7
150pts.

RUN

Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

Los mejores libros para tu SPECTRUM


CODIGO	TITULO	P.V.P.
370610954	**SPECTRUM INTRODUCCION AL CODIGO MAQUINA** Para usuarios que no se contentan con el Basic y desean aprender más cosas sobre su computador.	1.450 Ptas.
370510601	**EL PROGRAMADOR DE SPECTRUM** Libro escrito para todos los usuarios de Spectrum que quieren aprender a escribir sus propios programas.	1.300 Ptas.
370510369	**SPECTRUM LIBRO DE JUEGOS** Colección de juegos escritos para Spectrum, con comentarios para mejorar su programación.	1.300 Ptas.
300170031	**PROGRAMACION PASO A PASO CON EL SPECTRUM** Libro fundamental para sacar todo el partido posible a la máquina y a los ordenadores que se usarán en el futuro.	750 Ptas.
300170048	**LAS PRIMERAS 15 HORAS CON EL SPECTRUM** Libro diseñado para guiar al nuevo usuario durante las primeras semanas de programación.	750 Ptas.
370510288	**ZX SPECTRUM COMO OBTENER EL MAXIMO RENDIMIENTO** El objetivo de este libro es ayudarle a conocer su ordenador para que pueda programar su Spectrum para que haga lo que Ud. desee.	1.300 Ptas.
300030132	**COMO USAR LOS COLORES Y LOS GRAFICOS EN EL SPECTRUM** Introducción al uso de gráficos y colores para ayudarle a transformar sus ideas en programas llenos de color y movimiento.	850 Ptas.
300030101	**COMO PROGRAMAR SU SPECTRUM** Curso de Programación Basic utilizando el Spectrum que permite descubrir las tremendas posibilidades de esta máquina.	850 Ptas.
370510873	**40 JUEGOS EDUCATIVOS PARA EL SPECTRUM** Colección de programas para ayudar al usuario a familiarizarse con el ordenador, aprendiendo y divirtiéndose a la vez.	1.300 Ptas.
370511250	**OBTENER EL MAXIMO RENDIMIENTO A SU ZX MICRODRIVE** Libro destinado a los que ya conocen el Spectrum y desean usar las posibilidades que permiten el Interface-1 y el Microdrive.	1.400 Ptas.
370511411	**GRAFICOS Y SONIDOS EN EL SPECTRUM** En este libro se estudian las técnicas necesarias para producir dibujos y diagramas sobre la pantalla.	1.900 Ptas.
340173195	**ZX SPECTRUM. INTRODUCCION AL PROCESAMIENTO DE TEXTOS** Este libro contiene técnicas de programación, y algunos programas comerciales para usar el Spectrum como potente procesador de textos.	1.690 Ptas.
370030025	**CODIGO MAQUINA ZX SPECTRUM** Este libro contiene las instrucciones fundamentales de Z 80 para iniciarse en el código máquina.	2.100 Ptas.
370270026	**LOS 20 MEJORES PROGRAMAS PARA EL ZX SPECTRUM** Este libro presenta una selección de excelentes programas en Basic para ser ejecutados en el Spectrum.	1.800 Ptas.
370170004	**PROGRAMACION AVANZADA PARA EL ZX SPECTRUM** Subrutinas, trucos y análisis para mejorar tus programas en el Spectrum.	2.200 Ptas.
370120023	**LAS 40 MEJORES SUBRUTINAS** Compendio de las más útiles rutinas en código máquina, reunidas en un sólo volumen.	1.950 Ptas.
340172971	**ZX SPECTRUM. TEORIA Y PROYECTO DE INTERFASES** Proyectos fáciles de construir para expandir el Spectrum y utilizarlo como computadora de control.	1.690 Ptas.

digital, s.a.

PILAR DE ZARAGOZA, 45 (Semiesquina a Cartagena) - TELS. 246 49 90-246 56 63 - 28028 MADRID - APDO. CORREOS 8287

ENVIAR A:
DIGITAL, S.A.
C/. Pilar de Zaragoza, 45
28028 MADRID

RELLENAR A MAQUINA O CON MAYUSCULAS

Apellidos: .....
Nombre: .....
Dirección: .....
Localidad: ..... **D.P.** .....
Provincia: ..... **Teléfono:** .....

CUPON DE PEDIDO

REFERENCIA	TITULO	CANTIDAD	PRECIO
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....

Forma de pago: Contra reembolso ☐

GASTOS DE ENVIO 200

Tarjeta de crédito: ☐ Visa ☐ Eurocard ☐ Access ☐ Master Card

TOTAL

N.º de tarjeta: ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

Fecha de caducidad: ☐☐-☐☐

FIRMA:

Imprescindible para pago con tarjeta

RELLENAR A MAQUINA O CON MAYUSCULAS

EL SPECTRUM DECIDE

A través de los programas propuestos hasta ahora, hemos podido comprobar que el ordenador ejecuta siempre las instrucciones que componen los mismos de una manera secuencial, es decir, una a continuación de otra, respetando su numeración. Este estado de cosas sólo puede ser modificado mediante la sentencia **GO TO**, que obliga a un cambio en la siguiente línea a ejecutar en el programa.

Vamos a ver ahora como nuestro Spectrum es realmente capaz de «tomar decisiones» en base a comparaciones, y seguir por un camino u otro a partir de una evaluación de condiciones. La estructura general de la instrucción a tal fin es:

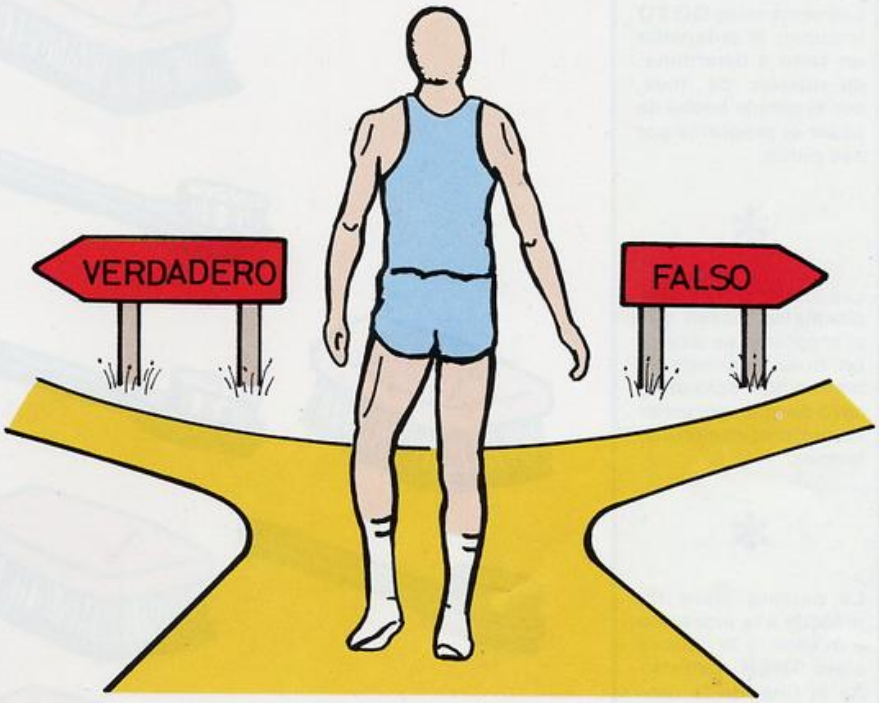
IF condición **THEN** instrucción

IF, que en inglés quiere decir SI (si condicional), precede a la condición a evaluar, y **THEN**, cuya traducción es ENTONCES, sirve para determinar el final de la condición e indicar el comienzo de la instrucción, que ha de ejecutarse sólo en caso de que la evaluación haya sido dada por «verdadera». Si, por el contrario, la evaluación de la condición resulta «falsa», pasa a ejecutarse la siguiente línea, haciéndose caso omiso de la instrucción que sigue a **THEN**. El comportamiento de este tipo de instrucciones podríamos describirlo llanamente de la siguiente manera:

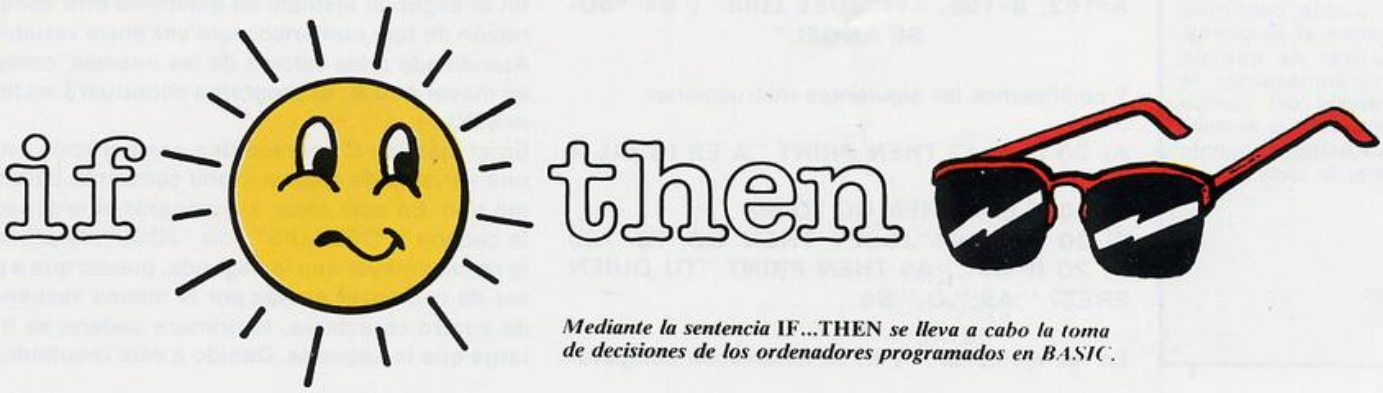
Si se cumple determinada condición ENTONCES haz lo que sigue SINO...

Podemos establecer comparaciones de tipo numérico o de cadena, pero nunca mixtas, es decir, comparar letras con números. Dados dos elementos, podemos realizar las siguientes comparaciones: si son iguales, si el primero es menor que el segundo, si el primero es mayor que el segundo, si el primero es menor o igual que el segundo (no es mayor), si el primero es mayor o igual que el segundo (no es menor), y si el primero es diferente del segundo (no es igual). Con esto se agotan todas las combinaciones posibles. Los símbolos que corresponden a estas comparaciones son:

=, <, >, <=, >= y <>



Las sentencias **IF** *obligan al Spectrum a tomar una decisión entre dos posibles opciones.*



Mediante la sentencia **IF...THEN** *se lleva a cabo la toma de decisiones de los ordenadores programados en BASIC.*

¡!

Las sentencias **GO TO** imponen al ordenador un salto a determinado número de línea, por el simple hecho de pasar el programa por ese punto.

*

Las bifurcaciones condicionales, hacen que el programa se dirija a un nuevo número de instrucción, sólo en el caso de que concurran determinadas circunstancias.

*

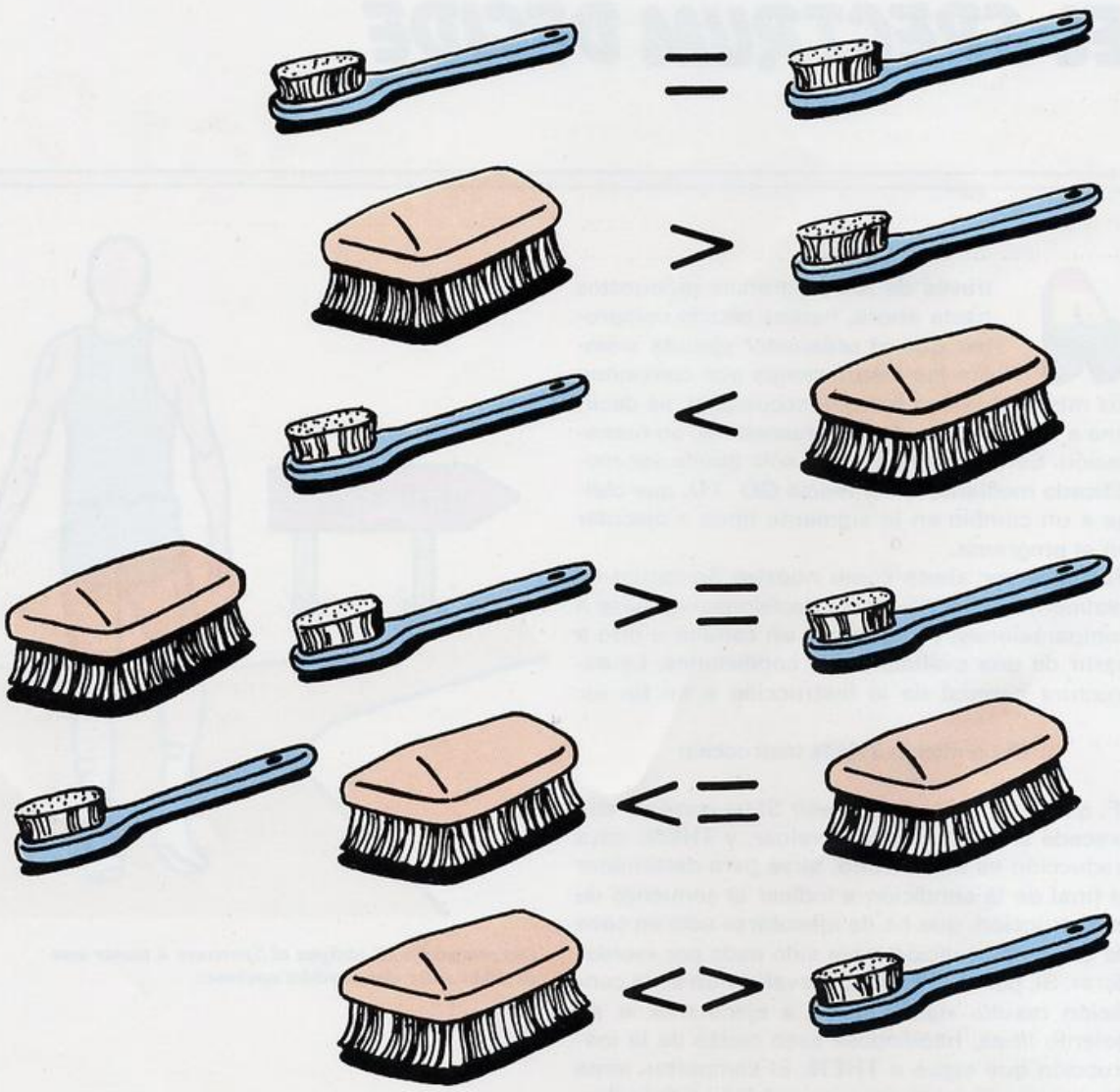
La palabra clave **IF**, precede a la expresión a evaluar, y la palabra clave **THEN**, determina el final de la condición y el principio del comando a ejecutar, si la evaluación de las condiciones resulta verdadera.

*

Las comparaciones deben establecerse siempre entre variables del mismo tipo, es decir, dos numéricas o dos de cadena.

*

Se puede continuar siempre el programa, a pesar de que no haya transcurrido la totalidad del tiempo asignado a la sentencia **PAUSE**, pulsando cualquier tecla.



Son seis las posibles comparaciones lógicas que se pueden realizar en el BASIC de Sinclair.

Como dicho así resulta un poco árido, aplicaremos nuestros conocimientos a algunos ejemplos. Vamos a suponer que, dentro de un programa, los valores de cuatro variables son:

A=162, B=165, A$="JOSE LUIS" y B$="JOSE ANGEL"

Y codificamos las siguientes instrucciones:

```
A) 20 IF A=12 THEN PRINT "A ES IGUAL A 12"
B) 20 IF B>A THEN GO TO 90
C) 20 IF A$<="JOSE" THEN GO TO 100
D) 20 IF B$<>A$ THEN PRINT "TU QUIEN ERES? ";A$;" O ";B$
```

En los ejemplos A y B, se establecen comparaciones entre elementos de tipo numérico, y en C y D de tipo cadena.

En el primer caso, se compara la variable **A** con la constante numérica **12**. Como la variable **A** tiene valor 162 en el momento de acceder el programa a la línea 20, no se ejecutará la sentencia **PRINT** que figura a partir de **THEN**, sino que el programa continuará en secuencia con la línea siguiente a la instrucción 20.

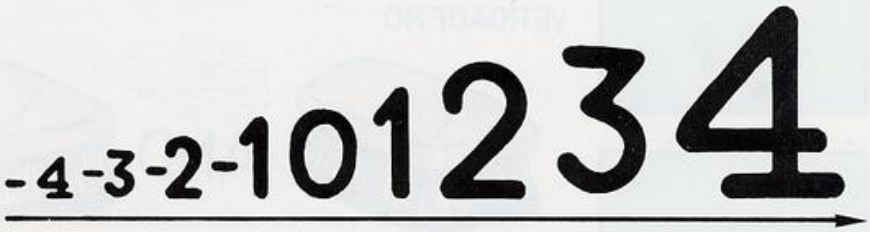
En el segundo ejemplo se establece otra comparación de tipo numérico, esta vez entre variables. Atendiendo a los valores de las mismas, como **B** es mayor que **A**, el programa continuará en la línea 90.

En el ejemplo C aparece una comparación entre una variable de cadena y una constante del mismo tipo. En este caso, al comparar el ordenador la cadena "JOSE LUIS" con "JOSE", la primera le resulta mayor que la segunda, puesto que a pesar de comenzar ambas por la misma secuencia de cuatro caracteres, la primera cadena es más larga que la segunda. Debido a este resultado, el

programa continuará ejecutando la siguiente instrucción en secuencia, y no la sentencia de bifurcación a la línea 100.
Finalmente, en el último ejemplo, podemos ver una comparación entre variables de cadena (**A$** y **B$**). En este caso, **A$** resulta mayor que **B$**, aunque no tiene mucha importancia, ya que lo que tratamos de averiguar en esta ocasión es solamente si las variables son diferentes en contenido, por lo que el programa ejecutará el **PRINT** situado tras **THEN**.
Debemos considerar que cuando tratamos variables de cadena en comparaciones, se utiliza el ordenamiento interno que el Spectrum tiene para cada uno de los caracteres del teclado (el orden alfabético). De esta forma, la letra «B» se considera menor que la «C» etc...
En las comparaciones con variables de tipo numérico, el ordenador tiene en cuenta en primer lugar los signos y luego los valores. En el caso de que los elementos a comparar sean de diferente signo, resulta siempre mayor el positivo. Si los elementos son del mismo signo, se analiza el valor absoluto (valor sin signo) de las variables. Si las cantidades son positivas, se toma como mayor la de valor absoluto superior, y si son negativas, la de valor absoluto inferior.
En el caso de las cadenas de caracteres, esta operación de comparación se realiza de izquierda a derecha, con el convenio de que si dos cadenas son idénticas en lo que se refiere a sus primeros caracteres, es considerada como menor la más corta. De esta forma, la cadena "ABC" se considera siempre menor que la "ABCD".
El juego de caracteres completo del Spectrum puede encontrarse al final de la Obra, aunque normalmente debe bastarnos con saber que los números son para el Spectrum menores que todas las letras, y que las letras mayúsculas son menores que las minúsculas.

LAS DECISIONES SE COMPLICAN

Hasta ahora hemos hablado de las comparaciones más simples, las de tipo unitario, en las que manipulamos un solo par de elementos. Manteniendo la misma estructura, podemos escribir instrucciones mucho más potentes apoyándonos en los OPERADORES RELACIONALES, también llamados OPERADORES LOGICOS.
Estos operadores, establecen la relación que debe cumplirse entre los diferentes elementos



En las comparaciones realizadas con datos numéricos, primero se tiene en cuenta el signo y en caso de ser éstos iguales, su valor absoluto.

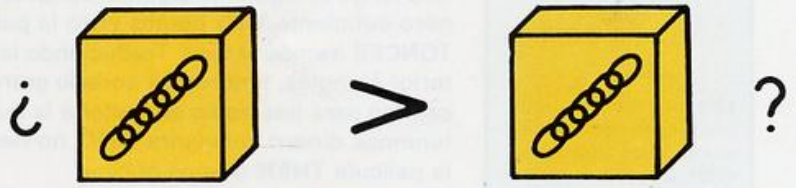
que componen la instrucción de comparación, siguiendo el formato:

IF condición OPERADOR condición ... **THEN** instrucción

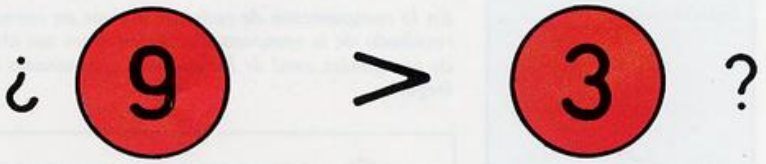
En este formato mantenemos una estructura parecida a la de los **IF** simples, pero haciendo intervenir diferentes condiciones relacionadas con los correspondientes operadores. Los tres tipos de operadores lógicos son: **AND, OR** y **NOT**; su traducción al castellano es **Y, O** y **NO**, respectivamente; su significado lo estudiaremos a continuación mediante unos ejemplos.
Comenzaremos hablando del operador **AND**. Su-

No debemos establecer comparaciones entre elementos de distinta clase (números y cadenas).

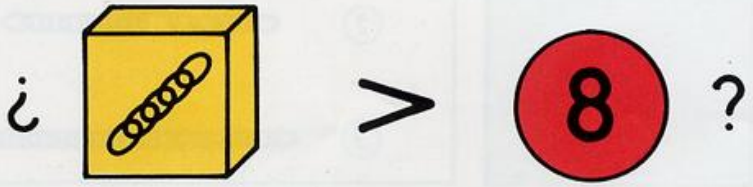
CORRECTO



CORRECTO



INCORRECTO



¡!

Si comparamos dos cadenas que comienzan por la misma secuencia de caracteres, se considera menor la más corta.

*

En la comparación entre variables de cadena, se sigue el orden establecido de códigos de caracteres del Spectrum, que podemos encontrar detallado al final de nuestra Obra o en el manual del aparato.

*

Al comparar variables de tipo numérico, se tiene en cuenta primero su signo y después su valor absoluto.

*

Los operadores lógicos permiten establecer más de una situación a evaluar, para determinar un salto en la secuencia del programa, o la ejecución de determinadas instrucciones.

*

Los posibles operadores de comparación son: =, <, >, <=, >= y <>.

VERDADERO

FALSO

El operador* AND *necesita que los dos sucesos que relaciona sean verdaderos para que la condición se cumpla.

pongamos que tenemos que decidir si vamos o no al cine, dependiendo el hecho de dos condiciones:

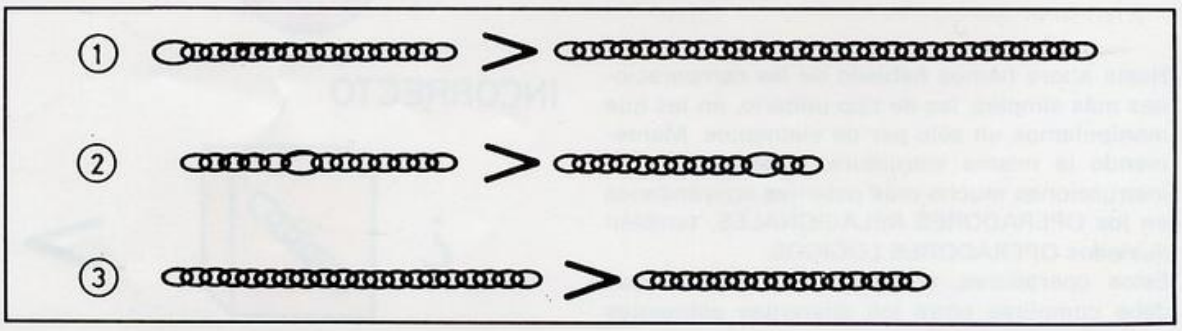
1. TENER DINERO PARA LAS ENTRADAS
2. NO HABER VISTO LA PELICULA

Evidentemente, tenemos que relacionar ambas condiciones con el operador **AND** (Y), ya que las dos deben ser ciertas al mismo tiempo, es decir, no vamos a ir si no tenemos dinero para las entradas ni, por supuesto, si la película la hemos visto ya. Por tanto, la decisión por parte del ordenador se tomará por medio de una instrucción que tenga el siguiente significado: SI tenemos dinero suficiente Y no hemos visto la película ENTONCES iremos al cine. Traduciendo la frase anterior al inglés, tendremos andado gran parte del camino para hacérselo entender a la máquina: **IF** tenemos dinero suficiente **AND** no hemos visto la película **THEN** iremos al cine.
Para terminar de trasladar nuestro problema a un programa BASIC, sólo nos resta hacer comprender al ordenador las condiciones "tenemos dinero suficiente" y "no hemos visto la película". ¡Manos a la obra! Supongamos que **D** es el dinero que tenemos, y **P$** una película que queremos ver. Vamos a escribir el siguiente programa:

```
  10 REM IR O NO IR, HE AQUI EL DI
LEMA
  20 CLS
  30 INPUT "Dinero:";D
  40 PRINT "Tenemos: ";D;" Ptas."
  50 INPUT "Pelicula:";P$
  60 PRINT "y queremos ir a ver: "
;P$
  70 IF D>=350 AND P$<>"TRON" THEN
  GO TO 120
  80 PRINT "NO PODEMOS IR porque "
;
  90 IF D<350 THEN  PRINT "NO tene
mos DINERO ";
 100 IF P$="TRON" THEN  PRINT "YA
la hemos VISTO";
 110 GO TO 130
 120 PRINT "PODEMOS IR SIN PROBLEM
AS."
 130 PRINT '''"PULSA UNA TECLA"
 140 PAUSE 0
 150 GO TO 20
```

En la comparación de cadenas se tiene en cuenta el resultado de la comparación lógica entre sus elementos y, de ser iguales, cual de las cadenas comparadas es más larga.

De la ejecución de este programa podemos sacar algunas conclusiones interesantes. Ante todo diremos que como datos fijos (constantes) hemos



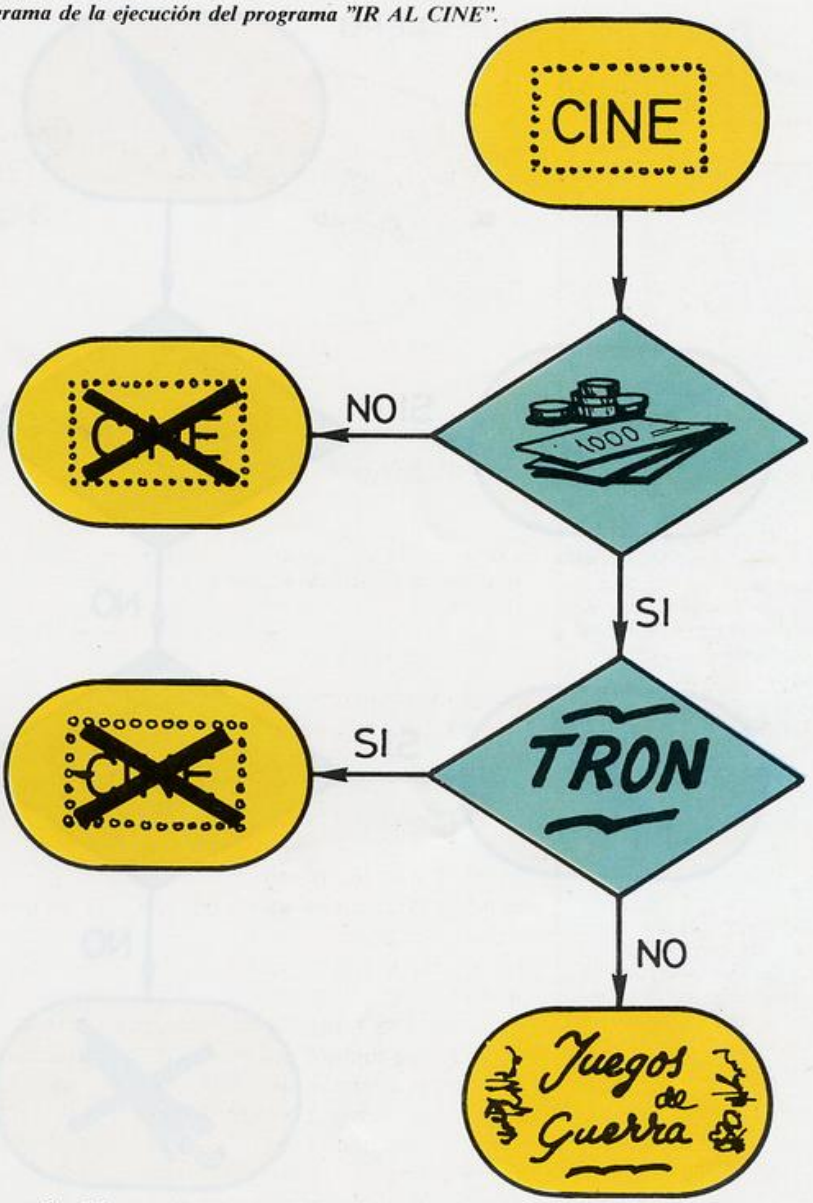
introducido el precio de la localidad y una de las películas que ya hemos visto (la única que el programa va a considerar).
Lo deseable para que el programa fuera un práctico consejero a la hora de resolvernos la tarde del fin de semana, sería poder contar con un archivo en la memoria de todas las películas que hemos visto. En el caso de que seamos muy aficionados al séptimo arte, esto podría traernos problemas de falta de memoria; por otro lado, nuestra única intención es que el programa nos sirva de ejemplo de la forma en que debemos codificar la toma de decisiones para el Spectrum. Como ya sabemos, las líneas 10 y 20 se ocupan, respectivamente, de dar información sobre el contenido del programa y de borrar la pantalla. Las líneas 30 a 60 efectúan la entrada de datos del programa y representan la información en la pantalla. Las variables que utilizamos para esta entrada son: **D** para el dinero de que disponemos y **P$** para el título de la película candidata. En la línea 70 establecemos la más ardua de las decisiones:

70 IF D>=350 AND P$<>"TRON" THEN GO TO 120

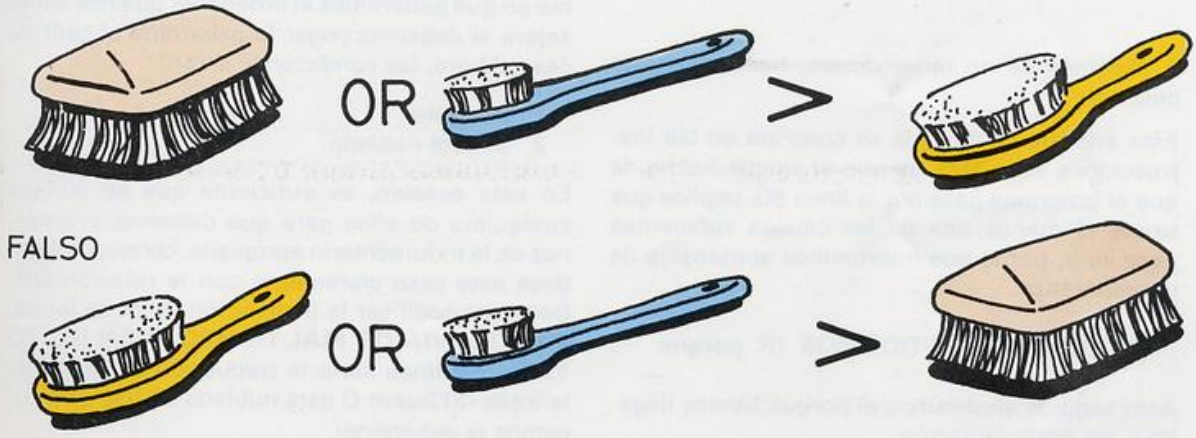
De acuerdo con dos condiciones vinculadas por el operador lógico **AND**, determinamos definitivamente en esta línea si podremos ir o no al cine. Si ambas condiciones resultan ser ciertas (no basta con que lo sea una sola), en base a los datos que introducimos (**D** y **P$**), el programa sigue a partir de la línea 120, señalándonos que no existe ningún problema. En caso contrario, el programa continuará en secuencia, haciendo caso omiso de la sentencia **GO TO**.
Una vez decidido que no podemos ir al cine, entramos en el análisis de los motivos. No existen más que tres posibilidades:

1. No tenemos dinero.
2. Ya hemos visto la película.

Diagrama de la ejecución del programa "IR AL CINE".



El operador **OR** *necesita que uno de los dos (o los dos) sucesos que relaciona se cumplan, para que la condición se cumpla.*



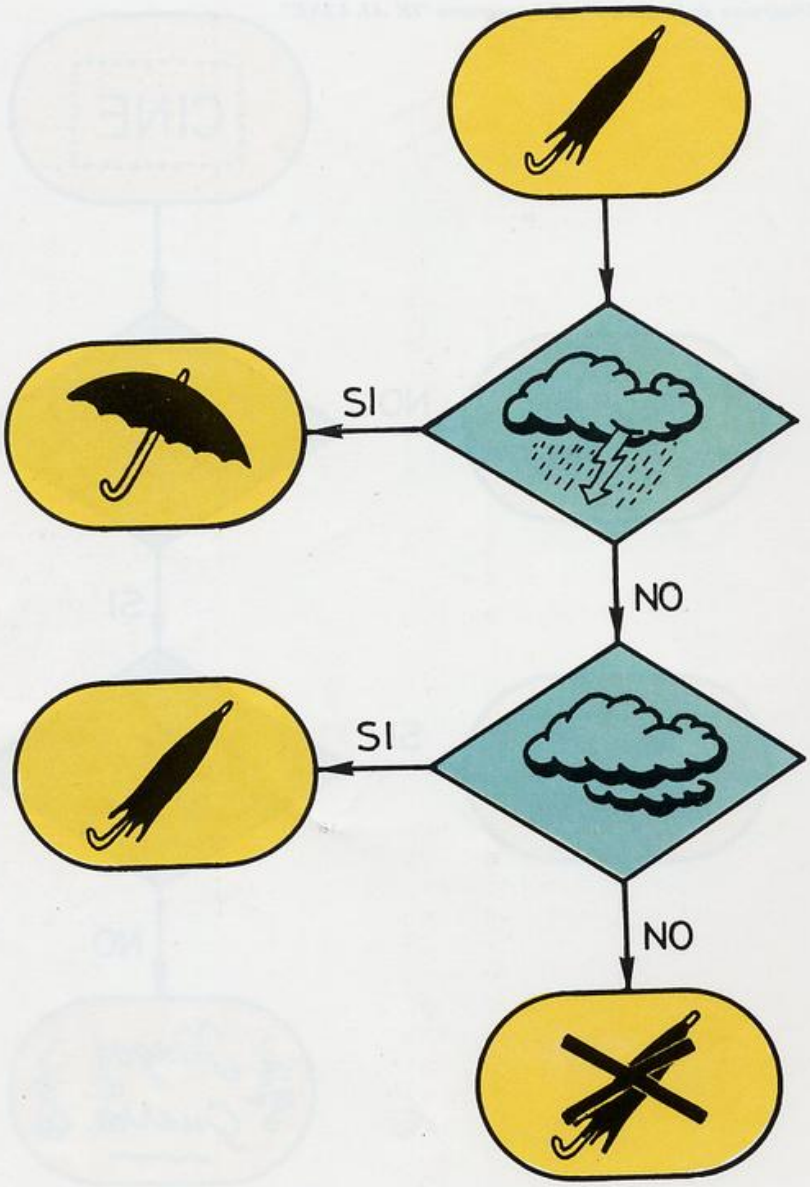


Diagrama de la ejecución del programa "CONDICIONES METEOROLOGICAS".

3. Además de no tener dinero, hemos visto la película.

Este análisis de motivos se concreta en las instrucciones 90 y 100, ya que el simple hecho de que el programa pase por la línea 80, implica que existe al menos una de las causas suficientes para no ir, por lo que imprimimos el mensaje de desesperanza:

```
80 PRINT "NO PODEMOS IR porque ";
```

Acto seguido analizamos el porqué hemos llegado a tan triste situación:

```
90 IF D<350 THEN PRINT "NO tenemos DINERO ";
100 IF P$="TRON" THEN PRINT "YA la hemos VISTO";
```

Si tenemos menos de 350 pesetas, se imprimirá el mensaje "NO tenemos DINERO", y si hemos indicado como candidata la película "TRON", lo hará el mensaje de "YA la hemos VISTO". Puede también darse la posibilidad de que concurran ambas circunstancias, en cuyo caso se imprimirán los dos mensajes.
Entramos por último en el análisis de las líneas 130 a 150, que nos deparan una sorpresa: la aparición en escena por primera vez de la palabra clave **PAUSE.**

```
130 PRINT""PULSA UNA TECLA"
140 PAUSE 0
150 GO TO 20
```

La sentencia **PAUSE** se emplea para detener momentáneamente la ejecución del programa, por un espacio de tiempo determinado: el especificado a continuación de la palabra clave. Dicha pausa se expresa en cincuentavos de segundo, de forma que si deseamos una detención de un segundo deberemos indicar **PAUSE 50,** si dos segundos **PAUSE 100,** etc...
En cualquier caso, el programa continuará al pulsar una tecla, a pesar de no haber concluido su tiempo completo de espera. Otra posibilidad es la de provocar una parada en el programa por tiempo indefinido, de la cual la única forma de salir, para que la ejecución continúe, es pulsando una tecla. Esta última opción es la que lleva un cero como parámetro de la sentencia **PAUSE (PAUSE 0).**
En nuestro ejemplo, nos servimos de esta instrucción para provocar una pausa indefinida, en la que se muestran los resultados impresos y se solicita que pulsemos una tecla para volver a ejecutar el programa.
De forma similar a la que hemos actuado en el programa propuesto anteriormente, podemos diseñar otro muy parecido en el que emplearemos el operador lógico **OR** (O). Podría ser un programa en que pidiéramos al ordenador que nos aconsejara si debemos coger la gabardina al salir de casa. Ahora, las condiciones serán:

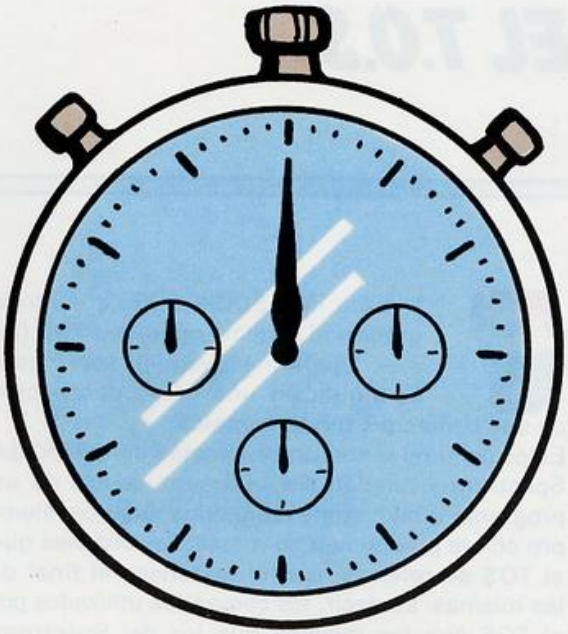
1. Si llueve.
2. Si está nublado.

En esta ocasión, es suficiente que se cumpla cualquiera de ellas para que debamos proveernos de la indumentaria apropiada, correspondiéndose este caso claramente con la relación **OR.** Debemos codificar la instrucción 70 de la forma: **70 IF LLUVIA OR MAL TIEMPO THEN GO TO 120.** Esta línea sería la traducción al BASIC de la frase «SI llueve O está nublado ENTONCES cogemos la gabardina».

Para facilitar la codificación de la instrucción hemos escogido dos variables cuyo nombre está directamente relacionado con el valor que contienen: LLUVIA y MAL TIEMPO; estas variables son numéricas, de forma que para la entrada de datos en los **INPUT**, debemos adoptar el convenio de considerar como «1» si el fenómeno atmosférico se produce y como «0» si no. Una novedad en la instrucción 70 es la de que no aparece el símbolo de igualdad detrás de las variables:

LLUVIA=1 y MAL TIEMPO=1

Esto podemos hacerlo gracias a que el ordenador supone que, al no existir el símbolo de igualdad, es nuestro deseo tratar estas variables numéricas como variables lógicas. Las variables lógicas adoptan para la comparación dos únicos valores: VERDADERO (diferente de 0) y FALSO (0).
Al escribir la instrucción de esta forma, conseguimos que el programa bifurque a la línea 120 cuando LLUVIA o MAL TIEMPO sean, las dos o cualquiera de ellas, diferentes de cero, es decir, verdaderas en sentido lógico.
La codificación del resto del programa será muy simple, puesto que sólo deberemos realizar modificaciones en las sentencias **PRINT** del programa, según las condiciones meteorológicas que se hayan producido, de manera similar a como procedimos con nuestro programa del cinéfilo.
Hablaremos por último del operador lógico **NOT**, que es el complementario (negación lógica) de los operadores lógicos que hemos visto hasta ahora. La verdadera utilidad del operador **NOT** es la de clarificar nuestras intenciones respecto a las condiciones que se avalúan en un programa. Expliquemos esto más claramente: supongamos que determinado suceso depende de una sola condición; por ejemplo, que ir a trabajar depende exclusivamente de que no sea domingo. La sentencia **IF** que deberíamos utilizar para codificar este suceso sería: SI hoy es diferente de domingo ENTONCES vamos a trabajar.
Para traducir esta instrucción al lenguaje BASIC, emplearemos una variable lógica que denominaremos DOMINGO, cuyo valor será cero en caso de que no sea domingo, y uno si hoy es domingo. La instrucción de decisión se podría concretar de la siguiente manera: **IF DOMINGO <> 1 THEN vamos a trabajar** o **IF DOMINGO = 0 THEN vamos a trabajar**. En este caso hubiera quedado mucho más clara una decisión del tipo. «Si hoy NO es domingo ENTONCES vamos a trabajar»; la traducción directa de esta frase al BASIC sería: **IF NOT DOMINGO THEN vamos a trabajar**. De lo dicho, sin duda habremos sacado la siguiente conclusión: **NOT** variable, tiene exactamente el mismo efecto que: variable **<> 0**. Para terminar, diremos que se pueden construir complicadas estructuras de comprobación de condiciones, utilizando combinaciones de los operadores lógicos. Del mismo modo que para las operaciones matemáticas, el Spectrum tiene un orden de prioridades de ejecución de estos operadores, de forma que el **AND** (producto lógico) se ejecuta con prioridad sobre el **OR** (suma lógica). En cualquier caso, pueden emplearse los paréntesis para alterar el orden en que se ejecutan normalmente las operaciones, de manera similar a lo que ocurre con las operaciones matemáticas. De todas formas, sobre este tema ahondaremos próximamente.



La sentencia **PAUSE** *establece una detención en la ejecución del programa.*

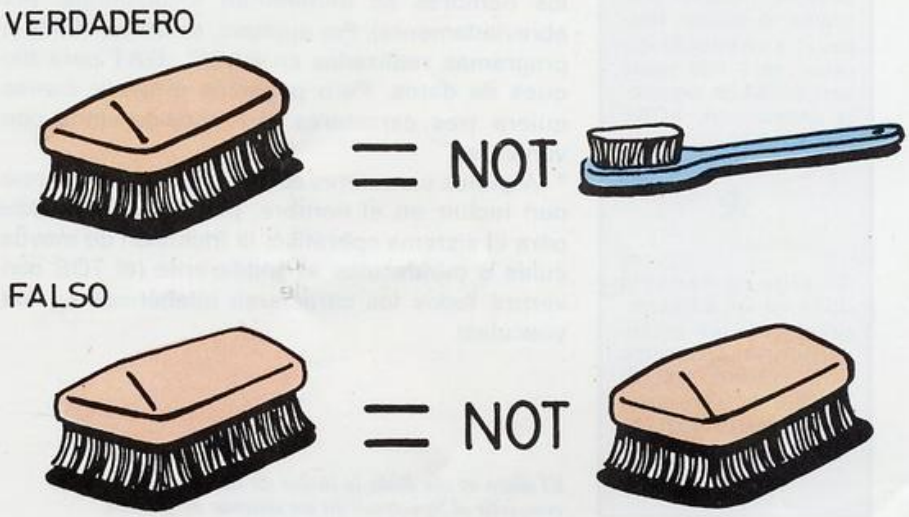
¡!

Con **PAUSE 0**, se provoca una parada indefinida, de la que sólo se puede salir cuando el usuario pulsa una tecla.

*

La instrucción **PAUSE**, produce una interrupción en la ejecución del programa, por el espacio de tiempo especificado como argumento de la sentencia, expresado en cincuentavos de segundo.

El operador **NOT** *es equivalente en su efecto a la comparación "distinto que".*



EL T.O.S.

EL disco INVESDISK 200 se controla gracias al sistema operativo TOS (Timex Operating System); veremos a continuación una breve descripción de sus comandos fundamentales.
Estos comandos son una extensión del BASIC del Spectrum y pueden ser utilizados dentro de un programa o bien como comandos directos, siempre con la precaución de marcar las órdenes que al TOS se refieran, con un asterisco al final de las mismas. Es decir, los comandos utilizados por el TOS son los mismos que los del Spectrum, pero marcados con el asterisco (aunque algunos no realizarán la misma función).

NOMBRES DE PROGRAMAS Y FICHEROS

Podemos utilizar cualquier nombre que deseemos con las siguientes limitaciones:
* No tendrán una longitud mayor que ocho caracteres; opcionalmente, a continuación del nombre y separados por un punto (.) se pueden incluir otros tres caracteres, que nos sirven para diferenciar tipos de programas; estos apéndices de los nombres se denominan extensiones, **(ext** abreviadamente). Por ejemplo, se utiliza **.BAS** en programas realizados en BASIC **.DAT** para bloques de datos. Pero podemos emplear cualesquiera tres caracteres que consideremos convenientes.
* Algunos caracteres como ↑, +, . o ?, no se pueden incluir en el nombre, por estar reservados para el sistema operativo; la inclusión de mayúsculas o minúsculas es indiferente (el TOS convertirá todos los caracteres alfabéticos en mayúsculas).

Cuando almacenamos información mediante un casete, no hay manera rápida de saber lo que tenemos grabado, ni lo que esto ocupa en una determinada cinta, a no ser que lo hayamos anotado cuidadosamente. En la unidad de discos, este problema se resuelve fácilmente con el uso de **CAT ***. Este comando nos va a informar, mediante una impresión en la pantalla, de todos los programas que se encuentran en la cara del disco con la que estamos trabajando. Proporciona además los siguientes datos: la longitud de cada uno de ellos, el número de Kbytes ocupados redondeados al entero superior más próximo, si están protegidos o no, el número de Kbytes que todavía quedan libres y los utilizados y, cuando se trata de un fichero, si está abierto o no.

El disco es sin duda la mejor de las soluciones para convertir el Spectrum en un sistema de gestión.

BITS

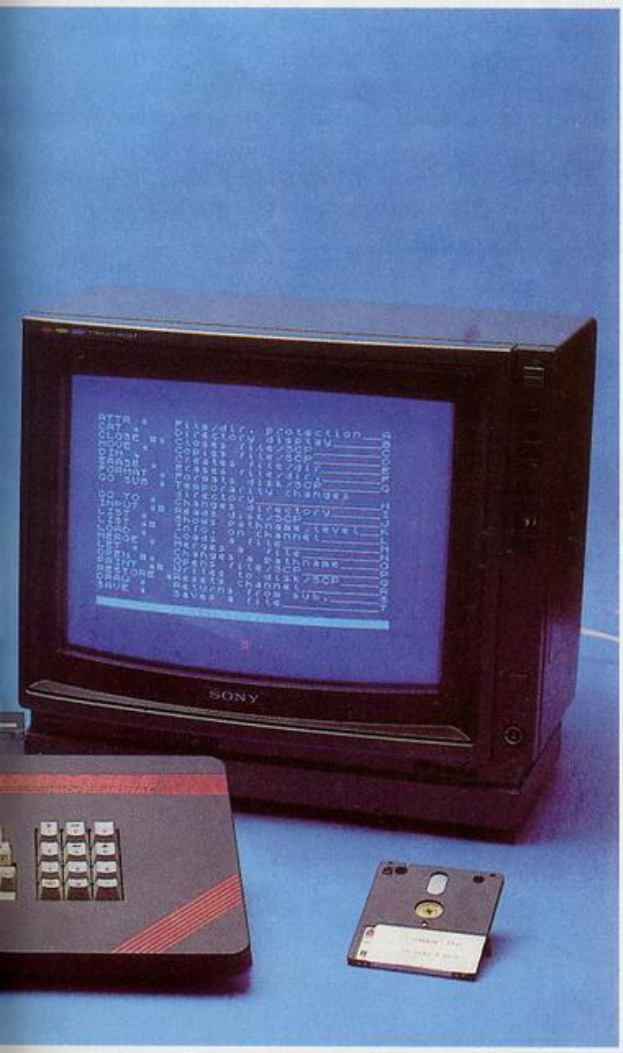
En algunas ocasiones, por ejemplo durante la ejecución de un **INPUT**, hacer desaparecer el cursor puede mejorar la presentación de nuestro programa, o incluso despistar a un posible «pirata»; para ello basta con incluir la siguiente instrucción: **POKE 23617,226**

*

Si alguno todavía duda de los sobresaltos que puede provocarnos el Spectrum, que teclee: **POKE 23659,1**. ¡Tranquilo!... pulsa **ENTER**.

El sistema operativo puede organizar los catálogos en forma jerárquica, es decir, como si se tratara de un árbol invertido, con la raíz en la parte superior y las ramas hacia abajo, Si al ajecutar el comando **CAT *** encontramos ficheros con la extensión **.DIR,** esto indica que tenemos un directorio dentro de otro en alguna de las distintas ramas del árbol. El TOS permite la creación de dieciséis ficheros de direcciones en cada cara del disco y de ocho niveles distintos en los que anidar los archivos. Se puede acceder a las ramas del árbol mediante los comandos **GOSUB *** o **GO TO * "nombre.DIR"**.

Para la carga desde el disco se utiliza la sentencia **LOAD,** cuya sintaxis es: **LOAD * "nombre"** o, si hemos hecho uso de los tres caracteres opcionales de extensión, **LOAD * "nombre.ext"**. Su empleo es muy similar a la sentencia homónima para el casete. Es importante tener en cuenta que con el disco no podremos cargar ficheros sin indicar el nombre, tal como haríamos con el casete; por tanto, el TOS no identificará la orden **LOAD * ""**. Siempre hará falta especificar el nombre del programa. Esto no produce ningún contratiempo pues, aunque no recordemos el nombre exacto, va a ser fácil averiguarlo utilizando el comando **CAT ***.



SENTENCIAS INCORPORADAS POR EL TOS

ATTR *	INPUT *
CAT *	LIST *
CLOSE # *	LOAD *
MOVE *	MERGE *
DIM *	LET *
ERASE *	OPEN # *
FORMAT *	PRINT *
GOSUB *	DRAW *
GOTO *	SAVE *

SAVE * es el comando inverso a **LOAD ***. Hay que destacar que cuando grabemos un programa o un fichero al que queremos dar un nombre que ya figura en el directorio del disco, el TOS nos advertirá de esta circunstancia mediante el mensaje:

"nombre del programa" **already exists**
Superse de (Y/N)?

Si nuestra respuesta es **Y,** borrará el programa existente ya mencionado y grabará la nueva versión desde la memoria. Si por el contrario pulsamos **N** ignorará la operación **SAVE ***.

De forma similar al casete o al ZX Microdrive podemos utilizar **SAVE * "nombre.ext" LINE xxxx,** que tiene el efecto de grabar nuestro programa con autoejecución a partir de la línea **xxxx.** También podremos utilizar los sufijos **SCREE N$, CODE** o **DATA** para almacenar pantallas, bloques de memoria o dimensionados, respectivamente.

El funcionamiento de **MERGE *** es también análogo al correspondiente comando del BASIC Sinclair estándar: mezcla un programa nuevo y sus variables, con el que se encuentre presente en

¡!

Recordemos que los dos tipos fundamentales de ficheros se denominan SECUENCIALES y ALEATORIOS. Estos se diferencian por el modo de acceso a cada uno de sus campos.

*

Para el acceso al cualquier campo de un fichero secuencial, es necesario leer previamente todos los campos anteriores a él.

*

El acceso a un fichero aleatorio se realiza libremente a cualquiera de sus campos, designándolo por su número de orden dentro del fichero, de forma similar a los dimensionados BASIC.

BITS

El espacio total de memoria disponible, incluyendo el dedicado a gráficos definidos por el usuario, es fácil de calcular mediante una sencilla llamada a la ROM del sistema. En cualquier momento podemos ejecutar la siguiente instrucción: **PRINT 65536-USR 7962;'' Bytes libres''**. Si el Spectrum es de 16 K, sustituiremos el 65536 por 32768.

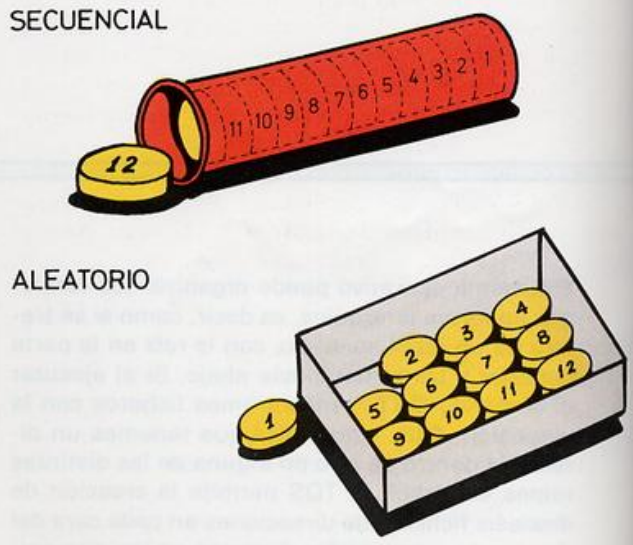
*

Incluir sonidos dentro de nuestros programas los hará sin duda mas entretenidos. A continuación se listan tres efectos sonoros que podremos aplicar muy fácilmente.
Caída: **10 FOR N=0 TO 60 STEP .5: BEEP .03,N: NEXT N**
Sirena: **10 BEEP .35,24: BEEP .35,12: GOTO 10**
«Click-Clack»: **10 FOR N=0 TO 200: BEEP .01,20: PAUSE 10: NEXT N**

*

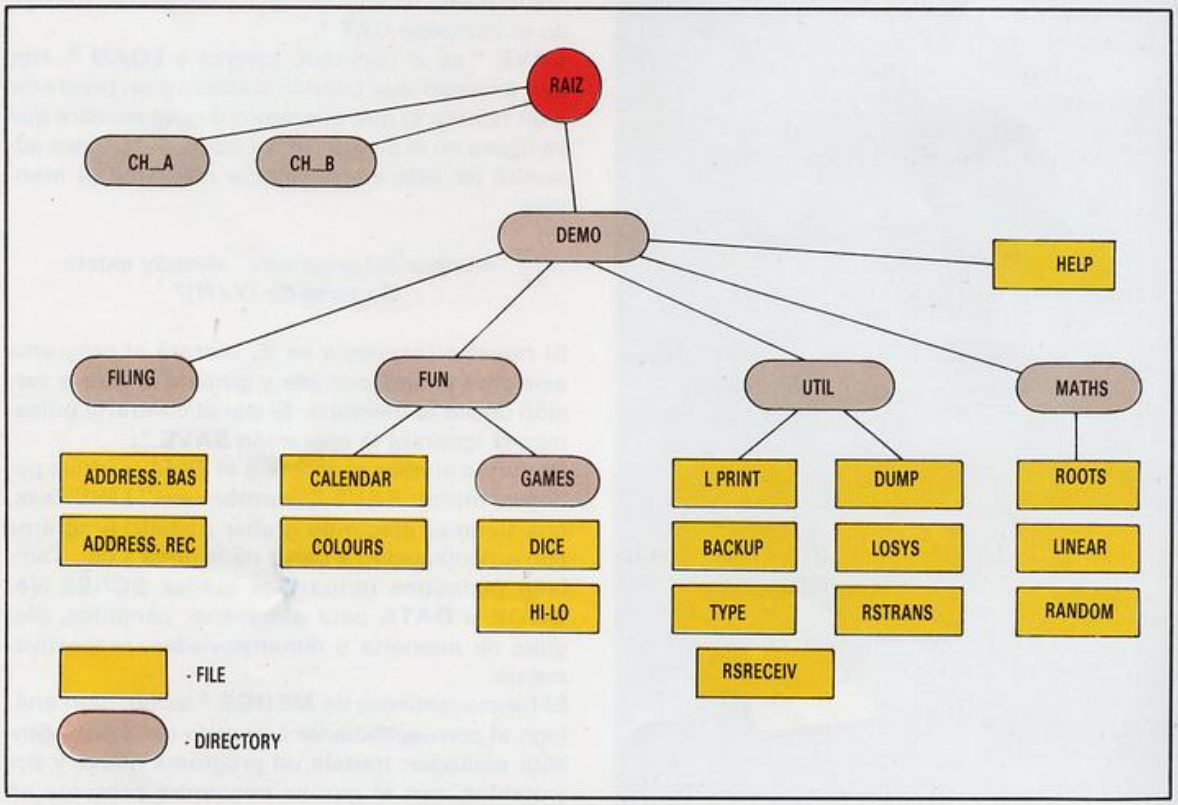
Si deseamos asegurarnos desde dentro de un programa, que una toma de datos es realizada en mayúsculas, nuestro problema se reduce a ejecutar la siguiente instrucción: **POKE 23658,8**. Si por el contrario lo que queremos es conectar el modo minúsculas, sólo tenemos que cambiar el valor del **POKE** a cero.

la memoria del Spectrum. Lógicamente, no es posible realizar un **MERGE** con un bloque de bytes o datos, sino sólo con programas. La sintaxis de la instrucción es: **MERGE * ''nombre.ext''**.
El comando **FORMAT** es probablemente el más importante de todos, puesto que es obligatoria su ejecución previamente a la utilización de cualquier disquete virgen.
Esta es la única sentencia del TOS en la que se debe especificar el nombre de la unidad de disco con la que vamos a trabajar. Para la identificación del *drive* (unidad de disco) afectada, utilizaremos las primeras letras del abecedario, asignándosele la **A** a la primera unidad, **B** a la segunda, **C** a la tercera y **D** a la cuarta. La sintaxis completa de la instrucción es: **FORMAT * ''nombre de la unidad '' TO ''nombre del disquete''**. Esta sentencia se introduce por el teclado como comando directo.
Cuando se ejecuta el comando **FORMAT ***, el sistema operativo divide la superficie del disquete insertado en la unidad de disco en cuarenta pistas de cuatro Kbytes, y cada pista en sectores. Es decir, lo prepara para que la información que allí va a ser almacenada, se distribuya ordenadamente y luego pueda tener fácil acceso a ella. El «formateado» de un disco dura poco más de treinta segundos, y el sistema operativo que se copia en el disquete ocupa 16 K. Otras 4 K se utilizan para el directorio o catálogo del mismo; de ahí que de las teóricas 160 K de cada cara del disquete, sólo podamos aprovechar 140 K.
Es importante tener cuidado con este comando, pues al ejecutarlo se destruye totalmente el contenido del disquete. De todos modos, el TOS se cerciora de que queremos «formatear» el disque-



El acceso a las pastillas de un tubo es secuencial; por el contrario, de hallarse dispuestas en una caja, podríamos hablar de acceso aleatorio.

El disquete de demostración que se suministra con la unidad de disco, utiliza varios directorios dispuestos en un sistema de árbol.



te, realizando una pregunta para confirmarlo antes de ejecutar el comando. Así mismo, cuando sólo disponemos de un *drive,* el TOS nos irá informando de cuando debemos retirar el disquete a «formatear» para insertar el del sistema operativo.

El comando para el borrado de ficheros es **ERASE** * y su sintaxis correcta es: **ERASE * ''nombre.ext''**. Antes de ejecutarlo, el sistema operativo realizará una pregunta de confirmación de la forma:

ERASE ''nombre.ext'' Y/N?

Para que la sentencia **ERASE** * tenga efecto es necesario que el fichero que queremos eliminar no se encuentre protegido. Sabremos si lo está ejecutando **CAT** * y mirando en la columna marcada con **P;** si encontramos una letra **P** en la línea correspondiente al fichero a borrar, esto será indicativo de que éste se encuentra protegido. Uno de los comandos del TOS que en BASIC Sinclair estándar tienen un significado absolutamente diferente es **ATTR** *. Este comando permite proteger, desproteger, hacer invisible o visibilizar un determinado fichero del directorio. Su sintaxis es:

ATTR * ''nombre.ext''p - protege el fichero.
ATTR * ''nombre.ext''i - hace invisible el fichero.
ATTR * ''nombre.ext''u - desprotege el fichero.
ATTR * ''nombre.ext''v - visibiliza el fichero.

El que un fichero o programa se haga invisible mediante el uso de **ATTR ***, solamente significa que no aparecerá en el directorio cuando ejecutemos **CAT ***; esta medida es similar a la adoptada por el ZX Microdrive cuando el nombre de un fichero comienza por el carácter cero **(CHR$ 0)**.

OTRAS PARTICULARIDADES DEL TOS

Una de las posibilidades que ofrece el TOS es la de incluir en el nombre del programa o fichero, los caracteres + o ? cuando usamos determinados comandos. Así por ejemplo, si ejecutamos: **CAT * ''+.BAS''**, obtendremos un directorio con el nombre de todos los ficheros de extensión **.BAS;** con **CAT * ''+''**, obtendremos los ficheros a los que no hemos dado extensión; y **CAT * ''R???????.+''**, buscará una lista de los archivos

Las sentencias básicas utilizadas por el TOS para el control del disco, son muy similares a las empleadas por el ZX INTERFACE 1, para la gestión del Microdrive.

o programas que comienzan por R. Estos caracteres se pueden utilizar con los comandos **ERASE *, ATTR *, MOVE *, LET *** y **CAT ***.

Mediante la instrucción **LET * ''nombre antiguo.ext'' TO ''nuevo nombre.ext''**, conseguiremos cambiar el nombre de un fichero por otro nuevo. Continuando con las alteraciones de denominación, **GOSUB * ''nombre de la unidad''d** se utiliza para cambiar la unidad de disco a que se hace referencia, donde el nombre de la unidad se designa con las letras A, B, C o D.

El comando* CAT * *proporciona una amplia información sobre el contenido de un disquete.

```
.DEMO
Level 0  Drive A

Name      Typ      Size  Alloc  S  P
------------------------------------
ALFA      DIR     41779    50K     P
COCHES             40     1K
CHUZOR            100     1K
FCHUZOR           360     1K     P
AUTOS     ENT     142     1K     P
AUTOS     LEC     135     1K     P
DIRECC    DAT   12000    12K
2+2               2600     3K
DIRECC    DEM     929     1K
CHULIN    DIR    2094     5K
INTERF    BAS   10118    10K     P
IBISA     CAR      28     1K     P
AFON              337     1K
79585     DIR      35     5K

Max 140K   Cur 93K   Res 47K

0 OK, 0:1
```

BITS

Este sencillo programa proporcionará dibujos lineales en la pantalla, siempre diferentes y sin el fantasma del desagradable mensaje *Integer out of range.*

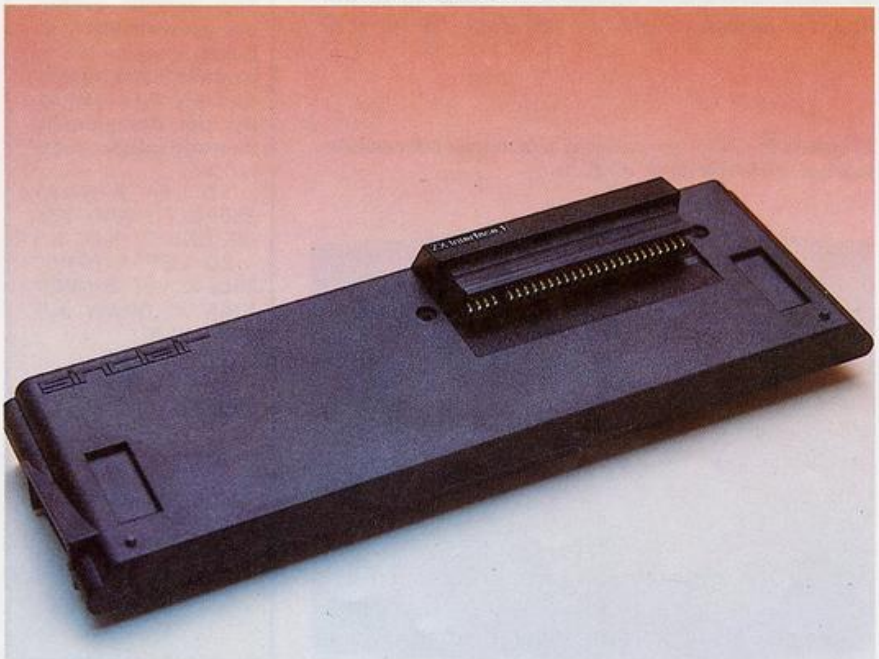
```
10 LET X=RND*255: LET Y=RND*175: INK RND*7: PLOT X,Y
20 LET A=(RND*255)-X: LET B=(RND*175)- Y: DRAW A,0: DRAW 0,B
30 GOTO 10
```

Prueba ahora eliminando la línea 20.

Una vez realizada la operación de «formateado», el disquete tiene 140 Kbytes netos para el usuario por cada cara.

Para finalizar con este breve repaso al TOS, mencionaremos que la copia de programas se consigue mediante **MOVE * "nombre1.ext" TO "nombre2.ext"**. La potencia del sistema operativo del INVESDISK es aún mayor que lo aquí expuesto. Se dispone de un total de dieciocho comandos destinados al disco, gracias a los cuales éste se convierte en un periférico muy útil y eficaz. Por último cabe añadir que el apoyo que presta el TOS al programador, va desde las preguntas de confirmación ante operaciones tan delicadas como el **FORMAT** o el borrado, hasta la emisión de un total de cuarenta y seis tipos diferentes de mensajes de error, todos ellos detallados en el manual de operación que acompaña a la unidad de disco.

El sistema INVESDISK 200 es incompatible con el ZX INTERFACE 1, y por tanto no permite la utilización simultánea de Microdrives.



FICHEROS DE DATOS

La principal ventaja de utilizar la unidad de discos frente a otros sistemas de almacenamiento, es la posibilidad de manejar ficheros de acceso aleatorio (en el ZX Microdrive, como con algunos sistemas de cinta, sólo se pueden manejar ficheros sencuenciales).

Cada uno de los componentes de un fichero se denomina *RECORD* o CAMPO (en ocasiones REGISTRO); la forma de acceder a cada uno de los campos de un fichero, es la que marca el tipo de éste. Fundamentalmente, existen dos tipos de ficheros: SECUENCIALES y ALEATORIOS.

Los primeros se caracterizan porque para acceder a cualquiera de sus campos es necesario haber leído previamente todos los anteriores. Imaginemos un tubo de pastillas dulces; si aquella cuyo sabor estamos buscando se encuentra al final del tubo, nos veremos obligados a sacar todas las pastillas anteriores para llegar hasta la que buscamos. Este es el tipo de acceso secuencial, utilizado en el ZX Microdrive.

Cuando podemos acceder a cada uno de los *records* de un fichero sin pasar por los anteriores, decimos que este fichero es de acceso aleatorio. Siguiendo el ejemplo anterior, podríamos imaginar que las pastillas han sido distribuidas en una caja formando un solo piso, gracias a ello podremos llegar directamente a cada una de ellas sin necesidad de desplazar el resto. Este tipo de ficheros es sin duda el más conveniente puesto que, aunque más lentamente, simula el tipo de acceso que en el BASIC se tiene a los dimensionados.

Para el control de ambos tipos de ficheros, el TOS emplea las sentencias **OPEN # ***, **CLOSE # ***, **INPUT *** y **PRINT ***. Cuando se trata de ficheros aleatorios, el *record* a acceder dentro del fichero se señala a continuacion del sufijo **AT.**

AVENTURA INTERMINABLE

ENTRE los programas que gozan de mayor aceptación dentro de los destinados a microordenadores, se cuentan los del tipo AVENTURA. Gracias a ellos nos trasladamos a un mundo irreal, en el que el destino de un personaje y el final de una historia depende directamente de nosotros, de nuestras decisiones y comportamiento. Este argumento, muy similar al de la conocida película de fantasía «La Historia Interminable», tiene también brillantes exponentes en el software destinado a nuestro aparato. Tal es el caso de programas tan conocidos como «El Hobbit» **(Melbourne House)** u otros muchos. Sin embargo, no debemos pensar que la realización de este tipo de programas nos está vedada a los desconocedores del código máquina. También en BASIC se pueden conseguir algunas «aventuras» interesantes.

Sirva como botón de muestra el programa que figura a continuación; aunque hemos de tener en cuenta que se trata de una versión reducida, puesto que de no ser así, la gran ocupación de espacio haría imposible su publicación.

LA HISTORIA

Este juego-aventura nos permitirá explorar nuevos y extraños parajes, encontrar tesoros, luchar contra monstruos; tendremos que sobrevivir en un inhóspito y engañoso desierto. En pocas palabras: si queremos acabar con vida, hemos de resolver todos y cada uno de los problemas que se nos plantearán en el nuevo mundo al que vamos a entrar. Obviamente, nosotros seremos los protagonistas de esta «historia interminable».

El Spectrum nos describirá los lugares que vayamos visitando y los objetos que aparezcan. Comenzaremos en un determinado lugar del condado de Kerkovia, perteneciente al territorio de nuestro gran enemigo: ASHAMM, el dragón verde.

Nuestro objetivo es hallar los tesoros de la princesa Shirka y depositarlos en el punto de partida. Para ello tendremos que hacer uso adecuado, en el momento oportuno, de los objetos que encontremos a nuestro paso, y con ellos salir airosos de los múltiples problemas con los que nos vamos a topar. Pero... ¡CUIDADO!, los sangrientos piratas del país vecino están al acecho.

Hemos de tener en cuenta, que todos los objetos que nos encontramos, por extraños que nos parezcan, tienen una utilidad y puede que nos sean necesarios en algún momento de la aventura. Veremos un ejemplo a modo de pista: si intentamos atravesar el lago del oasis del desierto, sin proveernos de un tubo hueco que poder utilizar para respirar mientras andamos por el fondo, pereceremos en el intento.

Lamentablemente, no podremos portar todos los objetos que nos vayamos encontrando, por lo que deberemos seleccionar con cuidado nuestro equipaje. Ante todo hemos de utilizar la imaginación, tengamos en cuenta que somos un personaje de fantasía que ¡hasta puede comer lámparas! Sin embargo, frenemos en lo posible nuestra hambre pantagruélica, puesto que esa misma lámpara nos puede servir para ir hacia la parte norte de la casa en que comienza nuestra aventura, donde reina la oscuridad; al intentar entrar allí, siem-

¡!

Las subrutinas han sido situadas al comienzo del programa, para conseguir un acceso más rápido a las mismas.

*

Los programas del tipo AVENTURA no han de estar necesariamente escritos en código máquina; también en BASIC podemos conseguir resultados bastantes aceptables.

*

El factor aleatorio influye en el programa, gracias a lo cual cada nueva aventura será diferente a las anteriores.

pre que no vayamos provistos de la lámpara, el programa emitirá el mensaje "ALGO TE FRENA" (la oscuridad), no permitiéndonos el acceso al lugar.

Bien, ya basta de pistas. Antes de meternos de lleno en el estudio del manejo del programa, aceptemos un último consejo: los programas de aventura precisan de mucha paciencia e imaginación. No hay que desesperarse por tanto si las dunas del «desierto itinerante», en su incesante viaje, bloquean nuestras posibles salidas. En último caso siempre podemos interrumpir el programa y recomenzar con **RUN**. Recordad: «No es mejor pescador el que tiene la caña más larga, sino el que tiene la paciencia más grande».

Los comandos a utilizar para «actuar» en nuestra historia consisten en una o dos palabras. Habrá que teclear en primer lugar el verbo (es decir, la acción que deseamos realizar), y a continuación (no siempre) nombrar el objeto con el que vamos a realizar dicha acción. En algunos casos el ordenador nos pedirá más información después de haber dado una orden; por ejemplo: si decimos "MATA DRAGON", el Spectrum nos preguntará con qué deseamos destruir a nuestro acérrimo enemigo; en este caso hay que introducir el nombre del arma o herramienta con que deseamos hacerlo, o nada si queremos utilizar nuestras propias manos.

Ante todo hay que introducir todos los comandos correctamente, esto es, bien deletreados. Si alguna letra de la orden a ejecutar estuviese equivocada, el ordenador hará caso omiso, y te presentará en la pantalla el mensaje "NO COMPRENDO", seguido del comando erróneo introducido.

Para ahorrarnos trabajo y evitar errores de escritura, la mayoría de las palabras-comando pueden ser abreviadas a 1, 2 ó 3 letras iniciales. Por otra parte, es conveniente dibujar un mapa de nues-

tro recorrido, para tener una referencia del punto en donde nos encontramos, evitando así nuestra desorientación.

Como pista, y sin que sirva de precedente, diremos que existe un oasis al sureste de la posición inicial, y que las dunas del desierto pueden bloquearte las salidas que hubieses descubierto con anterioridad. En la tabla adjunta se detallan todos los comandos que el ordenador reconoce. Con suerte, paciencia e imaginación, y valiéndose sólo de las palabras-comando reseñados más adelante, podremos llegar a conseguir nuestro objetivo.

La información que el ordenador necesita de cada objeto consiste en su propia descripción: dónde está, qué clase de objeto es (tesoro, monstruo, llave...), cómo es de fuerte, etc... Este sistema permite una mayor flexibilidad para la actuación del programa, según la actitud que el protagonista adopte ante el objeto en cuestión.

Tanto los objetos como los lugares y comandos, se encuentran codificados en el programa mediante un número, que supone un diferente significado en cada caso. Cuando se trata de objetos, éstos pueden poseer tres valores diferentes:

* valor -1: objetos que no se pueden ver o usar, u objetos cogidos con anterioridad.
* valor 0: objetos que el jugador lleva consigo.
* valor 1: resto de los objetos que no son utilizados en ese momento por la situación del jugador.

Los números que han sido asignados a los verbos (coge, deja, come...) son utilizados por el programa para localizar la subrutina que realiza la acción de dicho verbo. La codificación de cada comando se corresponde con la posición que ostenta cada uno en la **DATA**. Así pues, el primer verbo llevará el número uno, y la subrutina de acceso se encontrará a partir de la línea de programa número 1469 + 31 * (número del verbo). Ya sólo resta añadir que el programa tiene asignados unos números claves, los cuales son empleados por el ordenador para localizar el lugar o escenario en el que discurre la acción presente; estos valores comienzan en el uno y van incrementándose hasta llegar al treinta.

```
  1 REM ************************
  2 REM * J.M.MAYORAL SERRANO *
  3 REM ************************
  5 GO SUB 9995
 10 BORDER 1
 12 PAPER 1: INK 9
 13 CLS
 30 LET L=1
 40 GO SUB 900
 50 POKE 23692,255
 51 RESTORE L+LS: READ M$
 52 PRINT "TE ENCUENTRAS ";M$
 53 PRINT : LET Q$="ESTAS VIENDO: "
 54 GO SUB 2000
 60 BEEP .1,10
 61 PRINT
 65 REM
 66 REM   PROGRAMA PRINCIPAL
 67 REM
 70 GO SUB 1000
 80 PRINT V$;" ";
 90 GO SUB VR+VN*20
100 IF F=0 THEN  GO TO 60
110 PRINT
111 GO TO 50
150 REM INIC.VARIABLES PEC.
900 RESTORE OS
901 READ OC
902 DIM O(OC)
910 FOR I=1 TO OC
911 READ O(I)
912 NEXT I
920 RESTORE VS
921 READ VC
930 LET BE=0: LET LU=0
931 LET GU=0
932 LET E$="EN UN DESIERTO SIN FIN"
933 LET P$="EN UN PASADIZO"
940 RETURN
1000 INPUT "DIME QUE HAGO ''' LINE S$
1001 IF LEN S$=0 THEN  GO TO 1000
1010 LET A$=""
1011 LET N$=""
1020 LET X=0
1021 FOR I=1 TO LEN S$
1030 LET I$=S$(I TO I)
1040 IF I$>="A" AND I$<="Z" THEN  LET I$=CHR$ (32+COD
E I$)
1050 IF I$=" " AND X=1 THEN  LET I=LEN S$: GO TO 1080
1051 IF I$=" " THEN  LET X=1: GO TO 1080
1060 IF X=0 THEN  LET A$=A$+I$
1070 IF X=1 THEN  LET N$=N$+I$
1080 NEXT I
1081 LET W$=A$
1082 LET C=VC
1083 LET D=VS+1
1084 GO SUB 1200
1090 IF VN=0 THEN  PRINT  PAPER 6;" NO COMPRENDO: ";
PAPER 1;"  ";A$: GO TO 1000
1100 LET V$=Y$
1101 RETURN
1200 LET VN=0
1210 RESTORE D
1220 FOR I=1 TO C
1221 READ M$
1222 LET X=LEN M$
1223 IF X>LEN W$ THEN  LET X=LEN W$
1240 IF W$( TO X)=M$( TO X) THEN  LET VN=I: LET Y$=M$
: LET I=C
1260 NEXT I
1261 RETURN
1300 IF N$="" THEN  GO TO 1311
1301 LET W$=N$
1302 LET C=OC
1303 LET D=8001
1304 GO SUB 1200
1305 IF VN=0 THEN  GO TO 1311
1310 PRINT Y$: GO TO 1999
1311 PRINT  PAPER 6;" NO COMPRENDO "; PAPER 1;N$
1312 GO TO 1998
1400 REM VALORES/DIRECCIONES
1500 LET D=1: GO TO 1601
1520 LET D=2: GO TO 1601
1540 LET D=3: GO TO 1601
1560 LET D=4: GO TO 1601
1580 LET D=5: GO TO 1601
1600 LET D=6
1601 PRINT : RESTORE L+LS: READ M$: FOR I=1 TO D: REA
D NL: NEXT I
1602 IF NL=0 THEN  PRINT "NO PUEDES COGER ESE CAMINO"
: GO TO 1998
1603 RESTORE NL+LS: READ M$,S,S,S,S,S,S
1604 READ S
1605 GO SUB 7100+S: IF F=0 THEN  GO TO 1611
1610 IF O(1)=0 THEN  LET L=NL: GO TO 1999
1611 PRINT "ALGO TE FRENA": GO TO 1998
1620 GO SUB 1300: IF F=0 THEN  RETURN
1621 IF O(VN)<>L THEN  PRINT "ESO NO ESTA AQUI": GO T
O 1998
1622 RESTORE OZ+VN: READ S: IF S<0 THEN  GO TO 1997
1624 READ S
1625 READ S
1626 GO SUB CS+S: IF F=0 THEN  GO TO CT
1630 IF CX<>0  THEN  LET CX=CX-1: LET O(VN)=0: RETURN
```

A) De avance:
1) NORTE
2) SUR
3) ESTE
4) OESTE
5) ARRIBA
6) ABAJO

B) De acción:
1) COGE
2) DEJA
3) MATA
4) CORTA
5) NADA
6) ABRE
7) CAVA
8) COME
9) SALTA
10) BEBE
11) AYUDA
12) INVENTARIO
13) PUNTOS

¡!

El programa carece de gráficos definidos, debido a lo cual no aparecerá en el listado ningún carácter subrayado.

*

La grabación del programa se debe llevar a cabo mediante el comando **SAVE**. Si utilizamos el sistema de autoejecución, ésta ha de comenzar a partir de la línea 1. (**SAVE "AVENTURA" LINE 1**).

*

Toda nuestra aventura transcurre en el fantástico «desierto itinerante», por tanto no nos debe extrañar que el camino por el que hemos accedido a determinado lugar, se encuentre poco más tarde bloqueado por una duna.

```
1631 PRINT "LLEVAS DEMASIADAS COSAS": GO TO 1998
1640 GO SUB 1300: IF F=0 THEN  RETURN
1641 IF O(VN)<>0 THEN  PRINT "NO LO TIENES": GO TO 19
98
1642 RESTORE OZ+VN: READ S
1643 LET O(VN)=L
1644 IF L=1 AND S<>0 THEN  LET SC=SC+S: LET O(VN)=-1:
 PRINT "EL ";Y$;" HA DESAPARECIDO"
1645 LET CX=CX+1: IF SC=220 THEN  GO TO 9980
1646 RETURN
1660 GO SUB 1300: IF F=0 THEN  RETURN
1661 LET T$=Y$: LET MN=VN: RESTORE OZ+VN: READ S,MS:
IF S>=0 THEN  GO TO 1997
1662 LET S=0: BEEP .2,25: INPUT "CON QUE VAS A MATAR
?"'W$: IF W$="" THEN  GO TO 1670
1663 LET C=OC: LET D=OS+1: GO SUB 1200: IF VN=0 THEN
 GO TO 1662
1664 IF O(VN)<>0 THEN  GO TO 1662
1665 RESTORE OZ+VN: READ S,S: IF S=0 THEN  GO TO 1662
1666 PRINT "  CON ";Y$;
1670 PRINT : LET S=S+ST: IF MS>S+RND*20 THEN  GO TO 1
679
1671 IF S>MS+RND*15 THEN  GO TO 1678
1672 PRINT "EL ";T$;" VUELVE A LA LUCHA": PRINT "TE S
IENTES DEVIL"
1673 LET ST=ST-RND*5: GO TO 1999
1678 PRINT "TE HAS CARGADO AL ";T$: PRINT "EL CUERPO
SE DESVANECE EN UNA    NUBE DE HUMO": LET O(MN)=-1: GO
 TO 1999
1679 PRINT "EL ";T$;" TE HA MATADO": GO TO 9990
1680 LET L1=L: LET L=0: LET Q$="ESTAS LLEVANDO": GO S
UB 2000: LET L=L1: GO TO 1999
1700 PRINT SC: GO TO 1999
1720 GO SUB 1300: IF F=0 THEN  RETURN
1721 IF VN<>20 OR O(2)<>0 THEN  GO TO 1997
1722 LET O(14)=O(VN): LET O(VN)=-1: PRINT "UNA LLAVE
DE MARFIL                CAE DEL TECHO": RETURN
1740 IF L<>11 AND L<>29 THEN  GO TO 1997
1741 IF L=11 AND O(3)<>0 THEN  PRINT "TE CAISTE AL LA
GO                Y TE HAS AHOGADO": GO TO 9990
1742 IF L=11 THEN  LET L=29: GO TO 1999
1743 LET L=11: GO TO 1999
1760 GO SUB 1300: IF F=0 THEN  LET O(22)=-1: LET O(23
)=L: LET LU=1: RETURN
1762 IF L=13 AND O(14)=0 THEN  LET O(24)=-1: LET O(25
)=L: LET GU=1: RETURN
1763 GO TO 1997
1780 IF O(16)<>0 OR L<3 OR L>10 THEN  GO TO 1997
1781 PRINT "ENCONTRASTE";
1782 IF L=6 AND O(4)=-1 THEN  LET O(4)=L: PRINT " ALG
O": GO TO 1999
1783 PRINT " NADA": GO TO 1999
1800 GO SUB 1300: IF F=0 THEN  RETURN
1801 RESTORE OZ+VN: READ S,S1,S1
1802 IF S<0 THEN  GO TO 1997
1803 GO SUB CS+S1: IF F=0 THEN  GO TO 1997
1804 PRINT "ESTA ALGO DURO PERO...": LET O(VN)=-1: RE
TURN
1820 GO SUB 1300: IF F=0 THEN  RETURN
1821 IF L<>24 OR VN<>7 THEN  GO TO 1830
1822 PRINT "UN PUENTE APARECE SOBRE          EL PRECIP
ICIO": LET O(15)=L: LET BE=1: GO TO TR
1830 PRINT "NO PASA NADA": GO TO 1998
1840 IF L=24 OR L=26 THEN  PRINT "EN LA CAIDA TE HAS
DESNUCADO": GO TO 9990
1841 GO TO 1830
1860 GO SUB 1300: IF F=0 THEN  RETURN
1861 IF VN<>18 THEN  GO TO CT
1862 PRINT "AQUELLO ERA DELICIOSO": LET O(19)=O(VN):
LET O(VN)=-1: RETURN
1997 PRINT "NO PUEDES"
1998 LET F=0: RETURN
1999 LET F=1: RETURN
2000 LET X=0: PRINT Q$
2010 RESTORE OS+1: FOR I=1 TO OC: READ M$
2020 IF L<>O(I) THEN  GO TO 2030
2021 LET X=1: PRINT : PRINT "  UN/A";
2024 PRINT " ";M$
2030 NEXT I
2040 IF X=0 THEN  PRINT "  NADA"
2050 RETURN
3000 REM BASE/DATOS DIRECCIONES
7000 DATA 19
7001 DATA "norte","sur","este","oeste","arriba","abaj
o"
7002 DATA "coge","deja","mata","inventario","puntos"
7003 DATA "corta","nada","abre","cava"
7004 DATA "come","ayuda","salta","bebe"
7100 GO TO 1999
7101 GO TO 1998+LU
7102 GO TO 1998+GU
7103 GO TO 1998+BE
7104 GO TO 1998+(O(3)=0)
7105 GO TO 1998+(O(11)=-1)
7106 GO TO 1998+(O(12)=-1)
7107 GO TO 1998+(O(10)=-1)
7150 IF O(18)<>0 THEN  GO TO 1999
7151 LET O(18)=-1: LET O(12)=-1: PRINT "AL ENTRAR,UN
PIRATA TE ROBA EL RON Y HUYE.": GO TO 1999
7160 PRINT "EL DIABLO TE MATO"
7161 GO TO 9990
7165 REM BASE/DATOS PROG.PRINC.
8000 DATA 25,1,12,27,-1,14,17,21,25,29,12,14,17,20,-1
,-1,11,2,2,-1,19,11,1,-1,13,-1
8001 DATA "lampara de metal"
8002 DATA "espada"
8003 DATA "daga"
8004 DATA "llave"
8005 DATA "alfombra persa"
8006 DATA "moneda de oro"
8007 DATA "vara de plata"
8008 DATA "ruby"
8009 DATA "diamante"
8010 DATA "gigante"
8011 DATA "dragon verde"
8012 DATA "pirata"
8013 DATA "diablo"
8014 DATA "llave de marfil"
8015 DATA "puente de cristal"
8016 DATA "pala"
8017 DATA "mesa"
8018 DATA "botella de ron"
8019 DATA "botella vacia"
8020 DATA "soga que cuelga del techo"
8021 DATA "gran lago"
8022 DATA "puerta cerrada"
8023 DATA "puerta"
8024 DATA "verja cerrada con candado"
8025 DATA "verja"
8501 DATA 0,0,0
8502 DATA 0,20,7
8503 DATA 0,0,0
8504 DATA 0,0,0
8505 DATA 10,0,5
8506 DATA 50,0,6
8507 DATA 20,0,0
8508 DATA 30,0,0
8509 DATA 100,0,0
8510 DATA -1,87,0
8511 DATA -1,110,0
8512 DATA -1,210,0
8513 DATA -1,210,0
8514 DATA 10,0,0
8515 DATA 0,0,0
8516 DATA 0,0,0
8517 DATA 0,0,-5102
8518 DATA 0,0,0
8519 DATA 0,0,0
8520 DATA 0,0,-5102
8521 DATA 0,0,-5102
8522 DATA 0,0,-5102
8523 DATA 0,0,-5102
8524 DATA 0,0,-5102
8525 DATA 0,0,-5102
9001 DATA "EN EL CUARTO DE ESTAR DE UNA CASA,UN LETRE
RO DICE:DEPOSITA TODOS LOS TESOROS AQUI",2,0,0,0,0,12
,0
9002 DATA "EN LA COCINA",Z,1,3,Z,Z,Z,Z,Z
9003 DATA E$,0,4,5,2,0,0,0
9004 DATA E$,3,7,6,0,0,0,0
9005 DATA E$,0,6,7,3,0,0,0
9006 DATA E$,5,8,0,4,0,0,0
9007 DATA E$,0,0,0,4,0,0,0
9008 DATA E$,6,9,0,0,0,0,0
9009 DATA E$,8,0,10,0,0,0,0
9010 DATA E$,11,0,0,9,0,0,0
9011 DATA "UN OASIS",0,10,0,0,0,0,0
9012 DATA "EN UNA CUEVA",0,16,13,0,1,0,1
9013 DATA "A LAS PUERTAS DEL INFIERNO",0,0,0,12,0,20,
0
9014 DATA "EN UNA OSCURISIMA CAVERNA",0,15,0,0,0,0,0
9015 DATA P$,14,18,0,16,0,0,0
9016 DATA P$,12,0,15,0,0,17,0
9017 DATA "EN LA GUARIDA DE LOS PIRATAS",0,0,0,0,16,0
,50
9018 DATA P$,15,19,0,0,0,0,0
9019 DATA P$,18,0,0,0,0,0,0
9020 DATA "EN EL INFIERNO... UN DIABLO DICE: !ENCUENT
RA EL CAMINO CORRECTO Y SOBREVIVE SINO MUERE",30,30,3
0,21,30,30,2
9021 DATA P$,0,24,22,20,0,0,0
9022 DATA "EN UN FINAL MORTIFERO",0,0,0,21,0,0,0
9023 DATA "EN EL INFIERNO",0,0,0,0,0,0,0
9024 DATA "AL BORDE DE UN PROFUNDO ABISMO",21,25,0,0,
0,0,0
9025 DATA "EN UN BONITO CORREDOR",24,0,27,0,0,0,3
9026 DATA "AL BORDE DE UN PROFUNDO POZO. HAY INDICIOS
 DE                      FUEGO Y AZUFRE",0,27,0,0,0,
23,0
9027 DATA P$,26,0,0,25,0,28,0
9028 DATA P$,0,0,0,0,27,18,0
9029 DATA "NADANDO EN UN GRAN LAGO",0,0,0,0,0,0,4
9030 DATA "",0,0,0,0,0,0,60
9035 REM FIN BASE/DATOS PROG.
9036 REM
9037 REM
9038 REM SBR.FIN JUEGO
9980 FOR N=1 TO 4: PRINT : NEXT N
9981 PRINT  PAPER 1; FLASH 1;" LO CONSEGUISTE"
9990 PRINT : PRINT "PUNTUACION ";SC: PRINT "QUIERES I
NTENTARLO DE NUEVO ?"
9991 IF INKEY$="" THEN  GO TO 9991
9992 BEEP .2,20: IF INKEY$="S" THEN  GO TO 5
9993 IF INKEY$="N" THEN  GO TO 10000
9994 GO TO 9991
9995 LET TR=1999: LET FA=1998: LET LS=9000
9996 LET OS=8000: LET OZ=8500: LET VS=7000
9997 LET CS=7100: LET SC=0
9998 LET VR=1480: LET Z=0: LET CT=1997: LET DR=1601:
LET CX=6: LET ST=100
9999 RETURN
```