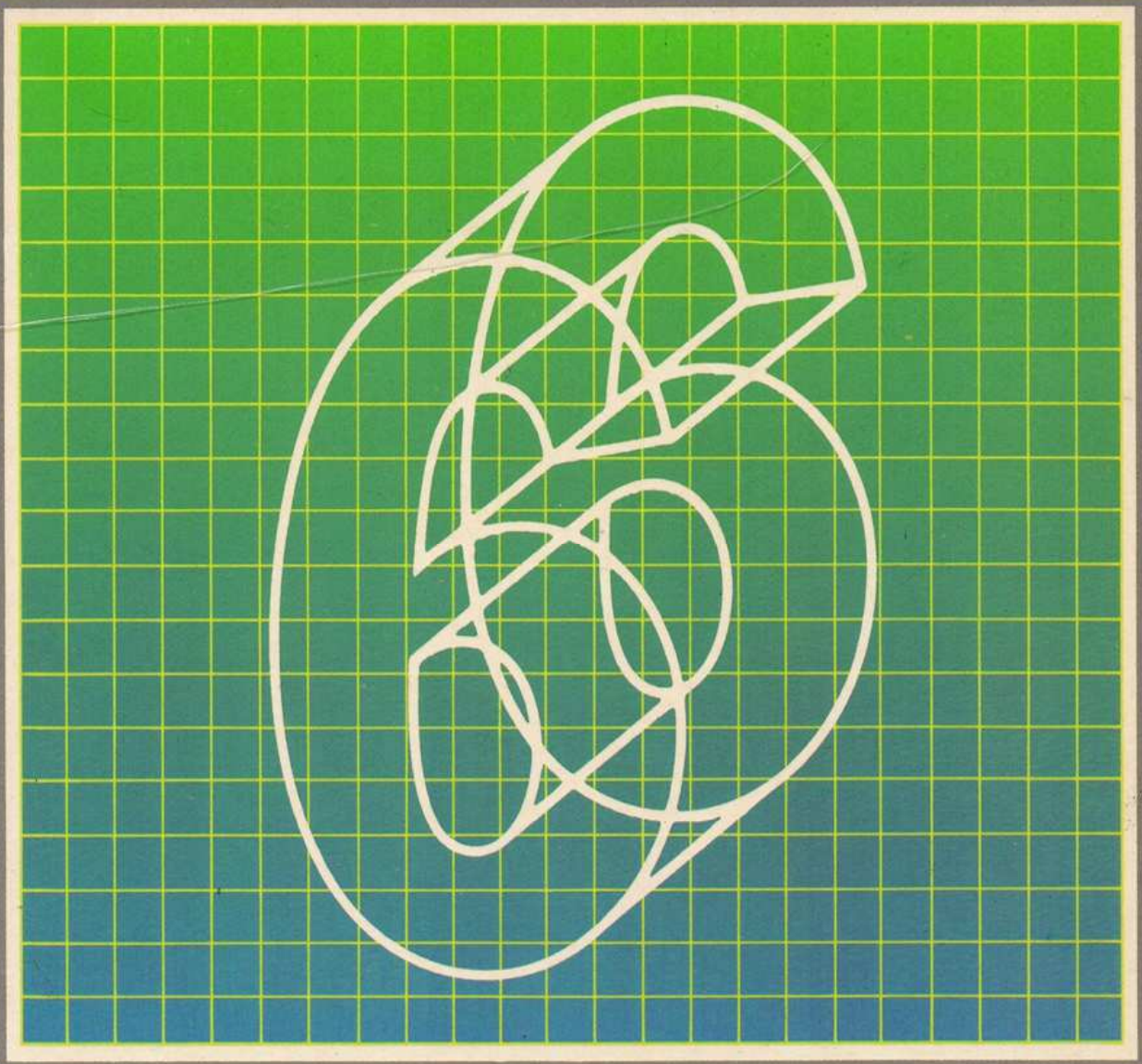
BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



APLICACIONES PRACTICAS EN BASIC.

PROCEDIMIENTOS Y RUTINAS

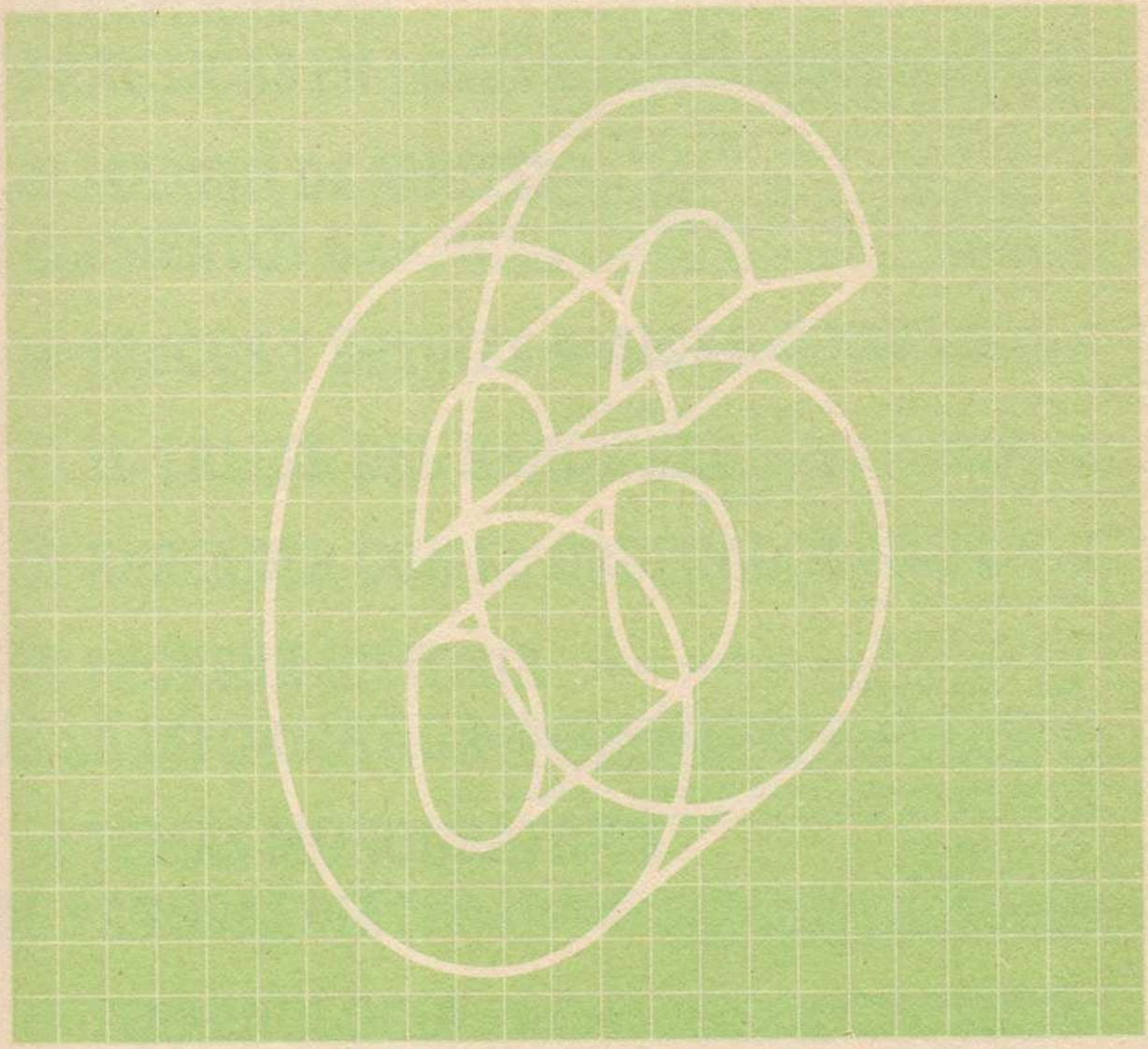
EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS · SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO · TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Director de la colección:
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:
BRAVO-LOFISH.

Maquetación:
D. S. M.

Dibujos:
JOSE OCHOA

Tomo 6. «Experiencias prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomuniación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-056-1.
ISBN de la obra: 84-7688-047-2

Fotocomposición:
ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. PINTO (Madrid).

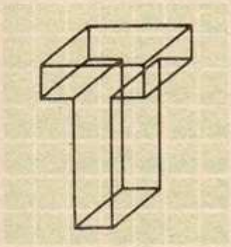
Depósito legal: M-43.185-1986
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S. A.
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Enero, 1987.

INDICE

EXPERIENCIA Y PRACTICAS EN LOGO



TRABAJEMOS en el ordenador que sea y con un lenguaje de programación u otro, la forma de comunicación entre el ordenador y el usuario es siempre la misma.

Nosotros nos comunicamos con el ordenador introduciéndole por el teclado las órdenes que queremos que realice. Todo aquello que vamos tecleando aparece al mismo tiempo sobre la pantalla, al igual que ocurre, por ejemplo, cuando estamos escribiendo a máquina, que todo lo que vamos tecleando se va imprimiendo sobre un papel. Así conseguimos saber si lo que estamos tecleando es, en realidad, lo deseado y si está escrito correctamente, pudiendo proceder a su corrección en el caso de que no lo sea.

Pero no basta con ir tecleando las órdenes. El ordenador se tiene que enterar que todo lo que estamos tecleando son órdenes que él debe realizar. Para ello tenemos que introducírselo en su memoria, para que sea analizado, procesado y, por último, recibamos alguna contestación. La forma de conseguir esto es presionando una tecla en concreto que poseen todos los ordenadores. Se trata de la tecla que, según el modelo de ordenador con el que estemos trabajando, puede recibir el nombre de ENTER, INTRO, RETURN, NEWLINE, etc.

Por lo general, esta tecla siempre se expresa de la siguiente forma:

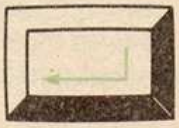


Fig. 1.

Y su significado podría ser "introduce en memoria".

Cuando trabajamos con el Logo, vamos dando órdenes, bien de una en una o bien en pequeños grupos, pero para que sean ejecutadas siempre hemos de pulsar la tecla de introducción, y todo aquello que hemos tecleado se realizará, siempre y cuando el ordenador lo entienda, es decir, que esté bien escrito y que la orden dada pertenezca al lenguaje con el que se está trabajando.

El ordenador, por su parte, se comunica con nosotros por la pantalla. Es allí donde nosotros recibimos sus mensajes e incluso sus órdenes. Los mensajes que nos da pueden ser muy variados; por ejemplo, mensajes de error, mensajes de información, etc. También es por la pantalla por donde el ordenador nos devuelve el resultado a las órdenes que nosotros les hemos dado. Bien sea un dibujo, una relación de datos o cualquier otro tipo de proceso.

De esta forma, está totalmente lograda la comunicación directa entre la máquina y el hombre.

Podemos hablar de dos formas de dar las órdenes a un ordenador. Una forma directa y una forma indirecta.

La forma directa corresponde a la ejecución inmediata de una o varias órdenes que se dan.

La forma indirecta corresponde al almacenamiento en memoria de una o varias órdenes, que no son ejecutadas hasta el momento que nosotros determinemos.

Hasta el momento, todos los ejemplos que hemos ido realizando con el Logo han consistido en un número determinado de órdenes dadas de una en una, o bien agrupándolas en

El Logo me da información sobre el estado en que se encuentra el lápiz de la Tortuga.

una misma línea que, una vez tecleadas, eran inmediatamente introducidas y ejecutadas.

Así obteníamos nuestros dibujos, que se iban realizando en pequeñas fases hasta que se completaban.

¿Pero qué pasa si queremos hacer, por ejemplo, un dibujo de una sola vez?

En este caso podemos hacer dos cosas. Si tecleamos todas las órdenes que realizan un dibujo seguidas una tras otra, y cuando se hayan tecleando todas se introducen, el dibujo se hará de una sola vez. Pero date cuenta que esto sólo es válido si el número de órdenes dadas es uno determinado. No se pueden dar más de un cierto número de órdenes seguidas, ya que las limitaciones que tiene el Logo no lo permite. Esta solución, pues, sólo será válida para la realización de procesos cortos, con pocas órdenes.

Si el número de órdenes que realizan un proceso es muy grande, tendremos que hacer lo que antes denominamos como:

"órdenes indirectas"

Iríamos tecleando las órdenes, pero no se ejecutarían hasta el momento que nosotros determinásemos. En este caso, ya no existen limitaciones al número de órdenes, pero sí hay que dar estas órdenes de una forma diferente a como lo hacíamos hasta ahora. Recuerda que estas órdenes se van a ir almacenando en la memoria, por tanto, no recibimos ningún tipo de contestación por parte del ordenador.

En todos los lenguajes de programación se puede realizar este proceso, y dependiendo de cada uno, la forma de hacerlo es diferente. Por ejemplo, en Basic la forma que existe para ir almacenando las órdenes es numerándolas. Se consigue de esta forma un conjunto de órdenes perfectamente ordenadas que permanecen en memoria.

En Logo, la forma de hacerlo es diferente, pero la lógica es la misma. Se consigue en todos los casos una serie de órdenes, las cuales tienen en común que pertenecen al mismo proceso. Se crea así una unidad, que recibe un nombre determinado. En la mayoría de los casos el nombre que recibe es PROGRAMA, y en el caso del Logo, se denomina PROCEDIMIENTO.

Esta unidad, bien sea un Programa o un Procedimiento, es la que se encuentra almacenada en memoria, y allí permanecerá hasta que demos la orden oportuna para que se borre o bien tengamos un corte de la corriente eléctrica.

Podremos obtener siempre que queramos el listado de las órdenes, así como realizar cambios, borrar e insertar nuevas órdenes y, por supuesto, ordenar la ejecución de esas órdenes tantas veces queramos. Estos Programas o Procedimientos son los que se pueden almacenar en soportes externos (cinta, discos, etcétera), y cargar en la memoria del ordenador tantas veces queramos. De esta forma podemos tener a nuestra disposición todo lo que hemos realizado en un momento dado sin necesidad de volver a empezar.

Vamos entonces a ver cómo se realiza este proceso en el Logo, pero ten en cuenta que a partir de este momento, siempre que hagamos algún ejemplo o ejercicio, nos referiremos a él con el nombre de PROCEDIMIENTO.

Creación de procedimientos

Supongamos que queremos realizar un dibujo de un tres. Tendremos que crearnos un procedimiento en el cual estén todas las órdenes que realizan dicho dibujo. Lo primero que haremos será buscar un nombre que diferencie a este procedimiento de otros. En principio, este nombre puede ser uno cualquiera. (Existen limitaciones en el nombre que se dé, que ya veremos más adelante.)

Intenta buscar un nombre que tenga relación con lo que vas a hacer. En nuestro caso el nombre más idóneo sería TRES.

Bueno, ya sabemos cómo llamar a nuestro procedimiento. Ahora tenemos que preparar al Logo para que vaya recibiendo las órdenes que van a realizar nuestro dibujo, pero, por el momento, sólo las irá almacenando. Esto lo hacemos dando la siguiente orden:

? PARA TRES

Puedo saber en cualquier momento las coordenadas de la posición de la Tortuga.

Con esta orden lo único que hacemos es decirle que las órdenes que vienen a continuación son las que se necesitan PARA dibujar un TRES.

Vamos a ver qué cambios se producen.

Sitúate dando la orden correspondiente en el modo de la pantalla texto. Esto no es necesario obligatoriamente, pero de esta forma tendremos toda la pantalla para poder escribir.

Una vez que lo hayas realizado, tu pantalla estará de la siguiente forma:

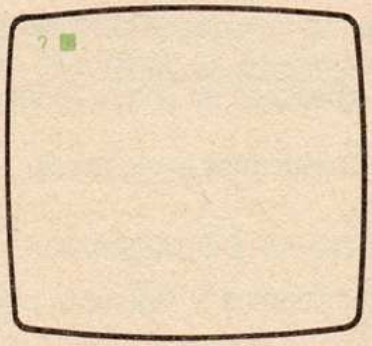


Fig. 2.

Ahora teclea e introduce lo siguiente:

? PARA TRES

La pantalla aparecerá así:

Fig. 3.

```
? PARA TRES
>
```

Observa que en lugar de haber una interrogación en la segunda línea hay el signo >. Este signo es, al igual que la interrogación, el que nos indica que podemos introducir órdenes. Pero en este caso todas las órdenes que vayamos introduciendo no se irán ejecutando.

Ya está todo preparado para comenzar a dar las órdenes que nos dibujan el tres. Como siempre hacemos, damos primero las órdenes de inicialización:

```
> PM
> BP
> OT
> SL
```

Fíjate que ahora las órdenes que has ido tecleando e introduciendo no han sido realizadas. Simplemente se han ido almacenando.

Aquí también puedes dar más de una orden por línea, pero recuerda que tienes que pulsar la tecla ENTER una vez que quieras pasar a otra línea.

Consideramos que una línea es la que empieza con el signo < y acaba justo antes de encontrar otro. Por esto una línea puede contener más de una línea de la pantalla (fila).

Seguiríamos dando las órdenes para completar el dibujo. Ahora vendrían las que nos centran el dibujo:

```
> GD 90
> AV 10
> BL
> GI 180
```

Y a continuación las que hacen el dibujo.

```
> AV 20 RE 20
> GD 90 AV 20
> GI 90 AV 15
> RE 15 GD 90
> AV 20 GI 90
> AV 20
```

Estas serían todas las órdenes para obtener un tres dibujado en la pantalla.

Pero ahora, ¿cómo se puede ver lo que hemos hecho?

Lo primero que hay que hacer es salir del estado en que estamos, es decir, avisar que ya hemos terminado de dar las órdenes y queremos ver cómo sale nuestro dibujo. Para ello hay que dar una última órden dentro de cada procedimiento. Se trata de la orden FIN.

Teclea e introduce al final de las órdenes anteriores:

```
> FIN
```

Si doy órdenes dentro de un procedimiento, no se van ejecutando a la vez que las introduzco.

Inmediatamente después de introducir esta orden aparece el mensaje:

TRES DEFINIDO

La interrogación ha vuelto a aparecer. Estamos en el mismo estado que al principio. Pero hemos hecho algo muy importante, definir nuestro primer procedimiento.

Si ahora das la orden BT (borra texto), todo lo que hay en la pantalla desaparecerá. Pero ¿todo lo que hemos hecho?, ¿se ha borrado también?

No todas las órdenes que hemos dado para definir nuestro procedimiento están guardadas, almacenadas en la memoria del ordenador.

Para poder ver nuestro dibujo lo único que hay que hacer es introducir el nombre que dimos a nuestro procedimiento. Así, pues, teclea e introduce:

```
? TRES
```

En la pantalla aparecerá el dibujo:

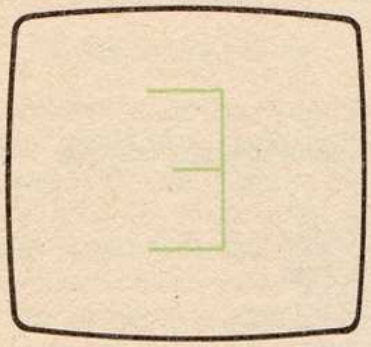


Fig. 4.

A partir de este momento cada vez que introduzcamos la palabra TRES aparecerá el dibujo.

TRES se ha convertido en una orden, nuestra propia orden. Al igual que, por ejemplo, la orden AV 10 hace avanzar diez pasos a la Tortuga, la orden TRES me dibuja un tres en la pantalla.

A partir de ahora cuando tengamos que hacer una misma cosa varias veces ya no tendremos que dar las órdenes necesarias cada vez que lo queramos ver. Simplemente daremos las órdenes una vez dentro de un procedimiento, y siempre que necesitemos ese procedimiento lo llamaremos.

Numerosas veces hemos utilizado un cuadro en nuestro dibujo y en todos ellos hemos tenido que dar las órdenes para que se dibujase. Si hacemos un procedimiento que lo haga, sólo tendremos que dar las órdenes una vez, y siempre que necesitemos un cuadrado nos bastaría con escribir el nombre del procedimiento.

Vamos a crear algunos procedimientos.

```
? PARA CUADRADO
> REPITE 4 [AV 20 GD 90]
> FIN
```

Si ahora introduces CUADRADO, en la pantalla aparecerá un cuadrado dibujado a partir de la posición en la que se encuentre la Tortuga.

Si introducimos lo siguiente:

```
? BP BL OT
? CUADRADO AV 2 CUADRADO
```

Obtendríamos la siguiente pantalla:

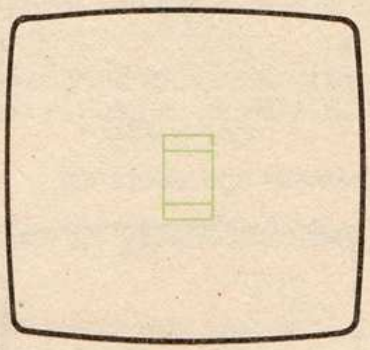


Fig. 5.

Con estas órdenes:

```
? BP BL OT
? REPITE 5 [CUADRADO AV 2]
```

Obtendríamos esta pantalla:

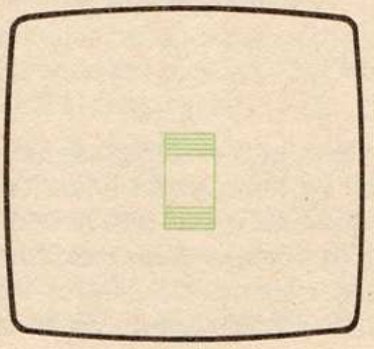


Fig. 6.

Puedo definir un montón de procedimientos y tenerlos guardados en la memoria.

Date cuenta cómo CUADRADO actúa como una orden.

En todos estos casos el tamaño del cuadrado es el mismo, es fijo. Ya veremos cómo podemos conseguir que este tamaño sea variable.

```
? PARA TRIANGULO
> REPITE 3 [AV 20 GD 120]
> FIN
```

Si ahora introducimos:

```
? CUADRADO TRIANGULO
```

Nos aparecerá:

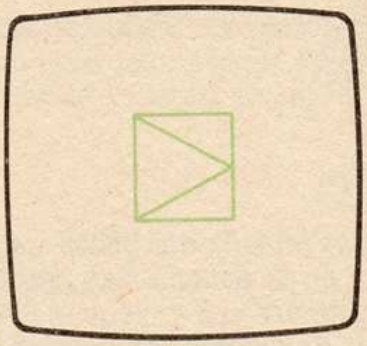


Fig. 7.

El siguiente procedimiento realiza el mismo dibujo anterior:

```
? PARA CUADTRIAN
> REPITE 4 [AV 20 GD 90]
> GD 60 AV 20 GI 120 AV 20
> FIN
```

Si introducimos:

```
? BP
? REPITE 6 [CUADTRIAN]
```

Conseguimos este dibujo:

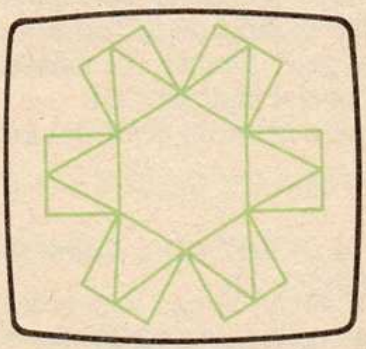


Fig. 8.

De esta manera puedes ir definiendo los procedimientos que quieras. No dés el mismo nombre a dos procedimientos, ya que se producirá un error.

Utilizando los procedimientos, vamos a dibujar una pared de ladrillos.

Con un procedimiento dibujamos un rectángulo, que servirá para enmarcar los ladrillos que pondremos.

```
? PARA MARCO
> SL
> PONPOS [-150 -60]
> BL
> REPITE 2 [AV 160 GD 90 AV 255 GD 90]
> FIN
```

Dibujamos el ladrillo con otro procedimiento.

```
? PARA LADRILLO
> REPITE 2 [AV 20 GD 90 AV 30 GD 90]
> FIN
```

Y lo vamos colocando en diferentes posiciones hasta construir la pared.

```
? BP
? MARCO
? SL
? PONY 100
? BL
? PONCL 2
? REPITE 4 [PONX -150 RE 20 REPITE 8 [LADRILLO GD 90 AV 30 GI 90] PONX -135 RE 20 REPITE [8 LADRILLO GD 90 AV 30 GI 90]]
```

Logos nos informa

Teclea las siguientes órdenes:

```
? BP
? MT
? PONCL 2
? BL AV 10 AV 20 AV 30 AV 40 AV 20 GD 90 AV 30 AV 40
? ESCRIBE COORX
```

El número que ha aparecido en la pantalla (70) es la coordenada X de la posición de la Tortuga cuando ha terminado de pintar.

```
? ESCRIBE COORY
```

Para que se realice uno de los procedimientos que he definido sólo es necesario introducir el nombre que le he dado.

Te habrá aparecido otro número (120), es la coordenada Y de la posición de la Tortuga.

Si anotas los dos números que han aparecido en la pantalla podrás, sin hacer cálculo alguno, realizar el mismo dibujo de esta forma:

```
? BP BL
? PONY 120
? PONX 70
```

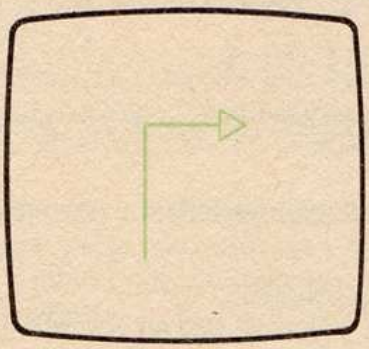


Fig. 9.

Cuando nos interese saber alguna coordenada del punto donde está situada la Tortuga en cualquier momento, ya sabemos que tenemos dos órdenes que nos informarán con exactitud.

Si no hubiésemos tecleado la orden ESCRIBE delante de cada una de estas dos órdenes informativas, el LOGO nos habría dado un error. La primitiva ESCRIBE indica que COORX y COORY deben ser mostradas en la pantalla.

— COORX de la coordenada de las X.
— COORY de la coordenada de las Y.

No vamos ha explicar la orden ESCRIBE ahora, lo veremos con detalle más adelante.

Si queremos conocer las dos coordenadas del punto donde está la Tortuga, podemos utilizar la orden POS que devuelve dos números, uno corresponde a la coordenada vertical y el otro a la horizontal.

```
? BP
? SL
? PONPOS [30 30]
? BL
? ESCRIBE POS
? REPITE 4 [AV 30 GD 90]
? ESPERA 30 SL
? CENTRO
? ESCRIBE POS
```

En el SPECTRUM las órdenes que nos devuelven la posición de la Tortuga son XCOOR e YCOOR de la coordenada X y de la coordenada Y, respectivamente.

El LOGO dispone de varias órdenes de este tipo, primitivas, que sirven para darnos alguna información; a continuación vamos a ver unas cuantas de ellas:

```
? BP
? MT
? PONCL 2
? AV 50 GD 90
? ESCRIBE CL
? ESPERA 40
? PONCL 3
? AV 60
? ESCRIBE CL
```

Hemos pintado dos líneas, cada una de un color, y en la pantalla han aparecido dos números, primero el 2 y luego el 3; si recuerdas la tabla de colores habrás apreciado que estos dos números corresponden a los códigos de color de las dos líneas dibujadas.

Hemos intercalado la orden ESPERA para poder ver más lentamente el proceso: dibujo de línea, aparición del código de color.

En este caso también utilizamos la primitiva ESCRIBE, con el mismo fin que anteriormente; ahora lo que queremos es información referente al lápiz: su color (CL).

- CL

En el logo del SPECTRUM la orden CL no hace lo que acabamos de explicar; su función es bajar el lápiz de la Tortuga, dejándola preparada para dibujar. La orden que nos dará el color del lápiz es COL.

¿Y para saber el color del fondo?

```
? BP
? BL
? REPITE 10 [PONFONDO AZAR 16 ESCRIBE
FONDO ESPERA 20]
```

La pantalla va apareciendo de diferentes colores, también aparece un número, es el

Dependiendo del ordenador con el que trabajemos, podemos guardar los procedimientos en un soporte o en otro.

código de color de fondo de la pantalla gráfica.

— FONDO.

En la versión para el SPECTRUM la orden que debemos teclear cuando queramos saber el código de color de fondo de la pantalla es CF.

¿Para conocer el estado del lápiz?

— LAPIZ.

Esta orden puede devolvernos cuatro estados diferentes:

IL, GOMA, SL o BL

Si nuestro LOGO es para un PC, también recibiremos información sobre el color del lápiz y sobre la paleta actual.

```
? BP
? BL
? AV 100
? IL RE 100
? ESCRIBE LAPIZ
? BL AV 100
? GOMA RE 100
? ESCRIBE LAPIZ
```

Hemos borrado la línea de dos formas: primero, invirtiendo el lápiz, y luego usando la goma, y hemos sido informados de ello con las dos órdenes ESCRIBE LAPIZ que aparecen en el ejemplo.

Si tu ordenador es un PC y solamente deseas saber el número de paleta con la que estás trabajando en ese momento, teclea ESCRIBE PALETA.

También disponemos de una primitiva que nos devuelve el rumbo actual de la Tortuga.

— RUMBO.

```
? BP
? BL
? AV 80 GD 70 AV 80 GD 200 A 50 GI 100 AV
60
? ESCRIBE RUMBO
```

Recuerda que con la orden PONRUMBO n damos a la Tortuga una orientación que puede variar de 0 a 359; el número que nos devolverá la orden RUMBO estará comprendido entre estos dos, ambos inclusive.

— HACIA [x y]

Con esta otra orden conocemos qué rumbo debemos darle a la Tortuga para que se dirija a un determinado punto de la pantalla.

```
? BP
? BL
? PONX 60
? PONY 70
? ESCRIBE HACIA [0 0]
? PONPOS [0 0]
```

En la pantalla te aparece un número, que corresponde al rumbo que lleva la Tortuga cuando se dirige al punto [0 0].

Aplicaciones con la información

Aparte de escribir en pantalla la información, podemos hacer algunas cosas más.

```
? BP BL
? REPITE 10 [PONFONDO AZAR 16 PONCL 2
REPITE 4 [AV FONDO GD 90]
? PONFONDO 10
```

Sí, todas estas órdenes que hemos visto podemos utilizarlas como datos de algunas primitivas.

Teclea este procedimiento:

```
? PARA CIRCULO
> REPITE 36 [AV CL GD FONDO]
> FIN
```

y estas órdenes:

```
? BP
? BL
? PONPOS [40 10]
? REPITE 4 [AV COORX GD 90]
? CENTRO
? ESPERA 30
? PONFONDO 10
? PONCL 1
? CIRCULO
```

Si vas a guardar procedimientos en un disquete, no te olvides que tiene que estar previamente formateado.

```
? PONCL 2
? CIRCULO
? PONCL 3
? CIRCULO
```

Ahora definiremos dos procedimientos: el primero dibuja un cuadrado y nos informa de las coordenadas de cada uno de sus vértices; el segundo tomará esta información para ir trazando líneas desde el centro hasta los vértices del cuadrado.

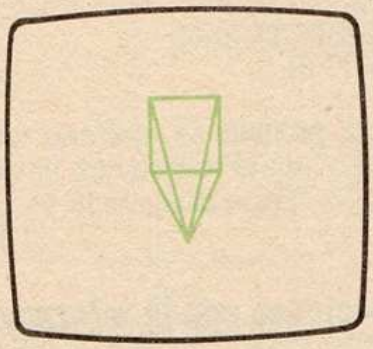


Fig. 10.

```
? PARA CUADRADO
> BP BL
> PONCL 2
> PONPOS [-15 60]
> REPITE 4 [AV 30 GD 90 ESCRIBE POS]
> FIN
```

Ejecuta el procedimiento:

```
? CUADRO
```

Anota los números que aparecen en la pantalla y haz el segundo procedimiento:

```
? PARA LINEAS
> CENTRO
> PONPOS [-15 90]
> CENTRO
> PONPOS [15 90]
> CENTRO
> PONPOS [15 60]
> FIN
```

Llama a los dos procedimientos:

```
? CUADRO
? LINEAS
```

Guardar y cargar procedimientos

Ya hemos mencionado que los procedimientos que realizamos los podemos guardar en un soporte de información externo para así tenerlos a nuestra disposición siempre que queramos con sólo cargarlos en memoria.

La forma, tanto de guardar como de cargar procedimientos, varía según el ordenador con el que estemos trabajando, ya que cada uno posee un soporte determinado para almacenar información.

Guardar procedimientos

La orden general que nos permite guardar procedimientos es:

GUARDA

que, dependiendo tanto de la versión de Logo con la que trabajemos como del ordenador, se tendrá que dar de una forma determinada:

— Para los PC-Compatibles:

GUARDA "nombre de fichero

Se guardarán en el disquete todos los procedimientos que se encuentran en el área de trabajo de la memoria del ordenador, bajo el nombre del fichero que se ha especificado.

Existen versiones del Logo para estos ordenadores en los que hay que especificar obligatoriamente los procedimientos que se desean guardar. En estos casos la orden habrá que darla de la siguiente manera:

GUARDA "nombre de fichero [lista de procedimientos]

Se guardan en el fichero especificado únicamente los procedimientos que aparecen en la lista.

— Para los MSX:

GUARDA "nombre de fichero

Se guardan todos los procedimientos que hay en el área de trabajo con el nombre de fichero que se ha especificado.

En estos dos ordenadores, una vez que se ha terminado de realizar este proceso, aparece el siguiente mensaje:

n procedimiento(s) guardado(s)

Da nombre tanto a los procedimientos como a los ficheros donde los guardes que tenga relación con lo que hacen. Así será más fácil distinguirlos.

donde n nos indica el número de procedimientos que se han guardado.

También en estos dos ordenadores el nombre del fichero no puede exceder de ocho caracteres.

– Para el SPECTRUM:

GUARDA "nombre de fichero [lista procedimientos]

Se guardan los procedimientos que aparecen en lista con el nombre del fichero que se ha especificado.

El nombre de fichero no puede exceder de siete caracteres.

En todos los casos existen opciones para guardar los procedimientos de una forma u otra, en un soporte o en otro, etc. Ya iremos viendo todo esto más despacio.

Carga de procedimientos

La carga de procedimientos es el proceso contrario al de guardar. En este caso se transfieren a la memoria del ordenador todos los procedimientos que hay en un fichero determinado desde un soporte externo.

La orden que nos permite este proceso es:

CARGA "nombre de fichero

El nombre del fichero ha de ser el que corresponda a un procedimiento que anteriormente hayamos guardado con el mismo nombre.

A medida que los procedimientos se van cargando en la memoria, van apareciendo sus nombres en la pantalla.

Cuadro resumen

– COORX

Informa en qué posición (coordenada X) se encuentra la Tortuga en ese momento.

– COORY

Informa en qué posición (coordenada Y) se encuentra la Tortuga en ese momento.

– POS

Devuelve el código de color que tiene en ese momento el lápiz de la Tortuga.

– FONDO

Devuelve las coordenadas (X e Y) de la posición de la Tortuga.

– CL

Devuelve qué código de color de fondo tiene la pantalla gráfica.

– LAPIZ

Nos informa sobre el estado del lápiz.

– PALETA

Nos dice el código de la paleta que estamos utilizando.

– RUMBO

Devuelve el rumbo que lleva la Tortuga.

– HACIA [X Y]

Nos dice qué rumbo debemos tomar para dirigirnos a un punto determinado, [X Y].

– PARA nombre

Señala que va a dar comienzo la definición o creación de un procedimiento determinado con el nombre que se ha especificado.

– FIN

Indica que se ha terminado la definición de un procedimiento.

– GUARDA "nombre de fichero
GUARDA "nombre de fichero [LISTA]

La primera forma almacena en el soporte externo los procedimientos que hay en memoria bajo el nombre que se especifique en "nombre.

La segunda forma guarda en el fichero que se especifica en "nombre de fichero" los procedimientos que se dan en la LISTA.

– CARGA "nombre de fichero

Carga el fichero especificado, en el área de trabajo de la memoria del ordenador. Este fichero puede contener uno o varios procedimientos.

Ejercicios

1. *Utilizando procedimientos, intenta dibujar este pequeño tren.*

Si tu versión de Logo no posee la orden pantalla mixta (PM), pasamos directamente a este modo de pantalla con cualquier orden que actúe sobre la Tortuga.

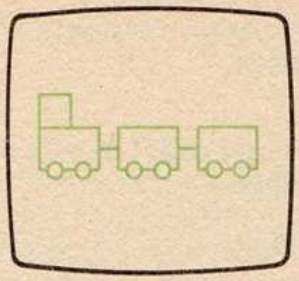


Fig. 11.

2. *Intenta hacer un procedimiento que dibuje una cámara de cine.*

3. *Este dibujo puede hacerse utilizando la orden PONRUMBO y un procedimiento; inténtalo.*

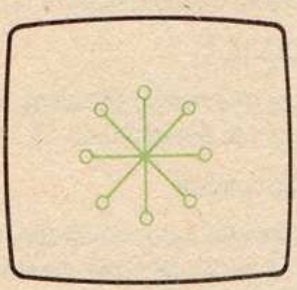


Fig. 12.

4. *Realiza el siguiente dibujo:*

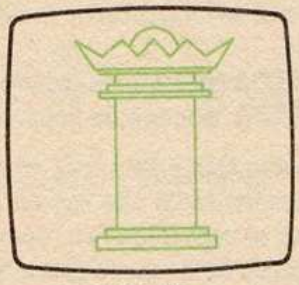


Fig. 13.

5. *Son correctas las siguientes órdenes:*
   - AVANZA CL
   - PONRUMBO HACIA [30 50]
   - PARA NOMBRE
     > FIN
   - AVANZA PONRUMBO
   - ESCRIBE HACIA (X Y)

Solución a los ejercicios

1: Utilizamos dos procedimientos:

Uno dibuja la cabina del vagón.

```
? PARA CABINA
> BL
> AV 50 GD 90
> AV 25 GD 90
> AV 20 GI 180
> FIN
```

y el otro dibuja el vagón

```
? PARA VAGON
> BL
> REPITE 2 [AV 30 GD 90 AV 70 GD 90]
> GD 90 AV 5 GI 90
> REPITE 36 [AV 2 GD 10]
> GD 90 AV 35 GI 90
> REPITE 36 [AV 2 GD 90]
> FIN
```

El cuerpo de órdenes que dibujan el tren las damos directamente.

```
? BP
? SL
? PONPOS [-100 0]
? CABINA
? SL PONPOS [-100 0]
? VAGON
? SL PONPOS [-30 10]
? BL GD 90 AV 20 GI 90
? SL PONPOS [-10 0]
? VAGON
? SL PONPOS [60 10]
? BL GD 90 AV 20 GI 90
? SL PONPOS [80 0]
? VAGON
```

2: Definimos el procedimiento CAMARA:

```
? PARA CAMARA
> BP
> SL
> PONPOS [-20 30]
> BL REPITE 2 [AV 30 GD 90 AV 60 GD
  90]
> AV 9 GI 90 AV 12
> SL PONPOS [-20 50]
> BL PONRUMBO 270 AV 12
> REPITE 36 [AV 1 GI 10]
> SL PONPOS [10 30]
> BL PONRUMBO 160
> AV 90
> SL PONPOS [10 30]
> BL PONRUMBO 200
> AV 90
> PONRUMBO 0
> SL PONPOS [10 35]
> BL REPITE 4 [AV 20 GD 90]
> FIN
```

Podemos utilizar como dato los valores que nos devuelven algunas órdenes.

Una vez tecleado el procedimiento lo ejecutamos:

? CAMARA

y obtendremos este dibujo:

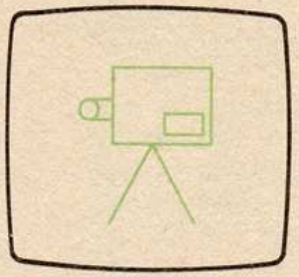


Fig. 14.

3: *Con un procedimiento dibujamos un brazo del dibujo.*

```
? PARA BRAZO
> AV 50 GI 90
> REPITE 36 [AV 2 GD 10]
> CENTRO
> FIN
```

El conjunto lo seguimos así:

```
? BP
? BRAZO
? PONRUMBO 45
? BRAZO
? PONRUMBO 90
? BRAZO
? PONRUMBO 135
? BRAZO
? PONRUMBO 180
? BRAZO
? PONRUMBO 225
? BRAZO
? PONRUMBO 270
? BRAZO
? PONRUMBO 315
? BRAZO
```

4:

```
? PARA REINA
INICIALIZACION
> PM
> BP
> OT
> SL
CENTRANDO DIBUJO
> GI 90 AV 35
> GD 90 RE 60
> BL
DIBUJO
> REPITE 2 [AV 10 GD 90 AV 70 GD 90]
> GI 90 RE 5 GD 90
> SL AV 10 BL
> AV 5 GD 90 AV 60
> GD 90 AV 5 RE 5
> GD 90 AV 15 GD 90
> AV 65 GD 90 AV 5
> GI 180 AV 40 GD 90
> AV 1 GI 90 RE 40
> AV 2 GD 90 AV 5
> GI 90 AV 35 RE 33
> GD 90 AV 3 GI 90
> RE 3 AV 38 GD 90
> AV 1 GD 90 AV 38
> SL CENTRO
> RE 45 GI 90 AV 15
> GD 90 BL AV 66
> GI 90 AV 3 GD 90
> AV 5 GD 90 AV 2

CORONA
> GI 90 AV 3 GI 30
> AV 10 GD 140 AV 10
> GI 110
> REPITE 3 [GD 45 AV 6 GD 90 AV 6 GI
  135]
> GD 60 AV 8 GD 140
> AV 10
> SL CENTRO
> AV 30 GI 45 AV 7
> GD 45 BL
> REPITE 6 [AV 4 GD 36]
> FIN
```

5:

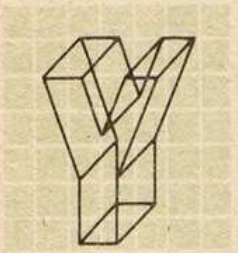
- AVANZA CL: Correcta
- PONRUMBO HACIA [30 50]: Correcta
- PARA NOMBRE
  > FIN

Correcto

- AVANZA PONRUMBO: Incorrecto, PONRUMBO es una orden que no devuelve ningún valor que sirva de dato a AVANZA.
- ESCRIBE HACIA (X Y): Incorrecta, las coordenadas X e Y van entre corchetes.

Dentro de un procedimiento puedo llamar a otro distinto.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS



Ya vimos en el tomo anterior cómo mover horizontalmente objetos o figuras compuestos de más de un carácter. En este tomo veremos cómo mover dichas figuras verticalmente. Para ello primero vamos a definirnos una serie de figuras que serán las que utilizaremos para realizar dichos movimientos.

La primera figura que vamos a dibujar va a ser un coche. Este coche tiene la forma que aparece en la figura 1.

Como ya vimos en el tomo anterior, los códigos ASCII de los caracteres semigráficos dependen mucho del ordenador que estemos usando. Por ello, en la figura 2 os damos los caracteres que componen dicho coche carácter a carácter.

El resto de los dibujos o figuras que utilizaremos para explicar estos tipos de movimiento los veremos más adelante.

Movimiento vertical de figuras complejas

Este tipo de movimiento es muy parecido al que ya vimos en tomos anteriores al tratar del movimiento de las figuras compuestas por un solo carácter.

La única diferencia que hay al realizar este movimiento con figuras complejas, en vez de con figuras sencillas compuestas por un solo carácter, es que hay que tener en cuenta que una figura compleja está compuesta de más de una columna de caracteres.

Un ejemplo de este tipo de movimiento lo podemos ver en el programa 1.

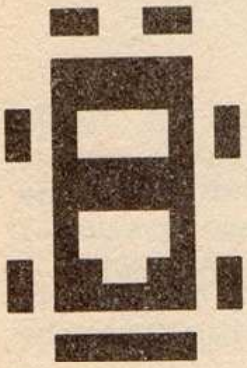


Fig. 1. Este es el coche que se mueve por la pantalla gracias al programa 1. Los caracteres de que está compuesto aparecen en la figura 2.

CARACTER	SPECTRUM	COMMODORE	AMSTRAD	IBM	MSX
	143	18 + 32 + 146	143	219	143
	131	18 + 162 + 146	131	220	131
	140	162	140	223	140
	138	161	133	221	133
	133	18 + 161 + 146	138	222	138

Fig. 2. Estos son los cinco caracteres que componen el coche de la figura 1 y los códigos ASCII según el ordenador.

```
 10 REM ************************************
 20 REM * MOVIMIENTO DE UN COCHE POR LA *
 30 REM * CARRETERA DE ARRIBA A ABAJO   *
 40 REM ************************************
 50 REM
 60 CLS
 70 REM
 80 REM *** DEFINICION DEL COCHE ***
 90 REM
100 LET A$=" "+CHR$(220)+" "+CHR$(220)
110 LET B$=" "+CHR$(219)+CHR$(219)+CHR$(219)+" "
120 LET C$=CHR$(221)+CHR$(221)+" "+CHR$(222)+CHR$(222)
130 LET D$=B$
140 LET E$=" "+CHR$(221)+" "+CHR$(222)+" "
150 LET F$=CHR$(221)+CHR$(219)+CHR$(223)+CHR$(219)+CHR$(222)
160 LET G$=" "+CHR$(223)+CHR$(223)+CHR$(223)
170 REM
180 REM *** MOVIMIENTO DEL COCHE ***
190 REM
200 LET X=10
210 FOR I=1 TO 15
220    LOCATE I,X
230    PRINT "     "
240    LOCATE I+1,X
250    PRINT A$
260    LOCATE I+2,X
270    PRINT B$
280    LOCATE I+3,X
290    PRINT C$
300    LOCATE I+4,X
310    PRINT D$
320    LOCATE I+5,X
330    PRINT E$
340    LOCATE I+6,X
350    PRINT F$
360    LOCATE I+7,X
370    PRINT G$
380    FOR J=1 TO 100
390    NEXT J
400 NEXT I
410 END
```

Programa 1.

Las modificaciones que son necesarias hacerle al programa para que funcione en ordenadores distintos del IBM o compatibles son:

COMMODORE:

```
60 PRINT "<SHIFT-HOME>"
100 LET A$=" "+CHR$(18)+CHR$(162)+CHR$(146)" "
+CHR$(18)+CHR$(162)+CHR$(146)
110 LET B$=" "+CHR$(18)+CHR$(32)+CHR(32)+CHR$(32)
+CHR$(146)
120 LET C$=CHR(161)+CHR$(161)+" +CHR$(18)+CHR$(161)
+CHR$(161)+CHR$(146)
140 LET E$=" "+CHR$(161)+" "+CHR$(18)+CHR$(161)+CHR$(146)
150 LET F$=CHR$(161)+CHR$(13)+" "+CHR$(146) +CHR$(162)
+CHR$(13)+" +CHR$(161)+CHR$(146)
160 LET G$=" "+CHR(162)+CHR$(162)+CHR$(162)
220 LET Y=I:GOSUB 9500
240 LET Y=Y+1:GOSUB 9500
260 LET Y=Y+1:GOSUB 9500
280 LET Y=Y+1:GOSUB 9500
300 LET Y=Y+1:GOSUB 9500
320 LET Y=Y+1:GOSUB 9500
340 LET Y=Y+1:GOSUB 9500
360 LET X=X+1:GOSUB 9500
```

También hay que unir el programa con la rutina LOCATE PARA COMMODORE que se dio en los tomos primero y quinto.

AMSTRAD

```
100 LET A$=" "+CHR$(131)+" "+CHR$(131)
110 LET B$=" "+CHR$(143)+CHR$(143)+CHR$(143)+" "
120 LET C$=CHR$(133)+CHR$(133)+" "+CHR$(138)+CHR$(138)
```

```
140 LET E$=" "+CHR$(133)+" "+CHR$(138)+" "
150 LET F$=CHR$(133)+CHR$(143) +CHR$(140)+CHR$(143)
+CHR$(138)
160 LET G$=" "+CHR(140+CHR$(140)+CHR$(140)
```

MSX:

Hay que cambiar las líneas 100, 110, 120, 140, 150 y 160 y ponerlas como las modificaciones que se han dado más arriba para el AMSTRAD. Las demás líneas que hay que variar son las siguientes:

```
220 LOCATE X,I
240 LOCATE X,I+1
260 LOCATE X,I+2
280 LOCATE X,I+3
300 LOCATE X,I+4
320 LOCATE X,I+5
340 LOCATE X,I+6
360 LOCATE X,I+7
```

SPECTRUM

```
100 LET A$=" "+CHR$(131)+" "+CHR$(131)
110 LET B$=" "+CHR$(143)+CHR$(143)+CHR$(143)+" "
120 LET C$=CHR$(138)+CHR$(138)+" "+CHR$(133)+CHR$(133)
140 LET E$=" "+CHR$(138)+" "+CHR$(133)+" "
150 LET F$=CHR$(138)+CHR$(143) +CHR$(140)+CHR$(143)
+CHR$(133)
160 LET G$=" "+CHR$140+CHR$(140)+CHR$(140)
220 PRINT AT I,X;
240 PRINT AT I+1,X;
260 PRINT AT I+2,X;
280 PRINT AT I+3,X;
300 PRINT AT I+4,X;
320 PRINT AT I+5,X;
340 PRINT AT I+6,X;
360 PRINT AT I+7,X;
410 STOP
```

El funcionamiento del programa es muy sencillo. Entre las líneas 100 y 160 se define el coche que vamos a hacer que se desplace por la pantalla. Dicho coche está compuesto de siete líneas con cinco caracteres cada línea. En la línea 130 se asigna a la variable alfanumérica D$ el valor de B$; porque, si te fijas, la línea segunda y la cuarta del dibujo del coche son iguales.

Entre las líneas 210 y 400 se realiza el movimiento. Para ello, lo primero que hacemos es imprimir una línea en blanco en la posición donde se va a colocar el coche. Luego se van imprimiendo, línea a línea, todos los caracteres que forman el coche. Tras una pequeña pausa, realizada con el bucle de retardo que se encuentra entre las líneas 380 y 390 del listado, se imprime el coche una línea más abajo de la que se encontraba borrando la primera línea de la antigua posición.

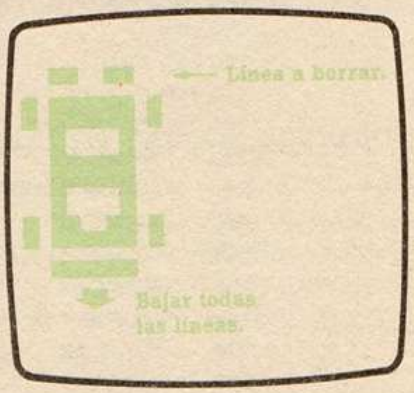


Fig. 3. En el programa 1, para dar la sensación de movimiento, tenemos que ir borrando la fila superior del dibujo.

Esta forma de realizar el programa no es muy bonita ni muy rápida, pero nos enseña muy bien cómo realizar un movimiento vertical. Una vez entendido cómo lo hemos hecho, podemos mejorarlo para que sea menos largo y más rápido. El resultado es el que aparece en el programa 2.

```
10 REM ************************************
20 REM * MOVIMIENTO DE UN COCHE POR LA *
30 REM * CARRETERA DE ARRIBA A ABAJO   *
40 REM ************************************
50 REM
60 CLS
70 REM
80 REM *** DEFINICION DEL COCHE ***
90 REM
100 DIM A$(8)
110 LET A$(1)="     "
120 LET A$(2)=" "+CHR$(220)+" "+CHR$(220)
130 LET A$(3)=" "+CHR$(219)+CHR$(219)+CHR$(219)+" "
140 LET A$(4)=CHR$(221)+CHR$(221)+" "+CHR$(222)+CHR$(222)
150 LET A$(5)=A$(3)
```

```
160 LET A$(6)=" "+CHR$(221)+" "+CHR$(222)+" "
170 LET A$(7)=CHR$(221)+CHR$(219)+CHR$(223)+CHR$(219)+CHR$(222)
180 LET A$(8)=" "+CHR$(223)+CHR$(223)+CHR$(223)
190 REM
200 REM *** MOVIMIENTO DEL COCHE ***
210 REM
220 LET X=10
230 FOR I=1 TO 15
240    FOR J=1 TO 8
260       LOCATE I+J,X
270       PRINT A$(J)
280    NEXT J
290    FOR J=1 TO 100
300    NEXT J
310 NEXT I
320 END
```

Programa 2.

Aunque las modificaciones que hay que hacer son casi iguales a las del programa anterior, éstas aparecen a continuación para todos los ordenadores que no sean IBM o compatibles.

COMMODORE

```
60 PRINT "<SHIFT-HOME>"
120 LET A$(2)=" "+CHR$(18)+CHR$(162)+CHR$(146)" "
+CHR$(18)+CHR$(162)+CHR$(146)
130 LET A$(3)=" "+CHR$(18)+CHR$(32) +CHR(32)+CHR$(32)
+CHR$(146)
140 LET A$(4)=CHR$(161)+CHR$(161)+" "+CHR$(18)+CHR$(161)
+CHR$(161)+CHR$(146)
160 LET A$(6)=" "+CHR$(161)+" "+CHR$(18)+CHR$(161)
+CHR$(146)
170 LET A$(7)=CHR$(161)+CHR$(13)+" "+CHR$(146)+CHR$(162)
+CHR$(13)+" "+CHR$(161)+CHR$(146)
180 LET A$(8)=" "+CHR$(162)+CHR$8162)+CHR$(162)
260 LET Y=I+J:GOSUB 9500
```

También hay que unir el programa con la rutina LOCATE PARA COMMODORE que se dio en los tomos primero y quinto.

AMSTRAD

```
120 LET A$(2)=" "+CHR$(131)+" "+CHR$(131)
130 LET A$(3)=" "+CHR$(143)+CHR$(143) +CHR$(143)+" "
140 LET A$(4)=CHR$(133)+CHR$(133)+" "+CHR$(138)
+CHR$(138)
160 LET A$(6)=" "+CHR$(133)+" "+CHR$(138)+" "
170 LET A$(7)=CHR$(133)+CHR$(143) +CHR$(140)+CHR$(143)
+CHR$(138)
180 LET A$(8)=" "+CHR$(140+CHR$(140)+CHR$(140)
```

MSX:

Hay que cambiar las líneas 120, 130, 140, 160, 170 y 180 y ponerlas como las modificaciones que se han dado más arriba para el AMSTRAD. La única línea que queda por variar es:

```
260 LOCATE X,I+J
```

SPECTRUM

```
100 DIM A$(8,5)
120 LET A$(2)=" "+CHR$(131)+" "+CHR$(131)
130 LET A$(3)=" "+CHR$(143)+CHR$(143) +CHR$(143)+" "
140 LET A$(4)=CHR$(138)+CHR$(138)+" "+CHR$(133)+CHR$(133)
160 LET A$(6)=" "+CHR$(138)+" "+CHR$(133)+" "
170 LET A$(7)=CHR$(138)+CHR$(143) +CHR$(140)+CHR$(143)
+CHR$(133)
180 LET A$(8)=" "+CHR$140+CHR$(140)+CHR$(140)
260 PRINT AT I+Y,X;
320 STOP
```

El funcionamiento, aunque distinto, no tiene ningún problema de comprensión. En vez de almacenar cada línea del coche en una variable, se almacenan en un vector (matriz de una sola dimensión y que en este caso, además, es alfanumérica) de ocho elementos. De esta manera es mucho más fácil imprimir el coche en la pantalla a la hora de realizar el movimiento.

Entre las líneas 230 y 310 hacemos que el coche se mueva de arriba a abajo por la pantalla. Como cada línea de la definición del coche está en uno de los elementos del vector A$, para imprimir unas líneas debajo de otras no hace falta utilizar varias sentencias LOCATE. Como podemos apuntar, mediante una variable índice, a cada uno de los elementos del vector, lo realizamos variando el valor de la variable numérica J mediante el bucle de la lí-

nea 240. Gracias a este bucle todo queda reducido a una sola sentencia de localización del cursor y a una sola sentencia PRINT.

Para realizar un movimiento que sea de abajo hacia arriba, en vez de arriba hacia abajo, se utiliza el mismo principio que cuando se trata de un solo carácter el que estamos moviendo. Como ejemplo, tenemos el programa 3, que mueve el mismo coche, pero en la dirección contraria. Como la orientación del movimiento ha cambiado, también tiene que cambiar la forma del coche, pero si te fijas en el listado, el coche está definido de la misma manera que antes. ¿Entiendes por qué?

```
10 REM ************************************
20 REM * MOVIMIENTO DE UN COCHE POR LA *
30 REM * PANTALLA DE ABAJO A ARRIBA    *
40 REM ************************************
50 REM
60 CLS
70 REM
80 REM *** DEFINICION DEL COCHE ***
90 REM
100 DIM A$(8)
110 LET A$(1)="      "
120 LET A$(2)=" "+CHR$(220)+" "+CHR$(220)
130 LET A$(3)=" "+CHR$(219)+CHR$(219)+CHR$(219)+" "
140 LET A$(4)=CHR$(221)+CHR$(221)+" "+CHR$(222)+CHR$(222)
150 LET A$(5)=A$(3)
160 LET A$(6)=" "+CHR$(221)+" "+CHR$(222)+" "
170 LET A$(7)=CHR$(221)+CHR$(219)+CHR$(223)+CHR$(219)+CHR$(222)
180 LET A$(8)=" "+CHR$(223)+CHR$(223)+CHR$(223)
190 REM
200 REM *** MOVIMIENTO DEL COCHE ***
210 REM
220 LET X=10
230 FOR I=13 TO 1 STEP -1
240    FOR J=8 TO 1 STEP -1
260       LOCATE I+J,X
270       PRINT A$(9-J)
280    NEXT J
290    FOR J=1 TO 100
300    NEXT J
310 NEXT I
320 END
```

Programa 3.

Las modificaciones que hay que realizar son exactamente las mismas que las que tuvimos que hacer para poder ejecutar el programa 2.

Lo único que hemos hecho para que el coche vaya en dirección contraria es cambiar los bucles de las líneas 230 y 240. El primer bucle, que nos indica la línea de la pantalla sobre la que tenemos que imprimir el coche, va contando desde 13 hasta 1. Esto es, va variando la línea donde se imprimirá el coche desde la línea número 13 hasta la número 1. Como la línea número 13 está más abajo que la número 1, el movimiento será ascendente.

En el caso del segundo bucle, el de la línea 240, lo que hace es imprimir el coche, línea a línea, pero, gracias a la línea 270, lo imprime al contrario, con el morro para atrás, con lo que se consigue que el coche haya cambiado de orientación.

Por fin, y casi terminado el movimiento vertical de figuras compuestas por más de un carácter, vamos a ver el programa 4. Este programa mueve el coche de abajo hacia arriba

y, cuando éste llega hasta el borde de la pantalla, se empieza a mover de arriba hacia abajo. Este movimiento continúa hasta que paramos el programa. Como podrás apreciar en el programa, la definición del coche sólo está una vez.

```
 10 REM ****************************************
 20 REM * MOVIMIENTO DE UN COCHE DE ARRIBA A *
 30 REM * A ABAJO Y DE ABAJO A ARRIBA        *
 40 REM ****************************************
 50 REM
 60 CLS
 70 REM
 80 REM *** DEFINICION DEL COCHE ***
 90 REM
100 DIM A$(8)
110 LET A$(1)="      "
120 LET A$(2)=" "+CHR$(220)+" "+CHR$(220)
130 LET A$(3)=" "+CHR$(219)+CHR$(219)+CHR$(219)+" "
140 LET A$(4)=CHR$(221)+CHR$(221)+" "+CHR$(222)+CHR$(222)
150 LET A$(5)=A$(3)
160 LET A$(6)=" "+CHR$(221)+" "+CHR$(222)+" "
170 LET A$(7)=CHR$(221)+CHR$(219)+CHR$(223)+CHR$(219)+CHR$(222)
180 LET A$(8)=" "+CHR$(223)+CHR$(223)+CHR$(223)
190 REM
200 REM *** MOVIMIENTO DEL COCHE ***
210 REM
220 LET X=10
230 LET ST=1
240 LET D1=1:LET D2=13
245 LET D3=1:LET D4=8
250 FOR I=D1 TO D2 STEP ST
260    FOR J=D3 TO D4 STEP ST
270       LOCATE I+J,X
280       LET JJ=J
290       IF ST=-1 THEN LET JJ=9-J
300       PRINT A$(JJ)
310    NEXT J
320    FOR J=1 TO 100
330    NEXT J
340    LET ST=-ST
350    LET D5=D1:LET D1=D2:LET D2=D5
360    LET D5=D3:LET D3=D4:LET D4=D5
370 GOTO 250
```

Programa 4.

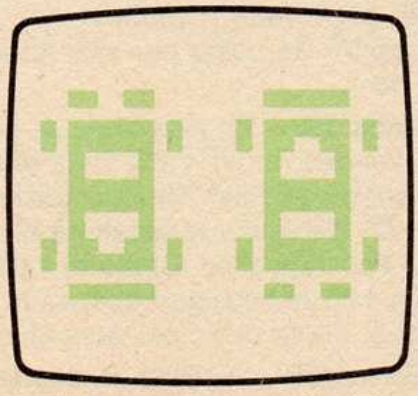


Fig. 4. Dibujo del coche en los dos sentidos del movimiento.

Las modificaciones que hay que realizar para utilizar este programa con ordenadores distintos del IBM son las siguientes:

COMMODORE:

```
160 PRINT "<SHIFT-HOME>"
120 LET A$(2)=" "+CHR$(18)+CHR$(162)+CHR$(146)" "
+CHR$(18)+CHR$(162)+CHR$(146)
130 LET A$(3)=" "+CHR$(18)+CHR$(32)+CHR(32) +CHR$(32)
+CHR$(146)
140 LET A$(4)=CHR$(161)+CHR$(161)+" "+CHR$(18)+CHR$(161)
+CHR$(161)+CHR$(146)
160 LET A$(6)=" "+CHR$(161)+" "+CHR$(18)+CHR$(161)
+CHR$(146)
```

```
170 LET A$(7)=CHR$(161)+CHR$(13)+" "+CHR$(146)+CHR$(162)
+CHR$(13)+" "+CHR$(161)+CHR$(146)
180 LET A$(8)=" "+CHR$(162)+CHR$(162)+CHR$(162)
270 Y=I+J: GOSUB 9500
```

También hay que unir el programa con la rutina LOCATE PARA COMMODORE que se dio en los tomos primero y quinto.

AMSTRAD

```
120 LET A$(2)=" "+CHR$(131)+" "+CHR$(131)
130 LET A$(3)=" "+CHR$(143)+CHR$(143)+CHR$(143)+" "
140 LET A$(4)=CHR$(133)+CHR$(133)+" "+CHR$(138)
+CHR$(138)
160 LET A$(6)=" "+CHR$(133)+" "+CHR$(138)+" "
170 LET A$(7)=CHR$(133)+CHR(143) +CHR$(140)+CHR$(143)
+CHR$(138)
180 LET A$(8)=" "+CHR$(140+CHR$(140)+CHR$(140)
```

MSX

Hay que cambiar las líneas 120, 130, 140, 160, 170 y 180 y ponerlas como las modificaciones que se han dado más arriba para el AMSTRAD. La única línea que queda por variar es:

```
260 LOCATE X,I+J
```

SPECTRUM

```
100 DIM A$(8,5)
120 LET A$(2)=" "+CHR$(131)+" "+CHR$(131)
130 LET A$(3)=" "+CHR$(143)+CHR$(143)+CHR$(143)+" "
140 LET A$(4)=CHR$(138)+CHR$(138)+" "+CHR$(133)
+CHR$(133)
160 LET A$(6)=" "+CHR$(138)+" "+CHR$(133)+" "
170 LET A$(7)=CHR$(138)+CHR$(143) +CHR$(140)+CHR$(143)
+CHR$(133)
180 LET A$(8)=" "+CHR$140+CHR$(140)+CHR$(140)
260 PRINT AT I+I,X;
```

Aunque estas modificaciones son las mismas que las del programa anterior, vuelven a aparecer aquí para que no tengas que estar mirando en hojas anteriores.

El funcionamiento del programa, a partir de la línea 220, línea a línea, es el siguiente:

Línea 220. Se le da a la variable X el valor 10. Esta será la columna donde empieza el primer carácter del coche y donde se desarrollará el movimiento.

Línea 230. Se asigna a **ST** el valor 1. Esta variable le dice al programa si el movimiento es hacia abajo, **ST=1,** o hacia arriba, **ST=-1.** Cada vez que el coche ha llegado a un extremo de la pantalla, esta variable cambia de signo.

Línea 240. Estas dos variables, **D1** y **D2,** Son las que dicen desde qué primera línea hasta qué última línea se va a desarrollar el movimiento. Estas variables intercambiarán su valor cada vez que el coche llegue a un extremo de la pantalla.

Línea 245. Las variables **D3** y **D4** nos servirán para indicarle al programa la dirección que ha de tener la figura del coche cuando la dibujemos. Varían de la misma forma que D1 y D2.

Línea 250. Aquí comienza el bucle que llevará el coche desde la línea D1 hasta la línea D2 con incremento ST. Cuando ST sea igual a 1, y, por tanto, D1 menor que D2, el movimiento será de arriba a bajo. Cuando ST sea igual a -1, y, por tanto, D2 menor que D1, el movimiento será de abajo hacia arriba.

Línea 260. Comienza el bucle que nos permitirá imprimir el coche mirando hacia arriba o hacia abajo.

Línea 270. Se coloca el cursor en la fila y la columna donde se debe de imprimir la fila que corresponda imprimir en ese momento.

Línea 280. Se asigna a la variable auxiliar **JJ** el valor de J. Esta variable servirá para poder imprimir el coche en cualquiera de las dos direcciones posibles.

Línea 290. Si el movimiento es hacia arriba (ST=-1), entonces se da a JJ el valor de **9-J** para que se imprima el coche en la dirección contraria a la habitual.

Línea 300. Se imprime la línea que corresponda de la definición del coche.

Línea 310. Aquí se termina el segundo bucle.

Líneas 320 y 330. Se realiza un tiempo de retardo gracias a la utilización de un bucle en vacío. Aunque este bucle utiliza como variable índice la J esto no influye en el desarrollo normal del programa, ya que ésta se utiliza después de haber terminado un bucle y antes de empezar otro.

Línea 340. Se cambia de signo el valor de ST. Si ST valía 1, ahora valdrá -1. Si valía -1, después de pasar por esta línea tendrá el valor de 1.

Línea 350. En esta línea se intercambian los valores de D1 y D2. Para ello utilizamos la variable auxiliar D5. Esta almacenará el valor de D1. Después se asignará el valor de D2 a D2. Por último, se asigna el valor de D5 a D2. Con esto se consigue que, en la siguiente vuelta, el bucle principal vaya al contrario.

Línea 360. Se hace la misma operación que en la línea anterior. Esta vez se rea-

liza con las variables D3 y D4, que afectan al segundo bucle. La forma de realizarlo es igual. Se utiliza la variable auxiliar D5 para almacenar momentáneamente el valor de D3. A continuación se pasa el valor que tenía D4 a D3 y se le da a D4 el antiguo valor de D3 y que ahora está almacenado en D5.

Línea 370. Se hace un salto incondicional a la línea 250 para que vuelva a empezar de nuevo el bucle principal, pero ahora con los valores al contrario para que el movimiento se invierta. Debido a que el programa no tiene ninguna instrucción que le diga al ordenador que se pare, para terminar el programa habrá que pulsar la tecla BREAK o STOP, según el ordenador de que se trate.

Para terminar, te propongo un programa que veremos resuelto en el próximo tomo.

Con los dibujos que puedes ver en la figura 5 puedes realizar el movimiento de un hombre de arriba a abajo (o de abajo a arriba) de la pantalla. Lo único que tienes que hacer es fijarte en lo que se ha dicho en este tomo, haber ejecutado y estudiado los programas que te hemos dado y recordar lo poco que se dijo en el tomo anterior sobre la animación. También es importante que recuerdes cómo se definen los caracteres del usuario (caracteres definibles por el usuario) y que ya se dijo cómo se hacía en uno de los primeros tomos.

Si no te sale el programa, no te preocupes. Nosotros publicaremos la solución y su explicación en el próximo tomo.

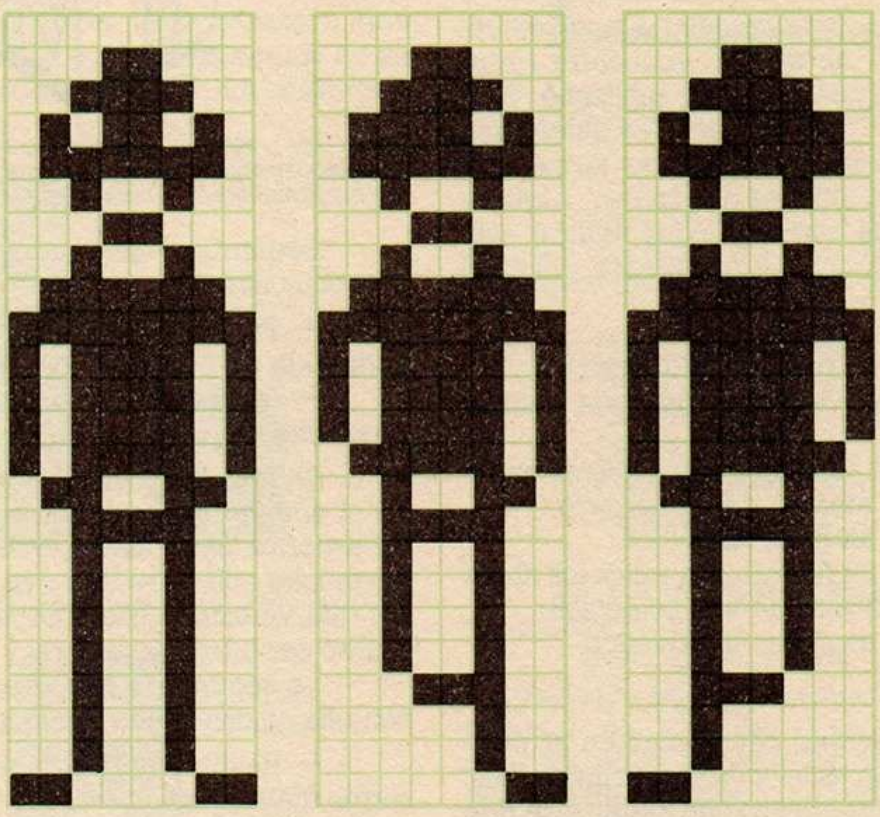
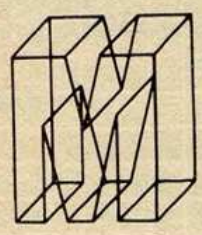


Fig. 5. Definición de las tres posiciones del movimiento de un hombrecillo.

Trucos de programación



MUCHAS de las personas que se compran un ordenador no lo compran sólo para jugar a matar marcianos o para rescatar a una princesa que se encuentra presa de un ogro malísimo. Este tipo de personas quiere que el ordenador le resuelva algunas tareas que sin él serían más difíciles de realizar. Algunas de las tareas que quieren poder realizar este tipo de usuarios con un ordenador personal pueden ser:

— Llevar la contabilidad casera.
— Calcular la media de cada alumno de la clase.
— Dibujar una serie de gráficas explicativas sobre la marcha de un negocio.
— Tener un listín telefónico.
— Tener un fichero con todos los libros o con todos los discos de una colección.
— Tener un diario informático.
— Etc.

Existen muchos programas en el mercado que realizan estas y otras muchas tareas. Los principales problemas que tienen este tipo de programas, llamados en conserva, son los siguientes:

— Hay tareas que nosotros necesitamos que realicen y que no hacen.
— Hacen cosas que no nos interesa para nada y que nos hace perder tiempo.
— Si son baratos, no suelen ser muy buenos.
— Si son buenos, suelen ser muy caros.
— No se puede mandar información de un programa a otro, pues la forma de almacenar los datos no es estándar y los programas que permiten hacerlo son carísimos.

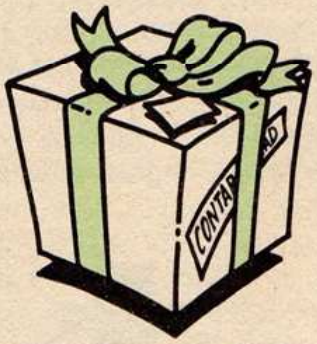


Fig. 1. Los programas verticales que se venden no suelen ser muy buenos porque no contemplan los deseos del usuario. Aquellos programas que sí lo hacían alcanzar precios exhorbitantes.

Según esto, sólo nos quedan dos posibles soluciones:

— Pedir que nos hagan un programa a la medida de nuestras necesidades.
— Hacernos nosotros mismos dicho programa.

La primera solución está muy difundida en el ámbito de las empresas, pero no en el de usuarios particulares, ya que suele resultar mucho más caro que comprar un programa que ya está hecho, aunque los resultados suelen ser bastante mejores.

La segunda solución es la mejor, pues con ella se consiguen al menos cuatro cosas:

— Realizar el programa tal y como nosotros queremos.
— Poder modificarlo a nuestro gusto en cualquier momento.
— Aprender algún lenguaje de programación y algo de informática.
— Poder hacer que dos o más programas se comuniquen entre sí compartiendo los datos que cada uno tiene almacenados.

De todos los programas que un usuario puede construir para sí mismo los más comunes son aquéllos que trabajan con ficheros, esto es, todos aquéllos que tienen que almacenar algún tipo de información, como los números de teléfono, direcciones, nombres, etc.

En este tomo, y en los siguientes, vamos

a ver qué es un fichero, cómo construirlo y cómo almacenar datos de formas muy diferentes, de manera que podamos ahorrar la mayor cantidad posible de memoria. Veremos cómo construirnos nuestro propio programa de manejo de ficheros.

Los ficheros

Antes de nada tenemos que explicar qué es un fichero. Qué es lo que se entiende, en el mundo de la Informática, por fichero.

La idea intuitiva de fichero que todos tenemos es la de un armario con cajones llenos de carpetas donde tenemos almacenados una gran cantidad de papeles sobre uno o varios temas. Esta idea no está muy lejos de la idea de un fichero informático. En el caso de un fichero informático, toda la información que queremos almacenar sobre algún tema no se guarda en un armario, sino en un diskette o cinta de cassette.

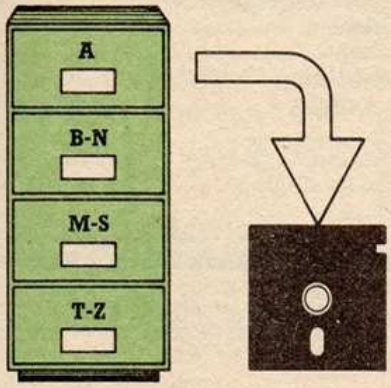


Fig. 2. Aunque la filosofía de un fichero no cambia, el volumen del mismo es mucho más reducido cuando está dentro de un disco.

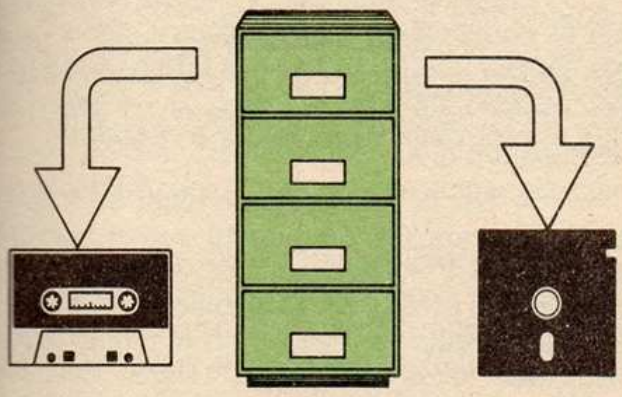


Fig. 3. Podemos realizar los ficheros en disco o en cinta. Da lo mismo, pero el disco, aparte de ser más rápido, da mucho mejor resultado.

Los papeles que componen el fichero suelen estar agrupados por nombres o por direcciones. Cada papel almacena los datos de un cierto disco, de una cierta persona, etc. A cada uno de estos papeles se le llama ficha.

La idea de ficha en Informática es la misma que la que acabamos de ver, pero teniendo en cuenta que no está escrita en un papel, sino en un trozo de la pista magnética del diskette o cinta donde se encuentra el fichero al que pertenece. En Informática, cada ficha también recibe el nombre de registro.

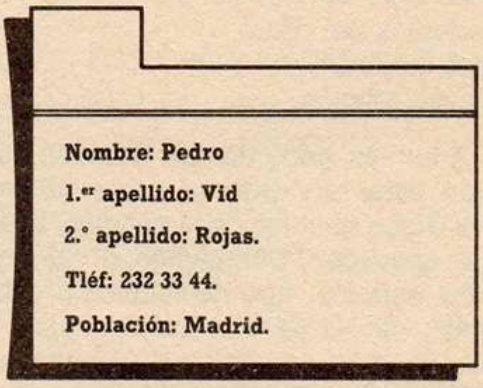


Fig. 4. Esta es la forma típica de una ficha. Dentro del diskette o cinta no se almacena como se ve aquí, pero la filosofía es la misma.

Cada ficha se compone de una serie de partes o de datos llamados campos. Estos, en un fichero de libros podrían ser:

- Nombre del Libro.
- Autor del libro.
- Editorial.
- Fecha de edición.
- Número de edición.
- Tipo de libro, etc.

En Informática la idea de campo es exactamente igual.

Cada campo puede, ocasionalmente, subdividirse en otros campos, llamados subcampos, que a su vez se pueden subdividir unos en otros de una forma indefinida. Un ejemplo de esto nos lo podemos encontrar en un fichero que contuviese todos los discos de un coleccionista de música. El fichero podría estar dividido en unos registros cuyo formato podría ser:

- Nombre del disco.
- Grupo:
  - Número de componentes del grupo.
  - Componente número 1 e instrumento.
  - Componente número 2 e instrumento.
  - Componente número n e instrumento.

  Cara A:
  - Número de canciones de la cara A.
  - Canción número 1.

- Canción número 2.
- Canción número n.

Cara B:

- Número de canciones de la cara B.
- Canción número 1.
- Canción número 2.
- Canción número n.

— Productor del disco.
— Casa editora.
— Año de edición.

A su vez, cada campo Canción número x puede estar dividido en otros subcampos que nos digan el autor de la canción, intérpretes que aparecen, puntuación o valorización de dicha canción, tipo de música o estilo de ésta, etc.

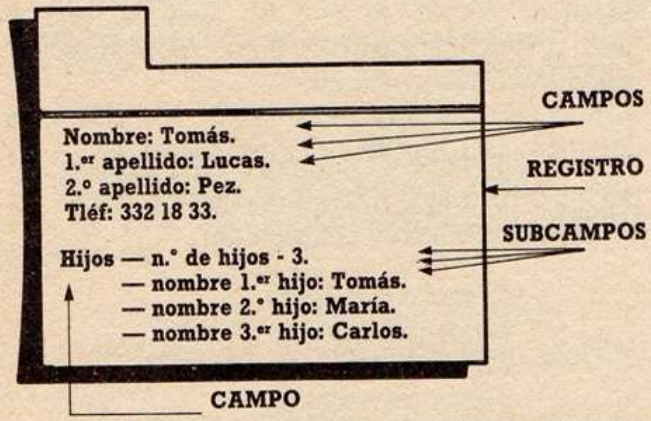


Fig. 5. En este dibujo se pueden apreciar más claramente todas las partes de un registro o ficha.

Como puede uno imaginarse, es más fácil realizar un fichero con un ordenador que tenerlo almacenado en un mueble. No sólo por la reducción de espacio que supondría realizarlo en un ordenador, sino porque es mucho más fácil la búsqueda de cualquiera de las fichas que componen dicho fichero con sólo saber uno de los campos que lo componen.

Por otro lado, el realizar un fichero con un ordenador nos permite realizar modificaciones de cualquier ficha en cualquier momento, búsqueda de todos los registros que cumplan una cierta condición, borrado de fichas, impresión de las mismas, pudiendo elegir el formato y el número de copias, etc.

Por todo esto, un programa gestor de ficheros tendría que poder realizar las siguientes funciones:

— Permitir la entrada de datos por el teclado.

— Ordenación alfabética por cualquier campo.

— Modificación de cualquier ficha en cualquier momento.

— Borrado de cualquier ficha en cualquier momento.

— Inserción de fichas nuevas en el fichero original.

— Visualización de todas o parte de las fichas dando el baremo sobre el que se basará la visualización o no visualización de una determinada ficha.

— Impresión de todas o parte de las fichas.

— Grabación del fichero cada vez que se termine una sesión de trabajo.

— Lectura de cualquier fichero creado con dicho programa.

— Posibilidad de unir dos ficheros distintos.

Fig. 6. Con los programas de gestión de ficheros, tenemos toda la información que necesitamos continuamente en pantalla.

El programa

Antes de comenzar a realizar el programa es conveniente que pensemos cómo va a ser. Se nos puede ocurrir dos formas esenciales, dependiendo de la forma de los ficheros que el programa va a crear y a gestionar.

— Los ficheros pueden tener un formato fijo e invariable, de manera que el número de campos y subcampos por registro viene dado por el programa. Esta solución, aun siendo más sencilla de realizar, no es buena, pues hace que todos los ficheros tengan que ser iguales, cosa que nunca nos va a pasar. Debido a esto, si realizamos el programa de esta manera, tendremos que variar el programa cada vez que queramos crear un fichero diferente.

— La otra solución es crear un programa que permita de paso definir el número de campos, el número de subcampos y el núme-

ro máximo de caracteres que puede utilizar cada campo. Esta solución, aun siendo más complicada, es la que nos conviene realizar para así no tener que preocuparnos por la forma de los ficheros.

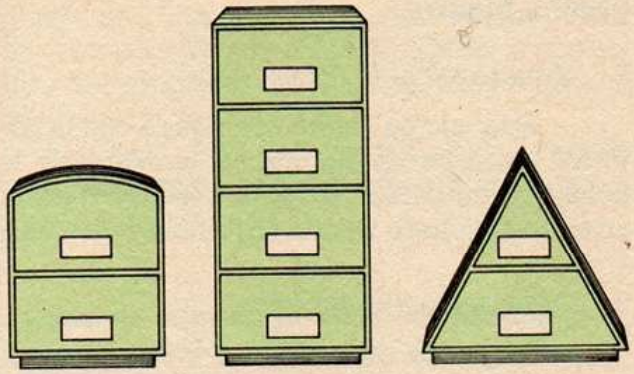


Fig. 7. Se permitimos que nuestro programa defina distintos tipos de ficheros, éste nos podrá servir para todo.

Una vez que sabemos cómo van a ser los ficheros (variables o invariables), tenemos que saber cómo vamos a almacenar todos los datos. También aquí se presentan dos soluciones:

— Tener los datos almacenados siempre en disco e ir a buscarlos sólo cuando los necesitemos. Esta solución es muy buena, porque nos permite tener toda la memoria del ordenador libre para el proceso interno de los registros. Por otro lado, esta solución no nos conviene, porque necesita que el usuario tenga disco y la capacidad de trabajar con ficheros aleatorios, caso que no se da en los usuarios de cinta de cassette.

— Tener almacenada continuamente en memoria toda la información del fichero y así trabajar siempre en memoria. Esto tiene de bueno que el proceso es más rápido por no tener que ir a buscar un registro al disco cada vez que lo necesitamos. De malo tiene que ocupa mucha más memoria, por lo que restringe el tamaño del fichero y hace el proceso más lento.

Como no todos los usuarios de ordenadores tienen una unidad de disco, nos decidimos por realizar un programa que almacene continuamente la información del fichero en memoria.

Una vez llegados a este punto sólo nos resta por decidir cómo almacenaremos en memoria, y en disco o cassette, cada registro. Como siempre, se nos aparecen al menos dos posibles soluciones:

— Realizar los registros por la unión de todos sus campos respetando la longitud de cada campo. Esto quiere decir que si, por ejemplo, tenemos un fichero cuyos registros se componen de:

- Número de teléfono con 10 caracteres como máximo.
- Nombre y apellidos del abonado con 30 caracteres.
- Dirección y ciudad con 30 caracteres.

un registro en memoria se encontraría así:

>91 8554454Antonio Miguel López González. Ppe. de Vergara, 12. Madrid.

y otro más corto:

>953 334422Pedro Pi Díaz.
Bañeza, 132. Alicante.

Como puede apreciarse, este tipo de almacenamiento ocupa mucha memoria, aunque tiene de bueno que siempre sabemos dónde empieza un campo. En este caso sabemos que el teléfono siempre empieza en el carácter número 1, el nombre en el carácter número 11 y la dirección en el número 41.

— El otro tipo de almacenamiento es mucho más compacto, pero nunca se sabe de antemano en qué posición comienza cada campo. En este tipo de almacenamiento en memoria se intercalan entre campo y campo una serie de caracteres, que el usuario no puede introducir, como delimitadores e indicadores de dónde comienza un campo cualquiera. Los dos registros, vistos anteriormente, pero almacenados de esta forma, presentarían el siguiente aspecto:

>91 8554454\Antonio Miguel López González\Ppe. de Vergara, 12. Madrid<
>953 334422\Pedro Pi Díaz\Bañeza, 132. Alicante<

Como puede apreciarse, la longitud de cada registro es notoriamente más corta.

El carácter actúa en este caso como delimitador y separador de los campos. Pueden utilizarse en vez de un solo delimitador, varios delimitadores, para así saber con más exactitud dónde empieza un cierto campo, sin necesidad de ir contando el número de delimitadores antes de llegar al que nos interesa.

Una vez llegados a este punto vamos a definir las restantes características del programa que vamos a realizar. Estas son:

1. *Definición del fichero*

El programa tiene que tener una opción para definir un nuevo tipo de fichero cada vez

Fig. 8. Podemos utilizar cualquier carácter que no pueda usar el usuario como delimitador. Detrás de éste puede ir un número para indicarnos el número campo en el que nos encontramos.

que se necesite. Esto incluye la capacidad de poder decir el número de campos por registro, el número de subcampos y el número de caracteres por campo. Después de definir un nuevo fichero, el ordenador grabará los datos que definen la estructura de dicho fichero en cinta o en disco.

2. *Introducción de fichas*

Se ha de permitir la introducción de fichas en cualquier momento. Las fichas nuevas se irán añadiendo al final del fichero para más comodidad y para no perder mucha velocidad en la ejecución.

3. *Modificación de fichas*

Se ha de prever la posible modificación de una ficha en cualquier momento. Dicha modificación no afectará al orden del fichero.

4. *Borrado de fichas*

En cualquier momento se podrá borrar una o un grupo de fichas. Estas no desaparecerán del fichero, sino que se les pondrá una marca que indicará que ya no existen.

5. *Recuperación de fichas*

Esta opción servirá para poder recuperar todas o algunas de las fichas que en un cierto momento borramos del fichero.

6. *Limpieza total del fichero*

Con esta opción se borran de verdad todas aquellas fichas que se anularon en cualquier momento. Esta opción se habrá de utilizar sólo cuando la memoria empiece a escasear.

7. *Búsqueda de fichas*

Se permitirá la búsqueda de fichas. Se permitirá ver las fichas con un cierto número de orden o que cumplan una determinada condición, como tener en el tercer campo la palabra **Madrid.**

8. *Visualización de fichas*

Esta opción estará conjugada con la anterior y nos permitirá, una vez encontrada la ficha que buscamos, ir hacia adelante o hacia atrás en el fichero viendo otras fichas distintas.

9. *Ordenación alfanumérica*

Se podrá ordenar el fichero en cualquier momento y por cualquier campo.

10. *Definición de salida de impresión*

Se permitirá al usuario que defina una forma de impresión. Dicha definición se almacenará junto con el fichero, pero puede alterarse en cualquier momento.

11. *Impresión instantánea*

El programa permitirá la impresión por impresora de cualquier ficha, tal y como aparece ésta representada en pantalla. Para ello, se proveerá al usuario de un comando que podrá utilizar desde la visualización de las fichas.

12. *Impresión de fichas*

Al elegir esta opción podrán imprimirse bien todas las fichas de que se compone el fichero, bien todas aquellas que cumplan una cierta condición. El formato de impresión será el que hayamos definido con anterioridad. Si dicho formato no ha sido definido, se avisará al usuario y se parará la impresión.

13. *Grabación de las fichas*

Se permitirá al usuario, en cualquier momento, la grabación de todas las fichas, o de aquéllas que cumplan una cierta condición, con el nombre que se desee. Al grabar el fichero también se grabará su forma y el último formato utilizado para impresión.

14. *Lectura de un fichero*

El programa, como es lógico, podrá leer cualquier fichero creado y grabado por el mismo. Al leer dicho fichero también se leerá su forma y el formato de impresión.

Con todo esto, podemos decir que ya está definida la forma que va a tener nuestro programa y las funciones que va a realizar. Ahora nos podemos dedicar a hallar las rutinas que vamos a necesitar para realizar este programa. Muchas de dichas rutinas se las puede uno imaginar sin pensar mucho. Otras es necesario pensar un poco más en la estructura del programa, para darse cuenta de que son necesarias.

Como primera aproximación, podríamos decir que algunas de las rutinas que vamos a necesitar son:

Rutina de entrada de datos de propósito general

Esta rutina se dio en el primer tomo, por lo que no la repetiremos hasta que tengamos el listado completo del programa.

Rutina de ordenación alfanumérica

Para ordenar el fichero. Tiene que estar capacitada para ordenar por cualquier campo.

Rutina de búsqueda

Para buscar cualquier ficha. Tiene que estar capacitada para buscar por el número de orden o por cualquier campo.

Rutina de creación de menús

Como el programa va a trabajar a base de menús es conveniente realizar una rutina que, dándole una serie de parámetros, nos imprima dichos menús en pantalla.

Rutina de impresión de fichas

Será una rutina mixta que sirva igual para imprimir en pantalla que para imprimir en la impresora.

Rutina de pulsa una tecla

Necesita para que la información no desaparezca de la pantalla. Se utilizará una de las rutinas que vimos en tomos anteriores.

Rutina de borrado de fichas

Se encargará de poner una máscara a todas las fichas que queramos borrar.

Rutina de limpieza del fichero

Nos permitirá borrar todas las fichas que tienen máscara por haber sido borradas.

Rutina de impresión formateada por impresora

Se encargará de imprimir todas o parte de las fichas, tal y como le hayamos dicho en el formato de impresión.

Rutina de grabación

Grabará el fichero.

Rutina de lectura

Leerá el fichero grabado y borrará, en caso necesario, el que estuviese en memoria.

Rutina de avance o retroceso

Será la encargada de ir hacia adelante o hacia detrás en el fichero cuando se está visualizando.

Rutina de compactación de datos

Se encargará de almacenar los campos de los registros y de poner los delimitadores o separadores.

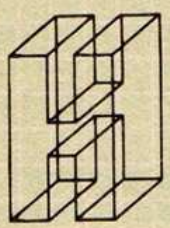
Rutina de descompactación de datos

Será la que, conociendo el número del campo, nos dé el valor de dicho campo.

Estas son las rutinas más importantes, aunque no las únicas con las que nos vamos a encontrar a la hora de realizar el programa. Las demás las iremos viendo según vaya avanzando el programa. Te doy éstas para que pienses sobre ellas e intentes encontrar alguna solución.

La razón por la que no aparece ningún programa en este tomo es que es necesario explicar primero qué es lo que vamos a hacer, y cómo, antes de empezar a programar. Por ello, empezaremos a ver las rutinas que componen este programa en el siguiente tomo.

Interfaz con señales analógicas



HASTA ahora hemos descrito dispositivos que suministran información de forma binaria o que necesitan solamente un bit para ser controlados. Por el contrario, el mundo físico que nos rodea presenta sus magnitudes de forma continuamente variable. Se dice en estos casos que las señales son de tipo analógico y con más propiedad cuando se miden de forma eléctrica magnitudes de otro tipo. Por ejemplo, estamos acostumbrados a ver la temperatura como desplazamiento de una columna de mercurio o de alcohol coloreado dentro de un tubo de vidrio. La temperatura determina el volumen de estos líquidos, que al verse confinados en un espacio con expansión a lo largo de un tubo, nos permite deducir la temperatura a la que se encuentran por el desplazamiento de la columna.

En estos casos de magnitudes que varían de forma continua hay un número infinito de valores que pueden presentarse. Los ordenadores, sin embargo, necesitan manipular las magnitudes de forma finita y discreta. Por ello, es necesario realizar la conversión de estas magnitudes a una representación directamente utilizable por las máquinas.

Las magnitudes pueden ser directamente de tipo eléctrico o bien presentarse como la señal generada mediante un dispositivo transductor. Se denomina transductores a los dispositivos que convierten una magnitud en otra. Algunos de estos transductores pueden trabajar en los dos sentidos: por ejemplo, un altavoz convierte una señal eléctrica en movimiento, pero se puede utilizar como micrófono si se invierte su funcionamiento, tomando la señal eléctrica generada al mover su membrana mediante ondas acústicas.

Para comunicar el ordenador con el mundo físico analógico es necesario disponer de instrumentos que permitan adaptar las señales generadas por los sensores analógicos a valores digitales, directamente utilizables por el ordenador. Estos dispositivos se denominan Conversores Analógico/Digital (C A/D) y Digital/Analógico (CD/A). Vamos a describir algunos métodos existentes para realizar las conversiones y presentaremos la realización de varios de ellos, empezando por los más sencillos, hasta llegar a los existentes en circuito integrado.

Para el diseño de conversores A/D y D/A es necesario tener presentes los siguientes parámetros:

— Resolución necesaria o número de niveles que han de poder distinguirse. La diferencia entre dos señales que puedan ser medidas como valores consecutivos.

— Valores máximos de las señales de entrada o de salida.

— Velocidad de conversión o tiempo mínimo entre muestras para conocer las características de la señal.

— Precisión necesaria (monotonicidad, estabilidad y linealidad).

— Polaridad de la señal: unipolar o bipolar.

— Presencia de ruido en la señal.

— Características del generador de la señal: impedancia, margen de frecuencias, margen de temperaturas.

Estos parámetros se utilizarán para seleccionar el equipo de medida o el sistema de adquisición de señales analógicas. Para nuestro caso diseñaremos circuitos de ejemplo utilizando componentes de fácil adquisición y de

sus características deduciremos las prestaciones máximas que podremos obtener del equipo. Es posible, sin embargo, realizar sistemas complejos y de aplicación práctica con cualquiera de nuestros ordenadores personales, siempre que pongamos las funciones necesarias en el equipo externo, si es que las limitaciones del ordenador no permiten realizarlas directamente con él.

Conversión digital/analógica

Empezamos describiendo el proceso de conversión de una señal digital en otra analógica equivalente, de acuerdo con una ley. Casi todos los conversores A/D llevan alguno de los tipos de conversor D/A para, mediante comparaciones sucesivas con valores de tanteo, realizar la conversión.

Una forma simple de convertir un número representado digitalmente en una magnitud eléctrica analógica consiste en hacer la suma ponderada de los valores de cada uno de los bits, asignando un coeficiente diferente a cada uno de ellos. El significado eléctrico es que la corriente o la tensión suministrada por cada uno sea proporcional a su peso en el número o, lo que es lo mismo, que cada entrada genere una corriente de acuerdo con un coeficiente asignado a cada bit, siguiendo las potencias de 2.

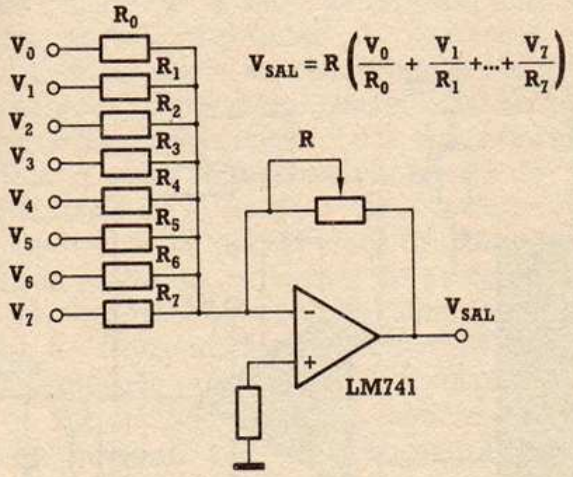


Fig. 1. Conversión digital/analógica por suma ponderada.

Con un circuito como el de la figura podemos generar corriente proporcional al cociente de la resistencia a través de la que introduce cada bit entre la resistencia de realimentación del amplificador operacional. El amplificador operacional funciona como sumador de corrientes y conversor corriente-tensión. La entrada se suele denominar "masa virtual", porque presenta casi cero voltios, pero con resistencia de entrada muy elevada, haciendo que la corriente en la resistencia de realimentación sea la suma de las corrientes de entrada. Hemos de hacer que la conversión tenga una resolución por lo menos equivalente al bit menos significativo, para lo cual deberán cumplirse algunos requisitos para los valores de los componentes. Por ejemplo, los generadores de las señales binarias deberán poder suministrar la corriente necesaria sin cambiar apreciablemente su tensión de salida y las resistencias deberán ser de la precisión apropiada para que los márgenes de tolerancia hagan que el error sea inferior al valor correspondiente al bit menos significativo. Si no se cumple esta condición ocurrirá que al asignar valores correlativos aparezca nivel menor para el código mayor. A este tipo de error se le denomina de falta de monotonicidad. Las resistencias que influyen con mayor importancia en el error son las de los bits más significativos. Se indica en la figura la forma de presentarse este error. La correspondencia entre los valores de entrada y los medidos no es monótonamente creciente. Hay otros errores que se presentan en los conversores como, por ejemplo, el desplazamiento del valor inicial, la falta de linealidad, la diferencia de ganancia con la señal de entrada y la dependencia de la temperatura.

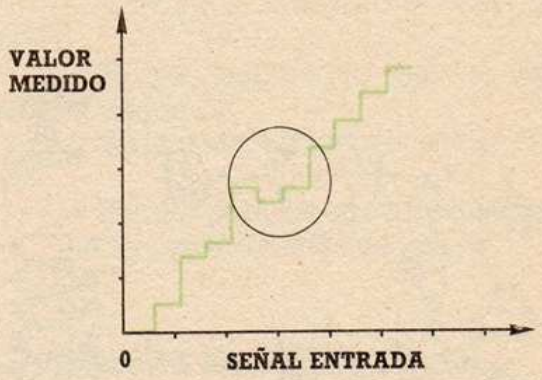


Fig. 2. Error de falta de monotonicidad.

El valor de la señal de salida correspondiente al conversor por suma ponderada se representa con la fórmula:

$$Vsal = R (V0/R0 + ... + V7/R7)$$

Los valores de las señales binaria aplicadas deberán ser iguales en su estado 1 y lo

más próximas a cero voltios en su estado 0. Mediante la resistencia R puede ajustarse la escala a los niveles apropiados a cada aplicación. En realidad, debemos sustituir en la fórmula los valores de las tensiones por los del producto del código binario por la tensión de salida de cada una de las entradas. En algunos casos puede interesar que la conversión no sea lineal, como, por ejemplo, en digitalización de voz en telefonía se utiliza un ley logarítmica de conversión, para mejorar la relación señal/ruido en los niveles bajos. Es decir, asignamos más bits de representación a los niveles bajos para que la influencia del ruido sea menor.

Para asignación binaria de pesos a los bits es necesario disponer de resistencias de valores en relación con las potencias de 2. Las resistencias normalizadas disponibles en el comercio a bajo precio (gama 5% o 10%) varían según saltos discretos de porcentaje, por lo que es necesario ajustar mediante dos resistencias en serie el valor adecuado para poder garantizar la resolución.

Para resolverlo de forma sencilla se dispone la red de resistencias de la forma indicada en la figura. Mediante la combinación R-2R puede conseguirse la asignación binaria de pesos utilizando solamente 2 valores diferentes de resistencias. Eléctricamente la solución es equivalente, pues en cada nodo la corriente se divide exactamente en dos ramas iguales. Un problema que se presenta con este tipo de red es que los valores de las señales analógicas a sumar deben ser: el nivel de referencia y 0. Si no ocurre así, por no ser muy perfectos los conmutadores empleados, aparecen efectos que hacen impracticable el método por encima de 4 bits. El circuito de alimentación de las señales para conversión puede hacerse con conmutadores MOS que presentan una relación grande abierto/cerrado. La señal de referencia también influye de forma importante en la señal generada, por lo que es recomendable generarla aparte de la alimentación general, con generadores estabilizados.

Para aplicaciones prácticas conviene disponer de por lo menos 8 bits de resolución, existiendo en el mercado gran número de circuitos integrados de coste relativamente bajo. Para la generación de señal convertida con mayor número de bit es necesario disponer de registros intermedios que se carguen secuencialmente desde el bus y que realicen la conversión de manera simultánea. Esta técnica se denomina "doble registro intermedio" y está disponible en algunos de los convertidores D/A del mercado.

En el circuito de la figura se describe una aplicación de uno de ellos. Para su montaje en los ordenadores personales descritos es recomendable utilizar la tarjeta de ampliación de puertos, descrita en el fascículo núme-

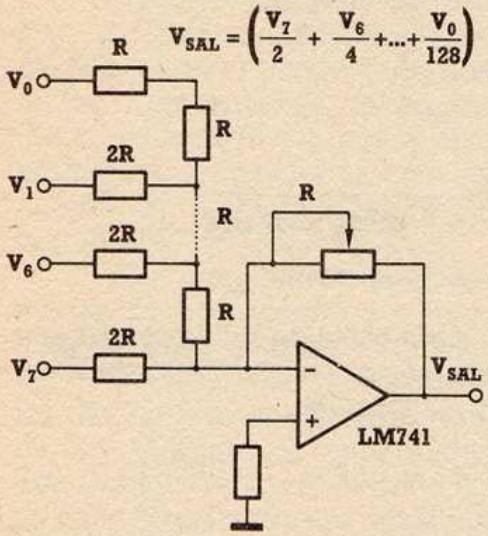


Fig. 3. Conversor D/A. Red de resistencias tipo R-2R.

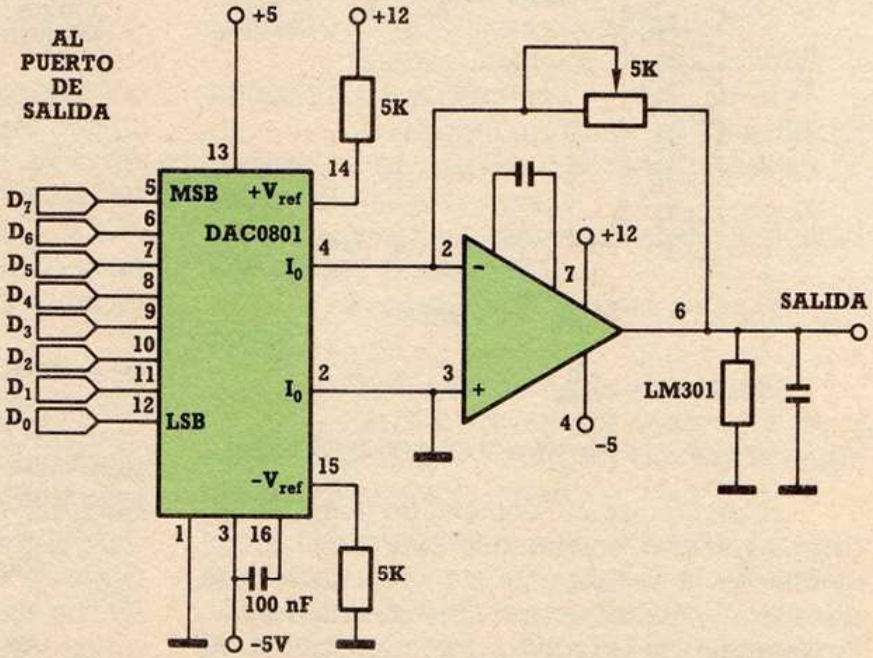


Fig. 4. Conversor digital/analógico.

ro 1. Se utiliza uno de los registros de salida para almacenar la información a convertir. La tarjeta podemos montarla sobre circuito de tiras y conectarla a la ampliación de puertos a través de un conector de borde. Las asignaciones de pines se dan en la figura. Conviene recubrir los terminales de las tiras que van a utilizarse como conector con estaño, para que el contacto sea mejor y menos oxidable. Hay que hacer notar que la tarjeta necesita tensiones de +5, +12 y -5 voltios.

```
SALIDA ANALOGICA
30 REM UTILIZA LAS DIRECCIONES DE PUER-
TO DE LA TARJETA
40 REM DE AMPLIACION DE PUERTOS
45 RETARDO=10
50 FOR I=0 TO 255
60 OUT PUERTO0,I
70 FOR J=0 TO RETARDO
80 NEXT J
90 NEXT I
100 GOTO 50
```

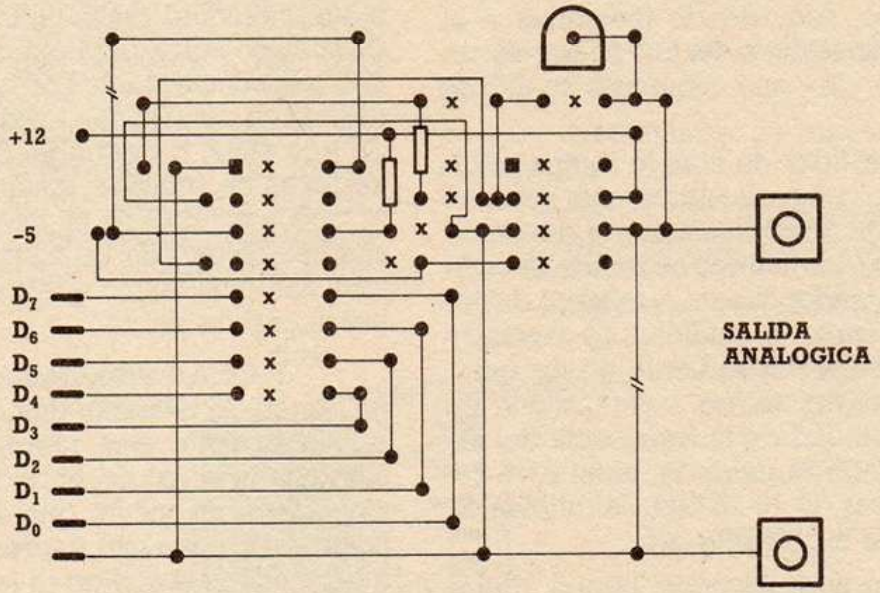


Fig. 5. Montaje práctico.

El circuito integrado empleado en la aplicación genera corriente proporcional a la señal binaria de entrada, de acuerdo con los pesos correspondientes de cada bit. La corriente de salida es convertida en tensión mediante el amplificador operacional. Se suele conectar un condensador a la salida del amplificador para reducir la pendiente de las transiciones de la salida y dar una onda más "redondeada". En aplicaciones de síntesis de señales mediante ordenador se suele conectar un filtro paso bajo para eliminar, en lo posible, las frecuencias por encima de las que realmente puede convertir el circuito.

Por ejemplo, podemos generar una señal en diente de sierra con el programa que se adjunta. Podemos variar la frecuencia introduciendo un retardo variable entre cada salida de nuevo valor de señal. La variable RETARDO fija la duración entre cada salida de muestras.

Programa CDADIENTE

```
10 REM PROGRAMA CDADIENTE
20 REM GENERA DIENTE DE SIERRA SOBRE
```

Lista de componentes del ejemplo de aplicación:

— Circuitos integrados (DAC0801, LM301).
— Resistencias según valores indicados en el esquema.
— Tarjeta de circuito impreso de soporte.
— Zócalos para los circuitos integrados.
— Conector exterior de salida.

Conversión analógico/digital

La conversión analógico/digital es la función complementaria de la anterior y permite utilizar los valores de las magnitudes físicas para ser procesadas por el ordenador, de forma digital.

Los métodos utilizados en la práctica junto con sus rangos de aplicación son:

— Conversor instantáneo o paralelo. Consiste en un conjunto de comparadores

igual al número de niveles discernible y que trabajan en paralelo, realizando cada uno de ellos la comparación con un nivel obtenido de un divisor de resistencias en cadena. La salida de los comparadores es tratada por un circuito digital combinacional, que genera el código apropiado correspondiente a la señal de entrada. Este procedimiento es muy costoso en circuitería, disponiéndose en la actualidad para resoluciones de 4 a 8 bits y velocidades de hasta 150 Megamuestras por segundo. Se utilizan en sistemas de tratamiento de señales y de imágenes, requiriendo memorias y el resto de los dispositivos del OP de prestaciones adecuadas para que compense su utilización.

— Convertidor de simple rampa. Utiliza el método de comparación con una señal en rampa generada analógicamente o mediante un conversor D/A. Mientras se realiza la comparación, un contador cuenta el número de impulsos hasta alcanzar la igualdad. La precisión conseguida resulta dependiente de los parámetros de los componentes: condensador, integrador, estabilidad de la frecuencia del oscilador. El método es, además, lento para resoluciones típicas de 10-12 bits. Se emplea en instrumentos de media precisión.

— Convertidor de doble rampa. Utiliza un método similar de comparación de la señal de entrada con una rampa, pero en este caso la rampa resulta generada a partir del valor máximo alcanzado durante un intervalo de integración fijo y descargado mediante una fuente de referencia. Se consiguen compensar en parte los defectos del primer método. El tiempo de conversión también es grande. Se emplea este método en instrumentación de precisión.

— Convertidor por aproximaciones sucesivas. Utiliza un procedimiento dicotómico de comparación de la señal de entrada con una señal generada a través de un C D/A. El algoritmo que tiene que seguir consiste en empezar por el bit de mayor signo comparándolo con la entrada. Si el resultado es de que es mayor la entrada, se deja el bit en 1 y si no se pone en cero. A continuación se prueba con el bit de orden siguiente, realizando otra vez la comparación. Se sigue sucesivamente con todos los bits, utilizando el mismo procedimiento de comparación. El tiempo de conversión será el tiempo de comparación de un bit multiplicado por el número de bits. Para el almacenamiento de los bits en comparación se emplea un circuito compuesto de biestables (Flip-Flops = FF) y montado para realizar la función de aproximaciones sucesivas. El circuito permite generar, a la vez que se hace la conversión, una señal digital serie con el valor convertido, empezando por el bit más significativo. Se utiliza en equipos de adquisición de datos en general.

— Convertidor de conmutadores múltiples. Consiste en una cadena de resistencias igual al número de niveles discernible, conectadas mediante conmutadores en estructura compuesta serie/paralelo a una de las entradas del comparador. Por aproximaciones sucesivas se determina el código más aproximado a la señal de entrada. Se emplea en conversores de relación, cuando es fundamental la monotonicidad de la conversión y se necesita flexibilidad en los límites superior e inferior. Son más lentos que los de aproximaciones sucesivas puros.

— Convertidor de seguimiento. El código generado alimenta un conversor D/A que constantemente está realizando la comparación con la señal de entrada. Si detecta que hay diferencia mayor que el bit menos significativo incrementa o decrementa el contador que mantiene el código en la dirección correspondiente a la diferencia. El circuito es muy simple y la velocidad de conversión no muy alta.

— Conversor tensión/frecuencia. Utilizan un oscilador de tipo estable controlado por tensión. La frecuencia generada es proporcional a la tensión de entrada. El método de medida consiste en contar el número de pulsos en un período determinado o el período de la señal. Se utilizan en los sensores remotos, en conexión a dos hilos a la unidad central, para señales de variación lenta. Permiten gran precisión, utilizando período de lectura grande.

Los criterios de diseño son los mismos que para los C D/A.

Ejemplo de aplicación

El ejemplo propuesto utiliza un circuito integrado que posee 8 canales analógicos de entrada, resolución de 8 bits y posibilidad de conexión directa al bus del ordenador. En nuestra aplicación podemos conectarlo a tra-

vés de la tarjeta de ampliación de puertos con las conexiones indicadas en la figura. La entrada de OE está permanentemente activada. Requiere para su funcionamiento solamente: un oscilador con frecuencia inferior a 1,2 MHz., direccionamiento del canal a medir y su señal de indicación de cuando la dirección es válida y el pulso de comienzo. Da una señal indicativa del final de la conversión, que puede ser leída por el programa para saber cuándo poder iniciar otra medida. Si se utiliza el intérprete de BASIC es casi seguro que el tiempo entre medidas va a ser superior al tiempo de conversión, por lo que podemos hacer cada medida sin consultar el bit de CONVERSION EN CURSO. Si hacemos la lectura desde lenguaje de máquina, sí será necesaria la consulta del bit, por ser el intervalo entre medidas mucho menor. El procedimiento de medida es el de comparación por conmutadores múltiples.

vas entre los valores de las referencias positiva y negativa, pudiendo conseguirse resoluciones de más de 8 bits si el rango de medida es restringido. Por ejemplo, para medir temperatura con un sensor lineal, dentro de un margen limitado de temperaturas, puede conseguirse una sensibilidad de 0,2 grados centígrados por bit. El tiempo de conversión es de 50 microsegundos aproximadamente.

Lista de componentes para el ejemplo de aplicación:

— Circuitos integrados (ADC0808, LM555).

— Resistencias de valores indicados en el esquema.

— Tarjeta de circuito impreso de tiras.

— Zócalos para los circuitos integrados.

— Conector para las entradas.

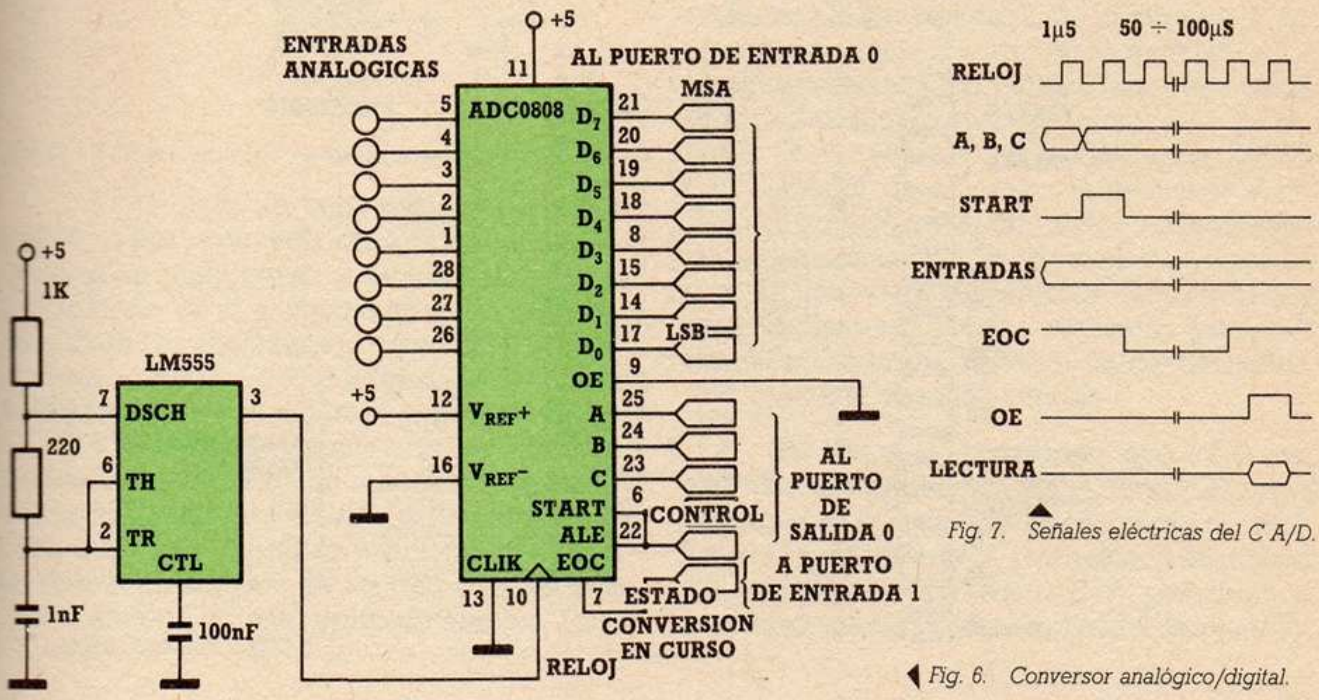


Fig. 7. Señales eléctricas del C A/D.

Fig. 6. Conversor analógico/digital.

El circuito puede conectarse directamente al bus de datos. Para ello hay que modificar ligeramente las conexiones en los puntos siguientes:

— Las entradas A,B,C van directamente al bus de datos D2, D1, D0.

— Las señales Start y ALE se conectan juntas al punto de decodificación de escritura.

— El control de salida OE se conecta al punto de decodificación de lectura.

Es interesante hacer notar que la conversión se realiza por aproximaciones sucesi-

Muestreo de señales

La digitalización de señales impone la representación de éstas de forma discreta en el número de niveles, pero también en su definición en el tiempo. Es decir, discretizamos también los instantes en los cuales tiene significación la señal para el ordenador. Es necesario considerar la información temporal de la señal para realizar el muestreo con la periodicidad adecuada. El hecho de conocer una señal de forma discreta puede hacer pensar que se pierde información significativa de la señal.

El conocimiento de la composición frecuencial de la señal nos indica que no se pierde información, siempre que el muestreo satisfaga las condiciones indicadas por el teorema de muestreo o de Nyquist, que expresa que para una señal con ancho de banda limitado es suficiente con muestrear a una frecuencia igual al doble del armónico más alto presente en la señal. Si se cumple esta condición, el proceso de reconstrucción o interpretación de la señal será posible, empleando un filtro reconstructor ideal. En la práctica lo que conviene es filtrar todo lo posible las frecuencias en la señal por encima de las que interesen y muestrear por encima del valor mínimo indicado, para facilitar la reconstrucción. Realmente no tiene sentido tomar más muestras que las necesarias para el ancho de banda de la señal dada, pues al ser su información limitada no se va a sacar más por tomar más muestras. Unicamente puede ser útil para simplificar el proceso de reconstrucción. El efecto observado, si no se cumplen las condiciones de muestreo, es el de que aparecen señales en la reconstrucción que son "alias" de las señales que realmente se muestrearon. En la figura puede verse el proceso por el cual aparece señal aparente de frecuencia en la banda útil, originada por la presencia de un armónico de frecuencia próxima a la de muestreo y, por tanto, fuera de las condiciones indicadas, por no ser inferior a la mitad de la frecuencia de muestreo.

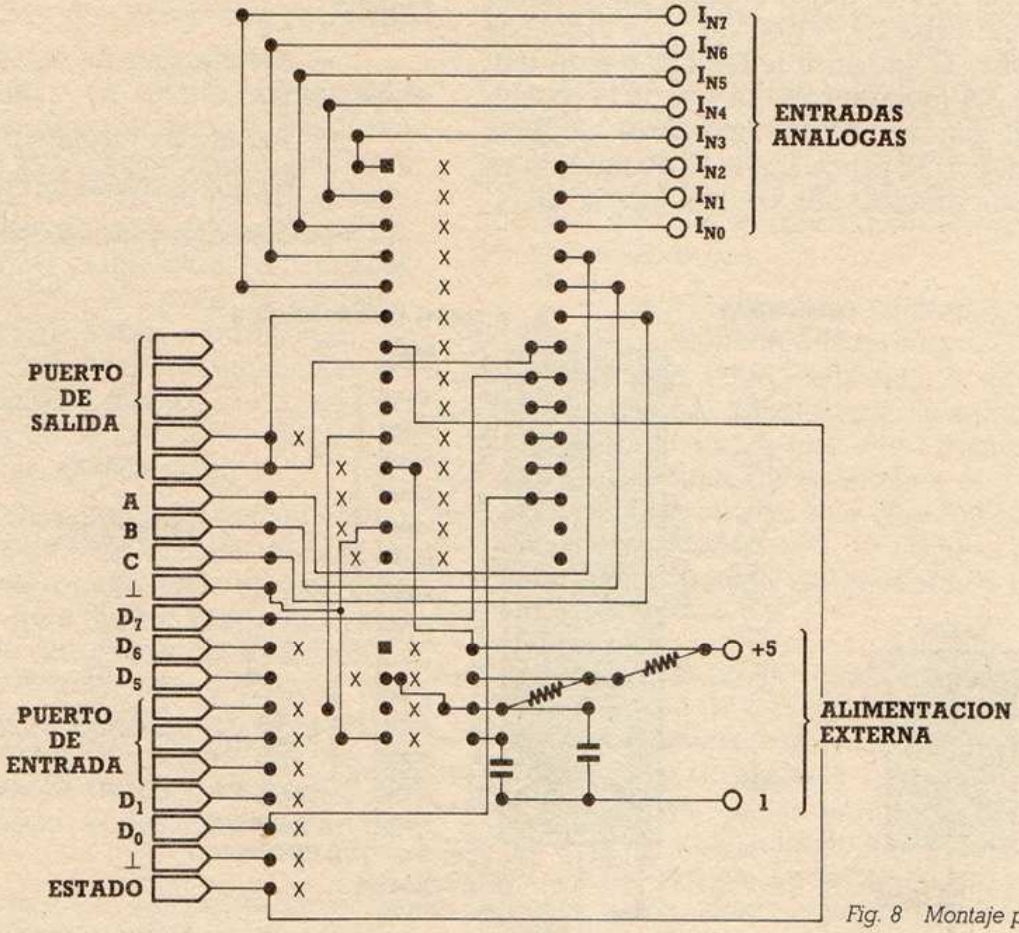


Fig. 8 Montaje práctico.

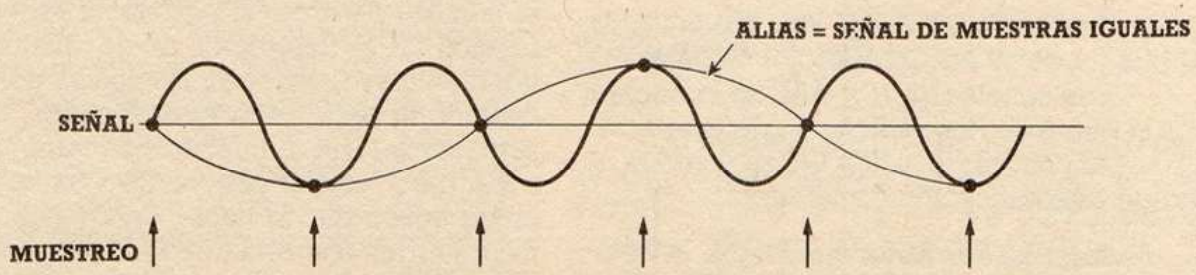


Fig. 9 Efecto del muestreo incorrecto.

La señal de puntos sería la señal leída e interpretada como señal de una frecuencia no existente en la señal original, por el efecto del muestreo a frecuencia inferior al doble de la componente más alta de la señal de entrada.

Multiplexado de señales

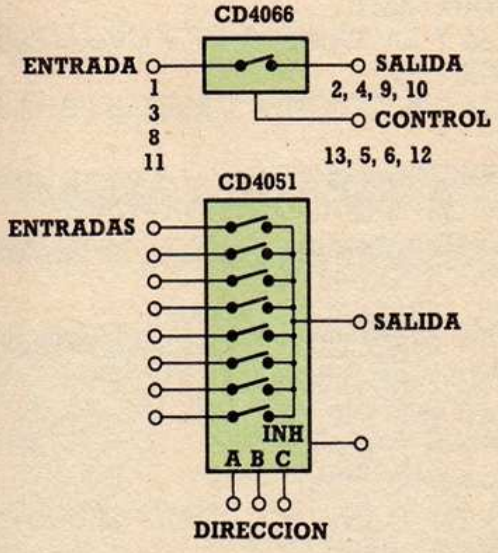


Fig. 10. Circuitos multiplexores analógicos.

El circuito de conversión A/D puede utilizarse para convertir señales de varios sensores si se multiplexan y se toma la muestra de cada uno en el momento en que sea requerido. Para ser considerado como ideal un multiplexor analógico deberá dejar pasar la señal seleccionada sin atenuación y deberá rechazar el efecto de las demás señales que comparten el equipo. Existen numerosos dispositivos comerciales que realizan la función de manera casi ideal utilizando conmutadores MOS. Se muestran algunos de los más usuales y se representa un circuito típico de empleo. En el ejemplo descrito para la conversión A/D se utiliza un circuito que incluye 8 posibles entradas para señales analógicas.

Muestreo y retención

El proceso de conversión A/D se ha indicado que puede necesitar un tiempo no despreciable, como es el caso del método de aproximaciones sucesivas. Es, por tanto, necesario mantener la señal en el valor de la muestra de forma estable durante la conversión. Solamente con los conversores de tipo instantáneo puede prescindirse del circuito de muestreo y retención; para los demás casos es recomendable, para garantizar que durante el período de conversión la señal varía menos que el valor del bit menos significativo. De todas maneras, será necesario evaluar para cada caso la conveniencia de incorporarlo, teniendo en cuenta las frecuencias de las señales muestreadas. Otro punto a tener en cuenta es la necesidad de muestreo simultáneo de señales. Si es necesario en la aplicación, habría que disponer de circuitos de muestreo y retención propios para cada señal y multiplexar la conversión A/D. Los circuitos de muestreo y retención están disponibles como módulos específicos, pero pueden realizarse mediante muestreadores analógicos y un condensador, que conviene sea muy estable.

Es necesario mencionar que cada vez son más asequibles circuitos integrados que disponen en un solo chip las funciones de: multiplexado, muestreo y retención, conversión A/D, proceso y conversión D/A, permitiendo la realización de bucles de control autónomos,

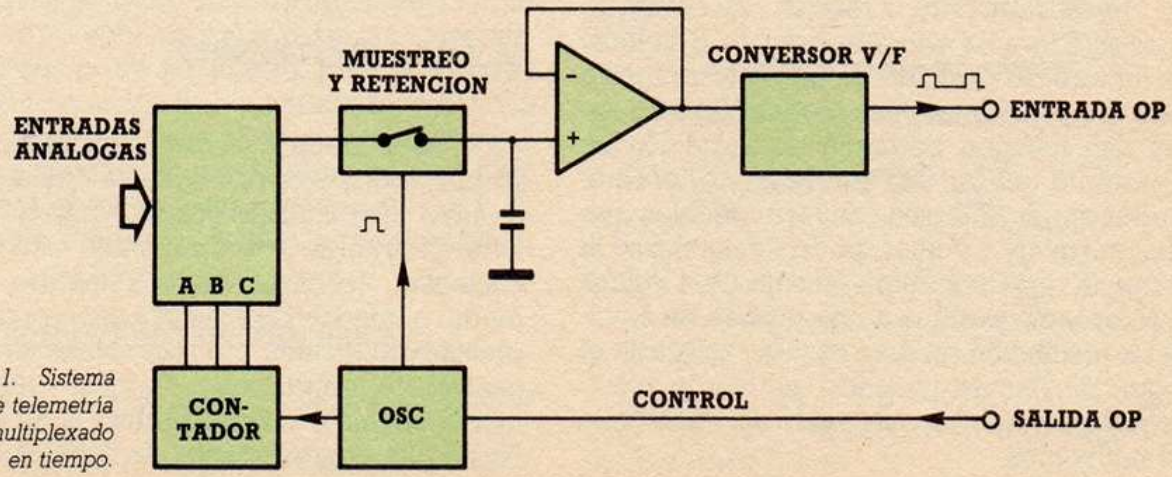


Fig. 11. Sistema de telemetría por multiplexado en tiempo.

con el solo requerimiento de la adición de una memoria y una fuente de alimentación.

Ejemplo de aplicación: Sistema de adquisición de datos analógicos

Como aplicación se utiliza el circuito montado en la figura 8 y descrito lógicamente en la figura 7 para adquisición de datos. Las señales son analógicas de bajo nivel, con valores entre 0 y 5 voltios. Los sensores alimentan amplificadores operacionales, que convierten las señales al rango admitido por el conversor. En la tarjeta diseñada para ser co-

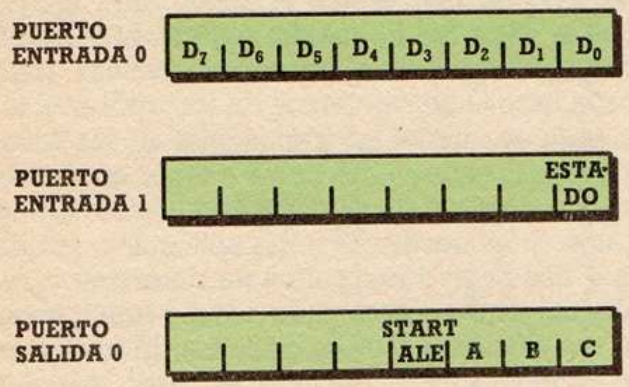


Fig. 12. Asignación de puertos y bits.

nectada a la tarjeta de ampliación de puertos se utilizan 4 bits de uno de los puertos de salida: 3 para direccionar la entrada a medir y 1 bit para iniciar la lectura. Un puerto de entrada completo se utiliza para leer los 8 bits del valor de la señal convertida. Puede emplearse 1 bit de entrada del otro puerto para leer el indicador de CONVERSION en CURSO.

La entrada para la tensión referencia se ha conectado a la alimentación de 5 voltios. Para lectura de posición de un potenciómetro puede ser suficiente, pero para lectura de señales con margen pequeño de variación es conveniente utilizar una referencia diferente. Por ejemplo, si se desea convertir señales que varían entre 4 y 5 voltios, podría conectarse la referencia negativa a una tensión de 4 voltios y la referencia positiva a una tensión de 5 voltios. La resolución en este caso se aplicaría al margen de lectura utilizado, pudiendo resolverse 255 niveles con un valor de cada salto de 4 milivoltios.

Un programa sencillo de lectura de señales de las 8 entradas podría ser el que se muestra más abajo. La activación de la lectura se realiza mediante el bit CONTROL, que se activa simultáneamente con los bits de dirección en el puerto de salida.

```
10 REM LECTURA DE DATOS ANALOGICOS
20 REM CON LA TARJETA DE ENTRADA
   ANALOGICA
30 REM Y LA TARJETA DE AMPLIACION DE
   PUERTOS
40 REM DIRECCIONES VALIDAS PARA IBM-
   PC
50 DIRECCION=&H201
60 DATOS=&H201: CONTROL=&H08
70 ESTADO=&H203
80 FOR CANAL=0 TO 7
90 OUT DIRECCION,CANAL
100 OUT DIRECCION,CANAL+CONTROL
110 OUT DIRECCION,CANAL
120 B=1 AND INP(ESTADO)
130 IF B=0 THEN 120
140 A=INP(DATOS)
150 PRINT "CANAL=";CANAL;" VALOR=";A
160 NEXT CANAL
170 GOTO 80
```

Es necesario cambiar las direcciones de puerto para los otros OP, poniendo en lugar de DIRECCION, DATOS y ESTADO las correspondientes a las que hayamos asignado en nuestra tarjeta de ampliación de puertos. Las recomendaciones hechas para la tarjeta de conversión digital/analógico son aplicables también a esta otra.

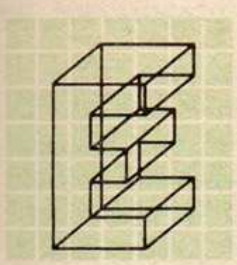
Hay que recordar también los diferentes códigos empleados en cada máquina para las instrucciones de ENTRADA y SALIDA, que deberán sustituir a las utilizadas en el ejemplo INP y OUT.

Otras aplicaciones

Con las tarjetas descritas en este número para conexión con señales analógicas podemos comunicarnos con un sinfín de dispositivos diferentes y realizar muy variadas aplicaciones. Veremos cómo utilizarlas directamente o integradas en otras tarjetas más complejas para diseñar equipos sofisticados, como control de temperatura de varios recintos simultáneamente o representación en la pantalla de nuestro ordenador de señales de voz.

NATURALEZA Y TECNOLOGIA

Física: el movimiento ondulatorio



En este apartado vamos a abordar uno de los fenómenos más interesantes de la Naturaleza. Se trata del movimiento ondulatorio. Veremos los conceptos fundamentales en los que se basa. En sucesivos tomos profundizaremos en el tema.

Si en alguna ocasión hemos tirado una piedra al río, habremos observado que en el momento de contactar aquélla con la superficie del agua se producen una serie de perturbaciones que se transmiten a lo largo del líquido elemento. Es de todos conocida la existencia de olas en el mar. Cuanto más embravecido se encuentre, mayores serán las olas. Un fenómeno semejante ocurre cuando flamean las banderas por acción del viento.

En general, estas perturbaciones se conocen como ondas. Su estudio constituye un apartado de la Física denominado movimiento ondulatorio.

El movimiento ondulatorio es una forma de transmitir energía a distancia sin transporte de masas ni de partículas. Este es el caso, de la propagación de las ondas de radio, de televisión o telefónicas. El comportamiento del sonido se ajusta a un movimiento de este tipo. Esto también sucede cuando se sujeta por un extremo un resorte y se tira del otro extremo.

Características de una onda

Una onda se caracteriza por una serie de parámetros:

Amplitud. Es el desplazamiento máximo de las partículas que vibran en el medio de propagación.

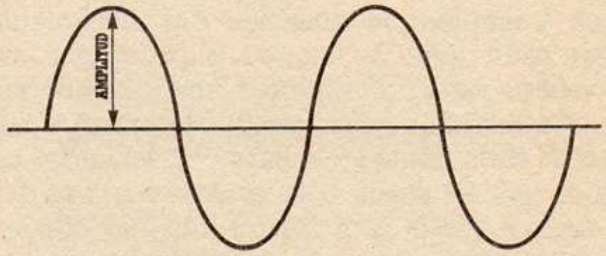


Fig. 1. Amplitud de una onda.

Longitud de onda. Es la distancia que hay entre dos partículas consecutivas de una onda que están en concordancia de fase. Se dice que dos partículas están en concordancia de fase cuando presentan el mismo desplazamiento respecto al eje de vibración.

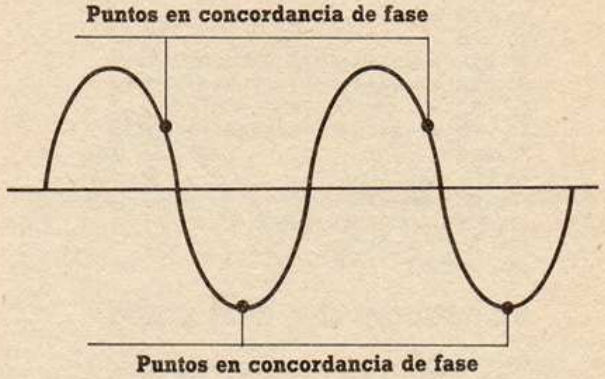


Fig. 2. Puntos en concordancia de fase.

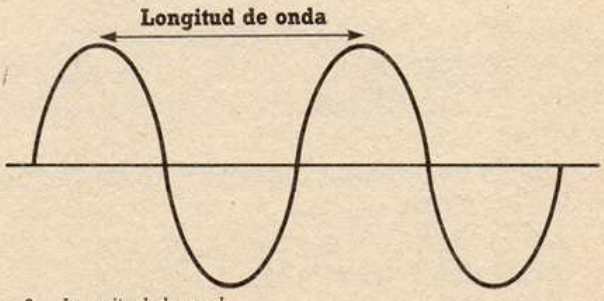


Fig. 3. Longitud de onda.

Frecuencia. Es el número de ondas que pasan por un punto dado en la unidad de tiempo.

Ecuación de una onda

Una onda responde a una ecuación sinusoidal que simplificadamente podría expresarse del siguiente modo:

$$y = A\,\text{sen}\ w$$

Podemos construir un programa que simule el movimiento de una onda. Para ello, vamos a emplear la ecuación del movimiento para cada punto de la onda. El programa está diseñado para ordenadores Amstrad, aunque puede adaptarse fácilmente para otros equipos. A cada punto se le hará corresponder un valor que se ajuste a la ecuación sinusoidal. Debido a que la pantalla del Amstrad presenta 640 x 400 puntos en alta resolución, hemos empleado un bucle que llegue hasta el número 640 para los puntos del eje de abscisa y hemos centrado el origen sumando 200 unidades a los valores de y.

```
10 REM ***************************
20 REM *         PROGRAMA        *
30 REM *           DE            *
40 REM * MOVIMIENTO ONDULATORIO *
50 REM ***************************
60 REM *  VALIDO PARA AMSTRAD    *
70 REM ***************************
80 CLS
90 FOR I=0 TO 640
100 LET AN=I*PI/180
110 LET A=50*SIN(2*AN)
120 PLOT I,A+200
130 NEXT I
```

MODIFICACIONES PARA OTROS EQUIPOS

SPECTRUM

```
90 FOR I=0 TO 255
120 PLOT I,A+70
```

IBM

```
75 SCREEN 2
77 PI=3.14159
120 PSET (I,100-A)
```

MSX

```
75 SCREEN 1
77 PI=3.14159
90 FOR I=0 TO 255
120 PSET (I,96-A)
```

Estudio de diferentes fenómenos asociados a las ondas

Ondas con distinta amplitud. En la ecuación de la onda, el parámetro A se corresponde con la amplitud, dado que el valor máximo que puede alcanzar "y", esto es, la amplitud, será aquél en el que el seno sea igual a uno, es decir:

$$y = A$$

Para ilustrar gráficamente el concepto de amplitud vamos a construir un programa en el que vaya aumentando la amplitud de una serie de ondas que presenten la misma longitud de onda. Por ello, en lugar de utilizar una amplitud constante e igual a 100, emplearemos los valores variables que proporciona un bucle.

```
10 REM ***********************
20 REM *      PROGRAMA       *
30 REM *         DE          *
40 REM *      VARIACION      *
50 REM *    DE LA AMPLITUD   *
60 REM *     DE UNA ONDA     *
70 REM ***********************
80 REM * VALIDO PARA AMSTRAD *
90 REM ***********************
100 FOR J=25 TO 75 STEP 25
110 CLS
120 PLOT 0,200
130 DRAWR 640,0
140 FOR I=0 TO 640
150 LET AN=I*PI/180
160 LET A=J*SIN(AN*2)
170 PLOT I,A+200
180 NEXT I
190 PLOT 45,200
200 DRAWR 0,J
210 FOR Z=1 TO 2000:NEXT Z
220 NEXT J
230 CLS
240 PRINT "COMO HABRAS OBSERVADO"
250 PRINT "EN ESTE PROGRAMA"
260 PRINT "VA AUMENTANDO PAULATINAMENTE"
270 PRINT "LA AMPLITUD DE LAS ONDAS"
```

MODIFICACIONES PARA OTROS EQUIPOS

SPECTRUM

```
120 PLOT 0,75
130 DRAW 255,0
140 FOR I=0 TO 255
170 PLOT I,A+75
190 PLOT 45,75
200 DRAW 0,J
```

IBM

```
95 SCREEN 2
97 PI=3.14159
120 LINE (640,100)-(0,100)
SUPRIMIR LA LINEA 130
170 PSET (I,100-A)
190 LINE (45,100-J)-(45,100)
SUPRIMIR LA LINEA 200
```

```
MSX

95 SCREEN 1
97 PI=3.14159
120 LINE (255,100)-(0,100)
```

```
SUPRIMIR LA LINEA 130
140 FOR I=0 TO 640
170 PSET (I,100-A)
190 LINE (45,100-J)-(45,100)
SUPRIMIR LA LINEA 200
```

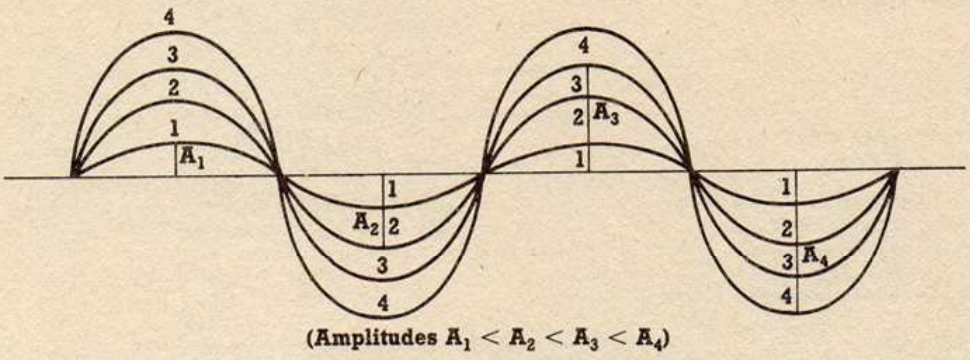


Fig. 4. Ondas con la misma longitud de onda y distinta amplitud.

Ondas con distinta longitud de onda. La frecuencia es inversamente proporcional a la longitud de onda. De modo simplificado podría decirse que el valor de w (argumento de la función seno) sería directamente proporcional a la frecuencia e inversamente proporcional a la longitud de onda. Para ilustrar gráficamente el concepto de longitud de onda hemos diseñado un programa en el que con un valor constante de la amplitud, varíen las longitudes de onda. Por ello se emplean valores crecientes de w, por lo que disminuirá paulatinamente la longitud de onda y aumentará la frecuencia.

```
10 REM ************************
20 REM *       PROGRAMA       *
30 REM *    DE VARIACION      *
40 REM *          DE          *
50 REM * LA LONGITUD DE ONDA  *
60 REM ************************
70 REM * VALIDO PARA AMSTRAD  *
80 REM ************************
90 FOR J=1 TO 6
100 CLS
110 FOR I=0 TO 640
120 LET AN=I*PI/180
130 LET A=50*SIN(AN*J)
140 PLOT I,A+200
150 NEXT I
160 PLOT 90/J,250
170 DRAWR 360/J,0
180 FOR Z=1 TO 2000:NEXT Z
190 NEXT J
200 PRINT "COMO HABRAS PODIDO OBSERVAR"
210 PRINT "EN ESTE PROGRAMA"
220 PRINT "VA AUMENTANDO PAULATINAMENTE"
230 PRINT "LA LONGITUD DE ONDA"

MODIFICACIONES PARA OTROS EQUIPOS

SPECTRUM

90 FOR J=2 TO 6
110 FOR I=0 TO 255
140 PLOT I,A+75
160 PLOT 90/J,125
170 DRAW 360/J,0

IBM
85 SCREEN 2
87 PI=3.14159
140 PSET (I,100-A)
160 LINE (450/J,50)-(90/J,150)
SUPRIMIR LA LINEA 170

MSX

85 SCREEN 1
87 PI=3.14159
90 FOR J=2 TO 6
140 PSET(I,100-A)
160 LINE (450/J,50)-(90/J,150)
SUPRIMIR LA LINEA 170
```

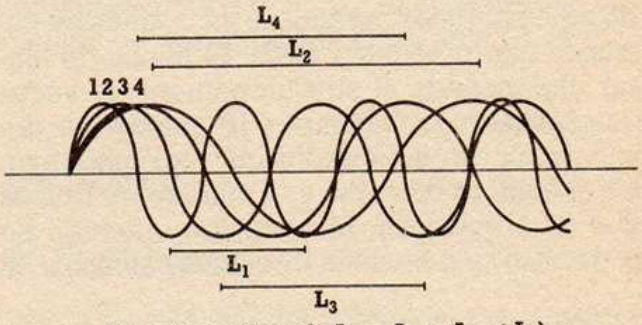


Fig. 5. Ondas con la misma amplitud y distintas longitudes de onda.

Este fenómeno sucede, por ejemplo, en el caso de las ondas de radio. Las emisoras locales emplean longitudes de onda pequeñas, o lo que es lo mismo, frecuencias grandes, por lo que su alcance será menor y su energía mayor. Se oyen mejor que las emisoras nacionales pero están limitadas a un ámbito local.

Ondas en distinta fase. El ángulo w puede ser el mismo o puede ser diferente para

dos ondas cualesquiera. Este hecho será decisivo, a la hora de estudiar el fenómeno de interferencias entre ondas. En este caso hemos elaborado un programa en el que w sea distinto entre dos ondas. La diferencia es de 90 grados, es decir, radianes.

```
PROGRAMA 4

10 REM ************************
20 REM *        PROGRAMA      *
30 REM *          DE          *
40 REM *       DOS ONDAS      *
50 REM *  CON DISTINTA FASE   *
60 REM ************************
70 REM * VALIDO PARA AMSTRAD *
80 REM ************************
90 CLS
100 FOR I=0 TO 640
110 LET AN=I*PI/180
120 LET A=50*SIN(2*AN)
130 LET B=50*SIN(2*AN+PI/2)
140 PLOT I,A+200
150 PLOT I,B+200
160 NEXT I
170 PRINT "COMO HABRAS OBSERVADO "
180 PRINT "LAS DOS ONDAS ESTAN DESFASADAS"
190 PRINT "EN 90 GRADOS"

MODIFICACIONES PARA OTROS EQUIPOS

SPECTRUM

100 FOR I=0 TO 255
140 PLOT I,A+75
150 PLOT I,B+75

IBM

85 SCREEN 2
87 PI=3.14159
140 PSET(I,100-A)
150 PSET(I,100-B)

MSX

85 SCREEN 1
87 PI=3.14159
100 FOR I=0 TO 255
140 PSET(I,96-A)
150 PSET(I,96-B)
```

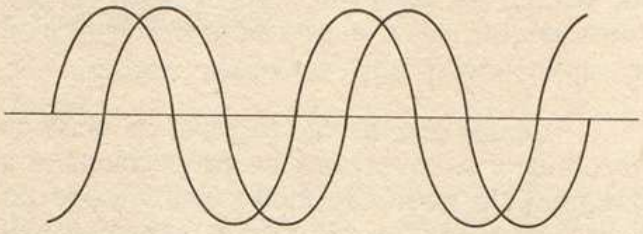


Fig. 6. Ondas desfasadas en 90°. La longitud de onda y la amplitud es la misma.

MATEMATICAS

Estadística: Cálculo de frecuencias y representación gráfica

La estadística es una ciencia que trata de agrupar datos basándose en cualidades comunes que presenten éstos. El hecho de lanzar una moneda al aire un número de veces determinado, por ejemplo, 10, puede ser objeto de un estudio estadístico. Podríamos anotar el número de caras y el número de cruces que se obtuviesen. El resultado obtenido en cada caso se denomina frecuencia absoluta. Si en el experimento se sacaron tres caras y siete cruces, la frecuencia absoluta del suceso "cara" sería tres, siendo siete la del suceso "cruz". El cálculo de frecuencias absolutas es un fenómeno que se repite en la vida real. Así, por ejemplo, el recuento de unas elecciones o el escrutinio quinielístico son casos de lo anterior.

En este apartado vamos a construir un programa que calcule la frecuencia de aparición de una serie de notas en una clase. Además realizaremos una representación gráfica en forma de diagrama horizontal.

```
10 REM ************************************
20 REM *              PROGRAMA            *
30 REM *                DE                *
40 REM * CALCULO DE FRECUENCIAS ABSOLUTAS *
50 REM ************************************
60 REM
70 REM  ************************************
80 REM  *        PROGRAMA VALIDO PARA      *
90 REM  * IBM,AMSTRAD,SPECTRUM,MSX Y COMMODORE*
100 REM ************************************
110 CLS:REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
120 DIM C(10):DIM P(10):DIM E(10)
130 INPUT "NUMERO DE NOTAS";M
140 CLS:REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
150 FOR I=1 TO M
160 PRINT "NOTA NUMERO";I
170 INPUT N
180 IF N<1 OR N>9 THEN GOTO 170
```

```
190 LET C(N)=C(N)+1
200 CLS:REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
210 NEXT I
220 REM ***********************
230 REM * CALCULO DEL MAXIMO *
240 REM ***********************
250 LET MAXIMO=C(1)
260 FOR I=1 TO 10
270 IF C(I)>MAXIMO THEN LET MAXIMO=C(I)
280 NEXT I
290 REM ****************************
300 REM * TRANSFORMACION A ESCALA *
310 REM ****************************
320 FOR I=1 TO 10
330 LET E(I)=C(I)*20/MAXIMO
340 NEXT I
350 REM ****************************
360 REM * CALCULO DEL PORCENTAJE *
370 REM ****************************
380 FOR I=1 TO 10
390 LET P(I)=C(I)*100/M
400 NEXT I
410 REM ********************************
420 REM * VISUALIZACION DE RESULTADOS *
430 REM ********************************
440 FOR I=1 TO 10
450 PRINT I;" ";
460 FOR J=1 TO E(I)
470 PRINT "*";
480 NEXT J
490 PRINT " ";TAB(28);P(I);"%"
500 NEXT I
```

```
1  *                    2 %
2  **                   4 %
3  ****                 8 %
4  ********             16 %
5  ********************  38 %
6  ********             16 %
7  *****                10 %
8  **                   4 %
9  *                    2 %
```

Fig. 7. Representación gráfica de una distribución de notas.

Comentario del programa

El programa está estructurado en tres partes. En la primera se introducen los datos. Al mismo tiempo se calculan las frecuencias de las notas. Para ello, se utiliza un contador con subíndice denominado C(10). Cada vez que se repita una nota tendremos el mismo subíndice, por lo que se incrementará el contador de dicha nota.

Debido a que no se conoce "a priori" el número total de notas, es interesante introducir un factor de escala. Esto es lo que se hace en la segunda parte. Para ello, se calcula, en primer lugar, la frecuencia máxima mediante una variable que se denomina MAXIMO. Luego, y en función del ordenador del que se trate, se hace la transformación a escala, dividiendo todas las frecuencias por el valor máximo y multiplicando por algo menos del número de columnas que tenga la pantalla del ordenador que estemos utilizando. En cualquier caso, en este programa multiplicaremos por 20 columnas para hacer que el programa sea estándar.

En la tercera parte del programa se visualizan los resultados mediante un gráfico horizontal. Hemos empleado el asterisco como símbolo, aunque pueden utilizarse otros, como el cuadrado relleno que, en general, corresponde al código ASCII 143, con lo que la función CHR$(143) podría servir. El gráfico se construye mediante un bucle variable que sea función de la frecuencia de la nota.

SOCIEDAD

El dadaísmo

En esta sección vamos a ver cómo visualizar mensajes en todas las direcciones. Este programa podría servir como ilustración del tratamiento que daban los dadaístas a las palabras.

El dadaísmo fue una escuela de arte y de literatura que apareció sobre 1917 y estuvo en boga en Francia y Suiza durante algunos años. Los dadaístas sostenían la teoría de que lo importante en una palabra no era el significado, sino su forma. Por ello, jugaban con las letras de las palabras escribiéndolas en distintas direcciones.

```
10 REM ******************************
20 REM *          PROGRAMA          *
30 REM *             DE             *
40 REM * VISUALIZACION DE PALABRAS *
50 REM *             EN             *
60 REM *   DISTINTAS DIRECCIONES    *
70 REM ******************************
80 REM
90  REM ******************************
100 REM *    PROGRAMA VALIDO PARA     *
110 REM * AMSTRAD,IBM,MSX y COMMODORE *
120 REM ******************************
130 CLS: REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
140 INPUT "DIME UNA PALABRA";N$
150 CLS: REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
160 LET N=LEN(N$)
170 FOR I=1 TO N
180 PRINT MID$(N$,I,1);
190 PRINT " ";
200 NEXT I
210 FOR Z=1 TO 2000:NEXT Z
220 CLS: REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
230 FOR I=1 TO N
240 PRINT MID$(N$,I,1)
250 PRINT " "
260 NEXT I
270 FOR Z=1 TO 2000:NEXT Z
280 CLS: REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
290 FOR I=1 TO N
300 PRINT TAB(I);MID$(N$,I,1)
310 NEXT I
320 FOR Z=1 TO 2000:NEXT Z
330 CLS: REM * PONER PRINT CHR$(147) PARA EL COMMODORE *
340 FOR I=N TO 1 STEP -1
350 PRINT TAB(I);MID$(N$,I,1)
360 NEXT I
370 REM ********************************
380 REM * MODIFICACIONES PARA SPECTRUM *
390 REM ********************************
400 REM
410 REM ************************************
420 REM * SUSTITUIR LAS SIGUIENTES LINEAS *
430 REM *    180 PRINT N$(I TO I);         *
440 REM *    240 PRINT N$(I TO I)          *
450 REM *    300 PRINT TAB(I);N$(I TO I)   *
460 REM *    350 PRINT TAB(I);N$(I TO I)   *
470 REM ************************************
```

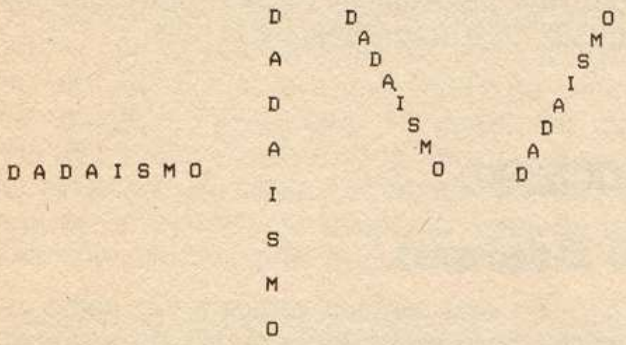


Fig. 8. Impresión de mensajes en distintas direcciones.

Comentario del programa

El programa utiliza funciones de tratamiento de cadenas. Concretamente, se utilizan las funciones LEN, que sirve para determinar la longitud de la cadena, y MID$, empleada para descomponer una palabra letra por letra. Para esto último se utiliza un bucle que llegue hasta la longitud de la palabra.

Por supuesto, el programa es susceptible de ser ampliado y mejorado. Sólo hemos pretendido dar una pequeña muestra de cómo visualizar mensajes en distintas direcciones.

PARA LOS MAS JOVENES

Cálculo del Máximo Común Divisor y Mínimo Común Múltiplo

Dentro del estudio de las matemáticas elementales está el cálculo del Máximo Común Divisor y el Mínimo Común Múltiplo.

Estos dos conceptos introducidos son muy sencillos, seguro que ya los conoces. Sin embargo, puede ser interesante y ameno el aplicar estos conceptos en el ordenador, cuya velocidad de resolución será infinitamente superior a la nuestra manual.

Para el cálculo del Máximo Común Di-

visor (M.C.D.) de dos números nos basaremos en un método matemático estudiado y demostrado que simplifica mucho el cálculo (de otra forma tendríamos que hacerlo mediante el estudio clásico de descomposición en factores primos y posterior estudio). Este método consiste en dividir el mayor por el menor; si el resto de esta división es cero, el menor es el M.C.D. de los dos. En caso contrario, entramos en un proceso reiterativo, en el cual repetimos la operación de la manera siguiente:

El resto de la división anterior pasa a ser el divisor de la próxima y el divisor de la división anterior pasa a ser el dividendo de la próxima. Este proceso concluye cuando la división sea de resto nulo y en este momento el divisor de esta división es el M.C.D. de los dos números.

Para el cálculo del Mínimo Común Múltiplo (M.C.M.) de dos números utilizaremos un nuevo método matemático, que consiste en emplear la expresión: M.C.M. = A*B/M.C.D., siendo A y B los números de los cuales queremos saber su M.C.M.

```
10 REM ************************
20 REM *          CALCULO       *
30 REM *            DEL         *
40 REM *  MAXIMO COMUN DIVISOR *
50 REM *          Y DEL         *
60 REM * MINIMO COMUN MULTIPLO *
70 REM ************************
80 REM
90  REM ******************************************
100 REM * VALIDO PARA AMSTRAD,COMMODORE,IBM y MSX *
110 REM ******************************************
120 CLS :REM * PONER PRINT CHR$(147) EN COMMODORE *
130 INPUT "DAME DOS NUMEROS";A,B
140 CLS :REM * PONER PRINT CHR$(147) EN COMMODORE *
150 LET E=A:LET F=B
160 PRINT TAB(18);"MENU"
170 PRINT TAB(10);"1.MAXIMO COMUN DIVISOR"
180 PRINT TAB(10);"2.MINIMO COMUN MULTIPLO"
190 INPUT "ELIJE OPCION";T
200 IF T<1 OR T>2 THEN PRINT "ELIJE 1 o 2":GOTO 100
210 ON T GOSUB 300,400
220 INPUT "DESEAS CONTINUAR (S/N)";A$
230 LET A$=LEFT$(A$,1)
240 IF A$="S" OR A$="s" THEN GOTO 120
250 END
300 REM ************************************
310 REM * CALCULO DEL MAXIMO COMUN DIVISOR *
320 REM ************************************
330 GOSUB 1000
340 PRINT "EL MAXIMO COMUN DIVISOR DE ";E;"y";F;"ES:";B
350 RETURN
400 REM ************************************
410 REM * CALCULO DEL MINIMO COMUN MULTIPLO *
420 REM ************************************
430 GOSUB 1000
440 LET MCM=F*E/B
450 PRINT "EL MINIMO COMUN MULTIPLO DE ";E;"y";F;"ES";MCM
460 RETURN
1000 REM ***************************
1010 REM * SUBRUTINA PARA CALCULOS *
1020 REM *   INTERMEDIOS COMUNES   *
1030 REM *    A AMBOS PROCESOS     *
1040 REM ***************************
1050 LET C=A/B
1060 LET R=A-INT(C)*B
1070 IF R<>0 THEN LET A=B:LET B=R:GOTO 1040
1080 RETURN
2000 REM ********************************
2010 REM * MODIFICACIONES PARA SPECTRUM *
2030 REM ********************************
2040 REM *   205 LET R=100*T+200         *
2050 REM *   210 GOSUB R                *
2060 REM *   230 LET A$=A$(1 TO 1)      *
2070 REM *   250 GOTO 9999              *
2080 REM ********************************
```

Comentario del programa

El programa que a continuación se presenta calcula el Máximo Común Divisor (M.C.D.) y el Mínimo Común Múltiplo (M.C.M.) de dos números cualesquiera.

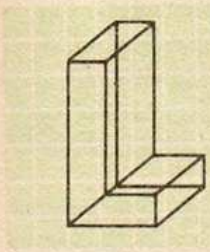
En la primera parte del programa se presenta un menú con dos opciones, de tal forma que se puede elegir calcular el Máximo Común Divisor o el Mínimo Común Múltiplo de dos números (que previamente han sido introducidos por el teclado). De esta forma hemos creado dos subrutinas, una para cada cálculo. La subrutina para el cálculo del Máximo Común Divisor comienza en la línea 300 y la del cálculo del Mínimo Común Múltiplo comienza en la línea 400.

Hemos abierto otra subrutina en la línea 1000, debido a que el cálculo del Máximo Común Divisor y del Mínimo Común Múltiplo son similares en sus pasos, ya que los hacemos por los métodos descritos anteriormente, y esto implica que para calcular el Mínimo Común Múltiplo tenemos que calcular previamente el Máximo Común Divisor.

En la línea 150 hemos asignado los valores de A y B (los números de los cuales tenemos que hallar el M.C.D. y el M.C.M.) a dos nuevas variables; esto es debido a que al realizar el cáiculo los valores de A y B cambian, y si no asignásemos estos valores a otras dos variables, no podríamos presentar en pantalla el resultado final.

El resto del programa son sentencias fáciles de entender y suponemos que el lector familiarizado con el lenguaje BASIC no tendrá dificultades de comprenderlo.

Los sistemas de numeración a través de la historia



OS sistemas de numeración han estado siempre influenciados por las culturas que los han utilizado. Las costumbres y necesidades de los pueblos han sido siempre el caldo de cultivo sobre el que se han desarrollado las artes y la sabiduría de éstos.

El que un sistema de numeración sea avanzado y pueda utilizarse en la resolución de problemas aritméticos con un cierto grado de comodidad tiene una importancia enorme; su alcance es mucho mayor que el que a primera vista puede vislumbrarse. Sin un sistema de numeración adecuado no pueden resolverse problemas matemáticos, y por ello, en algunas culturas esta ciencia no se ha desarrollado tanto como cabía de esperar. Y sin las Matemáticas, otras ciencias, como la Física, Astronomía, etc., que se apoyan en ella, sufren un desarrollo sustancialmente menor.

Consideraremos elementalmente dos tipos principales de sistemas numéricos: el de tarjas, que para indicar la cantidad repite el mismo signo el número de veces que sea necesario, y el de código, en el que cada código simboliza una determinada cantidad (cantidades diferentes tienen códigos diferentes).

El primer sistema se utilizaba 3000 años antes de Cristo, en las dinastías egipcias. Más adelante, basándose en la mano, inventaron otro símbolo para indicar el 10. El sistema se volvió a ampliar posteriormente, creándose otro símbolo diferente para el 100 y para el 1000. En el papiro de Ahmes, datado en el 1600 antes de Cristo, pueden verse los distintos códigos utilizados para representar las cantida-

Número	Babilónicos	Chinos	Egipcios	Romanos	Griegos
1	V	—	|	I	A
2	VV	=	||	II	B
3	VVV	≡		III	Γ
4		𝍠		IV	Δ
5		𝍡	𓂭	V	E
0	>	𝍢	∩	X	I
T	V (60)	𝍣 (20)	𓆼 (1.000)	D (50)	P (100)
R		𝍤 (30)	𓆐 (10.000)	C (100)	Σ (200)
O		𝍥 (400)	𓂭 (100.000)	M (1.000)	Φ (500) (900)
S					

des. Son indicativos de la cultura de la época: el uno era una marca elemental (I); para indicar el diez se dibujaba un símbolo parecido a un talón, 100 era una cuerda, elemento utilísimo en esa civilización, etc. Más tarde también apareció otro símbolo indicador del 5 (basándose en los cinco dedos de la mano).

El sistema babilónico es, desde luego, el más original de todos los sistemas de numeración. Utilizaban una base enorme... 60. Los símbolos para representar las cantidades eran sencillamente muescas, o marcas, muy parecidas unas a otras, en las que tenía significado especial el tamaño y la dirección del trazo (escritura cuneiforme).

El concepto de representación numérica posicional fue introducido también por las civilizaciones mesopotámicas. Este avance es de una importancia enorme, ya que simplifica los cálculos considerablemente, y constituye un acercamiento al sistema de numeración moderno.

Los babilonios eran grandes comerciantes, por lo que se interesaron (y desarrollaron) los sistemas de numeración, hasta tratar incluso números fraccionarios. Aunque puede parecernos que la base seleccionada es absurda por enorme, pensemos que hoy en día esa base es utilizada en la medición de tiempos y ángulos.

El problema principal de la notación babilónica consistía en que el símbolo 𒐕, que indicaba el 1, también representaba el 60, y todas las potencias de 60 ($60 \times 60 = 3600$), de forma que el número de que se tratara debía deducirse del contexto. Lo mismo ocurría con el símbolo que indicaba el 10 —representación además de 10 por cualquier potencia de 60.

En investigaciones recientes se ha determinado que parece que llegaron incluso a representar el concepto del cero (siguiente paso trascendental para modernizar el sistema). Sin embargo, no han aparecido pruebas suficientes de que se haya utilizado el símbolo en cálculos aritméticos.

Los chinos, otro pueblo antiquísimo, han utilizado en su largo recorrido histórico diversos sistemas de numeración en distintas épocas. En un principio utilizaron un sistema binario, semejante al tantas veces utilizado en los primeros ordenadores, y que sigue vigente. Sin embargo, más tarde pasaron al sistema actual en el que las cantidades se expresan

¿Sabía usted que...

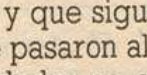
En los enormes ordenadores de los años cincuenta trabajaban científicos interesados en la velocidad de cálculo y la enorme cantidad de información que podían procesar las máquinas. Las disciplinas interesadas eran de todo tipo, Astronomía, Medicina, Química, etc.

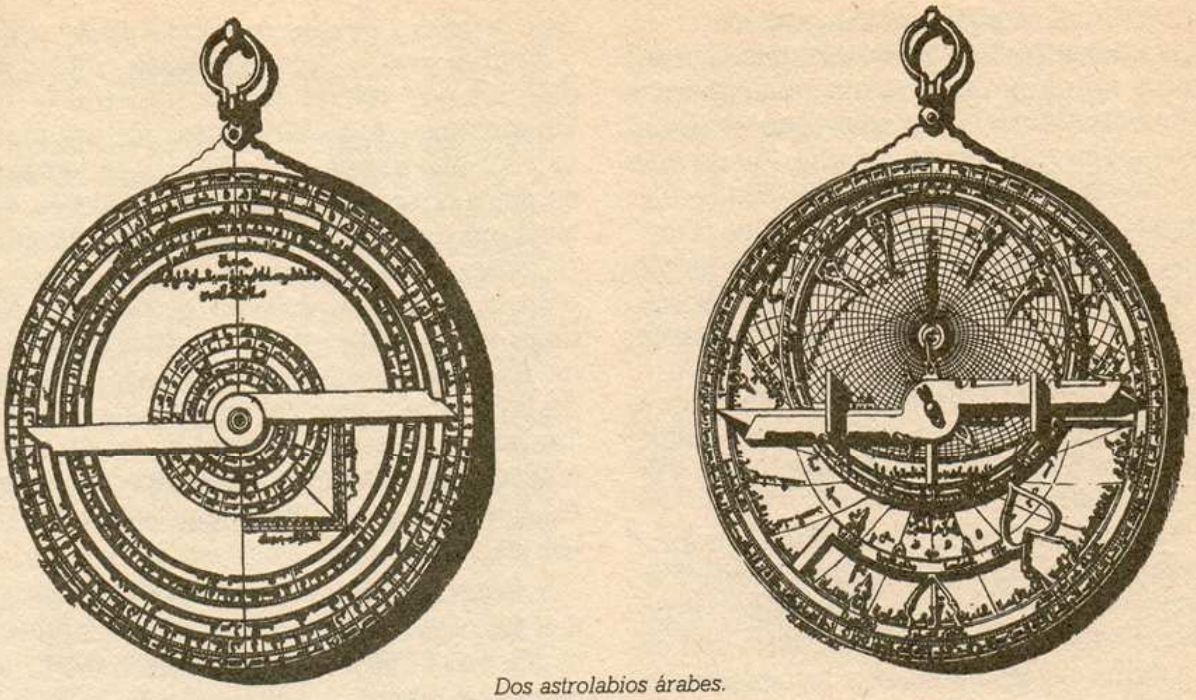
Entre 1954 y 1956 se investigó en un ordenador la estructura molecular de las sustancias producidas por organismos vivos. Entre las moléculas estudiadas se encontraba la Vitamina B_{12}. El mérito de estas investigaciones sobre sustancias orgánicas debe repartirse entre las Universidades de Oxford, Princeton y Los Angeles. También colaboró la Universidad de Manchester (con el Markl, de Ferranti).

En 1926 los médicos habían descubierto que la anemia perniciosa mejoraba al administrar al paciente extracto de hígado. Varios años después, y tras larga labor de investigación de laboratorio, los científicos localizaron por fin el factor activo: la vitamina B_{12}.

Pero, para seguir adelante, hacía falta conocer a fondo el factor descubierto; su fórmula empírica es C68,H88,P,Co. (Como se ve, lleva un metal en la molécula.) Sin embargo, en Química Orgánica la fórmula empírica tiene una importancia relativa, y la que interesa realmente es la disposición espacial de la molécula, aspecto fundamental para descubrir sus propiedades. Para llegar a conocer esta disposición fueron necesarios numerosísimos cálculos (además de la labor experimental). Finalmente, se llegó a conocer con una cierta aproximación. (En los ordenadores modernos se han seguido sistemas de trabajo similares, llegándose a conocer la molécula con un grado de exactitud enorme.)

código que lleva un sufijo indicador de decenas, centenas, etc.

Las civilizaciones precolombinas, en especial la maya, utilizaban también una notación posicional. Sus estudios de Astronomía y del calendario la hacían necesaria. Era más avanzada que la mesopotámica, pero tenía unas bases de numeración muy poco prácticas (5, 20 y 360-5 dedos de la mano, 20 dedos en total y 360 días del año maya). Fueron los primeros en darse cuenta de la importancia del cero, adjudicándole además un valor de posición. El símbolo que representaba el cero es una especie de ojo. Por ejemplo, para indicar 60 se escribirá un  ... (3 grupos de 20). Es increíbe cómo llegaron a avanzar tanto



Dos astrolabios árabes.

sin sufrir influencias de ninguna otra civilización hasta 1492.

A través de los siglos, los griegos utilizaron dos representaciones diferentes. Las dos eran de código. Hasta el siglo III utilizaban la notación romana, desde luego nada práctica. A partir de esa fecha pasaron a utilizar la notación alejandrina, que tampoco mejoraba mucho el problema. Esta notación utilizaba las letras del alfabeto (que eran veinticuatro) y les asignaba un valor: las nueve primeras indicaban los dígitos del 1 al 9, las nueve restantes del 10 al 90 (de diez en diez), y las restantes del 100 al 900. Como el alfabeto sólo disponía de 24 símbolos, tuvieron que adoptar otros tres símbolos más. El sistema no es en absoluto práctico, pero a pesar de ello los griegos se las arreglaron para ampliarlo hasta poder representar números superiores a 999. Los hebreos han conservado mucho tiempo un sistema parecido, y al utilizar grupos de letras obtenía algunos conjuntos con algún significado. Eso ha dado lugar a las numerosas supersticiones adjudicadas a ciertos números, como, por ejemplo, el 18 (o símbolo de la vida, ya que al escribir este número en caracteres alejandrinos se podía leer «vida» en hebreo. El Imperio de Oriente utilizó este sistema de numeración hasta bien entrado el siglo XV.

Hoy en día no podemos imaginar que los griegos, tan «cartesianos» en su forma de pensar, no tuvieran una idea moderna para representar los números. Cómo es posible que cerebros como el de Pitágoras o Arquímedes pudieran pensar teoremas tan interesantes y no supieran calcular adecuadamente. La explicación a este enigma puede ser que en Grecia el uso del ábaco estaba muy extendido y, por tanto, todos los cálculos se hacían con él, dejando al sistema de numeración un papel de «constatador» de la operación, sin otro valor práctico. Quizá por esta razón los griegos estudiaron mucho más la Geometría que otras ramas de las matemáticas.

Sabemos bastante más sobre el sistema de numeración de los romanos que sobre el de otras culturas. Al principio el cuatro se representaba por cuatro trazos, pero más tarde se pasó a utilizar el símbolo que hoy día conocemos. Evidentemente este sistema de numeración no era en absoluto práctico, ya que lo único que indicaba era que una determinada cantidad era la suma de otras. No era una notación posicional (el orden de los códigos no es en absoluto importante, y sólo tiene una función estética), y además no conocían el concepto de cero.

Nuestro sistema de numeración actual es una herencia de los hindúes. Realmente lo recibimos de los árabes, y lo transmitimos al mundo desde nuestras universidades, pero la idea primitiva era hindú. Los primeros fundamentos de la notación posicional se debieron, como sabemos, a los babilonios, unos 3000 años antes de Cristo. Sin embargo, los hindúes adoptaron este sistema (mejorado en siglos posteriores) aportando una idea genial para el manejo de cifras: el cero. Así, pues, el sistema era posicional y además podía indicar cual-

quier cantidad cómodamente utilizando el concepto del cero. La base seleccionada, desde luego, no es la mejor, pero es aceptable (10). (Es importante hacer notar que el ábaco no tuvo mucha vigencia en la India. Puede que las mejoras introducidas en la notación se debieran a la necesidad de flexibilizar los cálculos, que se efectuaban sin ábaco.) Los indios se interesaron mucho por las matemáticas, aunque las disciplinas que experimentaron mayores avances fueron el Algebra y la Aritmética. Aparecieron algunos tratados sobre estas materias, y tienen especial importancia unos tratados de Aritmética del astrónomo y filósofo Brahmagupta.

El imperio musulmán tuvo una enorme importancia hacia el siglo VII. En esta época el comercio con Oriente era floreciente. Además, estos siglos coindicieron en parte con un auge enorme de las ciencias y las artes en la India. Como el intercambio comercial entre la India y el imperio musulmán era grande, los comerciantes volvían a Bagdad, capital del imperio, aportando los conocimientos adquiridos en aquellas lejanas tierras. En seguida adoptaron el sistema decimal para transacciones, etc. De los indios, estudiaron matemáticas y astronomía. Estudiaron las obras de Brahmagupta. Alkarsimi fue el matemático de más relieve. Introdujo el estudio de álgebra y fue uno de los principales vehículos de diseminación del nuevo sistema decimal.

Sin embargo, nuestro sistema decimal no fue introducido al resto del mundo por los árabes de oriente. Fueron las Universidades de Córdoba y Sevilla, que durante los siglos X y XI eran lo más avanzado de la cultura mundial, las que realizaron esa importante tarea. Sin embargo, era difícil aprender esa cultura, ya que los árabes no permitían el acceso de alumnos cristianos en las Universidades, y sólo consiguieron introducirse un pequeño número, «abrazando temporalmente» la nueva religión, y esto bien entrado el siglo XII. Entre ellos, el Papa Silvestre II, naturalmente cuando sólo era Gerbert de Aurillac, y Adelardo de Bath, que tradujo el álgebra de Alkarsimi. También Leonardo Fibonacci, célebre matemático, que publicó el Liber Abaci donde hacía hincapié en la utilización del nuevo sistema de numeración.

Sin embargo, el nuevo y práctico sistema de numeración no tuvo en absoluto aceptación en la Europa medieval. Sólo fue aceptado a nivel científico a partir del siglo XV, y tardó todavía un siglo más en ser adoptado por el pueblo.

La notación fraccionaria apareció más tarde, en el siglo XVI, y el uso de la coma decimal quizá unos años antes. Al mismo tiempo, aparecieron tratados de aritmética, simplificando enormemente los procedimientos.

La aparición de la numeración indoarábiga contribuyó a la caída en desuso del ábaco, ya que los cálculos podían hacerse en papel, quedando constancia de ellos, lo que no ocurría con el ábaco. Con el nuevo sistema se podían realizar comprobaciones, además de crear pequeños «dossiers» con las cuentas, inspeccionables en cualquier momento.

El auge de la navegación también tuvo importancia en el avance de las matemáticas, y los sistemas de numeración, y posteriormente las máquinas de calcular. Las tablas de navegación no eran fiables, y era importante rehacerlas, ya que muchos barcos se perdían por esta causa.

Sin embargo, en el siglo pasado comenzó a utilizarse (sólo para ciertos fines) una notación nueva (en realidad, la habían utilizado los chinos, e incluso otros hombres de ciencia como Pascal y Leibniz se habían interesado en ella): la notación binaria. La necesidad de este tipo de notación se debió a que representaba perfectamente dos posibles estados de una máquina, facilitando enormemente su programación. Efectivamente, con la era de las máquinas esta notación resultó igualmente buena para representar impulsos eléctricos.

¿Sabía usted que...

En 1953 la empresa de Hostelería Lyons utilizó su ordenador Leo por primera vez. La tarea encomendada era el cálculo de una nómina de personal. Fue todo un éxito. Para 1957, el mismo ordenador producía una nómina semanal para 15.000 personas, calculaba costos y llevaba distintas anotaciones con fines contables. Esa firma es pionera en la utilización de las computadoras. Piénsese que en esas fechas los ordenadores eran enormes, fabricados prácticamente a medida, por lo que su coste era realmente altísimo, siendo casi siempre organismos oficiales los compradores de dichas máquinas. Es increíble que una empresa comercializadora de productos alimenticios, en aquella época, tuviera una visión de futuro tan acertada.

¿Sabía usted que...

¿Sabía usted que la destrucción del buque más temido durante la Segunda Guerra Mundial fue debida en gran medida a la labor de un solo hombre?

El *Bismark*, acorazado de la Marina alemana, era el buque más odiado por los ingleses durante la Segunda Guerra Mundial. Su objetivo era impedir la comunicación marítima entre Inglaterra y América. Inglaterra, imperio marítimo poderoso, decidió enfrentarse al acorazado en el Báltico. Para la acción envió a los cruceros *Hood* y *Prince of Wales*, entre otros. La derrota fue terrible, y los 1500 hombres del crucero *Hood* murieron. Siete días más tarde, y gracias a la labor de desciframiento criptográfico desarrollada por Alfred Knox y su equipo, se pudieron descifrar las órdenes de la comandancia alemana al buque, preparándose con todo cuidado el ataque para destruirlo. La operación fue un éxito.

Alfred Knox, artífice fundamental de esta victoria, se encontraba en un estado físico deplorable. Su muerte le sobrevino tras grandes sufrimientos un año después. Evidentemente, fue un gran golpe para la nación, ya que sólo el hundimiento del *Bismark* suponía un avance considerable en las posiciones marítimas, junto con un ahorro enorme en vidas.

Para pasar de binario a decimal, basta con aplicar la fórmula

$$n_m\, n_{m-1} \ldots n_2\, n_1{}_2 = n_1 \times 2^0 + n_2 \times 2^1 + \ldots + + n_{m-1} \times +2^{m-2} + n_m \times 2^{m-1}$$

La notación binaria no supone un nuevo sistema de numeración, sino una nueva «base» para el sistema general ya utilizado ampliamente (un símbolo para cada valor hasta el número tomado como base, y valor variable en función de la posición del dígito dentro del número).

Modernamente se utilizan diversas bases de numeración (2, 8, 16...) para facilitar los cálculos cuando se manejan datos representados en un ordenador electrónico (es decir, en base 2).

El sistema octal dispone de ocho signos distintos, del 0 al 7. Su peso depende de la posición (n) en que se encuentra, contando de derecha a izquierda, según la fórmula:

Valor del dígito = 8 elevado a n-1.

La utilidad que presenta este sistema se basa únicamente en que el paso de sistema binario a octal es muy sencillo, ya que 8 es igual a 2 elevado a 3.

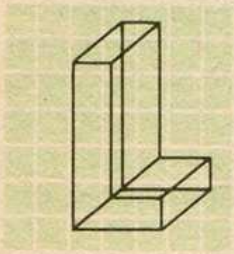
El hexadecimal es también un sistema de numeración muy utilizado. Dispone de 16 dígitos, los diez números arábigos de todos conocidos más las letras del abecedario de la A a la F. El paso de sistema binario a hexadecimal es también inmediato, y la ventaja que presenta este sistema es que en las representaciones utiliza menos dígitos que el sistema decimal.

La fórmula de paso de un sistema al sistema decimal es la siguiente:

Número en base A = 0784.

Valor en base $10 = 4 \times A^0 + 8 \times A^1 + + 7 \times A^2 + 0 \times A^3$

Calendario azteca.



A palabra «robot» viene del checo y significa «trabajo de esclavo». Esta definición puede parecer desagradable, pero aplicada a máquinas nos relaja y disipa nuestros temores internos. Realmente el hombre siempre ha intentado simplificar su trabajo, en suma, trabajar menos o trabajando igual obtener un grado superior de confort. Los primeros humanos inventaron las herramientas y utilizaron animales para realizar los trabajos más pesados. Es más, para poder usar los animales necesitaron domesticarlos, en suma, «enseñarles» la tarea que debían realizar (burros, elefantes, rènos, bueyes, perros, etc.) En suma, los robots sustituyen al hombre en algunas tareas físicas e incluso de trabajo intelectual.

Hoy en día los robots se encuentran en todos los campos de la industria. Están sustituyendo a los humanos en tareas impracticables, peligrosas o bien repetitivas. Los robots pueden aportar mucho a los distintos procesos de producción. Constituyen un sistema de automatización de alto nivel en aquellos sistemas que requieren un mínimo de aportación humana.

Una de las ventajas de los robots modernos es que pueden reprogramarse fácilmente para realizar tareas diferentes de las que realizaban (las diferencias se basan principalmente en cambios en las secuencias de la tarea, cambios de la herramienta final, etc.).

El parque mundial de robots ha aumentado en progresión geométrica en los últimos quince años. En este incremento han sido de vital importancia consideraciones de mejora de la productividad, así como de seguridad en el trabajo (trabajos especialmente peligrosos para el hombre, desde fundiciones, minería, otros trabajos en los que las condiciones ambientales sean especialmente peligrosas para la salud de los trabajadores, etc.).

Los antecesores de los robots han sido ingenios mecánicos, generalmente construidos por relojeros. Su fin primordial era lúdico. Así, sobre todo en los países anglosajones, encontramos muchos museos dedicados exclusivamente a autómatas, algunos de ellos ingeniosísimos y de una precisión digna de encomio, sobre todo considerando los procesos de fabricación de la época (casi siempre manuales).

En realidad, las máquinas o artilugios capaces de reproducir ciertas facetas del trabajo humano pueden ser de muchos tipos diferentes:

a) Médicos. Resuelven problemas de excepcional importancia. Son prótesis de todo tipo, que llevan su propio sistema de movimiento y están accionados por estímulos reflejos. Se han desarrollado enormemente en estos últimos años y constituyen un campo importantísimo de estudio.

b) Móviles. Su objetivo principal es desplazarse por el suelo salvando obstáculos de todo tipo. Tienen un interés grande, aunque están en una fase incipiente de desarrollo: pueden constituir un producto importante para sistemas defensivos y en tareas peligrosas para el hombre.

c) Especiales. Se diseñan específicamente para cubrir las necesidades de un objetivo muy concreto. Por ejemplo, salvamentos de todo tipo, tareas muy específicas de minería, rescates, submarinos, etc. Son robots muy especializados que merecen un estudio individualizado. Normalmente, la propia estructura y diseño de la máquina están totalmente influidos por la misión a cumplir.

d) Robots industriales. Son los que nos

interesan en esta pequeña disertación. Llevan a cabo procesos industriales de todo tipo y automatizan tareas variadísimas y de las que nos beneficiamos continuamente en nuestra vida cotidiana.

Sin embargo, se suelen incluir bajo el nombre de robots un conjunto enorme de máquinas que, en realidad, no son tales, a nuestro entender.

En general, cualquier dispositivo capaz de realizar movimientos «semejantes» a los humanos se llama manipulador.

Los manipuladores pueden clasificarse (según el método de control que incluyen) en tres tipos: manuales, automáticos e interactivos.

A. Los manipuladores manuales constan de:

— Elemento maestro (de control) cuya misión es preparar las señales de control o ejecutar los movimientos de control.

— Elemento esclavo (de trabajo) que realiza la acción deseada de acuerdo con las instrucciones recibidas del elemento maestro.

— Elemento de unión, que transmite las señales o movimientos de control desde el elemento maestro al esclavo.

— Elemento terminal, que manipula los objetos o ejecuta los movimientos finales. La forma usual del elemento terminal es de mano, garfio, herramienta, etc.

Un manipulador manual puede ser:

a) Manipulador esclavo, si se limita a reproducir los movimientos de brazo y mano del operador.

b) Manipulador de control remoto. En este tipo de dispositivo el elemento terminal no está unido cinemáticamente a la unidad maestra (como sucede en el manipulador esclavo), sino que cada grado de libertad es controlado separadamente por teclas, manejadores, etc.

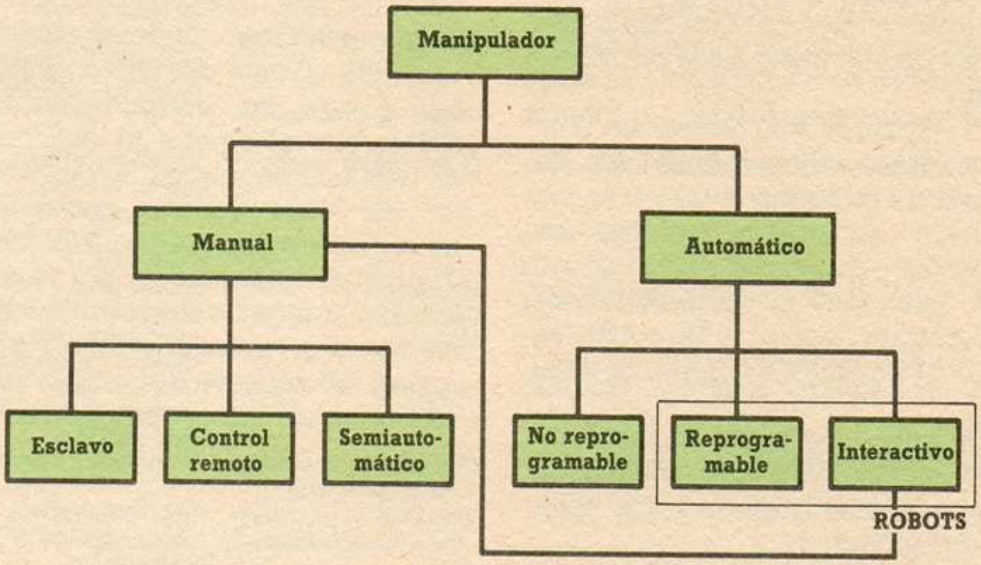


Fig. 1. Clasificación de los dispositivos manipuladores según el método de control que incorporan.

c) Manipulador semiautomático, que incluye un microordenador o calculador específico para transformar las señales de control en los movimientos que deben realizar los elementos esclavos y terminal.

B. Los manipuladores automáticos difieren de los manuales básicamente en que incluyen memoria y no requieren, por tanto, la permanente intervención del operador. Se suelen distinguir dos tipos de manipuladores automáticos: los autooperadores (cuya característica principal es que no son reprogramables) y los robots industriales propiamente dichos (reprogramables y de uso general; válidos, por tanto, para infinidad de tareas con sólo cambiar la secuencia de las instrucciones que se le envían o su contenido).

C. Se consideran usualmente en un grupo diferente (de características más sofisticadas) los manipuladores interactivos. Su propio nombre indica que su característica esencial es que pueden interactuar con el operador. Normalmente, disponen de memoria para realizar ciertas operaciones independientemente de las instrucciones recibidas.

Robótica (1)

Hay casos en que el automatismo es parcial y requiere la intervención del opera-

dor para tomar decisiones en ciertos momentos del desarrollo de la secuencia de instrucciones, y en otros casos la independencia del robot es casi absoluta (naturalmente, una vez recibidas las órdenes descriptivas de las tareas a abordar).

Incluso en ocasiones hay interactuación con el operador en modo conversacional mediante lenguajes de alto nivel, hasta por sintetización de voz.

Según Yu Kozyrev sólo deben ser considerados verdaderos robots industriales los dos últimos tipos de manipuladores.

Por otro lado, este modo de clasificar los manipuladores, en general (y los robots en particular), por su sistema de control, no es el único válido.

Según Angulo y No, entre los robots industriales, que son máquinas multifuncionales, programables, podemos distinguir dos tipos, por orden de complejidad:

a) Manuales. Manejan piezas o herramientas, pero requieren un cierto grado de intervención humana para la culminación del proceso.

b) Secuenciales. Realizan una determinada secuencia de trabajo. Dentro de este tipo pueden distinguirse dos subtipos:

— De secuencia variable, es decir, que se puede alterar según sean las necesidades (más adelante explicaremos cómo se puede alterar la secuencia, bien por programación o mecánicamente, cambiando levas, topes, etc.).

— De secuencia fija, que evidentemente tienen mucho menor interés.

c) De aprendizaje. Este aprendizaje es una especie de programación. Consiste en «enseñar», es decir, hacer que el robot en algún sentido siga la trayectoria que va a realizar y pueda repetirla con posterioridad. (Naturalmente, al ir realizando esa trayectoria de acciones, el robot va «tomando nota» de las coordenadas internas y externas de cada punto del camino; pero más adelante explicaremos esto con más detalle.)

d) De control numérico. A este tipo de robots, o máquinas automáticas, basta con indicarles (programarles o «enseñarles») algunos puntos de la trayectoria, realizando el propio robot las trayectorias entre esos distintos puntos.

Característica	Tipo de robot
Utilidad	Producción / Auxiliar (selección y colocación) / Universal.
Especialización	Especial / Monopropósito / Multipropósito.
Campos de aplicación	Forja / Soldadura / Pintura / Fundiciones / Maquinaria / Revestimientos / Control Automático / Almacenes / Otros.
Sistema de coordenadas	Rectangulares (en el plano o en el espacio) / Polares (en el plano o cilíndrica) / Esféricas.
Número de grados de libertad	Un eje / Dos ejes / Varios ejes.
Capacidad de cargas	Superligeros (hasta 1 kilo) / Ligeros (hasta 10 kilos) / Medios (hasta 200 kilos) / Pesados (hasta 1.000 kilos) / Superpesados (más de 1.000 kilos).
Movilidad	Estacionarios / Móviles.
Ubicación	Integrados / Tipo suelo / Elevados.
Motor(es)	Electromecánico / Neumático / Hidráulico / Compuesto.
Disposición de los motores	Como un bloque único / Como actuadores individuales / Organización compleja.
Ejecución de la secuencia de programa	No flexible / Adaptable / Flexible.
Velocidad y programación del movimiento	Secuencia limitada / Paso a paso / Camino continuo / Algunos puntos / Multipunto.

Fig. 2. Clasificación de los robots industriales.

e) Inteligentes. Son los robots más avanzados. Disponen de diversos sensores muy especializados (por ejemplo, para la visión artificial), que cumplen en cierto sentido la función de los sentidos humanos. Todos ellos llevan algún sistema de inteligencia artificial.

Existe cierto desacuerdo en lo que se refiere a la nomenclatura de los robots, que incide en estadísticas a nivel mundial. Así, los japoneses contabilizaban enormes parques de estas máquinas, ya que incluían en el término robot a todos los tipos antes descritos. Modernamente, la R.I.A. sólo considera como tales robots los de secuencia programable y demás máquinas de automatismos más complejas.

Los robots industriales, que son los que nos interesan aquí, están dedicados fundamentalmente a mejorar la producción.

El diagrama de bloques de un robot industrial (RI) incluye tres grupos de elementos:

A. El sistema de control. Preparado para controlar la operación del robot, almacenar, ejecutar y verificar el programa de control. Consta de lector de programa, memoria, unidad de introducción del programa, transductor (transforma el programa de control en datos inteligibles por los actuadores y moto-

res), retroalimentación, comparador, programa de control y consola de control.

B. Sistema de información. Selecciona, preprocesa y transfiere al sistema de control información, tanto sobre el propio RI (sus unidades y mecanismos, incluido el sistema de control) como sobre las condiciones del entorno. En los RI más completos, suele disponer de tres subsistemas: el de información del entorno, el estado operacional del robot y el subsistema de seguridad del equipo. La información sobre el estado del robot puede contener diversos datos sobre los dispositivos de que consta el robot: dispositivos que evalúan la posición y rapidez del movimiento respecto de un sistema de coordenadas; seguros que evitan el deterioro del equipo o el daño en cualquiera de sus elementos; equipos de diagnosis de averías y de prevención de fallos futuros, etc.

C. Un sistema mecánico que realiza físicamente las tareas encomendadas al robot. Los elementos básicos componentes del sistema mecánico son:

1. Una estructura mecánica articulada (numerosas articulaciones: su número total define los grados de libertad del mecanismo). Este es el manipulador propiamente dicho del robot (en contraposición con los restantes elementos de control, accionadores, etc.).

2. Una fuente de energía que permita que se lleven a efecto los movimientos (motores, sistemas hidráulicos, neumáticos, etc.) que funcionen con energía eléctrica u otros tipos de energía.

3. Un sistema de control de los movimientos (estamos hablando de un motor que mueva la estructura y que esté controlado para que los movimientos sean eficaces y pre-

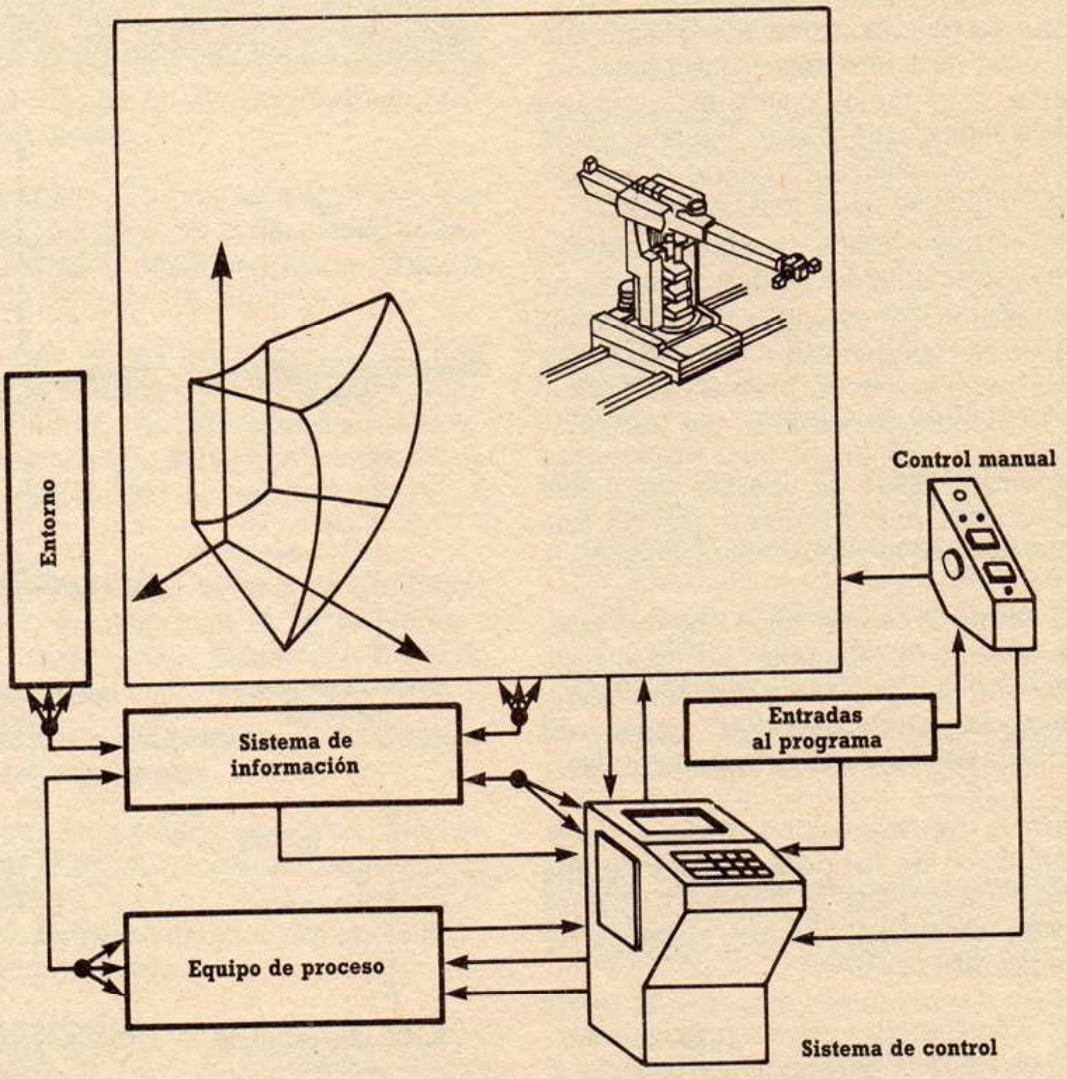


Fig. 3. Diagrama de bloques de un robot industrial.

cisos, tanto en la fase de «enseñanza» como en la de ejecución). Los robots «inteligentes» llevan sensores muy especializados que toman «anotaciones» del entorno exterior. Esas «anotaciones» y las demás órdenes se almacenan en la memoria del dispositivo.

4. Un elemento terminal, diseñado especialmente para la tarea específica que va a desarrollar (garra, pinza, herramienta especializada, etc.).

La estructura articulada es, como antes indicamos, una estructura formada por unos elementos rígidos, metálicos, unidos unos a otros por articulaciones. (Del número de éstas y de sus posibilidades dependerá la flexibilidad de la máquina y, por tanto, el nivel de automatismo que se obtenga.)

Los nombres de esta estructura, y en general de la Robótica, suelen aludir con frecuencia a elementos del cuerpo humano, o de la naturaleza. Así tenemos «enseñar» —programar un robot, brazo, muñeca, etc.

Esta estructura articulada o manipulador del robot lleva una base, que puede ser fija o móvil. Si es móvil, puede estar unida a un sistema motriz que puede llevar correas, raíles, etc., o sencillamente puede ir provista de ruedas o incluso patas móviles o fijas.

Las articulaciones (también llamadas ejes) son fundamentales, ya que permiten los movimientos. Pueden ser giratorias (se suelen representar mediante una R = rotación) o prismáticas (es decir, que se desplazan sobre sí mismas en sentido longitudinal, se representan por una P). Cada una de ellas proporciona un parámetro o grado de libertad del mecanismo, de forma que el número total de articulaciones es igual al número de grados de libertad.

La estructura tiene como objetivo posicionar y orientar el elemento terminal (que puede ser una garra, pinza o bien una herramienta especializada). A todo el conjunto de articulaciones se le denomina cadena cinemática.

Para la representación gráfica, ubicación y definición de las posibilidades de una estructura dada (grados de libertad y tipos de articulación) se utilizan los tres sistemas de coordenadas más usuales:

a) Coordenadas rectangulares o cartesianas. Los ejes son perpendiculares entre sí y la posición sobre ellos se define mediante números. Se puede trabajar en un plano (dos ejes) o en el espacio (tres ejes): estructura de tipo PPP.

b) Coordenadas polares, en que la representación se efectúa fijando un ángulo de giro y un desplazamiento. Se puede utilizar un solo ángulo y dos desplazamientos (estructura de tipo cilíndrico o RPP) o bien dos giros y un desplazamiento (lo que da lugar a un posicionamiento de tipo esférico que responde al esquema RRP).

c) Coordenadas angulares. Se determina un punto mediante dos ángulos en un plano (con estos dos giros y un desplazamiento se «barre» un espacio cilíndrico, RRP) o con tres ángulos (en una disposición de tipo esférico, RRR).

Existen incluso organizaciones más complicadas que disponen de elementos de un tipo y de otro, o que incluyen más articulaciones que las imprescindibles para aumentar la precisión (como es el caso de las articulaciones de tipo Scara, desarrolladas modernamente por la compañía sueca ASEA).

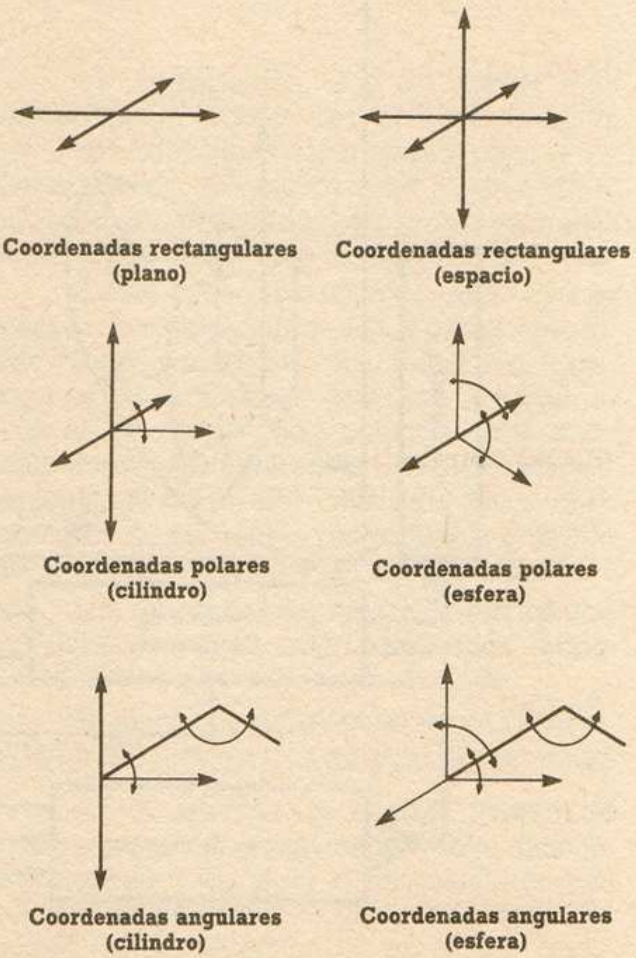


Fig. 4. Movimientos básicos según los sistemas de coordenadas.

TERMINOLOGIA

GLOSARIO DE TERMINOS UTILIZADOS EN ROBOTICA

Armadura. Conjunto de elementos del manipulador, donde se articula el brazo para realizar su misión.

Autooperadores. Manipulador automático no reprogramable.

Balanceo. Uno de los tres movimientos permitidos a la muñeca del robot. Llamado así por similitud con el correspondiente movimiento de un barco o avión. Movimiento de giro alrededor de un eje longitudinal (horizontal) del buque.

Brazo del robot. Una de las partes del manipulador. Soportado en la base de éste, sostiene y maneja la muñeca (donde va instalado el útil de toma de objetos).

Cabeceo. Uno de los tres movimientos permitidos a la muñeca del robot. Llamado así por similitud con el correspondiente movimiento de un barco o avión. Movimiento de giro alrededor de un eje transversal al buque.

Cadena cinemática. Conjunto de elementos mecánicos que soportan la herramienta o útil del robot (base, armadura, muñeca, etcétera).

Cartesianas, coordenadas. (Ver **Coordenadas rectangulares.)**

Cinemático. En robótica se utiliza este término para referirse a los accionamientos de un manipulador que suponen una unión física directa entre los mandos del operador y el elemento terminal.

Control analógico. La información de control es dada en forma de valores (variables de un modo continuo) de ciertas cantidades físicas (analógicas).

Control numérico. Los datos están representados en forma de códigos numéricos almacenados en un medio adecuado. Se llaman también sistemas de punto a punto, o de camino continuo.

Control remoto, manipulador de. Aquél en que cada grado de libertad está actuado por un dispositivo independiente, con lo que puede no estar unido cinemáticamente al actuador del operador.

Coordenadas. Sistema de ejes para el posicionamiento de un punto en el plano o en el espacio. Pueden ser: *a)* Angulares. Si la referencia de un punto se hace mediante la definición de ángulos a partir de los ejes (origen de los ángulos). *b)* Polares. Se establece un punto mediante la indicación de un ángulo y un valor escalar (numérico). *c)* Rectangulares. Cuando los puntos están definidos por varios números (dos o tres).

Eje. Cada una de las líneas según las cuales se puede mover el robot o una parte de él (algún elemento de su estructura). Pueden ser ejes o líneas de desplazamiento longitudinal sobre sí mismo (articulación prismática) o ejes de giro (rotación). Cada eje define un «grado de libertad» del robot.

Elemento. Cada uno de los componentes de la estructura de un manipulador. Pueden ser elemento maestro, esclavo, de unión, terminal, etc.

Garra. Una de las configuraciones típicas del elemento terminal de un manipulador. Es un elemento de precisión y potencia medias.

Giro. Movimiento básico de un manipulador. (Véase **Eje.**)

Grado de libertad. Cada uno de los movimientos básicos que definen la movilidad de

un determinado robot. Puede indicar un movimiento longitudinal o de rotación. (Véase **Eje.**)

Hidráulico. Es un manipulador cuya energía de movimiento viene proporcionada por un fluido que presiona émbolos. Se consigue una gran potencia en la operación del robot, aunque se pierda precisión.

Muñeca. Dispositivo donde se articula el elemento terminal (garfio, pinza, etc.) de un manipulador. Es un elemento básico para la definición de la flexibilidad y precisión del manipulador. Las posiciones del elemento terminal vienen dadas por los grados de libertad de la muñeca.

Manipulador. En general, cualquier dispositivo mecánico capaz de reproducir los movimientos humanos para la manipulación de objetos. En particular, suele referirse a los elementos mecánicos de un robot que producen su adecuado posicionamiento y operación.

Neumático. Es un manipulador cuya energía de movimiento viene proporcionada por un sistema de aire comprimido (conductos que lo contienen, émbolos de empuje, sistema compresor, etc.).

Paso a paso, motor. Motor eléctrico que gira un número exacto de grados al recibir una adecuada secuencia de comandos de control. Son motores sumamente precisos.

Pinza. Una de las configuraciones características del elemento terminal de un manipulador o de un robot. Se articula con el resto de la estructura a través de la muñeca.

RI. Siglas utilizadas para referirse a un robot industrial.

Robot. Manipulador mecánico, reprogramable y de uso general. En particular, los robots utilizados en la fabricación o procesamiento de objetos (los más numerosos) se suelen llamar robots industriales.

Rotación. Movimiento básico en un manipulador. (Véase **Eje.**)

VOCABULARIO DE INFORMATICA

Caracteres, llenado de. Introducción de caracteres sin valor en un medio de almacenamiento. Normalmente se utiliza para borrar datos no deseados.

Caracteres, impresora de. Dispositivo que va imprimiendo carácter por carácter, en contraposición a impresora de líneas.

Caracteres, reconocimiento de. Identificación de caracteres gráficos o audibles por medios automáticos.

Caracteres, subconjunto de. Selección de un grupo de caracteres que tienen alguna característica común.

Característica. (Véase **Exponente.**).

Cargador. Programa que sirve para cargar en memoria otros programas. Al conectar el ordenador el programa que realiza la primera carga de forma automática se llama BOOTSTRAP o IPL (Initial Programs Loader).

Cargar. Acto por el que se introduce en la memoria uno o más programas o datos.

Carriage return. Operación por la cual el siguiente carácter se imprime o visualiza en la primera posición de la línea siguiente.

Carriage return, carácter. Al pulsar la tecla, la impresión del texto (en pantalla o impresora) pasa a la primera posición de la línea siguiente. Abreviadamente, CR.

Carro. Parte móvil de una máquina de escribir (rodillo donde se inserta el papel). Aunque hoy en día las impresoras tienen una configuración distinta, y el rodillo no se desplaza, sino que lo hace la cabeza de escritura, por inercia se sigue llamando igual al sistema de escritura.

Carro, vuelta del. Efecto de pasar a escribir en la línea siguiente: el rodillo avanza una línea y la cabeza de escritura «vuelve» a la primera posición de impresión.

Carro, carácter de vuelta de. Carácter que, enviado a la impresora, provoca que se realice una «vuelta de carro».

Carry (arrastre). Caracteres o dígitos producidos durante una operación aritmética. Como resultado de la misma, se ven afectados uno o más dígitos o caracteres de orden superior.

Cartridge. (Véase **Cartucho.**) Sistema extraíble de almacenamiento de los datos (cintas o discos magnéticos).

Cassette. Dispositivo de cinta magnética para almacenar datos. Sirve también para grabar música. Es barata y menos fiable que otros sistemas de almacenamiento.

Catálogo. Lista descriptiva de elementos. La información que aparece en él debe ser suficiente para acceder a dichos elementos. Puede ser sinónimo de directorio.

Catódicos, tubo de rayos. (CRT). Dispositivo para la presentación de los datos que utiliza un rayo de electrones para barrer la pantalla.

CCITT. Siglas de Comité Consultivo Internacional de Telegrafía y Telefonía. Reglamenta las transmisiones en general (incluida la transmisión de datos).

CDA. Conversor Digital/Analógico. Dispositivo que convierte señales digitales (o discontinuas) en analógicas (o continuas). (Véase **CAD.**)

Celda. Pequeña unidad elemental de almacenamiento. Puede ser binaria, decimal, etc.

Cerrada, subrutina. Subrutina que puede almacenarse y conectarse a otra rutina en más de un punto. En contraposición a rutina abierta.

Cibernética. Rama del conocimiento que estudia las comunicaciones y el control en organismos vivos y máquinas. Sus primeras teorías y estudios se debieron a N. Wiener, y fueron desarrolladas a mediados de este siglo.

Cíclica, carácter de control de redundancia. Un carácter usado en un código cíclico modificado, para detección y corrección de errores. Suele representarse por las siglas CRC.

Cíclico, desplazamiento. Un desplazamiento mediante el cual sale al exterior algún dato de un registro de almacenamiento y se reintroduce por el otro extremo, como en un lazo cerrado.

Cíclico, almacenamiento. Ver **Circulante, registro.**

Ciclo. Intervalo de tiempo necesario para completar un determinado fenómeno o acontecimiento. Puede referirse a operaciones que se repiten con regularidad.

Ciclo de máquina. Tiempo necesario para que el ordenador realice una tarea interna a nivel elemental.

Circuito. En comunicaciones, camino de enlace entre dos puntos, que incluye un canal de «ida» y otro de «vuelta». En contraposición a canal. (Véase **Canal.**)

Circuito integrado. Circuito electrónico realizado completamente en una sola pieza (con un componente único llamado chip). Todos los elementos del circuito (transistores, resistencias, condensadores, etc.) se integran en un solo bloque mediante técnicas adecuadas de difusión (normalmente en varias capas). Según la cantidad de elementos (o componentes) que incluya, se dice que está hecho en MSI (Medium scale integration - Integración a media escala), LSI (Large scale integration - Integración a gran escala), VLSI (Very large integration - Integración a muy gran escala) e incluso SLSI (Super gran escala).

Circulante, registro. Registro en el que los datos que se obtienen en uno de sus extremos se reintroducen en el otro, como si se tratara de un bucle cerrado.

Clave. *a)* Campo utilizado para el control de las operaciones de manejo de datos (clave de búsqueda, clave de selección, etc.) *b)* Llave o dispositivo de control para limitar el acceso a unos datos, a un fichero o a un proceso.

Clear. Sirve para liberar (mediante borrado) una o más posiciones de un sistema de almacenamiento. Normalmente se trata de introducir en esas posiciones ceros o espacios.

Clear, área. En reconocimiento de caracteres, área específica que debe mantenerse libre de impresión o de cualquier otra marca no relacionada con la lectura.

Clock (reloj). *a)* Dispositivo que genera señales periódicas. Se utiliza para sincronizar dos o más señales. *b)* Dispositivo que mide y proporciona la hora. *c)* Registro cuyo contenido varía a intervalos regulares, de forma muy parecida a la medida del tiempo.