AÑO II N.º 27

MICROHOBBY AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO II N.º 27

160 Ptas.

Canarias **165** pts.

NUEVO ORDENADOR DE AMSTRAD CON 512 K DE RAM: PCW8512

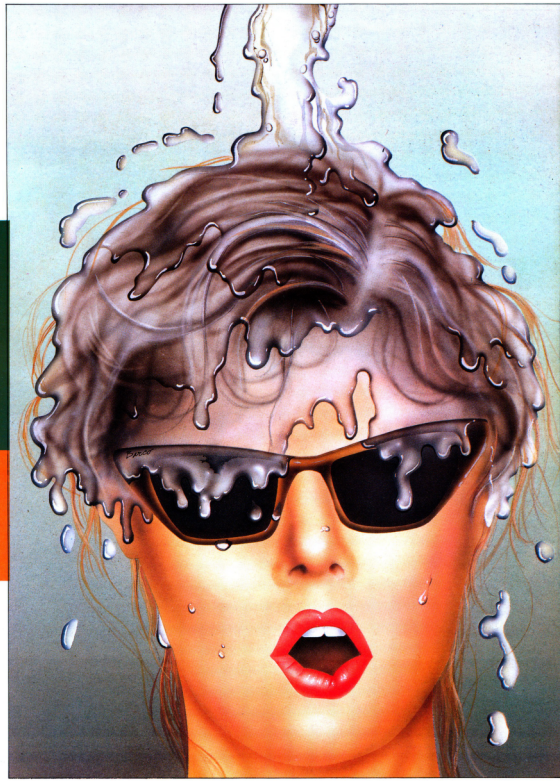
COPIA POR IMPRESORA DE LA PANTALLA DE TU ORDENADOR

DEFIENDE LA BASE ALFA DE LOS ATAQUES ALIENIGENAS

PROGRAMACION CON SUBRUTINAS

SOFTWARE

***Tornado* LOW LEVEL, Acción supersónica en 3D**



HOBBY PRESS

En COMPUTIQUE

el AMSTRAD PCW 8256 es

¡Aún cuás increible!

TODO SON VENTAJAS

Conseguir tu ordenador AMSTRAD PCW 8256 en COMPUTIQUE es jugar con ventaja. Gracias a la amplia experiencia profesional de COMPUTIQUE, te será más fácil obtener un mayor rendimiento y prestaciones del PCW 8256 ¡y en un tiempo récord!

Y es que COMPUTIQUE te obsequia con un CURSO GRATUITO DE MANEJO DEL PROCESADOR DE TEXTOS, HOJA DE CALCULO, BASE DE DATOS para que, en pocos días, manejes tu ordenador AMSTRAD como un experto.

TODO UN SISTEMA INFORMATICO

No olvides que el AMSTRAD PCW 8256 es, en sí, un auténtico Sistema Informático puesto a tu servicio, ya que incluye en su precio (129.900 + I.V.A.): Teclado profesional. Unidad central. Pantalla de alta resolución. Unidad de disco. Impresora. Programas en discos:

- *Mallard BASIC con sistema JETSAM para ficheros indexados.*
- *Sistema Operativo CP/M Plus.*
- *Procesador de textos LOCOSCRIPT.*
- *Lenguaje Dr. LOGO.*
- *Diversas utilidades. Completa documentación y manuales en castellano.*

Nada pierdes con hacernos una visita. Te haremos una demostración sin compromiso. De paso nos cuentas tus necesidades informáticas y cambiaremos impresiones sobre un tema que a ambos nos gusta: LOS ORDENADORES. ¡Te estamos esperando!

Te da cuás

COMPUTIQUE

Servimos a tiendas.
Abrimos sábados por la tarde.

Embajadores, 90. Tel. 227 09 80
28012 Madrid

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

Edita
HOBBY PRESS S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

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

Secretaria de Dirección
Marisa Cogorro

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

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

Dto. Circulación
Carlos Peropadre

Distribución
Coedis, S. A. Valencia, 245
Barcelona

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

MICROHOBBY

AMSTRAD

sumario

Año II • Número 27 • 4 al 10 de Marzo de 1986
160 ptas. (incluido I.V.A.)
Canarias, 155 ptas. + 10 ptas. sobretasa aérea
Ceuta y Melilla, 155 ptas.

Si eres lector habitual de esta revista

¡Llámame! de 7 a 10 de la tarde

Te estoy esperando.
Tengo muchas cosas que contarte... y muy interesantes. De momento, te propongo la posibilidad de *AHORRAR 1.600* ptas. y, además, con un poco de suerte, *GANAR UNA VESPINO* ¿Qué te parece? Pues esto es sólo un avance. Cuando me llames te contaré más cosas que seguro te gustarán.
Pero no te demores, porque a una mujer nunca se le hace esperar. Tienes de plazo hasta el 31 de marzo. Después, habrás perdido tu oportunidad.

HOBBY PRESS, S.A.

(91) 654 32 11

Primera PLANA

NACE EL PCW8512

Si nuestros lectores al leer el titular de la noticia se temen lo peor, aciertan plenamente. **Amstrad** va a hacerlo de nuevo.

El lanzamiento de su quinto ordenador es una realidad.

La criatura se llamará **Amstrad** PCW8512, con las mismas especificaciones que el 8256 pero con 512 Kbytes de RAM.

¡Qué nadie se asuste! Todo el software existente hasta ahora para el 8256 correrá perfectamente en la nueva máquina; no habrá problemas de compatibilidad.

Se espera que **Amstrad** no retire de su gama de modelos el 8256 en favor del 8512, sino que continuará comercializando ambos, aunque probablemente habrá un movimiento gradual hacia la versión con más memoria.

Se acostumbra en estos casos a rebajar el precio de la primera versión, aunque esta posibilidad se presenta difícil debido a que el 8256 se vende muy bien *(claro que en España, Indescomp ya ha bajado el precio del 8256, muy a tiempo)*.

En Inglaterra, el PCW8512 costará 100 libras más que el que ya podemos llamar su hermano menor.

No obstante, ahí no acaba la historia de las sorpresas que Sugar nos tiene reservadas.

Desde hace algún tiempo, el rumor de que pronto existiría un compatible IBM PC ha venido sonando con insistencia en el mundo informático español.

Según nuestro corresponsal en Inglaterra, **Amstrad** tendrá un ordenador de 16 bits compatible IBM PC para finales del verano de este año. De momento, no existe confirmación acerca de precios y configuraciones, aunque se especula en torno a las 200.000 ptas., y a que la máquina llevará el sistema operativo GEM de Digital Research, lo que presupone un mínimo de 512 K de RAM en su configuración base.

NUEVO ORDENADOR DE APPLE

Apple Computer va a presentar próximamente en USA su nuevo modelo de ordenador, el MacIntosh Plus.

Como su nombre indica, la nueva máquina es un Mac pero con una serie de importantes modificaciones: se le ha dotado de un Megabyte de RAM en configuración base y la ROM se extiende hasta 128 Kbytes.

En ésta, además del sistema de iconos que ha hecho famoso al Mac, se ha incluido un potente software para el manejo jerarquizado de ficheros, herramienta imprescindible para optimizar al máximo el proceso de datos.

También existe una puerta de comunicaciones tipo serie, pero que permite transmitir datos 6 veces más rápido que las convencionales.

Apple ha dotado al teclado de un **«keypad»** numérico y teclas del movimiento del cursor, y dice que la memoria se puede ampliar a 4 Megabytes.

Todo el software del MacPlus y sus manuales se encontrarán traducidos al castellano desde el momento que el primer equipo se venda en las tiendas.

SORTEO ESPECIAL "YOUR COMPUTER"

En el sorteo especial de **«Your Computer»** celebrado el día 5 de febrero entre todos los usuarios que nos mandaron sus datos, resultó agraciado con un **Amstrad** 128 K:

JOSE MANUEL GARCIA LAGOS
PINTO (Madrid)

Nuestra más sincera enhorabuena a este estudiante de 14 años, cuyo principal **«hobby»** es pasarse todo el tiempo que puede delante de su ordenador.

LAS SUBRUTINAS: EN ORDEN

La semana pasada vimos alguna de las aplicaciones que pueden tener nuestro nuevo descubrimiento: las subrutinas. Pero no era todo. Todavía nos queda mucho camino por recorrer para tener un «dominio absoluto» sobre ellas. Así que: ¡manos a la obra!

Se acuerda de nuestra pequeña calculadora del artículo anterior? Nos calculaba sencillamente cada una de las cuatro operaciones matemáticas básicas con dos datos que nosotros le dábamos al principio del proceso. Por si le falla la memoria le aconsejamos que eche una ojeada al programa 1 y verá que le resulta conocido.

Programa uno

En el programa principal —*o cuerpo principal*— existen cuatro llamadas a diferentes subrutinas una por cada operación. Están en las líneas 50, 60, 70 y 80.

Analicemos una de ellas. Si encuentra:

```
50 GOSUB 1000
```

¿Qué hará su Amstrad? Bueno, él lo tiene muy claro. Saltará a ejecutar la subrutina que empieza en la línea 1000 y una vez que la complete —*encuentre RETURN*— volverá al programa principal.

Pero, **¿y usted?** ¿Lo tiene tan claro como el ordenador? Los humanos no tenemos una mente tan **«numérica»** como su **Amstrad** y por tanto ante una instrucción tal como:

```
50 GOSUB 1000
```

solamente sabemos que el programa saltará a una rutina que hay en la línea 1000, pero nada más.

Aquí surge un problema. A menos que estudiemos y sigamos la subrutina paso a paso, no tendremos ni la más ligera idea de lo que hace visto desde el programa principal.

Por **«enésima»** vez vamos a decirle algo sobre la claridad. ¡No nos llame pesados! Recuerde que su programa no sólo ha de funcionar sino que además es muy conveniente que **«todos»**, incluso los demás, podamos entenderlo y seguirlo.

Y según esto la línea anterior:

```
50 GOSUB 1000
```

poca claridad nos aporta sobre qué es lo que hace la subrutina.

¿Qué le parece si en vez de llamar a una subrutina solamente con el número de línea le añadimos un comentario que nos indique lo que queremos hacer?

Sería cosa de cambiar la línea 50 por:

```
50 GOSUB 1000: REM SALTO A SUMA
```

Seguro que de esta manera se entiende bastante mejor qué operación vamos a realizar cuando el programa principal llega a esta línea: saltar a una subrutina que hace una suma. Por supuesto, para que este método cumpla su objetivo, es necesario que el comentario sea un buen indicativo de lo que va a ocurrir cuando se ejecute la subrutina. De no hacerlo así, sólo añadiríamos confusión a nuestro listado no, sería recomendable poner como un comentario **«multiplicación»** cuando hagamos un salto a una rutina que nos calcule una suma.

Lógicamente también podríamos cambiar de esta forma todas las demás llamadas a subrutinas. Corregiríamos las líneas 60, 70 y 80 por:

```
60 GOSUB 2000: REM RESTA
70 GOSUB 3000: REM MULTIPLICACION
80 GOSUB 4000: REM DIVISION
```

y conseguiremos tener un programa bastante majete, sencillo y simple pero sumamente claro.

El ejemplo que estamos utilizando es muy facilito y tiene pocas llamadas a subrutinas, pero **¿qué ocurrirá si nos encontramos dentro de un programa muy largo que está plagado de cientos y cientos de llamadas?** Puede parecerles una exageración pero les aseguramos que encontrar cada una de las llamadas puede llegar a ser una tarea de chinos. Y al final nos olvidaremos alguna.

Sin embargo, siguiendo el método que les hemos explicado, dentro del programa principal podríamos ir viendo a grandes rasgos qué es lo que hace cada una de ellas y por tanto tendremos una visión general de la forma como trabaja el conjunto. **¿Está un poco más claro?**

Pasemos de página. Las subrutinas a las que llamamos pueden ser de dos tipos:

— Unas realizan una determinada misión siempre con los mismos datos y valores de las variables. Son las subrutinas sin parámetros.

— Otras realizan la misma función o los mismos cálculos pero cada vez utilizan para ello valores diferentes de las variables. Se trata de las rutinas con parámetros.

Las primeras son subrutinas que siempre hacen lo mismo, realizan la misma función o los mismos cálculos independientemente de los valores de las variables que manejan. El programa 2 es un buen ejemplo de ellas.

Programa dos

Su misión es construir un cuadrado de asteriscos en el centro de la pantalla. Para ello utiliza dos subrutinas, una para escribir los lados horizontales —*líneas 1000 a 1040*— y otra para los verticales 2000 a 2040.

Como puede observar, el cuerpo del programa —*o programa principal*— es el que está comprendido entre las líneas 10 y 80. Desde aquí es donde se ordenan todas las operaciones que se van a efectuar a lo largo del programa y donde están colocadas las llamadas a las subrutinas.

Advierta la forma, cómo hacemos las llamadas. En eso hemos quedado, **¿no?**

Tanto la rutina que hemos bautizado como «horizontal» como la de siempre **«vertical»** siempre hacen lo mismo: líneas del mismo número de asteriscos bien sea en horizontal o en vertical. Son independientes del resto del programa así como de los valores de las variables y datos: no necesitan parámetros.

Vamos a intentar hacer un pequeño cambio en la estructura de este programa. En vez de dibujar siempre el mismo cuadrado, sería muy interesante poder construirle con lados de tamaños variables. Con esto nos estamos metiendo dentro de las subrutinas con parámetros.

En esta ocasión las rutinas **«horizontal»** y **«vertical»** no nos van a pintar un número fijo de asteriscos sino que, dependiendo de un valor que les pasemos, los lados van a ser más cortos o más largos.

Teclee con paciencia el Programa 3, en el que utilizamos subrutinas con parámetros, y a continuación pasaremos a su análisis.

Programa tres

Para determinar la longitud de los lados del cuadrado vamos a usar la variable **«parámetro»** cuyo contenido es el que van a utilizar las dos subrutinas para realizar una misma tarea —*dibujar una línea de asteriscos*— pero con distinto número de ellos cada vez.

Primeros pasos

El único requisito que necesita este programa para funcionar correctamente es que antes de llamar a las subrutinas, «parámetro» debe contener la longitud que queremos que tengan los lados del cuadrado.

Para ello utilizamos simplemente la instrucción INPUT de la línea 30.

El resto del programa es semejante al anterior. Las únicas diferencias son:

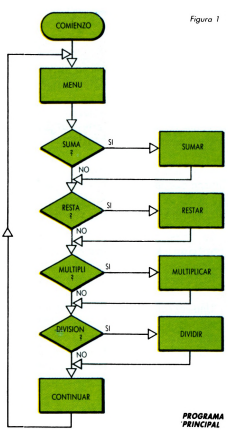
— La inclusión de la línea 30 que nos pide el valor del parámetro utilizado.

— El uso del contenido de la variable «parámetro» —o número de asteriscos— como límite superior de los bucles que componen las subrutinas.

Por lo demás, el funcionamiento de ambos es semejante. Dese cuenta de que por el hecho de pasar, o enviar, un parámetro a **«horizontal»** y **«vertical»**, los resultados obtenidos son bastante diferentes: el Programa 3 es mucho más flexible que el 2. ¡Maravilloso!

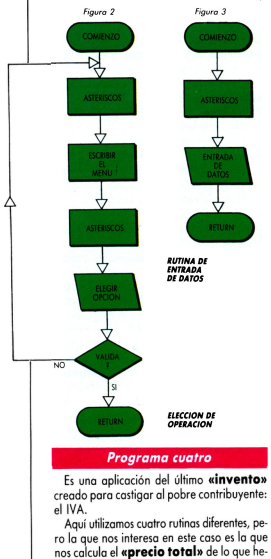
Podemos pasarle valores a una subrutina desde el programa principal por medio de una variable a la que hemos llamado **«parámetro»** en el ejemplo anterior. Son los **«parámetros de entrada»**.

Figura 1



Pero no son los únicos. También podemos ir en sentido contrario, es decir, pasando —*o devolviendo*— al programa principal el valor de un dato dentro de una variable de vuelta. A éstos les hemos dado el nombre de «parámetros de salida» de la subrutina.

Con el Programa 4 vamos a intentar comprender su sentido más claramente.



Programa cuatro

Es una aplicación del último **«invento»** creado para castigar al pobre contribuyente: el IVA.

Aquí utilizamos cuatro rutinas diferentes, pero la que nos interesa en este caso es la que nos calcula el **«precio total»** de lo que hemos comprado *(IVA incluido)*.

Lo primero que hacemos es saltar a la subrutina de entrada de datos mediante la instrucción:

20 GOSUB 1000: REM ENTRADA DE DATOS

Su misión es introducir en las variables **«precio»** e **«IVA»** los valores que nosotros le damos desde el teclado. *(Procure que el valor de la segunda variable sea bastante bajo.)*

Una vez conocidos los datos, asignamos su contenido a las variables **«entrada1»** y **«entrada2»** que van a ser las que utilizará la subrutina «cálculo» para encontrar un precio total. Son los «parámetros de entrada» de dicha subrutina, o sea, los que le da, o pasa, el programa principal.

Y llegamos al punto clave: la llamada a **«cálculo»**. ¡Ojalá sea pequeño el impuesto!

GOSUB 2000: REM CALCULO

nos hace saltar a la línea 2000 que es donde comienza realmente la subrutina. Vemos que en ella se opera con los valores que le hemos pasado en las variables **«entrada1»** y **«entrada2»** y el resultado lo metemos en la variable **«salida»**: ésta es la que vamos a utilizar para devolver el número que hemos calculado —línea 2010.

Por eso se llama **«parámetro de salida»**. El valor de nuestras operaciones **«sale»** en la variable **«salida»** desde la subrutina hacia el programa principal.

De vuelta en él, pasamos esta cantidad a **«precio total»** *—línea 60—* y sacamos el informe de todos los cálculos efectuados haciendo una llamada a la rutina «informe» —3000.

Nuestra única intención con este último programa es conseguir que le quede un poquito claro el concepto de parámetro de entrada o de salida. Por eso no hemos entrado en explicaciones sobre la forma como funciona todo él. Si lo hemos conseguido, ¡enhorabuena a todos!

Hasta ahora hemos construido cualquier programa como un solo bloque codificándole de principio a fin, todo seguido y de una sola vez. Pero hemos avanzado y no podemos quedarnos ahí.

Vamos a intentar **«formar»** un programa utilizando piezas prefabricadas que cumplan con una función específica y determinada. **¿Intuyen que va a ser como una especie de «kit» que vamos a montar a base de subrutinas?**

De esta forma puede organizarse convenientemente de acuerdo con las tareas que va a realizar en cada momento y una subrutina, o pieza de este rompecabezas ejecutará una porción determinada del proceso completo. Es simplemente dividir el problema general en bloques cada vez más específicos. Y como dice el refrán: **«A menor bulto, mayor claridad.»** Lo que podemos interpretar como que un problema pequeño y sencillo se soluciona más fácilmente que uno grande y complejo.

Si después ensamblamos todos estos bloques de subrutinas bajo el mando del **«programa principal»**, nuestro enredo dejará de serlo: no habrá pegas a la hora de resolverlo.

Vamos a trabajar. Partimos de un caso práctico en el que se nos pide un programa que nos sirva para calcular cualquier operación aritmética básica con dos números.

Para diseñar la forma de encontrarle una solución lógica utilizaremos una herramienta bastante conocida, aunque poco utilizada, en el mundo de la informática: los ORGANIGRAMAS.

Es un modo de desarrollar, por medio de unos gráficos ya establecidos, todo el proceso por el que tenemos que pasar para que el programa cumpla con todos los requisitos —o especificaciones— que nos han dado. Descompongamos nuestro problema en distintas fases.

Lo primero que tenemos que hacer es ELEGIR la operación matemática que queremos efectuar: suma, resta, multiplicación o división. En principio vamos a ver cada parte de una manera general y suponerla ya resuelta *(ya llegará el momento de desarrollarla con más detalle)*.

A continuación, y una vez determinada la operación, lo que procede es REALIZARLA e IMPRIMIR EL RESULTADO.

Por si acaso no queremos hacer solamente un solo cálculo sino varios, tendremos que dar al programa la opción de CONTINUAR hasta que nosotros queramos o TERMINAR si «nos hemos quedado sin números».

Si observa la Figura I podrá seguir mejor gráficamente todo este proceso.

FIGURA I

Como se ve lo hemos dividido en una serie de «cajas» que cumplen, en teoría, una misión específica. Podemos establecer una correspondencia entre lo que hemos llamado «Programa principal» y esta representación gráfica. Cada una de las «cajas» sería equivalente a una llamada a la subrutina necesaria en ese momento efectuada desde el programa principal.

Podríamos ir codificando con instrucciones que conozca nuestro **Amstrad** este **«centro de operaciones»**. Seguramente que usted habrá pensado ya la forma de hacerlo y que será bastante parecida al Programa 5.

Programa cinco

Consiste en una serie de llamadas a subrutinas, una por «caja» como dijimos anteriormente, y poco más. ¡Es enormemente sencillo!, pero si seguimos su listado nos dará una perspectiva general del problema a resolver **¿verdad que sí?**

Ahora solamente nos queda ir desarrollando los bloques mediante unos organigramas más precisos: **«Reduzcamos nuestros porblemas al tamaño en que podamos manejarlos.»**

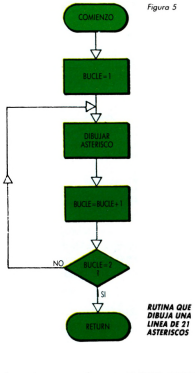


Figura 5

RUTINA QUE DIBUJA UNA LINEA DE 21 ASTERISCOS

La primera **«caja»** es ELEGIR OPERACION. **¿La desmenuzamos?**. El gráfico de la Figura II nos da una visión de cómo lo hemos hecho.

FIGURA II

Simplemente nos presenta en la pantalla un menú, lo más estético posible, y nos pide un dato que en este caso será el número distintivo de la operación.

Observe el retorno que se produce cuando le damos un número erróneo: volvemos al principio del organigrama y comenzamos el proceso de nuevo.

Si le pedimos que escriba la subrutina encargada de hacer todo esto, estamos seguros que no tendrá ninguna dificultad a la hora de hacerlo. Por si acaso le proporcionamos el Programa 6 que puede utilizar como guía de consulta.

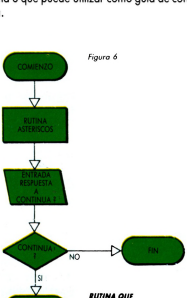


Figura 6

RUTINA QUE INVESTIGA SI HAY MAS OPERACIONES

Programa seis

Y así iríamos construyendo cada una de las subrutinas que van a dar forma a nuestro programa. Primero hacemos el organigrama de la misma y a continuación la codificamos con las instrucciones apropiadas y sobre todo **«con mucho cuidado»**. Una observación. Vea que en algunas de ellas hacemos también llamadas a otras subrutinas, pero esto ya no representa ningun problema para nosotros.

Intente repetir esta operación para cada una de las subrutinas que vamos a necesitar. Primero haga el organigrama lo más detallado posible y después páselos a instrucciones BASIC, su **Amstrad** se lo agradecerá.

Por si acaso no dispone de mucho tiempo para dedicar a estos temas, le damos la facilidad de observar los programas y figuras restantes —*que nos hemos encargado de hacer*— y estudiar sobre ellos la forma en que están construidos. Pero a pesar de ello insistimos en la conveniencia de que sea usted mismo quien realice los organigramas y los codifique después. Luego compare pero por si acaso... se los damos hechos.

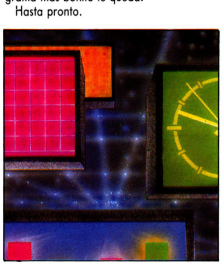
Para terminar, una cosa ha de quedar clara, es muy necesario que organice sus programas de acuerdo con la función concreta que desempeña una o más funciones. Si así lo hacemos seguro que es mucho más fácil leerlos e irlos siguiendo al contrario que si tenemos un listado con las tareas mezcladas y desordenadas.

A la hora de probar que funciona el programa correctamente también llevamos ventaja, ya que si empleamos este método podremos ir probando cada **«bloque»** de programa por separado. Y recuerde: los problemas pequeños se resuelven mejor que los grandes.

Y además una subrutina, por ejemplo de ENTRADA DE DATOS, no sólo nos sirve para un único programa, sino que podemos intercalarla dentro de otro que la necesite y ahorrarnos el trabajo de hacerla de nuevo.

Bueno, llega la hora de despedirnos. Mezcle convenientemente todas las últimas subrutinas que hemos desarrollado y ¡verá qué programa más bonito le queda!

Hasta pronto.



```
10 REM PROGRAMA I
20 CLS
30 INPUT "PRIMER OPERANDO: ",primer
o
40 INPUT "SEGUNDO OPERANDO: ",segun
do
50 GOSUB 1000
60 GOSUB 2000
70 GOSUB 3000
80 GOSUB 4000
90 INPUT "MAS CALCULOS";s$
100 IF s$="s" THEN GOTO 20 ELSE END
1000 REM RUTINA SUMA
1010 PRINT
1020 suma=primero+segundo
1030 PRINT primero;"+";segundo;"=";
suma
1040 PRINT
1050 RETURN
2000 REM RUTINA RESTA
2010 PRINT
2020 resta=primero-segundo
2030 PRINT primero;"-";segundo;"=";
resta
2040 PRINT
2050 RETURN
3000 REM RUTINA MULTIPLICACION
3010 PRINT
3020 multiplicacion=primero*segundo
3030 PRINT primero;"*";segundo;"=";
multiplicacion
3040 PRINT
3050 RETURN
4000 REM RUTINA DIVISION
4010 PRINT
4020 division=primero/segundo
4030 PRINT primero;"/";segundo;"=";
division
4040 PRINT
4050 RETURN
```

```
10 REM PROGRAMA II
20 CLS
30 LOCATE 11,5
40 GOSUB 1000
50 GOSUB 2000
60 LOCATE 11,19
70 GOSUB 1000
80 END
1000 REM ASTERISCOS HORIZONTALES
1010 FOR bucle=1 TO 15
1020 PRINT"*";
1030 NEXT bucle
1040 RETURN
2000 REM ASTERISCOS VERTICALES
2010 FOR bucle=1 TO 13
2020 PRINT TAB(11)"*";TAB(25)"*"
2030 NEXT bucle
2040 RETURN
```

```
10 REM PROGRAMA III
20 CLS
30 INPUT"NUMERO DE ASTERISCOS POR L
ADO: ",numero
40 parametro=numero
50 LOCATE 11,5
60 GOSUB 1000
70 GOSUB 2000
80 LOCATE 11,5+numero-1
90 GOSUB 1000
100 END
1000 REM ASTERISCOS HORIZONTALES
1010 FOR bucle=1 TO parametro
1020 PRINT"*";
1030 NEXT bucle
1040 RETURN
2000 REM ASTERISCOS VERTICALES
2010 FOR bucle=1 TO parametro-2
2020 PRINT TAB(11)"*";TAB(11+parame
tro-1)"*"
2030 NEXT bucle
2040 RETURN
```

Primeros pasos

```
10 REM PROGRAMA IV
20 GOSUB 1000: REM ENTRADA DE DATOS
30 entrada1=precio
40 entrada2=iva
50 GOSUB 2000:REM CALCULO
60 preciototal=salida
70 GOSUB 3000:REM INFORME
80 END
1000 REM ENTRADA DE DATOS
1010 CLS
1020 INPUT"PRECIO: ",precio
1030 INPUT"PORCENTAJE DE I.V.A.: ",
iva
1040 RETURN
2000 REM CALCULO
2010 salida=entrada1+(entrada1*entr
ada2/100)
2020 RETURN
3000 REM INFORME
3010 CLS
3020 LOCATE 1,10
3030 GOSUB 4000:REM ASTERISCOS
3040 PRINT TAB(10)"PRECIO: "precio
3050 PRINT TAB(10)"I.V.A.: "iva;"%
"
3060 PRINT TAB(10)"PRECIO TOTAL: ";
preciototal
3070 GOSUB 4000:REM ASTERISCOS
3080 RETURN
4000 REM ASTERISCOS
4010 FOR bucle=1 TO 39
4020 PRINT"*";
4030 NEXT bucle
4040 RETURN
```

```
10 REM PROGRAMA V (PROGRAMA PRINCIP
AL)
20 GOSUB 1000: REM MENU
30 IF opcion=1 THEN GOSUB 3000:REM
SUMA
40 IF opcion=2 THEN GOSUB 4000:REM
RESTA
50 IF opcion=3 THEN GOSUB 5000:REM
MULTIPLICACION
60 IF opcion=4 THEN GOSUB 6000:REM
DIVISION
70 GOSUB 8000:REM CONTINUO ?
80 GOTO 20
```

```
1000 REM PROGRAMA VI (MENU)
1010 CLS
1020 LOCATE 10,7
1030 GOSUB 7000:REM ASTERISCOS
1040 PRINT:PRINT
1050 PRINT TAB(11);" 1  SUMA"
1060 PRINT TAB(11);" 2  RESTA"
1070 PRINT TAB(11);" 3  MULTIPLICAC
ION"
1080 PRINT TAB(11);" 4  DIVISION"
1090 PRINT
1100  LOCATE 10,14
1110 GOSUB 7000:REM ASTERISCOS
1120 LOCATE 11,16
1130 INPUT "ELIGE UNA OPCION  ",opc
ion
1140 IF opcion<1 OR opcion>4 THEN G
OTO 1000
1150 RETURN
```

```
2000 REM PROGRAMA VII (ENTRADA DE D
ATOS)
2010 CLS
2020 LOCATE 10,7
2030 GOSUB 7000:REM ASTERISCOS
2040 PRINT:PRINT:PRINT TAB(10);
2050 INPUT "PRIMER OPERANDO: ",prima
ero
2060 PRINT TAB(10);
2070 INPUT "SEGUNDO OPERANDO: ",segu
undo
2080 RETURN
```

```
3000 REM PROGRAMA VIII (RUTINA SUMA
)
3010 GOSUB 2000:REM ENTRADA DE DATO
S
3020 PRINT
3030 suma=primero+segundo
3040 PRINT TAB(13)primero;"+";segun
do;"=";suma
3050 PRINT
3060 RETURN
```

```
4000 REM PROGRAMA IX (RUTINA RESTA)
4010 GOSUB 2000:REM ENTRADA DE DATO
S
4020 PRINT
4030 resta=primero-segundo
4040 PRINT TAB(13) primero;"-";segu
ndo;"=";resta
4050 PRINT
4060 RETURN
```

```
5000 REM PROGRAMA X (RUTINA MULTIPL
ICACION)
5010 GOSUB 2000:REM ENTRADA DE DATO
S
5020 PRINT
5030 multiplicacion=primero*segundo
5040 PRINT TAB(13) primero;"*";segu
ndo;"=";multiplicacion
5050 PRINT
5060 RETURN
```

```
6000 REM PROGRAMA XI (RUTINA DIVISI
ON)
6010 GOSUB 2000:REM ENTRADA DE DATO
S
6020 PRINT
6030 division=primero/segundo
6040 PRINT TAB(13)primero;"/";segun
do;"=";division
6050 PRINT
6060 RETURN
```

```
7000 REM PROGRAMA XII (ASTERISCOS)
7010 FOR bucle=1 TO 21
7020 PRINT"*";
7030 NEXT bucle
7040 RETURN
```

```
8000 REM PROGRAMA XIII (RUTINA CONT
INUAR ?)
8010 PRINT TAB(10);
8020 GOSUB 7000:REM ASTERISCOS
8030 PRINT:PRINT
8040 PRINT TAB(13);
8050 INPUT "MAS CALCULOS";s$
8060 IF s$="s" THEN RETURN ELSE END
```

Hasta ahora hemos estado trabajando sin darle importancia al hecho de que las letras fueran mayúsculas o minúsculas, pero como ya sabrá, en Basic las palabras clave deben ir siempre escritas en mayúsculas. Su **Amstrad** ya viene preparado para que esto ocurra así, pues aunque tecleemos:

10 print **«Hola»**

saldrá impreso como:

10 PRINT **«Hola»**

es decir, que la palabra clave será transformada a mayúsculas, pero como podrá ver, la palabra **«Hola»** no sufre ningún cambio, puesto que al venir entre comillas se la considera como una cadena y éstas son totalmente inviolables.

Todas las sentencias siguientes:

10 pRinT **«Hola»**
10 PrInt **«hola»**
10 PRiNT **«Hola»**

saldrán listadas como:

10 PRINT **«Hola»**

En el programa II vemos que todas las variables *(nombre, hecho, tratamiento)*, vienen escritas en minúsculas. Esto ocurre porque yo lo he tecleadp así, pero aquí hay una cuestión interesante.

En su **Amstrad**:

10 PRINT tratamiento$

y

10 PRINT TRATAMIENTO$

aparecerá escrita de diferente manera, pero la variable será la misma, puesto que el **Amstrad** no diferencia entre tratamiento$ y TRATAMIENTO$. En las variables las mayúsculas y las minúsculas son equivalentes. Así pues:

tratamiento$
TRATAMIENTO$
TRAtaMienTo$
trataMIENTO$

serán consideradas como la misma variable.

Medidas de seguridad

Mi recomendación es que siempre teclee las variables en minúsculas, de esta manera podrá distinguir fácilmente entre éstas y las palabras clave.

BASIC EN 2 DIMENSIONES

Primeros repasos

Reglas de nomenclatura

Actualmente podemos usar letras mayúsculas para las variables e intercalarlas con minúsculas y con números. Las reglas que rigen esto son:

— Todo nombre de una variable debe comenzar con una letra, aunque puede ir seguido por cualquier tipo de letra, número o carácter. Las letras mayúsculas y minúsculas son equivalentes.

— No está permitido intercalar espacios en blanco en el nombre de una variable.

— Las variables deben permanecer separadas de las papabras clave de Basic. Uno de los errores más comunes es colocar una variable junto a un comando.

Una de las mayores ventajas de utilizar variables en lugar de cadenas es que nos permite alterar fácilmente la salida del programa. En el caso de el programa 1, si queremos que nuestra carta llegue a otra víctima, tan sólo deberemos cambiar la línea 30. Por ejemplo:

30 nombre$ = **«Sr. Pérez»**

A partir de ese nombre todos los usos de nombre$ estarán referidos a el Sr. Pérez.

En este caso, al ser un programa corto no reviste mucha importancia, pero en otros más largos si ha utilizado la cadena «Sr. Gómez», en lugar de una variable, se verá obligado a realizar un gran número de modificaciones, una por cada vez que la haya utilizado.

El hecho de escribir las variables en minúculas nos ayudará, sobre todo a los recién iniciados en la programación, a no confundirlas con las palabras clave.

Formateando el texto: el comando TAB ()

Este nos ayudará para etudiar una nueva idea, el uso de la función TAB (). Esta función nos permitirá especificar a qué distancia a lo largo de una línea queremos que nos imprima el texto de un comando PRINT.

En el Mode 1 la línea tiene 40 caracteres y, por tanto, la pantalla se puede considerar dividida en 40 columnas. TAB () decidirá en cuál de esas columnas empezará a escribir. La numeración de las mismas va desde 1 a 40.

Según en el Mode con el que estemos trabajando, así será el número de caracteres de la pantalla, es decir, el número de columnas. Por ejemplo, el Mode 2 sólo tiene 20 columnas.

Pruebe a cambiar la línea 20 de este programa por:

20 MODE 0

¿Has visto lo que pasa?

Uno de los mayores errores de algunos supuesto buenos programas es su mala colocación en la pantalla. Un uso correcto a la función TAB () puede evitar estos problemas.

Para coger alguna práctica, veamos el programa 2. Este imprime un triángulo de asteriscos. **¿Podría usted inventar un programa que, con la ayuda de TAB (), dibujara una estrella de asteriscos?**

Antes de que empiece le diré que lo podrá realizar más fácilmente si vuelve a Mode 1 de la siguiente manera:

MODE 1 (ENTER)

Y por el momento, esto es todo lo que tenemos que decir acerca de las variables de cadena. Sin embargo, existe otro tipo de variables, que son las llamadas variables numéricas.

Las variables numéricas: segunda clave

No existe prácticamente ninguna diferencia en la asignación de éstas y las de cadena, excepto que, lógicamente, en lugar de estar formadas por una serie de caracteres, lo estarán por números y, por tanto, podremos realizar con ellas una serie de preparaciones matemáticas.

Pruebe a correr el programa 3.

La línea 30 utiliza la variable numérica A para asignar al número 10. Dese cuenta de que para las variables numéricas no es necesario el uso del carácter $, pues es suficiente con asignarle una letra del alfabeto. También habrá comprobado que tampoco se necesita poner el valor de estas variables entre comillas, como se puede ver claramente en la línea 30:

30 A = 10

La línea 40 por supuesto que no imprimirá A, sino el valor que ésta tiene asignado, que en este caso es 10.

Lo más interesante ocurre en la línea 50. Aquí, multiplicamos el valor de A por 2 y, por tanto, se imprimirá 20.

Este es el punto más importante de las variables numéricas, el poder operar con ellas.

El comando Input

El programa 4 nos saca de apuros. El truco consiste en el uso de INPUT nombre$ de la línea 40. La línea 30 de el programa VIII asigna el valor MIGUEL a la variable nombre$. En el programa IX esta variable no tiene aún un valor determinado.

Cuando el programa llega a la línea:

40 INPUT nombre$

espera hasta que le introduzcamos (INPUT), el valor que queremos de nombre$ para imprimir dicho valor.

En otras palabras: cuando el ordenador se encuentra con un INPUT seguido de una variable, le pregunta qué quiere que valga la variable, de hecho le muestra un signo de interrogación en la pantalla.

Ahora está en disposición de teclear la respuesta seguida de ENTER, que será quien se encargue de enviarlo al ordenador, para que éste siga con el desarrollo del programa.

Así, cuando corra este programa, la línea 30 le preguntará: **«¿Cómo te llamas?»,** pero dese cuenta de que no son necesarias las interrogaciones, pues la línea 40 se encarga de esto.

Entonces el micro esperará nuestra respuesta y la enviará mediante ENTER. Sea lo que sea lo que hayamos escrito, pasará a ser el valor de nombre$, aunque le hayamos mentido.

La línea 50 imprimirá el mensaje.

```
10 REM PROGRAMA 1
20 MODE 1
30 nombre$="Sr. Perez"
40 hecho$="Usted me debe dinero"
50 causa$="pagueme o si no..."
60 PRINT
70 PRINT "Pobre "nombre$
80 PRINT TAB(4) hecho$;causa$
90 PRINT TAB(15) "Atentamente"
100 PRINT TAB(17) "Miguel"
```

```
10 REM PROGRAMA 2
20 MODE 1
30 PRINT
40 PRINT TAB(5) "*"
50 PRINT TAB(4) "**"
60 PRINT TAB(3) "***"
70 PRINT TAB(1) "****"
```

```
10 REM PROGRAMA 3
20 MODE 1
30 A = 10
40 PRINT A
50 PRINT 2 * A
```

```
10 REM PROGRAMA 4
20 MODE 1
30 PRINT "Cual es tu nombre ";
40 INPUT nombre$
50 PRINT "Buenos dias, "nombre$
```

COPY DE PANTALLA POR IMPRESORA

Por Robert y Christine Chatwin

***Una de las cosas en las que el Amstrad no resiste la comparación con sus más inmediatos competidores, como el Spectrum, es la posibilidad de realizar en todo momento un volcado del contenido de la pantalla a la impresora, lo que se conoce como una copia de pantalla o HARDCOPY** («copia dura» en inglés).*
Así, al Spectrum, por ejemplo, basta darle la orden COPY y, si la impresora es la adecuada, ¡voila!, aparece la copia en papel de la pantalla.
Obviamente, este problema clamaba por una solución a grandes voces, y nuestros colaboradores Robert y Christine lo han resuelto... para varios tipos de impresoras.
En el artículo que sigue todo el mundo podrá aprender cómo y por qué se realiza un hardcopy, bien sea en Basic o en lenguaje máquina, en este último caso creando nuevos comandos RSX.

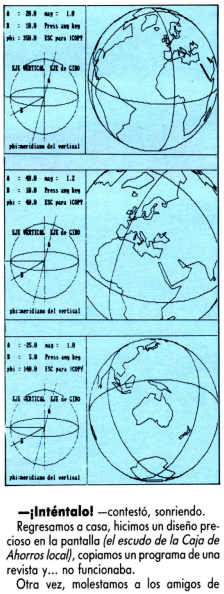
Después de varias semanas leyendo montañas de propaganda y contando con mucho cuidado las pesetas disponibles, la familia votó por comprar el ordenador **Amstrad,** modelo barato, casette y marcianitos en verde y todo. En desfile fuimos a unos grandes almacenes y media hora después llegamos a casa con nuestro tesoro. Ya habíamos preparado el lugar de honor en el cuarto de estar con nueva mesa y estantería lista para las cintas y los libros que íbamos a comprar.

Orgullosamente, empezamos a comparar nuestro **Amstrad** con otras máquinas —*hicimos varias pruebas y encontramos que el aparato era muchísimo mejor en velocidad, en su pantalla, etc. que muchas máquinas con procesador de 16 bits, el IBM-PC incluso*—. Así hemos hecho muchos enemigos, comprobando que nuestro **Amstrad** de precio de mercadillo era un gigante comparado con las máquinas suyas, hasta que hablamos con el propietario de un simple Spectrum de 28.000 ptas.

Con el Amstrad no se puede

—Bien —dijo él—, **muy bonito. Pero no se puede imprimir la pantalla por impresora, como yo hago con mi Spectrum.**

—No digas tonterías —le contestamos—, **por supuesto que se puede.**



—¡Inténtalo! —contestó, sonriendo.

Regresamos a casa, hicimos un diseño precioso en la pantalla *(el escudo de la Caja de Ahorros local)*, copiamos un programa de una revista y... no funcionaba.

Otra vez, molestamos a los amigos de **Amstrad,** preguntándoles cómo ellos pasan diseños a la impresora. O no lo habían intentado, o no funciona, con feas líneas verticales por doquier. ¡Qué vergüenza! Nuestro **Amstrad,** la crème de la crème del micro, vencido por un Spectrum. ¡Imposible! Entonces con hielo sobre la cabeza, nos pusimos a trabajar para intentar solucionar el problema.

¡Claro que se puede!

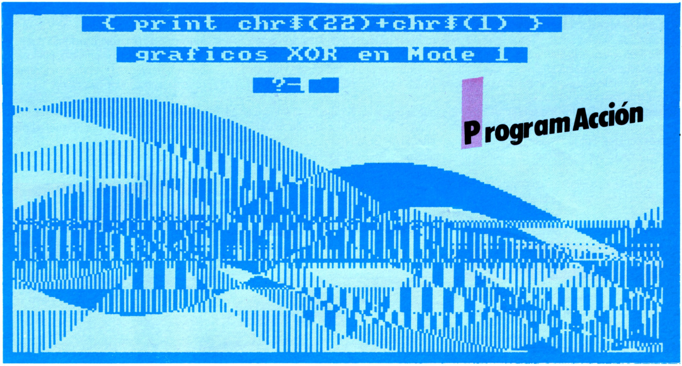
Primero ves si tienes impresora. Si no, no te preocupes. Nosotros tampoco. Teclea sólo el programa final, ése del código máquina y busca un amigo con impresora. Y regálale una copia del programa.

Con la impresora, ver el manual para MODO GRAFICO, que también se llama imagen de bits *(bit image)* porque un bit *(cifra binaria en el número enviado a la impresora)* sale como un punto *(dot)* en el papel. Este modo puede ser de dos tipos.

Lo más probable es el tipo de COMPATIBILIDAD EPSON. Habrá un diagrama mostrando bit 0 enviado al punto *(dot)* más bajo en el cabezal de impresión. *(Eso de los bits es curioso* —los ingenieros siempre cuentan desde 0 y lo que para mí es el octavo bit, para ellos es sólo bit 7. Tal vez así les sale más barato el IVA.)

Los programas que siguen, se han escrito para este tipo. Si, al contrario, bit 0 se muestra dirigido al punto *(dot)* más alto de impresión, este tipo vamos a llamar de SEIKOSHA, porque lo hemos probado en una SEIKOSHA GP550A. El programa final tiene unos pokes adicionales que lo adopte a este tipo de impresora.

Si no está claro de qué clase es la impresora, aplica el método conocido entre los cien-



ProgramAcción

tíficos de SIAS. Esto es **«Suck it and see».** Pruébalo y verás. Teclea los programas y si todo sale invertido y al revés, debe ser del otro tipo.

Aparte de la SEIKOSHA GP550A, hemos probado los programas en otros dos: INDESCOMP 80 y STAR GEMINI 10X, los dos del tipo EPSON. Todas son máquinas económicas de unos 100 cps de velocidad.

ADVERTENCIA: El uso de RUN antes de SAVE puede ser perjudicial para su salud mental. Si no tienes impresora conectada.

?#8, **«Hola»**

y —adiós. Tendrás que apagar todo el aparato. Entonces teclea el programa y grábalo ANTES de probarlo. *Microhobby* no quiere ser causa de suicidios en masa de las mejores mentes del país.

Probando, probando

Programa 1: PRUEBA. Este es de introducción. Los expertos en el asunto ya pueden saltar a programa 2, pero entonces les faltarán muchas líneas del presente programa que se usan allí. Cuando tecleas, no hace falta contar espacios: las indentaciones *(p. ej. líneas 61020-61100)* se usan para ayudar a comprender la estructura de los bucles y no necesitas seguirlas. Las observaciones *(REMs)* que siguen al apóstrofe (') pueden reemplazarse con un par de asteriscos ('**). Las líneas DATA también son de este estilo: terminan con una ètiqueta de postizo para indicar lo que los números representan. Puedes teclear más simplemente

64000 DATA 27, 64, 27, 177, **

respetando las comas.

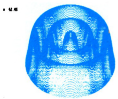
```
60000 '  prog PRUEBA graficos impresora  Microhobby 1986
60010 '
61000 GOSUB 63020'                  leer codigos de control
61010 PRINT#8,espacio$;
61020 FOR linea=1 TO 6
61030  PRINT#8,CHR$(10);
61040  FOR c=1 TO 4
61050    PRINT#8,grafico$;
61060    FOR b=1 TO 50
61070     car7bit=64+8'              enviar bit 3 y bit 6
61080     PRINT#8,CHR$(car7bit);CHR$(car7bit);
61090   NEXT b,c
61100 NEXT linea
61120 PRINT#8,goodbye$
61130 END
63000 '
63010 '
63020  RESTORE 64000
63030  GOSUB 65000:espacio$=control$
63040  GOSUB 65000:grafico$=control$
63050  GOSUB 65000:goodbye$=control$
63240 RETURN
63250 '
64000 DATA 27,64,27,177,esc @ esc 1 (espacio 7 bit carac.)
64010 DATA 27,75,100,128, esc K 100 0 (100 carac.graficos)
64020 DATA 10,27,192,   LF esc @                 ( adios )
64030 '
65000 '                 rutina para leer codigos de control
65010  control$="":dato=0
65020   WHILE dato<128
65030    READ dato
65040    control$=control$+CHR$(dato MOD 128)
65050    WEND
65060   READ postizo$
65070 RETURN

Para estilo SEIKOSHA, tecleas las siguientes lineas

64000 DATA 27,84,49,180,                  espacio (para 7 puntos)
64010 DATA 14,27,71,49,48,176,elongacion+esc G100(100 car.graf)
64020 DATA 10,27,54,143,      LF esc 6 chr(15)          ( adios )
```

Teclar y grabarlo con SAVE «prueba» en casette o disco. Ahora encender la impresora y RUN. Si salen 6 líneas dobles, largas, horizontales, fantástico. Pero, si cuando yo lo hago, salen estas líneas parcialmente rotas por **«gremlins»** en la máquina, con unos pocos **«H»** y **«,»** esparcidos, bien. Estos gremlins saben Basic, pero no Código Máquina, y la versión final va a funcionar correctamente.

Si no hay nada de líneas dobles horizontales, en verdad falla algo. Comprobar el programa y leer la parte del manual de la impresora que trate de modo gráfico. Al menos, no vale la pena seguir tecleando más programa, hasta que esta parte, de prueba, funcione.



¿Cómo funciona el programa?

En el programa PRUEBA se ven cosas como **«car7but»** o **«espacio 7 bit».** Un carácter para la máquina es un número entre 0 y 255. Estos números consisten de ocho bits *(cifras binarias).* Si quieres ver los bits en acción, prueba lo siguiente:

for car=115 to 135:? car; bin$ (car, 8): next car

y se nota que bit 7, el primero desde la izquierda, se alza *(cambia de 0 a 1)* entre 127 y 128. Hasta 255 se queda alzado.

La puerta Centronics *(Printer)* en el **Amstrad** corta la cabeza de cualquier carácter que pasa por ahí y entonces no llega a la impresora el octavo bit, sólo bits 0-6. Línea 64000 son los DATA para un espacio entre líneas que corresponde a siete **«dots** en el papel. Compruébalo y si hace falta cambia los DATA, enviando un carácter diferente. Teclea:

61070 car 7 bit=63

y RUN. Ahora *(sin hacer caso de* **«gremlins»** *si los hay)* se ve una fina línea blanca entre bloques de negro. Esta línea corresponde al bit 6 (64) siendo el único que no pone un punto en el papel.

61070 car 7 bit=127

y las dos franjas de negro deben unirse justamente, pero sin solapar. Si piensas que el espacio hay que ajustarle, cambia los datos correspondientes en línea 64000.

Los datos terminan en un número al cual se ha añadido 128 como señal de fin de datos, por ejemplo, 177=49+128 y 49 dice el manual de la impresora es el código para espacio entre líneas de siete puntos *(dots).*

Los datos de línea 64010 son los códigos de control para enviar 100 caracteres gráficos a la vez.

La línea 64020 es la que más probablemente aprovecharás cambiando. Esta repone la impresora en el estado que normalmente uses. En el programa hay los códigos más sencillos de reiniciación. Puede ser que tu impresora no reconoce ESC@. Entonces cámbialos por otros códigos. Por lo menos, juega con esta línea y ver lo que los cambios hacen con RUN seguido por LIST 64000-64020, #8.

Estos mismos códigos de control pueden pokearse en el programa final en código máquina.

Hardcopy en Basic

No borres el programa de memoria. Vamos a aumentarlo para copiar la pantalla a papel, usando Basic.

```
10  'prog  COPION.BASIC  Microhobby Semanal 1986
20  'Dr Robert Chatwin, Badajoz
30  '
100 MODE 1:PAPER 1:CLS:PAPER 0
110 ORIGIN 0,0,8,631,8,390:CLG 0:MOVE 700,42
120 gra.or$=CHR$(23)+CHR$(3):PRINT gra.or$
130 DEG:ang=0:alfa=120:dalfa=-0.04
140 long=400:dlong=-3
150 WHILE long>0
160  FOR lado=1 TO 3
170   ang=ang+alfa
180   DRAWR 2*long*COS(ang),long*SIN(ang),lado
190   alfa=alfa+dalfa:long=long+dlong
200   NEXT lado
210  WEND
220 ON BREAK GOSUB 63000
230 WHILE UPPER$(INKEY$)<>"C":WEND
240 gra.and$ =CHR$(23)+CHR$(2)
250 gra.norm$=CHR$(23)+CHR$(0)
260  PRINT gra.and$
270  mask=1
280  FOR colum=0 TO 640 STEP 2
290    MOVE colum,0:DRAWR 0,400,mask
300    NEXT colum
310  PRINT gra.norm$
320 GOSUB 63000
330 END
63000  ON BREAK STOP'                   rutina copiar 640x400 puntos
63010 t0=TIME
63020  RESTORE 64000
63030  GOSUB 65000:espacio$=control$
63040  GOSUB 65000:grafico$=control$
63050  GOSUB 65000:goodbye$=control$
63060  PRINT#8,espacio$;
63070  ORIGIN 0,0,0,640,0,400'                          pantalla entera
63080  MOVER -1,398
63090  WHILE XPOS<640
63100   PRINT#8,CHR$(10);
63110   MOVER 7,-400'                                    nueva columna
63120   FOR c=1 TO 4'     cada columna sera 4x50x2 puntos   400
63130    PRINT#8,grafico$;
63140    FOR b=1 TO 50
63150     car7bit=(TESTR(-6,2)=0)+1'      punto aqui? devuelve 1
63160     FOR bc=1 TO 6
63170      car7bit=(TESTR(1,0)=0)+1+2*car7bit
63180     NEXT bc
63190     PRINT#8,CHR$(car7bit);CHR$(car7bit);'        dos veces
63200   NEXT b,c
63210   WEND
63220  PRINT#8,goodbye$
63230 t0=(TIME-t0)/300:PRINT"tiempo = "t0\60"min"t0 MOD 60"seg"
63240 RETURN
63250 '
64000 DATA 27,64,27,177,          esc @ esc 1     (espacio)
64010 DATA 27,75,100,128,         esc K 100 0     (grafico)
64020 DATA 10,27,192,             LF esc @        (adios  )
64030 '
65000 '                       rutina para leer codigos de control
65010  control$="":dato=0
65020   WHILE dato<128
65030    READ dato
65040    control$=control$+CHR$(dato MOD 128)
65050    WEND
65060   READ postizo$
65070 RETURN
```

Notarás que no necesitas teclear otra vez las líneas 63020-50 y 63240-final, porque son iguales que antes. Tampoco te molestes en borrar los bloques antiguos 60000 y 61000, que no se usan aquí.

Las impresoras de la otra clase *(como la SEIKOSHA GP550A)* tienen dos líneas diferentes, aparte aquéllas de los DATA del programa anterior. Son las dos siguientes:

63150 car 7 bit=64* *((TESTR (—6,2)=0)+1)* punto aquí? —devuelve 64

63170 car 7 bit=64* *((TESTR (1,0)=0)+1)*+car 7 bit/2

cuyo efecto es copiar el grupo de siete puntos de la pantalla en orden inverso. La multiplicación *2 ha sido reemplazado por división /2, que en el programa de código máquina será RL- rotación hacia la izquierda, reemplazado por RR- rotación hacia la derecha.

Grabar el programa con SAVE **«copión. bas».** Coge unos cinco pliegues de papel continuo usado, que estabas a punto de tirar, porque lo que saldrá probablemente será un desastre. Cuando lo hice en una INDESCOMP 80 y STAR GEMINI, salieron beeps, alimentación de hoja y todo. Pero también...

Enciende la impresora, RUN y cuando el dibujo está acabado pulsa la tecla **«C».** Y a comer, toma un café o algo. Porque el pobre Basic tarda. Para mí, el tiempo mostrado en la pantalla al final era de 21 min y 20 seg.

Lo que hace el programa es procesar los 640×2000 pixels de la pantalla en bloques de siete, preguntando a través de la función TESTR si están encendidos o no y pasar la información a la impresora en forma de bits —un bit por pixel —para imprimir los puntos en papel. *(Cada bloque se repite su impresión para tener una copia 640×400).*

Pero, 640×200 veces en Basic puede ser demasiado. Se diseñó Basic para ser interactiva, no repetitiva. Ahora vamos a ver el programa en código máquina, pero la lógica será igual a lo que acabamos de ver.

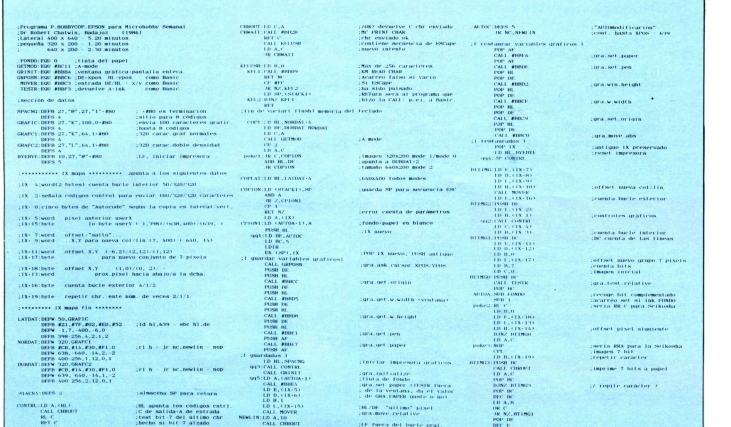
Lenguaje máquina: el plato fuerte

Esta vez teclea NEW. Hay una comprobación en el programa que los datos numéricos

Para las impresoras estilo SEIKOSHA GP550A, hay que seguir tecleando estas lineas:

```
2300 '                                            Seikosha pokes
2310 poke2=&1EF
2320 poke3=&1FF
2330 POKE (h+1)+poke2+1,&19'RR C
2340 POKE (h+1)+poke3   ,&1F'RRA
2350 '
2360 '                                     codigos de ctrl - pokes
2370 '
2380 e$=CHR$(27)'ESC
2390 spacng=&65
2400 grafic=&6D
2410 grafc1=&75
2420 grafc2=&7D
2430 byebye=&85
2440  control$=e$+"T14" '  spacng$ espacio para 7 puntos (dots)
2450 direc=spacng:GOSUB 2900
2460  control$=CHR$(14)+e$+"G100"'grafic$ elong.+ 100 car.graf.
2470 direc=grafic:GOSUB 2900
2480  control$=CHR$(14)+e$+"G320"'grafc1$ elong.+ 320 car.graf.
2490 direc=grafc1:GOSUB 2900
2500  control$=e$+"G320"         'grafc2$          320 car.graf.
2510 direc=grafc2:GOSUB 2900
2520  control$=CHR$(10)+e$+"6"+CHR$(15)'      Seikosha goodbye$
2530 direc=byebye:GOSUB 2900
2899 RETURN
2900 '                                             pokear control$
2910 direc=direc+h
2920 FOR q=1 TO LEN(control$)-1
2930  POKE direc+q,ASC(MID$(control$,q,1))
2940  NEXT q
2950 POKE direc+q,ASC(MID$(control$,q,1))+&80'        terminator
```

```
100 DATA CD11BC3B3BE333332B2B2BE5112F0019C15E235623C57AB3C10986
110 DATA 2811EB09D5E55E2356EB09EBE1732372E118E5C317022D00180A85
120 DATA 021B028F00A700BF00D900DC00DF00F100FF001001160128O106E9
130 DATA 2C0138013C01620165016B0181018401AF01B2010602CE01000519
140 DATA 001B401BB1000000001B4B6480000000001B4B408100000000398
150 DATA 1B4C4081000000000A1BC000000000032006D00217F02ED52048D
160 DATA FFFF070070FEFAFF00008E0402010240017500CB1430F1007E0937
170 DATA 0280FDF2FF0200FEFF90010C000140017D00CB1430F1007F02094C
180 DATA 80FDF2FF0100FEFF90020C000100000000E100C31501C3270108B0
190 DATA 434F50D9434F50594C41D40000007ECDF900CB11D82318F64F09CF
200 DATA CD2BBDD8CD04017918F60600CD09BBD0FEFC2004ED7BED00100BD0
210 DATA F2C921A9001118004FCD11BCFE027938061918032191 00ED730894
220 DATA ED00A72806FE01C0DD7E0032EC01E511890101050 0EDB0DDE30ADE
230 DATA CDC6BBD5E5CDCCBBD5E5CDD5BBD5E5CDD8BBD5E5CDE1BBF5CD1472
240 DATA E7BBF5216500CDEF00CDBABB3AEC01CDE4BBDD5E05DD5606260D4D
250 DATA 01DD6E0FCDC3BB3E0ACDF900CDB401CDC6BB000000000030EE0AA2
260 DATA F1CDE4BBF1CDDEBBE1D1CDD2BBE1D1CDCFBBE1D1CDC9BBE1D1147E
270 DATA CDC0BBDDE1218500C3EF00DD5E07DD5608DD6E09DD660ACDC30D0C
280 DATA BBDD5E10D5DD6EFEDD66FFCDEF00DD4EFCDD46FDC5DD5E0BDD1051
290 DATA 560C2600DD6E1106070E00C5CDF3BBC1D600D601CB111600DD097C
300 DATA 5E12DD6E0DDD660E10E879002FDD4613C5CDF90079C110F8C10B7D
310 DATA 0B78B120C4D11D20B0C901D90021D500C3D1BC08BF
1000 endit =&220'longitud de codigo
1001 spacng=&65 'longitud de basura de reubicar
1002 ON ERROR GOTO 4000
1003 permit.error=5:SYMBOL AFTER 256
1004 permit.error=0
1005 OPENOUT"microhobby":h=HIMEM:rsx=h-8:h=h-endit:MEMORY h
1006 CLOSEOUT
1007    direc=h:GOSUB 2000'cargador
1008   'save"hobbycop"
1009    CALL h+1
1010 GOTO 3000'fin
2000 'rutina carga datos con checksum
2010 FOR linea=100 TO 310 STEP 10
2020   READ postizo$
2030   check.sum=VAL("&"+RIGHT$(postizo$,4))
2040   FOR puntero=1 TO LEN(postizo$)-5 STEP 2
2050     dummy=VAL("&"+MID$(postizo$,puntero,2))
2060     check.sum=check.sum-dummy
2070     direc=direc+1
2080     POKE direc,dummy
2090     NEXT puntero
2100   IF check.sum<>0 GOTO 4100
2110   NEXT linea
2120 'sitio para acomodacion propia
2130 'etc
2999 RETURN
3000 MEMORY h+spacng
3010 PRINT"Si hace falta, reiniciar comandos con
3020 PRINT" CALL &"HEX$(rsx)"  en Basic
3030 PRINT" user(#"HEX$(rsx)") en Pascal
3040 PRINT"|COPY     =  copy horizontal  fondo=ink 0
3050 PRINT"|COPYLAT =  copy lateral     fondo=ink 0
3060 PRINT"|COPY,TINTA opcion no imprime esta tinta
3070 PRINT"|COPYLAT,TINTA  etc.
3080    PRINT
3090 END'prog COPY for Microhobby Semanal 14Jan86 Robert Chatwin
4000 IF ERR<>permit.error THEN PRINT"**** Error ****":STOP
4001 RESUME NEXT
4100 MODE 2:PRINT CHR$(7)"Error en"linea:LIST-999
```



han sido entrados correctamente. Un error se señala con el mensaje **«Error en xxx»** y las líneas de DATA se alistarán. Teclea en MODE 2: el formato es de 64 caracteres por línea, para ajustarse a la norma de la revista y si saltas un dato, será claro a simple vista.

Una vez terminado, grabarlo con SAVE «hobbycop».

Características

Nuevos comandos Basic:

| COPY *(, tinta)*
| COPYLAT *(, tinta)*

— Copia de la pantalla de un solo color enviado a impresora con modo gráfico (bit image).

— Si no se especifica el parámetro opcional, INK 0 en la pantalla saldrá de papel blanco y los demás INKs saldrán negro.

— Con el parámetro, el fondo de papel sin imprimir corresponderá al INK «tinta», y los demás (INK 0 incluido) saldrán negro.

— Comprobado en las impresoras SEIKOSHA GP550A, INDESCOMP 80, y STAR GEMINI 10X.

— **|** COPY en mode-0/mode-1 da una imagen de 320×200 puntos *(80 seg)* en mode-2 da una imagen de doble densidad de 640×200 *(tardando 150 seg)*.

— **|** COPYLAT en cualquier mode da 640×400 puntos, copia lateral *(5 min y 20 seg)*.

— Al terminar la copia la impresora está reiniciada para la impresión de texto normal.

— La tecla de **«break»** ESC-ESC está reconocida, parando de emergencia la copia. En este caso la impresora necesita ser apagada para su reiniciación.

— Compatible con toda la familia de CPC464 *(472, 664, 6128)*, por reubicarse dinámica y automáticamente en memoria.

— Longitud después de reubicación, 443 bytes.

La barra vertical **|** que hay en el bloque de líneas 3000 se encuentra encima del signo **@** en el teclado. Se usa para dar acceso a nuevos comandos en Basic, en este caso comandos que residen en la misma memoria llamados RSX (extensiones residentes).

Teclea MODE 1. Enciende la impresora. RUN. Debe verse en la pantalla la información contenida en las líneas 3010-3070. Teclea simplemente.

| copy

y (ENTER) claro, y sale una copia de la pantalla en el papel.

| copy, 1

y debe salir ahora con blanco y negro invertido. *(Esta forma del comando significa: elegir INK 1 como fondo de papel blanco, e imprime todas las demás tintas)*.

Comienza una nueva hoja en la impresora, y

| copylat

y sale una copia lateral de la pantalla, tamaño grande.

Los dos nuevos comandos se quedan en la memoria, hasta que apagues la máquina. Carga otra vez el programa COPION.BAS con

LOAD **«copión.bas»**

El programa cargador HOBBYCOP se ha borrado, claro, pero su efecto queda. Tecleas:

63005 ! COPY: RETURN

y RUN. Cuando termina el dibujo en la pantalla, pulsa la tecla **«C»** *(de copiar)* y sale la copia al papel. Inténtalo otra vez, pero ahora en lugar de **«C»,** *pulsa ESC/ESC* para **«break»,** y saldrá el mismo dibujo, pero en otra forma. Tecleas:

63005 ! COPYAT: RETURN

y RUN para una copia grande.

La pantalla en MODE 1 tiene 320×200 pixels. En MODE 2, 640×200 pixels. Entonces **|** COPY hace una copia en doble densidad automáticamente, si la pantalla está en MODE 2, con cada pixel un punto en el papel. Pero, es posible que la impresora que uses no

tenga doble densidad de impresión en modo gráfico. En tal caso, adapta el programa HOBBYCOP con el siguiente poke de corto-circuito.

2200 'JR C, COPION cambiado a JR COPION en la dirección poke1.
2210 poke1=&122.
2220 POKE *(h+1)*+poke1, &18.

En una aplicación como el Pascal de Hisoft, uno tiene que reiniciar cualquier RSX al principio de cada programa. La llamada necesaria, user(rsx), se da en la pantalla al final de HOBBYCOP.

Código fuente del programa HOBBYCOP

El listado del programa fuente en Ensamblador se ha escrito en forma autodocumentada. Etiquetas en mayúsculas representan nudos de la estructura del programa. En minúsculas se usan sólo para reubicación o para pokes desde Basic.

Personalizar HOBBYCOP

Para acomodar el programa a tu propia impresora y gustos, cárgalo con LOAD «hobbycop» y añadir líneas de la misma forma que cambiamos el programa para máquinas estilo SEIKOSHA. Por ejemplo, me gusta reiniciar la impresora INDESCOMP 80 a letras pequeñitas y juego ASCII. Entonces pongo los códigos de control para decir adiós gráficos *(byebye)* en la manera siguiente:

2350 '
2360 ' códigos de ctrel - pokes.
2370 '
2380 eS=CHR$ (27) 'ESC.
2390 spacng=&65 'espacio entre líneas para 7 dots.
2400 grafic=&6D '100 caracteres gráficos densidad normal.
2410 grafc1=&75 '320 caracteres gráficos densidad normal.
2420 grafc2=&7D '320 caracteres gráficos densidad normal.
2430 byebye=&85 'adiós gráficos reinicio impresión de texto.
2520 control$=CHR$ (10)+eS+«R»+CHR$ (0)+CHR$ (15)+eS+«3»+CHR$ (16).
2530 direc=byebye: GOSUB 2900.
2899 RETURN.
2900 «pokear control$.
2910 direc=direc+h.
2920 FOR q=1 TO LEN (control$)—1.
2930 POKE direc+q, ASC (MID$ (control$, q, 1)).
2940 NEXT q.
2950 POKE direc+q, ASC (MID$ (control$, q, 1))+&80'terminator.

Ahora RUN. Y después de usar COPY, me deja la impresora otra vez con la forma de letra que me gusta. Las direcciones de los cinco bloques de códigos de control se encuentran en líneas 2390-2430. Hay sitio para hasta ocho códigos en cada bloque.

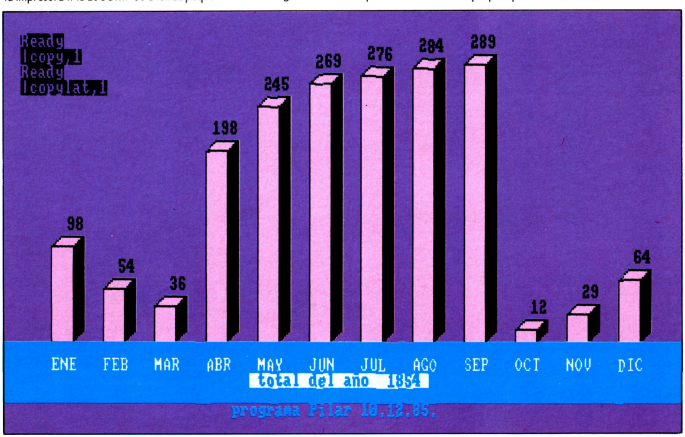
Para usar HOBBYCOP en programas propios, hay que añadir:

3085 RUN **«¡nombre.de.tu.programa»**

a HOBBYCOP. Entonces graba HOBBYCOP en frente de tu programa, el cual ahora puede usar COPY y COPYLAT sin haber perdido más de 443 bytes de memoria.

No olvides que las copias son monocromáticas. Si tienes muchos colores en la pantalla, sólo uno puede salir en blanco y los demás en negro. Tal vez puedes usar algo como la técnica de procesar la pantalla a sólo dos colores que se ha visto en el ejemplo gráfico de COPION.BAS. **«What you see is what you get»** es el dicho inglés. Es mejor ver en la pantalla exactamente lo que va a salir en el papel.

Quiero dar gracias a mis amigos *Paco y Juan de Portalo* aquí en Badajoz, colaboradores de MICROHOBBY Semanal, para el uso de su impresora Star Gemini. Paco es uno de los dos Paco pacenses inmortales por su éxito **«BUGABOO»** —la pulga. De él aprendí este método tan hermoso de automodificación en código máquina, donde el programa se escribe a sí mismo. Las etiquetas AUTOA y AUTOC en el programa Ensamblador señalan a tales autoamodificaciones, sin las cuales el programa habría sido algo como 30 por 100 más largo. Y gracias, también, a mi amigo Angel Hernández del Instituto Reino Aftasí en Badajoz, no sólo por el uso de la Indescomp 80 y la Seikosha GP550A, sino también por su bonito programa de geografía vista desde el espacio, que he usado en preparar ejemplos para este artículo.



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

BATTLE OF THE PLANETS

UNETE AL "COMANDO G" EN SU ULTIMA AVENTURA CONTRA ZOLTAR EN UN MARAVILLOSO JUEGO REALIZADO CON UNOS GRAFICOS TRIDIMENSIONALES QUE HAN DE SER VISTOS PARA SER CREIDOS.

BATTLE OF PLANETS ES EL PROGRAMA SELECCIONADO PARA EL

CAMPEONATO INTERNACIONAL DE JUEGOS DE ORDENADOR.

CONSIGUE LA MAXIMA PUNTUACION ESPAÑOLA Y PARTICIPA EN LA FINAL DE LONDRES.
(LAS BASES DEL CONCURSO ESTAN EXPLICADAS EN EL PROGRAMA)

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

TORNADO LOW LEVEL

El telele *(TLL)*, no es una contracción nerviosa espasmódica como todo el mundo piensa, sino que en el mundo del software, representa las siglas del Tornado Low Level, el juego co nel scroll más rápido del Amstrad.

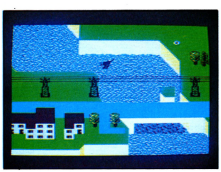
El Tornado, es uno de los aviones de caza europeos, con mayor poder de armamento y operatividad a baja cota. Comparable en muchos aspectos a los míticos Phantom, F-18 de McDonnall Douglas y los modelos más sofisticados de Mirage.

A los mandos de nuestro avión supersónico, afrontamos una misión de reconocimiento y localización de objetivos, en terreno enemigo.

Siendo esta una misión ultrasecreta, en la cual el factor sorpresa es determinante para el éxito, siempre debemos volar a baja altura para no ser detectados por el radar enemigo.

Como en todo avión moderno, la zona de combate, se encuentra perfectamente reproducida en la memoria de nuestro ordenador de ruta, el cual nos indica en todo momento el estado del combustible y las bombas que transportamos, así como el momento más oportuno para regresar a nuestra base a repostar.

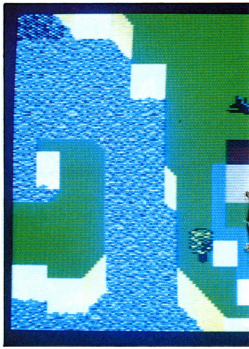
Compatible: CPC/464, CPC/664 y CPC/6128



Disponemos de dos velocidades de vuelo: reconocimiento a ras de tierra, a velocidad moderada y vuelo supersónico a cota baja, aerodinámico de las alas, consiguiendo una mayor penetración aerodinámica.

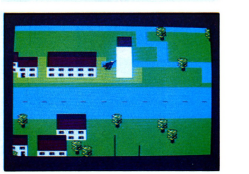
Tornado **Low Level** fue el juego que dio a conocer a la firma Vortex en el mundo del Spectrum; como pionero en los programas de la casa, no es un producto de ultimísima actualidad, pero aunque la versión para **Amstrad** ha tardado su tiempo en aparecer, sigue teniendo plena vigencia.

Además de ser pionero en los productos de la firma, TLL introdujo en su tiempo un nuevo concepto del espacio y la perspectiva en el mundo del software. Aprovechando que debemos circular por un mapa a distintas alturas, Vortex hace uso de una perspectiva plana, fácil y muy efectiva para simular las tres dimensiones.

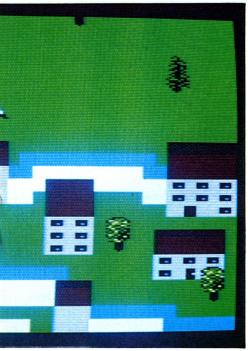


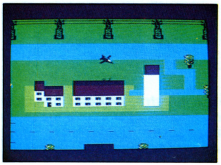
En la publicidad del juego podemos leer **«TLL tiene el más rápido y suave efecto de Scroll, jamás conseguido en la historia del Amstrad».**

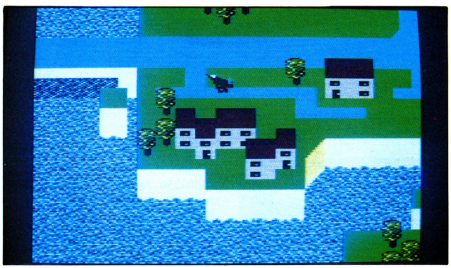
Afirmación que no es nada exagerada, pues cuando el Tornado varía el ángulo de las alas y vuela a velocidad supersónica, el paisaje pasa ante vuestros ojos a una velocidad increíble.

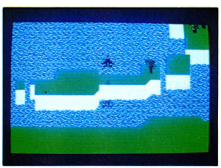


Mister JOYSTICK









Las técnicas de programación utilizadas para conseguir estos Scrolls a gran velocidad, son muy elogiables, teniendo en cuenta los 16 Kb. de pantalla del **Amstrad** y el hecho de que el avión puede moverse en ocho direcciones diferentes.

Para lo cual se han utilizado rutinas de Scroll con las siguientes características:

Scroll horizontal de izquierda a derecha y derecha a izquierda, y vertical de arriba a abajo y abajo arriba.

Diagonal ángulo superior izquierdo, inferior derecho e inferior derecho superior izquierdo.

Diagonal ángulo inferior izquierdo, superior derecho y superior derecho inferior izquierdo.

Con esta profusión de rutinas, se consigue un control del aparato de 360 grados, permitiendo cualquier maniobra real de combate.

Las misiones de caza que debemos realizar sobre objetivos en tierra, son transmitidas desde la división de inteligencia al ordenador de abordo, el cual nos da informes de situación y estado del aparato, con un simple toque de tecla.

Para cumplir cada una de las misiones que nos dicta el ordenador, debemos destruir los cinco objetivos fijados en el mapa de inteligencia.

Las dificultades con que nos encontramos son; la autonomía del aparato, el número limitado de proyectiles, y los obstáculos urbanos. Tendidos eléctricos, antenas de comunicaciones, edificios altos, etc, son muy peligrosos para el vuelo a baja cota, pudiendo chocar con ellos en cualquier momento.

Es importante repostar en los momentos oportunos, para lo cual debemos aterrizar en las pistas distribuidas por el mapa, donde se repondrán las existencias de proyectiles y combustible, permitiéndonos seguir con la misión en curso.

Nuestro principal enemigo es el tiempo. Si en plazo marcado por el ordenador de ruta no hemos destruido la totalidad de los objetivos y tomado tierra en nuestra base, somos localizados y destruidos.

Un programa de acción, en el que superar nuestro propio récord constituye el principal aliciente.

¡Buena caza!

Análisis CRONOTEST

Se puede aprender informática mientras uno está divirtiéndose, o a lo mejor se debe. En este sentido, Amstrad Análisis os propone un pequeño juego en el que os demostraréis a vosotros mismos vuestra rapidez de reflejos y aprenderéis, casi sin daros cuenta, cómo se usan los comandos de manejo de interrupciones del Basic AFTER y REMAIN.

El comando AFTER permite cronometrar un período de tiempo, y una vez pasado éste, desviar el programa hacia una subrutina. Y el comando REMAIN desactiva dicho cronómetro, dando además el tiempo que quedaba para cumplirse el plazo. Veámoslo con el ejemplo.

50 Establece el modo de pantalla. Inicializa la variable (aS), y el control (t).

60-90 Presentan el texto en la pantalla.

100 Va a la subrutina que dibuja un cuadrado en la pantalla.

110 Para el programa hasta que se pulsa ENTER.

120 Instrucciones.

130 La función RANDOMIZE TIME mejora la elección de números aleatorios, al hacer que éstos dependan del tiempo que lleva conectado el ordenador.

140 Asigna las operaciones aritméticas a las cadenas respectivas (OP$).

150-160 Elige dos números aleatorios, los multiplica por 1000 y los redondea con dos decimales. Esto se hace para que el número esté entre 1 y 999.99.

170 Almacena en *(a)* el mayor y en *(b)* el menor de los dos números anteriores.

180 Elige un número aleatorio entre 1 y 4 para poder seleccionar la operación aritmética (sel).

190 Conecta el temporizador número 1 para que haga un GOSUB a la línea 360 cuando pasen 30 segundos *(1500*0.02=30)*.

200 Presenta en pantalla la operación a realizar.

210 Inicia el bucle que detecta que se tiene el resultado. Esto se podría hacer con la instrucción INPUT, pero entonces si se acabase el tiempo, no se ejecutaría el GOSUB hasta que se pulsase ENTER.

220 Si se ha ejecutado el GOSUB *(t)* valdrá cero y saltará a la línea 330.

230 Asigna a *(aS)* la tecla pulsada. El bucle hace que se ignore cualquier otra tecla que no sea la *(s)*

240-270 Calcula el resultado real, dependiendo de la operación, que a su vez depende del valor de (sel).

280 Fin del bucle.

290 Pregunta el resultado si no se ha cumplido el plazo, y lo almacena en la variable (respuesta).

300 Se imprime si se ha acertado o si se ha fallado.

310 Imprime el resultado al lado de la operación.

320 Imprime el tiempo que quedaba para responder, calculado en segundos *(multiplicándolo por 0.02)*.

330-350 Pregunta si se quiere volver a probar, si se responde que sí, vuelve al principio, y si no acaba el programa.

360 Empieza la subrutina invocada por el comando AFTER.

370 Imprime que se ha pasado el tiempo.

380 Imprime el resultado correcto.

390 Pone el control a cero, para que al retornar al programa principal no pida el resultado otra vez.

400 Retorna al punto donde se dejó el programa principal al acabarse el tiempo.

410-470 Dibuja un cuadrado en la pantalla para escribir dentro la operación aritmética, y retorna al programa principal.

Si se quiere cambiar el tiempo de respuesta, basta con calcular el tiempo deseado multiplicando los segundos por 0.02, y sustituyendo el valor obtenido en la línea 190 después de AFTER. Para significar las operaciones, se pueden quitar los decimales quitando de las líneas 150, 160, 240, 250, 260 y 270 los doses que hay después de las comas.

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

Te ofrecemos algo "muy Especial"

En el mes de septiembre nació una ***AMSTRELLA*** *que vino a demostrar que los* ***Amstrad*** *estaban ahí y había que contar con ellos.*

Hoy, miles de personas nos dan la razón.

Por este motivo, y después de los 6 primeros meses de andadura juntos con nuestros lectores, ha vuelto a suceder algo muy importante: ha nacido una ***AMSTRELLA MUY ESPECIAL.***

¡Ya está a la venta en tu quiosco!
por sólo ***350*** *ptas.*

Relleno este cupón y envíalo a: **HOBBY PRESS, S. A.** Apartado de Correos 232. Alcobendas. Madrid.
Nombre _____ Apellidos _____ Edad _____
Localidad _____ Provincia _____ C. Postal _____
Teléfono _____ Profesión _____
¿Eres suscriptor de **MICROHOBBY AMSTRAD?** Sí ☐ No ☐
DESEO RECIBIR
☐ El Especial de **MICROHOBBY AMSTRAD** n.º 1 al precio de 350 ptas (IVA incluido)
FORMA DE PAGO
☐ Talón bancario adjunto a nombre de **HOBBY PRESS, S. A.**
☐ Mediante Tarjeta de Crédito, N.º _____ Fecha de caducidad _____
☐ Contra reembolso (supone 75 ptas. de gastos de envío).
Fecha y firma:

HOBBY PRESS
Para gente inquieta.

LAS LEYES DE MURPHY

Tarde o temprano, todo el mundo se pregunta en un alarde de sentido cumún cómo es posible que en la tarea de desarrollar un programa, aparentemente pristina y clara como el cristal, todo se desarrolle lentamente, por el camino más difícil posible y ocurran sin parar fenómenos inexplicables, a los cuales el programador, repleto de la Ira del Señor, culpa al ominoso azar, a la implacable Naturaleza y al Gobierno (ya en el culmen de la desesperación). No obstante, el calumniado azar no tiene vela en este entierro, para nada.

La verdadera razón de los parafenómenos informáticos obedece a una conspiración fríamente calcualda, y que el Doctor Murphy descubrió en su tesis titulada «Sobre la perversidad de los objetos inanimados» (Editorial Malamilk, de la Universidad de Hardware), la cual resumimos en sus puntos principales con el doble objeto de aclarar al personal las causas de sus cuitas, convencerles de la inevitabilidad de las mismas, y limitar en lo posible el creciente número de ordenadores estrellados contra el suelo en un arrebato de pasión.

Nota de redacción: *La persona que encuentre la manifiesta contradicción que se oculta en el último párrafo, pues bien. Si además nos lo dice a nosotros, le regalamos un precioso conjunto Sadomasok, creado por Cutrefiel, en cuero negro repujado para que se vista adecuadamente a la hora de programar. Mola ¿eh?*

1. Ley de Murphy

Si algo puede ir mal en tu programa, puedes apostar a que irá mal.

2. Teorema de Patrick

Si tu programa funciona a la primera, seguro que estás empleando el algoritmo y tal vez el ordenador equivocado.

3. Constante de Skinner

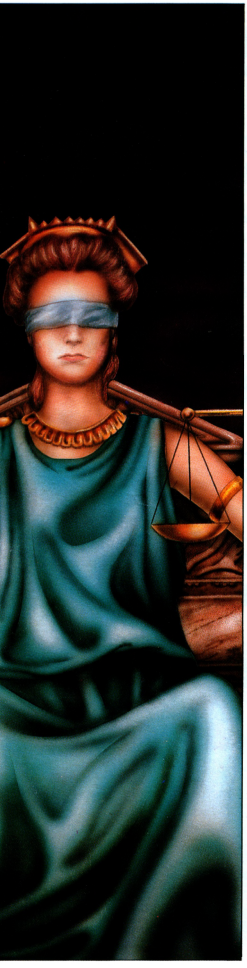
Es la cantidad de líneas pertenecientes a tu programa, tales que, añadidas al mismo, impiden que quepa en la memoria y, sustraídas de él, no permiten que funcione como habías previsto en un principio.

4. Postulados de las lenguas vivas

La pericia en el arte de programar es directamente proporcional al número de injurias que uno profiere durante el desarrollo del programa, e inversamente proporcional al cuadrado de las distancias que se recorren de casa al parque más cercano para **«estirar las piernas»**.

5. Ley de Flape sobre la perversidad de los programas concluidos

Todo programa, prescindiendo de su propósito, estructura y configuración, será concluido de la forma más insospechada y confusa posible, por razones completamente oscuras.



6. Axioma de Allen

Cuando todo falla, es el momento de consultar el manual que naturalmente, no tenemos idea de dónde puede encontrarse oculto.

7. Principio de las subrutinas dispersas

Al depurar un programa, la accesibilidad de una subrutina crítica para su funcionamiento, es inversamente proporcional al número de veces que dicha rutina desfila por delante de nuestros ojos al listar el programa en su búsqueda.

8. Corolario de compensación

La estructura de un programa puede considerarse un éxito si no más del 50 por 100 de las líneas originales deben modificarse para obtener cierto parecido con el resultado deseado.

9. Ley de Gumperson

La probabilidad de que ocurra un determinado suceso es inversamente proporcional a su importancia y al deseo que tengamos de que suceda.

10. En busca del material perdido

Los discos necesarios para el programa de ayer, deben pedirse no más tarde de mañana al mediodía.

11. Principio principal

Por definición, cuando uno programa en el dominio de lo desconocido, no sabe lo que se va a encontrar.

12. Regla de Ketterin (la más sencilla)

No funciona, pero no funciona por una razón distinta de la que uno piensa que no funciona.

13. Factor de futilidad

Ningún programa es nunca un completo fracaso: puede servir siempre como un mal ejemplo.

14. Ley de Anderson

Nunca se perderá un programa del que uno tenga copias.

15. Principio de la gravitación selectiva

Cuando el ordenador cae al suelo, inevitablemente aterriza por donde el daño puede ser mayor.

16. Teorema del cálculo mental

1. Si puede cometerse un error en los cálculos, desde luego que ocurrirá y de tal forma, que hay que rehacer todas las operaciones.

2. Todos los valores de las constantes, a fin de cuentas resultan ser variables.

3. En todos los cálculos, el valor que se creía más correcto es el causante de todos los errores.

4. La coma decimal se las arregla para colocarse por su cuenta en el peor sitio.

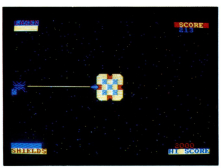
SPACE BASE

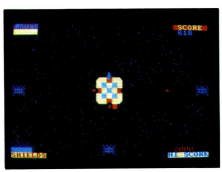
Los programas de ataques espaciales mando de naves galácticas y espantosos confrontaciones alienígenas nunca pasan de moda. Siempre renace de sus cenizas, como un ave Fénix simpática, para divertirnos.

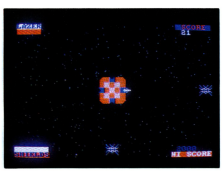
El año es 2036 después de Cristo. Después de décadas de costosos intentos en vidas y naves, el hombre ha sido capaz de establecer su primera base espacial en el asteroide, belt.

El aseteroide es rico en oro y minerales claves para la supervivencia de la civilización terrícola.

Sin embargo, los nativos de Andrómeda también codician los mismos minerales y están dispuestos a conseguirlos por la vía más rápida, la guerra. Debes defender la base contra sus enemigos.







VARIABLES PRINCIPALES

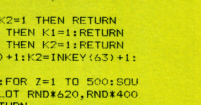
A(5)	Posición de las naves atacantes
B(5)	Posición de las bombas
SC	Score
LV	Nivel
SH	Combustible
LZ	Energía láser
LZ1	¿Se acabó el láser?
LZ2	Contador
LZC	Color del láser
A1$	
A2$	Los alienígenas
A3$	
LB	Posición del láser
R	Variable aleatoria
K1	Cierto si se pulsa Z o joystick a la izquierda
K2	Cierto si se pulsa X o joystick a la derecha

ESTRUCTURA DEL PROGRAMA

220-250	Datos para las bombas
260-340	Datas para las naves alienígenas
350-480	Datas para la base
490-640	Inicialización e instrucciones
650-880	Bucle principal. Inicializa la pantalla y, dependiendo del nivel, decide la velocidad y frecuencia de los ataques enemigos
890-960	Dirección de la bomba
970-1000	Coloca un alien en pantalla al azar
1010-1060	Mantiene la bomba en curso
1070-1080	¿Choque de la bomba con la base?
1090-1270	Mueve láser
1280-1370	Fuego del láser. Destruye bombas y naves
1380	Incrementa el score
1390	Incrementa el score
1400-1430	Examina el teclado y/o joystick
1440	Efectos de sonido

```
100 REM **************************
110 REM *                        *
120 REM *       SPACE BASE       *
130 REM *                        *
140 REM *     By Robin Nixon     *
150 REM *                        *
160 REM * (c) AMSTRAD SEMANAL     *
170 REM *                        *
180 REM *                        *
190 REM **************************
200 REM
210 SYMBOL AFTER 220:DEFINT A-Z
220 SYMBOL 220,0,60,24,60,60,60,24
230 SYMBOL 221,0,0,58,126,126,58,0,
0
240 SYMBOL 222,0,24,60,60,60,24,60,
0
250 SYMBOL 223,0,0,92,126,126,92,0,
0
260 SYMBOL 230,0,0,0,0,32,16,8,4
270 SYMBOL 231,0,0,0,0,129,66,36,24
280 SYMBOL 232,0,0,0,0,4,8,16,32
290 SYMBOL 233,34,17,15,1,1,15,17,3
4
300 SYMBOL 234,36,231,60,36,36,60,2
31,36
310 SYMBOL 235,68,136,240,128,128,2
40,136,68
320 SYMBOL 236,4,8,16,32,0,0,0,0
330 SYMBOL 237,24,0,0,0,0,0,0,0
340 SYMBOL 238,32,16,8,4,0,0,0,0
350 SYMBOL 240,15,31,63,127,255,255
,255,255
360 SYMBOL 241,255,255,255,255,255,
255,255,255
370 SYMBOL 242,129,195,231,255,255,
255,255,255
380 SYMBOL 243,240,248,252,254,255,
255,255,255
390 SYMBOL 244,255,127,63,31,31,63,
127,255
400 SYMBOL 245,255,254,252,248,248,
252,254,255
410 SYMBOL 246,255,255,255,255,127,
63,31,15
420 SYMBOL 247,255,255,255,255,255,
231,195,129
430 SYMBOL 248,255,255,255,255,254,
252,248,240
440 SYMBOL 249,165,90,165,90,90,165
,90,165
450 SYMBOL 250,24,24,60,24,126,24,2
55,24
460 SYMBOL 251,64,80,84,255,255,84,
80,64
470 SYMBOL 252,24,255,24,126,24,60,
24,24
480 SYMBOL 253,2,10,42,255,255,42,1
0,2
490 CALL &BC02:INK 0,0:INK 2,2:BORD
ER 0:DIM A(5),B(5):HS=2000
500 MODE 1:INK 2,26:SC=0:LV=1:SH=56
:PEN 2:PAPER 3:CLS:LOCATE 16,2:PRIN
T"SPACE BASE"
510 PEN 1:PRINT:PRINT:PRINT"Eres el
 capitan de la primera base"
520 PRINT"espacial terrestre en el
asteroide belt."
530 PRINT:PEN 0:PRINT"Tu estacion a
ctua como una base de"
540 PRINT"reparacion y repostaje pa
ra naves."
560 PRINT:PEN 2:PRINT"Sin embargo,
naves hostiles de Andromeda"
570 PRINT"se acercan a belt y no pa
raran hasta"
580 PRINT"acabar con toda competenc
ia de la Tierra";
600 PRINT:PEN 1:PRINT"Tu mision es
defender la base."
620 PRINT:PEN 0:PRINT"Usa Z  X para
 controlar tu laser y"
630 PRINT"pulsa ENTER para disparar
."
640 PAPER 2:PEN 0:LOCATE 14,24:PRIN
T"Pulsa una tecla";:WHILE INKEY$<>"
" OR JOY(0)>0:WEND:WHILE INKEY$=""
AND JOY(0)=0:WEND:INK 2,2
650 WHILE SH>-1:MODE 0:PAPER 3:CLS:
PEN 2:LOCATE 7,13:PRINT"LEVEL";LV;:
FOR Z=1 TO 5000:NEXT:MODE 1:PAPER 0
:CLS:LZ=40:LZ1=0:LZ2=0:LZC=1:HT=0:E
RASE A,B:DIM A(5),B(5)
660 GOSUB 1440:PEN 3:PAPER 2:LOCATE
 1,1:PRINT"LAZER";:PEN 1:PAPER 3:LO
CATE 33,1:PRINT" SCORE ";:PEN 3:PAP
ER 1:LOCATE 1,25:PRINT"SHIELDS";:PE
N 2:PAPER 1:LOCATE 32,25:PRINT"HI S
CORE";:PAPER 1:LOCATE 1,2:PRINT STR
ING$(5,32);:PAPER 2:LOCATE 1,24
670 PRINT STRING$(7,32);:PEN 2:PAPE
R 0:LOCATE 33,2:PRINT SC;:PEN 3:LOC
ATE 32,24:PRINT HS;:FOR Z=57 TO SH+
1 STEP -1:MOVE Z*2,16:DRAW Z*2,31,0
:NEXT
680 PAPER 0:PEN 1:LOCATE 18,11:PRIN
T CHR$(240);CHR$(241);CHR$(242);CHR
$(241);CHR$(243);
690 LOCATE 18,12:PRINT STRING$(5,24
1);:LOCATE 18,13:PRINT CHR$(244);ST
RING$(3,241);CHR$(245);
700 LOCATE 18,14:PRINT STRING$(5,24
1);:LOCATE 18,15:PRINT CHR$(246);CH
R$(241);CHR$(247);CHR$(241);CHR$(24
8);
710 PEN 2:PAPER 1:LOCATE 19,12:PRIN
T CHR$(249);:LOCATE 21,12:PRINT CHR
$(249);:LOCATE 20,13:PRINT CHR$(249
);:LOCATE 19,14:PRINT CHR$(249);:LO
CATE 21,14:PRINT CHR$(249);
720 A1$=CHR$(230)+CHR$(231)+CHR$(23
2):A2$=CHR$(233)+CHR$(234)+CHR$(235
):A3$=CHR$(236)+CHR$(237)+CHR$(238)
730 PEN 3:PAPER 0:LOCATE 20,11:PRIN
T CHR$(242);:LOCATE 18,13:PRINT CHR
$(244);:LOCATE 22,13:PRINT CHR$(245
);:LOCATE 20,15:PRINT CHR$(247);
740 HT=0:LB=1:IP=0:PEN 3:PAPER 2:GO
SUB 1170:PEN 2:GOSUB 1220:PAPER 0
750 WHILE HT<LV*25+25 AND SH>-1
760 R=RND*(10-LV):IF R<1 THEN GOSUB
 890
770 FOR X=1 TO 4:IF B(X)=0 THEN GOT
O 780 ELSE GOSUB 1010
780 GOSUB 1400:NEXT
790 GOSUB 1090:R=RND*(10-LV):IF R<1
 THEN GOSUB 910
800 IF INKEY(18)=0 OR JOY(0)>15 THE
N GOSUB 1280
810 LZ2=LZ2+1:IF LZ2=6 THEN LZ2=0:I
F LZ<39 THEN LZ=LZ+1:MOVE LZ*2,368:
DRAW LZ*2,382,LZC
820 IF LZ<0 THEN LZ1=1:LZC=3
830 IF LZ1=1 AND LZ>14 THEN PAPER 1
:LOCATE 1,2:PRINT STRING$(2,32);:LZ
```

```
C=1:LZ1=0:PAPER 0
840 WEND:LV=LV+1
850 PEN 2:PAPER 0:FOR Z=SH TO 0 STE
P-1:MOVE Z*2,16:DRAW Z*2,31,0:SC=SC
+LV:LOCATE 33,2:PRINT SC;:SOUND 1,2
0,1,7:FOR Z1=1 TO 25:NEXT Z1,Z
860 WEND
870 FOR Z=1 TO 1000:INK RND*3,RND*2
6:SOUND 1,7,1,7:NEXT:INK 0,0:INK 1,
24:INK 2,2:INK 3,6:MODE 0:PAPER 3:P
EN 1:CLS:LOCATE 6,13:PRINT"GAME  OV
ER";:FOR Z=1 TO 5000:NEXT:IF SC>HS
THEN HS=SC
880 GOTO 500
890 R=RND*3+1
900 IF A(R)=0 THEN A(R)=1:PEN 2:FOR
 Z=500 TO 0 STEP -23:SOUND 1,Z,1,6:
NEXT:ON R GOTO 970,980,990,1000
910 R=1:WHILE R<5 AND (A(R)=0 OR B(
R)>0):R=R+1:WEND:IF R=5 THEN RETURN
920 FOR Z=200 TO 400 STEP 20:SOUND
1,Z,1,6:NEXT:ON R GOTO 930,940,950,
960
930 B(R)=4:RETURN
940 B(R)=36:RETURN
950 B(R)=22:RETURN
960 B(R)=4:RETURN
970 LOCATE 19,1:PRINT A1$;:GOSUB 14
00:LOCATE 19,2:PRINT A2$;:GOSUB 140
0:LOCATE 19,3:PRINT A3$;:RETURN
980 LOCATE 37,12:PRINT A1$;:GOSUB 1
400:LOCATE 37,13:PRINT A2$;:GOSUB 1
400:LOCATE 37,14:PRINT A3$;:RETURN
990 LOCATE 19,23:PRINT A1$;:GOSUB 1
400:LOCATE 19,24:PRINT A2$;:GOSUB 1
400:LOCATE 19,25:PRINT A3$;:RETURN
1000 LOCATE 1,12:PRINT A1$;:GOSUB 1
400:LOCATE 1,13:PRINT A2$;:GOSUB 14
00:LOCATE 1,14:PRINT A3$;:RETURN
1010 ON X GOTO 1020,1030,1040,1050
1020 LOCATE 20,B(X):GOSUB 1080:IF B
(X)+1=11 THEN GOTO 1070 ELSE B(X)=B
(X)+1:PEN 3:LOCATE 20,B(X):GOTO 106
0
1030 LOCATE B(X),13:GOSUB 1080:IF B
(X)-1=22 THEN GOTO 1070 ELSE B(X)=B
(X)-1:PEN 3:LOCATE B(X),13:GOTO 106
0
1040 LOCATE 20,B(X):GOSUB 1080:IF B
(X)-1=15 THEN GOTO 1070 ELSE B(X)=B
(X)-1:PEN 3:LOCATE 20,B(X):GOTO 106
0
1050 LOCATE B(X),13:GOSUB 1080:IF B
(X)+1=18 THEN GOTO 1070 ELSE B(X)=B
(X)+1:PEN 3:LOCATE B(X),13
1060 PRINT CHR$(219+X);:RETURN
1070 SOUND 1,0,15,5,0,0,5:FOR Z=0 T
O 24 STEP 6:INK 1,Z:INK 2,Z+2:INK 3
,Z/6:FOR Z1=1 TO 30:NEXT Z1,Z:INK 2
,2:INK 3,6:MOVE SH*2,16:DRAW SH*2,3
0,0:SH%=SH%-1:B(X)=0:PEN 2:GOSUB 12
20:RETURN
1080 PEN 0:PRINT" ";:RETURN
1090 GOSUB 1400:IF INKEY(71)=-1 AND
 INKEY(63)=-1 AND JOY(0)<>4 AND JOY
(0)<>8 THEN IP=0:K1=0:K2=0:RETURN
1100 IF IP=1 THEN RETURN
1110 GOSUB 1400
1120 IF K1=0 AND K2=0 THEN RETURN
1130 IP=1:PEN 3:PAPER 0:GOSUB 1170:
PEN 0:GOSUB 1220:IF K1=1 THEN LB=LB
-1:IF LB=0 THEN LB=4
1140 IF K2=1 THEN LB=LB+1:IF LB=5 T
HEN LB=1
1150 K1=0:K2=0
1160 PEN 3:PAPER 2:GOSUB 1170:PEN 2
:GOSUB 1220:RETURN
1170 ON LB GOTO 1180,1190,1200,1210
1180 LOCATE 20,11:PRINT CHR$(242):R
ETURN
1190 LOCATE 22,13:PRINT CHR$(245):R
ETURN
1200 LOCATE 20,15:PRINT CHR$(247):R
ETURN
1210 LOCATE 18,13:PRINT CHR$(244):R
ETURN
1220 ON LB GOTO 1230,1240,1250,1260
1230 PAPER 0:LOCATE 20,10:GOTO 1270
1240 PAPER 0:LOCATE 23,13:GOTO 1270
1250 PAPER 0:LOCATE 20,16:GOTO 1270
1260 PAPER 0:LOCATE 17,13:GOTO 1270
1270 PRINT CHR$(249+LB):RETURN
1280 GOSUB 1400:IF LZ1=1 THEN RETUR
N ELSE MOVE LZ*2,368:DRAW LZ*2,382,
0:LZ=LZ-1:FOR Z=30 TO 10 STEP-3:SOU
ND 1,Z,1,5:NEXT:ON LB GOSUB 1300,13
20,1340,1360
1290 PEN 2:GOSUB 1220:RETURN
1300 FOR Z=1 TO 4:GOSUB 1400:MOVE 3
10,260:DRAW 310,362,Z:NEXT:IF B(LB)
>0 THEN SOUND 1,0,20,7,0,0,15:LOCAT
E 26,B(LB):GOTO 1380
1310 IF A(LB)=0 THEN RETURN ELSE SO
UND 1,0,20,7,0,0,1:PEN 0:FOR Z=1 TO
 3:LOCATE 19,7:PRINT"   ";:GOSUB 14
00:NEXT:GOTO 1390
1320 FOR Z=1 TO 4:GOSUB 1400:MOVE 3
70,200:DRAW 580,200,Z:NEXT:IF B(LB)
>0 THEN SOUND 1,0,20,7,0,0,15:LOCAT
E B(LB),13:GOTO 1380
1330 IF A(LB)=0 THEN RETURN ELSE SO
UND 1,0,20,7,0,0,1:PEN 0:FOR Z=12 T
O 14:LOCATE 37,Z:PRINT"   ";:GOSUB
1400:NEXT:GOTO 1390
1340 FOR Z=1 TO 4:GOSUB 1400:MOVE 3
10,140:DRAW 310,42,Z:NEXT:IF B(LB)>
0 THEN SOUND 1,0,20,7,0,0,15:LOCATE
 20,B(LB):GOTO 1380
1350 IF A(LB)=0 THEN RETURN ELSE SO
UND 1,0,20,7,0,0,1:PEN 0:FOR Z=23 T
O 25:LOCATE 19,Z:PRINT"   ";:GOSUB
1400:NEXT:GOTO 1390
1360 FOR Z=1 TO 4:GOSUB 1400:MOVE 2
50,200:DRAW 50,200,Z:NEXT:IF B(LB)
>0 THEN SOUND 1,0,20,7,0,0,15:LOCAT
E B(LB),13:GOTO 1380
1370 IF A(LB)=0 THEN RETURN ELSE SO
UND 1,0,20,7,0,0,1:PEN 0:FOR Z=12 T
O 14:LOCATE 1,Z:PRINT"   ";:GOSUB 1
400:NEXT:GOTO 1390
1380 PEN 0:PRINT " ";:B(LB)=0:SC=SC
+2*LV:PEN 2:PAPER 0:LOCATE 33,2:PRI
NT SC;:HT=HT+1:RETURN
1390 A(LB)=0:SC=SC+1*LV:PEN 2:PAPER
 0:LOCATE 33,2:PRINT SC;:HT=HT+1:RE
TURN
1400 IF K1=1 OR K2=1 THEN RETURN
1410 IF JOY(0)=4 THEN K1=1:RETURN
1420 IF JOY(0)=8 THEN K2=1:RETURN
1430 K1=INKEY(71)+1:K2=INKEY(63)+1:
RETURN
1440 PAPER 0:CLS:FOR Z=1 TO 500:SOU
ND 1,500-Z,1,5:PLOT RND*620,RND*400
,RND*2+1:NEXT:RETURN
```



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

MINI OFFICE

© DATABASE PUBLICATIONS

PROGRAMA DEL AÑO EN INGLATERRA

FINALIST BRITISH MICRO COMPUTING AWARDS 1985

1 PROCESADOR DE TEXTOS

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

2 HOJA DE CALCULO

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

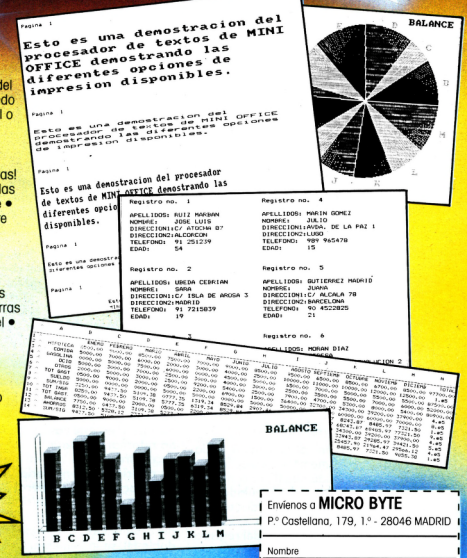
3 GRAFICOS

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

4 BASE DE DATOS

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

*En Castellano
*Servimos en 48 Horas



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

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

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

Nombre
Apellidos
Dirección
Población
D.P. Teléfono

Deseo que me envíen ☐ ejemplar/es del programa **MINI OFFICE**

PARA EL MICROORDENADOR SEÑALADO

☐ AMSTRAD ☐ COMMODORE ☐ SPECTRUM
☐ AMSTRAD VERSION DISCO
Sin gastos de envío

INCLUYO TALON NOMINATIVO ☐
CONTRA-REEMBOLSO ☐

Pedidos por teléfono
91 - 442 54 33/44

INSTRUCCIONES DE CAMBIO, TRANSFERENCIA Y BUSQUEDA (y III)

Terminaremos hoy el estudio de este bloque de órdenes refiriéndonos a las instrucciones de búsqueda que constituyen la última parte de este capítulo.

F. J. Frontán

Su misión consiste en la comparación de un byte determinado con el contenido del acumulador; si la comparación es positiva, es decir, el byte leído de una posición de memoria y el contenido del acumulador es el mismo, entonces se fija un bit de condición a 1. Si la comparación es negativa, o sea, el contenido del acumulador difiere del contenido de la posición de memoria, entonces el bit de condición permanece inalterado.

La primera instrucción de búsqueda que estudiaremos se representa como indicamos a continuación:

CPI

(Comparar-Incrementar)

Al ejecutarla, el contenido de la posición de memoria especificada por el contenido de HL, se compara con el contenido del acumulador. En caso de comparación verdadera, se fija un bit de condición a 1; ese bit de condición es el flag Z del registro F. Después des esto, se incrementa HL y se decrementa BC.

Veremos algunos ejemplos que nos aclararán el funcionamiento de dicha instrucción.

Supongamos que los registros HL, BC y el acumulador contienen los siguientes valores:

HL - #7000 (HL) - 45 BC - #0001
A - 45

Después de la ejecución de CPI obtendremos el siguiente resultado:

HL - #7001 (HL) - 45 BC - #0000
A - 45

y además el flag Z del registro F estará puesto a 1 ya que la comparación ha resultado verdadera.

Programa 1

En el programa número 1 podemos ver un ejemplo práctico de actuación de CPI. En primer lugar cargamos el registro doble HL con el valor #8000, a continuación cargamos en el acumulador el valor 255 y luego ejecutamos CPI; si el contenido de la posición de memoria #8000 es de 255, o sea, el mismo valor que el contenido en el acumulador, entonces el flag Z se pondrá a 1, por lo que nos imprimirá una 'S' en pantalla; si, por el contrario, el contenido de esa posición de memoria es distinto del contenido del acumulador, entonces Z estará a 0, por lo que el programa nos imprimirá en pantalla una 'N'.

Programa 2

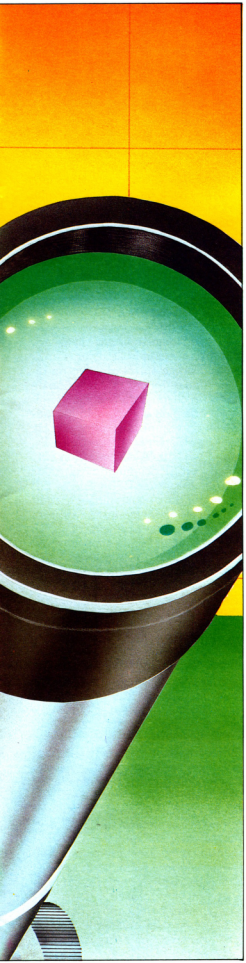
El programa número dos utiliza también la misma instrucción anterior, pero en este caso el programa no retornará al Basic hasta que no encuentre el valor 255 en alguna de las posiciones de memoria. Así pues si al ejecutarse la primera vez CPI, el contenido de esa posición de memoria no coincide con el contenido del acumulador, entonces el registro doble HL se incrementará en uno y así sucesivamente, hasta que el contenido de una posición de memoria sea igual al contenido del acumulador.

La siguiente instrucción perteneciente a este bloque se representa de la siguiente forma:

CPIR

(Comparar-Incrementar-Repetir)

Tras la ejecución el contenido de la dirección de memoria especificada por el registro doble HL se compara con el contenido del acumulador. Si la comparación es verdadera, es decir, si el contenido del acumulador y el contenido de la posición de memoria indicada por HL son iguales, en-



tonces se pone a 1 un bit de condición en este caso el flag Z del registro F. Además el registro doble HL se incrementa y el registro BC se decrementa.

Si el decremento de BC hace que este registro valga cero, o si el contenido del acumulador es igual al contenido de HL, entonces la instrucción finaliza su actuación.

Si el contenido del registro BC es distinto de cero y el contenido del acumulador diferente del contenido de HL, entonces BC se decrementa y se repite la instrucción.

Debemos tener en cuenta que si BC se pone a cero antes de ejecutarse la instrucción, ésta se repetirá 65535 veces si el contenido del acumulador es distinto al de cualquier posición de memoria. También debemos tener en cuenta que después de cada comparación se reconocerán las interrupciones.

Programa 3

El programa número tres nos muestra un ejemplo de cómo actúa CPIR. En primer lugar cargamos HL con el valor #8000, y BC con #1000, por lo tanto CPIR comparará el contenido del acumulador con el contenido de las posiciones de memoria que van desde la #8000 hasta la #9000, si el contenido de alguna de estas posiciones de memoria es igual al contenido del acumulador, entonces el programa nos imprimirá una 'S' en pantalla. Si, por el contrario, el contenido de ninguna de estas posiciones de memoria es idéntico al contenido del acumulador, entonces el programa nos imprimirá una 'N' en pantalla.

Programa 4

En el programa cuatro también se utiliza la instrucción anterior, pero, en este caso, lo que hace el programa no es buscar un valor igual al valor contenido en el acumulador, sino que aquí lo que hacemos es buscar una cadena de números.

En este caso la cadena que pretendemos encontrar es en primer lugar el carácter **'A'**; para ello cargamos en primer lugar el acumulador con ese valor, y en el momento en que éste sea encontrado, comprobaremos que la siguiente posición de memoria contenga el carácter **'B'**; si no es así, volveremos a buscar otra vez **'A'** y, cuando sea encontrado, miraremos si el carácter que le sigue es el **'B'**; si es así, el programa terminará imprimiéndonos en pantalla la posición de memoria donde se encuentra dicha cadena, en caso contrario seguiremos buscando hasta que el registro BC sea cero, o hasta que la cadena buscada sea encontrada.

Debemos tener en cuenta que hemos cargado BC con el valor 0 por lo que el programa mirará toda la memoria para ver si encuentra la cadena deseada.

La siguiente instrucción a estudiar se representa de la siguiente forma:

CPD

(Comparar-Decrementar)

Tras la ejecución el contenido de la dirección de memoria contenido en el registro doble HL, se compara con el contenido del acumulador. Si la comparación es verdadera, se pone a uno un bit de condición, el flag Z del registro F. Además se decrementan los registros dobles HL y BC.

Así, por ejemplo, si el contenido del registro par HL es #8000, la posición de memoria #8000 contiene 25, el acumulador contiene 25 y el registro doble BC contiene #0001, entonces después de la ejecución de CPD el registro par BC contendrá #0000, el contenido del registro doble HL será #7FFF y el falg Z del registro F se pondrá a 1. No se producirá ningún efecto sobre el acumulador ni sobre el contenido de la dirección de memoria #8000.

Programa 5

Una aplicación práctica de lo anterior podemos observarla en el programa número 5. En primer lugar, cargamos HL con la dirección #7000 y a continuación cargamos el acumulador con el valor 255, y ejecutamos CPD. Si el contenido de la dirección de memoria #7000 es igual al contenido del acumulador, el programa nos imprimirá una **'S'**, y si el contenido de esa dirección de memoria es distinto del contenido del acumulador, nos imprimirá una 'N'.

Tras la ejecución de este programa el registro doble HL contendrá el valor #6FFF.

Programa 6

El siguiente programa también utiliza la instrucción anterior, pero en este caso lo que haremos será comparar todas las posiciones de memoria con el contenido del acumulador.

Para ello cargamos el registro doble HL con el valor #FFFF y el registro BC con el valor 0, con lo que conseguiremos que el programa efectúe una comparación de las 65535 posiciones de memoria. A continuación cargamos el acumulador con el valor 22 y ejecutamos CPD. Cada vez que la comparación sea verdadera se imprimirá en pantalla un 1. Seguidamente miramos el contenido del registro BC; si éste es cero, terminará el programa; de lo contrario, efectuará la siguiente comparación hasta que dicho registro contenga el valor cero.

La última de las instrucciones de búsqueda es la que estudiaremos a continuación:

CPDR

(Comparar-Decrementar-Repetir)

Tras la ejecución de esta instrucción el contenido de la posición de memoria especificada por el registro par HL se compara con el contenido del acumulador.

En el caso de que la comparación sea verdadera, se fija un bit de condición, el flag Z del registro F. Además los registros pares BC y HL se decrementan. Si debido a este decremento el registro BC se iguala a cero, o bien si el contenido del acumulador es igual al contenido de la posición de memoria indicada por el registro HL, entonces se termina la instrucción y se pasa a la siguiente.

Si el registro BC no es igual a cero y si el contenido del acumulador es diferente del contenido de la posición de memoria indicada por HL, entonces BC se decrementa en dos y se repite la instrucción.

Debemos indicar que si BC está a cero cuando se inicia la instrucción, entonces ésta se repetirá durante 16K si no se encuentra ninguna posición de memoria cuyo contenido sea igual al del acumulador.

Diremos también que después de cada comparación de datos se reconocen las interrupciones.

Programa 7

El programa número siete nos muestra un ejemplo de actuación de dicha instrucción. El objetivo de dicha rutina es buscar una posición de memoria cuyo contenido sea idéntico al contenido del acumulador, para ello cargamos el registro par HL con el valor #FFFF y el registro BC con el valor 0. A continuación cargamos en el acumulador el valor 255. Al ejecutarse CPDR ésta se parará cuando el contenido de cualquier posición de memoria sea igual al contenido del acumulador.

El objetivo del siguiente programa es encontrar las instrucciones CALL que existen en la memoria de nuestro ordenador, y además nos dará las direcciones donde se efectúan dichos CALL. Así pues si dentro de la memoria se encuentra la instrucción siguiente:

CALL #A000

el programa nos mostrará en pantalla la dirección #A000 que es la dirección a la cual llama dicha instrucción.

Para ello cargamos en primer lugar los registros dobles HL y BC con #FFFF y 0 respectivamente, a continuación cargamos el acumulador con el valor #CD que es el correspondiente a la instrucción CALL, y seguidamente ejecutamos CPDR.

Si la posición de memoria chequeada contiene el mismo valor que el acumulador, entonces cargamos en el registro DE los valores contenidos en las dos posiciones de memoria siguientes con lo cual tendremos la dirección a la cual se llama y a continuación pasamos a imprimir ese valor en pantalla.

PROGRAMAS

```
                    10 ;PROGRAMA-1
                    20 ;
A000                30              ORG  #A000
A000  210080        40              LD   HL,#8000
A003  3EFF          50              LD   A,255
A005  EDA1          60              CPI
A007  2806          70              JR   Z,BIEN
A009  3E4E          80              LD   A,"N"
A00B  CD5ABB        90              CALL #BB5A
A00E  C9           100              RET
A00F  3E53         110 BIEN:        LD   A,"S"
A011  CD5ABB       120              CALL #BB5A
A014  C9           130              RET
```

ETIQUETAS

```
BIEN   A00F
```

```
                    10 ;PROGRAMA-2
                    20 ;
A000                30              ORG  #A000
A000  210080        40              LD   HL,#8000
A003  3EFF          50              LD   A,255
A005  EDA1          60 BUC:         CPI
A007  C8            70              RET  Z
A008  18FB          80              JR   BUC
```

ETIQUETAS

```
BUC    A005
```

```
                    10 ;PROGRAMA-3
                    20 ;
A000                30              ORG  #A000
A000  210080        40              LD   HL,#8000
A003  010010        50              LD   BC,#1000
A006  3EFF          60              LD   A,255
A008  EDB1          70              CPIR
A00A  2806          80              JR   Z,PAS
A00C  3E4E          90              LD   A,"N"
A00E  CD5ABB       100              CALL #BB5A
A011  C9           110              RET
A012  3E53         120 PAS:         LD   A,"S"
A014  CD5ABB       130              CALL #BB5A
A017  C9           140              RET
```

ETIQUETAS

```
PAS    A012
```

```
                    10 ;PROGRAMA-4
                    20 ;
A000                30              ORG  #A000
A000  210000        40              LD   HL,0
A003  010000        50              LD   BC,0
A006  3E41          60 BUC:         LD   A,"A"
A008  EDB1          70              CPIR
A00A  3E42          80              LD   A,"B"
A00C  EDA1          90              CPI
A00E  20F6         100              JR   NZ,BUC
A010  2B           110              DEC  HL
A011  2B           120              DEC  HL
A012  CD16A0       130              CALL HEXA
A015  C9           140              RET
A016  7C           150 HEXA:        LD   A,H
A017  CD1FA0       160              CALL PRHEX
A01A  7D           170              LD   A,L
A01B  CD1FA0       180              CALL PRHEX
A01E  C9           190              RET
A01F  F5           200 PRHEX:       PUSH AF
A020  E6F0         210              AND  #F0
A022  CB3F         220              SRL  A
A024  CB3F         230              SRL  A
A026  CB3F         240              SRL  A
A028  CB3F         250              SRL  A
A02A  CD30A0       260              CALL VHEX
A02D  F1           270              POP  AF
A02E  E60F         280              AND  #F
A030  C630         290 VHEX:        ADD  A,"0"
A032  FE3A         300              CP   ":"
A034  3802         310              JR   C,NOSUM
A036  C607         320              ADD  A,7
A038  CD5ABB       330 NOSUM:       CALL #BB5A
A03B  C9           340              RET
```

ETIQUETAS

```
BUC    A006  HEXA   A016  NOSUM  A038
PRHEX  A01F  VHEX   A030
```

```
                    10 ;PROGRAMA-5
                    20 ;
A000                30              ORG  #A000
A000  210070        40              LD   HL,#7000
A003  3EFF          50              LD   A,255
A005  EDA9          60              CPD
A007  2806          70              JR   Z.BIEN
A009  3E4E          80              LD   A,"N"
A00B  CD5ABB        90              CALL #BB5A
A00E  C9           100              RET
A00F  3E53         110 BIEN:        LD   A,"S"
A011  CD5ABB       120              CALL #BB5A
A014  C9           130              RET
```

ETIQUETAS

```
BIEN   A00F
```

```
                    10 ;PROGRAMA-6
                    20 ;
A000                30              ORG  #A000
A000  21FFFF        40              LD   HL,#FFFF
A003  010000        50              LD   BC,0
A006  3E16          60 BUC:         LD   A,22
A008  EDA9          70              CPD
A00A  CC12A0        80              CALL Z,UNO
A00D  78            90              LD   A,B
A00E  B1           100              OR   C
A00F  20F5         110              JR   NZ,BUC
A011  C9           120              RET
A012  3E31         130 UNO:         LD   A,"1"
A014  CD5ABB       140              CALL #BB5A
A017  C9           150              RET
```

ETIQUETAS

```
BUC    A006  UNA  A012
```

```
                    10 ;PROGRAMA-7
                    20 ;
A000                30              ORG  #A000
A000  21FFFF        40              LD   HL,#FFFF
A003  010000        50              LD   BC,0
A006  3EFF          60              LD   A,255
A008  EDB9          70              CPDR
A00A  C9            80              RET
```

```
                    10 ;PROGRAMA-6
                    20 ;
A000                30              ORG  #A000
A000  210A50        40              LD   HL,#500A
A003  010A00        50              LD   BC,10
A006  3ECD          60 BUC:         LD   A,#CD
A008  EDB9          70              CPDR
A00A  E5            80              PUSH HL
A00B  C5            90              PUSH BC
A00C  23           100              INC  HL
A00D  23           110              INC  HL
A00E  5E           120              LD   E,(HL)
A00F  23           130              INC  HL
A010  56           140              LD   D,(HL)
A011  CD1CA0       150              CALL HEXA
A014  C1           160              POP  BC
A015  E1           170              POP  HL
A016  2B           180              DEC  HL
A017  78           190              LD   A,B
A018  B1           200              OR   C
A019  20EB         210              JR   NZ,BUC
A01B  C9           220              RET
A01C  7A           230 HEXA:        LD   A,D
A01D  CD2AA0       240              CALL PRHEX
A020  7B           250              LD   A,E
A021  CD2AA0       260              CALL PRHEX
A024  3E0D         270              LD   A,13
A026  CD5DBB       280              CALL #BB5D
A029  C9           290              RET
A02A  F5           300 PRHEX:       PUSH AF
A02B  E6F0         310              AND  #F0
A02D  CB3F         320              SRL  A
A02F  CB3F         330              SRL  A
A031  CB3F         340              SRL  A
A033  CB3F         350              SRL  A
A035  CD3BA0       360              CALL VHEX
A038  F1           370              POP  AF
A039  E60F         380              AND  #F
A03B  C630         390 VHEX:        ADD  A,"0"
A03D  FE3A         400              CP   ":"
A03F  3802         410              JR   C.NOSUM
A041  C607         420              ADD  A,7
A043  CD5ABB       430 NOSUM:       CALL #BB5A
A046  C9                            RET
```

ETIQUETAS

```
BUC    A006  HEXA   A01C  NOSUM  A043
PRHEX  A02A  VHEX   A03B
```

PROGRAMA 1

```
10 REM * PROGRAMA 1 *
20 FOR N=&A000 TO &A015
30 READ A:SUMA=SUMA+A
40 POKE N.A
50 NEXT
60.IF SUMA<>&AEE THEN PRINT "ERROR
EN DATAS"
70 DATA 33.0.128,62,255.237.161
80 DATA 40,6,62,78,205,90,187
90 DATA 201.62.83.205,90,187,201
```

PROGRAMA 2

```
10 REM * PROGRAMA 2 *
20 FOR N=&A000 TO &A00A
30 READ A:SUMA=SUMA+A
40 POKE N.A
50 NEXT
60 IF SUMA<>&547 THEN PRINT "ERROR
EN DATAS"
70 DATA 33,0,128.62,255,237,161
80 DATA 200,24,251,0,0,0,0
```

PROGRAMA 3

```
10 REM * PROGRAMA 3 *
20 FOR N=&A000 TO &A018
30 READ A:SUMA=SUMA+A
40 POKE N.A
50 NEXT
60 IF SUMA<>&A2E THEN PRINT "ERROR
EN DATAS"
70 DATA 33.0.128.1,0,16,62
80 DATA 255,237,177,40,6,62,78
90 DATA 205.90,187.201,62.83,205
100 DATA 90,187,201,0,0,0,0
```

PROGRAMA 4

```
10 REM * PROGRAMA 4 *
20 FOR N=&A000 TO &A03C
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&1D35 THEN PRINT "ERROR
 EN DATAS"
70 DATA 33.0,0,1,0,0,62
80 DATA 65,237,177,62,66,237.161
90 DATA 32,246,43,43,205.22.160
100 DATA 201,124,205,31,160,125,205
110 DATA 31,160,201,245,230,240,203
120 DATA 63,203,63,203,63,203,63
130 DATA 205,48,160,241,230,15.198
140 DATA 48,254,58,56,2,198,7
150 DATA 205,90,187,201.0,0,0
```

PROGRAMA 5

```
10 REM * PROGRAMA 5 *
20 FOR N=&A000 TO &A015
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&A05 THEN PRINT "ERROR
EN DATAS"
70 DATA 33.0.112,62,255,237,169
80 DATA 40,6,62,78,205,90,187
90 DATA 201,62,83.205,90,187,201
```

PROGRAMA 6

```
10 REM * PROGRAMA 6 *
20 FOR N=&A000 TO &A018
30 READ A:SUMA=SUMA+A
40 POKE N.A
50 NEXT
60 IF SUMA<>&BA9 THEN PRINT "ERROR
EN DATAS"
70 DATA 33,255,255,1,0,0,62
80 DATA 22,237,169,204,18,160,120
90 DATA 177,32,245,201,62,49,205
100 DATA 90,187,201,0,0,0,0
```

PROGRAMA 7

```
10 REM * PROGRAMA 7 *
20 FOR N=&A000 TO &A00B
30 READ A:SUMA=SUMA+A
40 POKE N.A
50 NEXT
60 IF SUMA<>&5CC THEN PRINT "ERROR
EN DATAS"
70 DATA 33.255,255,1,0,0,62
80 DATA 255,237,185,201,0,0,0
```

PROGRAMA 8

```
10 REM * PROGRAMA 8 *
20 FOR N=&A000 TO &A047
30 READ A:SUMA=SUMA+A
40 POKE N.A
50 NEXT
60 IF SUMA<>&23CE THEN PRINT "ERROR
 EN DATAS"
70 DATA 33,10,80,1,10,0,62
80 DATA 205,237,185,229,197,35,35
90 DATA 94,35,86,205,28,160,193
100 DATA 225,43,120,177,32,235,201
110 DATA 122,205,42,160,123,205,42
120 DATA 160,62,13,205,93,187,201
130 DATA 245,230,240,203,63,203,63
140 DATA 203,63,203,63,205,59,160
150 DATA 241,230,15,198,48,254,58
160 DATA 56,2,198,7,205,90,187
170 DATA 201,0,0,0,0,0,0
```

Sin duda alguna

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

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

SIN DUDA ALGUNA está abierta a todos.

Mi problema es el siguiente: tengo un programa elaborado totalmente por mí, que trata sobre la contabilidad casera, es decir, desde luz, teléfono, gas, etc., hasta el gasto en gasolina y otras cosas del coche, pero mi programa no funciona con personas que no tengan algo de idea de programación, es decir, los datos de consumo y tarifas hay que escribirlos en el propio programa para que luego salgan a la pantalla con todos los demás datos (meses, etc.).

Podría de alguna manera sencilla introducir en el programa una subrutina, con la cual al escribir en la pantalla y en la posición debida quedaran los datos «escritos» en el programa. ¿Es complicado?

Fernando Martín Duarte (Madrid)

Por el contenido de su carta, deducimos que ha debido usted introducir sus números en sentencias DATA y luego leerlos mediante la orden READ. Si realmente ha escogido este método, lo único que necesita es crear una serie de matrices, preferiblemente alfanuméricas (de caracteres), e ir colocando en ellas los datos que vaya leyendo de las datas.

Si además necesita que se introduzcan más datos al programa, use la sentencia INPUT, y si los nuevos datos requieren ser grabados para uso posterior, métalos en las matrices.

Estamos suponiendo que su CPC-464 no lleva unidad de disco; si la lleva, todo sería mucho más sencillo, grabando la información en ficheros secuenciales, según explica claramente el manual.

Si está usted empleando para manipular sus datos un método distinto del expuesto anteriormente, le recomendamos use éste, dado lo sencillo que resulta.

LOS MEJORES PROGRAMAS PROFESIONALES DEL MUNDO

¡a precios "AMSTRAD"!

PARA AMSTRAD PCW 8256 Y AMSTRAD CPC 6128

MICROSOFT

DIGITAL RESEARCH
The creators of CP/M™

MULTIPLAN

Una de las más prestigiosas y completas "hojas de cálculo" del mundo. Rápido y versátil, ofrece prestaciones, como la de relacionar varias hojas entre sí, que no son frecuentes. La capacidad de ejecutar ordenaciones alfabéticas o numéricas, sus posibilidades en cuanto a formato en pantalla y en impresora; los menús en pantalla y la potencia de cálculo, son características distintivas y destacables de MULTIPLAN.

PVP: 15.100.- Pts. (+ IVA)

MBASIC INTERPRETER

Reconocido como el estándar mundial de los lenguajes intérpretes para microordenadores. Fácil de aprender y utilizar.

PVP: 15.100.- Pts. (+ IVA)

MBASIC COMPILER

Totalmente compatible con el MBASIC Interpreter para con una velocidad de ejecución de 3 a 10 veces más rápido. Traduce el código fuente a código objeto y permite una utilización más eficaz del espacio.

PVP: 15.100.- Ptas. (+ IVA)

MS COBOL COMPILER

Lenguaje COBOL según el estándar ANSI, especialmente útil para manejar grandes volumenes de datos.

PVP: 48.500.- Ptas. (+ IVA)

MS SORT

Flexible programa de ordenación según la técnica de la inserción binaria, utilizable independientemente o incluible en programas escritos en MS COBOL.

PVP: 15.100.- Ptas. (+ IVA)

MS-FORTRAN COMPILER

El lenguaje más utilizado en aplicaciones científicas y de ingeniería, es una potente implementación del ANSI-FORTRAN X3.9

PVP: 24.900.- Ptas. (+ IVA)

MS MACRO

Un completo paquete de desarrollo que incluye: MS-MACRO ASSEMBLER; MS-LINK, MS-LIB, MS-CREF y DEBUG.

PVP: 12.000.- Ptas. (+ IVA)

ASHTON·TATE

dBASE II

El Generador de Programas por excelencia. Permite crear bases de datos relacionados a partir de comandos sencillos y sin requerir conocimientos de programación. Las aplicaciones de dBASE II son incontables y cada usuario puede desarrollar las que mejor se adapten a sus necesidades: ficheros y mailings, contabilidades, nóminas, control de costos, control de almacen, facturación, etc. Ampliamente acreditado como uno de los programas más útiles y recomendables de cuantos existen para microordenadores. Manual en castellano.

PVP: 17.800.- Ptas. (+ IVA)

DR. DRAW

Programa interactivo para la creación y edición de gráficos y diagramas. Tres elementos básicos —líneas, texto y símbolos— son utilizados para producir gráficos de alta calidad... logos, diagramas de bloques, diagramas de flujo, etc. Los símbolos, tipos de letra y estilos de líneas, pueden alterarse y modificarse a voluntad del usuario.

PVP: 15.100.- Pts. (+ IVA)

DR. GRAPH

Generador de gráficos —de líneas, barras, columnas y de pastel— de muy sencillo manejo. Permite incluir textos y leyendas con gran flexibilidad de creación y edición.

PVP: 15.100.- Ptas. (+ IVA)

PASCAL MT+

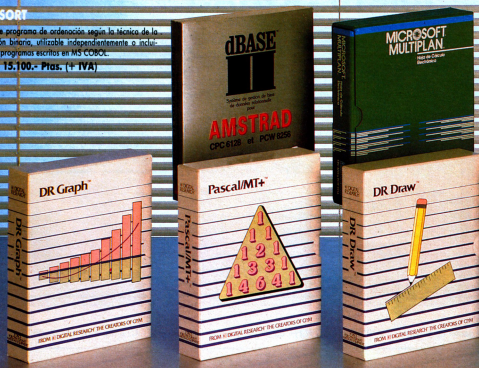
El más rápido PASCAL existente con implementación completa del estándar ISO. Un compilador de código nativo que genera en formato reubicable para usar con su montador de enlace (linker).

PVP: 15.100.- Ptas. (+ IVA)

CBASIC COMPILER

Versión mejorada del clásico lenguaje CBASIC, con mayor velocidad de ejecución y altamente flexible diseñado especialmente para el desarrollo de programas de gestión. Incluye el linker LK-80, que cambia la velocidad del compilador con la rutinas de biblioteca y permite el encadenamiento de módulos.

PVP: 15.100.- Ptas. (+ I.V.A.)



microSIS

P.º CASTELLANA, 179-1.º - 28046 MADRID
Telf. 442 54 33/44

MICRO-1

el IVA lo paga MICRO-1

C/ Duque de Sesto, 50. 28009 Madrid
Tel.: (91) 275 96 16/274 53 80
(Metro O'Donell o Goya)

SOFTWARE: por cada programa GRATIS ¡¡1 BOLIGRAFO CON RELOJ DE CUARZO!!

HYPER SPORTS	2.300 ptas.
TORNADO LOW LEVEL	1.950 ptas.
EXPLODING FISTT	2.300 ptas.
JUMP JET	2.495 ptas.
ZORRO	2.600 ptas.
SABREWULF	1.650 ptas.
GHOSTBUSTERS	1.950 ptas.
GYROSCOPE	2.300 ptas.
HYGHWAY ENCOUNTER	1.750 ptas.
HIGHWAY ENCOUNTER DISCO	3.300 ptas.

DYNAMITE DAN	2.100 ptas.
RAID OVER MOSCOW	2.300 ptas.
THEY SOLD A MILLION	2.500 ptas.
FIGHTER PILOT	1.975 ptas.
MASTER OF T. LAMP	1.950 ptas.
NIGHTSHADE	1.950 ptas.
HACKER	1.950 ptas.
SUPER TEST	2.300 ptas.
MAPGAME	2.700 ptas.
TONADO LOW LEVEL DISCO	3.300 ptas.

JOYSTICK QUICK SHOTT II.. 2.295 ptas.
JOYSTICK QUICK SHOT V ... 2.595 ptas.

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

TAPA METACRILATO PARA TECLADO ¡¡1.900 ptas.!!

UNIDAD DISKETTE 5.25"
¡¡45.900 ptas.!!
(incluido controlador)

LAPIZ OPTICO
¡¡4.900 ptas.!!

IMPRESORA MARGARITA
¡¡49.900 ptas.!!

CASSETTE ESPECIAL
ORDENADOR 5.295 ptas.

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

IMPRESORAS
¡¡20% DTO. SOBRE P.V.P.!!

SINTETIZADOR DE VOZ
Y AMPLIFICADOR:
7.900 ptas.

MODULADOR TV
8.400 ptas.

INTERFACE DISCO
5 1.4" **5.300** ptas.

CINTA C-15 ESPECIAL
ORDENADOR **85** ptas.
DISKETTE 3" **990** ptas.

UNIDAD DE DISCO 3" CON
CONTROLADOR: **49.900** ptas.

Libros:

Curso autodidáctico Basic I	2.525 ptas.
Curso autodidáctico Basic II	2.525 ptas.
Programando con Amstrad	2.195 ptas.
Juegos sensacionales Amstrad	1.950 ptas.
Hacia la Inteligencia Artific.	1.295 ptas.
Música y sonidos con Amstrad	995 ptas.

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

microbyte PRESENTA... AMSTRAD

NUEVOS PROGRAMAS EN CASSETTE Y DISCO

ARGO NAVIS

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

ROCK RAID

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

WIZARD'S LAIR

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

MACADAM FLIPPER

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

JUMP JET

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

MUSIC MAESTRO

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

PAZAZZ

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

SYCLONE 2

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

ZEDIS II

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

SYSTEM X

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

ODDJOB

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

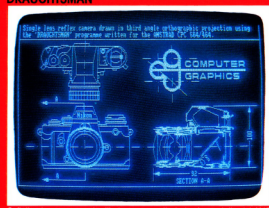
TRANSMAT

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

OTROS PROGRAMAS EN STOCK

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

DRAUGHTSMAN



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

ENVIENOS A **MICROBYTE** S.L.
P.º Castellana, 179, 1.º - 28046 Madrid

Nombre
Apellidos
Dirección
Población
D.P. Teléfono

ENVIOS GRATIS

JUEGO	C	D	Precio	TOTAL

PRECIO TOTAL PESETAS

Incluyo talón nominativo ☐
Contra-Reembolso ☐

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

El único ordenador concebido para sustituir a la máquina de escribir.

AMSTRAD PCW 8256

UN COMPLETO EQUIPO QUE INCLUYE:

- Unidad Central (256 K RAM) • Teclado en castellano
- Unidad de disco (180 K por cara) • Pantalla de alta resolución • Impresora alta calidad (NLQ)
- Programas: • Procesador de textos, sistema Operativo CP/M Plus, Mallard Basic con JET SAM para ficheros indexados, lenguaje DR LOGO.

PROGRAMAS PROFESIONALES

- Contabilidades • Almacenes • Facturación • HOJAS DE CALCULO: **Multiplán,** Supercalc 2, Cracker, Plannercalc. BASES DE DATOS: **DBase II,** Amsfile, Flexifile, **Boriar.** LENGUAJES: Cobol, Fortran, Pascal MT +, Pilot, etc.

SOLICITE DEMOSTRACION EN:

División informática de: El Corte Inglés, División On-line de GALERIAS, Tiendas especializadas en informática y Equipos de oficina.

NOTA: El Amstrad también puede ser utilizado como "Terminal Inteligente" de grandes equipos informáticos.

¡¡Increíble!!

AMSTRAD ESPAÑA
GRUPO INDESCOMP

Avda. del Mediterráneo, 9. 28007 Madrid. Télex 47660 Tels.: 433 45 48 - 433 48 76 - FAX - 433 24 50 Delegación en Cataluña: C/ Tarragona, 110. Tel. 325 10 58. 08015 Barcelona.