

12
150pts.

AULT

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek



RUN

Enciclopedia Práctica del Spectrum

una publicación de

EDICIONES NUEVA LENTE, S. A., Y
EDICIONES INGELEK, S. A.

Director editor por NUEVA LENTE
MIGUEL J. GÓÑI

Director editor por INGELEK

ANTONIO M. FERRER

Director de producción

RICARDO ESPAÑOL

Jefe de producción

SANTOS ROBLES

Director de la obra

FERNANDO LOPEZ MARTINEZ

Coordinación

VICENTE ROBLES

Colaboradores

JUAN MANUEL LOPEZ MARTINEZ

CARLOS DE LA OSSA

JUAN MANUEL MAYORAL

Diseño gráfico

JOSE OCHOA

Maquetación

CARLOS GONZALEZ-AMEZUA

Ilustraciones

JOSE OCHOA

ALFONSO MENDEZ

ANTONIO PERERA

Fotografía

(Equipo Gálata)

ALBINO LOPEZ y

EDUARDO AGUDELO

Ediciones Nueva Lente, S. A.

Dirección y Administración:

Benito Castro, 12

28028 Madrid. Tel. 2454598

Ediciones Ingelek, S. A.

Números atrasados y suscripciones:

Avda. Alfonso XIII, 141

28016 Madrid. Tel. 2505820

Publicidad:

LOLA GONZALEZ

CARMINA FERRER

Tel. 4576923

Plan general de la obra:

52 fascículos de aparición semanal

encuadrables en cuatro tomos

de 13 fascículos

Distribución en España:

COEDIS, S. A. Valencia, 245.

08007 Barcelona

Distribución en Argentina:

Capital: Ayerbe

Interior: DGP

Distribución en Colombia:

DISUNIDAS, Ltda.

Distribución en Chile:

Alfa Ltda.

Distribución en Ecuador:

Muñoz Hermanos, S. A.

Distribución en México:

INTERMEX, S. A.

Lucio Blanco, 435

México D. F.

Distribución en Paraguay:

Selecciones, SAC.

Distribución en Perú:

DISELPESA

Distribución en Puerto Rico:

Agencia de Publicaciones de Puerto

Rico, Inc.

Distribución en Uruguay:

Ledian, S. A.

Distribución en Venezuela:

CONTINENTAL

Editor para Chile:

Editorial Andina, S. A.

La Concepción, 311. Santiago-9

Importador exclusivo Cono Sur:

CADE, SRL

Pasaje Sud América 1532. Tel. 212464

Buenos Aires-1.290. Argentina.

© Ediciones Nueva Lente, S. A.

Fotomecánica: OCHOA

Ricardo Ortiz, 74. Madrid.

Impresión: Gráficas Reunidas, S. A.

Avda. de Aragón, 56. Madrid

ISBN de la obra: 84-7534-118-7

ISBN del fascículo: 84-7534-119-5

ISBN del tomo primero: 84-7534-120-9

Déposito legal: M-9896-1985

PRINTED IN SPAIN

Ediciones Nueva Lente, S. A. y Ediciones Ingelek, S. A. garantizan la publicación de todo los fascículos que componen esta obra y el suministro de cualquier número atrasado o tapa mientras dure la publicación y hasta un año después de terminada. El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra si las circunstancias del mercado así lo exigen.

JUNIO 1985

Los programas de RUN ya están a la venta en casete

AHORA, LAS UNICAS TECLAS QUE TIENES QUE TOCAR SON AQUELLAS QUE TE PERMITIRAN HUNDIR UN PORTAAVIONES ENEMIGO Y GANAR LA GUERRA, RESOLVER EL ENIGMA MARSUPIAL O VIAJAR ENTRE LOS ANILLOS DE SATURNO EN UNA MISION FANTASTICA Y PELIGROSA.

HEMOS GRABADO LOS PROGRAMAS DE RUN EN CINTAS DE GRAN CALIDAD. NADA MENOS QUE 4 PROGRAMAS POR CINTA (CON UN DUPLICADO DE SEGURIDAD). OLVIDATE DE TECLEAR LOS LISTADOS Y DE LOS ERRORES. COLOCA LA CINTA Y JUEGA. ¡ASI DE FACIL!



CUATRO
SENSACIONALES
PROGRAMAS POR
SOLO 300 PTAS.

Cada 4 fascículos ponemos a la venta un casete con los programas que en ellos se publican. Estas cintas solo se venden por suscripción o pedido (rellenando el cupón al efecto).

AHORA, UNA HORA FRENTE A TU SPECTRUM SON 60 MINUTOS DE EMOCION.

Recorta o copia este cupón y envíalo a
EDICIONES INGELEK. Apdo. de Correos 61294. 28080 MADRID

- ☐ Deseo recibir las 13 cintas de aparición mensual con los programas publicados en RUN, al precio de 3.900 pesetas.
☐ Deseo recibir las cintas siguientes con los programas de RUN.

- ☐ CINTA N.º 1. Programas de los fascículos 1 al 4
☐ CINTA N.º 2. Programas de los fascículos 5 al 8
☐ CINTA N.º 3. Programas de los fascículos 9 al 12
☐ CINTA N.º 4. Programas de los fascículos 13 al 16
☐ CINTA N.º 5. Programas de los fascículos 17 al 20
☐ CINTA N.º 6. Programas de los fascículos 21 al 24
☐ CINTA N.º 7. Programas de los fascículos 25 al 28

- ☐ CINTA N.º 8. Programas de los fascículos 29 al 32
☐ CINTA N.º 9. Programas de los fascículos 33 al 36
☐ CINTA N.º 10. Programas de los fascículos 37 al 40
☐ CINTA N.º 11. Programas de los fascículos 41 al 44
☐ CINTA N.º 12. Programas de los fascículos 45 al 48
☐ CINTA N.º 13. Programas de los fascículos 49 al 52

La fecha de salida de cada cinta es la de publicación del último de los 4 fascículos que comprende.

NOMBRE																EDAD	
APELLIDOS																	
DOMICILIO																	
CIUDAD																PROVINCIA	
C. POSTAL																TELEFONO	
															PROFESION		

Marco con una X en el casillero correspondiente la forma de pago que más me conviene.

- ☐ Talón bancario adjunto a nombre de INGELEK S. A.
☐ Giro postal n.º _____
☐ Contra reembolso del importe más gastos de envío.

Firma,



¡HUNDIDO!



CANGURIN



FUNCIONES 3D



SATURNO JET



TRATAMIENTO NUMERICO



ASTA el momento hemos hablado de algunas de las funciones de que dispone el vocabulario BASIC, concretamente, de las aplicables al tratamiento de expresiones y variables del tipo cadena (*string*); tales son: **LEN**, **STR\$**, **VAL** y **VAL\$**. Sin embargo, existe otro tipo de funciones de notable importancia, aplicables de manera general a expresiones numéricas, a las que denominamos **FUNCIONES NUMERICAS**, cuyo funcionamiento y misión trataremos en este capítulo. El formato general de estas funciones es similar al de las ya descritas de cadena; así pues, constan de nombre de **FUNCION** y **ARGUMENTO** (constante, variable o expresión) que será evaluado para obtener el resultado.

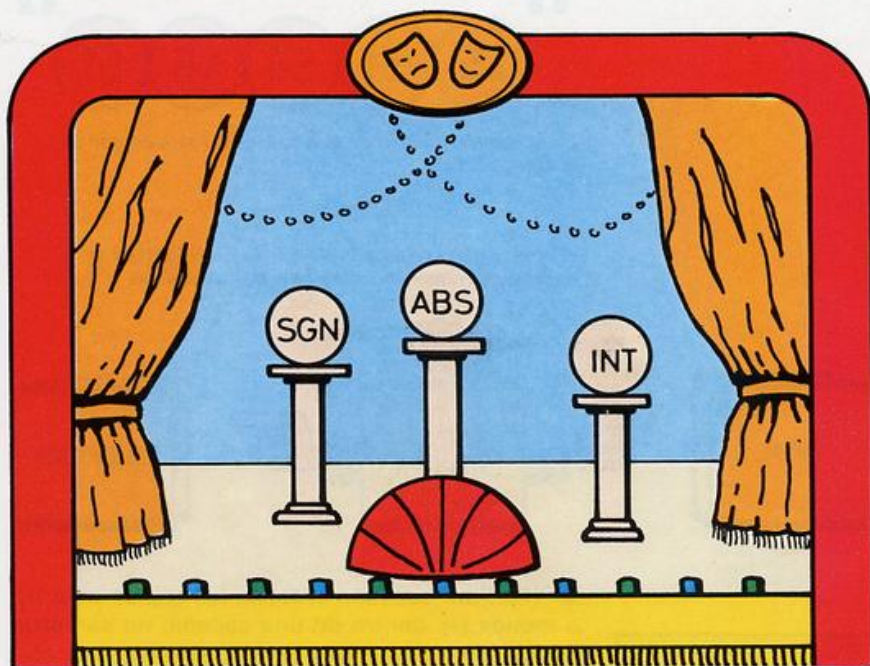
Podemos decir por tanto, que la sintaxis general de cualquier función BASIC, sea del tipo que sea, es: **FUNCION + ARGUMENTO**; donde el argumento puede ser una constante, una variable o una expresión, siempre y cuando se ajuste al tipo de dato sobre el que opera la función: de cadena o numérico.

Entre las funciones numéricas, de las que nos ocuparemos en las próximas páginas dedicadas al lenguaje BASIC, se cuentan: **SGN**, **ABS**, **INT**, **SQR**, **LN**, **EXP** y las denominadas funciones trigonométricas (**SIN**, **COS**, **TAN**, **ASN**, **ACS** y **ATN**). Además de todas estas funciones, en su mayor parte matemáticas, hablaremos de algunos otros elementos BASIC relacionados con ellas. Pero lo mejor que podemos hacer es comenzar sin más preámbulos.

SGN

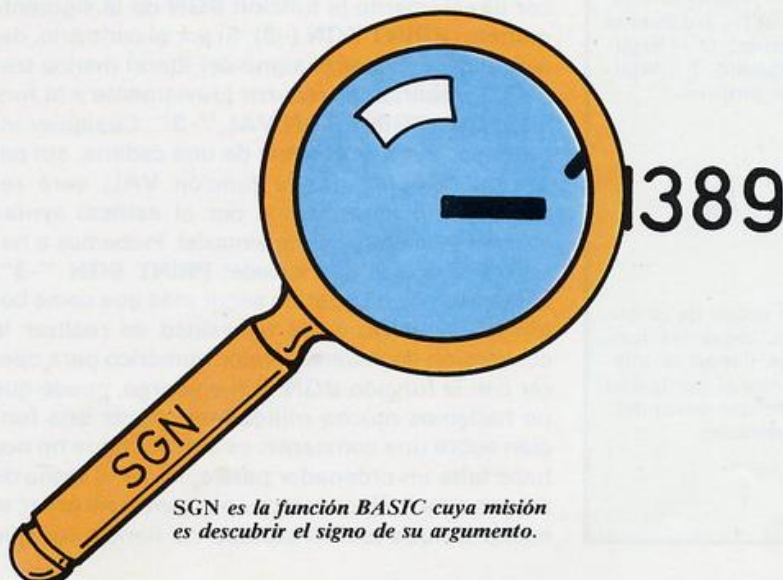
Por medio de esta función puede evaluarse el signo de un argumento numérico. Evidentemente, **SGN** no es aplicable a variables de cadena, y sí en cambio a constantes, variables o expresiones numéricas.

Tengamos en cuenta, que aunque dentro de una variable de cadena algunos de sus caracteres, o más concretamente el primero de ellos, puede



Las funciones BASIC juegan un papel muy importante en el teatro de la programación.

ser un signo más (+) o menos (-), a efectos del ordenador, por encontrarse entrecomillados, éstos carecen de su función habitual de signo matemático, y no dejan de ser meros caracteres, como lo son una letra A o cualquier símbolo o signo de puntuación, tales como la admiración (!) o el dó-



SGN es la función BASIC cuya misión es descubrir el signo de su argumento.

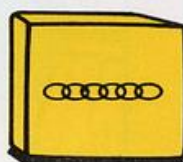
" -38.3 "



" 45 51 56 46 51 "

Cada número dentro de una cadena no es más que su código A.S.C.I.I.

el syntax checker no admite que la función SGN se aplique directamente sobre un valor de cadena.



i!

La función **SGN** devuelve el signo de su argumento, codificando la información según el siguiente convenio: -1 = argumento negativo, 0 = argumento cero, 1 = argumento positivo.

*

En el orden de prioridades, todas las funciones tienen la misma entre sí, por lo que se efectúan de izquierda a derecha.

lar (\$). Para ser más exactos, los signos más (+) o menos (-), dentro de una cadena, no son otra cosa que los caracteres cuyo código A.S.C.I.I. corresponde a 43 y 45, respectivamente.

En este momento, conviene recordar la existencia de la función **VAL**, que ya vimos anteriormente, cuya misión es convertir una variable de cadena en numérica. Gracias a ella, podemos someter la cadena a un proceso previo de conversión, tras el cual poderle aplicar, ya en forma de valor numérico, la función **SGN**.

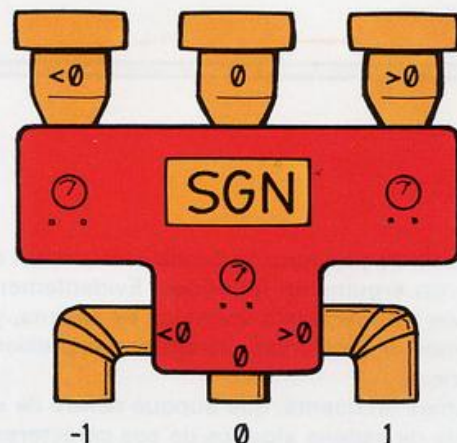
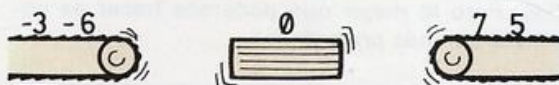
Supongamos que queremos saber cual es el signo de la constante menos tres (-3). En este caso, por tratarse de un valor numérico, podemos aplicar directamente la función **SGN** de la siguiente manera: **PRINT SGN (-3)**. Si por el contrario, deseáramos conocer el signo del literal menos tres ("-3"), deberíamos recurrir previamente a la función **VAL**: **PRINT SGN VAL "-3"**. Cualquier intento por evaluar el signo de una cadena, sin pasar previamente por la función **VAL**, será rechazado sin miramientos por el estricto *syntax checker* (comprobador de sintaxis). Probemos a hacerlo y observar que sucede: **PRINT SGN "-3"**. El ejemplo no ha querido servir más que como botón de muestra, de la necesidad de realizar la conversión de cadena a valor numérico para operar con la función **SGN**. Sin embargo, puede que no hallemos mucha utilidad en aplicar una función sobre una constante: es evidente que no nos hace falta un ordenador para conocer el signo de menos tres (-3); conviene, por tanto, recordar el hecho de que las constantes no tienen por qué

ser tan simples como la que se ha empleado en el ejemplo. Para reafirmar esta teoría, transcribamos una constante cuya complejidad habla por sí sola: $((-3*5 + \text{SGN}(15*3/2) - (28 * 5 / 3)) * 2 - 3) - 66$. Esta vez no estaría de más el concurso del ordenador para encontrar su signo ¿verdad?

En estos casos en los que no entran en juego variables que un programa puede alterar, es conveniente que en vez de introducir en el programa el cálculo de la función, lo llevemos a cabo nosotros mismos, mediante el modo directo, y lo insertemos ya calculado en el programa. Gracias a ello, ahorraremos memoria y tiempo de ejecución, puesto que la expresión completa siempre ocuparía más espacio que su resultado, y además tendría que ser recalculada cada vez que el ordenador pasara por ese punto del programa; operación bastante inútil si tenemos en cuenta que al tratarse de una expresión constante, su valor nunca variará.

Como hemos dicho al comienzo de este epígrafe, la función **SGN** indica el signo de su argumento, pero ¿de qué manera lo señala? Quizá podríamos pensar que simplemente mediante un carácter menos ("-") o más ("+"). Sin embargo, esta solución no es válida: en primer lugar hemos de notar que el nombre de la función es **SGN** (abreviatura de la palabra inglesa *SiGN*, signo), por tanto debe devolver un valor numérico, puesto que de no ser así, como ya hemos visto al tratar las funciones de cadena, su denominación debería terminar en un símbolo dólar (\$), siendo por tanto **SGN\$**.

Los valores devueltos por la función **SGN** son: -1 para argumentos negativos, 1 para argumentos positivos y 0 para argumento cero.

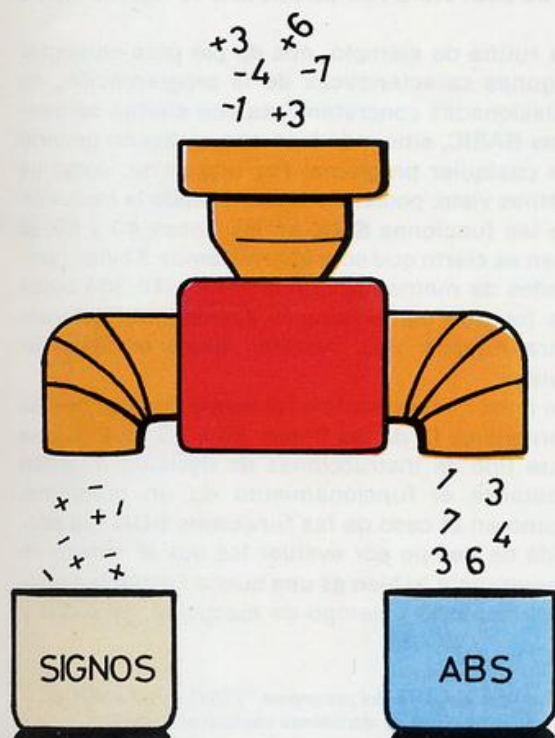


Por otra parte, el ordenador siempre proporciona una mayor facilidad de manejo con datos numéricos, por lo que suelen ser más fácilmente utilizables los datos de este tipo que los de cadena. Así pues, ¿cuál es el convenio que se adopta para indicar un signo mediante números?: como resultado de la función obtenemos tres valores posibles; 1 si el argumento es positivo, 0 si es cero y -1 si el argumento es negativo. Veamos algunos ejemplos:

```
10 REM - SGN
20 LET A=-2:LET B=0:LET C=3
30 PRINT SGN A,SGN B,SGN C,SGN
  A/B,SGN A/C,SGN C/A
```

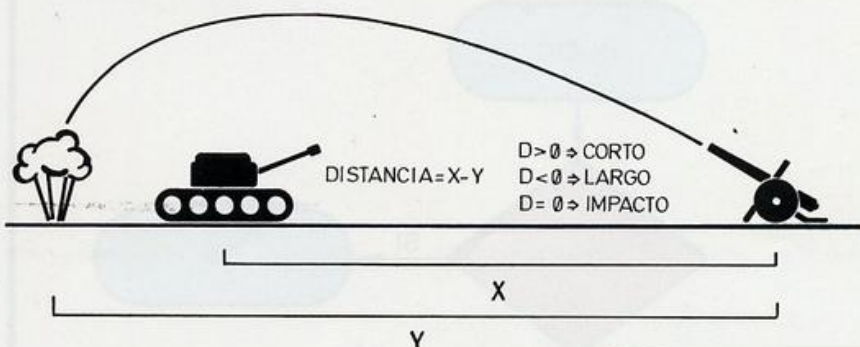
La utilidad de esta función puede que todavía no se nos manifieste claramente, pero según vayamos avanzando en la programación, iremos tomando conciencia del servicio que presta. No olvidemos que algunas operaciones matemáticas, incluso de uso muy común, como la radicación, no admiten valores negativos. Por ejemplo, si tenemos una instrucción del tipo **10 PRINT SQR A**, corremos el riesgo de que la variable A adopte un valor negativo; de ser así, al ejecutarse la línea 10 el programa se detendría con un error del tipo **A Invalid argument**. Una forma de evitar esto, sería utilizar la función **SGN**, para no dejar llegar a la operación nada más que valores positivos o cero: **10 IF SGN A>=0 THEN PRINT SQR A**.

La función **ABS** priva a los números de su signo.



La radicación, al igual que algunas otras operaciones matemáticas, no admite un operando negativo.

Planteamiento de la rutina "TIRO AL TANQUE". La variable D contiene la distancia que separa el tanque de la posición de impacto del proyectil.



En el caso concreto del ejemplo, hubiera sido más sencillo emplear directamente la comparación sobre A (**10 IF A>=0 THEN PRINT SQR A**), aunque sin duda, mediante **SGN** el programa gana en claridad, en cuanto a las intenciones del programador sobre esa línea del programa. En algunas otras ocasiones no es sólo conveniente o clarificador, sino prácticamente inevitable recurrir a **SGN**.

ABS

Esta función nos permite obtener el valor absoluto de un número. Matemáticamente, entendemos por obtener el valor absoluto privar de su signo al número afectado; así pues, el valor absoluto de tres ($|3|$); el valor absoluto de un número se representa convencionalmente entre barras verticales, es igual al de menos tres ($|-3|$). Para ser más concretos, la obtención del valor absoluto,

i!

La función **ABS** calcula el valor absoluto de su argumento; para ello, convierte en positivo su argumento, cualquiera que sea la magnitud o signo original de este.

Un sistema sencillo para averiguar si determinado número es o no entero, consiste en comparar su forma original y su forma entera, ya sea mediante el operador "igual que" ($=$), "distinto que" (\neq) o simplemente restando sus valores numéricos ($-$).

más que desproveer del signo al argumento, lo que hace es forzarlo a ser positivo, cualquiera que fuera su signo anterior.

De lo dicho anteriormente, se desprende que al aplicar la función **SGN** sobre un valor absoluto, nunca obtendremos el valor -1. La utilidad de esta función es también muy grande; pongamos un ejemplo de ello:

Vamos a suponer que en el diseño de un programa de juego, en el que se intenta alcanzar mediante un proyectil un tanque en movimiento, deseamos indicar al usuario lo alejado que quedó el último disparo de su objetivo, con el fin de corregir la trayectoria en la próxima andanada. Llegando a este punto veremos que se pueden producir dos circunstancias: o bien el disparo quedó

largo o bien corto (desecharemos las posibilidades de izquierda y derecha para no complicar el ejemplo).

En la variable D, tendremos el valor de la posición del tanque menos la de impacto del proyectil. Si suponemos que el origen de coordenadas se encuentra en nuestra base, cuando la variable D resulte positiva, el disparo habrá sido corto; de ser negativo, largo, y siendo cero, un éxito completo. Pasemos ahora a la transcripción efectiva del planteamiento a un programa BASIC:

```

10 REM D CONTIENE LA DISTANCIA AL
   BLANCO
20 IF D=0 THEN PRINT "*** FELICIDADES
   *****IMPACTO DIRECTO**": GO TO 70
30 PRINT "EL DISPARO HA QUEDADO ";
40 IF SGN D>0 THEN PRINT "CORTO"
50 IF SGN D<0 THEN PRINT "LARGO"
60 PRINT "POR ";ABS D;" METROS."
70 REM CONTINUACION DEL PRO-
   GRAMA
  
```

Con esta sencilla rutina, se imprimirán en la pantalla los mensajes indicando el resultado del disparo. Indudablemente, habremos mejorado la presentación de datos del programa, imprimiendo, por ejemplo:

**EL DISPARO HA QUEDADO LARGO
POR 15 METROS**

que no habiendo presentado D en su valor con signo (sin utilizar **ABS**), como sería el caso de:

EL DISPARO HA QUEDADO A -15 METROS

La rutina de ejemplo, nos da pie para comentar algunas características de la programación, no relacionadas concretamente con ciertas sentencias BASIC, sino más bien con el diseño general de cualquier programa. Por una parte, como ya hemos visto, podíamos haber evitado la inclusión de las funciones **SGN** en las líneas 40 y 50, si bien es cierto que sólo ahorraríamos 2 bytes (unidades de memoria), que entre los 16.384 bytes de los que como mínimo dispone el Spectrum para nuestro uso, resultan bastante insignificantes.

De mayor importancia es la forma de disponer las sentencias **IF** de las líneas 40 y 50, puesto que este tipo de instrucciones de decisión retardan bastante el funcionamiento de un programa. Como en el caso de las funciones **SGN**, la pérdida de tiempo por evaluar los dos **IF** carece de importancia, si bien es una buena costumbre ahorrar memoria y tiempo de ejecución, en todos y

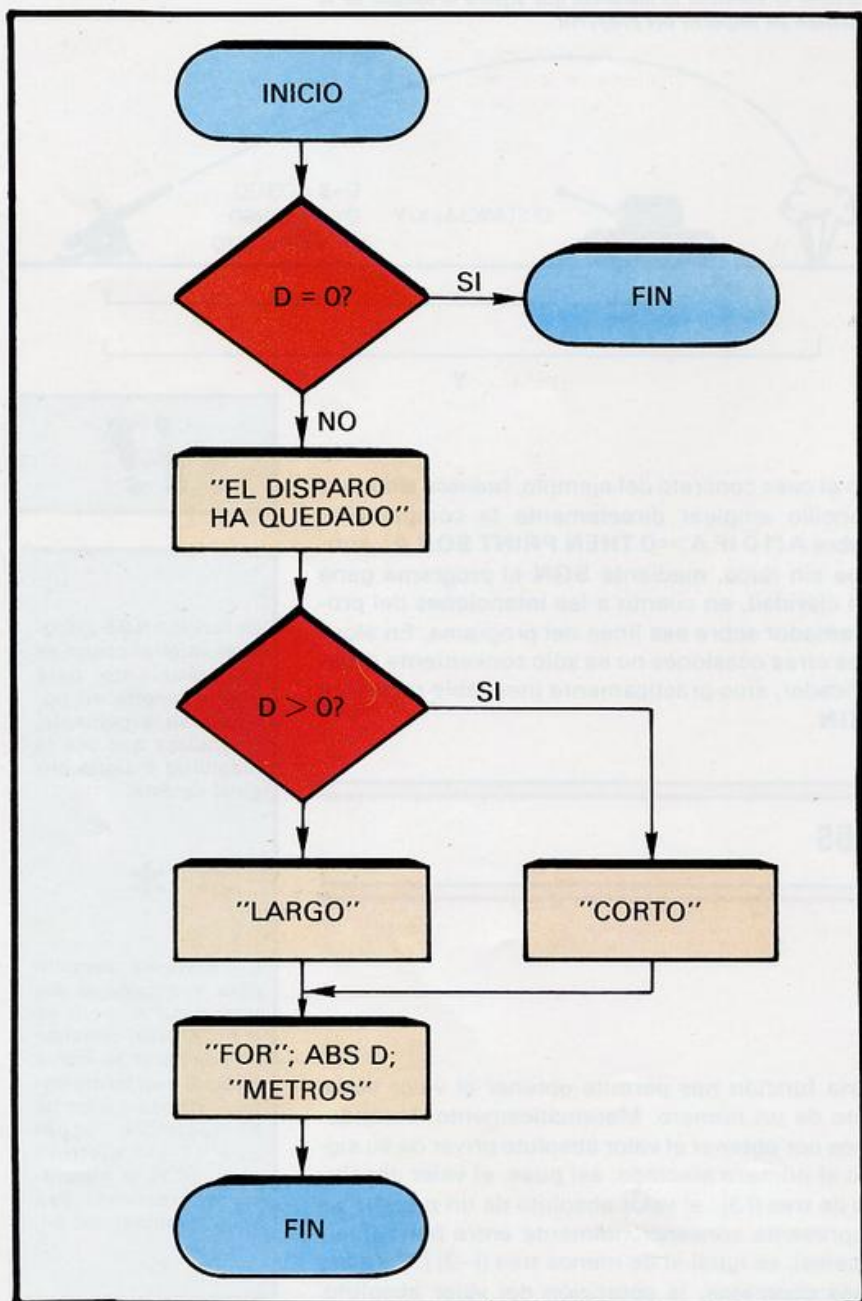


Diagrama de flujo del programa "TIRO AL TANQUE", con la estructura de decisiones exclusivas evitada mediante un puente.



cada uno de los elementos de la programación. En la totalidad de un programa de relativa longitud, la diferencia de ese conjunto de pequeños ahorros, sí se hace sentir.

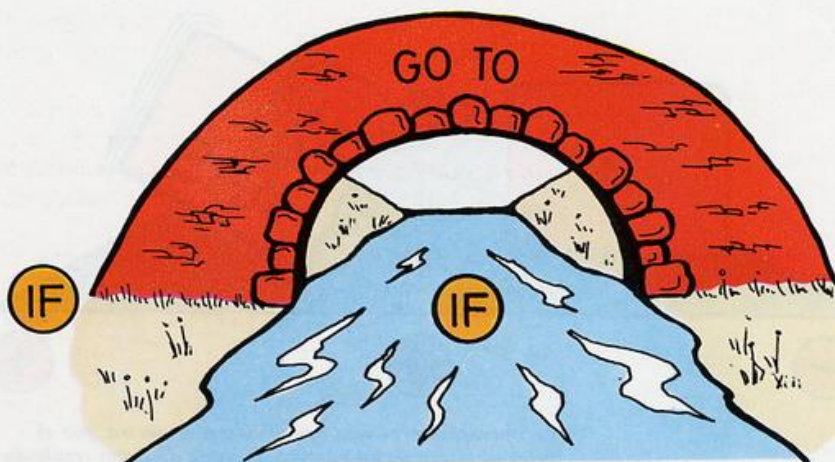
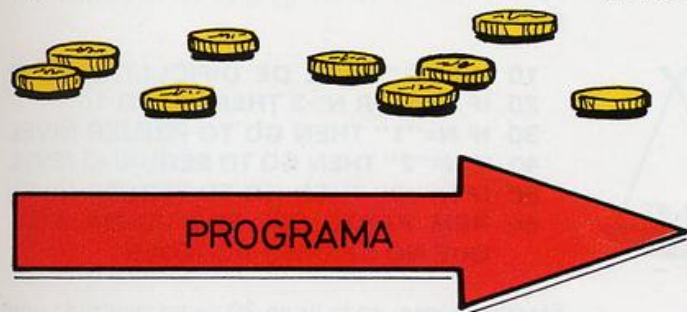
En el caso concreto de la rutina que nos ocupa, encontramos en dos líneas de programa seguidas (40 y 50), lo que en programación BASIC se conoce generalmente como **IF** exclusivos o **DECISIONES EXCLUSIVAS**. Este término implica que dadas dos decisiones consecutivas, cuando una de ellas se cumple, la otra no puede cumplirse bajo ninguna circunstancia.

En nuestro caso, dado que el posible valor cero de la variable D ha sido tratado ya en la línea 20, al llegar a las instrucciones 40 y 50 del programa, nos encontramos con dos decisiones exclusivas: $D > 0$ y $D < 0$. Evidentemente, si se cumple que D es mayor que cero, jamás se puede cumplir en la línea siguiente (puesto que no ha habido alteración del valor de D), que D sea menor que cero, y viceversa.

Para la eliminación de este tipo de estructuras, nos apoyamos en la sentencia **GO TO**, construyendo lo que denominamos un "puente". En el caso concreto de nuestro ejemplo, "puentearemos" la línea 50 de programa, mediante un **GO TO** a la 60 al finalizar la primera comparación, y eliminaremos el **IF** ya innecesario en la línea siguiente.

De esta forma, sólo se realizará una comparación (la de la línea 40), ya que si ésta se cumple, se imprimirá el mensaje correspondiente y pasaremos directamente a la línea 60; por el contrario, si el **IF** de la línea 40 no se hubiera cumplido, se pasaría directamente a la instrucción 50, donde se tiene libre acceso a la impresión del mensaje sin la evaluación de ninguna condición. Veamos el programa bajo su nueva configuración:

```
10 REM D CONTIENE LA DISTANCIA AL
    BLANCO
20 IF D=0 THEN PRINT "*** FELICIDADES
    *****IMPACTO DIRECTO": GO TO 70
30 PRINT "EL DISPARO HA QUEDADO ";
40 IF D>0 THEN PRINT "CORTO": GO TO
60
50 PRINT "LARGO"
60 PRINT "POR ";ABS D;" METROS."
70 REM CONTINUACION DEL PROGRAMA
```



*Mediante una técnica de "puente" basada en la sentencia **GO TO**, podemos ahorrar espacio de memoria y tiempo de ejecución, evitando la evaluación de un **IF**.*

Gracias a esta sencilla técnica, hemos conseguido reducir la evaluación innecesaria de dos condiciones a una sola. Ello nos ha obligado a incluir una nueva instrucción: **GO TO 60**, en la línea 40. Sin embargo, ha merecido la pena la modificación, puesto que hemos ahorrado en espacio de memoria, y lo que es más importante, en tiempo de ejecución.

Es de destacar la considerable diferencia en cuanto al tiempo que el ordenador emplea en tratar una sentencia **IF**, respecto a una **GO TO**. Nos bastará saber que el espacio que en la "inteligencia" del Spectrum (su memoria ROM) se utiliza para el tratamiento de los **IF**, es decenas de veces superior al empleado para la ejecución de un **GO TO**.

Depende de nosotros que, según las condiciones particulares de cada programa, decidamos ahorrar memoria (suprimiendo por ejemplo sentencias **SGN** que puedan ser obviadas), o aumentar

Los pequeños ahorros de memoria y tiempo de ejecución en un programa, conducen al final del mismo a un ahorro total considerable.

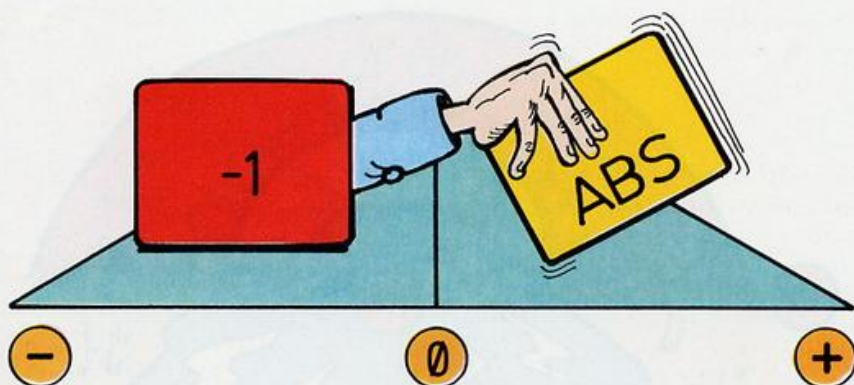


i!

Combinando la función **ABS** con la multiplicación por el elemento neutro negativo, podemos cerciorarnos de que un determinado valor no es positivo. Para ello, hallaremos el valor absoluto del número en cuestión, y multiplicaremos este por -1.

*

El sistema de "puenteado", para evitar estructuras de decisiones exclusivas, debe ser utilizado siempre que aparezca un problema de este tipo.



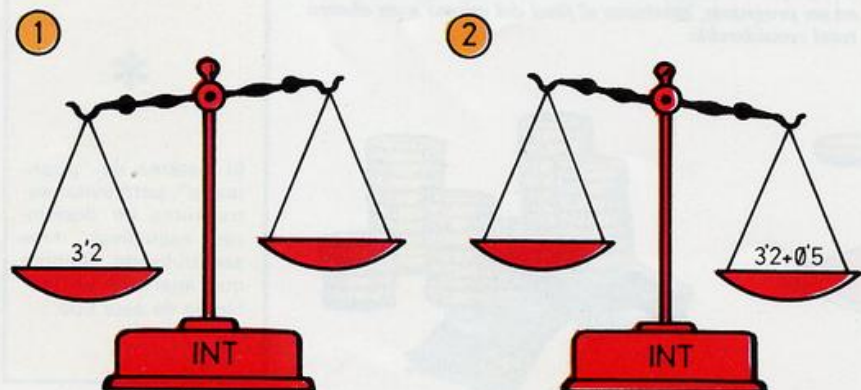
La operación producto con el elemento neutro, por el valor absoluto de un número, siempre da como resultado el número original con signo negativo.

la claridad del mismo. No obstante, la técnica del "puente" en los **IF** exclusivos es recomendable en todo momento.

Otra utilidad de la función **ABS** puede ser la de asegurarse la obtención de un número negativo; esto se puede conseguir forzando previamente el signo del número a positivo (mediante **ABS**), y multiplicando posteriormente este resultado por **-1**: **PRINT -1*ABS N**. Al multiplicar por **-1**, no alteramos la magnitud (valor numérico sin signo) del número, puesto que 1 es el elemento neutro de la operación producto; además de lo cual, cambiamos el signo de la expresión al multiplicar por un número negativo. Si esta operación hubiera sido realizada sin efectuar anteriormente el valor absoluto, correríamos el riesgo de encontrar un número originalmente negativo y cambiarlo a signo positivo.

Finalizaremos estas líneas de estudio de la función **ABS** (del inglés *ABSolute*, absoluto), haciendo notar que su uso sólo afecta al signo del argumento, y en ningún momento a la magnitud del mismo. Probemos con algunos ejemplos:

El "peso" de 0.5 hace que la función **INT** se decante por un redondeo comercial.



```
10 REM - ABS
20 LET A=-2:LET B=0:LET C=3
30 PRINT ABS A,ABS B,ABS C,ABS
  A/B,ABS A/C,ABS C/A
```

INT

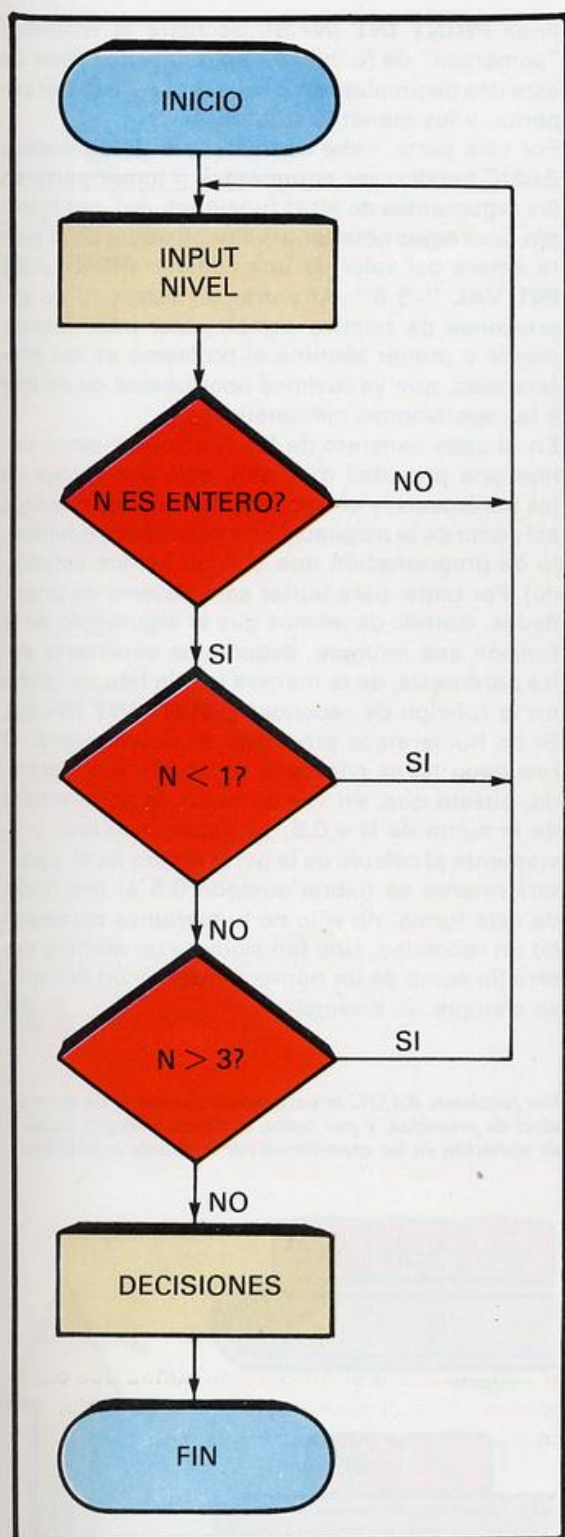
Entre las funciones del BASIC del Spectrum dedicadas al tratamiento numérico, que no constituyen una operación matemática, probablemente sea **INT** la de uso más frecuente. La misión de esta función es encontrar el valor entero de su argumento, ya sea este entero o decimal, positivo o negativo, despreciando sus decimales.

Aunque actualmente los microordenadores consideran como habitual el uso de valores numéricos con coma flotante (números decimales), gran cantidad de sucesos representables mediante números y que se pueden tratar en una computadora, son valores enteros. Por ejemplo, podríamos citar los números de teléfono, los valores que se pueden obtener al lanzar un dado, etc...

Debido a esto, es muy interesante disponer de una forma de cerciorarnos que el número sometido a tratamiento es entero y no decimal. Como el resto de las funciones, **INT** no modifica su argumento, sino que simplemente obtiene su parte entera. Del mismo modo, tampoco es de importancia el hecho de que el argumento de la función sea ya entero, puesto que aunque aparentemente no se produzca efecto alguno, la tarea de conversión a entero será llevada a cabo, y no se producirá ningún tipo de detención por error. Generalmente, **INT** se utiliza durante las tomas de datos, para adaptar al formato entero determinadas entradas. Supongamos, por ejemplo, que durante la realización de un juego, interrogamos al usuario sobre el nivel de dificultad que desea escoger. Una depuración evidente es la del rango máximo que puede tener este dato, por ejemplo, entre 1 y 3. Observemos el siguiente programa:

```
10 INPUT "NIVEL DE DIFICULTAD ? ";N
20 IF N<1 OR N>3 THEN GO TO 10
30 IF N="1" THEN GO TO PRIMER NIVEL
40 IF N="2" THEN GO TO SEGUNDO NIVEL
50 IF N="3" THEN GO TO TERCER NIVEL
60 REM PARTE DEL PROGRAMA A LA
  QUE NO SE DEBE ACCEDER
```

Efectivamente, en la línea 20 se ha realizado una depuración para que no entren valores superior-



res a tres ni inferiores a uno, ya que al no cumplir ninguna de las condiciones del bloque 30-50, pasarían a la zona reservada de programa, a partir de la 60.

Sin embargo, qué sucederá si el usuario introduce un valor decimal, ya sea por descuido o por poner a prueba nuestra habilidad como programadores. Si este valor supera a tres, o es inferior a uno, será capturado por nuestra depuración de la

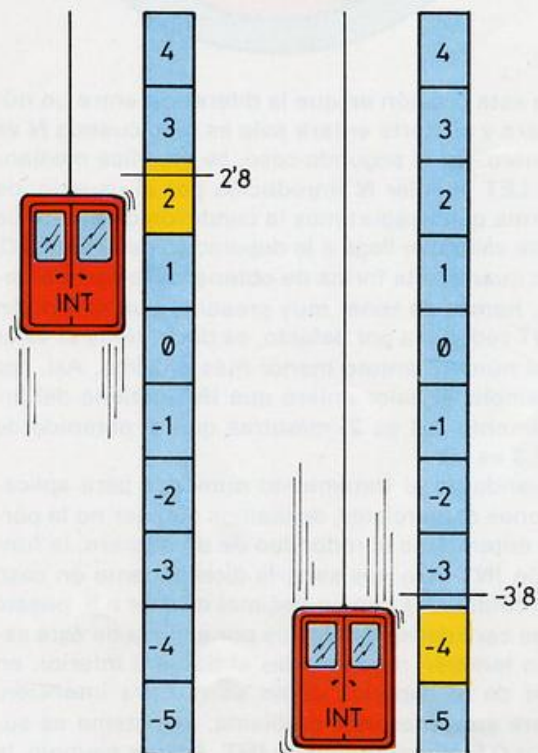
Diagrama de flujo de la rutina de depuración de entrada, una vez que le ha sido aplicado el control de números decimales.

línea 20, pero ¿y si no es así?; supongamos que el número introducido es 2.3. El valor erróneo pasará la instrucción 20, al estar comprendido entre uno y tres, y además, no se bifurcará en ninguna de las condiciones del bloque 30-50, puesto que no es ni uno, ni dos, ni tres. Evidentemente, el programa ejecutará la instrucción 60, con el consiguiente error.

Gracias a la función **INT**, nos aseguraremos de que el valor que llega a la línea 20 sea entero, de forma que no evite la depuración. Para ello, podemos seguir dos sistemas: o bien realizar una depuración en la línea 15 de valores enteros, o bien modificar el valor de N, para convertirlo incondicionalmente en entero. Las líneas podrían quedar como sigue: **15 IF N < INT N THEN GO TO 10** o **15 LET N=INT N**.

En el primer caso, como depuración de si el valor es entero o no, se emplea la condición **SI N ES DISTINTO DEL VALOR ENTERO DE N**, puesto que sólo N será igual a su valor entero, cuando efectivamente sea entero. Otro sistema habría sido codificar la instrucción como: **15 IF N-INT N THEN GO TO 10** (equivalente a **15 IF N-INT N < 0 THEN GO TO 10**), el algoritmo se basa

*La función **INT** efectúa un redondeo por defecto, es decir, obtiene el entero inferior más próximo a su argumento.*



i!

La función **INT** toma la parte entera de su argumento, redondeando siempre por defecto, es decir, al entero inferior más próximo.

*

Las funciones **BASIC** no alteran sus argumentos, sino que simplemente obtienen determinado valor en base a ellos.

*

En base a la función **INT** podemos realizar operaciones de redondeo comercial, es decir, al entero superior aquellos valores que superen o igualen a un medio sobre su parte entera, y al inferior los restantes; para realizar esta operación, sumaremos 0.5 al valor a redondear, y posteriormente hallaremos la parte entera del resultado obtenido.



La función INT corta la parte decimal de su argumento, extrayendo su parte entera.

Todas las funciones BASIC, pueden utilizar como argumento otras funciones, siempre y cuando sean del tipo adecuado (numéricas o de cadena).

i!

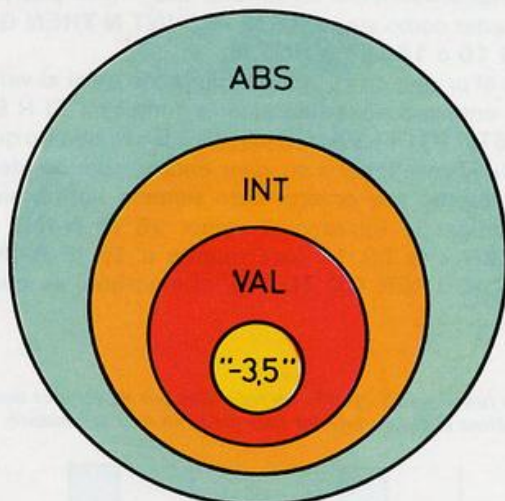
Las funciones BASIC pueden servir a su vez como argumentos de otras funciones, bien sea individualmente, o como parte de expresiones más complejas.



El ahorro de memoria y tiempo de ejecución, por pequeño que parezca, conduce, al cabo de un programa de relativa longitud, a un efecto de ahorro sensible.



En base a la función INT, podemos construir una estructura que diferencia los números pares de los impares: si la división del número entre dos da un resultado entero, el número será par; de no ser así, será impar (IF NUMERO/2=INT(NUMERO/2) THEN PRINT "PAR").



en esta ocasión en que la diferencia entre un número y su parte entera sólo es cero cuando N es entero. En el segundo caso, se modifica mediante LET el valor N introducido por el usuario, de forma que aseguramos la condición de entero de este valor que llega a la depuración de la línea 20. En cuanto a la forma de obtenerse la parte entera, hemos de tener muy presente que la función INT redondea por defecto, es decir, toma el valor del número entero menor más próximo. Así, por ejemplo, el valor entero que INT obtiene del argumento 2.3 es 2, mientras que el obtenido de -2.3 es -3.

Cuando en el tratamiento numérico para aplicaciones comerciales, deseamos obtener no la parte entera sino el redondeo de un número, la función INT sólo nos serviría directamente en caso de cantidades con un decimal inferior a 5, puesto que cantidades decimales por encima de éste serán también redondeadas al número inferior, en vez de al superior, como es nuestra intención. Para subsanar este problema, el sistema es sumar 0.5 al argumento de INT. Así por ejemplo, la

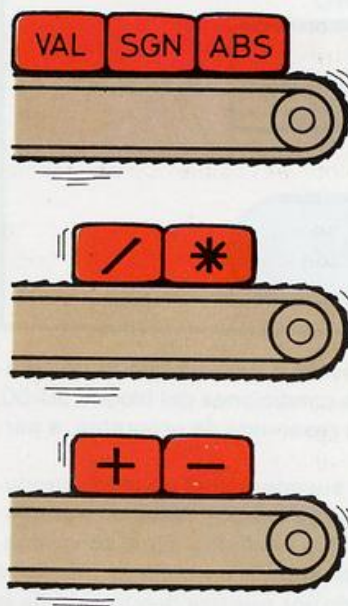
línea PRINT INT (N+.5), escribirá el redondeo "comercial" de N, cualquiera que sea el valor de éste (los decimales mayores o iguales a 0.5 al superior, y los menores al inferior).

Por otra parte, cabe destacar que las funciones BASIC pueden ser argumentos o tomar parte en los argumentos de otras funciones. Así, por ejemplo, podremos obtener el valor absoluto de la parte entera del valor de una cadena: PRINT ABS INT VAL "-3.5". Al entrar en este tipo de expresiones de relativa complejidad, pasa nuevamente a primer término el problema de las preferencias, que ya tuvimos oportunidad de aplicar a las operaciones matemáticas.

En el caso concreto de las funciones, estas tienen una prioridad muy alta, sólo por debajo de los paréntesis, y el troceado de cadenas (slicing), así como de la asignación de subíndices (elemento de programación que aún no hemos estudiado). Por tanto, para burlar este sistema de prioridades, cuando deseemos que el argumento de la función sea múltiple, deberemos encerrarlo entre paréntesis, de la manera que lo hemos hecho en la función de redondeo (PRINT INT (N+.5)). Si no hubiéramos procedido de esta manera, el resultado no se parecería en absoluto al obtenido, puesto que, en vez de hallar la parte entera de la suma de N y 0.5, se habría realizado previamente el cálculo de la parte entera de N, y posteriormente se habría sumado 0.5 al resultado; de esta forma, no sólo no hubiéramos conseguido un redondeo, sino tan siquiera un número entero (la suma de un número entero y un decimal, es siempre un decimal).



Las funciones BASIC se encuentran dentro de un mismo nivel de prioridad, y por tanto, son evaluadas por orden de aparición en las expresiones (de izquierda a derecha).



EL MAPA DEL SPECTRUM



N el capítulo anterior hicimos un buen repaso de los principales componentes de nuestro Spectrum, aclarando diversos aspectos sobre sus características, cometido, etc... Así pues, ha llegado la hora de poner una nota gráfica sobre el tema, finalizando esta etapa de nuestro conocimiento del ordenador con un completo mapa de la situación general de los principales componentes.

Nos consta que entre los aficionados a la microinformática, se encuentra un gran número de valiosos «manitas», que ya han hecho algún que otro pinito en el mundillo del *hardware*; especialmente dedicadas a ellos van las próximas páginas, en las que hacemos un rápido recorrido por el interior de nuestro ordenador.

Por tanto, aquellos que no seamos expertos en el manejo de destornilladores, soldadores, martillos... (no, no... martillos, no), más vale que nos abstengamos de poner en práctica las sugerencias de las páginas siguientes (convenientemente señalizadas con un PROHIBIDO INEXPERTOS y FIN DE PROHIBICION), y pasemos directamente al mapa de la doble página que finaliza el artículo, la cual conseguirá que los «excursionistas» del *hardware* no «perdamos el norte», si en alguna ocasión vemos un Spectrum tal cual, ¡como Sir Clive Sinclair lo trajo al mundo!

Antes de comenzar este *tour* hemos de hacer algunas aclaraciones: en primer lugar, conviene saber que nuestro ordenador está protegido por la garantía del Distribuidor Oficial: cualquier manipulación en el interior del aparato nos hace perder todo derecho a una reparación gratuita; por tanto, antes de romper el sello de garantía, tengamos muy en cuenta las posibles consecuencias de esta acción.

En segundo lugar, se dan breves indicaciones sobre los puntos desde los que podemos fabricar una salida de monitor para nuestro Spectrum; aunque esta operación se puede considerar muy sencilla, y consiste simplemente en la soldadura de dos cables a las entradas del modulador de U.H.F., no es recomendable, si no nos sentimos suficientemente capaces, aventurarnos con un soldador entre la nube de componentes de nuestro Spectrum.

El hablar de las memorias, nos da pie para aclarar una pequeña confusión que se puede producir sobre las características del Spectrum 48 K y el PLUS: la única diferencia existente entre am-

bos es el teclado, y un pequeño pulsador de *RESET* que se conecta a la placa mediante dos cablecillos. Excepción hecha de estas dos distinciones, y por supuesto las inherentes a pertenecer a diferentes series de producción (forma y situación de la placa disipadora de calor, distribución de pistas en la placa, etc.) ambos modelos son exactamente iguales.

Por último añadiremos que los Spectrum en su interior no son todos exactamente iguales, puesto que a lo largo del proceso de fabricación se han ido estableciendo determinadas mejoras que han dado lugar a diferentes versiones, aunque sólo desde el punto de vista *hardware*, puesto que externamente y a efectos de programación son iguales (salvo los valores devueltos por la función *IN*). Estas diferencias consisten en el trazado de las pistas, situación y forma de la placa disipadora de calor, etc. o incluso la posibilidad de que los modelos de 16 K sean ampliados internamente a 48 K, ya que en la primera versión del aparato, puesta a la venta por correo en el Reino Unido, esta posibilidad no existía, no pudiendo ser ampliables los modelos de 16 K.

Si nos pica la curiosidad de conocer la versión de nuestro Spectrum, encontraremos su nombre grabado en el anverso de la placa, en su zona inferior derecha (a la izquierda del altavoz), escrito tras la palabra *ISSUE* (serie).



He aquí los protagonistas de nuestra aventura: el Spectrum y el Plus.



!

Si no tenemos experiencia en electrónica, lo mejor que podemos hacer es dejar el interior del Spectrum para los profesionales.



A aquellos que no nos atrevamos a poner nuestro ordenador en la «mesa de operaciones», nos bastará echar un vistazo al mapa general que finaliza esta sección; en él encontraremos una fácil referencia sobre la situación aproximada de los principales componentes del Spectrum.

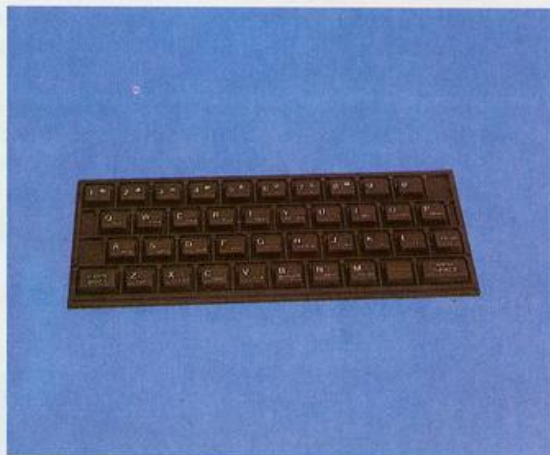


Recordemos que al romper el sello de garantía que se encuentra sobre el tornillo superior central, perdemos cualquier derecho a la reparación gratuita del ordenador.

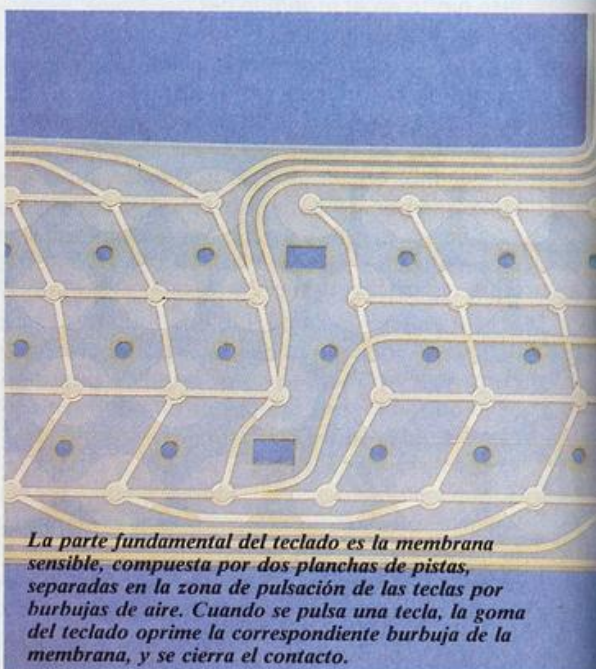
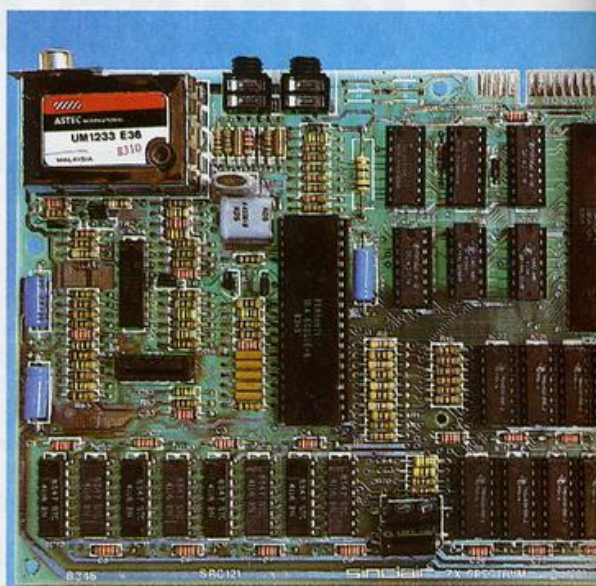
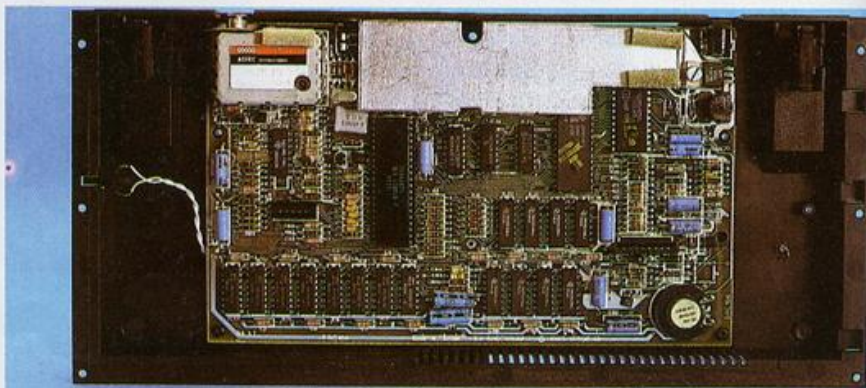


Si nuestro Spectrum no ha sido abierto nunca, el sello de garantía cerrará el paso hasta el tornillo superior central, sin quitar el cual no es posible acceder al interior del aparato.

La parte del teclado en contacto con el usuario es una plancha de goma; como podemos apreciar en la foto, se trata de un elemento continuo, en el cual las teclas son bloques sobresalientes de la superficie. Por tanto, al pulsar una tecla, esta no se desliza, sino que se «hunde» junto con las inmediaciones de la plancha.



Un par de cablecillos de color blanco, unidos a un pulsador en el lateral izquierdo de la caja, constituyen el sistema de RESET del Spectrum Plus. En la parte inferior derecha de la placa (a la izquierda del altavoz), aparece serigrafiado el tipo de versión de que disponemos.



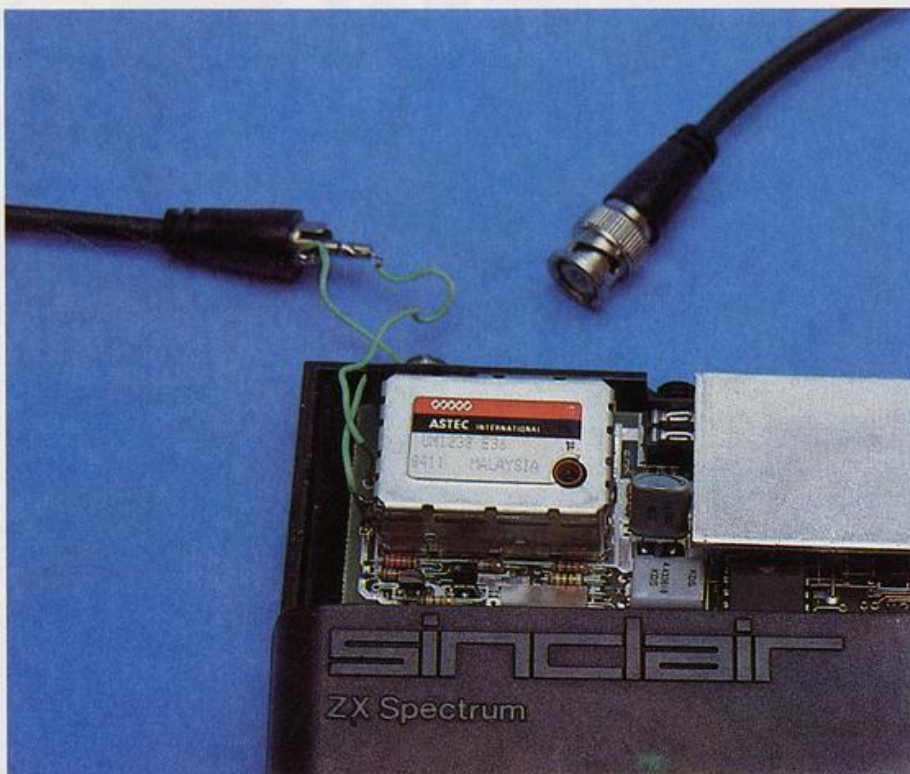
La parte fundamental del teclado es la membrana sensible, compuesta por dos planchas de pistas, separadas en la zona de pulsación de las teclas por burbujas de aire. Cuando se pulsa una tecla, la goma del teclado oprime la correspondiente burbuja de la membrana, y se cierra el contacto.



Al Spectrum que se observa en la foto, se le ha desprovisto de su placa disipadora de calor, con el fin de poder apreciar claramente todos y cada uno de los componentes del aparato.



Una vez desmontado el Spectrum, se divide en dos partes: la superior, con todo lo relacionado con el teclado; y la inferior, que contiene la placa. La unión entre ambas la efectúan los dos cables planos que parten del teclado, y se ajustan a presión en los conectores correspondientes en la placa. En la parte superior se aprecian cuatro elementos superpuestos consecutivamente: carcasa plástica, membrana sensible, teclado y placa metálica.



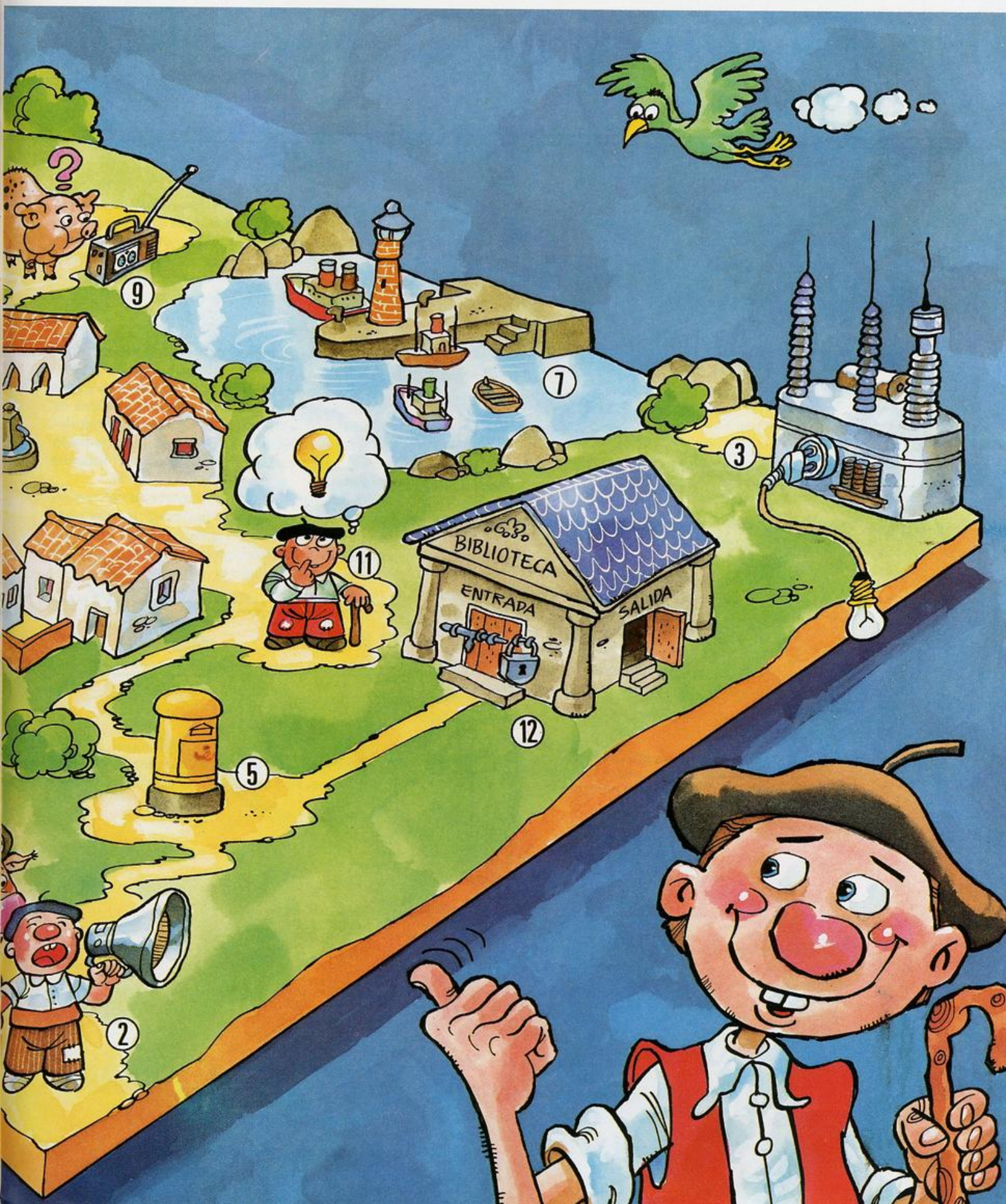
Probablemente una de las operaciones más sencillas a realizar en el interior de un Spectrum, sea fabricar una salida de monitor. Sólo es necesario soldar dos cables a los conductores de entrada al modulador, los cuales serán soldados a su vez al conector del cable del monitor.





Mapa aproximado del interior del Spectrum.

1. Modulador de televisión.
2. Altavoz
3. Fuente de alimentación
4. Memoria RAM
5. Conectores del teclado
6. Reloj
7. Port de comunicación exterior
8. Chips lógicos
9. Interface para casete
10. U.L.A.
11. C.P.U.
12. Memoria ROM



ANTIAEREO



E

L radiotelescopio de Arecibo ha detectado cientos de extraños objetos en dirección a la Tierra. Se cree que dichos artefactos voladores proceden de alguna estrella del cúmulo globular de Hércules. Vienen en formación de combate y dispuestos para la lucha.

Un radioaficionado ha captado una conversación entre dos naves. La posterior decodificación de dicha comunicación ha sacado a la luz las verdaderas intenciones de los extraterrestres: quieren llevarse a su planeta todos los Spectrum de la tierra, para su posterior destrucción, privando así a

la humanidad terrícola de uno de sus pasatiempos preferidos.

Los ejércitos de todo el mundo se han movilizado para hacerles frente. En lugares estratégicos de la corteza terrestre se han situado barreras defensivas, con lanzaderas de proyectiles de fisión nuclear, para tratar de evitar un Spectrumcidio en masa.

Nuestro deber es destruir la totalidad del convoy extraterrestre enemigo, y preservar la vida electrónica de nuestro gran amigo. Todo esto es lógicamente una ficción, ¿pero qué tal si nos vamos entrenando por si llega el momento...?

EL PROGRAMA

Dependiendo de la dificultad del blanco, el programa nos otorga más o menos puntos, a la hora de hacer impacto sobre un Spectrumcida extraterrestre.

El objetivo del juego es abatir el mayor número de OVNIS posible, con los 50 proyectiles de que disponemos. Hay que tener en cuenta que, como





es habitual, hasta que los misiles no alcancen el tope superior de la pantalla, no se puede volver a disparar, por lo que es importante que conservemos unos nervios de acero para hacer frente al bélico compromiso.

La puntuación que se obtiene por cada impacto está en relación a la dificultad de hacer blanco; así, por ejemplo, tendrá más valor destruir un OVNI cuanto más arriba de la pantalla aparezca; análogamente, proporciona mayor puntuación utilizar los proyectiles de los bunkers 1 y 2, que los del 3 y, a su vez, los del uno más que los del 2. Para el lanzamiento de proyectiles, emplearemos las teclas 1, 2 y 3, atendiendo a la lanzadera desde la que deseemos disparar.

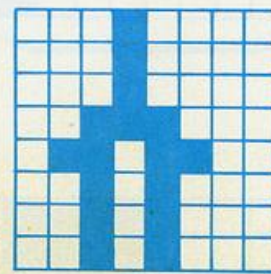
Para el proyectil de fisión nuclear, se ha definido especialmente el gráfico de la letra D.

Al finalizar el juego, el Spectrum compara el número de puntos obtenidos con el record que se tenga establecido hasta el momento, informándonos sobre la pericia demostrada en relación con la de nuestros predecesores.

Este juego es sin duda uno de los clásicos de la microinformática, y por su sencillez, sirve en muchas ocasiones de ejemplo para mostrar la estructura de un programa de acción. Es una buena idea que aquellos que queráis empezar a programar, utilicéis un programa con este esquema, al cual podréis ir añadiendo, según el ánimo con que conteis, todas las mejoras que consideréis oportunas.

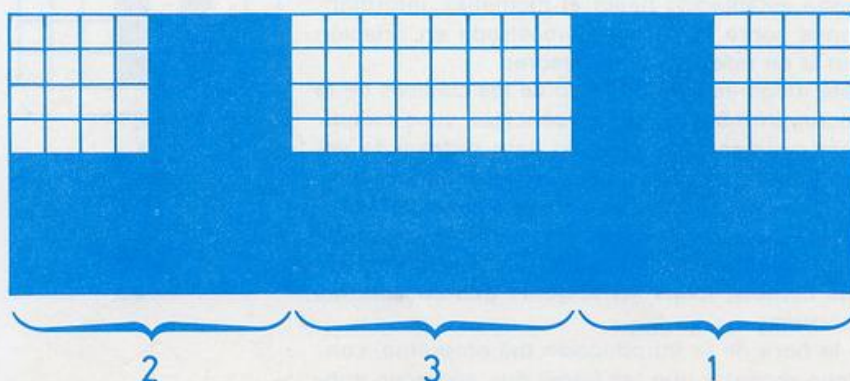
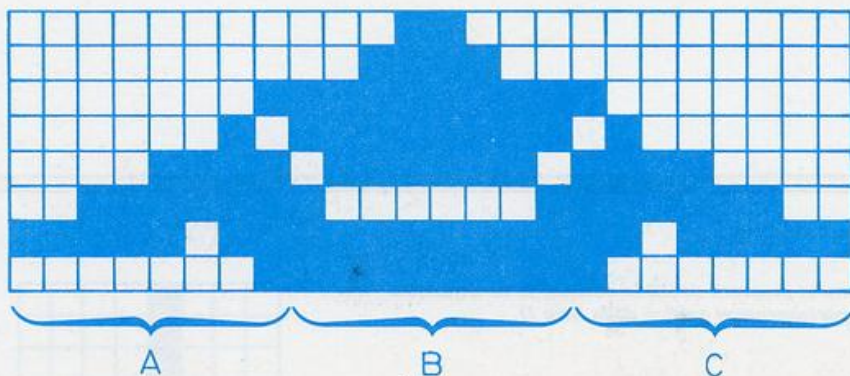
A la hora de la introducción del programa, conviene recordar que las letras que aparecen subrayadas en el listado, corresponden a los gráficos definidos de las teclas subrayadas. Si además el subrayado fuera doble (hecho que sólo encontraremos en los números), el gráfico es el que se obtiene pulsando simultáneamente **CAPS SHIFT**, lo que se conoce como **GRAFICO CAMBIADO**.

Veamos un ejemplo práctico: al llegar a la línea 338, encontraremos una **D** subrayada; debere-



D





La nave extraterrestre está compuesta por tres gráficos definidos, correspondientes a las teclas A, B y C.

mos por tanto pasar el ordenador a modo **GRAP-HICS (CAPS SHIFT + 9)**, y a continuación pulsar la tecla **D**; esto hará aparecer en la pantalla dicha letra mayúscula, no habiendo surtido efecto aparente la introducción de **GRAPHICS**; no debemos preocuparnos por ello, puesto que una vez ejecutado el programa, el ordenador se ocupará de cambiar esa **D** por el correspondiente proyectil de fisión nuclear. Una vez que hayamos introducido el gráfico, deberemos abandonar el modo **GRAPHICS (CAPS SHIFT + 9)**, para poder continuar con la introducción normal del listado.

La grabación del programa se efectuará mediante el uso combinado de los comandos **SAVE** y **LINE** de la siguiente forma: **SAVE "INVASION" LINE 630**.



Las lanzaderas de proyectiles, se han simulado, mediante los gráficos predefinidos que corresponden a las teclas 2, 3 y 1.

```

10 REM *****
20 REM * F. LOPEZ MARTINEZ *
30 REM *****
40 REM * Adapt. J.M. MAYORAL *
50 REM *****
60 REM
70 LET R=0: POKE 23658,8
80 BORDER 0: PAPER 0: INK 9
90 CLS
100 PRINT AT 21,0: PAPER 6:
110 PRINT AT 20,5: INK 2: '231': TAB 15: INK 3: '231': TAB 25: INK 4: '231'
120 PRINT PAPER 1: BRIGHT 1: AT 0,0: 'PUNTOS: 0', 'MUNICION: 50'
130 PRINT PAPER 8: OVER 1: AT 21,6: 1: AT 21,16: 2: AT 21,26: 3
140 LET D=0
150 LET P=0
160 LET M=50
170 LET A=INT (RND*14)+2
180 FOR L=0 TO 28
190 PRINT AT A,L: 'ABC'
200 BEEP .0015,10
210 IF D THEN GO TO 300
220 IF INKEY$="" THEN GO TO 400
230 IF INKEY$<"1" OR INKEY$>"3" THEN GO TO 400
240 LET K$=INKEY$: LET K=VAL K$+1
250 LET B=10*VAL INKEY$-4
260 LET D=1
270 LET M=M-1
280 LET H=19
290 PRINT PAPER 1: BRIGHT 1: AT 0,26: ' ': AT 0,26: M
300 LET H=H-1
310 IF H<A THEN GO TO 330
320 IF B>L AND B<L+4 THEN GO TO 430
330 PRINT INK K: BRIGHT 1: AT H,B: 'D': AT H+1,B: ' '
340 BEEP .0015,50
350 IF H>2 THEN GO TO 400
360 LET D=0
370 PRINT AT H,B: ' '

```

```

380 IF NOT M THEN GO TO 530
390 GO TO 400
400 NEXT L
410 PRINT AT A,29: ' '
420 GO TO 170
430 PRINT AT H+1,B: ' '
440 PRINT AT A,L: 'BOOM': INK K: BRIGHT 1: AT A,L-1: 'A': AT A,L+5: 'C'
450 FOR Q=1 TO 10
460 BEEP .1,40
470 NEXT Q
480 LET D=0
490 PRINT AT A,L-2: ' '
500 LET P=P+(16-A)*((4-((B-6)/10+1))
510 PRINT PAPER 1: BRIGHT 1: AT 0,8: P
520 IF M THEN GO TO 170
530 IF P<R THEN GO TO 570
540 PRINT AT 9,3: 'ESTABLECIO UN NUEVO RECORD'
550 LET R=P
560 GO TO 580
570 PRINT AT 9,9: 'EL RECORD SIGUE'
580 PRINT AT 11,16-((10+LEN STR$ R)/2): 'EN ': R: ' PUNTO S'
590 PRINT AT 13,3: 'DESEA INTENTARLO DE NUEVO ?'
600 IF INKEY$="S" THEN GO TO 80
610 IF INKEY$<"N" THEN GO TO 600
620 GO TO 10000
630 REM GRAFICOS DEFINIDOS
640 DATA 0,0,1,2,15,63,251,1
650 DATA 24,60,255,255,126,129,255,255
660 DATA 0,0,128,64,240,252,223,128
670 DATA 16,16,16,56,108,40,40,40
680 LET A$="ABCD"
690 FOR N=1 TO LEN A$
700 FOR F=0 TO 7
710 READ A
720 POKE USR A$(N)+F,A
730 NEXT F
740 NEXT N
750 RUN

```