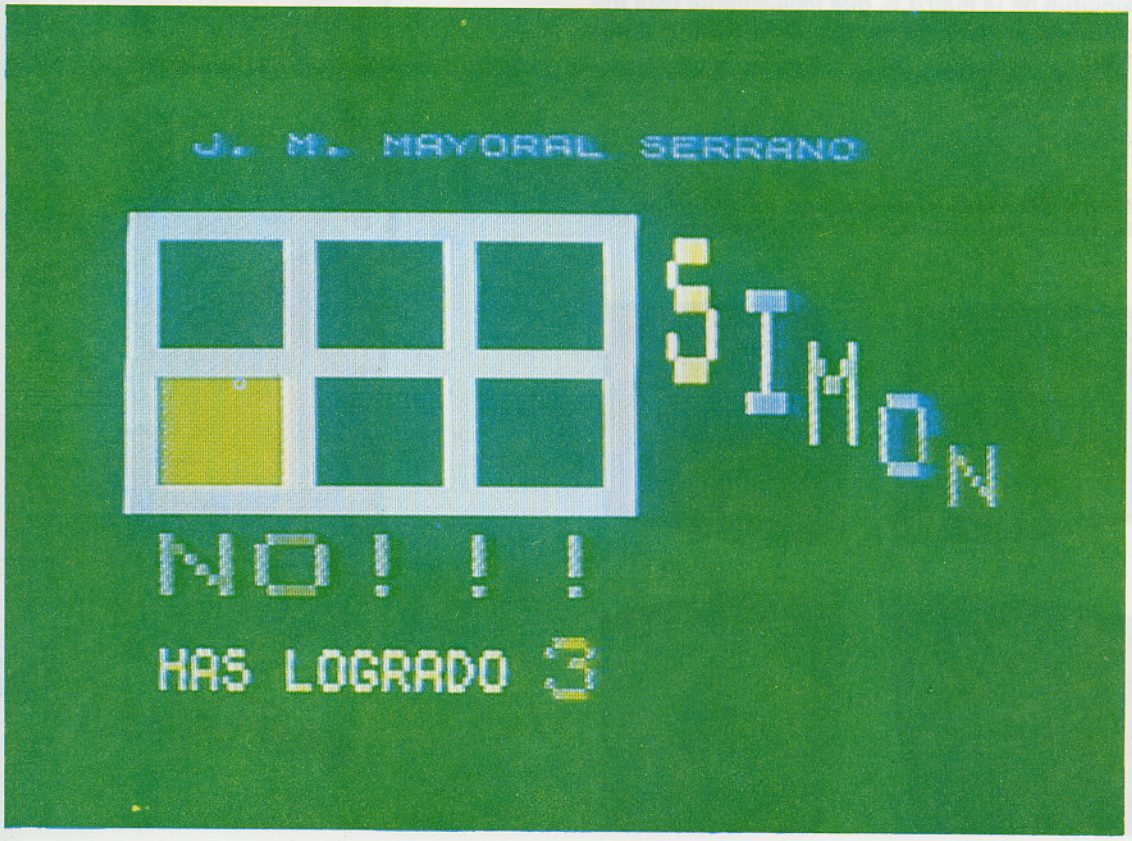
El ordenador emite un determinado sonido, al tiempo que presenta un cuadrado de color en la pantalla; a continuación debemos pulsar la tecla correspondiente a ese color, o si lo preferimos, a ese sonido. Si conseguimos acertarlo, al sonido y color anteriores se añadirá uno nuevo, diferente o no. Esta vez debemos repetir los dos colores. El juego continuará aumentando la secuencia, hasta que nos equivoquemos en alguna de ellas.

Nuestro programa posee dos niveles de dificultad. El primero y más sencillo, hace uso de los cuatro primeros colores básicos del Spectrum: azul, rojo, magenta y verde. El nivel restante, utiliza dos colores más, siguiendo el orden antes expuesto: azul, rojo, magenta, verde, cián y amarillo.

Estos colores son generados aleatoriamente uno a uno, y almacenados en una tabla para su posterior presentación en pantalla. Por cada secuencia de colores acertada, nuestro ordenador elige otro, añadiéndole al final de la cadena de colores, que va aumentando a medida que avanzamos en el juego.

Por cada color generado, se emite siempre un mismo sonido, de forma que si no somos capaces de recordar la secuencia policromada presentada en la pantalla, nuestro Spectrum nos da otra

La presentación del programa en la pantalla, gana mucho en calidad con la utilización de la subrutina de caracteres gigantes.



¡!

El programa carece de gráficos definidos, debido a lo cual en el listado no aparecen caracteres subrayados.

*

Para grabar el programa, utilizaremos la acción combinada de los comandos **SAVE** y **LINE** de la siguiente forma:
SAVE ''SIMON'' LINE 1480.

opción: recordarla dependiendo de la secuencia de notas que «deja» escapar. En el momento en que fallemos un color o nota, el Spectrum nos lo hará saber inmediatamente, a la vez que nos informará de los puntos obtenidos, finalizando el juego en ese momento.

La forma de puntuar de nuestro Spectrum es la siguiente: sumará al marcador tantos puntos como notas o colores tenga la secuencia presentada, siempre y cuando se acierte dicha secuencia. Es decir, si el ordenador ha generado 7 colores y hemos sido capaces de recordarlos por su orden cronológico de aparición, sumará 7 al marcador, mientras que si fallamos en alguno de ellos, la puntuación en esa secuencia será 0 y la partida habrá finalizado.

EL PROGRAMA

Este programa puede almacenar una serie de 200 colores con sus notas correspondientes. Si queremos aumentar o disminuir dicha cantidad, no tenemos más que variar el número que define la longitud de la matriz S, que lógicamente está inicializada en 200 elementos: **DIM S(200)** (línea 130).

¡!

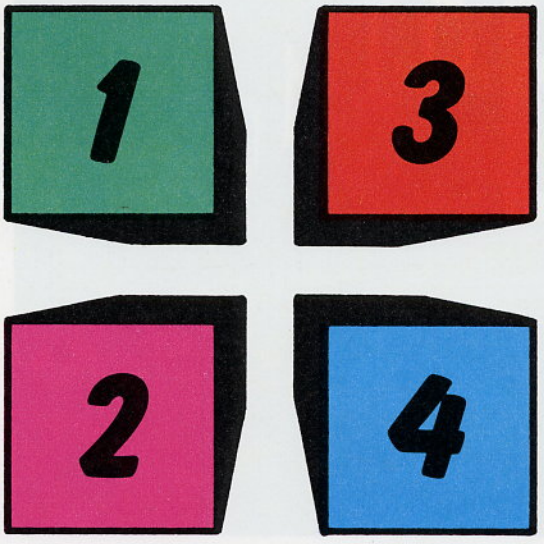
Los dos niveles de dificultad que posee el juego, hacen referencia al número de colores que pueden entrar en la secuencia a recordar (cuatro o seis).

*

Las teclas que hay que pulsar para emitir la secuencia de colores y sonidos, son las de los números correspondientes a los códigos de color deseados.

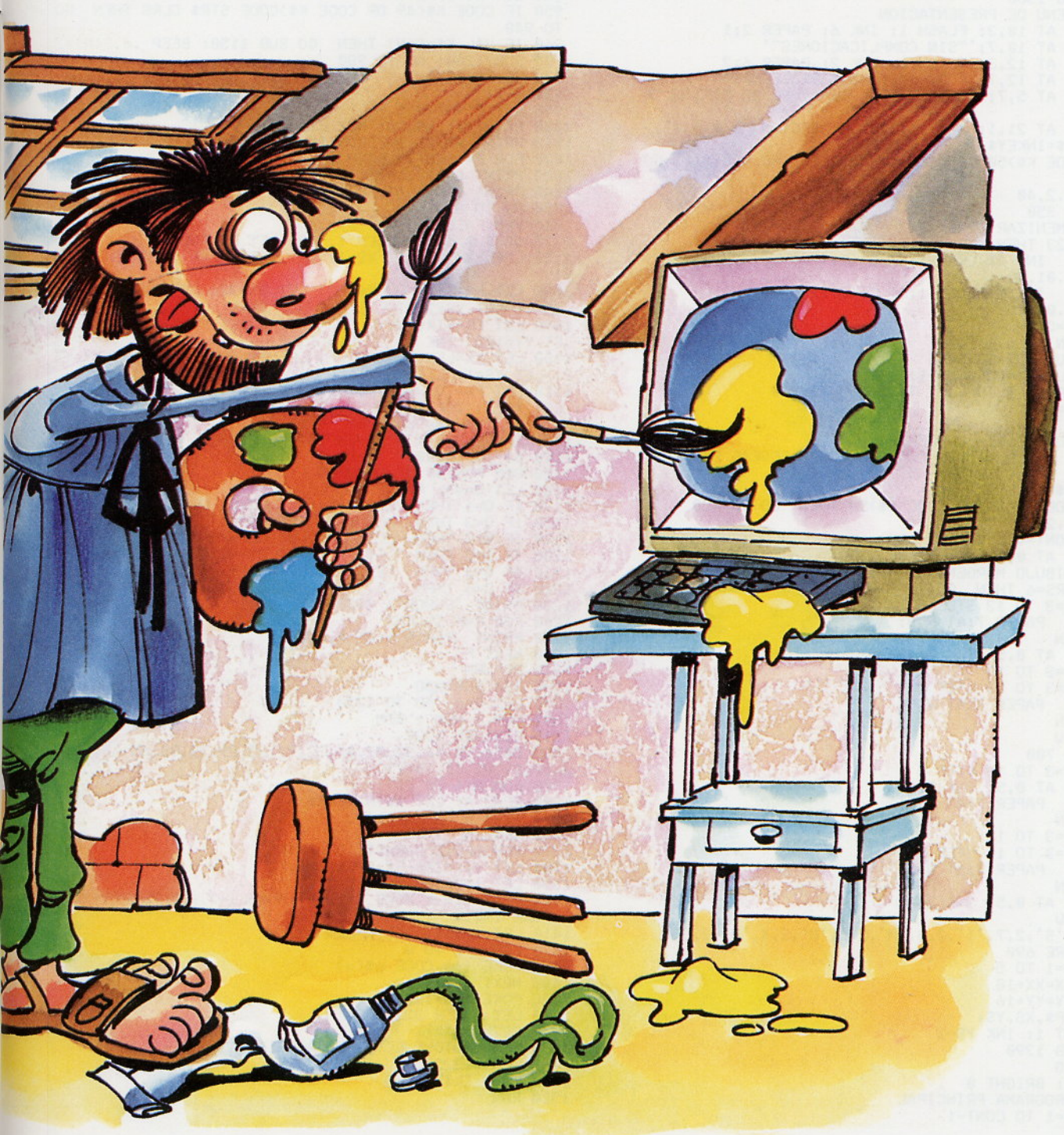
El tiempo que un color está presente en la pantalla, depende del nivel de dificultad, y se controla mediante la variable **T,** que se utiliza como parámetro del comando **BEEP.** Esta variable numérica, según el nivel de dificultad elegido, puede tener dos valores: en el nivel uno, **T** adquiere el valor 0.5, y en el segundo nivel su valor es 0.05. De esta forma, cuanto más pequeño sea **T,** menos tiempo estará el color generado en la pantalla.

El nivel más sencillo de nuestro juego del SIMON, se basa en los cuatro primeros colores del Spectrum.



En SIMON volvemos a utilizar la subrutina de caracteres gigantes de PSION, cuidando así la estética del programa en la pantalla. Por tanto, seguiremos para la adopción de la rutina las especificaciones que a este respecto se han brindado en programas anteriores, que también hayan hecho uso de ella.

En esta ocasión, echaremos de menos los caracteres subrayados en el listado, puesto que no se emplean gráficos, sino diferentes códigos de **PAPER** para la generación de los colores.
Es importante recordar la absoluta necesidad de tener cargada la subrutina de caracteres gigantes para el correcto funcionamiento del programa; así pues, para almacenarlo, utilizaremos el siguiente comando: **SAVE "SIMON" LINE 1480**. Disponiendo la autoejecución en la línea 1480, que es la encargada de la lectura del código máquina.



```
 10 REM ************************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ************************
 40 REM *   Program. SIMON    *
 50 REM ************************
 60 REM INICI. - VARIABLES
 70 BORDER 0: PAPER 0: INK 9: CLS : POKE 23658,8
 80 LET XX=140
 90 LET YY=8
100 LET C=10
110 DIM A$(21)
120 LET N1=0
130 DIM S(200): LET IND=1
140 GO SUB 1300
150 REM MENU DE PRESENTACION
160 PRINT AT 10,3; FLASH 1; INK 6; PAPER 2;1
170 PRINT AT 10,7;"~SIN COMPLICACIONES~"
180 PRINT AT 12,3; FLASH 1; INK 2; PAPER 6;2
190 PRINT AT 12,7;"~CON BASTANTES PROBLEMAS~"
200 PRINT AT 5,7; PAPER 5; INK 1;"NIVEL DE DIFICULTA
D"
210 PRINT AT 21,1;"# # # #  ELIGE OPCION  # # # #"
220 LET K$=INKEY$
230 IF CODE K$>50 OR CODE K$<49 THEN  GO SUB 260: GO
TO 220
240 BEEP .2,40
250 GO TO 350
260 REM AMENIZAR LINEA 210
270 IF C=23 THEN  LET C=10
280 PRINT  INK 4; OVER 1;AT 21,C;"  "
290 BEEP .01,40
300 PRINT  INK 7; OVER 1;AT 21,C-1;" "
310 BEEP .01,45
320 LET C=C+1
330 RETURN
340 REM NUM. DE COLORES
350 LET CLRS=2*(VAL K$+1)
360 IF CLRS=4 THEN  LET T=.5: GO TO 380
370 LET T=.05
380 CLS
390 REM NUM. SONI. COMIENZO
400 PRINT AT 19,0;"CON CUANTOS TONOS DESEAS EMPEZAR"
410 PRINT AT 21,12;"1 - 5"
420 LET K$=INKEY$
430 IF CODE K$<49 OR CODE K$>53 THEN  GO TO 420
440 BEEP .2,40: CLS
450 LET CONT=VAL K$
460 PRINT AT 0,5; INK 3;A$
470 REM DIBUJO MARGEN
480 IF CLRS=6 THEN  GO TO 590
490 FOR N=3 TO 13 STEP 5
500 PRINT  PAPER 7;AT N,3;"          "
510 NEXT N
520 PRINT AT 0,5; INK 3;A$
530 FOR W=3 TO 13
540 FOR N=3 TO 13 STEP 5
550 PRINT  PAPER 7;AT W,N;" "
560 NEXT N
570 NEXT W
580 GO TO 700
590 FOR N=3 TO 13 STEP 5
600 PRINT AT 0,5; INK 3;A$
610 PRINT  PAPER 7;AT N,3;"                "
620 NEXT N
630 FOR W=3 TO 13
640 FOR N=3 TO 18 STEP 5
650 PRINT  PAPER 7;AT W,N;" "
660 NEXT N
670 PRINT AT 0,5; INK 3;A$
680 NEXT W
690 DATA "S",2,7,"I",2,6,"M",2,5,"O",2,4,"N",2,3
700 RESTORE 690
710 FOR G=1 TO 5
720 LET XX=XX+18
730 LET YY=YY+16
740 READ P$,XS,YS:
750 BRIGHT 1: INK YS-1
760 GO SUB 1390
770 NEXT G
780 INK 9: BRIGHT 0
790 REM PROGRAMA PRINCIPAL
800 FOR N=1 TO CONT-1
810 LET S(IND)=INT (RND*CLRS)+1
820 LET IND=IND+1
830 NEXT N
840 REM COMIENZO SONIDOS
850 PRINT AT 0,5; INK 3;A$
860 LET S(IND)=INT (RND*CLRS)+1
870 FOR N=1 TO IND
880 LET K$=STR$ S(N): GO SUB 1150
890 BEEP T,S(N)*7: GO SUB 1290
900 NEXT N
910 LET IND=IND+1
920 REM INTENTOS ADVERSARIO
930 FOR N=1 TO IND-1
940 LET K$=INKEY$
950 IF CODE K$<49 OR CODE K$>CODE STR$ CLRS THEN  GO
TO 940
960 IF VAL K$=S(N) THEN  GO SUB 1150: BEEP .6,S(N)*7
: GO SUB 1280: GO TO 980
970 GO TO 1070
980 NEXT N
990 LET N1=N1+N-1
1000 PRINT AT 18,10;"MUY BIEN"
1010 PRINT AT 21,10;"CONTINUO"
1020 FOR P=1 TO 7: BEEP .1,35: NEXT P
1030 PRINT AT 18,10;"          "
1040 PRINT AT 21,10;"          "
1050 GO TO 840
1060 REM FALLOS
1070 GO SUB 1150: BEEP 1,-10: LET P$="NO!!!": LET YY=
115: LET XX=30: LET XS=3: LET YS=3: INK 4: GO SUB 139
0:
1080 LET P$="HAS LOGRADO": LET XX=32: LET YY=150: LET
 XS=1: LET YS=2: INK 6: GO SUB 1390
1090 LET P$=STR$ N1: LET XX=127: LET XS=2: LET YS=3:
LET YY=145: INK 2: GO SUB 1390
1100 LET XX=LEN P$*23+127: LET P$="PUNTOS": LET YY=15
0: LET XS=1: LET YS=2: INK 6: GO SUB 1390
1110 LET P$="OTRA PARTIDITA ?": LET YY=175: LET XS=2:
 LET YS=1: INK 5: GO SUB 1380
1120 IF INKEY$="S" THEN  RUN
1130 IF INKEY$<>"N" THEN  GO TO 1120
1140 GO TO 10000
1150 REM PINTOR DE CUADROS
1160 LET Q=VAL K$
1170 IF Q=1 THEN  LET F=4: LET CO=4: GO TO 1240
1180 IF Q=3 THEN  LET F=4: LET CO=9: GO TO 1240
1190 IF Q=5 THEN  LET F=4: LET CO=14: GO TO 1240
1200 IF Q=2 THEN  LET F=9: LET CO=4: GO TO 1240
1210 IF Q=4 THEN  LET F=9: LET CO=9: GO TO 1240
1220 IF Q=6 THEN  LET F=9: LET CO=14: GO TO 1240
1230 PRINT "ERROR": STOP
1240 FOR P=F TO F+3
1250 PRINT  PAPER Q;AT P,CO;"    "
1260 NEXT P
1270 RETURN
1280 REM BORRADO
1290 LET Q=0: GO TO 1240
1300 REM : REM : REM
1310 RESTORE 1320
1320 DATA 74,46,32,77,46,32,77,65,89,79,82,65,76,32,8
3,69,82,82,65,78,79
1330 FOR N=1 TO 21
1340 READ A: LET A$(N)=CHR$ A
1350 NEXT N
1360 RETURN
1370 REM BASIC CARACT. GIGANTES
1380 LET xx=(256-8*xs*LEN p$)/2
1390 LET i=23306
1400 POKE i,xx: POKE i+1,yy: POKE i+2,xs: POKE i+3,ys
: POKE i+4,8
1410 LET i=i+4: LET w=LEN p$
1420 FOR q=1 TO w
1430 POKE i+q,CODE p$(q)
1440 NEXT q
1450 POKE i+w+1,255
1460 LET w=USR 32256
1470 RETURN
1480 REM  AUTO-EJEC. OBLIGATORIA
1490 CLEAR 32255
1500 LOAD "LIT"CODE
1510 RUN
```