

A. BELLIDO

BASIC

ZX Spectrum+

Curso acelerado
para principiantes
Guía de comandos



Curso acelerado para principiantes Guía de comandos

Editado por GEASA

C./ S. Telmo, 24

Tfno.: 259 31 06 - 259 30 06

28016 Madrid

© 1985 A. Bellido.

Reservados los derechos de edición, reproducción y adaptación para todos los países.

Diseño: GORRINDO

Impreso en España

Printed in Spain

I. S. B. N.: 84 - 86019 - 15 - X

Depósito Legal: M. 16417 - 1985

Imprime: SIGNO IMPRESORES, S.A.

Indice

Introducción

1. *Conceptos previos*

2. *Guía de comandos*

3. *BASIC. Curso acelerado
para principiantes*

4. *Apéndices*

— Apéndice 1. Manejo del teclado

— Apéndice 2. Facilidades de edición

— Apéndice 3. Funciones lógicas aplicadas ..

— Apéndice 4. Troceado de cadenas

Introducción

Con las iniciales «Beginners All-Purpose Symbolic Instruccion Code» se forma la palabra BASIC.

El BASIC es un lenguaje de programación de alto nivel que fue desarrollado en el Dartmouth College en los Estados Unidos de América del Norte.

El BASIC de Sinclair es un dialecto que, además de utilizar la «filosofía» y comandos generales del BASIC, ofrece características que le son propias y de gran comodidad para el usuario.

Esta Guía de Comandos pone a disposición del lector una exposición detallada, y en orden alfabético, de todas las palabras que componen el BASIC DE SINCLAIR.

Las personas sin conocimientos previos de programación, pueden encontrar al final del libro un CURSO ACELERADO PARA PRINCIPIANTES, con el que podrán, posteriormente, comprender la GUIA DE COMANDOS.

En todo caso las páginas dedicadas a CONCEPTOS PREVIOS deben ser leídas.

En el Apéndice I se explica el manejo del teclado y, se ofrece una Carta para las entradas a las diferentes teclas.

En el Apéndice 2 se tratan las facilidades de borrado e inserción.

Conceptos previos

Programas

Un programa es una secuencia de instrucciones susceptibles de ser interpretadas por un computador, el cual las ejecutará, una tras otra, de principio a fin.

Línea de programa

La secuencia de instrucciones que conforma el programa, se establece gracias a las líneas de programa, cada una de las cuales está estructurada de la siguiente forma:

<i>Número de línea</i>	<i>Comando</i>	<i>Argumento</i>	<i>Fin de línea</i>
------------------------	----------------	------------------	---------------------

El *Número de línea* establece el orden de ejecución de las instrucciones, desde el menor número de línea al mayor, sin que por ello deban ser consecutivos.

Los números de línea deben estar comprendidos entre 0 y 9999.

El *Comando* puede ser cualquier palabra del BASIC, la cual, cumplirá su función sobre el *Argumento*.

El *Fin de línea* se establece al apretar la tecla ENTER, con lo que la línea en cuestión pasa a la memoria del computador.

Una línea de programa puede estar formada por una sola sentencia como la esquematizada más arriba, o bien por varias, con lo que se tendría una línea

multisentencia como la que sigue:

Número de línea Comando Argumento : Comando Argumento Fin de línea

Como puede apreciarse, una sentencia se separa de la siguiente por medio de los dos puntos (:), excepto la última que requiere el ENTER para establecer el Fin de línea.

Mediante el símbolo «>», denominado *puntero*, que aparece entre el número de línea y el comando, el Spectrum indica cuál es la última línea manipulada.

Modos de operar

Cuando una sentencia es introducida en el Spectrum sin número de línea, se ejecutará inmediatamente al apretar ENTER. Esta forma de operar se conoce como *MODO DIRECTO*.

En modo directo también se pueden introducir multisentencias.

El *MODO PROGRAMA* exige introducir las sentencias con sus número de línea correspondientes y sólo serán ejecutadas secuencialmente mediante RUN y ENTER/

El cursor y sus modos

Al introducir una sentencia en el computador, un caracter parpadeante va precediendo al último caracter o comando tecleado; a este indicador se le conoce por el nombre genérico de *cursor*.

En el Spectrum, este cursor queda representado por diferentes caracteres, cada uno de los cuales indica el *modo* en que va a trabajar el computador. Así:

- K. Avisa que aceptará la palabra BASIC que figure sobre la superficie de la tecla. En modo programa también aceptará los números correspondientes a la línea de programa.
- L. Previene que, al apretar cualquier tecla, interpretará el caracter —letra o número— que figure sobre la superficie de la tecla. En el caso de las letras, las considerará minúsculas; para obtener mayúsculas debe apretarse simultáneamente CAPS SHIFT.
- C. Produce el mismo efecto que el cursor en modo L, pero con todas las letras en mayúsculas. Este modo se obtiene apretando simultáneamente CAPS SHIFT y CAPS LOCK.
- E. Equivale a modo extendido y se consigue apretando EXTEND MODE. Una vez el cursor en este modo, bastará apretar la tecla donde figure la palabra BASIC deseada para obtenerla, y que corresponderá a la más externa de todas las que aparecen. La palabra BASIC que figura debajo de la anterior se consigue, una vez en modo E, apretando, a la vez, SYMBOL SHIFT y la tecla en cuestión.
- G. Permite la impresión de los caracteres gráficos situados en las teclas donde

figuran los dígitos decimales (1 a 8) y los caracteres definidos por el usuario. En este modo se entra apretando la tecla GRAPH, y se sale de la misma forma.

Para ampliar detalles sobre estos conceptos ver el APENDICE 1 «TECLADO».

Algoritmo

Un *algoritmo* está asociado a cualquier procedimiento de cálculo, pero, en términos informáticos, nos referimos al conjunto de operaciones realizadas sobre unos datos para obtener un resultado.

En definitiva, con los algoritmos lo que se pretende es organizar las operaciones y minimizar su número.

Los *datos* son valores dados a las variables que intervienen en el algoritmo.

El *resultado* es el valor obtenido al efectuarse las operaciones que componen el algoritmo sobre los datos.

Un algoritmo debe estar definido de tal manera que no produzca confusión.

Constantes

Las *constantes* son aquellos elementos de un algoritmo cuyo valor permanece invariable.

En términos de programación, hay dos clases de constantes: *numéricas* y *alfanuméricas*.

Las constantes numéricas pueden ser cualquier valor numérico positivo o negativo, entero o decimal. Como por ejemplo 3 ó —33.33.

Las alfanuméricas están compuestas por cualquier combinación de caracteres escritos entre comillas.

Ejemplos: "Spectrum"
"24 de diciembre"

Variables

Las *variables* son aquellos elementos de un algoritmo capaces de tomar cualquier valor.

En programación hay dos tipos de variables: *numéricas* y *alfanuméricas*.

Las primeras quedan representadas por cualquier cadena de caracteres, con la única condición de que el primero de todos ellos sea una letra. Ejemplos: LET A = 1 o LET A2 = 1520.

Las alfanuméricas o de caracteres deben ser representadas por una sola letra seguida del símbolo \$, para que el computador la distinga de las numéricas.

El valor que se asigne a una variable alfanumérica, debe ir entre comillas. Ejemplo:

LET A\$ = "Spectrum" o LET R\$ = "24 de diciembre".

Denominamos genéricamente por *nombre de la variable* a la representación de las mismas, así, en los ejemplos anteriores, tenemos los siguientes nombres de variables:

A, A2, A\$ y R\$.

Operadores

Los *operadores* son los símbolos que determinan las operaciones que se han de realizar entre valores.

Hay cuatro tipos de operadores: *Aritméticos, de relación, lógicos y funcionales.*

Expresiones

Una *expresión* es un conjunto de valores relacionados entre si por medio de operadores, cuyo resultado final es una cantidad.

Con respecto a las variables se dijo que eran elementos capaces de contener un valor, una expresión, sin embargo, *es* un valor.

Ejemplo: Definir una expresión que determine el valor de un viaje en taxi, sabiendo que cada «salto» del contador cuesta 25 ptas.

Si admitimos que la variable X representa el número de «saltos» del contador, el importe del viaje vendrá dado por la expresión: $25 * X$.

El operador que relaciona ambos valores es `""`.

Claro está, para cada valor de X tendremos un nuevo valor de la expresión $25 * X$.

Las expresiones matemáticas se escriben en BASIC prácticamente igual que con lápiz y papel, pero con las variaciones impuestas por los símbolos que figuran en el teclado del ordenador. Tal es el caso, por ejemplo, de x^3 (equis elevado a 3), que nos obligará a teclear `x 3`.

El orden de prelación de las diferentes operaciones matemáticas se verá más tarde, pero, antes de llegar al mismo, es importante hacer observar unas sutiles diferencias que existen entre escribir una expresión matemática a mano o en un computador. Supongamos que deseamos transcribir la expresión $\frac{a+b}{2c}$ de forma entendible para el ordenador. Para lograrlo, no debemos

olvidar ningún signo —como `*` entre 2 y c en el denominador— y no magnificar la capacidad de la máquina, ya que:

- Si escribimos `a + b/2*c`, el ordenador interpretará $a + \frac{b}{2} * c$.
- Si escribimos `(a + b)/2*c`, interpretará $\frac{a + b}{2} * c$.
- Sólo podrá interpretar correctamente la expresión citada si escribimos `(a + b)/(2*c)`.

Operadores aritméticos

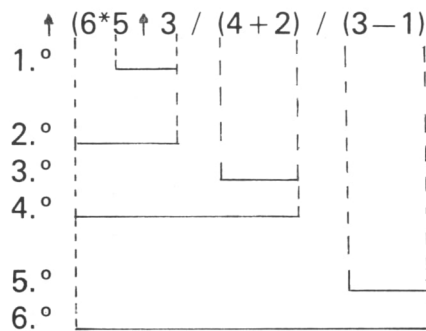
Se denominan así porque actúan entre valores aritméticos, siendo el conjunto de los símbolos que los componen y su orden de prioridad establecidos por el ordenador los siguientes:

PRIORIDAD	OPERACION	OPERADOR
Máxima	exponenciación	
	multiplicación y división	*
	división	/
	suma y resta	+
mínima	resta	—

Para operadores de la misma prioridad, la máquina ejecuta las operaciones de izquierda a derecha.

Las operaciones dentro de un paréntesis las efectúa primero y siguiendo el orden de prioridad anterior.

Ejemplo:



Calcula la potencia $5 \uparrow 3$ (5^3).

Multiplica el resultado por 6.

Calcula $4 + 2$.

Divide el producto (2.º) entre la suma (3.º).

Calcula $3 - 1$.

Divide el cociente (4.º) entre la diferencia (6.º).

Las cadenas de caracteres sólo pueden sumarse.

La forma de yuxtaponer una cadena de caracteres a otra es intercalar entre ellas el operador +.

Ejemplo:

PROGRAMA

COMENTARIO

10 LET A\$ = "ABC" Con las líneas 10 y 20 asignamos valores a las variables de caracteres A\$ y B\$

20 LET B\$ = "DEF"

30 PRINT A\$ + B\$ Con la línea 30 ordenamos la impresión del contenido actual de A\$ y B\$, justamente uno a continuación del otro.

El resultado de ejecutar este programa será: ABCDEF

Operadores de relación

Estos operadores sólo actúan en proposiciones entre dos operandos, cuyo resultado únicamente puede ser CIERTO, que se representará por 1, o FALSO, que se representará por 0.

Sus símbolos son:

igual que		=
distinto que		< >
menor que		<
mayor que		>
menor o igual que	..	< =
mayor o igual que	..	> =

Si un operador de relación compara dos expresiones en las que intervienen operadores aritméticos, antes de efectuar la comparación, actúan los operadores aritméticos.

Ejemplo:

Determinar si $2 + 5$ es menor que $(15 + 4)^3 / 20250$.

Solución:

Es evidente que, sin efectuar las operaciones aritméticas que determinan el valor de cada expresión, es difícil responder a la proposición anterior.

Las cadenas de caracteres pueden ser comparadas con estos mismos operadores. Para hacerlo se cotejan, carácter a carácter, ambas cadenas valorando cada carácter de acuerdo con su *código ASCII*. Más adelante se estudiará todo lo referente a los códigos, baste por ahora saber que cada carácter tiene su propio valor.

Operadores lógicos

Responden directamente a las *funciones lógicas del álgebra de BOOLE*.

Los operadores lógicos actúan entre operandos en los que, a su vez, intervienen operadores de relación. Esto quiere decir que los operandos de los operadores lógicos tendrán siempre el valor 1 (CIERTO) ó 0 (FALSO), y, por consiguiente, *los operadores lógicos* sólo responden 1 ó 0 (CIERTO o FALSO).

Los operadores lógicos NOT, AND y OR que se estudiarán con detalle más adelante.

Operadores funcionales

Estos son algoritmos que residen en el propio BASIC y operan sobre datos suministrados al computador por medio del teclado o el programa, obteniéndose como resultado un valor.

Coma flotante

El ingeniero Lonardo Torres Quevedo fue el primero en aplicar el concepto de la *coma flotante* a los computadores.

Cualquier número, positivo o negativo, puede ser representado en forma exponencial, para ver esto, supongamos que al medir una distancia escribimos 525 mts.

Evidentemente, podríamos haber representado lo mismo con: 0.525×10^3 y esto, en definitiva, es lo que representa el concepto de la coma flotante.

Ejemplo: PRINT 124*1000 000

Al ejecutar esta instrucción en modo directo el ordenador nos retorna 1.24 E + 8.

La mantisa queda representada por 1.24 y el índice por 8.

Este sistema permite, además de un medio económico de codificación, un amplio campo de representación numérica.

Conexión y puesta en marcha

Para convertir su computador en un ordenador propiamente dicho deberá disponer de :

- Un Spectrum Plus (o Convencional)
- Un televisor (o monitor)
- Un casete (si desea guardar y reproducir información)
- Un transformador 220V/9V (con el equipo estandar)
- Una impresora (si desea sacar copias en papel de la información)

La función que cumple cada elemento del ordenador es la siguiente:

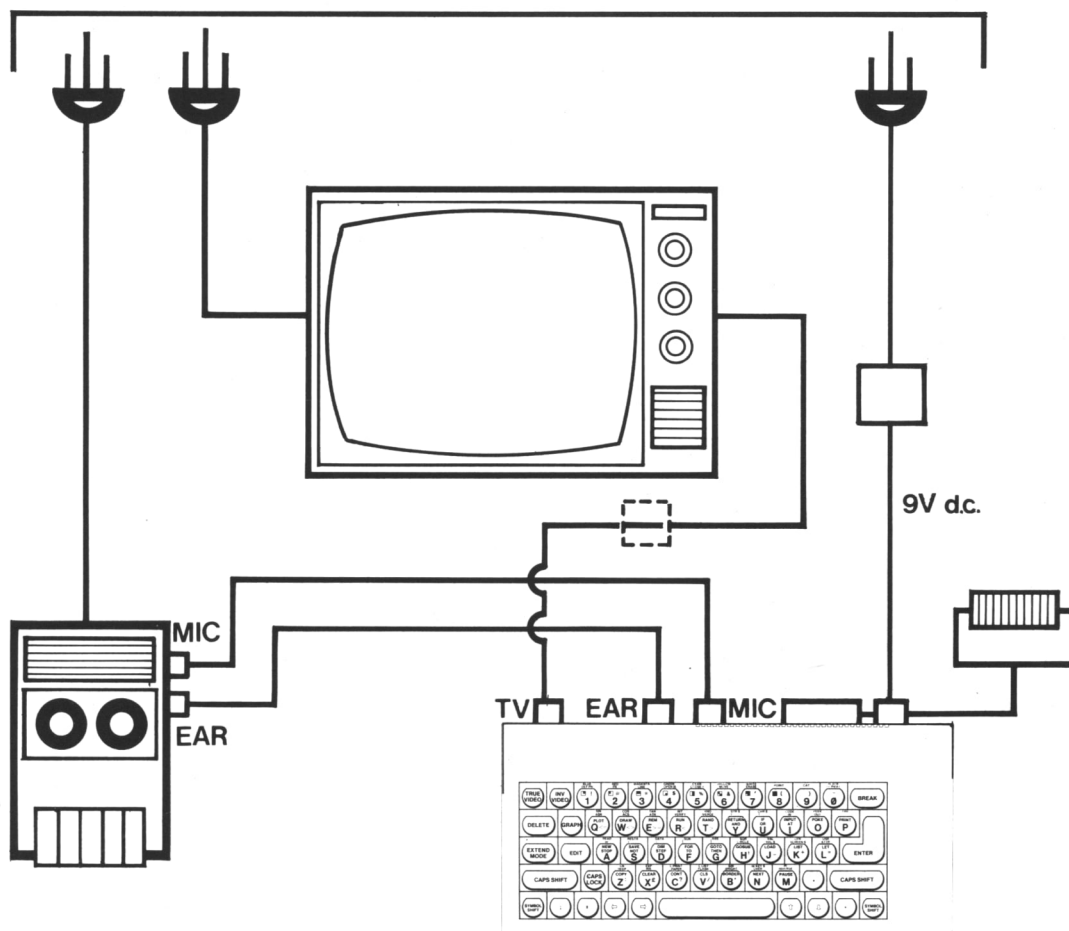
Computador: El Spectrum contiene toda la electrónica necesaria para el manejo y almacenamiento de la información y posee la adecuada conexión, interna, al teclado.

En la parte posterior se pueden ver los enchufes *EAR*, *MIC*, *TV* y *9VDC*. Mediante el primero se da entrada de información procedente de un casete. Con los dos siguientes (*MIC*, *TV*) se permite la salida de información hacia el televisor y el casete. Mediante *9VDC* se conecta el transformador que adapta la energía eléctrica a las necesidades de trabajo de la máquina (9V/1,2A).

Teclado: Gracias al teclado se consigue una comunicación directa entre el usuario y el ordenador, introduciendo información a través de él. Este dispositivo es sólo de entrada.

Televisor: Este dispositivo es sólo de salida, ya que permite al computador emitir sus respuestas por la pantalla del mismo.

Esquema de conexiones del Spectrum



Casete: Es un medio de almacenamiento externo que permite transferir información desde una cinta magnética al computador y, en sentido contrario, desde el computador a la cinta.

Impresora: Es un dispositivo de salida que permite al usuario proveerse de copias en papel de información contenida en el computador.

Guía de comandos

Interpretación

Todos los comandos del BASIC de Sinclair están descritos en esta Sección y en orden alfabético.

La explicación de cada comando se ciñe a la siguiente estructura:

Comando y forma de acceder al mismo.

Sintaxis: En este apartado se detalla la forma correcta de escribir la sentencia en la que está incluido el comando en cuestión.

Aplicación: Resume el campo de utilización del comando.

Ejemplo: Uno o varios ejercicios explicados, apoyan lo expuesto en los apartados anteriores.

La notación aplicada responde a los siguientes criterios:

- 1.— Las palabras en mayúsculas corresponden a comandos del BASIC.
- 2.— La información contenida entre dos signos de interjección — ¡.....! *debe ser* suministrada por el programador imprescindiblemente.
- 3.— La información contenida entre dos barras verticales — — *puede ser* suministrada por el programador, si así lo estima conveniente.
- 4.— Si la palabra *expresión* va entrecomillas, se refiere a una expresión alfanumérica. En caso contrario a una expresión numérica.
- 5.— μd significa microdrive.
- 6.— μc significa microcinta.

ABS

G. Cursor E

ABS

Sintaxis: ABS ¡expresión!

Aplicación: Convierte la **expresión**, sea cual sea su signo, en la misma **expresión** numérica pero con signo positivo.

Ejemplo: Al teclear en modo directo:
PRINT ABS-1.2 obtendremos como resultado 1,2.

ACS

W y SYMBOL SHIFT. Cursor E

ACS

Sintaxis: ACS ¡expresión!

Aplicación: Determina el arco coseno de la **expresión** numérica.

Ejemplo: Para determinar el arco cuyo coseno es -0.41614684 , bastará con teclear en modo directo:
PRINT ACS-.41614684.

AND

Y y SIMBOL SHIFT. Cursor K

AND

Ver apéndice funciones lógicas.

ASN

Q y SIMBOL SHIFT. Cursor E

ASN

Sintaxis: ASN ¡**expresión**!

Aplicación: Determina el arco seno de la **expresión** numérica.

Ejemplo: Para determinar el arco cuyo seno es -0.41614684 , bastará con teclear en modo directo:
PRINT ASN-.41614684.

AT

I y SIMBOL SHIFT. Cursor K

AT

Sintaxis: Este comando debe ir precedido en la sentencia por PRINT, de esta forma
PRINT AT ¡**expresión A**, **expresión B**!; *expresión numérica o alfanumérica*.

Donde **expresión A** debe dar — o ser — un número comprendido entre 0 y 21, ya que determina la **línea** donde debe iniciarse la impresión. Y **expresión B** debe dar — o ser — un número comprendido entre 0 y 31, ya que determina la columna donde debe iniciarse la impresión.

Aplicación: Fija la línea y la columna donde debe iniciarse la impresión de la *expresión numérica o alfanumérica* que vaya a continuación del punto y coma.

Ejemplo: Teclear en modo directo:
PRINT AT 0,0: "TOTAL"; AT 10,20;250
Un solo comando PRINT al comienzo de la sentencia, como puede apreciarse, es válido para los sucesivos AT que estén encadenados mediante los correspondientes puntos y comas.

ATN

ATN E y SIMBOL SHIFT. Cursor E

Sintaxis: $\overset{\vee}{\text{ATN}}$ ¡expresión!

Aplicación: Determina el arco tangente de la **expresión** numérica.

Ejemplo: Para determinar el arco cuya tangente es -0.41614684 , bastará con teclear en modo directo:
PRINT ATN-.41614684.

ATTR

L y SIMBOL SHIFT. Cursor E **ATTR**

Sintaxis: ATTR ¡expresión A, expresión B!
Donde **expresión A** debe dar —o ser— un número comprendido entre 0 y 23 y **expresión B** debe dar —o ser— un número comprendido entre 0 y 31.

Aplicación: Al aplicar esta función, el Spectrum devuelve un número, el cual, debidamente interpretado, determina los atributos que están afectando a la posición de carácter prefijada por **expresión A** y **expresión B**.

El número así obtenido es el resultado de la suma de cuatro sumandos, cada uno de los cuales representa lo que sigue:

FLASH .. 128 si parpadea, 0 en otro caso

BRIGHT .. 64 si tiene brillo, 0 en otro caso

PAPER .. Resultado de multiplicar 8 por el número que identifica el color del papel

INK número que identifica el color de la tinta

TOTAL .. número obtenido al aplicar ATTR

Ejemplo: 10 PRINT AT 0, 0; FLASH 0; BRIGHT 0; PAPER 0;
 INK 0; " "
 20 PRINT ATTR (0,0)

El número obtenido en pantalla como consecuencia de la línea de programa 20 será 0 que es el menor posible.

Si modificamos la línea 10 de esta forma:

10 PRINT AT 0, 0; FLASH 1; BRIGHT 1; PAPER 7; INK 7; " "

El número devuelto por la función ATTR será 256 que es el mayor posible.

BEEP

Z y SIMBOL SHIFT. Cursor E

BEEP

Sintaxis: BEEP ¡expresión A, expresión B!

Donde **expresión A** debe dar —o ser— un número mayor que 0 y determina la duración del sonido, y **expresión B** debe dar —o ser— un número que —a efectos prácticos— debe estar comprendido entre -69 y $+69$, y representa la nota musical.

Aplicación: No es posible producir un sonido y, simultáneamente, hacer alguna otra cosa sobre el Spectrum.
La equivalencia entre las representaciones de las notas y la **expresión B** es:

do	< >	C	< >	0
re	< >	D	< >	2
mi	< >	E	< >	4
fa	< >	F	< >	5
sol	< >	G	< >	7
la	< >	A	< >	9
si	< >	B	< >	12

Para subir o bajar una octava por nota, se debe sumar o restar 12.

Ejemplo:

```
10 LET X=RND
20 BEEP X, INT (12-24*X)
30 GOTO 10.
```

BIN

B Cursor E

BIN

Sintaxis: BIN ¡Seguido de 8 dígitos binarios!

Aplicación: A todos los efectos se puede utilizar el sistema binario de representación mediante este comando, aunque los resultados que devuelve el Spectrum sean siempre en base 10.

Ejemplo: PRINT 0000010*00000010.

BORDER

B Cursor K

BORDER

Sintaxis: BORDER ¡expresión!

Donde **expresión** debe ser —o dar— un número comprendido entre 0 y 7.

Aplicación: Fijar un determinado color a la zona de pantalla que bordea el área de impresión.

Ejemplo: 10 FOR X=0 TO 7
20 BORDER X
30 NEXT X
40 GOTO 10.

BRIGHT

B y SIMBOL SHIFT. Cursor E

BRIGHT

Sintaxis: BRIGHT ¡expresión!
Donde **expresión** debe ser —o dar— 0 ó 1.

Aplicación: BRIGHT 1 introduce en la posición o posiciones de carácter que le siguen la condición de aparecer en pantalla sobreiluminada, con BRIGHT 0, se cancela tal condición.

Ejemplo: PRINT BRIGHT 1; AT 11, 15; “*”.

CAT

9 y SYMBOL SHIFT. Cursor E

CAT

Sintaxis: CAT#r; ¡n.º μ d!
n.º μ d = número del microdrive a ser manipulado.
r = es un número representativo del **rumbo** (stream) que ha de seguir la información.

Aplicación: Muestra el catálogo de ficheros que contiene la μ c instalada en el μ d definido.
Al final de este listado aparece un número representativo de la μ c aún disponible y medida en Kb.
Al comienzo de este listado aparece, en mayúsculas, el **nombre** dado a la μ c en el momento de su formateado.
Si el **rumbo** no se define, el catálogo se muestra en pantalla.
Si el **rumbo** se define, el catálogo será enviado, según

el número del mismo, a la impresora o a la red de comunicación interior (**network**).

Observar, en el ejemplo que sigue, la sintáxis al usar **stream**, en lo relativo al símbolo # y al punto y coma.

Ejemplo: Sin rumbo : CAT 1

En el supuesto de querer obtener el catálogo por impresora:

CAT # 3;1

U Cursor E

CRH\$

CRH\$

Sintaxis: CHR\$ ¡expresión!

Donde **expresión** debe ser —o dar— un número comprendido entre 0 y 255.

Aplicación: Cada número del campo citado se corresponde con un símbolo de los manejados por el Spectrum.

En el ejemplo que se ve a continuación, algunos números dan como caracter equivalente "?", esto implica que no hay símbolo correspondiente.

Ejemplo: 10 FOR X=24 TO 255
20 PRINT X, CRH\$ X
30 NEXT X

Los 24 primeros códigos son de control, o por el contrario, no son utilizados.

CIRCLE

H y SIMBOL SHIFT. Cursor E

CIRCLE

Sintaxis: CIRCLE ¡X,Y,r!
Donde X e Y son las coordenadas del centro de una circunferencia de radio r.

Aplicación: Trazado de circunferencias.
Los valores de **X** deben estar comprendidos entre 0 y 225, los de **Y** entre 0 y 175, el valor de **r** tiene que estar en consonancia con los de **X** e **Y** para evitar que la circunferencia se «salga» de la pantalla, lo que daría un error.

Ejemplo:
10 FOR Z=0 TO 80
20 CIRCLE Z, Z, Z
30 NEXT Z

CLEAR

X. Cursor K

CLEAR

Sintaxis: CLEAR **expresión**

Aplicación: Si CLEAR se utiliza sin **expresión** se borran las variables, se borra la pantalla y se inicializa el fichero de los DATA.
Si se utiliza con **expresión**, además de los efectos indicados anteriormente se sitúa la RAMTOP en la dirección de memoria indicada por **expresión** y se sitúa la pila de los GOSUB en concordancia.

Ejemplo: Si se corre el programa del comando CIRCLE y a conti-

nuación se hace un CLEAR, un PRINT Z, en modo directo, dará un mensaje de «variable no localizada».

5 y SIMBOL SHIFT. Cursor E

CLOSE

CLOSE

Sintaxis: CLOSE# ¡r!

r = es un número representativo del rumbo (stream) que ha de seguir la información.

Aplicación: Cierra un canal que previamente fue abierto (OPEN#) a la CPU mediante un **rumbo** (stream) determinado. Si el canal en cuestión está siendo utilizado como entrada de información, los datos no recibidos antes del CLOSE# , se pierden.

¡Atención! No hacer pruebas con este comando sin tener conectado el Interface 1.

V. Cursor K

CLS

CLS

Sintaxis: CLS

Aplicación: Borra la pantalla.

Ejemplo: 10 FOR X=0 TO 21
20 PRINT AT X, X; X
30 NEXT X
40 CLS.

CODE

I. Cursor K

CODE

Sintaxis: CODE ;"expresión"!

Aplicación: Se obtiene el código numérico correspondiente al primer caracter de "expresión".

Ejemplo: PRINT CODE "Ab".

CONT

C. Cursor K

CONT

Sintaxis: CONT

Aplicación: Permite continuar la ejecución tras un BREAK o un STOP, según estas posibilidades:

Si el mensaje del Spectrum es "D BREAK-CON repeats" se repite la instrucción donde se detuvo el programa.

Si el mensaje es "L BREAK into program", el programa sigue la ejecución en la próxima instrucción. Igual sucede con el mensaje "STOP statement".

Ejemplo: 10 FOR X=1 TO 20: PRINT "P": STOP: NEXT X
Cada vez que aparezca el mensaje apriete CONT.

COPY

Z. Cursor K

COPY

Sintaxis: COPY.

Aplicación: Copia en papel, y mediante impresora, el contenido actual de la pantalla.

Ejemplo: Ejecutar el siguiente programa:

```
10 FOR X=0 TO 21  
20 PRINT AT X, X; X  
30 NEXT X
```

Y a continuación teclear en modo directo:
COPY seguido de ENTER.

COS

W. Cursor E

COS

Sintaxis: COS ¡expresión!.

Aplicación: Calcula el coseno de un ángulo medido en radianes, por tanto **expresión** debe dar —o ser— un valor comprendido entre 0 y 2PI.

Ejemplo: PRINT COS 0.

DATA READ

D. Cursor E

DATA

A. Cursor E

READ

Sintaxis: DATA ¡**constante 1, constante 2..., constante n!**
Cada constante puede ser numérica o alfanumérica, en este ultimo caso debe ir entre comillas.

Aplicación: Almacena valores y los deja a disposición del comando READ. Dichos valores deben ir separados unos de otros con una coma.
En una sentencia DATA, se puede colocar tantas constantes como sea conveniente o, en sentido contrario, las constantes a almacenar se pueden distribuir en tantos DATAS como sea adecuado.

Ejemplo: Para almacenar en un DATA, cinco constantes alfanuméricas, se podrá proceder así:
9000 DATA "aa", "bb", "cc", "dd", "ee".

READ

A. Cursor E

READ

Sintaxis: Read ¡**variable 1, variable 2..., variable n!**
Cada **variable** puede ser numérica o alfanumérica y debe ir separada de la siguiente por una coma.

Aplicación: Léa los valores contenidos en los DATA de tal forma, que la **variable** 1 tomará el valor de la **constante** 1, y la **variable** n tomará el valor de la **constante** n, debiendo ser cada variable contenida en el READ del mismo tipo — numérica o alfanumérica— que la constante a la cual accede.

Las variables contenidas en una sola sentencia READ, puede acceder a las constantes almacenadas en varias DATA, comenzando por el primer valor del primer DATA y, cuando no haya más constantes pasará al siguiente DATA y así sucesivamente.

En el caso de que hubiera más variables en una sentencia READ, que constantes en el conjunto de sentencias DATA se producirá el error OUT OF DATA.

Con el comando RESTORE se coloca el puntero que controla la constante a leer por READ, al comienzo de todas ellas.

Ejemplo: Para acceder a las cinco constantes alfanuméricas citadas en el ejemplo de DATA, se puede proceder, p. e.:

```
10 READ A$,B$,C$,D$,E$  
20 PRINT A$,B$,C$,D$,E$  
9000 DATA "aa", "bb", "cc", "dd", "ee"
```

```
10 FOR x=0 TO 4  
20 READ Y$  
30 PRINT Y$  
40 NEXT x
```



```
9000 DATA "aa", "bb"  
9010 DATA "cc", "dd", "ee"
```

DIM

D. Cursor K

DIM

Sintaxis: DIM ¡**nombre** (elementos)!

Aplicación: Reserva en memoria espacio suficiente para todas las variables que configuran los **elementos**, creando simultáneamente un conjunto de variables de igual **nombre**, distinguiéndose unas de otras por los subíndices.

La matriz de variables así configuradas, sólo puede tener por **nombre** una letra, y si las variables que la integran son alfanuméricas, esa letra debe ir seguida del símbolo \$. El comando DIM debe ser programado de forma que se procese antes de pretender usar alguna de las variables que componen la matriz.

En el momento de procesarse una sentencia DIM, todas sus variables están a 0, si se trata de una matriz numérica, y si es alfanumérica todas las variables están integradas por **espacios**.

En el caso de las matrices de caracteres — alfanuméricas — los **elementos** se definen indicando en primer lugar el número de variables y, separados por una coma, los caracteres que, como máximo, pueden integrar cada variable.

Para las matrices numéricas, basta con explicitar el número de variables por línea y columna.

Ejemplo:

Matrices numéricas

```
10 DIM A(2,3)
20 FOR X=1 TO 2
30 FOR Y=1 TO 3
40 INPUT A(X,Y)
50 NEXT Y
60 NEXT X
```

Con este programa creamos en la línea 10, un grupo de variables numéricas, cuyo nombre genérico es A, y cuyos subíndices son:

A(1,1) A(1,2) A(1,3)
A(2,1) A(2,2) A(2,3)

Inicialmente estas variables contienen el valor 0.

El resto del listado sirve para valorar cada una de las variables.

En estas condiciones, teclear en forma directa

```
PRINT A(1,3)
```

para comprobar que tal variable ha sido valorada.

Ejemplo:

Matrices de caracteres

```
10 DIM A$(2,3)
20 FOR x=1 TO 2
30 INPUT A$(x)
40 NEXT x
```

En la línea 10 se define una matriz que está compuesta por 2 variables de caracteres, cuyo nombre genérico es

A\$ y, cada una de las cuales puede tener, como máximo, 3 caracteres.
Inicialmente estas variables están compuestas por espacios en blanco.
El resto del programa sirve para valorar las variables.

DRAW

W. Cursor K

DRAW

- Sintaxis:** DRAW ¡x,y!
Donde x e y son las coordenadas que fijan el punto (pixel) donde acaba la recta a trazar.
- Aplicación:** Dibuja en pantalla una recta cuyo punto inicial se define con el comando PLOT y el final con los parámetros de DRAW x,y.
- Ejemplo:** DRAW 100, 50
Trazará una recta partiendo del origen —ángulo inferior izquierdo— hasta el **pixel** situado 100 **pixels** a la derecha y 50 hacia arriba.
Ahora tenemos situado el «puntero» de **pixels**, justo donde acaba la recta anterior (100,50), de tal forma que si se aplica, p.e.:
DRAW 50, 100, la nueva recta comenzará en este punto.

ERASE

7 y SYMBOL SHIFT. Cursor E

ERASE

Sintaxis: ERASE ¡“m”; n.º μ d; “nombre del fichero”!

Aplicación: Borra de la μ c instalada en el μ d definido por el n.º μ d, el fichero denominado “nombre de fichero”.

Ejemplo: ERASE “m”; 1; “Preguntas”

En este caso el nombre del fichero es “Preguntas” y va a ser borrado de una microcinta instalada en el microdrive 1.

EXP

X. Cursor E

EXP

Sintaxis: EXP ¡expresión!

Aplicación: Calcula el valor del número e elevado a la potencia que determine **expresión**.

Ejemplo: 10 FOR x=0 TO 9
20 PRINT x, EXP x
30 NEXT x.

FLASH

V. SYMBOL SHIFT. Cursor E **FLASH**

Sintaxis: FLASH ¡expresión!

Donde expresión puede tomar los valores 0 ó 1.

FOR TO NEXT STEP

Aplicación: Permite la impresión en pantalla de una, varias o incluso la totalidad de las posiciones de caracter en forma parpadeante.

Este modo de impresión se consigue dando a **expresión x** el valor 1 y se elimina dando a **expresión x** el valor 0.

Ejemplo: 10 FOR x=8 TO 15
20 PRINT FLASH 1; x; FLASH 0, x 12
30 NEXT x
40 PAUSE 100: FLASH 1: CLS: PAUSE 100: FLASH 0:
CLS.

F. Cursor K **FOR**

F y SIMBOL SHIFT. Cursor K **TO**

N. Cursor K **NEXT**

D y SIMBOL SHIFT. Cursor K **STEP**

Sintaxis: FOR ¡variable = expresión 1! TO ¡expresión 2! STEP
expresión 3 .

Donde **variable** cambia de valor desde el indicado por **expresión 1** hasta el de **expresión 2**.

La cadencia de ese cambio de valor viene dada por la **expresión 3** que sigue a STEP. Si STEP no figura, la cadencia del cambio es la unidad.

Este cambio de valor se produce al llegar la lectura del programa a la instrucción:

NEXT variable.

Aplicación: Produce un bucle de lectura que arranca en la línea de programa donde figura la sentencia FOR...TO, hasta la línea en que está NEXT. Este bucle se repite, una y otra vez, hasta que la variable alcanza el valor dado por **expresión 2**.

Ejemplo: Con el siguiente programa se obtiene la tabla de multiplicar por 2.

```
10 FOR x=0 TO 9
20 LET A=2*x
30 PRINT A
40 NEXT x
```

0 y SYMBOL SHIFT. Cursor E

FORMAT

Sintaxis: FORMAT j"m"; n.º µd; "nombre" de la µc"

Aplicación: Formatea la microcinta instalada en el microdrive n.º µd, asignándola el "nombre de la µc", el cual puede estar compuesto por un máximo de 10 caracteres. El proceso de formateado borra cualquier contenido previo de la µc.

Ejemplo: FORMAT "m"; 2; "UTILIDADES"

FORMAT

Sintaxis:	FORMAT ¡“n”; n.º Spectrum !
Aplicación:	Asigna al Spectrum sobre el cual se aplica esta instrucción, el número dado en n.º Spectrum dentro de una red de comunicación local (network). Este número debe estar comprendido entre 1 y 64.
Ejemplo:	FORMAT “n”; 12
Sintaxis:	FORMAT ¡“b”; baudios !
Aplicación:	Acopla la transmisión/recepción de información en serie por el canal “b” a la velocidad definida por baudios .
Ejemplo:	FORMAT “b”; 300
Sintaxis:	FORMAT ¡“t”; baudios !
Aplicación:	Acopla la transmisión/recepción de información en serie por el canal “t” a la velocidad definida por baudios . Este tipo debe usarse para manejar textos.
Ejemplo:	FORMAT “t”; 1200 Las velocidades de transmisión admitidas son: 50, 110, 300, 600, 1200, 2400, 4800, 9600 y 19200.

GOSUB RETURN

	H. Cursor K	GOSUB
	Y. Cursor K	RETURN
Sintaxis:	GOSUB ¡expresión!	

La expresión puede ser sustituida por una variable numérica que haya sido definida con anterioridad.

Aplicación: Permite cambiar la secuencia de ejecución de un programa a la línea indicada por **expresión** (o, alternativamente, al valor que tenga la variable que se use), a partir de la cual seguirá hasta encontrar la instrucción RETURN, en ese momento la lectura continúa en la instrucción inmediata siguiente al GOSUB inicial.

Ejemplo:

```
10 LET T = 3
20 GOSUB T*100
30 GOSUB 200
40 STOP
200 PRINT "SEGUNDO"
210 RETURN
300 PRINT "PRIMERO"
310 RETURN
```

G. Cursor K

GOTO

GOTO

Sintaxis: GOTO ¡expresión!
Expresión puede ser sustituida por una variable que haya sido definida con anterioridad.

Aplicación: Permite cambiar la secuencia normal de ejecución de un programa a la línea indicada por **expresión** (o, alternativamente, al valor que tenga la variable que se use).

IF THEN

Ejemplo:

```
10 LET T=3
20 GOTO T*100
30 PRINT "SEGUNDO"
40 STOP
300 PRINT "PRIMERO"
310 GOTO 30
```

U. Cursor K

IF

G y SIMBOL SHIFT. Cursor K

THEN

Sintaxis: IF **comparación** THEN **sentencia**
Donde el resultado de **comparación** sólo puede ser **cierto** o **falso**.

Aplicación: Obliga a pasar la secuencia de lectura a **sentencia** si el resultado de la comparación es **cierto**, en otro caso continúa en la siguiente línea de programa.

Ejemplo:

```
10 INPUT A
20 IF A=3 THEN PRINT "TRES"
30 PRINT "NO ES TRES".
```

I SIMBOL SHIFT. Cursor E

IN

IN

Sintaxis: IN ¡expresión!

Donde expresión debe dar —o ser— un número comprendido entre 0 y 65535.

Aplicación: El Spectrum capta un byte de información desde el dispositivo periférico correspondiente a **expresión**.

Ejemplo: Ejecute el programa

```
10 LET * = 64510
```

```
20 PRINT AT 0,0; IN x
```

```
30 GOTO 20
```

Y teclee un —o varias— teclas correspondientes a Q, W, E, R, T.

X SIMBOL SHIFT. Cursor E

INK

INK

Sintaxis: INK ¡expresión!

Donde **expresión** puede tomar cualquier valor entre 0 y 7.

Aplicación: Permite dar un color de tinta determinado, de acuerdo con los disponibles en la primera fila de teclas, a los caracteres impresos en pantalla.

Ejemplo: PRINT INK 2; "Spectrum Plus".

INKEY\$

N. Cursor E

INKEY\$

Sintaxis: INKEY\$.

Aplicación: Capta el caracter de la tecla apretada en el momento de ser procesada esta instrucción, si no hubiera ninguna, se considera la cadena vacía.

Ejemplo: 10 PRINT "Apretar una tecla"
20 IF INKEY\$=" " THEN GOTO 20
50 PRINT "Tecla apretada".

En la línea 20 se produce un bucle sobre sí misma, ya que mientras no se apriete una tecla, el computador capta la cadena vacía y, consiguientemente, pasa la lectura a la línea 20.

INPUT

I. Cursor K

INPUT

Sintaxis: INPUT "Comentarios"; ¡**Variable!**.

Aplicación: Provoca una parada en el programa para, de esta forma, dar un contenido a la variable.

Si el «texto» ha sido escrito, éste aparecerá en la pantalla durante la interrupción y hasta que el dato solicitado haya sido tecleado y se apriete la tecla ENTER.

Si el «texto» no existe, sólo el **prompt** (con la expresión **prompt** se viene a indicar que el ordenador está prepara-

do para hacer su trabajo y, para ello, hace aparecer en pantalla un símbolo, al que por extensión se llama así) típico del ordenador aparecerá en pantalla, indicando que está a la espera de información. En este caso, el dato que se introduzca será asignado a la variable.

Esta variable puede ser numérica o de caracteres y, consiguientemente, el dato introducido tiene que estar en consonancia con el tipo de variable. Es decir, si la variable se ha considerado como numérica, no se aceptará una cadena de caracteres como entrada del INPUT.

Si a continuación de INPUT colocamos LINE (apretar 3 y SIMBOL SHIFT en modo E) seguido de una variable de caracteres, el ordenador quedará a la espera de una cadena de caracteres, pero en pantalla —al correr el programa— no aparecerán las comillas de la instrucción equivalente que sería: INPUT seguido de la variable de caracteres.

Ejemplo:

Escriba un programa que, una vez ejecutado, nos pida el número de «pasos» consumidos en una conferencia telefónica y, a continuación, el valor de cada «paso», para obtener finalmente el importe de la conferencia.

```
10 INPUT "¿Número de pasos?"; N
```

```
20 INPUT "Valor de un paso?"; P
```

```
30 PRINT "El importe de su conferencia es";
```

```
40 PRINT N*P.
```

Con las dos primeras líneas provocaremos, al ejecutar el programa, dos interrupciones para asignar valores a las variables N y P.

Con la tercera línea, ordenaremos la impresión del texto que figura tras el comando PRINT.

Con la cuarta línea, ordenamos la impresión del resultado de multiplicar el contenido actual de las variables N y P.

INT

R. Cursor E

INT

Sintaxis: INT ¡expresión!

Aplicación: Se calcula el menor de los números enteros del valor que da —o es— **expresión**.

Ejemplo: 10 PRINT INT 1,2 '
20 PRINT INT -1,2.

INVERSE

M. y SIMBOL SHIFT. Cursor E

INVERSE

Sintaxis: INVERSE ¡expresión!

Donde **expresión** puede tomar los valores 0 ó 1.

Aplicación: Permite intercambiar los colores de papel y tinta en una o varias de las posiciones de carácter de pantalla.

Ejemplo: 20 PRINT INVERSE 1; "INVERSO"; INVERSE 0;
"NORMAL".

LEN

K. Cursor E

LEN

Sintaxis: LEN ¡expresión!

Aplicación: Calcula el número de caracteres que componen la cadena definida en «**expresión**».

Ejemplo: PRINT LEN "PLUS".

LET

L. Cursor K

LET

Sintaxis: LET ¡variable = expresión!

Aplicación: Asigna a una variable el valor de una expresión. Recordemos previamente que una **variable** contiene un **valor** que puede cambiar a lo largo de un algoritmo, pero que permanece fijo mientras no haya una instrucción que así lo determine.

Existen dos clases de variables: **numéricas y de caracteres**.

Las variables numéricas se representan por cualquier conjunto de caracteres —a veces, por una sola letra seguida de un número— con la condición de que el primero sea una letra. Ejemplo: LET ABc = 45.

Las variables de caracteres se representan por una sola letra seguida del símbolo \$. De esta forma, el computador sabe que el contenido de la variable será un texto,

que deberá ser introducido entre comillas. Ejemplo: LET A\$ = "Cervantes".

Al referirnos a las variables, dijimos que son elementos capaces de contener un valor. **Una expresión**, por el contrario es **un valor**.

Nada impide que el valor de una expresión varíe de acuerdo con los valores de las variables que forman parte de la expresión.

Ejemplo: 10 LET A = 5
20 LET A\$ = "TOTAL"
30 INPUT B
40 PRINT A\$, A*B.

LINE

3 y SIMBOL SHIFT. Cursor E

LINE

Sintaxis: 1.^a Posibilidad INPUT LINE ¡**variable alfanumérica!**
2.^a Posibilidad SAVE "título" LINE ¡**expresión!**

Aplicación: Las dos posibilidades explicitadas más arriba se exponen en INPUT y SAVE.

LIST

K. Cursor K

LIST

Sintaxis: LIST ¡**expresión!**

Aplicación: Proporciona un listado en pantalla del programa que actualmente esté en memoria, y, a partir de la línea de programa que indique **expresión**. Si **expresión** no se escribe, entonces se obtiene el listado completo.

Ejemplo: Sobre cualquier programa introducido en el Spectrum pruebe lo indicado anteriormente.

V. Cursor E

LLIST

LLIST

Sintaxis: LLIST ¡**expresión**!

Aplicación: Se obtiene un listado, por impresora, del programa que actualmente hay en memoria.
Es válido todo lo expuesto en el comando LIST.

Z. Cursor E

LN

LN

Sintaxis: LN ¡**expresión**!

Aplicación: Calcula el logaritmo natural —en base e— del valor que da —o es— **expresión**.

Ejemplo: PRINT LN 1,1.

LOAD

I. Cursor K

LOAD

Sintaxis: LOAD " nombre "

Aplicación: Instruye al computador para que cargue en su memoria RAM información procedente de un soporte de memoria externo del tipo casete o microcinta.

LOAD " " carga el primer programa que encuentra en la cinta.

LOAD "**nombre**" carga exclusivamente aquel programa que haya sido grabado (salvado) en cinta con el nombre "**nombre**".

LOAD "**nombre**" DATA **denominación de la matriz**

() La denominación de la matriz se ajusta a lo indicado en la sentencia DIM, de forma que las matrices numéricas requerirán una letra para su denominación y las alfanuméricas una letra seguida del símbolo \$.

Con esta instrucción se carga la matriz que haya sido grabada (salvada) en cinta con el "**nombre**" y la reconocerá por la **denominación de la matriz**.

LOAD " " CODE carga el primer programa en código máquina que encuentra en la cinta.

LOAD "**nombre**" CODE **dirección de comienzo, longitud** cumple la misma función que la instrucción anterior, por tanto, esta instrucción se utiliza cuando la dirección de memoria desde donde queremos iniciar la carga del programa "**nombre**" en código máquina es distinta de aquella en la que fue grabada (salvada).

LOAD "**nombre**" SCREEN\$ carga directamente en el fichero de imágenes la **pantalla** que haya sido grabada (salvada) en cinta con el nombre "**nombre**". Si "**nombre**" no existe y sólo las comillas son utilizadas, carga la primera **pantalla** que encuentre en la cinta.

LOAD * "**m**"; **n.º microdrive**; "**nombre**" nos servirá para la carga procedente del **microdrive**, teniendo en cuenta que **n.º microdrive** debe indicar el microdrive que debe ser manipulado por el ordenador y que **todas** las referencias generales indicadas en la instrucción deben figurar, por lo demás es válido lo dicho anteriormente refiriéndonos al casete.

LOAD * "**m**"; **n.º microdrive**; "**nombre**" nos servirá para la carga procedente del **microdrive**, teniendo en cuenta que **n.º microdrive** debe indicar el microdrive que debe ser manipulado por el ordenador y que **todas** las referencias generales indicadas en la instrucción deben figurar, por lo demás es válido lo dicho anteriormente refiriéndonos al casete.

LOAD* "**n**"; **n.º Spectrum**, carga información en el Spectrum sobre el que se trabaja, procedente del Spectrum **n.º Spectrum**, dentro de una red de comunicación local (network).

Ejemplo:

Si se espera recibir un programa procedente del Spectrum, cuyo número de identificación dentro de la red local es 12 —asignado mediante el FORMAT correspondiente—, sólo se necesita la siguiente instrucción: LOAD*"**n**"; 12. LOAD*"**n**"; 0 carga la información que sea transmitida

con la instrucción `SAVE*"n"; 0`. Para averiguar de qué **n.º Spectrum** procede la información recibida, basta un `PEEK` sobre el byte 23759.
`LOAD*"b"` carga información en serie por el canal "b".

LPRINT

LPRINT

C. Cursor E

Sintaxis: LPRINT **expresión**

Aplicación: Produce la impresión por impresora de **expresión**, teniendo en cuenta las alternativas explicitadas más arriba.
Si **expresión** no se escribe, obtendremos línea en blanco.

Ejemplo: 10 LPRINT "Spectrum"
20 LPRINT
30 LPRINT "PLUS".

MERGE

MERGE

T_X y SIMBOL SHIFT. Cursor E

Sintaxis: MERGE "nombre "

Aplicación: Mezcla un programa en BASIC procedente de cinta (o micro-cinta) con el que actualmente esté en memoria, sustituyendo en éste aquellas líneas de programa que —procedentes del programa a fundir— coincidan.
`MERGE ""` Mezcla con el programa actualmente en memoria el primero que encuentre en la cinta.
`MERGE "nombre"` cumple la misma función, pero sólo con un programa que previamente haya sido salvado con el nombre "nombre".

MERGE * "m"; n.º **microdrive**; "nombre" donde n.º de **microdrive** indica el microdrive donde está instalada la microcinta que contiene el programa a mezclar. Todas las referencias indicadas debe aparecer.

MERGE*"n"; n.º **Spectrum** mezcla, en las condiciones ya explicitadas, el programa procedente del Spectrum, cuyo número dentro de la red local sea n.º **Spectrum**, con el que actualmente esté en memoria.

Sólo programas y sus variables pueden ser mezcladas.

6 y SIMBOL SHIFT. Cursor E

MOVE

MOVE

Sintaxis: MOVE # r1 TO # r2

Donde **r1** es el **rumbo** (stream) actual de la información y **r2** es el **rumbo** al cual queremos **mover**.

Aplicación: Permite recibir información procedente del **rumbo r1** y enviarla por el **rumbo r2**.

Sintaxis MOVE "m"; n.º μ d; "nombre fichero" TO r
nombre fichero debe corresponder a un fichero de datos. De otra forma dará error.

Aplicación Mueve un fichero de datos desde la μ c instalada en el μ d determinado por μ d al **rumbo r**.

Sintaxis: MOVE "m"; n.º μ d; "nobre fichero 1" TO "m"; n.º μ d;
"nombre fichero 1" a la nueva denominación "nobre fichero 2".

NEW

A. Cursor K

NEW

Sintaxis: NEW

Aplicación: Produce un borrado de la pantalla, programa y variables.

Ejemplo: Aplicar un NEW sobre cualquier programa y solicitar la impresión de alguna variable y pedir un listado del mismo.

NEXT

N. Cursor K

NEXT

ver FOR/TO....NEXT

NOT

S y SIMBOL SHIFT. Cursor K

NOT

ver Apéndice «FUNCIONES LOGICAS»

OPEN

4 y SIMBOL SHIFT. Cursor E

OPEN

Sintaxis: OPEN # r; "m"; n.º µd; "nombre fichero"
r debe estar comprendido entre 0 y 15.

Aplicación Abre al **rumbo** (stream) r, el fichero denominado "nombre fichero" situado en el microdrive n.º µd.

Sintaxis: OPEN # r; "n"; n.º Spectrum
r debe estar comprendido entre 0 y 15
n.º Spectrum debe estar comprendido entre 0 y 64.

Aplicación Abre el **rumbo** (stream) *r*, entre el Spectrum sobre el que se trabaja y el Spectrum cuyo número en la red local es **n.º Spectrum**.

Al abrir un **rumbo** dentro de la red local, este queda disponible para enviar o recibir datos, pero la primera acción (enviar o recibir) determina que ese **rumbo** sólo pueda enviar o recibir.

Sintaxis: OPEN # *r*; "b"
OPEN # *r*; "t"

r debe estar comprendido entre 0 y 15

"b" = canal b "t" = canal t

Aplicación Abre el μ (stream) *r* al canal "b" o "t" —según el tipo— para recibir o transmitir.

U y SIMBOL SHIFT. Cursor K

OR

ver Apéndice «FUNCIONES LOGICAS»

OR

O SIMBOL SHIFT. Cursor E

OUT

Sintaxis: OUT ¡expresión A, expresión B!

Donde **expresión A** debe dar —o ser— un número comprendido entre 0 y 65535.

Aplicación: El Spectrum emite un byte de valor **expresión B** al dispositivo periférico externo correspondiente al puerto definido por **expresión A**.

OUT

Ejemplo: 10 FOR x=0 TO 255
20 OUT 254, x
30 NEXT x.

OVER

N y SIMBOL SHIFT. Cursor E

OVER

Sintaxis: OVER ¡expresión!
Expresión sólo puede dar —o ser— 0 ó 1.

Aplicación: Permite sobreimprimir en la misma posición de carácter.

Ejemplo: 10 PRINT AT 0, 0; " "
20 PRINT A 0, 0; OVER 1; "/".

PAPER

C y SIMBOL SHIFT. Cursor E

PAPER

Sintaxis: PAPER ¡expresión!
Expresión sólo puede dar —o ser— un número entre 0 y 7.

Aplicación: Permite dar un color de papel determinado, de acuerdo con los disponibles en la primera fila del teclado, al fondo de los caracteres impresos en pantalla.

Ejemplo: PRINT PAPER 6; "Spectrum".

PAUSE

M. Cursor K

PAUSE

Sintaxis: PAUSE ¡expresión!

Aplicación: Produce una interrupción del proceso durante el tiempo definido en **expresión** y a razón de 50 unidades por segundo.
PAUSE 0, produce una interrupción hasta que se aprieta cualquier tecla.

Ejemplo: 10 PRINT "Spectrum"
20 PAUSE 50*2
30 PRINT "PLUS".

O. Cursor E

PEEK

PEEK

Sintaxis: PEEK ¡expresión!
Expresión debe dar —o ser— un número comprendido entre 0 y 65535.

Aplicación: Permite averiguar el contenido de la posición de memoria cuya dirección viene dada por **expresión**.

Ejemplo: PRINT PEEK 0.

Q. Cursor K

PLOT

PLOT

Sintaxis: PLOT ¡expresión A, expresión B!.
Donde **expresión A** debe dar —o ser— un número comprendido entre 0 y 255, y **expresión B** entre 0 y 175.

Aplicación: Imprime en pantalla un punto (pixel) del color actual de tinta posicionado de acuerdo con la abscisa y ordenada definida por **expresión A** y **expresión B**, respectivamente.

Ejemplo: PLOT 200,20

POINT

8 y SIMBOL SHIFT. Cursor E

POINT

- Sintaxis:** POINT ¡(expresión A, expresión B)!
- Donde **expresión A** y **expresión B** tienen el mismo significado que en PLOT.
- Aplicación** Determina si un determinado punto (**pixel**) de la pantalla está encendido (color INK) o apagado (color PAPER). En el primer caso, el resultado de aplicar POINT, será 1, de otra forma se obtendrá 0.
- Ejemplo:** Si previamente imprimimos un punto de la pantalla, p. e. PLOT 200, 20, y a continuación aplicamos: PRINT POINT (200,20) obtendremos un 1, para indicarnos que tal punto está encendido o, dicho de otro modo, está afectado por el color de tinta actual.

POKE

O. Cursor K

POKE.

- Sintaxis:** POKE ¡expresión A, expresión B!.
- Aplicación:** Permite introducir el byte definido por **expresión B** en la posición de memoria cuya dirección viene dada por **expresión A**.
- Ejemplo:** POKE 30000,8.

Sintáxis: PRINT ¡expresión!.

Aplicación: Ordena la impresión en pantalla de la expresión o expresiones que figuran, optativamente, en las expresiones.

- * Si las expresiones son omitidas, una línea de pantalla salta en blanco.
- ** Si las expresiones son incluidas y estas expresiones son cadenas de caracteres, debemos escribirlas entrecomilladas.
- *** Si las expresiones son incluidas y estas expresiones son matemáticas, aparecerán en pantalla sus valores. Recuerde que las expresiones matemáticas no se escriben entre comillas.

La posición de impresión de cada expresión viene determinada por los símbolos que separan una expresión de otra:

- Un punto y coma (;) hace que la siguiente expresión se imprima justo a continuación de la inmediata anterior.
- Una coma (,) obliga a imprimir a partir de la siguiente mitad de la pantalla o al comienzo de la línea siguiente si la otra mitad ya se ocupó.
- Un apóstrofe imprime la expresión que la sigue al comienzo de la siguiente línea.

Si las expresiones finalizan con una (,) o un punto y coma (;) la siguiente instrucción PRINT, que pudiera aparecer a lo largo del programa, comenzaría la impresión de

RANDO- MIZE

sus expresiones conforme al criterio expuesto más arriba. En caso de no acabar con una coma o un punto y coma, la siguiente instrucción PRINT comenzaría la impresión de sus expresiones en la siguiente línea de pantalla que esté libre.

RANDOMIZE

T. Cursor K

Sintáxis: RANDOMIZE **expresión** .

Aplicación: Inicializa el generador de números aleatorios con la **expresión**, con lo que la secuencia de números RND comenzará con valores distintos.

El valor de la expresión numérica debe estar comprendido entre 0 y 65535, con lo que las secuencias RND comenzarán en diferentes lugares.

RANDOMIZE 0 obliga a RND a producir números lo más aleatorios posible.

En algunos desarrollos técnicos y/o científicos, conviene simular fenómenos aleatorios controlados, de forma tal que, variando algunos parámetros, se repita el proceso con los mismos valores.

Si **expresión** se omite, produce el mismo efecto que RANDOMIZE 0.

Ejemplo: Este programa que produce números lo más aleatorios posible.

```
10 RANDOMIZE 0.  
20 FOR x = 1 TO 20.  
30 PRINT RND.  
40 NEXT x.
```

Córralo varias veces y comprobará que, a diferencia del ejemplo expuesto en la ficha RND, los resultados son diferentes en cada ocasión.

A. Cursor E

READ

Ver comando DATA/READ

READ

E. Cursor K

REM

Sintaxis: REM **expresión**

Aplicación: Permite introducir los comentarios dados en “**expresión**” dentro de cualquier línea de programa sin que interrumpa ni intervenga en el proceso general.

Ejemplo: 10 REM “cálculo de cargas”.

REM

RESTORE

S. Cursor E

RESTORE

Sintaxis: RESTORE **expresión**

Aplicación: Con esta sentencia se permite comenzar la lectura de las constantes desde la primera del DATA situado en el número de línea dada por **expresión**.

Si el número de línea no se especifica, la lectura de constantes se iniciará en la primera del primer DATA del programa.

Después de un programa lee una sentencia RESTORE, la primera sentencia READ que ejecute el programa accederá a la primera constante, según el criterio expuesto más arriba.

Teclear RESTORE y el **número de línea** seguido de ENTER, para volver a leer las constantes que hay en el **DATA de la línea del número dado**. Teclear RESTORE, seguido de ENTER, para volver a leer las constantes desde **el primer DATA del programa**.

Ejemplos:

```
10 FOR X= 1 TO 5.  
20 READ V: PRINT V  
30 RESTORE  
40 NEXT X  
100 DATA 3
```

La variable V es leída cinco veces en la línea 20 aunque tiene una sola constante en el DATA de la línea 100. Esto es posible gracias al RESTORE de la línea 30, que obliga a comenzar la lectura en el primer valor del primer DATA.

```

10 FOR X= 1 TO 10
20 READ V$: PRINT V$
30 RESTORE
40 NEXT X
100 DATA "caballo", "jinete"

```

La variable V\$ es leída diez veces en la línea 20 tomando el primer valor del DATA de la línea 100.

Y. Cursor K

RETURN

Ver comando GOSUB/RETURN

RETURN

T. Cursor E

RND

Sintáxis: RND

Aplicación: Cada vez que se utiliza RND, se obtiene un valor diferente y aleatorio igual o mayor que 0 y menor que 1. Esta función da la misma secuencia de valores cada vez que se ejecuta el programa que la contiene, tecleando RUN/ENTER.

Ejemplo:

```

10 FOR X=0 TO 20
20 PRINT RND
30 NEXT X

```

RND

RUN

R. Cursor K

RUN

Sintaxis: RUN

Aplicación: Obliga a la ejecución del programa que acutalmente esté en memoria.

Ejemplo: Como comando directo basta con teclear RUN seguido de ENTER para ejecutar el programa que esté en memoria. Como instrucción en una línea de programa produce el mismo efecto.

SAVE

S. Cursor K

SAVE

Sintaxis: SAVEj"nombre"!

Aplicación: Transfiere la información contenida en la memoria RAM de un computador a un soporte de memoria externo, salvándola o guardándola para ser utilizada posteriormente, transfiriéndola nuevamente al computador mediante LOAD.

Es necesario recordar que la memoria RAM no es permanente y, al desconectar el computador de la corriente, toda la información que contiene se pierde. Gracias a SAVE y a la función que cumple, la información queda salvada, bien sobre cinta magnética, bien sobre disco.

SAVE "nombre" graba (salva) en cinta (casete) el programa que actualmente esté en la memoria RAM con el nombre "nombre".

SAVE "nombre" DATA **denominación de la matriz** ().

La denominación de la matriz se ajusta a lo indicado en la sentencia DIM, de forma que las matrices numéricas requerirán una letra para su denominación y las alfanuméricas una letra seguida del símbolo \$. Con esta instrucción se graba o salva — con el nombre "**nombre**" — la matriz que actualmente reconozca el computador con la denominación **denominación de la matriz**.

SAVE "nombre" LINE **número de línea** cumple la misma función que la anterior pero con la diferencia de que al ser cargado en el computador con un LOAD se autoejecuta a partir de la línea indicada en **número de línea**.

SAVE "nombre" SCREENS\$ graba o salva en cinta todo el contenido que actualmente esté en pantalla.

SAVE "nombre" CODE **dirección de comienzo, longitud** graba o salva en cinta — con el nombre «**nombre**» — el programa en código máquina — o bytes simplemente — que actualmente está en memoria a partir de la dirección dada por **dirección de comienzo** y con una longitud — medida en bytes — dada por **longitud**.

SAVE * "m"; **número microdrive; "nombre"** graba o salva en microcinta — con las mismas especificaciones dadas para el casete —, teniendo en cuenta que **número de microdrive** debe indicar el microdrive dónde está instalada la microcinta dónde se ha de reproducir la grabación y que todas las referencias generales indicadas más arriba deben figurar.

SAVE * "n"; **n.º Spectrum**, envía el programa que ac-

tualmente está en la memoria del Spectrum sobre el que se trabaja, al Spectrum cuyo número en la red es **n.º Spectrum**.

SAVE * "b" SCREEN\$, envía el fichero de imágenes (pantalla) que actualmente tiene el Spectrum sobre el que se trabaja al canal "b".

SCREEN\$

K. y SYMBOL SHIFT. Cursor E **SCREEN\$**

Sintáxis: Debe ir precedido de **SAVE** o **LOAD** para que sea operativa, siendo la instrucción completa:

SAVE ¡"expresión"! SCREEN\$

LOAD¡"expresión"! SCREEN\$

Donde **"expresión"** da nombre a la pantalla a salvar o cargar.

Aplicación: Con el comando **SAVE**, permite salvar la pantalla que a la sazón figure, con el nombre dado por **"expresión"**. Con el comando **LOAD**, permite cargar la pantalla que previamente se haya salvado con el nombre **"expresión"**. En el caso de que al cargar una pantalla no se utilice **expresión**, entonces el Spectrum tomará la primera que se encuentre en la cinta de la casete.

Ejemplo: Salvar cualquier pantalla para recuperarla posteriormente siguiendo las indicaciones anteriores.

Sintaxis: **SCREEN\$ ¡expresión A, expresión B)!**
Siendo ambas expresiones valores representativos de una

	línea (0-23) y una columna (0-31) de pantalla respectivamente.
Aplicación	Toma el carácter que actualmente esté en pantalla en la posición determinada por expresión A y expresión B .
Ejemplo:	10 PRINT AT 10,10; "p" 20 PRINT SCREEN\$ (10,10)

F. Cursor E

SGN

Sintaxis:	SGN ¡(expresión)!
Aplicación:	Devuelve -1, 0 ó 1 según que el valor de expresión sea negativo, nulo o positivo.
Ejemplo:	PRINT SGN (2-7)

SGN

Q. Cursor E

SIN

Sintaxis:	SIN ¡expresión!
Aplicación:	Calcula el seno de un ángulo medido en radianes, por tanto expresión debe dar -o ser- un valor comprendido entre 0 y 2 Pi.
Ejemplo:	PRINT SIN 0

SIN

SQR

H. Cursor E		SQR
Sintáxis:	SQR ¡(expresión)!	
Aplicación:	Calcula el valor de la raíz cuadrada de la expresión .	
Ejemplo:	10 PRINT SQR 4 20 PRINT SQR (6-2) 30 PRINT SQR 6-2	

STEP

D y SYBOL SHIFT. Cursor K		STEP
<i>Ver comando FOR/TO....NEXT</i>		

STOP

A y SYMBOL SHIFT. Cursor K		STOP
Sintáxis:	STOP.	
Aplicación:	Produce una interrupción en el proceso emitiendo un mensaje de «Sentencia STOP». (STOP Statement.) Se puede seguir el proceso normal con el comando directo CONT.	
Ejemplo:	Ver el aplicado al comando CONT.	

STR\$

Y. Cursor K		STR\$
Sintáxis:	STR\$ ¡expresión!	

Aplicación: Convierte la **expresión** numérica en la alfanumérica correspondiente, convirtiendo cada dígito del número dado por **expresión** en una cadena de caracteres idéntica.

Ejemplo: 10 LET A \$ = TR\$ 1234.5
20 PRINT A \$

TAB

P. Cursor E

TAB

Sintaxis: Este comando debe ir precedido en la sentencia por PRINT, de esta forma:
PRINT TAB **¡(expresión)!**; *expresión numérica o alfanumérica.*

Donde **expresión** debe dar —o ser— un número comprendido entre 0 y 31 ya que determina la columna donde debe iniciarse la expresión.

Aplicación: Fija la columna, dentro de la línea actual, dónde debe iniciarse la impresión de la expresión —numérica o alfanumérica— que vaya a continuación del punto y coma.

Ejemplo: Teclear en modo directo:
PRINT TAB 8; "TOTAL"; TAB 20; 250
Un solo comando PRINT al comienzo de la sentencia, como puede apreciarse, es válido para los sucesivos TAB que estén encadenados mediante los correspondientes puntos y comas.

TAN

E. Cursor E

TAN

Sintáxis: TAN ¡expresión!

Aplicación: Calcula la tangente de un ángulo medido en radianes, por tanto **expresión** debe dar —o ser— un valor comprendido entre 0 y 2PI.

Ejemplo: PRINT TAN 0

THEN

G y SYMBOL SHIFT. Cursor K

THEN

Ver comando IF/THEN

TO

F y SYMBOL SHIFT. Cursor K

TO

*Ver comandos FOR /TO.... NEXT
Ver apéndice «TROCEADO DE CADENAS»*

USR

L. Cursor E

USR

Sintaxis: USR ¡expresión!

Expresión debe dar —o ser— un número comprendido entre 0 y 65535.

Este comando debe ir precedido de otro del tipo RANDOMIZE, PRINT. etc.

Aplicación

Dirige la lectura del programa en BASIC a una subrutina en código máquina cuya dirección de comienzo viene dada por **expresión**, cuando se produce el retorno al BASIC, se obtiene el valor del contenido del par de registros **bc**, para lo cual la llamada a la subrutina debería ser PRINT USR **expresión**.

También puede ser usada para generar caracteres gráficos por el usuario, debiendo, en este caso, ir sucedido del correspondiente número.

J. Cursor E

VAL

Sintaxis: VAL ¡**expresión**!

Aplicación: Convierte el resultado de la **expresión** alfanumérica en su resultado numérico.

Ejemplo: 10 LET a=VAL "212"
20 PRINT a

VAL

R y SYMBOL SHIFT. Cursor E

VERIFY

VERIFY

Sintáxis: VERIFY " nombre "

Aplicación: Comprueba que lo grabado o salvado en la cinta coincide exactamente con el contenido en memoria y que — previamente— ha sido transferido del computador al casete (o microcinta).

VERIFY " " compara el programa que actualmente está en memoria con el primer programa que se encuentre en la cinta.

VERIFY "nombre" cumple la misma función que la instrucción anterior pero no inicia la comparación hasta que encuentra un programa que haya sido salvado con el nombre «nombre».

VERIFY "nombre" DATA **denominación de la matriz**, compara la matriz denominada **denominación de la matriz** y que actualmente reconoce así el computador con la matriz que haya sido grabada en la cinta con el nombre "nombre".

VERIFY "nombre" CODE **dirección de comienzo, longitud** compara los bytes grabados en cinta previamente con el nombre "nombre" a partir de la dirección **dirección de comienzo** y a lo largo de una longitud — medida en bytes— dada por **longitud**.

VERIFY * "m"; **número microdrive**; "nombre" donde **número microdrive** indica el microdrive donde está instalada la microcinta. Todas las referencias indicadas deben aparecer, por lo demás es válido lo dicho anteriormente con respecto al casete.

BASIC: Curso acelerado para principiantes

En este curso vamos a estudiar aquellos comandos que implican el mayor esfuerzo intelectual, por parte del aprendiz, para adaptarse al modo de razonar que le exige el computador.

En el planteamiento didáctico, he asumido que el lector no tiene conocimiento alguno de la lengua inglesa y, consiguientemente, lo primero que haremos al enfrentarnos con cada instrucción será adaptarla a la mentalidad de un hispano-parlante.

También supongo que sabe manejar el teclado (Ver apéndice «Manejando el teclado») y, puesto que de un curso acelerado se trata, pasemos ya a iniciarnos en la programación BASIC.

Una nota aclaratoria previa se hace imprescindible: en todo lo que sigue, el lector deberá tener su equipo correctamente conectado, estar situado enfrente de él y probar todas y cada una de las indicaciones que se sugieren, así como los ejercicios.

Haga cuantas pruebas crea conveniente, vaya tan rápido en la lectura como quiera y vuelva atrás sin el menor reparo.

PRINT

Print significa «imprimir» y, por tanto, este comando nos transmite la idea de ordenar al computador que imprima en pantalla.

Como ya se explicó en CONCEPTOS PREVIOS, una cadena de caracteres es distinguida de un número o una expresión numérica, porque aquélla va siempre entrecomillada. Así, si nosotros le ordenamos en modo directo al Spectrum que nos escriba la cadena "*Spectrum*", debemos proceder de la siguiente forma:

PRINT "*Spectrum*"; seguido de ENTER

con lo cual la palabra *Spectrum* saldrá impresa en pantalla.

Si por el contrario queremos que nos imprima el resultado de la expresión numérica $2 + 8$, solo tendremos que teclear:

PRINT $2 + 8$, seguido de ENTER e inmediatamente obtendremos un 10 en la pantalla.

En modo programación, podríamos utilizar este comando escribiendo líneas de programa numeradas de forma tal que las instrucciones se ejecuten en el orden que nos interese. Veamos un ejemplo.

Supongamos que se desea obtener en pantalla la siguiente secuencia de impresiones:

Dos más dos son 4

Y cuatro por dos son 8

Esto se podría obtener con este listado

10 PRINT "Dos más dos son"

20 PRINT 4

30 PRINT "Y cuatro por dos son"

40 PRINT 8

Para «ejecutar» o «correr» este programa tendremos que apretar RUN y ENTER, con lo cual el ordenador comenzará a interpretar la primera línea de programa —la 10— que le ordena imprimir *Dos mas dos son* y eso, exclusivamente, es lo que hace. En la 20 imprime el número 4. Al llegar a la línea 30 imprimirá el texto que figura entrecomillado, para acabar con la instrucción de la línea 40 que le ordena imprimir el número 8.

Esto, y no otra cosa, es lo que hace el comando PRINT.

Ahora bien, el BASIC dota a este comando de una serie de posibilidades de impresión que aquí no se tocan, pero que pueden ser estudiadas en la ficha PRINT de la GUIA DE COMANDOS.

Let significa «dejar» en el sentido de dejar una variable igual a un valor y, por tanto, al usar LET estamos ordenando al computador que asigne un determinado valor a una variable.

LET

Si, en modo directo, le damos al Spectrum la orden:

LET X = 25, seguido de ENTER

habremos asimilado, en memoria, la variable x al valor 25, de tal forma que si le pedimos que nos imprima el valor x —con un PRINT— obtendremos el número 25.

En el caso de querer igualar una variable de caracteres a una cadena de caracteres, procederemos de la siguiente forma:

```
LET X$ = "Spectrum", seguido de ENTER
```

a partir de este momento la variable X\$ contiene la cadena de caracteres *Spectrum*. Podemos comprobar esto mediante un PRINT X\$, lo cual nos dará en pantalla la palabra Spectrum.

Apoyándonos en los dos comandos estudiados hasta el momento (PRINT y LET), desarrollaremos un ejemplo:

"Primer valor" y "Segundo valor" y a dos variables numéricas los números 10 y 20.

```
10 LET A$ = "Primer valor"  
20 LET B$ = "Segundo valor"  
30 LET X = 10  
40 LET Y = 20  
50 PRINT A$, X  
60 PRINT B$, Y
```

Con las cuatro primeras líneas conseguimos los objetivos propuestos, en las dos últimas exigimos la impresión de estas variables con el formato que da la coma.

Esta es la función que cumple el comando LET, para dominarlo en profundidad hay que revisar detalladamente todo lo expuesto en CONCEPTOS PREVIOS sobre variables y, en la GUIA DE COMANDOS, la ficha LET.

INPUT

Input es equivalente a «entrada», lo que, en términos de programación, nos viene a transmitir la idea de «poner dentro» de la memoria del computador un valor, ya sea numérico o alfanumérico, procedente del teclado.

Cuando el ordenador, a lo largo del programa, se encuentra con una instrucción INPUT para —detiene— todo proceso hasta que, a través del teclado, recibe la entrada que espera, cosa que sucede al apretar la tecla ENTER.

Como es de suponer, estas «entradas» pueden ser numéricas o alfanuméricas, las cuales se distinguen, como es usual, por estar —o no— entre comillas según sean de caracteres o numéricas.

En el caso de pretender una entrada numérica, el siguiente ejemplo servirá de aclaración:

```
10 PRINT "Dame un número"  
20 INPUT N  
30 PRINT "El número que me has dado es"  
40 PRINT N
```

Al llegar el ordenador en la lectura del programa a la línea 20, se parará hasta que se le de por teclado un número, p. e. el 8, seguido de ENTER, en ese momento el proceso sigue, con lo cual se imprimirá en pantalla el texto que figura en la línea 30 y, a continuación, obedecerá la instrucción que figura en la línea 40, según la cual debe imprimir el valor de la variable N, que, a su vez, ha recibido el número 8 —en el caso del ejemplo sugerido— en la entrada (INPUT) de la línea 20 y, a través del teclado.

Para las entradas de variables alfanuméricas, la situación es similar.

Veamos este ejemplo:

```
10 PRINT "Dime tu nombre"  
20 INPUT N$  
PRINT "Hola"; N$
```

En la línea 20 el proceso se detendrá hasta que se escriba una cadena de caracteres seguida de ENTERM p. e. *Spectrum*, con lo cual el proceso continúa en la línea 30, con lo cual se imprimirá la expresión *Hola*, seguida del contenido de las variables N\$ que, en nuestro caso, es Spectrum.

Las demás posibilidades que ofrece este comando se pueden ver en la ficha INPUT de la GUIA DE COMANDOS.

GOTO

GO TO significa «ir a», lo que interpretado por el computador en un programa quiere decir *vete a la línea de programa número tal*.

Como sabemos el ordenador «lee» las líneas del programa, una tras otra, en el orden que le indican sus números, pero si en una de ellas se encuentra con la instrucción GOTO seguida de un número, inmediatamente saltará esta lectura secuencial de líneas de programa, a la línea cuyo número le indica la instrucción GOTO en cuestión.

Un ejemplo dejará esto claro:

```
10 PRINT "COMIENZO"  
20 GOTO 40  
30 PRINT "Esta línea no sirve para nada"  
40 PRINT "FINAL"  
50 GOTO 10
```

La línea 30 jamás será leída, ya que en la línea 20 el GOTO que figura en ella, manda la lectura a la línea 40, saltándose la 30.

En la línea 50 se envía la lectura a la línea 10 con lo cual el proceso comienza otra vez.

«Si apruebas *entonces* te regalo un libro». El «Si» y el «*entonces*» que figuran en la sentencia anterior son equivalentes al IF y al THEN que vamos a estudiar a continuación y que, en definitiva, representan la forma de tomar decisiones de un computador, a través de una instrucción de un programa.

IF...THEN

Supongamos que nos interesa cambiar la secuencia de lectura de un programa en el caso, y sólo en el caso, de que la variable numérica X sea igual a 9. Para ello escribiríamos un programa de este estilo:

```
10 INPUT "dime un número"; X  
20 IF X=9 THEN GOTO 100  
30 PRINT "El número es distinto de 9"  
40 STOP
```



```
....  
100 PRINT "El número es 9"  
110 GOTO 10
```

En la línea 10 se utiliza el comando INPUT con una de sus facilidades de uso —según se estudia en su ficha— dándose un valor, por el teclado, a la variable X. En la línea 20 se le ordena al computador que «sí» la variable X es igual a 9 *entonces* cambie la secuencia normal de lectura a la línea 100.

De la misma forma que en el ejemplo anterior, hemos utilizado el comando GOTO tras THEN, podríamos utilizar, si fuera de nuestro interés, otro comando BASIC. En el siguiente ejemplos dedicado a IF/THEN, pero manejando variables de caracteres.

En el caso de que quisiéramos p. e. continuar la ejecución de un programa en función de una cadena de caracteres introducida por el teclado, este podría ser un ejemplo válido:

```
10 INPUT "Quiere continuar"; R$  
20 IF R$ = "SI" THEN GOTO 40  
30 STOP  
40 ...  
50 ...  
60 ...
```

Si el resultado de la entrada de la línea 10, fuera SI entonces la lectura del programa pasa a la línea 30 con lo cual el ordenador detiene la ejecución del programa como consecuencia del comando STOP que figura en ella.

Este tema se puede ampliar en la ficha correspondiente de la GUIA DE COMANDOS y en el APENDICE dedicado a las fuciones lógicas aplicadas.

Bucles de repetición controlada

Este tipo de bucles se repiten DESDE (FOR), un valor predeterminado de una variable numérica, HASTA (TO) que la citada variable alcance el valor que se indica tras TO, ejecutando todas las instrucciones que están a continuación. Cuando la lectura del programa alcanza la instrucción NEXT, entonces toma el siguiente valor de la variable inicial y se repite el proceso, una y otra vez, hasta alcanzar la variable en cuestión el valor citado.

FOR...TO
...
NEXT

Este bucle sería muy práctico para obtener en pantalla la tabla de multiplicar por 5:

```
10 FOR X=0 TO 9
20 PRINT X*5
30 NEXT X
40 STOP
```

El proceso de ejecución y lectura de este bucle FOR...TO/NEXT es muy sencillo.

En la línea número 10 se determina que, desde el valor 0 de la variable X y hasta el valor 9, se repita el proceso que sigue hasta donde está la instrucción NEXT. En el caso del ejemplo, el proceso que sigue se limita a una instrucción —PRINT X*5— en la cual se le exige al ordenador que imprima el resultado de multiplicar el valor que actualmente tenga la variable X por cin-

co. (En el primer ciclo será 0, en el siguiente 1 y así sucesivamente). Al encontrar la instrucción NEXT X, la variable X toma el siguiente valor, comprendido entre 0 y 9, secuencialmente y la lectura del programa retorna a la línea 20 para repetir el proceso de impresión, con el valor que acaba de tomar la variable X.

Cuando X sobrepasa el valor 9, la lectura continúa en la línea 40.

Si deseamos que la cadencia con que cambia de valor la variable sea distinta de la unidad, utilizaremos la instrucción STEP de la siguiente forma:

```
10 PRINT X=0 TO 9 STEP 2
20 PRINT X*5
30 NEXT X
40 STOP
```

Este programa es igual al anterior, excepto por el hecho de haber introducido un STEP 2 al final de la línea 10, con lo cual la variable X tomará los valores de 0 a 9, de dos en dos, es decir: 0, 2, 4, 6 y 8.

Corra este programa y compare los resultados con el primer ejemplo.

En la ficha correspondiente de la GUIA DE COMANDOS, encontrará una generalización del uso de estos comandos.

GOSUB/ RETURN

De la misma forma que GOTO obligaba a cambiar la secuencia de lectura a la línea cuyo número completa esta instrucción, GOSUB cambia la lectura normal del programa a la línea cuyo número figura tras ella, pero con la diferencia de que la lectura del programa retornará a la línea siguiente a aquella en que figure el GOSUB, justo en el momento en que aparezca la instrucción de volver: RETURN.

En otras palabras, GOSUB significa «ir a subrutina» y RETURN significa «volver de la subrutina».

Un ejemplo despejará todas las dudas:

```
10 INPUT "Dime tu nombre"; N$  
20 IF N$ = "Pepe" THEN GOSUB 100: STOP  
30 PRINT "No te llamas Pepe": CLS  
40 GOTO 10  
100 PRINT "Hola Pepe"  
110 RETURN
```

Como podemos observar, siempre que el nombre que escribamos por teclado sea *Pepe*, el ordenador, gracias a la instrucción de la línea 20, dirigirá la lectura a la subrutina que empieza en la línea 100 y, que acaba en la línea donde figura la instrucción RETURN que, en nuestro caso, es la 110.

Un repaso a la ficha dedicada a estos comandos en la GUIA DE COMANDOS le dará una visión completa del tema.

Resumen

Esta rápida introducción al lenguaje BASIC, le permitirá desarrollar sus conocimientos de programación, para lo cual nos hemos apoyado en aquellas instrucciones cuyo significado es esencial conocer y que son las estudiadas en este CURSO ACELERADO DE PROGRAMACION.

Si al llegar a este punto, Ud. está interesado en profundizar en este área, le recomendamos que elija aquellas «palabras» que le interesen a sus fines, dentro de la GUIA DE COMANDOS y haga cuantos ejemplos y supuestos le parezca conveniente, combinándolas con las instrucciones analizadas en este curso.

En este sentido, debemos resaltar que, si bien es interesante dominar el mayor vocabulario BASIC posible, no es imprescindible aprender de memoria todas ellas. Véase como ejemplo, las funciones trigonométricas, las cuales, salvo casos muy especiales, no suelen ser de interés.

Bibliografía:

Para profundizar en aquellos temas que necesita dominar el usuario del Spectrum, les sugerimos los siguientes títulos:

- **Como programar su Spectrum.**
Curso detallado de programación BASIC para principiantes.
- **Los colores y los gráficos del Spectrum.**
Contiene los conocimientos necesarios para introducirse en el mundo de la generación de gráficos y sirve como paso previo para acceder a conocimientos más profundos.
- **Spectrum: Iniciación al código máquina.**
Este libro incorpora un ensamblador en casete y se hace imprescindible para dar el primer paso dentro del código máquina, sin frustraciones.
- **Los trucos del Spectrum.**
Un resumen de tretas que normalmente, no están al alcance de los usuarios del Spectrum.
- **Spectrum: Plus Ultra.**
Este libro es una verdadera enciclopedia del Spectrum, ya que contiene a todos los libros anteriores y mucho más.

APENDICE 1

Manejo del teclado

Denominamos *cursor* al caracter que parpadea en la pantalla, y cuyo fin es indicarnos donde se producirá la siguiente impresión.

El cursor se puede desplazar, a lo largo de una línea de programa o instrucción, mediante las flechas situadas a la izquierda de la barra espaciadora.

En el Spectrum, este cursor tiene además otro cometido. Según el caracter que figure en el cursor, nos estará indicando el modo en que va a trabajar el teclado.

Así, si el cursor es un *K* nos está indicando que, al apretar una tecla, va a imprimir en pantalla la palabra del BASIC que figura sobre la superficie de la misma y en la parte superior.

Las palabras del BASIC citadas también son conocidas como KEYWORDS.

Cuando el cursor es una *L*, cualquier tecla que se apriete, implicará la impresión en pantalla de la letra o número que aparece prominentemente en ella.

Si quisiéramos conseguir palabras BASIC que están sobre la superficie de la tecla e inmediatamente encima de la letra que está en ella, tendríamos que apretar, simultáneamente, la citada tecla y una de las teclas donde figura SYMBOL SHIFT. De igual forma se obtienen los símbolos que figuran en la superficie de la teclas.

Fuera de lo que es propiamente la superficie de la tecla, vemos dos palabras BASIC —en algunos casos hay símbolos— que se obtienen pasando el cursor al modo *E*.

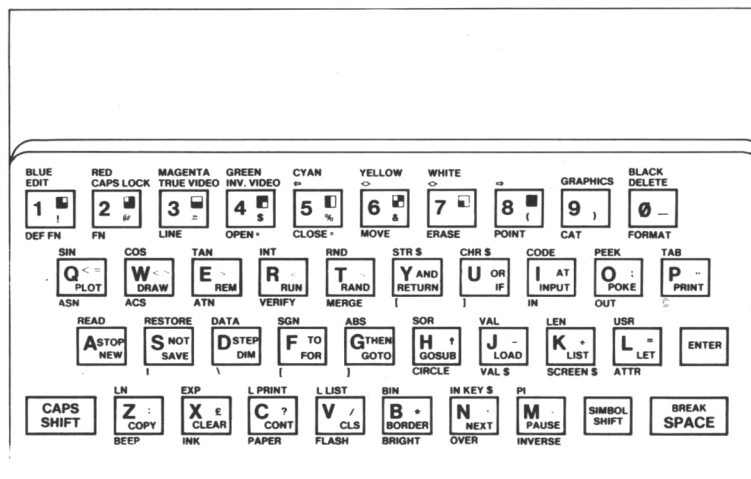
Al modo *E* se pasa, simplemente, apretando la tecla EXTENDED MODE.

De estas dos palabras, la que está situada más al borde, se obtiene apretando directamente la tecla en cuestión. La otra palabra, situada inmediatamente debajo de la anterior —a veces es un símbolo—, se consigue apretando simultáneamente la tecla que nos interese y SYMBOL SHIFT.

Apretando CAPS SHIFT y una tecla de las que contienen letras, obtendremos esa letra en mayúsculas, pasaremos del modo *L* al *C*, apretamos CAPS LOCK.

Con GRAPH, entramos en el modo *G* o gráficos con lo cual, al apretar cualquier tecla que contenga un caracter gráfico como p. e., del 1 al 8, obtendremos ese caracter gráfico.

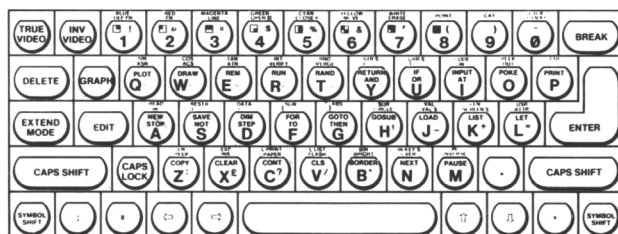
ENTRADAS A LAS TECLAS



Comandos	Tecla n.º	Teclas a apretar	Modo	Comandos	Tecla n.º	Teclas a apretar	Modo
ABS	26	G	E	BRIGHT	35	B + SS	E
ACS	12	W + SS	E	CAT	9	9 + SS	E
AND	16	Y + SS	K	CHR\$	17	U	E
ASN	11	Q + SS	E	CIRCLE	26	H + SS	E
AT	18	I + SS	K	CLEAR	32	X	K
ATN	13	E + SS	E	CLOSE	5	5 + SS	E
ATTR	29	L + SS	E	CLS	34	V	K
BEEP	31	Z + SS	E	CODE	18	I	E
BIN	35	B	E	CONT	33	C	K
BORDER	35	B	K	COPY	31	Z	K

CS signi fica CAPS SHIFT

SS significa SYMBOL SHIFT



Comandos Tecla n.º Teclas a Modo apretar

COS	12	W	E	FN	2	2 + SS	E
DATA	23	D	E	FOR	24	F	K
DEF FN	1	1 + SS	E	FORMAT	0	0 + SS	E
DELETE	0	0 + CS		GOSUB	26	H	K
DIM	23	D	K	GOTO	25	G	K
DRAW	12	W	K	IF	17	U	K
EDIT	1	1 + CS		IN	18	1 + SS	E
ERASE	7	7 + SS	E	INK	32	X + SS	E
EXP	32	X	E	INKEY\$	36	N	E
FLASH	34	V + SS	E	INPUT	18	I	K

Comandos	Tecla n.º	Teclas a apretar	Modo
INT	14	R	E
INVERSE	37	M+SS	E
LEN	28	K	E
LET	29	L	K
LINE	3	3+SS	E
LIST	28	K	K
LLIST	34	V	E
LN	31	Z	E
LOAD	27	J	K
LPRINT	33	C	E
MERGE	15	T+SS	E
MOVE	6	6+SS	E
NEW	21	A	K
NEXT	36	N	K
NOT	22	S+SS	K
OPEN	4	4+SS	E
OR	17	U+SS	K
OUT	19	O+SS	E
OVER	36	N+SS	E
PAPER	33	C+SS	E
PAUSE	37	M	K
PEEK	19	O	E
PI	37	M	E
PLOT	11	Q	K
POINT	8	8+SS	E

Comandos	Tecla n.º	Teclas a apretar	Modo
POKE	19	O	K
PRINT	10	P	K
RAND	15	T	K
READ	21	A	E
REM	13	E	K
RESTORE	22	S	E
RETURN	16	Y	K
RND	15	T	E
RUN	14	R	K
SAVE	22	S	K
SCREEN\$	28	K+SS	E
SGN	24	F	E
SIN	11	Q	E
SQR	26	H	E
STEP	23	D+SS	K
STOP	21	A+SS	K
STR\$	16	Y	E
TAB	10	P	E
TAN	13	E	E
THEN	25	G+SS	K
TO	24	F+SS	K
USR	29	L	E
VAL	27	J	E
VAL\$	27	J+SS	E
VERIFY	14	R+SS	E

APENDICE 2

Facilidades de edición

Denominamos *puntero*, a una pequeña flechita (>) que aparece en la pantalla situada entre el número de línea de programa y la instrucción que dicha línea contiene.

Este puntero nos señala la línea que, en ese momento, puede ser manipulada o editada.

Para editar una determinada línea de programa, es necesario desplazar el puntero hasta esa línea, y esto se consigue mediante las flechas situadas a la derecha de la barra espaciadora.

Una vez el puntero en la línea deseada, basta apretar la tecla EDIT para que una reproducción de la misma aparezca al pie de la pantalla. Sobre esta nueva línea, sólo tendremos que desplazar el cursor justo a la derecha del carácter equivocado y apretar DELETE, para que tal carácter desaparezca.

En el caso de querer insertar algo, igualmente tendríamos que desplazar el cursor hasta el lugar en que hay que introducir la variación, y comenzar a escribir.

Para reincorporar la línea editada al listado, con las modificaciones realizadas, apretamos ENTER/

APENDICE 3

Funciones lógicas aplicadas

Al hablar sobre los *operadores de relación*, dijimos que éstos sólo actúan sobre dos operandos cuyo resultado únicamente puede ser *cierto* o *falso*. Por ejemplo, en $A > B$ sólo hay dos posibilidades esperadas: A es mayor que B o no lo es.

Los *operadores lógicos* actúan entre *operadores* conformados por *operadores lógicos* sólo cuando actúan entre dos posibilidades *cierto* y *falso*.

En éste Apéndice vamos a estudiar *la aplicación de los operadores lógicos AND, OR y NOT* en combinación con la instrucción *IF/THEN*.

Analicemos desde un punto de vista estrictamente lógico la siguiente expresión:

```
IF (A = 15) AND (B > 100) THEN PRINT «bien»
```

Las dos posibilidades que se nos ofrecen son que A sea o no igual a 15 y que B sea o no mayor que 100. En función de ambas alternativas, escribiremos o no la palabra *“bien”*.

En el ejemplo tenemos dos operadores de relación, *“=”* y *“>”*, y el resultado de aplicarlos como sabemos entre dos operandos, sólo puede ser *cierto* o *falso*. Asociemos *cierto* con el símbolo 1 y *falso* con el 0.

De aquí deducimos que, las combinaciones posibles de 1 y 0 que nos ofrecen ambas *operaciones de relación* son:

AND

$A = B \Rightarrow \text{cierto} \Rightarrow 1$
 otra posibilidad $\Rightarrow \text{falso} \Rightarrow 0$
 $B > 100 \Rightarrow \text{cierto} \Rightarrow 1$
 otra posibilidad $\Rightarrow \text{falso} \Rightarrow 0$

NOTA: El símbolo $\Rightarrow$ significa «implica»

Volviendo a la expresión inicial, escribiremos las posibilidades de que la palabra “bien” sea escrita:

Si $A = B$ (cierto ó 1) Y $B > 100$ (cierto ó 1) entonces sí se escribe
 Si (falso ó 0) Y (falso ó 0) entonces no se escribe
 Si (falso ó 0) Y (cierto ó 1) entonces no se escribe
 Si (cierto ó 1) Y (falso ó 0) entonces no se escribe

Es decir, que el operador lógico y exige que los dos operadores sobre los que actúa sean 1 —cierto— para permitir que suceda la acción que le sigue.

Para concluir, hagamos igual a X el resultado (0 ó 1) de la primera operación de relación ($A = 15$, en el ejemplo) e igual a Y el resultado (0 ó 1) de la segunda operación de relación ($B > 100$, en el ejemplo); pudiendo presentar las posibilidades que ofrece el operador lógico Y (AND, en inglés) mediante la siguiente

TABLA DE LA VERDAD: AND

x	y	x AND y	COMENTARIOS
1	1	1	Se permite la acción subsecuente
0	0	0	Se prohíbe la acción subsecuente
0	1	0	Se prohíbe la acción subsecuente
1	0	0	Se prohíbe la acción subsecuente

En lo sucesivo nos referiremos directamente a la *tabla de verdad* de cada operador lógico, evitando la explicación que no lleva a ella.

Ejemplo:

Para comenzar los estudios de Bachillerato es necesario tener 14 años y haber obtenido una calificación mínima de 5 puntos en la E.G.B. Estudie éste programa y ejecútelo.

Programa:

```
10 INPUT "Algún alumno?"; A$
20 IF A$ = "si" THEN GO SUB 60
30 IF A$ = "no" THEN STOP
40 RETURN
50 CLS
60 INPUT "Nombre del alumno?"; N$ : PRINT N$
70 INPUT "Edad?"; E : PRINT E
80 INPUT "Calificación?"; C : PRINT C
90 IF (E ≥ 14) AND (C < 5) THEN GOTO 140
100 PRINT "Admitido"
110 PRINT
120 INPUT "Otro alumno?"; A$
130 GOTO 20
140 PRINT "Rechazado" : GOTO 110
```

OR

Este *operador lógico* permite la acción subsecuente siempre que uno cualquiera de los operadores de relación sea 1 (cierto).

Volviendo al ejemplo del operador anterior, pero cambiando éste por el actual (OR), tendremos en términos de BASIC:

```
IF (A = 15) OR (B ≥ 100) THEN PRINT "bien"
```

Según la siguiente

TABLA DE VERDAD: OR

x	y	x OR y	COMENTARIOS
1	1	1	Se permite la acción subsecuente
0	0	0	No se permite la acción subsecuente
0	1	1	Se permite la acción subsecuente
1	0	1	Se permite la acción subsecuente

Podemos deducir que la palabra "bien" será impresa en pantalla en el caso de que A sea igual a 15 ó B sea mayor que 100.

Para comenzar los estudios de bachillerato es necesario haber aprobado la E.G.B. o tener más de 21 años. Estudia este programa y ejecútalo.

Programa:

```
10 INPUT "E.G.B. aprobada?"; A$
20 INPUT "Edad?"; E
30 IF (A$ = "SI") OR (E > 21) THEN GOTO 60
40 PRINT "RECHAZADO"
50 STOP
```

```
60 PRINT "ADMITIDO"  
70 STOP
```

Este es el más sencillo de *los operadores lógicos*, ya que el resultado es falso (0) cuando el operador de relación es cierto (1), y cierto (1) cuando el operador de relación es falso (0), según la siguiente

NOT

TABLA DE LA VERDAD: NOT

x	NOT X	COMENTARIOS
1	0	No se permite la acción subsecuente
0	1	Sí se permite la acción subsecuente

Ejemplo:

```
IF NOT (A = B) THEN PRINT "bien"
```

Esto implica que la palabra "bien" será impresa en pantalla sólo en el caso de que A no sea igual a B.

APENDICE 4

Troceado de cadenas

Por *cadena*, y en términos de programación, entendemos cualquier concatenación de caracteres.

Entre otras cosas, hay que tener en cuenta que una cadena sólo se puede igualar a una variable de caracteres. Tal sería el caso de

A\$ = "ROMARVELLAVE, es una ? > 1"

En este ejemplo la cadena está compuesta por letras mayúsculas y minúsculas, dos signos de puntuación, un espacio y un par de símbolos matemáticos. En definitiva caracteres en general.

Trocear una cadena significa subdividirla en cadenas menores. A este tema dedicaremos el Apéndice presente.

Sobre la variable A\$ fijada más arriba, aplicaremos los comandos oportunos.

Este sistema lo seguiremos con todas las instrucciones de éste Apéndice.

Entendemos que, como introducción a ésta materia, son suficientes los COMENTARIOS para hacerse frente idea de la aplicación de cada comando.

Posibilidad 1

Extraer de la cadena representada por A\$, una subcadena con los cuatro caracteres situados más a la izquierda.

Ejemplo:

```
50 LET B$ = A$ (1 TO 4)
60 PRINT B$
```

Comentario:

La ejecución de éstas dos líneas de programa obligará la impresión en pantalla de la cadena ROMA.

Posibilidad 2

Extraer una subcadena del A\$ con los cinco caracteres situados más a la derecha.

Ejemplo:

```
50 LET B$ = A$ (20 TO 24)
60 PRINT B$
```

Comentario:

La ejecución de estas dos líneas nos dará en pantalla: a ? $\geq$ 1
Fíjese el lector que el primer carácter de ésta subcadena es un espacio.
Idéntico resultado.

Posibilidad 3

Extraer una subcadena de A\$ que comience en su tercer carácter y tenga una longitud de 8 caracteres.

Ejemplo:

```
50 LET B$ = (3 TO 10)
60 PRINT B$
```

Comentario:

Al ejecutar éstas líneas obtendremos: MARBELLA

Posibilidad 4

Extraer un solo carácter de la cadena A\$

Ejemplo:

```
50 LET B$ (6 TO 6)
60 PRINT B$
```

Comentario:

En pantalla aparecerá *B*

Todo lo visto más arriba y algunas otras forma de aplicar el comando TO con idéntico resultado, lo podemos ver en el siguiente ejemplo:

Programa

```
10 LET A$ =
«ROMARBELLAVE»
```

Comentarios

La variable A\$ es valorada.

20 PRINT A\$ (TO 4)

30 PRINT A\$ (3 TO 10)

40 PRINT A\$ (8 TO)

50 PRINT A\$ (6 TO 6)

60 PRINT A\$ (1 TO 2)

70 LET B\$ = A\$ (6 TO 10)

Imprimo todos los caracteres desde el principio hasta el cuarto, de la A\$.

Imprimo todos los caracteres desde el tercero al décimo, ambos inclusive, de la A\$.

Imprimo todos los caracteres del octavo al final de la A\$.

Imprimo el sexto carácter únicamente

Imprimo la cadena del principio al final. Igual resultado obtendría con PRINT A\$.

Dejamos la variable B\$ igual a la subcadena A\$ indicada.

OTROS LIBROS DEL AUTOR

- **COMO PROGRAMAR SU SPECTRUM**
Ed. Paraninfo
- **LOS COLORES Y LOS GRAFICOS EN EL SPECTRUM**
Ed. Paraninfo
- **CURSO-INCIACION AL CODIGO MAQUINA**
Ed. Paraninfo
- **BASIC PARA MAESTROS**
Ed. Paraninfo
- **BASIC PARA ESTUDIANTES**
Ed. Paraninfo

sinclair

ZX Spectrum +

