

MICROHOBBY

AMSTRAD

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

Especial AÑO I N.º 3

350 ptas.

Canarias, Ceuta
y Melilla 335 ptas.

**SIMULADORES
DE VUELO:
¡COMPARATIVO
MONSTRUO!**

**ROBOTS:
EL FUTURO
EMPEZO AYER**

**EL INCREIBLE MUNDO
DE BLOQUES. ¿ES UN
PROGRAMA O ERES TU?**

**EL NO VA MAS
DE LOS LENGUAJES,
ESCRITO POR
NOSOTROS PARA TI.
TE PRESENTAMOS A...
FORTH**

**EL FASCINANTE
JUEGO DE LA VIDA**



Sound-on-Sound

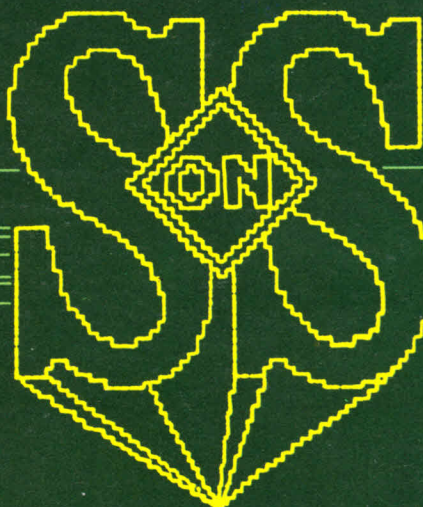
La cinta virgen para ordenador

C15 y C20

¡NUEVA!



Fabulosos
REGALOS



Cintas de alta resolución

Sound-on-Sound es una marca registrada, producida y distribuida por
IBEROFON, S.A. Avda. de Fuentenar, 35. Pol. Ind. de Coslada (Madrid)
Teléfonos: 671 22 00 - 04 - 08 - 12

Comprando una cinta Sound-on-Sound, usted puede obtener uno de estos regalos:

- Un ordenador PCW 8256 AMSTRAD.
- Un ordenador CPC 6128 AMSTRAD.
- Un ordenador CPC 6128 AMSTRAD y una IMPRESORA.
- Una IMPRESORA para AMSTRAD.
- Un cassette electrónico y un cassette software INDESCOMP.
- Un cassette electrónico.
- Un cassette software INDESCOMP.

MICROHOBBY

AMSTRAD

sumario

Año 1 • Número 3 • Octubre 1986
Precio 350 ptas. Canarias, Ceuta y Melilla 335 ptas.

Director Editorial

José I. Gómez-Centurió

Director Ejecutivo

José M.ª Díaz

Redactor Jefe

Juan José Martínez

Diseño gráfico

Fernando Chaumel

Colaboradores

Eduardo Ruiz

Javier Barceló

David Sopuerta

Robert Chatwin

Francisco Portalo

Pedro Sudón

Miguel Sepúlveda

Francisco Martín

Jesús Alonso

Pedro S. Pérez

Amalio Gómez

Secretaría Redacción

Carmen Santamaría

Fotografía

Carlos Candel

Portada

M. Barco

Ilustradores

J. Igual, J. Pons, F. L. Frontán,

J. Septien, Pejo, J. J. Mora

Edita

HOBBY PRESS, S.A.

Presidente

María Andrino

Consejero Delegado

José I. Gómez-Centurió

Jefe de Producción

Carlos Peropadre

Marketing

Marta García

Jefe de Publicidad

Concha Gutiérrez

Publicidad Barcelona

José Galán Cortés

Tél.: (93) 303 10 22/313 71 62

Secretaría de Dirección

Marisa Cogorro

Suscripciones

M.ª Rosa González

M.ª del Mar Calzada

Redacción, Administración y Publicidad

Ctra. de Irún km 12,400

(Fuencarral) 28049 Madrid

Teléfonos: Suscrip.: 734 65 00

Redacción: 734 70 12

Dto. Circulación

Paulino Blanco

Distribución

Coedis, S. A. Valencia, 245

Barcelona

Imprime

Gráficas Reunidas

Avda. Aragón, 56 (MADRID)

Fotocomposición

Novocomp, S.A.

Nicolás Morales, 38-40

Fotomecánica

GROF

Ezequiel Solana, 16

Depósito Legal:

M-5836-1986

Derechos exclusivos

de la revista

COMPUTING with the AMSTRAD

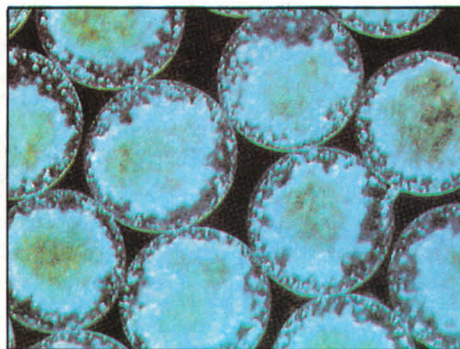
Representante para Argentina, Chile, Uruguay y Paraguay, Cia. Americana de Ediciones, S.R.L. Sud América 1.532. Tel.: 21 24 64. 1209 BUENOS AIRES (Argentina).

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

5 El juego de la vida

Lo que está vivo, ¿tiene leyes? ¿Cómo evoluciona una forma viviente? Este fascinante juego lo muestra de principio a fin. No es el comienzo de la genética asistida por ordenador, pero se le parece tanto... Además, todo en lenguaje máquina.



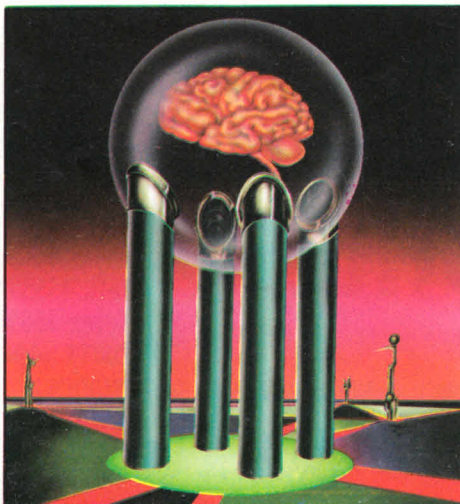
45 Simuladores de vuelo

Hacia falta una cosa así. Alguien tenía que probar todos los simuladores de vuelo que existen para Amstrad, es decir, los «juegos» más espectaculares para ordenador, hábil y objetivamente juzgada por nuestro equipo de expertos en software.



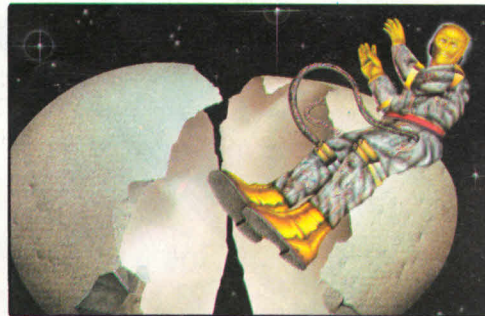
12 El lenguaje Forth

El lenguaje Forth es por antonomasia uno de los favoritos de los programadores, usado a menudo como herramienta de desarrollo. Su funcionalidad, junto a su proximidad a la máquina trabajando a un nivel verdaderamente bajo, le capacitan como un verdadero sustitutivo del en ocasiones trabajoso y cansino código máquina. **AMSTRAD ESPECIAL** número 3 os ofrece un intérprete de FORTH, ideal para iniciarse en este lenguaje.



56 Robótica

Los ordenadores están aquí. Sus hijos, los robots, también, y ya están empezando a ejercer una importante influencia en nuestras vidas. La Segunda Revolución Informática empezó ayer.



Mundo de 66 bloques

Hay cosas que son increíbles, como un elefante que vuela, o como este programa, dentro de la más pura tradición de la IA, que «existe» en un mundo imaginario regido por una serie de leyes, las cuales le permiten entender castellano normal y ejecutar órdenes de gran complejidad. Simplemente, es increíble.

SOFTWARE de muchos rombos, para mayores

TOTALMENTE EN ESPAÑOL

C
Compilador C

Versión completa del famoso C-Hisoft para CP/M. Capacidades de E/S, ficheros aleatorios y modos de acceso binario y ASCII. Incluye editor ED 80 compatible WORDSTAR.

15.000 ptas.

PASCAL 80
Compilador Pascal

Especial para Z-80. Deja el programa fuente en un programa directamente ejecutable. Incluye ED 80, editor compatible con WORDSTAR.

15.000 ptas.

KNIFE
Editor sectores

Permite trabajo directo sobre disco, bien en hexadecimal o ASCII, recuperar ficheros perdidos o borrados, alterar y/o proteger directorios, todo bajo AMSDOS y CP/M.

7.900 ptas.

DEVPAC 80
Ensamblador/des

ED 80: Editor Configurable GEN 80: Macros, inclusión en disco, ensamblador condicional, manipulación bit a bit. MON 80: Monitor y debugger, puntos de ruptura y presentación de memoria.

15.000 ptas.

MODULA-2
Comp. Modula -2

Implementación total del lenguaje MODULA-2 para CP/M. Compilador en un único paso, listo para ser linkado.

19.900 ptas.

TORCH
Tutor de CP/M

Diseñado específicamente para AMSTRAD. Incluye THE WAND, creador de menús de programas.

7.900 ptas.

POLYPRINT
Multitipos

Transforme su impresora en una imprenta. Permite la impresión en 8 tipos distintos de letras; configurable para cualquier impresora.

***11.900 ptas.**

POLY TYPEFACES
Multitipos

Añade a la potencia del programa POLYPRINT 8 juegos adicionales de impresión a los ya existentes.

***9.900 ptas.**

WRITE HAND MAN
Sidekick en CP/M

Residente en memoria, sin interferir en su programa principal le ofrece: Calculadora (Hex-Dec), Block de notas y teléfonos, Calendario, Directorios, etc...

11.900 ptas.

POLYPLOT
Impresora/Plotter

Permite realizar gráficos sofisticados en su impresora. Gráficos de pastel, histogramas comparativos, gráficos de líneas, imágenes de 980 PIXELS de densidad.

***11.900 ptas.**

POLYMAIL
Mailing

Sencillo sistema de MAIL-MERGE. Idóneo para producir circulares. Incluye editor. Permite la realización de etiquetas autoadhesivas.)

***10.900 ptas.**

CATALOG
Clasificador

Asigna a cada disco un número de serie y además indexa y cataloga los ficheros en ese disco.

8.900 ptas.

MULTI-TEXT
Módulo de textos

Módulo de textos, preparado para ser empleado con nuestro lápiz óptico ESP o con las teclas de cursor.

6.900 ptas.

FIRST STEPS
Tutor de Newword

Explore las enormes capacidades del procesador de textos NEWWORD; guiado desde los fundamentos del proceso de textos.

7.000 ptas.

MASTER LOCOSCRIPT

Dos cintas audio con instrucciones claras para aprendizaje y apoyo al manual del tratamiento de textos LOSOSCRIPT.

3.000 ptas.

DRAUGHTS-MAN II

Nueva versión mejorada y compatible con nuestra tableta GRAFPAD II: Gran capacidad en gráficos.

6.200 ptas.

TYPING CRASH COURSE
Inicia a teclear

Curso de iniciación a los teclados, recomendado para personas no acostumbradas a su uso.

9.900 ptas.

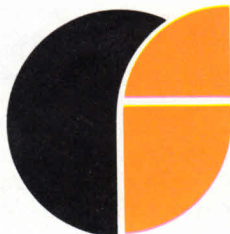
TWO FINGERS
Curso mecanográfico

Conozca a fondo las posibilidades del teclado, escribiendo con sus diez dedos en lugar de sólo dos.

9.900 ptas.

*** los 4 juntos 23.800 ptas.**

IVA no incluido



DE VENTA EN LOS MEJORES COMERCIOS DE INFORMÁTICA
Si Vd. tiene alguna dificultad para obtener los programas, puede dirigirse a:

Ofites
Informática

Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES
EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA

EL JUEGO DE LA VIDA

Cuando John Horton Conway, famoso matemático de la Universidad de Cambridge, creó por primera vez en su ordenador el conocido juego de la vida, no podía imaginar el revuelo que éste causaría; aun después de tanto tiempo, sería tremendamente difícil evaluar las pérdidas económicas que su juego provocó. En todas las grandes empresas de EE.UU., allí donde existiese un ordenador, miles de seres unicelulares nacían y morían tras un monitor verde.

Ahora, años después, AMS-TRAD Especial ofrece a todos sus lectores la posibilidad de recrearse en este universo de fantasía, donde la vida y la muerte se aúnan en una lucha eterna en pos de la simetría.



El juego de la vida, a causa de sus semejanzas con el nacimiento, muerte y alteraciones que experimentan las sociedades de seres vivos, pertenece a la clase de los llamados juegos de simulación.

Todo el proceso del juego encaja con la teoría de autómatas celulares, la cual propone la posibilidad de que una máquina provista de las instrucciones necesarias, puede ser capaz de construir una copia de sí misma.

Cada una de estas máquinas sería a su vez capaz de construir otras, y estas cuatro se convertirían en ocho, y así sucesivamente.

Ahora bien, tal proceso conduciría a un número infinito de autómatas, lo cual sería insostenible. Para evitar esta proliferación, Conway propuso la aplicación de unas leyes genéticas que regularan los nacimientos, muertes y sucesivas alteraciones.

EL CICLO DE NACIMIENTO Y MUERTE

Dicho juego debe jugarse teóricamente sobre un tablero cuadrículado infinito. Cada una de estas casillas puede encontrarse en dos estados, vacío o lleno. Cada uno de estas cuadrículas tiene asociadas un conjunto finito de cuadrículas 'vecinas' que pueden tener influencia sobre su estado.

La configuración de estados cambia a intervalos temporales, en conformidad con unas reglas de transición, aplicadas simultáneamente a todas las cuadrículas.

Conway eligió sus reglas, tras un período de experimentación, intentando satisfacer tres condiciones:

1. No debe existir ninguna configuración inicial para la que se pueda demostrar fácilmente que su población crecerá ilimitadamente.

2. Deben existir configuraciones iniciales que aparentemente crezcan sin límite.

3. Han de existir configuraciones iniciales sencillas que sean capaces de crecer y cambiar durante períodos de tiempo considerables, antes de finalizar de una de las siguientes formas posibles:

- a) Extinguirse completamente, ya sea por superpoblación o por encarecimiento.

- b) Adoptar una configuración estable invariable en lo sucesivo.

- c) Entrar en fase oscilatoria, donde se repiten sin fin dos o más estados.

LAS LEYES DE CONWAY

Las leyes genéticas elegidas por Conway, que son sobre las que está basado el programa que os ofrecemos del '**Juego de la vida**', son las siguientes:

1. Supervivencia. Cada ficha que tenga dos o tres fichas vecinas sobrevive y pasa a la siguiente generación.

2. Fallecimiento. Cada ficha que tenga cuatro o más vecinas, muere por superpoblación. Las fichas con sólo una o ninguna vecinas muere por aislamiento.

3. Nacimientos. Cada casilla vacía adyacente a exactamente tres fichas vecinas, es casilla generatriz. Por lo que deberá colocarse allí una ficha.

Debemos hacer notar que cada ficha posee ocho vecinas, es decir, cuatro ortogonalmente y otras cuatro diagonalmente.

Es importante darse cuenta que todos los nacimientos y muertes ocurren simultáneamente, y constituyen en su conjunto una generación, o como se las suele llamar un '**tic**' o '**latido**' de la vida de la configuración inicial.

Descubriremos, una vez iniciado el juego, que la población experimenta constantemente cambios insólitos, bellos e inesperados. En ciertos casos, la sociedad termina por extinguirse, si bien antes de que esto suceda, deberán pasar gran número de generaciones.

Casi todas las configuraciones iniciales terminan por alcanzar figuras estables (que Conway llama naturalezas muertas), incapaces de cambio y que quedan oscilando.

Las formaciones iniciales que no poseen simetría, tienden a ir adquiriéndola, y una vez que esto sucede ya no puede perderse.

Una familia formada únicamente por dos fichas, se extinguirán al primer latido.

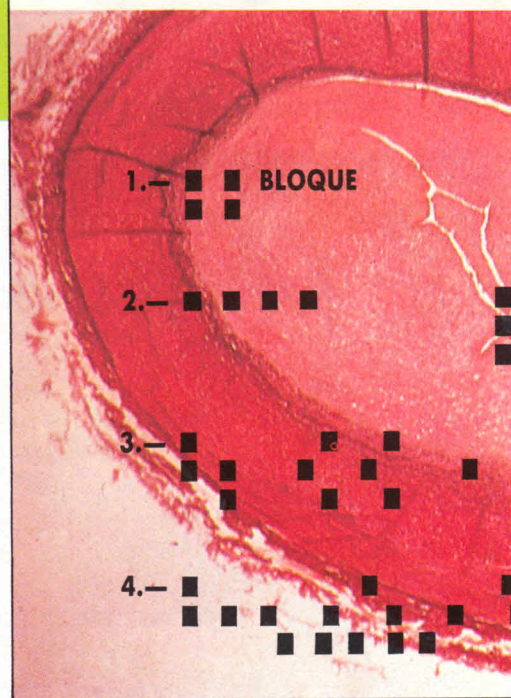
Una formación inicial de tres casillas moriría de inmediato a menos que una de ellas tenga un par de vecinas.

Debemos hacer notar que una cadena diagonal aislada de fichas, por larga que ésta sea, termina desapareciendo, ya que en cada latido pierde las dos fichas situadas en sus extremos.

La tercera configuración se transforma en un '**BLOQUE**' estable en el segundo latido.

La última de ellas es la más sencilla de las llamadas '**FLIP-FLOPS**' que son figuras oscilantes de período dos. Conway la denomina '**intermitente**'.

Veamos ahora algunas figuras compuestas por cuatro fichas conectadas entre sí por movimientos de torre, denominadas tetróminos:



El primero de ellos es como hemos visto anteriormente, una naturaleza muerta.

Los tetróminos 2 y 3, alcanzan una configuración estable al segundo latido, dicha configuración estable, se denomina '**colmena**'. El último tetrómino se convierte en colmena al tercer latido.

Vamos a ver a continuación algunas de las

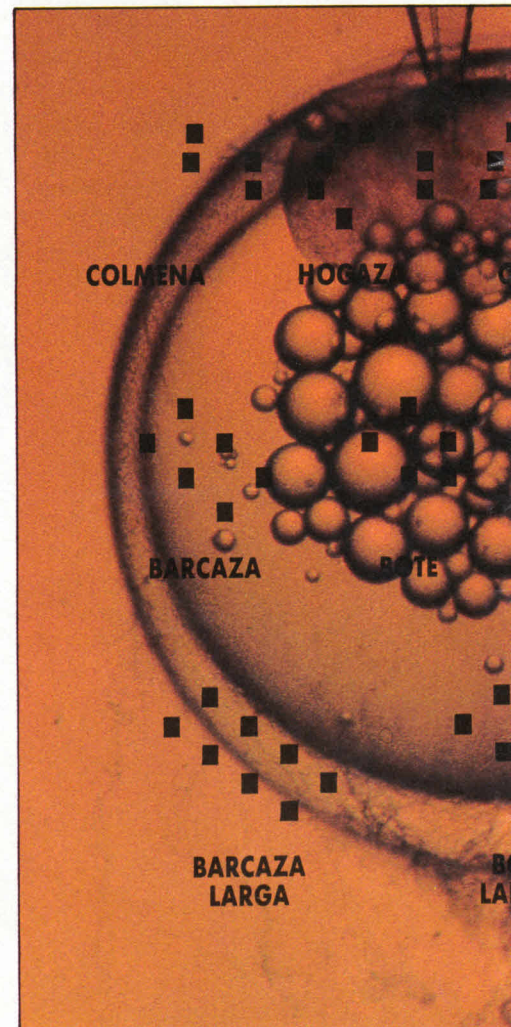
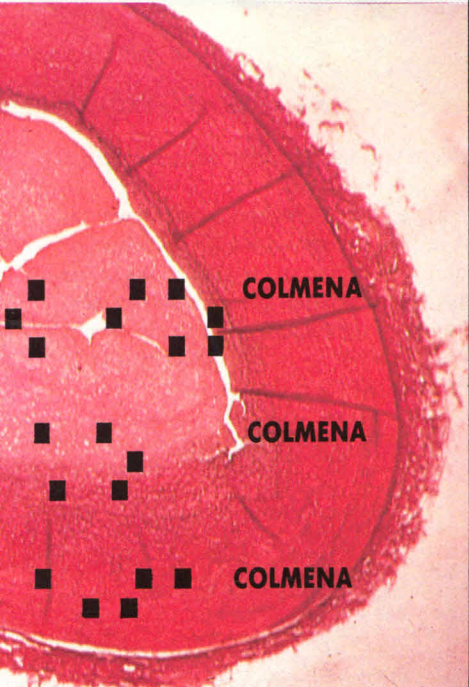
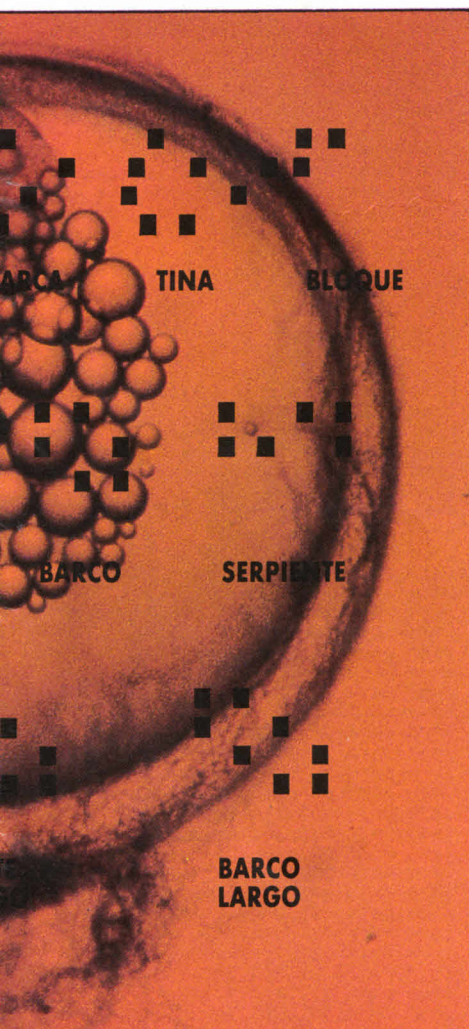


gráfico n.º 1



figuras más interesantes que se pueden presentar a través de las diferentes fases del juego de la vida. En el primer gráfico podemos observar algunas de las formas estables más corrientes. Dichas formas son las fases iniciales de muchas configuraciones, y no varían a lo largo del tiempo, a no ser que otras configuraciones se aproximen a ellas.



Una de las figuras más interesantes con que nos podemos encontrar, es el llamado «deslizador», formado por cinco casillas, y que podemos ver en el gráfico 2.

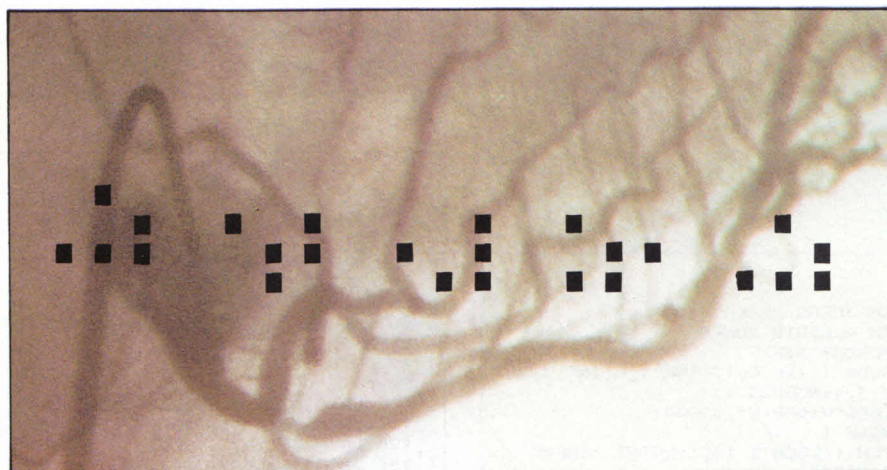


gráfico n.º 2

Dicho deslizador, al cabo de dos latidos, se desplaza y queda reflejado respecto de una recta diagonal. Al cabo de otros dos latidos, se vuelve a enderezar, desplazándose una cuadrícula digonalmente hacia abajo y hacia la derecha de su posición inicial.

Una vez explicadas las curiosidades más interesantes del juego de la vida, pasaremos ahora a ver cómo funciona el programa que aparece al final del artículo.

EL PROGRAMA

El programa consta de un bloque en Basic y otro en código máquina. El primero de ellos se encarga de imprimir en pantalla el menú de opciones que posee el juego de la vida, así como todas las teclas que pueden ser utilizadas en cada parte del programa.

Las tres opciones disponibles, son las siguientes: Creación de pantalla, Pulsaciones y Pantalla aleatoria.

La primera de ellas nos ofrece la posibilidad de crear nuestra propia pantalla utilizando las teclas de cursor para desplazarnos por la pantalla.

Pulsando la tecla «P» podremos pintar utilizando los cursores y pulsando la tecla «O» podremos borrar o desplazarnos por la pantalla sin pintar.

Una vez hayamos finalizado nuestra pantalla bastará con pulsar la tecla «COPIA» para que empiecen a producirse los latidos.

Cuando estemos en la fase de pulsaciones, tendremos las opciones de parar las pulsaciones, pulsando la tecla «P» o bien retornar el menú principal pulsando la tecla «B».

Cuando nos encontremos en «PAUSA», podremos volver a iniciar los latidos pulsando la tecla «E» o bien salvar en disco o cinta la pantalla actual, pulsando la tecla «S».

La segunda de las opciones con la que nos encontramos, es la de entrar directamente a la fase de pulsaciones. Lógicamente si utilizamos esta opción cuando no tengamos ningun

efecto. Así pues, esta opción servirá cuando estando en la fase de pulsaciones se haya vuelto al menú y se desee retornar a las pulsaciones.

Por último, tenemos la opción de crear una pantalla aleatoria. Eligiendo esta opción, nuestro **Amstrad** creará una pantalla al azar y después de esto pasará directamente a la fase de pulsaciones.

Veamos ahora cuáles son los pasos que realiza el programa en código máquina.

En primer lugar, se crea un buffer de 1.200 bytes, sobre el cual se producen todos los cálculos para comprobar cuáles son las células que sobreviven, nacen o mueren.

Dentro de ese buffer de trabajo, cada bit contiene la información de una casilla, y puede tener dos estados: 1 indica que la célula está viva y 0 indicará que dicha célula está muerta.

Así pues, en cada latido se deben comprobar cada uno de los bits que componen ese buffer de 1.200 bytes, por lo tanto deberemos tratar 1.200*8 bits.

Como hemos dicho anteriormente las reglas sobre las que se basa el juego contemplan las ocho casillas inmediatamente próximas a la que se está estudiando.

De este modo, cuando deseemos mirar lo que ocurriría con un bit en concreto, deberemos mirar cada uno de los ocho bits que lo rodean.

Si por ejemplo deseamos mirar el bit 4 de un byte cualquiera dentro del buffer, deberemos mirar los siguientes bits de los bytes que se indican:

a	b	c
d	x	e
f	g	h

Así pues, teniendo en cuenta que el número de bytes que contiene cada fila del buffer

es 10 y si cargamos en el registro indexado IX la posición de memoria en la cual está el bit que intentamos investigar, los bits que deberemos chequear serán los siguientes:

- a. bit 3 de (IX-10)
- b. bit 4 de (IX-10)
- c. bit 5 de (IX-10)

- d. bit 3 de (IX+0)
- e. bit 5 de (IX+0)
- f. bit 3 de (IX+10)
- g. bit 4 de (IX+10)
- h. bit 5 de (IX+10)

De esta forma cuando uno de estos bits esté puesto a uno, llamamos a una rutina que se encarga de incrementar el contador que nos

indicará finalmente cuántos de esos ocho bits están puestos a uno.

Así pues, una vez revisados cada uno de los bits vecinos, tomaremos el valor de dicho contador, y de esta forma se podrá decidir, de acuerdo con las reglas indicadas anteriormente, si ese bit debe sobrevivir o morir en caso de que esté vivo, o bien si debe nacer en el caso de que esté muerto.

```

1 REM JUEGO DE LA VIDA
2 REM ALBERTO SUNER
10 MEMORY &3FFF
20 MODE 1:INK 0,13:INK 1,0:INK 2,20
:INK 3,1:BORDER 13
30 LOAD"VIDABIN",&A000
40 MODE 1
50 PEN 1:LOCATE 13,1:PRINT "JUEGO D
E LA VIDA"
60 PEN 2:LOCATE 5,2:PRINT "MENU":PE
N 3
70 LOCATE 5,4:PRINT "1.....
CREACION PANTALLA"
80 LOCATE 5,5:PRINT "2.....
PULSACIONES"
90 LOCATE 5,6:PRINT "3.....
PANTALLA ALEATORIA"
100 PEN 2:LOCATE 5,8:PRINT "CREACIO
N DE PANTALLA"
110 PEN 3:LOCATE 5,10:PRINT "CURSOR
ES..... MUEVEN PUNTO"
120 LOCATE 5,11:PRINT "COPIA.....
.. LATIDOS"
130 LOCATE 5,12:PRINT "P.....
.. PINTAR"
140 LOCATE 5,13:PRINT "O.....
.. BORRAR"
150 LOCATE 5,14:PRINT "L.....
.. CARGA PANTALLA"
160 LOCATE 5,15:PRINT "B.....
.. MENU"
170 PEN 2:LOCATE 5,17:PRINT "LATIDO
S":PEN 3
180 LOCATE 5,19:PRINT "P.....
.. PAUSA"
190 LOCATE 5,20:PRINT "B.....
.. MENU"
200 PEN 2:LOCATE 5,22:PRINT "PAUSA"
:PEN 3
210 LOCATE 5,24:PRINT "S.....
.. SALVA PANTALLA"
220 LOCATE 5,25:PRINT "E.....
.. SEGUIR CON LATIDOS"
230 IF INKEY(64)=0 THEN 350
240 IF INKEY(65)=0 THEN 390
250 IF INKEY(57)=0 THEN 270
260 GOTO 230
270 MODE 2
280 FOR N=&9000 TO &9200
290 POKE N,INT(RND*256)
300 NEXT
310 POKE &A553,0:POKE &A554,0
320 GOSUB 430
330 CALL &A134
340 GOTO 40
350 MODE 2
360 GOSUB 430
370 CALL &A000
380 GOTO 40
390 MODE 2
400 GOSUB 430
410 CALL &A134
420 GOTO 40
430 LOCATE 30,25:PRINT "LATIDOS-000
00"
440 RETURN

```

```

10 REM JUEGO DE LA VIDA
20 REM PROGRAMA CARGADOR
30 FOR N=&A000 TO &A5A3
40 READ A:SUMA=SUMA+A
50 POKE N,A
60 NEXT
70 IF SUMA<173577 THEN PRINT "ERRO
R EN DATAS"
80 DATA 33,0,0,34,83,165,33
90 DATA 0,144,17,1,144,1,176
100 DATA 4,54,0,237,176,33,165
110 DATA 144,34,6,165,62,54,205
120 DATA 30,187,192,62,36,205,30
130 DATA 187,40,3,205,58,165,175
140 DATA 205,30,187,40,3,205,238
150 DATA 160,62,2,205,30,187,40
160 DATA 3,205,252,160,62,8,205
170 DATA 30,187,40,3,205,7,161
180 DATA 62,1,205,30,187,40,3
190 DATA 205,29,161,62,27,205,30
200 DATA 187,40,4,175,50,9,165
210 DATA 62,34,205,30,187,40,5
220 DATA 62,1,50,9,165,205,114
230 DATA 160,62,9,205,30,187,194
240 DATA 52,161,24,165,42,6,165
250 DATA 58,8,165,254,0,32,2
260 DATA 203,198,254,1,32,2,203
270 DATA 204,254,2,32,2,203,214
280 DATA 254,3,32,2,203,222,254
290 DATA 4,32,2,203,230,254,5
300 DATA 32,2,203,238,254,6,32
310 DATA 2,203,246,254,7,32,2
320 DATA 203,254,205,57,144,58,9
330 DATA 165,167,32,1,201,42,6
340 DATA 165,58,8,165,254,0,32
350 DATA 2,203,134,254,1,32,2
360 DATA 203,142,254,2,32,2,203
370 DATA 150,254,3,32,2,203,158
380 DATA 254,4,32,2,203,166,254
390 DATA 5,32,2,203,174,254,6
400 DATA 32,2,203,182,254,7,32
410 DATA 2,203,190,205,57,164,201
420 DATA 42,6,165,17,10,0,55
430 DATA 63,237,82,34,6,165,201
440 DATA 42,6,165,17,10,0,25
450 DATA 34,6,165,201,58,8,165
460 DATA 60,50,8,165,254,8,192
470 DATA 42,6,165,43,34,6,165
480 DATA 175,50,8,165,201,58,8
490 DATA 165,61,50,8,165,254,255
500 DATA 192,42,6,165,35,34,6
510 DATA 165,62,7,50,8,165,201
520 DATA 33,0,149,17,1,149,1
530 DATA 88,2,54,0,237,176,33
540 DATA 0,149,221,33,0,144,1
550 DATA 88,2,221,94,0,22,0
560 DATA 203,123,40,11,205,62,162
570 DATA 205,14,164,204,22,164,24
580 DATA 9,205,62,162,205,9,164
590 DATA 204,22,164,22,0,203,115
600 DATA 40,11,205,119,162,205,14
610 DATA 164,204,25,164,24,9,205
620 DATA 119,162,205,9,164,204,25
630 DATA 164,22,0,203,107,40,11
640 DATA 205,176,162,205,14,164,204
650 DATA 28,164,24,9,205,176,162
660 DATA 205,9,164,204,28,164,22
670 DATA 0,203,99,40,11,205,233
680 DATA 162,205,14,164,204,31,164
690 DATA 24,9,205,233,162,205,9
700 DATA 164,204,31,164,22,0,203
710 DATA 91,40,11,205,34,163,205
720 DATA 14,164,204,34,164,24,9
730 DATA 205,34,163,205,9,164,204
740 DATA 34,164,22,0,203,83,40
750 DATA 11,205,91,163,205,14,164
760 DATA 204,37,164,24,9,205,91

```

```

770 DATA 163,205,9,164,204,37,164
780 DATA 22,0,203,75,40,11,205
790 DATA 148,163,205,14,164,204,40
800 DATA 164,24,9,205,148,163,205
810 DATA 9,164,204,40,164,22,0
820 DATA 203,67,40,11,205,205,163
830 DATA 205,14,164,204,43,164,24
840 DATA 9,205,205,163,205,9,164
850 DATA 204,43,164,221,35,35,11
860 DATA 120,177,194,75,161,205,46
870 DATA 164,205,85,165,62,27,205
880 DATA 30,187,196,10,165,62,54
890 DATA 205,30,187,202,52,161,201
900 DATA 221,203,255,70,196,7,164
910 DATA 221,203,0,118,196,7,164
920 DATA 221,203,9,70,196,7,164
930 DATA 221,203,10,126,196,7,164
940 DATA 221,203,10,118,196,7,164
950 DATA 221,203,246,118,196,7,164
960 DATA 221,203,246,126,196,7,164
970 DATA 221,203,245,70,196,7,164
980 DATA 201,221,203,0,126,196,7
990 DATA 164,221,203,0,110,196,7
1000 DATA 164,221,203,246,126,196,7
1010 DATA 164,221,203,246,118,196,7
1020 DATA 164,221,203,246,110,196,7
1030 DATA 164,221,203,10,126,196,7
1040 DATA 164,221,203,10,118,196,7
1050 DATA 164,221,203,10,110,196,7
1060 DATA 164,201,221,203,0,118,196
1070 DATA 7,164,221,203,0,102,196
1080 DATA 7,164,221,203,246,118,196
1090 DATA 7,164,221,203,246,110,196
1100 DATA 7,164,221,203,246,102,196
1110 DATA 7,164,221,203,10,118,196
1120 DATA 7,164,221,203,10,110,196
1130 DATA 7,164,221,203,10,102,196
1140 DATA 7,164,201,221,203,0,110
1150 DATA 196,7,164,221,203,0,94
1160 DATA 196,7,164,221,203,246,110
1170 DATA 196,7,164,221,203,246,102
1180 DATA 196,7,164,221,203,246,94
1190 DATA 196,7,164,221,203,10,110
1200 DATA 196,7,164,221,203,10,102
1210 DATA 196,7,164,221,203,10,94
1220 DATA 196,7,164,201,221,203,0
1230 DATA 102,196,7,164,221,203,0
1240 DATA 86,196,7,164,221,203,246
1250 DATA 102,196,7,164,221,203,246
1260 DATA 94,196,7,164,221,203,246
1270 DATA 86,196,7,164,221,203,10
1280 DATA 102,196,7,164,221,203,10
1290 DATA 94,196,7,164,221,203,10
1300 DATA 86,196,7,164,201,221,203
1310 DATA 0,94,196,7,164,221,203
1320 DATA 0,78,196,7,164,221,203
1330 DATA 246,94,196,7,164,221,203
1340 DATA 246,86,196,7,164,221,203
1350 DATA 246,78,196,7,164,221,203
1360 DATA 10,94,196,7,164,221,203
1370 DATA 10,86,196,7,164,221,203
1380 DATA 10,78,196,7,164,201,221
1390 DATA 203,0,86,196,7,164,221
1400 DATA 203,0,70,196,7,164,221
1410 DATA 203,246,86,196,7,164,221
1420 DATA 203,246,78,196,7,164,221
1430 DATA 203,246,70,196,7,164,221
1440 DATA 203,10,86,196,7,164,221
1450 DATA 203,10,78,196,7,164,221
1460 DATA 203,10,70,196,7,164,201
1470 DATA 221,203,0,78,196,7,164
1480 DATA 221,203,1,126,196,7,164
1490 DATA 221,203,246,78,196,7,164
1500 DATA 221,203,246,70,196,7,164
1510 DATA 221,203,247,126,196,7,164
1520 DATA 221,203,10,78,196,7,164
1530 DATA 221,203,10,70,196,7,164

```


1540 DATA 221,203,11,126,196,7,164
 1550 DATA 201,201,20,201,122,254,3
 1560 DATA 200,201,122,254,2,200,254
 1570 DATA 3,200,201,203,254,201,203
 1580 DATA 246,201,203,238,201,203,2
 30
 1590 DATA 201,203,222,201,203,214,2
 01
 1600 DATA 203,206,201,203,198,201,3
 3
 1610 DATA 0,149,17,0,144,1,88
 1620 DATA 2,237,176,33,0,64,17
 1630 DATA 1,64,1,0,64,54,0
 1640 DATA 237,176,205,77,164,205,19
 5
 1650 DATA 164,201,17,0,64,33,0
 1660 DATA 144,6,60,197,237,83,182
 1670 DATA 164,6,10,175,203,126,196
 1680 DATA 184,164,18,175,19,203,118
 1690 DATA 196,184,164,18,175,19,203
 1700 DATA 110,196,184,164,18,175,19
 1710 DATA 203,102,196,184,164,18,17
 5
 1720 DATA 19,203,94,196,184,164,18
 1730 DATA 175,19,203,86,196,184,164
 1740 DATA 18,175,19,203,78,196,184
 1750 DATA 164,18,175,19,203,70,196
 1760 DATA 184,164,18,19,35,16,189
 1770 DATA 229,42,182,164,1,80,0
 1780 DATA 237,176,33,80,0,25,235
 1790 DATA 225,193,16,164,237,83,182
 1800 DATA 164,201,0,0,203,247,203
 1810 DATA 255,203,239,203,231,203,2
 23
 1820 DATA 201,33,1,1,17,0,64
 1830 DATA 213,235,33,176,191,66,22
 1840 DATA 0,29,25,17,80,0,25
 1850 DATA 16,253,221,225,6,192,24
 1860 DATA 21,124,230,56,254,56,40
 1870 DATA 6,124,198,8,103,24,8
 1880 DATA 17,80,0,124,238,56,103
 1890 DATA 25,197,229,6,80,221,126
 1900 DATA 0,119,221,35,35,16,247
 1910 DATA 225,193,16,218,201,0,144
 1920 DATA 0,0,62,58,205,30,187
 1930 DATA 192,62,60,205,30,187,194
 1940 DATA 26,165,24,240,6,4,33
 1950 DATA 79,165,17,0,169,205,140
 1960 DATA 188,33,0,149,17,88,2
 1970 DATA 1,0,0,62,2,205,152
 1980 DATA 188,205,143,188,205,146,1
 88
 1990 DATA 201,6,4,33,79,165,17
 2000 DATA 0,169,205,119,188,33,0
 2010 DATA 144,205,131,188,205,122,1
 88
 2020 DATA 201,86,73,68,65,0,0
 2030 DATA 38,38,46,25,205,117,187
 2040 DATA 42,83,165,35,34,83,165
 2050 DATA 55,17,16,39,35,62,47
 2060 DATA 60,237,82,48,251,205,152
 2070 DATA 165,17,232,3,60,237,82
 2080 DATA 48,251,205,152,165,17,100
 2090 DATA 0,60,237,82,48,251,205
 2100 DATA 152,165,17,10,0,60,237
 2110 DATA 82,48,251,205,152,165,133
 2120 DATA 205,152,165,201,205,90,18
 7
 2130 DATA 62,47,32,1,35,25,35
 2140 DATA 201,0,0,0,0,0,0

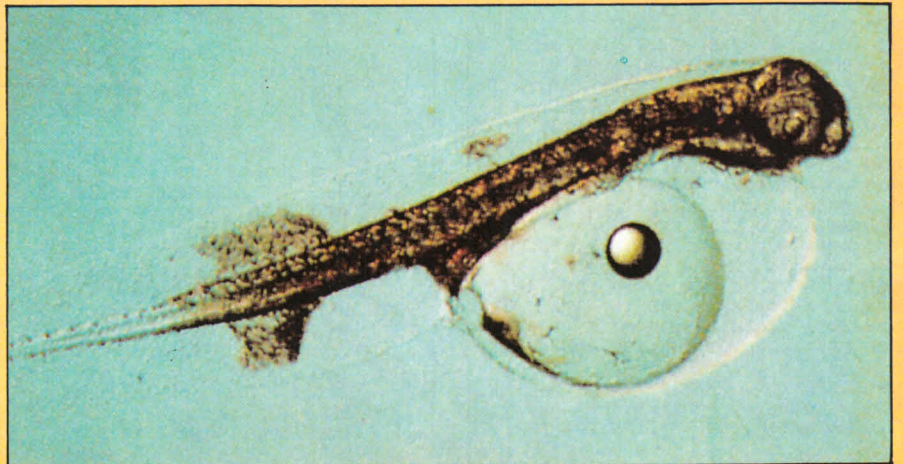


P ara que tus dedos
 no realicen el trabajo duro, M.H. AMS-
 TRAD lo hace por ti. Todos los listados que incluyen
 este logotipo se encuentran a tu disposición en un cas-
 sette mensual, solicítanoslo.

10 ; JUEGO DE LA VIDA
 20 ORG #A000
 30 LD HL,0
 40 LD (LATID),HL
 50 LD HL,#9000
 60 LD DE,#9001
 70 LD BC,1200
 80 LD (HL),0
 90 LDIR
 100 LD HL,#90A5
 110 LD (POSCUR),HL
 120 TEC: LD A,54
 130 CALL #BB1E
 140 RET NZ
 150 LD A,36
 160 CALL #BB1E
 170 JR Z,TECLO
 180 CALL LOAD
 190 TECLO: XOR A
 200 CALL #BB1E
 210 JR Z,TECL1
 220 CALL UP
 230 TECL1: LD A,2
 240 CALL #BB1E
 250 JR Z,TECL2
 260 CALL DOWN
 270 TECL2: LD A,8
 280 CALL #BB1E
 290 JR Z,TECL3
 300 CALL LEFT
 310 TECL3: LD A,1
 320 CALL #BB1E
 330 JR Z,TECL4
 340 CALL RIGHT
 350 TECL4: LD A,27
 360 CALL #BB1E
 370 JR Z,TECL5
 380 XOR A
 390 LD (PINTI),A
 400 TECL5: LD A,34
 410 CALL #BB1E
 420 JR Z,TECL6
 430 LD A,1
 440 LD (PINTI),A
 450 TECL6: CALL PINBU
 460 LD A,9
 470 CALL #BB1E
 480 JP NZ,INIC
 490 JR TEC
 500 PINBU: LD HL,(POSCUR)
 510 LD A,(CONBY)
 520 CP 0
 530 JR NZ,BYT1
 540 SET 0,(HL)
 550 BYT1: CP 1
 560 JR NZ,BYT2
 570 SET 1,(HL)
 580 BYT2: CP 2
 590 JR NZ,BYT3
 600 SET 2,(HL)
 610 BYT3: CP 3
 620 JR NZ,BYT4
 630 SET 3,(HL)
 640 BYT4: CP 4
 650 JR NZ,BYT5
 660 SET 4,(HL)
 670 BYT5: CP 5
 680 JR NZ,BYT6
 690 SET 5,(HL)
 700 BYT6: CP 6
 710 JR NZ,BYT7
 720 SET 6,(HL)
 730 BYT7: CP 7
 740 JR NZ,BYT8
 750 SET 7,(HL)
 760 BYT8: CALL PINTU
 770 LD A,(PINTI)
 780 AND A
 790 JR NZ,BORRA
 800 RET
 810 BORRA: LD HL,(POSCUR)
 820 LD A,(CONBY)
 830 CP 0
 840 JR NZ,BITO
 850 RES 0,(HL)
 860 BITO: CP 1
 870 JR NZ,BIT1

880 RES 1,(HL)
 890 BIT1: CP 2
 900 JR NZ,BIT2
 910 RES 2,(HL)
 920 BIT2: CP 3
 930 JR NZ,BIT3
 940 RES 3,(HL)
 950 BIT3: CP 4
 960 JR NZ,BIT4
 970 RES 4,(HL)
 980 BIT4: CP 5
 990 JR NZ,BIT5
 1000 RES 5,(HL)
 1010 BIT5: CP 6
 1020 JR NZ,BIT6
 1030 RES 6,(HL)
 1040 BIT6: CP 7
 1050 JR NZ,BITB
 1060 RES 7,(HL)
 1070 BITB: CALL PINTU
 1080 RET
 1090 UP: LD HL,(POSCUR)
 1100 LD DE,10
 1110 SCF
 1120 CCF
 1130 SBC HL,DE
 1140 LD (POSCUR),HL
 1150 RET
 1160 DOWN: LD HL,(POSCUR)
 1170 LD DE,10
 1180 ADD HL,DE
 1190 LD (POSCUR),HL
 1200 RET
 1210 LEFT: LD A,(CONBY)
 1220 INC A
 1230 LD (CONBY),A
 1240 CP B
 1250 RET NZ
 1260 LEF1: LD HL,(POSCUR)
 1270 DEC HL
 1280 LD (POSCUR),HL
 1290 XOR A
 1300 LD (CONBY),A
 1310 RET
 1320 RIGHT: LD A,(CONBY)
 1330 DEC A
 1340 LD (CONBY),A
 1350 CP 255
 1360 RET NZ
 1370 LD HL,(POSCUR)
 1380 INC HL
 1390 LD (POSCUR),HL
 1400 LD A,7
 1410 LD (CONBY),A
 1420 RET
 1430 ;
 1440 INIC: LD HL,#9500
 1450 ;
 1460 LD DE,#9501
 1470 LD BC,600
 1480 LD (HL),0
 1490 LDIR
 1500 LD HL,#9500
 1510 LD IX,#9000
 1520 LD BC,600
 1530 MIRBU: LD E,(IX)
 1540 LD D,0
 1550 BIT 7,E
 1560 JR Z,PASB11
 1570 CALL MIR7
 1580 CALL MIRBIT
 1590 CALL Z,PON7
 1600 JR PASB12
 1610 PASB11: CALL MIR7
 1620 CALL MIRBI1
 1630 CALL Z,PON7
 1640 PASB12: LD D,0
 1650 BIT 6,E
 1660 JR Z,PASB13
 1670 CALL MIR6
 1680 CALL MIRBIT
 1690 CALL Z,PON6
 1700 JR PASB14
 1710 PASB13: CALL MIR6
 1720 CALL MIRBI1
 1730 CALL Z,PON6
 1740 PASB14: LD D,0

1750	BIT 5,E	2640	CALL NZ,INCCON	3250	BIT 2,(IX-10)
1760	JR Z,PASB15	2650	RET	3260	CALL NZ,INCCON
1770	CALL MIR5	2660	MIR6: BIT 7,(IX+0)	3270	BIT 4,(IX+10)
1780	CALL MIRBIT	2670	CALL NZ,INCCON	3280	CALL NZ,INCCON
1790	CALL Z,PON5	2680	BIT 5,(IX+0)	3290	BIT 3,(IX+10)
1800	JR PASB16	2690	CALL NZ,INCCON	3300	CALL NZ,INCCON
1810	PASB15: CALL MIR5	2700	BIT 7,(IX-10)	3310	BIT 2,(IX+10)
1820	CALL MIRB11	2710	CALL NZ,INCCON	3320	CALL NZ,INCCON
1830	CALL Z,PON5	2720	BIT 6,(IX-10)	3330	RET
1840	PASB16: LD D,0	2730	CALL NZ,INCCON	3340	MIR2: BIT 3,(IX+0)
1850	BIT 4,E	2740	BIT 5,(IX-10)	3350	CALL NZ,INCCON
1860	JR Z,PASB17	2750	CALL NZ,INCCON	3360	BIT 1,(IX+0)
1870	CALL MIR4	2760	BIT 7,(IX+10)	3370	CALL NZ,INCCON
1880	CALL MIRBIT	2770	CALL NZ,INCCON	3380	BIT 3,(IX-10)
1890	CALL Z,PON4	2780	BIT 6,(IX+10)	3390	CALL NZ,INCCON
1900	JR PASB18	2790	CALL NZ,INCCON	3400	BIT 2,(IX-10)
1910	PASB17: CALL MIR4	2800	BIT 5,(IX+10)	3410	CALL NZ,INCCON
1920	CALL MIRB11	2810	CALL NZ,INCCON	3420	BIT 1,(IX-10)
1930	CALL Z,PON4	2820	RET	3430	CALL NZ,INCCON
1940	PASB18: LD D,0	2830	MIR5: BIT 6,(IX+0)	3440	BIT 3,(IX+10)
1950	BIT 3,E	2840	CALL NZ,INCCON	3450	CALL NZ,INCCON
1960	JR Z,PASB19	2850	BIT 4,(IX+0)	3460	BIT 2,(IX+10)
1970	CALL MIR3	2860	CALL NZ,INCCON	3470	CALL NZ,INCCON
1980	CALL MIRBIT	2870	BIT 6,(IX-10)	3480	BIT 1,(IX+10)
1990	CALL Z,PON3	2880	CALL NZ,INCCON	3490	CALL NZ,INCCON
2000	JR PASB10	2890	BIT 5,(IX-10)	3500	RET
2010	PASB19: CALL MIR3	2900	CALL NZ,INCCON	3510	MIR1: BIT 2,(IX+0)
2020	CALL MIRB11	2910	BIT 4,(IX-10)	3520	CALL NZ,INCCON
2030	CALL Z,PON3	2920	CALL NZ,INCCON	3530	BIT 0,(IX+0)
2040	PASB10: LD D,0	2930	BIT 6,(IX+10)	3540	CALL NZ,INCCON
2050	BIT 2,E				
2060	JR Z,PASB11				
2070	CALL MIR2				
2080	CALL MIRBIT				
2090	CALL Z,PON2				
2100	JR PASB12				
2110	PASB11: CALL MIR2				
2120	CALL MIRB11				
2130	CALL Z,PON2				
2140	PASB12: LD D,0				
2150	BIT 1,E				
2160	JR Z,PASB13				
2170	CALL MIR1				
2180	CALL MIRBIT				
2190	CALL Z,PON1				
2200	JR PASB14				
2210	PASB13: CALL MIR1				
2220	CALL MIRB11				
2230	CALL Z,PON1				
2240	PASB14: LD D,0				
2250	BIT 0,E				
2260	JR Z,PASB15				
2270	CALL MIRO				
2280	CALL MIRBIT				
2290	CALL Z,PON0				
2300	JR PASB16				
2310	PASB15: CALL MIRO				
2320	CALL MIRB11				
2330	CALL Z,PON0	2940	CALL NZ,INCCON	3550	BIT 2,(IX-10)
2340	PASB16: INC IX	2950	BIT 5,(IX+10)	3560	CALL NZ,INCCON
2350	INC HL	2960	CALL NZ,INCCON	3570	BIT 1,(IX-10)
2360	DEC BC	2970	BIT 4,(IX+10)	3580	CALL NZ,INCCON
2370	LD A,B	2980	CALL NZ,INCCON	3590	BIT 0,(IX-10)
2380	OR C	2990	RET	3600	CALL NZ,INCCON
2390	JP NZ,MIRBU	3000	MIR4: BIT 5,(IX+0)	3610	BIT 2,(IX+10)
2400	CALL PINTA	3010	CALL NZ,INCCON	3620	CALL NZ,INCCON
2410	CALL INCLAT	3020	BIT 3,(IX+0)	3630	BIT 1,(IX+10)
2420	LD A,27	3030	CALL NZ,INCCON	3640	CALL NZ,INCCON
2430	CALL #BB1E	3040	BIT 5,(IX-10)	3650	BIT 0,(IX+10)
2440	CALL NZ,ESPER	3050	CALL NZ,INCCON	3660	CALL NZ,INCCON
2450	LD A,54	3060	BIT 4,(IX-10)	3670	RET
2460	CALL #BB1E	3070	CALL NZ,INCCON	3680	MIRO: BIT 1,(IX+0)
2470	JP Z,INIC	3080	BIT 3,(IX-10)	3690	CALL NZ,INCCON
2480	RET	3090	CALL NZ,INCCON	3700	BIT 7,(IX+1)
2490	MIR7: BIT 0,(IX-1)	3100	BIT 5,(IX+10)	3710	CALL NZ,INCCON
2500	CALL NZ,INCCON	3110	CALL NZ,INCCON	3720	BIT 1,(IX-10)
2510	BIT 6,(IX+0)	3120	BIT 4,(IX+10)	3730	CALL NZ,INCCON
2520	CALL NZ,INCCON	3130	CALL NZ,INCCON	3740	BIT 0,(IX-10)
2530	BIT 0,(IX+9)	3140	BIT 3,(IX+10)	3750	CALL NZ,INCCON
2540	CALL NZ,INCCON	3150	CALL NZ,INCCON	3760	BIT 7,(IX-9)
2550	BIT 7,(IX+10)	3160	RET	3770	CALL NZ,INCCON
2560	CALL NZ,INCCON	3170	MIR3: BIT 4,(IX+0)	3780	BIT 1,(IX+10)
2570	BIT 6,(IX+10)	3180	CALL NZ,INCCON	3790	CALL NZ,INCCON
2580	CALL NZ,INCCON	3190	BIT 2,(IX+0)	3800	BIT 0,(IX+10)
2590	BIT 6,(IX-10)	3200	CALL NZ,INCCON	3810	CALL NZ,INCCON
2600	CALL NZ,INCCON	3210	BIT 4,(IX-10)	3820	BIT 7,(IX+11)
2610	BIT 7,(IX-10)	3220	CALL NZ,INCCON	3830	CALL NZ,INCCON
2620	CALL NZ,INCCON	3230	BIT 3,(IX-10)	3840	RET
2630	BIT 0,(IX-11)	3240	CALL NZ,INCCON	3850	RET



```

3860 INCCON: INC D
3870 RET
3880 MIRBI1: LD A,D
3890 CP 3
3900 RET Z
3910 LD A,D
3920 MIRBIT: LD A,D
3930 CP 2
3940 RET Z
3950 CP 3
3960 RET Z
3970 LD 7,(HL)
3980 PON7: SET 7,(HL)
3990 RET
4000 PON6: SET 6,(HL)
4010 RET
4020 PON5: SET 5,(HL)
4030 RET
4040 PON4: SET 4,(HL)
4050 RET
4060 PON3: SET 3,(HL)
4070 RET
4080 PON2: SET 2,(HL)
4090 RET
4100 PON1: SET 1,(HL)
4110 RET
4120 PON0: SET 0,(HL)
4130 RET
4140 PINTA: LD HL,#9500
4150 LD DE,#9000
4160 LD BC,600
4170 LDIR
4180 PINTU: LD HL,#4000
4190 LD DE,#4001
4200 LD BC,#4000
4210 LD (HL),0
4220 LDIR
4230 CALL MIRA
4240 CALL IMPRE
4250 RET
4260 MIRA: LD DE,#4000
4270 LD HL,#9000
4280 LD B,60
4290 LLL2: PUSH BC
4300 LD (POSIN),DE
4310 LD B,10
4320 LLL1: XOR A
4330 BIT 7,(HL)
4340 CALL NZ,PON67
4350 LD (DE),A
4360 XOR A
4370 INC DE
4380 BIT 6,(HL)
4390 CALL NZ,PON67
4400 LD (DE),A
4410 XOR A
4420 INC DE
4430 BIT 5,(HL)
4440 CALL NZ,PON67
4450 LD (DE),A
4460 XOR A
4470 INC DE
4480 BIT 4,(HL)
4490 CALL NZ,PON67
4500 LD (DE),A
4510 XOR A
4520 INC DE
4530 BIT 3,(HL)
4540 CALL NZ,PON67
4550 LD (DE),A
4560 XOR A
4570 INC DE
4580 BIT 2,(HL)
4590 CALL NZ,PON67
4600 LD (DE),A
4610 XOR A
4620 INC DE
4630 BIT 1,(HL)
4640 CALL NZ,PON67
4650 LD (DE),A
4660 XOR A
4670 INC DE
4680 BIT 0,(HL)
4690 CALL NZ,PON67
4700 LD (DE),A
4710 INC DE
4720 INC HL
4730 DJNZ LLL1

```

```

4740 PUSH HL
4750 LD HL,(POSIN)
4760 LD BC,80
4770 LDIR
4780 LD HL,80
4790 ADD HL,DE
4800 EX DE,HL
4810 POP HL
4820 POP BC
4830 DJNZ LLL2
4840 LD (POSIN),DE
4850 RET
4860 ;
4870 POSIN: DEFS 2
4880 ;
4890 PON67: SET 6,A
4900 SET 7,A
4910 SET 5,A
4920 SET 4,A
4930 SET 3,A
4940 RET
4950 ;RUTINA-IMPRESION
4960 ;H-POSICION-VERTICAL-INICIO-1
4970 ;L-POSICION-HORIZONTAL-INICIO-1
4980 ;DE-DIRECCION-GRAFICO
4990 ;
5000 IMPRE: LD HL,#0101
5010 LD DE,#4000
5020 PUSH DE
5030 EX DE,HL
5040 LD HL,#C000-80
5050 LD B,D
5060 LD D,0
5070 DEC E
5080 ADD HL,DE
5090 LD DE,80
5100 S_BUC: ADD HL,DE
5110 DJNZ S_BUC
5120 POP IX
5130 LD B,200
5140 JR COLOC
5150 P_BUC: LD A,H
5160 AND 56
5170 CP 56
5180 JR Z,P_PAS
5190 LD A,H
5200 ADD A,B
5210 LD H,A
5220 JR COLOC
5230 P_PAS: LD DE,0080
5240 LD A,H
5250 XOR 56
5260 LD H,A
5270 ADD HL,DE
5280 COLOC: PUSH BC
5290 PUSH HL
5300 LD B,80
5310 P_BUC1: LD A,(IX+0)
5320 LD (HL),A
5330 INC IX
5340 INC HL
5350 DJNZ P_BUC1
5360 POP HL
5370 POP BC
5380 DJNZ P_BUC
5390 RET
5400 DEFS 0
5410 POSCUR: DEFW #9000
5420 CONBY: DEFB 0
5430 PINTI: DEFB 0
5440 ESPER: LD A,58
5450 CALL #BB1E
5460 RET NZ
5470 LD A,60
5480 CALL #BB1E
5490 JP NZ,SAVE
5500 JR ESPER
5510 SAVE: LD B,4
5520 LD HL,NAME
5530 LD DE,#A900
5540 CALL #BCBC
5550 LD HL,#9500
5560 LD DE,600
5570 LD BC,0
5580 LD A,2
5590 CALL #BC9B
5600 CALL #BCBF
5610 CALL #BC92

```

```

5620 RET
5630 LOAD: LD B,4
5640 LD HL,NAME
5650 LD DE,#A900
5660 CALL #BC77
5670 LD HL,#9000
5680 CALL #BC83
5690 CALL #BC7A
5700 RET
5710 NAME: DEFW "VIDA"
5720 LATID: DEFW 0
5730 INCLAT: LD H,38
5740 LD L,25
5750 CALL #BB75
5760 LD HL,(LATID)
5770 INC HL
5780 LD (LATID),HL
5790 SCF
5800 LD DE,10000
5810 INC HL
5820 LD A,47
5830 DMIL: INC A
5840 SBC HL,DE
5850 JR NC,DMIL
5860 CALL PRINT
5870 LD DE,1000
5880 MIL: INC A
5890 SBC HL,DE
5900 JR NC,MIL
5910 CALL PRINT
5920 LD DE,100
5930 CIEN: INC A
5940 SBC HL,DE
5950 JR NC,CIEN
5960 CALL PRINT
5970 LD DE,10
5980 DIEZ: INC A
5990 SBC HL,DE
6000 JR NC,DIEZ
6010 CALL PRINT
6020 ADD A,L
6030 CALL PRINT
6040 RET
6050 PRINT: CALL #BB5A
6060 LD A,47
6070 JR NZ,PAS
6080 INC HL
6090 PAS: ADD HL,DE
6100 INC HL
6110 RET

```

ETIQUETAS

BIT1	A0C6	BIT2	A0CC
BIT4	A0D8	BITS	A0DE
BIT8	A0EA	BORRA	A0B4
BYT2	A0B6	BYT3	A0BC
BYT5	A098	BYT6	A09E
BYT8	A0AA	CIEN	A580
CONBY	A508	DIEZ	A58B
DOWN	A0FC	ESPER	A50A
INCCON	A407	INCLAT	A555
LATID	A553	LEF1	A111
LLL1	A45C	LLL2	A455
MIL	A575	MIRO	A3CD
MIR2	A35B	MIR3	A322
MIR5	A2B0	MIR6	A277
MIRA	A44D	MIRBI1	A409
MIRBU	A14B	NAME	A54F
PASB10	A1D0	PASB11	A1E1
PASB13	A1FB	PASB14	A204
PASB16	A21E	PASB11	A15F
PASB13	A179	PASB14	A182
PASB16	A19C	PASB17	A1AD
PASB19	A1C7	PINBU	A074
PINTA	A42E	PINTI	A509
PONO	A42B	PON1	A42B
PON3	A422	PON4	A41F
PON6	A419	PON67	A4BB
POSCUR	A506	POSIN	A4B6
P_BUC	A4DF	P_BUC1	A4FB
RIGHT	A11D	SAVE	A51A
TEC	A019	TECLO	A029
TECL2	A03C	TECL3	A046
TECL5	A05B	TECL6	A067
		UP	A0EE
		PINTU	A439
		PON2	A425
		PON5	A41C
		PON7	A416
		PRINT	A59B
		P_PAS	A4EC
		S_BUC	A4D6
		TECL1	A032
		TECL4	A050

FORTH: POTENCIA Y VELOCIDAD EN ALTO NIVEL

David Sopena

Como todos sabemos, el lenguaje utilizado por excelencia para programar nuestros ordenadores caseros ha sido el Basic. Pero los aficionados inquietos, se habrán dado cuenta rápidamente que no es el lenguaje de programación ideal para que nuestro Amstrad actúe de una manera eficaz en todas las ocasiones y, que algunas aplicaciones como, por ejemplo, los juegos de arcade y similares, están pidiendo a gritos que utilicemos un sistema más rápido y compacto a la hora de codificarlos.



La primera solución que se nos ocurre es escribir este tipo de programas en código máquina. ¡Qué problema! A la mayoría de nosotros puede que hacerlo nos resulte todavía bastante difícil y penoso ya que se trata de un lenguaje en el que hay que «hilar» muy fino, teniendo muy claro qué es lo que queremos hacer y conociendo con precisión las instrucciones internas del micro.

Además, los programas generados son largos y de difícil seguimiento en caso de error. Tiene un gran número de instrucciones elementales entre las que nos resultará relativamente fácil perdersenos. Así que de momento...

Un camino más fácil y entretenido es utilizar alguno de los muchos lenguajes de alto nivel que ya están a disposición de nuevo Amstrad. Entre ellos están incluidos el Pascal, el Forth y el Logo y, si su ordenador está equipado con una unidad de disco, también podrá hacer sus programas en Lisp, Prolog, Fortran, C y muchos más.

Cada uno de ellos tiene sus ventajas e inconvenientes en las diferentes aplicaciones que les demos. Mientras un lenguaje puede parecer ideal para una cosa en concreto, a lo mejor resulta ser demasiado lento o necesitar demasiada memoria en otras ocasiones.

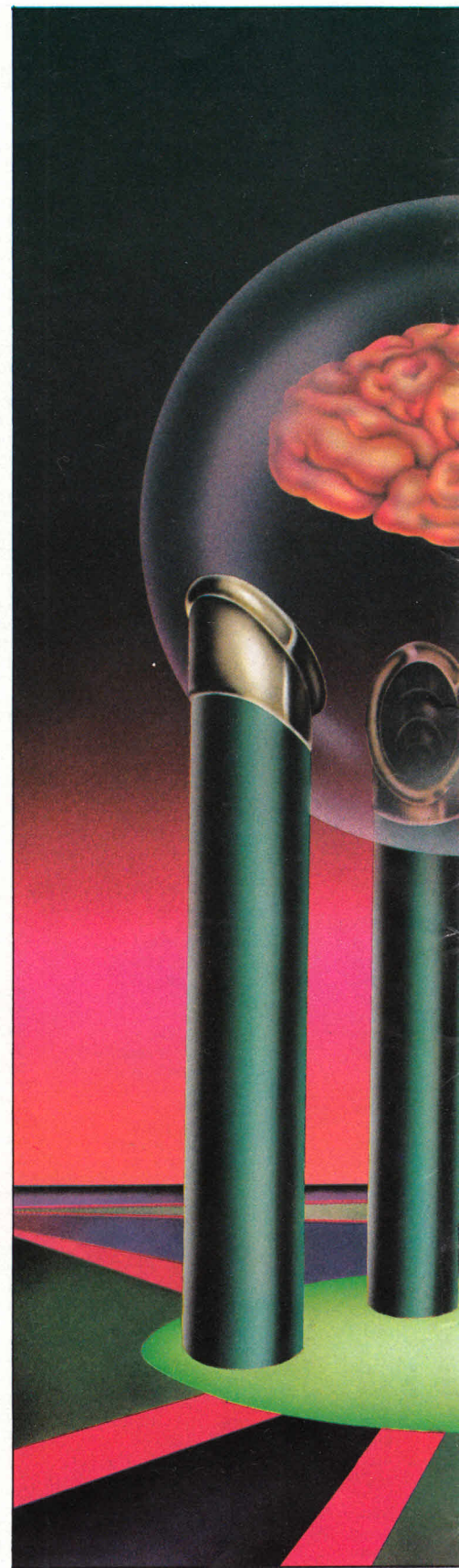
En pocas palabras. Lo que realmente estamos necesitando es un lenguaje «ideal» que, en conjunto, sea suficientemente rápido para cualquier requisito, que no «despilfarre» demasiada memoria y que, por supuesto, sea relativamente fácil de aprender y utilizar.

Forth: historia de la eficacia

El lenguaje que, en principio, se adapta mejor a estos requisitos es el Forth y, no debe sorprenderle que sea el segundo en popularidad entre los usuarios de ordenadores caseros. Es sencillo, compacto, pensado para aplicaciones de uso general, ideal para emplearlo en multitud de problemas y su aprendizaje no entraña ninguna dificultad, a pesar de su estructura y vocabulario poco usuales.

El Forth nació alrededor de 1969 y en un principio se utilizó para controlar los complejos movimientos de los grandes telescopios. Desde entonces ha tenido una amplia y variada gama de usuarios para una no menos extensa galería de aplicaciones.

Posee alguna de las características avanzadas de los lenguajes de alto nivel, tales como estructuras de bucles y análisis de condiciones muy complejas, y genera programas muy uni-



formes que se ejecutan a una gran velocidad, aproximadamente 10 veces más deprisa que el Basic. A esto hay que añadir que podemos modificar y ampliar el lenguaje dotándolo de nuevas palabras «clave» a medida que vayan surgiendo a lo largo de cualquier aplicación.



Nuestra primera tarea, antes de comenzar a desentrañar las maravillas de este nuevo lenguaje, es, por supuesto, teclear el programa que nos servirá para ejecutar en el **Amstrad** las instrucciones que escribamos en Forth.

Un intérprete para aprender

Dese cuenta de algo importante. Este programa no nos genera realmente una **versión del Forth**, simplemente simula esta operación. Su misión es convertir cada nueva palabra en un código que responda a un formato interno especial que se corresponde con unas instrucciones Basic que son las que realmente realizan el trabajo.

Y como a continuación el **Amstrad** ha de interpretar estas sentencias Basic, no se extrañe que en esta ocasión cualquier programa creado en Forth se ejecutará bastante lentamente. Pero no se desespere. Por lo demás esta versión nos permite utilizar un Forth bastante parecido al que se usa realmente. A pesar de su lentitud, nos permitirá hacer experimentos con este potente lenguaje empleando las técnicas sugeridas en este artículo. Cuando su aprendizaje haya terminado ya estaremos capacitados para decidir si Forth es **«nuestro»** lenguaje. De ser así, ya sabe lo que le tocará hacer: adquirir una de las versiones comerciales disponibles para el **Amstrad**.

Al principio puede resultarnos un poco incómodo de utilizar debido principalmente a que emplea una notación diferente a la que nosotros estamos acostumbrados. En vez de escribir instrucciones en la forma que lo hace el Basic, todas las sentencias Forth, o **«palabras»** como de ahora en adelante las conoceremos, necesitan tener sus argumentos —o números con los que trabajan— **«delante»** de cada comando y no después como es el caso de la mayor parte de los lenguajes.

El Forth: es un lenguaje basado en el concepto de stack

Por ejemplo, si queremos sumar dos números y después imprimir su resultado, lo conseguiremos con la siguiente instrucción:

```
PRINT 3+8
```

Pero en Forth no se haría así. Tendremos que escribir algo parecido a:

```
3 8+.
```

donde el punto (.) es la «palabra» Forth empleada para visualizar en la pantalla. Esta forma de escribir las instrucciones es lo que se conoce como «Notación Polaca Inversa» y le aseguramos que no es tan enrevesada como parece, así que no se sienta desanimado por esta novedad.

Sin embargo, no es la única diferencia existente. El Forth trabaja empleando una parte de la memoria llamada «stack» o pila para guardar los valores que en un determinado momento van a tener los argumentos que vamos a pasar a una **«palabra»**.

Su funcionamiento es muy sencillo. Imagine que está recogiendo platos y **«apilándolos»** uno encima del otro. Es evidente que el último plato que hayamos tomado será el que coloquemos en la parte superior de la **«pila»**. Una vez colocados, necesitamos coger uno. *¿Cuál será?*

A menos que sea un malabarista e intente demostrarlo sacando uno de los platos de abajo, lo más normal y lógico es que el elegido sea precisamente el que está colocado encima de todos. Así sucede con el «stack», el último dato que hayamos colocado en él es generalmente el primero que después vamos a sacar de allí.

Es lo que los técnicos han bautizado como memoria **«LIFO»** (o **«last input-first output»**) que quiere decir: el último elemento que entra es el primero que sale.

La manera en la que el Forth interpreta las instrucciones, o **«palabras»**, que escribimos anteriormente es bastante sencilla. Coloca en el stack los números que vamos a sumar en el orden que se los hemos dado —primero el 3 y encima el 8. A continuación se ejecuta la palabra Forth +.

El modo de operación es muy semejante en la mayor parte de las sentencias Forth: se sacan los dos números del stack, se procesan en la forma oportuna y el resultado se vuelve a guardar en el stack para que pueda ser manejado por sucesivas palabras.

En este caso, se cogen los dos datos que ocupan las posiciones más altas de la pila, se suman y se devuelve este dato al stack.

A continuación la siguiente palabra (.) extrae el número que ahora esté colocado encima (el resultado de la operación anterior) y lo saca en la pantalla. El Forth además imprime el mensaje de **«OK»** para convencernos de que la instrucción se ha ejecutado sin errores. Observe y tenga siempre en cuenta que tanto los datos como las palabras Forth han de estar separadas por un espacio en blanco. ¡No lo olvide!

Estos comandos dejan el stack exactamente igual que estaba antes de ejecutarlos. Por nuestra parte le damos un consejo: es siempre conveniente que así ocurra ya que de esta forma permitimos a las **«palabras»** siguientes, operar con los valores que se colocaron en la pila antes de ejecutarse la anterior secuencia de instrucciones.

El límite superior del stack siempre contiene el último número introducido y si metemos en él un nuevo valor, el anterior queda debajo de modo que el elemento que hemos almacenado más recientemente es el que ocupará la parte más alta de la pila, de ahí que se le llame **«top of stack»** (superior de la pila).

Quizá nos llame la atención, por lo que hemos visto hasta ahora, que el Forth sólo sea capaz de sumar dos números y no tres, cuatro o todos los que queramos. Pero es lógico, la «palabra» Forth «+» opera siempre solamente con dos números (al fin y al cabo es lo que nosotros hacemos con cada una de sus ci-

fras) y a continuación devuelve al stack un único resultado. Y esta forma de trabajo no sólo es válida para la suma sino que podemos entenderlo al resto de las operaciones aritméticas empleadas en Forth, tales como multiplicaciones y divisiones.

Pero este modo de desarrollar el lenguaje no nos impide utilizar expresiones más complejas. Solamente será necesario poner un poco más cuidado en lo que estamos haciendo a la hora de decidir cómo organizarlas según esta nueva notación.

Evaluación de expresiones en Forth

Pongamos un ejemplo. Si queremos evaluar en Forth la expresión:

$$15 + 2 \times 9$$

debemos comenzar multiplicando 2 por 9 (recuerde la prioridad de operaciones) para encontrar un resultado intermedio que sumado luego a 15, nos dé el valor final de la expresión.

Otra forma de calcularlo sería sumando primero 15 y 2 y después multiplicar el resultado por 9. Observe que la solución obtenida es completamente diferente de la anterior. ¿Cuál es la buena?

Recuerde que en Basic existe un orden de ejecución en las operaciones matemáticas y que el producto es más prioritario que la suma. Por tanto, el Basic y la mayoría de los lenguajes utilizarán el primer método de evaluación.

Sigámonos también en Forth. Para multiplicar 2 y 9 tendremos que teclear:

$$2\ 9\ *$$

y nos dejará el resultado (18) en el elemento superior del stack. Podemos comprobarlo visualizándolo en la pantalla usando la palabra «.» pero tenga en cuenta que, si así lo hacemos desaparecerá de la pila. De manera que como vamos a necesitar este valor en la siguiente parte del cálculo es mejor que lo dejemos donde estaba.

Después necesitamos sumar 15 al elemento superior de la pila. Pues teclee:

$$15\ +$$

y colocará el resultado en el nuevo «TOS» (top of stack) pudiéndolo imprimir a continuación. La primitiva expresión Basic se ha convertido en:

$$2\ 9\ *15\ +.$$

que es el comando Forth que nos va a sacar en la pantalla el resultado correcto de la evaluación: 33.

Sobre los cálculos matemáticos queremos puntualizar algo. La mayor parte de las versiones del Forth tiene unos operadores aritméticos que utilizan solamente números enteros y además dan resultados que son también enteros.

Escriba ahora:

$$9\ 4/.$$

que tiene toda la pinta de ser el equivalente en Forth a:

PRINT 9/4

y la respuesta obtenida es 2 en lugar de 2.25 que sería el valor exacto. De esta forma confirmamos que el número que nos devuelve la operación es también entero. Así que de momento queda bastante claro que no podemos usar números con punto decimal en esta versión del lenguaje.

Pero también está restringido el rango de los números. Intente teclear:

$$1000\ 200\ *.$$

Parece que la respuesta que aparece en la pantalla no está muy de acuerdo con lo que esperábamos. La aritmética en Forth está pensada para ser utilizada en micros, del mismo modo que el lenguaje. Por tanto, trabajará con números que puedan ser representados con 16 bits (o dígitos binarios) o lo que es lo mismo, con valores comprendidos dentro del rango de -32768 a 32767. De ahí el mensaje, aunque sea ficticio, de «**stack lleno**»: la pila no está llena, lo que se produce es un «**overflow**» o desbordamiento de la capacidad numérica de su micro.

Esta notación aritmética no es tan compleja como parece. El mejor camino para utilizarla con soltura es intentar emplearla muchas veces para hacer en Forth todas las operaciones que se nos ocurran. Después de realizar unas cuantas prácticas verá que su uso es casi tan sencillo como la empleada en la aritmética habitual con una ventaja: es mucho más potente.

La potencia del Forth está en crear nuevas órdenes en Forth

Sin embargo, donde radica el poder real del lenguaje Forth es en el hecho de poder definir nuevas «palabras» que se añaden inmediatamente a su vocabulario básico así como definir con otro nombre las ya existentes.

La «**palabra**» Forth es algo equivalente a las subrutinas, o partes de un programa Basic a las que se salta tras un GOSUB y siempre terminan con RETURN.

De un modo sencillo, podemos decir que una «**palabra**» es una serie de instrucciones, y sus correspondientes parámetros, que están agrupadas bajo un nombre que incorporamos, tras definirla, al vocabulario Forth.

Para ejecutar esta secuencia basta con invocarla, o llamarla, por su nombre y el programa salta a la serie de instrucciones, así bautizadas, y las ejecuta.

Supongamos que preferimos utilizar palabras castellanas como operadores aritméticos en lugar de los signos matemáticos y también

emplear ESCRIBIR en lugar del punto Forth. Todo lo que hay que hacer es definir las nuevas palabras.

Para ello escribimos:

```
: SUMAR+;
: RESTAR-;
: MULTIPLICAR*;
```

para las próximas palabras aritméticas, y

```
: ESCRIBIR.;
```

para conseguir que nos aparezcan los resultados en la pantalla.

Con estas definiciones, la expresión aritmética que anteriormente analizamos podría transformarse en:

2 9 MULTIPLICAR 15 SUMAR ESCRIBIR

obteniendo el mismo resultado en ambos casos.

La primitiva expresión sigue siendo válida aunque actualmente hayamos definido otras, con nombre diferente, pero que realizan las mismas operaciones.

Todas las nuevas palabras Forth que que-

Los programas siguen el método de diseño «Top-down»

Así es exactamente como se genera un programa en Forth. Dividimos un problema general en grupos de acciones más particulares. Estos últimos ya se pueden definir utilizando las palabras standard Forth.

Y es en este punto donde comenzamos para luego ir definiendo otras cada vez más complejas. Al final llegaremos a un programa completo compuesto por una sola palabra que, al teclearla, hace que se ejecuten todas las asociadas.

Si seguimos usando el ejemplo anterior, podemos definir la siguiente palabra:

```
: CALCULO 2 9 MULTIPLICAR 15 SUMAR  
  ESCRIBIR;
```

y ahora, con sólo teclear:

```
CALCULO
```

se ejecutarán todas las palabras agrupadas bajo este nombre, así como también las standard asociadas a ellas.

Por supuesto que los programas Forth no consisten únicamente en la evaluación de expresiones aritméticas —como en los casos anteriores— sino que también tienen a su disposición otras palabras que nos permitirán realizar cosas bastante más interesantes y atractivas.

Todas estas palabras básicas, incluidas en cada una de las versiones Forth que existen, es lo que se conoce como «**vocabulario**» del lenguaje, y en él están recogidas todas las que nos van a permitir utilizar variables, implementar bucles, explorar el teclado y un montón de cosas que son necesarias para que sus programas sean más complejos y le muestren la auténtica potencia del lenguaje.

Algunas órdenes del intérprete

Si quiere, puede hacer que le aparezca en la pantalla todo el vocabulario básico completo empleando el comando:

```
*VLIST
```

Y no sólo eso. También podemos visualizar todas las palabras que hayamos definido nosotros mismo. Si no lo ha hecho, le sugerimos que teclee las nuevas palabras de creación propia, para que por lo menos exista alguna.

A continuación teclee:

```
*LIST
```

y podrá conocer todas las que tiene a su disposición.

Pero empleando este comando puede también ver la serie de instrucciones que forman parte de cualquiera de ellas. Bastará con escribirlo seguido del nombre que queremos conocer y su listado estará «**servido**».

¿Qué ocurre si por equivocación definimos dos veces una misma palabra? Sencillamente esta versión del Forth no acepta la segunda y nos da un mensaje de error:

Palabra ya definida

En otras, sí lo admite sin problemas. Guarda la segunda palabra además de la anterior. Pero cuando invocamos el nombre con el que está definida, el Forth accede a la más reciente de todas las que hayamos hecho sin tener en cuenta ninguna de las más antiguas.

Con este intérprete no se nos dará nunca este caso, ya que, como hemos dicho, no admitirá dos definiciones bajo el mismo nombre.

Entonces si queremos corregir o modificar una de las ya existentes el camino más cómodo a seguir sería listar la palabra para tener constancia de ella en la pantalla (no es necesario aunque sí muy conveniente). Después borrar del diccionario la que queramos cambiar mediante el comando:

```
*FORGET palabra
```

que le dirá al Forth que elimine de su diccionario la definición que hayamos hecho de «palabra».

A continuación creamos una nueva, auxiliándonos del listado existente en la pantalla con el cursor de copia, incluyendo las modificaciones que queramos hacer.

Sin embargo, para utilizar *FORGET debemos pensar muy bien lo que estamos haciendo ya que no sólo nos elimina la palabra que le hayamos dicho, sino que también borra todas las que estén definidas con posterioridad al nombre que sigue a *FORGET. Así que, ¡jojo con lo que borramos!

Hasta este momento debe tener por lo menos una pequeña idea de qué es lo que hay que hacer para trabajar con el Forth, pero seguro que todavía no tiene una noción clara de cómo podemos hacer un programa: no conocemos lo que hace cada una de las instrucciones del vocabulario standard del lenguaje.

Podemos dividir las en grupos. El primero es el compuesto por todas las instrucciones encargadas de mantener en orden el stack cuando éste contiene parámetros. Son las instrucciones de tratamiento o manejo de la pila.

Instrucciones de manejo de la pila

Suponga que tenemos dos valores colocados en el stack y queremos hacer una división entre ellos. Por ejemplo:

```
6 2/
```

nos calcularía el cociente entre 6 y 2 y nos lo sacaría en la pantalla.

Pero quizás el resultado que a nosotros nos interesaba es el inverso, o sea, el obtenido al

ramos crear se definen de la misma forma. Comenzamos poniendo dos puntos seguidos por el nombre que queremos darle, a continuación ponemos la serie de instrucciones que queremos agrupar y cerramos la definición con un punto y coma.

Los dos puntos indican al Forth que vamos a definir una nueva palabra y deben ir seguidos forzosamente por el nombre que queremos darle.

Después viene la secuencia de parámetros y palabras Forth ya definidas que se ejecutarán al llamar a la nueva. No es necesario que sean palabras standard del lenguaje —tales como los signos + y —del ejemplo anterior— sino que podemos usar también cualquier palabra de las que nosotros ya hemos creado.

Finalmente, la definición se cierra con un punto y coma. De esta manera podemos conseguir construir un lenguaje completo y compacto, en el que se emplean palabras standard del lenguaje para crear otras nuevas, y después utilizar las nuevas para otras sucesivas definiciones. El lenguaje crece de esta manera indefinidamente.

dividir 2 entre 6. La forma inmediata de hacerlo sería escribir:

```
2 6/.
```

No siempre es posible hacer esto, ya que en la mayoría de las ocasiones no conocemos los valores contenidos en el stack. Ahora bien, el Forth dispone de un operador que intercambia los dos valores superiores de la pila. En nuestro caso ya podríamos realizar el inverso de la operación anterior mediante:

```
6 2 SWAP/.
```

Para comprobar que con esta instrucción existe intercambio sin variar para nada el resto de los elementos del stack, teclee:

```
1 2 3 SWAP...
```

y aparecerán en la pantalla estos números pero con el 2 y el 3 cambiados de orden, o sea:

```
2 3 1
```

Pero no es este el único operador que maneja los elementos de la pila. ROT hace que giren los tres superiores. Coloca el tercero en la posición más alta y desplaza a los otros dos un lugar más abajo. Como siempre, la manera más efectiva de ver cómo funciona es un ejemplo práctico, así que escriba:

```
1 2 3 ROT...
```

El orden como quedan colocados es el que aparece en la pantalla y vemos que el 1 es el que está colocado encima de los otros dos. ¿Comprendido?

Otro operador utilizado en Forth es DUP. Su misión es simplemente duplicar la última entrada que haya existido en el stack. Si quiere calcular el cuadrado de un número sería suficiente con hacer:

```
12 DUP *.
```

para que nos aparezca 144 (el cuadrado de 12) en la pantalla.

Si en lugar de esto lo que necesitamos es duplicar el elemento que está inmediatamente debajo del superior, el Forth posee un operador que realiza este trabajo: OVER. Compruébelo tecleando:

```
1 2 OVER...
```

que copiaría el 1 encima de 2 (elemento superior en ese preciso instante) e imprimiría después los tres elementos existentes sacándolos de la pila y dejándola vacía.

El último de los operadores de manejo de stack es DROP, que saca del mismo la última entrada de datos. Ejemplo al canto:

```
1 2 DROP...
```

Y éstos son los más importantes. Su conocimiento y manejo pueden resultarnos un poco liosos al principio pero poco a poco nos iremos habituando a ello y podremos «sabarrear» sus ventajas.

Las estructuras del control del lenguaje son muy potentes

Pero el Forth no sólo dispone de operadores aritméticos o de manejo del stack, sino que además posee unas estructuras de control clásicas en otros lenguajes que le dan una gran potencia y versatilidad. Veámoslas.

La sentencia que nos permitirá alterar el orden de ejecución de comandos dependiendo de que se cumpla o no una determinada condición es IF... THEN... ELSE. Su forma general es:

```
< condición > IF < acción-1 >  
ELSE < acción-2 >  
THEN < continuar >
```

Su modo de funcionamiento es como sigue. La evaluación de la condición deja un «falso» o «verdadero» en el stack. Este resultado se analiza y si es verdadero se ejecuta la acción-1, si no, realiza la acción-2 y después continua el programa en el punto que sigue a THEN.

El Programa I sería un ejemplo de utilización de esta estructura. Para utilizarlo bastaría con teclear la nota seguida de la palabra «NOTA» una vez que haya sido definida para que se nos informe de si hay aprobado o no.

Programa I

```
:NOTA 5 < IF. "SUSPENSO" ELSE. "PROBADO"  
THEN CR. "CORECTO ?";
```

El Forth dispone también de una serie de estructuras de bucle que hace que se repitan una serie de instrucciones durante un determinado número de veces. Es el conocido FOR... NEXT del lenguaje Basic.

```
< valor final >  
< valor inicial > DO < acción > LOOP
```

Con él estamos ejecutando acción desde que el «índice» es igual al valor inicial hasta que alcanza el valor final.

La palabra clave LOOP hace que volvamos a DO mientras no se haya alcanzado el valor final. Le sugerimos que intente hacer variaciones sobre este pequeño ejemplo, Programa II, para clarificar sus ideas.

Programa II

```
:BUCLE1 12 0 DO. «ESTO ES UN BUCLE  
QUE SE REPITE 12 VECES» CR LOOP;
```

Podemos acceder al índice del bucle mediante la palabra Forth «I» que obtiene el valor

en curso de la variable índice del bucle DO y le sitúa en la parte superior del stack. Si teclea el Programa III lo verá en funcionamiento.

Programa III

```
:BUCLE 13 1 DO I. SPACE. «ESTO ES UN  
BUCLE QUE SE REPITE 12 VECES» CR LOOP;
```

Si deseamos que el índice no se incremente de 1 en 1 sino con un valor diferente, se puede utilizar la estructura:

```
DO... número + LOOP
```

poniendo en «número» el incremento deseado, tanto positivo como negativo. Pero ésta es una facilidad que no contempla nuestro intérprete Forth. ¡Es una pena!

Existen también otra clase de bucles que van a repetir una serie de comandos dependiendo de que se cumpla o no una condición. En Forth este tipo de bucle puede ser de dos formas.

He aquí la primera de ellas:

El bucle WHILE nos va a permitir repetir una serie de acciones mientras se esté cumpliendo una determinada condición. Su forma general es:

```
BEGIN < condición > WHILE < acción >  
REPEAT
```

La palabra BEGIN indica simplemente el punto donde comienza el bucle. Después la condición deja un valor en el stack dependiendo de que se cumpla o no.

WHILE analiza dicho valor y si no es cero (falso) ejecuta la «acción». REPEAT nos devuelve otra vez a BEGIN.

Si la condición no se cumple y, por tanto, el valor que coloca en el stack es cero, entonces no se ejecuta «acción» y se continúa por los siguientes a la palabra REPEAT.

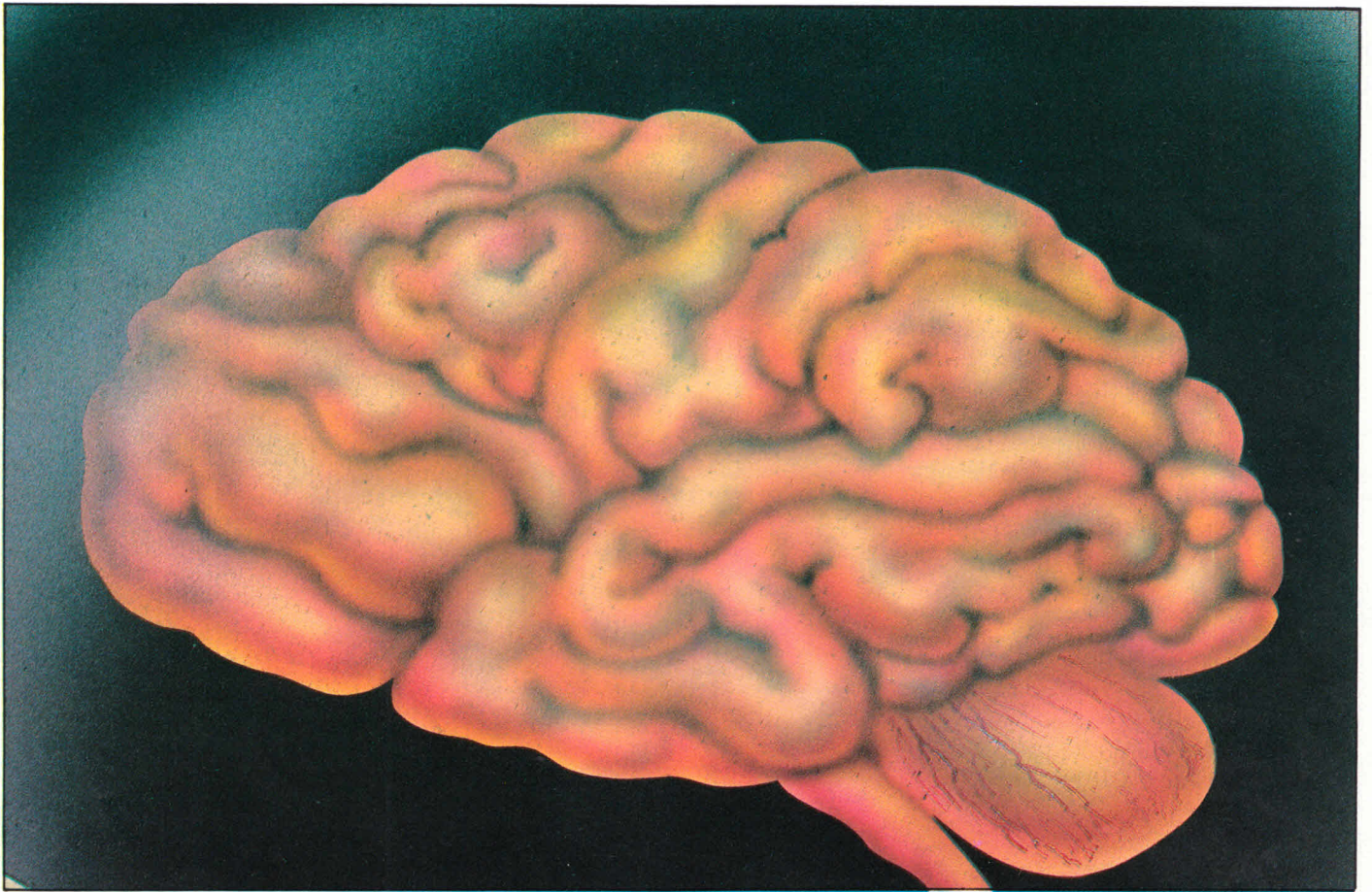
El Programa IV es un ejemplo de la utilización de este tipo de estructuras.

Programa IV

```
:BUCLAS 1 DUP BEGIN 13 < WHILE. «ESTO  
ES UN BUCLE QUE SE REPITE 12 VECES»  
CR 1 + DUP REPEAT;
```

La segunda forma de bucle indefinido es la que hace que se esté ejecutando un proceso hasta que se cumpla una condición. Se trata de la instrucción BEGIN... UNTIL.

El formato más general de esta sentencia es: BEGIN < acción > < condición > UNTIL y su forma de trabajar es la siguiente: BEGIN marca, como en el caso anterior, el comienzo del bucle. El cuerpo del mismo es



«acción» y será lo que se repita en el caso de no cumplirse la condición.

Después se evalúa la condición y deja en el stack un valor acorde con el resultado de la misma. UNTIL es la palabra clave que se encarga de encaminarnos a un sitio o a otro diferente. Cuando se encuentre con que el resultado lógico de la condición es falso (*cero*), vuelve a BEGIN y repite el proceso.

Sin embargo, si es verdadero (*distinto de cero*) el programa continúa por lo que hay detrás de UNTIL.

Programa V

```
: BUCLE4 0 BEGIN. «ESTO ES UN BUCLE
QUE SE REPITE 12 VECES» CR 1+DUP
12=UNTIL. «FIN»;
```

Estos dos últimos bucles funcionan de una manera muy parecida: se evalúa una condición y dependiendo del resultado se continúa fuera del mismo o se repite la ejecución de un determinado proceso.

Pero existe una gran diferencia entre ambos: el momento donde se comprueba si se cumple la condición.

En el primer caso realizamos el análisis antes de ejecutar el cuerpo del bucle, por tanto,

puede darse el caso que la condición sea falsa y no ejecutemos acción alguna.

Sin embargo, en el segundo siempre se va a ejecutar el proceso que hayamos definido entre BEGIN y UNTIL (al menos una vez). Se debe esto a que ahora la evaluación de la condición se hace al final del mismo. Así que, ¡cuidado con este matiz!

Parece que se nos ha olvidado algo al hablar de este lenguaje: las variables. ¿Es que no se emplean?

La verdad es que sí, existen y se usan. Pero una de las cosas que da potencia y velocidad al Forth es precisamente que utiliza las variables solamente cuando es imprescindible hacerlo.

Para pasar parámetros a rutinas o a operadores empleamos el stack tal y como lo hemos venido haciendo hasta ahora ya que se trata de valores temporales y no tiene mucho interés almacenarlos continuamente.

Las variables en Forth

Pero quizá necesitemos alguna vez guardar un determinado dato permanentemente. Y para esto están las variables en éste y en la mayoría de los lenguajes. Pero en este caso estarán definidas a nivel global, es decir, que pueden ser utilizadas por cualquiera de las palabras que estén en el vocabulario.

La forma de declararlas es:

```
< valor inicial > VARIABLE < nombre >
```

y el nombre que le hayamos dado se incorpora directamente al vocabulario de variables. Con:

```
0 VARIABLE PUNTOS
```

estamos creando una variable que se llama PUNTOS, a la que damos un valor inicial cero.

Podemos dar, o asignar, un valor distinto del inicial a una variable, ya definida, a lo largo de un programa. La forma de hacerlo en Basic sería:

```
PUNTOS=100
```

pero en Forth no se hace así. Es un poco diferente, compárelo usted mismo:

```
100 PUNTOS!
```

Y también podemos hacer el proceso inverso, es decir, recuperar en el stack el valor que previamente habíamos asignado a una variable.

Para colocar el valor de PUNTOS en el elemento superior del stack bastaría con hacer:

```
PUNTOS
```

y después imprimirlo con la palabra «.».

Pero estas dos últimas instrucciones se podrían agrupar en una sola. Si escribimos:

```
PUNTOS ?
```

obtendríamos el mismo resultado.

EUROPA

El software educativo, por desgracia, no tiene un papel protagonista dentro de la inmensa cantidad de programas que rodean a los ordenadores Amstrad.

Tal vez porque sea muy difícil de hacer, tal vez porque los que comercializan programas no están muy seguros del mercado potencial en este campo, o quizás porque en nuestro país la enseñanza asistida por ordenador está en «mantillas».

El caso es que siempre resulta agradable ver que un lector se preocupa por explorar un terreno considerablemente virgen, contando con que los medios de que dispone son inferiores, con mucho, a los de cualquier casa de soft.

El programa «Europa» es una buena idea para aprender geografía en varios sentidos a la vez. Su presentación, a mi juicio, es impecable, y se nota que el programa está hecho con amor, cuidando al máximo el detalle.

Alguien podría objetar que «Europa» es excesivamente largo y tedioso de teclear, y que no hubiera estado mal emplear un algoritmo en lugar de dibujar el mapa de Europa punto a punto.

Bien, no hubiera estado mal, pero hay programas escritos con la cabeza y programas escritos con el corazón; yo creo que «Europa» es de los segundos, y apuesto por él.



El juego consta de cinco niveles:

- Nivel 1: Naciones.
- Nivel 2: Capitales.
- Nivel 3: Naciones-capitales.
- Nivel 4: Organización militar.
- Nivel 5: Organización económica.

En el primer nivel aparecerá el mapa de Europa y un cuadrado rojo en una nación europea.

El ordenador te preguntará qué nación es donde se encuentra el cuadrado rojo. Si adivinas la nación te da 20 puntos y te pregunta otra nación.

Cuanto tengas 40 puntos, pulsando «L» cambias de nivel.

En el segundo nivel, lo que hay que poner es la capital de la nación. Cuando tengas 80 puntos, pasas a otro nivel pulsando «L».

... y así sucesivamente con «Naciones-capitales», «Organizaciones militares» y «Organizaciones económicas».

Los cambios de nivel se producen a:

- los 40 pts;
- los 80 pts;
- los 120 pts;
- los 160 pts;
- los 200 pts.

A los 200 puntos vuelves a comenzar.

Variables

- E Valor «x» del cuadrado en pantalla de gráficos.
- F Valor «y» del cuadrado en pantalla de gráficos.
- H Valor del incremento.
- I Color del cuadrado (Siempre rojo 11)
- Q\$ Nombre de la nación, capital, etc.
- BC\$, BD\$, BE\$, BF\$, BG\$, BH\$, BI\$, BJ\$, BK\$, BL\$, BLL\$, BM\$, BN\$, BO\$, BP\$, BQ\$, BR\$, BS\$, BT\$, BU\$, BW\$, BX\$, BY\$, Datos.
- Ca Nivel.
- K\$ Nombre de la nación, capital, etc. que introduces.
- LL Puntuación.
- AZ Número de países ya apelados.

Instrucciones

- Simplemente hacer lo que el ordenador diga, tiene las instrucciones incorporadas.
- Si cuando pones el nombre de la nación, capital, etc. escribes «L» y tu puntuación es 40, 80, 120, 160 ó 200, cambias de nivel.
- Si escribes «F» finaliza el juego.
- Si escribes «R» te da la respuesta el ordenador.


```

6230 REM
6240 REM ***** DATOS *****
**
6250 REM
6260 BC$="ESPAÑA":BD$="FRANCIA":BE$="IRLANDA":BF$="INGLATERRA":BG$="ITALIA":BH$="SUIZA":BI$="BELGICA":BJ$="PORTUGAL":BK$="HOLANDA":BL$="ALEMANIA OCCIDENTAL":BLL$="ALEMANIA ORIENTAL":BM$="AUSTRIA":BN$="YUGOSLAVIA":BO$="CHECOSLOVAQUIA":BP$="POLONIA"
6270 BQ$="RUMANIA":BR$="HUNGRIA":BS$="ALBANIA":BT$="GRECIA":BU$="BULGARIA":BV$="NORUEGA":BW$="SUECIA":BX$="FINLANDIA":BY$="RUSIA"
6280 GOTO 6460
6290 BC$="MADRID":BD$="PARIS":BE$="DUBLIN":BF$="LONDRES":BG$="ROMA":BH$="BERNA":BI$="BRUSELAS":BJ$="LISBOA":BK$="AMSTERDAM":BL$="BONN":BLL$="BERLIN":BM$="VIENA"
6300 BN$="BELGRADO":BO$="PRAGA":BP$="VARSOVIA":BQ$="BUCAREST":BR$="BUDAPEST":BS$="TIRANA"
6310 BT$="ATENAS":BU$="SOFIA":BV$="OSLO":BW$="ESTOCOLMO":BX$="HELSINKI":BY$="MOSCU":GOTO 6460
6320 BC$="ESPAÑA-MADRID":BD$="FRANCIA-PARIS":BE$="IRLANDA-DUBLIN":BF$="INGLATERRA-LONDRES":BG$="ITALIA-ROMA":BH$="SUIZA-BERNA":BI$="BELGICA-BRUSELAS":BJ$="PORTUGAL-LISBOA":BK$="HOLANDA-AMSTERDAM":BL$="ALEMANIA OCCIDENTAL-BONN"
6330 BLL$="ALEMANIA ORIENTAL-BERLIN":BM$="AUSTRIA-VIENA"
6340 BP$="POLONIA-VARSOVIA":BQ$="RUMANIA-BUCAREST"
6350 BR$="HUNGRIA-BUDAPEST":BS$="ALBANIA-TIRANA"
6360 BN$="YUGOSLAVIA-BELGRADO":BO$="

```

```

"CHECOSLOVAQUIA-PRAGA"
6370 BT$="GRECIA-ATENAS":BU$="BULGARIA-SOFIA":BV$="NORUEGA-OSLO":BW$="SUECIA-ESTOCOLMO":BX$="FINLANDIA-HELSINKI":BY$="RUSIA-MOSCU"
6380 GOTO 6460
6390 BC$="OTAN":BD$="OTAN":BE$="NULO":BF$="OTAN":BG$="OTAN":BH$="NULO":BI$="OTAN":BJ$="OTAN":BK$="OTAN":BL$="NULO":BLL$="PACTO DE VARSOVIA":BM$="NULO":BN$="NULO"
6400 BO$="PACTO DE VARSOVIA":BP$="PACTO DE VARSOVIA":BQ$="PACTO DE VARSOVIA":BR$="PACTO DE VARSOVIA":BS$="PACTO DE VARSOVIA":BT$="NULO":BU$="PACTO DE VARSOVIA":BV$="OTAN":BW$="NULO":BX$="NULO":BY$="PACTO DE VARSOVIA"
6410 GOTO 6460
6420 BC$="CEE":BD$="CEE":BE$="CEE":BF$="CEE":BG$="CEE":BH$="NULO":BI$="CEE":BJ$="CEE":BK$="CEE":BL$="CEE":BM$="NULO":BN$="NULO":BO$="COMECON":BP$="COMECON":BQ$="COMECON":BR$="COMECON":BS$="NULO":BT$="CEE":BU$="COMECON"
6430 BV$="NULO":BW$="NULO":BX$="NULO":BY$="COMECON"
6440 GOTO 6460
6450 REM
6460 A=INT(RND(1)*8)
6470 IF A=1 THEN GOTO 4080
6480 IF A=2 THEN GOTO 4140
6490 IF A=3 THEN GOTO 4200
6500 IF A=4 THEN GOTO 4260
6510 IF A=5 THEN GOTO 4320
6520 IF A=6 THEN GOTO 4380
6530 IF A=7 THEN GOTO 4440
6540 IF A=8 THEN GOTO 4490
6550 IF A=9 THEN GOTO 4550
6560 IF A=10 THEN GOTO 4610

```

```

6570 IF A=11 THEN GOTO 4670
6580 IF A=12 THEN GOTO 4730
6590 IF A=13 THEN GOTO 4790
6600 IF A=14 THEN GOTO 4850
6610 IF A=15 THEN GOTO 4910
6620 IF A=16 THEN GOTO 4970
6630 IF A=17 THEN GOTO 5030
6640 IF A=18 THEN GOTO 5090
6650 IF A=19 THEN GOTO 5150
6660 IF A=20 THEN GOTO 5210
6670 IF A=21 THEN GOTO 5270
6680 IF A=22 THEN GOTO 5330
6690 IF A=23 THEN GOTO 5390
6700 IF A=24 THEN GOTO 5450
6710 REM
6720 REM ***** PND *****
**
6730 REM
6740 FOR DE=1 TO 6
6750 PLOT E,F-5,0:DRAW E+H,F-5
6760 F=F+1
6770 NEXT
6780 LOCATE 1,19:PRINT"
6790 GOTO 4040
6800 GOTO 6460
6810 CLS:END
6820 CLS:RUN

```



Correo..., más rápido...



Con el fin de acelerar lo más posible el **correo**, y poder resolver o contestar a todas las dudas y sugerencias que llegan a nuestra redacción, a partir de esta semana os rogamos, en beneficio de todos, consignar en el sobre, en lugar bien visible, una de las denominaciones siguientes:

- **Suscripciones AMSTRAD.** Para todos aquellos casos relacionados con petición de cintas, números atrasados, formalización de suscripciones, devoluciones etc...
- **Mercado Común AMSTRAD.** Compras, ventas, intercambios, clubs...
- **Sin duda alguna AMSTRAD.** Para que nos enviéis todas vuestras dudas.
- **Serie Oro AMSTRAD.** Para los programas que nos enviéis para su publicación.
- **Sugerencias AMSTRAD.** Para vuestras críticas, sugerencias o cualquier opinión que queráis vertir sobre la revista.

ORDEMANIA SOFT

DE VENTAS
ESTADÍSTICA

por fin..

CONTABILIDAD

No. ASIENTO.: 1

FECHA.: 09/06/86

MANUAL

GESPACK

Paquete integrado de gestión que le permite a Vd., de forma fácil y optimizando el tiempo, la gestión global de su empresa.

Este paquete incorpora los programas de CONTABILIDAD, FACTURACION y CONTROL DE STOCKS además de un programa de CONTROL DE PEDIDOS y todo de forma interactiva. Controle su empresa por sólo

29.900 pts.

Contabilidad

Contabilidad de fácil manejo y de gran potencia que permite trabajar con cuentas de hasta cuatro niveles, con capacidad según diskette de 500/1.000 cuentas y de 2.000/10.000 asientos.

Permite modificar o dar de baja apuntes ya integrados en el Mayor, programaciones de cierres, ficheros de Contabilidad y Cuenta de Explotación, ejecución de balances comparativos, reenlazándolos por meses, clave acceso restringido, etc...

19.900 ptas.

Facturación

Programa de gran agilidad y rapidez que incorpora el Control de Clientes, con gran capacidad de datos, Artículos, Albaranes, Facturas y recibos.

Generación automática y manual de documentos, valoración, a voluntad de los albaranes, todo tipo de listado, incluido el del IVA de las facturas emitidas para la declaración de Hacienda, etc...

15.500 ptas.

Control de stocks

Gran capacidad de datos, le permitirán a Vd. llevar con claridad y sencillez el control de su stock.

El programa le permite llevar un libro de entradas/salidas, reorganizarlo, hacer listados de stocks..., le avisará de los límites de stocks, mínimo y máximo por artículo, etc...

Todo para la llevanza de su almacén.

14.900 ptas.

disponibles para: PCW 8256
PCW 8512

Asimismo, Contabilidad disponible para CPC 664/6128 **9.900 ptas.**

OFITES INFORMATICA
DISTRIBUIDOR EXCLUSIVO DE ORDEMANIA SOFT

● IVA NO INCLUIDO

Condiciones especiales para distribuidores

Si tiene alguna dificultad en obtenerlos dirijase a

Ofites
Informatica
Avda. Isabel II, 16 - 8º
Tels. 455344 - 455333
Telex 366908
20011 SAN SEBASTIAN

CONTROL DEL CRT

Pero, ¿qué es el CRT? Pues, sencillamente, las iniciales de «Cathode Ray Tube», que traducido a nuestro idioma quiere decir, literalmente, «Tubo de Rayos Catódicos», o en términos menos técnicos, la pantalla «física» del monitor de su Amstrad, que también es un tubo de rayos catódicos como el de un televisor cualquiera.



Como lo anterior se deduce que el artículo que presentamos está dedicado a cómo controlar la pantalla. Y efectivamente, así es, pero desde un punto de vista distinto al habitual del BASIC y, sin embargo, sin utilizar código máquina ni PEEK's ni POKE's, a menos que, por razones de velocidad de ejecución, se decida emplear otros medios más rápidos que sólo son posibles desde código máquina, y para los que la base teórica que expondremos seguirá siendo igualmente válida.

Con las órdenes existentes en BASIC podemos hacer un montón de cosas «**EN la pantalla**», pero también hay un comando que nos permitirá hacer otras tantas «**CON la pantalla**». Y dicho comando no es, ni más ni menos, que el comando «**OUT**».

EN LA PANTALLA Y CON LA PANTALLA

El comando OUT generalmente pasa desapercibido para el programador normal, sobre todo porque en los manuales no se explica cuál puede ser su utilidad práctica, limitándose a dar en dos líneas una escueta descripción de la acción que realiza y nada más. Transcribimos para aquéllos que no tengan el manual a mano:

«OUT < número de portal > , < expresión entera >

Envía el valor del parámetro entero al portal

de SALIDA correspondiente a la dirección mencionada.»

Aun cuando tan sucinta definición es totalmente exacta, lo menos que puede ocurrir es que nos deje absolutamente indiferentes si no se complementa con un ejemplo o con una explicación de lo que es un portal de salida.

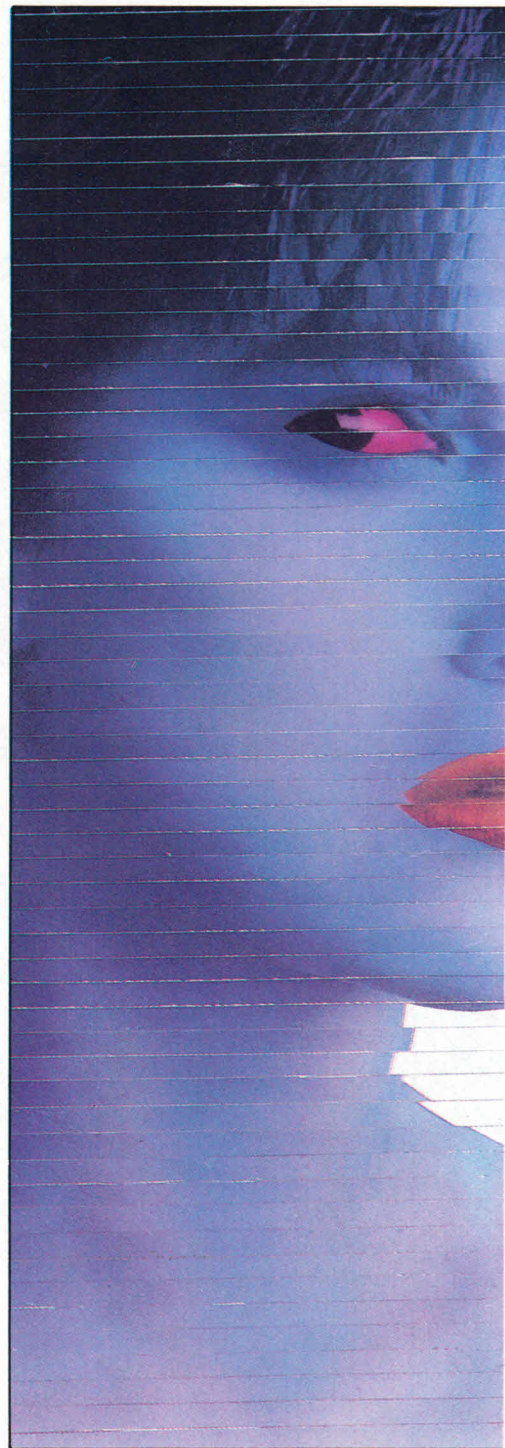
Intentaremos dar un poco de luz sobre el tema a lo largo del artículo, pero antes, y para desperezar los dedos, teclearemos en modo directo las siguientes instrucciones, operando en **MODE 1** y procurando no equivocarse:

```
OUT &BC00, 1 [ENTER] OUT &BD00, 20 [ENTER]
```

¿Qué es lo que ha ocurrido? Aparentemente, se han insertado líneas en blanco entre las que acaba de escribir, pero intente escribir algo, de más de 20 caracteres, o moverse con el cursor más allá de la columna 20 y verá cómo el cursor salta a la primera columna de la línea siguiente. ¡EUREKA! Pensará que hemos descubierto el comando WINDOW, sólo que más complicado. PUES NO. Puede comprobarlo tecleando MODE 1, con lo que volveríamos a la pantalla normal, pero que en esta ocasión no sirve de nada. Para ver lo que ha sucedido teclee:

```
BORDER 15 [ENTER]
```

Esta orden pone el borde la pantalla de color naranja y, en estas circunstancias, además, la mitad derecha de la pantalla también aparecerá de este color, lo que quiere decir que la pantalla útil se ha reducido a la mitad. Para ser más exactos, los 40 caracteres de una línea normal ahora se distribuyen en dos líneas de 20 caracteres, es decir, la pantalla de $25 \times 40 = 1.000$ caracteres ahora se ha transformado en 50 líneas de 20 caracteres. Imagínese la pantalla como una masa de letras que comprimimos horizontalmente: las letras al no poder salirse de la pantalla, se desplazarán hacia abajo, reestructurando la visualización, pero sin perderse ni una, solamente que no las vemos. Compruébelo desplazando el cursor por debajo de la última línea visible, pulse ESCAPE y siga bajando el cursor hasta que empiece el scroll hacia arriba, con



lo que aparecerá el «*Break*» que no veíamos antes.

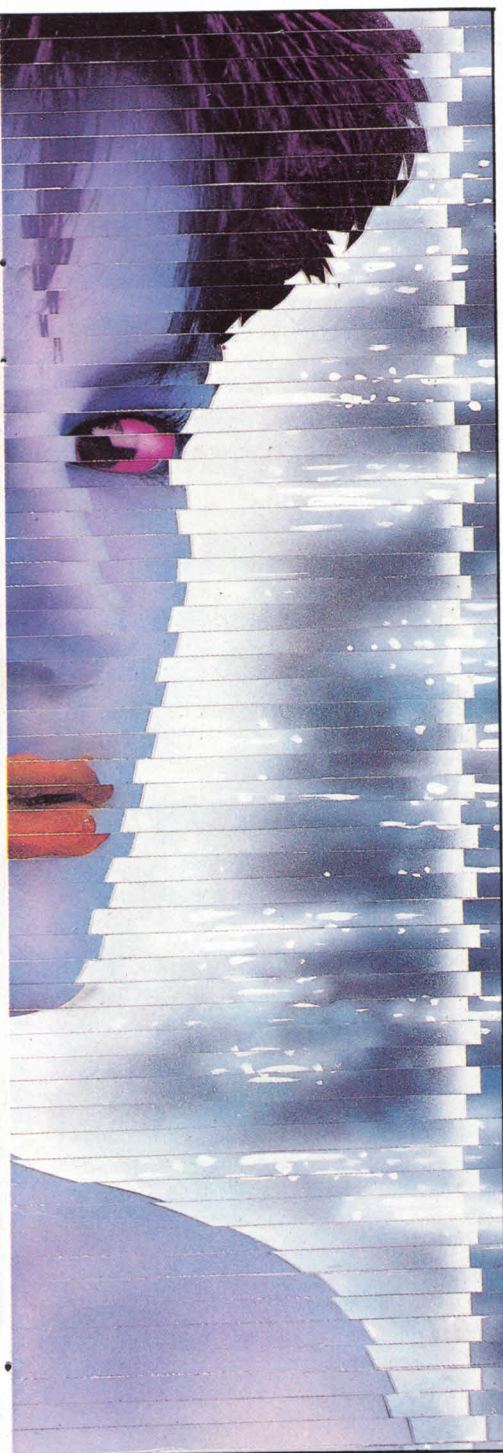
Para volver a la pantalla normal escriba:

```
OUT &BC00,1: OUT &BD00,40 [ENTER]
```

o resetee el ordenador.

Ahora se comprenderá la diferencia entre hacer cosas «**EN la pantalla**» y «**CON la pantalla**», puesto que la orden OUT afectará a toda la pantalla y no a zonas de la misma.

Antes de que se le ocurra empezar a experimentar por su cuenta haciendo OUT's a diestro y siniestro, estamos en la obligación moral de advertir que un OUT de cualquier valor en cualquier portal de salida, sin un ligero



conocimiento de lo que se hace, puede producir un «hermoso» bloqueo del sistema, siendo necesario desconectar el ordenador, y si no pruebe con:

```
OUT &BC00,0: OUT &BD00,1
```

UN POCO DE TEORIA

Todos los ordenadores están compuestos físicamente por una serie de elementos que nos permiten trabajar con él. El elemento más importante es el microprocesador o Unidad de

Control de Proceso (CPU), que en los **Amstrads** es el renombrado Z80A, y que no es más que una *pastillita* de silicio de unos cuantos milímetros cuadrados conteniendo cerca de 8.000 transistores, interconectados de tal manera que posibilitan hacer cosas tales como jugar a marcianitos o llevar la contabilidad de una empresa.

Pero decimos que «posibilitan» porque, a pesar de la potencia del microprocesador, éste por sí solo no es incapaz de hacer absolutamente nada si no se encuentra conectado con otros dispositivos como un teclado, una pantalla o la memoria; mediante los cuales los podemos dar órdenes o recibir los resultados de una serie de operaciones o almacenar datos, respectivamente.

El Z80A es el cerebro que controla, directa o indirectamente, todas las acciones que realiza el ordenador, y entre ellas se encuentran: leer el teclado, manejar la pantalla, generar sonidos, controlar el cassette, enviar datos a la impresora, leer y escribir en la memoria, etc.

Nos centraremos en el área del control de la pantalla. El Z80 maneja la pantalla indirectamente a través de un chip o circuito integrado llamado CRTC, cuyas siglas corresponden a «**Cathode Ray Tube Controller**» (Controlador del Tubo de Rayos Catódicos).

El CRTC realiza las funciones de mantenimiento de la visualización en pantalla, controlando las dimensiones y la posición de la zona de exposición de información en la superficie del Tubo, y generando las direcciones de memoria donde están los datos de pantalla, entre otras cosas.

Todo esto se consigue utilizando el contenido de los 18 registros de que dispone el CRTC. Podemos entender un registro como un dispositivo cuya misión es retener una información para posteriormente ser tratada por el CRTC. Cada uno de estos registros puede contener un número de un byte de longitud, esto es, con un valor entre cero y 255.

Modificando el contenido de los registros del CRTC modificaremos las características de la pantalla, según hemos podido comprobar. ¿Y cómo modificar un registro? Nada más sencillo, con la orden **OUT**.

Como ya hemos dicho, el CRTC es un circuito conectado al microprocesador, y éste se comunica con aquél a través de unos canales que se encuentran en los llamados **PORTALES** de SALIDA. Un PORTAL no es más que una dirección de memoria donde se depositan datos para ser enviados al dispositivo correspondiente a ese número de portal. Explicándolo de otra forma, hacer un OUT de un Número a un Portal es como enviar una carta con un mensaje a una persona. El OUT es la acción de enviar la carta; el Número sería el mensaje, y el Portal sería la dirección de esa persona.

Necesitamos, pues, conocer la dirección del portal de salida correspondiente al Controlador del Tubo de Rayos Catódicos para poder enviarle los datos que queramos.

PROGRAMACCION

El CRTC tiene dos portales o direcciones, cada uno de los cuales tiene una misión específica. El primer portal se utiliza para indicarle al CRTC el número del registro que queremos seleccionar. Y el segundo portal es por donde le enviaremos el valor al registro previamente seleccionado. Las direcciones son:

&BC00: para seleccionar número de registro.

&BD00: para modificar contenido del registro.

Aun cuando hemos dicho que el CRTC tiene 18 registros, el sistema sólo permite acceder a los primeros 16, por lo que el número que tendremos que enviar al primer portal del CRTC, el &BC00, será un número entero entre cero y 15, inclusive, por aquello de que los ordenadores empiezan a contar desde cero. Lo haremos del siguiente modo:

```
OUT &BC00, < num. reg > siendo num. reg > un valor entre 0 y 15.
```

para modificar el contenido del registro seleccionado:

```
OUT &BD00, < dato > siendo < dato > un valor entre 0 y 255.
```

En la tabla 1 damos los nombres de cada uno de los registros, así como los valores estándar y los valores mínimos y máximos en cuya gama no se bloquee el ordenador. Insistimos en que cualquier valor fuera de esta gama nos obligará a tener que desconectar nuestro **Amstrad**.

También existen portales de ENTRADA al microprocesador, donde recoger datos mediante el orden del BASIC IN, pero no los trataremos por salirse del tema que abordamos. Únicamente decir al respecto que un uso inapropiado de las entradas puede ocasionar **daños físicos** a los circuitos del **ordenador**.

Comentar todos y cada uno de los efectos que se producen al modificar los registros del CRTC con los distintos valores posibles es prácticamente imposible. Por otra parte, enviando OUT's en modo «directo» y de forma desordenada nos encontraremos con que es una forma un tanto pesada de investigar, y otras veces habremos cambiado tantos registros que ya no sabremos cómo volver a la pantalla normal.

Por todo ello, creemos que la mejor solución está en facilitar las cosas con un programa que nos permita manejar los registros a nuestro antojo, pudiendo recuperar la pantalla normal cuando queramos y ver qué registros hemos modificado hasta ese momento, para, de esta manera, sacar nuestras propias conclusiones de cómo inciden los diferentes parámetros enviados al CRTC, sobre todo, pensando en el dicho de que una imagen vale más que mil palabras. Todo esto nos lo permitirá el programa «**CONTROL DEL CRT**» que luego comentaremos.

TABLA 1. REGISTROS DEL CRTC

N. Reg	Estand.	Min.	Max.	Nombre del Registro
0	63	46	100	Total
1	40	0	64	Horizontal Visualizado
2	46	0	63	Posición de sincronismo horizontal
3	142	(*)	(*)	Ancho del sincronismo
4	38	0	255	Total Vertical
5	0	0	255	Ajuste del Total Vertical
6	25	0	50	Vertical Visualizado
7	30	0	255	Posición de sincronismo vertical
8	30	0	255	Modo solapado y sesgado
9	7	0	255	Máximo número de líneas de barrido
10	0	0	255	Comienzo del cursor sintetizado
11	0	0	255	Fin del cursor sintetizado
12	48	0	255	Dirección comienzo (byte alto) rastreo memoria de pantalla
13	0	0	255	Idem (byte bajo)
14	192	0	255	Registro del cursor (byte alto)
15	0	0	255	Idem (byte bajo)

(*) = Dependiente del contexto de los demás registros.

Volviendo sobre el famoso dicho, pruebe a teclear el siguiente programa que nos muestra una forma más vistosa de presentar las pantallas en nuestros programas:

```
10 BORDER 15: MODE 0
20 FOR a=40 TO 0 STEP-1: GOSUB 60:
NEXT: '— Recoge Pantalla
30 GOSUB 70: '—Pintar Pantalla
40 FOR a=0 TO 40: GOSUB 60: NEXT: '—
—Extiende Pantalla
50 CALL &BB06: GOTO 20: '—Espera pulsación y vuelve a empezar
60 CALL &BD19: OUT &BC00,1: OUT &BD00,a: RETURN: 'Ejecuta OUT's
70 t=TIME
80 CLG RND*13
90 WHILE TIME < t+300
100 DRAW 640*RND, 400*RND, 13*RND:
TAG: PRINT CHR$( RND*26+64);
110 WEND
120 RETURN
```

A los que posean el juego **BEACH HEAD** les resultará familiar esta forma de recoger la pantalla. Está basada en variar el contenido del registro 1 del CRTC entre los valores 40 y 0, mediante un bucle FOR... NEXT STEP -1, para recoger la pantalla, y al revés, entre 0 y 40, para extenderla de nuevo.

Antes de cambiar el registro con los OUT's correspondientes, hacemos una llamada al FIRMWARE, en &BD19, para sincronizar el cambio que se va a realizar con el haz de barrido de pantalla, consiguiéndose una mayor suavidad en el movimiento. Es lo mismo que hacer un FRAME en el CPC 6128. Para ver la necesidad del CALL &BD19, pruebe a quitarlo.

Otro de los ejemplos que se me ocurren como consecuencia de indagar con los registros del CRTC utilizando el programa «CONTROL CRT» es el siguiente:

```
10 FOR d=1 TO 3: READ men$
20 men$=SPACE$( (40-LEN (men$))/2)+
+men$: '—Centra mensaje
30 MODE 1
40 LOCATE 1,20
50 FOR a=1 TO 40
60 letra$=MID$( men$,a,1)
70 GOSUB 140: '—Ejecuta OUT's
80 PRINT letra$;
90 NEXT
100 FOR r=0 TO 1500: NEXT: '—Retardo
110 LOCATE 1,25: PRINT STRING$( 20,10):
'—Sube mensaje
120 NEXT
130 RUN
140 CALL &BD19: OUT &BC00,13: OUT &BD00,a: RETURN
150 DATA MICROHOBBY AMSTRAD ESPECIAL
160 DATA Presenta:
170 DATA *CONTROL DEL CRT*
```

Esta vez se modifica el registro 13, que contiene el byte bajo de la dirección de comienzo de rastreo de la memoria de pantalla, por lo que, al ir desplazando dicha dirección con el comando OUT, se desplaza a su vez la pantalla, produciendo un suave **SCROLL** horizontal de la pantalla.

Esta última técnica, en combinación con unos cuantos conocimientos de código máquina para obtener la rapidez necesaria, nos permitiría construir un programa que utilizara la pantalla de su **Amstrad** como un tablón de anuncios electrónico a todo color, como ya he podido ver en algunos sitios.

Antes de hacerlo correr es conveniente salvarlo en cinta/disco por si hubiera algún OUT perdido que nos lo echara todo a perder.

Primero tenemos una pantalla de presentación en la que se explica la función y el manejo del programa, para, después de ver una pequeña demostración del uso del CRTC, aparecer la pantalla de trabajo con un informe del estado actual de los registros, el valor estándar entre corchetes y el nombre de los mismos.

Abajo, a la derecha, y recuadrado, tenemos los dos Modos de funcionamiento, apareciendo en video inverso el modo activo.

En modo «Ejecutar» podemos modificar los registros sin que se observe su efecto, hasta pulsar las teclas CTRL+E.

En modo «Inmediato», nada más modificar un registro, se ejecuta la acción inmediatamente.

Se puede cambiar de modo en cualquier momento pulsando las teclas SHIFT.

En cuanto seleccionamos un registro, su nombre aparece en video inverso y el programa espera la pulsación de una de las teclas posibles que se visualizan en la última línea y que son:

«V»: Visualiza el rango permitido y espera la entrada del dato a enviar al registro.

«Cursor Arriba/Cursor Abajo»: Incrementa/Decrementa el contenido del registro en una unidad.

Cualquier otra tecla, así como un valor fuera de rango, anula la acción, produciendo una señal acústica de error.

En la pantalla se informa con un asterisco del último registro modificado, y, en video inverso aparecerán los valores que no sean estándar, al objeto de una más fácil localización.

Si se llega a perder el control de la pantalla, pulsando la tecla «N» volvemos a la pantalla normal, apareciendo el mensaje: PULSE UNA TECLA PARA VOLVER AL ESTADO ANTERIOR.

Si en ese momento estamos en modo Inmediato y pulsamos cualquier tecla, excepto SHIFT, volveremos al caso anterior; pero si pulsamos SHIFT, cambiamos al modo Ejecutar, por lo que, al pulsar otra tecla, se mantiene la pantalla normal, pudiendo restaurar los registros al valor que queramos hasta que pulsamos las teclas CTRL+E para observar el efecto que se produce.

Con la tecla «X» veremos cuatro de las infinitas posibilidades que nos presenta el manejo del CRTC.

Por fin, pulsando «T» termina el programa restaurando los valores normales del CRTC.

Finalizar aclarando que el registro **r3**, Ancho del Sincronismo, es muy inestable, por lo que el programa no permite acceder al mismo. No obstante, el que quiera aventurarse sólo tiene que modificar la línea 710 quitando la condición «OR tec = 4». ¡Ah!, y que alterar los registros que se refieren al Cursor no produce efecto alguno.

PROGRAMACION

```

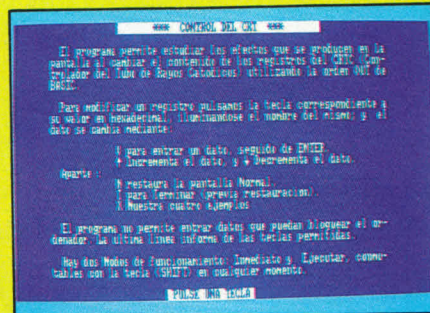
10 '*-----*
20 '* CONTROL CRT *
30 '* por J.F.Bayo *
40 '* Mayo 1986 *
50 '*-----*
60 :
70 ' ::: PRESENTACION :::
80 :
90 MODE 2:WINDOW 8,80,1,25
100 i$=CHR$(24)
110 PRINT ,i$ SPC(7) "*** CONTROL
DEL CRT ***" SPC(7)i$:PRINT
120 PRINT" El programa permite est
udiar los efectos que se producen e
n la"
130 PRINT"pantalla al cambiar el co
ntenido de los registros del CRTC (
Con-"
140 PRINT"trolador del Tubo de Rayo
s Catodicos) utilizando la orden OU
T de"
150 PRINT"BASIC.":PRINT
160 PRINT" Para modificar un regis
tro pulsamos la tecla correspondien
te a"
170 PRINT"su valor en hexadecimal,
iluminandose el nombre del mismo; y
el"
180 PRINT"dato se cambia mediante:"
:PRINT
190 PRINT,"V para entrar un dato, s
eguido de ENTER."
200 PRINT, CHR$(240)+ " Incrementa e
l dato, y "+CHR$(241)+ " Decrementa
el dato."
210 PRINT" Aparte : "
220 PRINT,"N restaura la pantalla N
ormal."
230 PRINT,"T para Terminar (previa
restauracion)."
240 PRINT,"X Muestra cuatro ejemplo
s":PRINT
250 PRINT" El programa no permite
entrar datos que puedan bloquear el
or-"
260 PRINT"denador. La ultima linea
informa de las teclas permitidas.":
PRINT
270 PRINT" Hay dos Modos de funcio
namiento: Inmediato y Ejecutar, co
nmu-"
280 PRINT"tables con la tecla <SHIF
T> en cualquier momento.":PRINT
290 PRINT TAB(23) i$+" PULSE UNA TE
CLA "+i$
300 :
310 ' ::: INICIALIZACION :::

```

```

320 :
330 DEFINT a-u
340 DIM stat(3,15),nom$(15)
350 FOR a=0 TO 15
360 READ stat(0,a),stat(1,a),stat(2
,a),nom$(a)
370 stat(3,a)=stat(2,a)
380 f$=MID$(STR$(stat(3,a)),2)
390 f$=SPACE$(3-LEN(f$))+f$
400 nom$(a)="["+f$+" ] "+nom$(a)
410 NEXT
420 ENT -1,1,10,1:ENT -2,1,-10,1
430 ENV 1,7,1,1,7,-1,1:ENV 2,1,10,1
,10,1,1,15,5,1,15,-1,15
440 masc1=i$+"&"&i$:masc2=i$+" ##
# "+i$

```



```

450 m$(0)="Ejecutar " :m$(1)="Inmedi
ato"
460 CALL &BB06:BORDER 12
470 REM MIN,MAX,STANDAR,NOMBRE DEL
REGISTRO
480 DATA 46,100,63,Total horizontal
490 DATA 0,64,40,Horizontal Visuali
zado
500 DATA 0,63,46,Posicion sincronis
mo horizontal
510 DATA 0,255,142,Ancho del sincro
nismo <No Accesible>
520 DATA 0,255,38,Total Vertical
530 DATA 0,255,0,Ajuste total Verti
cal
540 DATA 0,50,25,Vertical visualiza
do
550 DATA 0,255,30,Posicion sincroni
smo vertical
560 DATA 0,255,0,Modo solapado o se
sgado
570 DATA 0,255,7,Numero lineas de b
arrido
580 DATA 0,255,0,Comienzo del curso
r
590 DATA 0,255,0,Fin del cursor
600 DATA 0,255,48,Direccion comienz

```

```

o rastreo (byte alto)
610 DATA 0,255,0,Idem (byte bajo)
620 DATA 0,255,192,Registro del cur
sor (byte alto)
630 DATA 0,255,0,Idem (byte bajo)
640 :
650 GOSUB 1400:'*** DEMOSTRACION **
*
660 :
670 ' ::: CONTROL ENTRADAS :::
680 :
690 GOSUB 840:'inkey$
700 tec=INSTR("0123456789ABCDEFNTX"
+CHR$(5),q$)
710 IF tec=0 OR tec=4 THEN SOUND 7,
600,-2,8,1,2:GOTO 690
720 IF tec=17 THEN GOSUB 1200:GOTO
690
730 IF tec=18 THEN 1200
740 IF tec=19 THEN GOSUB 1340:GOSUB
1300:GOTO 690
750 IF tec=20 THEN GOSUB 1270:GOTO
690
760 reg=tec-1
770 LOCATE 20,reg+3:PRINT USING mas
c1$;nom$(reg)
780 LOCATE 20,23:PRINT <V> <Flechas
cursor> Cualquier otra anula la ac
cion"
790 GOSUB 840
800 IF q$="v" OR q$="V" THEN LOCATE
1,23:PRINT CHR$(10)+CHR$(11):PRINT
"ENTRAR DATO (<stat(0,reg)>..."stat
(1,reg)> <ENTER para volver>";:LI
NE INPUT "",v$:v=VAL(v$):IF STR$(v)
=" "+v$ THEN IF v<=stat(1,reg) AND
v>=stat(0,reg) THEN stat(3,reg)=v:G
OTO 890
810 IF q$=CHR$(240) AND stat(3,reg)
<stat(1,reg) THEN stat(3,reg)=stat(
3,reg)+1:GOTO 890
820 IF q$=CHR$(241) AND stat(3,reg)
>stat(0,reg) THEN stat(3,reg)=stat(
3,reg)-1:GOTO 890
830 SOUND 7,600,-2,8,1,1:LOCATE 20,
reg+3:PRINT nom$(reg):GOSUB 1300:G
OTO 690
840 IF INKEY(21)<>-1 THEN m=m XOR 1
:LOCATE 48,20+m:PRINT USING masc1$;

```

```

m$(m):LOCATE 48,21-m:PRINT m$(1-m):
GOSUB 1830
850 q$=UPPER$(INKEY$):IF q$="" THEN
840 ELSE RETURN
860 :
870 ':::: OUT :::::
880 :
890 IF m=0 THEN 910
900 OUT &BC00,reg:OUT &BD00,stat(3,
reg)
910 IF TEST(140,84)<>1 THEN GOSUB 1
130
920 LOCATE 20,reg+3:PRINT nom$(reg)
930 LOCATE 8,reg+3:PRINT "*"
940 LOCATE 15,reg+3:IF stat(3,reg)=
stat(2,reg) THEN PRINT USING " ###
";stat(3,reg) ELSE PRINT USING masc
2$;stat(3,reg)
950 IF reg<>ureg THEN LOCATE 8,ureg
+3:PRINT " "
960 ureg=reg
970 GOSUB 1300:GOTO 490
980 :
990 ':::: ESTADO DE LOS REGISTROS ::
:
1000 :
1010 ORIGIN 0,0:DRAW 0,399:DRAW 639
,399:DRAW 639,0:DRAW 0,0:MOVE 1,0:D
RAWR 0,399:MOVE 638,0:DRAWR 0,399
1020 FOR a=0 TO 20:INK 0,12:INK 0,6
:INK 0,24:NEXT:INK 0,1
1030 PRINT TAB(6) i$+" ESTADO ACTUA
L, ESTANDAR Y NOMBRE DE LOS REGISTR
OS DEL CRTC "+i$
1040 PRINT
1050 FOR a=0 TO 15
1060 PRINT TAB(10)"r "+CHR$(a+40-(a
>9)*7)+" =";:PRINT USING " ### ";st
at(3,a):LOCATE 20,a+3:PRINT nom$(a)
1070 BORDER 26-a:INK 1,9+a:NEXT
1080 LOCATE 48,20:PRINT USING masc1
$;m$(0)
1090 LOCATE 48,21:PRINT m$(1)
1100 PLOT 404,84:DRAWR 144,0:DRAWR
0,-40:DRAWR -144,0:DRAWR 0,40
1110 PLOT 482,70:TAG:PRINT"<CTRL+E>
";:TAGOFF
1120 GOTO 1160
1130 LOCATE 5,20:PRINT "* indica el
ultimo registro modificado"
1140 PRINT TAB(5)"En video inverso
los datos no Estandar"
1150 PLOT 60,84:DRAWR 312,0:DRAWR 0
,-40:DRAWR -312,0:DRAWR 0,40
1160 GOSUB 1300:RETURN
1170 :
1180 ':::: PANTALLA NORMAL :::

```

```

1190 :
1200 FOR a=0 TO 15:OUT &BC00,a:OUT
&BD00,stat(2,a):NEXT
1210 IF tec=19 THEN RETURN
1220 IF tec=18 THEN LOCATE 1,1:END
1230 LOCATE 1,23:PRINT CHR$(18)
1240 LOCATE 10,23:PRINT i$+" PULSE
UNA TECLA PARA VOLVER AL ESTADO AN
TERIOR "+i$
1250 GOSUB 840
1260 GOSUB 1300:IF m=0 THEN RETURN

```



```

1270 FOR a=0 TO 15:OUT &BC00,a:OUT
&BD00,stat(3,a):NEXT
1280 RETURN
1290 :
1300 LOCATE 1,23:PRINT i$+" Teclas
posibles: "+i$+" 0,1,2,4,5,6,7,8,9,
A,B,C,D,E,F (N) (T) (X)=Ejemplos."+
CHR$(18):RETURN
1310 :
1320 ':::: EJEMPLOS :::::
1330 :
1340 GOSUB 1200
1350 LOCATE 1,23:PRINT"ELEGIR EJEMP
LO (1,2,3,4) (ENTER para volver)+C
HR$(18)
1360 GOSUB 840
1370 ej=INSTR("1234",q$):IF ej=0 TH
EN RETURN
1380 ON ej GOSUB 1540,1620,1700,179
0:RETURN
1390 :
1400 '==== DEMOSTRACION ====
1410 :
1420 FOR a=25 TO 0 STEP-1:GOSUB 147
0:NEXT
1430 MODE 2:WINDOW 5,79,2,24
1440 FOR a=0 TO 25:GOSUB 1470:NEXT
1450 SOUND 7,0,0,0,2,0,31:OUT &BC00
,8:OUT &BD00,1
1460 GOTO 1480
1470 CALL &BD19:OUT &BC00,1:OUT &BD
00,INT(1.6*a):OUT &BC00,6:OUT &BD00
,a:SOUND 2^(a MOD 3),0,7,13,0,0,a+5

```

```

:RETURN
1480 INK 1,8:GOSUB 1010:'ESTADO DE
LOS REGISTROS
1490 OUT &BC00,8:OUT &BD00,0
1500 RETURN
1510 :
1520 '==== EJEMPLO 1 ====
1530 :
1540 FOR a=40 TO 0 STEP -1:GOSUB 15
80:NEXT
1550 GOSUB 1830
1560 FOR a=0 TO 40:GOSUB 1580:NEXT
1570 RETURN
1580 CALL &BD19:OUT &BC00,1:OUT &BD
00,a:RETURN
1590 :
1600 '==== EJEMPLO 2 ====
1610 :
1620 FOR a=25 TO 0 STEP -1:GOSUB 16
60:NEXT
1630 OUT &BC00,1:OUT &BD00,0:GOSUB
1830:OUT &BC00,1:OUT &BD00,40
1640 FOR a=0 TO 25:GOSUB 1660:NEXT
1650 RETURN
1660 CALL &BD19:OUT &BC00,6:OUT &BD
00,a:RETURN
1670 :
1680 '==== EJEMPLO 3 ====
1690 :
1700 FOR a=40 TO 0 STEP -1:GOSUB 17
40:NEXT
1710 GOSUB 1830
1720 FOR a=0 TO 40:GOSUB 1740:NEXT
1730 RETURN
1740 CALL &BD19:OUT &BC00,1:OUT &BD
00,a:OUT &BC00,2:OUT &BD00,a+6:RETU
RN
1750 RETURN
1760 :
1770 '==== EJEMPLO 4 ====
1780 :
1790 FOR b=0 TO 30:CALL &BD19:FOR a
=0 TO 5:GOSUB 1810:NEXT:FOR a=4 TO
0 STEP -1:GOSUB 1810:NEXT:NEXT
1800 RETURN
1810 OUT &BC00,5:OUT &BD00,a:RETURN
1820 :
1830 FOR t=0 TO 500:NEXT:RETURN

```



Ofites Informática

Presenta: la tableta gráfica

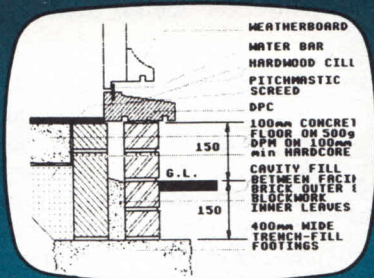
GRAFPAD II-

LO ULTIMO EN DISPOSITIVOS DE ENTRADA DE GRAFICOS PARA AMSTRAD, COMMODORE Y BBC

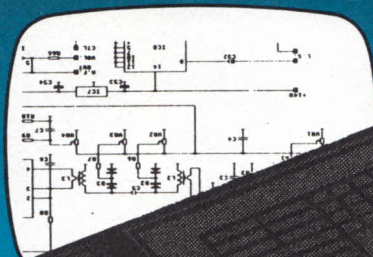
La primera tableta gráfica, de bajo costo, en ofrecer la duración y prestaciones requeridas por las aplicaciones de negocios, industria, hogar y educación. Es pequeña, exacta y segura. No necesita ajustes ni mantenimiento preventivo. GRAFPAD II es un producto único que pone la potencia de la tecnología moderna bajo el control del usuario.



DIBUJO A MANO ALZADA
SOFTWARE DE ICONOS



DISEÑO DE ARQUITECTURA
CON SOFTWARE DDX



COMBINA EN UN UNICO DISPOSITIVO TODAS LAS PRESTACIONES DE LOS INTENTOS PREVIOS DE MECANISMOS DE ENTRADA DE GRAFICOS. LAS APLICACIONES SON MAS NUMEROSAS QUE EN LOS DEMAS DISPOSITIVOS COMUNES E INCLUYEN:

- selección de opciones
- entrada de modelos
- recogida de datos
- diseño lógico
- diseño de circuitos
- creación de imágenes
- almacenamiento de imágenes
- recuperación de imágenes
- diseño para construcción
- C.A.D. (diseño asistido por ordenador)
- ilustración de textos
- juegos
- diseño de muestras
- educación
- diseño PCB.

ESPECIFICACIONES

RESOLUCION:

1.280 x 1.024 pixels.

PRECISION:

1 pixel.

TASA DE SALIDA:

2.000 pares de coordenadas por segundo.

INTERFACE:

paralelo.

ORIGEN:

borde superior izquierdo o seleccionable.

DIMENSIONES:

350 x 260 x 12 mm.

DISPONIBLE AMSTRAD:

CASSETTE 23.900 ptas.

DISCO 25.900 ptas.

(IVA NO INCLUIDO)

- FACIL DE USAR.
- TRÁZADO PCB.
- C.A.D.
- AREA DE DISEÑO DIN A4.
- COLOR EN ALTA RESOLUCION.
- USO EN HOGAR Y NEGOCIOS.
- VARIEDAD DE PROGRAMAS DISPONIBLES.
- DIBUJO A MANO ALZADA.
- DIAGRAMAS DE CIRCUITOS.

**TRADUCIDO
AL ESPAÑOL**

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener la tableta gráfica, puede dirigirse a:



**Ofites
Informática**

Avda. Isabel II, 16 -8º

Tels. 455544 - 455533

Télex 36698

20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

MODIFICADOR DE CADENAS

Muchos de nosotros, en alguna ocasión, nos hemos visto obligados a cambiar una cadena de caracteres que se repetía en varias ocasiones en un programa Basic. Aquellos que hayan vivido esta experiencia, conocerán lo monótona y trabajosa que resulta esta tarea.



A partir de ahora, la modificación de cadenas de un programa Basic, resultará lo más sencillo del mundo si nos ayudamos del programa que aparece listado en este artículo.

Debemos advertir en primer lugar que nuestro programa trabaja únicamente con programas que estén salvados en formato fichero.

Si deseamos trabajar con un programa que no se encuentre en este formato, deberemos hacer lo siguiente:

— Cargar el programa en memoria con:
LOAD"PROGRAMA"

— Salvar el programa de la forma siguiente:

SAVE"PROGRAMA",A

de esta forma nuestra rutina ya podrá trabajar con él, puesto que ahora se encontrará salvado como fichero de caracteres o fichero ASCII.

La rutina encargada de efectuar el trabajo está ubicada a partir de la dirección hexadecimal &A000, y tiene una longitud de 1135 bytes. Por lo tanto no podremos trabajar con programas Basic que ocupen dichas direcciones.

Debemos aclarar que esto no es un gran inconveniente, debido a que apenas ningún programa escrito en Basic llega a ocupar esas direcciones, ya que generalmente no suelen ser demasiado extensos.

De lo dicho anteriormente, se deduce que el programa modificador de cadenas no podrá trabajar con programas Basic que superen los 39 K de longitud.

La solución del peor de los casos

Si en alguna ocasión necesitáramos trabajar con programas de gran extensión, que superarán los 39 K de memoria, no deberemos alarmarnos, puesto que todo tiene solución.

Dicha solución sería partir nuestro programa en dos, es decir, deberíamos cargar el programa en memoria y salvar únicamente la mitad, y a continuación volver a cargarlo en memoria y salvar la otra mitad.

Ahora estaríamos en condiciones de trabajar con cada una de las dos mitades sin problemas de memoria.

Una vez hubiésemos finalizado el trabajo cargaríamos el primer programa y a continuación haríamos un MERGE del segundo programa, con lo cual tendríamos el programa otra vez completo y modificado.

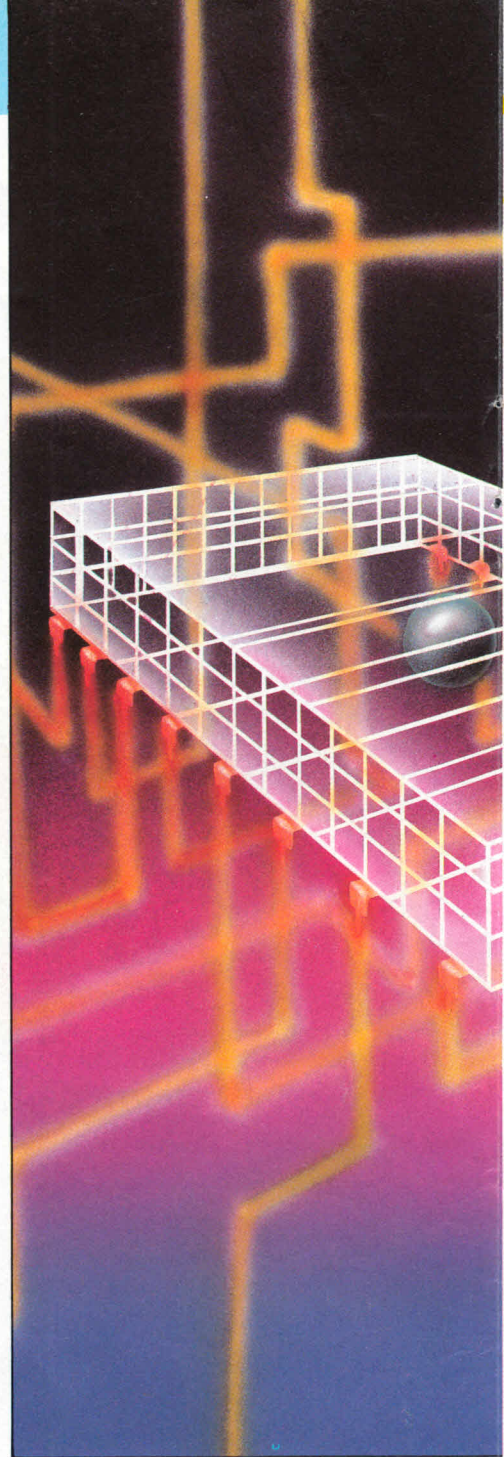
Realizadas estas advertencias y apuntadas las posibles soluciones a los problemas con los que nos podemos encontrar, pasaremos ahora a explicar el manejo del programa modificador de cadenas.

El programa

Con lo primero que nos encontramos al ejecutar el programa, es con un menú de ayuda en el que se encuentran las siguientes opciones:

ORDENES DEL MODIFICADOR

B — Basic
L — Listar
C — Cargar (LOAD)
S — Salvar (SAVE)
F — Cambiar cadenas



La primera de las opciones tiene como único objeto, permitirnos retornar al Basic.

La opción "LISTAR", nos permitirá listar el programa que se encuentre en memoria en ese momento. Si no hemos cargado ningún programa, al elegir esta opción, aparece un mensaje indicándonos que no existen ningún programa en memoria.

Cuando se elija la opción **CARGAR**, se nos pedirá el nombre del programa que deseamos cargar en memoria. Una vez escrito dicho nombre, deberemos pulsar "ENTER".

Mientras se esté ejecutando la carga del programa, aparecerán unas líneas en la pantalla, pero eso no nos debe asustar, ya que dichas rayas son debidas a que se ha elegido dicha zona de memoria como buffer de carga.

Se ha hecho de esta forma con el fin de ahorrar memoria ya que dicho buffer necesita: 2 k.

A continuación se nos preguntará cuál es la cadena que debe sustituir a la anterior.

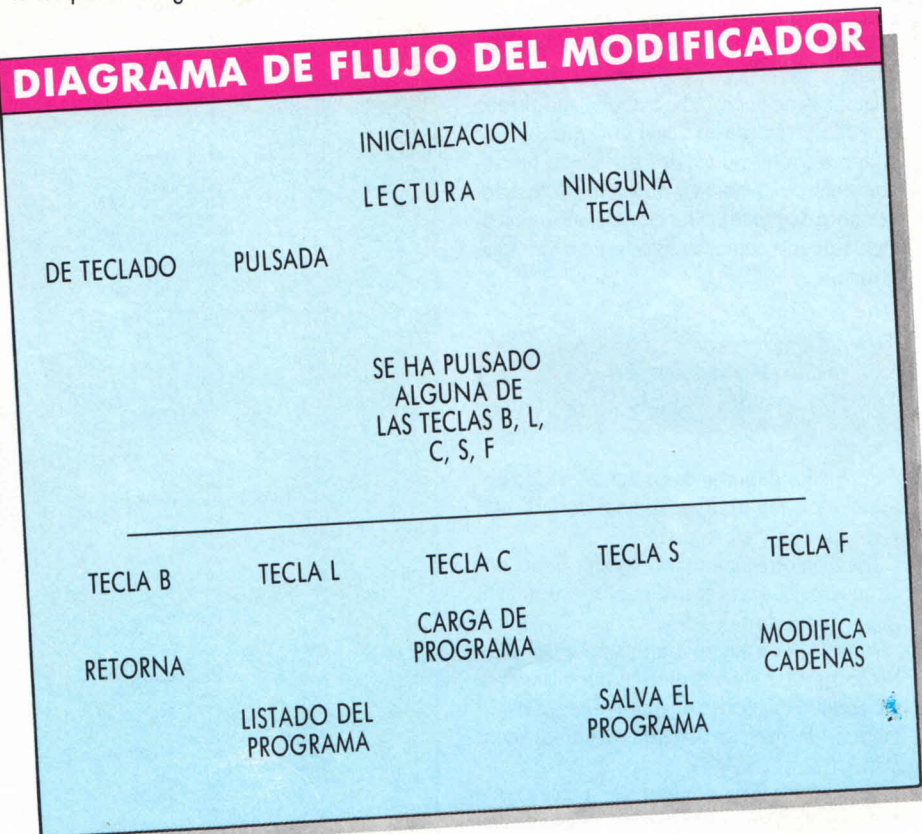
Cuando se hayan dado estos parámetros, el programa buscará una cadena idéntica a la primera que hayamos dado, y la cambiará por la última.

Cuando haya finalizado el trabajo, se imprimirá un mensaje indicando que dichas cadenas se han modificado. En el caso de que no encuentre ninguna cadena de caracteres igual a la que hemos introducido, nos lo indicará con otro mensaje.

Funcionamiento y rutinas principales

Vamos a ver ahora cuáles son las rutinas más importantes de que se compone nuestro programa.

Para hacernos una idea de conjunto del programa realizaremos una especie de diagrama de flujo con el cual podremos ver globalmente los pasos a seguir.



Otra de las opciones que posee es la que permite salvar el programa que tenemos en memoria en disco o cinta. En esta opción volverán a aparecer las rayas mencionadas anteriormente, debido a que se utiliza el mismo buffer de memoria.

Por último, se nos ofrece la posibilidad de modificación de cadenas de caracteres. Esta opción nos permitirá cambiar una cadena por otra de una longitud no superior a 20 caracteres. Si la longitud de la cadena de caracteres que se desea modificar es superior a dicha cifra, entonces se deberá efectuar el cambio en dos pasadas, es decir, deberemos modificar en primer lugar los primeros 20 caracteres y a continuación los restantes. Cuando deseemos modificar alguna cadena, lo primero que hará el programa será preguntarnos cuál es la cadena que se desea cambiar; una vez introducida, deberemos pulsar "ENTER".

lectura del teclado. La rutina permanecerá en dicho bucle hasta que no sea pulsada alguna de las teclas predeterminadas.

Cuando se detecte la pulsación de alguna de las teclas que indican al programa que se ha elegido una opción, se enviará el control a la rutina que corresponda, y una vez finalizado el trabajo se retornará al bucle principal de lectura de teclado.

La primera subrutina con la que nos encontramos en el programa, es la encargada de efectuar la carga.

Para averiguar cuál es el nombre del programa que se desea cargar, efectuamos una llamada a la rutina INPUT, la cual nos pedirá el nombre del fichero que deseamos introducir en la memoria del ordenador.

Hecho esto se retorna a la rutina LOAD para efectuar dicha carga, utilizando las rutinas del firmware capaces de leer un programa en forma de fichero ASCII.

Una vez finalizada su tarea, reinicializa la pantalla y devuelve el control del programa al bucle principal para detectar la elección de otras opciones.

Rutina de listar

Otra de las rutinas que componen el programa es la encargada de producir un listado en pantalla. Para ello se toma la dirección inicial del programa, y a partir de ahí va tomando los caracteres que encuentra en memoria y los imprime en pantalla.

Cuando se detecta que se ha llegado al final del programa, retorna al bucle principal.

A continuación nos encontramos con la rutina más importante de nuestro programa, se trata de la rutina encargada de modificar las cadenas de caracteres.

En primer lugar, se llama a la rutina INPUT, para conseguir la información de qué cadena debe modificar y por cuál ha de sustituirla. Cuando ha conseguido dichos datos, intenta detectar una cadena idéntica a la dada, sino lo consigue devuelve el control al bucle principal imprimiendo el mensaje de que no ha encontrado ninguna cadena.

En caso de que se encuentre con una cadena de caracteres idéntica a la que se ha dado, entonces se encarga de sustituirla por la nueva, reservando el espacio de memoria necesario para la nueva cadena de caracteres a introducir, y eliminando al mismo tiempo el espacio de memoria que ocupaba la antigua.

Una vez realizada la misma operación con todas las cadenas de caracteres idénticas a la dada, imprime el mensaje correspondiente y retorna al bucle de lectura de teclado.

Por último nos queda comentar la rutina de SAVE, ésta hace una llamada a INPUT para saber qué nombre debe dar al nuevo fichero, y a continuación se almacena en cinta o disco.

Después de finalizar su trabajo, inicializa la pantalla y retorna al bucle principal.

Una vez realizada la descripción del funcionamiento de la rutina y dadas las instrucciones para su correcto funcionamiento, únicamente queda copiar el listado de dicho programa.

Manipulación del listado

Para ello, deberemos copiar el listado ensamblador que aparece al final del artículo, y salvarlo en cinta o disco.

También ofrecemos un cargador Basic, para aquellos que prefieran teclearlo directamente en forma de DATAS.

Aquellos que elijan esta última opción, deberán ejecutar el programa cargador a vez teclado, y en el caso de que aparezca algún mensaje de error, se deberán revisar las líneas DATA.

Si dicho mensaje no aparece, indicará que todo ha ido bien, y por lo tanto estaremos en condiciones de salvarlo en cinta o disco.

Para ello haremos lo siguiente:

SAVE "CADENAS",B,&A000,1135

Cuando se quiera trabajar con el modificador de cadenas, deberemos escribir un programa BASIC como el que se indica a continuación:

```
10 MEMORY &9FF
20 LOAD "CADENAS"),&A000
30 CALL &A000
```

No queda nada más que decir, sólo desear que este programa os sea de gran utilidad.

```
10 ;MODIFICADOR DE CADENAS
20 ORG #A000
30 LD HL,0
40 LD (FINPRD),HL
50 CALL VENTAN
60 JP TEC
70 VENTAN: CALL A,2
80 LD A,#COE
90 LD A,1
100 CALL #BBB4
110 LD HL,0
120 LD DE,#5001
130 CALL #BB66
140 LD A,1
150 CALL #BB96
160 XOR A
170 CALL #BB90
180 CALL #BB6C
190 LD HL, TXT
200 DTRPD: LD A,(HL)
210 CP ZSS
220 JR Z, FUERA
230 CALL #BB5A
240 INC HL
250 JR DTRPD
260 FUERA: XOR A
270 CALL #BBB4
280 LD HL,#0002
290 LD DE,#5019
300 CALL #BB66
310 RET
320 ;
330 ;
340 TEC: LD A,54
350 CALL #BB1E
360 RET NZ
370 LD A,36
380 CALL #BB1E
390 JR Z, PAST1
400 CALL IMPRE
410 PAST1: LD A,60
420 CALL #BB1E
430 JR Z, PAST2
440 CALL SAVE
450 PAST2: LD A,62
460 CALL #BB1E
470 JR Z, PAST3
480 CALL LDAD
490 PAST3: LD A,53
500 CALL #BB1E
510 JR Z, TEC
520 CALL BUSCA
530 JR TEC
540 ;
550 ;
560 LOAD: CALL #BB6C
570 CALL I_LOAD
580 CALL #BB6C
590 LD A,(L_LOAD)
600 LD B,A
610 LD HL,N_LOAD
620 LD DE,#COAO
630 CALL #BC77
640 JP NC, ERROR
650 LD HL,5000
660 BULEC: CALL #BC80
670 JR NC, FINIT
680 LD (HL),A
690 INC HL
700 JR BULEC
710 FINIT: DEC HL
720 LD (FINPRD),HL
730 CALL #BC7A
740 CALL VENTAN
750 LD HL,#0B0D
760 LD DE, TXTFL
770 CALL PRINT
780 RET
790 IMPRE: CALL #BB6C
800 LD HL,(FINPRD)
810 LD A,H
820 OR L
830 JP Z, NOPRD
840 LD DE,5000
850 DTRA: PUSH DE
860 LD A,66
870 CALL #BB1E
880 CALL NZ, PAUSA
890 POP DE
900 LD A,(DE)
910 CALL #BB5A
920 INC DE
930 LD HL,(FINPRD)
940 LD A,H
950 CP D
960 JR NZ, DTRA
970 LD A,L
980 CP E
990 JR NZ, DTRA
1000 RET
1010 PAUSA: LD BC,40000
1020 PAUS: DEC BC
1030 LD A,B
1040 OR C
1050 JR NZ, PAUS
1060 CALL #BB03
1070 CALL #BB1B
1080 RET
1090 ;
1100 ;BUSCA CADENA
1110 ;
1120 BUSCA: LD HL,(FINPRD)
1130 LD A,H
1140 OR L
1150 JP Z, NOPRD
1160 XOR A
1170 LD (NUMCA),A
1180 CALL #BB6C
1190 CALL NOMCA1
1200 CALL #BB6C
1210 CALL NOMCAN
1220 CALL #BB6C
1230 LD HL,CADENV
1240 LD DE,5000
1250 LD IX,1000
1260 VULV: LD B,D
1270 BUC: LD A,(LONCV)
1280 CP B
1290 JR Z, FIN
1300 PUSH HL
1310 LD HL,(FINPRD)
1320 LD A,D
1330 CP H
1340 JR NZ, SIGU
1350 LD A,E
1360 CP L
1370 JR Z, FINRU
1380 SIGU: POP HL
1390 LD A,(DE)
1400 LD (IX+0),A
1410 CP (HL)
1420 INC DE
1430 INC IX
1440 JR Z, INCCO
1450 LD B,0
1460 LD HL,CADENV
1470 JR BUC
1480 INCCO: INC B
1490 INC HL
1500 JR BUC
1510 FIN: LD (DIREC),DE
1520 OTRDE: DEC IX
1530 DJNZ OTRDE
1540 LD A,(NUMCA)
1550 INC A
1560 LD (NUMCA),A
1570 LD A,(LONCN)
1580 LD B,A
1590 LD HL,CADENN
1600 PONBU: LD A,(HL)
1610 LD (IX+0),A
1620 INC IX
1630 INC HL
1640 DJNZ PONBU
1650 LD HL,CADENV
1660 JR VULV
1670 FINRU: POP HL
1680 PUSH IX
1690 POP HL
1700 LD DE,4000
1710 ADD HL,DE
1720 LD (FINPRD),HL
1730 LD DE,5000
1740 SCF
1750 CCF
1760 SBC HL,DE
1770 LD B,H
1780 LD C,L
1790 LD HL,1000
1800 LD DE,5000
1810 LDIR
1820 LD A,(NUMCA)
1830 CP 0
1840 JR NZ, SIGCA
1850 LD HL,#0B0D
1860 LD DE, TXTNOC
1870 CALL PRINT
1880 RET
1890 SIGCA: LD HL,#0B0D
1900 LD DE, TXTSIC
1910 CALL PRINT
1920 RET
1930 SAVE: CALL #BB6C
1940 LD HL,(FINPRD)
1950 LD A,H
1960 OR L
1970 JP Z, NOPRD
1980 CALL I_SAVE
1990 CALL #BB6C
2000 LD A,(L_SAVE)
2010 LD B,A
2020 LD HL,N_SAVE
2030 LD DE,#COAO
2040 CALL #BC8C
2050 LD DE,5000
2060 OTRCA: LD A,(DE)
2070 CALL #BC95
2080 LD HL,(FINPRD)
2090 LD A,H
2100 CP D
2110 JR NZ, PASI
2120 LD A,L
2130 CP E
2140 JR Z, FINSA
2150 PASI: INC DE
2160 JR OTRCA
2170 FINSA: CALL #BCBF
2180 CALL #BC92
2190 CALL VENTAN
2200 RET
2210 NOMCA1: LD A,20
2220 LD (MIRLON+1),A
2230 LD HL,#0B0D
2240 LD DE, TXTO
2250 CALL PRINT
2260 CALL INPUT
2270 LD A,(LONG)
2280 LD (LONCV),A
2290 LD DE,CADENV
2300 CALL TRASP
2310 RET
2320 NOMCAN: LD A,20
2330 LD (MIRLON+1),A
2340 LD HL,#0B0D
2350 LD DE, TXT1
2360 CALL PRINT
2370 CALL INPUT
2380 LD A,(LONG)
2390 LD (LONCN),A
2400 LD DE,CADENN
2410 CALL TRASP
2420 RET
2430 I_LOAD: LD A,B
2440 LD (MIRLON+1),A
2450 LD HL,#0B0D
2460 LD DE, TXT2
2470 CALL PRINT
2480 CALL INPUT
2490 LD A,(LONG)
2500 LD (L_LOAD),A
2510 LD DE,N_LOAD
2520 CALL TRASP
2530 RET
2540 I_SAVE: LD A,B
2550 LD (MIRLON+1),A
2560 LD HL,#0B0D
```


LISTADOR DE VARIABLES

Otro de los programas de utilidades que os ofrecemos en este número, es el listador de variables, utilizable a través de nuevos comandos RSX.



Mediante dichos comandos, estaremos en condiciones de listar tanto las variables numéricas como alfanuméricas que se utilicen en un programa.

La rutina en código máquina que realiza dicha función, está ubicada a partir de la dirección hexadecimal \$A000 y tiene una longitud de 585 bytes.

Debido a la zona de memoria que ocupa nuestro programa, no podremos trabajar con programas de gran longitud, ya que de lo contrario corromperíamos esas direcciones de memoria. Así pues, la longitud de nuestros programas, no debe superar 39k.

Para que dicha rutina funcione correctamente, deberemos ejecutar el siguiente programa Basic:

```
10 MEMORY &9FFF
20 LOAD"RSX
30 CALL &A000
```

Una vez hecho esto, borraremos todo lo que hay en la memoria mediante el comando:

```
NEW
```

y a continuación podremos cargar en memoria el programa con el cual vayamos a trabajar.

La línea 30 del anterior programa Basic, se utiliza para indicar al ordenador que existen nuevos comandos, ya que de lo contrario, los ignoraría.

Una vez tengamos en memoria el programa con el que deseemos trabajar, **no deberemos ejecutarlo**, ya que de lo contrario se inicializarían las variables, y la rutina sería incapaz de identificarlas.

A partir de este momento, estamos en condiciones de utilizar nuestros nuevos comandos.

Vamos a indicar a continuación cuáles son dichos comandos y de qué forma funcionan.

El primero de ellos se utiliza para listar las variables alfanuméricas, y se puede usar de dos formas distintas, según la opción deseada.

Este nuevo comando es el siguiente:

```
IVALFA
```

escrito de esta forma, dicho comando produciría un listado de todas las variables alfanuméricas, indicando asimismo la línea en la que cada una de ellas se encuentra.

Si por ejemplo existiera en memoria un programa que tuviera la variable alfanumérica A\$ en las líneas 20, 40, la variable HOLA\$ en las líneas 50 y 100 y la variable B\$ en las líneas 20, 50 y 60, tras la ejecución del anterior comando, nos aparecería el siguiente listado:

```
00020 A$ B$
00040 A$
00050 B$ HOLA$
00060 B$
00100 HOLA$
```

Otra manera de utilizarlo sería de la forma que se indica a continuación:

```
IVALFA, "HOLA"
```

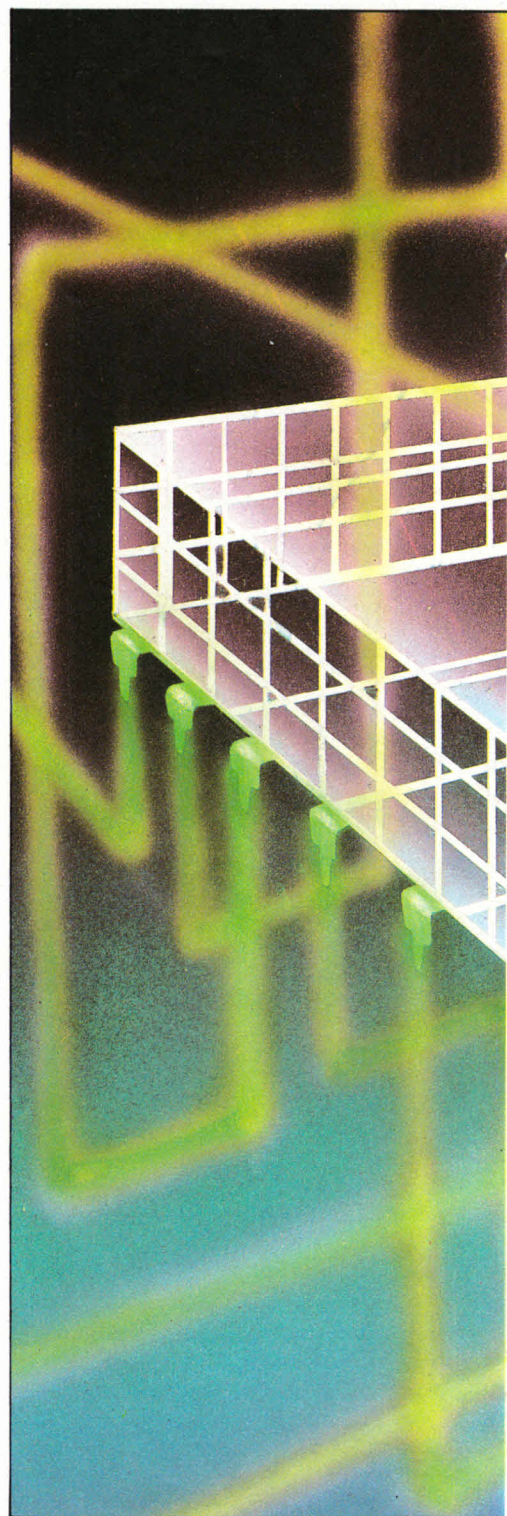
La ejecución de este comando tal como está escrito, produciría el listado de la variable alfanumérica 'HOLA\$', indicando al mismo tiempo en qué líneas de programa se encuentra.

Utilizando el ejemplo del hipotético programa citado anteriormente, este comando, nos produciría un listado en pantalla, como el siguiente:

```
00050 HOLA$
00100 HOLA$
```

Este último comando nos será de utilidad cuando se desee conocer en qué partes del programa actúa una variable alfanumérica, y en particular en qué líneas de programa se está utilizando.

Así pues, cuando se ejecute, se producirá la búsqueda de la variable introducida, y en el caso de ser encontrada se imprimirá en pantalla. En el caso de que no exista ninguna variable de este tipo, no se producirá ningún tipo de impresión en pantalla.

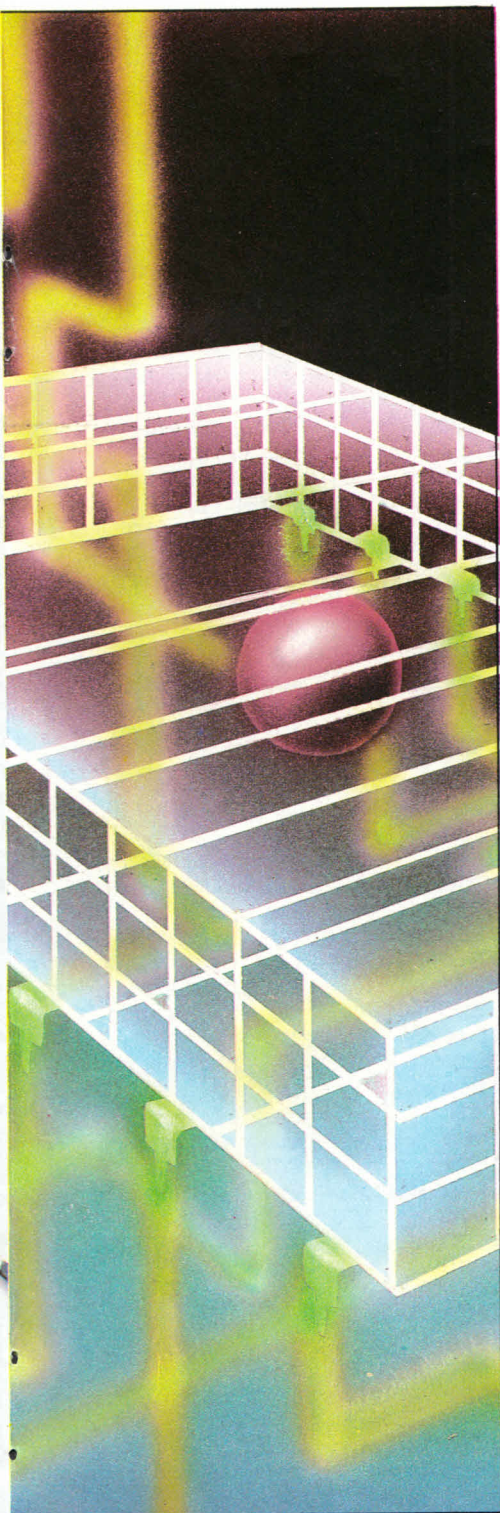


Otro de los comandos de que disponemos, es el que se indica a continuación:

```
IVNUME
```

Dicho comando provoca el listado de todas las variables numéricas que contenga nuestro programa, así como los números de línea donde se encuentran.

Como hemos hecho anteriormente, supondremos que existe en memoria un programa que contiene en las líneas 10 y 60 la variable numérica 'POSX', en las líneas 20 y 90 la variable 'YPOS', y en la línea 40 la variable 'DI-



Este comando es el siguiente:

IVNUME, "POSX"

Su ejecución provoca la búsqueda de la cadena introducida entre comillas. En el caso de tener en memoria el programa anteriormente mencionado, provocaría la impresión en pantalla del siguiente texto:

```
00010 POSX
00060 POSX
```

También por impresora

El último comando del que disponemos, y no por ello menos importante es:

IP

el cual nos ofrece la opción de imprimir en pantalla o bien en impresora.

Este comando actúa como una especie de conmutador, es decir cuando, se pulsa, activa lo que anteriormente estuviera desactivado.

Así, por ejemplo, si estamos produciendo un listado de variables en pantalla y ejecutando dicho comando, obligaríamos a que el listado saliera por impresora. Si ahora deseamos volver a listar en pantalla, únicamente deberemos ejecutarlo otra vez.

Vamos a ver a continuación cuáles son las principales rutinas de que consta nuestro programa.

En primer lugar se efectúa la instalación de los nuevos comandos RSX en el sistema; para ello se definen dichos comandos y se efectúa una llamada al firmware, que anuncia al sistema su creación.

La primera rutina con que nos encontramos es la que se encarga de activar el comando con parámetros, o bien sin utilizar ningún parámetro.

Registro A=0 indica que no hay parámetros
Registro A < > 0 indica la existencia de parámetros.

De esta forma, el programa distinguiría las posibles opciones del comando RSX.

Cuando no se utilicen parámetros, el programa saltará inmediatamente a la rutina encargada de buscar todas las variables alfanuméricas que existan en el programa.

Para detectar dichas variables, se ha definido en un buffer la cadena de bytes por la que pueden ser identificadas, y que es la siguiente:

Variable alfanumérica... 3,0,0,nombre

dado que en este caso el nombre de la variable no nos interesa, únicamente se buscará una cadena que contenga los tres primeros bytes.

Una vez encontrada, conoceremos la existencia de una variable alfanumérica en esa dirección de memoria. Para reconocer de qué variable se trata tomaremos los bytes que si-

guen a continuación, hasta encontrarnos con uno que contenga el valor de un carácter ASCII al que se le ha sumado 128, ya que es de esta forma como el Basic almacena dichas variables.

Veamos por ejemplo cómo se encontraría almacenada en memoria la variable HOLA\$:

3,0,0,"H","O","L","A"+128

Cuando se utilice el anterior comando con parámetros, en primer lugar se reconocerá qué parámetros se han introducido, y los colocará en un buffer, detrás de los bytes 3,0,0 identificativos de variable alfanumérica.

A continuación se enviará el control del programa a la rutina encargada de buscar cadenas en memorias. Así por ejemplo si el parámetro es la cadena "POSX", dicha rutina intentará localizar la siguiente secuencia de números:

3,0,0,"P","O","S","X"+128

Por último nos queda por describir el comando encargado de localizar las variables numéricas. La rutina utilizada, también chequea en primer lugar el contenido del acumulador para averiguar si se han enviado o no parámetros.

En el caso de que no existan dichos parámetros, se llama a la rutina encargada de localizar este tipo de variables, las cuales pueden ser reconocidas por la siguiente secuencia de bytes:

13,0,0

De esta forma, cuando se encuentre la cadena, se conocerá la existencia de una variable numérica en esa dirección de memoria, por lo que tomaremos los bytes que siguen hasta encontrarnos con uno que contenga el valor de un carácter ASCII más 128, para conocer cuál es el nombre de dicha variable.

Como podemos comprobar este tipo de variables se almacenan en memoria igual que las tratadas anteriormente.

Cuando se utilice este último comando con algún parámetro, se tomará dicho parámetro y se colocará en un buffer a continuación de los bytes indicadores de variable numérica, y se llamará a la rutina de búsqueda para que los localice, y una vez encontrados se imprimirán en pantalla.

Una vez dadas las instrucciones de funcionamiento y explicadas las rutinas más interesantes, estamos en condiciones de poder utilizar correctamente el programa.

Para ello deberemos copiar el listado ensamblador que aparece a continuación, o bien teclear el programa cargador. Aquellos que utilicen esta última opción, deberán ejecutar el programa una vez copiado, y en caso de que no dé ningún mensaje de error, se deberá salvar de la forma siguiente:

SAVE "RSX", B, \$A000, 585

Cuando se dese ejecutar, deberemos cargarlo en memoria de la forma indicada al principio de ese artículo, sin olvidarnos de efectuar la llamada a la dirección \$A000, con el fin de inicializar los nuevos comandos.

NERO'. La ejecución del anterior comando, produciría un listado semejante al que se indica:

```
00010 POSX
00020 YPOS
00040 DINERO
00060 POSX
00090 YPOS
```

La otra variante de este último comando, nos permitirá buscar una variable numérica en particular, produciéndose únicamente la impresión de dicha variable en pantalla, así como los números de línea en que se encuentra.

AMSTRAD CPC - 464

AMSTRAD



ORDENADOR

SERIE CPC

- **TECLADO** • Teclado profesional con 74 teclas en 3 bloques - Hasta 32 teclas programables - Teclado redefinible
- **PANTALLA** • Monitor RGB verde (12" color (14"))

	Normal	Alta Res.	Multicolor
Col x líneas	40 x 25	80 x 25	20 x 25
Colores	4 de 27	2 de 27	16 de 27
Puntos	320 x 200	640 x 200	180 x 200

- Se pueden definir hasta 8 ventanas de texto y 1 de gráficos • **SONIDO**
- 3 canales de 8 octavas moduladas independientemente - Altavoz interno regulable - Salida estéreo • **BASIC**
- Locomotive BASIC ampliado en ROM - Incluye los comandos AFTER y EVERY para control de interrupciones

AMSTRAD CPC 464

UNIDAD CENTRAL. MEMORIAS

- Microprocesador Z80A - 64K RAM ampliables - 32K ROM ampliables

CASSETTE • Cassette incorporada con velocidad de grabación (1 o 2 Kbaudios) controlada desde Basic • **CONECTORES**

- Bus PCB multiuso, Unidad de Disco exterior, paralelo Centronics, salida estéreo, joystick, lápiz óptico, etc.

• **SUMINISTRO** • Ordenador con monitor verde o color - 8 cassettes con programas - Libro "Guía de Referencia BASIC para el programador" - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR 59.900 Pts. (monitor verde)
90.900 Pts. (monitor color)

AMSTRAD CPC 6128

UNIDAD CENTRAL. MEMORIAS

- Microprocesador Z80A - 128 K RAM ampliables - 48 K ROM ampliables

UNIDAD DE DISCO • Unidad incorporada para disco de 3" con 180K por cara • **SISTEMAS OPERATIVOS**

- AMSDOS, CP/M 2.2, CP/M Plus (3.0)

• **CONECTORES** • Bus PCB multiuso, paralelo Centronics, cassette exterior, 2.ª Unidad de Disco, salida estéreo, joysticks, lápiz óptico, etc.

• **SUMINISTRO** • Ordenador con monitor verde o color - Disco con CP/M 2.2 y lenguaje DR. LOGO - Disco con CP/M Plus y utilidades - Disco con 6 programas de obsequio - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR 84.900 Pts. (monitor verde)
119.900 Pts. (monitor color)

PCW - 8256

AMSTRAD CPC - 6128



ES AMSTRAD

Increible!!

AMSTRAD PCW 8256

UNIDAD CENTRAL. MEMORIAS

- Microprocesador Z80A - 256K RAM de las que 112K se utilizan como disco RAM
- **TECLADO** • Teclado profesional en castellano (ñ, acento...) de 82 teclas
- **PANTALLA** • Monitor verde de alta resolución - 90 columnas x 32 líneas de texto
- **UNIDAD DE DISCO** • Disco de 3" y 173K por cara - Opcionalmente, 2.ª Unidad de Disco de 1 Mbyte integrable
- **SISTEMA OPERATIVO** • CP/M Plus de Digital Research • **IMPRESORA** • Alta calidad (NLQ) a 20 c.p.s. - Calidad estándar a 90 c.p.s. - Papel continuo u hojas sueltas - Alineación automática del papel - Caracteres normales, comprimidos, expandidos, control del paso de letra (normal, cursiva, negrita, subíndices, superíndices, subrayado, etc).
- **OPCIONES** • Kit de Ampliación a 512K RAM y 2.ª Unidad de Disco - Interface Serie RS 232C y paralelo

Centronics • **SUMINISTRO** • Ordenador completo con teclado, pantalla, Unidad de Disco e Impresora - Discos con el procesador de Texto LocoScript, CP/M Plus, Mallard BASIC DR. LOGO y diversas utilidades - Manuales en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR 129.900 Pts.



Los más prestigiosos paquetes de Software Profesional, en formato AMSTRAD... a "precios AMSTRAD"

Existe también la versión **PCW 8512** con **512K RAM** y la 2.ª Unidad de Disco de 1 Mbyte incorporada. **PVP. 149.900 Pts.**
* El **PCW 8256** puede utilizarse como terminal y en comunicaciones.

El I.V.A. no está incluido en los precios.

NOTA: Es muy importante verificar la garantía del aparato ya que sólo **AMSTRAD ESPAÑA** puede garantizarle la ordenada reparación y sobre todo materiales de repuesto oficiales (Monitor, ordenador, cassette o unidades de discos).

AMSTRAD

ESPAÑA

C/ Aravaca, 22. Tel. 459 30 01. Télex 47660 INSC E.
Fax 459 22 92. 28040 Madrid.

Delegación en Cataluña: C/ Tarragona, 110. Tel. 325 10 58.
08015 Barcelona.

EL MUNDO DE LOS «PLOTTERS»

Francisco G.R.

Cuando realizamos unos gráficos y deseamos observarlos desde una perspectiva no tan horizontal, la realizada por el usuario ante el monitor, nos planteamos la necesidad de volcarlo en papel. Esto se puede realizar en una impresora mediante pequeñas rutinas en máquina que nos realizan esta labor.

A

l principio esta solución en válida para un tanto por ciento muy importantes de usuarios. Pero, poco a poco, vamos comprobando que no es una reproducción muy perfecta la que obtenemos en la impresora. Hay varios puntos muy claros en este aspecto: entre ellos podemos citar las circunferencias, que no son tales sino más bien elipses, y luego tenemos las intersecciones o las líneas rectas que enlazan un par de puntos. Los enlaces con los puntos son perfectos, pero la trayectoria de la recta deja mucho que desear con la realidad.

Esto y la necesidad de adaptación a necesidades concretas nos obligan a buscar un periférico de impresión más específico: el PLOTTER. Este periférico está diseñado para trasladar al papel el desarrollo de un plano (en construcción), del diseño de piezas especializadas, de diseño cartográfico, etc.

Salidas por el canal de impresión

En un ordenador hay que tener en cuenta las entradas y salidas de datos por los diferentes canales de la máquina. Como ejemplo, podemos citar que, cuando estamos introduciendo datos por el teclado, éstos se imprimen en la pantalla por el canal de vídeo. Lo mismo sucede cuando mandamos información hacia la impresora o el Plotter: la llevamos por su canal. Este es el encargado de transmitir la in-

formación contenida en la pantalla, o en un fichero, hacia el periférico que tengamos conectado.

Comenzamos tratando las rutas que comunican el procesador y los módulos de entrada y salida de un microordenador. Para ello, empezamos con una introducción sobre las diferencias de comunicación Serie y Paralelo.

Comunicación en Serie

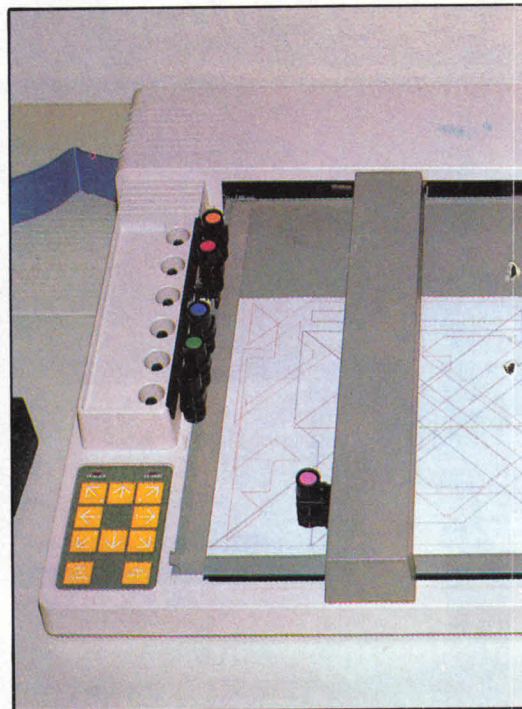
Especialmente cuando se trata de transmitir datos a larga distancia, la comunicación en Paralelo resultaría inviable por el número de hilos necesario para la misma. Por ello, recurrimos a la transmisión en Serie, en la que los bit se suceden ordenadamente en el tiempo; sólo son estrictamente necesarios dos hilos activos de interconexión, uno para salida y otro para entrada de datos.

La diferencia fundamental entre los interfaces para la transmisión en Serie y en Paralelo es que las series deben realizar la conversión del dato de Paralelo a Serie para la emisión, y viceversa para la recepción.

Un ejemplo de un interface Serie es el RS-232C, cuyas siglas significan Recommended Standard. En este interface, además de llevar los dos hilos de interconexión, se dispone de una serie de señales que permite controlar la transmisión.

Comunicación en paralelo

Al contrario de lo que sucede con los interfaces Serie, los paralelos tienen la ventaja de poder realizar transferencias de datos a muy alta velocidad, ya que se transmiten los caracteres en una sola emisión. La desventaja fun-



damental es el mayor número de hilos de interconexión y la corta distancia que pueden cubrir, no sólo en razón de su costo y complejidad, sino porque no son aptas para las comunicaciones por vía telefónica.

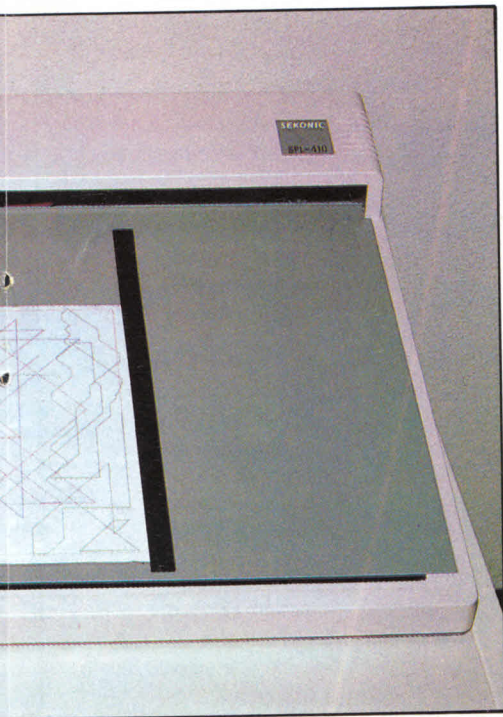
Los interfaces paralelos, por no tener la necesidad de adaptarse a un medio común (en el caso de los interfaces en Serie las líneas telefónicas eran este medio común), no se han sujetado por lo general a ningún standard, y esto provoca que frecuentemente existan problemas de incompatibilidades. Mediante una norma que se elaboró, llamada IEEE 488 y que incorporan la mayoría de los equipos actualmente, aunque aún no todos, se intentó solucionar el problema de las incompatibilidades.

Programas con opción de Plotter en el mercado

Cuando decidimos comprar un Plotter, miramos las posibilidades de resolución que tiene éste, y el tamaño del folio que acepta. También procuramos que disponga del mayor número de plumas de colores posibles, para darle una mayor calidad a los trabajos que realizamos.

El primer paso a seguir será la adaptación de éste a nuestra máquina. Aunque dependiendo de la calidad del Plotter, se puede o no controlar manualmente, tenemos la necesidad de gobernarlo a través de nuestro equipo. Si comenzamos por estudiar los puertos de entrada y salida y cómo mandar la información, probablemente no utilizaríamos el Plotter desde el ordenador.

Para ello nos vamos a programas ya comercializados y que nos den de alguna forma este problema resuelto. Si buscamos y miramos



bien los programas que existen, comprobaremos que hay dos programas en el que se da la opción de salida por Plotter. Estos programas son el D.R. Draw y D.R. Graph, los cuales en las opciones de imprimir tienen la salida de Plotter.

Cuando realizamos un dibujo con el D.R. Draw, una vez retocado, lo podemos mandar por la salida de Plotter, pero podemos encontrarnos con la respuesta de «programa no preparado para salidas de ese periférico», o bien no recibir ninguna señal el Plotter. El mismo caso ocurre con el D.R. Graph: tenemos que preparar la salida del Plotter antes de su utilización.

La inicialización del programa para el ordenador de que dispongamos viene explicada en el manual, pero ya no es tan clara para instalar el Plotter. Lo primero que debemos tener en cuenta es el interface que lleva incorporado nuestro Plotter. Dependiendo de ésta tenemos que obrar de una forma o de otra.

Si tuviera un interface Serie, lo primero que necesitaríamos, tanto para la gama de los CPC o de los PCW, sería la conexión de un interface Serie para el equipo. Este interface lo conectaremos a la salida que llevan nuestros CPC (hay que hacer constar que en el 6128, se co-



necta en el Bus de salida impresora, mientras que en el 464 y los PCW 8256 y 8512, se conectan al Bus trasero de expansión).

En caso contrario al anterior (interface Paralelo), la cosa se reduce a adquirir un cable Centronics y conectarlo directamente al equipo. Esto se puede hacer con la gama de los CPC. Desgraciadamente para los usuarios de los últimos modelos lanzados de **Amstrad**, hay que decir que obligatoriamente necesitan la compra del interface Paralelo, por carecer este equipo de una salida Centronics.

Cuando hemos solucionado todos los problemas de tipo Hardware, de conexión entre ambos, tenemos que instalar el Software. Con la compra de nuestro ordenador nos dan unos discos de Sistemas Operativos, en los cuales también existen unas utilidades para la máquina. Efectivamente, uno de esos programas que hemos sacado por el directorio tantas veces y no sabemos para qué nos sirve y del cual no disponemos de ninguna información, es ahora mismo el programa más importante para la instalación.

Cuando instalemos el programa para nuestro equipo, debemos incorporar la salida pertinente. Lo realizaremos colocando en el disco los DEVICE de OUTPUT de SID (cuando tenemos conectados un interface en Serie), o CEN (caso contrario al anterior, el interface es Paralelo o Centronics). Cuando se trata de una salida en Paralelo, en los modelos **Amstrad** CPC no hace falta preparar la salida, pues están tomados estos valores por defecto. En el caso de Serie sí son necesarios, debido a tener que incorporarles al equipo un periférico externo.

Una vez preparados los DEVICE CEN o DEVICE SID, el ordenador tiene conectados los canales de impresión para su uso. Seguidamente debemos introducir en el disco el programa con el cual el ordenador mande señales concretas y admitidas por el Plotter. Este programa nos viene con el equipo y se denomina:

A > DDHP7470.PRL

De él existen dos versiones dependiendo del Sistema Operativo y del equipo. Existe una en el disco número dos del CP/M Plus del 6128, y la otra se encuentra en la cara número tres de las utilidades del CP/M Plus del 8256/8512. Este programa debe colocarse inmediatamente después de los DEVICE, para su perfecto funcionamiento.

Comandos de manejo de Plotter

Al igual que en una impresora, un Plotter tiene sus comandos de uso interno. En una impresora tenemos comandos de salto de página, retroceso de carro y salto de línea, entre otros.

En un Plotter estos comandos no existen, pero sí tenemos otros, como pueden ser el cambio de tinta, posicionamiento en un punto del papel, radio de una circunferencia, etc.

Como punto de interés por el usuario, vamos a definir la mayoría de los comandos del Plotter para los intrépidos que deseen hacer uso de ellos.

Comando: AA — Arc Absolute (Arco Absoluto)
— Formato: AA x,y,a(,c)
x=Punto de eje x.
a=Grados
y=Punto eje y. c=Grados de retroceso

Comando: AR — Arc Relative (Arco Relativo)
— Formato: AR ix,iy,a(,c)
ix=Incremento de x.
a=Grados
iy=Incremento de y.
c=Grados de retroceso

Comando: CI — CIRCLE (Circunferencia)
— Formato: CI r(,c)
c=Grados

Comando: LB — Label (Etiqueta)
— Formato: LB c1,c2,...
cN[t]
cN= Punto de comienzo, alto, ancho, etc.
t=Espacio entre etiqueta

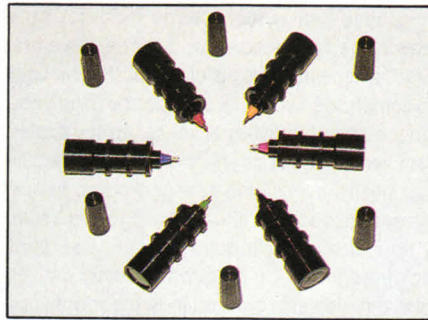


Comando: PU — Pen Up (Pluma Arriba)
— Formato: PU;

Comando: LB — Label (Etiqueta)
— Formato: LB c1,c2,...
cN[t]

cN=Punto de comienzo, alto, ancho, etc.
t=Espacio entre etiqueta

- Comando: PU — Pen Up (Pluma Arriba)
— Formato: PU;
- Comando: PD — Pen Down (Pluma Abajo)
— Formato: PD;
- Comando: PA — Plot Absolute (Plot Absoluto)
— Formato: PA;
- Comando: PR — Plot Relative (Plot Relativo)
— Formato: PR;
- Comando: FT — Fill Type (Relleno de Figura)
— Formato: FT;
- Comando: RA — Rectangle Absolute (Rectángulo Absoluto)
— Formato: RA x,y
x=Punto de eje x. y
Punto eje y.
- Comando: CS — Designate Standard Character Set (Definir Carácter.)
— Formato: CSn
n=Número de Set de caracteres



- Comando: CA — Designate Alternate Character Set (Definir Carácter.)
— Formato: CSn;
n=Número de Set creado por usuario

- Comando: SS — Select Standard Set (Seleccionar un Standard)
— Formato: SS;
- Comando: SA — Select Alternate Set (Seleccionar uno propio)
— Formato: SA;
- Comando: DT — Define Terminator (Definir el término)
— Formato: DT c;
c=valor del punto
- Comando: DI — Absolute Direction (Dirección Absoluta)
— Formato: DI ix,iy ;
ix=Incremento en la dirección X
iy=Incremento en la dirección Y
- Comando: DR — Relative Direction (Dirección Relativa)
— Formato: DR ix, iy ;
ix=Incremento en la dirección X
iy=Incremento en la dirección Y
- Comando: SL — Character Slant (Elongación de carácter)
— Formato: SL c ;
c=Grados de elongación

¿Te falta algún número?

Si estás interesado en algún número de los ya publicados por **Microhobby Amstrad**, realiza hoy mismo tu pedido porque ya hay algunos ejemplares agotados.

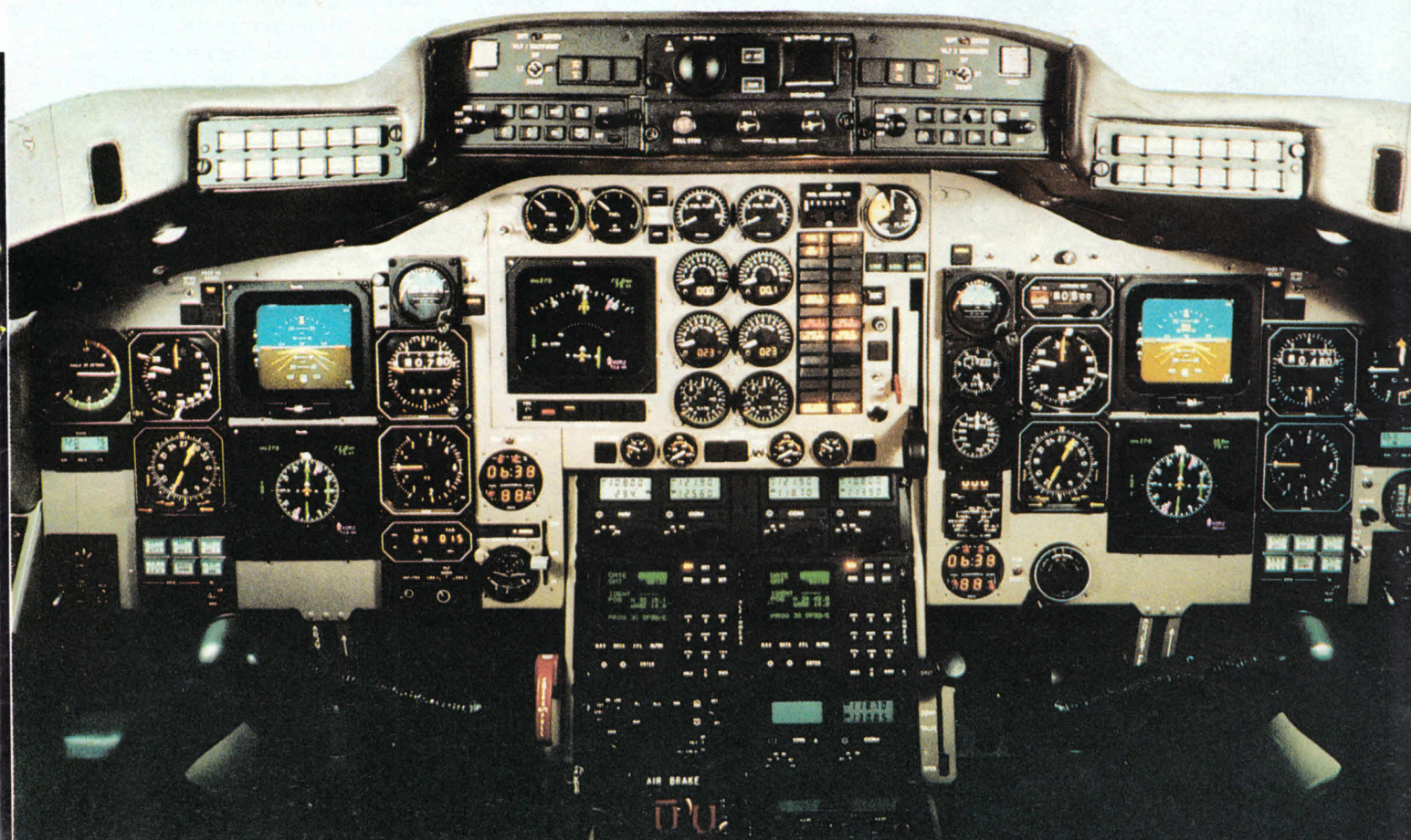
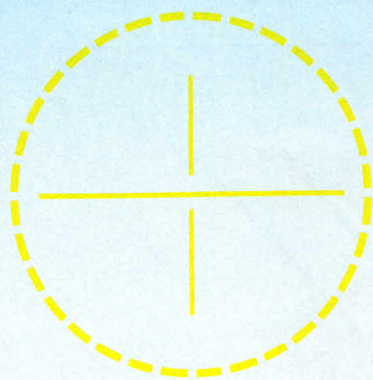
No pierdas la oportunidad de disponer de la mejor obra publicada sobre ordenadores Amstrad. En todos sus números encontrarás interesantes artículos de iniciación, pokes, trucos, curso de código máquina, etc...

¡No te pierdas detalle!

Recorta o copia el cupón que aparece cosido en las páginas de la revista.

SIMULADORES DE COMBATE

De todas las aventuras que podemos vivir sobre el teclado de un ordenador, tal vez una de las más reales y emocionantes sea la de pilotar un avión de caza en busca de aparatos enemigos.



Mucho más que tirar del joystick

Estos programas estaban destinados a llenar los ratos libres de ejecutivos. Mientras se tomaba el café después de una reunión, surgía el momento idóneo para lucir ante los demás nuestras habilidades como piloto.

En un momento la pantalla del ordenador se convertía en el centro de atención, y las palabras de ánimo de los camaradas llenaban la sala.

Con la llegada de los ordenadores domésticos, y la aparición de un elevado número de clientes potenciales de los juegos, ha llegado también la producción masiva de software de entretenimiento, y cómo no, los simuladores de vuelo han tenido un lugar destacado en esta invasión de productos.

Desde las primeras piezas de museo, en las que sólo salía en pantalla la línea del horizonte, y unos cuantos indicadores digitales de altitud, velocidad, etc., hasta los actuales, en los que impresionantes efectos tridimensionales hacen aparecer aviones enemigos que nos lanzan misiles, la cosa ha evolucionado.

Uno de los clásicos en los programas de juego para ordenador, son los simuladores de vuelo. Su nacimiento data de hace tiempo, cuando los ordenadores personales sólo se utilizaban para negocios.

Dentro del mundo de los juegos, los simuladores de combate ocupan un sitio especial. Nos encontramos ante un tipo de juego que precisamente no es de matar marcianitos, estos programas requieren mucho más que tirar del joystick para su utilización. En ellos nos encontramos a los mandos de un caza, y hemos de interceptar al enemigo, que se encuentra en una posición detectada por el radar en tierra.

Una vez conocida la posición de nuestro enemigo, hemos de reproducir todo el proceso que realizaría un piloto de verdad.

En primer lugar hemos de despegar de la pista, para lo cual encenderemos el motor, accionaremos los flaps, aceleraremos y nos elevaremos.

Cuando hemos cogido altura, es la hora de ocultar el tren de aterrizaje y consultar el mapa para localizar la posición del enemigo.

Si el modelo de simulador es avanzado, el ordenador de abordo nos llevará automáticamente al encuentro con él, si no, tendremos que navegar en su búsqueda, valiéndonos de todos los instrumentos de navegación.

Tras varios minutos de singladura, nos encontramos en la posición del avión a cazar: el mapa nos indica su proximidad y nuestros ojos escudriñan el aire a nuestro alrededor en su búsqueda.

Al fin vemos un punto que se aproxima hacia nosotros; es el momento de armar el sistema de misiles y prepararnos para abordarle.

Con el dedo sobre el botón de disparo, intentamos ponernos a la cola del avión enemigo aumentando la velocidad; cuando nos encontramos tras su estela, una ligera pulsación del disparador y presenciamos la trayectoria del misil tras el reactor del aparato. Unos segundos después la explosión del aparato, nos permite añadir una muesca más a nuestro récord de derribos.

Pero mientras observamos con orgullo cómo caen los restos del caza enemigo, nuestros momentos de falta de concentración, han sido aprovechados por otro caza para colocarse a nuestra cola.

Sólo nos queda intentar una maniobra de defensa, o seremos derribados inmediatamente. Forzando al máximo las posibilidades de nuestro caza, realizamos una tijera para intentar colocarnos a su cola.

Hemos fallado en nuestro intento de evasión, y la situación se complica, intentamos un giro en el aire, pero ya es demasiado tarde, un misil ha sido lanzado y su impacto llega certero con una terrible explosión que inunda de fuego nuestra cabina.

En el espacio de tiempo transcurrido desde nuestro despegue, hasta el contacto con el avión enemigo, hemos tenido que accionar un elevado número de controles, al igual que vi-



gilar el calentamiento de motores, gasto de combustible, controlar el rumbo, etc.

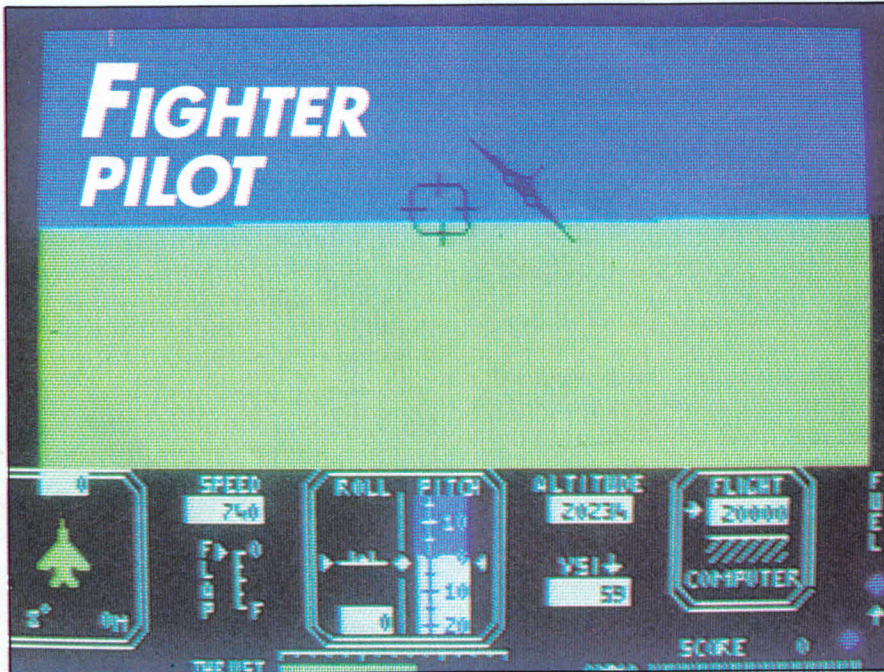
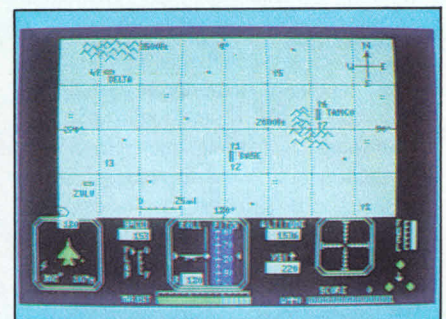
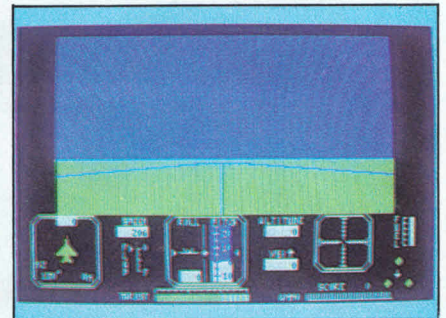
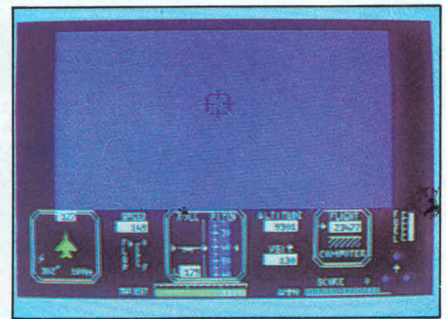
En el combate hemos de situar la mira, aumentar velocidad, colocarnos a la cola del enemigo, armar misiles, tirar de la ametralladora y derribar al otro caza.

Estos son los factores que hacen de un simulador de combate, un programa mucho más complicado e interesante que una simple manzanita de marcianitos.

Computadora de vuelo, se utiliza indistintamente para servir de guía en el aterrizaje, así como para el combate aéreo, donde se demuestra su eficacia.

Indicador de aterrizaje, con él podemos comprobar que nuestra maniobra es correcta, midiendo el ángulo de inclinación que llevamos.

Combustible, indicador del tren de aterrizaje y mapa.



El camino de los simuladores de combate en el **Amstrad**, se inicia con el Fighter Pilot, programa de la casa Digital Integration.

En este caso estamos a los mandos de un F-15, y nos disponemos a interceptar al enemigo.

La cabina de mandos aparece ante nuestros ojos, cuajada de instrumentos e indicadores. Para pilotar nuestro avión disponemos de una amplia gama de controles:

Horizonte artificial, con un avión indicador de la posición y los valores numéricos de los ángulos.

Altimetro e indicador de velocidad.

Indicador de velocidad vertical, con el cual podemos determinar la velocidad de ascenso o descenso de nuestro caza.

Empuje, con indicador gráfico de porcentaje de fuerza de propulsión y calentamiento de motores.

Radar y brújula, con los cuales dirigimos nuestro caza, apoyándonos en las distintas bases de tierra, o siguiendo en el mapa la posición de nuestro enemigo.

Modo de combate.

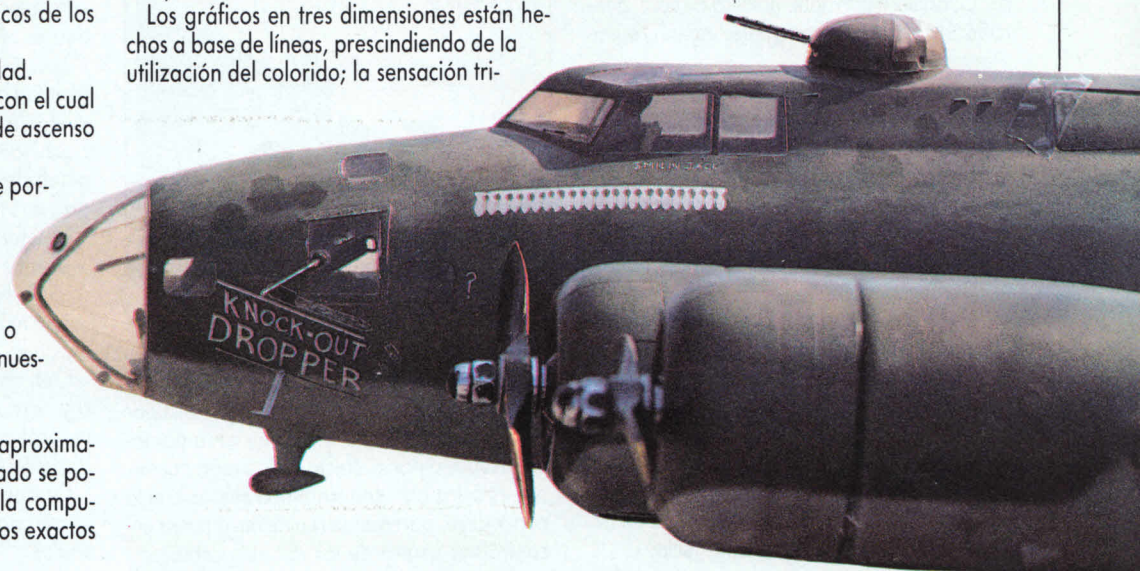
Con él iniciamos la maniobra de aproximación a nuestro blanco. Al ser activado se ponen en funcionamiento el radar y la computadora de vuelo, dándonos los datos exactos de su posición y altitud.

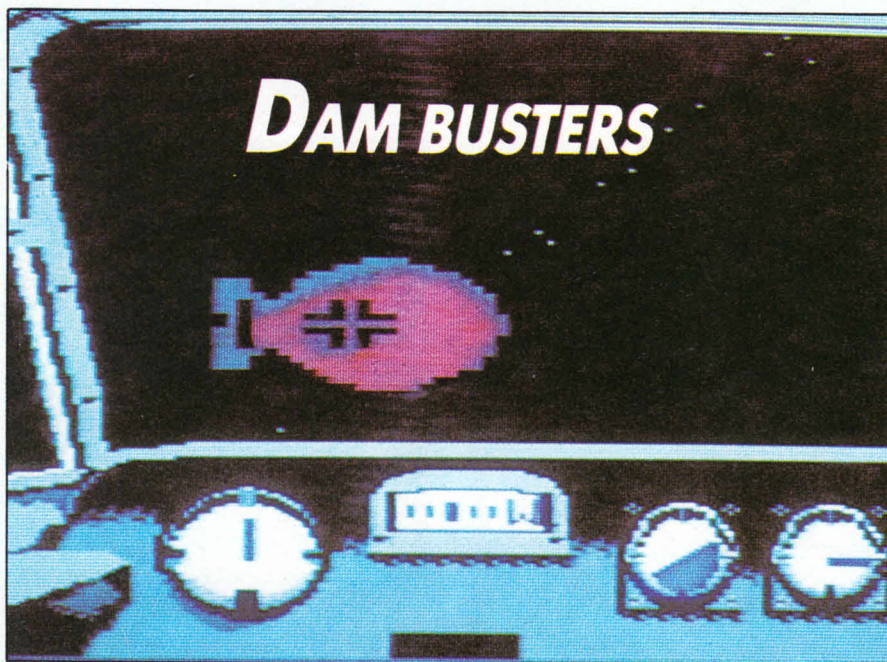
Indudablemente una amplia gama de instrumentos e indicadores, que gobiernan el caza. Para pilotar el F-15 tenemos que manejar junto con el joystick 16 teclas, lo que da idea de la complejidad de la operación de pilotar este aparato. Nos encontramos ante un programa en el que el manejo del avión, debido al elevado número de controles, resulta un poco costoso; en todo momento ante nuestros ojos tenemos presentes los tableros de mandos y a la vista a través de la cabina del piloto.

Los gráficos en tres dimensiones están hechos a base de líneas, prescindiendo de la utilización del colorido; la sensación tri-

dimensional es aceptable, pero poco espectacular.

Un programa con excesivo número de teclas a utilizar, en el cual la navegación se hace lenta y la localización del enemigo bastante laboriosa, recomendado especialmente para los metódicos de las técnicas de vuelo, que se tendrán que armar de paciencia hasta ponerse a la cola del enemigo.





DAM BUSTERS

Todos los programas tratados anteriormente, tenían una característica en común; aunque se tratara de modelos más antiguos o modernos, siempre pilotábamos un caza, el Harrier, el Spitfire, el F-15, o el futurista Skyfox.

Pero como la calenturienta imaginación de los creadores de software no puede estancarse en la vulgaridad, se les ocurre romper con todos los moldes, reproduciendo una acción de guerra real de la Segunda Guerra Mundial.

En primer lugar, ya no pilotamos un caza, sino que estamos a los mandos de un bombardero pesado Lancaster, y como toque distintivo, nuestro objetivo son las presas de la zona industrial del valle del Ruhr, en pleno corazón de la Alemania nazi.

Como se trata de manejar un bombardero pesado, ya no basta con la visión a través de la cabina del piloto, como en los cazas, aquí tenemos una completa dotación de personal con distintos puestos asignados, a los que nosotros debemos sustituir.

Los puestos que hemos de ocupar son los de piloto, artillero principal, artillero de cola, operador de bombas, navegante, ingeniero jefe e ingeniero segundo.

El programa está concebido de forma que el cambio de puesto se realice de forma fácil, y permita el desarrollo paralelo de la misión sin interrupciones.

Cada elemento de la tripulación, tiene asignada una pantalla distinta, de forma que con un simple toque de tecla cambiamos al puesto deseado.

Con este método conseguimos dominar el aparato, solamente con el joystick y el uso de siete teclas, facilitando la complicada tarea que representa hacer las veces de la dotación completa de un Lancaster.

Para romper más aún con la línea clásica de los simuladores de vuelo, Dam Busters introduce un nuevo sistema de pilotaje, que no requiere la pulsación ni de media tecla.

La sala de máquinas, donde se encuentran los mandos que gobiernan los motores del avión, asombrosamente está toda gobernada por el joystick.

Una vez que nos encontramos en la pantalla del ingeniero jefe, aparecen los ocho indicadores de revoluciones y de inyección, y las ocho palancas que los accionan, más las cuatro que sirven para extinguir el fuego en cada uno de los motores.

No hemos de olvidar que el Lancaster está dotado de cuatro motores, cada uno de los cuales se puede manejar por separado.

Para accionar toda esta serie de mandos sólo hemos de hacer uso exclusivo del joystick. Un punto negro aparece debajo del grupo de palancas y para manejar la deseada sólo hemos de mover el punto hasta ella, y una vez situada debajo tirando del joystick hacia arriba o hacia abajo, la palanca se mueve en la misma dirección.



Con este método gráfico de accionamiento de controles, no hay problemas de teclas ni de vigilar números.

Todo el programa está tratado con unos gráficos excelentes, el dibujo de cada pantalla, ayudado por el efecto de la visión nocturna, está tratado con un hiperrealista efecto perspectivo, pareciendo realmente que nos encontramos dentro de las distintas cabinas.

Un programa con un concepto totalmente nuevo, realizado con excelentes gráficos, y en el que los distintos puestos que hemos de ocupar, y la minuciosidad con que hemos de estudiar el lanzamiento de las bombas rompedoras, le dotan de un interés y emoción insuperables. Su creadora la casa U.S. Gold.

Como podemos ver los usuarios de Amstrad interesados por el mundo de los simuladores de vuelo, tienen una amplia gama de posibilidades para elegir.

Sea cual sea el tipo de programa que más se ajuste a nuestras preferencias, encontraremos uno a nuestra medida.

Si deseamos un simulador de los de la vieja escuela, con largas maniobras de vuelo y aproximación, manejando multitud de controles, tenemos el Fighter Pilot.

Si por el contrario nuestro gusto se dirige hacia maniobras más rápidas, utilizando un número más reducido de controles; aderezadas con unas pantallas de cabina y panel de mandos, de un sobervio realismo gráfico, nuestro programa es el Spitfire 40.

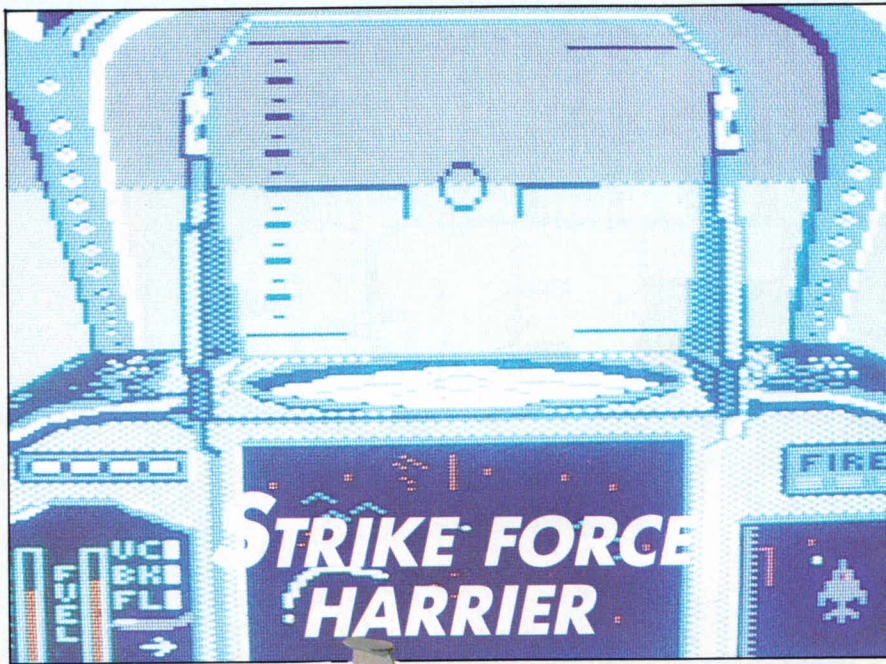


Para los amantes de aviones más modernos y sofisticados, en los que hemos de manejar una gran cantidad de controles, y en los que la misión pasa a ser estratégica, incluyendo combates tierra aire y aire aire, la solución idónea es el Strike Force Harrier.

Los que no quieran complicarse la vida con multitud de teclas y largas aproximaciones de vuelo, lo tienen muy fácil: el Skyfox ofrece la posibilidad de aniquilar a base de misiles o ametralladora; tanques y escuadrillas enemigas, en un programa de acción total y los mejores efectos tridimensionales.

Los que en cambio quieran reproducir el curso de la historia, se encontrarán de lleno en la misión más audaz de la Segunda Guerra Mundial, a los mandos de un bombardero Lancaster, en un programa en que todo ha sido hecho para manejarse sin esfuerzo, con excelentes gráficos y en el que las emociones de una incursión nocturna se saborean hasta el final.

Después de ver todo esto, ¿quién no se pregunta cómo serán los nuevos simuladores de vuelo?



El protagonista de esta nueva pieza de software, es el famoso Harrier, un avión mucho más próximo a nosotros que el anterior, y con la especial característica de que sus motores orientables, le permiten despegar y aterrizar en vertical.

El programa es de la misma casa que el anterior, por lo cual a primera vista ya sabemos que encontraremos un

defensa y ataque en vuelo que permite nuestro Harrier.

En la cabina de mandos tenemos los siguientes controles: VAV dispositivo de información de vuelo. Con él podemos obtener los datos de velocidad vertical, velocidad relativa, girocompás, altura sobre el nivel del suelo y cabeceo del avión respecto a la horizontal.

Mira de bombas, con el punto de posible impacto si se suelta una bomba.

rrizaje, bases de misiles SAM, tanques, aviones y misiles enemigos.

Por último tenemos la pantalla de mensajes, que es la terminal del ordenador de vuelo, en la que se nos da información de inestimable valor.

Sin ninguna duda estamos ante el más completo y sofisticado de los simuladores tratados hasta ahora. Las características del Harrier con sus motores orientables y el concepto totalmente nuevo de este avión de combate, hacen que su conducción requiera una técnica distinta a la de los aviones convencionales.

El número de teclas que accionan los distintos controles y dispositivos es de 20 más el joystick; un completo panel de mandos.

Este programa, además de las características de simulación de vuelo, introduce una nueva variante en los simuladores y nuestra misión no sólo se limita a derribar aparatos enemigos, sino que se trata de una completa incursión en su territorio.

Nuestro objetivo es su Cuartel General, que se encuentra a 500 millas de nuestra posición. Para llegar hasta él, hemos de combatir a las fuerzas acorazadas enemigas que hostigan nuestros puntos de apoyo en tierra, las cuales nos servirán para reabastecer las deficiencias de armamento y combustible.

Aparte de los combates aire tierra, también tenemos que eliminar a los cazas MIG-26, que patrullan los aires en nuestra búsqueda.

Harrier es un completo simulador de vuelo,

producto de una calidad inmejorable.

Si hay algo que impresiona al abrir la carpeta de Harrier, es el librito de instrucciones, que contiene 26 páginas, las cuales están cuajadas de explicaciones de los instrumentos que manejamos, técnicas de vuelo, controles, descripción de la misión, y una completa guía ilustrada de las distintas técnicas de



Localización de puntos de aterrizaje, donde el personal de tierra puede abastecernos.

Mira de misil.

Visualización multifuncional; con información del vuelo y estado del armamento, potencia de empuje, vector de empuje, alimentación de combustible, posición de flaps, tren de aterrizaje y frenos.

Radar de ataque aéreo; que detecta en un radio de cinco millas la posición tanto de aviones enemigos, como de misiles SAM y radar de identificación y seguimiento; con él podemos reconocer el terreno que se encuentra dentro del área de operaciones, en él se identifican montañas, bases de tierra, pistas de ate-

dotado de buenos gráficos tridimensionales realizados a todo color, y en el que se introducen conceptos de misión completa, en la que la estrategia y el aprovechamiento de las excepcionales características de nuestro caza marcan la diferencia.

Otro gran programa de Mirrorsoft.

JUEGA EL JUEGO DEL QUE TODOS HABLAN !

Distribuido en
España por
ERBE
Software

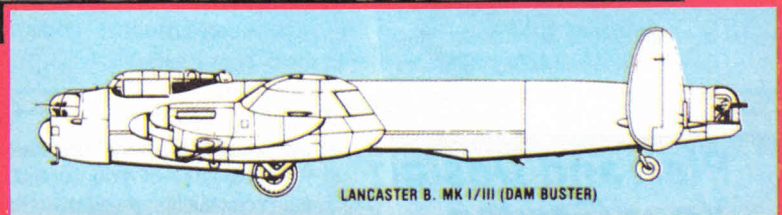
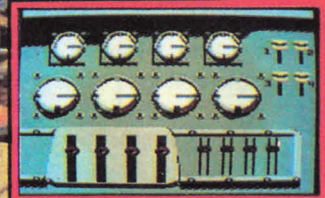
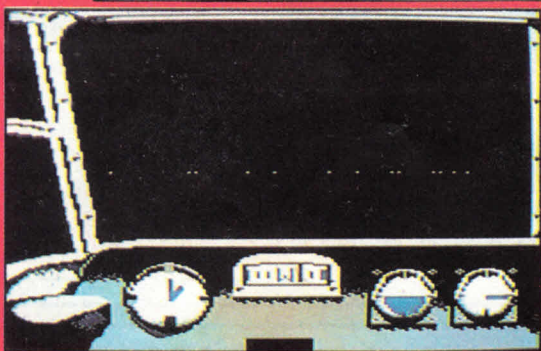
THE DAM BUSTERS



OFFICIALLY ENDORSED BY
R.A.F. No. 617 SQUADRON



Available for
SPECTRUM
48k £9.95



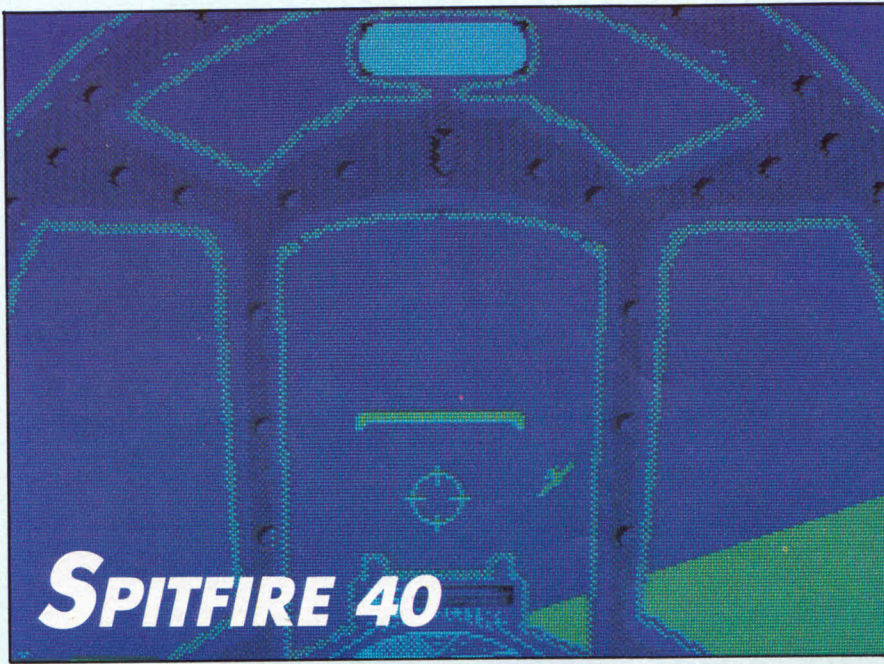
LANCASTER B. MK I/III (DAM BUSTER)

¡Apasionante!

Son las 21'15 horas del 16 de Mayo. Un bombardero Lancaster en vuelo especial, despegando de Inglaterra hacia Alemania. Después de meses de preparación, el escuadrón 617 vuela en una operación destinada a cambiar el curso de la II Guerra Mundial. Su objetivo es destruir las más importantes presas alemanas para paralizar los puntos vitales de sus fábricas de armamento.

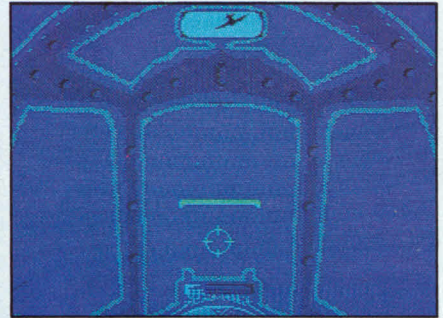
Este detallado y auténtico simulador te permite ocupar los puestos de: **Piloto, Ingeniero de vuelo, Artillero delantero y trasero, Bombardero y Navegante.** Volarás a través del Canal de la Mancha y Europa intentando evitar a los temibles ME-110 alemanes, zeppelins, focos antiaéreos y todos los demás peligros a los que se enfrentó el comando Inglés.

PIDE ESTOS PROGRAMAS A ERBE, SANTA ENGRACIA, 17, 28010 MADRID. TFN. (91) 447 34 10 - Y EN LAS MEJORES TIENDAS DE INFORMÁTICA TIENDAS Y MAYORISTAS... CUMPLIMENTAMOS SUS PEDIDOS EN 24 HORAS



En Spitfire 40 encontramos excelentes gráficos, emoción y tensión; el manejo del aparato no requiere horas de práctica, y la localización de enemigos se realiza de forma bastante rápida. Un programa en el que en la fase de combate se pueden ejecutar técnicas de evasión, picados, loopings y lo que nos apetezca para ponernos a la cola del caza enemigo y aniquilarle.

La casa autora del software es Mirrorsoft.



En este programa nos encontramos ante un modelo histórico, el mítico avión de caza de la II Guerra Mundial.

Nuestra cabina de mandos tiene los siguientes indicadores: horizonte artificial, indicador de velocidad vertical, revoluciones del motor, brújula, altímetro, timón de cola, indicador de giro y deslizamiento e indicador de inclinación.

El conjunto de indicadores y manecillas, reproducen exactamente los controles de este avión, permitiendo un pilotaje preciso y no excesivamente complicado.

Unos minutos de práctica siguiendo los consejos de las instrucciones, nos permitirán hacernos con el aparato; en total manejamos 10 teclas junto con el joystick.

Hasta aquí, podríamos decir que nos encontramos ante un simulador más, brújula, altímetro, tren de aterrizaje, ¿Pero qué aporta de nuevo este programa?

Hemos de decir que lo mejor del Spitfire 40, son sin lugar a dudas los gráficos. En este programa se ha salido del concepto tradicional de simulador de combate, para entrar en una nueva dimensión en esta clase de programas.

La diferencia de los demás, en el Spitfire tenemos la auténtica sensación de encontrarnos en la cabina del piloto.

Tenemos dos vistas básicas; la del tablero de mandos y la de visión del exterior, cada una de las cuales está trata-

da con una riqueza gráfica, que parecen de verdad.

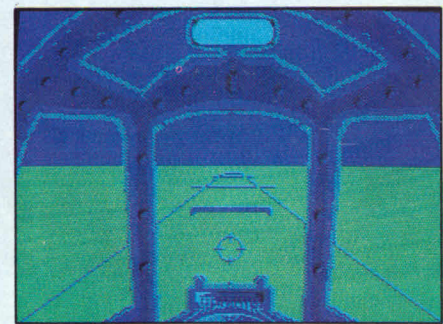
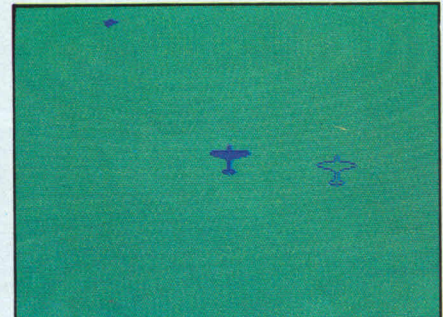
En el tablero de mandos, los indicadores están compuestos por relojes, manecillas y brújulas, todas hechas reproduciendo exactamente los mandos del avión de la II Guerra Mundial.

La visión a través de la cabina está enmarcada por las barras que soportan el fuselaje, dibujadas con un gran efectismo tridimensional, que nos incita a descubrir lo que hay detrás.

Todos los gráficos están hechos en el modo de dieciséis colores, lo que ayuda a darle espectacularidad al juego.

Si hasta aquí el programa parece bueno, no es nada con lo que nos espera en el combate.

Cuando nos aproximamos al avión enemigo, entonces sí podemos apreciar que es un programa en tres dimensiones: delante de nosotros la negra silueta de un caza, se mueve intentando esquivarnos. Comenzamos a ametrallarle sin piedad, y nuestro adversario intenta un giro para escapar. Cuando nos acercamos a él su tamaño crece desmesuradamente y una certera ráfaga le hace caer.



El caza de los Vencedores, Skyfox de DRO SOFT

**El número 1
ahora en disco
para AMSTRAD**



Simulación de vuelo tridimensional de combate aéreo y ataque al suelo.



Cinco niveles, quince escenarios y capacidad de juego estratégico.

Estás en la cabina del caza que sería el sueño de cualquier piloto, pero desde luego eres un mal sueño para el pobre tipo que tienes delante, confiado en una misión sin problemas. Calientale la tobera con tus láser y apartate mientras estalla en una bola de fuego. Rápidamente ponte en picado para caer sobre los blindados

enemigos, como la peste entre los cerdos. SKYFOX es el juego que más rápidamente se está vendiendo en toda la historia de Electronic ARTS.

Posee la más asombrosa animación de alta velocidad que hayas visto en tu ordenador.

Ahora puede ser tuyo totalmente traducido al castellano.

SKYFOX™

EN CASTELLANO

P.V.P.: 2500 pts.



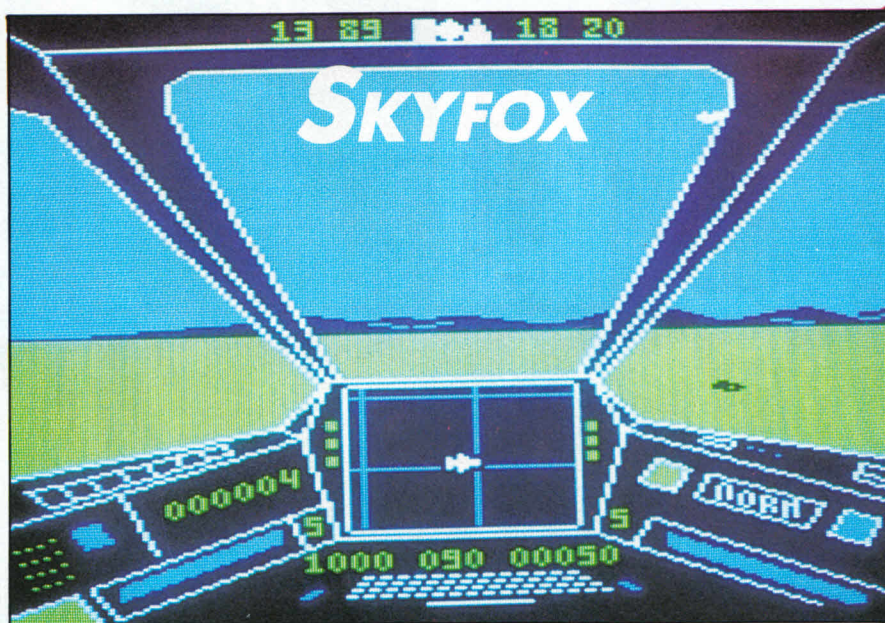
DRO SOFT



ELECTRONIC ARTS

CARACTERISTICAS: NACIONALIDAD: Federación galáctica. **FABRICANTE:** TOBEY ASTRONAUTICS **TIPO:** Caza interceptor multipropósito **PROPULSION AUXILIAR:** Un generador antigravitatorio a 66 MKI **TRIPULACION:** Un humano. **ARMAMENTO:** Dos cañones láser de fuego continuo de 70 kilojulios 10 toneladas de empuje. 5 misiles rastreadores de calor tipo PHOENIX 5 misiles guiados por radar tipo TYPHOON **DEFENSA:** - 2 unidades deflectoras WCRC **AYUDAS ELECTRONICAS:** Radar SCANNER de largo y corto alcance conectable al piloto automático. **VELOCIDAD EN ATMOSFERA:** - 3.000 MPH (Mach IV a 35.000 piés).

Editado por DRO SOFT. Fundadores, 3 - 28028 MADRID
Tlfs.: 255 45 00/09



Si hasta ahora hemos tratado simuladores de vuelo, en los que el control del aparato requería el manejo de una larga serie de teclas y controles, en Skyfox se simplifica este proceso para hacer que la conducción del caza se realice única y exclusivamente con el joystick.

Con Skyfox, salimos de los típicos simuladores de vuelo, y nos adentramos en una versión más cercana a los arcades. En este programa la acción se desarrolla a ritmo trepidante, y nos vemos envueltos en una lluvia de fuego, aniquilando uno tras otro tanques y aviones enemigos.

Nuestro panel de mandos está compuesto por los siguientes indicadores:

Coordenadas de vuelo, reloj digital, pantalla de radar, brújula digital, indicadores de velocidad y altitud, nivel de combustible, estado del escudo protector y número de misiles disponibles.

En la misma pantalla aparece la cabina del piloto junto con el panel de mando. La visión a través de la cabina es uno de los puntos fuertes de este programa.

En él encontramos los mejores efectos tridimensionales producidos por cualquier programa de esta clase. La sensación de velocidad y acercamiento a los elementos hostiles, es verdaderamente apabullante.

Los gráficos de tanques y aviones aumentan de tamaño considerablemente al acercar-

nos a ellos, siendo sin duda los más grandes que se ha visto en esta clase de programas.

Esto refuerza considerablemente la sensación de acercamiento y velocidad, pareciendo incluso que podríamos hasta chocar contra ellos.

Una innovación interesante, que facilita considerablemente la aproximación a los cazas enemigos, es la inclusión del piloto automático de intercepción; con este sofisticado artilugio, el ordenador de vuelo del Skyfox nos lleva directamente hacia los aviones detectados por el radar.

De esta forma evitamos largas navegaciones de aproximación a los objetivos, haciendo que el combate se desarrolle de forma rápida y sin que dejemos de accionar el disparador del joystick ni un solo instante.

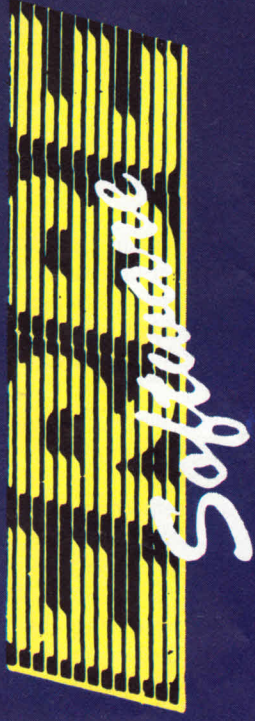
Para hacer fuego contra blancos difíciles disponemos de dos tipos de misiles: los guiados por el calor y los autodirigidos, con los cuales podemos atacar a las escuadrillas de cazas que aparecen ante nosotros.

Nos encontramos ante el simulador de combate que se maneja con más facilidad. En Skyfox no hay que realizar complicadas operaciones de despegue y mantenimiento del rumbo, etc.

Aquí todo se desarrolla de forma rápida y persiguiendo la total aniquilación de lo que aparezca por pantalla, los blancos son detectados por el radar, y el piloto automático nos dirige a su posición con sólo pulsar un botón.

Un caza de la tercera guerra mundial, que pilotaremos con total soltura, sin haber transcurrido más de dos minutos desde que nos hallamos sentados a los mandos.

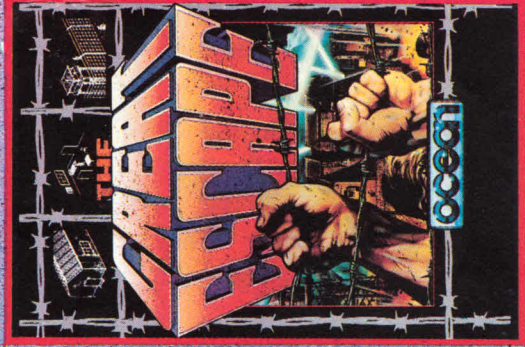
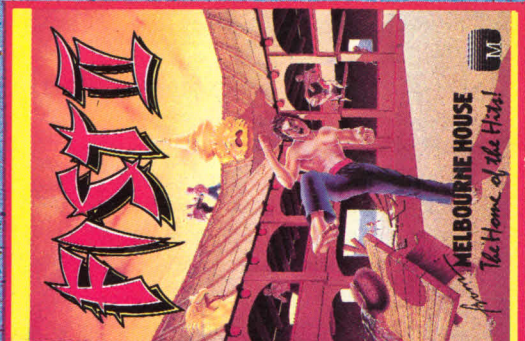
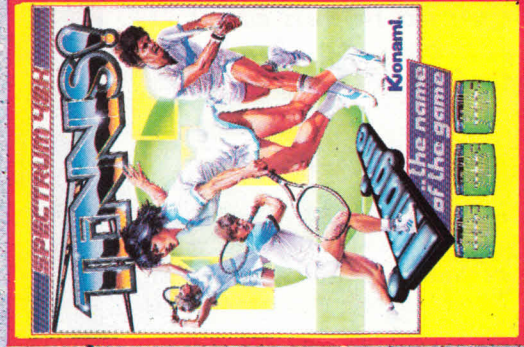
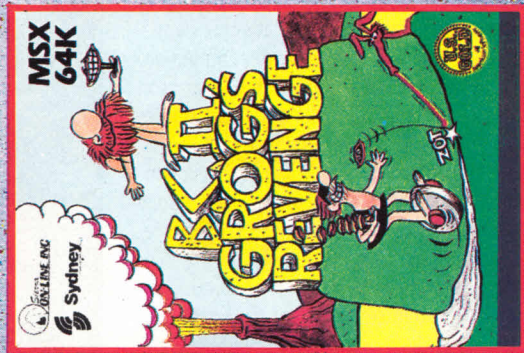
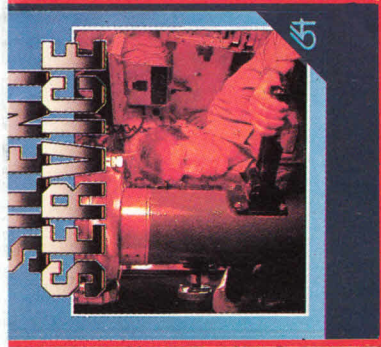
Skyfox es un producto de Electronics Arts.



TE LO OFRECE

ELIGE LO MEJOR





ERBE SOFTWARE
 C/. STA. ENGRACIA, 17
 DISTRIBUIDOR EXCLUSIVO
 PARA ESPAÑA:
 28010 MADRID. TEL. (91) 447 34 10
 DELEGACION BARCELONA:
 AYDA. MISTRAL, N.º 10
 TEL. (93) 432 07 31



ROBOTICA

Los avances técnicos logrados durante los últimos años del siglo XX superan a todos los realizados durante el resto de la Historia del hombre. Si esto ya de por sí es suficiente atractivo, hay un tema cuyo desarrollo lo hace aún mucho más excitante:

¡Estamos hablando del robot!

Y ¿qué poder tiene el robot que no tienen otros descubrimientos, incluido el ordenador (mal llamado «cerbro electrónico»)?

D

Desde los tiempos más remotos se ha convertido al robot en un mito universal que aparece en todas las culturas y religiones que tienen un mínimo de tradición. Vamos a tomar dos ejemplos representativos de todos los que existen en la nuestra para no extendernos mucho y además porque es la que mejor conocemos. *¿De acuerdo?*

¿Quién no recuerda la odisea de aquel escultor, llamado Pygmalión, que se enamoró tan tiernamente de una de sus figuras de mármol que conmovió el corazón de la diosa Atenea hasta el punto que dio vida a la obra y la convirtió en una Galatea de carne y hueso?

Y, por cierto, esta historia tiene un reflejo mucho más moderno. Suponemos que a nadie se le habrá pasado por alto el paralelismo que tiene con la historia de Pinocho que más tarde escribió Collodi. Esto nos confirma que a pesar del paso de los años, la idea de la escultura animada sigue interesando.

Pero no son los únicos. La tradición hebrea nos habla de una figura de barro, a la que se da forma bajo las leyes de la Cábala y debe modelarse en un momento especialmente difícil para la Humanidad, para después insuflarle la vida por medio de una Estrella de David colocada a la altura del «corazón». Se trata del «**Golem**».

La reflexión más importante que podemos sacar de esta historia es que el hombre creaba al Golem para salvar a la Humanidad de algún mal y, una vez superado el peligro, se le destruía. Era como una especie de héroe-esclavo que «**nacía**» con un determinada finalidad y cuando ya había conseguido cumplirla, sencillamente desaparecía.

Además de esta vertiente mitológica, los ro-

bots nos atraen por un montón de cosas más. No cabe duda que para la mayoría de las personas es muy difícil dejar atrás esa especie de «**morbo**» ligado íntimamente a los muñecos, las figuras de cera, los androides que representan algo que se queda a medio camino entre la vida y la falta de ella, la imitan pero son incapaces de participar, al menos conscientemente, en alguno de sus aspectos.

Hasta ahora hemos estado utilizando con toda soltura y sin el mínimo recato la palabra «**robot**», y tenemos que decir que hemos tenido un tremendo lapsus. En el año 1917 se emplea por primera vez, y con propiedad dicha palabra en la obra *Opilec* de Karen Capek, donde se da forma a cada uno de los conceptos que la componen.

Y es en 1942 cuando se comienza a oír hablar de la «**robótica**» como ciencia que trata de los robots, por supuesto. Isaac Asimov fue el iniciador e impulsor de esta nueva parcela del conocimiento, así como de otras muchas ya conocidas por todos nosotros.

De este modo lo anterior a estas fechas entra dentro del hombre artificial o del autómatas, que sería el más claro precedente del robot.

Para conocer datos certeros sobre el campo de los autómatas nos encontramos con problemas verdaderamente insalvables. Muchos de ellos han desaparecido y no podemos comprobar que fueran capaces de realizar las maravillas que sus creadores cuentan largamente en sus crónicas.

Sin embargo, sí tenemos pruebas de verdaderas e ingeniosas maravillas que funcionan a base de aire, vapor de agua o complicados sistemas de relojería que, como ya hemos dicho, pueden considerarse antecesores (*lejanos, eso sí*) de los robots. Pero dejemos este tema aconsejando a los interesados que busquen en su localidad algún museo donde estén recogidas algunas de estas sorprendentes máquinas.



Fue a partir de la Segunda Guerra Mundial cuando las grandes empresas comenzaron a interesarse por la automatización para mejorar la producción en cadena de fábricas y talleres. Los primeros robots industriales se utilizaron en aplicaciones tales como coger piezas y colocarlas en un determinado sitio.

Dentro de este campo hay que destacar la figura de George C. Deval, considerado como el padre de la robótica. Desde nuestra perspectiva actual sus creaciones pueden parecerse un poco ingenuas. Por otra parte era un fiel seguidor de Asimov, lo que quiere decir que quizá fuera más imaginativo que teórico, pero en 1954 ya había postulado todos los elementos y conceptos que forman parte de los actuales robots industriales. ¿Está de acuerdo con nosotros en que todo eso lo convierte en el verdadero padre de la «**robótica**»?



Sin ir más lejos reconoció que todavía existía un largo camino por recorrer en este campo ya que los prototipos de su creación tenían grandes deficiencias. Y también contagió su entusiasmo a Joseph Engelberger (otro enamorado de Asimov), creador de la primera empresa cuya finalidad era la fabricación robótica. Hoy en día la Engelberger Unimation Inc. continúa siendo una de las mayores suministradoras de robots industriales.

Para ser sinceros donde la robótica ha experimentado un verdadero auge tanto por el interés que despierta su estudio como por su utilización masiva ha sido, sigue siendo y probablemente lo será por mucho tiempo en Japón. En este país la robótica industrial no sólo dobla las cantidades de cualquier país industrializado, sino que se renueva constantemente. Día a día podríamos decir sin miedo a exagerar.

Hasta aquí puede servir la introducción, pero entremos directamente en el tema. Y así, de golpe, lo primero que se nos ocurre es preguntarnos: *¿qué es un robot?*

Todos nosotros tenemos dos ideas claramente diferenciadas, que son incluso contradictorias, respecto a la palabra robot. Por un lado tenemos una tradición cultural, de la que ya hemos hablado, potenciada poco a poco por la literatura de ficción (*el género de ciencia-ficción, para ser exactos*) y más recientemente el cine.

Según esta idea romántica, el robot es un ser hecho a semejanza del hombre (más o menos), con unas capacidades cada vez más humanas, de forma que algunos son capaces de tomar decisiones por sí mismos y otros tienen claros deseos de inmortalidad o incluso de procreación.

Frente a este concepto se encuentra la rea-

lidad. No todos los robots están creados a imagen del hombre, es más, la mayor parte de ellos están fabricados imitando funciones humanas, no su aspecto. Se les diseña funcionalmente para que sean capaces de realizar una serie de trabajos muy definidos tal como decíamos anteriormente al referirnos a los robots industriales.

Bueno pero, **¿qué es un robot?** Tal vez podamos decir que es una máquina con capacidad para ser programada y que, por tanto, puede memorizar una secuencia de operaciones y repetirla incansablemente.

Parece un definición muy pobre pero, sin embargo, es lo suficientemente amplia como para abarcar las diferentes generaciones de robots así como sus distintas configuraciones mecánicas y funcionales.

Tal vez debíamos olvidar por un momento la pregunta anterior, realizada tan directamente, para intentar rodearla y llegar a encontrar su respuesta a través del estudio de la robótica actual y así desde el conocimiento de casos particulares tal vez seamos capaces de llegar a un concepto general válido.

De momento, lo que sí parece claro, y aplicable a la mayoría de los casos, es que el robot es una máquina especializada. Y cada uno está pensado, diseñado y montado para cumplir su misión con la mayor exactitud.

Vamos a verlos despacito y de cerca. Para ello los dividiremos en dos grandes apartados: los que no se parecen en nada a los humanos, en cuanto a su estructura y aspecto, y los llamados «androides».

Y antes de empezar a recorrerlos uno por uno, vamos a pedir disculpas a los robots de la lavadora, el horno, la plancha, etc. porque en una sabia decisión unilateral hemos decidido sacarles de la definición de robot. Para ser más exactos diremos que están mucho más cerca de ser pequeños ordenadores. En fin, perdón.

Son ya muy antiguos los mecanismos que tienen por finalidad la manipulación a distancia. Pero es a partir de la Segunda Guerra Mundial y la necesidad de trabajar con material radiactivo la que impulsa el perfeccionismo de estas manos teledirigidas. Desde este momento puede hablarse de una verdadera especialización de estas máquinas y de los brazos de robots, que son los que nos interesan en este caso y que pueden considerarse sus descendientes.

Los fines para los que se crearon fueron meramente industriales. Se usan para coger, colocar, atornillar, perforar, etc. Como verá se trata de realizar trabajos muy específicos dentro de una cadena de montaje, sustituyendo la operación mecánica del hombre. Tienen la ventaja de que no se cansan, no se distraen y tienen una mayor fuerza, pero en cambio son mucho más pesados.

Pero lo verdaderamente interesante de estos brazos, aunque no es una de las áreas más explotadas, es su combinación con un vehículo

móvil. Ciertamente este conjunto perdería algo de fuerza en comparación con un brazo sólidamente apoyado, pero a cambio facilitaría su presencia allí donde fuera necesario.

Si está unido a un ordenador podríamos conseguir evitar choques. Dirigido desde una pantalla, por ejemplo, el robot móvil recoge y transporta cajas de un almacén, libros de una biblioteca, etc. para después colocarlos con precisión en su lugar exacto.

En cuanto a las bases sobre las que están colocados estos brazos son también robots. Están unidas a un ordenador que las dirige a través de unas «pistas» que están formadas por raíles, cables eléctricos e incluso líneas en color. Es evidente que en este último caso deberán estar provistas de unas células sensibles a la luz.

Todos ellos deben disponer de sensores que eviten choques en cualquier circunstancia, deteniendo el vehículo cuando en su camino aparezcan obstáculos más o menos imprevistos, incluido el operario humano que tiene todo el derecho del mundo a cruzar la fábrica o el almacén.

Pero no sólo podemos destinarlos a estas misiones tan concretas. Sofisticando y complementando estos brazos con una serie de células, sensores, luces, cámaras de T.V., etc. son capaces de realizar operaciones mucho más delicadas en lugares donde, bien por condiciones hostiles, bien por economía o por ambas cosas a la vez, sea más beneficiosa la presencia del robot que la humana.

Hablamos, por ejemplo, del espacio exterior, donde la toma de muestras se realiza con estos brazos aplicados a las naves, satélites o vehículos espaciales. Y podemos aplicar esto mismo al fondo del mar, las zonas subterráneas, las prospecciones petrolífera; también pueden resultarnos imprescindibles en algo realmente peligroso: las minas. En este caso estarán dotados también de algún accesorio especial que corte las rocas antes que el brazo comience a actuar.

Otra aplicación importante de este tipo de herramientas es puramente militar. Son unos desactivadores de explosivos realmente eficaces y no ya sólo por la importantísima circunstancia de salvar vidas humanas, sino porque ellos mismos tienen lo que nosotros entenderíamos como «instinto de conservación» y se autoprotegen de la mejor forma para que los daños que sufran sean menores, con lo que la parte económica sale también beneficiada.

Pero a nosotros, usuarios, entusiasmados y enloquecidos con nuestro **Amstrad**, lo que nos interesa más bien es la aplicación que uno de estos brazos podría tener respecto a nuestro ordenador.

En efecto, desde hace ya unos años algunas casas han comercializado unos genios cuya finalidad esencial es la educativa. Un brazo de robot en casa va a servirnos ni más ni menos que para aprender su funcionamiento,

sus posibilidades, su manejo. Para investigar por nosotros mismos e intentar crear o mejorar un programa que lo mueva con precisión. Un ejemplo clásico que le brindamos es intentar jugar al ajedrez o las damas con un brazo educacional y un buen programa, quizá codificado por nosotros mismos, ejecutado por nuestro ordenador.

La gran diferencia entre estos brazos y los industriales es el tamaño y, consecuentemente la fuerza que desarrollan. Por tanto, la gran posibilidad que tenemos es intentar potenciar sus habilidades, por ejemplo, cambiando la pinza (o dedos robóticos) por imanes o viceversa. Incluso los más «manitas» pueden intentar realizar sus propias creaciones con diferentes materiales, darle más potencia (y quizá nos ayude a colocar nuestros libros, discos o videos). No se limite a montar los kits tal como vienen preparados de la fábrica, sino intente hacer un robot a su medida.

Pero quizá mucho más divertidos que estos brazos sean los robots de suelo. Su fabricación no tiene por qué ser más cara que la de los anteriores y, sin embargo, sus posibilidades son bastante superiores.

Su diseño abarca desde la imitación de las más avanzadas tecnologías hasta los clásicos animales de peluche que son, sin duda, de lo más atractivo para los «informáticos» más jovencitos.

Este tipo de robots tiene además una mayor autonomía que los brazos. Por un lado pueden unirse al ordenador por medio de un cable, pero también se pueden utilizar rayos infrarrojos para dirigirlos, aunque en este caso necesitaremos un espacio diáfano entre uno y otro.

En cuanto a sus posibilidades podemos decir que son infinitas. Son capaces de realizar increíbles dibujos y adaptándoles células y sensores que le ayuden a orientarse, evitar obstáculos o cualquier otra cosa que se nos ocurra. Podemos desarrollar toda nuestra experiencia en ingeniería y programación y conseguir adaptar a estos robots de suelo cualquiera de las novedades que podamos imaginar.

Y pasamos ya a los que por su aspecto y funcionamiento son más semejantes a los humanos: los androides. Son, quizá, los más sugerentes pero no por ello los más avanzados técnicamente. Antes de entrar de lleno en ellos, vamos a hacer una advertencia, quizá un tanto superflua, pero que es necesaria para impedir que nuestra imaginación se dispare.

Al hablar de este tipo de robots todos nuestros recuerdos y experiencias se centran en los ejemplos que nos ha mostrado el cine. Todos conocemos a Roddy, a C3PO y R2D2 y alguna que otra estrella del momento. Pero todos ellos son productos de la imaginación literaria, la ciencia no va por ese camino. De hecho tendrán que pasar bastantes años para conseguir prototipos semejantes.

Y además parece que su aspecto físico no interesa en demasía a los científicos que dan más importancia a la especialización y funcionalidad.

Dicho esto, continuemos. Los menos sofisticados son los llamados «maniqués», una serie que intenta imitar la apariencia humana o animal con la mayor perfección posible. Sus movimientos son toscos, limitados y su capacidad de memoria mínima. Son típicos en los videos, escaparates y en alguna que otra manifestación a modo de reclamo. Por todo ello el interés que tienen para nosotros es también bastante anecdótico y superfluo.

¿Recuerda la película «**Almas de metal**»? Es algo que, aunque no sea de vital importancia, al menos puede resultar muy divertido. Se trata de una serie de robots con una perfecta apariencia de estar dotados de vida que les hacía vivir su aventura favorita a cualquier humano que pudiera costeársela.

¿Puede imaginarse a sí mismo luchando con un «**marcianito**» de tres dimensiones en lugar de enfrentarse en la pantalla de su monitor? Con un poco de tiempo, paciencia y perfeccionamiento podremos conseguir hacerlo realidad.

Una aplicación mucho más interesante de los «**androides**» es la que consiste en «**relle-nerles**» de microprocesadores, conectados a su vez con uno central, que imiten los órganos vitales de un humano. De esta manera pueden experimentarse situaciones límite reales, sin el menor peligro, que nos permitan desarrollar métodos capaces de evitarlos o corregirlos.

Los más conocidos son los llamados «**domi**». Se trata de un grupo de androides de diferentes tamaños y pesos que trabajan dentro de los automóviles. Se puede decir que son los «**inventores**» del cinturón de seguridad, cabezales, colchón de aire... Nos explicamos, los domis son los encargados de demostrar las consecuencias que los accidentes de automóvil pueden acarrearle al ser humano de verdad.

Acompañándoles están sus primos hermanos, que trabajan en la Facultad. Si los domis han sufrido toda serie de traumas imaginables, sus familiares se han visto afectados por paros cardíacos, asfixias, tumoraciones y cuantos etcéteras se le puedan ocurrir.

Después de quitarnos el sombrero ante ellos, seamos prácticos. ¿Qué pasa con los androides y los ordenadores caseros?

Pues poco más o menos lo mismo que ocurriría en el caso de los brazos de robot. Por ahora estamos mucho más cerca del juego educativo que de la utilidad realmente práctica.

Dentro de los que tenemos en casa, podemos hacer dos grandes grupos: los de juguete y los domésticos. La verdad es que su apariencia no difiere mucho ya que el cine influye en gran manera sobre los gustos de las personas y R2D2 ha tenido demasiado «**gancho**».



Los androides de juguete se han desarrollado sobre todo en Japón, cómo no. Su apariencia es cada día más benigna y amistosa, aunque todavía recuerdan en parte sus orígenes belicosos, cuando iban cargados de armas espaciales, pistolas de rayos láser o de cualquier otro tipo (simuladas claro).

Sus limitaciones vienen dadas por las del microprocesador que llevan incorporado y que permite programarles para que ejecuten y recuerden una serie de movimientos e incluso alguna que otra frase que podrán intercambiar con los humanos, una vez reconozcan su voz.

En cuanto a los robots domésticos se puede decir que todos ellos se mueven sobre sí mismas, o sea, son autónomas. Se controlan por un ordenador que generalmente llevan en su interior, o bien mediante infrarrojos que lo unen con un micro de mesa.

Pero realmente, *¿qué es lo que pueden hacer?* Aparentemente multitud de tareas caseiras pueden ser realizadas por estos androides. Parece que se acabó el barrer, poner la mesa, ordenar un armario. La situación, sin em-

bargo, es muy diferente. La tecnología con la que están fabricados no está lo suficientemente desarrollada como para que sean unos perfectos ayudantes del ama de casa. De momento nos conformaremos con aprender robótica, electrónica y mecánica con ellos, así como idear un programa que podamos «meter» en su memoria y conseguir perfeccionar o incluso ampliar las primitivas tareas de las que están dotados.

Pero lo más atractivo de estos robots es que, al igual que ocurría con los brazos, con un prototipo, nuestro ordenador y nuestra imaginación, podemos aprender, investigar, inventar y desarrollar cualquier idea por peregrina que nos parezca. Tal vez al principio los resultados no sean todo lo impresionantes que deseemos pero de lo que no cabe duda alguna es que nuestros conocimientos aumentarán. ¿Por qué no se anima y fabrica un prototipo?

A estas alturas no sabemos si estamos en condiciones de responder a aquella pregunta que hicimos sobre la naturaleza de los robots, pero a cambio haremos una reflexión «filosófica».

Hay quien ve en los robots una amenaza. Bueno, pues no lo son.

En principio las posibilidades del robot son todavía limitadas y no pueden considerarse en ningún momento como peligrosas.

Sin embargo, sí podemos tener en ellos una gran ayuda ya que se «encargan» de todas las tareas pesadas y rutinarias, dejando a los humanos las labores creativas y, por supuesto, una mayor cantidad de tiempo libre.

En un futuro el robot cumplirá una labor semejante a la del antiguo esclavo, pero evitando el problema moral que la esclavitud significa en este caso.

El empleo de este concepto es de muy dudosa aplicación al referirnos al trabajo que realiza un robot.

Si le parece que esta premisa es lo suficientemente lógica piense por un momento que hemos llegado al futuro.

Los robots serán lo suficientemente inteligentes como para descubrir que el papel de los humanos es más apasionante que el suyo.

Y entonces...

BLACK JACK

Los juegos de cartas por ordenador, al parecer, constituyen un tema favorito de los programadores y nuestros lectores no son una excepción. Una y otra vez llegan a nuestra redacción multitud de ellos, unos muy buenos, otros menos buenos, pero todos bienvenidos.

Hemos seleccionado uno de ellos, que «computariza» el famoso juego inglés del «Black Jack», conocido en España como 21 y, pariente cercano de las «siete y media».

El autor del programa ya se toma la molestia, unas líneas más abajo, de describir el juego con detalle, así que no lo repetiré aquí. Sólo quisiera llamar la atención de los lectores acerca de la precisión con que están realizados los gráficos de las caras, así como la original y, acertada, disposición de las mismas en la pantalla.

Más de una vez hemos dicho en la revista que, por la magia del ingenio, el byte se convierte en padre y madre del Arte. Black Jack es una prueba de ello.

Antonio Fernando Aguilera Romera



Este programa es una recreación del tradicional juego del Black Jack, también conocido como veintiuno. En este caso se juega a doce partidas, tras las cuales se muestra una pantalla con las máximas puntuaciones, pudiéndose, en caso de entrar entre las mejores puntuaciones, introducir previamente el nombre del jugador. El objetivo del juego es alcanzar los 21 puntos o, en su defecto, una mayor puntuación que la banca, que en este caso es el ordenador. Este juego permite las jugadas habituales, que son las siguientes:

— Doblar la apuesta, lo que se puede hacer cuando nuestra puntuación es de 9, 10 ó 11 puntos.

— Asegurarse, es decir, apostar la mitad de lo que se lleva apostado a la banca. Si ésta obtiene un black Jack (un as y una carta de valor diez), el jugador gana esa cantidad.



— Abrir el juego en dos ramas independientes. Esto sólo se puede hacer cuando las 2 primeras cartas del jugador son del mismo valor y éste no ha doblado la apuesta. Existe una jugada que supera a todas las de-

más: ésta es el Black Jack, que se obtiene cuando se tiene un as y una carta de valor diez (10, J, Q, K). Caso de no alcanzarse ni Black Jack ni 21 por ninguna de las 2 partes, ganará el que más se acerque a 21.

Tasman

SOFTWARE

por fin en España, software a precios británicos

RUN "TASWORD"

© 1984-TASMAN SOFTWARE LTD.

TASMAN
SOFTWARE

TASWORD

¿Se imagina su ordenador convertido en una máquina de escribir? TASWORD es la mejor relación calidad-precio en tratamiento de texto profesional.

Totalmente en castellano, permitiendo realizar MAIL MERGE, trabajar en bloques sin ninguna interrupción incrementando su velocidad, etc... (en versión 6128 aprovecha las 128 K creando un disco virtual de 64 K).

- Acentos, ñ, ü, ?, etc...
- Compatible Productos TASMAN.
- Adaptación impresoras.
- Configuración propia por usuario.
- Ensamblaje de textos.



9.900 pts.

AMSTRAD
COMMODORE
EINSTEIN
MSX



6.900 pts.

AMSTRAD
COMMODORE
MSX
SPECTRUM



7.900 pts.

SPECTRUM

TAS-SPELL

Primer auxiliar que corregirá la ortografía de sus escritos y pondrá los acentos olvidados no dando margen a ningún error. Contiene un potente diccionario con más de 20.000 vocablos pudiendo Vd. ampliarlos. Complemento ideal para su TASWORD con disco.



7.600 pts.

AMSTRAD

Próximamente
en versión PCW 8256
8512

TAS-PRINT

Con TAS-PRINT la escritura elevada a arte. Utiliza las grandes posibilidades gráficas de su ordenador. Las posibilidades tipográficas las explota al máximo al dar una doble pasada optimizando la calidad.

Los tipos de escritura son: COMPACTA MEDIAN DATA-RUN
LECTURA LIGHT PAPER SCRIPT



7.600 pts.

AMSTRAD
EINSTEIN



5.900 pts.

AMSTRAD
SPECTRUM



6.900 pts.

QL
SPECTRUM

TASCOPY

Sin necesidad de un PLOTTER podrá obtener sus gráficos de pantalla a través de la impresora. Un increíble ZOOM le permite realizar sus gráficos en 4 hojas formando un póster de gran tamaño.



7.600 pts.

AMSTRAD



5.900 pts.

AMSTRAD
SPECTRUM



6.900 pts.

QL
SPECTRUM

GRAFMAN

Programa de E.G. Computer Graphics especialmente diseñado para trabajar conjuntamente con TASCOPY representando las funciones matemáticas en desarrollo de diagramas por coordenadas, permitiendo su efecto "ZOOM" ampliar sectores de dichos diagramas.



5.600 pts.

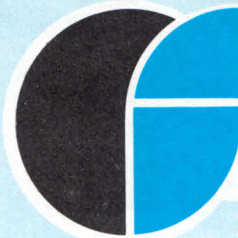
SOLO AMSTRAD



6.200 pts.

• IVA NO INCLUIDO

TOTALMENTE
EN
ESPAÑOL



DE VENTA EN LOS MEJORES COMERCIOS DE INFORMÁTICA
Si Vd. tiene alguna dificultad para obtener los programas, puede dirigirse a:

Ofites
Informática

CONDICIONES ESPECIALES PARA DISTRIBUIDORES
EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA Y PORTUGAL

Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

EL MUNDO DE BLOQUES

Uno de los temas más espinosos de la IA, todavía no resuelto, es lo que los investigadores en este terreno llaman «proceso del lenguaje natural» o «comprensión del lenguaje natural», es decir, conseguir que los ordenadores comprendan el lenguaje humano.

Creo que todos podemos imaginar la complejidad que este propósito entraña, ya que nuestros idiomas están llenos de ambigüedades, frases hechas y coloquiales, etc, que nosotros mismos comprendemos sólo gracias al enorme banco de memoria que tenemos en nuestro cerebro, y a algo indefinible, por ahora al menos, que llamamos inteligencia.

Los computadores no tienen de eso, pero, como ya hemos aprendido en el curso de IA, pueden simularlo de manera bastante convincente (recuerden el programa Eliza).

En el caso del lenguaje, éste sólo puede ser comprendido dentro de un entorno predecible, esto es, que posea una serie de posibilidades o «movimientos» fijos.

Hubo un hombre, Terry Winnograd, que marcó un hito hasta ahora insuperado con un programa, escrito en «Planner», en la comprensión del lenguaje natural; podía mantener conversaciones con su «obra» de una increíble complejidad. Nosotros, con algunas lógicas limitaciones, hemos escrito en Basic una versión del Mundo de Bloques, y, lo que es mucho más decisivo, hemos explicado con todo detalle las técnicas que se usan para que una máquina comprenda nuestra lengua y actúe en consecuencia.

Sin más preámbulos, AMSTRAD Semanal presenta... El Mundo de Bloques.

Por José Antonio Morueco González





La inteligencia artificial es un área de la informática muy importante, que nos permite atacar tareas que no podían ser realizadas hasta hace pocos años. La inteligencia artificial tiene unas técnicas propias especiales y muy variadas según sea el objetivo a conseguir. Estos objetivos se dividen en tres grandes grupos:

- Los sistemas expertos.
- Los procesadores de lenguaje natural.
- La robótica.

Tanto la parte de los sistemas expertos como la de la robótica no tienen mucho que ver con el programa aquí presentado, aunque sí es bueno saber un poco sobre su objetivo primordial: ayudar al hombre a resolver situaciones y a tomar decisiones basándose en el conocimiento «inteligente» que tiene el ordenador en el área del saber en la que estamos. Este conocimiento inteligente, su presentación y su gestión, es el que distingue la inteligencia artificial del resto de las técnicas informáticas.

Lenguaje natural

Por otra parte, que nos atañe más en este artículo, está el procesamiento del lenguaje natural. Para una persona que no esté familiarizada con el mundo de la programación, este problema puede parecerle no muy complicado, porque a él le cuesta muy poco comprender lo que se dice en un escrito, o lo que se dice por la radio. Pero el problema de hacerle entender al ordenador nuestro lenguaje natural, aun teniendo ciertas estructuras, es muy poco sumiso a unas reglas de construcción severas. Esta flexibilidad del lenguaje humano es la que complica las cosas al ordenador.

Los lenguajes con los que el hombre se comunica con el ordenador son de unas estructuras muy severas, y en los que hay pocas palabras distintas, aunque desde fuera pueda parecer todo lo contrario. Cuando una persona hace un programa en BASIC, por ejemplo, tiene que andarse con mucho cuidado de utilizar exactamente las palabras reservadas que hagan falta; en caso contrario tendrá muchos errores. Cuando hablamos con un amigo no es necesaria tanta precisión, y de hecho no pensamos qué palabra usar en esta frase, o qué estructura sintáctica en aquella otra: simplemente decimos lo que se nos ocurra.

Todo esto nos conduce a reflexionar sobre cómo hacer que podamos conseguir que el ordenador comprenda castellano directamente. Para ello es necesario hacer un programa que sirva de interfase entre nuestras palabras y la acción que se quiera realizar. Esta interfase es la que traduce, a código más comprensible para el ordenador, dichas palabras.

Hasta ahora, no se han hecho procesadores de lenguaje natural que puedan considerarse completos, sino que se tiene un conjunto de palabras limitado o palabras claves, que son las que entiende el programa y mediante las cuales intenta comprender las frases dadas, como se verá luego más prácticamente con el programa que se presenta.

Claves del procesamiento del lenguaje natural

El procesamiento de lenguaje natural tiene varios puntos fundamentales:

- El diccionario, o conjunto de palabras que son conocidas y pueden traducirse.



- Las reglas sintácticas y estructuras del lenguaje natural que son analizadas por el procesador.

- Las herramientas que pueden usarse, etc.

Con respecto a las herramientas, es muy importante hacer notar que el BASIC no es precisamente muy adecuado para la inteligencia artificial, lo que produce en algunos casos ciertas limitaciones; se están desarrollando lenguajes, como LISP o PROLOG, más adecuados para estas cuestiones, y permiten unas representaciones de las estructuras más satisfactorias.

De todas formas esto no es un obstáculo insalvable; de hecho, pienso que es interesante ver, en un lenguaje de programación de gran extensión, algunas de estas técnicas, aunque la forma de representación no sea la más óptima.

En el caso que nos ocupa, tenemos un conjunto limitado de palabras que realmente traduce el programa, pero a la hora de usarse parece que hay más, debido a que pueden usarse muchas más palabras, aunque sean ignoradas en la comprensión de la frase. Pasemos pues al programa.

Un procesador de lenguaje natural

¿Qué se puede hacer con el programa?

Antes de ver cómo funciona, veamos un poco cómo se usa. Cuando des al «run», para ejecutar el programa, te aparecerán varios cuadros y triángulos de distintos colores y tamaños, y debajo de éstos, unas líneas de texto.

Inteligencia ARTIFICIAL

Aquí es donde debes ir dándole las órdenes al programa. Estas órdenes consisten en cuál de las figuras quieres que se mueva, y dónde. Por ejemplo puedes escribir:

pon el triángulo pequeño rojo encima del cuadro grande azul

Como puedes observar, tienes también unos números debajo de las figuras, del 1 al 12, que también pueden usarse; por ejemplo, después del movimiento anterior, la posición donde estaba el triángulo pequeño rojo, la número 10, queda vacía. Por ello, puedes ahora llevar allí otra figura; por ejemplo puedes dar la orden:

lleva el cuadro pequeño verde a la posición 10

De esta forma puedes ir realizando los movimientos que desees de una forma natural. Así, por ejemplo, el primer movimiento lo puedes expresar de muchas maneras diferentes en castellano, igualmente comprensible para el programa; algunas de estas formas podrían ser:

pon sobre el cuadro azul grande el triángulo rojo pequeño apila el triángulo rojo pequeño en el cuadro azul grande sitúa debajo del triángulo pequeño rojo el cuadro grande azul

Como ves, hay bastantes variaciones ideológicas para expresar una cosa tan sencilla como ésta. Este es uno de los problemas de traducir por ordenador el lenguaje natural, como decíamos en la introducción. Debido a ello hay ciertas limitaciones, pero bastante razonables. Por ejemplo, no intentes comprender frases que pueden ser ambiguas en su significado, ni frases tratadas de forma poética. — como podrían ser «pon el gran cuadro rojo sobre el triángulo cielo»— cosa que complicaría más todavía la lógica del programa, ya de por sí complicada sólo con un lenguaje normal.

Las leyes semánticas del programa «bloques»

Aparte de las cuestiones sintácticas, también he introducido algunas reglas semánticas sencillas como son:

- No puedes poner figuras del mismo color apiladas juntas.
- No poner nada encima de un triángulo, porque, visto con naturalidad, se caería.
- Algo grande no puede ir encima de algo más pequeño, cosa también razonable, etc.

Estas reglas tienen por misión que se vea cómo se entiende la frase y, o bien se lleva a cabo el movimiento indicado, si es posible, o bien se indica el porqué no se puede hacer.

Para que lo veas, haz todas las combinaciones posibles de movimientos, e irás viendo que hay muchos posibles.

Por último notar que para acabar basta con decirle «adiós».

Visto ya cómo usarlo entremos un poco en su construcción. La idea general consiste en adivinar qué palabras clave están en la frase del usuario y cuál es la estructura de la frase.

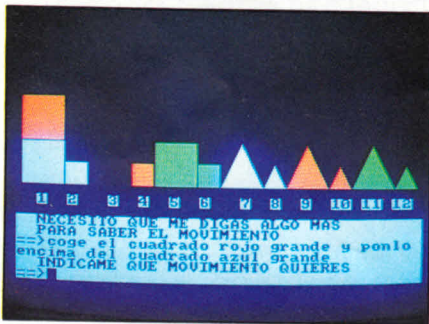
Representación Del conocimiento

Cuando empiezas a pensar en cómo construir un programa así, lo primero que hay que limitar es el número de palabras y estructuras que van a ser comprendidas. En nuestro caso tenemos 5 estructuras generales, las cuales son bastante razonables.

Estas estructuras son:

1. Poner el objeto A encima del objeto B.
2. Poner el objeto B debajo del objeto A.
3. Poner encima del objeto B el objeto A.
4. Poner debajo del objeto A el objeto B.
5. Poner el objeto A en la posición n (con n entre 1 y 12).

donde el objeto puede ser un triángulo o un cuadrado, de un cierto tamaño (grande o pequeño), y un cierto color (verde, azul o rojo).



Cada objeto, junto con sus atributos, se considerará como un todo a la hora de analizar la frase. También hay que notar que llamo objeto A al que está encima, y objeto B al que va a estar debajo; esto se usará al hacer el programa.

En el programa hay varias partes, que iremos viendo:

— El programa principal, es la mayoría del programa, y donde se va analizando la estructura de la frase y, una vez hecho esto, se llevan a cabo las acciones semánticas (movimiento a hacer o mensaje de por qué no hacer el movimiento).

— Las subrutinas 2500 y 3000, cuya misión es dibujar, o borrar, cuadrados y triángulos, respectivamente.

— La subrutina 2000 es clave y tiene por misión decir en qué posición está una palabra (pal\$) en una frase (fr\$), si está.

Vayamos al programa principal línea a línea. Para comprenderlo, veamos primero las variables más importantes y su función:

— $px(i), py(i)$ — con i entre 1 y 12 — tienen la posición horizontal y vertical de los objetos. Estos objetos están numerados del 1 al 12 y son:

- 1 = cuadrado azul grande.
- 2 = cuadrado azul pequeño.
- 3 = cuadrado rojo grande.
- 4 = cuadrado rojo pequeño.
- 5 = cuadrado verde grande.
- 6 = cuadrado verde pequeño.
- 7 = triángulo azul grande.
- 8 = triángulo azul pequeño.
- 9 = triángulo rojo grande.
- 10 = triángulo rojo pequeño.
- 11 = triángulo verde grande.
- 12 = triángulo verde pequeño.

En las primeras líneas se inicializan estas posiciones para la configuración inicial. También, asociada a éstas, está $p(i)$, que mantiene siempre las posiciones iniciales de $px(i)$, y sirve para los números que aparecen en pantalla del 1 al 12, y para cuando se indique que se ha de poner algo en dichas posiciones.

— $db(i)$, indica qué figura está debajo de la i -ésima; si no hay ninguna valdrá 0. Por ejemplo, si el triángulo rojo pequeño está encima del cuadrado azul grande entonces $db(10) = 1$.

— $oc(j)$ — con j entre 1 y 6 — indica si el cuadrado número j está cubierto — $oc(j) = -1$ — o si está descubierta — $oc(j) = 0$.

— La variable p va a usarse para posiciones en frases donde está cierta palabra, según se calcula en la subrutina 2000. Además hay variables pr , pa y pb que indican posiciones en la frase de la preposición, el objeto A y el objeto B. Está también la variable n , que indica la posición de la frase donde se ha de empezar a buscar la palabra.

— na y nb contienen al final los números asociados a los objetos A y B respectivamente — números del 1 al 12.

— Otra variable importante es hn que indica si la estructura de la frase es la quinta, es decir, en la que se indica un movimiento a una posición numérica, en cuyo caso hn vale -1 . En caso contrario valdrá 0.

— También está la variable cs , que vale 1 para las estructuras 1 y 4, en las cuales el objeto A está delante, en la frase, del objeto B; y vale 2 en las estructuras 2 y 3, que ocurre lo contrario.

— La variable ar indica si la estructura usa preposiciones como en, encima de, sobre, en cuyo caso ar vale -1 , o si se usan preposiciones como bajo, debajo, en cuyo caso vale

0. Es decir, es otra variable para discernir la estructura.

— La variable aux indica si el objeto A es un triángulo — $aux = 6$ — o es un cuadrado — $aux = 0$.

— La variable t indica si el objeto es grande o pequeño, según valga 32 ó 16; t es la apotema del objeto. Este valor depende de si el número del objeto es par —pequeño— o impar —grande.

— Para indicar el color están las variables ca y cb .

Hay alguna variable más, pero éstas son las más importantes.

Las demás las iremos viendo cuando aparezcan.

Veamos ya la realización del programa.

* Las primeras líneas, hasta la 280, es la inicialización de las variables, dibujar en pantalla el estado inicial, con el control de colores conveniente, la construcción de la ventana de texto.

Esta parte es sencilla y su única complicación es el cálculo de las posiciones en pantalla en donde se han de dibujar los objetos.

Después se escribe el texto inicial y, en la línea 300, se pide ya el movimiento deseado, que se mete en la variable $fr\$$.

Se mira si la palabra es «adiós» para terminar.

A partir de aquí, se empieza ya el análisis de la frase con la intención de saber al final del análisis el valor de na y nb , que representan el número del objeto de arriba y abajo.



Entre las líneas 330 y 370 se mira si hay en la frase algún número del 1 al 12. Si lo hay estamos en la quinta estructura y hacemos $hn = -1$. Si no la hay hacemos $hn = 0$.

Si hemos encontrado el número, que será el ji , miramos —líneas 410 a la 450— si dicha posición ji está ocupada; esto ocurrirá cuando exista algún j con $p(ji) = px(j)$, es decir, cuando algún objeto tenga su posición horizontal en la misma de ji . Si está ocupada, entonces se manda el mensaje correspondiente y se vuelve a empezar con otra orden. En caso contrario, se va a la línea 620, para ver qué objeto es el que ha de llevar a la posición ji .

Si $hn = 0$ entonces sigue por las líneas 460 hasta la 550, en donde se va viendo cuál es la preposición que se usa en la frase. Según cual sea, ar valdrá -1 ó 0 , según vimos al ver las variables. Si no se encuentra ninguna de estas preposiciones se manda el mensaje correspondiente y se empieza de nuevo. En la 610 ponemos en pr la posición donde estaba dicha preposición.

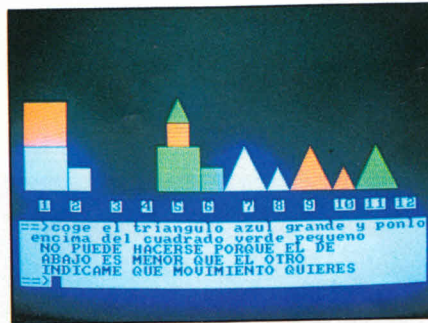
En las 620-630 se comprueba si la palabra cuadrado está en la frase. Si no está y no estamos en la quinta estructura, entonces es imposible porque el objeto de abajo tiene que ser un cuadrado y se manda el mensaje conveniente de la línea 650.

En caso de la quinta estructura, si hn , es posible que no esté ningún cuadrado en la frase, y será un triángulo. Si no estamos en la quinta estructura, entonces hacemos $pb = p$, lo cual es provisional, y buscamos en la frase si hay, después de el cuadro encontrado, otro cuadro —por eso tomo $n = pb + 8$, posición

posterior a la de cuadrado. Si lo hay, no hace falta buscar si está el triángulo y nos saltamos las líneas 710-740; y variamos las posiciones pa y pb , suponiendo el caso 1. Las posiciones se cambiarán o se dejarán según la estructura que tengamos.

Si no hay dos cuadrados se mira si hay triángulo, como antes con los cuadrados; si no lo hay se manda el mensaje como pa , y $aux = 6$, como se vio al ver las variables.

Así ya tenemos en pa y pb las posiciones de los dos objetos. En caso de la quinta estructura no es necesario y simplemente no influirá en su ejecución dichos parámetros.



Una parte fundamental está entre las líneas 750-760, en donde se analiza la estructura en la que nos encontramos. En las líneas 750 y 753, por medio de dos instrucciones condicionadas, se consigue saber cuál es el valor de cs :

- Si $pr < pa$ y $pr < pb$ y $ar = 1$ entonces estaremos en la estructura 2 con lo que $cs = 2$.
- Si $pr < pa$ y $pr < pb$ y $ar = 0$ entonces estaremos en la estructura 4 con lo que $cs = 1$.
- Si no se da la condición de las posiciones y $ar = -1$, estaremos en la estructura 3 y $cs = 2$.

Tras esto sabremos ya el orden del objeto A y el objeto B:

- $cs = 1$: el objeto A delante del objeto B.
- $cs = 2$: el objeto B delante del objeto A.

Recordar que el objeto A es el de arriba y el objeto B es el de abajo, por lo cual es muy importante saber su orden. Todos estos cálculos se consiguen viendo todos los casos posibles que se pueden presentar en las estructuras, y, por ello, es lo más complicado de entender.

Después, en la línea 756, se comprueba si el objeto B, el de abajo, es un triángulo, por medio de un if un poco complicado, pero sólo ve las posibilidades de que así sea; en dicho caso nos vamos a la línea 650, donde se da el nombre conveniente.

Por último, en la línea 758, si $cs = 2$, es decir, si estamos en una estructura con el objeto B delante del objeto A, se intercambian pa y pb , para que tengan ya la posición de A y B de forma correcta, respectivamente.

Entre las líneas 760 y 950 se calcula el color que tiene cada objeto A y B actuales. Para ello, se usa un $data$ con las palabras azul, rojo y verde, y sus números asociados, 1, 2

y 3. Si estamos con $hn = -1$ entonces es más sencillo porque sólo se busca un color.

Veamos qué pasa en los demás casos. Para ello se toman dos subpartes de la frase total:

- $fr\$ (0)$ es la correspondiente al objeto A.
- $fr\$ (1)$ es la correspondiente al objeto B.

Estas se calculan según estemos en unas estructuras o en otras, dependiendo del valor de cs y vía unos cálculos sencillos de las posiciones pb y pa . Después se van leyendo de los datos estos colores para cada subfrase adecuada, obteniendo en $n(0)$ el color de A, y en $n(1)$ el de B, valores que se traspasan a ca y cb respectivamente. Si no se encuentran estos colores se saca el mensaje adecuado, saltando en la línea 910 a la 720.

Si los colores coinciden, también se lanza el mensaje adecuado. En el caso de $hn = -1$ sólo tenemos, como es de esperar, un sólo color en ca .

Entre las líneas 960-1080, se calcula, de forma parecida, los tamaños de los objetos A y B, los cuales están al final en $n(0)$ y $n(1)$ respectivamente. El procedimiento es similar al anterior, incluido para $hn = -1$.

Ahora, calculo ya los valores de na y nb , que son los números asociados a los objetos A y B que se han calculado. Si estamos en la estructura 5, $hn = -1$, $nb = 0$ y no hay problemas. En la línea 1110 se calcula, en caso de la estructura 5, si el objeto cabe en la posición en donde se le intenta colocar.

En estas líneas se comprueba también si el cuadrado de abajo, objeto B, estaba ya ocupado, $oc(nb) = -1$, o si el de arriba está ya cubierto. En ambos casos el movimiento no se hace y se manda el mensaje conveniente. También se ha comprobado si el objeto de arriba no es mayor que el de abajo.

Lo que queda de programa principal es borrar el objeto A de donde esté y colocarlo sobre el objeto B. Para borrarlo se va a la subrutina 2500 ó 3000, según sea cuadrado o triángulo, y con $br = -1$ se pinta con el color de fondo dicho objeto. Para dibujarlo en su sitio actual se calcula, según hn y según el objeto sea grande o pequeño, los valores de las nuevas posiciones; la variable xx se usa para saber, antes de borrar el objeto, si va a caer, en la nueva posición, dentro de la pantalla.

Por último poner debajo del objeto A el objeto B — $db(na) = nb$ — y desocupar el anterior, lo cual se hace antes, como debe ser. Ya sólo decir que las subrutinas de dibujo son más sencillas. Sólo indicar que la variable p indica el color.

Conclusión:

Espero que hayas entendido las ideas de la construcción del programas y que te sirva para que veas lo complicado que resulta llegar a comprender, digitalmente, el lenguaje humano.

También es importante notar que el BASIC no es muy adecuado, como ya quedó dicho y como se demuestra al seguir el programa paso a paso.

```

10 DIM PX(12),PY(12),p(12),DB(12)
20 MODE 1
30 INK 0,11:INK 1,0:INK 2,9:INK 3,6
40 BORDER 0
50 PAPER 1:CLS
60 PAPER 0:WINDOW #1,1,40,20,25
70 FOR I=1 TO 6:PRINT #1:NEXT
80 PX(I)=43:P(I)=43
90 FOR I=2 TO 12
100 PX(I)=PX(I-1)+52
110 P(I)=PX(I)
120 NEXT I
130 FOR I=1 TO 12
140 IF I MOD 2=0 THEN PY(I)=161 ELSE
E PY(I)=177
150 NEXT I
160 FOR I=1 TO 6
170 GOSUB 2500
180 NEXT I
190 FOR I=7 TO 12
200 GOSUB 3000
210 NEXT I
220 FOR J=1 TO 12
230 K=K+3
240 IF K=9 OR K=22 THEN K=K+1
250 LOCATE K,18:PRINT MID$(STR$(J),
2,2)
260 NEXT J

```

```

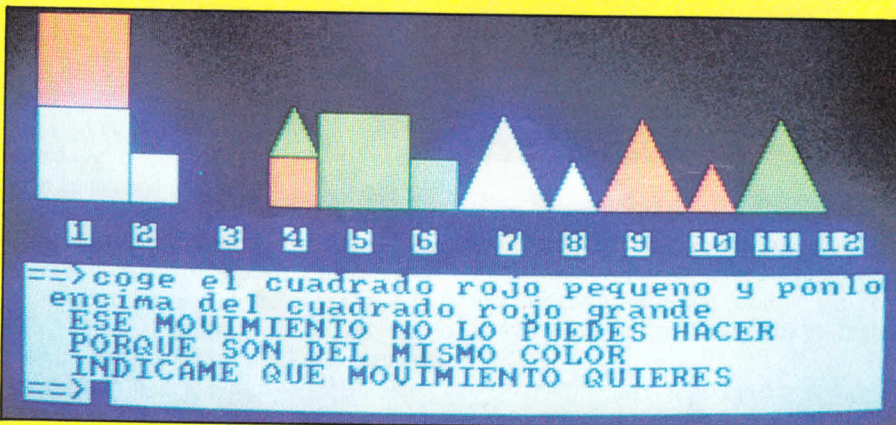
580 AR=-1
590 GOTO 610
600 AR=0
610 PR=P
620 PAL$="CUADRADO":GOSUB 2000
630 IF P>0 THEN 660
640 IF HN THEN 700
650 PRINT #1," SI PONES ALGO ENCIM
A DE UN TRIANGULO PUEDE CAERSE.R
EPITE":GOTO 290
660 IF HN THEN 740 ELSE PB=P
670 N=PB+8
680 GOSUB 2000
690 IF P<>0 THEN PA=PB:PB=P:GOTO 7
50
700 N=1
710 PAL$="TRIANGULO":GOSUB 2000
720 IF P=0 THEN PRINT #1," NO ME D
AS SUFICIENTE INFORMACION":GOTO 290
730 PA=P:AUX=6
740 IF HN THEN I=0: P=0:GOTO 860
750 IF PR<PA AND PR<PB THEN CS=2 EL
SE CS=1
753 IF NOT AR THEN CS=3-CS
755 IF AUX=0 THEN 758
756 IF (CS=1 AND PB<PA) OR (CS=2 AN
D PB>PA) THEN 650 ELSE 760
758 IF CS=2 THEN AB=PA:PA=PB:PB=AB

```

```

1170 IF (NB MOD 2)<(NA MOD 2) THEN
PRINT #1," NO PUEDE HACERSE PORQUE
EL DE ABAJO ES MENOR QUE
EL OTRO":GOTO 290
1190 IF OC(NB) THEN PRINT #1," ESE
CUADRADO YA ESTABA CUBIERTO":GOTO
290
1200 IF NA<7 THEN IF OC(NA) THEN PR
INT #1," EL CUADRADO DE ABAJO ESTA
CUBIERTO":GOTO 290
1210 IF NA THEN 1220 ELSE xx=PY(NB)
+20+16*((NB MOD 2)+1)+16*(NA MOD 2)
:GOTO 1230
1220 IF I MOD 2=0 THEN XX=161 ELSE
XX=177
1230 IF XX+T>400 THEN PRINT #1," T
E SALES DE LA PANTALLA":GOTO 290
1240 I=NA:BR=-1
1250 IF NA>6 THEN GOSUB 3000 ELSE G
OSUB 2500
1260 IF HN THEN PX(I)=P(JI) ELSE PX
(I)=PX(NB)
1270 IF NOT HN THEN OC(NB)=-1
1280 PY(NA)=XX
1290 BR=0
1300 IF NA>6 THEN GOSUB 3000 ELSE G
OSUB 2500
1310 OC(DB(NA))=0
1320 DB(NA)=NB
1330 GOTO 290
2000 REM posicion de fr$ donde esta
pal$
2010 M=N
2020 p=INSTR(M,fr$,pal$)
2030 IF p=0 THEN 2120
2040 IF p=1 THEN 2070
2050 A$=MID$(fr$,p-1,1)
2060 IF A$(">" " AND A$(">")," AND A$(">
");" THEN 2100
2070 IF p+LEN(pal$)-1=LEN(fr$) THEN
2120
2080 A$=MID$(fr$,p+LEN(pal$),1)
2090 IF A$(">" " AND A$(">")," AND A$(">
");" THEN 2100 ELSE 2120
2100 M=p+LEN(pal$)
2110 GOTO 2020
2120 RETURN
2500 REM cuadrado
2510 IF BR THEN P=1:GOTO 2540
2520 p=(i-1)\2
2530 IF p=1 THEN p=3
2540 GRAPHICS PEN p
2550 IF i MOD 2=0 THEN t=16 ELSE t=
32
2560 PLOT px(i)-t,py(i)-t
2570 DRAW px(i)+t-1,py(i)-t
2580 DRAW px(i)+t-1,py(i)+t-1
2590 DRAW px(i)-t,py(i)+t-1
2600 DRAW px(i)-t,py(i)-t
2610 MOVE px(i),py(i)
2620 FILL p
2630 RETURN
3000 REM triangulo
3010 IF BR THEN P=1:GOTO 3040
3020 p=(i-7)\2
3030 IF p=1 THEN p=3
3040 GRAPHICS PEN p
3050 IF i MOD 2=0 THEN t=16 ELSE t=
32
3060 PLOT px(i)-t,py(i)-t
3070 DRAW px(i)+t-1,py(i)-t
3080 DRAW px(i),py(i)+t-1
3090 DRAW px(i)-t,py(i)-t
3100 MOVE px(i),py(i)
3110 FILL p
3120 RETURN
3500 DATA "AZUL",1,"ROJO",2,"VERDE"
,3,"NO",0

```



```

270 PRINT #1," TIENES EN LA PANTAL
LA CUADRADOS Y TRIANGULOS GRA
NDES Y PEQUENOS"
280 PRINT #1," PUEDES INDICAR SI Q
UIERES PONER ALGO EN DONDE ESTAN
LOS NUMEROS, DICIENDOLO"
290 PRINT #1," INDICAME QUE MOVIMI
ENTO QUIERES"
300 AUX=0
310 LINE INPUT #1,"==>",fr$
320 FR$=UPPER$(FR$)
330 N=1
335 PAL$="ADIOS":GOSUB 2000
336 IF P>0 THEN CLS:END
340 FOR JI=1 TO 12
350 PAL$=MID$(STR$(JI),2,2)
360 GOSUB 2000
370 IF P>0 THEN 400
380 NEXT JI
390 HN=0:GOTO 460
400 HN=-1
410 FOR J=1 TO 12
420 IF P(JI)=PX(J) THEN 450
430 NEXT J
440 GOTO 620
450 PRINT #1," NO PUEDES PONER NAD
A EN EL NUMERO ";JI;" PORQUE E
STA OCUPADO":GOTO 290
460 PAL$="EN":GOSUB 2000
470 IF P>0 THEN 580
480 PAL$="ENCIMA":GOSUB 2000
490 IF P>0 THEN 580
500 PAL$="SOBRE":GOSUB 2000
510 IF P>0 THEN 580
520 PAL$="DEBAJO":GOSUB 2000
530 IF P>0 THEN 600
540 PAL$="BAJO":GOSUB 2000
550 IF P>0 THEN 600
560 PRINT #1," NECESITO QUE ME DIG
AS ALGO MAS PARA SABER EL
MOVIMIENTO"
570 GOTO 310

```

```

760 FT$=FR$
770 IF CS=2 THEN 810
780 FR$(0)=LEFT$(FT$,PB)
790 FR$(1)=RIGHT$(FT$,LEN(FT$)-PB)
800 GOTO 830
810 FR$(0)=RIGHT$(FT$,LEN(FT$)-PA)
820 FR$(1)=LEFT$(FT$,PA)
830 FOR I=0 TO 1
840 N=1:P=0
850 FR$=FR$(I)
860 WHILE PAL$(">" "NO" AND P=0
870 READ PAL$,N(I)
880 GOSUB 2000
890 WEND
900 RESTORE
910 IF P=0 THEN 720
920 IF HN THEN CA=N(0):GOTO 960
930 NEXT I
940 CA=N(0):CB=N(1)
950 IF CA=CB THEN PRINT#1," ESE MO
VIMIENTO NO LO PUEDES HACER P
ORQUE SON DEL MISMO COLOR":GOTO 290
960 PL$(0)="GRANDE":PL$(1)="PEQUENO
"
970 IF HN THEN I=0:GOTO 1010
980 FOR I=0 TO 1
990 FR$=FR$(I)
1000 ENC=0
1010 FOR J=0 TO 1
1020 PAL$=PL$(J)
1030 GOSUB 2000
1040 IF P<>0 THEN N(I)=J:ENC=-1
1050 NEXT J
1060 IF NOT ENC THEN 720
1070 IF HN THEN 1090
1080 NEXT I
1090 NA=2*CA-1+N(0)+AUX
1100 IF NOT HN THEN 1130 ELSE NB=0
1110 IF JI MOD 2<NA MOD 2 THEN PRIN
T #1," HAY NO CABE":GOTO 290
1120 GOTO 1200
1130 NB=2*CB-1+N(1)

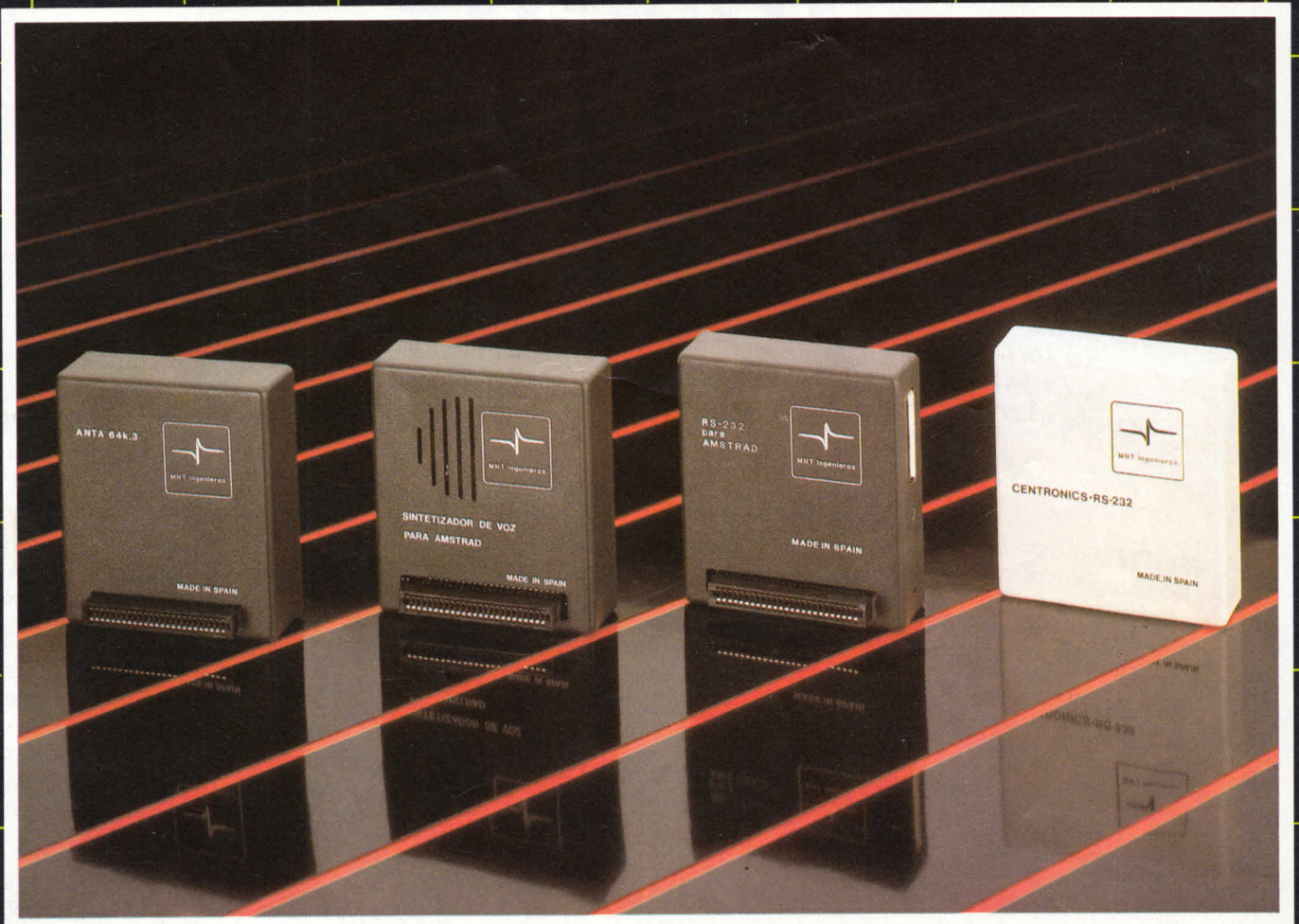
```



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

Periféricos
para AMSTRAD

CONECTAMOS CON TUS IDEAS.



PERIFERICOS PARA LOS MODELOS CPC 464, CPC 664 Y CPC 6128

- ANTA 64 K.3 Ampliación de memoria, buffer de impresora y Ram Disk.
- SINTETIZADOR DE VOZ

El programa que controla este sintetizador, contiene las reglas básicas de pronunciación en castellano y permite su funcionamiento, tanto en modo directo, como bajo el control de un programa.

- RS-232-C
Permite comunicar el ordenador con impresoras y plotters con entrada serie, modems, y otros ordenadores.

PERIFERICOS PARA LOS MODELOS PCW 8256 Y PCW 8512

CENTRONICS-RS 232

Proporciona al ordenador dos canales de comunicación:

- Canal paralelo (centronics) para el manejo de impresoras.
- Canal serie (RS-232) para comunicar con otros ordenadores, modems, plotters, etc.).

MHT ingenieros



Distribución: L.S.B., S.A. Sánchez Pacheco, 78 - 28002 Madrid.
Para mayor información, escribir a:
L.S.B., S.A. Sánchez Pacheco, 78 - 28002 Madrid
Nombre/Apellidos
Domicilio
Localidad
Provincia
C.P.
Profesión
Edad

Información sobre: Periféricos Amstrad otros

DICCIONARIO DE TERMINOS

La informática es una ciencia que, como todas las demás, emplea su propia terminología, accesible sólo a los iniciados. Los profanos que desean introducirse a fondo en las entretelas de dicha disciplina no deben verse frenados en su justo empeño durante más tiempo. Aquí está **AMSTRAD Semanal** para hacerles la vida un poco más fácil. Y si no lo creen, observen detenidamente nuestro diccionario de términos informáticos; el que después de leerlos no entienda cualquier texto o programa del ramo por abstruso que sea, es un azimut, ¿vale?

VERIFYFY: orden del dialecto Basic de la comunidad Gay «Pompiup usada para averiguar si su **Amstrad** graba más blanco que el del grupo feminista «stickdown».

COMPUTER: jueves tarde y sábados noche. Se admite Visa. Seriedad y discreción.

TAPE: orden perentoria que, en círculos conservadores, se aplicó intensivamente a las morbideces femeninas exitosamente durante mucho tiempo, pero con el advenimiento de los ordenadores la cosa acabó encinta..., de cassette (pila de lágrimas).

SKIP: el comando de Basic que limpia más blanco.

RANDOM: variante de acceso de ficheros, llamada también aleatoria, protagonista no ha mucho de un famoso film de Silvester Stallone.

RANDOM SPLASH: ficheros aleatorios para hombres curtidos.

MASK: dícese de la marca de chicle preferida por **Amstrad**.

FILTRO: instrucción de un programa que prevé que, si el ordenador se quema durante la ejecución del mismo, el humo que suelte lleve menos nicotina (ver **LIGHT**).

EDITOR: Maravillosa persona que publica revistas de informática, aceptando trabajos de los lectores, incluso pagándoles.

DIGITAL: proceso consistente en encontrar al astuto asesino por la huella del dedo meñique del pie izquierdo dejada en el interruptor de la luz.

DELETE: Dícese de algo que es propiedad de un extraterrestre.

DECIMAL: programa que siempre apuesta, tras horas de proceso, por el número de la ONCE que nunca toca.

SYNTAX: famoso lingüista griego, mudo todo él. Inventor del bien hablar en la informática (ver **ERROR**).

ERROR: para empezar, **SYNTAX** no era mudo. Para terminar, error es un mensaje u orden que da el **Amstrad** durante la ejecución de un programa, pensado para instar al usuario a pronunciar alguna que otra palabra malsonante, como «caquita».

ENSAMBLADOR: persona que junta en un todo coherente las partes de un ordenador. Gracias a ellos no nos dan una caja llena de piezas aisladas al comprar un **Amstrad**.

BYTE: es la bebida preferida por el ordenador.

DIAGRAMA DE FLUJO: dibujo explicativo de un proceso, natural o artificial, que algunas personas se ven obligadas a realizar periódicamente (ver *INCONFESABLE*).

DEBUG: en honor de Jacques Debug, el mayor idiota informático de la historia, autor del programa más largo del mundo capaz de escribir en pantalla «mi mamá me mima». El nombre enmarca un proceso destinado a la captura de errores en un programa, o, en su defecto, a la creación de otros nuevos mientras se eliminan los antiguos.

VARIABLE: parte de la memoria de un ordenador a la que se le asigna un valor que nunca coincide con el previsto, alterándose de forma misteriosa (ver *RANDOM*).

TERMINAL: configuración **Amstrad** que incorpora, en lugar de un monitor a color, una ventanilla expedidora de todo tipo de billetes impresos. Incluye, completamente gratis, funcionario «cutre», silla odiosa y cafetería con precios abusivos (más IVA).

SORT: rutina especializada que poseen la mayoría de los sistemas operativos que se precien, pensada para colocar los «slips» de los usuarios en orden alfabético.

ROM: barrilito de bebida espirituosa que los **Amstrad** llevan de fábrica. Dicen que es imprescindible para que la máquina funcione (ver *CUBA*).

PROCESADOR: juez militar chileno, apodado «Z80», programado para aplicar la máxima pena aleatoriamente (ver *CRIMINAL*).

RESTORE: del prefijo «RE», repetir, y «STORE», almacenar, es decir, guardar lo que ya está guardado. ¡Vaya estupidez! (Ver «*West Side Store*», preferiblemente un miércoles).

SILICIO: gobernador romano de California del Siglo I D.J.

SUBROUTINA: trabajo mecánico que se realiza a escondidas y con la desaprobación de todos (ver *HACIENDA*).

MONITOR: especie primate, pequeña y simpática, que habita encima de las mesas unida por un cable a la unidad central de un ordenador, en lugar de los árboles, como Dios manda. Originario de Sudáfrica, sólo existe en dos colores: verde y multicolor.

NETWORK: ciudad norteamericana en la cual todos sus habitantes trabajan juntos conectados por cables, compartiendo calles, coches y edificios por riguroso orden.

REGISTRO: subdivisión de un fichero armada de orden judicial, que normalmente acaba revolviendo los cajones (con perdón) de la morada de algún probo usuario.

CHAIN: país asiático especializado en el encadenamiento de programas, llamados ciudadanos por las malas lenguas.

FICHERO: palo inmaterial, compuesto de células de memoria y acabado en un gancho, que se utiliza en las faenas de pesca de unicornios.

Empleado provisto de tarjeta perforada por un reloj.

Bueno, pues aquí acaba, por ahora, el extracto de lo más florido de nuestro diccionario informático. Tiene algunas pequeñas deficiencias: está incompleto, no está ordenado alfabéticamente y hace referencias a palabras que no existen en el texto, pero por lo demás es una joya. En caso de duda, no lo dude: no nos llame, nosotros tampoco le llamaremos.



Ofites

Presenta: el universo del software, y

DELTA +

La más moderna base de datos DELTA, superándose a sí misma, "DELTA +", desarrollada para CP/M por COMPSOFT con todo en español.

Diseña sus propios ficheros; desde un simple fichero de nombres y direcciones hasta su propio sistema contable. El formato standar DIF permite intercambiar datos en DELTA, desde las hojas de cálculo CRACKER II, etc... y viceversa. Intercambio de datos con la mayoría de los tratamientos de texto como NEWWORD para MAILING.

Incluye un sencillo y funcional sistema de impresión de etiquetas con: hasta 5 columnas de etiquetas, 65 caracteres por etiquetas, 20 líneas con 3 campos cada una.

- PROGRAMABLE Y RELACIONAL.
- FICHEROS INDEXADOS.
- HASTA 90 CAMPOS ó 2.000 CARACTERES.
- MULTIPLES SISTEMAS DE BUSQUEDA, 8 CLAVES.
- FICHEROS DE HASTA 8 Mb.
- 8 GRUPOS DE TRANSACCION POR REGISTRO.

BASE
DE DATOS

17.850 pts.

NEWWORD

Programa de tratamiento de textos mejorando todo lo anterior. Manual y programa en español, que le enseñarán con facilidad y rapidez lo más avanzado en procesadores de textos. Compatibilidad funcional con WORDSTAR incluyendo muchas capacidades adicionales.

Tiene un potente MAIL-MERGE con opción de selección de destinatarios por criterios base de datos, creación de documentos, impresión de etiquetas. Utiliza todo el espacio de disco. Ensamblaje de textos, sustitución, etc., de la forma más fácil: autohace copias de seguridad. ¡NUNCA PERDERA UN TEXTO!

- Ñ, ACENTOS, DIERESIS, ETC...
- PRESENTACION EXACTA EN PANTALLA DEL FUTURO DOCUMENTO IMPRESO.
- INTERCAMBIOS DE FICHEROS CON CRACKER.
- VARIABLES SUSTITUIBLES EN IMPRESORA.
- POTENTE CALCULADORA.
- COMPROBADOR ORTOGRAFICO Y GRAN DICCIONARIO (45.000 TERMINOS AMPLIABLES).
- POSIBILIDAD DE LECTURA DE FICHEROS DE DELTA, CARD BOX, SUPERCALC, DBASE II, ETC...

TRATAMIENTO
DE TEXTOS

17.850 pts.

CRACKER II

El CRACK de las hojas de cálculo, la que deja detrás al resto. Funciones nunca vistas, formato de fechas, salvaguardia continua sobre un fichero. Realiza automáticamente copias de seguridad. Además de las tradicionales funciones, CRACKER II posee funciones lógicas, estadísticas y de alta matemática. Intercambia datos con NEWWORD, bases de datos y la mayoría de las hojas de cálculo.

- CELDAS PROGRAMABLES.
- FUNCIONES ESPECIALES: Fecha, días; desde y hasta la fecha de la semana, del año, lapso de tiempo, retraso, beep entrada, saludo usuario.
- SISTEMA DE AYUDA ON-LINE.
- SUMA CONDICIONAL.
- TOMAR DECISIONES EN LA HOJA.
- 18 MODOS GRAFICOS DISTINTOS.
- TRADICIONALES FUNCIONES MATEMATICAS Y AMPLIACION, FUNCIONES ESTADISTICAS Y LOGICAS.
- GENERA GRAFICOS EN BASE A LOS DATOS.

HOJA
DE CALCULO

17.850 pts.

EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA

IVA
no
incluido

TOTALMENTE
EN
ESPAÑOL

Informática

estas son sus estrellas.

NUCLEUS

NUCLEUS más que una estrella una constelación; tres ESTRELLAS en un SUPERPROGRAMA, la solución a cualquier aplicación por compleja que sea, NUCLEUS es GENERADOR DE PROGRAMAS, BASE DE DATOS Y GENERADOR DE INFORMES.

Toda la información es multi-intercambiable y de libre acceso por cualquiera de los demás programas. Así los datos de la base los condicionamos y utilizamos en el generador de programas y los imprimimos a través del generador de informes.

- GENERADOR DE PROGRAMAS EN MALLARD BASIC.
- CREACION DE BASES DE DATOS RELACIONALES.
- GENERADOR DE INFORMES.
- DISEÑADOR DE FORMATOS.
- DISEÑADOR DE PANTALLAS.
- CODIGO FUENTE DE LIBRE ACCESO Y LIBRE DE ERROR.
- DISEÑA SU PROPIO SISTEMA.
- MAILMERGE.

GENERADOR DE PROGRAMAS

26.780 pts.

BRAINSTORM

La revolución del pensamiento, BRAINSTORM es un programa que piensa con Vd.

El compañero ideal para el empresario, director o cualquier persona que tenga que planificarse o tomar decisiones.

BRAINSTORM es la ayuda necesaria para su organización. El programa que se ha standarizado en Inglaterra, tan necesario, útil y popular como una base de datos o un tratamiento de textos.

- ORGANIZA POR RANGOS.
- ACCESO DESCENDENTE POR-MENORIZADO.
- PLANIFICACION A NIVEL DIA.
- DECISIONES A LARGO PLAZO.
- REVISION DE PROBLEMAS.
- SIMULTANEIZACION DE TAREAS.
- PROCESO TOP/DOWN.

ORGANIZADOR DE IDEAS

17.850 pts.

STARCOM

Piii... su ordenador le comunica:

La revolución de las comunicaciones, de la mano de OFITES INFORMATICA, llega a España. El nuevo mundo de las comunicaciones digitales lo tiene a su disposición, las redes de transmisión electrónica digitalizada, con su PCW 8256 o PCW 8512 a través de un interface RS 232-C con otros ordenadores, redes de transmisión de datos, etc..., Vd. podrá enviar o recibir ficheros de texto o de datos, ASCII, etc..., creados por NEWWORD y otros...

- TRANSICIONES DIRECTAS EN RED.
- COMPATIBILIDAD CON NEWWORD.
- POSIBILIDADES DE TRANSMISIONES VIA MODEM, RED TELEFONICA.
- COMUNICACION INSTANTANEA.

COMUNICACIONES

17.850 pts.

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener los programas, puede dirigirse a:



Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

HISOFT-C

Daniel Palomo Ortega

Todos los usuarios de Amstrad estabamos deseando que apareciese en España un compilador de C para nuestro ordenador. Por fin vemos nuestros deseos cumplidos, con un magnifico compilador como es el de HISOFT. MICROHOBBY AMSTRAD ha probado este compilador, en versión inglesa, para daros una visión del funcionamiento del mismo.

El lenguaje C se creó en 1972 como una herramienta de programación. El creador del C es Dennis Ritchie, de los laboratorios Bell. Este lenguaje surgió cuando Ritchie trabajaba, junto con Ken Thompson, en el diseño del sistema operativo UNIX.

Este lenguaje no surgió por arte de magia, como es de suponer, sino que derivó del lenguaje B de Thompson, que a su vez... Pero eso es otra historia.

Fue creado como herramienta de programadores, por tanto es un lenguaje útil, ya que es potente, flexible y rápido.

El C es un lenguaje muy extendido, existen compiladores para multitud de sistemas aparte del UNIX, tan importantes como: Cray I, IBM, Sperry, Apple, Commodore, Amstrad, etc.

El C se utiliza en aplicaciones muy variadas, por ejemplo, el sistema operativo UNIX está escrito en C. Muchos compiladores, además de multitud de juegos, se han hecho con este lenguaje e incluso se utilizó en *El retorno del Jedi* para la animación de las secuencias de la película. Paquetes de software, programas de gestión y un largo etcétera han sido realizados en C.

El C es un lenguaje estructurado que produce programas compactos, eficientes, transportables, y de modificación muy sencilla.

¿Qué es lo que da esta potencia? C es un lenguaje moderno que trata las tareas por separado, es decir, es un lenguaje modular, todo esto se realiza gracias a la flexibilidad de sus expresiones. La filosofía de diseño de C se basa en el adecuado uso de las funciones, éstas son parecidas a las subrutinas, procedimientos y funciones de otros lenguajes, el C trabaja siempre con funciones como palabras clave, **PRINTF()**, **SCANF()**, **FSEEK()**, son funciones predefinidas, y nosotros podemos crear

nuestra biblioteca muy fácilmente e incluir funciones propias en las ya creadas. Por todo esto, muchas de las funciones a las que estamos acostumbrados no están implementadas en C.

La implementación de HISOFT es buena y además tiene el detalle de suministrar dos compiladores en el mismo disco, uno que funciona bajo AMSDOS y otro bajo CPM. Pasemos a analizarlos.

Crear un programa fuente

Lo primero que tenemos que hacer para crear un programa C es almacenarlo en un fichero de texto, esto lo podemos realizar con los editores que se suministran con cada compilador.

El perteneciente a **AMSDOS** es el típico de Hisoft con órdenes parecidas a las de anteriores productos de esta marca.

El de **CP/M** es, el ya conocido por sus usuarios, ED80 compatible con WORDSTAR.

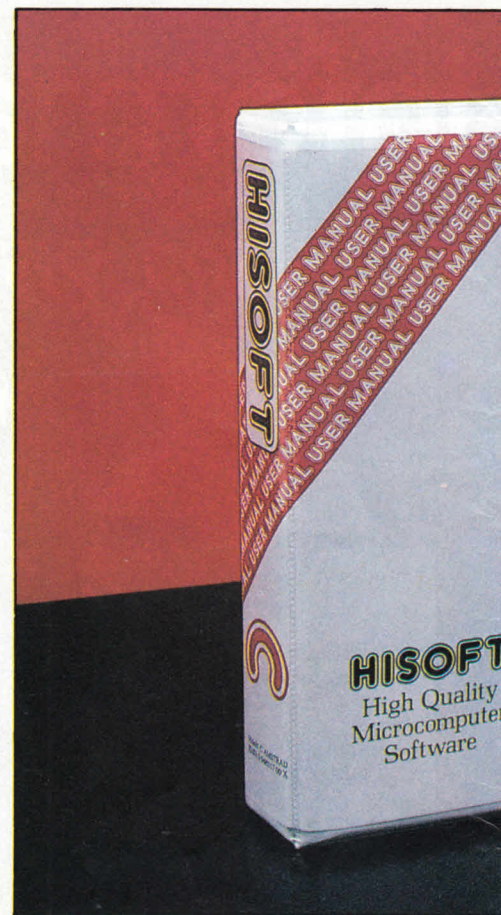
Podemos utilizar cualquier editor al que estemos más acostumbrados.

Veamos cómo funcionan.

El editor de líneas

El editor que va implantado en el compilador de **AMSDOS** es muy parecido al original de **Amstrad**, teniendo habilitado el cursor de copia y todas las demás funciones.

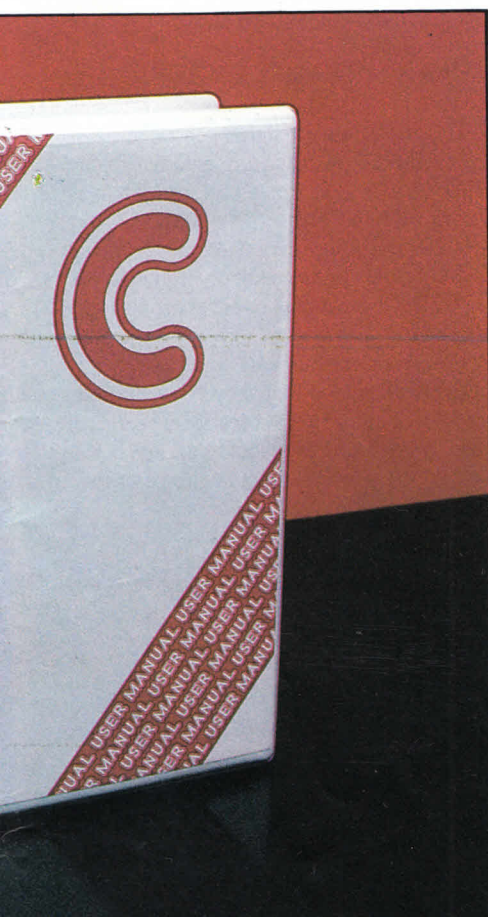
Los números de línea son una referencia para el programador y el compilador sólo los tiene en cuenta en el tratamiento de errores, para referenciarlos más rápidamente a la línea en cuestión. Si en este momento se pulsa la tecla [E] se editará la línea que produjo el error, cualquier otra tecla retorna al modo directo. Las llaves son imprescindibles en C, con ellas se indica el principio ({} y el final ()) de funciones, condiciones, bucles, etc. En el CPC6128 se pueden obtener con [CTRL] más los paréntesis.



COMANDOS DEL EDITOR

- In,i:** El editor imprime los números de línea siendo n el número inicial e i el incremento.
- Ln,m:** Lista un número indicado de líneas, siendo n la primera línea a listar y m la última.
- Wn,m:** Lista por impresora.
- S,,d:** Selecciona el delimitador a utilizar por los comandos del editor, muy útil cuando queremos buscar cadenas que contienen comas, d es el delimitador elegido, éste no puede ser un espacio.
- C:** Retorna al compilador. Si todo es correcto tecleando #INCLUDE comenzará a compilarse nuestro programa.
- Dn,m:** Borra los números de línea comprendidos entre n y m.

Los RSX de AMSDOS están todos a nu



EDITOR DE LINEAS

- Nn,i:** Renumerar el programa siendo n el primer número de línea e i el incremento.
- Fn,m,c,c':** Busca una cadena c en los números de línea comprendidos entre n y m, y la sustituye por la cadena c'.
- V:** Imprime en pantalla los valores actuales de n, m, c y c', así como el delimitador actual.
- Pn,m,nom:** Graba un programa en disco o cinta, n es la primera línea que se quiere grabar, m la última y nom el nombre del fichero con su extensión.
- Gn,m,nom:** Carga un programa de disco o cinta.
- En:** Edita la línea especificada por n.
- ... otra disposición.

El editor de CP/M + (ED80)

Este editor, compatible con WORDSTAR, es muy potente y no tiene nada que envidiar a los procesadores comerciales.

Todos los comandos se consiguen con [CTRL] y una o varias letras.

A continuación damos una lista de ellos clasificados por tareas:

Movimiento del cursor	
Ŝ Carácter izq.	Đ Carácter der.
H Carácter izq.(bor)	
Â Palabra izq.	F palabra der.
Ô S Tabula izq.	Ô D Tabula der.
Q S Principio línea	Q D Final línea
Ê Línea superior	Ë Línea inferior
Ô E Principio texto	Ô X final de texto
R Página posterior	Ĉ Página anterior
Borrado	
ÿ Borra línea	
[DEL] Borra último carácter	Ĝ Borra este carácter
Ô T Borra palbr. izq.	Ô Borra palbr. der.
Q [DEL] Borra principio línea	Q Y Borra final de línea
Ĥ B Marca princ. bloque	Ĥ K Marca fin bloque
Ĥ V Mueve bloque	Ĥ C Copia bloque
Ĥ Y Borra bloque	Ĥ R Lee bloque de disco
Ĥ W Graba bloque disco	
Movimiento rápido del cursor	
Ô G Ir a línea	
Q B Ir a princ. de bloque	Q K Ir a fin de bloque
Ĥ O Recuerda posición	Q O Ir a posición
Búsqueda y sustitución	
Q F Busca primero	Ĥ Busca siguiente
Ô L Sustituye y busca	Ô A Sustituye todo
Ĥ Abandona y sale	Ô Q Salir sin Backup
Ĥ X Salir con Backup	

Banco de PRUEBAS

Varios

Ĥ F Directorio	Ĥ J Borra fichero de disco
Ĥ P Control meta-key	Ĥ J Ayuda

Un editor muy completo, como se puede ver, y con posibilidades de adaptarlo a nuestro gusto, permitiéndonos remodelar la pantalla de acuerdo al monitor utilizado, ya sea de la serie CPC o PCW.

Un poco de... C

El C es el lenguaje que se maneja muy bien. En eso de los números el rango es el mismo que el de LOCOMOTIVE BASIC, pero hay algunas aclaraciones que hacer; los números octales se representan añadiendo un 0 a la izquierda si tiene significado, por lo tanto, no debemos utilizar esta notación, ya que sería entendido erróneamente por el compilador. Los números hexadecimales se representan con un cero seguido por una X y a continuación el número hexadecimal (OX4F3B).

Los caracteres simples pueden ser manejados encerrándolos entre comillas simples o dobles ('A'; "B").

Tenemos una serie de caracteres, llamados en C caracteres de escape, que son utilizados para imprimir retorno de carro, salto de página, comillas, etc. Estos se representan precedidos de la barra atrás (\) y son los siguientes:

- Es la animación d \ n nueva línea
- \ b espacio atrás
- \ f avance de página
- \ t tabulador horizontal
- \ r retorno de carro
- \ " comillas
- \ \ barra atrás
- \ " comillas simples

También podemos utilizarlos escribiendo su valor ASCII en octal precedido por la barra atrás, esto sirve para imprimir todos los caracteres gráficos de Amstrad por ejemplo \32 sería EOF; \377 imprimiría el último carácter del juego (las pesas), el valor a continuación de la barra atrás únicamente se puede poner en las bases disponibles.

Las cadenas se almacenan en una matriz conteniendo el valor ASCII de cada componente de la misma, cada carácter ocupa un byte.

Los nombres e identificadores pueden tener la longitud que deseamos, para hacer más claro lo que queremos representar con ellos, pero sólo tendrán significado para el compilador, los 8 primeros caracteres del nombre, se pueden utilizar todas las letras, números y el guión abajo, (A-Z, a-z, 0-9, _).

Por convenio se dejan las palabras mayúsculas para los nombres que utilizemos en los comandos del preprocesado, se escribe el programa, siempre que sea posible, en minúsculas.

Palabras reservadas

C es mucho menos extenso, en lo que a palabras reservadas se refiere, ya que al estar compuesto por funciones es más difícil, por no decir imposible, utilizar un nombre de una función como nombre de variable, ya que los paréntesis forman parte del nombre de la función y no son caracteres permitidos para un nombre de variable.

Pero sí tenemos algunas palabras con las que el compilador se haría un lío si las utilizáramos como nombre de variable.

Estas palabras son las siguientes:

PALABRAS RESERVADAS

Auto	break
char	continue
double	else
float	for
inline	int
return	short
struct	switch
unsigned	while
case	cast
default	do
entry	extern
goto	if
long	register
sizeof	static
typedef	union

Programas en C

Los programas en C se consiguen gracias a una colección de funciones creadas por nosotros, las implementamos o las de librería, que hacen que todo eso funcione. ¿Cómo consigue saber el compilador por donde ha de empezar a ejecutar el programa? Esto lo sabe porque nosotros hemos de llamar, obligatoriamente, Main () a la función principal, esta es la traducción de Main, de nuestro programa, así sabrá por dónde empezar, sencillo, ¿no?

A continuación del nombre de la función se pueden poner los parámetros de la misma, encerrada entre paréntesis vacíos a continuación del nombre. La función comienza en el punto

en que encuentra una llave de apertura (), todo lo que existe entre el nombre y esta llave es considerado por el compilador como declaración de variables. La sintaxis de una función es la siguiente:

```
Nombre de función (lista de
parámetros)
Declaración de variable

Asignación de variable
y llamadas a funciones
```

La llave de cierre finaliza la función. Los comentarios pueden ser incluidos entre barras y asteriscos (/* comentario*/)

Tablas de operadores

A continuación se da la tabla de operadores, ordenados, de acuerdo con su prioridad de mayor a menor:

Operador Prioridad					
()	16	Paréntesis	/	13	División.
func()	15	Llamada a una función.	%	13	Halla el resto de una división.
[]	15	Subíndice.	+	12	Suma.
.	15	Suministro del valor de un incremento de una estructura.	-	12	Resta.
→	15	Puntero a un miembro de una estructura.	>>	11	Operador de shift+ a la derecha.
*	14	Operador de indirección	<<		Operador de shift+ a la izquierda.
&	14	Dirección de var.	<		Operadores de relación.
-	14	Signo menos.	< 10		
!	14	NOT negación lógica.	<=		
++	14	Operador de incremento.	>=	9	Operador de igualdad.
--	14	Operador de decremento.	==	9	Operador de no igualdad.
sizeof	14	Suministra el número de bytes ocupados por una variable.	!=	9	Operador de no igualdad.
cast	14	Fuerza a que el tipo sea el especificado.	&	8	Y lógico bit a bit.
*	13	Multiplicación.	:	7	XOR lógico bit a bit.
				6	O lógico bit a bit.
			&&	5	Y lógico.
				4	O lógico.
			?:	3	Operador condicional.

Tipos de datos

En C hemos de declarar todas las variables que vayamos a utilizar en el programa, tal como ocurre en otros lenguajes compilados; para

ello tenemos algunos tipos ya implementados y éstos son:

CHAR para declarar variables y matrices en cadena

INT para declarar enteros con signo

LINSIGNED para declarar enteros sin signo

FLOAT para declarar enteros variables en punto flotante (reales)

SHORT para declarar enteros cortos (1 byte)

LONG para declarar enteros largos (2 bytes)

TYPED para definir nuestros propios tipos de datos

* para declarar punteros

Las matrices en C son muy parecidas a las de BASIC. Esto se hace con el identificador de tipo correspondiente, seguido del nombre de la variable y los subíndices entre corchetes.

CHAR nombre [10] [30]

Si no se usa ningún subíndice, se utiliza el espacio requerido por los elementos de la matriz.

Las estructuras es algo que posee C y que da gran flexibilidad a la hora de hacer programas, ya que nos permiten referirnos a cualquiera de los elementos de la estructura muy fácilmente, consiguiendo crear un base de datos, por ejemplo, sin problemas, ya que en la estructura se puede incluir cualquier tipo de datos.

Banco de PRUEBAS

Y la biblioteca en sí, que es donde se encuentran todas las funciones utilizables y algunas para el manejo interno.

Hay que agradecer a Hisoft que sea tan explicativo en estas bibliotecas, dando siempre la oportunidad de aprender algo más, ya que éstas se encuentran llenas de comentarios y nombres de variables con un significado muy claro, que hacen muy fácil su lectura y comprensión.

En las figuras 1 y 2 se encuentran las listas de funciones de las bibliotecas de AMSDOS y CP/M, stdio.h es la misma para los compiladores.



Los modos de almacenamiento determinan qué variable conoce cada función y hasta cuándo deben permanecer en memoria estas variables.

Los tipos de almacenamiento son:

AUTO: Es el modo asumido por defecto, estas variables tienen alcance local, es decir, sólo las conoce la función en la cual se declaran estas variables, desapareciendo al finalizar la misma.

EXTERN: Cuando una variable se declara fuera del cuerpo de una función se dice que es externa y estas variables pueden ser utilizadas por todas las funciones que declaren en su interior como «extern», incluso estando esta variable en otro módulo que queramos compilar junto con el que utiliza esa variable.

Si no se declara como «extern» se considera una variable nueva y asumirá el modo por defecto (**auto**).

STATIC: Es del mismo alcance que auto, local, pero permanece en memoria durante toda la ejecución del programa.

REGISTER: Estas variables son de modo automático, pero el compilador intentará que estén situadas en los registros de la CPU en vez de estar en la memoria RAM. Es más una síplica que una orden.

El preprocesador

El preprocesador es una parte del compilador C, se encarga de realizar algunas tareas «domésticas», tales como definir constantes y macros, incluir ficheros y librerías etc. Los comandos son los siguientes:

#define: Define una constante, una macro instrucción, o cualquiera de las palabras reservadas pudiéndose utilizar el nombre impuesto por nosotros en su lugar. Por ejemplo: #define EOF -1

#error: Este comando retira de memoria el texto de los códigos quedando como única diferencia los números de error. Esto es útil cuando tenemos que compilar programas muy largos ya que el texto de errores ocupa 2K

#data: Se sitúa al principio de la zona de datos, la dirección de memoria debe ir en hexadecimal, sólo utilizable en CPM.

#list: Activa(+) o desactiva(-) el listado del programa.

#direct: Activa o desactiva la ejecución directa de comandos, parecido a la ejecución directa en basic.

#include: Incluye el fichero escrito a continuación ya sea para compilar juntos los módulos o para incluir funciones de biblioteca. Tiene dos modos posibles:

1) #include «Nomfich»

#include nomfich

incluye todo el fichero compilador junto con el actual.

2) #include ?nomfich?

incluye sólo funciones utilizadas por el programa que da la orden **#include**.

#translate Graba en disco el código objeto con el nombre especificado por nosotros en CP/M sólo sirve para cambiar el nombre del fichero objeto.

Las bibliotecas de programa

El compilador de C incluye varias bibliotecas, con las que podremos cubrir casi todas nuestras necesidades. Estas bibliotecas contienen todas las funciones típicas de C, las rutinas de manejo del disco, etc. Tenemos dos bibliotecas con casi todas las palabras reservadas de Basic pero al estilo C, con ellas se pueden aprovechar todas las prestaciones del AMTRAD tanto gráficas como de sonido e interrupciones.

En CP/M encontramos todas las funciones para manejar a nuestro antojo, tanto el disco como pantalla o teclado. Los usuarios de CP/M+ podrán disfrutar de los gráficos, ya que incluye una biblioteca para la extensión gráfica GSX.

Las bibliotecas se dividen en dos partes:

El encabezamiento que es donde se definen todas las constantes que se utilizan en la correspondiente biblioteca.

STUDIO.LIB

```
int abs(n)
int sgn(n)
int poek (address)
void poke (address,value)
in out(data, port)
int inp(port)
iont atoi(s)
void qsort(list, nu-items, size, cmp-func)
char *strcat(base, add)
char *strncat(base,add,number)
int strcmp(s,t)
int strncmp(s1, s2, n)
char *strcpy(s1, s2, n)
char *strncpy(dest, source)
unsigned strlen(s)
char *strchr(string, ch)
int strstr(s1, s2)
int strcspn(s1, s2)
char *strchr(s, c)
char *strrchr(s, c)
int ispunct(c)
int isalnum(c)
int isxdigit(c)
int isascii(c)
int iscntrl(c)
int isprint(c)
int isgraph(c)
int toascii(c)
char *fgets(s, n, fp)
char *gets(s)
void fputs(s, fp)
char *calloc(n, size)
void free(block)
char *sbrk(n)
exit(n)
- exit(n)
void srand(n)
void long-multiply(c, a, b)
void long-init(a, n1, no)
void long-set(a, n, d)
void long-copy(c, a)
```

BASIC1.LIB SONIDOS

```

setup-sound()
play(control-string,status)
S-arm-event(channel-bit,seb-add)
S-queue(sp)
sound-check(chanbit)
S-release(channel-bits)
S-ampl-envelope(number, envelope)
S-tone-envelope(number, envelope)
S-hold()
S-CONTINUE()
after(delay-in-ticks, control-block,
function-name)
every(period-in-ticks, control-block,
function-name)
add-ticker (ctrl-block, initial-time-delay,
recharge-delai, function-name)
init-event (event-block, function-name)
border(colour)
cass-speed (speed)
catalog()
cls()
event-enable()
event-disable()
flash-speed(time1, time2)
ink(ink-to-setup, colour1, colour2)
int inkey(key-number)
char *instr(main-string, sub-string)
itob(n, string, precision)
joy(joystick-number)
int key-function(traslated-key-number,
expansion-string)
K-arm-breaks(event-routine, ROM-select)
K-disarm-break()
read-file(filename, adress)
char *strlower(string)
char *strupper(string)
time(array)

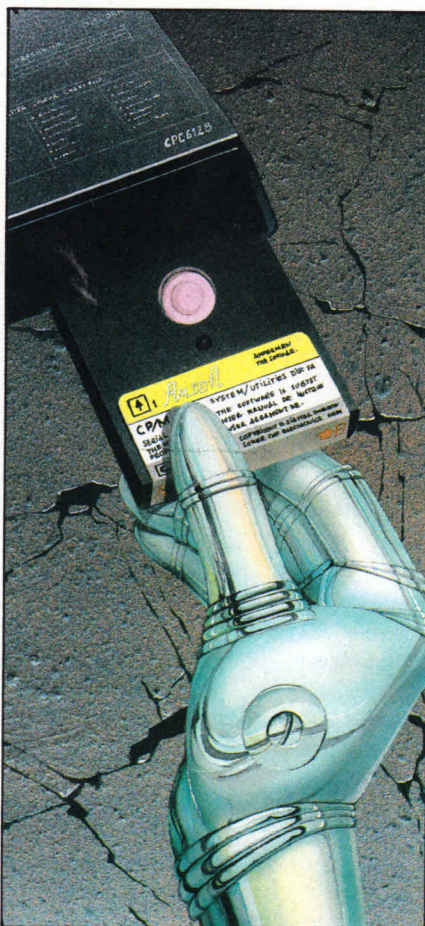
```

BASIC2LIB

```

draw(control-string)
T-set-graphic(on)
T-swap-streams(stream-number,
another-stream-number)
T-get-cursor(px-column, py-row,
p-roll-count)
G-ask-cursor(pdx, pdy)
G-set-origin(x,y)
G-win-width(x1,x2)
G-win-height(y1,y2)
G-clear-window()
G-set-pen(ink)
G-set-paper(ink)
G-wr-char(c)
G-move-absolute(x,y)
G-move-relative(dx, dy)
G-plot-absolute(x, y)
int G-test-absolute(x, y)
int G-test-relative(dx, dy)
G-lie-absolute(x, y)
G-line-relative(dx, dy)

```



CPM.LIB

```

void cpm-cmd-line(aargc, aargv, buffer)
int parse-args(s, argv, dest)
cpm-dir(drive, user, afn, sp, width)
void fcb-to-name(filename, fcb)
cpm3-bios(func, a-param, bc-param,
de-param, hl-param)
int cpm-punout(c)
int cpm-version()
int unlink(filename)
rename(oldname, newname)
int cpm-user(new-user)
void cpm-bdos(func, param)
void fseek(stream, offset, mode)
read-file(filename, address)
char *strupper(string)
char *strlower(string)
write-file(filename, address, length)

```

GSX.LIB gráficos de CP/M

```

v-clswk(handle)
v-updwk(handle)
v-pline(handle,count,pxyarray)
v-pmarker(handle,count, pxyarray)
v-gtext(handle,x,y,string)
v-bar(handle,pxyarray)
vst-height(handle,height,char-width,
char-height,cell-width,cell-height)
int vst-rotation(handle,angle)

```

```

int vsi-type(handle,style)
int vsm-type(handle,symbol)
int vsf-color(handle, color-index)
int vsm-locator(handle,x,y,xout,yout,term)
vsin-mode(handle,dev-type,mode)
v-exit-cur(handle)
v-enter-cur(handle)
v-currigh(handle)
v-curhome(handle)
v-eeol(handle)
v-rvon(handle)
v-dspcur(handle,x,y)
v-rmcur(handle)
gsx()

```

Conclusiones

Este es un compilador fácil de usar en sus dos versiones, trae en sus bibliotecas multitud de funciones con las que podremos hacer casi todo, lo que no podamos hacer sólo tenemos que implementarlo.

Se probó su rapidez con dos programas, 1 y 2, demostrando escasa diferencia entre el compilador de AMSDOS y el de CP/M. Las pruebas efectuadas dieron los siguientes resultados:

Compilación	3 segundos
Bucle vacío	2 segundos en 1.000 vueltas
Bucle simple	6 segundos en 500 vueltas
Bucle complejo	49 segundos en 500 vueltas
Bucle texto	18 segundos en 100 vueltas

Podéis probar a ejecutar los mismos bucles en basic, midiendo los tiempos tardados, y haciendo las oportunas comparaciones. Pero por si era poco miramos también cómo maneja C los ficheros. El programa 2 leía un fichero de texto, concretamente stdio.h, con más de 2.000 caracteres en tan sólo 4 seg., el procesador de textos con que se ha escrito este artículo, tarda el mismo tiempo en realizar esa operación, a pesar de estar escrito en puro ensamblador.

Los manuales son escuetos pero bastante completos, tratando cada punto lo justo, sin pasarse en ningún sentido. La única pega encontrada fue la de estar escritos en inglés, a ver si tenemos pronto la versión española.

EQUIPO: CPC 464-664-6128 PCW
8256-8512
SISTEMA OPERATIVO: AMSDOS Y CP/M
DISTRIBUIDOR:
OFITES INFORMÁTICA
PRECIO: 15.000 PTAS.

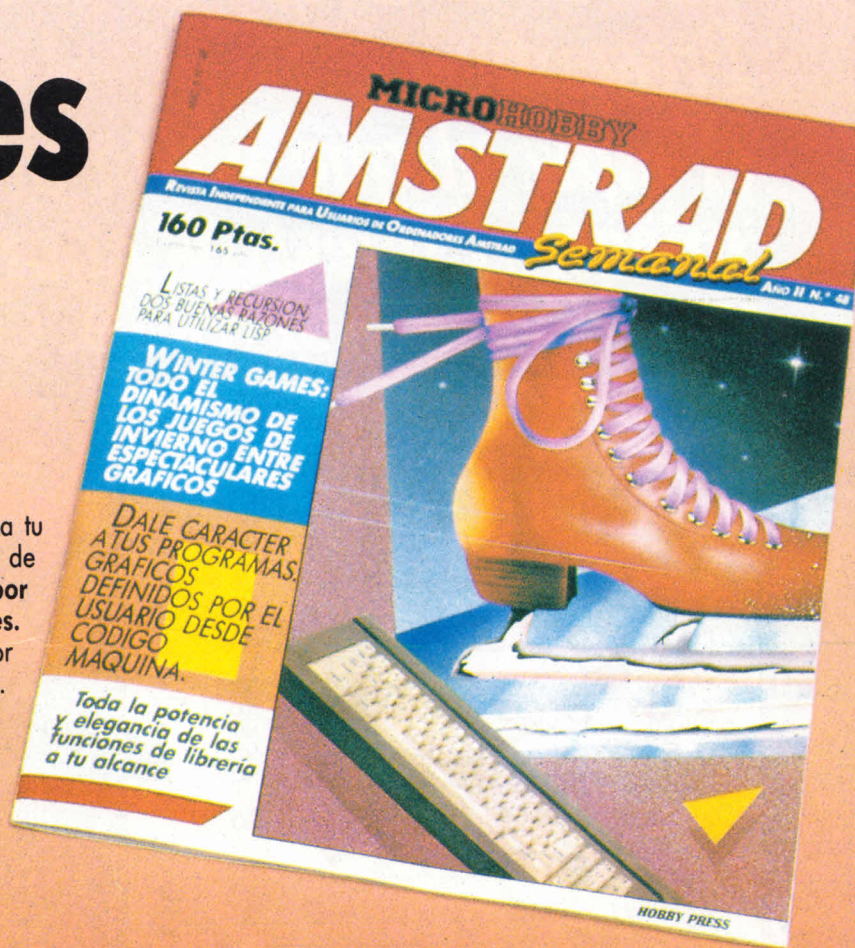
OFERTA ESPECIAL I ANIVERSARIO

6 meses Gratis de AMSTRAD CASSETTE

Suscríbete ahora a Microhobby Amstrad, o realiza tu renovación, y recibirás, totalmente gratis, un regalo de excepción: una suscripción a Amstrad Cassette por seis meses.

Cada cinta contiene los programas publicados por Microhobby Amstrad durante un mes.

Todos los programas de nuestras cintas se encuentran desprotegidos, con el objeto de facilitar su copia en disco y la revisión de los listados.



En cada cinta encontrarás:

- Apasionantes juegos llenos de acción y dinamismo.
- Utilidades con las que sacar mayor partido a tu ordenador.
- Rutinas en código máquina, para que las utilices en tus propios programas.
- Y pequeños trucos de programación, para que, poco a poco, te conviertas en un experto.

Recorta o copia el cupón que aparece cosido en las páginas de esta revista.
APROVECHA ESTA OFERTA ÚNICA
válida sólo para España
hasta el 31 de
noviembre de
1986.

Ofites Informática

Presenta:

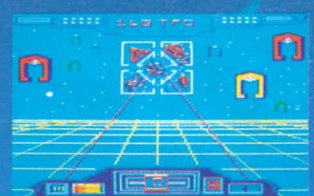
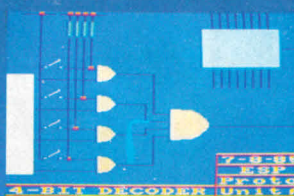
el lápiz al que gusta decir **SI**
mientras nuestros competidores dicen no
UNICO PARA AMSTRAD, CON PRECISION PIXEL

FUNCIONES	ESP	dt'ronics	OTROS
UNICO MENU DE PANTALLA	SI	NO	
ARRASTRE OBJETOS PANTALLA	SI	NO	
TRASLADO OBJETOS PANTALLA	SI	NO	
TRASLADO DE CURSOR	SI	NO	
CAJAS ELASTICAS	SI	SI	
LINEA ELASTICA	SI	SI	
TRIANGULO ELASTICO	SI	NO	
ELIPSE ELASTICO	SI	NO	
DIAMANTE ELASTICO	SI	NO	
POLIGONO ELASTICO	SI	NO	
HEXAGONO ELASTICO	SI	NO	
OCTOGONO ELASTICO	SI	NO	
CUBO ELASTICO	SI	NO	
PIRAMIDE ELASTICA	SI	NO	
CIRCUNFERENCIAS	SI	SI	
CIRCULOS RELLENOS	SI	NO	
CAJAS RELLENAS	SI	NO	
ELIPSES RELLENAS	SI	NO	
CUNAS	SI	NO	
SIMULADOR DE CORTES	SI	NO	
DISEÑO DE ZOOM	SI	SI	
IMAGEN ESPEJO E INVERTIDA	SI	NO	
FONDO DE REFERENCIA	SI	NO	
REJILLA DE FONDO	SI	NO	
OPCION DISPLAY X, Y	SI	NO	
RELLENADO CON COLOR	SI	SI	
LAVADO DE COLOR	SI	NO	
VOLCADO PANTALLA RESIDENTE	SI	NO	
DIBUJO DE BORDES EN 3 D	SI	NO	
TEXTO	SI	SI	
9 TAMAÑOS DE BROCHA	SI	NO	
18 TOBERAS MOSTRADORAS	SI	NO	
4 MEZCLAS BASICAS	SI	NO	
VARIADOR DE MEZCLAS	SI	NO	
SOMBREADO DE MEZCLAS XOR	SI	NO	
FICHERO ICONOS RESIDENTES	SI	NO	
FICHERO RELLENOS RESIDENTES	SI	NO	
26 COLORES DE PAPEL	SI	NO	
PALETA DE 15 TONOS DE COLOR	SI	NO	
POSICIONAMIENTO DE PUNTO	SI	SI	
RAYOS DESDE UN PUNTO FIJO	SI	NO	
DIBUJO REFLEJADO (ESPEJO)	SI	NO	
FUNCION HOME	SI	NO	
CONTROL DESDE TECLADO	SI	SI	
CONTROL CON JOYSTICK	SI	NO	
DISPONIBLES MODOS 1 Y 2	SI	?	

Compare con otros lápices



ESTOS SON
ALGUNOS EJEMPLOS
DE LOS GRAFICOS QUE VD.
PODRA REALIZAR CON NUESTRO
LAPIZ OPTICO



DE VENTA EN LOS MEJORES COMERCIOS
DE INFORMÁTICA

Si Vd. tiene alguna dificultad para obtener el lápiz óptico,
puede dirigirse a:



Avda. Isabel II, 16 -8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

DISPONIBLE PARA:
CPC 464 CASSETTE 4.900 Ptas.
CPC 464-664 DISCO 6.900 Ptas.
CPC 6128 DISCO 6.900 Ptas.
(IVA no incluido)
CONDICIONES ESPECIALES PARA DISTRIBUIDORES

Version
SPECTRUM,
AMSTRAD Y COMMODORE

Alistate a **Juegos & ESTRATEGIA**

LA BATALLA DE INGLATERRA ha comenzado



Todas las unidades de la RAF y la Luftwaffe —tu ordenador— intentará neutralizarlas. El destino del mundo libre depende de ti.

Oferta especial hasta el 31 de noviembre: PIDE TRES NUMEROS Y PAGA SOLO DOS.

ENVIE HOY MISMO ESTE CUPON AL APARTADO 232 DE ALCOBENDAS (Madrid)

- Deseo recibir en mi domicilio tres ejemplares de **Juegos & Estrategia** al precio especial de **2.255 ptas.**, lo que me supone adquirir tres y pagar sólo dos. Marco los tres ejemplares que deseo con una cruz.
- Deseo recibir un solo ejemplar de **Juegos & Estrategia** al precio de 1.125 ptas. Marco con una cruz el ejemplar que deseo recibir.

Spectrum

- N.º 1 Arnhem
N.º 2 Ratas del Desierto
N.º 3 OTAN Alerta
War Zone

Especial 1 Elecciones Generales

N.º 4 Su mejor hora (La batalla de Inglaterra)

Amstrad

- Arnhem
 Ratas del Desierto
 Teatro de Europa
War Zone

La batalla de Inglaterra

Commodore

Teatro de Europa

La batalla de Inglaterra

NOMBRE

Fecha de nacimiento

DIRECCION

LOCALIDAD

PROVINCIA

C. POSTAL

TELEFONO

PROFESION

Forma de pago:

Talón bancario a nombre de Hobby Press, S.A. Giro Postal a nombre de Hobby Press, S.A., n.º de giro

Tarjeta de crédito: Visa n.º

Master Charge n.º

American Express n.º

Fecha de caducidad de la tarjeta

Fecha y firma

SINCLAIR STORE

EL CENTRO DE LAS NOVEDADES



SPECTRUM 128 K+2



INVES PC 640 X

INVES 100 HF

Venga a Sinclair Store.

Los primeros en tener lo último.

Le presentamos las más recientes
PC totalmente compatibles por menos de
Convertidor TV para tu Amstrad, hasta las cadenas de sonido con un precio inferior a 30.000
ptas., que van a revolucionar el mercado. **¡VA A SER UN ESCANDALO!**

novedades. Desde los ordenadores
90.000 ptas., lo último en Spectrum.

OFERTAS

Convertidor TV Amstrad
Ampliación memoria Amstrad 464, 64 K
Ampliación memoria Amstrad 464, 256 K
Disco de silicio 256 K
Lápiz óptico Amstrad
Sintetizador de voz
Fundas teclado, desde
Opus Discovery
Software Amstrad, Commodore, desde
Joystick Quick Shot II + Interface Kempston

Pesetas
Lanzamiento
8.500
21.500
20.600
5.600
9.450
800
44.000
500
3.000

ABRIMOS SABADOS TARDE

sinclair store

SOMOS PROFESIONALES

BRAVO MURILLO, 2
(Glorieta de Quevedo)
Tel. 446 62 31 - 28015 MADRID
Aparcamiento GRATUITO Magallanes, 1

DIEGO DE LEON, 25
(Esq. Núñez de Balboa)
Tel. 261 88 01 - 28006 MADRID
Aparcamiento GRATUITO Núñez de Balboa, 114

AV. FELIPE II, 12
(Metro Goya)
Tel. 431 32 33 - 28009 MADRID
Aparcamiento GRATUITO Av. Felipe II