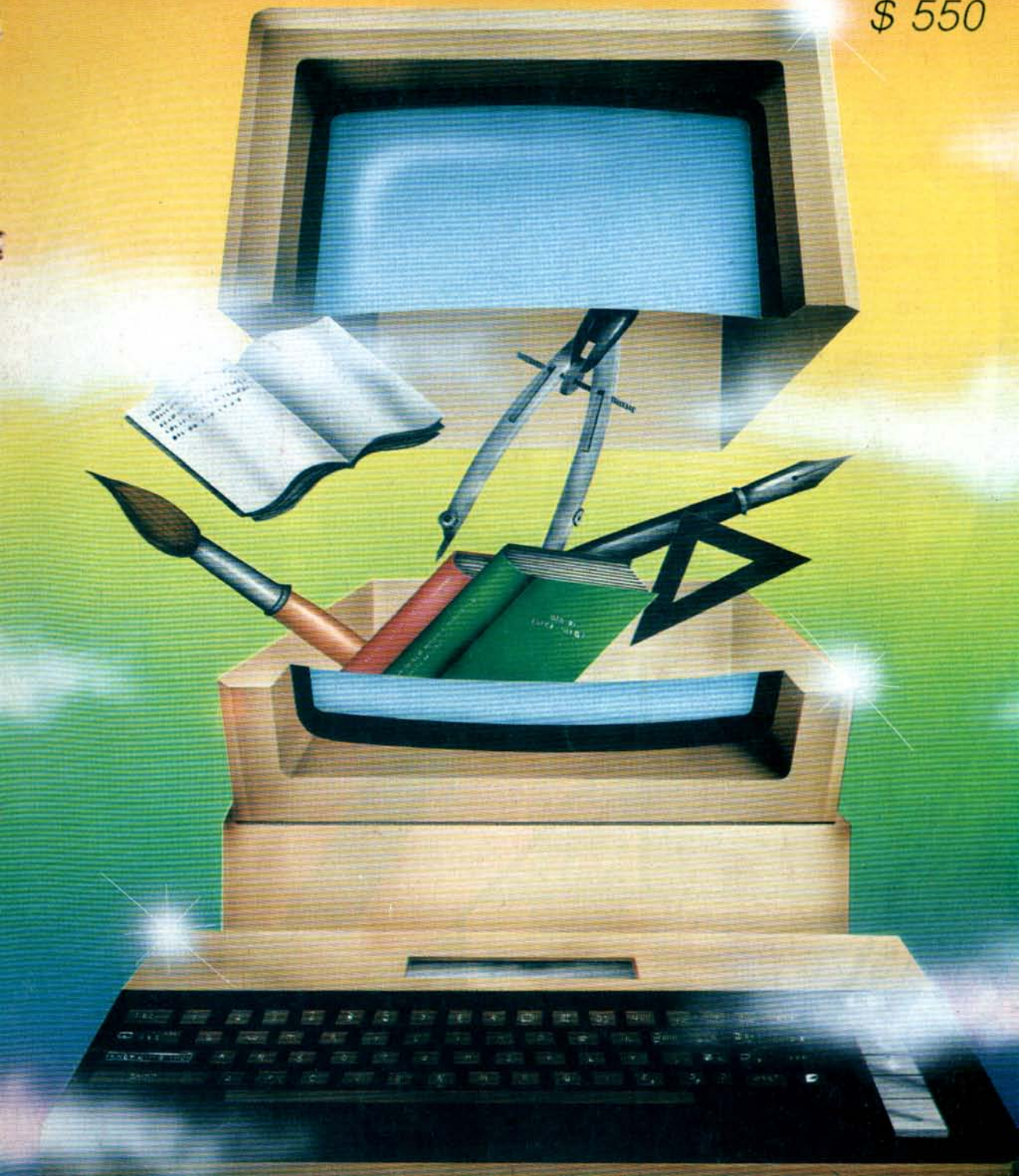
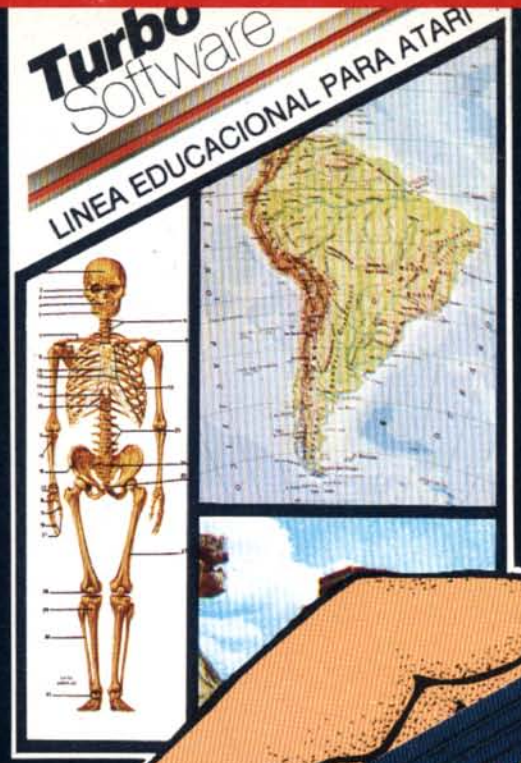
TURBO news

Revista para Computadores ATARI® N° 9 – ABRIL 1990

$ 550





TURBO LIGHT PEN

- El **Turbo Light Pen** es un novedoso periférico para los Computadores Atari XL/XE.
- Con él podrás extender tus horizontes en la generación de gráficos para tus programas, y también utilizarlo como dispositivo de control para juegos.
- **Turbo Light Pen** es muy fácil de usar y no necesita ningún conocimiento previo.
- Con el **Turbo Light Pen** se incluyen como regalo, un programa graficador y un juego de la línea Turbo Software.

LAPIZ OPTICO PARA COMPUTADORES ATARI

OFERTA
PROMOCION INCLUYE 2 CASSETTES

ES OTRO PRODUCTO M.P.M.

ATARI ES MARCA REGISTRADA DE ATARI CORPORATION.

ADQUIERELOS EN LOS SIGUIENTES PUNTOS DE VENTAS

• **ANTOFAGASTA:** COOPERCARAB KW VIDEO LA ESPAÑOLA • **VIÑA DEL MAR:** FALABELLA VIÑA INSIS MPR COMPUTACION • **VALPARAISO:** COMPUTRONIC • **SANTIAGO:** AUDIO BICICLETA INTERNAC CASA ROYAL CENTRO ATARI COMERCIAL ESTADO COMPUMANQUE COMPUCENTER FALABELLA AHUMADA FALABELLA P. ARAUCO IMACO INFOGROUP PC STORE PETERSEN ROLEC SUPERMERCADOS UNIMARC TASCO VIDEO CLUB INTERNACIONAL • **RANCAGUA:** CASA ZUÑIGA • **CURICO:** MULTIHOGAR • **TALCA:** LIBRERIA "EL AHORRO" MULTICENTRO VIDEO CLUB CASSAL • **CHILLAN:** CASA EDISON • **CONCEPCION:** COOPERCARAB DISMAR DISMAR 2 EQUS PHANTER RAPSODIA SESCO • **LOS ANGELES:** DISTRIBUIDORA MERINO • **ANGOL:** SCORPIO • **VICTORIA:** CASA SIGMUND • **TEMUCO:** COMERCIAL MANQUEHUE ESTABLECIMIENTOS GEJMAN FALABELLA • **PUCON:** ELTIT • **VILLARRICA:** JOYERIA KETTERER • **VALDIVIA:** ELECTROMUSICA • **LA UNION:** IMPORTADORA COSMOS • **OSORNO:** CASA REAL FOTO EXPRESS • **PUERTO VARAS:** ELECTRO HORN • **PUERTO MONTT:** COMERCIAL MANQUEHUE

EDITORIAL

Estimados amigos:

Ya el año se encuentra en pleno desarrollo y, como todos los meses, volvemos a reencontrarnos ahora ya instalados en nuestras nuevas oficinas ubicadas en la Avda. Fco. Bilbao 4226 de la comuna de Las Condes, para poder así mejorar nuestra comunicación. Una comunicación que se ha ido incrementando mes a mes. Sus sugerencias e interrogantes estamos tratando de resolverlas. Sobre todo los referidos a cómo conseguir los números atrasados de nuestra Revista. Con la nueva ubicación esperamos poder recibirlos mejor, para entregarles así las viejas ediciones. Ediciones que no pierden vigencia con el paso del tiempo y que son necesarias para darle una continuidad a los cursos que aquí se están desarrollando.

En este número hemos incorporado el primer capítulo de un curso de protección de Software, que esperamos sea de su agrado. Siguiendo sus sugerencias, hemos incrementado el número de programas para tipear en el computador.

CONTENIDO

TURBO news

Circulación Mensual, Nacional e Internacional.
Destinada a los usuarios de computadores ATARI (R) como material didáctico de Programación. TURBO news (R) es una publicación de EDITORA TURBO LTDA. Domicilio: Av. Fco. Bilbao 4226 - Teléfono: 486506.

DIRECTOR RESPONSABLE: Mauro Pieressa. REPRESENTANTE LEGAL: Marcelo Waldbaum. PRODUCCION: Marcelo Waldbaum y Mauro Pieressa, Programadores y Diseñadores de Computación. DIRECTORA DE ARTE: Odalí Guerrero L. CORRECTOR: Marcial Valenzuela S. PUBLICIDAD Y RR.PP.: Liliana Muñoz Otárola, Hernán Vittini. COLABORACION: Mariana Pizarro. PUZZLE: Mario Calvo A. FOTOCOMPOSICION: Brubytes. IMPRESION: Servigraf. DISTRIBUCION: Alfa Ltda. Agradecemos la colaboración de COELSA S.A. Centro Atari. (Augusto Leguía Sur 75). Profesor Emilio Antileff. Atari es marca registrada de ATARI CORPORATION. TURBO news es marca registrada de EDITORA TURBO LTDA. (Reg. Marca N° 342428 9-05-89).

ENTREVISTAMOS AL

COLEGIO OMEGA

En este número, hemos entrevistado al director de otro de los colegios que han elegido al computador Atari, para ingresar a sus alumnos al fascinante mundo de la computación.

Todo comenzó hace cuatro años, comenta el director del colegio Omega, Don Sergio Alvarez Hidalgo, cuando 4 profesores del establecimiento, realizaron un curso intensivo organizado por la Universidad Católica de Chile y, por supuesto, se basó en los computadores Atari.

El mismo, nos cuenta, duró todo el mes de enero de ese año y se basó en los principios del lenguaje Logo y del lenguaje Basic. Fue un curso dirigido únicamente para docentes. Al final del curso, se tomó la decisión de adquirir dos computadores Atari 800 XL, los cuales al corto tiempo se hicieron insuficientes ante el gran interés despertado entre el alumnado y se decidió por la compra de otros dos.

Los cursos que aquí se brindan son totalmente voluntarios. Se desarrollan dos veces por semana, incluido uno el día sábado. Como se destina un computador por cada dos personas, generalmente se debe crear una lista de espera para poder ingresar a los cursos.

Se enseña el lenguaje Logo para los niños de la enseñanza Básica y el lenguaje Basic para los de la enseñanza Media.

Los niños más pequeños mostraron un interés mucho mayor que los mayores, demostrado por el hecho que un 70% de éstos manifestaron interés en los cursos, cifra que bajaba hasta un 20% entre los de educación Media. Claro que estos últimos cuando se ponían frente al computador, no había quien los detenga.

El método de enseñanza utilizado se podría sintetizar

de la siguiente manera:
- Primero, se muestran las instrucciones generales.
- Segundo, se hacen programas en papel y lápiz.
- Tercero, se verifican en el computador.
- Cuarto, se deben encontrar y analizar los errores aprendiendo de ellos.

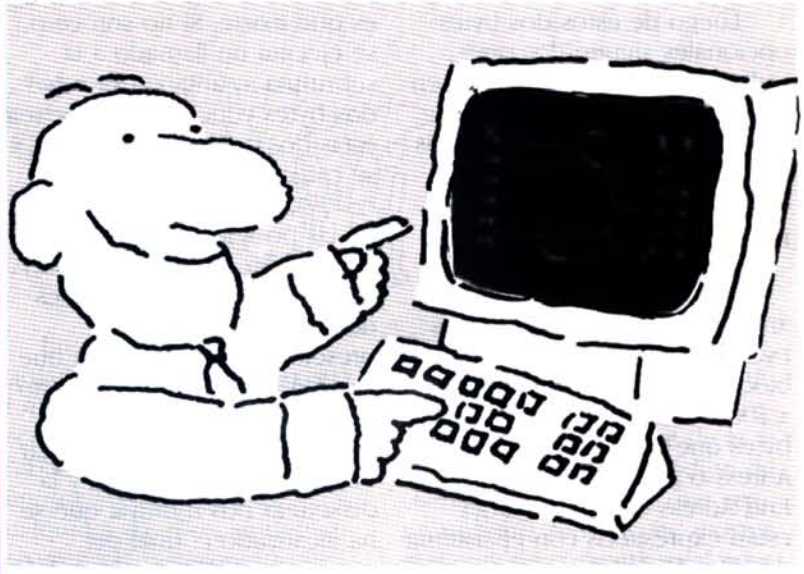
No se utiliza la calificación sobre los trabajos, simplemente se busca que, cada cual a su medida, vaya incorporando los conceptos de la computación en base a su propia creatividad.

Los resultados logrados son sorprendentes, cuenta orgulloso el Director del Establecimiento y nos cuenta cómo, un niño realizando sencillos ejercicios con figuras geométricas en lenguaje Logo, terminó dibujando una iglesia.

También logramos beneficios extras, comenta, como ser que los niños a través del Logo, empiecen a manejar ángulos, por lo que al llegar a esa etapa de las matemáticas, pueden resolverlo con mucha facilidad.

Sin duda, concluye, el computador nos ha facilitado mucho la tarea de educar a los niños.

Files Binarios



Muchas veces los Atarianos nos encontramos investigando los sistemas de cargas de programas para agilizarlos o bien para lograr que determinado archivo que carga de diskette pueda hacerlo también desde el cassette.

Al no existir mucha documentación en los manuales del computador y al recibir muchas consultas sobre el tema, incorporamos en este artículo una primera introducción acerca de la carga de programas con la estructura de Files Binarios.

Un File Binario está compuesto por una sucesión de áreas de memoria a cargar.

Cada área de memoria está constituida por dos secciones, un vector de carga y una secuencia de bytes que se cargarán en la memoria.

El vector de carga contiene toda la información necesaria para indicarle al computador en dónde tiene que almacenar la secuencia de bytes que vienen a continuación y cuánta información viene en el área de memoria a cargar.

El vector de carga está compuesto por 6 ó 4 bytes. Los dos primeros son $FF, que son obligatorios en el primer área de memoria del archivo y no en los siguientes. Estos dos bytes le indican al D.O.S. que el archivo es binario, pero en los

Files Binarios

BASIC

siguientes stages o zonas a cargar no son necesarios.

Luego de estos dos bytes opcionales siguen dos que indican la zona de memoria en donde va a comenzar a cargar el computador y dos bytes más que le indican hasta dónde cargar. Por ejemplo:

```
FF FF 00 06 02 06
```

Está indicando que el archivo es binario por los dos bytes opcionales y el primer bloque se cargará desde $0600 a $0602. Esto es así, pues los bytes que indican el Load Adress o zona de carga y el End Adress o final de carga están expresados con el sistema de LO y HI bytes.

En este ejemplo de vector de carga se estarían cargando 3 bytes en la memoria. Si después del vector de carga vienen los bytes 03 06 y 01, luego de realizar la carga de este bloque las direcciones quedarían con el siguiente contenido:

```
$600 --> 03
$601 --> 06
$602 --> 01
```

Cuando un stage es cargado por completo, quiere decir que los bytes comprendidos entre el Load Adress y el End Adress han sido leídos y almacenados en su posición de memoria, las direcciones $2E2 y $2E3 son examinadas.

Si ambas contienen ceros, se vuelve a cargar otro stage si es que existe. Si no son cero, se ejecuta un llamado a la subrutina apuntada por estos dos bytes y luego se devuelven estas dos posiciones en cero y se carga otro stage.

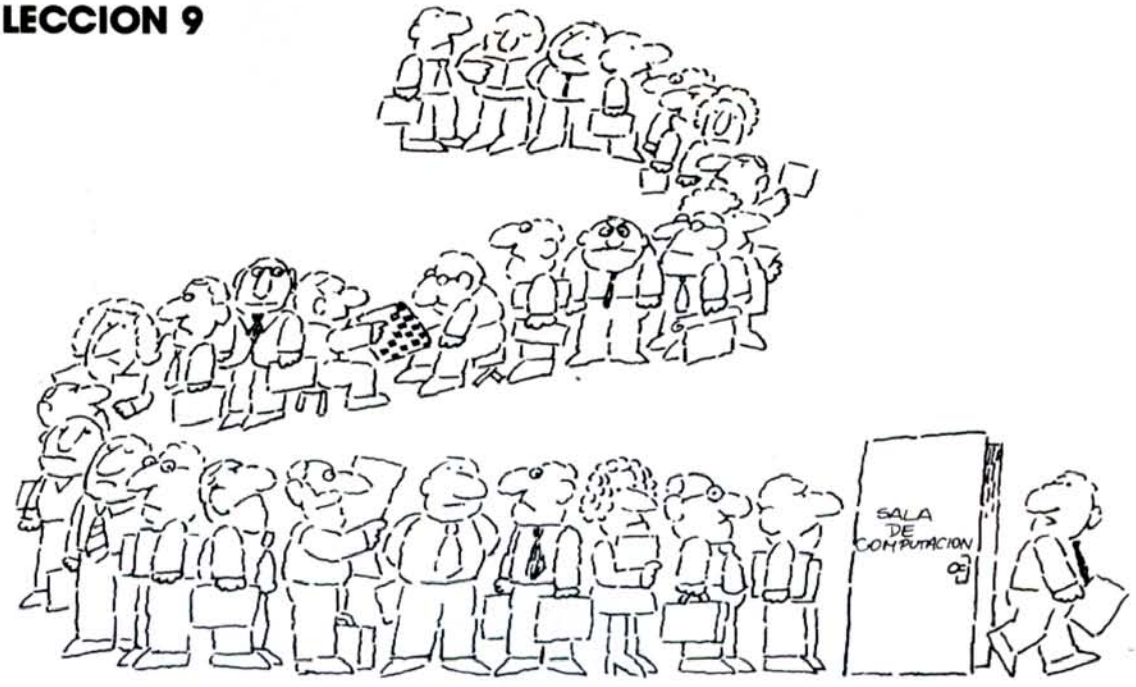
Esta estructura de análisis de las posiciones $2E2 y $2E3, permite en el medio de una carga que se ejecuten partes del programa como presentaciones o sonidos que se mantienen durante el resto de la carga del programa.

Cuando se trata de cargar otro stage del programa y la diskettera nos informa que se ha alcanzado el final del programa (EOF), la carga del programa finaliza y viene el proceso de ejecución del programa.

En este momento, el computador analiza las posiciones de memoria $2E0 y $2E1. Si ambas son cero, termina la carga y no se ejecuta el programa, pero si no lo son estos valores indican el comienzo del programa y el computador realiza un llamado a la zona de memoria apuntada por estas dos posiciones.

Una vez analizada la estructura de Files Binarios, en la próxima edición de TURBO news, incorporaremos la carga de discos Boot y trabajaremos con el Omnimon para describir cómo se pasan los archivos de una estructura a la otra.

LECCION 9

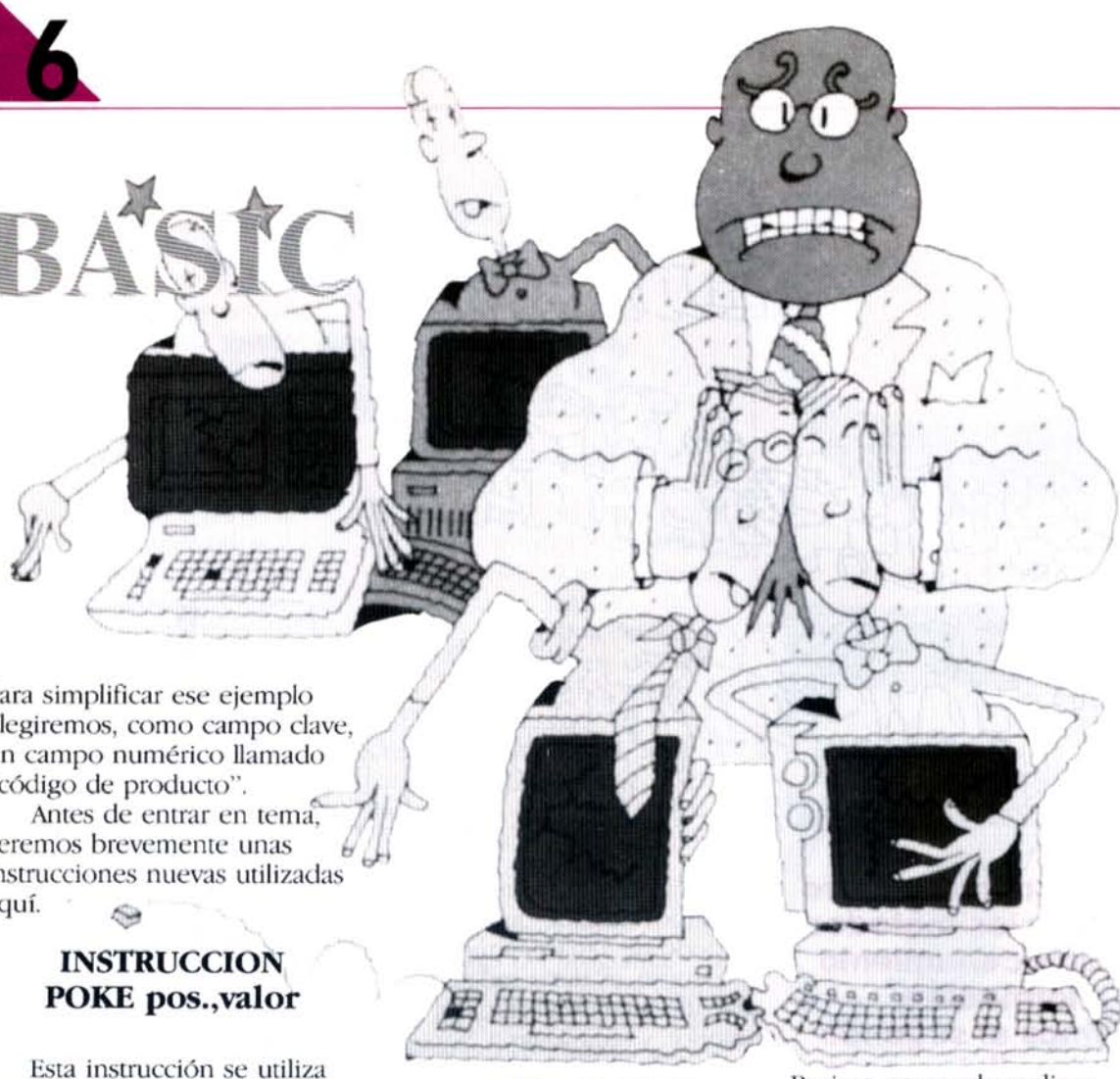


En este número, para concluir los métodos de almacenamiento de información, veremos el segundo de ellos: "el acceso directo". También veremos otro pequeño grupo de nuevas instrucciones.

En el número anterior habíamos hablado que existía el modo de almacenamiento secuencial, es decir cada dato grabado a continuación del anterior, y para llegar a cualquiera de ellos, se debían recorrer todos los anteriores, método utilizado tanto con la cassettera como con la diskettera. También contamos que existía un segundo método para utilizar, sólo con diskettera, que era el acceso directo. Este se basaba en que si queríamos llegar a un dato cualquiera, no debíamos pasar por los anteriores. Esto no se podía hacer con la cassettera, ya que no puede alcanzar cualquier dato en forma inmediata, sino que debe esperar que pase la cinta en forma secuencial. El disco, al estar en continuo movimiento de rotación y la cabeza poder avanzar o retroceder a cada instante, puede alcanzar cualquier dato rápidamente.

Para usar un archivo de acceso directo se debe elegir un campo que sea clave, es decir, que todo el archivo esté ordenado según ese criterio. Por ejemplo el nombre de la persona, la edad, un código de cliente, etc. Conviene que sea un número. Si no lo es, existen maneras de convertir cualquier texto en número, por ejemplo asignándole a cada letra un valor, para luego sumar todas las letras y obtener un resultado. En el ejemplo que daremos en la parte de programas veremos cómo hacer un programa que maneje un stock de hasta 2080 productos. Notarás, con el uso, que podrás preguntar por cualquiera de los 2080 productos siendo la respuesta casi inmediata, cosa que no sucede en el caso del acceso secuencial ya que para llegar al registro 2080 debería buscar por los 2079 anteriores.

BASIC



Para simplificar ese ejemplo elegiremos, como campo clave, un campo numérico llamado "código de producto".

Antes de entrar en tema, veremos brevemente unas instrucciones nuevas utilizadas aquí.

INSTRUCCION POKE pos.,valor

Esta instrucción se utiliza para poner un determinado valor en una posición de memoria. Para poder utilizar esta instrucción hay que poseer un mapa de memoria, como el que estamos entregando en la revista, que nos diga para qué sirve cada posición de memoria, y qué valores pueden ser depositados ahí. Un ejemplo de su utilización sería hacer:

```
POKE 710,1
```

La posición 710 maneja el color de fondo de la pantalla y si le ponemos el valor 1, logramos que ésta tome el color negro. Puedes probar de ponerle otros valores. Hay que tener en cuenta que el valor que se puede "pokear" varía de 0 a 255.

INSTRUCCION PEEK(pos.)

Esta instrucción se utiliza para "leer" el contenido de alguna posición de memoria. Puedes probarla haciendo:

```
PRINT PEEK(710)
```

También, para poder utilizarla, se necesita poseer un mapa de memoria.

INSTRUCCION USR

Esta instrucción se utiliza cuando queremos agregar una rutina en lenguaje de Máquina a un programa Basic. Esta instrucción la utilizan programadores avanzados cuando encuentran que el Basic o no puede realizar alguna cosa o no lo hace a la suficiente velocidad. A efectos de este curso, donde se supone no existen conocimientos previos del Assembler, la veremos como una "caja negra" la que le indicaremos, cuando la vayamos utilizando, para qué sirve.

INSTRUCCION ADR

Esta instrucción también es utilizada sólo por los programadores más avanzados y se utiliza para saber en qué dirección se encuentra almacenada alguna variable. Ejemplo:

```
PRINT ADR(NOMBRE$)
```

Ahora sí estamos en condiciones de retomar el tema del acceso directo. Para ello utilizaremos una rutina interna que posee el computador, llamada SIO. Para poder utilizarla, ésta necesita que se le entreguen varios datos. Estos son:

- Qué dispositivo va a utilizar (en este caso la diskettera 1)
- Qué tipo de operación va a realizar (lectura o escritura)
- De qué zona tiene que tomar o dónde debe dejar la información tomada del disco (en cuál variable)
- De cuál sector del disco debe tomar o dejar la información.

Cuando concluya cada operación, el computador entregará el resultado correspondiente.

El tipo de dispositivo que se va a utilizar se indica a través de la posición de memoria 768. Pokeándole allí un 49 se indica que se va a utilizar una diskettera.

Mediante la posición de memoria 769 se le indica cuál es el número de diskettera que se va a utilizar. En este caso 1.

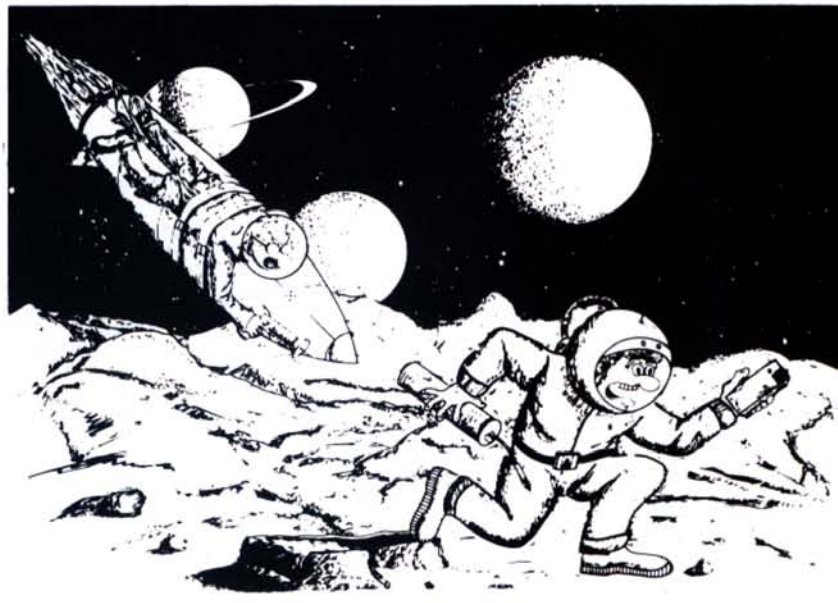
En la posición 770 debemos poner la operación que se va a realizar. Un 82 indica lectura y un 87 grabación.

En las posiciones 772 y 773 debemos poner la dirección de la variable que vamos a utilizar para tomar o dejar los datos. Esto se realiza mediante una fórmula matemática que podrás tomar del ejemplo.

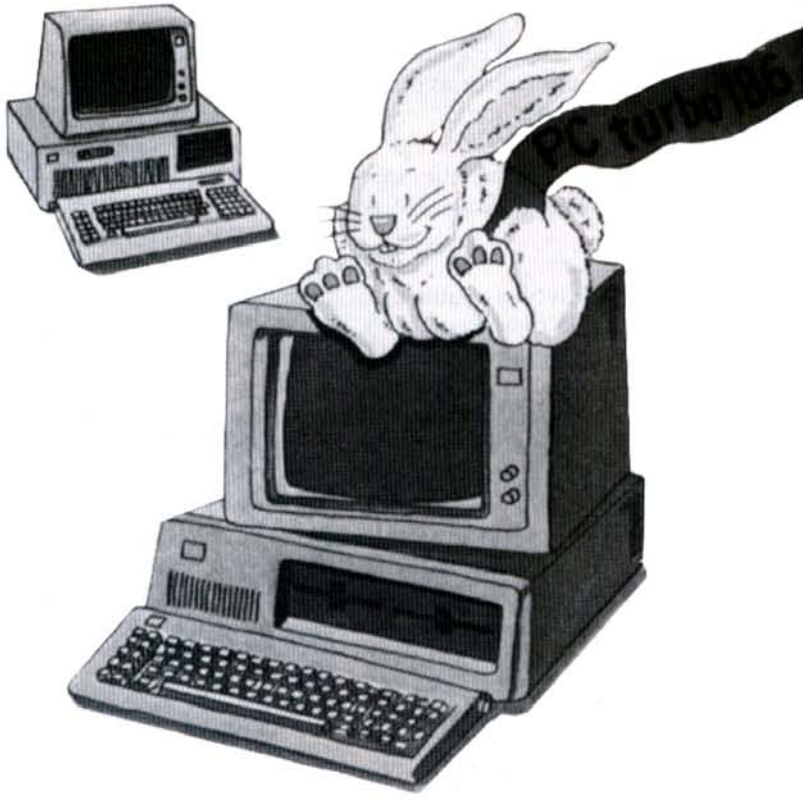
A través de las posiciones 778 y 779, debemos indicarle qué sector del disco queremos leer, aplicando la misma fórmula matemática en el caso anterior.

Finalmente leyendo con la instrucción PEEK el contenido de la posición 771, sabremos si se produjo un error o no durante la operación de lectura o grabación.

Veamos cómo se realiza todo esto en programación, si quisiéramos leer o grabar el sector 50.



BASIC




```
10 DCB=768
20 DIM BUFFER$(128),PRG$(6)
30 POKE DCB,49:POKE DCB+1,1
35 REM TOMAMOS LA
DIRECCION DE LA VARIABLE Y
LA GUARDAMOS EN SU LUGAR
40 ADDR=ADR(BUFFER$)
50 ADDRHI=INT(ADDR/256)
60 ADDRLO=ADDR-
(ADDRHI*256)
70 POKE DCB+4,ADDRLO
80 POKE DCB+5,ADDRHI
90 SECTOR=50
95 REM TOMAMOS EL NUMERO
DE SECTOR Y LO GUARDAMOS
EN SU LUGAR
100 SECTORHI=INT(SECTOR/
256)
120 SECTORLO=SECTOR-
(SECTORHI*256)
130 POKE DCB+10,SECTORLO
140 POKE DCB+11,SECTORHI
150 POKE DCB+2,82:REM PARA
LEER
160 POKE DCB+2,87:REM PARA
GRABAR
170 FOR I=1 TO 5:READ
A:PROG$(I,I)=CHR$(A):NEXT I
180 X=USR(ADR(PRG$))
190 DATA 104,32,83,228,96
```

Atari divide al disco en 1040 sectores, pudiendo almacenar cada uno de ellos 128 caracteres o bytes. Para mayor información de la estructura del disco, podrá consultar nuestros números anteriores.

Lo que hemos logrado ahora, es que dado un número de sector podamos tomarlo rápidamente y guardarlo en una variable de longitud 128. Allí podemos poner la información que deseemos, nombres, teléfonos, etc. Si hacemos por ejemplo que el número de sector coincida con el campo clave del archivo, como ser código de cliente, buscar el cliente número 5, puede convertirse en buscar el sector número 5, pudiendo por consiguiente tener hasta 1040 clientes. Podemos hacer aún más: podemos poner dos clientes por cada sector, entonces en el sector 1 estarán el cliente 1 y el 2. Si queremos buscar el cliente 19, estará en la primera mitad del sector 10 y el 20 en la segunda mitad. Esto es lo que hemos hecho en el programa que se encuentra al final de la revista. Dicho programa maneja hasta 2080 productos con acceso directo. También se ha puesto incapié en dicho programa, de cómo manejar de manera más agradable a la vista las pantallas. Un mismo programa, realizado con una presentación bonita, aunque haga menos cosas, es mucho mejor cotizado.

Para lograr esa presentación se ha utilizado una nueva instrucción:

INSTRUCCION POSITION X,Y

Esta instrucción se ha creado para ser utilizada precediendo a la instrucción PRINT o INPUT y le indica a éstas, en qué posición de la pantalla debe imprimir o tomar algún dato. Por ejemplo:

```
POSITION 4,5:PRINT "HOLA"
```

Imprimirá en la cuarta columna y quinta fila, la palabra hola.

¡¡Nos vemos en el próximo número!!

PROMOCION turbo

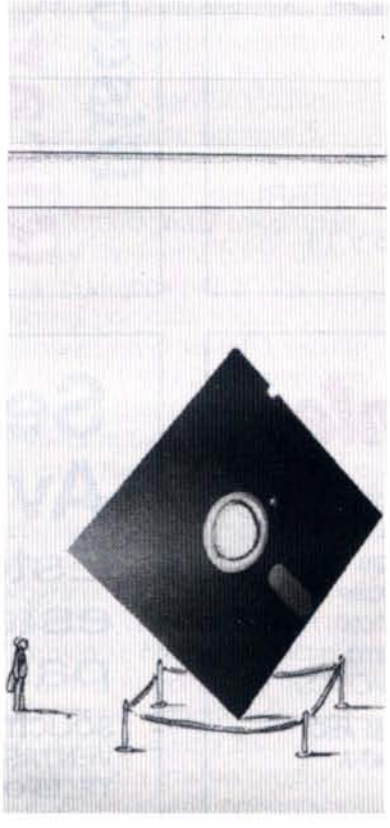
La protección del SOFTWARE

LECCION 1

A partir de este número y en los sucesivos, les iremos entregando técnicas para proteger el Software que vayamos realizando.

INTRODUCCION

Muchas veces, cuando luego de mucho esfuerzo, hemos logrado hacer un programa que deseamos comercializar, la primera pregunta que nos hacemos es cómo evitar que éste sea reproducido sin nuestro consentimiento y beneficio. Esto nos lleva a tener que proteger nuestros programas. La falta de protección lleva a que los programadores desistan del esfuerzo de resolver un problema cualquiera, debido que por cada copia que él pueda llegar a vender, se venden un número mayor de programas no originales. La protección, al mismo tiempo que soluciona uno de los problemas que es la copia pirata, crea uno nuevo, que es el hecho de que el comprador de un ejemplar legítimo, desee tener BACK-UP o copia de él, por las dudas que se le heche a perder. Por ello en todas las protecciones habría que tener en cuenta la posibilidad de que el dueño desee tener respaldo de sus programas.



Todos los programas son copiables de una u otra manera, si no fuera así el dueño del programa no podría reproducirlo para poder venderlo. El tema consiste en hacer la copia lo más difícil posible para así desalentar a la mayoría.

También es cierto que un programa que se copia fácilmente, también se difunde fácilmente con la consiguiente publicidad que eso ocasiona. Muchos excelentes programas, sumamente protegidos, no han logrado mayor difusión justamente por ello.

Comenzaremos a ver cómo proteger los programas escritos en Basic.

PROTECCION DE PROGRAMAS BASIC

El principal problema en la protección de programas escritos en lenguaje BASIC, es la posibilidad de detener los programas, ya que al hacerlo, cualquier persona desde el Basic puede ejecutar la instrucción SAVE o CSAVE y guardarlo en un disco o

cassette propio. Las maneras que existen de detener un programa son:

- Mediante la tecla BREAK.
- Mediante la tecla RESET.
- Forzando un error.
- Cargando el programa con LOAD o CLOAD y luego grabarlo con SAVE o CSAVE.

TECLA BREAK

Oprimiendo la tecla BREAK se pueden detener los programas sin que se borre el contenido de la memoria. Para evitar esto, existen dos métodos.

El primero es modificar el vector de interrupciones. Cuando se oprime la tecla BREAK, se produce una interrupción. El computador posee un vector donde tiene guardada la dirección de la rutina que atiende dicha interrupción. Lo que hacemos es modificar el vector para que en lugar de apuntar a la rutina de atención de la interrupción, apunte a una zona de memoria donde sabemos que no hace nada y simplemente le devuelva el control al programa. Esto se logra poniendo las siguientes instrucciones al principio del programa:

```
POKE 566,PEEK(518)
POKE 567,PEEK(519)
```

La segunda manera es modificar el vector de interrupción del POKEY para que no atienda esta interrupción. Lo hacemos con las siguientes instrucciones.

```
POKE 16,112
POKE 53774,112
```

Cabe destacar que la instrucción GRAPHICS, vuelve estas últimas dos posiciones a sus valores originales, por lo que se debería repetir estas instrucciones cada vez que se use.

La protección del SOFTWARE

El proteger la tecla BREAK también puede ser útil para evitar errores involuntarios durante la utilización de un programa BASIC.

TECLA RESET

El oprimir esta tecla durante la ejecución de un programa, también produce su detención, sin que se borre el contenido de la memoria.

Para evitarlo, también existen dos formas.

Poniendo al principio del programa:

```
POKE 580,1
```

Cuando se oprima la tecla RESET el computador se BOOTEARA de nuevo, es decir producirá el mismo efecto que haber apagado y prendido el computador de nuevo.

Poniendo:

```
POKE 9,255
```

Causará que se bloqueen las teclas del computador, cuando la tecla RESET sea oprimida.

FORZANDO UN ERROR

Cuando en un programa se produce un error, éste se detiene, como por ejemplo sucede si ingresamos un valor alfanumérico cuando se nos solicita uno numérico.

Para evitar este tipo de "trampas" se debe procurar que el programa pueda evitar que se produzcan errores. Eso se logra con la utilización de la instrucción Basic TRAP.

COMBINACIONES DE INSTRUCCIONES LOAD-SAVE, CLOAD Y CSAVE.

Para evitar este tipo de copia, debemos agregar al final de nuestro programa la siguiente instrucción:

```
32120 POKE PEEK(138)+256*PEE
K(139)+2,0:S AVE "D:nombre prog
rama":NEW
```

Cuando guardemos el programa, en lugar de hacerlo con una instrucción SAVE debemos hacerlo con la instrucción GOTO 32120. Conserva una copia guardada con SAVE, para tener así un ejemplar listable. En el caso de tener cassettera, deberás reemplazar el SAVE "D: por SAVE "C:. Una vez que el programa esté así protegido, cuando se encuentre en memoria, la ejecución de cualquier comando en modo inmediato, hará que el computador se bloquee. Para ejecutar un programa que tenga esta protección, se deberá hacer RUN "D:nombre" o RUN "C:, pero no se podrá hacer LOAD "D:NOMBRE" o LOAD "C:.

Si lo que queremos hacer es proteger un programa no contra copia, sino para que no se pueda modificar, existe una manera para encriptar los programas de la siguiente manera:

```
FOR N19=PEEK(130)+PEEK(131)*
256 TO PEEK(132)+PEEK(133)*25
6:POKE N19,155:NEXT N19
```

Lo que hacemos aquí es reemplazar todos los nombres de las variables por el caracter 155 (RETURN), quedando los listados totalmente distorsionados. Puedes intentar cambiar el 155 por otros caracteres produciéndose distintos efectos.

Mapa de memoria

En esta edición de tu revista TURBO news continuamos desarrollando el análisis de las distintas posiciones de memoria afectadas al trabajo del Sistema Operativo y a los Chips del Hardware de tu Atari.

En la Edición de Diciembre desarrollamos conceptos muy importantes dentro del mapa de memoria del computador que manejan el tema de las interrupciones. No hicimos ninguna mención práctica, pues está en nuestros planes incluirlo en unas cuantas lecciones del curso de Assembler cuando éste avance en su nivel.

En esta edición trabajaremos la zona de memoria comprendida entre $230 y $2BE.

Siempre al tratar el mapa de memoria dejamos sin explicación muchas de las posiciones de memoria utilizadas por el Sistema

Mapa de memoria

Operativo, pues éstas no revisten demasiada importancia para el programador, pues pueden operar como variables temporales dentro de rutinas poco trascendentes.

560,561 $230,$231 SDLSTL: Estas dos posiciones de memoria, definen el comienzo de la dirección del Display List.

564 $234 LPENH: En esta posición de memoria, el Sistema Operativo del Atari almacena la posición horizontal de la lectura del Lápiz de Luz.

565 $235 LPENV: Posición vertical del Lápiz de Luz sobre la pantalla.

580 $244 COLDST: Esta dirección del mapa de memoria contiene un flag que define la acción que va a tomar el computador en caso de ser presionada la tecla Reset. Si esta posición contiene un cero, al presionar Reset no se genera un reboot del sistema. Si en cambio su contenido es 1, al presionar Reset el equipo realiza las mismas tareas que hace cuando se enciende el computador.

624 $270 PADDL0: En esta posición de memoria, el sistema operativo almacena el valor de la lectura del paddle Nro. 0. Este valor puede oscilar entre 0 y 228.

625 $271 PADDL1: valor del paddle 1.

626 $272 PADDL2: valor del paddle 2.

632 $278 STICK0: Esta posición registra la posición en la que se encuentra el bastón del joystick 1.

633 $279 STICK1: Esta posición similar a la anterior, registra el contenido de la lectura del joystick 2.

694 $2B6 INVFLG: Esta posición de memoria refleja a un flag que indica si el teclado está o no en video inverso. Un valor cero en esta posición indica que el teclado está en video común, pero si pokeamos un 128 el teclado se convierte en video inverso.

702 $2BE SHFLOK: Esta posición contiene un flag que indica si el teclado está trabajando en mayúsculas o minúsculas o bien si está siendo ocupada la tecla Control. Pokeando un cero, el teclado trabaja con minúsculas, con un 64 lo hace con mayúsculas y cuando posee un 128 la tecla Control está siendo accionada.

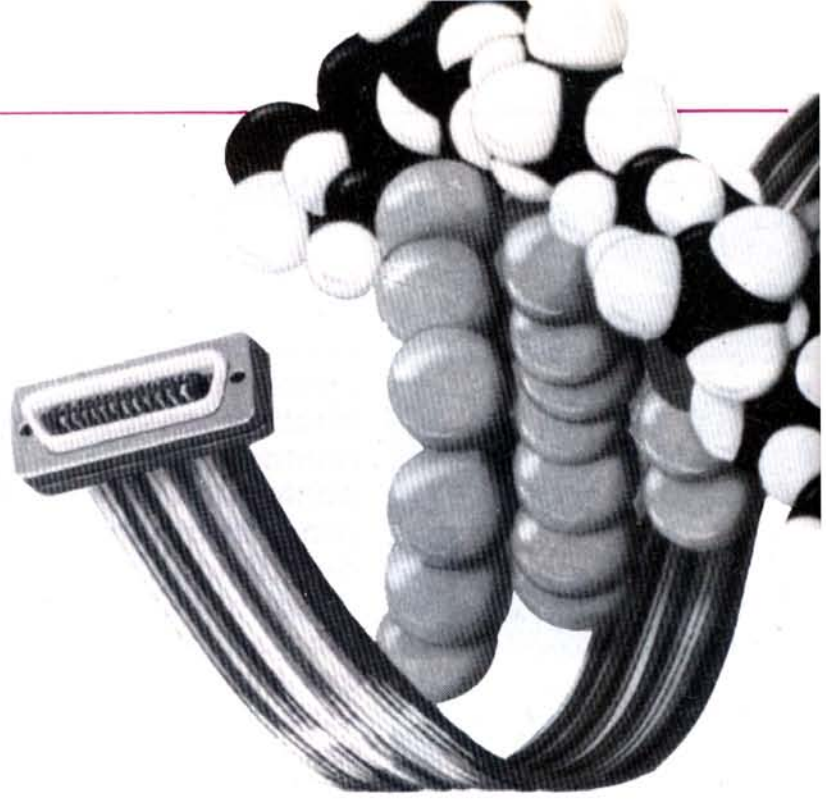
Solución Puzzle (Edición N° 8)

ASSEMBLER ASSEMBLER

LECCION 9

En las lecciones anteriores de este curso hemos comenzado a analizar las distintas rutinas que conforman el programa que decidimos desarrollar y en esta lección continuaremos con esta apasionante tarea.

ASSEMBLER



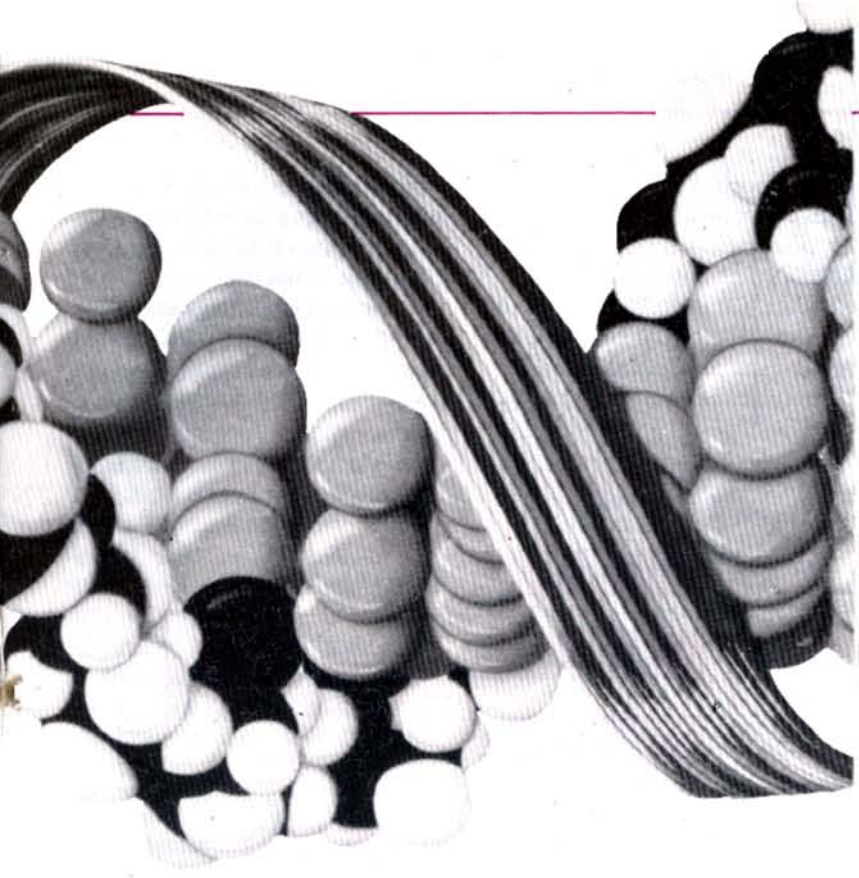
Tomaremos como patrón, para seguir analizando el programa, el diagrama de flujo que desarrollamos en la Edición Nro. 7 en su página 12.

Ocuparemos cada uno de los símbolos del diagrama para estudiarlos en la práctica en el listado del programa. Comenzaremos con el primer rectángulo del programa que define a Flagnum en cero.

El programa debe saber, en cada instante, si debe o no numerar automáticamente cada vez que el usuario termina de agregar una línea en el programa, para esto definimos la variable Flagnum que puede tener dos valores: cero o distinto de cero. Si el valor que posee es un cero, el programa no numera automáticamente a menos que el usuario ingrese el comando de numeración automática "Numerar". Si se ingresa el comando de numeración automática, la variable Flagnum toma un valor distinto de cero.

Cuando se inicia el programa luego de definir algunos parámetros importantes en el Rótulo INICIO, en la línea 1330 del programa EDITOR.MAC, encontramos las instrucciones que definen a Flagnum en cero. Luego antes de que el computador solicite al usuario otra línea, consulta si Flagnum está o no en cero. Esto lo realiza en las líneas siguientes a la 1570. Las líneas LDA FLAGNUM y BNE NUMERACION, vienen a representar al primer rombo del diagrama de flujo, en el cual se decide si se imprime o no en la pantalla un número de instrucción.

El segundo rectángulo del diagrama de flujo del programa se utiliza para definir a un puntero que va a recorrer



durante todo el programa a la tabla de instrucciones. Esta tabla se va a ubicar dentro de la memoria del sistema, a continuación del listado del programa. Para esto, en la línea 6900 del programa EDITOR.MAC, definimos el rótulo Tabla. En las líneas 1460 a 1490, definimos a FINTAB que va a apuntar a la primera posición de memoria disponible para almacenar alguna instrucción en la tabla. Para realizarlo ingresamos las líneas:

```
LDA #<TABLA
STA FINTAB
LDA #>TABLA
STA FINTAB+1
```

Esta variable FINTAB es modificada cada vez que se agrega algún byte en la tabla de instrucciones para que siempre mantenga la primera posición libre dentro de la tabla. Otra de las aplicaciones de esta variable, es tener en cuenta que si estamos recorriendo toda la tabla con algún fin como el de listar todo el programa, si nos excedemos de la posición apuntada por FINTAB, estamos invadiendo zonas de memoria que no contienen el listado del lenguaje.

Luego de comparar a Flagnum con cero y determinando que este posee el valor que indica que no existe numeración automática, el programa ejecuta un JMP SINUMERACION para que el usuario ingrese una línea de programa o un comando a su elección. Esta tarea está representada por el siguiente rectángulo en el diagrama de flujo del programa, ubicado a la izquierda del rombo que interroga sobre el valor de Flagnum.

En el rótulo SINUMERACION, que se encuentra a partir de la línea 2670, se ejecuta un llamado a la subrutina TOMOLINEA que se queda esperando hasta que el usuario presione la tecla Return luego de ingresar una línea. Esta subrutina Tomolínea, afecta a la zona de memoria denominada por el programa como LINEA, definida en $9000 en la instrucción 190 de EDITOR.MAC. Cuando el usuario termina de ingresar la instrucción o el comando, el contenido de las teclas presionadas se localiza en la memoria a partir de esta posición denominada Línea. Por eso en las siguientes instrucciones se consulta si la primera posición de la zona Línea es un $9B que representa un Return.

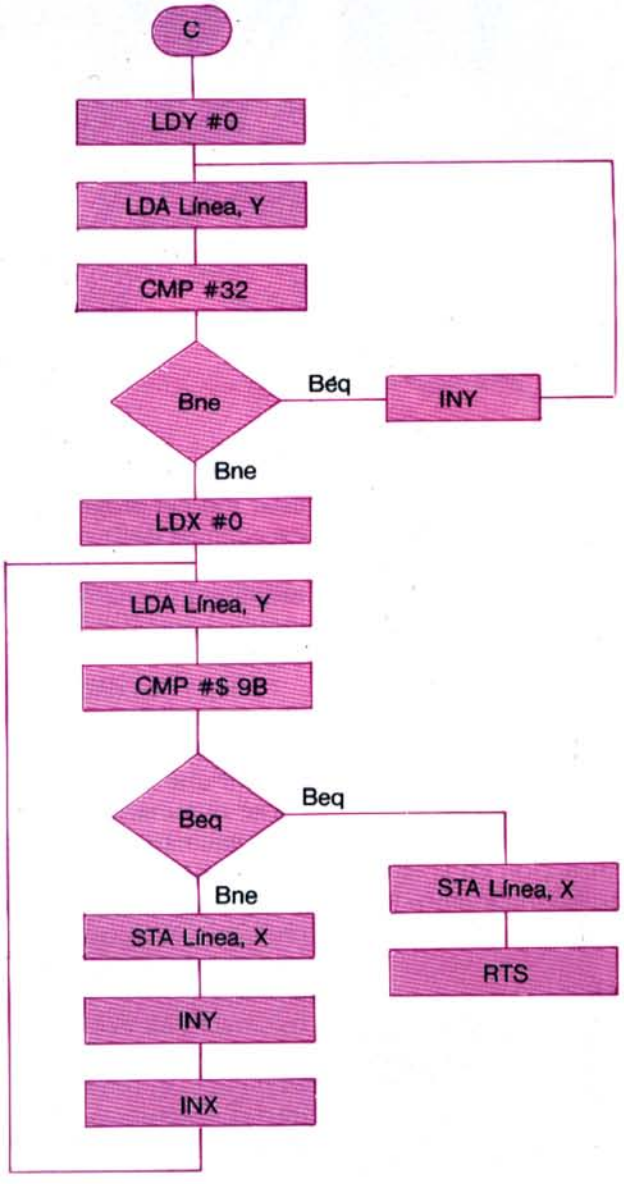
Si el primer caracter de línea es un Return, significa que el usuario sólo presionó esta tecla y no existe ningún

comando o instrucción en la línea. Esto si bien no se ve razonable es muy útil para anular la numeración automática del programa. Quiere decir que si el sistema está numerando automáticamente, el usuario puede abortar la numeración ingresando una línea nula.

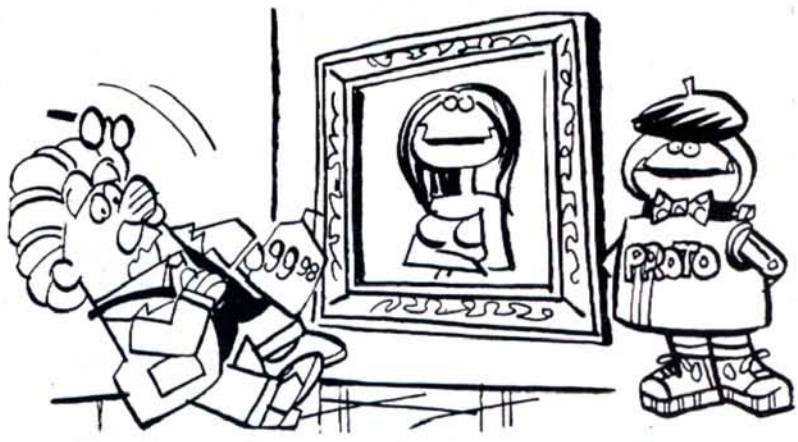
Para lograr este efecto, en la línea 2770 definimos nuevamente a Flagnum en cero y luego de imprimir el mensaje LIST en la pantalla el programa vuelve a esperar otra línea. Esta operatoria viene a representar al siguiente rombo del diagrama de flujo, en donde se cuestiona si la línea ingresada fue un Return y si lo fue se ejecuta un Flagnum=0.

Si la línea ingresada por el usuario no fue un Return debemos eliminar los posibles espacios en blanco a la izquierda de la instrucción. Para esto en la línea 2860 llamamos a la subrutina BLANCIZ.

Esta rutina recorre la zona de memoria LINEA incrementando al registro Y por cada espacio en blanco que la línea posea antes de algún caracter. Cuando encuentra un caracter distinto al byte #32 que representa al espacio en blanco, realiza un movimiento a izquierda de toda la línea superponiéndose ésta en todos los espacios en blanco. Para ver clara esta situación, incorporamos el diagrama de flujo de esta rutina para que puedas entender con mayor precisión su funcionamiento:



Esta operación, viene a representar al rectángulo que se ejecuta en el diagrama de flujo del sistema si la línea ingresada no fue un Return. Luego se verifica en el diagrama si la línea ingresada en realidad es un comando o bien una instrucción. Para esto lo que se verifica es si el primer caracter de Línea está comprendido entre los caracteres 0 y : Esto es así, pues en la tabla de caracteres ATASCII, después del número 9 viene el : Entonces si el primer byte de la línea ingresada está comprendido entre el rango 0Y:, la línea contiene una instrucción para el cartel.



Este trabajo en el listado del programa se realiza en las líneas comprendidas entre la 2930 y la 2970. Es importante mencionar en esta parte del curso que cuando queremos comparar si algún byte es menor a otro ejecutamos la instrucción BCC y si la consulta es por mayor o igual es un BCS la que debemos encarar.

Luego de determinar si la línea fue un comando saltamos al rótulo COMANDO, ubicado en la instrucción 3390 del listado. En ella lo que estamos analizando es si el comando fue válido o no y de serlo lo ejecutamos y volvemos a tomar una nueva línea del usuario. Si en cambio la línea fue una instrucción, ejecutamos un llamado a la rutina NUMEROLINEA, que transforma el número de línea en dos bytes HI y LO byte, para que ésta ocupe menos espacio en la tabla del sistema.

Luego de hacerlo, comprimimos de la línea al número de instrucción y saltamos a la rutina que almacena en forma ordenada según el número de instrucción la línea dentro de la tabla.

En las siguientes lecciones del curso seguiremos analizando este entretenido sistema y describiremos las rutinas que mencionamos en esta lección y no las hemos tratado con detalle para no complicarnos tanto por ahora.

No te olvides que en la sección de programas seguimos listando todos los programas que componen este sistema. Ingrésalos en tu computador y grábalos en tu medio de almacenamiento con 1 ó 2 copias para que no pierdas tu trabajo.

Hasta la próxima lección...

Gráficos por computadora

CUARTA PARTE

Ahora ya estamos en condiciones de realizar los primeros programas utilizando PLAYERS-MISSILE. En este número encontrarás varios ejemplos para clarificar todas las dudas.

Ya hemos visto cuáles eran las características que daban vida a cada uno de los Players-Missile. En este número veremos cuáles son los pasos necesarios para crearlos.

Lo primero que debe hacerse es seleccionar en qué parte de la memoria vamos a ubicar los dibujos de los Players-Missile. Para ello conviene utilizar la zona de RAM que se encuentra al final del Basic, antes de donde empieza la utilizada por el ROM del ATARI. Existe una posición de memoria que el computador utiliza para saber cuántas páginas de 256 bytes cada una, posee el computador disponible para el Basic. Dicha posición es la 106 y normalmente posee el valor 160 lo que significa que hay 40960 bytes disponibles. De ellos, una parte lo ocupa la pantalla (que va a variar dependiendo del modo gráfico con el cual se trabaje). Para los ejemplos vamos a utilizar el modo gráfico 3 que ocupa 240 bytes. Es fundamental la elección del modo gráfico con el cual trabajar. Antic interrumpe la CPU 6502 y se encarga del control de buses para su acceso directo a memoria una vez por cada byte de memoria de pantalla. Durante ese tiempo la CPU se encuentra en suspenso. Cuanta mayor cantidad de memoria ocupe la pantalla, más veces ANTIC interrumpirá a la CPU y no le dejará trabajar en nuestro programa BASIC. Todo ese proceso de actualización, se realiza 60 veces por segundo. Por ello, cuando no se necesita, conviene utilizar modos gráficos que consuman menos memoria. También al trabajar con PLAYERS-MISSILE, aumenta el trabajo del ANTIC, robándole tiempo a la CPU. Por ello es aconsejable que si no se van a

usar misiles o cuando se deja de trabajar con Players, sean desconectados como veremos más adelante.

Como habíamos dicho debemos primero seleccionar la zona de memoria que debemos utilizar. Como hemos seleccionado el modo gráfico 3 y los Players en resolución doble ocupan 8 páginas, reservamos 16 páginas de memoria. Debemos recordar que la zona elegida debe ser divisible por 8. O sea que si tomamos el PEEK de 106, le restamos 16 páginas y multiplicamos al resultado por 256 tendremos la dirección (PMBASE), donde deben comenzar a dibujarse los Players, según el esquema de utilización de memoria dada en números anteriores.

Ese valor de página donde deben almacenarse los Players deben pokearse en la posición 54279. A continuación debe indicársele a ANTIC que debe empezar a buscar información de PLAYERS en la memoria. Se hace a través de la posición 559. Dicha posición está compuesta de la siguiente manera:

NUMERO BIT	7	6	5	4	3	2	1	0
VALOR DEL BIT	128	64	32	16	8	4	2	1
HABILITACION PANTALLA			1					
RESOLUCION PM				1				
HABILITACION PLAYER					1			
HABILITACION MISSILE						1		

BITS	7	6	5	4	3	2	1	0
VAL	0	0	1	0	1	0	1	0
			32	+	8	+	2	= 42

El resto de los bits veremos en próximos números para qué se utilizan. Para saber qué valor pokear en esta posición de memoria debemos sumar los valores de los bits de aquellos que deseamos prender. El bit 5 se utiliza para indicarle al ANTIC que debe interrumpir o no a la CPU. Normalmente está puesto en 1, pero si el computador está realizando complejos cálculos en un programa cualquiera y no nos interesa ver nada en la pantalla durante ese proceso, podemos apagarla poniendo en 0 este bit, aumentando la velocidad de procesamiento un 20%.

El bit 4 maneja la altura del Player, es decir si cada bit que dibujamos ocupa una o dos líneas de alto. Un uno aquí significa gráficos de 1 línea (mayor resolución) y un cero, 2 líneas.

El bit 3 determina si el player está habilitado o no. Lo mismo hace el 2 con los missiles. Normalmente en un programa cualquiera están en cero. Si queremos utilizarlos debemos ponerlos en uno.

Los bits 1 y 0 debemos decir únicamente que el 1 normalmente está encendido y el 0 apagado.

Por ejemplo, si queremos poner la pantalla normal y trabajar con players de resolución de dos líneas, debemos hacer la siguiente cuenta:

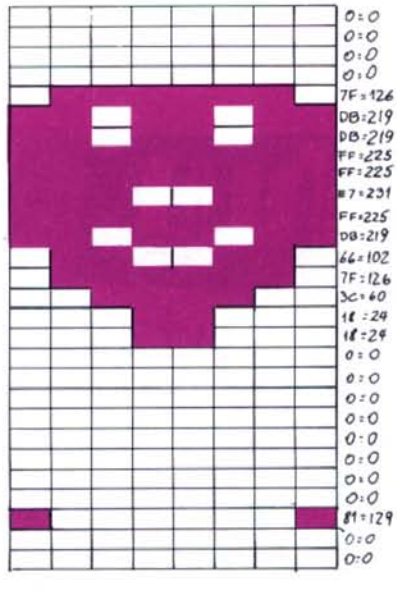
También debe indicársele a la GTIA, otro componente del computador, que se van a utilizar gráficos de PM. Esto se hace a través de la posición 53277.

Poniéndole un valor 2, le indicamos que vamos a utilizar PLAYERS y poniéndole un 3, PLAYERS y MISSILES.

En resumen, todos los gráficos que utilicen PM deben tener en un comienzo las siguientes instrucciones:

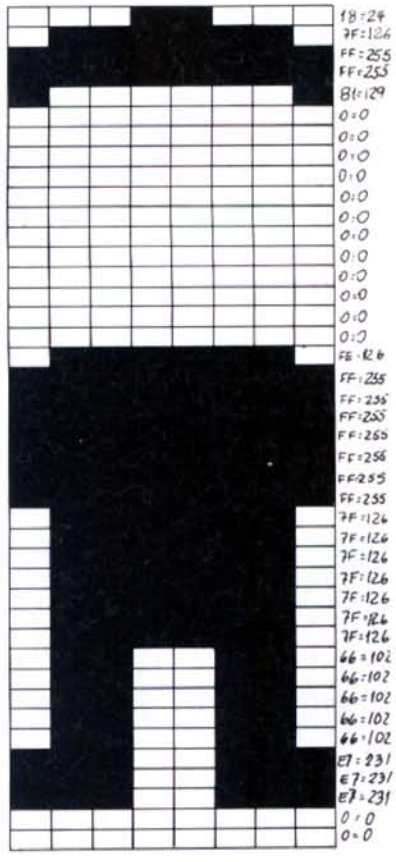
```
10 BASE=PEEK(106)-nro de
páginas a reservar
20 POKE 54279,BASE
30 PMBASE=BASE*256
40 POKE 559,valor seleccionado
50 POKE 53277,valor seleccionado
```

A continuación crearemos un PM que sea un hombre. Lo haremos con dos Players para poder hacerlo así de dos colores.



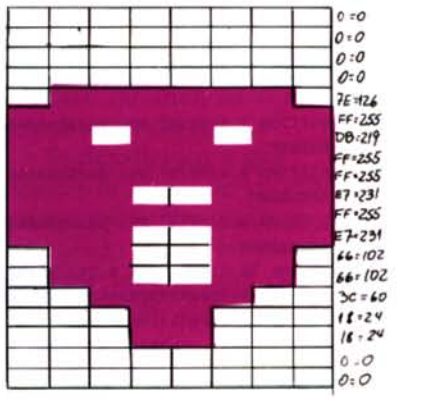
Recordemos que lo primero que hay que hacer, es crear las variables alfanuméricas para poder llegar así, a través de la tabla de contenido de variables, a saber la dirección donde se encuentran sus contenidos, como vimos en números anteriores.

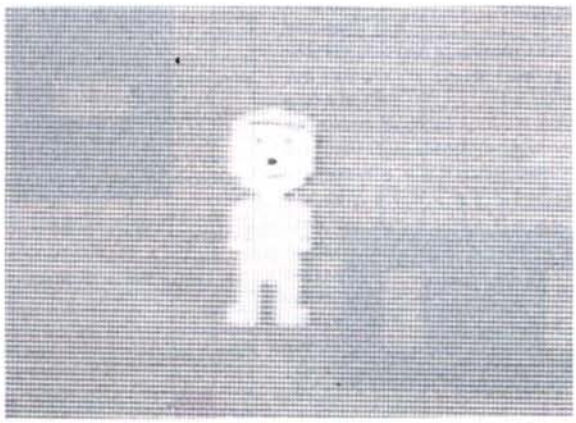
Los dibujos de las personas los creamos de la siguiente manera:



```
10 DIM CABE$(256),PELO$(256),DIBI$(42)
,DIB2$(42)
20 GRAPHICS 3+16:POKE 710,66
30 BASE=PEEK(106)-16
40 POKE 54279,BASE
50 PMBASE=BASE*256:FOR I=PMBASE+1024 T
O PMBASE+1536:POKE I,0:NEXT I:REM DEBE
MOS BORRAR TODA  LA ZONA DE PLAYERS
60 POKE 559,58: REM PANTALLA Y PLAYERS
 DE RESOLUCION 1 LINEA
70 POKE 53277,2: REM SOLO PLAYERS
80 POKE 704,15:POKE 705,7
85 REM MODIFICAMOS LA VVTP PARA QUE AP
UNTEN A LA ZONA DE PLAYERS
90 STARP=PEEK(140)+PEEK(141)*256
100 VVTP=PEEK(134)+PEEK(135)*256
110 I=PMBASE+1024-STARP
120 HB=INT(I/256)
130 POKE VVTP+2,I-HB*256
140 POKE VVTP+3,HB
150 I=PMBASE+1280-STARP
155 HB=INT(I/256)
160 POKE VVTP+10,I-HB*256
170 POKE VVTP+11,HB
175 PONEMOS LOS DIBUJOS EN LAS VARIABL
ES
180 FOR I=1 TO 40:READ A:DIBI$(I,I)=CH
R$(A):NEXT I
190 FOR I=1 TO 40:READ A:DIB2$(I,I)=CH
R$(A):NEXT I
195 REM PONEMOS LOS PLAYERS EN LAS POS
ICIONES 120,100
200 CABE$(100)=DIBI$
210 PELO$(100)=DIB2$
220 POKE 53248,120:POKE 53249,120
230 GOTO 230
1000 DATA 0,0,0,0,126,219,219,255,255,
231,255,219,102,126,60,24,24,0,0,0,0,0
,0,0,0,123,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1010 DATA 24,126,255,255,127,0,0,0,0,0
,0,0,0,0,0,0,126,255,255,255,255,255,25
5,255,255,126,126,126,126,126,126,126,12
6
1020 DATA 102,102,102,102,102,231,231,
231
```

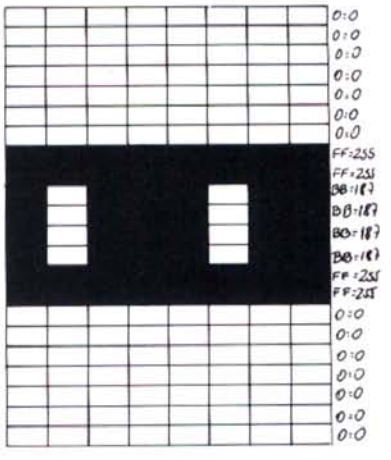
Tomando este mismo programa como base, le agregaremos el desplazamiento del Player en sentido horizontal, el manejo de prioridades y el cambio de dibujo, utilizado para darle más realismo al programa. Para ello creamos un tercer dibujo que va a reemplazar la cara de la persona cambiándole el gesto, como se ve en el dibujo más abajo.

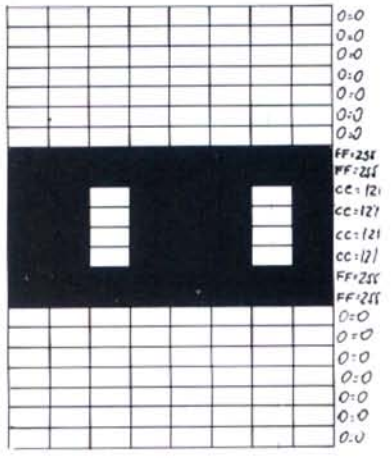


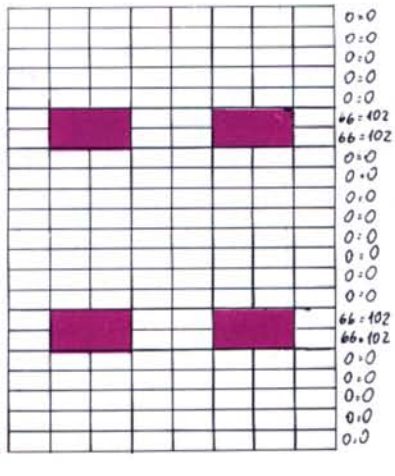


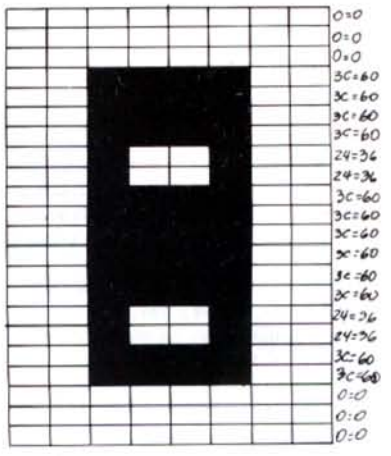
Gráficos por computadora

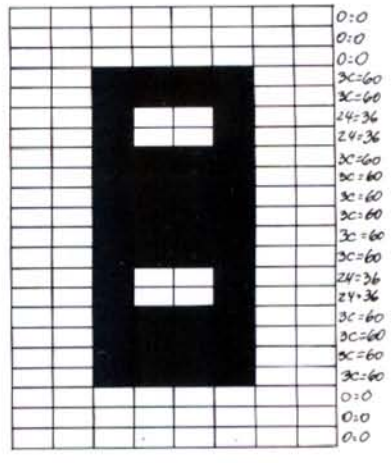
A continuación veremos con otro programa, cómo hacemos el movimiento del PLAYER hacia arriba y hacia abajo. También mostraremos aquí, el uso de colisiones. Los dibujos utilizados son los siguientes:

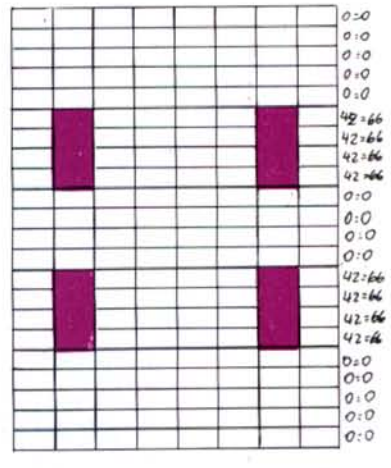






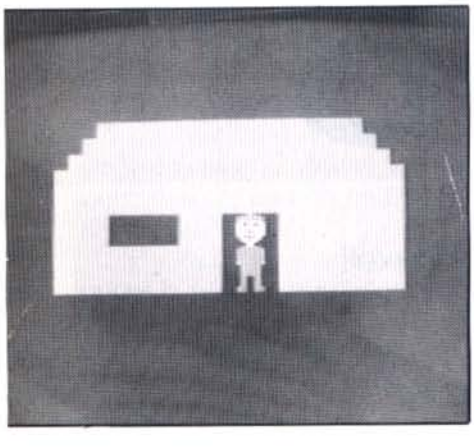






Se utilizan cuatro, uno por cada una de las direcciones que pueda tomar el auto (arriba, abajo, izquierda, derecha).

```
15 DIM DIBI$(28)
200 CABE$(113)=DIBI$:REM PONEMOS EL PL
AYER EN 120,113
210 PEL0$(113)=DIB2$
220 POKE 53248,120:POKE 53249,120
221 REM DIBUJAMOS LA CASA
223 POSITION 8,4:PRINT #6;"ÀÀÀÀÀÀÀÀÀÀ
ÀÀÀÀÀÀÀÀ"
225 POSITION 7,5:PRINT #6;"ÀÀÀÀÀÀÀÀÀÀ
ÀÀÀÀÀÀÀÀÀÀ"
230 POSITION 6,6:PRINT #6;"ÀÀÀÀÀÀÀÀÀÀ
ÀÀÀÀÀÀÀÀÀÀÀÀ"
240 POSITION 5,7:PRINT #6;"ÀÀÀÀÀÀÀÀÀÀ
ÀÀÀÀÀÀÀÀÀÀÀÀÀÀ"
250 FOR I=1 TO 2:POSITION 5,7+I:PRINT
#6;"BBBBBBBBBBBBBBBBBBBBBBBBBB":NEXT I
270 FOR I=1 TO 2:POSITION 5,9+I:PRINT
#6;"BBBB     BBB    BBBBBBBB":NEXT I
290 FOR I=1 TO 3:POSITION 5,11+I:PRINT
 #6;"BBBBBBBBBBBB    BBBBBBBB":NEXT I
300 FOR I=1 TO 17:READ A:DIBI$(I,I)=CH
R$(A):NEXT I
310 POKE 623,4:REM FIJAMOS LA PRIORIDA
D PARA QUE PASE POR ATRÁS DE LA CASA
320 CABE$(113)=DIBI$:FOR I=80 TO 130:P
OKE 53248,I:POKE 53249,I:FOR J=1 TO 10
:NEXT J:NEXT I
325 REM LE CAMBIAMOS EL GESTO CUANDO V
UELVE
330 CABE$(113)=DIBI$:FOR I=130 TO 80 S
TEP -1:POKE 53248,I:POKE 53249,I:NEXT
I
500 GOTO 320
1030 DATA 0,0,0,0,126,255,219,255,255,
231,255,231,102,102,60,24,24
```



```
10 DIM JEEP$(256),JEEP2$(256),AUTO$(22
),RUEDA$(22),RUEDA2$(22),BLANCO$(256)
20 DIM AUTO2$(22),AUTO3$(22),AUTO4$(22
)
22 REM LA VARIABLE BLANCO$ SE UTILIZA
PARA BORRAR LOS PLAYERS A VELOCIDAD DE
 LENGUAJE DE MAQUINA
25 FOR I=1 TO 256:BLANCO$(I,I)=CHR$(0)
:NEXT I
30 GRAPHICS 3+16
40 TEMP=PEEK(106)-16
50 POKE 54279,TEMP
60 PMBASE=256*TEMP
70 POKE 559,58:POKE 53277,2:POKE 704,1
5:POKE 705,7
80 POKE 623,1
90 STARP=PEEK(140)+PEEK(141)*256
100 VVTP=PEEK(134)+PEEK(135)*256
110 I=PMBASE+1024-STARP
120 HB=INT(I/256)
130 POKE VVTP+2,I-HB*256
140 POKE VVTP+3,HB
150 I=PMBASE+1280-STARP
160 HB=INT(I/256)
170 POKE VVTP+10,I-HB*256
180 POKE VVTP+11,HB
185 REM CARGAMOS LOS DIBUJOS
190 FOR I=1 TO 22:READ A:AUTO$(I,I)=CH
R$(A):NEXT I
200 FOR I=1 TO 22:READ A:AUTO2$(I,I)=C
HR$(A):NEXT I
210 FOR I=1 TO 22:READ A:AUTO3$(I,I)=C
HR$(A):NEXT I
220 FOR I=1 TO 22:READ A:AUTO4$(I,I)=C
HR$(A):NEXT I
230 FOR I=1 TO 22:READ A:RUEDA$(I,I)=C
HR$(A):NEXT I
240 FOR I=1 TO 22:READ A:RUEDA2$(I,I)=
CHR$(A):NEXT I
245 DIBUJAMOS LA PISTA
250 PRINT #6;"AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA"
260 FOR I=1 TO 22:? #6;"A
                        A":NEXT I
270 PRINT #6;"AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA"
280 POSITION 8,5:PRINT #6;"AAAAAAAAAA
AAAAAAAAAA"
285 FOR I=1 TO 12:POSITION 8,5+I:PRINT
 #6;"A                A":NEXT I
290 POSITION 8,18:PRINT #6;"AAAAAAAAAA
AAAAAAAAAA"
295 POSITION 31,12:PRINT #6;"AAAAAA"
298 REM LOOP PRINCIPAL
300 HPOS=90:VPOS=198:JEEP$(VPOS)=AUTO$
:JEEP2$(VPOS)=RUEDA$
310 JOY=STICK(0):POKE 53278,0
320 IF JOY=7 THEN HPOS=HPOS+3-(HPOS)20
0)*3:JEEP$(VPOS)=AUTO$:JEEP2$(VPOS)=RU
EDA$:GOTO 400:REM DERECHA
330 IF JOY=11 THEN HPOS=HPOS-3+(HPOS<5
0)*3:JEEP$(VPOS)=AUTO2$:JEEP2$(VPOS)=R
UEDA$:GOTO 400:REM IZQUIERDA
340 IF JOY=14 THEN VPOS=VPOS-3+(VPOS<I
5)*3:JEEP$(VPOS)=AUTO3$:JEEP2$(VPOS)=R
UEDA2$:GOTO 400:REM ARRIBA
350 IF JOY=13 THEN VPOS=VPOS+3-(VPOS)2
38)*3:JEEP$(VPOS)=AUTO4$:JEEP2$(VPOS)=
RUEDA2$:GOTO 400:REM ABAJO
400 POKE 53248,HPOS:POKE 53249,HPOS
410 IF PEEK(53252)<>0 THEN 510:REM HUB
O CHOQUE?
420 GOTO 310
500 REM RUTINA DE CHOQUE
510 FOR I=1 TO 100:N=(100-I)/10:SOUND
0,I,6,N:NEXT I
520 JEEP$=BLANCO$
530 JEEP2$=BLANCO$
580 SOUND 0,0,0,0:GOTO 300
600 DATA 0,0,0,0,0,0,0,0,255,255,187,187
,187,187,255,255,0,0,0,0,0,0,0,0
610 DATA 0,0,0,0,0,0,0,0,255,255,221,221
,221,221,255,255,0,0,0,0,0,0,0,0
620 DATA 0,0,0,0,60,60,60,60,60,36,36,60,60
,60,60,60,60,36,36,60,60,0,0,0,0
630 DATA 0,0,0,0,60,60,36,36,60,60,60,60
,60,60,36,36,60,60,60,60,0,0,0,0
650 DATA 0,0,0,0,0,0,102,102,0,0,0,0,0,0
,0,0,102,102,0,0,0,0,0,0
660 DATA 0,0,0,0,0,0,66,66,66,66,0,0,0,0
,66,66,66,66,0,0,0,0,0,0
```

Para evitar que el auto se vaya de los límites de la pantalla, se ponen unas fórmulas matemáticas para asegurarse que el auto quede siempre en pantalla. Por ejemplo, si queremos que HPOS nunca sea mayor a 200, hacemos HPOS=HPOS+3-(HPOS>200)*3.

Si HPOS es menor a 200, no hay problema, la expresión (HPOS>200) vale 0 y HPOS se incrementa en tres. Cuando HPOS es mayor a 200, dicha expresión vale 1 multiplicado por 3 da 3, es decir, le suma tres pero también se lo resta, permaneciendo igual.

Te desafiamos a que mejores la rutina de choque, por ejemplo dibujando un auto destrozado. También puedes agregarle un contador de vueltas y fabricar más pistas.

Una

Cuando queramos detener cada uno de los programas anteriores, si lo hacemos con la tecla BREAK, al estar los PLAYERS conectados, quedará una franja muy molesta en pantalla (compruébalo). Por eso conviene detener el programa con la tecla RESET.

¡¡Nos vemos en el siguiente número!!

RANKING DEL MES

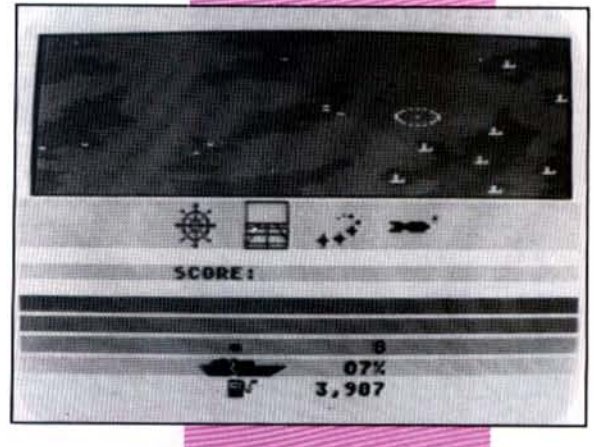
POS. DEL MES	POS. MES ANTERIOR	TITULO
1	4	HARDBALL
2	9	INTERNATIONAL KARATE
3	6	CRYSTAL RAIDER
4	11	HENRY'S HOUSE
5	5	GHOSTCHASER
6	-	LEGACY II
7	-	GREMLINS
8	3	NINJA
9	2	MONTEZUMA'S REVENGE
10	14	FIGHTER PILOT II
11	18	BOULDER DASH I
12	1	SUPER SOCCER
13	-	ZAXXON
14	-	BC'S QUEST FOR TIRES
15	13	GREAT AMERICAN RACE
16	7	LEADER BOARD GOLF
17	8	TABLE SOCCER
18	12	RAID OVER MOSCOW
19	22	SCREEMINGS WINGS 1942
20	-	GHOSTBUSTER
21	21	SWAT
22	15	RIVER RAID
23	-	BMX SIMULATOR
24	-	STAR RAIDERS II
25	-	SUPER PACMAN

ASCENSO **CONSTANTE** **DESCENSO**

Este es el Ranking correspondiente al mes de Abril, obtenido en base a las estadísticas de ventas de cassettes Turbo Software en todo Chile. Recuerda que tus preferencias también serán tenidas en cuenta, para lo cual podrás escribir a Av. Fco. Bilbao 4226, Las Condes, con los juegos de tu elección.

LEGACY II

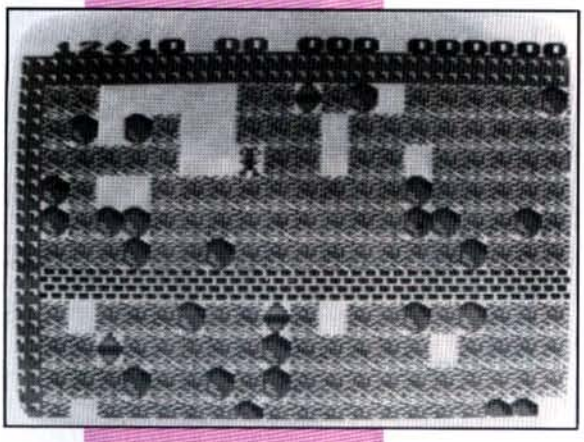
En este excelente simulador de combate, debes evitar que el enemigo destroce tus ciudades y hunda tus barcos. Posees un mapa sobre el cual te podrás desplazar, con el consiguiente gasto de combustible, para atacar sus ciudades, representadas por pequeños rectángulos, o sus barcos y dibujados como puntos blancos sobre el mapa. Para el ataque cuentas con tres armas: barcos, aviones bombarderos y un satélite defensivo para destruir las oleadas de misiles dirigidas hacia nuestras ciudades.
Cuando enfrentes un barco, debes hacerlo con uno propio, con la ayuda de un radar y poderosos torpedos. Ten cuidado de no ser alcanzado por los misiles mar-mar que te serán arrojados. Las ciudades enemigas podrás destruirlas con los bombarderos. Cuando sientas la alarma, significará que te han lanzado misiles. Un contador comenzará la cuenta regresiva que te indicará el tiempo que tienes para eliminarlos con el satélite, antes que éstos hagan impacto en tus ciudades.



SONIDO	6.4
GRAFICACION	6.6
ADICCION	6.5
PRESENTACION	6.6
PROMEDIO	6.525

BOULDER DASH I

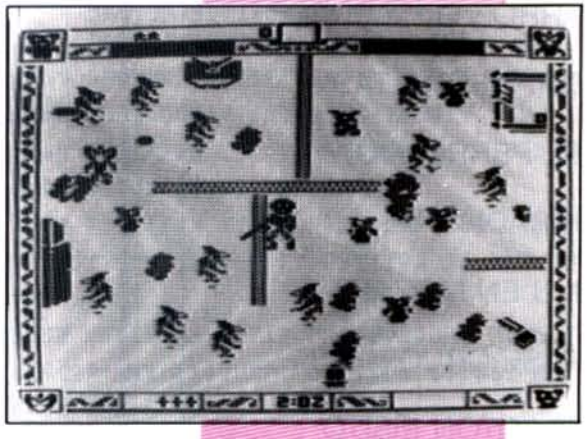
SONIDO	6
GRAFICACION	6.7
ADICCION	6.7
PRESENTACION	6.6
PROMEDIO	6.35



En este entretenido juego, comandas a Boulder Dash, un simpático personaje, que deberás conducir cavando a través del terreno en búsqueda de valiosos diamantes para poder, luego de juntar un número determinado de ellos, pasar de nivel. Este contiene un laberinto de paredes de ladrillo y de pesadas rocas. Deberás descubrir, en cada uno de los niveles, cómo hacer para transformar rocas en diamantes, para juntar así la cantidad necesaria para poder pasar a la pantalla siguiente. Debes tener cuidado al cavar debajo de una roca, ya que ésta te puede caer encima. También debes tener cuidado de los extraños seres que se moverán por los túneles cavados por ti, que tratarán de alcanzarte y matarte. Los podrás eliminar haciendo que les caigan rocas encima de ellos. Este juego es similar al Boulder Dash II, del que sólo difiere en sus escenarios.

GREMLINS

De la famosa película GREMLINS ha sido extraído este entretenido juego. Los GREMLINS, extrañas criaturas venidas de otro planeta, se han instalado en tu casa. Son dóciles y fáciles de capturar y encerrar. Pero cuando se alimentan se vuelven feroces y únicamente matándolos con una espada podrá evitar que lo ataquen. Tienes hasta las seis de la mañana para capturar a los extraños seres. De lograrlo, obtendrás un puntaje extra. Deberás, también, ir recogiendo la comida que hay en el suelo para evitar, así, que los pequeños seres se alimenten y se transformen. Existen unas zonas que harán que los extraterrestres que pasen por allí se dupliquen. Deberás evitar también que los malignos seres se acerquen a la jaula y liberen a los que allí se encuentren encerrados.



SONIDO	6.7
GRAFICACION	6.6
ADICCION	6.5
PRESENTACION	6.5
PROMEDIO	6.55

TURBO PUZZLE:

DIAGRAMA 1

TIZARAN

EXPORTAR TINTE H

RADIO KANT

NI AJEN

MARCAN P

SIMON BARROS

CUELE BREA

TUMBE LA X

BOLLITO FA

HONRE SU YESO H

TENDI EL TV LLI

ARROZ O

DR. SIERRALTAS

EL PERRO KARPI

IRRADIAR VE

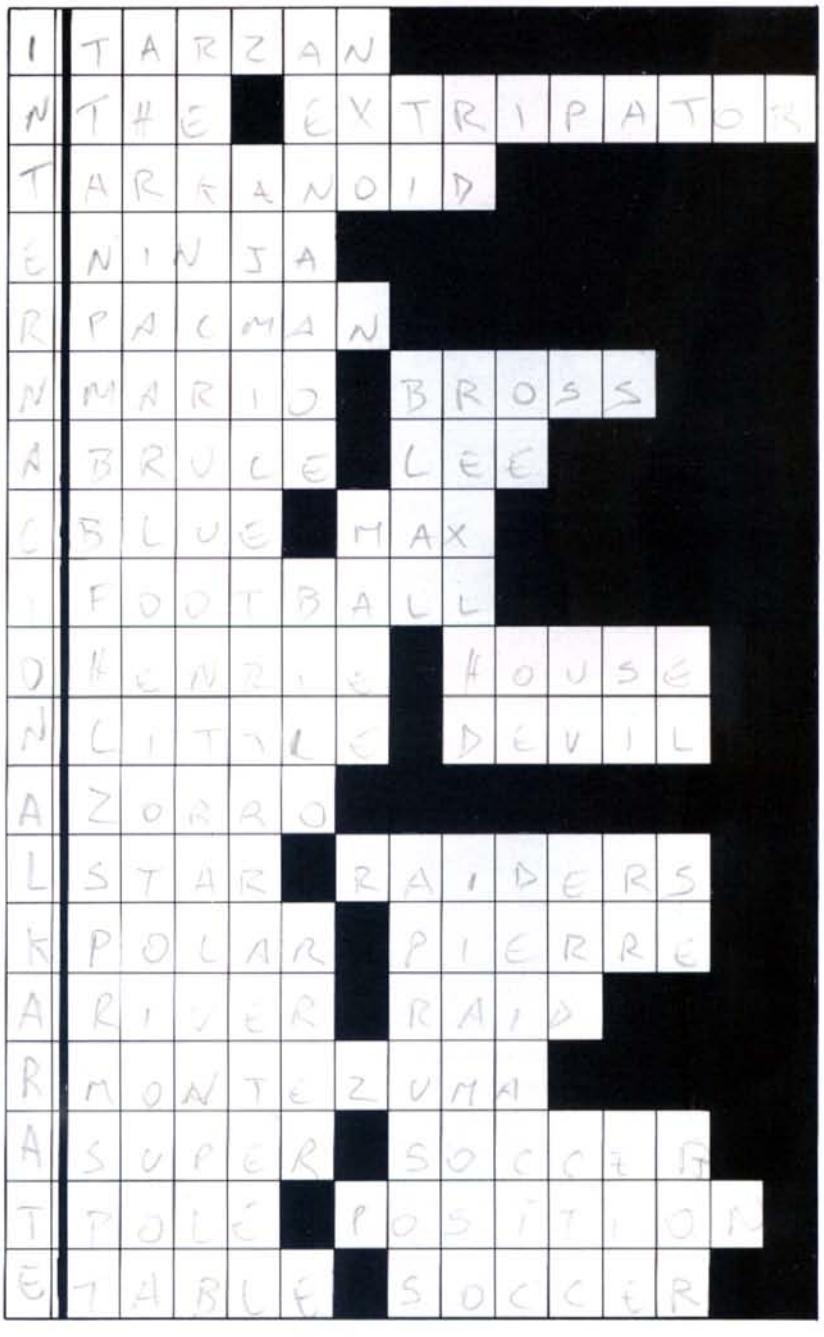
MU EN MOZART

PASE CRUCEROS

PELOTON PISITO

CABLE SECRETO

DIAGRAMA 2



En las extrañas frases del Diagrama 1, están escondidos los nombres de algunos de los más entretenidos juegos ATARI; para encontrarlos debes reordenar las letras de cada una, teniendo en cuenta que hay una de más, la cual deberá ir poniéndose en la columna señalada, donde se podrá leer verticalmente, el nombre de otro de tus juegos favoritos.

Todos los títulos de juegos, puestos aquí, han aparecido en alguno de los rankings de ediciones anteriores.

BASIC

```
1 REM SAVE "D:BASIC9.LST
100 DCB=768:POKE 752,1
200 DIM SECTOR$(128),CANT$(6),PRECIO$(
7),MODIF$(5),DESCRIP$(26),BLANCOS$(200
),RESP$(1),RAYA$(50),PROVEEDOR$(16)
210 DIM PROG$(6)
220 FOR I=1 TO 5:READ A:PROG$(I,I)=CHR
$(A):NEXT I
250 FOR I=1 TO 50:RAYA$(I,I)="-":NEXT
I
300 FOR X=1 TO 200:BLANCOS$(X)=" ":NEX
T X
400 GOSUB 20000
1000 GRAPHICS 0:POKE 710,1:? "
     Menu principal"
1100 ? "↓↓↓↓1 ALTAS"
1200 ? "↓↓2 MODIFICACIONES/CONSULTAS"
1360 ? "↓↓3 INICIALIZAR DISKETTE"
1400 ? "↓↓↓↓SU OPCION: ";:INPUT RESP
1410 ? "":ON RESP GOSUB 2400,3355,110
00
1450 GOTO 1000
1850 REM ******************************
         ***     E/S DE SECTORES   ***
         ******************************
1900 SECTORHI=INT(SECTOR/256)
2000 SECTORLO=SECTOR-(SECTORHI*256)
2100 POKE DCB+10,SECTORLO
2200 POKE DCB+11,SECTORHI
2300 X=USR(ADR(PROG$)):RETURN
2350 REM ******************************
         ***          ALTAS        ***
         ******************************
2400 POSITION 1,1:? "ARTICULO:":POSITI
ON 11,2:? RAYA$(1,4)
2500 POSITION 1,4:? "DESCRIPCION:":POS
ITION 14,5:? RAYA$(1,26)
2550 POSITION 1,7:? "PROOVEDOR:":POSIT
ION 12,8:? RAYA$(1,16)
2600 POSITION 1,10:? "CANTIDAD:":POSIT
ION 11,11:? RAYA$(1,6)
2700 POSITION 1,13:? "PRECIO:":POSITIO
N 9,14:? RAYA$(1,7)
2850 POSITION 10,1:INPUT CODIGO
2870 SECTOR=CODIGO/2+0.5
2875 SECTOR$=BLANCOS$
2892 SECTOR=INT(SECTOR)
2894 POKE DCB+2,82:GOSUB 1900
2900 POSITION 13,4:INPUT DESCRIP$
2920 POSITION 11,7:INPUT PROVEEDOR$
2950 POSITION 10,10:INPUT CANT$
3000 POSITION 8,13:INPUT PRECIO$
3070 SECTOR=CODIGO/2+0.5
3080 AUX=64:IF INT(SECTOR)=SECTOR THEN
 AUX=0
3085 SECTOR=INT(SECTOR)
3100 SECTOR$(AUX+1,AUX+64)=BLANCOS$(1,
64):SECTOR$(1+AUX,26+AUX)=DESCRIP$
3130 SECTOR$(27+AUX,43+AUX)=PROVEEDOR$
:SECTOR$(44+AUX,50+AUX)=CANT$
3150 SECTOR$(51+AUX,58+AUX)=PRECIO$
3200 POKE DCB+2,87:GOSUB 1900
3250 RETURN
3350 REM ******************************
         ***     MODIFICACIONES    ***
         ******************************
3355 ? "ARTICULO A MODIFICAR: ":INPUT
CODIGO:SECTOR=CODIGO/2+0.5
3356 AUX=64:SECTOR$=BLANCOS$
3357 IF INT(SECTOR)=SECTOR THEN AUX=0
3359 SECTOR=INT(SECTOR)
3360 POKE DCB+2,82:GOSUB 1900:? ""
3400 POSITION 1,1:? "ARTICULO: ";CODIG
O:POSITION 11,2:? RAYA$(1,4)
3500 POSITION 1,4:? "DESCRIPCION: ";SE
CTOR$(1+AUX,26+AUX):POSITION 14,5:? RA
YA$(1,26)
3550 POSITION 1,7:? "PROOVEDOR: ";SECT
OR$(27+AUX,43+AUX):POSITION 12,8:? RAY
A$(1,16)
3600 POSITION 1,10:? "CANTIDAD: ";SECT
OR$(44+AUX,50+AUX):POSITION 11,11:? RA
YA$(1,6)
3700 POSITION 1,13:? "PRECIO: ";SECTOR
$(51+AUX,58+AUX):POSITION 9,14:? RAYA$
(1,7)
3850 POSITION 10,1:INPUT CODIGO
3900 POSITION 13,4:INPUT DESCRIP$
3920 POSITION 11,7:INPUT PROVEEDOR$
```

```
3950 POSITION 10,10:INPUT CANT$
4000 POSITION 8,13:INPUT PRECIO$
4070 SECTOR=CODIGO/2+0.5
4080 AUX=64:IF INT(SECTOR)=SECTOR THEN
 AUX=0
4085 SECTOR=INT(SECTOR)
4100 SECTOR$(AUX+1,AUX+64)=BLANCOS$(1,
64):SECTOR$(1+AUX,26+AUX)=DESCRIP$
4130 SECTOR$(27+AUX,43+AUX)=PROVEEDOR$
:SECTOR$(44+AUX,50+AUX)=CANT$
4150 SECTOR$(51+AUX,58+AUX)=PRECIO$
4200 POKE DCB+2,87:GOSUB 1900
4250 RETURN
10900 REM ******************************
          ***   INICIALIZACION     ***
          ******************************
11000 ? "¨  ESTA SEGURO S/N":INPUT RES
P$:IF RESP$<>"S" THEN RETURN
11005 XIO 254,#1,0,0,"D1:"
19999 REM ******************************
          *** DIRECCION VARIABLE   ***
          ******************************
20000 ADDR=ADR(SECTOR$)
20010 ADDRHI=INT(ADDR/256)
20020 ADDRLO=ADDR-(ADDRHI*256)
20030 POKE DCB+4,ADDRLO
20040 POKE DCB+5,ADDRHI:RETURN
20050 DATA 104,32,83,228,96
```

ASSEMBLER

```
0100 ;SAVE#D:ANALCMD.MAC
0110 ;
0120 ; RUTINA QUE ANALIZA SI EL
0130 ; COMANDO ES VALIDO. Y SI LO
0140 ; ES SALTA A LA SUBRUTINA QUE
0150 ; LO EJECUTA.
0160 ;
0170 ANALIZOCMD
0180 ;
0190 ; POINTERAUX APUNTA A LA PRIMER
0200 ; POSICION DE LA TABLA DE COMAN-
0210 ; DOS PARA RECORRERLA.
0220 ;
0230     LDA # <CMD
0240     STA POINTERAUX
0250     LDA # >CMD
0260     STA POINTERAUX+1
0270     LDY #0
0280 LOOPCMD
0290 ;
0300 ; SI EN LA TABLA DE COMANDOS
0310 ; APARECE UN $FF, ES PORQUE
0320 ; NO EXISTE NINGUN OTRO COMANDO,
0330 ; POR LO TANTO, EL COMANDO INGRE
0340 ; SADO NO ESTA EN LA TABLA Y ES
0350 ; INVALIDO.
0360 ;
0370     LDA (POINTERAUX),Y
0380     CMP #$FF
0390     BNE CMDVALIDO?
0400     LDA # <ROTINV
0410     STA MANDO
0420     LDA # >ROTINV
0430     STA MANDO+1
0440     JSR IMPRIMO
0450     RTS
0460 CMDVALIDO?
0470 ;
0480 ; SI NO APARECIO EL $FF, COMPARO
0490 ; EL COMANDO INGRESADO EN LINEA
0500 ; CON LA SINTAXIS DEL COMANDO
0510 ; ESCRITA EN LA TABLA.
0520 ;
0530     LDY #1
0540     LDX #0
0550 SONIGUALES?
0560     LDA LINEA,X
0570     CMP (POINTERAUX),Y
0580     BNE OTROCMD
0590     INY
0600     INX
0610     LDA LINEA,X
0620     CMP #$9B
0630     BNE SONIGUALES?
0640     DEY
0650     TYA
0660     LDY #0
0670     CMP (POINTERAUX),Y
0680     BEQ SALTOALCOMANDO
0690 OTROCMD
0700 ;
0710 ; SI LOS COMANDOS NO SON IGUALES
0720 ; ANALIZO EL SIGUIENTE.
0730 ;
0740     CLC
0750     LDY #0
0760     LDA (POINTERAUX),Y
0770     STA AUXSUM
0780     LDA POINTERAUX
0790     ADC #3
0800     STA POINTERAUX
0810     LDA POINTERAUX+1
0820     ADC #0
0830     STA POINTERAUX+1
0840     CLC
0850     LDA POINTERAUX
0860     ADC AUXSUM
0870     STA POINTERAUX
0880     LDA POINTERAUX+1
0890     ADC #0
0900     STA POINTERAUX+1
0910     JMP LOOPCMD
0920 SALTOALCOMANDO
0930 ;
0940 ; SI LOS COMANDOS FUERON IGUALES
0950 ; SALTO A LA DIRECCION DE LA
0960 ; RUTINA DE ATENCION DEL COMANDO
0970 ; QUE SE ENCUENTRA APUNTADA POR
0980 ; LOS DOS BYTES SIGUIENTES AL
```

```
0990 ; NOMBRE DEL COMANDO EN LA TABLA
1000 ;
1010     LDA (POINTERAUX),Y
1020     TAY
1030     INY
1040     LDA (POINTERAUX),Y
1050     STA JUMPER
1060     INY
1070     LDA (POINTERAUX),Y
1080     STA JUMPER+1
1090     JMP (JUMPER)



0100 ;SAVE#D:LIST.MAC
0110 ;
0120 ; RUTINA QUE EJECUTA EL COMANDO
0130 ; LIST. REALIZA UNA IMPRESION
0140 ; DEL LISTADO DEL PROGRAMA EN
0150 ; LA PANTALLA DEL COMPUTADOR.
0160 ;
0170 LIST
0180 ;
0190 ; POINTERAUX1, APUNTA AL INICIO
0200 ; DE LA TABLA DE INSTRUCCIONES
0210 ; PARA RECORRERLA E IMPRIMIR SU
0220 ; CONTENIDO.
0230 ;
0240     LDA # <TABLA
0250     STA POINTERAUX1
0260     LDA # >TABLA
0270     STA POINTERAUX1+1
0280 INITLIST
0290 ;
0300 ; SI LLEGUE AL FINAL DE LA TABLA
0310 ; YA EJECUTE TODO EL COMANDO POR
0320 ; LO TANTO CONTINUO CON EL
0330 ; PROGRAMA PRINCIPAL.
0340 ;
0350     LDA FINTAB+1
0360     CMP # >TABLA
0370     BNE NOVAC1
0380     LDA FINTAB
0390     CMP # <TABLA
0400     BNE NOVAC1
0410     RTS
0420 NOVAC1
0430 ;
0440 ; UTILIZANDO LAS SUBRUTINAS DEL
0450 ; SISTEMA OPERATIVO DEL COMPUTA
0460 ; DOR, TRANSFORMO LOS DOS BYTES
0470 ; DEL NUMERO DE INSTRUCCION,
0480 ; LO Y HI BYTES, EN NUMEROS
0490 ; ATASCII QUE PODEMOS IMPRIMIR
0500 ; EN LA PANTALLA.
0510 ;
0520     LDY #0
0530     LDA (POINTERAUX1),Y
0540     STA FR0
0550     INY
0560     LDA (POINTERAUX1),Y
0570     STA FR0+1
0580     JSR $D9AA
0590     JSR $D8E6
0600 ;
0610 ; EL RESULTADO DE ESTA RUTINA
0620 ; QUEDA EN LBUFF.
0630 ;
0640     LDY #0
0650 L20
0660     LDA LBUFF,Y
0670     BMI INV
0680     STY GUARDOY
0690     JSR $F2B0
0700     LDY GUARDOY
0710     INY
0720     JMP L20
0730 INV
0740 ;
0750 ; EL ULTIMO CARACTER DEL NUMERO
0760 ; EL SISTEMA OPERATIVO LO DA
0770 ; EN VIDEO INVERSO PARA SABER
0780 ; QUE ESTE ES EL ULTIMO.
0790 ;
0800     AND #$7F
0810     JSR $F2B0
0820     LDA #32
0830     JSR $F2B0
0840 ;
0850 ; UNA VEZ QUE DESPLEGAMOS EL
0860 ; NUMERO DE LINEA, RECORREMOS
0870 ; LA INSTRUCCION Y LA IMPRIMIMOS
0880 ;
0890     LDY #2
0900     LDA (POINTERAUX1),Y
0910     TAX
0920     INY
0930 L21
0940     LDA (POINTERAUX1),Y
0950     STY GUARDOY
0960     STX GUARDOX
0970     STA GUARDOA
0980     LDA #27
0990     JSR $F2B0
1000     LDA GUARDOA
1010     JSR $F2B0
1020     LDY GUARDOY
1030     LDX GUARDOX
1040     INY
1050     DEX
1060     BNE L21
1070 ;
1080 ; IMPRIMO UN RETURN PARA QUE
1090 ; EL CURSOR CAMBIE DE LINEA.
1100 ;
1110     LDA #$9B
1120     JSR $F2B0
1130 ;
1140 ; MODIFICO A POINTERAUX1 PARA
1150 ; QUE APUNTE A LA NUEVA
1160 ; INSTRUCCION PARA SU LISTADO.
1170 ;
1180     CLC
1190     LDY #2
1200     LDA (POINTERAUX1),Y
1210     ADC #3
1220     STA CANT
1230     CLC
1240     LDA POINTERAUX1
1250     ADC CANT
1260     STA POINTERAUX1
1270     LDA POINTERAUX1+1
1280     ADC #0
1290     STA POINTERAUX1+1
```

```
1300      CMP FINTAB+1
1310      BCC NOVAC1
1320      LDA POINTERAUX1
1330      CMP FINTAB
1340      BCC NOVAC1
1350      RTS


0100 ;SAVE#D:GENERAL.MAC
0110 ;
0120 ; EN ESTE LISTADO, INGRESAMOS
0130 ; ALGUNAS RUTINAS DE USO
0140 ; GENERAL EN EL PROGRAMA.
0150 ;
0160 OVERFLOW
0170 ;
0180 ; RUTINA QUE IMPRIME EL MENSAJE
0190 ; DE MEMORIA EXCEDIDA.
0200 ;
0210      LDA # <ROTOVER
0220      STA MANDO
0230      LDA # >ROTOVER
0240      STA MANDO+1
0250      JSR IMPRIMO
0260      RTS
0270 IMPRIMO
0280 ;
0290 ; RUTINA QUE IMPRIME UN TEXTO
0300 ; APUNTADO POR MANDO Y MANDO+1
0310 ; HASTA QUE ENCUENTRA UN *.
0320 ;
0330      LDY #0
0340 LOOPIMPRIMO
0350      LDA (MANDO),Y
0360      CMP #'*
0370      BEQ FINIMPRIMO
0380      STY GUARDOY
0390      JSR $F2B0
0400      LDY GUARDOY
0410      INY
0420      JMP LOOPIMPRIMO
0430 FINIMPRIMO
0440      RTS
0450 SETPOINTERAUX
0460      LDA # <TABLA
0470      STA POINTERAUX
0480      LDA # >TABLA
0490      STA POINTERAUX+1
0500      RTS
0510 NEW
0520 ;
0530 ; RUTINA QUE EJECUTA EL COMANDO
0540 ; NUEVO, QUE BORRA TODO EL PRO-
0550 ; GRAMA. EN REALIDAD LO UNICO
0560 ; QUE HACE ES CAMBIAR EL PUNTERO
0570 ; FINTAB AL INICIO DE LA TABLA,
0580 ; SIN BORRAR LA MEMORIA, PUES
0590 ; NO ES NECESARIO.
0600 ;
0610      LDA # <TABLA
0620      STA FINTAB
0630      LDA # >TABLA
0640      STA FINTAB+1
0650      RTS
0660 EXISTELIN
0670 ;
0680 ; RUTINA QUE RECIBE UN NUMERO
0690 ; DE LINEA EN DESDENUM Y
0700 ; DESDENUM+1 Y VERIFICA SI ESTE
0710 ; NUMERO DE LINEA EXISTE EN EL
0720 ; LISTADO.
0730 ;
0740      JSR SETPOINTERAUX
0750 LOPESTA?
0760      LDA POINTERAUX+1
0770      CMP FINTAB+1
0780      BCC PUEDESER
0790      LDA POINTERAUX
0800      CMP FINTAB
0810      BCC PUEDESER
0820      RTS
0830 PUEDESER
0840      LDY #1
0850      LDA DESDENUM+1
0860      CMP (POINTERAUX),Y
0870      BEQ VEOLO
0880      BCS AVANZO
0890      RTS
0900 AVANZO
0910 ;
0920 ; AVANZO A LA SIGUIENTE
0930 ; INSTRUCCION.
0940 ;
0950      CLC
0960      LDY #2
0970      LDA (POINTERAUX),Y
0980      STA AUXSUM
0990      LDA POINTERAUX
1000      ADC #3
1010      STA POINTERAUX
1020      LDA POINTERAUX+1
1030      ADC #0
1040      STA POINTERAUX+1
1050      CLC
1060      LDA POINTERAUX
1070      ADC AUXSUM
1080      STA POINTERAUX
1090      LDA POINTERAUX+1
1100      ADC #0
1110      STA POINTERAUX+1
1120      JMP LOPESTA?
1130 VEOLO
1140      LDY #0
1150      LDA DESDENUM
1160      CMP (POINTERAUX),Y
1170      BEQ ESTALINEA
1180      BCS AVANZO
1190      RTS
1200 ESTALINEA
1210 ; SI LA LINEA ES HALLADA
1220 ; FLAGESTA VUELVE CON $FF
1230 ;
1240      LDA #$FF
1250      STA FLAGESTA
1260      RTS
1270 FLAGESTA
1280      .BYTE $00
```

La nueva generación de Software para Computadores Atari

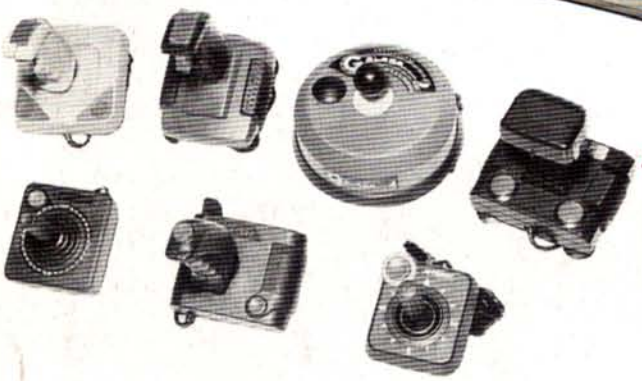
ADQUIERALOS EN LOS SIGUIENTES PUNTOS DE VENTAS

• **ANTOFAGASTA:** COOPERCARAB KW VIDEO LA ESPAÑOLA • **VIÑA DEL MAR:** FALABELLA VIÑA INSIS MPR COMPUTACION • **VALPARAISO:** COMPUTRONIC • **SANTIAGO:** AUDIO BICICLETA INTERNAC CASA ROYAL CENTRO ATARI COMERCIAL ESTADO COMPUMANQUE COMPUCENTER FALABELLA AHUMADA FALABELLA P. ARAUCO IMACO INFOGROUP PC STORE PETERSEN ROLEC SUPERMERCADOS UNIMARC TASCO VIDEO CLUB INTERNACIONAL • **RANCAGUA:** CASA ZUÑIGA • **CURICO:** MULTIHOGAR • **TALCA:** LIBRERIA EL AHORRO MULTICENTRO VIDEO CLUB CASSAL • **CHILLAN:** CASA EDISON • **CONCEPCION:** COOPERCARAB DISMAR DISMAR 2 EQUS PHANTER RAPSODIA SESCO • **LOS ANGELES:** DISTRIBUIDORA MERINO • **ANGOL:** SCORPIO • **VICTORIA:** CASA SIGMUND • **TEMUCO:** COMERCIAL MANQUEHUE ESTABLECIMIENTOS GEJMAN FALABELLA • **PUCON:** ELTIT • **VILLARRICA:** JOYERIA KETTERER • **VALDIVIA:** ELECTROMUSICA • **LA UNION:** IMPORTADORA COSMOS • **OSORNO:** CASA REAL FOTO EXPRESS • **PUERTO VARAS:** ELECTRO HORN • **PUERTO MONTT:** COMERCIAL MANQUEHUE DIMARSA • **COYHAIQUE:** FACI HOGAR • **PUNTA ARENAS:** BALFER LTDA.

COMPUTACION

1 COMPUTADOR 65 XE
1 CASSETTERA
2 JOYSTICKS
6 JUEGOS

$ 64.700



JOYSTICK

JOYSTICK GALAGA	$ 3.750
BG - 201	
CON MICROSWITCH	$ 6.040
BG - 747	$ 3.000
BG - 105	$ 2.500
BG - 124	$ 1.500
QUICK SHOT I	$ 2.200
QUICK SHOT II (AUTODISPARO)	$ 5.250

CAJAS PORTADISKETTES

3.1/2 40 UNIDADES	$ 3.750
5.1/4 50 UNIDADES	$ 3.650
5.1/4 100 UNIDADES	$ 4.500

PROGRAMAS EDUCATIVOS

SERIE TM (TELEMATICA)
DESDE $ 1.790

CASSETTES EDUCATIVOS
TURBO SOFTWARE DESDE $ 895

DISKETTES 5 1/5 2S-2D:

VERBATIN	$ 390 C/U
FUJI	$ 355 C/U
BASF	$ 395 C/U
MEMOREX	$ 310 C/U
CIS	$ 290 C/U
GOLDSTAR	$ 225

DISKETTES 3,5"

BASF	$ 630
VERBATIN	$ 640
FUJI	$ 695
MEMOREX	$ 720

ELECTRONICA
CASA ROYAL M.R. LTDA.
PRIMER CENTRO ELECTRONICO CHILENO

AV. L. B. O'HIGGINS 845 FONOS: 333908 - 391524
MONJITAS 813 FONOS: 399046 - 392714

FAX: 399047 - TELEX 340517

DESPACHOS A PROVINCIA PREVIO ENVIO DE CHEQUE, VALE VISTA O GIRO TELEGRAFICO A CORREO 21 - CASILLA 395 -V -STGO.

•ELECTRONICA •ELECTRICIDAD •COMPUTACION •AUDIO - VIDEO