MICROHOBBY

# AMSTRAD

Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO II N.º 21

**160 Ptas.** (incluido I.V.A.)

GESTION CONTABLE EN DISCO

**TRATAMIENTO AVANZADO DE PALABRAS Y CARACTERES**

**Desafía a tu Amstrad a una partida de Black Jack**

**COMO SE HACE UN TEST POR ORDENADOR**

**SOFTWARE**

*Gyroscope: una peonza mecánica en el laberinto*


HOBBY PRESS, S.A.

SI BUSCAS LO MEJOR ERBE Software LO TIENE

LA REPLICA AL CELEBRE "ROLLING" DE LAS MAQUINAS.
EL JUEGO MAS ADICTIVO QUE PUEDAS ENCONTRAR

# GYROSCOPE


DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA ERBE SOFTWARE, SANTA ENGRACIA, 17. Tel.: 447 34 10. DELEGACION BARCELONA, Avd. MISTRAL, 19. Tel. (93) 432 07 31

**Director Editorial**
José I. Gómez-Centurión
**Director Ejecutivo**
Víctor Prieto
**Subdirector**
José María Díaz
**Redactora Jefe**
Marta García
**Diseño**
José Flores
**Colaboradores**
Francisco Portalo, Pedro Sudón
Miguel Sepúlveda,
Francisco Martín,
Jesús Alonso, Pedro S. Pérez
Amalio Gómez
Juan J. Martínez,
David Sopuerta, Alberto Suñer,
Eduardo R. Velasco
**Corresponsal en Londres**
Alan Heap
**Secretaria Redacción**
Carmen Santamaría
**Fotografía**
Carlos Candel
Javier Martínez
**Portada**
M. Barco
**Ilustradores**
Javier Igual, J. Pons, F. L.
Frontán, J. Septien, Pejo, J. J.
Mora, Luigi Pérez

**Edita**
HOBBY PRESS S.A.

**Presidente**
María Andrino
**Consejero Delegado**
José I. Gómez-Centurión

**Jefe de Publicidad**
Concha Gutiérrez
**Publicidad Barcelona**
José Galán Cortes
Tel: (93) 303 10 22/313 71 62

**Secretaria de Dirección**
Marisa Cogorro

**Suscripciones**
M.ª Rosa González
M.ª del Mar Calzada

**Redacción, Administración y Publicidad**
La Granja, 39
Polígono Industrial de Alcobendas
Tel.: 654 32 11
Telex: 49 480 HOPR

**Dto. Circulación**
Carlos Peropadre

**Distribución**
Coedis, S. A. Valencia, 245
Barcelona

**Imprime**
ROTEDIC, S. A. Crta. de Irún.
Km. 12,450 (MADRID)
**Fotocomposición**
Novocomp, S.A.
Nicolás Morales, 38-40
**Fotomecánica**
GROF
Ezequiel Solana, 16
**Depósito Legal:**
M-28468-1985

Derechos exclusivos de la revista **COMPUTING with the AMSTRAD**

Representante para Argentina, Chile, Uruguay y Paraguay, Cía. Americana de Ediciones, S.R.L. Sud América 1.532. Tel.: 21 24 64. 1209 BUENOS AIRES (Argentina).

M. H. AMSTRAD no se hace necesariamente solidaria de las opiniones vertidas por sus colaboradores en los artículos firmados. Reservados todos los derechos.

Se solicitará control OJD

MICROHOBBY

# AMSTRAD

## sumario

*Año II • Número 21 • 21 al 27 de Enero 1986*
*160 pts. (incluido el IVA)*
*Precio Canarias, Ceuta y Melilla*
*155 + 10 ptas. sobretasa aérea*

# Es usted capaz de tomar el relevo del General Montgomery...

?

## & Juegos ESTRATEGIA

le presenta en exclusiva el WAR GAME, para Spectrum, de mayor éxito en Inglaterra:

**ARNHEM**

(operación «*Market Garden*», basada en un hecho real de la Segunda Guerra Mundial)

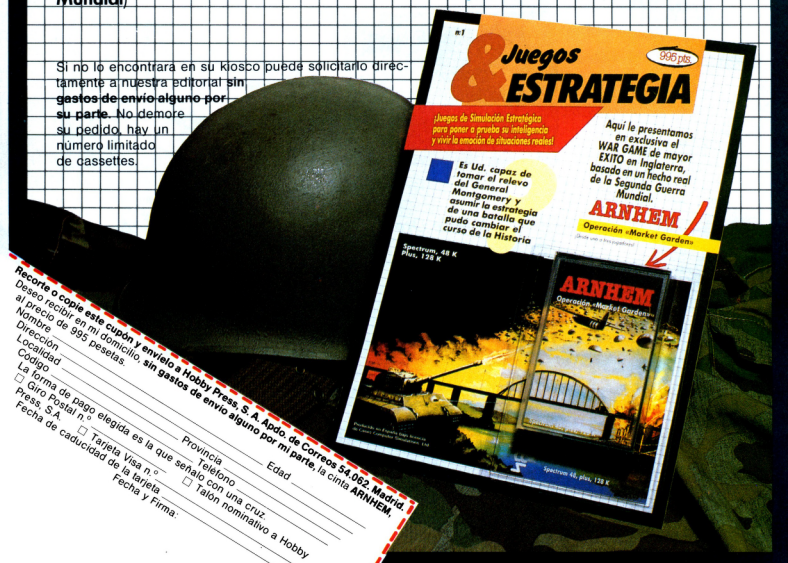
Ya está a la venta!

*Y ahora también para*

**AMSTRAD**

*Compatible todos los modelos*

Si no lo encontrara en su kiosco puede solicitarlo directamente a nuestra editorial **sin ~~gastos de envío alguno por su parte~~**. No demore su pedido, hay un número limitado de cassettes.




**Recorte o copie este cupón y envíelo a Hobby Press, S.A. Apdo. de Correos 54.062. Madrid.** Deseo recibir en mi domicilio, **sin gastos de envío alguno por mi parte**, la cinta **ARNHEM**, al precio de 995 pesetas.

Nombre _____

Dirección _____

Localidad _____ Provincia _____ Edad _____

Código _____ Teléfono _____

La forma de pago elegida es la que señalo con una cruz

☐ Giro Postal n.º _____ ☐ Tarjeta Visa n.º _____

☐ Talón nominativo a Hobby Press S.A.

Fecha de caducidad de la tarjeta _____

Fecha y Firma _____

**HOBBY PRESS, S.A.** *Editamos para gente inquieta.*

# Novedades Apple

Los ordenadores Apple II y McIntosh se verán complementados por una interesante gama de periféricos que Apple Computer acaba de lanzar al mercado.

Entre ellos destacan la Apple RAM Expansion Card de 256 Kbytes, que permite aumentar la memoria de los Apple II, II+ y IIe hasta un máximo de un megabyte. Esta ingente cantidad de memoria puede emplearse como RAM disc o bien como memoria de acceso aleatorio normal. En el Firmware de la Card existen las necesarias rutinas de gestión de memoria.

Otro periférico que acaba de aparecer es un disco de 800 K formateadas para el Apple IIc, formato 3,5 pulgadas, el mismo que el del Mac. Otras máquinas de la serie II necesitan modificaciones en la ROM para poder manipular el disco.

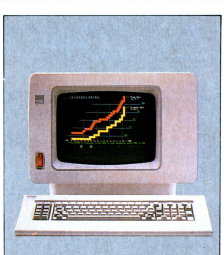
Los productos que soportan el Uni-Disk 3.5 y la tarjeta de ampliación de memoria, pueden trabajar con algo que Apple ha llamado **«Guark Catalyst 3.0»**, que no es más que un interface tipo MacIntosh para el Apple II.

# NCR ESPAÑA

NCR ha incrementado su facturación en la primera mitad del presente ejercicio en un 37 por 100.

Según su director, NCR ha tenido y tiene una gran penetración en todos los sectores informáticos, especialmente en lo que respecta a los cajeros automáticos y al sistema financiero 5000. Este último mercado constituye la mayor parte de su volumen de negocios.

NCR posee muchos modelos de ordenadores; tal vez uno de los más conocidos sea el que incorpora dos microprocesadores: un Z80 (compatibilidad CP/M) y un 8088 *(compatibilidad IBM)*.



# Mas compatibles para IBM AT

Bull, primer fabricante francés de ordenadores, conocido entre otras cosas por sus tarjetas de crédito inteligentes, con un microprocesador incluido, lanzará próximamente un ordenador, denominado AJAX, compatible con el IBM AT, completando así a otros modelos que ya posee el fabricante francés, compatibles con el IBM PC.

El nuevo ordenador posee prácticamente las mismas características que el AT: 512 K de RAM, varios modos gráficos y el 80286 de Intel como unidad central de proceso y unos cuantos extras, como un precio inferior en un 10 por 100 a IBM, conexión con otros ordenadores BULL, etc.

La galaxia de compatibles se ve ampliada con un nuevo miembro *(tachín, tachiiiin)*.

# Base de datos para el Tribunal Supremo

Este venerable organismo contará a partir de ahora con una base de datos, suponemos que inmensa, con toda la jurisprudencia y los datos del Tribunal que necesiten ser consultados por juristas y demás profesionales. La conexión se hará entre un ordenador del Supremo y la base de datos jurídica Datalex.

# Primera plana

## Informatica en el Ejercito

Anisa y CTI se han llevado un contrato de 48 millones de pesetas para el desarrollo de un sistema informático para nuestra Armada.

Concretamente será el encargado del control de configuración *(¡tela!)* del nuevo grupo de combate integrado por el portaaeronaves **«Príncipe de Asturias»** y las fragatas tipo FGG.

# El lenguaje chino en ordenador

La República Popular China está tratando de ponerse rápidamente al día en lo que respecta a temas informáticos, para tratar de alcanzar a Occidente lo antes posible.

A pesar de que los chinos llevan un cierto retraso en lo relativo a ordenadores, han conseguido crear un sistema de procesamiento del lenguaje chino único en el mundo.

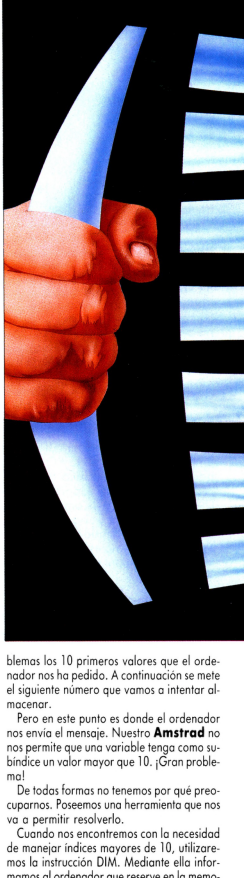
Creado en la Universidad de Educadores de Beijing, permite establecer la frecuencia de uso de caracteres escritos, editar diccionarios e investigar acerca del lenguaje.

El software del sistema incorpora tres bases de datos de caracteres auxiliares chinos, archivo indexado de palabras e información sobre frases y lenguajes.

Fuentes de International Data Corp. informaron que el sistema ha sido estudiado a fondo y aprobado por más de 60 lingüistas y expertos en computación.

# VARIABLES CON SUBINDICE (MATRICES II)

***Las variables con subíndice o matrices han entrado de lleno en nuestra «vida informática». Han revolucionado nuestro concepto de variable «sencilla» y nos han proporcionado una herramienta muy valiosa para poder agrupar diversos valores de un mismo tipo bajo el nombre de una única variable.***



Javier Igual

En nuestro último encuentro recibimos una avalancha de conceptos, palabras nuevas, definiciones, etc. sobre matrices, dimensiones y subíndices. Nuestra idea es que se vayan familiarizando poco a poco con este nuevo tipo de variables, entiendan el porqué de su utilización y se adentren con cuidado y atención en el maravilloso mundo de las matrices. Para conseguirlo vamos a detenernos minuciosamente en cada uno de los conceptos expuestos.

Recordemos lo que era una variable con subíndice. Es un nombre de una variable sencilla seguida de un índice encerrado entre paréntesis.

conjunto (7)

será el elemento número siete de la matriz «conjunto»; «conjunto» sería el nombre de la matriz y 7 el subíndice del elemento.

Observen que mientras:

conjunto (7)

es un elemento de una matriz.

conjunto 7

es una variable sencilla y no una variable con subíndice. Si nos las encontramos en un mismo programa que no se nos olvide que son variables distintas, aunque ambas contienen valores numéricos.

Del mismo modo:

grupo$ (8)

sería el elemento 8 de la matriz «grupo$». Como habrán adivinado esta última matriz contendrá valores alfanuméricos —termina en $.

Uno de los misterios maravillosos de las matrices es que los subíndices pueden ser a su vez variables. Sería válido asegurar:

conjunto (elemento) = >

siempre que anteriormente hayamos dado algún valor a la variable «elemento».

Los subíndices pueden ser también expresiones más complejas siempre que su evaluación sea un número entero. Podríamos hacer:

gruposS (3*5) = «PEPE»

sin tener ningún tipo de problemas —a menos que PEPE se enfade por utilizar su nombre.

Para dar valores a todos los elementos de una matriz utilizamos generalmente las instrucciones INPUT o READ repitiendo un número fijo de veces un bucle (FOR... NEXT) o haciéndolo hasta que introduzcamos un valor previamente establecido que haga que nos salgamos del bucle (WHILE.. WEND). Veamos lo que decimos con el Programa 1.

## PROGRAMA UNO

Con la línea 30 le decimos al ordenador el número de elementos que queremos tenga la matriz **«conjunto».**

Las líneas 50 a 80 dan a dichos elementos los valores que nosotros metemos por el teclado. Observemos que la variable «elementos» se utiliza como límite superior de la variable de control del bucle «indice1» y así nos aseguramos de rellenar el número de elementos deseado.

Las líenas 120 a 140 nos sacan un listado con el contenido completo de la matriz.

Si metemos en la variable **«elementos»** un 3, los dos bucles FOR... NEXT se repetirán 3 veces cada uno y por lo tanto en el primero se nos pedirá que introduzcamos por el teclado otros tantos números y en el segundo se nos imprimirán todos los valores que hayamos metido.

Utilicemos ahora este programa para intentar rellenar con unos determinados valores una matriz de 16 elementos.

Nuestro **Amstrad** se revela y nos da un mensaje de error:

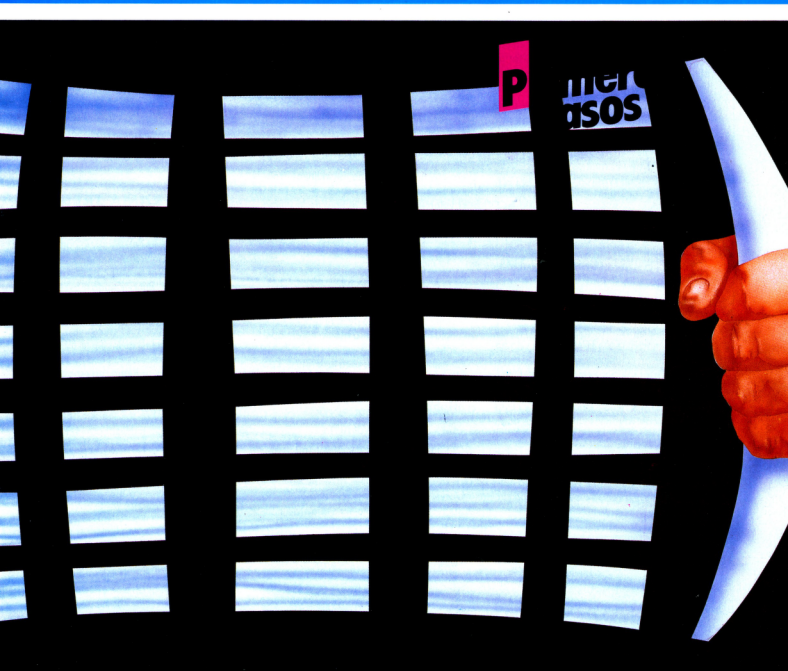
Subscript out of range in 80

**¿Qué ha ocurrido?** Analice los pasos que ha seguido. Hemos introducido sin problemas los 10 primeros valores que el ordenador nos ha pedido. A continuación se mete el siguiente número que vamos a intentar almacenar.

Pero en este punto es donde el ordenador nos envía el mensaje. Nuestro **Amstrad** no nos permite que una variable tenga como subíndice un valor mayor que 10. ¡Gran problema!

De todas formas no tenemos por qué preocuparnos. Poseemos una herramienta que nos va a permitir resolverlo.

Cuando nos encontremos con la necesidad de manejar índices mayores de 10, utilizaremos la instrucción DIM. Mediante ella informamos al ordenador que reserve en la memoria espacio suficiente para contener una matriz que tiene más de 10 elementos. Además también le decimos el valor máximo del subíndice que va a tener uno de dichos elementos.



Agregue al programa anterior la línea:

15 DIM conjunto(16)

y repita los pasos anteriores. Ahora sí nos admitirá 16 valores.

Y si ponemos:

15 DIM conjunto(1000)

nos permite rellenar una matriz que tenga 1.000 elementos.

Podemos sacar en conclusión que el **Amstrad** permite trabajar con arrrays que no estén DIMensionados siempre que el valor máximo de los subíndices no sea mayor que 10. En caso contrario tendremos que avisarle de ello —DIMensionarlos.

En este programa hemos utilizado dos índices diferentes para acceder a cada uno de los elementos de la lista —«índice1» e «índice2». Los hemos elegido así arbitrariamente ya que al ser bucles FOR... NEXT diferentes y no estar anidados uno dentro del otro podríamos usar una sola variable de control para ambos bucles.

Veamos cómo rellenar una matriz por medio de las instrucciones WHILE... WEND y READ... DATA.

## PROGRAMA DOS

En la línea 20 definimos y dimensionamos en nuevo array o lista que en este caso va a ser alfanumérica y a poder contener 100 elementos.

La línea 210 contiene todos los datos que queremos meter. El primero es el número de los mismos. Lo leemos en la línea 50 y su valor lo colocamos en **«grupo$ (0)»** —ya hablaremos sobre él en el subíndice 0. Nos sirve de momento para hacer por primera vez el bucle WHILE...WEND de las líneas 70 a 110.

En este bucle vamos leyendo las sucesivas cadenas que hay en la línea 210 y las almacenamos en los distintos elementos del array hasta que encontramos un valor que sea la cadena vacía. Esta es la condición de salida del bucle: encontrar la cadena vacía en uno de los elementos de la matriz. Por eso lo primero que hacemos es meter una cadena literal con los mismos dígitos que el número de elementos en **«grupo$ (0)»** y así tenemos ya un valor distinto de la cadena vacía —recuerde que el **Amstrad** inicializa las listas con el valor «»— y se ejecuta por primera vez el bucle WHILE... WEND.

En las líneas 150 a 180 realizamos un proceso semejante para sacar en pantalla todos los

elementos a los que hemos dado algún valor.

Como subíndice para recorrer los elementos que vamos a rellenar utilizamos la variable «índice» y la vamos incrementando en las líneas 80 y 170 respectivamente.

Con la línea 200 escribimos el número de datos que hemos dado a dichos elementos.

Vamos a ver ahora un supuesto práctico. Nos han dirigido una encuesta para elegir la mejor revista de informática y nos han propuesto realizar un programa que determine el número de votos que obtiene cada revista. Cada papeleta va a contener un dígito del 1 al 4. Si el voto contiene un 1 le damos un punto a «MicroHobby», si contiene un 2 es un punto a «MicroHobby-Cassette». Cuando contenga un 3 votamos a **«MicroHobby-AMSTRAD»** y con un 4 a **«Micromanía»**.

Le pasamos la bola y le invitamos a que intente hacer este programa antes de seguir. Mientras, nosotros hacemos una pausa.

¿Lo ha conseguido? ¡Bien! Veamos si lo ha pensado de una manera semejante a la nuestra. Una de las soluciones más eficaces al problema planteado sería:

1. Declarar y dimensionar un array cuyos elementos contengan los valores de los datos recibidos por cada una de las revistas:

```
DIM votos(3)
```

2. Tomar la viable **«revista»** como índice de la matriz **«votos»** dimensionada anteriormente.

A partir de aquí podemos proceder a contar normalmente los votos emitidos para cada una de las revistas pero no lo vemos muy práctico, **¿verdad?**

Mejor, y más útil, es fabricarnos una o varias líneas de DATA en las que tengamos registrados todos los votos emitidos.

Sería semejante a:

```
DATA 1,2,3,3,1,4,3,2,-1
```

¿Pero qué hace —1 al final de la línea? Tranquilo, se lo vamos a explicar.

El valor —1 colocado al final de la última línea de DATAs nos indica sencillamente que nuestros votos se acabaron: ya no existen más votos y podemos proceder al recuento. No es un voto más.

Nuestro programa va a leer cada uno de los votos. Dependiendo del número que lleve va a sumar un punto al elemento de la matriz que contenga los puntos de la revista utilizando como índice el número escrito en el papel del voto.

Por ejemplo, el voto es para la revista 1. Esto implica qué **«revista»** va a valer 1 y tenemos que sumar un punto a «votos(1)» —**«MicroHobby»:**

```
votos(1) = votos(1)+1
```

y generalizando, esta última expresión sería:

```
votos (revista) = votos (revista)+1
```

Observe que esta sentencia de asignación es equivalente a la forma en la que incrementábamos la variable «índice» en el programa anterior:

```
índice = índice+1
```

Cuando nos llega el indicador de fin de datos (—1) detenemos la cuenta de los votos y podemos hacer el recuento e imprimir los resultados.

Pero no le vamos a hacer rabiar más. Vea si su programa coincide con el nuestro.

## PROGRAMA TRES

En la línea 20 DIMensionamos el array. En el desarrollo posterior del programa podrá observar que ningún índice es superior a 10 y por tanto no será necesario utilizar esta instrucción pero creemos que, aunque nada más sea por razones de claridad, es muy conveniente usarla siempre.

El recuento de votos los realizamos con el bucle WHILE... WEND de las líneas 60 a 100. En él leemos los datos de la línea 190 y anotamos los votos en su lugar correspondiente en la línea 80.

Los resultados los sacamos simplemente con instrucciones PRINT en las líneas 110 a 180.

Aquí hemos vuelto a utilizar otra vez el elemento con subíndice 0 —«votos(0)». Esta vez lo hemos usado como contador de los votos totales emitidos. Por cierto que vamos incrementando su valor en la línea 70. Pero no es esta su única misión. ¡Ya lo verá!

No olvide que siempre tenemos en nuestras manos el elemento (o los elementos si la matriz es de varias dimensiones) que tiene como índice 0 (recuerde nuestra juguetona mentirijilla que, sobre el tema, le dijimos la semana anterior) y podemos utilizar estos elementos para los fines que mejor nos parezcan. Le insistimos que vamos a procurar ver las mejores utilizaciones de los mismos.

Y ahora una pregunta: **¿Qué valor tendrán todos los elementos de la matriz?** La respuesta es bien sencilla: CERO, ya que nuestro **Amstrad** las inicializa a 0 aunque nosotros no se lo digamos. Hace algo parecido a lo que hemos visto que ocurre con los arrays de literales solamente que ahora, en vez de imicializarlos con la cadena vacía, los inicializa con ceros.

Le proponemos ahora que intente añadir una condición al Programa III para que no admita votos para la revista número 16, por ejemplo, ya que no existe, así como registrar los votos de las personas que no estén conformes con las revistas indicadas y sean partidarias del RESTO (mal llamado competencia).

También puede modificar este programa para que además de los votos nos aparezca el % de votos de cada una de las revistas redondeando al entero más próximo.

Bueno, ¡hasta la próxima semana! Intentaremos adentrarle en el uso de variables con dos subíndices o matrices de dos dimensiones y ¡anímese, que es fácil!

## PROGRAMA I

```
10 REM PROGRAMA I
20 CLS
30 INPUT "NUMERO DE ELEMENTOS: ",el
ementos
40 REM DAR VALOR A LOS ELEMENTOS
50 FOR indice1=1 TO elementos
60 PRINT "VALOR";indice1;
70 INPUT ":",valor
80 conjunto(indice1)=valor
90 NEXT indice1
100 REM LISTADO VALORES
110 PRINT
120 FOR indice2=1 TO elementos
130 PRINT"CONJUNTO(";indice2;") =";
conjunto(indice2)
140 NEXT indice2
150 PRINT
160 PRINT"LA MATRIZ TIENE";elemento
s;"ELEMENTOS"
```

## PROGRAMA II

```
10 REM PROGRAMA II
20 DIM grupo$(100)
30 CLS
40 REM DAR VALOR A LOS ELEMENTOS
50 READ valor$
60 grupo$(0)=valor$
70 WHILE grupo$(indice)<>""
80 indice=indice+1
90 READ valor$
100 grupo$(indice)=valor$
110 WEND
120 REM LISTADO VALORES
130 indice=1
140 PRINT
150 WHILE grupo$(indice)<>""
160 PRINT"GRUPO$(";indice;") =";gru
po$(indice)
170 indice=indice+1
180 WEND
190 PRINT
200 PRINT"LA MATRIZ TIENE";VAL(grup
o$(0));"ELEMENTOS"
210 DATA "6","PEPE","RICARDO","PACO
","JESUS","LUIS","PEDRO",""
```

## PROGRAMA III

```
10 REM PROGRAMA III
20 DIM votos(4)
30 CLS
40 REM RECUENTO DE VOTOS
50 READ revista
60 WHILE revista<>-1
70 votos(0)=votos(0)+1
80 votos(revista)=votos(revista)+1
90 READ revista
100 WEND
110 REM IMPRESION DE RESULTADOS
120 PRINT
130 PRINT"MICROHOBBY:";votos(1);"VO
TOS"
140 PRINT"MICROHOBBY-CASSETTE:";vot
os(2);"VOTOS"
150 PRINT"MICROHOBBY-AMSTRAD:";voto
s(3);"VOTOS"
160 PRINT"MICROMANIA:";votos(4);"VO
TOS"
170 PRINT
180 PRINT"TOTAL VOTOS EMITIDOS:";vo
tos(0)
190 DATA 1,3,2,4,3,1,2,3,1,3,4,2,3,
4,3,-1
```

# PONTOON

***Pontoon es una simulación por ordenador del conocido juego de cartas del mismo nombre, tan parecido a nuestras entrañables «siete y media».***
***Se trata de ir almacenando puntuación sin pasar de 21, y batir a la banca, que en este caso es el Amstrad.***

El ordenador nos da y se da inicialmente dos cartas; las nuestras están ambas boca arriba, pero sólo vemos una del ordenador.

Tenemos dos opciones: o nos plantamos pulsando la «S» o pedimos otra carta mediante la **«T»**. No hay límite al número de cartas que podemos recibir, siempre y cuando la suma de sus puntuaciones no pase de 21.

A todo esto, cada carta vale su número de orden, excepto los reyes, que valen 10, y los ases, que valen 1 u 11.

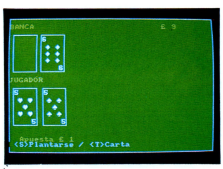
## VARIABLES MAS IMPORTANTES

| | |
|---|---|
| x, y | Coordenadas de las cartas. |
| c | Número de carta (1-13). |
| player | 0=computer, 1=jugador. |

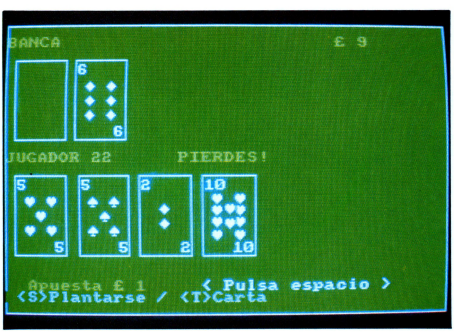
Cuando nos plantamos, el ordenador estudia nuestro juego y el suyo, y se va echando cartas hasta que nos gane o se pase.

Comenzamos con un capital a nuestra disposición de 10.

El autor del programa ha incluido al principio del mismo una curiosa opción de autodestrucción. Si quiere saber cómo funciona, ¡salve el programa en disco antes!








## ESTRUCTURA DEL PROGRAMA

| | |
|---|---|
| 10-60 | Lee los valores de las cartas. |
| 70-135 | Dibuja la pantalla. |
| 140 | Comienza el bucle principal. |
| 220 | Comienza la rutina que rige el juego del ordenador. |
| 260 | Rutina del jugador. |
| 470 | Ganadores. |
| 570 | Dibuja cartas. |
| 870 | «Voz» del ordenador. |
| 940 | ¡Pontoon! |
| 1030 | Se acabó el dinero. |

# Serie Oro

```
5 REM(c)Amstrad Semanal
10 DIM  card$(13)  , pack(4,13)
20 MODE 1 : BORDER 0 : INK 0,0 : IN
K 1,26 : INK 2,18 : INK 3,22
30 REM lee los valores de las carta
s
40 FOR n=1 TO 13 : READ card$(n) :
NEXT
50 DATA A,2,3,4,5,6,7,8,9,10,J,Q,K
60 money=10  : REM dinero para apos
tar
70 REM ******* pantalla inicial ***
****
80 PEN 2 : LOCATE 15,1 : a$="¥ PONT
OON ¥" : GOSUB 870 : GOSUB 920
85 LOCATE 1,15 : PRINT "instruccion
es ?"
86 a$=INKEY$ : IF a$="" THEN 86 ELS
E IF LEFT$(UPPER$(a$),1)<>"Y" THEN
 120
90 CLS : PRINT : PRINT"El objeto de
l juego es ir sumando las  puntuac
iones de las cartas que recibas ylu
ego compararlas con las del ordenad
or."
95 PRINT "Si el total es mayor, gan
as. Pero el valor no debe exceder d
e 21. Si sucede,puespierdes."
100 PRINT "Si sumas 21, te dan un b
onus":PRINT"Las teclas de control s
on:":PRINT"S... Para plantarse":PRI
NT"T... Para pedir otra carta."
110 PRINT:PRINT"Para hacer el juego
 mas interesante hay una opcion de
autodestruccion: el programa se bor
rara si se te acaba el dinero."
```


```
114 LOCATE 11,23:PRINT"< Pulsa una
tecla >"
115 WHILE INKEY$="":WEND
120 CLS:LOCATE 7,10 : a$="Auto-dest
ruc  ON/OFF " : GOSUB 870
130 INPUT autodestruct$ : autodestr
uct$=UPPER$(autodestruct$)
135 a$="Apuesta minima es " : LOCAT
E 12,18 : GOSUB 870 : PRINT CHR$(16
3);" 1" : GOSUB 920 : GOSUB 920
140 REM ******* bucle principal ***
****
150 playertotal=0 : dealertotal=0 :
 dealeraces=0 : playeraces=0
160 CLS : PEN 1 : INK 0,4 : BORDER
4 : GOSUB 920 :a$=" Tienes ":LOCATE
 1,10:GOSUB 870:PRINT CHR$(163);" "
;money
170 GOSUB 920 : a$=" Cuanto apuesta
s " :LOCATE 1,14: GOSUB 870
180 INPUT bet : IF bet < 1 OR bet >
 money THEN LOCATE 2,18 : a$="Que?
Trata de nuevo. Apuesta " : GOSUB 8
70 : LOCATE 25,18 : GOTO 180
190 CLS : INK 0,9 : BORDER 0 : LOCA
TE 1,23 : PEN 2 : PRINT"  Apuesta";
" ";CHR$(163);bet
200 money=money-bet : LOCATE 31,2 :
 PRINT CHR$(163);money
205 LOCATE 1,2 : PRINT "BANCA" : LO
CATE 1,12 : PRINT "JUGADOR"
210 MOVE 0,0 : DRAW 639,0 : DRAW 63
9,399 : DRAW 0,399 : DRAW 0,0
220 player=0 : start=1 : x=15 : y=2
50 : GOSUB 570 : REM primera carta
de la banca
230 x=105 : GOSUB 570      : REM seg
unda carta de la banca
240 player=1 : x=15 : y=90 : GOSUB
570 : x=110 : GOSUB 570
250 TAGOFF : LOCATE 20,23 : PEN 3 :
 PRINT " <S>Plantarse / <T>Carta"
260 WHILE playertotal<22
270  reply$=UPPER$(INKEY$) : IF rep
ly$="" THEN 270
280  IF reply$<>"T" THEN  310
290  SOUND 1,100,5 : SOUND 2,110,10
 : x=x+90 : GOSUB 570 : REM otra ca
rta
300 WEND
```

```
310 WHILE playeraces<>0
320  IF playertotal+10>21 THEN play
eraces=playeraces-1
330  IF playertotal+10<22 THEN play
eraces=playeraces-1 : playertotal=p
layertotal+10
340 WEND
345 LOCATE 8,12 : PEN 2 : PRINT pla
yertotal
350 IF playertotal>21 THEN LOCATE 1
7,12 : PRINT "PIERDES!      " : SOUN
D 1,3000,40 : GOTO 510
360 IF playertotal=21 THEN  GOSUB 9
40 : REM pontoon !
370 player=0 : x=15: y=250 : c=star
tcard : s=startsuite
380 TAG : GOSUB 700 : x=105 : REM d
ibuja la primera carta de la banca
390 WHILE dealertotal < playertotal
400  IF dealeraces=0 OR dealertotal
=2 THEN 430
410  IF dealertotal+10 > 21 THEN de
aleraces=dealeraces-1 : pace=pace+1
420  IF dealertotal+10 < 22 THEN de
alertotal=dealertotal+10 : pace=pac
e+1 : dealeraces=dealeraces-1 : GOT
O 450
430 GOSUB 920
440  SOUND 1,90,5 : SOUND 2,70,10 :
 x=x+90 : GOSUB 570 : REM otra cart
a
450 WEND
460 IF pace<>0 AND dealertotal >21
AND dealeraces<>0 THEN dealertotal=
dealertotal-10 : pace=pace-1 : GOTO
 390
```


```
465 LOCATE 8,2 : PEN 2 : PRINT deal
ertotal
470 REM ******* quien gana ? ******
*
480 IF dealertotal > playertotal AN
D dealertotal < 22 THEN LOCATE 17,1
2 : PRINT"YO GANO !!!     "
490 IF dealertotal < playertotal OR
 dealertotal >21 THEN LOCATE 12,12
: PEN 3 : PRINT "TU GANAS ";CHR$(16
3);bet*2 : money=money+bet*2
500 IF dealertotal = playertotal TH
EN LOCATE 15,12 : PRINT"* EMPATE *
   " : money=money+bet
510 IF money < 1 THEN GOTO 1030
520 FOR suite=0 TO 3 : FOR c=1 TO 1
3
530  pack(suite,c)=0          : REM n
ueva baraja
540 NEXT c : NEXT suite
550 LOCATE 18,23 : PEN 1 : PRINT "
< Pulsa espacio >"
560 IF INKEY$="" THEN 560 ELSE CLS
: GOTO 140
570 REM ******* cartas *******
580 PEN 1 : MOVE x,y-10 : DRAW x,y+
100 : DRAW x+75,y+100 : DRAW x+75,y
-10 : DRAW x,y-10 : TAG : MOVE x,y+
90
590 suite=INT(RND*4) : c=INT(RND*13
)+1 : REM selecciona carta al azar
600 IF suite=0 THEN s=226 : REM clu
bs
610 IF suite=1 THEN s=227 : REM dia
mantes
620 IF suite=2 THEN s=228 : REM cor
azones
630 IF suite=3 THEN s=229 : REM esp
adas
640 IF pack(suite,c)=1 THEN 590
650 IF pack(suite,c)=0 THEN pack(su
ite,c)=1
660 IF c>10 THEN cardvalue=10 ELSE
cardvalue=c
670 IF player=1 THEN playertotal=pl
ayertotal+cardvalue : IF c=1 THEN p
layeraces=playeraces+1
680 IF player=0 THEN dealertotal=de
alertotal+cardvalue : IF c=1 THEN d
ealeraces=dealeraces+1
690 IF start=1 THEN startsuite=s :
startcard=c : start=0 : RETURN
700 MOVE x+2,y+95 : PRINT card$(c);
710 IF c=10 THEN MOVE x+42,y+8 : PR
INT card$(c);: ELSE MOVE x+58,y+8 :
 PRINT card$(c);
720 IF c=1 THEN MOVE x+32,y+50 : PR
INT CHR$(s);: REM ace
730 IF c=2 THEN MOVE x+30,y+40 : PR
```

# Serie Oro

```
INT CHR$(s);: MOVE x+30,y+60 : PRIN
T CHR$(s);
740 IF c=3 THEN MOVE x+30,y+70 : PR
INT CHR$(s);: MOVE x+30,y+50 : PRIN
T CHR$(s);: MOVE x+30,y+30 : PRINT
CHR$(s);
750 IF c=4 THEN MOVE x+15,y+70 : GO
SUB 860 : MOVE x+15,y+30 : GOSUB 86
0
760 IF c=5 THEN MOVE x+15,y+70 : GO
SUB 860 : MOVE x+15.y+30 : GOSUB 86

0 : MOVE x+32,y+50 : PRINT CHR$(s);
770 IF c=6 THEN MOVE x+15,y+70 : GO
SUB 860 : MOVE x+15,y+50 : GOSUB 86
0 : MOVE x+15,y+30 : GOSUB 860
780 IF c=7 THEN MOVE x+15,y+70 : GO
SUB 860 : MOVE x+15,y+50 : GOSUB 86
0 : MOVE x+15,y+30 : GOSUB 860 : MO
VE x+32,y+50 : PRINT CHR$(s);
790 IF c=8 THEN MOVE x+15,y+70 : GO
SUB 860 : MOVE x+15,y+50 : GOSUB 86
0 : MOVE x+15,y+32 : GOSUB 860 : MO
VE x+32,y+40 : PRINT CHR$(s);: MOVE
 x+32,y+60 : PRINT CHR$(s);
800 IF c=9 THEN MOVE x+15,y+75 : GO
SUB 860 : MOVE x+15,y+25 : GOSUB 86
0 : MOVE x+15,y+58 : GOSUB 860 : MO
VE x+15,y+42 : GOSUB 860 : MOVE x+3
2,y+50 : PRINT CHR$(s);
810 IF c=10 THEN MOVE x+15,y+75 : G
OSUB 860 : MOVE x+15,y+25 : GOSUB 8
60 : MOVE x+15,y+58 : GOSUB 860 : M
OVE x+15,y+42 : GOSUB 860 : MOVE x+
32,y+40 : PRINT CHR$(s);: MOVE x+32
,y+60 : PRINT CHR$(s);
820 IF c=11 THEN MOVE x+15,y+45 : P
RINT CHR$(141);CHR$(140);CHR$(142);
: MOVE x+20,y+45 : DRAW x+40,y+70 :
 DRAW x+59,y+45 : MOVE x+32,y+55 :
PRINT CHR$(s);
830 IF c=12 THEN MOVE x+15,y+37 : P
RINT CHR$(131);CHR$(131);CHR$(131);
: MOVE x+15,y+55 : PRINT CHR$(197);
CHR$(238);CHR$(199);: MOVE x+32,y+7
0 : PRINT CHR$(s);
840 IF c=13 THEN MOVE x+15,y+35 : P
RINT CHR$(131);CHR$(131);CHR$(131);
: MOVE x+15,y+50 : PRINT CHR$(203);
CHR$(203);CHR$(203);: MOVE x+32,y+6
5 : PRINT CHR$(s);
850 TAGOFF : RETURN
860 PRINT CHR$(s);" ";CHR$(s);: RET
URN
870 REM ***** voz del ordenador ***
**
880 FOR t=1 TO LEN(a$) : PRINT MID$
(a$,t,1);
890  FOR delay=0 TO 30 : NEXT delay
900 SOUND 1,20+INT(RND*10),3
910 NEXT t : RETURN
920 REM ******* retardo *******
930 FOR delay=0 TO 700 : NEXT delay
 : RETURN
940 REM ******* pontoon ! *******
950 LOCATE 13,12 : PEN 3 : PRINT "P
ONTOON !!  BONUS ";CHR$(163);bet
960 money=money+bet
970 FOR loop=1 TO 3
980  FOR note=70 TO 30 STEP -10 : S
OUND 1,note : SOUND 2,note+loop*5,7
990  NEXT note : SOUND 1,70
1000 NEXT loop
1010 FOR delay=0 TO 3000 : NEXT del
ay
1020 LOCATE 11,12 : PRINT SPC(25) :
 GOSUB 920 : RETURN
1030 REM ***** sin dinero *****
1040 FOR t=1 TO 3 : GOSUB 920 : NEX
T
1050 CLS : LOCATE 5,12 : PRINT"Te q
uedaste sin dinero !!"
1060 LOCATE 1,23 : GOSUB 920
1070 IF autodestruct$="ON" THEN CAL
L 0 : REM Adios !!
1080 LOCATE 5,18 : PRINT"Otro juego
? Y/N";:INPUT answer$
1085 answer$=UPPER$(answer$)
1090 WHILE answer$<>"Y" AND answer$
<> "N":WEND
1100 IF answer$="N" THEN END
1110 money = 10:INK 0,0: INK 2,18:
GOTO 120
```

*Para que tus dedos no realicen el trabajo duro, M.H. AMS-TRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.*

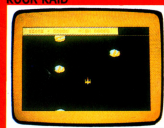
# microbyte PRESENTA... AMSTRAD

## NUEVOS PROGRAMAS EN CASSETTE Y DISCO

### ARGO NAVIS

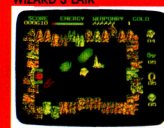
El comandante de nave AMSTRAD 1 se encuentra atrapado en las profundidades de una central nuclear y debe salir con vida. Excelentes gráficos y sonido. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

### ROCK RAID



Debes pilotar con acierto la nave que a lo largo de su viaje galáctico sufrirá encuentros con meteoritos, residuos planetarios, etc. Gran movilidad y excelentes efectos. **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

### WIZARD'S LAIR



Te encuentras atrapado en las profundidades de una caverna, llena de obstáculos, adversidades, etc. ¿Serás capaz de salir con vida? **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

### MACADAM FLIPPER

Arcade, programa que nos trasladará al mundo de la máquina-flipper del mejor casino de Las Vegas. Posibilidad de creación del tablero, puntuaciones, etc. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

### JUMP JET

Te encuentras a los mandos de la nave "Aircraft". En una perfecta maniobra debes despegar del portaviones. (Excelente versión simulador vuelo-combate). **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

### MUSIC MAESTRO

El más completo programa de musica creado para el AMSTRAD. Permite crear sonidos, melodías y convertir tu ordenador en la mejor "caja de música". **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

### PAZAZZ

Programa que permite de una manera sencilla la creación de pantallas con gráficos, dotaries de movimiento, acompañados de música. **P.V.P.: DISCO 2.900 pts.**

### SYCLONE 2

Programa de utilidad que permite realizar copias de seguridad (back-ups) a distintas velocidades (baudios). **P.V.P.: CASSETTE 1.800 pts. DISCO 2.500 pts.**

### ZEDIS II

Editor-desensamblador del Z-80, para el programador más avanzado. **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

### SYSTEM X

Ampliación del lenguaje Basic. Conjunto de 30 nuevas instrucciones (fill, circle, protect) para ayudar en la programación. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

### ODDJOB

La mejor utilidad para el mejor conocimiento del Disco. (Copias de disco, Disk map, Disk track, sector, etc.) **P.V.P.: DISCO 2.600 pts.**

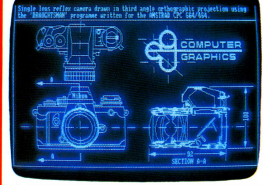
### TRANSMAT

Pasar los mejores programas de cinta a disco ya no es problema. Con Transmat este proceso será fácil y sencillo. **P.V.P.: DISCO 2.600 pts.**

### OTROS PROGRAMAS EN STOCK

| | |
|---|---|
| MIÑI OFFICE | P.V.P. CASS. 3.200 pts. P.V.P. DIS. 3.900 pts. |
| WORLD CUP FOOTBALL | P.V.P. CASS. 1.800 pts. |
| BATLE FOR MIDWAY | P.V.P. CASS. 1.800 pts. |
| FIGHTER PILOT | P.V.P. CASS. 2.200 pts. |
| SURVIVOR | P.V.P. CASS. 1.800 pts. |
| MOON BUGGY | P.V.P. CASS. 1.800 pts. |
| TECHNICIAN TED | P.V.P. CASS. 1.800 pts. |
| FRUITY FRANK | P.V.P. CASS. 1.800 pts. |
| DATABASE | P.V.P. CASS. 2.100 pts. |
| LOGO TURTLE GRAPHICS | P.V.P. CASS. 2.400 pts. |
| TASCOPY Y TASPRINT | P.V.P. CASS. 2.600 pts. |
| FONT EDITOR | P.V.P. CASS. 1.900 pts. |

### DRAUGHTSMAN




Sofisticado programa de dibujo que permite tratar la pantalla del AMSTRAD como un complejo tablero de dibujo, sus resultados son espectaculares. **P.V.P.: CASSETTE 4.500 pts. DISCO 5.200 pts.**

ENVIENOS A **MICROBYTE** S.A.

P.º Castellana, 179, 1.º - 28046 Madrid

Nombre

Apellidos

Dirección

Población

D.P. Teléfono

**ENVIOS GRATIS**

| JUEGO | C | D | Precio | TOTAL |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**PRECIO TOTAL PESETAS**

Incluyo talón nominativo ☐

Contra-Reembolso ☐

Pedidos por teléfono **91 - 442 54 33 / 44**

# CONTABILIDAD

***Esta semana presentamos en Banco de Pruebas otro programa de contabilidad, adaptado también al Plan Contable Español y que destaca fundamentalmente por dos cosas:***

***1. Es increíblemente sencillo de usar.***
***2. Puede emplearse con una o dos unidades de disco.***

***Para analizarlo, seguiremos manteniendo el mismo enfoque adoptado con el Placon, sin pretender por ello entrar en comparaciones, siempre odiosas y en este caso más, porque cada uno de los programas cumple su función dentro del hueco que pretende llenar. Sin más preeámbulos, vamos a pasar a describir detalladamente el programa:***

La aplicación de Contabilidad, pretende que de una manera sencilla tengamos una contabilidad completa y al día, adaptada al Plan General Contable, y de la que se obtiene toda la información necesaria para el análisis contable, pudiéndose obtener incluso, ratios y porcentajes de los gastos e ingresos, sumamente informativos del estado contable.

## Equipo necesario

—AMSTRAD CPC-464
—MONITOR (F. VERDE O COLOR)
—UNA O DOS UNIDADES DE DISCO
—IMPRESORA *(OPCIONAL)*

Una vez cargado el programa, dos cosas nos pueden suceder, según sea la primera vez que utilizamos el programa, o no.

**a)** Si es la primera vez, entramos en el apartado **«Proceso de Instalación»**, donde seremos interrogados sobre el nombre de la empresa, que ulitizará el programa para imprimir en la cabecera de los listados. Procura no equivocarte, ya que no habrá posibilidad de modificarlo a lo largo del programa.

**b)** No es la primera vez, el nombre de la empresa ya está **«guardado»** en un archivo, por lo tanto no aparecerá más el proceso anterior, se entrará directamente en la aplicación.

Las funciones del programa de Contabilidad están divididas en seis apartados, que son:

1. ASIENTOS CONTABLES
2. PLAN CONTABLE
3. LISTADO DE CUENTAS DE MAYOR.
4. BALANCE DE SUMAS Y SALDOS
5. CUENTA DE RESULTADOS
6. CIERRE PERIODICO

Para escoger una u otra opción utilizaremos las flechas del cursor (↑↓), y a continuación pulsaremos la tecla 'ENTER', dando así paso, a la opción deseada.

A esta colección de opciones denominamos Menú, y a este primero, **«Menú Principal»**; la adopción de un determinado proceso, normalmente, da paso a otro menú con otra serie de opciones.

Para regresar o salir de un menú al anterior **(Menú Principal)**, se pulsará siempre la tecla del ángulo superior izquierdo del teclado 'ESC', utilizada desde el menú principal, saldremos de la aplicación de Contabilidad al sistema operativo, dando por concluido el trabajo.

## Asientos contables

En esta opción están recogidos todos los procesos a realizar con los asientos de la Contabilidad.

Al entrar en este capítulo nos aparecerá en la pantalla, otro menú, con los siguientes apartados:

ENTRADA DE ASIENTOS
CONSULTA DE ASIENTOS
LISTADO DEL LIBRO DIARIO

Y, por supuesto, con la tecla 'ESC' salimos de este menú para volver al principal, redundando en lo anteriormente expuesto.

ENTRADA DE ASIENTOS

Consiste en la grabación de las partidas contables, como información de base para la elaboración de la totalidad de documentos contables que se pueden obtener con este programa.

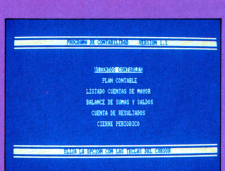
Los datos que la pantalla nos pide son:

P.No. FECHA CUENTA CONCEPTO IMPORTE D/H

**P.No.** o Número de Partida es el número de orden de los apuntes, que el ordenador asigna automáticamente a cada uno, empezando desde 1.

**FECHA.** El primer dato, por tanto, que debemos aportar, es la fecha del apunte. Esta fecha puede corresponder, o bien a la fecha en que se produjo el hecho contable, o bien, a la fecha en que se registra el asiento, (ya que no tiene por qué ser el mismo día que pagamos una factura, y el día que se los decimos al ordenador). Una u otra fecha son igualmente válidas.

**CUENTA.** Previamente a la entrada de asientos se ha de INTRODUCIR el PLAN CONTABLE, es decir, las cuentas a las que vamos a asignar las partidas que vamos anotando.



Si no ha sido definida previamente cuenta alguna, aparecerá en pantalla el mensaje: **«NO HAY CUENTAS DEFINIDAS»,** y no se podrán introducir asientos tanto en cuanto no realicemos dicha función.

**CONCEPTO.** Descripción de la naturaleza del asiento. Admite todas las teclas del ordenador *(mayúsculas, minúsculas, números y signos)*, con un total de 12 caracteres como máximo para definir el apunte. P. Ej.: «Fra.A/123-85», etc.

**IMPORTE.** Cantidad que importa el asiento. Sólo admite dígitos en un total de nueve: 999.999.999; y el signo «-», no se deben poner los signos de puntuación, ya que tras escribir la cantidad y pulsar 'ENTER', éstos aparecen en su sitio.

**D/H.** DEBE o HABER, se consigna **«D»** o **«H»**, *(en Mayúsculas)*, según deseemos que el asiento sea deudor o acreedor.

CONSULTA DE ASIENTOS

Listado en pantalla de los asientos, discriminados entre dos fechas.

Se pedirá una fecha inicial, y otra final, encuadrando así los asientos a consultar en un período de tiempo.

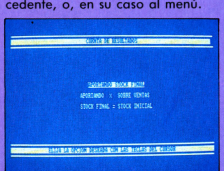
LISTADO DEL LIBRO DIARIO

Al igual que el anterior proceso, pero esta vez sobre el papel, el listado del libro diario imprimirá los asientos limitados por dos fechas dadas. Pulsando 'ENTER', se asumirá la fecha dada en «Fecha Inicio del Listado».

A continuación, se solicita el número de la primera página. Pulsado 'ENTER', el listado comenzará con el número de página del último diario, más uno.

Un último mensaje aparece en la pantalla: «PREPARE LA IMPRESORA», dándonos así tiempo para encender la impresora, (que ya debería estarlo), colocar el papel al principio de un hoja, etc. Pulsando cualquier tecla, comienza el listado.

Abundando en lo mismo, si pulsas 'ESC', se volverá a la cuestión precedente, o, en su caso al menú.




## Plan contable

Este apartado recoge todas las funciones a realizar con el Plan de Cuentas.

Tenemos el siguiente menú:

ALTA DE CUENTAS
MODIFICACION DE CUENTAS
CONSULTA DE CUENTAS
LISTADO DEL PLAN CONTABLE
CLASIFICACION DE CUENTAS

Como ya se dijo en el apartado de Entrada de Asientos, el primer proceso a realizar con el programa, será la definición de las cuentas, es decir...

ALTAS DE CUENTAS

Permite la grabación en el archivo, de todas las cuentas que conforman el Plan. Se deben tener en cuenta dos condiciones:

**a)** Cuentas a grabar han de tener 5 dígitos, es decir, se grabarán directamente, las subcuentas, pues los grupos, subgrupos y cuentas, los asume el programa. Esto permite que para cada cuenta tengamos hasta 100 subcuentas.

| | | |
|---|---|---|
| P. Ej.: Cuenta | 430 | Cliente |
| Subcuentas: | 43000 | Cliente A |
| | 43001 | Cliente B |
| | 43002 | Cliente C |
| | ..... | ..... |
| | 43099 | Cliente X |

**b)** Conveniente ajustarse a las cuentas definidas por el Plan General Contable, dado que el programa utiliza este mismo orden para la elaboración del Balance y la cuenta de Resultados.

Cuando se digita un número de cuenta que ya existe en el archivo, el mensaje **«LA CUENTA nnnnn YA EXISTE»**, nos avisa del hecho, no pudiendo definir cuentas con el mismo número.

El título de la cuenta, tendrá como máximo 12 caracteres, de capacidad, pudiéndose utilizar letras, números y signos. El título ha de reflejar el sentido o finalidad de la cuenta.

Pulsando 'ESC' se pasa nuevamente a introducir el número de cuenta, si no estamos conformes con él, y otro 'ESC' provoca la salida del menú.

MODIFICACION DE CUENTAS

Cuando nos percatamos de que el título de una cuenta no está bien escrito, o simplemente queremos cambiar su denominación a otra más exacta, recurriremos a este proceso, que es casi idéntico al anterior. Se requiere el número de cuenta a modificar, (recuerde que se utilizan 5 dígitos), para que el ordenador la busque en el archivo. Si no la encuentra, (no ha sido dada de Alta), el cursor vuelve a la petición de cuenta. Por el contrario, si la cuenta existe, el título aparece a continuación, y sobre la primera letra, el cursor, para poder modificar.


Se avanza con las flechas del cursor (→ ←) si se quiere mantener algún caracter.

Pulsando 'ESC', se retorna sin grabar la modificación. Pulsando 'ENTER', el título modificado sustituirá al anterior.

CONSULTA DE CUENTAS

Listado por pantala de parte o la totalidad del Plan Contable, así como la suma de DEBE, HACER ySALDO.

El listado puede obtenerse, o bien de todas las cuentas, pulsando 'ENTER', al principio del listado, o bien se podrán listar las subcuentas de un determinado grupo (pulsando el número del grupo), o a nivel de subgrupo (pulsando las dos cifras del mismo), o también a nivel de cuenta (tres dígitos) o por supuesto, de una sola subcuenta, pulsando los cinco dígitos de la misma.

LISTADO DEL PLAN CONTABLE

Para tener un listado impreso de las cuentas, pero en papel, y sin los saldos, únicamente el número de cuenta y título.

Nos pedirá fecha de edición, para salir impresa en el listado.

CLASIFICACION DE CUENTAS

Este proceso debe realizarse forzosamente siempre que se produzcan altas en el Plan Contable. De no realizase, las cuentas dadas de alta después de la última clasificación, no podrán ser usadas, como si no existieran.

## Listado de cuentas de mayor

Para obtener el listado de asientos de una determinada cuenta y en de-

terminada fecha, utilizaremos este proceso, en dos opciones diferentes:

LISTADO POR PANTALLA
LISTADO POR IMPRESORA

Como su nombre indica, los dos apartados realizan la misma función, con la salvedad de que el primero muestra el resultado en pantalla, (para ir confeccionando el Libro-Mayor).

Para ambas opciones, el prólogo es idéntico:

Primero, petición de la cuenta a listar, *(5 dígitos, cuentas que existan en el Plan de cuentas, etc.)*.

A continuación, fecha inicial y fecha final, que ya vimos en el apartado Consulta de Asientos, para listar sólo aquellos movimientos de un período.

Por último, para la opción de impresora, se solicita la también conocida, **«Fecha de Edición».**

### Balance de sumas y saldos

Para la obtención de listado del Balance de Sumas y Saldos, en la impresora.

En este caso no hay ninguna opción a tomar, tan sólo se requiere la **«Fecha de Edición».**

### Cuenta de resultados

Documento de análisis de resultado económico del período. Para su cálculo se necesitan algunos datos que hemos de aportar, y que según éstos, habrá distintas maneras de realizarlo, por lo que tampoco el resultado final será el mismo.

Estas opciones son:

APORTANDO STOCK FINAL
APORTANDO PORCENTAJES SOBRE VENTAS
STOCK FINAL = STOCK INICIAL

APORTANDO STOCK FINAL

Eligiendo esta opción, el análisis, así como el resultado obteniendo es el más correcto de los tres posibles. Sabiendo el inventario final (inventario = stock = existencias), el márgen bruto obtenido, será el margen real que estamos cargando a los productos que vendemos, y el resultado, exacto.

APORTANDO EL PORCENTAJE SOBRE VENTAS

Si lo que conocemos es el margen que cargamos sobre los productos, el cálculo se hace a la inversa, siendo entonces el stock final calculado, solamente una aproximación,, (téngase en cuenta que aun teniendo un margen idéntico para todos los productos que vendemos, se realizan descuentos, se pierde mercancía, etc.).

STOCK FINAL = STOCK INICIAL

Cuando la empresa no tiene stock, (o resulta imposible conocerlo), y son bastante variables los márgenes aplicados a los productos, o simplemente se trata de una empresa de servicios (sin existencias ni márgenes), con este proceso se obtiene un resultado no muy exacto (si realmente hay existencias), o bastante aproximado si se trata de una empresa como la mencionada.

*NOTA: Recuérdese que para la consecución de la Cuenta de Resultados, es conveniente que todo los asientos a cuentas de ingresos (Grupo 7), salvo las ventas (Cuentas del Subgrupo 70), se realicen sobre cuentas del Subgrupo 74 (Ingresos Financieros).*

### Cierre periódico

Consiste en la anulación de todos los asientos del disco, pero manteniendo los saldos de las cuentas. De esta manera tendremos una capacidad ilimitada de apuntes en el archivo.

Es conveniente realizar previamente una «Copia de Seguridad» del disco de asientos, para no perderlos, y así recurrir a ello cuando sea preciso. Así por ejemplo, si realizamos el cierre cada trimestre, utilizaremos siempre el mismo disco de trabajo, pero tendremos un total de 4 discos con los asientos de cada trimestre.

# Banco de pruebas

## FICHA DEL PROGRAMA CONTABILIDAD

Comercializado por: **Indescomp.**
Soporte: **Disco** *(una o dos unidades).*
Sistema operativo: **CP/M** *(una o dos unidades).*
Compatible: **CPC464, CPC664, CP6128.**
Precio: **14.500** pts.


## MICROHOBBY AMSTRAD SEMANAL

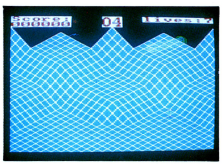
LE OFRECE AHORA SUS PROGRAMAS YA GRABADOS, PARA QUE VD. NO TENGA QUE TECLEARLOS

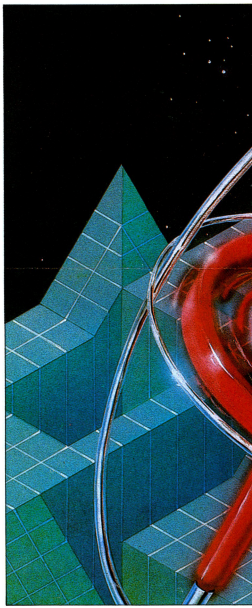
Todos los programadores y aficionados a la microinformática sabemos lo tedioso y propenso a errores que resulta el teclear un listado de un programa. Para facilitar tu labor al máximo y que no tengas que estar horas sobre el teclado de tu ordenador tratando de descifrar incomprensibles mensajes de error, **AMSTRAD** SEMANAL te ofrece cada mes los programas publicados de los cuatro números correspondientes en una cinta de cassette, sólo por **675 ptas. (sin más gastos por envío).**

**Envíanos con la menor demora posible, el cupón correspondiente.**

# GYROSCOPE

***Dotado de una extraña energía cinética, el incansable mecanismo giroscópico, circula por los planos multiformes de una construcción espacial, en la que el mínimo error de trayectoria nos precipita a abismos insondables.***



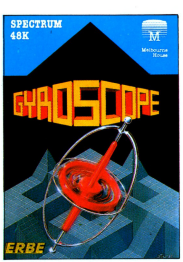


Todos conocemos el soberbio juego de Atari, para máquinas de bares y billares, en el cual dirigimos una bola por pasadizos y pasillos elevados, en busca de las distintas metas que culminan las sucesivas fases del juego.

Melbourne House, aprovechando la brillante idea de Atari, realiza un juego muy parecido, con la salvedad de que en vez de dirigir una bola, conducimos una peonza.

Nos aguarda el paisaje geométrico y frío de una ciudad espacial sin habitantes, solamente algunos seres hostiles se interponen en nuestro camino.

A pesar de todo, nuestro principal enemigo somos nosotros mismos.




COMPATIBLE
CPC 464
CPC 664
CPC 6128

El artilugio mecánico que dirigimos, está dotado de una fuerza centrífuga, que hace bastante difícil el control del mismo, provocando la salida de la ruta que debemos seguir.

Cualquier salida de ésta es fatal.

Los planos horizontales que constituyen el terreno por el que debemos circular, están bordeados por insondables abismos por los que cae nuestra peonza.

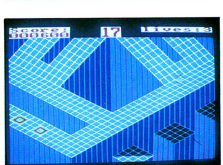
Nuestro único objetivo, es superar el mayor número de fases posibles, incrementando de esta manera nuestro marcador de puntos.

La dificultad de cada recorrido, aumenta progresivamente, según se van superando sucesivas fases.

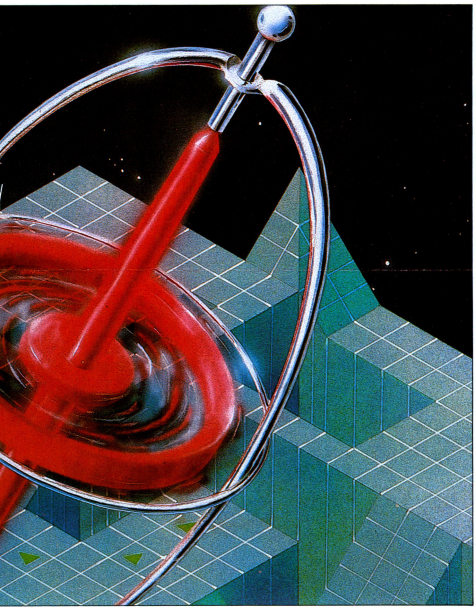
Cosa que incita a superar pantalla por pantalla, con el único propósito de saber cómo es la siguiente y las sorpresas que nos depara.

La conducción del aparato giroscópico, se puede realizar mediante joystick o teclado, siendo este último el que permite tener mayor control de la dirección.

El joystick, por muy bueno que sea para juegos de acción trepidante, no es el idóneo en aquéllos en los que la precisión y la delicadeza son piezas clave.



# Mr. Joystick



Utilizando el teclado, se puede graduar la inercia hacia cierta dirección. Con leves toques o con enérgicas pulsaciones de tecla, podemos obtener movimientos suaves o bruscos cambios de dirección.

Las direcciones de movimiento a las que obedece el giroscópico, son las clásicas; arriba, abajo, izquierda y derecha, pero éstas no son exactamente las direcciones en las que nos tenemos que mover.

Los planos de circulación, generalmente se encuentran en posición oblicua a las direcciones indicadas anteriormente. Razón por la cual, caer por los bordes se hace demasiado frecuente.

Una técnica muy útil para superar este problema, es pulsar simultáneamente dos teclas. Si queremos movernos en la dirección que va hacia el borde superior izquierdo de la pantalla, basta con pulsar a la vez las teclas arriba e izquierda, cosa que con el joystick sería imposible.

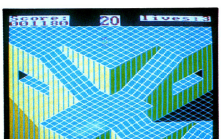
La parte más vistosa del juego, son los gráficos que constituyen las superficies por las que circula nuestro artilugio, siendo el mayor atractivo del juego.

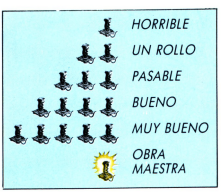
Están realizadas con unos gráficos en tres dimensiones, que simulan perfectamente la inclinación de los distintos planos, diferenciando claramente las zonas horizontales de aquellos planos que tienen una cierta pendiente.

Para conseguir este efecto, se ha recurrido a la utilización de una trama cuadriculada sobre las zonas horizontales y demás planos con pendiente, mientras que los planos verticales están tratados solamente con rayado vertical y distinto color de fondo, acentuando de esta manera la diferencia entre ambos.

La utilización de esta técnica permite un efecto tridimensional muy real, creando una auténtica sensación de volumen.

Un juego, que como su antecesor de las máquinas de bares y billares es totalmente adictivo, llegando a constituir una verdadera obsesión pasar a la siguiente pantalla.






# UN "MILLON" EN JUEGO

**Gráficos a color, rapidez de acción y sonido dan a este juego la sensación de una máquina de millón real.**

**P.V.P.**
**1.900 (cinta)**
**2.900 (disco)**


ACE

ACTIVIDADES COMERCIALES ELECTRONICAS, S.A.
Tarragona, 110. Tel. 325 10 58* 08015 Barcelona. Telex 93133 ACEE E

YA DISPONIBLE EN El Corte Inglés Y EN TODAS LAS TIENDAS ESPECIALIZADAS

# Cómo realizar un programa test

**Análisis**

***Los ordenadores sirven para muchas cosas: de acuerdo. Una de ellas, curiosa y útil, os la presentamos en Amstrad análisis esta semana. Se trata de un programa muy sencillo que permite implementar los típicos test (carnet de conducir) en los que se plantean 4 respuestas posibles a una pregunta y sólo una de ellas es cierta. Como lo importante es la idea, las preguntas y respuestas están en forma de DATAs, pero una aplicación más realista requeriría que el usuario introdujera preguntas, respuestas y cuál es la correcta mediante INPUTs y archivar todo eso en fichero de disco, que el programa principal podría posteriormente utilizar siempre. De todas maneras, el método sería más o menos así:***

**10-30** Titulo del programa.

**40-50** Número de preguntas y dimensionado de las matrices que guardan las preguntas *(preguntas)*, las respuestas correctas *(respuestaS)*, las respuestas al azar incorrectas *(aleatoriaS)* y los números que indican cuál de las 4 respuestas que aparecerán en pantalla es la válida *(número)*.

**60** Borrar pantalla y llamar a la subrutina que introduce datos en las matrices.

**80** Bucle principal sin fin. Una vez más, nos aprovechamos de que la variable **«fin»** siempre será cero y la condición del WHILE será siempre falsa.

**90** Imprime el test en pantalla.

**100** Pide la respuesta mediante un INPUT. No se comprueban errores; se supone que daremos un número dentro de rango.

**110** Se comprueba la validez de la respuesta, comparando el número respondido con el valor de la matriz de respuestas acertadas número(x).

**120** Repetición de la jugada.

**130** Fin del programa.

**140-120** Introduce valores en las matrices mediante la orden READ. No merece mayor comentario.

**230** Calculo de la pregunta. X varía de 1 a 5, a MAXNUM.

**250-280** coloca la respuesta correcta en el lugar adecuado comprobando cuándo «orden» es igual a número(x). Recordemos que en esta matriz se almacenaban los números que corresponden al lugar correcto donde aparecerá la respuesta válida y hay una correspondencia biunívoca entre esta matriz y respuestaS(x). Si el número no concuerda, se elige al azar una respuesta falsa, de «despiste».

**300-320** Respuestas correctas.

**340-370** Respuestas de **«despiste»** que se escogen al azar.


```
 0 REM ****************
 20 REM * PROGRAMA TEST *
 30 REM ****************
 40 MAXNUM=5
 50 DIM pregunta$(MAXNUM),respu
 esta$(MAXNUM),aleatoria$(MAXNU
 M*4),numero(MAXNUM)
 60 CLS:GOSUB 140:REM * inicial
 izar *
 70 REM * bucle principal *
 80 WHILE NOT fin
 90 GOSUB 220
 100 INPUT"Cual es la respuesta
 (1-4)";numresp
 110 IF numresp<>numero(x) THEN
  PRINT"Error.La respuesta corr
 ecta es: "respuesta$(x) ELSE
 PRINT"correcto"
 120 WEND
 130 END
 140 REM * inicializar matrices
  *
 150 FOR n=1 TO MAXNUM
 160 READ pregunta$(n),respuest
 a$(n),numero(n)
 170 NEXT n
 180 FOR i=1 TO MAXNUM*4
 190 READ aleatoria$(i)
 200 NEXT i
 210 RETURN
 220 REM * imprimir test *
 230 x=INT(1+RND (1)*MAXNUM)
 240 PRINT "dime un "pregunta$
 (x)"s"
 250 FOR orden=1 TO 4
 260 y=INT(1+RND (1)*20)
 270 IF orden=numero(x) THEN PR
 INT orden;SPC(2); respuesta$(x
 ) ELSE PRINT orden;SPC(2);alea
 toria$(y)
 280 NEXT orden
 290 RETURN
 300 REM * respuestas correctas
  *
 310 DATA nombre de flor,rosa,3
 ,elemento quimico,vanadio,1
 320 DATA personaje historico,h
 itler,3,pais antiguo,egipto,4,
 avion americano,huracrane,1
 330 REM * respuestas aleatoria
 s *
 340 DATA segmento,razon,linea,
 terraza,peso
 350 DATA casa,hombre,feliz,avi
 on,merluza
 360 DATA azul,verde,rojo,amari
 llo,neon
 370 DATA database,calendario,h
 otella,libro,programa
```

*Para que los dedos no realicen el trabajo duro, M. H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un casete mensual, solicítanoslo.*

# EL AMSTRAD Y LAS CADENAS

J.J. MARTIN

***Dos son los tipos de variables que podemos encontrar en un programa basic, variables numéricas y alfanuméricas o literales. En esta ocasión vamos a ocuparnos de las alfanuméricas, así como de todas las instrucciones que se relacionan de forma directa, con el tratamiento de ellas.***

Si la versión de Basic que utiliza CPC *(Locomotive)*, es capaz de provocar el encantamiento de cualquier Basic-adicto, el tratamiento que de las cadenas alfanuméricas realiza, no podría por menos, que sorprendernos agradablemente, ante su versatilidad y potencia de uso. Comenzemos dando unos, detalles en el uso general de las variables alfanuméricas, para posteriormente acabar por sumerginos de lleno en el tratamiento de éstas.

## ¿Cómo se usan?

El uso de una variable alfanumérica debe ser siempre advertido al procesador de nuestra máquina. Así, si ejecutamos x=**"palabra"**, obtendremos inmediatamente una detención con el informe **"TYPE MISMACHT"**. Lo que viene a significar que el tipo de variable que esperaba, numérico en este caso, no coincide con el recibido, alfanumérico. Dos son las formas que tenemos que realizar esta advertencia. La primera, segurámente será la que más use, así como lo que más se encuentre en otros programas, consiste en colocar tras el nombre de la variable el signo $.

Ahora si hiciésemos x$=**"palabra"**, no obtendríamos queja alguna de nuestro fiel CPC. Observe que siempre que hagamos uso de una cadena alfanumérica, ésta deberá ir encerrada entre comillas, salvo en el caso de las sentencias REM, en las que puede omitirlas. Si suprimiésemos éstas, el ordenador intentaría encontrar una variable con ese nombre. Por ejemplo:

```
10 PALABRA=34
20 PRINT PALABRA
```

Nos daría como resultado en pantalla 34. Mientras que:

```
10 PALABRA=34
20 PRINT "PALABRA"
```

Daría como resultado PALABRA.

La segunda forma de marcar una variable con la etiqueta de literal, consiste en hacer una declaración previa mediante la instrucción DEFSTR. Haciendo uso de esta orden podemos suprimir la $. Así, sí ejecutamos:

```
DEFSTRS X
```

provocamos que todas las variables que empiezen por X o x, sean tratadas como variable literal. Así, después de esta declaración:

```
x="PALABRA"
```

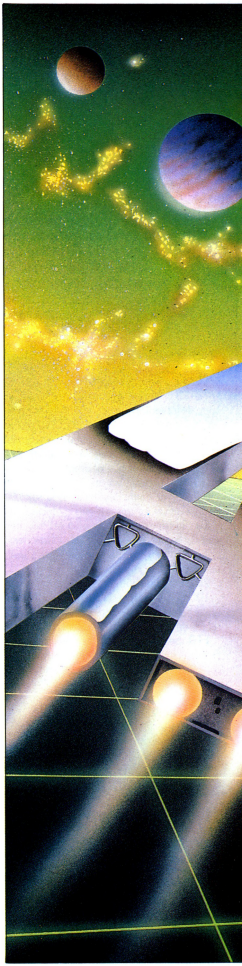
sería perfectamente válido. El único problema, ahora, es que no podrá utilizar como variable numérica, ningún nombre que comience por x o X. Existe una forma más sofisticada de usar esta orden, que consiste en utilizar dos letras separadas por un guión. Por ejemplo:

```
DEFSTRS D-L
```

provoca que todas las variables cuyos nombres comiencen por una letra comprendida entre la D y la L, sean tratadas como variables literales.

Mi consejo, es que, por lo menos al principio, empleé el sistema de etiquetar con $, lo que le facilitará la comprensión del programa, con un simple vistazo.

Lo último importante que puede decirse sobre este tipo de variables, es que el número máximo de caracteres que pueden albergar es de 255, el famoso número mágico que siempre ronda a nuestro alrededor por ser el mayor que puede contener una posición de memoria.




# ProgramAcción

### *Instrucciones de manejo de cadenas*

El grupo de instrucciones que se relacionan con el tratamiento de variables literales son:

| | |
|---|---|
| DEFSTRS | INSTRS |
| LEFT$ | LOWER |
| MID$ | RIGTH$ |
| STR$ | STRING$ |
| UPPER$ | VAL |
| SPACE$ | LEN |

Y además de éstas, las operaciones +, —, <, >, < >, =, pueden también ser usadas de un modo parecido a como se emplean con los números.

Para explicar el significado que dentro del lenguaje Basic tienen estas palabras, realizaremos un programa que nos ayudará a su comprensión. Siéntese delante de su ordenador y enciéndalo.

Comenzamos.

### *El programa del ahorcado*

El programa que vamos a realizar se basa en el conocido juego del ahorcado. Si no lo conoce, su descripción es muy sencilla. Un jugador piensa una palabra e informa al otro de su longitud mediante guiones, que deberá ir rellenando a medida que el oponente le vaya comunicando letras. De todos modos teclee el programa del cual le damos, al final del artículo, el listado completo, y úselo antes de intentar entender su lectura.

```
10 REM *EL AHORCADO*
20 REM *JJ MARTIN para MICRO-
       HOBBY AMSTRAD*
30 REM *          EL AHORCADO*
40 MODE O
```

Mediante esta orden, fijamos el tamaño de las letras al máximo disponible.

```
50          a$ = ''AHORCADO'':
       b$ = SPACE$ (15)
```

Cargamos la variable a$ con la palabra ahorcado, observe cómo etiquetamos la variable con $ y cómo la palabra AHORCADO va en-

tre comillas. El uso de este literal se comprenderá más adelante. La segunda asignación provoca la carga de la variable b$ con 15 espacios, mediante el uso de la función SPACE$. Esta nos servirá posteriormente para borrar un mensaje aparecido en pantalla.

```
60 errores=0
```

Esta variable numérica llevará la cuenta de los errores cometidos.

```
70 INPUT palabra$
```

Mediante esta línea, detenemos la ejecución del programa hasta que sea introducida la palabra que nuestro contrincante debe buscar.

```
80 CLS
90 palabras$=UPPER$ (palabras$)
```

Aquí encontramos una de las instrucciones de nuestro estudio UPPER$. Esta función transforma la variable alfanumérica encerrada entre paréntesis por la misma cadena, sustituyendo las letras minúsculas que encuentre por letras mayúsculas. Los caracteres que contenga la variable que no sean letras minúsculas no serán modificados. La función inversa de UPPER$ es LOWER$ que realiza la misma operación, pero, sustituyendo las mayúsculas por minúsculas.

La utilidad que en nuestro programa vamos a dar a esta función consiste en asegurarnos que vamos a comparar siempre mayúsculas. Tenga en cuenta que para el ordenador **"a"** es distinto de **"A"** lo cual comportaría un verdadero problema, a la hora de intentar hallar la solución. Ya que significaría, tener que averiguar no sólo la palabra, sino con qué letras, mayúsculas o minúsculas, fue escrita.

```
100 solución$=palabra$
110 lon=LEN (palabra$)
```

En la línea 100 cargamos solución$ con el contenido de palabra. Observe en este caso la ausencia de comillas. La razón de crear esta nueva variable es que la variable palabra$ la vamos a modificar durante la ejecución y hemos de asegurarnos de no perder la palabra buscada.

En la línea 110 nos encontramos con una nueva función, LEN su uso es muy simple, nos da la longitud de la variable encerrada entre paréntesis. Observe que LEN *(palabra$)* no es 8, es decir, la longitud de **"palabra$"**, si no que es la longitud de la cadena que alberga la variable cuyo nombre es palabra$. También podríamos hacer LEN *("palabra$")*, en este caso sí obtendríamos 8 como resultado, la explicación es que "palabra$" no es considerada como el nombre de una variable, sino como una cadena alfanumérica, por ir encerrado entre comillas.

```
120 trabajo$=STRING$ (lon, "—")
```

La función STRING genera una cadena alfanumérica de un caracter, el encerrado entre comillas, repetido tantas veces como indique el número que lo acompaña. Así STRING$ (10, "*") nos daría como resultado "**********"

En nuestro programa creamos la variable trabajo$ de longi-tud igual a la de palabra$ y formada por la repetición de "—". En esta variable iremos colocando las letras que vayamos acertando.

```
130 LOCATE 8,20: PRINT trabajo$
140 IF trabajo$=solución$ THEN
    GOTO 350
```

Escribimos trabajo$. Por el momento sólo es una cadena de guiones. En la línea 140 comparamos trabajo$ con solución$ y si es igual enviamos la ejecución a la línea 350 en la que haremos notar que se ha acertado la palabra.

```
145 LOCATE 1,23: PRINT "PULSE
    UNA TECLA"
    150 IF INKE$ < > "" THEN GOTO 150
    160 letra$=INKEY$
170 IF letra$="" THEN GOTO 160
    180 letra$=UPPER$ (letra$)
    185 LOCATE 1,23: PRINT b$
    185 LOCATE 1,23: PRINT ""
```

Mediante estas seis líneas hacemos una captura de datos, la tecla pulsada es leída en la línea 150, apuntada en la variable letra$ mediante la línea 160, y transformada en mayúscula por la sentencia UPPER$ de la línea 180. La línea 150 es un pequeño truco que simula la instrucción CLEAR INPUT sólo accesible para los poseedores del 664. La finalidad de ésta es obligarnos a levantar esto último no se preocupe, hoy no es el tema que nos atañe, en un próximo artículo hablaremos de ello con más detalle.

```
190 dicha$=dicha$+
    "—"+letra$
```

```
200 LOCATE 1,4: PRINT dicha$
```

Creamos una nueva variable, la variable dicha$, en ésta se irán apuntando las letras que se van pulsando. Observe con suma atención

la curiosa suma. Lo primero que puede destacarse es que estamos sumando cadenas alfanuméricas. Esto no debe asustarle, la definición de esta operación es tan sencilla como puede estar imaginando, un ejemplo será suficiente. **"ABC"+"DEF"+"GH"** nos daría como resultado la cadena **"ABCDEFGH"**. La segunda sorpresa que podemos llevarnos es la estraña igualdad: dicha$=dicha$+"—"+letra$. **¿Cómo algo puede ser igual a ese mismo algo más otra cosa?** El motivo de nuestra extrañeza se encuentra en considerar esto como una igualdad, consideración más que justificada por el empleo del =. Realmente estas dos barras horizontales no representan en este caso una igualdad, sino una asignación. En los principios del Basic era necesario poner LET lo cual permitía una diferenciación evidente entre la igualdad y la asignación, después, con el uso, se modificó y el empleo de LET quedó como algo puramente opcional y condenado a desaparecer. Lo que hacemos es cargar la variable dicha$ con lo que era dicha$ más el signo "—" más letra$, que recordemos, contenía la última tecla pulsada. Mediante la instrucción de la línea 200 escribimos en pantalla las letras que llevamos dichas.

```
210 posi=INSTR (1, palabra$, letra$)
220 IF posi=O THEN GOTO 290
```

En la línea 210 nos encontramos con una nueva instrucción. La misión de ésta es informarnos sobre si una cadena alfanumérica se encuentra contenida en otra. El primer número encerrado en el paréntesis da el número de caracter a partir del cual vamos a comparar en la cadena buscada. La primera cadena es en la que se busca y la segunda lo que hay que buscar. El resultado de esta función es numérico. O en el caso de que no se halle la cadena buscada dentro de la cadena en la que se busca, y un número mayor que O si el resultado de la búsqueda fuese positivo. Además en este último caso el número nos va a señalar el caracter a partir del cual aparece la cadena buscada en la que se busca. Un ejemplo resultará clarificador. INSTR (1, **"abcdabf"**, "ab") será igual a 1 e INSTR *(3, "abcdabf", "ab")* será igual a 5 nótese que en este último caso comenzamos a buscar a partir de **"c"**, como consecuencia del 3 inicial.

Pero regresemos a nuestro programa. Ya se habrá dado cuenta que mediante esta función, averiguamos si la letra que da el que intenta solucionar el acertijo se encuentra en la solución. Si llevamos esto al término de nuestras variables tendríamos si letra$ se encuentra contenida en palabra$. Además, no sólo eso, sino que sabemos en qué posición de palabra$ se encuentra letra$, esta posición la guardamos en otra variable numérica que llamaremos posi.

Mediante la siguiente línea de nuestro programa separaremos dos casos. Que posi sea igual a O, es decir, la letra no se encuentra en la palabra, en este caso enviaremos la ejecución a la subrutina **"errores"**, para anotar el error. Y en el caso de que posi sea distinto de O continúa la ejecución en la línea:

```
230 MID$ (trabajo$, posi, 1)=letra$
240 MID$ (palabra$, posi, 1)="*"
```

Tratemos de comprender el significado de esta instrucción así como la misión que tiene en nuestro programa.

La orden MID$ lo que hace es coger la primera cadena del paréntesis y sustituye desde el carácter que indica el primer número, tantos caracteres como indica el segundo, por la cadena que se encuentra tras el igual. Nuevamente recurrimos a un ejemplo para esclarecer esto.

Supongamos que A$="**federico**", entonces MID$ (A$, 3, 2)="pa" transforma el contenido de A$ por "feparico".

La sentencia MID$ es quizá la única palabra del Basic Locomotive que puede utilizarse como orden y como función, al final del artículo hablaremos de la diferencia de estos dos términos, así como el empleo de ésta como función.

En nuestro programa gracias a esta instrucción, en la línea 230 cambiamos un guión de trabajo$ por la letra que se halla acertado, y mediante la línea 240 tachamos de palabra$ la letra encontrada, sustituyéndola por un **"*"**.

Simulemos la primera jugada suponiendo que la palabra buscada sea HELIPUERTO, en un principio tendríamos:

```
palabra$="HELIPUERTO"
trabajo$="——————————"
```

Ahora si hubiésemos elegido la letra "P" para ver si se encuentra en la palabra buscada, estas dos variables quedarían del siguiente modo:

```
palabra$="HELI*UERTO"
trabajo$="————P———"
```

El motivo de tachar de la palabra la letra encontrada es penalizar las repeticiones.

## Program Acción

```
250 LOCATE 8,20: PRINT trabajo$
260 posi=instr (1, palabra$, letra$)
    270 IF posi < >
    O THEN GOTO 230
    280 GOTO 130
```

En la primera de estas líneas escribimos en un lugar de la pantalla la variable trabajo$. Gracias a ésta iremos viendo como va evolucionando nuestra palabra hacia la solución.

La línea 260 ya no debe ser ningún misterio para usted, aunque quizá sí lo sea su misión. La razón de volver a ponerla estriba en que la palabra buscada puede contener alguna letra repetida, por lo que volvemos a mirar si existe otra letra, después de haber borrado la primera encontrada. En el caso de que exista otra letra enviamos la ejecución a la línea 230 para que se repita todo el proceso descrito anteriormente.

En el caso de que no existan más letras iguales a la seleccionada enviamos la ejecución, mediante el GOTO 130 de la línea 280 a la línea 130, para que de este modo se nos vuelva a pedir otra letra.

Bien ya casi tenemos terminado nuestro programa, ahora usted recordará que en la línea 220 si la letra seleccionada no se encontraba en la palabra buscada, enviábamos la ejecución a la línea 290 donde tendremos que penalizar este error. Realizemos esta subrutina.

```
290 REM= =la letra no está en la
          palabra= =
    300 errores=errores+1
310 fallo$=LEFT$ (a$, errores)
320 LOCATE 1,1: PRINT fallo$
```

Nuevamente nos encontramos con otra de las instrucciones de nuestra lista. Pero no nos adelantemos. En la línea 300 sumamos 1 a la cuenta de los errores. Recuerde que en este caso= es una asignación, errores queda cargado con lo que era errores más 1.

Veamos cuál es el significado de la instrucción de la línea 310, LEFT$. Lo que hace esta función es coger de la cadena del paréntesis tantos caracteres comenzando desde la izquierda, como indique el número que se

encuentra como segundo miembro del paréntesis. Así LEFT$ (''12345'', 3), daría como resultado ''123''. Existe también otra función dual a ésta, RIGTH$, que realiza lo mismo, salvo que comienza a contar por la derecha.

Volviendo a nuestro programa, recuerde que a$ era igual a ''AHORCADO''. Luego la línea 310 lo que hará será ir cargando la variable fallo$ con ''A'' en el primer error cometido, ''AH'' en el segundo, ''AHO'' en el tercero, y así hasta el octavo en el que tendremos fallo$=''AHORCADO'', con lo que el juego habrá finalizado sin ser hallada la solución. Detención que provocaremos gracias a la siguiente línea.

330 IF fallo$=''AHORCADO'' TEHN STOP

Ahora si el juego aún no ha terminado, sólo son permitidos ocho fallos, observe que puede fácilmente modificar este número cambiando la palabra ahorcado por otra más corta; a pesar de haber cometido un error, deberemos enviar la ejecución del programa a la línea 145 para que se nos solicite nuevamente una letra.

340 GOTO 145

Y ya para terminar hagamos la rutina que nos informará de que la palabra ha sido terminada y por tanto encontrada.

350 REM———la palabra es encontrada———
360 CLS: LOCATE 5,9: PRINT ''MUY BIEN''
370 END

Recordará que en la línea 140 si trabajo$=solución$ la ejecución del programa era enviada a la línea 350. A partir de ésta borramos la pantalla gracias al CLS de la línea 360, y escribimos en un lugar de la pantalla ''MUY BIEN''.

El resto es labor suya: siéntese delante de su CPC y recuerde que la programación es como el ajedrez, por mucho que aprenda a mover las piezas. no aprenderá a jugar mientras no juegue. Anímese y modifique el programa con todo lo aprendido en números anteriores, ventanas y gráficos pueden transformar este programa algo estéril en presentación, dada la sencillez que se pretendía, en una verdadera maravilla. Próximamente analizaremos más órdenes para el manejo de cadenas alfanuméricas, junto con programas ejemplo, breves y claros, que ilustren su uso.

# Program Acción

```
10 REM * EL AHORCADO *
20 REM JJ MARTIN PARA MICROHOBBY AM
STRAD
30 REM * EL AHORCADO *
40 MODE 0
50 A$="AHORCADO":B$=SPACE$(15)
60 errores=0
70 INPUT palabra$
80 palabra$=UPPER$(palabra$)
90 CLS
100 solucion$=palabra$
110 lon=LEN(palabra$)
120 trabajo$=STRING$(lon,"-")
130 LOCATE 8,20:PRINT trabajo$
140 IF TRABAJO$=SOLUCION$ THEN 350
145 LOCATE 1,23:PRINT "PULSE UNA LE
TRA"
150 IF INKEY$<>"" THEN GOTO 150
160 letra$=INKEY$
170 IF letra$="" THEN GOTO 160
180 letra$=UPPER$(letra$)
185 LOCATE 1,23:PRINT B$
190 dichas$=dichas$ + "-" + letra$
200 LOCATE 1,4:PRINT dichas$
210 posi= INSTR(1,PALABRA$,letra$)
220 IF POSi=0 THEN GOTO 290
230 MID$(trabajo$,posi,1)=letra$
240 MID$(palabra$,posi,1)="*"
250 LOCATE 8,20:PRINT trabajo$
260 posi= INSTR(1,PALABRA$,letra$)
270 IF posi<>0 THEN 230
280 GOTO 130
290 REM-la letra no esta en la pala
bra--
300 errores=errores+1
310 fallo$=LEFT$(a$,errores)
320 LOCATE 1,1:PRINT fallo$
330 IF fallo$=a$ THEN STOP
340 GOTO 145
350 REM--la palabra es encontrada--
360 LOCATE 5,9:PRINT "MUY BIEN"
370 END
```

Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

# ¡GUERRA al I.V.A.!

Suscríbete ahora a **MICROHOBBY AMSTRAD** y benefíciate de un

# 26% DESCUENTO

(Oferta válida sólo hasta el 28 de febrero de 1986)

Microhobby AMSTRAD **te ofrece ahora una oportunidad excepcional. Hasta el 28 de febrero de 1986 podrás suscribirte a nuestra revista sin tener que pagar el recargo correspondiente al Impuesto sobre el Valor Añadido.**

Microhobby AMSTRAD **lo abona por ti. Ahora puedes recibir** Microbobby AMSTRAD **en tu domicilio durante todo un año por sólo 5.900 ptas., es decir, 2.100 ptas. menos de su valor real.**

**¡APROVECHA ESTA OPORTUNIDAD!**

RECORTA O COPIA ESTE CUPON Y ENVIALO A HOBBY PRESS, S.A. APDO. DE CORREOS 232 ALCOBENDAS (MADRID)

Deseo suscribirme a **Microhobby AMSTRAD** durante un año (50 números) por **sólo 5.900 ptas.,** lo que me supone **un ahorro de 2.100 pesetas.**

El primer número que deseo recibir es el _____

**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 HOBBY PRESS, S. A.**
- ☐ **Giro Postal a nombre de HOBBY PRESS, S. A., N.º** _____
- ☐ **Contra reembolso del primer envío**
- ☐ **VISA N.º** _____

**Fecha de caducidad de la tarjeta** _____

- ☐ **Domiciliación bancaria (50 NUMEROS MAS 1 DE REGALO)**

**Banco** _____ **Sucursal** _____ **Localidad** _____

**N.º de cuenta** _____

**Si lo prefiere puede suscribirse por teléfono: (91) 654 28 98**

**Firma y fecha**

# INSTRUCCIONES DE PUESTA A UNO, A CERO Y PRUEBA BITS

***Hasta ahora hemos estado trabajando con bytes, o sea, números comprendidos entre el 0 y 255, que son los valores máximos que podemos almacenar en un byte. Pues bien, en este capítulo vamos a hablar de ciertas instrucciones que nos van a permitir introducirnos en un byte y de esta forma poder examinarlo de una manera mucho más exhaustiva.***

n número realmente es un byte en formato binario, es decir, en forma de unos y ceros, que son los valores con los que trabaja el microprocesador. Podemos desgranar un byte en varios ceros y unos, a cada uno de estos valores se les llama bits, por lo que un bit únicamente puede tener un valor que puede ser un cero o un uno.

### Sistema binario y bits

Un byte está compuesto por ocho bits que van del bit 0 que es el menos significativo hasta el bit 7 que es el más significativo. Vamos a ver cuál es el valor de cada uno de estos bits dentro del byte:

| | | | |
|---|---|---|---|
| bit 7 | bit 6 | bit 5 | bit 4 |
| 128 | 64 | 32 | 16 |
| bits 3 | bit 2 | bit 1 | bit 0 |
| 8 | 4 | 2 | 1 |

Algunos ejemplos de números representados en este formato serían:

| | |
|---|---|
| 128 | 10000000 |
| 1 | 00000001 |
| 129 | 10000001 |
| 255 | 11111111 |
| 0 | 00000000 |

si el bit está puesto a uno, se le suma el valor correspondiente a ese bit, y si está puesto a cero, se le suma cero, y así obtenemos el número deseado. El mayor número que podemos obtener de esta forma, es el 255 y el menor es el cero, por eso un byte estará siempre comprendido entre esos dos valores.

### Instrucción SET

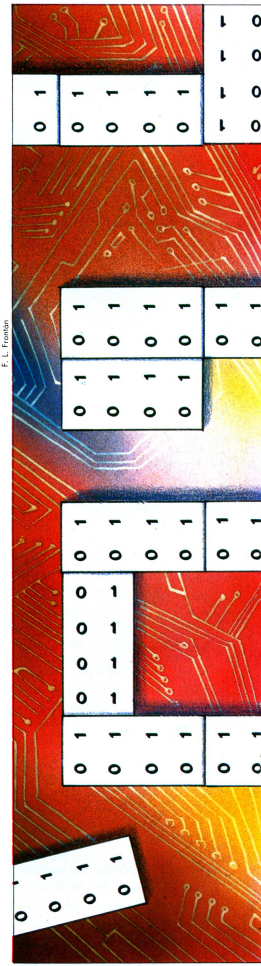
Una de las instrucciones de tratamiento de bits es la instrucción de puesta a uno, o sea, pone el bit correspondiente del registro indicado a uno. Esta instrucción se representa de la siguiente forma:

SET b,r

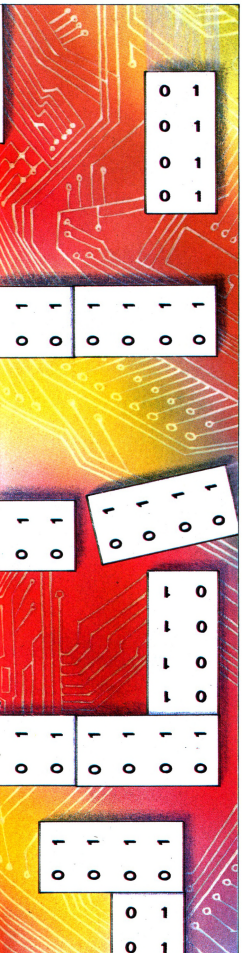
donde b puede ser cualquier bit del 0 al 7 del registro r (cualquiera de los registros simples B, C, D, E, H, L o A). Así pues la instrucción SET b,r pondrá a uno el bit b de r.

Un ejemplo de funcionamiento de esta instrucción lo tenemos en el programa número uno; lo que hacemos únicamente, es poner el bit 4 regristro B, a uno, independientemente del valor que tuviera ese registro.

Ahora el programa número 2. Para entenderlo mejor, vamos a ver en primer lugar la diferencia que existe entre los códigos de las letras mayúsculas y las minúsculas. Los códigos de



F. J. Frantzen



estas últimas van desde el 97 hasta el 122, y los códigos de las primeras van desde el 65 hasta el 90. Si nos fijamos en los códigos de las letras mayúsculas, veremos que en éstos, el bit 5 está siempre a cero, en cambio en los códigos de las letras minúsculas el bit 5 siempre está puesto a uno. Así pues lo que hace el programa 2, es cargar en el acumulador el código de una letra mayúscula, y luego pone a uno el bit 5 de ese código, por lo que esa letra mayúscula queda convertida en minúscula, y luego la imprime en pantalla.

Esta instrucción de puesta a uno, también puede actuar con el contenido de la posición de memoria indicada por HL, IX o IY. Esto lo podemos ver en el programa 3. Aquí lo que hacemos es poner a uno el bit 3 de la posición de memoria indicada por el registro doble HL. Así pues si la posición de memoria &7000, que es la que hemos cargado en HL, contenía un cero, después de la ejecución del programa, esa posición de memoria contendrá un 8, pues el bit 3 de esa posesión de memoria contendrá un 1.

### Instrucción RES

Vamos a ver ahora otra instrucción de manejo de bits, esta instrucción es la de puesta a cero y se representa de la siguiente forma:

RES b,m

b representa cualquier bit del 0 al 7 del registro m. Este registro puede ser cualquiera de los registros simples H, L, D, E, B, C o A, o bien (HL), (IX+d) o (IY+d).

Esta instrucción lo que hace es poner a cero el bit correspondiente del registro indicado.

En el programa número 4 se da un ejemplo de esta instrucción, lo que hacemos es cargar el acumulador con uno y luego ponemos el bit 0 del acumulador a cero, con lo cual el acumulador se queda otra vez con cero.

El programa número 5, es muy similar a uno visto anteriormente, pero a la inversa: se carga el acumulador con la letra minúscula «a», y luego ponemos a cero el bit 5, con lo cual en el acumulador obtendremos el código de la letra mayúscula «A» y luego la imprimimos.

Así pues, el manejo de la instrucción de puesta a cero es similar a la instrucción vista anteriormente de puesta a uno.

# Código máquina

### Instrucción BIT

Existe una última instrucción de manejo de bits, que es la instrucción de prueba de bits. Esta es una instrucción muy interesante, ya que nos permite averiguar en cualquier momento, cuáles son los bits puestos a uno o a cero de un registro cualquiera.

La instrucción de prueba de bits se representa de la siguiente manera:

BIT b,r

donde b es cualquier bit del 0 al 7 del registro r. Este registro puede ser cualquiera de los registros H, L, B, C, D, E o A.

Tras la ejecución de esta instrucción, el flag Z (flag de cero) del registro F, contendrá el complementario del bit indicado dentro del registro indicado.

Así pues si ejecutamos la siguiente instrucción:

BIT 2,A

y el bit 2 del acumulador contiene un uno, el flag Z del registro F se pondrá a cero.

Si el bit 2 del acumulador fuese un cero, el flag Z se pondría a uno.

El programa número 6 es un ejemplo práctico del manejo de la instrucción de prueba de bits. Lo que hace es mirar si el bit 7 del acumulador está puesto a uno, si es así vuelve al Basic, de lo contrario pasa e incrementa el contenido del acumulador. De este modo, el programa volverá al Basic cuando el contenido del acumulador sea exactamente 128, ya que éste es el valor correspondiente al bit 7. En ese momento el bit contendrá un uno y el flag Zedel registro F contendrá su complementario, o sea un cero.

Volvamos de nuevo al problema de las letras mayúsculas y minúsculas; como hemos dicho, las primeras contienen en el bit 5 un cero y las últimas contienen en dicho bit un uno.

Así pues el programa número 7, lo que hace es comprobar si un código de una letra dada, corresponde a una letra mayúscula o minúscula; para ello, se chequea el bit 5 del código de dicha letra, si en dicho bit hay un uno, entonces el flag Z del registro F contendrá un cero, lo que querrá decir que la letra en cuestión es minúscula. Si por el contrario el bit 5 de dicho código contiene un cero, el flag Z contendrá un uno, por lo que la letra en cuestión será mayúscula.

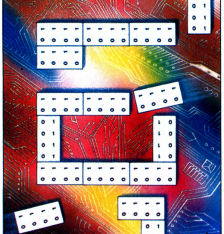
Así pues si en la línea 10 de este programa, cargamos el acumulador con el código de una letra mayúscula, al ejecutarlo se imprimirá en pantalla una «M», si por el contrario cargamos el código de una letra minúscula, nos aparecerá una «m» en pantalla.

Con esta instrucción también podemos chequear los bits que componen una posición de memoria. Esta posición de memoria la deberemos cargar en los registros dobles HL, IX o bien IY.

En el programa 8 cargamos el registro indexado IX con la posición de memoria &7000, luego con la instrucción:

BIT 1, (IX+1)

comprobamos si el bit 1 de la posición de memoria &7001 está puesto a uno o a cero. Si ese bit está a uno, el flag Z contendrá un cero y el programa imprimirá en pantalla un uno. Si por el contrario el bit 1 de esa posición de memoria contiene un cero, el flag Z contendrá un uno y el programa nos imprimirá un cero.



Como hemos visto hasta ahora, con esta instrucción de prueba de bits, podemos chequear los diferentes bits de cualquier registro o posición de memoria. Así pues con dicha instrucción podemos desgranar todos los bits de cualquier registro y conocer de este modo cuáles están puestos a uno y cuáles están puestos a cero.

El programa número nueve aprovecha esta posibilidad de la instrucción de chequeo de bits para convertir un número decimal dado a binario.

En primer lugar, dicho programa carga en el registro par HL con la posición de memoria &7000. en dicha dirección será donde nosotros deberemos colocar el número decimal que deseemos pasar a binario. Esto lo podemos hacer mediante la instrucción Basic:

POKE &7000, n

donde n será el número decimal deseado, que deberá estar comprendido entre 0 y 255.

A continuación el programa irá chequeando cada uno de los bits que componen esa posición de memoria. Si el bit chequeado contiene un cero, el programa hará que se imprima un cero en pantalla y por el contrario el bit chequeado contiene un cero, el programa hará que se imprima un uno en pantalla. Y así hasta terminar la lectura de los ocho bits que componen un byte. De esta forma tan sencilla habremos conseguido pasar un número decimal dado a forma binaria.

El programa anterior, como habréis visto no puede transformar a binario un número superior a 255, pero esto no representa ninguna dificultad, ya que si en lugar de chequear los bits de una posición de memoria, chequeáramos los bits de dos posiciones de memoria consecutivas, lograríamos pasar a binario un número comprendido entre 0 y 65535.

Por último decir que las instrucciones de puesta a uno y puesta a cero, es decir, las instrucciones:

SET b,r
RES b,r

no afectan a ningún flag del registro F.

Por el contrario la instrucción de prueba de bits, o sea la instrucción:

BIT b,r

afecta únicamente al flag Z del registro F.

## PROGRAMAS

```
                        10 ;PROGRAMA-1
                        20 ;
8000                    25          ORG  #8000
8000  CBE0              30          SET  4,B
8002  C9                40          RET
                        10 ;PROGRAMA-2
                        20 ;
8000                    25          ORG  #8000
8000  3E41              30          LD   A,"A"
8002  CBEF              40          SET  5,A
8004  CD5ABB            50          CALL #BB5A
8007  C9                60          RET
                        10 ;PROGRAMA-3
                        20 ;
8000                    25          ORG  #8000
8000  210070            30          LD   HL,#7000
8003  CBDE              40          SET  3,(HL)
8005  C9                50          RET
                        10 ;PROGRAMA-4
                        20 ;
8000                    25          ORG  #8000
8000  3E01              30          LD   A,1
8002  CB87              40          RES  0,A
8004  C9                50          RET
                        10 ;PROGRAMA-5
                        20 ;
8000                    25          ORG  #8000
8000  3E61              30          LD   A,"a"
8002  CBAF              40          RES  5,A
8004  CD5ABB            50          CALL #BB5A
8007  C9                60          RET
                        10 ;PROGRAMA-6
                        20 ;
8000                    25          ORG  #8000
8000  3E00              30          LD   A,0
8002  CB7F              40 BUC:     BIT  7,A
8004  2003              50          JR   NZ,FIN
8006  3C                60          INC  A
8007  18F9              70          JR   BUC
8009  3A0070            80 FIN:     LD   (#7000),A
800C  C9                90          RET
```

## ETIQUETAS

```
BUC     8002  FIN     8009
                        10 ;PROGRAMA-7
                        20 ;
8000                    25          ORG  #8000
8000  3E41              30          LD   A,65
8002  CB6F              40          BIT  5,A
8004  2006              50          JR   NZ,MINUS
8006  3E4D              60          LD   A,"M"
8008  CD5ABB            70          CALL #BB5A
800B  C9                80          RET
800C  3E6D              90 MINUS:   LD   A,"m"
800E  CD5ABB           100          CALL #BB5A
8011  C9               110          RET
```

## ETIQUETAS

```
MINUS   800C
                        10 ;PROGRAMA-8
                        20 ;
8000                    25          ORG  #8000
8000  DD210070          30          LD   IX,#7000
8004  DDCB014E          40          BIT  1,(IX+1)
8008  2806              50          JR   Z,CERO
800A  3E31              60          LD   A,"1"
800C  CD5ABB            70          CALL #BB5A
800F  C9                80          RET
8010  3E30              90 CERO:    LD   A,"0"
8012  CD5ABB           100          CALL #BB5A
8015  C9               110          RET
```

## ETIQUETAS

```
CERO    8010
                        10 ;PROGRAMA-9
                        20 ;
8000                    25          ORG  #8000
8000  210070            30          LD   HL,#7000
8003  CBFE              40          BIT  7,(HL)
8005  CD2C80            50          CALL MIRAR
8008  CB76              60          BIT  6,(HL)
800A  CD2C80            70          CALL MIRAR
800D  CB6E              80          BIT  5,(HL)
800F  CD2C80            90          CALL MIRAR
8012  CB66             100          BIT  4,(HL)
8014  CD2C80           110          CALL MIRAR
8017  CB5E             120          BIT  3,(HL)
8019  CD2C80           130          CALL MIRAR
801C  CB56             140          BIT  2,(HL)
801E  CD2C80           150          CALL MIRAR
8021  CB4E             160          BIT  1,(HL)
8023  CD2C80           170          CALL MIRAR
8026  CB46             180          BIT  0,(HL)
8028  CD2C80           190          CALL MIRAR
802B  C9               200          RET
802C  CD33B0           210 MIRAR:   CALL 2,CERO
802F  C43980           220          CALL NZ,UNO
8032  C9               230          RET
8033  3E30             240 CERO:    LD   A,"0"
8035  CD5ABB           250          CALL #BB5A
8038  C9               260          RET
8039  3E31             270 UNO:     LD   A,"1"
803B  CD5ABB           280          CALL #BB5A
803E  C9               290          RET
```

## ETIQUETAS

```
CERO    8033    MIRAR   802C    UNO     8039
```

# MICRO-1

C/ Duque de Sesto, 50
28009 Madrid
Tels. (91) 275 96 16/274 53 80
MICROLID: Gregorio Fdez.

el IVA lo paga MICRO-1

**SOFTWARE: Por cada programa GRATIS ¡¡1 bolígrafo con reloj de cuarzo!!**

| | | | |
|---|---|---|---|
| RAID | **2.300** ptas. | EXPLOIDING FIST | **2.300** ptas. |
| JUMP JET | **2.495** ptas. | FIGHTER PILOT | **1.975** ptas. |
| TALES OF ARABIAN | **1.950** ptas. | MASTER OF T. LAMP | **1.950** ptas. |
| SABREWULF | **1.650** ptas. | NIGHTSHADE | **1.950** ptas. |
| GHOSTBUSTERS | **1.950** ptas. | HACKER | **1.950** ptas. |
| HIGHWAY ENCOUNTER | **1.750** ptas. | MAPGAME | **2.700** ptas. |

IMPRESORAS
¡¡20% DE DESCUENTO SOBRE P.V.P.!!

LAPIZ OPTICO
¡¡4.900 ptas.!!

YOYSTICK QUICK SHOT II _____ **2.395** ptas.
JOYSTICK QUICK SHOT V _____ **2.795** ptas.

CASSETTE ESPECIAL
ORDENADOR 5.295 ptas.

TOSHIBA MSX 64 K
¡¡34.900 ptas.!!

CINTA C-15 ESPECIAL ORDENADOR _____ 85
DISKETTE 3'' _____ 990

PC-COMPATIBLE IBM 256 K
MONITOR FOSFORO VERDE
2 BOCAS DISKETTE 360 K
SOLO ¡¡243.900!!

PC-PORTATIL
2 BOCAS 360 K
¡¡174.900!!

PRECIOS SUPER-EXCEPCIONALES PARA
AMSTRAD CPC-472 y CPC-6128
¡¡LLAMANOS, TE ASOMBRARAS!!

TAPA METACRILATO PARA
TECLADO ¡¡2.300 ptas.!!

SINTETIZADOR DE VOZ
¡¡8.495 ptas.!!

UNIDAD DE DISCO CON
CONTROLADOR: 49.900

MODULADOR TV
8.400 ptas.

INTERFACE DISCO
5 1/4'' 5.300

**LIBROS:**

| | |
|---|---|
| Curso autodidáctico Basic I | **2.525** ptas. |
| Curso autodidáctico Basic II | **2.525** ptas. |
| Programando con Amstrad | **2.195** ptas. |
| Juegos sensacionales Amstrad | **1.950** ptas. |
| Hacia la inteligencia artificial | **1.295** ptas. |
| Música y sonidos con Amstrad | **995** ptas. |

IMPRESORA MARGARITA
¡¡49.900 ptas.!!

Pedidos contra reembolso sin ningún gasto de envío. Tels. (91) 275 96 16/274 53 80 o escribiendo a Micro-1. C/ Duque de Sesto, 50. 28009 Madrid.

# Sin duda alguna

A través de esta sección se pretende resolver, en la medida de lo posible, todas las posibles dudas que **«atormenten»** a todas las personas interesadas en el mundo del AMSTRAD, sean o no poseedores de uno y, si lo son, se encuentren en cualquier nivel de destreza en su manejo.

Semanalmente, aparecen en estas páginas las consultas de la mayor cantidad de usuarios posible; ello redundará en un mejor servicio y en un contacto más estrecho entre todos nosotros a través de la revista.

**SIN DUDA ALGUNA** está abierta a todos.

Tengo desde junio un 664, y quisiera saber dónde están las famosas 64 K RAM, porque el ordenador responde que tiene 40,25.

También querría conocer si los programas del 6128 y del 8256 serían compatibles con el mío caso de instalar la ampliación de memoria que anunciáis en vuestro n.º 10 (los programas del 6128 en la ampliación de 128 K y los del 6128 y 8256 en la de 256 K).

Otra cosa: ¿por qué son mejores los procesadores de 16 bits?, porque siempre os estáis metiendo con los 8 bits del Z80.

Por último, dedicar a **Amstrad** un efusivo corte de mangas, por su 6128, naturalmente, y otro a INDESCOMP, por su precio de venta. Desde luego, una broma de mal gusto.

**Antonio García** *(Barcelona)*

*Hay que distinguir entre la memoria total de un ordenador y la memoria disponible para el usuario.*

*En el caso del* ***Amstrad****, el sistema operativo, el firmware y otras cosas también ocupan memoria, por lo que sólo te quedan para programas Basic los Kbytes que indica la orden PRINT FRE (0).*

*De todas maneras, piensa que tu* ***Amstrad*** *maneja más de 64 Kbytes, sólo que no de una vez. En efecto, las cuarenta y pico libres, más las 32 de ROM suman más de 64. La publicidad, por razones obvias, omite esta explicación referente a las diferencias entre memoria total y libre para el usuario.*

---

*Una vez ampliado tu 664, debiera ser absolutamente compatible con el 6128 a nivel de programas Basic y a nivel de CP/M.*

*En cuanto a los programas para el 8256, los programas para esta máquina que corren bajo CP/M+ deben ser compatibles con mínimos cambios. A nivel de programas Basic, olvidate. El Basic del 8256 no tiene nada que ver con el de la serie CPC.*

*Los procesadores de 16 bits son mejores que los de 8 bits, así, en abstracto, por tres razones:*

*a) Pueden direccionar mucha más memoria sin paginarla.*

*b) Su capacidad de cálculo es mucho mayor porque son capaces de manejar números más grandes.*

*c) Son mucho más rápidos que los de 8 bits, aunque esto depende del diseño del ordenador que lleve uno de estos procesadores.*

*Los procesadores de 8 bits, como el Z80, pueden imitar las prestaciones de los otros, pero a costa de una notable disminución de velocidad.*

Cuando copio un programa y escribo mal una variable como por ejemplo: escore$ en vez de score$, el programa no funciona. Entonces me gustaría saber:

¿Hay alguna forma de hacer que el ordenador lo indique, o que en su defecto liste todas las variables, para de esa forma poder saber si hay alguna mal escrita, o algo parecido...?

*Por desgracia, no existe una manera sencilla de hacer lo que pretendes. En otros ordenadores, como el Spectrum por ejemplo, cuando usas una variable que no existe aparece en la pantalla el mensaje de «VARIABLE NOT FOUND» o algo parecido, pero el* ***Amstrad****, cuando encuentra una variable por primera vez que no está inicializada, la da el valor cero o cadena vacía según sea numérica o de cadena, automáticamente.*

*Para crear un comando que listara todas las variables, sería necesario recurrir al código máquina, al sistema RSX. En numerosos artículos de* ***AMSTRAD*** *Semanal encontrarás información acerca de cómo hacerlo.*

# Amstrad Ideas

**AMSTRAD** Semanal comunica a todos sus lectores la apertura de una nueva sección dedicada a recoger las mejores ideas que exploten al máximo las posibilidades del ordenador, materializadas en programas claros y cortos *(máximo 25 líneas)*. Los mejores de entre todos ellos serán publicados con el nombre de su autor en la revista, recibiendo como premio, gratuitamente en su domicilio los cuatro primeros números de nuestra cinta mensual. Los programas enviados deberán incluir:

**— Cinta de cassette con el programa o programas grabados.**

**— Explicación detallada del funcionamiento y propósito del programa, mecanografiado a 2 espacios o con letra clara.**

Es imprescindible indicar en el sobre claramente: **AMSTRAD IDEAS.**

La dirección es:

**Hobby Press, S. A.**
*La Granja, s/n.*
*Polígono Industrial de Alcobendas.*
*Madrid*


**¡Operación cambio!**

*Valoramos:*
Tu AMSTRAD 464 en 50.000 ptas.
Un Spectrum+ en 30.000 ptas.
Amstrad CPC 664 en 70.000 ptas.
En la compra del AMSTRAD CPC 6128 o PCW 8256.
Consulte para monitor color.
*Precios especiales en impresoras y accesorios.*
☎ Tardes 270 34 97.

# MINI OFFICE

## PROGRAMA DEL AÑO EN INGLATERRA

FINALIST MICRO COMPUTING BRITISH AWARDS 1985

© DATABASE PUBLICATIONS

### 1 PROCESADOR DE TEXTOS

¡Ideal para escribir cartas e informes! Características: Visualización continua del tiempo • Contador de palabras (indicando las palabras por minuto) • Texto normal o doble, en pantalla o impresora.

### 2 HOJA DE CALCULO

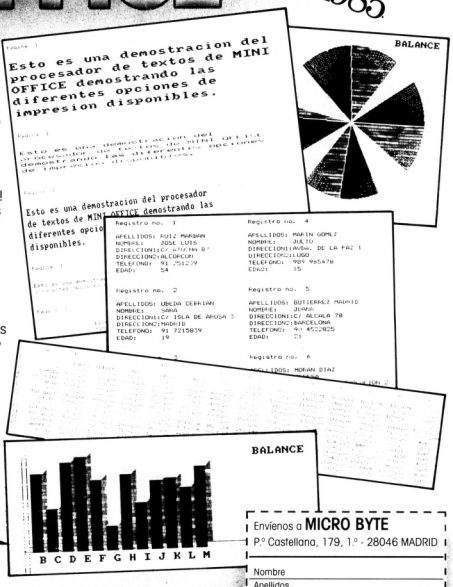
¡Utiliza tu micro para controlar tus cuentas! Características: Cifras visualizadas en filas y columnas • Actualización permanente • Actualización reflejada instantáneamente en toda la hoja • Grabación de los resultados para futuras modificaciones.

### 3 GRAFICOS

¡Convierte esos números en maravillosos gráficos! Características: Gráficos de barras en tres dimensiones • Gráficos de pastel • Histogramas.

### 4 BASE DE DATOS

¡Igual que los archivos de la oficina! Características: Cargar ficheros con solo pulsar una tecla • Clasificación • Modificación • Listados • Búsqueda.




*En Castellano
*Servimos en 48 Horas

4 PROGRAMAS P.V.P. 3.200 Ptas AL PRECIO DE 1

*Versión disco AMSTRAD P.V.P. **3.900 ptas.**




Envíenos a **MICRO BYTE**
P.º Castellana, 179, 1.º - 28046 MADRID

Nombre
Apellidos
Dirección
Población
D. P. Teléfono

Deseo que me envíen ☐ ejemplar/es del programa **MINI OFFICE**
**PARA EL MICRORDENADOR SEÑALADO**
☐ AMSTRAD ☐ COMMODORE ☐ SPECTRUM
☐ AMSTRAD VERSION DISCO
Sin gastos de envío

INCLUYO TALON NOMINATIVO ☐
CONTRA-REEMBOLSO ☐

**Pedidos por teléfono**
**91 - 442 54 33/44**

# Mercado común

Con el objeto de fomentar las relaciones entre los usuarios de AMSTRAD, *MERCADO COMUN* te ofrece sus páginas para publicar los pequeños anuncios que relacionados con el ordenador y su mundo se ajusten al formato indicado a continuación.

En *MERCADO COMUN* tienen cabida, anuncios de ventas, compras, clubs de usuarios de AMSTRAD, programadores, y en general cualquier clase de anuncio que pueda servir de utilidad a nuestros lectores.

Envíanos tu anuncio mecanografiado a: **HOBBY PRESS, S.A.**
**AMSTRAD SEMANAL.**
Apartado de correos 54.062
28080 MADRID
**¡ABSTENERSE PIRATAS!**

**Cambio** programas comerciales, últimas novedades. Interesados llamar al Tel. (91) 478 12 15. Preguntar por Alejandro.

**Vendo Amstrad** CPC-664 con monitor de fósforo verde. Nuevo. Con garantía oficial. Regalo Pascal en disco. Por 89.000 ptas. Tel. (960) 285 17 84 de Oliva *(Valencia)*. Preguntar por Pedro. Transportaría a cualquier punto de la provincia de Valencia.

**Vendo Amstrad** 464, monitor fósforo. En garantía. Regalo juegos y manuales de uso y Basic **Amstrad**, 55.000 ptas. Eduardo Ruiz de Velasco. Tel. (91) 204 52 76.

Desearía **contactar** con usuarios del **Amstrad** para intercambio de programas, sobre todo de utilidades. Los interesados escribir a: Orlando Alonso Martínez. Ed. «A Casiña» 1.º Fase, Portal 2 - 3.º dcha. Las Lagunas *(Orense)*.

**Vendo Amstrad** CPC 664, monitor color con sus programas, más 4 disquetes y conector de casset de regalo por sólo 112.000 pts. con garantía. Tel. (977) 32 03 13.

**Deseo** comprar un **Amstrad** 464 de segunda mano con monitor en fósforo verde o en color, interesados llamar al Tel. 705 03 57 de Madrid. De 17 a 20 horas.

**Cambio** monitor fósforo verde GT-64 y alimentador/modulador MP-1, por monitor color CTM-640. Abono diferencia precio. Fernando Corredera. Tel. 759 25 28. Madrid.

**Vendo** el siguiente lote de programas al precio de 2.750 pts.: Decathlon, Easi-amsword (Procesador de textos), Fred, roland in then caves The Galactic plage, Harrier attack. Llamar al Tel. (976) 34 61 95. O escribir a Joaquín López Vela. C/ Parque de Roma, F-1 - 8-G. 50010 Zaragoza.

**Urge** vender **Amstrad** CPC-464 por cambio de ordenador, regalo programas de facturación y control de stocks y libro de juegos, precio a convenir, es con pantalla de fósforo verde, y está comprado en junio. Estoy en el Tel. (964) 47 15 80 de Benicarló (CS). Preguntar por Jordi de 2 a 4.

**Queremos** deshacernos de unos programas entre ellos el Exploding-Fist, Souther Belle, Knight Lore. Nos gustaría venderlos o cambiarlos: Estamos interesados por el Pole Position. Llamadnos al Tel. (988) 75 16 02. C/ Avda. Gral. Goded, 23 - 3.º C. 34005 Palencia. Preguntad por José.

**«Vendo,** por cambio de equipo, Spectrum Plus en perfecto funcionamiento, con todos sus accesorios, y más de 50 programas comerciales en cintas, todo por 26.000 pts. Llamad al Tel. (923) 24 91 86. Jesús.

**«Deseo** comprar impresora compatible con un **Amstrad** 664, en buen estado. Tel. (93) 236 03 15. «Juan Antonio».


NL New Line
GABINETE DE INFORMATICA
- **Clases de Informática sobre AMSTRAD**
  En grupos o individuales
- **Ordenadores AMSTRAD y periféricos**
  Los mejores precios
- **Software a la medida**

ZURBANO, 4 ☎ 410 47 63
28010 MADRID


**Desearía** cambiar el DALEY THOMPSONS DECATHLON (original), por uno de estos programas:
Exploding fist.
Rocki o Frank Bruno's Boxing.
Beach Head.
Tenis.
Knight Lore.
Pole Position.
Raid over Moscow.
Dragontore.
Southern Belle.
También me interesarían algunos otros títulos. Interesados llamar a: José María Crespo. C/ Jaime Fabré, 2 - 10.º-1.º. 08019 Barcelona. Tel. 305 69 28.

**Vendo Amstrad** 464-F verde con modulador MP-1 para televisión color y B. & N., 24 juegos

# SACALE EL JUGO A TU ORDENADOR. DISEÑA TUS PROPIAS PANTALLAS Y DIVIERTETE JUGANDO CON...

Este mes:

**YOUR COMPUTER**

Te ofrece algo realmente sabroso:

## MUSICA

Este magnífico programa escrito en código máquina te permitirá manejar el sonido y la música en tu Amstrad desde Basic, mediante un nuevo juego de comandos creados especialmente para ello.

## CROSS

Tienes cuatro revólveres para destruir a tus enemigos en el mínimo tiempo posible. Necesitarás toda tu habilidad, rapidez de reflejos y suerte, mucha suerte.

## JUMPER

Debes alcanzar la cima del Valle de las Cintas Deslizantes. Tienes que saltar por los huecos de las vallas, que se desplazarán a derecha e izquierda con una rapidez de vértigo.

## MAGGOT

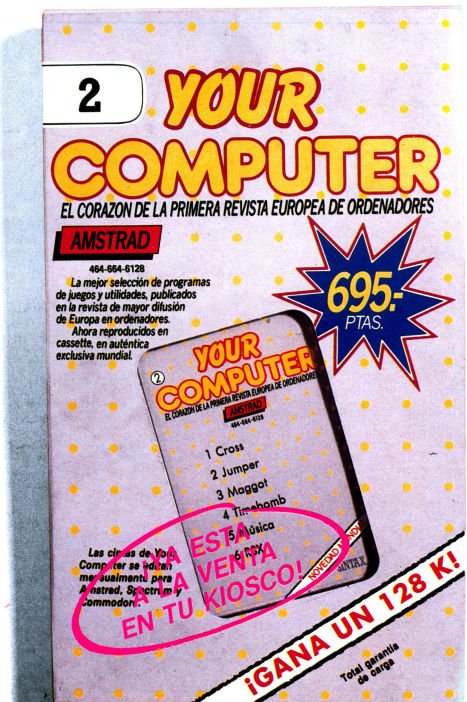
Te encuentras en la amable tierra de las setas gigantes. Tu misión es guardarla del ataque y la invasión de una peligrosísima serpiente polimórfica que las ataca sin piedad.

## TIMEBOMB

Una organización terrorista de Oriente Medio ha colocado una bomba de tiempo en el laberinto de defensa del Laboratorio de investigación bacteriológica de Lexington.

## RSX

Your Computer ha pensado en los usuarios del Amstrad CPC464 y ha creado un nuevo juego de comandos completo para tu ordenador, de forma que el Basic así ampliado no tenga nada que envidiar al de los otros modelos de la serie.




SINTAX, S.A.

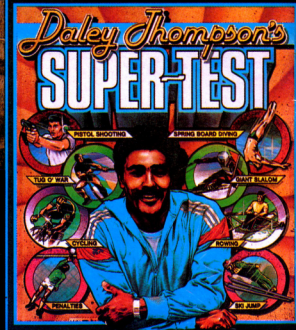
Si no lo encontrara en su kiosco, puede solicitarlo directamente a nuestra editorial:
**Paseo de la Castellana, 268. Tel.: (91) 733 25 99. 28046 Madrid.**

SI BUSCAS LO MEJOR **ERBE** *Software* LO TIENE

# TODO EL DEPORTE EN TU ORDENADOR


**NATACION, TIRO AL PLATO, SALTO DE POTRO, LEVANTAMIENTO DE PESO, TIRO AL ARCO, TRIPLE SALTO**






**PENALTIES, CICLISMO, TIRO DE PISTOLA, REMO, SALTO DE ESQUI, SLALOM GIGANTE TIRO DE CUERDA, SALTO DE TRAMPOLIN**

DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA ERBE SOFTWARE, SANTA ENGRACIA, 17. Tel.: 447 34 10
DELEGACION BARCELONA, Avd. MISTRAL, 10. Tel. (93) 432 07 31