AÑO II N.º 48

MICROHOBBY AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO II N.º 48

160 Ptas.

Canarias **165** pts.

LISTAS Y RECURSION, DOS BUENAS RAZONES PARA UTILIZAR LISP

WINTER GAMES: TODO EL DINAMISMO DE LOS JUEGOS DE INVIERNO ENTRE ESPECTACULARES GRAFICOS

DALE CARACTER A TUS PROGRAMAS. GRAFICOS DEFINIDOS POR EL USUARIO DESDE CODIGO MAQUINA.

Toda la potencia y elegancia de las funciones de librería a tu alcance

HOBBY PRESS

SOMOS MAYORISTAS

MICRO-1

PRECIOS INCLUIDO IVA

C/ Duque de Sesto, 50. 28009 Madrid (Metro O'Donnell o Goya)
Tel.: (91) 274 53 80 Aparcamiento gratuito en Felipe II

Ofertas en software: 2 programas al precio de 1 y además regalo fin de curso una calculadora completamente gratis. ¡¡Asombroso!! ¿Verdad?

BAT MA N	**2.300** ptas.	KUNG-FU MASTER	**2.300** ptas.
ROCK'N LUCHA	**2.300** ptas.	SABOTEUR	**2.300** ptas.
YIER AR KUNG FU	**2.300** ptas.	PING PONG	**2.300** ptas.
THE WAY OF THE TIGER	**2.300** ptas.	MILLION II	**0000** ptas.
WEST BANK	**2.100** ptas.	OLE TORO	**2.300** ptas.
CAMELOT WARRIORS	**2.300** ptas.	TURBO ESPRIT	**2.100** ptas.
RAMBO	**2.300** ptas.	SABRE WULF	**1.650** ptas.
WORLD CUP (DISCO)	**3.300** ptas.	BATALLA DE LOS PLANETAS	**2.100** ptas.
MILLION II (DISCO)	**3.300** ptas.	SABOTEUR-COMBAT LINX DISCO	**3.300** ptas.

RAMBO-MATCH DAY (DISCO) **3.300** ptas.

SOFTWARE DE REGALO (OFERTA 2×1)
DECATHLON BEACH HEAD SOUTHERN BELLE
DRAGONTORC

LAPIZ OPTICO
3.295 PTAS.

CASSETTE ESPECIAL ORDENADOR
5.295 PTAS.

SINTETIZADOR DE VOZ EN CASTELLANO
7.650 PTAS.

AMPLIACION DE MEMORIA ANTA 64 K.3
12.500 PTAS.

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

TAPA DE METACRILATO 464	895
CABLE CENTRONICS	3.175
CABLE SEGUNDA UNIDAD D.	1.790
CABLE SEPARADORES 6128	1.975
INTERFACE RS232	9.265
CINTA VIRGEN C15	69

CABLE AUDIO	795
CABLE ADAPTADOR 2 JOYSTICK	2.390
CABLE SEPARADORES 464	1.390
CABLE SEPARADOR 8256	2.900
CABLE RS232	2.500
DISKETTES 3''	990

PRECIOS EXCEPCIONALES PARA TU AMSTRAD
CPC-464, CPC-6128, PCW-8256

¡¡LLEGARON LAS REBAJAS DE VERANO A MICRO 1!!

OFERTAS EN JOYSTICKS

QUICK SHOT I	1.395
QUICK SHOT II	1.695
QUICK SHOT V	1.695

PEDIDOS CONTRA REEMBOLSO SIN NINGUN GASTO DE ENVIO TEL. (91) 274 53 80
O ESCRIBIENDO A: MICRO-1. C/ DUQUE DE SESTO, 50. 28009 MADRID.

Tiendas y distribuidores grandes descuentos.
Dirigirse a Diproimsa. C/ Galatea, 25. Tel. (91) 742 20 19 ó 742 79 68

Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
José M.ª Díaz
Redactor Jefe
Juan José Martínez
Diseño gráfico
Fernando Chaumel
Colaboradores
Eduardo Ruiz
Javier Barceló
David Sopuerta
Robert Chatwin
Francisco Portalo
Pedro Sudón
Miguel Sepúlveda
Francisco Martín
Jesús Alonso
Pedro S. Pérez
Amalio Gómez
Alberto Suñer
Secretaria Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Chema Sacristán
Portada
M. Barco
Ilustradores
J. Igual, J. Pons, F. L. Frontán, J. Septien, Pejo, J. J. Mora

Edita
HOBBY PRESS, S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

Jefe de Producción
Carlos Peropadre

Marketing
Marta García
Jefe de Publicidad
Concha Gutiérrez
Publicidad Barcelona
José Galán Cortés
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
Ctra. de Irún km 12,400
(Fuencarral) 28049 Madrid
Teléfonos: Suscrip.: 734 65 00
Redacción: 734 70 12

Dto. Circulación
Paulino Blanco

Distribución
Coedis, S. A. Valencia, 245
Barcelona

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

MICROHOBBY

AMSTRAD

sumario

Año II • Número 48 • 29 de Julio al 4 de Agosto
160 ptas. (incluido I.V.A.)
Canarias, 155 ptas. + 10 ptas. sobretasa aérea.
Ceuta y Melilla, 155 ptas.

¿Tienes ya tu hobby?

Ya está a la venta el nº 2

HOBBY sale este mes a la calle con un montón de temas insólitos que seguro te apasionarán. ¿Sabías, por ejemplo, que puedes ser un campeón de baloncesto aunque midas menos de 1,80? ¿O que es posible domar una planta carnívora y medir sus estímulos? HOBBY te enseña también cosas tan dispares como los pasos a seguir para irte a un pueblo abandonado o el arte de volar cometas. Y, demás, todos los meses regalamos una cadena de música.

Ten un HOBBY todos los meses.

¡Ya está en tu kiosco el n.º 2!

Primera PLANA

DAVID PARK, NOMBRADO DIRECTOR DE NORTHERN COMPUTERS

David Park, mano derecha de Clive Sinclair durante doce años, ha sido nombrado director general de NORTHERN COMPUTERS. Será el responsable de las ventas y distribución de la compañía AMSTRAD NETWORK and HARD DISK SYSTEM PRODUCTS.

El señor Park trabajó en SINCLAIR RADIONICS en 1969 como el primer vendedor de Clive Sinclair.

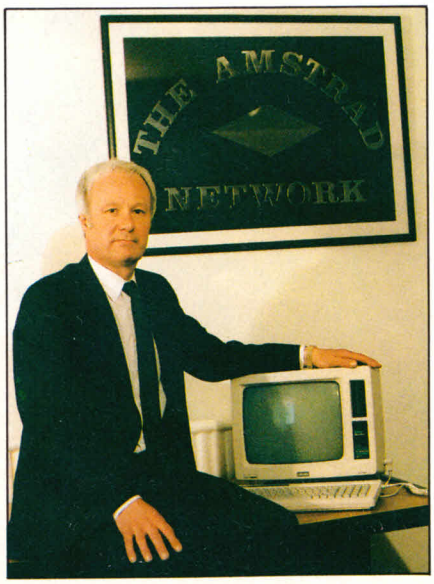
Mientras trabajó con Sinclair fue escalando posiciones tales como director de ventas para Inglaterra, director de exportación y director de marketing.

Introdujo los ordenadores Sinclair en países tales como Alemania, España, China, India, Iraq, Arabia Saudita y la URSS.

Recientemente David Park dimitió de SINCLAIR PRODUCTS a principios de abril.

NORTHERN COMPUTERS se encuentra enclavada en York y exportará a Europa —tanto Oriental como Occidental— Australia y América del Norte los productos de HARD DISK y NETWORK SYSTEMS.

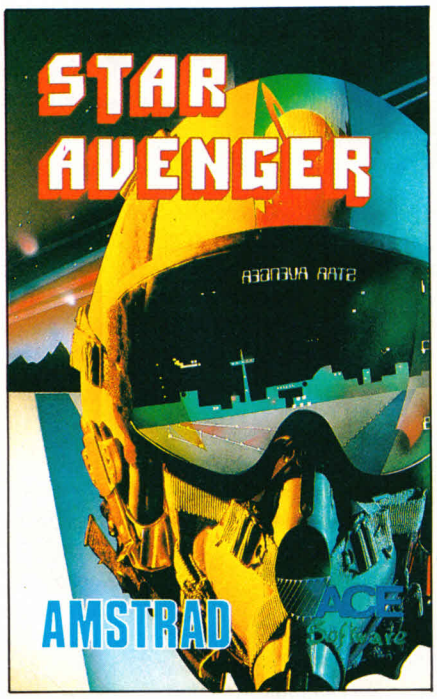
La primera tarea de David Park será nombrar distribuidores en más de los 30 ó 40 países en los que se utilizan ordenadores **Amstrad.**



STAR AVENGER EN VERSION AMSTRAD

Ya tenemos para Amstrad el conocido juego, en versión para MSX, STAR AVENGER.

Este juego tuvo una etapa dorada, hace tiempo ya, en las máquinas de salas de juego o billares y bares. Posteriormente y con algunas modificaciones llegó a los ordenadores domésticos. Pero como no a perdido nada de su encanto, pasaremos buenos momentos guiando nuestra nave a través de angostos túneles don-



de pondremos a prueba nuestra habilidad de pilotos, lo mismo tendremos que hacer en tétricas cavernas, ejercitaremos la puntería para derribar las naves enemigas que salen a nuestro paso y escapar sorteando las gigantescas bolas de fuego que se empeñan en complicar nuestro vuelo.

Pero no dudamos que con nuestra poderosa nave portadora de un mortífero láser y nuestra pericia, llena de serena frialdad, venceremos todos los peligros que se nos presenten.

STAR AVENGER es de ACE Software y está distribuido por MICROBYTE.

NUEVO PRECIO DEL AMSTRAD PCW 8512

AMSTRAD ESPAÑA, del Grupo Indescomp, acaba de anunciar el nuevo y espectacular precio de su ordenador PCW 8512 que, a partir de ahora, será de 149.900 ptas. (más IVA).

El PCW 8512, que incluye Unidad Central con 512 K de RAM, dos Unidades de Disco (173 y 720 K), monitor e impresora de alta calidad, venía comercializándose a 169.900 ptas. (más IVA). **AMSTRAD ESPAÑA**, siguiendo su ya habitual política de precios, ha decidido trasladar al usuario final el descenso en los costes de fabricación de este modelo, obtenido gracias al enorme número de unidades producidas y vendidas en los últimos meses.

GOGLY EN VERSION DISCO

Está a la venta la versión en disco del simpático juego GOGLY.

Este, después de perderse en el remoto pasado, sigue intentando volver al presente, atravesando diferentes épocas de la historia. En estas deberá encontrar la llave que le abra la puerta de una etapa cada vez más moderna y que le acerque a su tiempo.

Pero con lo que hay que tener mucho cuidado es con los proyectiles que quieren impedir que GOGLY regrese a su época.

En su viaje a través de tiempo, nuestro amigo puede moverse en tres velocidades, rápida, normal o lenta.

Siendo la velocidad rápida de una acción trepidante y sumamente divertida.

El juego es de la casa ACE Software y está distribuido por MICROBYTE.

FUNCION MID$

LA DOBLE VIDA DE MID$

Ya conocemos un gran número de funciones para un tratamiento avanzado de las cadenas de literales. Podemos fraccionarlas, agruparlas, tomar el número de caracteres que deseemos, encontrar su longitud, etc. Pero lo bueno es que todavía nos queda alguna por descubrir e incluso veremos la posibilidad de utilizar alguna no como función, sino como orden.

Este último es el caso de **MID$.** ¿Recuerda lo que hacía? No le importará que se lo volvamos a contar muy someramente no sea que...

MID$ (cadena, posición, longitud)

era, hasta ahora, una función que nos devolvía un valor alfanumérico que ya podíamos asignar a una variable o tratarle de acuerdo con las necesidades de nuestro programa.

Consistía en una subcadena literal compuesta por tantos caracteres como indique el parámetro **«longitud»** tomados de la **«cadena»** especificada a partir del que ocupa el lugar indicado por **«posición».** Por ejemplo:

PRINT MID$ **(MICROHOBBY,** 3,3)

nos imprimirá tres caracteres del valor literal **MICROHOBBY** empezando a contar desde el que ocupa la tercera posición. Si nos ha seguido, no tendrá ninguna dificultad en deducir correctamente la solución: la subcadena **«CRO»** nos aparece en la pantalla.

Sobre esta definición general de la función podemos hacer alguna que otra puntualización. Escriba ahora:

PRINT MID$ **(MICROHOBBY),** 3)

Aparentemente hemos cometido un error sintáctico puesto que nos falta especificar el valor del parámetro que determina la longitud de la subcadena. Esperamos, pues, que aparezca el mensaje al que ya estamos suficientemente acostumbrados:

SYNTAX ERROR

Y en esta posición el **Amstrad** nos sorprende, acepta la instrucción como correcta y la ejecuta sin problemas. Algo no concuerda con lo que conocemos de momento.

Antes de pensar que nuestro ordenador no funciona, analicemos el resultado que nos aparece en la pantalla:

«CROHOBBY»

El carácter de comienzo de esta subcadena es el que hemos seleccionado —o sea, el que ocupa la posición número tres— y desde ahí coincide con el resto de la primitiva cadena.

Repita la misma experiencia con cualquier otra cadena, comprobará que los resultados son semejantes.

Con esto deducimos una nueva facilidad que nos ofrece la función MID$. Cuando falta el parámetro que nos indica la longitud de la subcadena, que como vemos, es opcional, el **Amstrad** toma todos los caracteres desde el que ocupa la posición seleccionada hasta el final de la cadena a la que aplicamos la función.

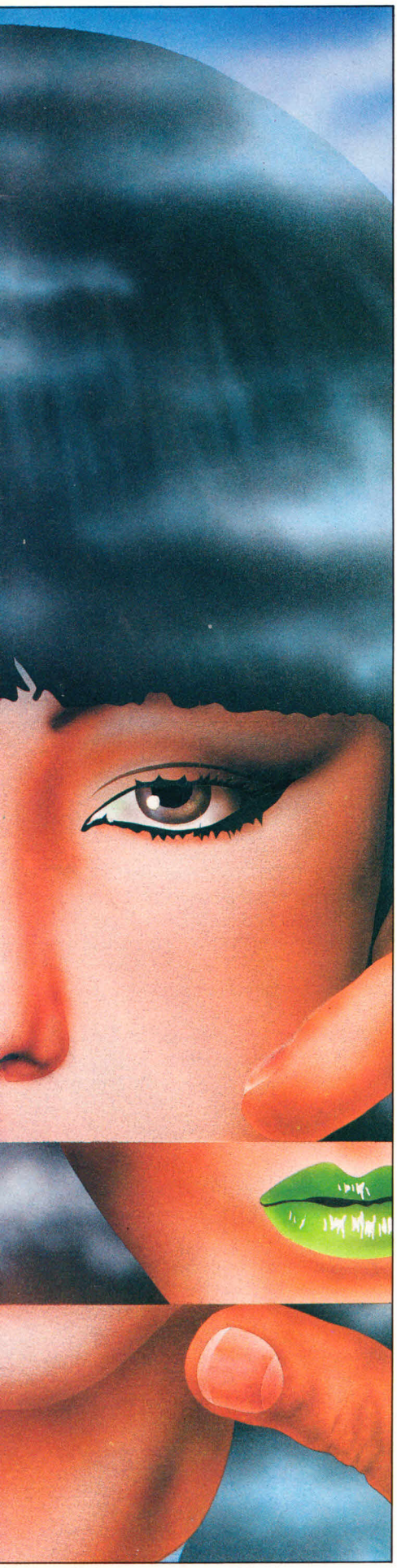
Puede decirnos que es como si en lugar de **MID$** usáramos otra de las funciones que ya conocemos. Y algo parecido ya es, pero no del todo. Observe la diferencia existente entre ambas tecleando:

PRINT MID$ **(MICROHOBBY,** 3)

y a continuación:

PRINT RIGHT$ **(MICROHOBBY,** 3)

Con la primera obtenemos el resultado comentado anteriormente. Sin problemas. Pero la segunda sentencia nos ofrece un resultado



muy diferente, *¿no es así?* En la pantalla nos aparecen los tres últimos caracteres solamente. ¿Extrañado?

Vuelva a leer la definición que dimos de la función RIGHT$. ¿Ya? Aplicada a este caso nos genera una subcadena de tres caracteres tomados a partir del final de la palabra. Por lo tanto, es muy lógico que se imprima solamente los caracteres:

«BBY»

Mientras en **MID$** el segundo parámetro nos indica el lugar a partir del que vamos a ir seleccionando caracteres, en RIGHT$ lo que nos está diciendo es el número de ellos que debemos tomar empezando a contar desde el último hacia la izquierda. De ahí la diferencia entre los resultados obtenidos.

Y, ¿qué ocurre cuando por error se le pasa a esta función MID$ una longitud de subcadena mayor que el número de caracteres disponibles? Compruébelo escribiendo:

PRINT MID$ («HOLA», 2,6)

Hemos pedido al **Amstrad** que imprima 6 caracteres comenzando a contar desde el segundo de la palabra **«HOLA».** Y él, sin complicarse mucho la vida, nos ha sacado las tres letras disponibles a partir de la **«Q».** No nos ha dado ningún mensaje de error, sino que sencillamente ha tomado los caracteres que ha podido.

Si le parece bien y tiene un poco de curiosidad, calcule la longitud de la subcadena generada por la función. Para ello teclee:

a$=MID$ **(«HOLA»,** 2,6)

después:

PRINT LEN (a$)

y se convencerá de lo que le estamos diciendo.

El rango, o margen de valores, de los parámetros de posición y longitud de la función MID$ está comprendido entre 0 y 255 en el caso de **«longitud»** y entre 1 y 255 para **«posición».** Si damos a cualquiera de ellos un número fuera de estos límites se nos producirá, evidentemente, un mensaje de error. Pero no lo dé vueltas, intente comprobarlo por sus propios medios.

PRINT MID$ **(«HOLA»,** 0,2)

es una instrucción errónea y así nos lo dice el micro:

IMPROPER ARGUMENT

No es muy difícil darse cuenta por qué se ha producido. Estamos diciendo al ordenador que seleccione 2 caracteres tomados a partir del que ocupa la posición «cero» dentro de la palabra **«HOLA».** ¡Un momento! ¿Cuál es la posición cero de una cadena de caracteres? Si el carácter que está al principio del literal está colocado en el lugar «uno», ¿cómo podemos empezar a seleccionar letras a partir del cero? Luego el mensaje de error no es el del todo ilógico, ¿verdad?

Primeros PASOS

Teclee ahora:

PRINT MID$ **(«HOLA»,** 2,270)

Nos vuelve a aparecer el mismo mensaje de error. Ahora no es debido a que la posición de comienzo no es correcta sino que se produce porque el parámetro de **«longitud»** de la subcadena se sale fuera del rango: es mayor de 255.

¿Recuerda el tamaño máximo que admitía el **Amstrad** para un literal? Era precisamente 255 caracteres. Imaginamos que después de esto no será necesario explicarle por qué un parámetro de **«longitud»** igual a 270 está fuera de rango, ¿no es así?

Puede ocurrir otra cosa extraña. Imagine que estamos aplicando esta función a una cadena que es de menor longitud que la posición especificada. En la práctica sería:

PRINT MID$ **(«HOLA»,** 64,3)

A simple vista podemos darnos cuenta que **«HOLA»** es una palabra de cuatro caracteres. Pero, practique con las funciones y teclee:

PRINT LEN **(«HOLA»)**

Si no nos hemos equivocado, el valor que aparece coincide con lo que era evidente, *¿no?*

Volvamos a lo nuestro. Con MID$ queremos generar una subcadena formada por caracteres de **«HOLA»** seleccionando 3 comenzando a contar a partir del que ocupa el lugar 64. Ahora bien, hemos dicho que **«HOLA»** sólo tiene 4 letras, así que, ¿cuál será el que esté en la posición 64?

Escriba de nuevo:

PRINT MID$ **(«HOLA»,** 64,3)

y obtendrá la respuesta en la pantalla.

Bueno, pues no aparece nada.

Y esa es precisamente la subcadena que produce. Estamos ante lo que hemos llamado anteriormente **«cadena vacía»** que consiste en un literal que no contiene ningún carácter.

Puede asegurarse de ello, también, comprobando la longitud que tiene. Así que:

PRINT LEN (MID$ **(«HOLA»,** 64,3)

y el cero que se visualiza confirma nuestra teoría.

Resumiendo. Cuando pedimos la función **MID$** a una expresión alfanumérica y le pasamos un parámetro «posición» mayor que la longitud de la expresión, el ordenador muy astutamente nos dice que allí no hay nada, nos devuelve una **«cadena vacía»** y punto.

Al principio de este artículo insinuamos que había alguna función de tratamiento de cadenas que además se podía utilizar como orden o comando. **MID$** es uno de ellas. Quizá sea una de las pocas palabras clave Basic que permitan esta doble vertiente.

Hasta ahora nos ha servido para seleccionar un cierto número de caracteres de los que componen una expresión literal. Este es su trabajo como función.

Pero ahora vamos a intentar cambiar alguno de ellos asignándoles el valor que queramos. La manera de hacerlo es asignarles un valor. ¡Así de fácil! Lo complicado es la manera de situarnos encima de los caracteres que necesitamos cambiar y una vez allí transformarlos en los nuevos.

Siempre hemos estado presumiendo de que el **Amstrad** poseía un conjunto de sentencias y funciones para tratamiento de cadenas muy potente. Esta es una buena ocasión para demostrarlo.

La forma de dar un valor a una variable literal era mediante una sentencia de asignación. Con:

saludo$ = **«HOLA»**

estamos almacenando en el espacio de memoria que el ordenador ha reservado para la variable «saludo$», la constante alfanumérica **«HOLA».** ¿Quiere comprobarlo?, pues adelante.

PRINT saludo$

le habrá sacado de cualquier duda que tuviera al respecto.

En nuestro caso seguiremos un camino parecido. Ahora seleccionaremos los caracteres que queramos cambiar y después le asignamos el valor.

Con:

a$ = **«HOLA»**
MID'' (a$, 2,3)

estamos tomando 3 caracteres de la palabra **«HOLA»** a partir de la **«O»,** que es el que ocupa el segundo lugar.

Y precisamente ahí es donde queremos darle el nuevo valor. Así que sólo queda asignárselo.

MID$ (a$, 2,3) = **«IPO»**

nos producirá el efecto deseado. Almacenará la palabra **«IPO»** a partir del segundo carácter de **«HOLA».**

Podemos comprobarlo de la siguiente manera. Primero asignamos **«HOLA»** a una variable y lo visualizamos. Después hacemos la modificación y a continuación imprimimos la variable donde se ha realizado el cambio. Esto es precisamente lo que hace el programa 1. Analicémosle.

Programa uno

En la línea 30 asignamos a la variable «palabra$» el valor elegido y la imprimimos con la 50.

Realizamos el cambio en la 80. Observe que tiene el mismo aspecto que la del ejemplo que pusimos. Sólo hemos cambiado la constante **«HOLA»** por la variable «palabra$».

Y visualizamos la modificación con la 90.

Es muy sencillo, pero creemos que lo suficientemente ilustrativo como para despejarle los puntos oscuros acumulados.

Compliquemos un poquito más el asunto.

```
10 REM PROGRAMA I
20 CLS
30 palabra$="HOLA"
40 PRINT"ANTES DEL CAMBIO: ";
50 PRINT palabra$
60 PRINT
70 PRINT"DESPUES DEL CAMBIO: ";
80 MID$(palabra$,2,3)="IPO"
90 PRINT palabra$
100 PRINT
```

Vamos a codificar, o al menos intentar hacerlo, un programa que nos permita cambiar cualquier carácter de cualquier palabra. El ordenador nos preguntará por la cadena que queremos modificar, luego el nuevo valor, la posición, el número de caracteres y ¡manos a la obra!

Y tras largos y penosos procesos lógicos y noches de insomnio (es broma) llegamos a la conclusión que sería algo parecido al programa 2.

Programa dos

Introducimos los datos mediante las líneas 30 a 60. Se hace el cambio en la 110 y se imprime el resultado en la 120. ¿Sencillo?

Vamos a intentar sacarle más jugo a este pequeño programa. De momento el argumento de MID$ no puede ser ya una cadena literal constante tal como ocurría antes, sino que es necesario que sea una variable literal. Por eso pasamos la palabra a la variable «palabra$» en la línea 30. Intente teclear:

MID$ **(«HOLA»,** 2,3) = «IPO»

y verá coomo el ordenador le avisa que está cometiendo un error.

En esta ocasión «longitud» contiene el número de los caracteres que tenemos que seleccionar del **«nuevo valor»** para introducirlos después en la variable literal. El parámetro **«posición»** sigue indicando lo que ya sabemos: es dónde comienza la inserción de los nuevos caracteres.

Este programa no comprueba si los parámetros están dentro de los márgenes válidos. Por ejemplo, podemos decir que en la posición 2 de **«HOLA»** sustituyamos cinco caracteres por **«IPO».** Investigue lo que pasa al escribir:

a$ = «HOLA»
MID$ (a$, 2,5) = «IPO»

y

PRINT a$

```
10 REM PROGRAMA II
20 CLS
30 INPUT"DIME UNA PALABRA ",palabra
$
40 INPUT"NUEVO VALOR ",valor$
50 INPUT"POSICION ",posicion
60 INPUT"NUMERO DE CARACTERES A SUS
TITUIR ",numero
70 PRINT"ANTES DEL CAMBIO: ";
80 PRINT palabra$
90 PRINT
100 PRINT"DESPUES DEL CAMBIO: ";
110 MID$(palabra$,posicion,numero)=
valor$
120 PRINT palabra$
130 PRINT
```

Estamos pidiendo que sustituya cinco caracteres a partir de la **«O»** de **«HOLA»** por otros cinco que coja de **«IPO»** (sólo tiene tres).

Puede preguntar después la longitud de la cadena obtenida como comprobación y, viendo los resultados, ¿sería capaz de sacar alguna conclusión? Imaginamos que sí, ya que estamos siguiendo un proceso análogo a lo que dijimos para MID$ empleada como función en los casos que los parámetros no concuerdan del todo. Haga las pruebas que se le ocurran, dentro y fuera del rango, y anote los resultados. Seguro que en algún momento pueden venirle como anillo al dedo todos estos apuntes.

Pero nosotros, por si acaso, hemos codificado el programa 3 en el que ya sí se chequea que los valores de «longitud» y «posición» sean medianamente correctos. Echele un vistazo.

Programa tres

Continuemos avanzando. Ahora, en vez de sustituir, vamos a presentar otra función que lo que va a permitir es detectar si una determinada cadena está o no incluida en una constante o variable literal. Sería muy interesante para corregir o modificar una palabra que está incluida repetidas veces en una cadena, por ejemplo.

La función a la que nos referimos es INSTR y esta es su forma general:

INSTR (posición, cadena-1, cadena-2)

que nos determina si la **«cadena-2»,** o cadena buscada, está contenida en **«cadena-1»,** o cadena en la que se busca, a partir de la posición indicada por **«posición».** ¿Enrevesado?

Veámoslo en la práctica. Necesitamos encontrar si una determinada palabra contiene la letra **«A».** Pues primero dele un valor a la palabra:

palabra$ = **«HOLA»**

y después

PRINT INSTR (1, palabra$, «A»)

```
10 REM PROGRAMA III
20 CLS
30 INPUT"DIME UNA PALABRA ",palabra
$
40 INPUT"NUEVO VALOR ",valor$
50 IF LEN(valor$)>LEN(palabra$) THE
N GOTO 40
60 INPUT"POSICION ",posicion
70 IF posicion>LEN(palabra$) THEN G
OTO 60
80 INPUT"NUMERO DE CARACTERES A SUS
TITUIR ",longitud
90 IF posicion+longitud-1>LEN(palab
ra$) THEN PRINT"NOS SALIMOS DE LA P
ALABRA":GOTO 60
100 PRINT
110 PRINT"ANTES DEL CAMBIO: ";
120 PRINT palabra$
130 PRINT
140 PRINT"DESPUES DEL CAMBIO: ";
150 MID$(palabra$,posicion,longitud
)=valor$
160 PRINT palabra$
170 PRINT
```

Con esta última instrucción le decimos al **Amstrad** que detecte e imprima si dentro de **«HOLA»** y a partir de la primera posición está incluida la letra **«A».** A simple vista es muy fácil ver que **«HOLA»** sí tiene **«A».** ¿Qué le ha aparecido en la pantalla? ¡Vaya!, resulta que ha sido un número: el 4.

Con él, INSTR nos está señalando la posición donde ha encontrado la «A». ¿Concuerda con lo que había deducido en su inspección ocular?

Así que INSTR nos devuelve un número entero que marca el lugar a partir de donde se encuentra la cadena buscada (cadena-2). ¿Qué pasará cuando no esté incluida en «cadena-1»? Compruébelo, como siempre, usted mismo escribiendo:

a$ = **«PEPITO»**

a continuación:

PRINT INSTR (1,a$, **«A»)**

y ahora el resultado es un cero. La posición 0 no existe, así que podemos asociar que este valor a que **«A»** no está contenida en «PEPITO». Y no nos faltaría mucha razón.

Primeros PASOS

El primer parámetro, el de posición, es opcional. Puede ir o puede faltar. Si le ponemos, el ordenador empezará a buscar a partir de ese lugar o lo que es lo mismo, le estamos indicando el punto donde debe empezar a buscar. Cuando no vaya como parámetro en la expresión de INSTR, el **Amstrad** asume que ha de comprobar si «A» está incluida en la variable literal, en este caso «a$», a partir del primer carácter de la misma. Compruébelo tecleando:

a$ = **«HOLA»**
PRINT INSTR (a$, «A)

y obtendrá los mismos resultados que con el ejemplo anterior.

¿Cree necesario que le contemos algo sobre los márgenes entre los que se debe mover el valor de este parámetro de «posición»? Seguro que no.

Y nada más por el momento. Le dejamos con unas cuantas instrucciones agrupadas que nos permitirán cambiar una cadena dentro de una palabra por otra a nuestra elección, con la única limitación que tengan la misma longitud. Estudie con detenimiento el programa 4 y anote la manera de emplear todas estas instrucciones de fraccionamiento y manejo de cadenas.

Programa cuatro

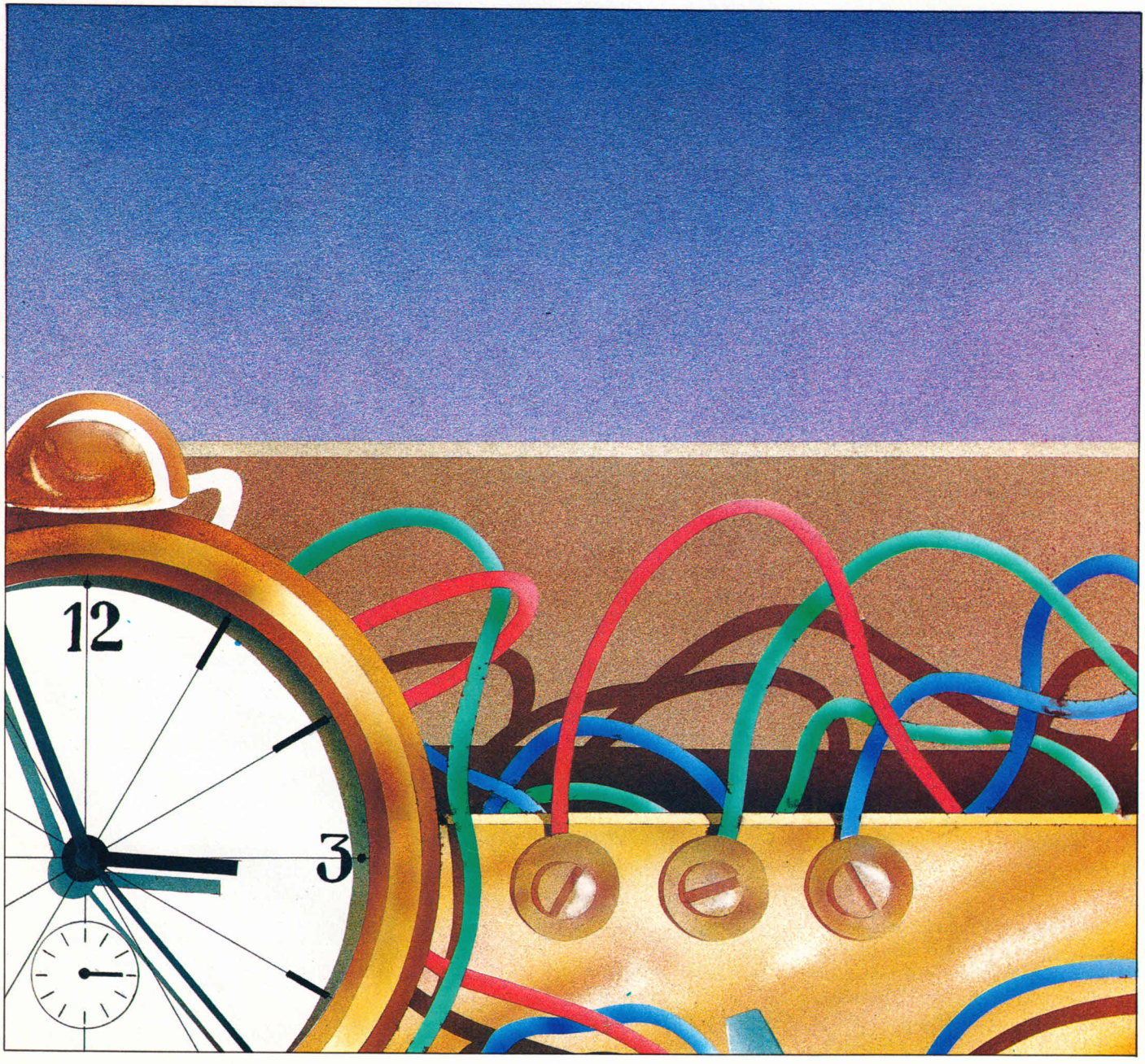
Con un helado bien frío en las manos, al borde de un maravilloso lago de aguas tranquilas, que es lo más propio para sobrellevar el verano, nos atrevemos a decirle: ¡Hasta pronto!

```
10 REM PROGRAMA IV
20 CLS
30 INPUT"DIME UNA PALABRA ",palabra
$
40 PRINT
50 INPUT"CADENA A SUSTITUIR ",caden
a$
60 contenido=INSTR(palabra$,cadena$
)
70 IF contenido=0 THEN PRINT"NO EST
A CONTENIDA":END
80 INPUT"POR ",cambio$
90 IF LEN(cambio$)<>LEN(cadena$) TH
EN PRINT"NO COINCIDE LA LONGITUD":G
OTO 80
100 MID$(palabra$,contenido,LEN(cam
bio$))=cambio$
110 PRINT
120 PRINT palabra$
130 END
```

MANIPULACION DE LISTAS EN LISP

Por: Roberto Garrote Bernal

Es probable que los dos últimos artículos que han aparecido en esta sección (programación funcional y recursión) os hayan parecido complicados, difíciles de leer e incluso aburridos. Ha sido necesario escribirlos porque teníamos que asegurarnos de que cuando hablásemos de funciones y de recursión no os íbais a preguntar ¿y eso qué es? Si lo he conseguido, os felicito; pero si no ha sido así no os preocupéis: mucha gente que sabe programar en LISP sigue sin saberlo muy bien. A lo largo de esta serie de artículos iréis comprendiendo un poco mejor esos dos conceptos. Esta semana relajaremos nuestras mentes aprendiendo a manejar listas con las herramientas que LISP nos proporciona.



Inteligencia ARTIFICIAL

Antes de empezar con el tema quisiera pedir disculpas por un fallo que se coló en el intérprete de **LISP** publicado en el **AMSTRAD** Especial, n.º 2 y que muchos de vosotros ya habréis descubierto. El problema surge cuando el intérprete intenta escribir valores booleanos (T o NIL). Los ejemplos que estaban en el artículo del Especial se habían probado con una versión anterior del intérprete y de ahí que yo no los detectara. La solución, sin embargo, es sencilla. Ahí va. La línea 60270 dice:

```
60270 DIM LO(1): LO(0)=NL: LO(1)=T
```

Debe decir:

```
60270 DIM LO(1)
```

Antes hay que añadir una nueva línea, que es:

```
60345 LO(0)=NL: LO(1)=T
```

Subsanando el error podéis volver a releer los artículos de LISP que han aparecido hasta ahora para comprobar que de verdad funciona. (Y tal vez algún día os explique en qué forma trabaja un intérprete de LISP.)

Vamos a empezar recordando algunas definiciones que utilizaremos a lo largo de los artículos que vayan apareciendo. Te aconsejo que las grabes para que así no tengas que teclearlas cada vez que hagas un programa. Para grabarlas puedes hacer dos cosas: la primera es escribirlas con un procesador de textos y la segunda es dárselas a tu intérprete de **LISP** y luego decirle que las guarde. Si eliges la primera opción deberás grabarlas en un fichero que tenga extensión «.LIS», por ejemplo, **LISTAS.LIS.** Si optaste por la segunda puedes ir escribiéndolas y luego te explicaré cómo se guardan en disco o en cinta.

```
(DE PRIMERO (LIS) (CAR LIS))
(DE RESTO (LIS) (CDR LIS))
(DE SEGUNDO (LIS) (PRIMERO (RESTO LIS)))
(DE TERCERO (LIS) (SEGUNDO (RESTO LIS)))
(DE N-ESIMO (N LIS)
(COND ((= N 0) '(NO EXISTE EL ELEMENTO DE NUMERO CERO))
(T (N-ESIMOAUX N LIS))
))
(DE N-ESIMOAUX (N LIS))
(COND ((NULL LIS) '(LA LISTA NO TIENE TANTOS ELEMENTOS))
((= N 1) (PRIMERO LIS))
(T (N-ESIMOAUX (SUB N 1) (RESTO LIS)))
))
```

Si optaste por escribir estas definiciones con un procesador, vamos a hacer que tu genio de LISP las lea. Para ello debes cargar antes el intérprete tecleando RUN''MINILISP desde BASIC. (Es posible que si alguno ha intentado cargar el programa en un 464 con disco se haya encontrado con errores del tipo «Encontrado fin de fichero». En ese caso no debes usar como cargador el programa **MINILISP** sino que debes ejecutar directamente el programa **LISP.BAS.** Este fichero tiene como comentarios las líneas 60.000 y 60.010. Debes suprimir las comillas de los comentarios de modo que queden como líneas de programa normales. Cuando quieras modificar el número de identificadores deberás cambiar el valor de la variable TTS, que se inicializa en la línea 60.000. Un valor de 300 suele ser adecuado en la mayor parte de las ocasiones.)

Ahora el genio de **MINILISP** ya está dispuesto a recibir tus órdenes. Para que lea el fichero con las definiciones de las funciones debes escribir

```
(OPEN ® nombre~ 'INPUT)
```

donde ® nombre~ debes sustituirlo por el nombre que le diste al fichero donde escribiste las definiciones, pero sin la extensión, que debe ser «.LIS». Por ejemplo, como nosotros llamamos al fichero **LISTAS.LIS,** debemos escribir:

```
(OPEN 'LISTAS 'INPUT)
```

El geniecillo de **MINILISP** se encargará de abrir el fichero, leerlo, ubicar las definiciones en su memoria y, cuando acabe con él, cerrarlo y avisarte de ello haciendo sonar un «beep».

Por si fuiste de los que prefirieron escribir las definiciones directamente en el intérprete y quieres guardarlas ahora, por si acaso luego hacemos alguna trastada, entonces debes escribir:

```
(OPEN ' < nombre > 'INPUT)
```

Como nosotros estamos usando el nombre LISTAS, debemos escribir:

```
(OPEN 'LISTAS 'DEFS)
```

Entonces el intérprete de **MINILISP creará un fichero de texto con el nombre LISTAS.LIS** en el que escribirá todas las definiciones que le hayamos dado hasta ese momento.

Ahora que ya tenemos guardadas las primeras definiciones podemos empezar a jugar con nuestro intérprete. Cuando se pretende programar en un lenguaje de **«alto nivel»,** , como **LISP,** no se puede pensar a «bajo nivel», es decir, no se puede pensar en la representación de los objetos con los que se trabaja, sino en los objetos mismos. Voy a explicar esto último. En su artículo anterior comentaba la forma en la que se representan las listas en LISP mediante la utilización de punteros. Esto puede ser de utilidad para entender cómo trabajan las funciones básicas, pero cuando se usan listas en un programa hay que pensar en ellas como una sucesión de objetos ordenada de una determinada forma, y no como una serie de direcciones de memoria que apuntan a determinados lugares. Este proceso se llama abstracción de datos y cuando conseguimos realizarlo podemos considerarnos un escalón por encima del resto de los programadores. En esta serie de artículos veremos bastantes ejemplos de abstracción de datos. Además esta es una técnica que te será de utilidad cuando programes en cualquier lenguaje de alto nivel (no necesariamente LISP) como por ejemplo, Pascal. ¡Enciende tus motores fotónicos y acompáñanos al mundo de los objetos abandonando el liviano mundo de los bits y los bytes! ¡Una fabulosa aventura te está esperando!

Para manipular objetos de cualquier tipo hacen falta unas cuantas operaciones. Por ejemplo, imagínate que nos encontramos en un campo y queremos trabajar con árboles: podarlos, plantarlos, etc. Pero claro, antes de podar algo tenemos que estar seguros de que se trata de un árbol. ¡La que se podría liar si podásemos otros objetos que no fuesen árboles! Necesitamos por tanto tres tipos de operaciones:

1. **Reconocedores,** para saber con qué objeto estamos trabajando.
2. **Constructores,** para generar nuevos objetos a partir de otros. Juntando una raíz y varias ramas podríamos construir un árbol.
3. **Selectores,** que sirven para separar un objeto en sus diversos componentes. Por ejemplo, un árbol se puede separar en raíz y ramas.

Para trabajar con listas sólo necesitamos unas pocas operaciones de estos tres tipos:

1. Reconocedores:

NULL: para saber si una lista es la lista vacía.

```
(NULL '(A B C)) vale NIL
(NULL '(A.B) vale NIL
(NULL '()) vale T
(NULL ()) vale T
(NULL NIL) vale T
```

Recordemos que la lista vacía se representa, bien como **()** , bien como NIL. Además la lista vacía no necesita la comilla porque su valor es ella misma.

ES—LISTA: responde a la pregunta de si un objeto es una lista.

```
(ES—LISTA '(A B C)) vale T
(ES—LISTA '(A.B)) vale NIL
(ES—LISTA ()) vale T
(ES—LISTA 'A) vale NIL
```

2. Constructores:

AÑADE: dados un objeto cualquiera y una lista, construye una nueva lista que tiene como primer elemento el objeto y como resto la lista dada. (El nombrecito se debe a los problemas con las e#es.)

```
(ANYADE 'A '(B C)) vale (A B C)
(ANYADE '(A B) '(C D E)) vale ((A B) C D E)
(ANYADE 'A ()) vale (A)
```

LIST: hace una lista con todos los objetos que se le dan en el orden dado.

```
(LIST 'A 'B 'C 'D) vale (A B C D)
(LIST 'A 1 'B 2) vale (A 1 B 2)
(LIST '(A)) vale ((A))
(LIST) vale ()
```

3. Selectores:

PRIMERO: devuelve el primer elemento de la lista que se le pasa como argumento.

```
(PRIMERO '(A B C)) vale A
(PRIMERO '((A B) C D E)) vale (A B)
(PRIMERO '(A)) vale A
```

RESTO: devuelve la lista que recibe como argumento sin su primer elemento.

```
(RESTO '(A B C)) vale (B C)
(RESTO '((A B) C D E)) vale (C D E)
(RESTO '(A)) vale ()
```

De todas estas funciones NULL y LIST están ya definidas (son primitivas). PRIMERO y RESTO las hemos definido nosotros a partir de las funciones CAR y CDR. La función ES—LISTA la podríamos construir de la siguiente forma:

```
(DE ES—LISTA (SEX)
(COND ((NULL SEX) T)
((ATOM SEX) NIL)
(T (ES—LISTA (RESTO SEX)))
))
```

ANYADE es fácil de definir usando la función CONS.

```
(DE ANYADE (OBJ LIS) (CONS OBJ LIS))
```

Si alguno se ha **«empollado»**ya el manual de MINILISP tal vez haya pensado que la función PAIRP puede servir para saber si un objeto es una lista. Esta función sólo sirve para saber si un objeto es un par, pero ni todas las listas están formadas por pares ni todos los pares son listas. Por ejemplo:

```
(PAIRP ()) vale NIL
(ES—LISTA ()) vale T
(PAIRP '(A.B)) vale T
(ES—LISTA '(A.B)) vale NIL
```

Con todo lo que ya sabemos de listas vamos a construirnos un diccionario. Haremos dos cosas: primero, dadas una lista de palabras en castellano y la lista de las traducciones en inglés (en cualesquiera otros lenguajes, que al genio de MINILISP eso le da igual) asociaremos a cada palabra su traducción; luego, construiremos una función para buscar la palabra inglesa que esté asociada a la palabra en castellano.

A la función encargada de realizar la primera labor la llamaremos HAZPARES.

```
(DE HAZPARES (LIS1 LIS2)
(COND ((NOT (ES—LISTA LIS1)) '(MAL EMPIEZAS))
((NOT (ES—LISTA LIS2)) '(VAYA, IBAS BIEN))
(T (HPAUX LIS1 LIS2))
))

(DE HPAUX (LIS1 LIS2)
(COND ((NULL LIS1) ())
(T (ANYADE (CONS (PRIMERO LIS1)
(PRIMERO LIS2))
(HPAUX (RESTO LIS1)
(RESTO LIS2))
)) ) )
```

Vamos a ejecutar este programa.

```
(HAZPARES '(UNO DOS TRES CUATRO)
'(ONE TWO THREE FOUR))
```

Su valor es:

```
((UNO.ONE) (DOS.TWO) (TRES.THREE)
(CUATRO.FOUR)
```

Para no tener que construir el diccionario cada vez que queremos buscar una palabra es mejor crear una constante que tenga como valor el diccionario. Llamemos a esta constante, por ejemplo, DICCIONARIO. Entonces

```
(PUT 'DICCIONARIO
'VALUE
(HAZPARES '(UNO DOS TRES CUATRO)
'(ONE TWO TRHEE FOUR)))
```

«liga» la constante DICCIONARIO a la lista de pares construida por HAZPARES con las listas dadas. Si quieres comprobar que, en efecto, el genio de MINILISP ha hecho esto puedes escribir

DICCIONARIO

y MINILISP te contestará con:

```
((UNO.ONE) (DOS.TWO) (TRES.THREE)
(CUATRO.FOUR))
```

Vamos ahora con la segunda función, **ASOCIA,** que se encarga de darnos la traducción de una palabra de nuestro diccionario.

```
(DE ASOCIA (PAL DICC)
(COND ((NULL DICC) '(NO CONOZCO ESA PALABRA))
((EQ PAL (CAR (PRIMERO DICC))) (CDR (PRIMERO DICC)))
(T (ASOCIA PAL (RESTO DICC)))
))
```

Ahora podemos preguntarle al genio la traducción de TRES escribiendo

```
(ASOCIA 'TRES DICCIONARIO)
```

devolviendo THREE.

Existe una función incorporada en MINILISP que hace más o menos lo que HAZPARES. Se llama PAIRLIS. Se diferencia de HAZPARES en que la lista que devuelve está invertida. Los mensajes que devuelve en caso de datos erróneos también son diferentes. Si te molesta que la lista de pares esté al revés puedes usar la función REVERSE para invertir la lista. Por ejemplo:

```
(REVERSE '(A B C D)) vale (D C B A)
```

Por tanto

```
(REVERSE (PAIRLIS '(UNO DOS TRES CUATRO)
'(ONE TWO THREE FOUR)))
```

vale lo mismo que

```
(HAZPARES '(UNO DOS TRES CUATRO)
'(ONE TWO THREE FOUR))
```

También existe una función incorporada en MINILISP que actúa de forma semejante a ASOCIA. Se llama ASSOC. Si no encuentra la palabra que le pedimos devuelve NIL y si la encuentra devuelve el par formado por la palabra que le hemos pasado y su «traducción».

Las funciones para construir y manejar un diccionario funcionarán también en el caso de que la segunda lista no sea una lista de palabras, sino una lista de cualquier otra cosa. Por ejemplo

```
(PAIRLIS '(1 2 3 4)
'((UNO ONE) (DOS TWO) (TRES THREE)
(CUATRO FOUR)))
```

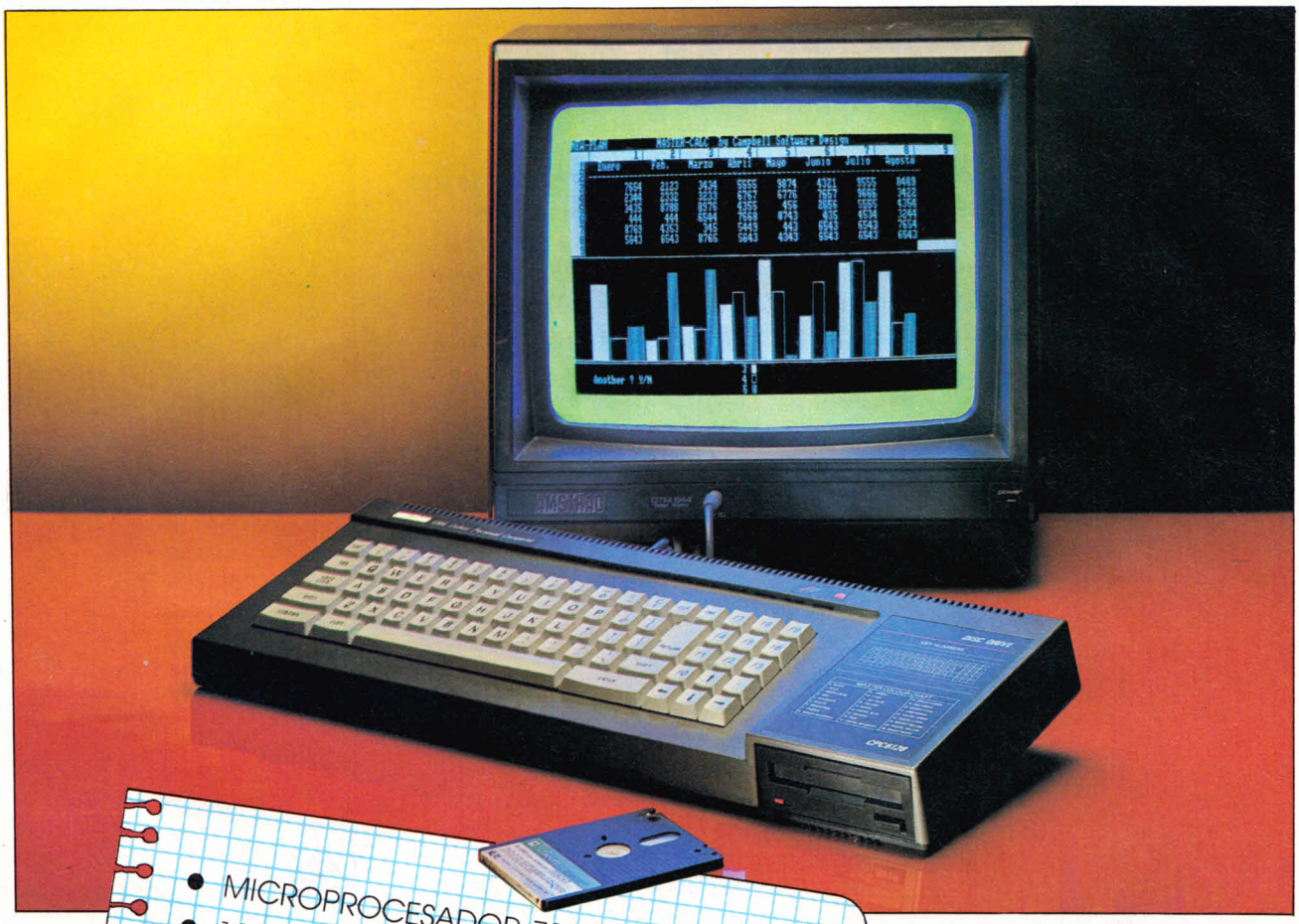
devuelve

```
((1 UNO ONE) (2 DOS TWO) (3 TRES THREE)
(4 CUATRO FOUR)))
(ASSOC 2 '((1 UNO ONE) (2 DOS TWO)))
```

vale (2 DOS TWO). Si en vez de ASSOC usamos ASOCIA el valor es (DOS TWO).

En el próximo artículo veremos cómo se pueden definir y manejar pilas, colas y árboles usando LISP. ¡Y muy pronto hablaremos de bases de datos inteligentes!

AMSTRAD CPC-6128



- MICROPROCESADOR Z80A.
- 128 K DE MEMORIA RAM (41 K DE USUARIO EN BASIC Y 61 K EN CP/M PLUS)
- 48 K DE MEMORIA ROM QUE INCLUYEN EL LOCOMOTIVE BASIC Y EL SISTEMA OPERATIVO.
- 76 TECLAS, TECLADO NUMERICO Y DE CURSOR INDEPENDIENTE.
- TEXTO EN MONITOR DE 20, 40 U 80 COLUMNAS Y GRAFICOS CON DEFINICION DE HASTA 640 × 200 PUNTOS. 27 COLORES DISPONIBLES.
- HASTA 8 VENTANAS EN PANTALLA.
- GENERACION DE SONIDOS EN 3 VOCES Y 8 OCTAVAS.
- UNIDAD DE DISCO DE 3" (169 K BYTES)
- SISTEMAS OPERATIVOS AMS-DOS Y CPM/PLUS
- CONECTORES PARA IMPRESORA, JOYSTICKS, CASSETTE, SEGUNDA UNIDAD DE DISCO, ETC.

SISTEMA COMPLETO CON MONITOR EN FOSFORO VERDE, MANUAL EN CASTELLANO, GARANTIA OFICIAL AMSTRAD ESPAÑA, DISCO CON SISTEMA OPERATIVO CP/M 2.2 Y LENGUAJE DR. LOGO, DISCO CON SISTEMA OPERATIVO CP/M PLUS (CP/M 3.0) Y UTILIDADES, DISCO CON SIETE PROGRAMAS DE OBSEQUIO

84.900 Pts. + I.V.A.

SISTEMA COMPLETO IGUAL AL ANTERIOR PERO CON MONITOR EN COLOR.

119.900 Pts. + I.V.A.

AMSTRAD ®

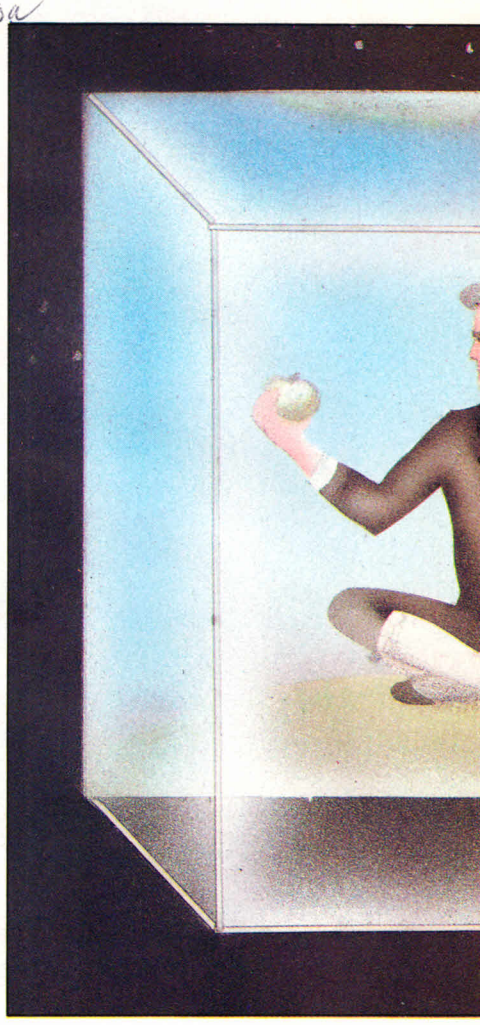
ESPAÑA

Avd. de Mediterráneo, 9, 28007 MADRID.
Tels. 433 45 48 - 433 48 76
Delegación Cataluña: C/. Tarragona, 110, 08015 BARCELONA - Tel. 325 10 58

PROGRAMAS DE PLANOS INCLINADOS NEWTON

COMENTARIO A NEWTON

Newton ha sido como una aplicación para la física de BUP. Intentamos con él ser el adecuado complemento informático a las clases o un útil instrumento para presentar, explicar y resolver problemas de máquina de Atwood (planos inclinados), mediante las ecuaciones de Newton.



Hemos tratado de presentar la información de la forma más sencilla posible, para ello hemos potenciado las pantallas gráficas, tratando de presentar la mayor cantidad posible de información por este método. Nuestro programa dibuja las masas, planos inclinados, pesos, resultantes y normales, poniéndoles nombre con la siguiente notación:

* M?: Masa ? (la interrogación es un número)
* P?: Peso ?
* R?: Fuerza resultante en la dirección del movimiento de la masa ?
* N?: Normal, fuerza resultante en la dirección perpendicular a la dirección del movimiento.

Otra forma sencilla de presentar la información es a través de pantallas de texto. No hemos escatimado de esta manera información adicional sobre el problema, junto con una explicación detallada de los resultados y los pasos intermedios. Para completar el programa he incluido un generador aleatorio de problemas junto con subrutinas que presentan el problema permitiéndote operar para conseguir el resultado que después compararás con la solución verdadera para poder de algún modo permitirte conocer tus errores.

Estructura

Cabecera:
  Resolver problema
  Poner problema

A) Resolver problemas:
  Una masa
  Dos masas

A.1. Dos masas:
  * Entrada de datos dinámica.
  * Presentación del problema y resultados intermedios dos masas.
  * Dibujar
  * Presentación de resultados

A.2. Menú una masa:
  Estática
  Dinámica

A.2.a. Estática:
  Hallar ángulo.
  Hallar coeficiente de rozamiento.

A.2.a.1. Hallar ángulo:
  * Entrada de datos estática.
  * Presentación de ecuaciones y resultados intermedios ángulo.
  * Dibujar.
  * Presentación de resultados una masa.

A.2.a.2. Hallar coeficiente de rozamiento:
  * Entrada de datos estática.
  * Presentación de ecuaciones y resultados intermedios coef. rozamiento.
  * Dibujar.
  * Presentación de resultados una masa.

A.2.b. Dinámica:
  * Entrada de datos dinámica.
  * Presentación del problema y resultados intermedios dinámica una masa.
  * Dibujar.
  * Presentación de resultados una masa.

B) Poner problema:
  * General problema.
  * Presentación del problema y de los datos.
  * Operaciones e introducción de resultados.
  * Control de soluciones:
    Bien
    Mal

B.1. Bien:
  Volver a empezar
  Otro problema

B.2. Mal:
  Volver a empezar
  Intentarlo de nuevo
  Ver resultados

Trabajando con Newton

Es muy fácil, sólo tienes que tener en cuenta el siguiente criterio:

PULSA significa que deberás pulsar la tecla que se te indica para realizar la operación que anuncia.

METE o si no se indica nada, sólo se pide información, deberás teclear la información que se pide, sólo numérica, y después dar a la tecla de vuelta de carro, return o enter.

NEWTON BB

```
1 REM **********************
2 REM *** programa hecho ***
3 REM ***                ***
4 REM ***        by      ***
5 REM ***                ***
6 REM ***  CESAR LOBATO  ***
7 REM **********************
8 REM
9 REM Este programa calcula las ace
leraciones de masas por NEWTON
10 CLS
20 REM PARTE DECLARATIVA
30 DIM rez(2),coef(2),oper(8),nomb$
(8):ERASE rez,coef,oper,nomb$
40 DIM masa(2),ang(2),ka(2,3)
50 ERASE masa,ang,ka
60 DIM a(2),b(2),pol(1,2),c(1,2)
70 ERASE pol,a,b,c
80 SYMBOL 253,16,16,124,16,16,0,124
,0:SYMBOL 254,0,24,24,24,24,116,116
,0
90 CLS:DEG:WINDOW#4,1,40,22,25:PAPE
R#4,3 :CLS#4
100 WINDOW#2,1,40,1,5:PAPER#2,3:CLS
#2
110 WINDOW#0, 1,40,6,21:CLS
120 DEG
130 REM CABECERA
140 PRINT#2,"pulsa H y resolvere pr
oblemas          pulse P y te pond
re problemas"
150 b(1)=302:b(2)=192
160 nm1=0:nm=0:nm2=0
170 nm1=nm1+1:IF nm1>4 OR nm1<=0 TH
EN nm1=1:ang(1)=0 ELSE IF nm1=1 THE
N ang(1)=0 ELSE IF nm1=2 OR nm1=3 T
HEN ang(1)=45 ELSE IF nm1=4 THEN an
g(1)=90
180 nm=nm+1:IF nm>4 THEN nm=1:nm2=n
m2+1
190 IF nm2>4 OR nm2<=0 THEN nm2=1:a
ng(2)=0 ELSE IF nm2=1 THEN ang(2)=0
 ELSE IF nm2=2 OR nm2=3 THEN ang(2)
=45 ELSE IF nm2=4 THEN ang(2)=90
200 a(1)=192:a(2)=192*2+32:pol(1,1)
=320:n1=2:n4=1:pol(1,2)=17*16:c(1,1
)=1:c(1,2)=2
210 ka(1,3)=nm1:ka(2,3)=nm2:GOSUB 9
10
220 FOR i=1 TO 500:NEXT i
230 WINDOW#0,1,40,1,25
240 e$=INKEY$
250 IF e$="H" OR e$="h" THEN CLS#2:
ajg=1:GOTO 270 ELSE IF e$="P" OR e$
="p" THEN ajg=1:GOTO 2030 ELSE GOTO
 170
260 REM MENU PRIMARIO
270 CLS#2:CLS#4:SOUND 1,239,50,8:PR
INT#2,SPC(17);"HOLA";SPC(17);SPC(8)
;"Vamos a resolver un problema";SPC
(9);SPC(18);"de";SPC(18);SPC(11);"p
lanos inclinados";SPC(12);SPC(12);"
maquina de ATWOOD"
280 PRINT#4,"masas maximas 2":INPUT
#4,"cuantas masas por favor";n1
290 IF FIX(n1)<>n1 THEN CLS#4:GOTO
280
300 IF n1<1 OR n1>2 THEN CLS#4:GOTO
 280
310 IF n1=1 THEN a(1)=312:n4=0:GOTO
 380
320 a(1)=192:a(2)=192*2+32:c(1,1)=1
:c(1,2)=2:pol(1,1)=(192+192*2+32)/2
:pol(1,2)=15*16:ka(1,3)=2:ka(2,3)=1
:ang(1)=45:ang(2)=45:n4=1
330 REM CUERPO PRINCIPAL 2 MASAS
340 GOSUB 1520
350 GOSUB 1740
360 GOSUB 2290
370 REM MENU 1 MASA
380 CLS#2:CLS#4
390 PRINT#2,"Estatica pulsa E
                    Dinamica pulsa D"
400 e$=INKEY$
410 IF e$="E" OR e$="e" THEN GOTO 4
30 ELSE IF e$="D" OR e$="d" THEN GO
TO 470 ELSE GOTO 400
420 REM MENU ESTATICA
430 CLS#2:CLS#4:s1=0
440 PRINT#2,"Si NO conoces...
                  Angulo pulsa A
                       Coeficiente
de rozamiento pulsa C"
450 e$=INKEY$
460 IF e$="A" OR e$="a" THEN s1=1 E
LSE IF e$="C" OR e$="c" THEN s1=2 E
LSE GOTO 450
470 REM CUERPO PRINCIPAL UNA MASA
480 IF s1=1 OR s1=2 THEN GOSUB 3570
 ELSE GOSUB 1520
490 IF s1=1 THEN GOSUB 3730 ELSE IF
 s1=2 THEN GOSUB 3840 ELSE GOSUB 39
70
500 IF s1=1 THEN ang(1)=ATN(coef(1)
)
510 ang(1)=ROUND(ang(1),3)
520 GOSUB 910
530 IF s1=1 THEN CLS#2:CLS#4:LOCATE
#2,1,1:PRINT#2,"Angulo=arctg(coefic
iento de rozamiento"
540 IF s1=1 THEN LOCATE#2,1,2:PRINT
#2,"Angulo=";ang(1)
550 IF s1=2 THEN GOSUB 910:CLS#2:CL
S#4:LOCATE#2,1,1:PRINT#2,"coeficien
to de rozamiento=                (mas
a1*9.8*sen(angulo)+Fx)/(masa1*9.8*c
os(angulo)+Fy)"
560 ac=(masa(1)*9.8*COS(ang(1))+fy1
):ac=ROUND(ac,3):IF ac=0 THEN ac=5
ELSE ac=(masa(1)*9.8*SIN(ang(1))+fx
1)/ac:ac=ROUND(ac,3)
570 IF (masa(1)*9.8*SIN(ang(1))+fx1
)=0 THEN ac=9
580 IF ac>1 AND s1=2 THEN LOCATE#2,
1,2:PRINT#2,"no existe un coeficien
te de rozamiento  que cumpla las co
ndiciones":GOTO 610
590 ac=ROUND(ac,4)
600 IF s1=2 THEN LOCATE#2,1,4:PRINT
#2,"coeficiento de rozamiento=";ac
610 IF s1<>2 AND S1<>1 THEN CLS#2:C
LS#4:LOCATE#2,1,1:PRINT#2,"("MASA(1
)"*Ac="mas(1)"*9.81*sen("ang(1)")+"
fx1;CHR$(253);"("masa(1)"*9.81*cos(
"ang(1)")+"fy1")"
620 IF s1<>2 AND S1<>1 THEN LOCATE#
2,1,3:PRINT#2,"MASA="masa(1),"ANGUL
O="ang(1):LOCATE#2,1,4:PRINT#2,"Coe
ficiente de rozamiento="coef(1):LOC
ATE#2,1,5:PRINT#2,"Fx1="fx1,"Fy1="f
y1
630 IF s1<>2 AND s1<>1 THEN res=mas
a(1)*9.81*SIN(ang(1))+fx1:fuer=coef
(1)*(masa(1)*9.81*SIN(ang(1))+fy1):
fuer=ROUND(fuer,3):res=ROUND(res,3)
640 IF fuer<0 THEN LOCATE#",1,1:?#2
,"LA MASA SE SALE DEL PLANO":goto 4
79
650 IF ABS(res)<ABS(fuer) THEN suma
=0 ELSE IF res<0 THEN suma=(res+fue
r)/masa(1):suma=ROUND(suma,3) ELSE
suma=(res-fuer)/masa(1):suma=ROUND(
suma,3)
660 IF s1<>2 AND S1<>1 THEN LOCATE#
4,1,3:PRINT#4,"ACELERACION="suma
670 PRINT#4,"pulsa C volver a empez
ar"
680 e$=INKEY$
690 IF e$="C" OR e$="c" THEN GOTO 7
00 ELSE 680
700 RUN
710 RETURN
720 REM ENTRADAS DE FUERZAS ADICION
ALES EXTERIORES
730 CLS#4: LOCATE#4,1,2:PRINT#4,"FU
ERZAS ADICIONALES EXTERIORES
   modulo y angulo pulsa M
        coordenadas pulsa C
              no fureza adicional pu
lsa N"
740 e$=INKEY$
750 IF e$="C" OR e$="c" THEN GOTO 7
60 ELSE IF e$="M" OR e$="m" THEN GO
TO 830 ELSE IF e$="N" OR e$="n" THE
N RETURN ELSE GOTO 740
760 CLS#4:PRINT#4,"Mete coordenada
horizontal de la fuerza"
770 INPUT#4,lola1:lola1=ROUND(lola1
,3)
780 CLS#4:PRINT#4,"Mete coordenada
vertical de la fuerza"
790 INPUT#4,lola2:lola2=ROUND(lola2
,3)
800 IF i=1 THEN fx1=lola1*COS(ang(1
))+lola2*SIN(ang(1)):fy1=lola1*SIN(a
ng(1))+lola2*COS(ang(1)):fx1=ROUND(
fx1,3):fy1=ROUND(fy1,3)
810 IF i=2 THEN fx2=lola1*COS(ang(2
))+lola2*SIN(ang(2)):fy2=lola1*SIN(
ang(2))+lola2*COS(ang(2)):fx1=ROUND
(fx1,3):fy1=ROUND(fy1,3)
820 RETURN
830 CLS#4:PRINT#4,"Mete modulo de l
a fuerza"
840 INPUT#4,modu:modu=ROUND(modu,3)

850 CLS#4:PRINT#4,"Mete angulo que
forma con horizontal     la fuerza"
860 INPUT#4,angu
870 IF i=1 THEN fx1=modu*COS(angu):
fy1=modu*SIN(angu)
880 IF i=2 THEN fx2=modu*COS(angu):
fy2=modu*SIN(angu)
890 fx1=ROUND(fx1,3):fy1=ROUND(fy1,
3):fx2=ROUND(fx2,3):fy2=ROUND(fy2,3
)
900 RETURN
910 REM PINTAR
920 ca=FIX(640/n1)
930 IF ajg=1 THEN CLS:CLS#2:CLS#4 E
LSE WINDOW#0,1,40,6,22:CLS
940 FOR i=1 TO n1
```

LISTA DE SUBRUTINAS

1-9	Nombre
10-120	Parte declarativa
130-250	Cabecera
260-320	Menú
330-360	Cuerpo principal dos masas
370-410	Menú una masa
420-460	Menú estática
470-710	Cuerpo principal una masa
720-900	Entrada de fuerzas adicionales
910-1510	Pintar
1520-1730	Entrada de datos dinámica
1740-2020	Presentación de ecuaciones dos masas
2030-2090	Cuerpo principal poner problema
2100-2280	Comparar resultados
2290-2660	Presentación de soluciones y pasos intermedios dos masas
2670-2990	Generador de problemas
2950-3120	Presentación de problema
3130-3560	Operaciones y entrada de resultado
3570-3720	Entrada de datos estática
3730-3830	Presentación de ecuación y resultados intermedios ángulo
3840-3960	Presentación de ecuación y resultados intermedios coeficiente de rozamiento
3070-4090	Presentación de ecuación y resultados intermedios dinámica una masa
4100-4180	Escribe datos guardados

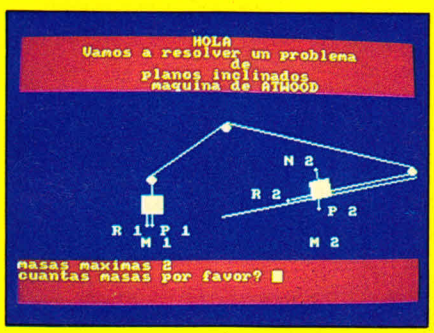
```
950 IF ang(i)<90 THEN GOTO 1060
960 IF n1>1 THEN MOVE a(i)+8,12*16:
TAG:PRINT CHR$(231);:TAGOFF
970 MOVE a(i),10*16:DRAW a(i)+32,10
*16:DRAW a(i)+32,10*16-25:DRAW a(i)
,10*16-25:DRAW a(i),10*16:MOVER 5,-
5:FILL 1
980 MOVE a(i)+16,12*16:DRAWR 0,-3*1
6
990 MOVE a(i),6*16:TAG:PRINT"M";i;:
TAGOFF
1000 MOVE a(i)+30,9*16-30:TAG:PRINT
"P";i;:TAGOFF
1010 MOVE a(i)-46,9*16-30:TAG:PRINT
"R";i;:TAGOFF
1020 MOVE a(i)+8,10*16-25:DRAWR 0,-
20:PLOT a(i)+7,10*16-44:PLOT a(i)+1
0,10*16-44
1030 MOVE a(i)+16,10*16-25:DRAWR 0,
-20:PLOT a(i)+15,10*16-44:PLOT a(i)
+18,10*16-44
1040 ka(i,1)=a(i)+16:ka(i,2)=12*16
1050 GOTO 1430
1060 IF ang(i)>0 THEN GOTO 1180
1070 MOVE ca*(i-1),9*16-26:DRAW ca*
i,9*16-26
1080 MOVE ca*(i-1)+(FIX((ca-48)/3))
,9*16:DRAWR 32,0:DRAWR 0,-25:DRAWR
-32,0:DRAWR 0,25:MOVER 5,-5:FILL 1
1090 MOVE ca*(i-1)+(2*FIX((ca-32)/3
))+32,9*16-9:TAG:PRINT CHR$(231);:T
AGOFF
1100 MOVE ca*(i-1)+(2*FIX((ca-32)/3
))+32,9*16-9:DRAW ca*(i-1)+(FIX((ca
-48)/3)),9*16-9
1110  MOVE ca*(i-1)+(FIX((ca-48)/3)
),6*16:TAG:PRINT"M";i;:TAGOFF
1120  MOVE ca*(i-1)+(FIX((ca-48)/3)
)+30,9*16-30:TAG:PRINT"P";i;:TAGOFF
1130  MOVE ca*(i-1)+(FIX((ca-48)/3)
)+16,9*16-25:DRAWR  0,-20:PLOTR 2,1
:PLOTR -4,0
1140  MOVE ca*(i-1)+(FIX((ca-48)/3)
)-48,9*16+34:TAG:PRINT"N";i;:TAGOFF
1150  MOVE ca*(i-1)+(FIX((ca-48)/3)
)+16,9*16:DRAWR  0,20:PLOT ca*(i-1)
+(FIX((ca-48)/3))+14,9*16+18:PLOT c
a*(i-1)+(FIX((ca-48)/3))+18,9*16+18
1160 ka(i,1)=ca*(i-1)+(2*FIX((ca-32
)/3))+40:ka(i,2)=9*16-9
1170 GOTO 1430
1180 IF ka(i,3)=2 THEN GOTO 1310
1190 MOVE ca*(i-1),12*16:DRAW ca*i,
8*16
1200 MOVE FIX(ca*(i-1)+(ca-35)/2),1
2*16-FIX(((3*16/ca)*(ca-32)/2))-5:D
RAWR 5,(ca/(3*16))*5:DRAWR 30,-(((3
*16)/ca)*30):DRAWR -5,-(ca/(3*16))*
5:DRAWR -30,(((3*16)/ca)*30):MOVER
5,5:FILL 1
1210 MOVE FIX(ca*(i-1)+(ca-35)/2),6
*16:TAG:PRINT"M";i;:TAGOFF
1220 MOVE ca*(i-1),13*16:TAG:PRINT
CHR$(231);:TAGOFF
1230 MOVE ca*(i-1),13*16-8:DRAW ca*
(i-1)+(((ca-35)/2)+3)*1.4,13*16-8-(
4*16)/ca*((((ca-35)/2)+3)*1.4):DRAW
R -1,1:DRAWR 0,-3
1240 MOVER 10,16:TAG:PRINT"R";i;:TA
GOFF
1250 MOVE FIX(ca*(i-1)+(ca-35)/2)+2
0,12*16-FIX(((3*16/ca)*(ca-32)/2))-
5+(ca/(3*16))*5-(((3*16)/ca)*15):DR
AWR 5,(ca/(16*3))*5:DRAWR 1,-1:DRAW
R -3,0
1260 MOVER 5,20:TAG:PRINT"N";i;:TAG
OFF
1270 MOVE FIX(ca*(i-1)+(ca-35)/2)+1
5,12*16-FIX(((3*16/ca)*(ca-32)/2))-
(((3*16)/ca)*30)-5:DRAWR 0,-20:PLOT
R -2,2:PLOTR 4,0
1280 MOVE FIX(ca*(i-1)+(ca-35)/2)-1
7,11*16-FIX(((3*16/ca)*(ca-32)/2))-
15:TAG:PRINT"P";i;:TAGOFF
1290 ka(i,1)=ca*(i-1)+8:ka(i,2)=13*
16
1300 GOTO 1430
1310 MOVE ca*(i-1),8*16:DRAW ca*i,1
2*16
1320 MOVE FIX(ca*(i-1)+(ca-25)/2),8
*16+FIX(((4*16/ca)*(ca-25)/2)):DRAW
R -5,(ca/(4*16))*5:DRAWR 30,((4*16)
/ca)*30:DRAWR 5,-(ca/(4*16))*5:DRAW
R -30,-((4*16)/ca)*30
1330 MOVER 5,5 :FILL 1
1340 MOVE FIX(ca*(i-1)+(ca-35)/2),6
*16:TAG:PRINT"M";i;:TAGOFF
1350 IF i=1 THEN MOVE ca*i,13*16:TA
G:PRINT CHR$(231);:TAGOFF ELSE MOVE
 ca*i-16,13*16:TAG:PRINT CHR$(231);
:TAGOFF
1360 MOVE ca*i,13*16-8:DRAW ca*(i-1
)+(ca-((ca-25)/2))*0.6,9*16-8+(64-(
(4*16)/ca*(ca-(ca-25)/2)))*0.6:DRAW
R 2,-1:DRAWR 0,1
1370 MOVER -58,16:TAG:PRINT"R";i;:T
AGOFF:PLOTR -5,-13
1380 MOVE FIX(ca*(i-1)+(ca-25)/2)-5
+15,8*16+FIX(((4*16/ca)*(ca-25)/2))
+(ca/(4*16))*5+((4*16)/ca)*15:DRAWR
 -20*(4*16/ca),20:DRAWR 2,-2:DRAWR
-4,0
1390 MOVER -48,20:TAG:PRINT"N";i;:T
AGOFF
1400 MOVE FIX(ca*(i-1)+(ca-35)/2)+1
5,8*16+FIX(((3*16/ca)*(ca-32)/2))+(
(3*16)/ca)*30+7:DRAWR 0,-20:PLOTR 1
,1:PLOTR -3,1
1410 IF i=1 THEN ka(i,1)=ca*i ELSE
ka(i,1)=ca*i-8
1420 ka(i,2)=13*16:MOVE FIX(ca*(i-1
)+(ca-35)/2)+30,8*16+FIX(((3*16/ca)
*(ca-32)/2))+((3*16)/ca)*30-10:TAG:
PRINT"P";i;:TAGOFF
1430 NEXT i
1440 IF n4=0 THEN WINDOW 1,40,1,25:
RETURN
1450 IF ka(1,3)=2 AND ka(2,3)=3 THE
N WINDOW 1,40,1,25:RETURN
1460 FOR i=1 TO n4
1470 MOVE pol(i,1),pol(i,2):TAG:PRI
NT CHR$(231);:TAGOFF
1480 IF c(i,1)<=n1 THEN MOVE pol(i,
1)+8,pol(i,2):DRAW ka(c(i,1),1),ka(
c(i,1),2) ELSE MOVE pol(i,1)+8,pol(
i,2):DRAW pol(c(i,1)-n1,1),pol(c(i,
1)-n1,2)
1490 IF c(i,2)<=n1 THEN MOVE pol(i,
1)+8,pol(i,2):DRAW ka(c(i,2),1),ka(
c(i,2),2) ELSE MOVE pol(i,1)+8,pol(
i,2):DRAW pol(c(i,2)-n1,1),pol(c(i,
2)-n1,2)
1500 NEXT i
1510 WINDOW 1,40,1,25:RETURN
1520 REM ENTRADA DE MASA,ANGULOS Y
COEFICIENTES DE ROZAMIENTO
1530 p2=0
1540 FOR i=1 TO n1
1550 CLS#2:CLS#4:PRINT#4,"Peso de l
a Masa";i;" en Kg";:INPUT#4,masa(i)
1560 SOUND 1,239,25,6:p1=0
1570 PRINT#2,SPC(6);"Si quieres que
 la masa este en";SPC(4);SPC(12);"e
l Suelo pulsa S";SPC(12);SPC(12);"V
ertical pulsa V";SPC(24);"un Plano
pulsa P";SPC(12);"la decision limit
a el movimiento de la M";
1580 e$=INKEY$
1590 IF e$="P" OR e$="p" THEN p2=p2
+1:p1=2 ELSE IF e$="V" OR e$="v" TH
EN p1=1:p2=0 ELSE IF e$="S" OR e$="
s" THEN p1=4:p2=0 ELSE GOTO 1580
1600 IF p1=2 AND p2=2 THEN p1=3:p2=
0
1610 ka(i,3)=p1
1620 IF p1=1 THEN ang(i)=90 ELSE IF
 p1=2 OR p1=3 THEN CLS#4:INPUT#4,"A
ngulo que forma el plano con el sue
lo",ang(i) ELSE IF p1=4 THEN ang(i)
=0
1630 IF p1>1 THEN CLS#4:INPUT#4,"Co
eficiente de rozamiento",coef(i)
1640 IF ang(i)>90 OR ang(i)<0 THEN
GOTO 1620
1650 IF ang(i)=90 THEN p1=1:p2=0 EL
SE IF ang(i)=0 THEN p1=4:p2=0
1660 IF masa(i)<0 THEN GOTO 1550
1670 IF coef(i)<0 OR coef(i)>=1 THE
N GOTO 1630
1680 masa(i)=ROUND(masa(i),3)
1690 coef(i)=ROUND(coef(i),4)
1700 ang(i)=ROUND(ang(i),3)
1710 IF p1>1 THEN GOSUB 720
1720 SOUND 1,190,20,6:NEXT i
1730 RETURN
1740 REM SALIDA DE DATOS 2 MASAS
1750 CLS
1760 h1$="(M1+M2)Ac=Res"+CHR$(253)+
"Fuer"
1770 h2$="Fy?=fuerza esterna masa?
sentido "+CHR$(254)+" mov"
1780 LOCATE 16,1:PRINT"NOTACION"
1790 LOCATE 1,2:PRINT"C=Coeficiente
 de rozamiento"
1800 LOCATE 1,3:PRINT"A=Angulo en g
rados"
1810 LOCATE 1,4:PRINT"Fx?=Fuerza es
terna en masa? sentido mov"
1820 LOCATE 1,5:PRINT h2$
1830 LOCATE 1,6:PRINT CHR$(253);" e
s + si Res<0 y - si Res>0"
1840 LOCATE 1,7:PRINT "g=Gravedad=9
.81 m/(sg^2)
1850 LOCATE 1,8:PRINT "Ac=Aceleraci
on incognita"
1860 LOCATE 1,9:PRINT "M?=Masa ? en
 Kg"
1870 LOCATE 14,10:PRINT "**********
**"
1880 LOCATE 14,11:PRINT "*ECUACIONE
S*"
1890 LOCATE 14,12:PRINT "**********
**"
1900 LOCATE 1,13:PRINT h1$
1910 LOCATE 1,14:PRINT "Res=(M1gsen
(A1))+Fx1-(M2gsen(A2))-Fx2"
```

```
1920 LOCATE 1,15:PRINT "Fuer=C1M1gs
en(A1)+Fy1+C2M2gsen(A2)-Fx2"
1930 LOCATE 16,16:PRINT "DONDE"
1940 LOCATE 1,17:PRINT "Res=resulta
nte en la direccion del mov"
1950 LOCATE 1,18:PRINT "Fuer=Ci.Ni"
1960 LOCATE 1,19:PRINT "N=resultant
e en la direccion";CHR$(254);"al mo
v"
1970 LOCATE 18,20::PRINT "NOTA"
1980 LOCATE 1,21:PRINT CHR$(253);"
significa que el roz se opone al mo
v"
1990 LOCATE 1,22:PRINT "recordar qu
e si Res<Fuer entonces Res=0"
2000 LOCATE 1,24:PRINT "Pulsa C Con
tinuar"
2010 e$=INKEY$
2020 IF e$<>"C" AND e$<>"c" THEN 20
10 ELSE RETURN
2030 REM CUERPO PRICIPAL PONER PROB
LEMA
2040 CLEAR  :DEG
 2050 GOSUB 2670
 2060 GOSUB 910
 2070 GOSUB 2950
 2080 GOSUB 3130
 2090 GOSUB 910
 2100 REM COMPARAR RESULTADOS
 2110 IF gua1=1 AND gua2=1 THEN IF A
BS(ABS(angulo)-ABS(ang(1)))<0.03 TH
EN GOTO 2150 ELSE GOTO 2240
 2120 IF gua1=1 AND gua2=2 THEN IF A
BS(ABS(coez)-ABS(coef(1)))<0.008 TH
EN GOTO 2150 ELSE GOTO 2240
 2130 IF gua1=1 AND gua2=3 THEN IF A
BS(ABS(kop)-ABS(fw))<0.02 THEN GOTO
 2150 ELSE GOTO 2240
 2140 IF gua1=2 THEN IF ABS(ABS(kop)
-ABS(fw))<0.02 THEN GOTO 2150 ELSE
GOTO 2240
 2150 SOUND 1,235,25,5:SOUND 1,200,5
0,6:SOUND 1,170,50,6:SOUND 1,140,75
,7:CLS#2:CLS#4
 2160 puntos=puntos+5:j1=8:j2=2:LOCA
TE#2,1,1:PRINT#2,"El problema esta
bien hecho                puntos="
 2170 FOR i=1 TO puntos
 2180 j1=j1+1:IF j1>40 THEN j1=1:j2=
j2+1
 2190 LOCATE#2,j1,j2:PRINT#2,"*"
 2200 NEXT i
 2210 SOUND 1,200,25,6:PRINT#4,"puls
a O otro problema
 pulsa M iniciar el programa"
 2220 e$=INKEY$
 2230 IF e$="O" OR e$="o" THEN GOTO
 2050 ELSE IF e$="M" OR e$="m" THEN
RUN ELSE GOTO 2220
 2240 SOUND 1,235,25,5:SOUND 1,280,3
0,6:SOUND 1,310,49,7:SOUND 1,340,60
,8
 2250 CLS#2:CLS#4:PRINT#2,"LO HAS HE
CHO MAL                   puls
a R ver Resultado
 pulsa I Intertarlo de nuevo
      pulsa M iniciar programa"
 2260 e$=INKEY$
 2270 IF e$<>"R" AND e$<>"r" THEN GO
TO 2280 ELSE IF n1=1 THEN GOTO 530
ELSE IF n1=2 GOTO 350
 2280 IF e$="I" OR e$="i" THEN GOTO
 2050 ELSE IF e$="M" OR e$="m" THEN
RUN ELSE GOTO 2260
 2290 REM PRESENTACION DE REULTADO Y
  PASOS INTERMEDIOS DOS MASAS
 2300 CLS
 2310 h1$="(M1+M2)Ac=Res"+CHR$(253)+
"Fuer"
 2320 h2$="Fy?=fuerza esterna masa?
sentido "+CHR$(254)+" mov"
 2330 LOCATE 14,10:PRINT "**********
**"
 2340 LOCATE 14,11:PRINT "*ECUACIONE
S*"
 2350 LOCATE 14,12:PRINT "**********
**"
 2360 LOCATE 1,13:PRINT h1$
 2370 LOCATE 1,14:PRINT "Res=(M1gsen
(A1))+Fx1-(M2gsen(A2))-Fx2"
 2380 LOCATE 1,15:PRINT "Fuer=C1M1gs
en(A1)+Fy1+C2M2gsen(A2)-Fx2"
 2390 LOCATE 16,16:PRINT "DONDE"
 2400 LOCATE 1,17:PRINT "Res=resulta
nte en la direccion del mov"
 2410 LOCATE 1,18:PRINT "Fuer=Ci.Ni"
 2420 LOCATE 1,19:PRINT "N=resultant
e en la direccion";CHR$(254);"al mo
v"
 2430 LOCATE 18,20::PRINT "NOTA"
 2440 LOCATE 1,21:PRINT CHR$(253);"
significa que el roz se opone al mo
v"
 2450 LOCATE 1,22:PRINT "recordar qu
e si Res<Fuer entonces Res=0"
 2460 g2$="Res=(":g5$="*"+"9.8":g6$=
"-":g7$="*"+"9.8"+"-"
 2470 g3$="Fuer=":g9$="*(":g10$="*"+
"9.8":g11$=")+":g15$=")"
 2480 res=masa(1)*9.81+fx1-masa(2)*9
.81-fx2:res=ROUND(res,3)
 2490 fuer=coef(1)*(masa(1)*9.81+fy1
)+coef(2)*(masa(2)*9.81+fy2):fuer=R
OUND(fuer,3)
 2500 LOCATE 1,3: PRINT "(";masa(1);
"+";masa(2);")";"Ac=";res;CHR$(253)
;fuer
 2510 LOCATE 1,1:PRINT"RESULTADOS IN
TERMEDIOS"
 2520 LOCATE 1,5: PRINT g2$;masa(1);
g5$;"+";fx1;g6$;masa(2);g5$;"-";fx2
;")"
 2530 LOCATE 1,6:PRINT g3$;coef(1);g
9$;masa(1);g10$;"+";fy1;g11$;coef(2
);g9$;masa(2);g10$;"+";fy2;g11$;g15
$
 2540 IF ABS(res)<ABS(fuer) THEN res
=0 ELSE IF res<0 THEN res=res+fuer
ELSE res=res-fuer
 2550 LOCATE#4, 1,3:PRINT#4,"pulsa C
 Continuar"
 2560 e$=INKEY$
 2570 IF e$<>"C" AND e$<>"c" THEN GO
TO 2560
 2580 GOSUB 910
 2590 LOCATE#2, 15,1:PRINT#2,"RESULT
ADOS"
 2600 ana=res/(masa(1)+masa(2)):ana=
ROUND(ana,3)
 2610 LOCATE#2, 1,5:PRINT#2,"La acle
racion de la masa1  es=";USING"###.
###";ana
 2620 LOCATE#2, 1,4:PRINT#2,"La acle
racion de la masa2 es=";USING"###.#
##";-1*ana
 2630 LOCATE#4, 1,3:PRINT#4,"pulsa V
 Volver a empezar"
 2640 e$=INKEY$
 2650 IF e$="V" OR e$="v" THEN GOTO
 2660 ELSE 2640
 2660 RUN
```

LISTA DE VARIABLES

MASA (?)	Valor de la masa ? en Kg
ANG (?)	Valor del ángulo ? en grados
COEF (?)	Valor del coeficiente de rozamiento ?
Fx?	Fuerza en la dirección del movimiento
Fy?	Fuerza en la dirección perpendicular al movimiento
A (?)	Coordenada horizontal masa ?
B (?)	Distancia entre las ? masas
C (?,2)	Conexiones de la polea ?
POL	Guarda las coordenadas de la polea
CA	Distancia reservada para dibujo de cada masa
ajg	Indica modo de ventana 0
ka (?,1.3)	Coordenadas donde se unirá la polea y tipo de suelo
n1	Número de masas
n4	Número de poleas
NM1 NM2}	Generan la sucesión de pantallas de la cabecera
NM	
P1	Dice si se pulsa vertical, plano o suelo
P2	Dice si el anterior ha sido plano inclinado
KOP KOP1 KOP2 AC} ANA	Utilizadas en los cálculos
RES	
FUER	
GUA1	Tipo de problema una o dos masas
GUA2	Tipo de problema una masa
VEA	Dice si el problema generado tiene solución numérica
MODU ANGU }	Empleadas en la construcción de fx1, fx2, fy1, fy2
LOLA1 LOLA2 ANGULO	
FW}	Guardan la solución que da el usuario
COEZ	
PUNTOS	Los puntos que saca el usuario, 5 puntos problema bien hecho
J1 }	Marcan las posiciones de las * que indican los puntos que has hecho
J2	
S1	Informa de si se ha pedido ángulo o coeficiente
OPER	Guarda los valores numéricos que quieras
OPERA	Valor de la operación realizada
OPER1	Valor primer operando
OPER2	Valor segundo operando
NOMB$	Nombre del dato que guardas

```
2670 REM GENERADOR DE PROBLEMAS
2680 RANDOMIZE TIME
2690 vea=0
2700 FOR i=1 TO 2
2710 ang(i)=RND*90:ang(i)=ROUND(ang
(i),3)
2720 IF ang(i)=0 THEN ka(i,3)=4 ELS
E IF ang(i)=90 THEN ka(i,3)=1 ELSE
IF i=1 THEN ka(i,3)=2 ELSE ka(i,3)=
3
2730 coef(i)=RND:coef(i)=ROUND(coef
(i),4)
2740 masa(i)=RND*100:masa(i)=ROUND(
masa(i),3)
2750 IF masa(i)=0 THEN vea=1
2760 NEXT i
2770 IF vea=1 THEN GOTO 2690
2780 fx1=RND*10*9.8:fx1=ROUND(fx1,3
)
2790 fx2=RND*10*9.8:fx2=ROUND(fx2,3
)
2800 fy1=RND*9.8:fy1=ROUND(fy1,3)
2810 fy2=RND*9.8:fy2=ROUND(fy2,3)
2820 gua1=RND*100
2830 gua2=RND*100
2840 IF gua1<25 THEN gua1=2:gua2=0
ELSE IF gua2<66 AND gua2>33 THEN gu
a1=1:gua2=1 ELSE IF gua2<33 THEN gu
a1=1:gua2=2 ELSE IF gua2>66 THEN gu
a1=1:gua2=3
2850 s1=gua2:n1=gua1:n4=n1-1
2860 IF gua1=1 THEN masa(2)=0:ang(2
)=0:coef(2)=0:fx2=0:fy2=0
2870 IF n4=1 THEN pol(1,1)=b(2)+(b(
2)+32)/2:pol(1,2)=16*16:c(1,1)=1:c(
1,2)=2
2880 IF gua1=2 THEN kop1=(masa(1)*9
.81*SIN(ang(1))+fx1-masa(2)*9.81*SI
N(ang(2))-fx2):kop2=coef(1)*(masa(1
)*9.81*COS(ang(1))+fy1)+coef(2)*(ma
sa(2)*9.81*COS(ang(2))+fy2):kop1=RO
UND(kop1,3):kop2=ROUND(kop2,3)
2890 IF gua1=1 AND gua2=3 THEN kop1
=masa(1)*9.81*SIN(ang(1))+fx1:kop2=
coef(1)*(masa(1)*9.81*COS(ang(1))+f
y1):kop1=ROUND(kop1,3):kop2=ROUND(k
op2,3)
2900 IF gua1=2 THEN IF ABS(kop1)<ko
p2 THEN kop=0 ELSE kop=(ABS(kop1)-k
op2)/(masa(1)+masa(2))
2910 IF gua2=1 THEN ang(1)=ATN(coef
(1)):ang(1)=ROUND(ang(1),3)
2920 IF gua2=2 THEN IF (masa(1)*9.8
*SIN(ang(1))+fx1)>(masa(1)*9.8*COS(
ang(1))+fy1) THEN GOTO 2700
2930 IF gua2=2 THEN coef(1)=(masa(1
)*9.8*SIN(ang(1))+fx1)/(masa(1)*9.8
*COS(ang(1))+fy1):coef(1)=ROUND(coe
f(1),4)
2940 RETURN
2950 REM PRESENTACION PROBLEMA
2960 CLS#2:CLS#4
2970 IF gua1=1 AND gua2=2 THEN LOCA
TE#2,1,2:PRINT#2,"MASA=";masa(1):LO
CATE#2,1,3:PRINT#2,"ANGULO=";ang(1)
:LOCATE#2,1,4:PRINT#2,"fuer. adicon
al sentido mov.=";fx1
2980 IF gua1=1 AND gua2=2 THEN LOCA
TE#2,1,5:PRINT#2,"fuer. adiconal se
ntido per mov.=";fy1:LOCATE#2,1,1:P
RINT#2,SPC(17);"DATOS"
2990 IF gua1=1 AND gua2=1 THEN LOCA
TE#2,1,2:PRINT#2,"MASA=";masa(1):LO
CATE#2,1,3:PRINT#2,"COEFICIENTE DE
ROZAMIENTO=";coef(1):LOCATE#2,1,1:P
RINT#2,SPC(17);"DATOS"
3000 IF gua1=1 AND gua2=3 THEN LOCA
TE#2,1,2:PRINT#2,"MASA=";masa(1);"A
NGULO=";ang(1):LOCATE#2,1,3:PRINT#2
,"COEFICIENTE DE ROZAMIENTO=;coef(1
):LOCATE#2,1,1:PRINT#2,SPC(17);"DAT
OS"
3010 IF gua1=1 AND gua2=3 THEN LOCA
TE#2,1,4:PRINT#2,"fuerza sentido mo
vimiento=";fx1
3020 IF gua1=1 AND gua2=3 THEN LOCA
TE#2,1,5:PRINT#2,"fuerza perpendicu
lar movimiento=";fy1
3030 IF gua1=2 THEN LOCATE#2,1,1:PR
INT#2,SPC(17);"DATOS":LOCATE#2,1,2:
PRINT#2,"MASA1=";masa(1):LOCATE#2,2
0,2:PRINT#2,"ANGULO1=";ang(1):LOCAT
E#2,1,3:PRINT#2,"MASA2=";masa(2)
3040 IF gua1=2 THEN LOCATE#2,20,3:P
RINT#2,"ANGULO2=";ang(2)
3050 IF gua1=2 THEN LOCATE#2,1,4:PR
INT#2,"COEFICIENTE DE ROZAMIENTO1="
;coef(1):LOCATE#2,1,5:PRINT#2,"COEF
ICIENTE DE ROZAMIENTO2=";coef(2)
3060 PRINT#4,"pulsa C Continuar"
3070 e$=INKEY$:IF e$<>"C" AND e$<>"
c" AND gua1=2  THEN GOTO 3070
3080 IF gua1=2 THEN LOCATE#2,1,2:PR
INT#2,"fuer. adiconal sentido mov1=
";fx1:LOCATE#2,1,3:PRINT#2,"fuer. a
diconal sentido per mov1.=";fy1:LOC
ATE#2,1,4:PRINT#2,"fuer. adiconal s
entido mov2=";fx2
3090 IF gua1=2 THEN LOCATE#2,1,5:PR
INT#2,"fuer. adiconal sentido per m
ov2.=";fy2
3100 IF gua1=2 THEN PRINT#4,"pulsa
C Continuar"
3110 e$=INKEY$:IF e$<>"C" AND e$<>"
c" THEN GOTO 3110
3120 RETURN
3130 REM OPERACIONES Y ENTRADA DE R
ESULTADOS
3140 FOR i=1 TO 5:monb$(i)=" ":oper
(i)=0:NEXT i
3150 WINDOW#5,1,17,6,21:CLS#5
3160 WINDOW#1,19,40,6,12:CLS#1
3170 WINDOW#3,19,40,14,21:CLS#3
3180 FOR i=6 TO 21:LOCATE 18,i:PRIN
T"*":NEXT i
3190 FOR i=19 TO 40:LOCATE i,13:PRI
NT"*":NEXT i
3200 GOSUB 4100
3210 CLS#4: PRINT#4,"pulsa V Ver da
tos                    pulsa O r
ealizar Operaciones                puls
a R escribir Resultado"
3220 e$=INKEY$
3230 IF e$="V" OR e$="v" THEN GOSUB
 2950:GOTO 3150 ELSE IF e$="O" OR e
$="o" THEN GOTO 3290 ELSE IF e$<>"R
```




```
" AND e$<>"r" THEN GOTO 3220
3240 CLS#5:CLS#1
3250 IF gua1=1 AND gua2=1 THEN INPU
T#5,"ANGULO=";angulo:angulo=ROUND(a
ngulo,3)
3260 IF gua1=1 AND gua2=2 THEN INPU
T#5,"COEFICIENTE DE     ROZAMIENTO="
;coez:coez=ROUND(coez,3)
3270 IF gua2=3 OR gua1=2 THEN INPUT
#5,"ACELERACION=";fw
3280 RETURN
3290 LOCATE#5,1,2:PRINT#5," + SUMAR
":LOCATE#5,1,4:PRINT#5," - RESTAR":
LOCATE#5,1,6:PRINT#5," * MULTIPLICA
R":LOCATE#5,1,8:PRINT#5," / DIVIDIR
":LOCATE#5,1,10:PRINT#5," S SENO":L
OCATE#5,1,12:PRINT#5," C COSENO":LO
CATE#5,1,14:PRINT#5," A ARCOTANGENT
E"
3300 CLS#4:PRINT#4,"HAGO LAS OPERAC
IONES DE UNA EN UNA
                                pulsa
 la operacion que queries realizar"
3310 e$=INKEY$
3320 IF e$="*" THEN LOCATE#1,1,3:IN
PUT#1,"METE MULTIPLICANDO",opera1:C
LS#1:LOCATE#1,1,6:PRINT#1,opera1"*"
:LOCATE#1,1,3:INPUT#1,"METE MULTIPL
ICADOR",opera2:opera=opera1*opera2:
opera=ROUND(opera,4):CLS#1:LOCATE#1
,1,6:PRINT#1,opera1"*"opera2"="oper
a
3330 IF e$="*" THEN GOTO 3410
3340 IF e$="/" THEN LOCATE#1,1,3:IN
PUT#1,"METE DIVIDENDO",opera1:CLS#1
:LOCATE#1,1,6:PRINT#1,opera1"/":LOC
ATE#1,1,3:INPUT#1,"METE DIVISOR",op
era2:opera=opera1/opera2:opera=ROUN
D(opera,4):CLS#1:LOCATE#1,1,6:PRINT
#1,opera1"/"opera2"="opera:GOTO 341
0
3350 IF e$="+" THEN LOCATE#1,1,3:IN
PUT#1,"METE SUMANDO1=",opera1:CLS#1
:LOCATE#1,1,6:PRINT#1,opera1"+":LOC
ATE#1,1,3:INPUT#1,"METE SUMANDO2=",
opera2:opera=opera1+opera2:opera=RO
UND(opera,4):CLS#1:LOCATE#1,1,8:PRI
NT#1,opera1"+"opera2"="opera:GOTO 3
410
3360 IF e$="-" THEN LOCATE#1,1,3:IN
PUT#1,"METE MINUENDO",opera1:CLS#1:
LOCATE#1,1,6:PRINT#1,opera1"-":LOCA
TE#1,1,3:INPUT#1,"METE SUSTRAENDO",
opera2:opera=opera1-opera2:opera=RO
UND(opera,4):CLS#1:LOCATE#1,1,6:PRI
NT#1,opera1"-"opera2"="opera:GOTO 3
410
3370 IF e$="S" OR e$="s" THEN LOCAT
E#1,1,4:INPUT#1,"METE ANGULO",opera
1:opera=SIN(opera1):opera=ROUND(ope
ra,4):CLS#1:LOCATE#1,1,8:PRINT#1,"S
EN("opera1")="opera:GOTO 3410
3380 IF e$="c" OR e$="C" THEN LOCAT
E#1,1,4:INPUT#1,"METE ANGULO",opera
1:opera=COS(opera1):opera=ROUND(ope
ra,4):CLS#1:LOCATE#1,1,8:PRINT#1,"C
OS("opera1")="opera:GOTO 3410
3390 IF e$="A" OR e$="a" THEN LOCAT
E#1,1,4:INPUT#1,"METE TANGENTE",ope
ra1:opera=ATN(opera1):opera=ROUND(o
pera,4):CLS#1:LOCATE#1,1,8:PRINT#1,
"ARCTG("opera1")="opera :GOTO 3410
3400 GOTO 3310
3410 LOCATE#1,1,12:PRINT#1,"guardo
el dato S/N"
3420 k=0:w1=0
3430 e$=INKEY$
3440 IF e$="S" OR e$="s" THEN GOTO
3450 ELSE IF e$="n" OR e$="N" THEN
GOTO 3150 ELSE GOTO 3430
3450 w1=0
3460 w2=w2+1:IF w2>8 THEN GOTO 3500
3470 LOCATE#1,1,8:INPUT#1,"pulsa el
 nombre que quieres 5 letras";nomb$
(w2):w1=1
3480 oper(w2)=opera
3490 IF w1=1 THEN GOTO 3150
3500 FOR i=2 TO 8
3510 oper(i-1)=oper(i)
3520 nomb$(i-1)=nomb$(i)
3530 NEXT i
3540 LOCATE#1,1,8:INPUT#1,"pulsa el
 nombre que quieres 5 letras";nomb$
(8):w2=8
3550 oper(8)=opera
3560 GOTO 3150
3570 REM ENTRADA DE DATOS ESTATICA
3580 SOUND 1,239,25,6:p1=0
3590 IF s1=1 THEN GOTO 3670
3600 PRINT#2,SPC(6);"Si quieres que
 la masa este en";SPC(4);SPC(12);"e
l Suelo pulsa S";SPC(12);SPC(12);"V
ertical pulsa V";SPC(24);"un Plano
pulsa P";SPC(12);"la decision limit
a el movimiento de la M";
3610 e$=INKEY$
3620 IF e$="P" OR e$="p" THEN p2=p2
+1:p1=2 ELSE IF e$="V" OR e$="v" TH
EN p1=1:p2=0 ELSE IF e$="S" OR e$="
s" THEN p1=4:p2=0 ELSE GOTO 3610
3630 IF p1=2 AND p2=2 THEN p1=3:p2=
0
3640 IF p1=1 THEN ang(1)=90 ELSE IF
 p1=2 OR p1=3 THEN CLS#4:INPUT#4,"A
ngulo que forma el plano con el sue
lo",ang(1):ang(1)=ROUND(ang(1),3) E
LSE IF p1=4 THEN ang(1)=0
3650 IF ang(1)>90 OR ang(1)<0 THEN
GOTO 3640
3660 IF ang(1)=90 THEN p1=1:p2=0 EL
```

```
SE IF ang(1)=0 THEN p1=4:p2=0
3670 IF s1<>2 THEN CLS#4:INPUT#4,"C
oeficiente de rozamiento",coef(1):c
oef(1)=ROUND(coef(1),4)
3680 IF coef(1)>=1 OR coef(1)<0 THE
N IF s1<>2 THEN GOTO 3670
3690 CLS#4:PRINT#4,"Peso de la Masa
1  en Kg";:INPUT#4,masa(1):masa(1)=
ROUND(masa(1),3)
3700 IF masa(1)<0 THEN GOTO 3690
3710 IF s1=2 THEN GOSUB 720
3720 RETURN
3730 REM PRESENTACION DE ECUACIONES
 Y RESULTADOS INTERMEDIOS ANGULO
3740 CLS:LOCATE 14,2:PRINT"********
********                          ***
 ECUACION ***
  *****************"
3750 LOCATE 1,6:PRINT"Angulo=arctg(
Coeficiente de rozamiento)"
3760 LOCATE 6,8:PRINT"**************
***************          ** RESUL
TADOS INTERMEDIOS **            ***
**************************"
3770 ang(1)=ATN(coef(1)):LOCATE 1,1
3:PRINT"Angulo=arctg("coef(1)")"
3780 LOCATE 18,16:PRINT"DATOS"
3790 LOCATE 1,18:PRINT"Coeficiente
de rozamiento="coef(1)
3800 LOCATE 1,19:PRINT"MASA="masa(1
)
3810 PRINT#4,"pulsa C Continuar"
3820 e$=INKEY$
3830 IF e$<>"C" AND e$<>"c" THEN GO
TO 3820 ELSE RETURN
3840 REM PRESENTACION DE ECUACIONES
 Y RESULTADOS INTERMEDIOS COEFICIEN
TE DE ROZAMIENTO
3850 CLS:CLS#4:LOCATE 14,2:PRINT"**
**************
   *** ECUACION ***
       *****************"
3860 LOCATE 1,7:PRINT"Coeficiente d
e rozamiento es igual  C=  (M*9.81s
en(ang)+Fx1)/(M*9.81cos(ang)+fy1)
3870 LOCATE 6,9:PRINT"**************
***************          ** RESUL
TADOS INTERMEDIOS **            ***
**************************"
3880 LOCATE 1,14:PRINT"El coefiente
 de rozamiento C es igual   C=("mas
a(1)"*9.81*sen("ang(1)")+"fx1") /("
masa(1)"*9.81*cos("ang(1)")+"fy1")"
3890  LOCATE 18,17:PRINT"DATOS"
3900 LOCATE 1,18:PRINT"MASA="masa(1
)
3910 LOCATE 1,19:PRINT"Angulo="ang(
1)
3920 LOCATE 1,20:PRINT"Fuerza perpe
ndicular al mov Fy1="fy1
3930 LOCATE 1,21:PRINT"Fuerza direc
cion del mov Fx1="fx1
3940 PRINT#4,"pulsa C Continuar"
3950 e$=INKEY$
3960 IF e$<>"C" AND e$<>"c" THEN GO
TO 3950 ELSE RETURN
3970 REM PRESENTACION DE ECUACIONES
 Y RESULTADOS DINAMICA DE 1 MASA
3980 CLS:CLS#4:LOCATE 14,2:PRINT"**
**************
   *** ECUACION ***
       *****************"
3990 LOCATE 1,7:PRINT"MASA*ACELERAC
ION=M*9.81*sen(ang)+Fx1";CHR$(253);
"C*(M*9.81*cos(ang)+fy1)"
4000 LOCATE 6,9:PRINT"**************
***************          ** RESUL
TADOS INTERMEDIOS **            ***
**************************"
4010 LOCATE 1,14:PRINT masa(1)"*AC=
"masa(1)"*9.8*sen("ang(1)")";CHR$(2
53);coef(1)"("masa(1)"*9.81*cos("an
g(1)"))"
4020  LOCATE 10,17:PRINT"DATOS"
4030 LOCATE 1,18:PRINT"MASA="masa(1
),"Angulo="ang(1)
4040 LOCATE 1,19:PRINT"Coeficiente
de rozamiento="coef(1)
```

Serie ORO

```
4050 LOCATE 1,20:PRINT"Fuerza perpe
ndicular al mov Fy1="fy1
4060 LOCATE 1,21:PRINT"Fuerza direc
cion del mov Fx1="fx1
4070 PRINT#4,"pulsa C Continuar"
4080 e$=INKEY$
4090 IF e$<>"C" AND e$<>"c" THEN GO
TO 4080 ELSE RETURN
4100 REM ESCRIBE DATOS GUARDADOS
4110 WINDOW#3,19,40,14,21:CLS#3:w1=
0
4120 FOR i=1 TO 8
4130 IF w1=1 THEN GOTO 4170
4140 IF oper(i)=0 THEN w1=1:GOTO 41
70 ELSE w1=0
4150 LOCATE#3,3,i:PRINT#3,nomb$(i):
LOCATE#3,9,i:PRINT#3,"=";oper(i)
4160 w1=0
4170 NEXT i
4180 RETURN
```

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

WINTER GAMES

La Olimpiada Blanca nos abre sus puertas. Sobre la nieve y el hielo, los atletas medirán sus fuerzas en pos de la medalla de oro.

Desde la primera edición de los Juegos de Invierno en Chamonix (Francia) en el año 1924, esta competición se ha celebrado 15 veces, la última en Sarajevo *(Yugoslavia)*.

La próxima edición de los juegos tendrá lugar en el año 1988, y la sede será la ciudad de Calgary, en Canadá. Es precisamente esta edición de los Juegos de Invierno la que se convierte en protagonista de un programa para ordenador.

Tomando como base de la competición ocho deportes, la casa Epix ha desarrollado un programa en el que la nieve sirve de marco incomparable a la competición entre los mejores deportistas de todo el mundo.

El programa está contenido en dos cassettes, de forma que cada cara agrupa dos pruebas distintas, quedando distribuidas las mismas de la siguiente manera:

Cara 1.- Bobsled y esquí acrobático.
Cara 2.- Velocidad sobre patines y saltos de esquí.
Cara 3.- Patinaje artístico y patinaje estilo libre.
Cara 4.- Esquí de fondo y tiro con rifle.

También existe una versión en disco, que, además de contener las distintas pruebas, incluye la ceremonia de apertura de los juegos, con el encendido de la llama olímpica y la suelta de palomas.

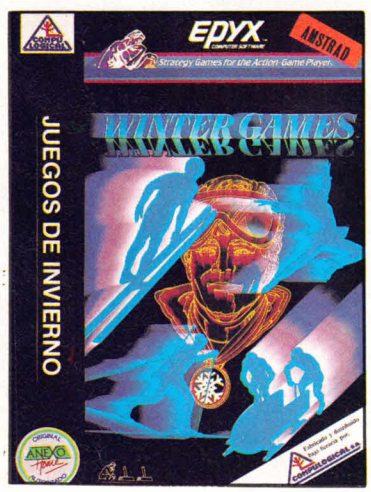
Esta versión permite la competición en todos los juegos, con lo que podemos desafiar a varios amigos en las distintas disciplinas deportivas, para ver quién es el que consigue más medallas a lo largo de toda la olimpiada.

La versión de cassette, por la distribución de los deportes en las distintas caras, sólo permite la competición simultánea en un máximo de dos pruebas.

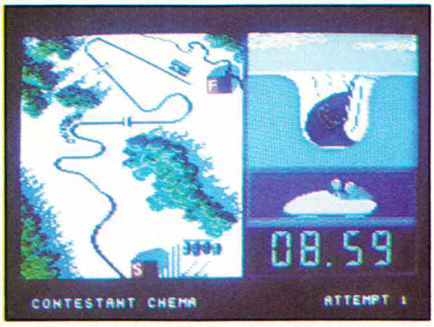
Como el dominio de las distintas disciplinas no es fácil, existe la posibilidad de practicar un juego determinado, lo que tras sucesivos intentos, nos permitirá mejorar nuestro estilo en este deporte.

En las distintas competiciones, pueden practicar desde uno hasta cuatro jugadores, cada uno de los cuales puede elegir entre jugar con teclado o con joystick.

Una vez encendido el fuego olímpico y concluida la ceremonia de apertura, nos encontramos de lleno en la primera competición.



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



Mister JOYSTICK

Bobsled

Puestos al volante de nuestro deslizador, debemos mantener la trayectoria en la pista sin chocar contra los bordes. La gran velocidad a la que nos lanzamos por ella puede hacernos volcar si no tomamos las curvas con precisión.

El conocimiento del circuito es de gran importancia para anticiparnos a las curvas; por la elevada velocidad, éstas aparecen repentinamente, y cualquier retraso en mover el timón puede sacarnos de la pista.

El mejor tiempo en el recorrido será el que decida la medalla de oro.

En esta prueba la pantalla se encuentra dividida en dos secciones, en una de las cuales aparece una vista aérea del circuito con la posición que ocupamos en él, y en la otra vemos nuestro vehículo por detrás y la sección transversal de la pista.

Cada vez que entramos en una curva, la pista cambia de color y nuestro bobsled es empujado hacia la parte alta por la inercia.

Esquí acrobático

Estamos en una prueba donde la habilidad es de vital importancia. Lanzados por una pendiente entramos en un trampolín natural el cual nos eleva en el aire; el fuerte desnivel existente nos deja espacio suficiente para realizar durante el vuelo las figuras deseadas.

Disponemos de seis piruetas diferentes para realizar: el botón de disparo del joystick nos lanza hacia el trampolín, y una vez en el aire efectuamos el movimiento elegido. Cuando caemos a tierra, la posición central del joystick nos hará caer en posición vertical.

Los movimientos clásicos del esquí acrobático están representados; podemos ejecutar la coz de mula, el chiflado, la vuelta hacia atrás, la vuelta adelante, el cisne y el rasgado.

Cuando adquiramos la suficiente destreza podremos realizar varias piruetas en el mismo salto, lo que aumentará considerablemente nuestra puntuación.

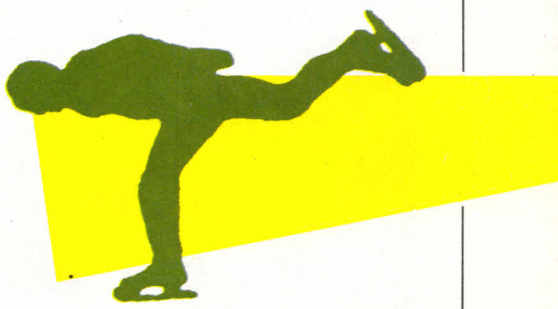
En esta competición, hay que destacar la perfección del movimiento del esquiador y la gracia de las piruetas que ejecuta, las cuales se ajustan exactamente a la realidad.

Patinaje de velocidad

Nuestro patinador, en posición aerodinámica, espera la orden de salida. En la otra pista nuestro adversario intentará batirnos por todos los medios; el que consiga imprimir mayor velocidad a sus patines será el vencedor.

Cuando se da la salida, empezamos a mover las piernas de nuestro patinador: tirando del joystick a izquierda y derecha llevamos el ritmo de las zancadas.

Para lograr un buen deslizamiento sobre la pista, debemos aprovechar adecuadamente el impulso de cada zancada. Si nos retrasamos o adelantamos, al final del movimiento de cada pierna perdemos velocidad.



Para poder vencer a nuestro adversario, debemos salir más despacio, aumentando paulatinamente el ritmo de nuestras zancadas, hasta conseguir aprovechar al máximo el impulso de cada patín y elevar el ritmo a una velocidad aceptable.

Cualquier fallo en la coordinación de los movimientos nos resta velocidad, haciendo que tengamos que recuperar de nuevo el ritmo que llevábamos.

Salto de trampolín

Nos encontramos en la parte superior de la rampa de saltos; cuando nos dan la salida nuestros brazos dan un fuerte tirón y comenzamos el deslizamiento.

Adoptamos una posición baja doblándonos sobre nuestras rodillas, de forma que nuestro cuerpo ofrezca la menor oposición al aire y alcancemos una velocidad más grande.

Esta postura se aguanta hasta el último momento, en el que al llegar al borde del trampolín un enérgico tirón del cuerpo nos lanza al aire.

Ahora nuestra posición es totalmente distinta; estamos extendidos con los esquíes paralelos al cuerpo, planeando sobre la nieve. Al final, otro tirón del cuerpo nos hará aterrizar sobre la blanca masa, marcando la distancia de nuestro salto.

En las pruebas de salto de trampolín, se califican tanto los metros alcanzados en el salto, como el estilo del vuelo; cualquier pérdida de la posición paralela a los esquíes, nos hace perder longitud de salto y penaliza nuestro estilo.

Para realizar un buen salto, debemos hacer una buena salida del trampolín, aprovechando al máximo el impulso del deslizamiento. Luego, en el vuelo, deberemos mantener la posición paralela de los esquíes durante toda la duración de éste, corrigiéndola con el joystick si no estamos en la posición óptima.

Podemos extender las rodillas, adelantar el cuerpo o retrasarlo, y nivelar los esquíes de forma que se mantengan paralelos: el jurado de estilo estará atento a nuestra forma de controlar el vuelo.

A la hora del contacto con la nieve, una fuerte flexión de piernas acompañada de la extensión de los brazos, nos hace tomar tierra sin perder el equilibrio.

Patinaje artístico

Estamos en el programa corto, donde, en el tiempo de un minuto tenemos que realizar seis movimientos obligatorios: Camel, Giro Sentado Doble Axel, Triple Axel, Doble Lutz, Triple Lutz, Camel con Giro Sentado.

El orden en que debemos efectuar las distintas figuras es completamente indiferente, lo importante es enlazarlos convenientemente y realizarlos con la mayor perfección posible.

En la modalidad de patinaje, donde destacan los grandes estilistas, no cuenta la espectacularidad, ni la inspiración que presiden en el estilo libre; lo requerido es un dominio total de la técnica y una perfecta ejecución de las figuras.

Debemos manejar el joystick con sensibilidad y precisión; cada figura que iniciamos debe ir acompañada de la pulsación del botón de disparo en el movimiento adecuado, de la misma forma hemos de acabar el movimiento correctamente sin caernos.

Es muy importante empezar un movimiento con las piernas en la posición correcta, de lo contrario nuestra caída será segura: todo depende de la pulsación del botón de fuego.

Durante todo nuestro ejercicio, el jurado toma nota de nuestros movimientos, calificando la ejecución y penalizando los errores.

Cada caída resta 0.7 puntos y cada movimiento mal ejecutado 0.2; la figura que más puntúa es el Camel Agachado con 1.2, y la máxima puntuación total es de 6.0 puntos, que correspondería a un ejercicio perfecto.

Estilo libre

Estamos en la prueba más espectacular y vistosa del patinaje; en ella, la gracia en la ejecución, la técnica, el riesgo y la belleza se unen para deleitar al público.

Esquí de fondo

Disponemos de dos minutos para realizar el ejercicio: en nuestra imaginación preside el encadenamiento de las distintas figuras, haciendo que la música se funda con el movimiento en una recreación artística de innegable belleza

Para alcanzar el máximo de puntuación, debemos realizar tres veces cada figura, procurando ejecutar dentro del tiempo los tres intentos de tantos movimientos difíciles como nos sea posible.

Las penalizaciones por caída son 0.5 puntos, y 0.2 por movimiento defectuosamente ejecutado.

El máximo de puntos posibles es de 6.9.

Estamos en una prueba donde resistencia y buena puntería se unen. Debemos efectuar un recorrido de fondo sobre nuestros esquíes, y disparar con el fusil a los cinco blancos que encontraremos a lo largo de la pista.

El tiempo empleado en completar el recorrido, es el que decide el campeón de la prueba; cada fallo al disparar sobre un blanco aumenta el tiempo del corredor en cinco segundos.

Con el joystick, dirigimos el movimiento de piernas del esquiador. En esta modalidad lo más importante es conseguir un ritmo adecuado y aprovechar bien el impulso de los esquíes.

En los tramos cuesta arriba, debemos aumentar nuestra velocidad, mientras que si nos encontramos en una pendiente aumentaremos nuestra velocidad de caída ayudándonos de los brazos.

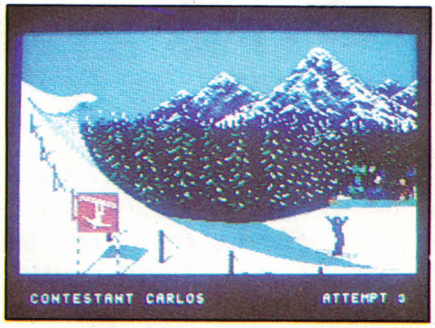
Para disparar a cualquiera de los blancos, debemos correr el cerrojo de nuestro fusil y apuntar. El cansancio del esquiador influye en su puntería y no debemos perder demasiado tiempo apuntando.

Winter Games es un programa que refleja fielmente lo que son los juegos de invierno: las siete pruebas que contiene son las más representativas de esta competición y están reproducidas con verdadero realismo.

Todas las pruebas se realizan a base de joystick, sin tener que utilizar teclas adicionales que complicarían el desarrollo de los ejercicios.

Aunque existen gran variedad de movimientos en cada prueba, el dominio de las distintas técnicas no resulta complicado; con un poco de práctica llega a ser intuitivo y permite todo el desarrollo de nuestras cualidades.

Mister JOYSTICK



La parte más destacable de este programa, sin duda, son el movimiento y el dibujo de los gráficos.

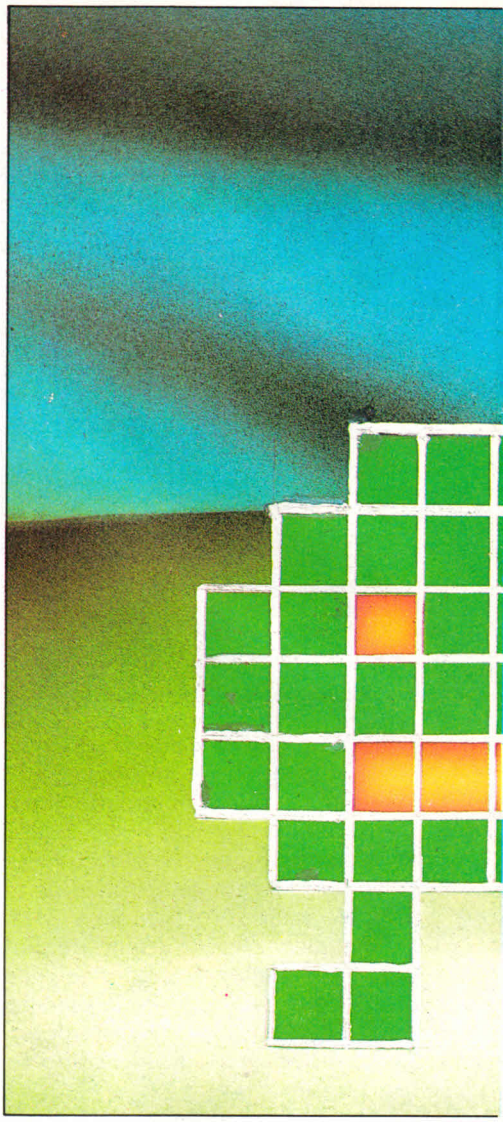
En lo referente al movimiento, éste reproduce totalmente la técnica de cada competición; son de destacar la gracia y fiel reproducción de los movimientos de esquí acrobático, del salto de trampolín, las pruebas de patinaje artístico y el esquí de fondo. Francamente el movimiento está bien conseguido.

En cuanto al dibujo de las distintas pantallas, éste está realizado con una técnica impresionista muy lograda; la nieve se diría que va a salir por la pantalla, los bosques y montañas alpinas nos rodean en un paisaje propio de una postal.

En definitiva, un programa de total espíritu deportivo donde nuestra habilidad se pone a prueba en siete deportes distintos.

GRAFICOS DEFINIDOS POR EL USUARIO

Esta semana centraremos nuestra atención en la forma de definir nuestros propios caracteres gráficos, (gráficos definibles por el usuario), desde lenguaje máquina.



Como todos sabemos, cada carácter viene determinado por 8 bytes, cada uno de los cuales está formado por 8 bits.

Estos últimos son los que indican los puntos que deben estar encendidos o apagados. Así pues, cada carácter estará formado por 64 puntos, cada uno de los cuales puede estar iluminado o no según si el bit está a 1 o a 0.

Para dejar completamente aclarado este punto, veamos cómo definiríamos un carácter gráfico que representara una **cruz**.

Indicaremos a continuación cada uno de los 8 bytes que formarán el gráfico, en notación binaria, decimal y hexadecimal.

La primera de ellas es la que nos dará una visión más clara, del gráfico que deseamos.

Binario	Decimal	Hexadecimal
10000001	129	81
01000010	66	42
00100100	36	24
00011000	24	18
00011000	24	18
00100100	36	24
01000010	66	42
10000001	129	81

De esta forma para obtener dicho gráfico, en el carácter 255 por ejemplo, desde Basic haríamos lo siguiente:

```
10 SYMBOL AFTER 254
20 SYMBOL
255,129,66,36,24,24,36,66,129
```

Para realizar la misma operación desde código máquina, deberemos efectuar una llamada al firmware, para indicar al sistema los caracteres a definir, y qué posición de memoria se desea utilizar para almacenarlos.

Esta rutina es la siguiente:

DEFINICION DE UNA TABLA DE CARACTERES. #BBAB

Reserva un espacio para la creación de caracteres gráficos.

Condiciones de entrada.

El registro doble DE, debe contener el primer carácter de la tabla, y el registro HL debe contener la dirección de inicio de dicha tabla.

```
10 REM **PROGRAMA CARGADOR**
20 FOR N=&A000 TO &A0F3
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&700E THEN PRINT "ERROR
 EN DATAS"
70 DATA 62,1,205,14,188,33,1
80 DATA 1,34,20,160,205,158,160
90 DATA 205,92,160,195,22,160,0
100 DATA 0,205,24,187,42,20,160
110 DATA 125,254,14,40,18,205,80
120 DATA 160,36,44,229,205,125,160
130 DATA 225,34,20,160,205,92,160
140 DATA 24,227,205,24,187,42,20
150 DATA 160,125,254,2,40,216,205
160 DATA 80,160,36,45,229,205,125
170 DATA 160,225,34,20,160,205,92
180 DATA 160,24,227,124,254,37,192
190 DATA 38,1,229,205,108,187,225
200 DATA 201,42,20,160,229,205,117
210 DATA 187,62,251,205,90,187,62
220 DATA 253,205,90,187,225,44,205
230 DATA 117,187,62,252,205,90,187
240 DATA 62,254,205,90,187,201,42
250 DATA 20,160,229,205,117,187,62
260 DATA 32,205,90,187,62,32,205
270 DATA 90,187,225,44,205,117,187
280 DATA 62,32,205,90,187,62,32
290 DATA 205,90,187,201,17,251,0
300 DATA 33,211,160,205,171,187,33
310 DATA 179,160,17,211,160,1,32
320 DATA 0,237,176,201,3,15,31
330 DATA 63,127,127,255,255,255,255
340 DATA 127,127,63,31,15,3,192
350 DATA 240,248,252,254,254,255,25
5
360 DATA 255,255,254,254,252,248,24
0
370 DATA 192,0,0,0,0,0,0
380 DATA 0,0,0,0,0,0,0
390 DATA 0,0,0,0,0,0,0
400 DATA 0,0,0,0,0,0,0
410 DATA 0,0,0,0,0,0,0
```

Si no se había definido una tabla anteriormente, se obtiene:

Carry falso

Registros A y HL corrompidos.

Si ya se había definido una tabla anteriormente, nos devuelve:

Carry a uno

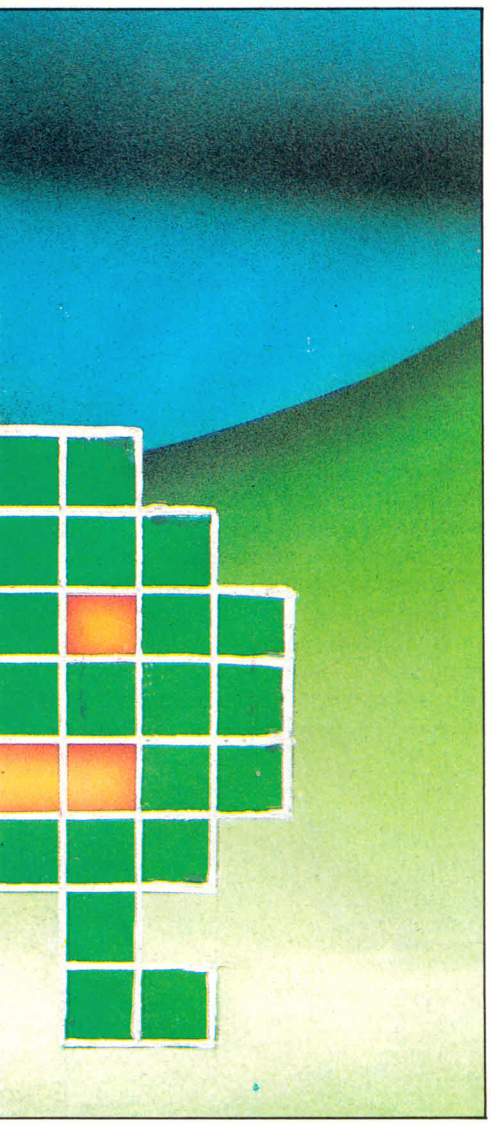
Contiene el primer carácter de la anterior tabla.

HL contiene la dirección de inicio de la antigua tabla.

En cualquier caso, a la salida de esta rutina, se corrompen todos los flags y los registros BC y DE. Los demás son preservados.

Cómo definir un gráfico en lenguaje máquina

Veamos, una vez explicada cómo funciona esta rutina, la forma en que



definiríamos desde máquina el carácter mencionado anteriormente.

En primer lugar, deberemos cargar en el registro doble DE, el código ASCII del primer carácter a definir. Como en nuestro caso es 255, cargaremos DE con dicho valor.

```
LD DE,255
```

A continuación deberemos indicar en HL la dirección de memoria en la cual colocaremos los valores correspondientes a dicho carácter. Esta dirección puede ser cualquiera dentro del rango que nos permita el ordenador, generalmente se deberá utilizar una dirección alta, menor a 42619.

En este caso, podemos colocar la tabla en la dirección 40000.

```
LD HL, 40000
```

Hecho esto, ya sólo queda llamar a la rutina citada anteriormente:

```
CALL #BBAB
```

En estos momentos ya estamos en condiciones de colocar lo correspondientes a nuestro a partir de la dirección 40 lo cual obtendremos nuestro gráfico cuando imprimamos en pantalla el carácter 255.

Para colocar los anteriores valores en esa dirección de memoria, utilizamos la siguiente rutina:

```
LD HL,DATOS ; dirección de
              los valores
LD DE,40000 ; dirección
              donde se de-
              ben colocar
LD BC,8     ; número de
              datos
LDIR        ; efectúa la
              transferencia

DATOS:          DEFB
  129,66,36,24,24,36,66,129
```

```
10 MODE 1
20 y=1:x=1:GOSUB 230
30 IF x=14 THEN GOTO 100
40 IF y=37 THEN y=1:CLS
50 GOSUB 200
60 x=x+1:y=y+1
70 GOSUB 170
80  WHILE INKEY$="":WEND
90 GOTO 30
100 IF x=2 THEN GOTO 30
110 IF y=37 THEN y=1:CLS
120 GOSUB 200
130 x=x-1:y=y+1
140 GOSUB 170
150 WHILE INKEY$="":WEND
160 GOTO 100
170 LOCATE y,x:PRINT CHR$(251);CHR$
(253)
180  LOCATE y,(x+1):PRINT CHR$(252)
;CHR$(254)
190 RETURN
200 LOCATE y,x:PRINT CHR$(32);CHR$(
32)
210  LOCATE y,(x+1):PRINT CHR$(32);
CHR$(32)
220 RETURN
230 SYMBOL AFTER 250
240 SYMBOL 251,3,15,31,63,127,127,2
55,255
250 SYMBOL 252,255,255,127,127,63,3
1,15,3
260 SYMBOL 253,192,240,248,252,254,
254,255,255
270 SYMBOL 254,255,255,254,254,252,
248,240,192
280 RETURN
```

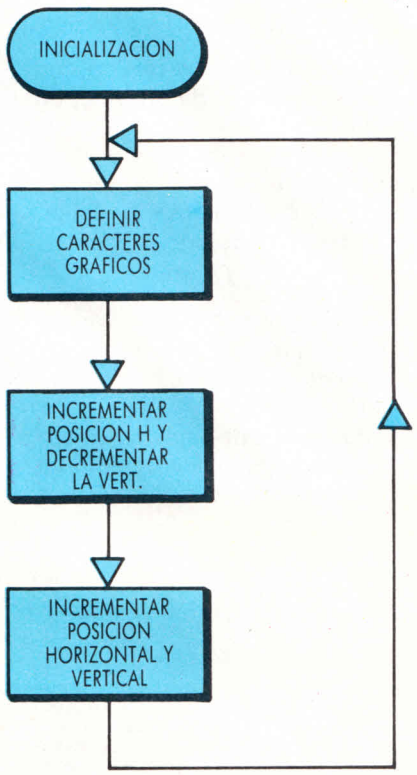
Los programas

Vamos a ver a continuación el programa que hemos preparado esta semana, pero antes deseo hacer algunas indicaciones para aquellos que sigan este curso en posteriores capítulos.

En primer lugar, decir que en los programas realizados en código máquina, se utilizarán siempre que se pueda los mismos nombres para aquellas rutinas que realicen indénticas funciones, así por ejemplo a las rutinas de impresión, se las llamará con el nombre **«PRINT»**.

Otra de las mejoras que deseamos

Código MAQUINA



introducir, es indicar mediante un diagrama de **flujo**, el funcionamiento de los programas, para que de esta forma se vea más claro el método de trabajo de cada una de las rutinas.

Aclarado este punto, empezaremos con la descripción de nuestro programa, en el cual se definen cuatro caracteres gráficos que unidos forman un bola, y a continuación se la hace mover a través de la pantalla.

Lo primero que debemos hacer es colocar la pantalla en el modo de trabajo deseado, en este caso modo 1. Seguidamente se realiza la inicialización de variables.

Dado que en lenguaje máquina tenemos una cantidad limitada de registros, para almacenar nuestras variables utilizaremos direcciones de memoria.

Así pues, cuando deseemos ver el valor de nuestras variables, deberemos observar esas direcciones de memoria. Del mismo modo cuando se desee alterar su valor, éste se tomará de la memoria, se modificará y a continuación se volverá a almacenar allí.

Para hacer esto, se debe reservar un espacio de memoria, al cual le daremos un nombre, para que sepamos en cualquier momento dónde está almacenada nuestra variable.

Esto se puede hacer de la siguiente forma:

```
LD A,50
LD (VARI),A
RET
VARI: DEFS 1
```

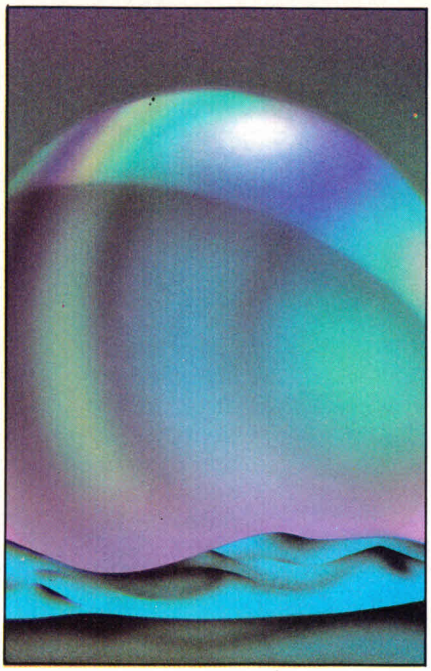
En este caso hemos reservado un espacio de memoria de un byte, en el cual hemos almacenado el valor 50.

Cuando se desee ver el contenido de esa dirección de memoria, únicamente debemos hacer lo siguiente:

```
LD A,(VARI)
```

Una vez hecha la inicialización, se llama a la rutina que crea los caracteres definidos, la cual se ha explicado anteriormente.

Seguidamente entramos en un bucle que se encarga de incrementar la posición horizontal de la bola y decrementar la vertical, hasta que se llegue a un valor determinado.



Una vez alcanzado dicho valor, se envía el control del programa a otro bucle que se encarga de incrementar la posición horizontal y vertical de la bola hasta un cierto valor, alcanzado éste, se envía el programa al bucle anterior.

Las rutinas de impresión y borrado

Por último, se encuentran las rutinas que se encargan de la impresión y borrado de la bola en pantalla. La primera de ellas imprime la bola en la posición de pantalla indicada por la variable «POSIC». La rutina de borrado, se encarga de borrar la posición anterior de la bola.

Para que dicho borrado se efectúe correctamente, se deberá llamar a esta rutina antes de actualizar la posición de la bola, y una vez efectuado el borrado, se renuevan sus coordenadas y se imprime en pantalla.

```
Pass 1 errors: 00

A000              10          ORG  #A000
                  20 ;
                  30 ;INICIALIZACION
                  40 ;
A000  3E01        50          LD   A,1
A002  CD0EBC      60          CALL #BC0E
A005  210101      70          LD   HL,#0101
A008  2214A0      80          LD   (POSIC),HL
A00B  CD9EA0      90          CALL UDGS
A00E  CD5CA0     100          CALL PRINT
A011  C316A0     110          JP   DECRE
A014             120 POSIC:   DEFS 2
                 130 ;
                 140 ;DECREMENTA POSICION
                 150 ;
A016  CD18BB     160 DECRE:   CALL #BB18
A019  2A14A0     170          LD   HL,(POSIC)
A01C  7D         180          LD   A,L
A01D  FE0E       190          CP   14
A01F  2812       200          JR   Z,INCRE
A021  CD50A0     210          CALL MIRHOR
A024  24         220          INC  H
A025  2C         230          INC  L
A026  E5         240          PUSH HL
A027  CD7DA0     250          CALL BORRA
A02A  E1         260          POP  HL
A02B  2214A0     270          LD   (POSIC),HL
A02E  CD5CA0     280          CALL PRINT
A031  18E3       290          JR   DECRE
                 300 ;
                 310 ;INCREMENTA POSICION
                 320 ;
A033  CD18BB     330 INCRE:   CALL #BB18
A036  2A14A0     340          LD   HL,(POSIC)
A039  7D         350          LD   A,L
A03A  FE02       360          CP   2
A03C  28D8       370          JR   Z,DECRE
A03E  CD50A0     380          CALL MIRHOR
A041  24         390          INC  H
A042  2D         400          DEC  L
A043  E5         410          PUSH HL
A044  CD7DA0     420          CALL BORRA
A047  E1         430          POP  HL
A048  2214A0     440          LD   (POSIC),HL
A04B  CD5CA0     450          CALL PRINT
A04E  18E3       460          JR   INCRE
                 470 ;
                 480 ;COMPRUEBA POSICION HORIZONTAL
                 490 ;
A050  7C         500 MIRHOR:  LD   A,H
A051  FE25       510          CP   37
A053  C0         520          RET  NZ
A054  2601       530          LD   H,1
A056  E5         540          PUSH HL
A057  CD6CBB     550          CALL #BB6C
A05A  E1         560          POP  HL
A05B  C9         570          RET
                 580 ;
                 590 ;RUTINA DE IMPRESION
                 600 ;
A05C  2A14A0     610 PRINT:   LD   HL,(POSIC)
A05F  E5         620          PUSH HL
A060  CD75BB     630          CALL #BB75
A063  3EFB       640          LD   A,251
A065  CD5ABB     650          CALL #BB5A
A068  3EFD       660          LD   A,253
A06A  CD5ABB     670          CALL #BB5A
A06D  E1         680          POP  HL
A06E  2C         690          INC  L
A06F  CD75BB     700          CALL #BB75
A072  3EFC       710          LD   A,252
A074  CD5ABB     720          CALL #BB5A
A077  3EFE       730          LD   A,254
A079  CD5ABB     740          CALL #BB5A
A07C  C9         750          RET
                 760 ;
                 770 ;RUTINA DE BORRADO
                 780 ;
A07D  2A14A0     790 BORRA:   LD   HL,(POSIC)
A080  E5         800          PUSH HL
A081  CD75BB     810          CALL #BB75
A084  3E20       820          LD   A,32
A086  CD5ABB     830          CALL #BB5A
A089  3E20       840          LD   A,32
A08B  CD5ABB     850          CALL #BB5A
A08E  E1         860          POP  HL
A08F  2C         870          INC  L
A090  CD75BB     880          CALL #BB75
A093  3E20       890          LD   A,32
A095  CD5ABB     900          CALL #BB5A
A098  3E20       910          LD   A,32
A09A  CD5ABB     920          CALL #BB5A
A09D  C9         930          RET
                 940 ;
                 950 ;CREACION DE CARACTERES
                 960 ;
A09E  11FB00     970 UDGS:    LD   DE,251
A0A1  21D3A0     980          LD   HL,TABLA
A0A4  CDABBB     990          CALL #BBAB
A0A7  21B3A0    1000          LD   HL,DATOS
A0AA  11D3A0    1010          LD   DE,TABLA
A0AD  012000    1020          LD   BC,32
A0B0  EDB0      1030          LDIR
A0B2  C9        1040          RET
A0B3  030F1F3F  1050 DATOS:   DEFB 3,15,31,63,127,127,255,255
A0BB  FFFF7F7F  1060          DEFB 255,255,127,127,63,31,15,3
A0C3  C0F0F8FC  1070          DEFB 192,240,248,252,254,254,255,255
A0CB  FFFFFEFE  1080          DEFB 255,255,254,254,252,248,240,192
A0D3            1090 TABLA:   DEFS 32

Pass 2 errors: 00

BORRA  A07D  DATOS  A0B3  DECRE  A016
INCRE  A033  MIRHOR A050  POSIC  A014
PRINT  A05C  TABLA  A0D3  UDGS   A09E

Table used:   121  from  1000
```

Ofites Informática

Presenta:
el lápiz al que gusta decir SI
mientras nuestros competidores dicen no

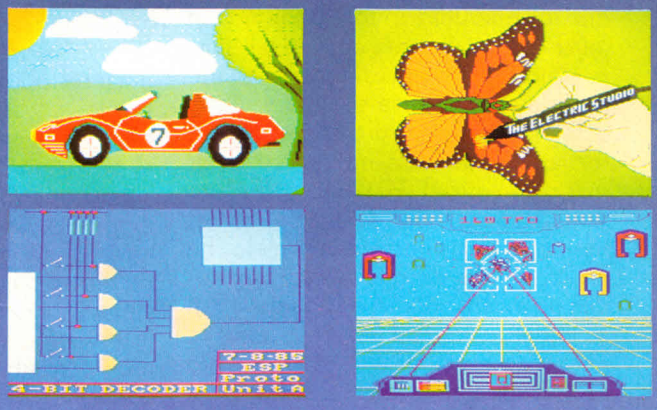
UNICO PARA AMSTRAD, CON PRECISION PIXEL

FUNCIONES	ESP	dk'tronics	OTROS
UNICO MENU DE PANTALLA	SI	NO	
ARRASTRE OBJETOS PANTALLA	SI	NO	
TRASLADO OBJETOS PANTALLA	SI	NO	
TRASLADO DE CURSOR	SI	NO	
CAJAS ELASTICAS	SI	SI	
LINEA ELASTICA	SI	SI	
TRIANGULO ELASTICO	SI	NO	
ELIPSE ELASTICO	SI	NO	
DIAMANTE ELASTICO	SI	NO	
POLIGONO ELASTICO	SI	NO	
HEXAGONO ELASTICO	SI	NO	
OCTOGONO ELASTICO	SI	NO	
CUBO ELASTICO	SI	NO	
PIRAMIDE ELASTICA	SI	NO	
CIRCUNFERENCIAS	SI	SI	
CIRCULOS RELLENOS	SI	NO	
CAJAS RELLENAS	SI	NO	
ELIPSES RELLENAS	SI	NO	
CUÑAS	SI	NO	
SIMULADOR DE CORTES	SI	NO	
DISEÑO DE ZOOM	SI	SI	
IMAGEN ESPEJO E INVERTIDA	SI	NO	
FONDO DE REFERENCIA	SI	NO	
REJILLA DE FONDO	SI	NO	
OPCION DISPLAY X, Y	SI	NO	
RELLENADO CON COLOR	SI	SI	
LAVADO DE COLOR	SI	NO	
VOLCADO PANTALLA RESIDENTE	SI	NO	
DIBUJO DE BORDES EN 3 D	SI	NO	
TEXTO	SI	SI	
9 TAMAÑOS DE BROCHA	SI	NO	
18 TOBERAS MOSTRADORAS	SI	NO	
4 MEZCLAS BASICAS	SI	NO	
VARIADOR DE MEZCLAS	SI	NO	
SOMBREADO DE MEZCLAS XOR	SI	NO	
FICHERO ICONOS RESIDENTES	SI	NO	
FICHERO RELLENOS RESIDENTES	SI	NO	
26 COLORES DE PAPEL	SI	NO	
PALETA DE 15 TONOS DE COLOR	SI	NO	
POSICIONAMIENTO DE PUNTO	SI	SI	
RAYOS DESDE UN PUNTO FIJO	SI	NO	
DIBUJO REFLEJADO (ESPEJO)	SI	NO	
FUNCION HOME	SI	NO	
CONTROL DESDE TECLADO	SI	SI	
CONTROL CON JOYSTICK	SI	NO	
DISPONIBLES MODOS 1 Y 2	SI	?	

Compare con otros lápices

DEBIDO A LA FALTA DE ESPACIO NO PODEMOS LISTAR LAS OTRAS 40 FUNCIONES MAS QUE NUESTRO LAPIZ ES CAPAZ DE HACER.

ESTOS SON ALGUNOS EJEMPLOS DE LOS GRAFICOS QUE VD. PODRA REALIZAR CON NUESTRO LAPIZ OPTICO



DISPONIBLE PARA:

CPC 464 CASSETTE	4.900 Ptas.
CPC 464-664 DISCO	6.900 Ptas.
CPC 6128 DISCO	6.900 Ptas.

(IVA no incluido)

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener el lápiz óptico, puede dirigirse a:

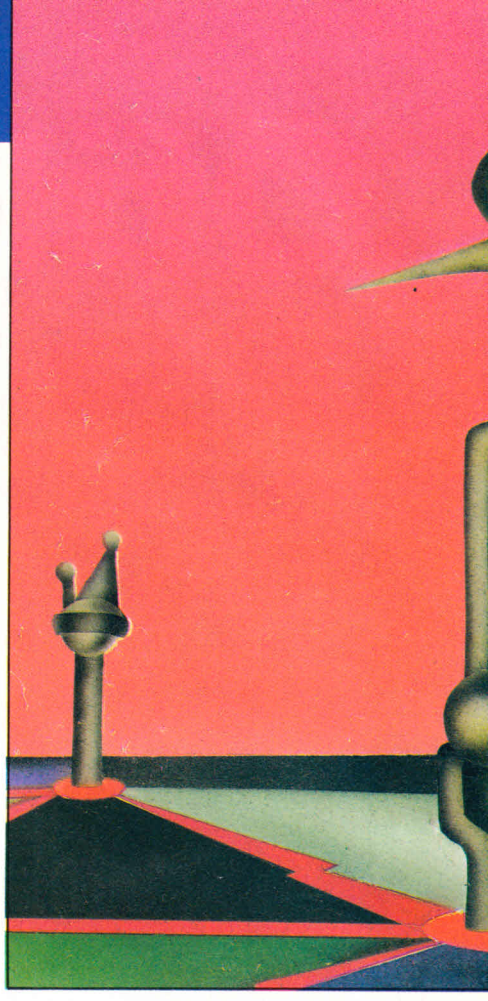
Ofites Informática

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

LA SENTENCIA GOTO ¿PARA QUE?

El GOTO, esa sentencia que nos permite ir de un lado a otro del programa tranquilamente, puede ser un arma de dos filos. Esto se debe a varias razones que serán expuestas a lo largo de este artículo.

Daniel Palomo Ortega



Kernighan y Ritchie, este último es el creador del lenguaje C, le dedican, al GOTO, el siguiente halago: *«utilizar esta sentencia es abusar de ella»;* y hacen la siguiente sugerencia: *«lo mejor es usarla con cuentagotas o no hacerlo nunca.»*

GOTO NO, GRACIAS

La gente que se inició en la programación con basic, casi todos, tienen tendencia a abusar de esta sentencia mucho más de lo necesario. Aunque los que utilizan fortran lo tienen mucho peor, ya que en este lenguaje el no utilizar esta sentencia es imposible. Por esto se suele decir que el GOTO es la **maldición** del basic y fortran. Todo esto no quiere decir que no exista el **GOTO** en otros lenguajes pero sí, que la utilización del mismo es bastante más compleja. En los lenguajes de alto nivel compilados no existen números de línea. Para la utilización del **GOTO** en éstos es necesario el poner una etiqueta, que es una identificación puesta al principio de la línea a la cual se quiere bifurcar. Esto complica mucho más la utilización de esta sentencia.

La versión basic de **Amstrad** es muy potente. Mucho más que la de otros intérpretes, ya que se le ha implementado algunas de las sentencias de otros lenguajes más potentes y estructurados. Por ello nosotros podemos prescindir, en la mayoría de las ocasiones, de la sentencia **GOTO**.

LOS CONTRAS DEL GOTO

La primera oposición que encontramos al **GOTO** es que altera la secuencia lineal del programa, la Instrucción a la que accede más rápido el ordenador, es por lógica la siguiente a la que acaba de ejecutar. Nosotros con un **GOTO** alteramos esa estructura secuencial que utiliza nuestro aparato. Ya que tiene que buscar la nueva instrucción y continuar con la ejecución del programa en ese punto. Esto aunque no lo parezca, le cuesta bastante tiempo y trabajo a nuestro querido intérprete; ya que no es tan rápido como creemos.

Además, un GOTO es muy raro que esté solo en el programa, es decir, su utilización implica que en algún lugar del mismo, vamos a tener que volver a utilizar esta sentencia, para restablecer de nuevo el flujo correcto. Lo que provoca que tengamos que hacer una serie de cruces en el programa.

La abundancia de esta sentencia influye en la legibilidad del programa, ya que nos obliga a ir de un lado a otro dificultando el que veamos claro lo que hace. Por lo que si esta instrucción es muy abundante, es muy probable que muy pronto nos cansemos de intentar entender ese programa.

Posiblemente, si utilizamos mucho el GOTO, alguna vez nos encontremos con que nuestro programa no funciona correctamente, y no sepamos por qué. Al usar esta sentencia debemos saber muy bien dónde bifurcamos y no equivocar el número de línea, y de estructurar muy bien dicha línea para que un posible error no nos lleve mucho tiempo.

El andar dando saltos alegremente por el programa **NO ES ACONSEJABLE Y SE DEBE INTENTAR EVITAR** porque:

es lento,
peligroso,
y poco elegante

Es muy fácil el caer en la tentación del **GOTO,** ya que es una gran comodidad el poder ir directamente al punto del programa que deseamos; y es comprensible que el programador novel guste de utilizar esta sentencia. Pero ésta es inevitable y además el suprimirlo nos ayudará mucho a desarrollar, mucho más, nuestra imaginación programando, algo que bien merece un esfuerzo.

¿COMO EVITARLO?

Lo primero que debemos hacer a la hora de empezar un programa, es saber qué deseamos que haga, cómo lo va a realizar y si podemos hacerlo, esto último depende mucho de nosotros.

La mejor forma de plantearnos un programa, es dividirlo en módulos cortos, que realicen tareas muy concretas; esto se llama estructurar un programa.

Cuando Basic se creó, no se hizo con la intención de que fuese un lenguaje estructurado, sino que se hizo

como un lenguaje de propósito general, fácil de aprender y como una aproximación a la lógica informática, fue el lenguaje que le permitía a cualquier persona programar un ordenador. Pero afortunadamente se le implantó esta posibilidad, y posteriormente se mejoró, esto es posible gracias al uso adecuado de las subrutinas. La mayoría de las veces los **«GOTO's»** pueden ser sustituidos por subrutinas muy fácilmente, esto no rompe la estructura lineal del programa, ya que al utilizar un programa principal, su secuencialidad está garantizada. Además las equivocaciones que podamos tener son mucho más fáciles de corregir, ya que podemos ejecutar cada módulo por separado y hacer las comprobaciones necesarias para su corrección. El uso adecuado de subrutinas es el primer paso en la *«larga carrera»* de evitar el **GOTO.**

Veamos ahora algunos ejemplos prácticos de cómo evitar el GOTO:

1) Elección entre dos alternativas:

```
100 IF A<B THEN A=A+1:J=A*4/3:GOTO 120
110 B=B+1:L=B*6/4
120 .....continúa...
```

Esto es mucho más fácil de entender si lo expresamos utilizando ELSE.

```
100 IF A<B THEN A=A+1:J=A*4/6 ELSE
    B=B+1:L=B*6/4
110....continúa...
```

2) Utilización de un bucle indefinido:

```
90  A=0
100 PRINT «PULSA ENTER PARA FINALIZAR»
110 INPUT «INTRODUCE EL NOMBRE», NOM$(A)
120 INPUT «APELLIDO», APE$(A)
130 A=A+1
140 IF NOM$<>"" OR APE$(A)<>""THEN 100
```

Utilizando la estructura WHILE/WEND lo podemos evitar muy elegantemente.

```
90  A=0
100 WHILE NOM$(A)<>"" OR APE$(A)<>""
110 A=A+1
120 PRINT «PULSA ENTER PARA FINALIZAR»
130 INPUT «INTRODUCE EL NOMBRE», NOM$(A)
140 INPUT «APELLIDO», APE$(A)
150 WEND
```

3) Abandonar la ejecución de un bucle FOR:

```
10  FOR N=1 TO 100
20  IF N MOD 50=0 THEN GOTO 50
30  ?N,N*2
40  NEXT N
50  ..... continúa...
```

Se puede evitar haciendo la variable de control del bucle igual al número mayor que acepta:

```
10  FOR n=1 TO 100
20  IF n MOD 50=0 THEN n=101
30  PRINT n,n*2
40  NEXT
50  ..... continúa...
```

4) Manejar un menú de opciones:

```
10  CLS:C=0
20  INPUT «OPCION»,OP
30  ON OP GOSUB 100, 200, 300, 400
40  IF C=4 THEN END
50  GOTO 10
100 PRINT «SUBRUTINA 1»:RETURN
200 PRINT «SUBRUTINA 2»:RETURN
300 PRINT «SUBRUTINA 3»: RETURN
400 PRINT «SUBRUTINA 4 FINALIZAR»: RETURN
```

De nuevo entra en acción el bucle WHILE/WEND. Aunque pueda parecer extraño el bucle WHILE es capaz de mantenerse después de haber ejecutado una subrutina.

```
10  CLS:c=0
20  WHILE c=0
30  INPUT«OPCION»,OP
40  ON OP GOBUS 100, 200, 300, 400
50  WEND
60  END
100 PRINT «SUBRUTINA 1»:RETURN
200 PRINT «SUBRUTINA 2»:RETURN
300 PRINT «SUBRUTINA 3»:RETURN
400 PRINT «SUBRUTINA 4 FINALIZAR»:C=1:RETURN
```

5) Esperar a que se pulse una tecla

```
1000 PRINT «PARA SEGUIR PULSA UNA TECLA»
1010 IF INKEY$="" THEN 1010
```

la potencia de un bucle WHILE es bastante como para resolver nuestro problema sin tener que recurrir a GOTO.

```
1000 PRINT «PARA SEGUIR PULSA UNA TECLA»
1010 WHILE INKEY$="":WEND
```

En algunas ocasiones no podremos evitarlo, eso no nos debe preocupar, ya que a basic le faltan bastantes de las órdenes de las que disponen otros lenguajes, para evitar la utilización de esta sentencia. Sobre todo para no liar mucho a la máquina y al que se interese por vuestros programas.

Y si algún día trabajáis con otro lenguaje que no acepte tan bien esta instrucción, agradecez a vuestro **Amstrad** el poseer un basic tan potente.

READ TODO SOBRE DATA

Ya vimos cómo programas que no son eficaces a la hora de trabajar, resultaban ser tan rígidos que cuando queríamos cambiar el valor de las variables debíamos reescribir líneas completas.

os ocurría cuando utilizábamos sentencias LET para asignar valores a las variables, así que vamos a intentar ver otros sistemas que den a nuestros programas mayor flexibilidad.

Empezaremos por ver el Programa I.

Programa I

Utilice nuestro familiar bucle FOR... NEXT, en él, la variable de control **«bucle»** que se incrementa en la misma cantidad a cada pasada, va a contener los valores sucesivos que queremos ir sumando en cada giro. Cuando acaba el programa, hacemos que la variable **«suma»** contenga el resultado de la ejecución o suma de los valores de **«bucle»**.

Y para demostrar su flexibilidad cambie la línea 30 por:

30 FOR bucle=4 TO 16 STEP 4

```
10 REM PROGRAMA I
20 suma=0
30 FOR bucle=1 TO 3
40 suma=suma+bucle
50 NEXT bucle
60 PRINT suma
```

¡Ya está! A propósito, la línea 20, en el caso del **Amstrad** es redundante, pero algunos micros si no se especifica el valor inicial interrumpen el programa y dan mensaje de error.

Sin embargo, el programa no es todo lo flexible que deseamos porque no siempre las variables deben tener un incremento (o decremento) fijo. De modo que vamos a utilizar otro sistema.

En esta ocasión se trata de la sentencia INPUT. Su sistema de trabajo es el mismo que el anterior, pero se diferencia en que ahora el **Amstrad** espera que los valores le lleguen desde el teclado. Véalo con el Programa II.

Programa II

La adaptabilidad es fantástica, pero tiene dos desventajas: una es que nos mantiene en el programa hasta que introducimos todos los datos y si son mil...

```
10 REM PROGRAMA II
20 suma=0
30 FOR bucle=1 TO 3
40 INPUT "Numero? ", numero
50 suma=suma+numero
60 NEXT bucle
70 PRINT suma
```

La otra es que debemos teclear todos los datos cada vez que ejecutemos el programa, por lo que un error nos llevaría directamente hacia la **«vuelta a empezar»** o al método de la operación aritmética mental.

Hasta ahora ninguno de los tres sistemas (sentencia LET, bucle FOR... NEXT y comando INPUT) nos ha proporcionado un programa que nos ofrezca la posibilidad de dar cualquier valor a las variables (dentro del rango del micro), que sea flexible y que no necesitemos teclear nada durante su ejecución. De modo que vamos a probar otra cosa.

Y vamos a hacerlo con el Programa III, que no es muy diferente del anterior, pero tiene un par de líneas muy interesantes.

Programa III

La primera es la línea 40 donde aparece la sentencia READ. Trabaja igual que Input pero en vez de esperar el valor desde el teclado, lo busca dentro del propio programa.

La segunda novedad está en la línea 80 que comienza con la palabra clave DATA. Esta sentencia por sí misma no hace nada, sólo sirve para contener los valores que READ va a leer y asignar a las variables correspondientes, por sí misma es completamente inútil.

La mecánica de trabajo es la siguiente: READ provoca que el programa busque una línea que comience por DATA, lee un valor y lo almacena en la variable que sigue al comando READ. El micro se mantiene

```
10 REM PROGRAMA III
20 total=0
30 FOR ciclo=1 TO 10
40 READ cifras
50 total=total+cifras
60 NEXT ciclo
70 PRINT total
80 DATA 12,5,34,6,7
90 DATA 3,67,54,1,2
```

al tanto del orden en la lista de datos y cada vez que obedece una instrucción READ toma el primero de los valores que todavía no ha utilizado.

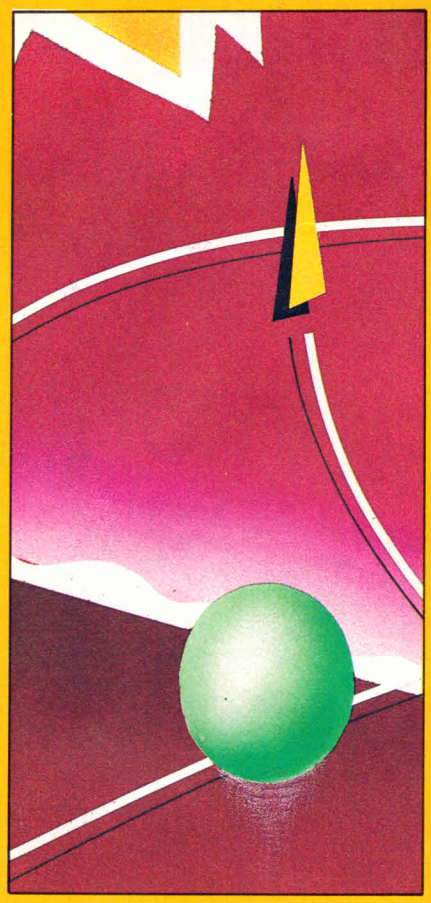
Habrá observado que hay dos líneas DATA. Es simplemente por comodidad, así evitamos errores humanos y no afecta para nada al micro. Cuando termina con todos los datos de una línea busca otra que empiece con DATA y procede de igual modo. Habrá observado, asimismo, que los datos van separados por comas y esto sí que es muy importante. Funcionan como limitadores diciendo al **Amstrad** dónde termina uno y comienza el siguiente.

Si omitimos una coma convertimos dos datos en uno solo, con lo que el micro se verá ante el dilema de encontrar un número de datos dado y con una línea DATA que le proporciona ese número menos uno, lo que le provoca una **«crisis nerviosa»** y parar el programa para darnos una disculpa:

DATA exhausted in 40

Pruébelo cambiando, por ejemplo, la línea 90 a:

90 DATA 3,67,541,2



Primeros repasos

Esto le demuestra que cuando el micro nos señale un error en una línea READ lo que debemos revisar es la línea DATA, que, por otro lado, es la más propicia para introducir errores.

En el caso contrario, si ponemos una coma de más, el micro no nos dará mensaje de error, sino que tomará el número de datos que le hemos indicado, ingnora el resto y obtendremos un resultado erróneo sin ser avisados. Pruébelo, cambie ahora a:

90 DATA 3,6,7,54,1,2

Y este problema puede complicarse aún más en un programa largo que utilice más de un READ, porque en ese caso el puntero quedó señalando al dato que no se utilizó y cuando entre en acción el READ siguiente lo primero que leerá será aquel dato anterior y todo los READs estarán desplazados un valor.

Un último problema se nos presenta si tecleamos una coma al final de una línea DATA. Entonces el **Amstrad** está esperando otro valor y como no lo encuentra asume 0 por defecto. Véalo gráficamente colocando una coma detrás del 7 de la línea 80; el bucle girará 10 veces y no habrá sitio para el último 2 de la línea 90.

Por todo ello sea muy cuidadoso al teclear sentencias DATA en general y con las comas en particular.

Finalmente debemos advertirle que no es necesario situar todos los DATAs juntos y al final del programa. Lo hacemos por claridad, pero pueden distribuirse por todo él, porque el puntero de datos se mantiene al tanto de las cosas. Sin embargo, para programas cortos esta colocación es un buen sistema; para programas largos y complicados es mejor agruparlas en los lugares más lógicos: cerca de la subrutina que los utiliza, por ejemplo.

Si estás en tu sano vicio

éstas son tus revistas

AMSTRAD

Una revista de vanguardia, dinámica, dirigida a todos aquellos que su pasión son los ordenadores Amstrad. Cada semana pequeños trucos y claves que te ayudarán a sacarle el máximo partido a tu ordenador.
Y además, puedes ganar hasta 100.000 pesetas al enviarnos un programa realizado por ti.

MICROHOBBY SEMANAL

Si tienes un Spectrum, ¿dónde encontrar toda la información que necesitas para estar a la última?
MICROHOBBY SEMANAL te la proporciona todos los martes. Juegos, programas de utilidades, periféricos, etc.
Una publicación pensada para los usuarios de los ordenadores Sinclair.

MICROMANIA

Todos los meses, 76 páginas repletas de la información que te interesa.
Para que no te quedes atrás y domines el campo de los ordenadores domésticos.
Para que sepas lo que realmente se está «cociendo» a este nivel.
No te pierdas el número de este mes.

HOBBY PRESS
Para gente inquieta.

Mercado común

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

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

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

Vendo mini órgano & calculadora CASIO VI-tone. Operaciones matemáticas básicas, diez ritmos, cinco instrumentos y la opción sintetizador. Pantalla y altavoz incorporados. Funda protectora, instrucciones en castellano y recambio de pilas. A estrenar, por solamente 3.000 ptas. Interesados llamar al (988) 74 41 38 y preguntar por Dani.

Vendo Amstrad PCW 8256, en garantía, comprado en febrero 86, o cambio por **Amstrad** CPC 128, en perfecto estado, abonándome diferencia a convenir. Interesados escribir a: Luis López Burgueño. Gral. Vives Camino, 3C-2.° A. 19004. Guadalajara.

Desearía contactar con usuarios del **Amstrad** CPC 464 para intercambio de programas de todo tipo, información, ideas, etc. Escribir a: Francisco López Pernas. República Argentina, 5, 7.° C. Castro Urdiales (Santander). Tel. (942) 86 22 95.

Desearía cambiar programas de todo tipo con usuarios de **Amstrad** CPC 464/664/6128. Interesados escribir a: José Luis Parra Bravo. Daoíz, 6-2-12. 29014. Málaga.

Vendo ordenador **Amstrad** CPC 664, monitor color, con programas originales, más diversos juegos y programas de utilidad. Precio todo incluido: 95.000 ptas. José Fernández Montalbán. Avda. de Roma, 19, 3-3. 08029. Barcelona. Tels. (93) 323 02 04 y 317 41 20.

PEQUEÑA ERRATA EN DBASEI

Los que hayáis tecleado el programa DBASEI habréis tenido la desagradable sorpresa de que a partir de 10 fichas el programa da error por matriz no dimensionada.
No os asustéis pues no váis a tener que teclearlo de nuevo entero.
El error, como quizá muchos de vosotros habréis notado, es culpa de una simple vocal:

EN LA LINEA 480 DONDE PONE COMPO$ DEBE PONER CAMPO$
EN LA LINEA 660 SUSTITUIR ROGI$ POR REGI$
EN LA LINEA 1270 DONDE PONE OCME$ DEBE PONER ACME$

Y esto es todo. Introduciendo estas modificaciones el programa funcionará perfectamente. Lamentamos el «sofocón» que os hayan podido ocasionar estas erratas.

Vendo Amstrad CPC 464 color, 20 juegos y utilidades (Knight Lore, Alien 8, Harrier attack, Roland on the ropes, Ensamblador, Desensamblador, copiones...). Comprado 24-6-1985. Perfecto estado. Precio a convenir. José R. Muñoz. Río Miño, 13, 3.° izda. Teruel. (De 7 de la tarde en adelante). Tel. (974) 60 55 64.

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.

EL «RATAS DEL DESIERTO» A DISCO

He comprado la cinta n.º 2 de juegos de estrategia «Ratas del desierto», poseo un CPC 6128 ***Amstrad****, lo he pasado al disco, pero desde éste no me funciona, les acompaño listado para que me indiquen, si puede ser, quéinstrucción debo cambiarle para que funcione con el disco.*

Les aseguro que no se trata de ningún tipo de ***piratería****, sólo que al disponer de disco es más rápido y más cómodo. Les ruego me contesten, tanto si me lo pueden decir como si no.*

Amaro Alcaide

El programa **«Ratas del desierto»**, carga los últimos bloques binarios a través de un cargador en código máquina, por lo que éste está preparado para cargar desde cassette. Así pues, la única forma de conseguir que cargue desde disco, es de corregir dicho cargador de código máquina.

LOS RSX IDISC Y ITAPE

Quiero felicitaros por vuestranuestra revista y preguntaros lo siguiente:

1.º Poseo un CPC 6128, quisiera saber ¿por qué al hacer un RUN'' '' de un programa binario desde disco, que a su vez cargue un segundo programa también en binario desde disco, el ordenador me pide este segundo programa a través del cassette y no del disco? (La rutina para cargar el segundo programa es la publicada en el n.º 25.)

2.º ¿Cómo se puede evitar este cambio de disco a cassette?, porque lo que yo hago es utilizar un programa ***BASIC*** *que cargue el primer programa binario y después un* ***CALL*** *a la dirección inicial, y de esta manera el segundo programa lo pide del disco, ¿hay otro procedimiento?*

3.º ¿Cómo se puede simular el efecto de ***IDISC*** *y* ***ITAPE*** *en código de máquina?*

Quisiera también que me publiquéis el siguiente anuncio:

BUSCO un manual de la impresora CENTRONICS 730-2, la 733-2 o similar. Pagaría fotocopias y gastos de envío. Tel. (91) 445 10 02, tardes o bien por carta a Juan Carlos Plaza. C/ Alberto Aguilera, 16. 28015 Madrid.

Juan Carlos Plaza

La rutina publicada en el n.º 25 de **MICROHOBBY AMSTRAD**, está preparada para funcionar tanto en disco como en cassette, y que funcione en una forma u otra sólo depende del sistema que esté elegido en este momento.

Para que funcione correctamente utilizando el disco, únicamente deberás escribir la siguiente orden Basic antes de llamar a la rutina:

IDISC

La simulación de las dos sentencias mencionada en tu carta desde código máquina, no son posibles mediante llamadas directas, aunque una forma de conseguirlo es llamar directamente a la ROM del disco o del cassette.

BLOQUES PERDIDOS

¡Hola! Soy un orgulloso propietario de un CPC 472. Mi problema es el siguiente:

Tenía grabado un programa de vuestra revista. Un día al cargarlo me confundí y pulsé REC y PLAY a la vez. Se me han borrado los dos primeros bloques y, como es normal, no puedo sacar la información de los restantes. ¿No habría alguna forma de copiar los dos primeros y unirlos a los otros sin necesidad de tener que copiar el programa de nuevo? O, ¿no hay otra forma de sacar la información de los que aún conservo?

Javier Martín

Dado que se ha perdido el primer bloque de programa, no existe ninguna forma desde Basic con la que puedas conseguir el resultado que nos propones, por lo que te recomendamos que en posteriores ocasiones rompas la patilla de las cintas en las que tengas grabado algo importante para que no vuelva a sucederte lo mismo que en esta ocasión.

UTILIZACION DE CARACTERES Y EL PROGRAMA «BIBLIOTECH»

Soy principiante de un CPC-464. Les rogaría me indicasen, lo antes posible:

1.º ¿Cómo se saca el signo ^?

2.º ¿Me pueden transcribir las líneas?

1560
1580
1585

del programa BIBLIOTECH, de su n.º 24, pues están casi borradas?

Un saludo.

Fernando González y Serrano *(Jaén)*

Para poder utilizar este carácter como cualquier otro bastará que hagas lo siguiente:

key 1, ''chr$(160)''

Siendo 160 el código ASCII del carácter que quieres utilizar en este caso, pero si quisieras utilizar otro tendrías tan sólo que modificar el código ASCII que se encuentra en el manual del ordenador.

Las líneas del programa «Bibliotech» son éstas:

1560 for k=0 to 1500:next:locate 5,23:print'' '':return

1580 window#1,4,79,8,21: return

1580 window #2,5,77,15,21:cls#2: locate 43,23:print '' < espacio > ^ < del > fin'':locate 27,13 print'' < < < temas > > > ''

En tu kiosco te espera algo muy inteligente

El AMSTRAD Especial número 2 incluye una cinta de cassette adherida a la portada con un lenguaje Lisp completo que te permitiá comprender y dominar las técnicas más complejas de inteligencia artificial. Por si fuera poco, en nuestra cinta se incluyen también dos concursos: uno, de diseño gráfico de pantallas, para artistas, te permitirá ganar hasta 170.000 pesetas en premios. En el segundo regalamos un ordenador Amstrad CPC6128.

El número 2 de AMSTRAD Especial trata un amplio espectro de interesantes temas, como un comparativo de impresoras, que le ayudará a elegir la más adecuada a sus necesidades, como multitud de programas y rutinas de utilidad en lenguaje máquina como un paquete de soft integrado, con tres programas en uno y un largo etcétera que sería demasiado prolijo detallar.

Si no lo encuentras en tu kiosco, solicítalo directamente a nuestra Editorial.

Rellena este cupón y envíalo a **HOBBY PRESS, S. A.** Ap. de Correos 232. Alcobendas. Madrid.

Nombre _____ Apellidos _____

Domicilio _____

Localidad _____ Provincia _____ C. Postal _____

Teléfono _____ Profesión _____ Fecha de nacimiento _____

¿Eres suscriptor de **MICROHOBBY AMSTRAD?** Sí ☐ No ☐

Deseo recibir el Especial de **MICROHOBBY AMSTRAD** n.º 2 al precio de 475 ptas. (IVA incluido)

FORMA DE PAGO

☐ Talón bancario adjunto a nombre de **HOBBY PRESS, S. A.**

☐ Mediante Tarjeta VISA. N.º ..... Fecha de caducidad .....

☐ Contra reembolso (supone 75 ptas. de gastos de envío).

Fecha y firma:

A.D.1

4X1

paga uno y llevate cuatro

AMSTRAD

COMBAT LINX
GREMLINS
DUMMY RUM
DRAGONTORC
MATCH DAY
BASEBALL
FIGHTING WARRIOR
MAP GAME
YIE AR KUNG-FU
HYPERSPORTS
ZORRO
SUPERTEST
PING-PONG

COMMODORE

BASEBALL
DROPZONE
BEACH HEAD
HYPERSPORTS
SUPER ZAXXON
FIGHTING WARRIOR
SPY HUNTER
TAPPER
BC-II
BOUNTY BOB
POLE POSITION

MSX

DISC WARRIOR
JET SET WILLY II
SHOWJUMPER

SPECTRUM

ZAXXON
FRANKIE
BLUE MAX
BRUCE LEE
RAID OVER MOSCOW
BASEBALL
DRAGONTORC
ASTROCLONE
GYROSCOPE
MAP GAME
ZORRO
COSMIC WARTOAD
N.O.M.A.D.
BATTLE OF PLANETS
DYNAMITE DAN
LEYENDA AMAZONAS
BRIAN BLOODAXE
PSYTRAXX

.... y mil titulos más

¡¡¡ absolutamente originales !!!

500 ptas.

sinclair store

SOMOS PROFESIONALES

BRAVO MURILLO, 2
(Glorieta de Quevedo)
Tel. 446 62 31 - 28015 MADRID
Aparcamiento GRATUITO Magallanes, 1

DIEGO DE LEON, 25
(Esq. Núñez de Balboa)
Tel. 261 88 01 - 28006 MADRID
Aparcamiento GRATUITO Núñez de Balboa, 114

AV. FELIPE II, 12
(Metro Goya)
Tel. 431 32 33 - 28009 MADRID
Aparcamiento GRATUITO Av. Felipe II