

# Ordeno y Aprendo



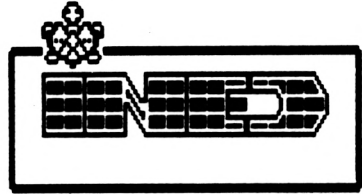
con

**AMSTRAD**

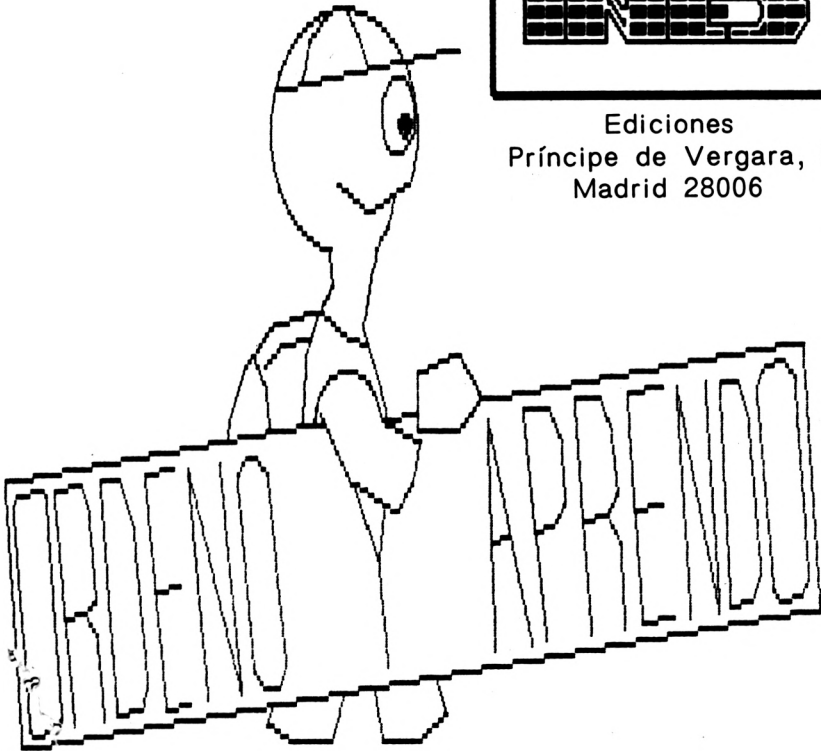
para EGB







Ediciones  
Príncipe de Vergara, 89  
Madrid 28006



Autor: J.L.Carralero

Equipo INED.S.A.

---

Material didáctico:

Curso autodidáctico por ordenador  
Libro de programas prácticos

Derechos reservados INED.S.A. Tlf: 2 62 67 02/03

## Prólogo.

La gran variedad de revistas, libros y manuales aparecidos recientemente sobre informática, hacen que el estudio y aprendizaje de esta materia se pierda en un sinnúmero de teorías y técnicas, sin llegar a encauzar la acción educativa de las mismas en el sentido teórico y práctico necesario en los Centros Docentes.

La acción pedagógica enfocada hacia los distintos niveles de enseñanza, representa uno de los mayores problemas a la hora de planificar un proyecto educativo, concreto y eficaz para la consecución de determinados objetivos.

ORDENO Y APRENDO (I), nace no solamente como un curso de informática, sino que, como elemento de apoyo y enriquecimiento de la actividad docente en otras materias, viene a centrar sus esfuerzos como nexo globalizador que permita un mayor aprovechamiento del alumno durante las etapas de su aprendizaje.

El deseo de ofrecer una informática práctica y asequible, a la vez que agradable en su estudio, nos ha hecho buscar un justo término entre la formación teórica y la práctica en un mismo texto, centrandolo en un objetivo en una mayor y más eficaz acción pedagógica que fomente la creatividad del alumno, haciéndolo partícipe de su propio aprendizaje.

El modelo de aprendizaje se configura en cada unidad bajo los siguientes apartados:

A. CONOCER. Area teórica.

- Aborda el conocimiento teórico necesario para la asimilación de los conceptos básicos y su aplicación en las técnicas de trabajo.

B. PRACTICAR. Area de acción práctica.

- Enfocada al dominio y manejo del ordenador mediante la utilización del lenguaje BASIC de programación.

Esperamos que ORDENO y APRENDO (I) ayude a profesores y alumnos a hacer del aprendizaje de la informática una herramienta útil y productiva en sus estudios, y que su utilización en el trabajo y momentos de ocio les permita dar una nueva dimensión a su capacidad creativa.

© José Luis Carralero Pereira  
Teléfono 262 67 02 / 03  
Diciembre 1985  
ISBN: 84-398-5578-8  
Depósito legal: M. 43.709-1985  
Imprime: Hijos de E. Minuesa, S. L.  
Ronda de Toledo, 24 - Madrid

**Indice**

UNIDAD 1 LA INFORMATICA ..... 9

CONOCER. 1 Historia de la Informática. 2 El Ordenador. 3 Hardware y Software. 4 La informática y el ordenador. 5 Las instrucciones.

PRACTICAR. 1 Nuestro ordenador: teclado, pantalla y cassette. 2 Primeros comandos.

UNIDAD 2 EL ORDENADOR ..... 33

CONOCER. 1 Partes del ordenador. 2 Memoria central. 3 Unidad central. 4 Unidad Aritmético Lógica. 5 Unidad de control. 6 La instrucción PRINT.

PRACTICAR. 1 Uso de la instrucción PRINT. 2 Memoria libre: instrucción FRE(0).

UNIDAD 3 LA INFORMACION ..... 49

CONOCER. 1 Los sistemas de numeración. 2 Conversión de bases de numeración. 3 Elementos del lenguaje BASIC. 4 Las constantes y las variables. 5 Reglas en el empleo de las variables. 6 Asignación de variables.

PRACTICAR. 1 Prácticas con sistemas de numeración. 2 Utilización de variables.

## Índice

### UNIDAD 4      LOS ALGORITMOS ..... 65

CONOCER. 1 Los Algoritmos. 2 Los Lenguajes de Programación. 3 El Programa del Ordenador. 4 Estructura de un programa. 5 El Programa BASIC. 6 La Instrucción INPUT.

PRACTICAR. 1 Mi Primer Programa. 2 Ejecución de un Programa. Comando RUN. 3 Uso de la Instrucción INPUT.

### UNIDAD 5      REPRESENTACION GRAFICA ..... 83

CONOCER. 1 Los Organigramas. 2 Simbología de los Organigramas. 3 Utilización de los Organigramas. 4 La Instrucción GOTO. 5 Experimento Nro. 1: La Mesa Incontrolada. 6 Los Contadores.

PRACTICAR. 1 Codificación de Organigramas. 2 Codificación del Experimento Nro. 1 : La Utilización de Contadores.

### UNIDAD 6      LAS BIFURCACIONES ..... 105

CONOCER. 1 ¿Qué es una Bifurcación?. 2 La Instrucción IF...THEN. 3 Experimento Nro 2 : EL SER EXTRAÑO.

PRACTICAR. 1 Codificación del Experimento Nro 2.

## Indice

UNIDAD 7      LOS BUCLES ..... 127

CONOCER. 1 Los Bucles.2 La Instrucción FOR...NEXT.  
3 Experimento Nro.3: Proyecto inter-espacio.

PRACTICAR. 1 Codificación del experimento Nro. 3.

UNIDAD 8      OTRAS INSTRUCCIONES ..... 153

CONOCER. 1 La Instrucción READ/DATA. 2 La ins-  
trucción REM. 3 Experimento Nro. 4: Contestador me-  
teorológico.

PRACTICAR. 1 Codificación del experimento Nro. 4.  
2 Comando EDIT.

UNIDAD 9      TABLAS DE UNA DIMENSION ..... 171

CONOCER. 1 ¿Qué es una tabla?. 2 Dimensionamiento  
de tablas. 3 El Comando AUTO. 4 La Instrucción  
RENUM. 5 Experimento Nro. 5: Proyecto ordenador.

PRACTICAR. 1 Codificación del experimento Nro. 5.





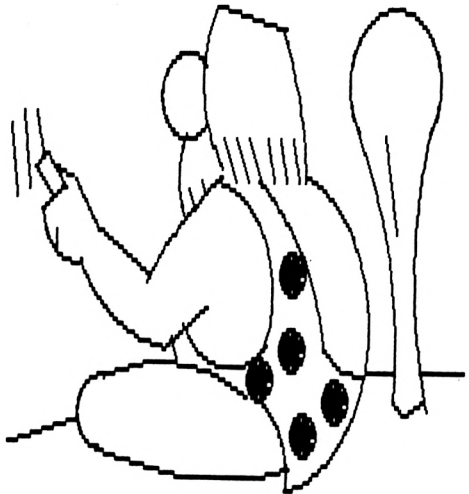
## UNIDAD 1 LA INFORMATICA



Conocer

## 1 Historia de la Informática.

A través de la historia, el hombre ha utilizado diferentes métodos para contar. Así, el hombre primitivo se servía de marcas para contar las piezas cazadas, pintando tantas rayas como ejemplares capturaba. Para pocos elementos, este sistema resultaba bastante útil, pero surgían problemas cuando el número a contar era muy alto. El desarrollo de la civilización trajo consigo grandes



avances en todas las ramas del saber. Dentro de los sistemas de numeración cabe destacar el ROMANO, con el que puede representarse cualquier cantidad, valiéndose de unos caracteres, mediante el sencillo método de sustituir un determinado número de unidades, por los signos correspondientes, (la V equivale a cinco marcas, la X a diez, etc).

El código romano supuso un avance importante, sin embargo era poco natural a la constitución del ser humano. El hombre tiene cinco dedos en cada mano y sus manos son sus principales herramientas para contar. Debido a todo ello, el método que le es más natural es la representación decimal, compuesta de diez dígitos (0-9) y es el sistema que habitualmente utilizamos para realizar nuestros cálculos.

El hombre ha buscado siempre una forma de facilitar su trabajo, tanto manual como intelectual. Así es como los chinos, mediante la utilización del popular ABACO, realizaban complicadas operaciones matemáticas, instrumento que aún en la actualidad, se utiliza en algunos pueblos orientales.

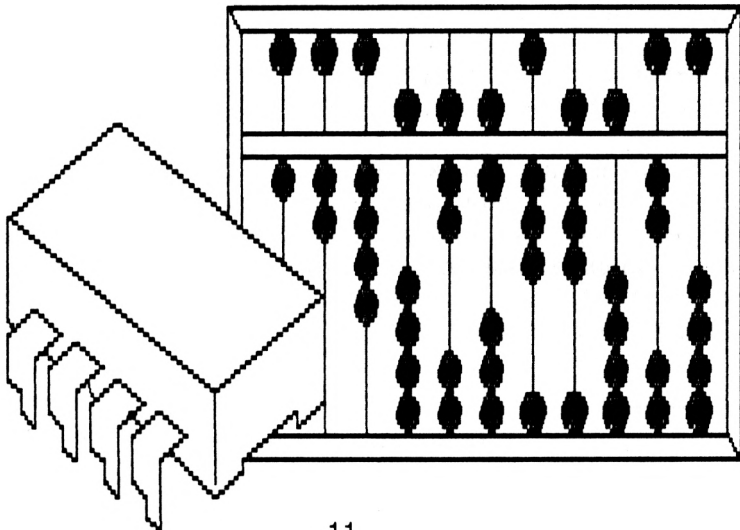
En el siglo VXII, PASCAL diseñó una máquina que mediante la utilización de ruedas dentadas, era capaz de realizar sumas de una manera automática. A este artilugio se le denominó PASCALINA.

Más tarde, en el mismo siglo, LEIBNIZ reformó y amplió la rueda dentada de PASCAL, consiguiendo multiplicar y extraer raíces cuadradas con ella.

En el siglo XIX, CHARLES BABBAGE, idea la llamada Máquina Analítica, a la que se puede considerar como la verdadera precursora de los ordenadores, ya que tenía los mismos elementos que los computadores actuales. Sin embargo, tendrían que pasar cien años para que se construyera un ordenador similar al que este científico inglés había ideado.

A finales del siglo XIX, el norteamericano HERMAN HOLLERITH diseñó las tarjetas perforadas, consistentes en unas fichas con perforaciones las cuales, según su posición, indican la información que contienen.

En 1937, se crea el Primer Ordenador MARK I, fabricado por H. AIKEN, basado en la idea de la Máquina Analítica de Babbage. Con el nacimiento de esta Máquina, comienza la PRIMERA GENERACION de ordenadores, los cuales consumían excesiva energía, y se averiaban con mucha frecuencia. Los típicos de aquella época ocupaban una nave, pesaban alrededor de 30 toneladas, estaban equipados con unas 120.000 válvulas de vacío y tenían una potencia de 50.000 Watios. Estos ordenadores, utilizaban fichas perforadas para introducir programas y datos, así como cintas magnéticas, similares a las empleadas actualmente en los cassettes.



La SEGUNDA GENERACION comienza en el año 1940 y se caracteriza por la aparición del transistor que viene a sustituir a la válvula de vacío, con las ventajas de ser más pequeño, más barato, consumir menos energía y estropearse con menor frecuencia. Utilizaba igualmente fichas perforadas y cintas magnéticas para su funcionamiento. En esta etapa los ordenadores empiezan a ser frecuentes en las empresas para almacenar datos y llevar su gestión, comenzando a utilizarse, así mismo, el disco magnético.

La TERCERA GENERACION se inicia en el año 1964 y se caracteriza por el desarrollo de los llamados "Circuitos Integrados", mejorándose la velocidad de proceso, las operaciones de entrada y salida y los terminales.

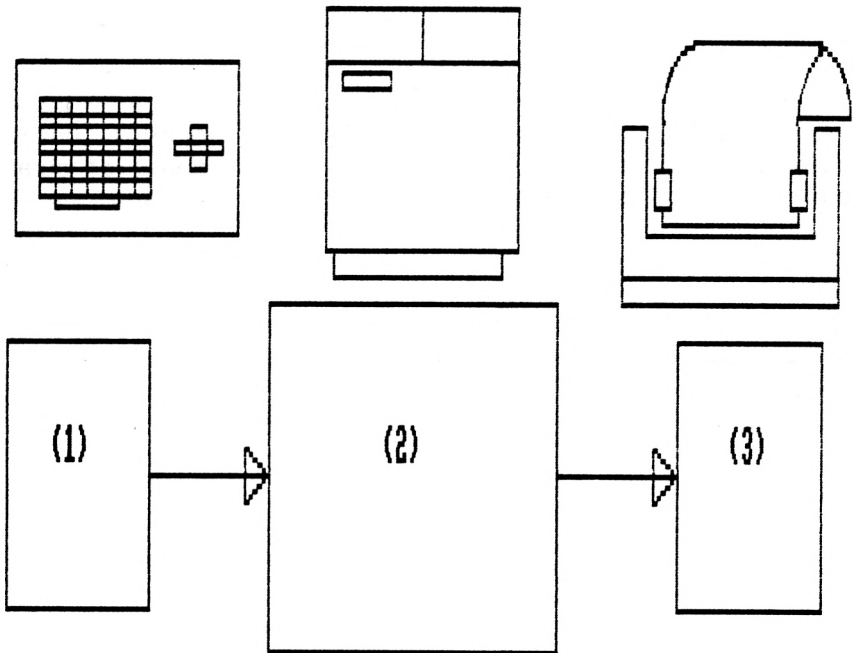
A mediados de 1970, se define la CUARTA GENERACION. El avance tecnológico, permite la fabricación de los LSI, (Circuitos de Gran Escala de Integración). Estos complejos circuitos, llamados CHIPS o pastillas, están constituidos por una pequeña superficie de SILICIO, en la que se insertan un gran número de circuitos simples. La reducción de volumen y precio permite la utilización masiva de los mismos en los ordenadores. Esta es la "generación" en la que nos encontramos hoy en día.

Se prevé, una próxima y nueva generación, (QUINTA GENERACION), que desarrollará el auge de la inteligencia artificial, en la cual los ordenadores se programarán por sí solos e incluso se podrá hablar con ellos. Nos espera un MUNDO DE ASOMBROSOS AVANCES TECNOLOGICOS en este campo.

## 2 El Ordenador

Básicamente, podemos definir un ordenador, como un dispositivo capaz de elaborar información a partir de los datos suministrados, mediante la utilización de un programa de instrucciones.

Fig.1

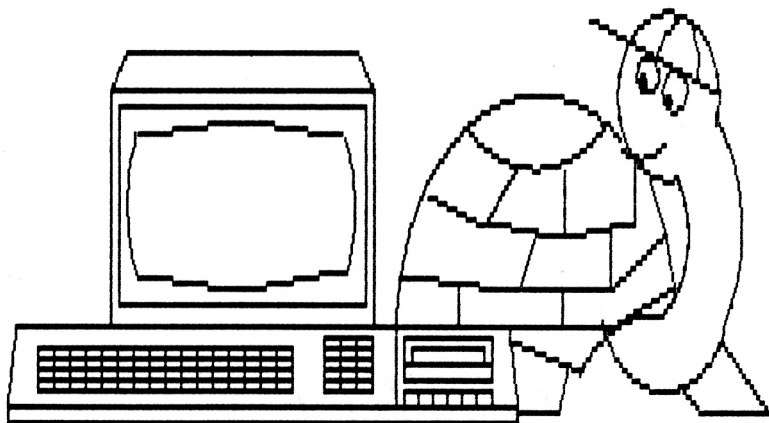


En la figura 1, se observan las etapas durante el trabajo de un ordenador. En el punto (1), se realiza la entrada de datos al ordenador; en el punto (2), los datos son tratados o "procesados" y en el punto (3), se obtienen los resultados de dicho proceso.

Ejemplo: Para realizar una suma de dos números, la entrada de datos consistiría en la introducción de los sumandos; la operación aritmética de la suma sería el "procesamiento" de los datos en el ordenador, y el total de la suma, que es lo que queremos saber, constituiría la salida de los resultados.

El hombre utiliza su cerebro de una forma parecida al ordenador: una parte para "pensar" y otra parte para "memorizar" los datos que ha asimilado. Aquellos datos que necesita memorizar inmediatamente, los anota en su agenda, para ser consultados en el momento adecuado.

De igual manera, el ordenador dispone de un pequeño cerebro, llamado CPU (Unidad Central de Proceso), que es, una memoria capaz de almacenar una cantidad limitada de datos, y otros elementos exteriores (cintas, discos, etc.), que utiliza para guardar y "apuntar" otros datos.

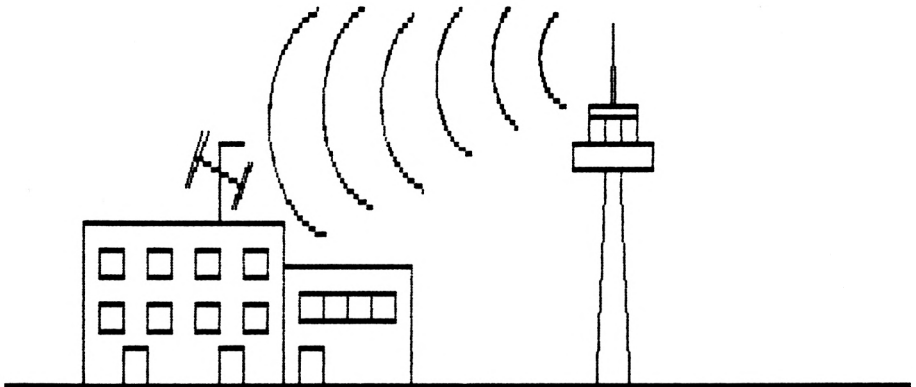


Teniendo en cuenta las características de todos estos elementos, podemos clasificar fundamentalmente a los ordenadores en:

- 1 Sistemas con microprocesador.
- 2 Microordenadores.
- 3 Miniordenadores.
- 4 Ordenadores propiamente dichos sean grandes, medianos o pequeños.

### 3 Hardware y Software

En informática, existen dos partes fundamentales relacionadas entre sí. Una, la parte "física", que se compone de un conjunto de circuitos, dispositivos electrónicos, teclado y todos aquellos elementos que podemos ver y tocar, denominada **HARDWARE** (parte dura), y otra, no física que es el conjunto de información (programas y datos), suministrados al ordenador, a la que se denomina **SOFTWARE**, (parte blanda).



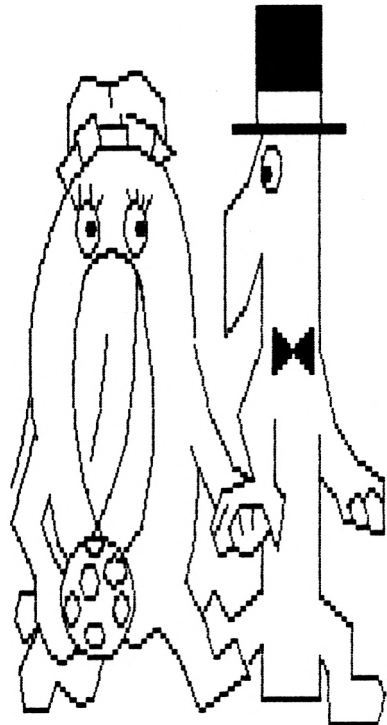
Tomemos un ejemplo: Haciendo una analogía con la TV, diremos que el **HARDWARE** lo constituyen: la pantalla, transistores, altavoz, antena, los circuitos y demás componentes electrónicos; Y el **SOFTWARE**, el conjunto de programas emitidos desde los estudios de TV., así como las películas de vídeo.



#### 4 La Informática y el Ordenador.

La numeración decimal, que por las razones apuntadas al principio, se ajusta muy bien a la conformación del hombre, no es la más adecuada para el ordenador. El ordenador es un aparato electrónico, y como tal solo sabe diferenciar dos niveles eléctricos de tensión: alta y baja. A la tensión alta le atribuye el valor (1), y a la tensión baja el (0), operando únicamente con estos dos valores. Esto significa, que toda la información existente en el ordenador, será un conjunto de señales binarias, denominadas DIGITALES.

Toda la información existente en el ordenador, se encuentra codificada mediante ceros (0) y unos (1), que debidamente combinados representan cualquier grupo de números o letras. A este tipo de codificación se la denomina BINARIA o en BASE DOS por estar representada en dos estados o unidades (0 y 1) denominados BITS, derivado de la expresión "Binary Digit".



La representación por medio de dos dígitos de cualquier número decimal, se realiza de la siguiente forma:

DECIMAL	BINARIO	Explicación
	c b a	
0	----> 0 0 0	---
1	----> 0 0 1	---
2	----> 0 1 0	---
3	----> 0 1 1	---
4	----> 1 0 0	---
5	----> 1 0 1	---
6	----> 1 1 0	---

Y así sucesivamente ...

Generalmente la representación en la memoria central de cualquier valor, viene dada por series de grupos de OCHO BITS, denominados BYTES u OCTETOS. El BYTE es la menor unidad direccionable en la "Memoria Principal", configurándose como un conjunto de celdas de almacenamiento que contienen 8 BITS cada una.

Estas pequeñas celdas (Bytes) son la unidad básica para medir la capacidad de memoria en los ordenadores. Para grandes cantidades de memoria, se utiliza como unidad el Kbyte, equivalente a 1024 Bytes. Esto quiere decir que un ordenador de 48 Kbytes, tiene aproximadamente 48.000 Bytes de memoria.

La representación de los datos, se realiza mediante la combinación de un determinado número de Bytes. Según naturaleza y utilidad de la información a registrar, se necesitará un mayor o menor número de Bytes. Dado que cada BIT tiene dos estados (0 y 1), con un solo Bit ( $2$  elevado a  $1 = 2$ ), representamos 2 elementos, con 2 Bits ( $2$  elevado a  $2 = 4$ ), 4 elementos, con 3 Bits 8 elementos,... y así sucesivamente. En consecuencia, con un solo Byte, podemos representar 256 ( $2$  elevado a 8), combinaciones o elementos distintos.

Generalmente, el almacenamiento en el ordenador de letras, números y otros símbolos especiales (A a la Z, 0 al 9, &, %, \$, etc), se representará internamente mediante un único Byte, no siendo usual la utilización de más de 256 caracteres distintos.

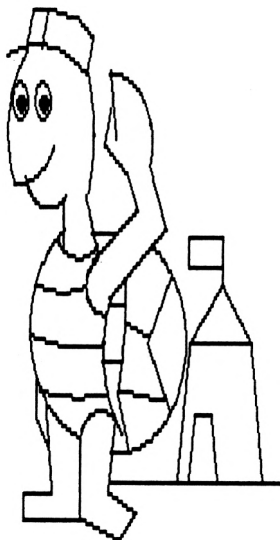
Una de las funciones más importantes de los ordenadores, es la gran capacidad que estos tienen para realizar operaciones matemáticas. Con un solo Byte, el ordenador únicamente puede registrar 256 números distintos ( $2$  elevado a 8). Para operar con números más altos, necesitaremos utilizar más de un solo Byte. Por ejemplo con 2 Bytes ( $8 + 8 = 16$  bits), tenemos la posibilidad de representar 65536 números ( $2$  elevado a 16), etc...

La representación de datos se realiza mediante combinaciones de bytes. En muchos ordenadores se emplean agrupaciones de datos, formadas por "MEDIAS PALABRAS" resultantes de la composición de dos bytes, "PALABRAS", compuestas de cuatro bytes, y "DOBLES PALABRAS", que se componen de ocho bytes.

## 5 Las Instrucciones.

"Seguir instrucciones" es una frase muy corriente y que seguramente habrás oído muchas veces.

Cada vez que quieras realizar una tarea, con un mínimo de claridad y seguridad en su consecución, necesitas obtener las instrucciones válidas, para llevarla a buen término.

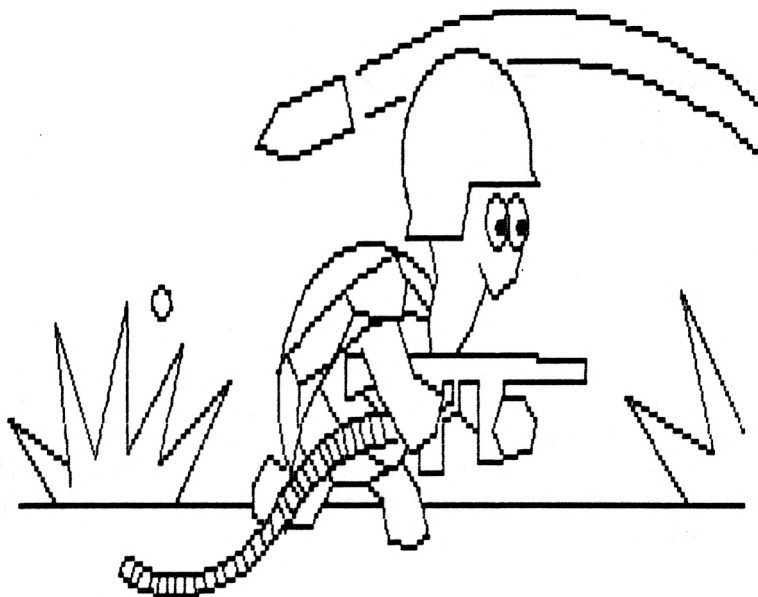


El ordenador es una máquina, construida para realizar un gran número de trabajos distintos, y por ello necesita que le suministren continuamente INSTRUCCIONES.

Una instrucción es, pues, aquella información, que bajo unas reglas específicas permite la consecución de una función para la cual ha sido creada.

El conocer y dominar las instrucciones será a partir de este momento nuestro principal objetivo, pues de ello dependerá el buen funcionamiento de nuestro ordenador.

En el lenguaje de las computadoras oírás muchas veces hablar de instrucciones y COMANDOS. Son lo mismo, con la diferencia de que las instrucciones vienen dadas en una "lista " programa para ser ejecutadas en un momento determinado y el COMANDO es cumplimentado en ese mismo instante. Por ello, dependiendo de la forma de su ejecución, serán llamadas INSTRUCCION O COMANDO.





## Practicar

### 1 Nuestro ordenador.

Nuestro ordenador se compone de un monitor o pantalla, de un teclado, y de un cassette o unidad de disco incorporado. Como puedes observar, el teclado va unido a la pantalla por medio de dos cables, (uno para la corriente y el otro para enviar datos desde el ordenador).

Al encender el ordenador, veremos aparecer en la pantalla el siguiente mensaje :

**Amstrad 64k Microcomputer (v1)**

**©1984 Amstrad Consumer Electronics plc  
and Locomotive Software Ltd.**

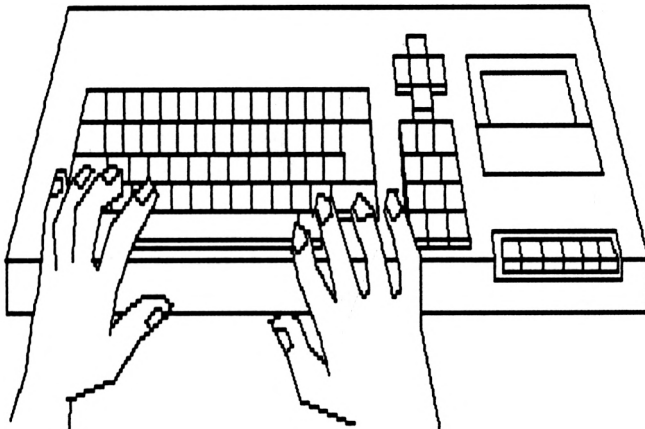
**BASIC 1.0**

Esta información es la que nos dice los datos del fabricante y la versión del lenguaje BASIC, con la que a partir de este momento nosotros empezaremos a trabajar. La palabra "Ready", nos indica que el ordenador se encuentra preparado para comenzar. Debajo de esta palabra observás un pequeño cuadrado al que de ahora en adelante denominaremos CURSOR.

## El teclado

Lo primero que tenemos que conocer para poder utilizar correctamente nuestro ordenador, es su teclado.

El teclado de un ordenador es muy similar al de cualquier máquina de escribir, salvo algunas diferencias, que veremos más adelante. Observarás también, la existencia de teclas con diferentes colores. Estas son las imprescindibles para su funcionamiento.



Teclas principales.

ENTER. Como verás hay dos teclas (ENTER) en el ordenador, una grande y otra pequeña. Su principal función es indicar a éste, que recoja la información que has tecleado. Esto quiere decir que has terminado de darle una instrucción o un dato y ya puede analizarlo.

No olvides nunca que, al finalizar cualquier instrucción o comando (se verá más adelante), tienes que pulsar esta tecla para que pueda ser cumplimentado.

Ahora pulsa varias teclas y (ENTER). Observarás que en la pantalla te aparecerá el mensaje : - SYNTAX ERROR- . Este mensaje te indica, que la información que le has dado al ordenador no es comprensible para él.

ESC. Esta peculiar tecla, desarrolla dos únicas funciones :

1. Cuando la pulsamos una sola vez, se produce una pausa en el proceso que se está realizando, el cual se vuelve a reanudar al pulsar cualquier otra tecla.
2. Pulsándola dos veces consecutivas, se producirá la ruptura o parada del proceso en marcha.

DEL. Utilizamos la tecla (DEL) para "borrar" caracteres , (números, letras, ...etc) previamente introducidos por el teclado. Esta tecla, te permitirá corregir todos aquellos errores que cometes al escribir las instrucciones. Manteniéndola pulsada, se irán borrando sucesivamente de la pantalla los caracteres situados a la izquierda del cursor.

Escribe tu nombre y al finalizar presiona la tecla (DEL) una vez, y comprueba como se borra la última letra. Repite la operación dejando la tecla pulsada.



SHIFT. Esta es una de las teclas, que deben siempre pulsarse a la vez que otras; es decir, su uso es conjunto con las demás. Permite escribir las letras mayúsculas, así como los caracteres que se especifican en la parte superior de algunas teclas.

Realiza las siguientes pruebas :

Pulsa las letras del "abecedario" desde la "a" a la "z" ( en caso de equivocarte emplea la tecla (DEL) ). A continuación aprieta la tecla (SHIFT) y sin soltarla vuelve a marcar el abecedario. Comprueba que las letras ahora aparecen en mayúsculas.

abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Realiza alguna prueba más con las teclas marcadas con dos símbolos (teclas numéricas, ...etc).

CAPS LOCK. Realiza las mismas funciones que la tecla (SHIFT), con la diferencia de que su efecto se mantiene hasta que se aprieta de nuevo. Realiza la siguiente prueba:

Pulsa una vez (CAPS LOCK), y a continuación las las teclas del 1 al 0. Verás aparecer en la pantalla los símbolos que se encuentran marcados sobre dichas teclas. Para volver a imprimir en pantalla los símbolos numéricos, deberás pulsar de nuevo (CAPS LOCK).

CTRL. Esta tecla tiene distintas funciones, dependiendo de otras teclas que se pulsen simultáneamente con ella. Una de dichas funciones es la de obtener caracteres gráficos que no vienen marcados en el teclado. Prueba a pulsar la tecla (CTRL) y seguidamente las de la tercera fila comenzando a contar por arriba. Verás aparecer en la pantalla algunas figuras que se pueden utilizar en los programas y juegos de tu invención.

Pulsa, manteniéndolas apretadas, las teclas (CTRL), (SHIFT) y (ESC) y observa lo que sucede.

La pantalla.

La pantalla del ordenador es un elemento auxiliar del mismo. Su misión más importante consiste en mostrarnos los datos que introducimos en el ordenador y aquellos que el obtiene. De igual forma que la pantalla de nuestra calculadora nos muestra las cantidades que marcamos y los resultados de los cálculos que efectúa, el ordenador utiliza la pantalla para presentarnos todos esos datos, sean números, letras u otros símbolos gráficos.

Normalmente, todas las pantallas disponen de unos mandos situados frontal o lateralmente para ajustar la visión de las mismas. Si pones atención, observarás que en el frente de tu pantalla se hallan situadas tres ruedas marcadas con los nombres : brightnes, contrast y v-hold. El primer mando sirve para regular el brillo, cuya intensidad debe variar en función de la luz del local en que te encuentres.

El mando de contrast te da un mayor o menor contraste o "resolución" de los caracteres que aparecen en la pantalla. Por último v-hold te ajustará la altura en vertical de toda la pantalla impidiendo que por cualquier interferencia esta comience a moverse verticalmente.

El Cassette.

Otro de los "periféricos" más utilizados en la actualidad para el almacenamiento de datos es el CASSETE. Este dispositivo no sólo se utiliza para reproducir o grabar nuestra música favorita, sino que también es de uso corriente como elemento auxiliar de los ordenadores pequeños tanto para almacenar o grabar datos (mandar datos de la memoria central a la cinta) como para reintroducirlos en el ordenador (pasar datos de la cinta a la memoria central). Observa las distintas teclas de que se compone el cassette.

2 Primeros Comandos.

Antes de continuar recuerda dos reglas importantes:

- 1 Después de teclear cualquier comando o instrucción deberás marcar la tecla (ENTER).

- 2 Siempre que aparezca el mensaje " SYNTAX ERROR " se volverá a marcar de nuevo la instrucción o comando correctamente.

Ahora tienes encendido y preparado el ordenador. Vamos a darle a continuación las primeras instrucciones:

CLS

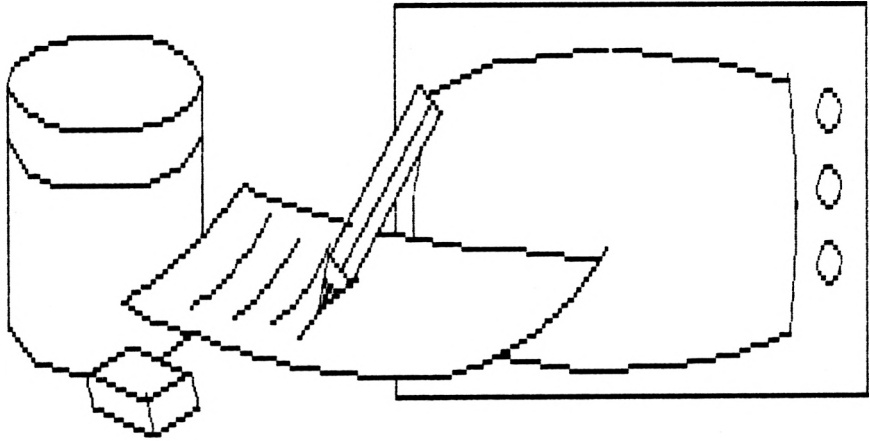
Escribe el comando CLS y pulsa (ENTER). Observa lo que sucede; Ha desaparecido todo el contenido de la pantalla, y esta se ha quedado "limpia" con la palabra READY (preparada), y escrita en la parte superior. CLS, pues, es una instrucción que borra la pantalla del ordenador, como has podido comprobar.

Vamos a seguir probando comandos que actúen sobre la pantalla.

MODE

Mediante este comando defines el MODO de escritura en la pantalla. Existen tres modos: MODE 0 (para 20 caracteres ), MODE 1 (40 caracteres),y MODE 2 (80 caracteres).Ahora te encuentras en " MODE 1 ", es decir que tu pantalla admite 40 espacios en cada línea horizontal. Teclea a continuación MODE 2 (ENTER).

Comprueba como cambia el tamaño de las letras según el modo que seleccionas. Pulsa varias teclas y cuenta la cantidad de signos que caben en cada línea.



## BORDER

Este comando o instrucción, te permite cambiar el color del borde de la pantalla. La gama de colores es muy amplia, y se obtienen marcando, a continuación del comando, el número correspondiente del 0 al 26. Prueba por ejemplo:

BORDER 15

(ENTER)

Observa como ha cambiado el color del borde. Sigue probando con otros números. Para dejarlo como al comienzo marca BORDER 1.

## PAPER

La selección del color de fondo de la pantalla, a excepción del borde, depende del comando PAPER. Es decir: actúa como si seleccionaras el color del papel sobre el que vas a escribir, de ahí su nombre. Realiza algunas pruebas cambiando de número al igual que lo hiciste con el comando BORDER.

## PEN

Al comando PEN le es aplicable lo dicho sobre PAPER, pero referido al color de la tinta. Actúa, pues, sobre el color de todo aquello que imprime en pantalla. Haz alguna prueba pero.... con cuidado. ¿Qué ocurriría si el número de color elegido para los comandos PAPER y PEN fuera el mismo?



## Preguntar

- 1 Haz una lista de los precursores de los ordenadores por orden de antigüedad.
- 2 Señala los componentes electrónicos que se han sucedido en la historia de las máquinas programables.
- 3 Indica los elementos que componen un ordenador, y la función de los mismos (entrada, procesamiento o salida de datos).
- 4 Pon un ejemplo para explicar el Hardware y Software.
- 5 Suma 1 al número binario 110111.
- 6 Explica la diferencia que hay entre una instrucción y un comando.
- 7 Indica las partes de tu ordenador.
- 8 Haz una lista de comandos de pantalla.





UNIDAD 2 EL ORDENADOR

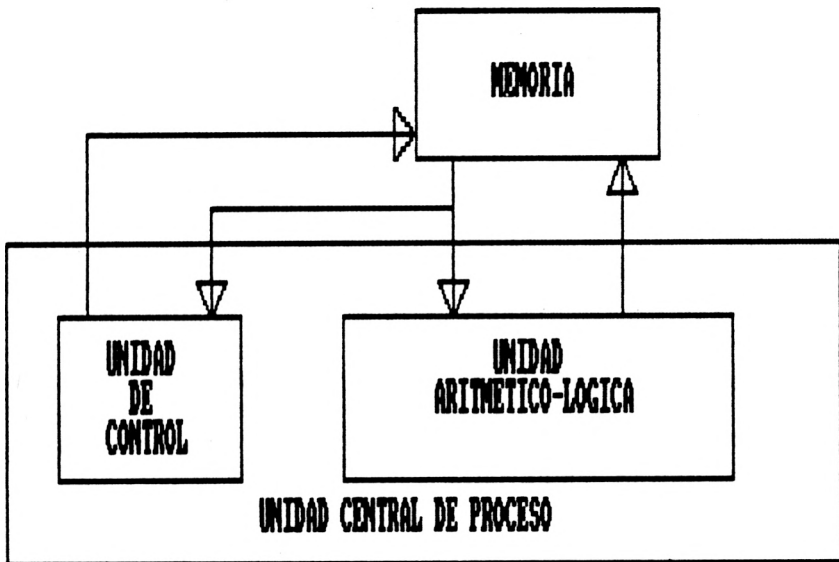


Conocer

1 Partes del ordenador.

Los ordenadores se componen internamente de varios "órganos" o partes estrechamente relacionadas entre sí, y sin las cuales no podría funcionar. Estos dispositivos permiten al ordenador realizar las funciones para las que ha sido fabricado, principalmente las de almacenamiento de datos, cálculo y control.

Observa en el gráfico siguiente la configuración del ordenador y la relación entre las distintas unidades.



Como puedes ver, existen dos elementos principales en el esquema básico de un ordenador: la UNIDAD CENTRAL DE PROCESO (U.C.P) y la MEMORIA. Entre estos dos órganos, existen unos cables de conexión llamados BUSES. A su vez la U.C.P se compone de dos elementos principales: la Unidad Aritmético Lógica (U.A.L) y la Unidad de Control (U.C).

### La Memoria Central.

La Memoria Central, es aquella pared del ordenador encargada de almacenar todos los datos, resultados, e instrucciones utilizadas por el mismo. Toda la información que pasa por el ordenador queda en algún momento retenida en ella, tanto la que recibe del exterior, como la procedente de los resultados obtenidos en las operaciones internas. Es un órgano pasivo y simplemente tiene como objetivo almacenar la información.

### Estructura.

La estructura de la memoria es comparable a un gran armario con muchos cajones, de los cuales algunos están llenos y otros vacíos. Cada cajón se encuentra en una posición determinada del armario y todos los cajones son del mismo tamaño. Para poder acceder a cualquier cajón debemos conocer la posición que éste ocupa en el armario. Por ejemplo: el tercer cajón de la segunda fila comenzando por la izquierda.

Cada uno de estos cajones o celdas equivale a una posición de memoria. La capacidad de cada cajón variará de un ordenador a otro. El nuestro tiene asignado un Byte u Octeto por cada uno de sus cajones. Esto quiere decir que cada cajón del armario se divide a su vez en ocho compartimentos correspondientes a cada uno de los ocho bits del que se compone un byte.

#### Capacidad de la Memoria.

Como ya vimos con anterioridad, la capacidad de la memoria se define en relación con la cantidad de bytes que ésta puede almacenar. De igual forma vimos que 1Kbyte es igual a 1024 Bytes ( $2^{10}$ ), aunque con frecuencia se toma el valor de 1000 Bytes para referirse a 1Kbyte. Este sistema de medición nos indicará el número de celdillas que un determinado ordenador dispone para almacenar la información.

#### Tipos de Memorias.

Las memorias se clasifican de diversos modos.

##### A. Según su uso :

**Memorias de Lectura Destructiva.** Son aquellas cuya información es destruída una vez realizada su lectura.

**Memorias de Lectura no Destructiva.** La información permanece inalterable después de realizarse la lectura.

### B. Según la forma de retener la información:

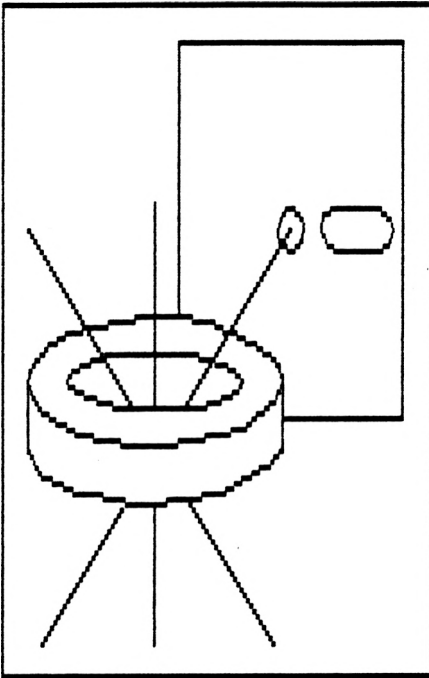
Memorias volátiles y no volátiles. En las primeras se pierde la información registrada al apagar el ordenador. En las no volátiles la información permanece.

Memorias Estáticas o Dinámicas. Las memorias estáticas tienen la facultad de permanecer inalterables hasta su modificación. Las dinámicas sufren una degradación en el tiempo por lo que necesitan impulsos periódicos de "refresco" para mantenerlas.

### 3. Según la forma de acceso.

Memorias ROM (Read Only Memory). Son memorias de lectura. Sólo se pueden leer y no podemos alterar su contenido. En ellas se han grabado previamente los "programas internos" que dirigen el funcionamiento del ordenador y que constituyen el denominado "SISTEMA OPERATIVO". Este tipo de memorias son no volátiles, pues la información registrada no se borra al desconectar el ordenador.

Memorias RAM (Random Acces Memory). En éstas podemos leer y escribir cualquier tipo de información. Todos los datos y programas creados por los usuarios quedan registrados en ellas. A diferencia de las ROM, las memorias RAM son volátiles, por lo que para conservar los programas y datos que almacena, es preciso grabarlos en algún soporte externo (cinta, disco, etc..), antes de apagar la máquina.



D. Según el soporte físico de la memoria :

Memorias de Núcleos de ferrita (no utilizadas en actualidad).

Memorias de Soporte magnético como : cintas, discos y tambores magnéticos.

Memorias de Burbujas magnéticas.

La Unidad Central de Proceso.

Es la parte más complicada del ordenador. La gran cantidad de componentes electrónicos que la configuran, permite la realización de las tareas más complejas.

Las funciones que realiza son las siguientes:

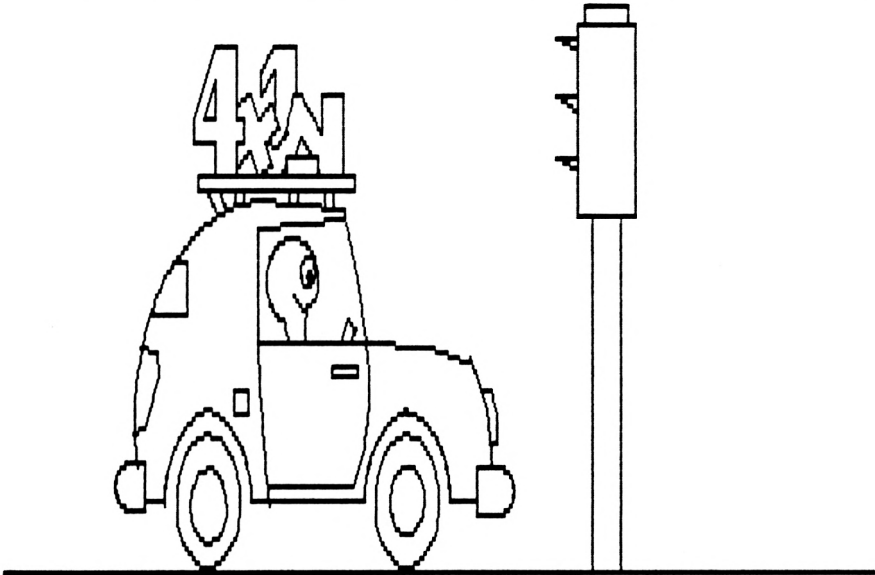
A. Analiza las instrucciones que recibe comprobando

si son válidas, es decir si pertenecen al repertorio de instrucciones del ordenador, activando el BUS DE CONTROL (que más adelante estudiaremos la forma en que proceda para actuar sobre los demás componentes del ordenador y ejecutar correctamente las instrucciones).

- B. Coordina la comunicación entre los periféricos de entrada y salida (E/S) y la Memoria Central. Sirve de enlace para llevar los datos de ésta a los periféricos y viceversa.
  
- C. Permite la comunicación entre otras memorias RAM/ROM por medio de distintos canales de enlace.

### La Unidad Aritmético Lógica. (U.A.L)

Como su nombre indica,este componente se encarga de realizar las operaciones con los datos suministrados al ordenador. No solamente actúa como la "calculadora" del ordenador, realizando operaciones aritméticas (suma, resta, multiplicación, etc...),sino que también realiza operaciones lógicas (comparaciones entre palabras, movimiento de datos, etc...).



La Unidad de Control (U.C).

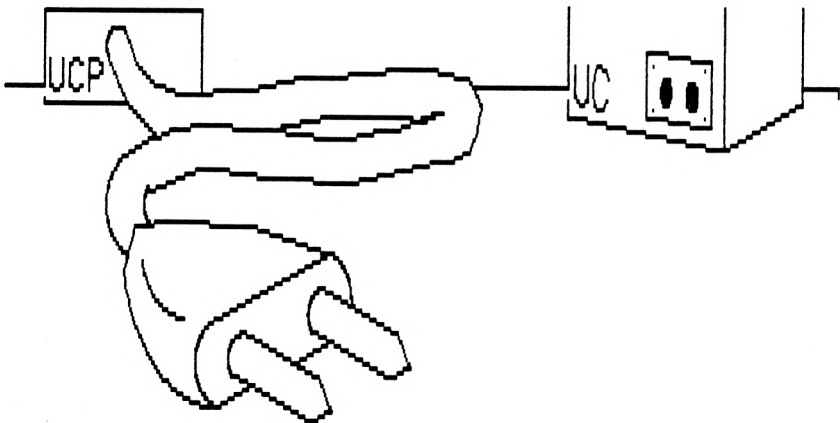
La Unidad de Control (U.C), tiene como objeto dirigir todas las actividades del ordenador. Sigue todas las instrucciones dadas en el programa, controlando las operaciones que han de efectuarse y ejecutándolas en el orden correspondiente. Coordina entre sí los distintos elementos y unidades que componen el ordenador. Dispone de un reloj para sincronizar todas estas operaciones en la secuencia y el orden establecido. La U.C, pues :

- A. Controla la entrada y salida de instrucciones y datos desde y hacia la memoria principal.
- B. Localiza las instrucciones por su dirección de memoria, las extrae y las traduce a su "lenguaje de ceros y unos" para conocer su significado y ejecutarlas.

## Los Buses

Se llama "BUS" al conjunto de cables que tienen como fin el conectar entre sí las diferentes partes de un ordenador. Existen varios tipos:

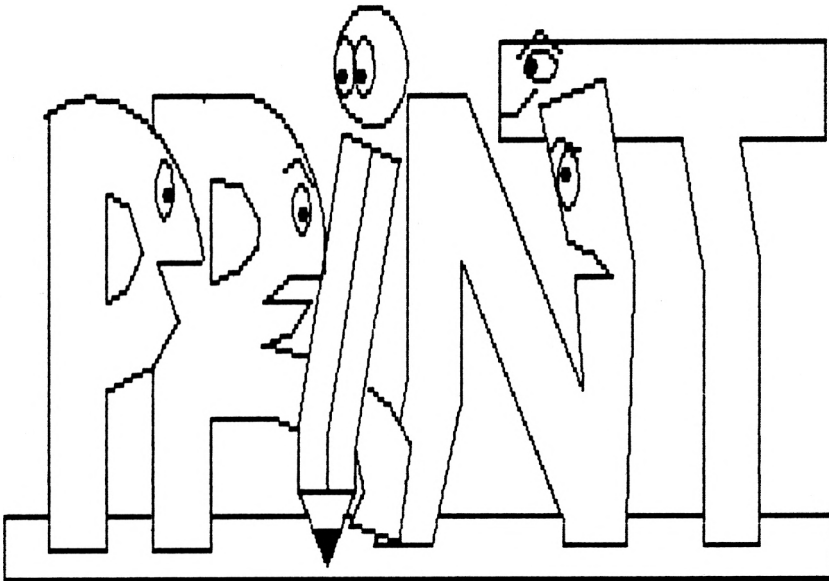
**BUS DE DATOS.** Se encarga de comunicar la U.C.P. con la memoria. A través de él se transfieren los datos e instrucciones utilizadas por la U.C.P en ambas direcciones. Está formado por pequeños cables paralelos (generalmente 8), que permiten la transmisión de un solo bit.



**BUS DE DIRECCIONES.** Sabemos que la memoria es como un gran armario compuesto por muchos cajones/celdas. Cada una de estas celdas se identifica por su dirección o "posición" en el armario. El Bus de Direcciones sirve para acceder a cada una de las mismas.

**BUS DE CONTROL.** Permite la comunicación de las señales de control para activar y desactivar las unidades del ordenador.





## 2 La instrucción PRINT.

En las páginas precedentes has podido estudiar las partes de que consta un ordenador. Recuerda también las tres etapas de su trabajo: (1) entrada de datos, (2) tratamiento de datos y (3) salida de datos.

Todas las instrucciones de BASIC que iremos examinando, tienen relación con una de esas tres fases de trabajo del ordenador.

La primera "verdadera" instrucción que vamos a estudiar es PRINT. Seguramente, ya habrás oído alguna vez la palabra "printar". Es una mezcla entre Print y pintar, y es que pintar o "sacar" en la pantalla, es justamente el cometido de la misma.

La instrucción PRINT permite visualizar y presentar en la pantalla cualquier dato ya se trate de resultados u operaciones, valores internos, funciones,... etc.

¿Sabrías decir a cuál de las etapas de funcionamiento del ordenador corresponde la instrucción PRINT ?

Evidentemente, tal como habrás pensado, corresponde a la fase 3 puesto que permite la salida de datos. Es, además, una de las instrucciones que más vas a utilizar en tus programas.



### Practicar

#### 1 Uso de la instrucción PRINT.

Como ya dijimos esta instrucción permite que el ordenador exponga los datos que estamos interesados en visualizar.

Veamos algunos ejemplos. Escribe :

```
PRINT "SOY UN ORDENADOR"
```

Fíjate como al pulsar (ENTER), el ordenador escribe la copia de tu mensaje.

¿ Has observado que ha hecho falta poner unas comillas (") al principio y final de la frase ?

Ello tiene una explicación que más adelante veremos, por ahora confórmate con saber que todas las frases necesitan una "cartulina" en la que poder escribirlas. Esta cartulina serán siempre las comillas.

PRINT 5+20

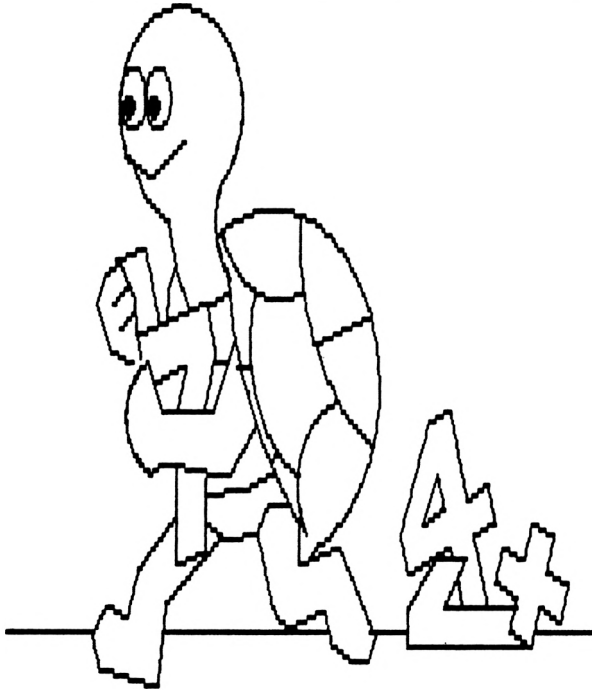
(ENTER)

Obtienes como resultado la cantidad 25, y es que tu ordenador es sobre todo una potente calculadora capaz de decirte al momento el resultado de cualquier operación que le plantees.

Para las expresiones numéricas (números) y operaciones, no es necesario poner las comillas ("), pues son elementos para operar con ellos y no para un uso informativo.

No olvides nunca escribir las frases en su "cartulina" (") al igual que escribirías tu nombre en una para ponerlo sobre tu mesa de trabajo.

La instrucción PRINT es, una de las más importantes, y para poder utilizarla correctamente en expresiones matemáticas, necesitaremos conocer algunas cosas.



Los cálculos matemáticos en BASIC son muy corrientes. Los símbolos utilizados para realizarlos son los siguientes: Para la suma y la resta los mismos que habitualmente empleamos ( + y - ). Para la multiplicación (\*), la división (/),y la potenciación (↑).

Realiza las siguientes operaciones y comprueba su resultado:

$$10 + 20 / 2 , 3000 * 4 + 31 - 2, 7^3, 365 / 12$$

No olvides utilizar la instrucción PRINT.

Intenta resolver algún problema empleando el ordenador como calculadora.

Vamos a continuar haciendo prácticas con PRINT, de modo que imprima en la pantalla cosas que se nos ocurran, pero ATENCION, siempre de acuerdo con las reglas antes establecidas, para que nuestro ordenador nos pueda entender.

Escribe :

```
PRINT "Una caja de manzanas tiene ";20;" unidades"
```

Marca la tecla (ENTER), y verás aparecer en la pantalla :

```
Una caja de manzanas tiene 20 unidades  
Ready
```

El punto y coma (;) se utiliza para separar letreos y números con la instrucción PRINT.

Ahora escribe:

```
PRINT "Una caja de manzanas tiene 20 unidades"
```

Observarás que el resultado al pulsar (ENTER) ha sido el mismo. Ello es debido a que el número 20 lo hemos incluido como parte de nuestro letrero, PERO NO PODEMOS REALIZAR NINGUNA OPERACION CON EL.

Sin embargo:

```
PRINT "En tres cajas de manzanas hay ";20*3;" unidades"
```

Obtendrás como respuesta:

En tres cajas de manzanas hay 60 unidades  
Ready

Los números 20 y 3, al estar fuera de nuestra "cartulina" ("), han sido interpretados como tales números y el ordenador ha efectuado la correspondiente operación.

Prueba seguidamente a introducirlos dentro de las comillas y verás la diferencia: el ordenador no hace la operación.

## 2 Memoria libre. Instrucción FRE(0).

En las páginas precedentes, se ha explicado ya lo que es la memoria y también que cualquier dato introducido en el ordenador, ocupa una determinada posición dentro de aquella. Cuanto más amplio es, por ejemplo, un programa, mayor espacio ocupa en la memoria y por ello, a veces, para saber si es posible o no su ejecución, necesitaremos conocer la cantidad de memoria que queda aún disponible en el ordenador.

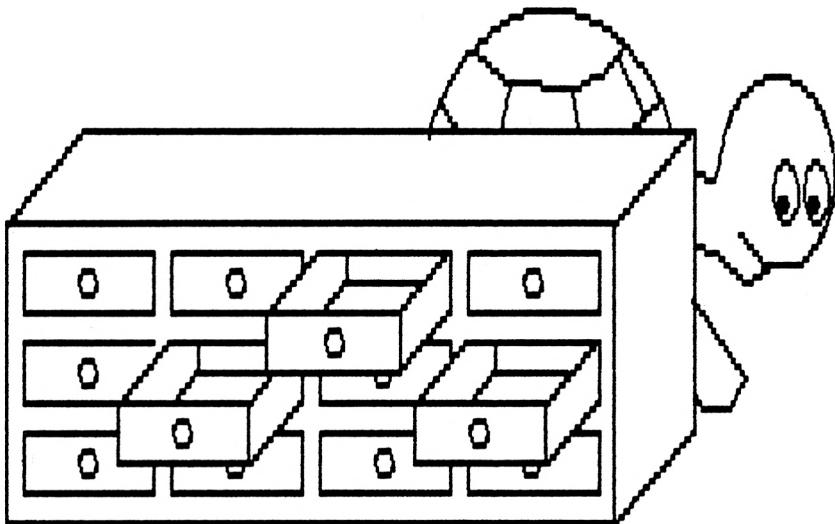
Para saberlo no tenemos más que preguntárselo. Existe una manera de pedirle a nuestro ordenador que nos la indique.

Escribe :

PRINT FRE(0)

(RETURN)

Si has marcado bien esta instrucción, el ordenador te responderá con un número que representa la cantidad de Bytes libres para almacenar datos. Repasa el AREA TEORICA y calcula su equivalencia en KBytes.





## Preguntar

1

Indica las partes de que se compone un ordenador y explica cuales son sus principales funciones.

2

Haz una lista de los tipos de memorias existentes.

3

¿Qué diferencias existen entre la memoria RAM y la memoria ROM.?

4

Señala la función más importante de la instrucción PRINT

5

Escribe en el ordenador tu nombre, dirección y teléfono utilizando la instrucción PRINT.

6

Resuelve utilizando el ordenador como una calculadora el siguiente problema:

Si un año tiene 365 días, calcula:

- ¿ Cuántos días tienen tres años ?
- ¿ De cuántas semanas consta ?
- ¿ Cuántos minutos transcurren en dos años ?



## UNIDAD 3 LA INFORMACION



Conocer

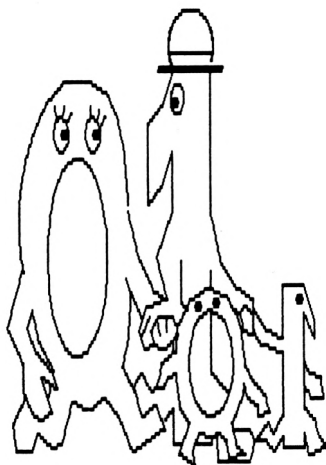
## 1 Los sistemas de numeración.

Los sistemas de numeración sirven para representar números. Nosotros empleamos el sistema decimal, es decir, que utilizamos diez dígitos diferentes para representar cualquier cifra.

Alguno de vosotros se preguntará ¿para qué son necesarios otros sistemas de numeración, si ya con el decimal podemos escribir cualquier número y realizar todo tipo de operaciones?.

La explicación es bien sencilla:

Como recordarás, los componentes electrónicos con los que están fabricados los ordenadores sólo reconocen dos estados: Conectado y Desconectado. Por ello que necesitan reducir la información que contienen a un sistema representado tan sólo por dos dígitos (0 y 1).



Este es el sistema binario de numeración (bi=dos), susceptible, a su vez, de reducirse a otros sistemas más sencillos para "ahorrar" memoria.

Así el número decimal 65430 (5 dígitos) equivale al binario (1111 etc) y al hexadecimal ff96 (4 dígitos).

Todo sistema de numeración es convencional, y en consecuencia, tan válido es uno como otro. La diferencia estriba en el número de caracteres empleados en cada uno de ellos: dos en el binario, ocho en el octal, diez en el decimal, etc.

#### 1. DECIMAL.

En este sistema, que es el utilizado habitualmente por nosotros, la representación numérica tiene lugar mediante los diez dígitos de todos conocidos (0-1-2-3-4.....).

#### 2. BINARIO.

Se sirve de dos dígitos (0 y 1). (Ver página 8).

#### 3. HEXADECIMAL.

El código hexadecimal consta de 16 caracteres distintos (0-1-2-3-4-5-6-7-8-9-A-B-C-D-E y F). Es muy útil para representar números binarios simplificando su complejidad, ya que el paso de binario a hexadecimal resulta muy sencillo.

## 4. OCTAL.

Otro sistema empleado frecuentemente por su fácil conversión al binario es la codificación OCTAL, que utiliza los caracteres: 0, 1, 2, 3, 4, 5, 6, 7. En esta codificación cada número octal se corresponde con tres dígitos binarios como veremos más adelante

## 2 Conversión de Bases de Numeración.

## De Base Decimal a Binaria.

Para reducir un número decimal a binario operamos de la siguiente forma : Se divide el número entre dos, y el cociente obtenido se divide nuevamente entre dos, repitiendo la operación tantas veces como sea necesario, hasta que el cociente sea igual a 1. El resultado vendrá dado tomando el ULTIMO cociente, es decir el 1, y añadiendo en orden inverso los restos de las divisiones (0 ó 1)

Ejemplo :

¿Cómo escribiremos en binario el número 57 ?

$$\begin{array}{r}
 57 : 2 \\
 17 \quad 28 : 2 \\
 1 \quad 08 \quad 14 : 2 \\
 - \quad 0 \quad 0 \quad 7 : 2 \\
 \quad \quad - \quad - \quad 1 \quad 3 : 2 \\
 \quad \quad \quad \quad - \quad 1 \quad 1 \\
 \quad \quad \quad \quad \quad - \quad -
 \end{array}$$

Es decir que 57 en base 10 es igual a 111001 en base 2.

De Binario a Decimal.

A partir de una codificación en binario obtendremos el número decimal equivalente, mediante el procedimiento que seguidamente se describe:

Conversión del número binario 111001 a decimal :

PRIMER	BIT 1	-->	1 * (2 <sup>0</sup> )	=	1
SEGUNDO	BIT 0	-->	0 * (2 <sup>1</sup> )	=	0
TERCER	BIT 0	-->	0 * (2 <sup>2</sup> )	=	0
CUARTO	BIT 1	-->	1 * (2 <sup>3</sup> )	=	8
QUINTO	BIT 1	-->	1 * (2 <sup>4</sup> )	=	16
SEXTO	BIT 1	-->	1 * (2 <sup>5</sup> )	=	32
					---
			Resultado		57

Como hemos visto en el ejemplo, comenzamos tomando el primer bit de derecha a izquierda y multiplicamos su valor (1) por 2 elevado a la posición que ocupa (siendo cero la primera), dando como resultado 1. A continuación tomamos el siguiente bit (0), multiplicándolo por 2 elevado a 1 (siguiente posición) dando como resultado 0, y así sucesivamente.

De Hexadecimal a Decimal.

Para convertir un número hexadecimal a decimal, se procede de forma similar a la anterior, con la diferencia de que el valor de las letras A,B,C,D,E y F son respectivamente 10,11,12,13,14 y 15 y cada una de sus cifras se multiplicará por 16 (elevado a la potencia correspondiente), en vez de por 2.

Ejemplo :

Pasar el número hexadecimal AC24 a base decimal.

PRIMERO	4	-->	4	*	(16 <sup>0</sup> )	=	4
SEGUNDO	2	-->	2	*	(16 <sup>1</sup> )	=	32
TERCERO	C	-->	12	*	(16 <sup>2</sup> )	=	3072
CUARTO	A	-->	10	*	(16 <sup>3</sup> )	=	40960
							-----
			Resultado				44068

De Decimal a Hexadecimal.

El procedimiento es idéntico al utilizado para la conversión de decimal a binario, es decir, cogiendo sus restos en orden inverso. A continuación veamos un ejemplo:

Convertir el número 44068 en base 10 (decimal) a número hexadecimal ( base 16 ).

```

44067 : 16
120
086      2754 : 16
068      115
04       034   172 : 16
--       02    12
--       --    --
12=C,10=A      10

```

resultado : AC24

De Octal a Binario.

Cada número octal se corresponde con tres dígitos en binario. Para convertir un número de codificación octal a su equivalente en binario, se cambia cada cifra octal por el grupo que le corresponde en el sistema binario. Por ejemplo:

OCTAL	BINARIO	OCTAL	BINARIO
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

¿Cuál es el correspondiente a 41 (Octal) en base 2?

$$41 = 4(100) 1(001) = 100001$$

Para obtener un número OCTAL a partir de otro BINARIO, se procede de igual forma pero a la inversa.

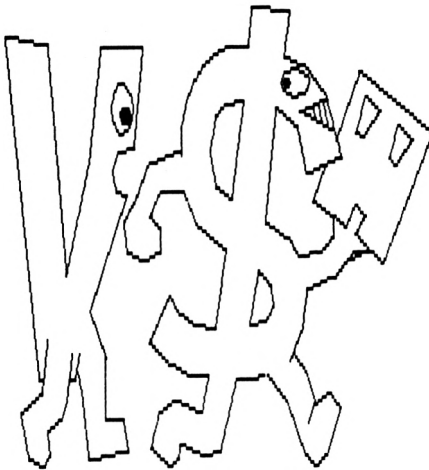
### 3 Elementos del lenguaje basic.

Para aprender un idioma, debemos conocer las reglas que nos permitan desenvolvernosen él. El BASIC no es una excepción, pero afortunadamente el número de reglas no es elevado. La práctica será el mejor aliado para su aprendizaje. Comencemos por diferenciar dos conceptos básicos como son el de CONSTANTE Y VARIABLE.

#### Las Constantes.

El propio nombre hace referencia a algo que "permanece". En nuestro caso se denomina constante, a la información registrada en el ordenador cuyo contenido no varía.

Las constantes pueden ser NUMERICAS (números) o ALFANUMERICAS (letras y números). Las constantes NUMERICAS son siempre números y las utilizamos generalmente para operar con ellas.



Nunca se te ocurriría multiplicar un número por el

nombre de una cosa, por ejemplo: 3X manzana. Pues bien las constantes ALFANUMERICAS no sirven para hacer operaciones aunque contengan números (XTR2 puede ser el nombre de tu robot preferido y su nombre tiene un número). Recuerda que una constante alfanumérica, siempre se introduce en el ordenador entre comillas (").

Repasa en la unidad anterior el uso de la instrucción PRINT y verás algunos ejemplos.

Las Variables.

Sí, son variables porque "varían". El valor que almacena una variable es fácilmente cambiado por otro. Imagínate un armario con muchos cajones, todos ellos con su correspondiente letrero; En cada cajón puedes meter, sacar y cambiar las cosas que se encuentran dentro. Con las variables puedes actuar de la misma forma que con los cajones. El letrero, en este caso, equivaldría al nombre elegido para identificar la variable.

Existen dos tipos de variables: NUMERICAS y ALFANUMERICAS.

Variables Numéricas.

Almacenan datos numéricos, es decir números. Llamemos LAPICES a una variable a la cual queremos asignar un determinado valor (p.ej. 5). Mediante una sencilla ins-



trucción el lenguaje BASIC nos permite, como verás más adelante, asignar el valor 5 o cualquier otro a la variable LAPICES o, lo que es lo mismo, introducir en el cajón marcado con ese nombre el número 5.

Ejemplos de variables numéricas :

LAPICES, CANTIDAD, A, TOT, X, SUMA, .....

RECUERDA: El contenido de una variable numérica ha de ser siempre una constante numérica.

Variables Alfanuméricas.

Sirven para retener información de toda índole, se trate de letras, números o ambos a la vez. Se identifican con un "letrero" que generalmente indica su contenido y que, como ocurre con las variables numéricas, es elegido por nosotros. Se diferencian de éstas en que todas terminan con el símbolo \$.

Ejemplos : LAPICES\$, h\$, NOM\$, ....

RECUERDA: El contenido de una variable alfanumérica ha de ser siempre una constante alfanumérica.

4 Reglas en el empleo de variables.

Existen algunas reglas a tener en cuenta en el momento de elegir el "letrero" o nombre de las variables. En

el ordenador han de introducirse siempre correctamente las instrucciones, comandos y demás información que posteriormente pasará a analizar. En el caso contrario pueden ocurrir dos cosas :

1. El ordenador detecta el error, indicándote su naturaleza y situación.
2. El ordenador no detecta el error y nuestro programa no funciona bien.

Para que no ocurra ésto, es preciso prestar mucha atención y conocer perfectamente las normas que regulan la redacción o presentación de datos al ordenador.

Así pues, recuerda que todas las variables (numéricas y alfanuméricas), se rigen por las siguientes reglas:

1. Todos los nombres de variables, "letreros", tienen que comenzar por una letra (mayúscula o minúscula), y no por un número.
2. Los nombres de las variables, no pueden contener espacios en blanco ni símbolos distintos a las letras o números.

Ejemplos :

78Total (MAL) CLASE.NR (MAL) CAJA (BIEN)

LIBR 1\$ (MAL) 2 Clase (MAL) NUM\$ (BIEN)

## 5 Asignación de variables.

Hemos visto ya las clases de variables existentes, así como las reglas para su uso. Veamos ahora como y para qué se utilizan.

Una de las propiedades más importantes del ordenador es la de "memorizar" gran cantidad de datos.

¿Has pensado de qué forma puede guardar todos esos datos?

La respuesta es bien fácil, utilizando las VARIABLES.

Si quieres "meter", en un cajón de tu ordenador alguna "cosa", primero tendrás que decirle en qué "cajón" la vas a guardar. Para ello buscarás un nombre o "letreiro" de TU invención, pero basándote en las REGLAS DE EMPLEO DE LAS VARIABLES, es decir que has de pensar previamente si lo que vas a guardar es un número (para realizar operaciones) o son letras/números (tu calle, un nombre, etc...). Pues bien, una vez que has encontrado un nombre válido para tu variable, escribirás ésta, seguida del signo (=) IGUAL, y a continuación lo que quieres guardar en ella. No olvides que los datos alfanuméricos van entre comillas ("").

Veamos un ejemplo :

- 1 Se quiere guardar en la memoria del ordenador el nombre de un animal : Gato

2 Pensamos el nombre de la variable (alfanumérica) para guardar en ella la palabra Gato: ANIM\$ u otra que se nos ocurra.

3 Escribimos : ANIM\$ = "Gato"



### Practicar

#### 1 Prácticas con sistemas de numeración.

En la práctica de la informática, como se ha dicho, se utilizan, además del binario, otros sistemas de numeración. El ordenador tiene instrucciones precisas para pasar de un sistema a otro sin dificultad. En cuestión de diezmilésimas de segundo, te convertirá el número que de-sees a otra base de numeración. Vamos a hacer algunas pruebas.

Recuerda el primero de los ejemplos: Pasar el número 57 en base 10(decimal) a base 2(binario). Pues bien, si escribimos :

```
PRINT BIN$(57,8)           (ENTER)
```

```
111001  
READY
```

Por medio de la instrucción PRINT, nuestro ordenador presenta en pantalla el número binario equivalente al decimal 57. Probemos a continuación con un número hexadecimal:

```
PRINT HEX$(44068)           (ENTER)
```

```
AC24  
READY
```

Correcto, tu ordenador funciona perfectamente.

Recuerda: El ordenador registra toda la información que le damos (instrucciones, datos, ...etc) en sistema Binario. Sin embargo, la presentación en pantalla la efectúa en Decimal, a no ser que nosotros le indiquemos lo contrario.

## 2 Utilización de variables

Recuerda que las variables, se utilizan para almacenar en la memoria del ordenador, todos aquellos datos que más tarde utilizaremos.

Si has leído con atención los pasos que hay que dar para la ASIGNACION DE VARIABLES, podrás realizar sin problemas los siguientes ejercicios prácticos.

Marca en el teclado de tu ordenador:

```
CLS                           (ENTER)
```

```
ABC = 365                       (ENTER)
```

```
CLS                             (ENTER)
```

```
PRINT ABC                       (ENTER)
```

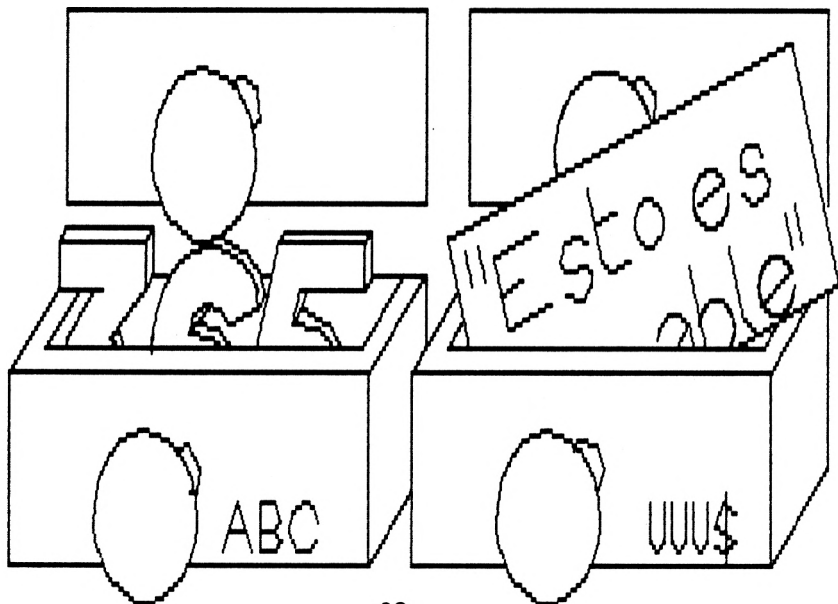
El ordenador te ha respondido con la cantidad 365 y READY. Esto quiere decir, que en la memoria del ordenador hemos creado un "cajón con su letrero" (una variable), en el que guardamos el número 365. Ahí permanecerá hasta que cambiemos su contenido o lo borremos.

Escribe ahora :

CLS (ENTER)

PRINT 5 + ABC (ENTER)

Dice 360. Claro, el ordenador no ha olvidado que la variable ABC contiene el valor 365. No hace falta poner ese número para operar con él. Con escribir el nombre de su "letrero" podemos utilizarlo para cualquier operación.



Hasta ahora hemos visto como se utilizan las variables numéricas. Veamos algunos ejemplos con el otro tipo de variables: las alfanuméricas.

Pulsa las siguientes teclas:

```
CLS                                     (ENTER)
UUU$ = "ESTO ES UNA VARIABLE "        (ENTER)
CLS : PRINT UUU$                       (ENTER)
```

El ordenador ha escrito con la instrucción PRINT el contenido de la variable UUU\$.

```
CLS                                     (ENTER)
N$="ALFANUMERICA"                     (ENTER)
CLS: PRINT UUU$;N$                    (ENTER)
```

Todos los datos asignados a variables alfanuméricas, van entre comillas por ser CONSTANTES ALFANUMERICAS. El (;) en un PRINT, hace que se escriban los contenidos de las variables uno detrás de otro.



## Preguntar

1

Explica en pocas palabras, por qué los ordenadores utilizan para su funcionamiento el sistema de numeración binario.

2

¿Cuántos sistemas de numeración conoces, y qué dígitos utilizan para representación?.

3

Convierte el número decimal 345 en binario, y realiza después el ejercicio contrario para su comprobación.

4

Traslada el número decimal 58725 a base 16 (hexadecimal)

5

Define una variable numérica y asígnale como contenido un número decimal, convirtiendo después este valor en hexadecimal, por medio de las instrucciones que ya conoces.

6

Crea una variable alfanumérica que informe del resultado del ejercicio anterior.



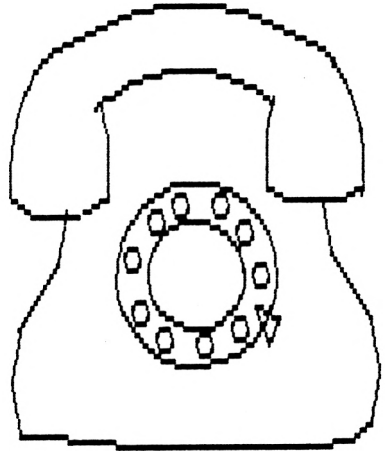
## UNIDAD 4 LOS ALGORITMOS



Conocer

## 1 Los Algoritmos.

Siempre que realizamos una tarea (comer, pintar un cuadro..etc), seguimos un orden en las acciones para llevarla a cabo. Este orden viene marcado por la necesidad de ir cumpliendo una serie de pasos, para lograr el fin deseado. Al conjunto de estos pasos le denominamos ALGORITMO. Son Algoritmos de la Vida Diaria.



Para hacer una llamada por teléfono, realizaremos las siguientes acciones :

- 1 Buscar el número.
- 2 Coger el teléfono.
- 3 Esperar la señal de marcar.
- 4 Marcar el número.
- 5 Esperar la respuesta.

Este es un Algoritmo sencillo. Existen algoritmos mas complejos y que requieren muchas "instrucciones" o pasos consecutivos.

El programa de cualquier ordenador, es en realidad un Algoritmo, pues consta de una serie de instrucciones encadenadas en un orden. Todas las acciones a realizar por el ordenador, han sido previamente estudiadas, y de igual forma que en el Algoritmo de "llamar por teléfono", éstas han sido ordenadas y numeradas para tener un verdadero control de las mismas. Más adelante, estudiaremos de qué forma tan sencilla podemos representar cualquiera de estos algoritmos.

## 2. Los Lenguajes de Programación.

Todo ordenador posee un vocabulario elemental que viene definido por su constructor. Desde el lenguaje de números binarios,(representados por ceros y unos), pasando por la codificación en hexadecimal, hasta llegar a los lenguajes como el BASIC, COBOL, PASCAL y otros llamados de "ALTO NIVEL", por estar formados de estructuras más complejas, ha existido una creciente búsqueda por aproximarlos al lenguaje natural del hombre.

Veamos a continuación, la historia de alguno de estos lenguajes :

### FORTRAN (FORMula TRANslator)

Fue el primer lenguaje simbólico evolucionado. Se desarrolla de 1955 a 1957, creándose para programar cálculos científicos.

### COBOL (COmmom Business Oriented Laguage)

Por iniciativa del Departamento Americano de Defensa se crea este lenguaje (1959/1961), con el propósito de estandarizar el lenguaje utilizado por los ordenadores.

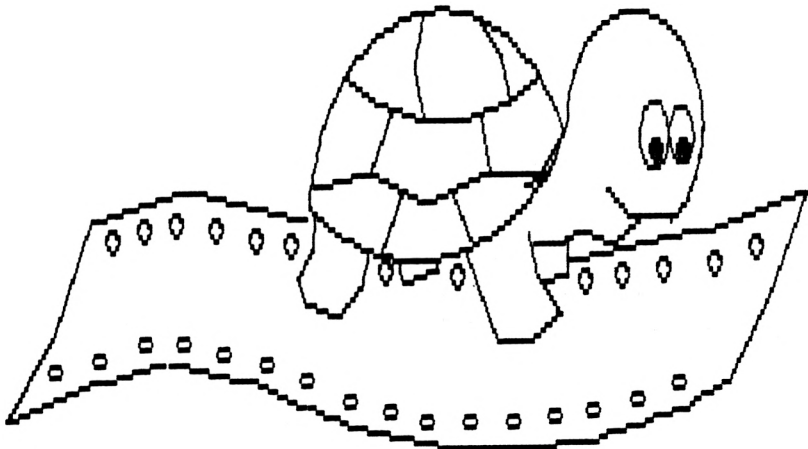
PL/1

Lenguaje desarrollado por IBM y primero que pretendió su utilización universal.

### BASIC (Beginner's All purpose Symbolic Instructions Code)

El interés en crear un lenguaje fácil de entender y flexible para los estudiantes, dio lugar al desarrollo de este lenguaje. Es el más difundido en la actualidad y existen varias versiones del mismo..

### 3. El Programa del Ordenador.



Como vimos al definir los algoritmos, las acciones que realizas normalmente necesitan seguir un orden. Sin darte cuenta estás siguiendo una LISTA INVISIBLE de instrucciones, que por supuesto no tienes anotadas, pero que se repiten invariablemente al ejecutar una misma acción: llamar por teléfono, coger un tren,...etc.

Los ordenadores, cada vez que tienen que hacer un trabajo también necesitan seguir un orden de actuaciones, pero como carecen de la capacidad de aprendizaje de las personas, este orden de actuación, hay que indicárselo en una "lista de instrucciones", que constituye el programa.

Un PROGRAMA DE ORDENADOR, es por tanto, una lista de instrucciones que, siguiendo un algoritmo determinado, cumple una función específica.

Esta lista, para poder ser útil, tiene que ser "comprendida" por el ordenador, es decir, ha de estar escrita en un lenguaje inteligible para él. Con este fin se han creado los distintos lenguajes de programación, entre los que se encuentra el BASIC.

#### 4. Estructura de un programa.

Vamos a estudiar la estructura básica que configura cualquier programa de un ordenador. Para ello nos serviremos de un algoritmo tomado de la vida real.

Siempre que realizamos una acción, estamos siguiendo un "programa" específico. Tomemos por ejemplo la lista de instrucciones del programa "Lavarse las manos".

#### PROGRAMA LAVARSE LAS MANOS

- 1 Abrir el grifo.
- 2 Coger el jabón.
- 3 Frotarse (n veces) las manos con jabón.
- 4 Si no están limpias, volver a 3.
- 5 Secarse.

Si piensas un poco en estas acciones, te darás cuenta de que han intervenido tres elementos en tu programa.

SECUENCIA : Primero el punto 1, después el 2, 3, etc.

REPETICION : En 3 se repite la misma acción.

ALTERNATIVA : En el punto 4 hay una pregunta que puede hacer cambiar la secuencia de tu programa.

Enhorabuena, con este ejemplo tan sencillo acabas de comprender la base de la ciencia de la programación.

## 5 El programa BASIC

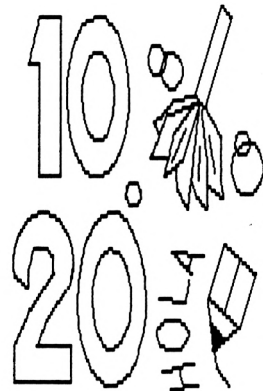
Eres conocedor de las partes y estructuras más importantes de un programa. Sabes de la existencia de los algoritmos y de la LISTA de instrucciones de la que aquellos se componen para llevar a cabo una tarea.

Lo mejor y más fácil para recordar una serie de actos que has de realizar, es hacer una lista numerada de los mismos, máxime si dichas acciones van a tener lugar en un orden determinado. Por ello, todas las instrucciones de un PROGRAMA BASIC, están numeradas en cada línea que se escribe.

Con ésto conseguimos dos cosas muy importantes:

- A SABER en qué número de línea está cada INSTRUCCION.
- B DARLE al ordenador la lista completa (PROGRAMA) para que pueda guardarlo en su memoria.

El hecho de numerar las instrucciones, te evitará su repetición, (introducirlas de nuevo por el teclado) cada vez que quieras ejecutar la misma operación, y por otra parte te va a permitir saltar desde una línea de la lista a otra, como viste en el ejemplo del programa LAVARSE LAS MANOS (saltó de la línea 4 a la 3).



Existen pues, dos formas de trabajar en un ordenador:

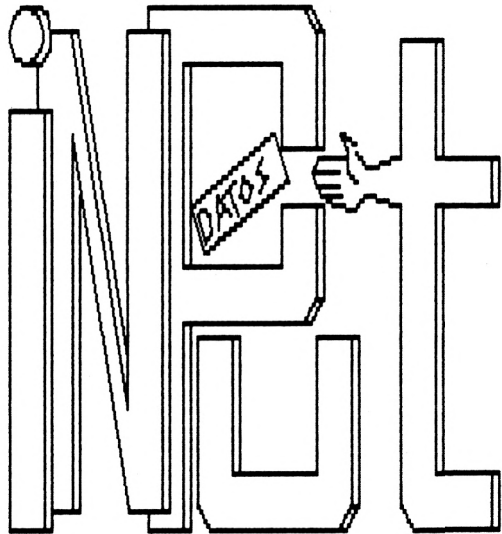
A INTRODUCIENDO programas.

B OPERANDO con ellos.

## 6 La instrucción INPUT.

Esta es una de las instrucciones más utilizadas al igual que PRINT. Al contrario de la instrucción PRINT, cuya finalidad es "sacar" datos al exterior, la instrucción INPUT se encarga de "introducirlos".

La mayor parte del tiempo necesario para un proceso, lo emplean los ordenadores en las ENTRADAS y SALIDAS de información; por ello, estos dos procesos son de tanta importancia. Generalmente la pantalla es un dispositivo de salida de datos, y el teclado, de entrada de los mismos.



La instrucción INPUT, va ligada a una entrada de información por el teclado del ordenador, por lo cual, siempre que se encuentra con esta instrucción, se produce una espera para la introducción de los datos requeridos.

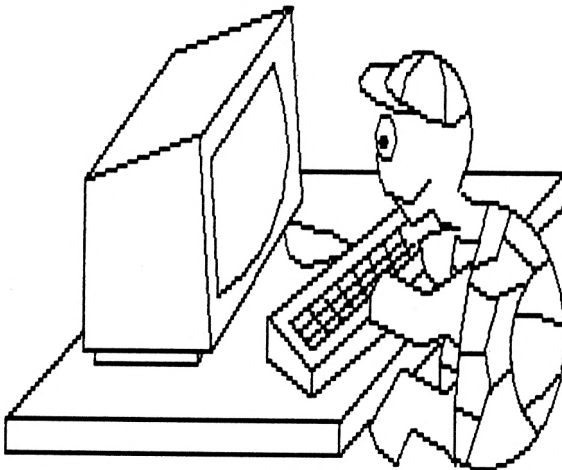
INPUT, utiliza siempre una variable para recoger el dato a recibir. Esta variable puede ser numérica o alfanumérica. Veamos un ejemplo:

```
INPUT JK      (JK recibe un dato numérico)
```

```
INPUT LPR$   (LPR$ recibe un dato alfanumérico)
```

La instrucción INPUT, puede ir acompañada de una explicación o letrero que sirve de ayuda a la hora de introducir el dato correspondiente. El letrero permanecerá encendido hasta que el dato se haya introducido.

```
INPUT      "Introduce un número" YF
```







## Practicar

### 1 Mi primer programa.

Eres ya conocedor de las técnicas y herramientas necesarias para poder realizar tu primer programa. Veamos que tal se te da.

Antes de redactar la lista de acciones a realizar por el ordenador, pensaremos la tarea que vamos a encomendarle.

Imaginemos que la pantalla es una gran "mesa" para realizar tus experimentos. Antes de comenzar con los experimentos, pondremos unos letreros indicando nuestro nombre y algunas cosas más.

Lo primero que haremos con nuestra mesa será limpiarla. Operando de la siguiente forma:

Pulsa (ENTER), una vez para asegurarte de que el ordenador no ha recogido ninguna información extraña.

READY

Muy bien. Los pasos a seguir son los siguientes:

PRIMERO: Recuerda que vas a hacer una "lista" de instrucciones y hay que numerarla. Comienza, pues, con 1 la primera línea de instrucciones a seguir por tu ordenador.

1 CLS

(ENTER)

SEGUNDO: no olvides marcar (ENTER) al terminar la línea con las instrucciones.

TERCERO: Pulsaste la tecla (ENTER) y !No se ha borrado la pantalla!. No, no has cometido ningún error. Recuerda: no estás dando un Comando (ejecución inmediata), al ordenador. Estas sólo dándole una lista de cosas, que tendrá que hacer más adelante

Hasta ahora, la lista introducida en el ordenador consta de una sola orden, que es CLS (limpiar la pantalla o nuestra mesa). A continuación le daremos otra instrucción para que la añada en su lista.

Una de las cosas que generalmente se hace, es poner el nombre de la persona en su mesa de trabajo. Nosotros también vamos a poner el nuestro, por lo que en esta segunda instrucción digitaremos:

2 PRINT "Me llamo XBKC-2" (ENTER)

O tu nombre. Con ésto, conseguiremos poner un cartel indicativo, de quien trabaja en esta pantalla.

En este momento, y si todo ha ido bien, nos encontramos con que nuestro ordenador tiene una lista con dos instrucciones numeradas con un uno (1) y un dos (2).

Vamos a introducir a continuación otra instrucción más, a la lista de acciones que tendrá que realizar el ordenador. Le indicaremos que escriba alguna frase, pero nos saltaremos un número de instrucción, por si se nos olvida después alguna que queramos intercalar.

```
4 PRINT "Esta es mi mesa de trabajo" (ENTER
```

Ha llegado el momento de consultar nuestra lista. Es posible que hayamos cometido algún error al introducir las instrucciones. Hagamos una consulta, pues nuestra pantalla tal vez se encuentre llena de letras, y la lista un poco desordenada.

Para ello pulsemos un par de veces la tecla (ENTER) y a continuación:

```
LIST (ENTER)
```

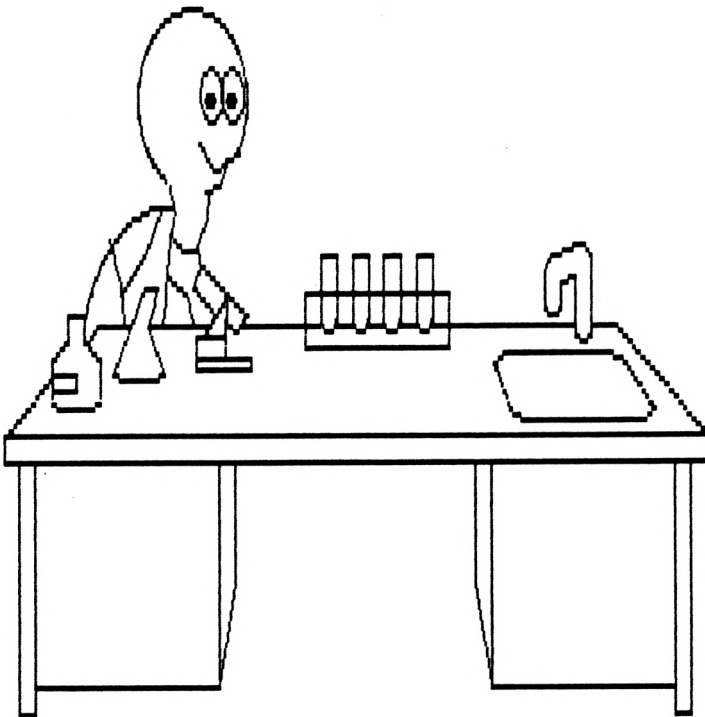
LIST: significa "lista" en inglés, y su función es mostrar nuestra lista "junta y ordenada" en la pantalla. Tu lista ahora está constituida por las siguientes instrucciones:

```
1 CLS  
2 PRINT "Me llamo XBKC-2"  
4 PRINT "Esta es mi mesa de trabajo"  
Ready
```

Si alguna de tus líneas de instrucciones no coincide con éstas, vuelve a marcar el número de la instrucción y a repetir su contenido. El ordenador se encargará de borrar la que está mal, y poner la nueva en su lugar.

¿Estás preparado para comenzar a darle trabajo al ordenador, según la lista que tiene ya guardada?

Adelante entonces. Vamos a preparar nuestra "mesa" para realizar todas nuestras pruebas de programación.



## 2 Ejecución de un programa. Comando RUN.

En este momento, el ordenador tiene su lista de instrucciones guardada y preparada para ejecutarla. Ninguna de esas instrucciones se ha "realizado" hasta ahora.

¿Cómo ejecutar el programa?, simplemente ordenando su ejecución mediante el comando RUN.

Marca pues (ENTER) dos o tres veces y teclea:

```
RUN                (ENTER)
```

```
Me llamo XBKC-2  
Esta es mi mesa de trabajo  
Ready
```

Toda la lista de instrucciones ha sido realizada en el orden que nosotros establecimos. En primer lugar se ha borrado la pantalla, y a continuación se han descrito los dos letreros con PRINT.

Repíte otra vez la instrucción RUN. Ha vuelto a suceder lo mismo. La lista sigue en el ordenador para utilizarla siempre que nosotros queramos, sin necesidad de repetir instrucciones, las cuales al formar una lista numerada no se borra de la memoria.

¿Recuerdas que saltamos un número de la lista sin ninguna instrucción?

Sí ¿verdad?, era el 3. Vamos a poner una instrucción más a nuestro programa y aprovechando que dejamos libre esa línea de la lista, la incluiremos entre otras.

Escribe:

```
3 PRINT "-----" (ENTER)
```

Volvamos a dar un LIST y veremos:

```
1 CLS
2 PRINT "Me llamo XBKC-2"
3 PRINT "-----"
4 PRINT "Esta es mi mesa de trabajo"
Ready
```

Y es que, aunque la última línea que hemos escrito ha sido PRINT "-----", al darle el número 3, el ordenador la ha situado en el lugar correspondiente, es decir, que el ordenador coloca siempre las líneas de la lista por orden de numeración.

Pulsemos de nuevo RUN.

```
RUN (ENTER)
```

```
Me llamo XBKC-2
-----
Esta es mi mesa de trabajo
Ready
```

Hemos realizado nuestro primer programa de ordenador, colocando unos "letreros informativos". Recuerda:

- 1 En los programas las líneas que contienen las instrucciones han de estar siempre numeradas.
- 2 Estas instrucciones no se ejecutan en el momento.
- 3 Puedes poner el número que quieras a tus instrucciones, pero respetando siempre, el orden de ejecución.
- 4 Puedes dejar líneas en blanco o sin utilizar para incluir posteriormente otras instrucciones

### 3 Uso de la instrucción INPUT.

La instrucción INPUT, como ya conoces, te permite introducir datos en el ordenador por medio del teclado.

Sabes que para guardar los datos el ordenador utiliza las VARIABLES. Conviene que repases la Unidad 2, (las reglas en el empleo de las variables y asignación de variables).

Escribe en tu ordenador:

INPUT MA\$

(ENTER)

Ahora el ordenador ha escrito una interrogación en la pantalla (?), y está esperando que marques un dato. Puesto que la variable elegida tiene el símbolo (\$) al final, el dato a introducir es ALFANUMERICO.

Pulsa pues cualquier dato, por ejemplo Perro, y marca a continuación la tecla (ENTER).

```
? Perro                (ENTER)
Ready
```

Así es como está la pantalla en este instante. Investiguemos que ha sucedido.

¿Recuerdas al estudiar las variables, que para darles un contenido utilizabamos el signo (=)?

Preguntemos el valor de MA\$ borrando previamente la pantalla.

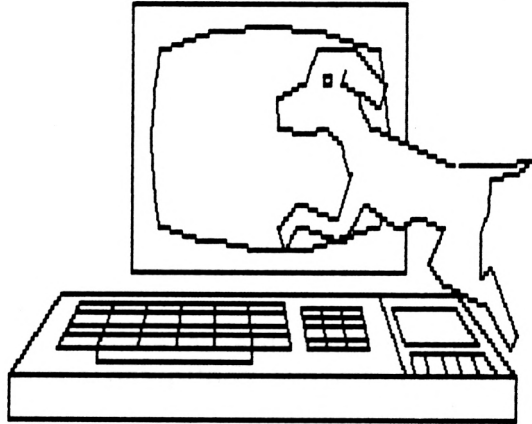
```
CLS                    (ENTER)
PRINT MA$              (ENTER)
Perro
Ready
```

La instrucción INPUT, ha realizado la "asignación" de un valor a una variable (alfanumérica en este caso).

En realidad ha actuado de igual forma que el signo (=), pero desde el teclado del ordenador. Es como si el teclado hubiera hecho de intermediario entre el nombre de



esa variable y su contenido.



Escribe:

```
INPUT "Número de manzanas ";TR      (ENTER)
```

Número de manzanas ?

Y es que, como ya comentamos anteriormente, la instrucción `INPUT` puede ir acompañada de un "letrero". Este letrero se coloca SIEMPRE entre la instrucción y la variable. Marca a continuación un número y pulsa la tecla (ENTER).

¿Qué pasaría si introdujésemos un dato alfanumérico ? Haz la prueba. El ordenador te hará algunas indicaciones.

A partir de ahora podremos continuar nuestros "experimentos" previendo ya el momento en que el ordenador va a detenerse en espera de que le suministremos datos.



## Preguntar

1

¿Qué es un algoritmo?. Pon algún ejemplo de algoritmo, haciendo una lista de las acciones de que consta.

2

¿Explica el significado de la palabra BASIC?.

3

Expón mediante un ejemplo las 3 estructuras o elementos que configuran cualquier programa de ordenador.

4

¿Qué diferencia existe entre una instrucción precedida de un número y otra introducida sin él?.

5

¿Para qué sirve la instrucción INPUT?.

6

Escribe algunos ejemplos de instrucción INPUT con:

una variable numérica, una variable alfanumérica, ídem con letrero indicativo del tipo de dato a introducir.

## UNIDAD 5 REPRESENTACION GRAFICA



Conocer

## 1 Los Organigramas.

Los organigramas son dibujos o formas gráficas que representan los pasos necesarios para la ejecución de una tarea o consecución de un fin.

Al estudiar los algoritmos se exponían, de forma narrativa, los distintos "pasos" a desarrollar para realizar un trabajo.

Un organigrama nos permite representar esos mismos pasos, pero de una forma gráfica que nos facilite el seguimiento de todos ellos.

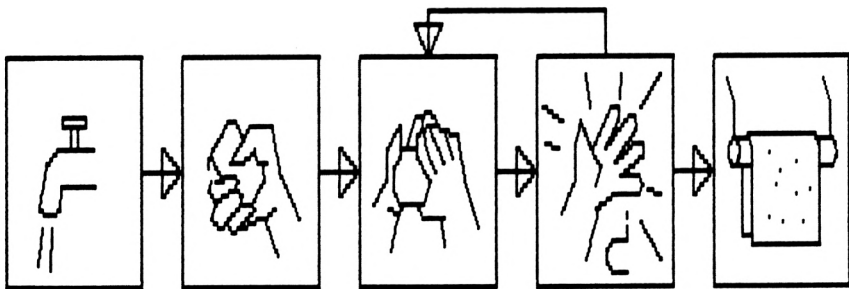
¿Recuerdas el ejemplo del algoritmo "Lavarse las manos"?. Volvamos a analizarlo:

Algoritmo : Lavarse las manos

- 1 Abrir el grifo.
- 2 Coger el jabón.
- 3 Frotarse (n veces) las manos con jabón.
- 4 Si no están limpias, volver a 3.
- 5 Secarse.

Este es un ejemplo de una tarea sencilla que contiene pocos pasos en su ejecución. Es fácil y corto de leer, pero imaginemos una lista de instrucciones que pudiéramos dar a nuestro ordenador. Podría estar compuesta de hasta mil, dos mil o más líneas como éstas.

Nos sería más fácil ver la lista de acciones de este algoritmo de la siguiente forma:



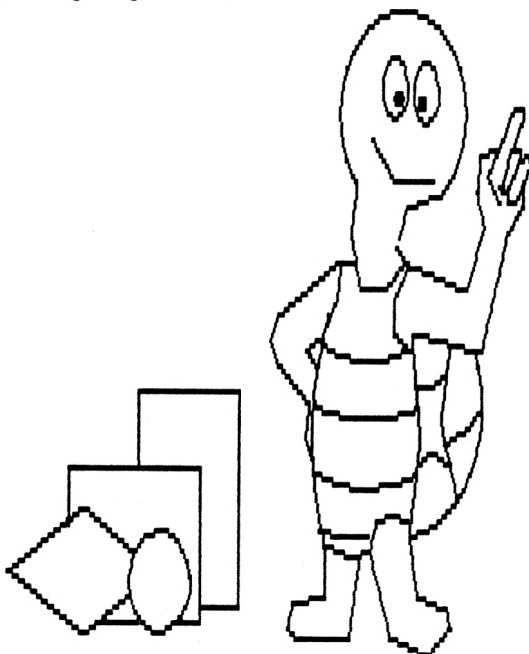
Representar cuadrados y flechas es lo más sencillo. Sin leer lo que pone dentro de ellos ya compruebas fácilmente que:

- A Las acciones a realizar son pocas (5).
- B Todas van en secuencia unitaria.
- C En la cuarta hay una posible "ruptura" de la secuencia normal.
- D La tercera recibe una flecha.

Estas pequeñas observaciones son fácilmente reconocibles en un diagrama. ¿Habrías podido detectar todo ello con la misma rapidez en la lista inicial del algoritmo?.

Claro que no. Y es que todas ellas hay que tenerlas en cuenta a la hora de realizar un programa de ordenador. Cada figura va a representar una instrucción y las flechas te servirán para enlazarlas entre sí. Por eso son tan importantes los organigramas. Recuerda: cuanto más complicados sean tus organigramas, más complejos serán tus programas.

## 2 Simbología de los organigramas.



Has estudiado ya algunas instrucciones utilizadas en el lenguaje BASIC de programación. Entre ellas: CLS, PRINT, INPUT, etc. Hemos explicado igualmente la función de dichas instrucciones. Unas tienen como objetivo determinados PROCESOS en el ordenador, tales como limpiar la pantalla (CLS) o informarnos sobre la cantidad de memoria libre (FRE); otras sirven para la SALIDA de información en la pantalla (PRINT), existiendo, por último, otras cuya finalidad es la ENTRADA de datos.

Todas las funciones del ordenador se reducen, en definitiva, a una de las tres siguientes:

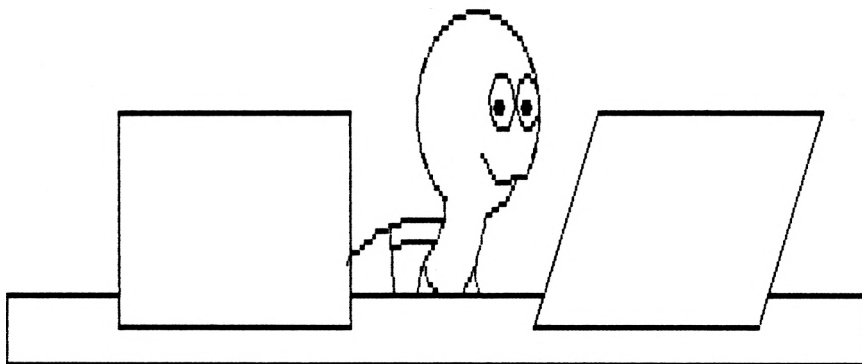
#### PROCESOS INTERNOS-ENTRADAS-SALIDAS

¿Pero qué relación tienen estas funciones con los organigramas?

El organigrama es una representación gráfica. No solamente podemos utilizar cuadrados para describir una acción a realizar en un proceso. Cualquier figura es válida si con ello conseguimos facilitar la comprensión del mismo. ¿Por qué no utilizar otras?.

Si, además de poder ver y seguir fácilmente un proceso, buscamos figuras que indiquen o tengan relación con las acciones que lo componen, tendremos a primera vista mucha más información.

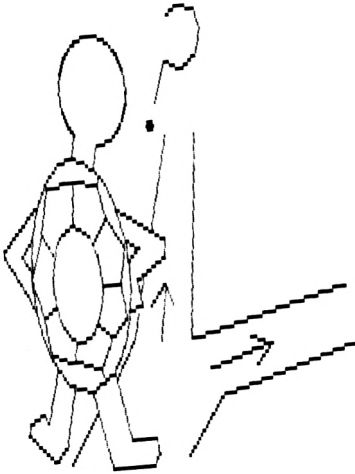
Los organigramas utilizan este sistema. Asignan a los distintos pasos de una "lista" de instrucciones una figura determinada. Así por ejemplo:



Estas dos son las figuras que con más frecuencia verás aparecer en los organigramas. Siempre que en tu lista de acciones planifiques realizar una acción simple, que no sea una entrada o salida, dibuja un RECTANGULO. Por el contrario, si tienes que expresar una recogida o salida de información, situarás el ROMBOIDE en su secuencia correspondiente.

Hay otra figura que también es utilizada con gran frecuencia en la realización de los organigramas. Se trata del ROMBO.

Muchas veces existe la posibilidad de seguir dos o más caminos distintos en las acciones que hemos de seguir. Son acciones encaminadas a la toma de una "decisión".

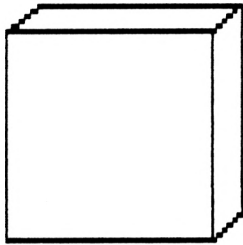


Como recordarás, en el algoritmo de "Lavarse las manos" nos planteábamos la cuestión de si éstas estaban limpias o no. En el caso de no estarlo volvíamos al punto 3. Se trataba, pues, de tomar una decisión (cambiar la secuencia de acciones) en base a dos posibilidades.

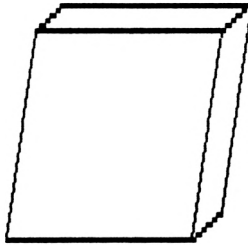
La importancia de este símbolo radica en la conveniencia de representar la existencia de dos o más opciones, de las cuales el ordenador tendrá que seleccionar una. La toma de decisiones es una de las "facultades" más notorias de los ordenadores. Ahora bien, ten en cuenta que tales decisiones dependen en todo caso de las previsiones del programador, es decir, de las instrucciones del programa.



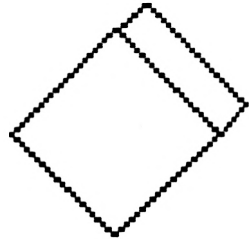
Repasemos los símbolos más utilizados en los organigramas.



proceso



entrada/salida



decisión

Sólo falta unirlos entre sí. Para ésto, nada más sencillo que la utilización de FLECHAS. Con ellas podrás indicar, dirigiéndolas de un símbolo a otro: el orden en la realización de las instrucciones, los distintos caminos a seguir y todas las intrincadas ramificaciones que tu inventiva pueda crear.

Otros símbolos auxiliares también utilizados son:

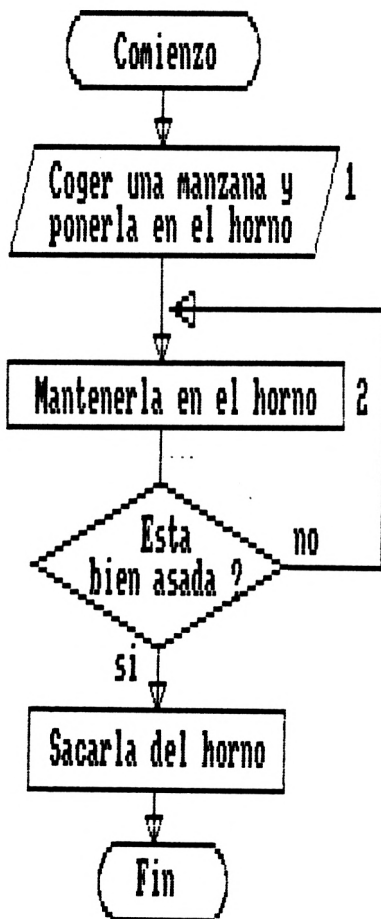


IMPRESO



CONECTOR

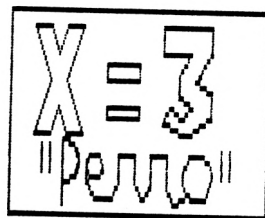
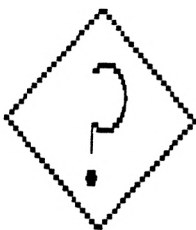
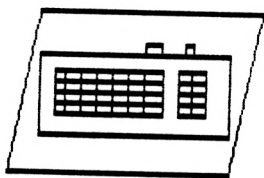
## 3 Utilización de los organigramas.



Este es un organigrama basado en una acción de cocina. Se han utilizado los símbolos que ya conoces. Veamos cómo.

En (1) la acción de coger la manzana se toma como una ENTRADA, por lo que utilizamos el símbolo de entradas y salidas. El (2) es simplemente un PROCESO y utiliza el rectángulo para su representación. En el paso (3) se nos plantea una pregunta o DECISION y en (4) ponemos el símbolo de PROCESO de nuevo (no confundirlo con el hecho de sacar la manzana=salida en este ejemplo).

Generalmente antes de hacer un programa debemos plantearnos por medio de un organigrama, los distintos pasos necesarios para su consecución. Esto nos ayudará a analizar más claramente el programa y plasmar de una forma gráfica todo el contenido del proceso.



Recuerda:

### ENTRADAS Y SALIDAS.

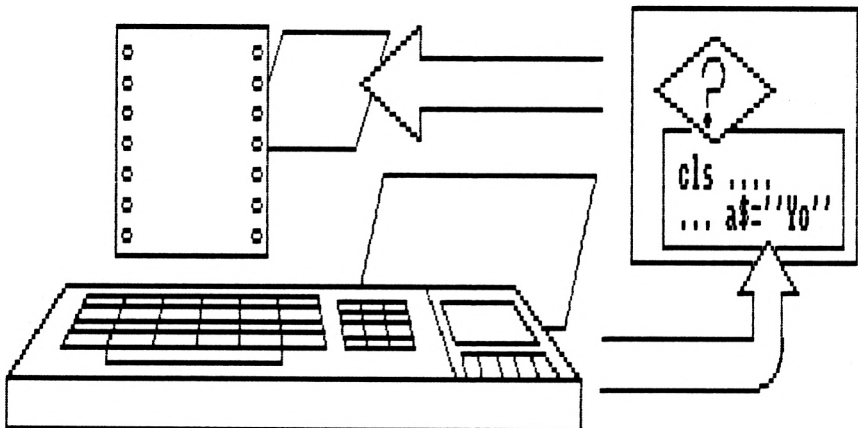
(Entradas por teclado, cintas, discos, etc. Salidas por pantalla, impresora etc).

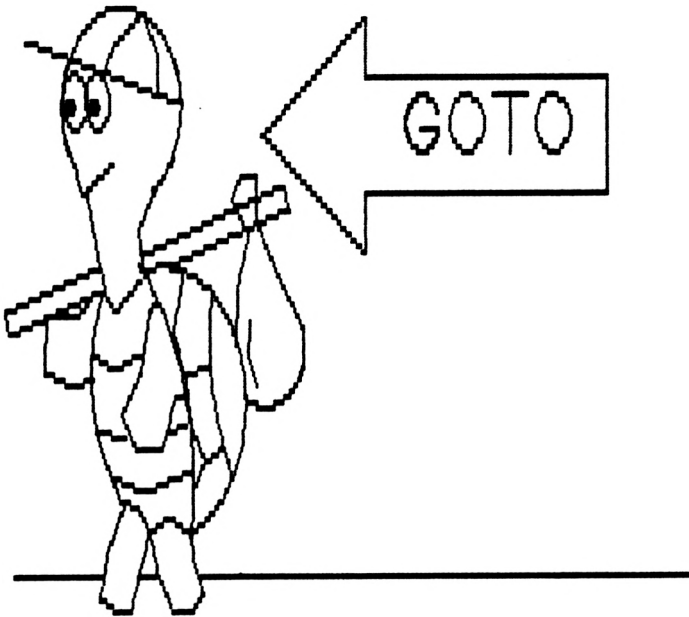
Decisiones

(Preguntas, selección de opciones).

Proceso

(Lo demás)





#### 4 La instrucción GOTO.

Conoces ya los símbolos utilizados para la elaboración de un organigrama. Vamos a utilizarlos en un ejemplo que nos permita ver su correspondencia con alguna de las instrucciones que hemos estudiado, conociendo al propio tiempo una nueva instrucción: GOTO.

La instrucción GOTO quiere decir "ve a". ¿Recuerdas que las líneas del programa llevan un número que las identifica en la lista? Pues bien, GOTO permite dirigirte a cualquiera de ellas con sólo señalar a continuación el número de la línea, (ejemplo: GOTO 3). Más adelante comprenderás con claridad su utilización.

Imaginemos de nuevo que la pantalla del ordenador va a ser nuestra mesa de trabajo. Vamos a realizar un experimento "incontrolado" para ver la velocidad que tiene nuestra máquina.

Primero plantearemos nuestro "experimento" para pasar seguidamente a elaborar el organigrama que nos permita PRACTICAR su codificación.

### 5 Experimento Nro 1. La mesa incontrolada

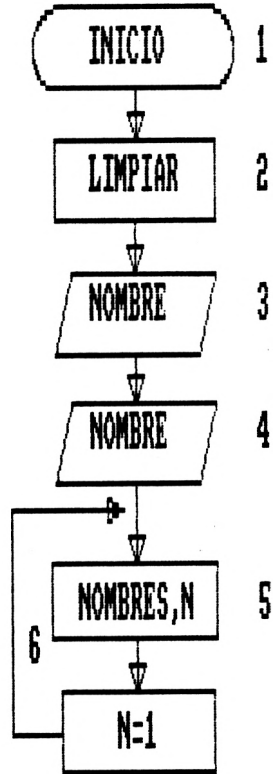
Se trata de realizar un experimento para ver la rapidez en poner y quitar de la mesa (pantalla) números y nombres de cosas. Para ello introduciremos el nombre de dos cosas en el ordenador mediante la instrucción correspondiente. A continuación los escribiremos en la mesa seguidos del número 1, después los repetiremos seguidos del 2 y así sucesivamente.

Intentemos ahora hacer el organigrama de este experimento.

Para la confección de un organigrama es necesario tener en cuenta el conjunto de las acciones que vayan a realizarse. Se requiere una gran previsión.

Veamos paso por paso cómo se dibuja un organigrama.

- 1 Colocamos el símbolo de INICIO.
- 2 Siempre conviene limpiar la mesa. Situamos, pues, un símbolo de PROCESOS.
- 3 Un símbolo de ENTRADA para el primero de los nombres que se van a introducir.
- 4 Otro símbolo de ENTRADA para el segundo nombre.
- 5 Hay que presentar en la mesa ambos nombres, pero con el número que varía a partir de 1. Puesto que se trata de un número fijo emplearemos una variable numérica N.
- 6 Los nombres han de imprimirse de nuevo. Utilizaremos una flecha que vaya a 5.



Si en este momento siguiésemos el diagrama a lo largo de sus flechas, notaríamos que lo único extraño y no muy claro es la N del número. Piensa que si nosotros pusiésemos los nombres de LIBRO y HOJA, y quisiéramos que apareciese LIBRO 1, HOJA 1 ... LIBRO 2, HOJA 2,...

no podríamos poner un número fijo, pues si no, siempre aparecería el mismo. Para ello nos serviremos de una suma. Una suma de una variable numérica.

## 6 Los contadores.

¿Recuerdas cómo se asignaba un valor a una variable?

Para dar un valor a una variable se utilizaba el signo (=) entre el nombre de ésta y su valor. Así pues si escribiésemos:

$N = 1$ , estaríamos indicando que la letra N ha recibido el valor 1. De esta manera cada vez que "sacamos" N obtendríamos 1 como respuesta. Pero hay más, podemos utilizar la propia letra N (nombre de la variable elegida, en este caso para multiplicarla, sumarla,...etc por otro número).

¿Qué pasaría si pusiésemos  $N = N + 1$  ?

Muy sencillo. Cada vez que pasásemos por esa acción el valor de la variable N aumentaría en 1. Veamos por qué:

- 1 Primera vez  $N = 1$
- 2  $N = N + 1$ , o lo que es lo mismo  $N = (1) + 1$  por lo que N ahora vale 2.
- 3  $N = N + 1$ , que internamente es  $N = (2) + 1$  por lo que N ahora vale 3.



#### 4 Y así sucesivamente....

Para comprender bien la asignación de valores a las variables debes comenzar a leerlas de derecha a izquierda. De esta forma te será más fácil saber el valor final de la misma.

Esto es lo que se llama un CONTADOR. La utilización de contadores en los programas es muy normal. Puedes estar seguro de que el 80 % de los programas que realices llevarán al menos un contador.

Ahora que sabemos como crear un CONTADOR, podemos continuar con el organigrama de "La mesa incontrolada"

Lo único que nos faltaba para completar nuestro experimento era el poder numerar correlativamente los nombres introducidos a medida que fueran apareciendo en la pantalla.

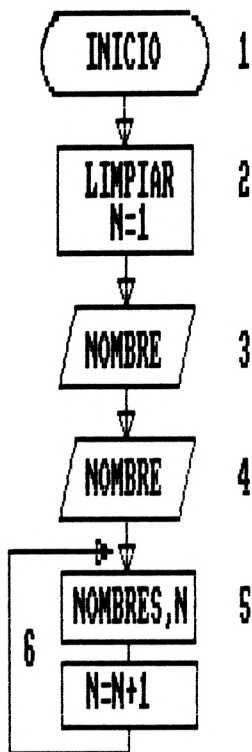
Para nuestro ejemplo hemos escogido los nombres de LIBRO Y HOJA. Con el fin de situar el contador en el organigrama, se han efectuado las modificaciones siguientes:

A. En el punto 2, aprovechando el símbolo de procesos, hemos incluido la variable N, dándole el valor 1 para la primera pasada.

B. Entre los pasos 5 y 6 se incluye el contador para que N aumente en 1.

Tanto para poner  $N=1$  como  $N=N+1$  se ha utilizado la figura del "rectángulo" puesto que se trata de PROCESOS internos del ordenador ajenos a las funciones de salida o entrada.

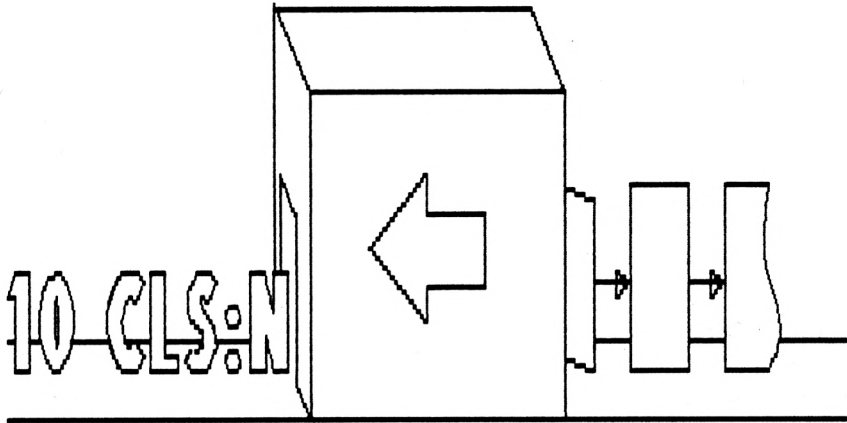
La parte teórica de nuestro experimento ha sido ya completada. Pasemos a la práctica





## Practicar

## 1 Codificación de organigramas.



"Codificar" equivale a traducir un mensaje a un código o vocabulario convencional distinto del empleado habitualmente. En realidad, el BASIC es un código de instrucciones. Resulta bastante sencillo codificar un programa previamente se ha analizado con detenimiento y su diagrama de flujo ha sido completado.

A la hora de codificar un programa es importante tener en cuenta:

- 1 Antes de comenzar, hacerse una idea completa y exacta del desarrollo del programa.
- 2 Analizar detenidamente cada uno de sus pasos.
- 3 Utilizar las instrucciones correspondientes a las funciones previstas en cada uno de los símbolos del organigrama.
- 4 Asignar números no correlativos a las líneas del programa (de 10 en 10 por ejemplo) por las posibles ampliaciones y variaciones a realizar en el mismo.

## 2 Codificación del experimento Nro. : la utilización de contadores

Comencemos, pues, la codificación de nuestro programa-experimento:

En el paso (1) aparece el símbolo de comienzo. Este no tiene codificación pues sólo sirve para indicar el principio de un organigrama.

En (2) comenzabamos por "limpiar la mesa de experimentación", es decir la pantalla. A continuación hemos incluido una instrucción para asignar el valor 1 a una variable.

Hagámoslo así:

Teclea (ENTER) una o dos veces y seguidamente:

```
10 CLS : N = 1           (ENTER)
```

Observa tres cosas:

A. Comenzamos con el número de línea 10. ¿Por qué no comenzamos con la 1?. Es muy sencillo, tenemos hasta el número 65.000 de línea para poner instrucciones. No importa, por ello, separarlas de 10 en 10, lo que nos permitirá, como antes se ha dicho, introducir líneas intermedias si fuera necesario.

B. En una misma línea hemos escrito dos instrucciones. Su finalidad no es otra que aprovechar espacio, pues  $N = 1$  podría haberse puesto en la línea 11. La ejecución de las instrucciones tiene lugar de izquierda a derecha, en el mismo sentido en que se lee. Con esta línea primero se borrará la pantalla (CLS) y después se asignará a la variable N el valor 1.

C. La separación de instrucciones en la misma línea se efectúa mediante los dos puntos (:). Continuemos. Ahora hay que introducir un nombre. Se trata de una entrada. Instrucción INPUT. ¿Qué tipo de variable es?... Alfanumérica. Asignemos, pues, un nombre a nuestra variable terminado con el signo \$. Por ejemplo OBJEP \$.

Tecleamos:

```
20 INPUT OBJEP$
```

```
(ENTER)
```

El siguiente símbolo con el que nos encontramos es también de ENTRADA/SALIDA. Al igual que antes, habrá que introducir otro dato por el teclado. Busquemos otra denominación para esta nueva variable alfanumérica, cuyo contenido será el segundo nombre de nuestro experimento.

```
30 INPUT OBJES$           (ENTER)
```

Advierte cómo utilizamos un nombre de variable muy parecido al anterior. Sólo cambiamos la última letra: P (de primero) y S (de segundo).

En el paso número (5), presentamos sobre la mesa (pantalla) el nombre de ambos objetos seguidos, cada uno de ellos, del número correspondiente a las veces que se hayan escrito. Instrucción PRINT. Teclea entonces:

```
40 PRINT OBJEP$ ; N : PRINT OBJES$ ; N
```

Recuerda que podemos poner dos instrucciones o más en la misma línea. El punto y coma (;) sirve sólo para separar el nombre de dos variables en el programa, pero en la pantalla aparecerán juntas sin signo alguno intermedio.

Punto (6). Variable de contador. Este hará que la variable N incluida en la instrucción PRINT anterior, tenga un valor distinto cada vez que el ordenador ejecute la línea 40.

```
50 N = N + 1
```

En el organigrama no nos quedan más símbolos,pero

!existe una flecha que va hacia el punto 5!. Hay que incluir, pues, en el programa alguna instrucción cuya función permita volver de nuevo al punto 5 del organigrama (línea 40 del programa). Esta función corresponde a la instrucción GOTO. Anota:

```
60 GOTO 40
```

Comprobemos si nuestro programa está correctamente escrito. Marca los siguientes comandos.

```
(ENTER)
CLS                               (ENTER)
LIST                              (ENTER)
```

Y aparecerá en tu pantalla:

```
10 CLS : N = 1
20 INPUT OBJEP$
30 INPUT OBJES$
40 PRINT OBJEP$ ; N : PRINT OBJES$ ; N
50 N = N + 1
60 GOTO 40
```

Rectifica aquellas líneas que no coincidan con las anteriores.

Sólo falta ponerlo en funcionamiento: recuerda la instrucción RUN.

Marca esta instrucción seguida de la tecla (ENTER) y tu experimento se pondrá en marcha. Suerte.



## Preguntar

1

¿Qué son los organigramas ?.¿Cuáles son los principales símbolos que se utilizan en su elaboración?

2

Confecciona el organigrama de un algoritmo basado en una acción de la vida real.

3

¿Para qué son utilizados los Contadores?. Pon un ejemplo.

4

Indica cual de estas expresiones es incorrecta.

A)  $NU = 1 + NU$

C)  $5 = K + 1$

B)  $Ma = PP * 5 + pp$

D)  $LI = 2 + LI - 1$

5

Amplía el organigrama del experimento "La mesa incontrolada", incluyendo otro contador para el segundo objeto.



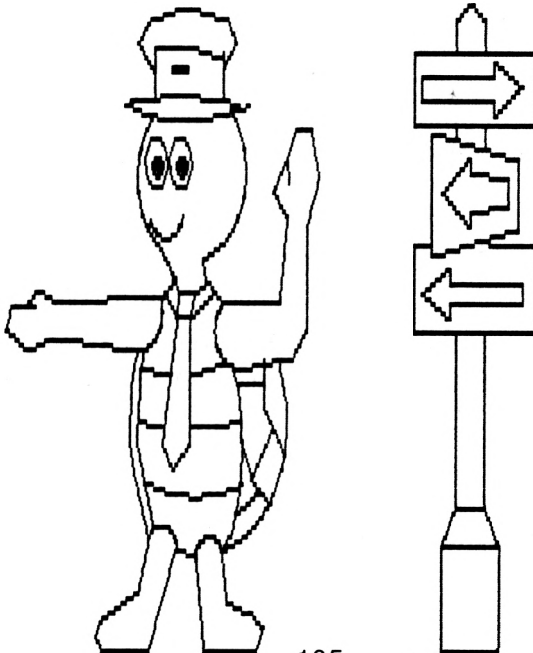
## UNIDAD 6 LAS BIFURCACIONES



Conocer

## 1 Las bifurcaciones.

Por bifurcación entendemos la división en dos ramales o brazos, de un río, árbol, camino etc. En los algoritmos seguimos una secuencia de acciones que constituye el camino "normal", pero existe la posibilidad de que en determinadas circunstancias tengamos que "desviarnos" de ese camino para tomar otros distintos.



Podemos decir que una bifurcación es la ruptura de la secuencia principal. Esta ruptura habrá de reflejarse en el organigrama para más tarde, ser trasladada al lenguaje de programación.

Recordemos el algoritmo "Lavarse las manos". Según veíamos estaba compuesto de las siguientes acciones:

- 1 Abrir el grifo.
- 2 Frotarse (n) veces las manos.
- 3 ¿Están limpias? si no ir a 2
- 4 Secarse.

Como puedes comprobar en el mismo existe una bifurcación. Esta no implica que "siempre" tengamos que salir del camino principal. En las bifurcaciones basadas en una CONDICION dependerá de que esa condición se cumpla o no. Siempre van ligadas a una pregunta.

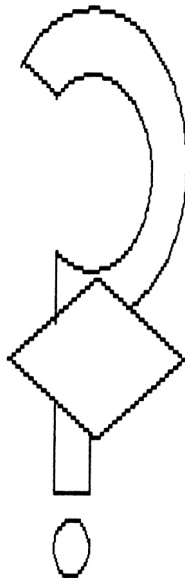
En nuestro algoritmo la bifurcación de la acción 3 viene dada por la pregunta: ¿Están limpias?. Es una bifurcación CONDICIONAL.

Existen otras bifurcaciones en las cuales no hace falta plantearse ninguna pregunta.

Tales bifurcaciones dan lugar a la inmediata elección de uno de los caminos sin necesidad de interrogarse cuál de ellos ha de seguir. Son bifurcaciones INCONDICIONALES. Veamos un ejemplo de cada una de ellas. En una parada existen varias líneas de autobuses.

INCONDICIONAL: Sería la que realiza una persona que toma siempre el mismo autobús para ir a su casa porque es el único que le puede llevar. No se plantea ninguna opción.

CONDICIONAL: Supongamos, en cambio, que otra persona puede tomar 3 autobuses, pues todos pasan por su domicilio. ¿Cuál llegará primero a la parada?



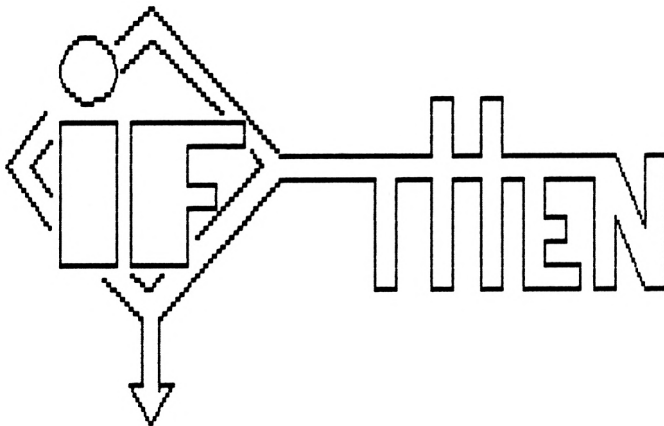
## 2 La instrucción IF...THEN.

Existen en el BASIC diversas instrucciones cuyo objeto son las bifurcaciones. La más importante de todas es IF...THEN.

De los dos tipos de bifurcaciones existentes, la instrucción IF THEN te va a permitir trabajar con las CONDICIONALES, es decir plantear preguntas dentro del programa.

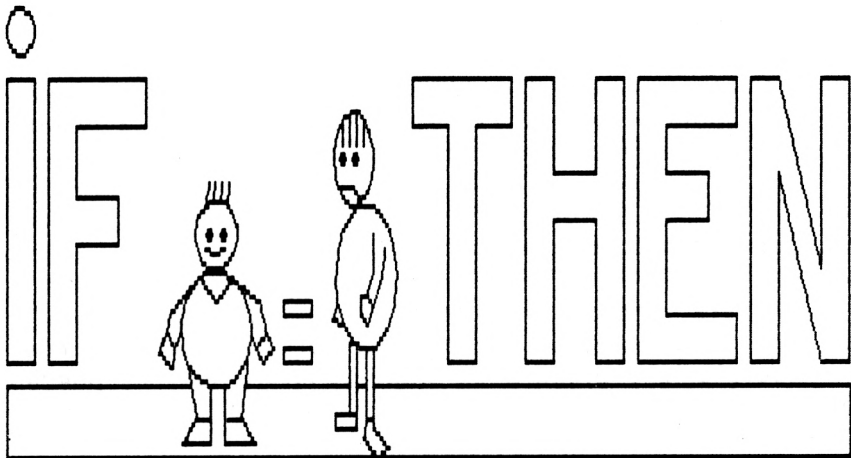
En realidad cuando hacemos una pregunta estamos comparando entre varias opciones. De igual forma podemos introducir una pregunta en nuestro programa. Para ello utilizaremos las diversas formas con que los símbolos matemáticos nos permiten efectuar comparaciones:

=	Si es igual	<>	Si es distinto
<	Si es menor	>	Si es mayor



Estos símbolos se emplean conjuntamente con la instrucción IF...THEN para formular todo tipo de comparaciones.

Si quisiéramos comparar nuestra altura con la de otra persona, la forma más sencilla de hacerlo sería acercándonos a ella. De una forma similar actúa el ordenador al utilizar esta instrucción.



Para la comparación de dos elementos IF THEN los une preguntándose si se cumple lo expresado por el símbolo existente entre los dos elementos a comparar. Por ejemplo, podríamos comparar el valor de dos variables numéricas del siguiente modo.

IF A = B THEN .....

Es bien sencillo, esto quiere decir:

- A Los elementos de la comparación se sitúan entre los términos IF (al principio) y THEN (al final)
- B Dichos elementos (en el ejemplo dos variables) se encuentran relacionados por uno de los cuatro símbolos antes señalados (en este caso = )
- C Si es verdad o se cumple la relación expresada por el símbolo correspondiente se ejecuta lo que viene a continuación del THEN.

### 3 Experimento 2. El ser extraño.

Hasta ahora sólo has realizado un "experimento " con tu ordenador. Vamos a continuar trabajando en nuestro laboratorio y probaremos algunas cosas que nos puedan ser útiles más adelante.

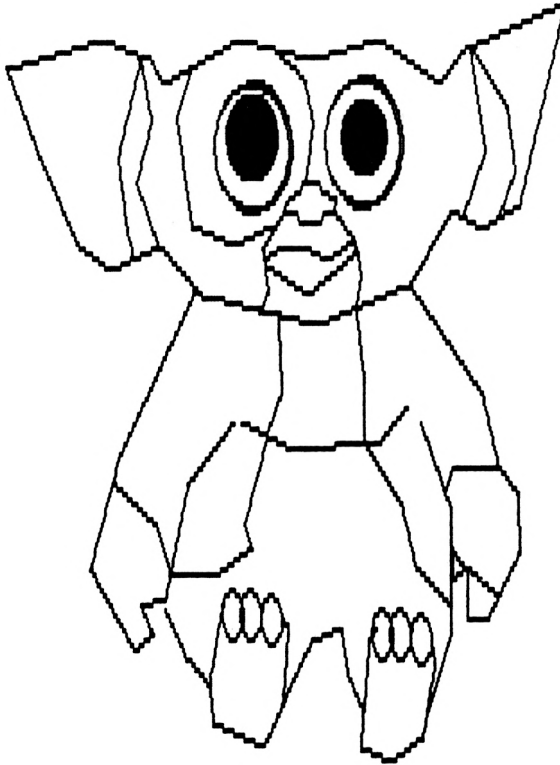
Procederemos a estudiar otro proceso que nos servirá para practicar las bifurcaciones y algunas de las instrucciones que ya hemos aprendido. Este nuevo experimento lo llamaremos "EL SER EXTRAÑO".

El director de nuestro laboratorio informático de pruebas, ha encontrado un extraño SER. Tras comprobar que no se parece a ninguna especie conocida, nos ha encargado que nos ocupemos de él. Quiere saber si está dotado de inteligencia, para lo cual debemos averiguar en el plazo más corto y por los medios disponibles si es "listo" o no.

El trabajo se nos presenta difícil dado que no tenemos mucho material en el laboratorio. Solamente contamos con los siguientes elementos:

- 1 Una urna de cristal.
- 2 Una caja grande de galletas.
- 3 Una caja pequeña vacía.
- 4 Varias bolsas de plástico.

¿A qué prueba podríamos someter a ese SER EXTRAÑO para saber si tiene inteligencia o no ?.



Tras largas horas de reflexión y grandes esfuerzos mentales, hemos llegado a la conclusión de que lo mejor es proceder de la siguiente forma:

En primer lugar cogemos a nuestro valioso espécimen con sumo cuidado y lo introduciremos en la urna. Aprovecharemos la ocasión para darle un nombre por el cual llamarle para entendernos.



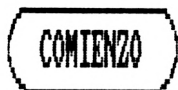
A continuación y para que no se aburra, le daremos también la cajita pequeña vacía, a fin de que la use según unas instrucciones que le vamos a dar. Si verdaderamente es inteligente tendrá que entendernos. Voy a decirle lo que tiene que hacer:

"SER EXTRAÑO, si verdaderamente eres listo y como pienso que tienes hambre, quiero que te comas las galletas que te iré dando en estas bolsitas de plástico. Te daré 10 bolsas con distinto número de galletas, pero al final tendrás que devolverme la bolsa con más galletas medidas dentro de la cajita que te he dado, porque si no, no volverás a comer".

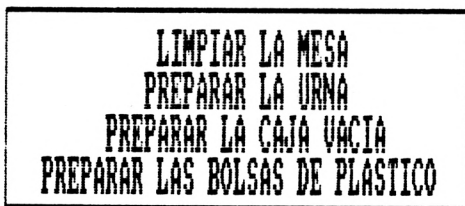
Ha llegado el momento de preparar nuestro experimento con el EL SER EXTRAÑO. Para que nada nos falle lo primero que haremos, antes de comenzar la fase práctica, es analizar teóricamente tanto nuestras acciones como las que EL SER tenga que realizar.

En primer lugar elaboraremos el ORGANIGRAMA con la meticulosidad que requiere cualquier proyecto científico.

Para iniciar el organigrama no tenemos más que dibujar el símbolo de COMIENZO con la palabra "comienzo" dentro de él.



A continuación haremos un recuento del material que vamos a utilizar. Para ello limpiaremos la mesa y pondremos todo el material encima de ella. Esta verificación no es ninguna entrada o salida de datos, sino un PROCESO. Coloquemos, pues, el símbolo correspondiente y situemos dentro de él las acciones a realizar.



Los símbolos que vienen a continuación van a representar acciones concretas. En este momento vamos a manipular los objetos que tenemos preparados.

Lo primero que haremos será introducir a nuestro SER EXTRAÑO en la urna. Dado que se trata de situarlo en el interior de la urna, el símbolo a utilizar es el de ENTRADAS/SALIDAS. A la variable la denominaremos con el mismo nombre de Urna.



Una vez dentro de la urna comenzaremos a darle bolsas con galletas. Iremos cogiendo cada vez una bolsa e introduciendo unas cuantas galletas. Hemos de controlar el número de galletas que vamos metiendo en cada bolsa para saber si, al final, el SER EXTRAÑO nos devuelve la que tiene mayor número de galletas. El símbolo correspondiente es de nuevo el de ENTRADA/SALIDA. Llamaremos BOL a la variable numérica que se encargará de anotar el número de galletas de cada bolsa.



Llegados a este punto, si el SER es inteligente se plantearía el problema de este modo: Al recibir una bolsita la compararía con la que ha guardado en la caja que le dimos para ver si tiene más galletas que ésta. La primera vez tendría que guardar la bolsita y aguantar sin comérsela, pues podría ser la que más tiene.

Evidentemente en este paso hay que poner un símbolo de BIFURCACION o condición.

Nuestro E1.7E establecerá una comparación entre el número de galletas de la bolsita que le acabamos de dar y el de las que tiene la bolsa de la caja, es decir que la comparación relacionaría dos variables: BOL por una parte y la variable que represente a la caja (a la que llamaremos CAJA) por otra. Por tratarse de una comparación el símbolo adecuado es el ROMBO.



Una vez entregada la primera bolsa, continuaremos suministrándole las demás pero sin olvidar que sólo hemos de darle 10 bolsitas. Hay que ir contándolas y como para contar lo mejor son los dedos daremos este nombre a la variable, a la cual encuadraremos, a su vez, en un RECTANGULO puesto que se trata de un PROCESO.



A rectangular box with a solid border. Inside the box, the text "DEDOS = DEDOS + 1" is written in a bold, monospaced font.

Además de contar hay que fijar otro paso para conocer si el contador ha llegado a 10. ¿Cómo?. Sencillamente comparando la variable DEDOS con el número 10.

¿ Qué ocurre cuando el contador DEDOS marca diez ? ¿ y si marca menos ?

Evidentemente si aún no ha llegado a 10 podemos continuar llenando bolsitas para EL SER.

Por el contrario si DEDOS es igual a 10 ya no le entregaremos más bolsas y será EL SER quien tenga que devolvernos la que haya guardado en la caja. Si todo ha ocurrido conforme a lo esperado, esa será precisamente la bolsa con mayor número de galletas. La función es de SALIDA. Pongamos el símbolo correspondiente con la variable a sacar (CAJA):



CAJA

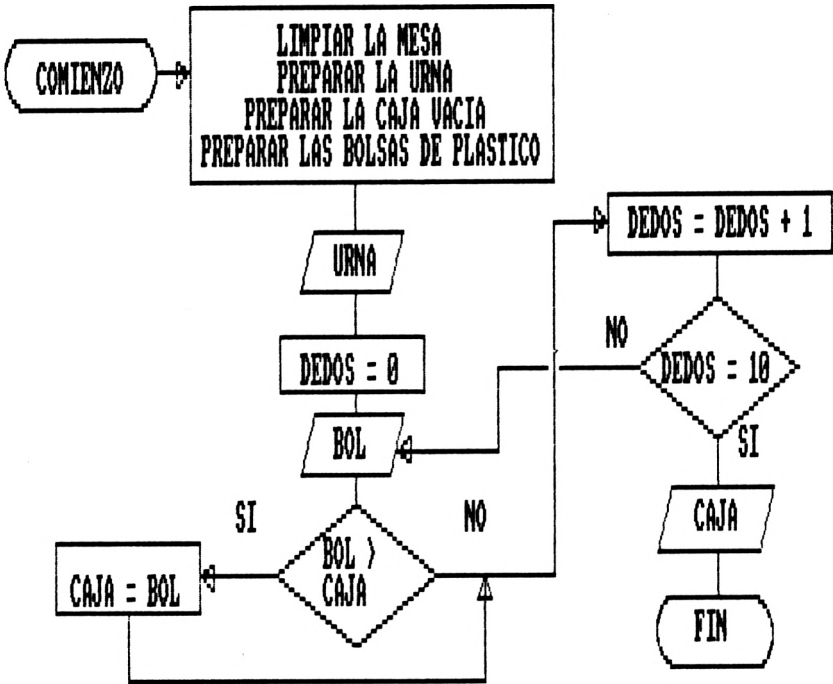
Con la impresión en pantalla del valor "CAJA" podemos dar por terminado el experimento. Solo falta llevarlo a la práctica. Situemos el símbolo de COMIENZO/ FIN de la siguiente forma:



FIN

¿Será o no inteligente nuestro SER EXTRAÑO?

Demos un repaso a todo nuestro organigrama para verificar como ha quedado:



Observa y sigue con detenimiento el significado de todas las flechas que unen los distintos símbolos.



## Practicar

### 1 Codificación del experimento 2.

El proyecto denominado EL SER EXTRAÑO va a ser puesto en marcha. Si todo el planteamiento teórico ha ido bien, no tenemos ningún problema en llevarlo a la práctica. Comencemos a introducirlo en nuestro ordenador, fiel ayuda en todos nuestros experimentos.

Cojamos el organigrama y empecemos la fase de codificación del programa. ¿Recuerdas lo que significa la palabra CODIFICAR ?

Primer símbolo: COMIENZO. Preparemos la máquina y marquemos un par de veces la tecla (ENTER). Con ésto evitamos que cualquier información anterior enturbie el nuevo programa.

Segundo símbolo: PROCESOS. Para limpiar la mesa empleamos la instrucción CLS. Preparar la urna, caja y bolsas de plástico equivale a poner a cero esas variables. Marquemos, pues, el primer número de instrucción 10 (Siempre espaciaremos las líneas de 10 en 10):

```
10 CLS: URNA$ = " ": CAJA = 0 : BOL = 0:
DEDOS =0
```

Observa:

A. URNA\$, es el nombre de la variable creada para definir la urna. En esta variable alfanumérica recogeremos el nombre de EL SER.

B. Poner = " " equivale a decir que esa variable alfanumérica no tiene contenido.No existe ningún carácter, número, letra o incluso espacio en blanco, dentro de ella.

Tercer símbolo: ENTRADA/SALIDA. En este caso se trata de una entrada porque tendremos que asignar un contenido a la variable URNA\$, que consistirá en la denominación del SER. Escribamos:

```
20 INPUT "Nombre del SER" ;URNA$
```

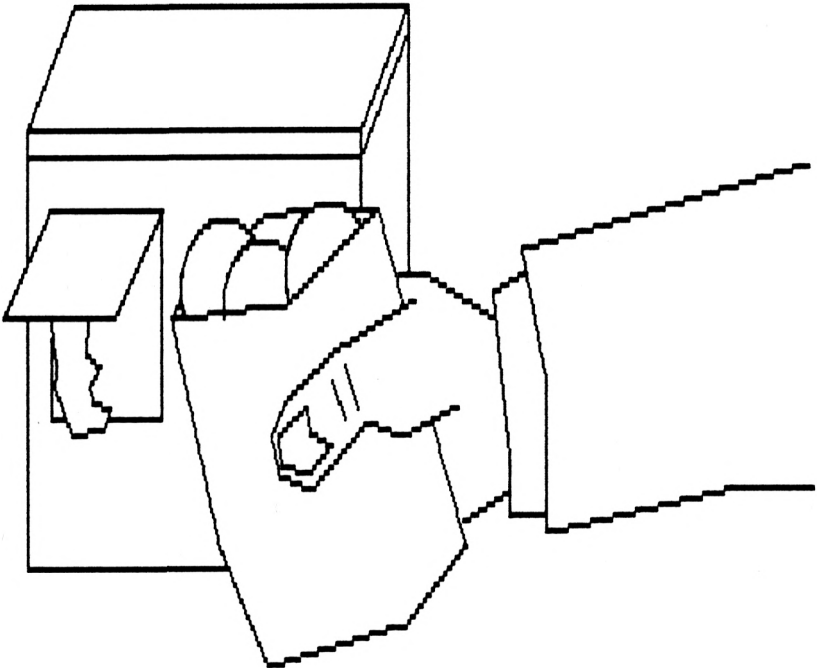
Observa:

A. "Nombre del SER" no es más que un letrero que queremos acompañe a la instrucción INPUT. Si dejásemos sólo INPUT URNA\$, en la pantalla únicamente aparecería una interrogación (?) y al existir otros INPUT más adelante, podríamos no saber a cuál de ellos se refiere.



Cuarto símbolo: ENTRADA/SALIDA. Nueva introducción de datos. Se trata de indicar el número de galletas que introduciremos en cada bolsita de plástico. Hagamos también algún letrero de ayuda:

```
30 INPUT "Bolsa de galletas ";BOL
```



Quinto símbolo: BIFURCACION. Realizamos una comparación entre la cantidad de galletas de las bolsitas (BOL) y la cantidad de galletas de CAJA.

¿Qué ocurriría en el caso de que la cantidad de galletas de BOL fuese mayor que la cantidad guardada en la CAJA ?

Sencillamente el SER podría comerse con tranquilidad las galletas de la caja y tendría que guardar la cantidad BOL en CAJA. Así ocurrirá en el caso de cumplirse la condición. Complimentamos el IF THEN de la siguiente forma:

```
40 IF BOL>CAJA THEN CAJA=BOL
```

Recuerda:

Sólo en el caso de cumplirse la condición (si el valor de la variable BOL es mayor que el valor de la variable CAJA) se ejecutará lo que viene a continuación. En caso de no cumplirse, el ordenador salta a la línea siguiente del programa.

Sexto símbolo: PROCESOS. Hemos de continuar dando bolsitas al SER hasta llegar a diez. Situemos el contador.

```
50 DEDOS = DEDOS + 1
```

Repasa los contadores.

Séptimo símbolo. ROMBO. Ahora es el momento de preguntar si el contador marca ya diez. A tal fin establecemos la comparación y bifurcación correspondientes.

```
60 IF DEDOS <> 10 THEN GOTO 30
```

Observa:

En el caso de cumplirse la condición de que la variable DEDOS no sea igual a 10, tendremos que continuar llenando bolsitas (instrucción 30). Por ello ponemos THEN GOTO 30.

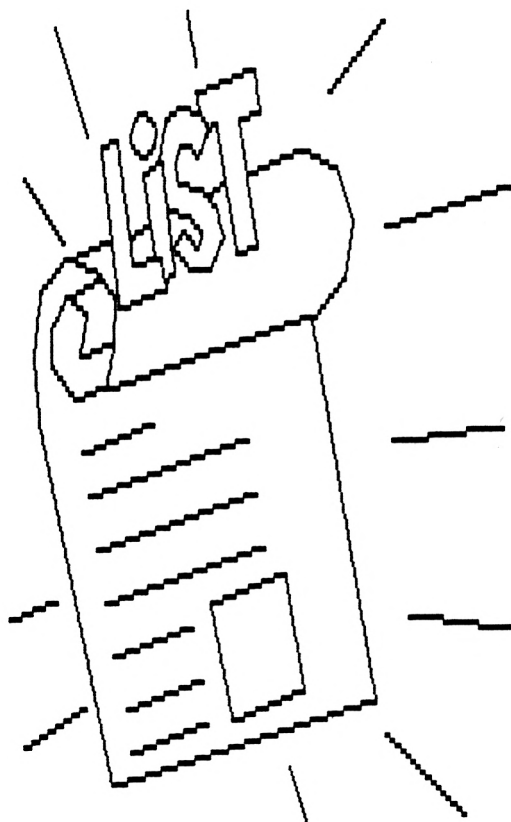
Octavo símbolo: ENTRADA/SALIDA. No se ha cumplido la condición anterior. DEDOS es igual a 10 y en consecuencia el SER debe de tener en la variable CAJA la bolsa con mayor número de galletas. Examinemos el contenido de dicha variable:

```
70 PRINT CAJA
```

Noveno símbolo: COMIENZO/FIN. La mejor forma de terminar es con un END. Pongámoslo:

```
80 END
```

Comprueba que todo tu programa está bien realizado. Para ello utiliza el comando LIST. Escribe LIST directamente sin ningún número de línea (más ENTER por supuesto) y verás aparecer en la pantalla el listado del programa.



Este es el listado completo del programa:

```
10 CLS: URNA$ = "": CAJA = 0 : BOL = 0 :  
    DEDOS = 0  
20 INPUT "SER llamado ";URNA$  
30 INPUT "Bolsas de galletas ";BOL  
40 IF BOL > CAJA THEN CAJA = BOL  
50 DEDOS = DEDOS + 1  
60 IF DEDOS < > 10 THEN GOTO 30  
70 PRINT CAJA  
80 END
```

Corrige las líneas que estén mal escribiéndolas correctamente y pon en marcha el programa mediante el comando RUN seguido de (ENTER). Si la respuesta final de CAJA coincide con la tuya EL SER es inteligente, si no, no.

¿ Lo es ?



## Preguntar

1

Explica con un ejemplo lo que es una bifurcación y señala qué tipos de bifurcaciones existen.

2

¿Para qué sirve la instrucción IF...THEN ?

3

Elabora el organigrama de un programa en el que haya que introducir el número de un día del mes y que por medio de dos instrucciones IF...THEN controle si el día está entre 1 y 31.

4

Escribe algunos ejemplos con la instrucción IF...THEN

5

Di si se cumplen las siguientes condiciones IF/THEN para los valores siguientes:

A=50 B=15 C=35

IF A>C THEN IF B+C<>A THEN IF A-B=C THEN

IF B<C THEN IF 45=A-C+B THEN IF 0<B-C THEN

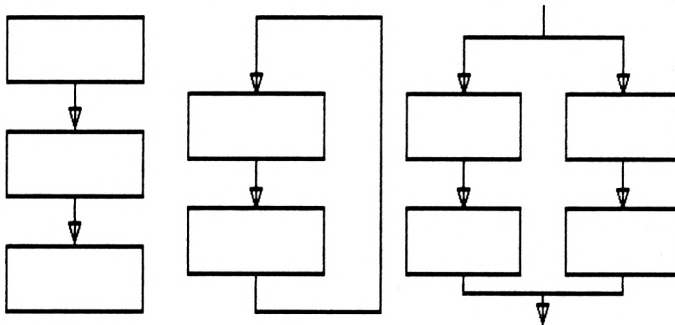
## UNIDAD 7 LOS BUCLES



Conocer

## 1 Los bucles.

Recordarás, que existen tres estructuras principales en la formación de los algoritmos. Estas estructuras, que han de reflejarse en la construcción de los organigramas, eran: La secuencia, la ruptura, y la repetición.



La Secuencia viene marcada por el orden que se establece normalmente en la ordenación encadenada de todas las instrucciones, la ruptura nos permite el cambio de la secuencia en un punto determinado del programa. IF... THEN es un ejemplo de instrucción de ruptura de secuencia.

El estudio de secuencia y ruptura se ha abordado ya en unidades anteriores. Nos queda estudiar, por último, la repetición como elemento constitutivo de la estructura de un proceso. Como claro exponente de la repetición podemos hablar de los bucles.

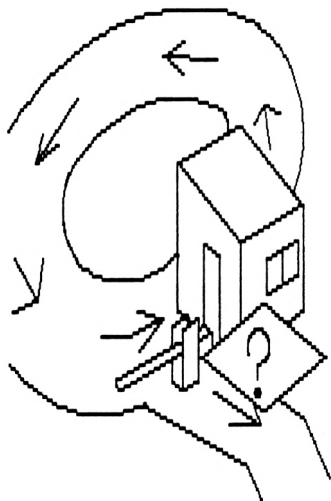
Un bucle es un conjunto de instrucciones, cuya ejecución se repite un número determinado de veces.

En todo algoritmo en el que se produce una repetición de una o varias de las acciones que lo componen, se está contemplando un bucle. En el algoritmo "Lavarse las manos", procedíamos a frotarlas varias veces hasta comprobar que estaban ya limpias. Este hecho repetitivo constituye un bucle.

En nuestro último experimento de laboratorio, determinamos dar al SER solamente 10 bolsas de galletas. La acción de llenar bolsas se repetía una y otra vez hasta que la variable DEDOS se igualaba a 10, momento en el cual el programa continuaba la secuencia principal presentando en pantalla la variable CAJA.

Como es obvio, siempre existe algún control en las tareas que requieren ser repetitivas. En "Lavarse las manos" dejábamos de frotarnos cuando estaban limpias. En el segundo experimento, terminábamos cuando contábamos hasta 10.

Todos los bucles tienen un elemento de control que indica cuándo el bucle debe terminar. ¿Qué pasaría si no existiese ese elemento?.

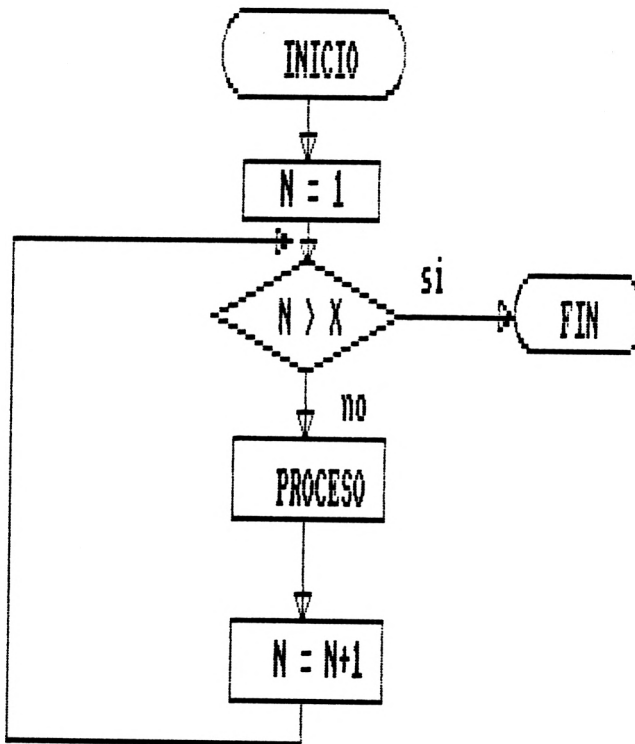




El bucle no terminaría nunca y seguiría indefinidamente hasta la desconexión del ordenador o la interrupción del programa mediante la instrucción BREAK.

Recuerda: Para el control de un bucle se necesita una condición.

Observa el gráfico siguiente. Es la estructura general de un bucle.



Como puedes observar, si sigues desde el principio el organigrama, siempre pasas por el mismo sitio. Dentro del bucle existe un símbolo de condición. Este es el control del bucle que hará que salgamos de la rutina repetitiva que lo construye.

Piensa en algún algoritmo con acción repetitiva y define cuál es la condición-control del bucle establecido.

## 2 La instrucción FOR...NEXT

Un contador sirve para "contar". Para ello utilizá-bamos hasta ahora una variable, por ejemplo DEDOS, cuyo valor aumentaba mediante la sentencia:

```
DEDOS = DEDOS + 1
```

Cada vez que pasá-bamos por esta instrucción , DEDOS incrementaba su valor en 1.

Para terminar la rutina de introducir bolsitas de galletas, usamos la instrucción IF...THEN. Cuando DEDOS era igual a 10 la rutina finalizaba continuando el programa su secuencia normal.

La instrucción FOR...NEXT nos permite repetir una acción sin necesidad de utilizar una instrucción condicional. ¿Es posible eso?.

Sí, porque la condición ya está incluida en la propia instrucción FOR..NEXT. Con una sola instrucción conseguimos el mismo efecto que antes con dos.

Veamos un ejemplo del uso de esta instrucción:

Imaginemos de nuevo que queremos repetir una acción un número determinado de veces. Utilizaremos también la variable DEDOS para contar, pero en este caso contaremos hasta 20. En el sistema de contador que conocemos, la fórmula sería la siguiente:

```
1 INSTRUCCIONES DE REALIZACION DE LA ACCION
2 DEDOS=DEDOS+1
3 IF DEDOS=20 THEN GOTO 1
4 END
```

Observa:

A. Hemos puesto una condición IF THEN para controlar el final del bucle.

B. Cuando DEDOS tiene el valor 20 no se cumple la condición (ya no va hacia 1) y se ejecuta el siguiente número de instrucción, parándose el programa.

Apliquemos ahora la instrucción FOR...NEXT al ejemplo para comprobar la facilidad de indicarlo:

```
1 FOR DEDOS = 1 TO 20
2 INSTRUCCIONES DE REALIZACION DE LA
  ACCION
3 NEXT DEDOS
4 END
```

Observa:

A. FOR DEDOS = 1 TO 20 quiere decir que la variable DEDOS comienza con el valor 1 ( = 1 ) y llegará como máximo hasta 20 ( TO 20 ).

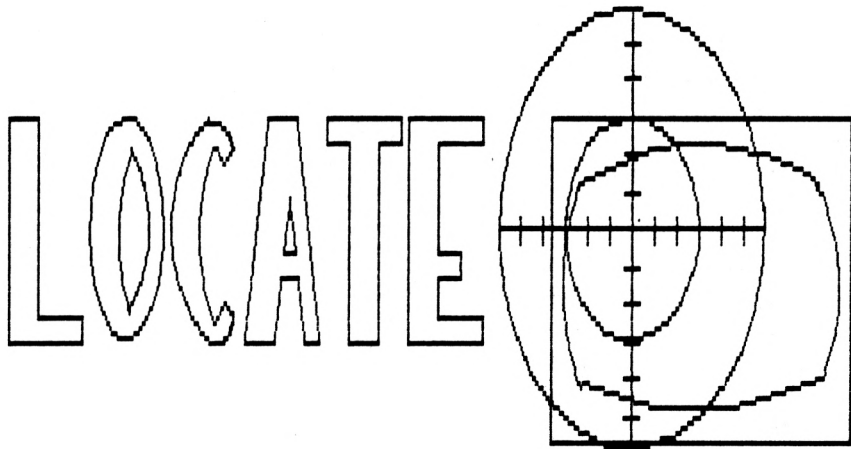
B. NEXT DEDOS aumenta la variable DEDOS en 1, mandando el control de programa a la instrucción 1 hasta que el valor de DEDOS sea 20.

### 3 La instrucción LOCATE

Habrás observado que , al utilizar la instrucción PRINT los datos aparecen en la línea siguiente a la última escrita.

Cuando en un programa existe una instrucción CLS, los datos correspondientes a instrucciones PRINT o INPUT ejecutadas a continuación de aquella se sitúan en la parte más alta de la pantalla (en la primera línea).

La instrucción LOCATE te permite situar en la parte que desees de la pantalla los datos o textos que vayan a mostrarse en la misma.



La pantalla está dividida en filas y columnas. Para situarte en la misma con la instrucción LOCATE deberás indicar en qué horizontal y en qué vertical quieres situar el primer carácter. Por ejemplo:

```
LOCATE 20,5 : PRINT "Aquí"
```

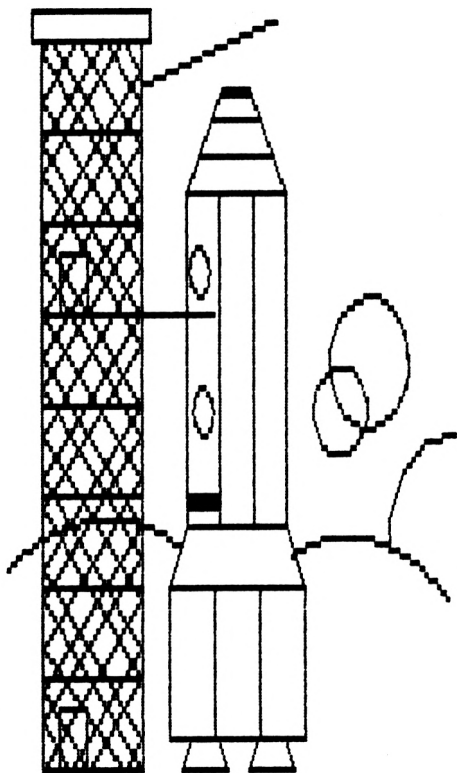
### 3 Experimento 3. Proyecto inter-espacio.

Después de estudiar los bucles y las instrucciones que permiten su ejecución práctica nos encontramos en disposición de acometer tareas más complejas como la que seguidamente se plantea.

Nuestro laboratorio se encuentra en estos momentos con los últimos avances tecnológicos necesarios para emprender las más difíciles pruebas y experimentos.

El departamento de experimentación ha llegado a la conclusión de que el próximo paso a seguir sea el desarrollo de un proyecto de gran envergadura:

La fabricación y puesta en funcionamiento de un cohete espacial. Este proyecto se denominará "INTER-ESPACIO", cuyos objetivos son los siguientes:



1. Fabricación y puesta a punto de la nave con el material suministrado en la lista que se especificará.
2. Lanzamiento el día DD, hora HH.
3. La distancia a recorrer por la nave dependerá de la cantidad de combustible líquido suministrado en el momento del despegue.
4. Caída libre al mar una vez agotado el combustible

El proyecto INTER-ESPACIO requiere un material muy sofisticado. Todos los elementos que se utilizarán han sido cuidadosamente elegidos. Su comprobación se ha de realizar meticulosamente. La lista se compone de:

- 1 DEPOSITO DE COMBUSTIBLE.
- 2 CONTADOR DE DEPOSITO.
- 3 LOCALIZADOR DE VUELO.
- 4 CONTROLADOR DE HORA.

Operaciones ha realizar durante el proceso:

1. En primer lugar se procederá a llenar el depósito de combustible líquido. Para esta tarea se ha estimado que la capacidad conveniente es de 35.000 unidades. También fijaremos la hora definitiva del lanzamiento según el último parte meteorológico.

2. Más adelante se introducirá la hora de lanzamiento. En el caso de que ésta sea igual a la elegida como hora HH, la nave despegará.

3. A continuación se procederá a situar la nave en su lugar de lanzamiento. Para ello utilizaremos la pantalla de nuestro ordenador como elemento de seguimiento de la nave.

4. Comienza a funcionar el localizador de vuelo entre los márgenes fijados por nuestra pantalla (de 1 a 80). El vuelo continuará hasta que se agote el combustible. Ten en cuenta en este punto que el combustible se quemará a razón de 500 unidades por cada paso horizontal de pantalla.

5. Cuando el combustible esté por debajo de cero, el motor se parará y la nave se perderá en el espacio.

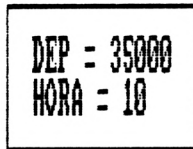
Puede parecerle muy complejo, pero verá que fácilmente desarrollamos nuestro proyecto. Veamos como hacer el organigrama.

En primer lugar colocamos, como siempre, un símbolo de comienzo. Una vez hecho esto, consultamos nuestro manual de operaciones y leemos las primeras instrucciones, éstas dicen: "llenar el depósito de combustible con 35.000 litros y fijar la hora de lanzamiento". Son PROCESOS puesto que lo que haremos será asignar a distintas variables el valor que les corresponda. El siguiente símbolo a situar es, pues, un rectángulo.





COMIENZO

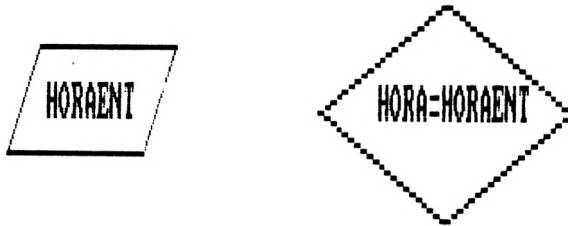


DEP = 35000  
HORA = 10

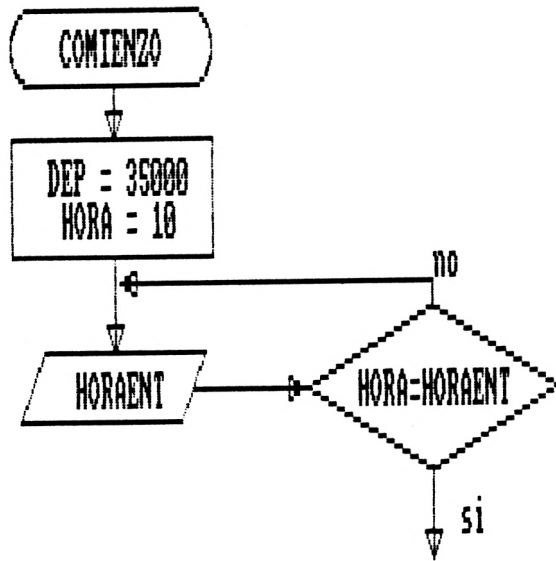
Observa que dentro del rectángulo hemos colocado dos variables: por una parte DEP a la que corresponde el valor 35.000. Y por otra HORA, a la que asignamos el valor 10. No obstante puedes fijar la hora de lanzamiento que creas más oportuna.

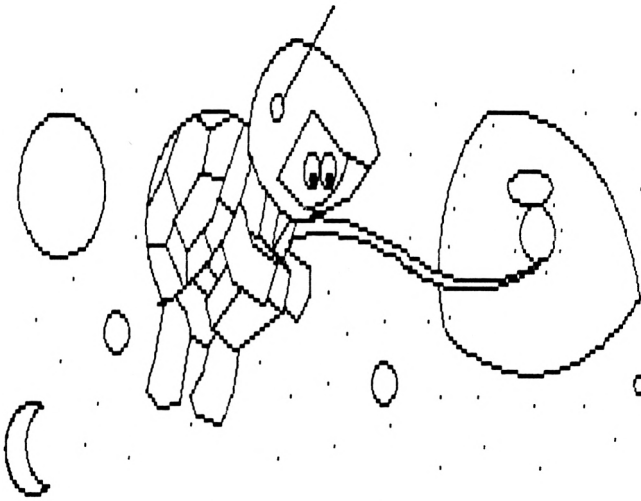
Continuando con el proyecto INTER-ESPACIO, llegamos al segundo punto de nuestro manual de operaciones, el cual indica que hemos de introducir la hora para proceder al despegue". Hay que dibujar sin lugar a dudas un símbolo de ENTRADA/SALIDA. A la variable correspondiente la denominaremos HORAENT.

Llegados a este punto, comprobaremos la hora de lanzamiento. Para ello verificaremos si esta HORA coincide con la que en ese momento se acaba de introducir HORAENT. Hay que comparar ambas variables y para ello utilizaremos el símbolo del ROMBO:



Hasta ahora nuestro organigrama aparece como sigue:

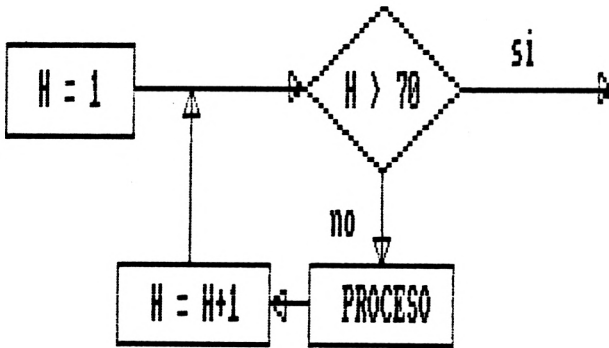




Pasadas las etapas preliminares de nuestro proyecto INTER-ESPACIO, nos disponemos a comenzar la fase más compleja: el despegue. Analicemos:

- 1 La nave tiene que recorrer la distancia que hay desde el extremo izquierdo de la pantalla hasta el extremo derecho de la misma. Como el MODO pantalla (recuerda MODE 2) es de 80 columnas, existen ochenta posiciones que tiene que recorrer.
- 2 El movimiento de la nave consistirá en un desplazamiento de izquierda a derecha a través de las distintas posiciones que recorrerá de una en una un número determinado de veces (acción repetitiva).

Lo más sencillo para ejecutar una repetición es utilizar un bucle. Representemos un bucle que cuente desde 1 hasta 70 (alguna posición menos que 80) con condición y contador implícitos:



En el bucle anterior se han situado varios símbolos:

Comparación:  $H > 70$ . Es el control horizontal para el desplazamiento de la nave. Cuando llegue a 71, el proyecto habrá finalizado.

Procesos: Dentro de este símbolo se van a realizar las más importantes operaciones de la nave. Las veremos dentro de un momento.

Procesos: En el segundo proceso la función a desarrollar es solamente de contador:  $H = H + 1$ . Nos servirá para ir controlando el vuelo de la nave (LOCALIZADOR DE VUELO).

Pasemos a continuación a analizar las instrucciones que deben de componer el primer símbolo de procesos.

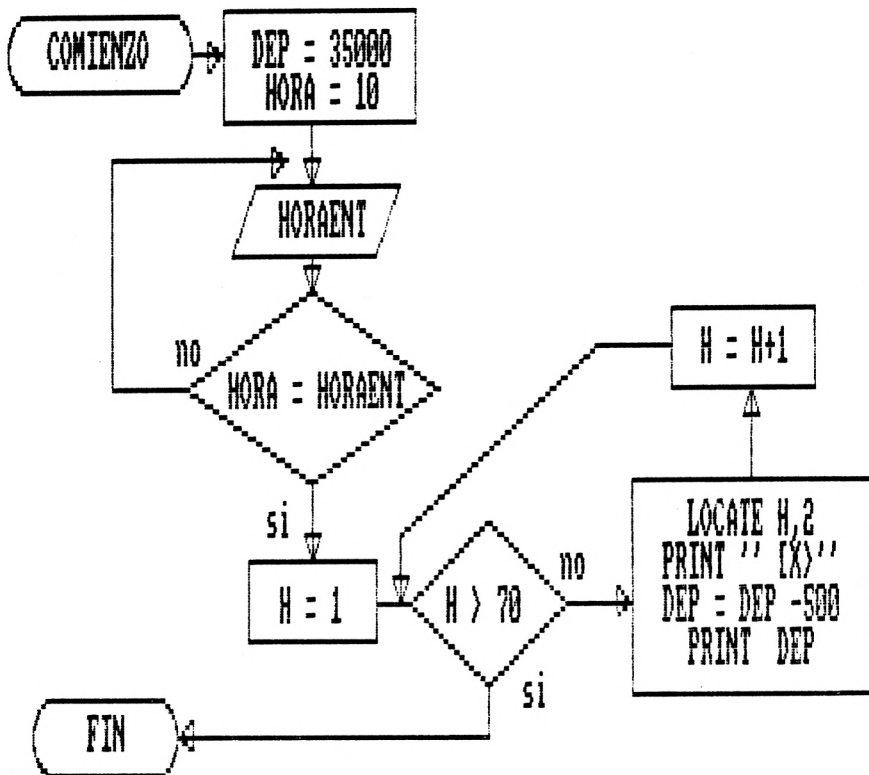
1 Para visualizar la nave en los puntos horizontales de la pantalla, nada mejor que la instrucción PRINT. Ahora bien, la nave no puede aparecer en cualquier lugar y por ello nos serviremos de LOCATE colocada antes de PRINT para hacer que la nave se desplace de izquierda a derecha.

2 Queremos ir viendo también el combustible que queda, para lo cual utilizaremos otro PRINT.

3 El CONTADOR DE DEPOSITO es otro de los elementos que necesitamos.  $DEP = DEP - 500$  será la instrucción que nos sirva para restar, cada vez que pasemos por esta línea 500 unidades al depósito de la nave.

```
LOCATE H,Z. PRINT " (X)"
DEP = DEP-500. PRINT DEP
```

Analicemos el organigrama completo de nuestro proyecto. En este momento ha de ser el siguiente:



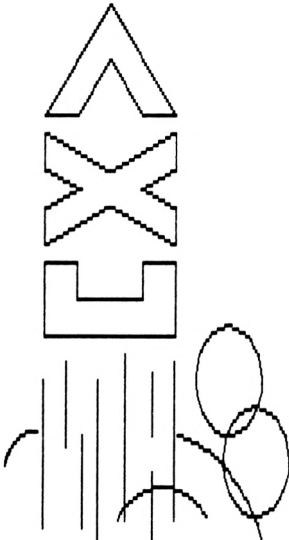


## Practicar

### 1 Codificación del experimento 3.

El proyecto INTER-ESPACIO, está preparado para su fase de ejecución. Comenzaremos por situar todos los elementos que han sido estudiados en la forma indicada en nuestro organigrama. Analizaremos paso a paso cada uno de los símbolos y procederemos a su codificación en el ordenador.

Primer símbolo: Comienzo. Preparamos nuestra máquina para la escritura del programa. Pulsamos una o dos veces la tecla (ENTER) y MODE 2.



Segundo símbolo: Rectángulo. Con el número de línea correspondiente, asignamos valor a variables que se indican en él.

```
10 DEP=3500:HORA=10:CLS
```

No olvides pulsar (ENTER) al finalizar de escribir cualquier línea de instrucciones.

Tercer símbolo: Romboide. Nos indica una entrada. La variable HORENT es la elegida para recoger la hora.

```
20 INPUT "Indicar la hora "; HORENT
```

Observar el letrero que acompaña a la instrucción INPUT.

Cuarto símbolo : Rombo. Pregunta o instrucción condicional con la que, en nuestro programa, controlamos la hora de salida estableciendo una comparación entre ésta y la que se introduce por teclado.

```
30 IF HORA <> HORAENT THEN 20
```

En esta línea si la condición se cumple, o sea si el valor de HORA (10) NO ES IGUAL al valor asignado a HORAENT, el programa vuelve a la línea 20. Sólo en el momento en que la condición no se cumpla, es decir cuando introducido para HORAENT sea 10, continuará su curso el programa.

Quinto símbolo: Rombo. Se trata de otra condición, pero en este caso no utilizaremos una instrucción IF. No hace falta. Es el comienzo de un bucle que va desde 1 hasta 70 (movimiento de la nave). Escribamos la primera parte de la instrucción FOR... NEXT.

```
40 FOR H = 1 TO 70
```



Recuerda:

- A Con la anotación H=1 queremos indicar al ordenador que comience a dar valores a la variable numérica H (podía ser otra) a partir de 1.
- B La expresión TO 70 señala el final del bucle. Es el valor máximo que tomará H y hasta no alcanzar dicho valor se repetirán todas las acciones comprendidas entre el FOR y la instrucción NEXT que marca su final.
- C A continuación indicaremos las acciones repetitivas, como son el movimiento de la nave y el contador del depósito.

Sexto símbolo: Rectángulo. Aquí se enmarcan todas las acciones que se producen dentro del bucle creado por la instrucción FOR...NEXT. Son las siguientes:

```
50 LOCATE H,2: PRINT "[ X>"
```

Significa que, desde el primer valor de H hasta el último (1 hasta 70), se van a ir imprimiendo sucesivamente en la horizontal de la pantalla los caracteres entrecorridos de la instrucción PRINT. La vertical no varía; se ha fijado en 2 con la cual los caracteres de PRINT aparecerán siempre en la línea segunda.

```
60 DEP = DEP - 500
```

En cada pasada dentro del bucle se restan 500 unidades de combustible al valor contenido en DEP.

```
70 PRINT DEP
```

De esta forma podemos ver la cantidad de combustible que nos queda en cada momento.

Septimo símbolo: Rectángulo. Es el contador de H. Podemos ver  $H = H + 1$ . No vamos a utilizar este tipo de contador. Como recordarás el bucle FOR...NEXT realiza esta tarea sin necesidad de indicárselo. Sólo con poner el final del bucle conseguimos que H aumente su valor de uno en uno. Escribe:

```
80 NEXT H
```

Con ello indicamos "siguiente valor de H", es decir que aumente en una unidad el valor anterior.

Sólo nos queda poner fin a nuestro programa por medio de la instrucción END (fin), indicando al ordenador que ha llegado al final de la ejecución del programa y del proyecto INTER-ESPACIO. Escribe:

```
90 END
```

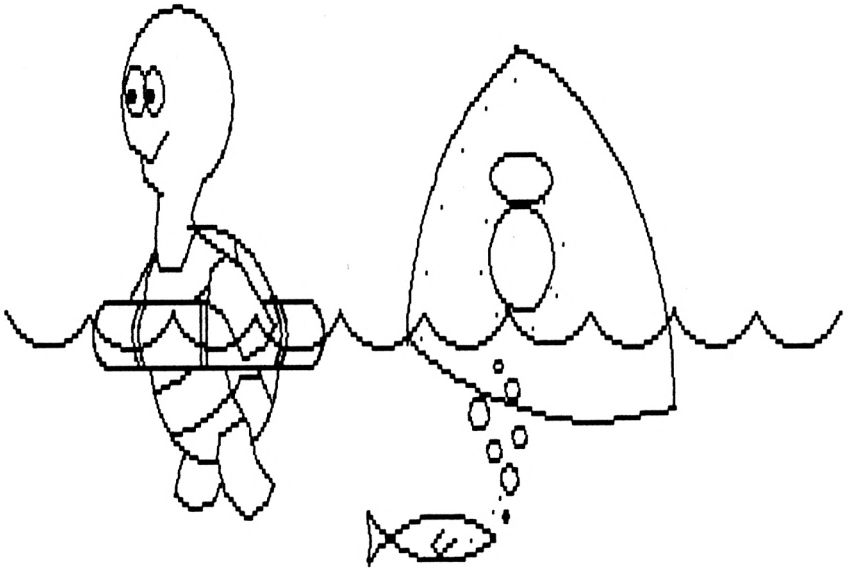
Estamos preparados para el lanzamiento de nuestra nave. No obstante vamos a comprobar antes si se encuentra a punto. Escribe:

```
LIST
```

El listado de tu programa habrá de coincidir con el que a continuación se presenta:

```
10 DEP=35000:HORA=10:CLS
20 INPUT "Indicar la hora";HORAENT
30 IF HORA <>HORAENT THEN 20
40 FOR H=1 TO 70
50 LOCATE H,2:PRINT " [ X>"
60 DEP=DEP-500
70 PRINT DEP
80 NEXT H
90 END
```

¿Coincide tu programa?. Corrige, en su caso las líneas erróneas volviéndolas a escribir completas.



Hemos terminado todas las fases de estudio y preparación para el lanzamiento de nuestra nave. Ha llegado el momento de ponerla en órbita. El proyecto INTER-EXPACIO llega a su fin. Procedamos a la prueba de despegue:

RUN

"Indicar la hora"

Se ha limpiado la pantalla y en estos momentos se nos pregunta por la hora de despegue. Si ésta coincide

con la establecida previamente (10), la nave partirá hacia el espacio. Prueba a introducir varias cantidades.

!Qué rapidez! La nave ha salido disparada hacia la derecha de la pantalla. Casi no hemos podido ver su trayectoria, pero el proyecto INTER-ESPACIO ha sido un gran éxito. !Enhorabuena!.

Los técnicos de nuestro proyecto han decidido repetir el experimento para tratar de controlar la velocidad punta de la nave. Quieren introducir un pequeño dispositivo de retardo de la velocidad. ¿Qué instrumento puede ser éste?.

Observa que el movimiento de la nave se ha producido al repetir 70 veces la instrucción PRINT. La primera vez la instrucción LOCATE tenía los valores 1,2, la segunda vez 2,2, después 3,2 ...etc. Al ejecutarse un PRINT en una posición seguido de otro en la posición de la derecha y así sucesivamente, la nave ha ido moviéndose.

El bucle ha ido muy rápido y la nave también. ¿Y si introdujéramos otro bucle dentro del que ya hay? Escribe:

```
FOR H=1 TO 3000:NEXT H      (ENTER)
```

Observa que la máquina ha estado un rato ocupada y después ha escrito Ready. Esto quiere decir que ha estado contando desde 1 hasta 3000.

Vamos a escribir un retardo dentro de nuestro programa. Vuelve a listarlo mediante la instrucción LIST.

Las líneas del programa están numeradas de diez en diez. Entre línea y línea podemos, pues, introducir otras (hasta nueve) para incluir nuevas instrucciones. Escribamos el bucle retardador entre las líneas 70 y 80:

```
75 FOR B=1 TO 500:NEXT B
```

Con hacer que el ordenador cuente hasta 500 en cada movimiento de la nave, será suficiente para que ésta vaya más despacio.

Marca de nuevo el comando RUN y observa cuán distinto es ahora el vuelo de la nave.



## Preguntar

- 1 Explica lo que es un bucle.
- 2 Escribe un algoritmo tomado de la vida real en el que se encuentre una acción repetitiva.
- 3 Haz el organigrama del algoritmo anterior.
- 4 ¿Qué elemento de control llevan incorporados los bucles ?
- 5 Explica para qué sirve la instrucción FOR ... NEXT
- 6 Pon un ejemplo para la utilización de la instrucción FOR..NEXT
- 7 Señala la función de la instrucción LOCATE. ¿Que parámetros la acompañan?.



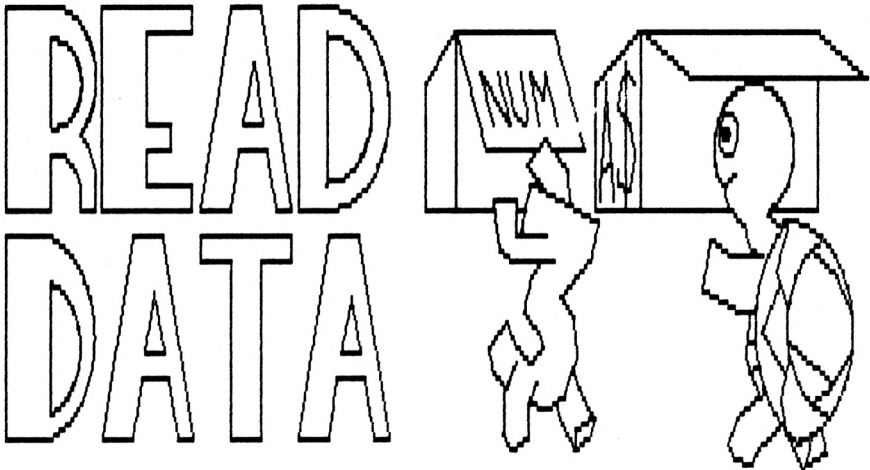


## UNIDAD 8 OTRAS INSTRUCCIONES



Conocer

## 1 La Instrucción READ/DATA



Existen algunos datos que normalmente vienen indicados dentro del mismo programa. Recuerda como para la variable HORA, nosotros introducíamos el valor 10 dentro de la misma línea de instrucción del programa.

La instrucción READ en combinación con la palabra DATA, nos permite hacer asignaciones o dar valores a variables dentro de nuestros programas. Imagina por un momento, si tuviésemos la necesidad de dar varias horas a otras tantas variables.

Busquemos cinco variables :

HORA1, HORA2, HORA3, HORA4 Y HORA5

Pues bien, si nosotros quisiéramos asignar el valor de una hora a cada una de estas variables, necesitaríamos hacer las siguientes indicaciones:

HORA1=8, HORA2=9, HORA3=10, HORA4=11,  
HORA5=12

La instrucción READ/DATA, nos permite hacer de otra manera la introducción de estas cantidades en cada una de estas variables en dos pasos muy sencillos. Veamos su aplicación en el presente caso:

1 Ponemos la instrucción READ, seguida de las variables a cargar y separadas por comas.

```
READ HORA1,HORA2,HORA3,HORA4,HORA5
```

2 A continuación, y dentro del programa, se introducen con la instrucción data los valores en el mismo orden que el expuesto para cada una de las variables.

```
DATA 8,9,10,11,12
```

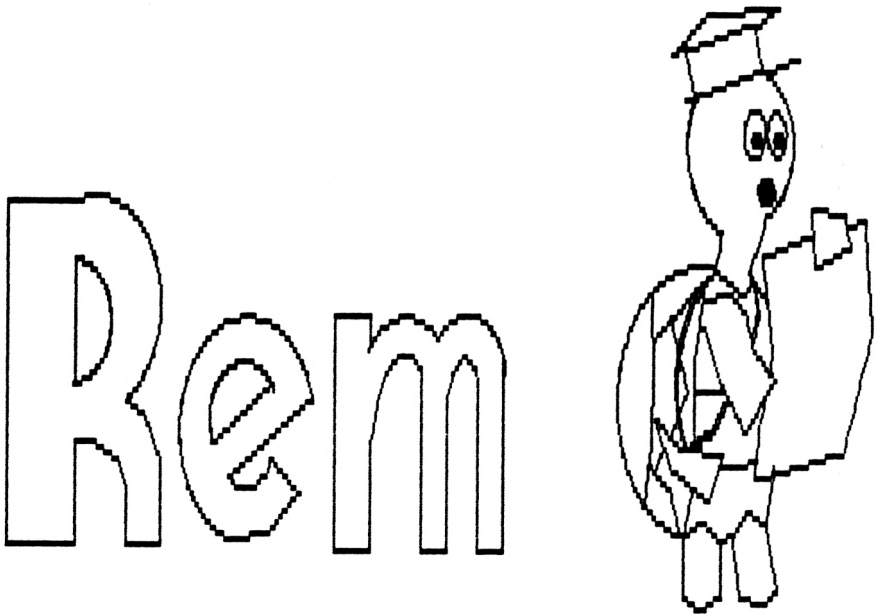
Con estas operaciones, en el momento de ponerse en funcionamiento el programa, la instrucción READ, se encargará de poner cada uno de los valores indicados en la DATA, en cada una de las variables que se la han indicado, de esta forma, HORA1 toma el valor 8, HORA2 toma el valor 9, HORA3 toma el valor 10, HORA4 toma el valor 11 y por último HORA5 toma el valor 12.

Recuerda que la DATA, con los valores, no tiene por que ir antes en el programa que READ. Esta especial instrucción puede incluirse en cualquier parte de tu programa. Más adelante veremos algunos ejemplos prácticos.

## 2 La instrucción REM.

El nombre de REM, viene de REMARK, es decir, marca o indicación. Esta instrucción sólo tiene como finalidad su utilización en el programa, para indicar letreros de la acción o grupo de acciones que vienen a continuación.

Esta instrucción no produce ninguna entrada, salida o proceso dentro de la operativa del ordenador, solamente sirve de indicativo al propio programador de las instrucciones en una parte del programa.



Veamos un ejemplo:

```
10 REM INSTRUCCIONES PARA ENTRADA  
20 INPUT "Dame tu nombre ";NO$
```

Observa:

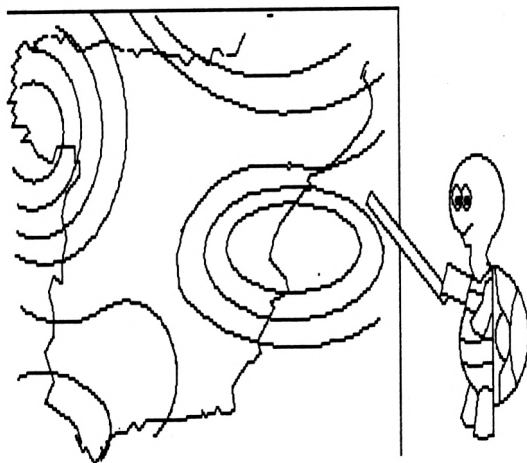
A La instrucción REM, va seguida del texto, que deseamos aparezca en el programa.

B La instrucción REM, no es operativa, es decir, no produce ningún resultado.

### 3 Experimento nro. 4. Contestador meteorológico.

En nuestro laboratorio nos hemos propuesto realizar una prueba de los utensilios que hasta ahora nos han proporcionado. Hasta el momento, contamos con casi todas las herramientas necesarias para trabajar en el lenguaje BASIC.

Poco a poco, iremos ampliando y conociendo las instrucciones para realizar procesos que requieren secuencia en las acciones, alternativa o bifurcación y repetición en las mismas.



El proyecto que nos plateamos, va a requerir la utilización de todos estos elementos. Para ello nos hemos propuesto realizar la fabricación de un elemento muy sofisticado para su utilización por el servicio meteorológico de la capital.

Este artilugio, servirá para dar información general a cualquier ciudadano sobre las previsiones meteorológicas de la semana, indicándole en todo momento el estado actual del tiempo y aconsejándole para preveerlo.

Este proyecto se denominará "CONTESTADOR METEOROLOGICO". Pasemos a estudiar las funciones del mismo.

El Contestador meteorológico tiene como misión el de informar a los ciudadanos del estado de el tiempo durante todos los días de una semana. Para ello deberá tener registrado cada uno de los días y su previsión meteorológica que, previamente habrá sido programada por nosotros. Otros de los datos que suministrará este contestador será el de la ropa o elementos auxiliares de uso en consonancia con el tiempo que hay que registrar.

Analicemos los elementos necesarios para proceder a la fabricación del contestador.

Imaginemos que fuéramos nosotros la persona encargada de dar la información sobre el tiempo. Primero llegaría la persona interesada en recibir la información. Esta persona nos indicaría su nombre o el número de usuario que le corresponde. A continuación nos preguntaría por el día de la semana que quiere consultar. Nuestra siguiente acción sería buscar el día de la semana y sacar la información correspondiente al mismo. Una vez encontrado, le indicaríamos a nuestro cliente la previsión meteorológica para el mismo. Los elementos que necesitaríamos serían:

- A Un receptor de usuarios para información meteorológica.
- B Un receptor de día para indicar información.
- C Buscador de días e información referente a cada uno de ellos.
- D Controlador de búsqueda, para saber si el día está encontrado.
- E Emisor de información al usuario.

Comencemos a realizar la planificación del proyecto "Contestador Meteorológico", mediante la realización en una primera fase de fabricación del organigrama de flujo de los distintos elementos que lo componen.

El organigrama nos indicará el estudio de "ensamblaje" de los componentes que permitirán el funcionamiento del contestador.

Primer símbolo. Símbolo de comienzo. No necesita ningún comentario.

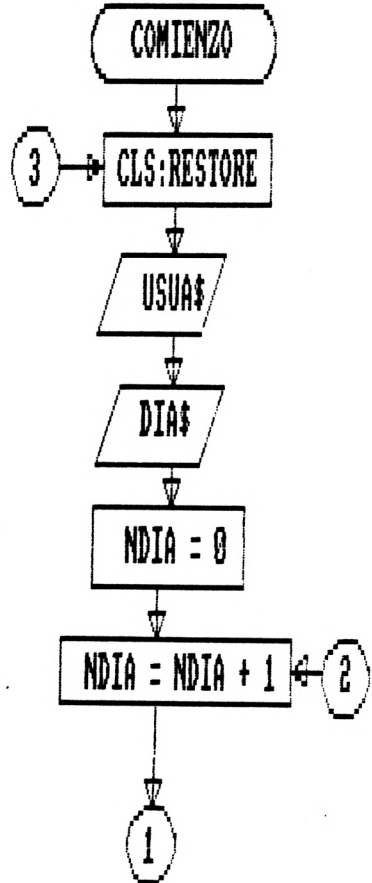
Segundo símbolo. Rectángulo. Se hace un borrado de la pantalla.

Tercer símbolo. Romboide. Entrada de nombre o código del usuario. Se utiliza la variable llamada USUA\$.

Cuarto símbolo. Romboide. Entrada de día de la semana a consultar. DIA\$ es la variable utilizada en este caso.

Aquí entramos en la fase búsqueda de días. Procedemos a la creación de un bucle y comenzamos con el control de final del mismo.

Quinto símbolo. Rectángulo. Se crea un contador para los días llamado NDIA utilizado para contarlos.





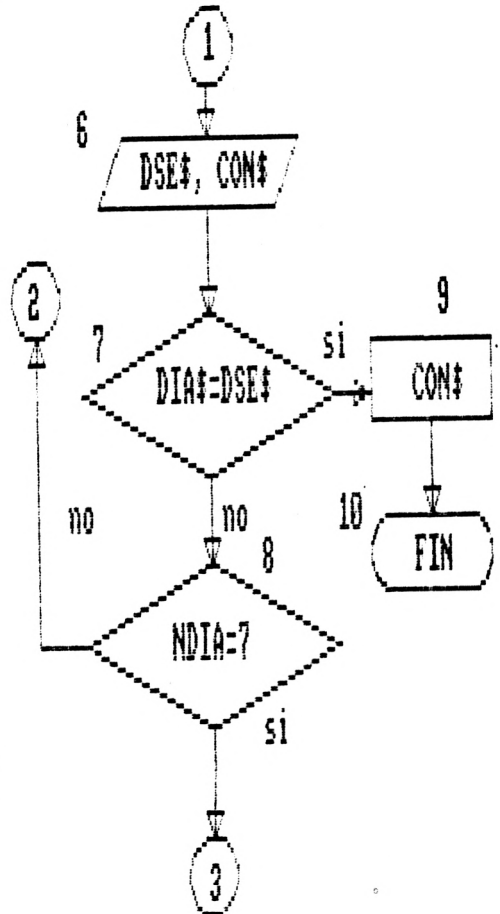
Sexto símbolo. Rectángulo. Realizamos la lectura de los días de la semana y su consulta mediante la utilización de la instrucción READ y las variables DSE\$, CON\$.

Séptimo símbolo. Romboide. Mediante la comparación de las variables DIA\$ y DSE\$ comprobamos la existencia del día señalado. En el caso de no coincidir pasamos al símbolo siguiente.

Octavo símbolo. Romboide. Siempre que el contador de día sea menor que 7, podemos seguir la búsqueda volviendo al símbolo quinto.

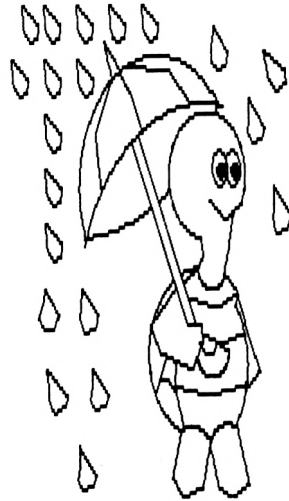
En caso contrario el día no habrá sido encontrado por lo que pediremos otro día.

Noveno símbolo. Rectángulo. Se procede a la presentación del día y el campo que le corresponde.



Hemos acabado de realizar el organigrama del proyecto "Contestador Meteorológico". Observa varias cosas:

1 El contador de días, comienza en el símbolo quinto, aumentando de uno en uno la variable NDIA para contar hasta siete



2 Dentro del bucle se produce la comparación entre las variables DS\$ y DD\$, para saber si coincide con el día introducido .

3 El símbolo séptimo es el control de final del bucle. En él se comprueba si NDIA ha llegado al valor de 7, mandando el control del programa de nuevo al símbolo quinto para continuar la búsqueda.

4 Cuando el bucle de búsqueda ha terminado sin encontrar el día, es decir NDIA es igual a 7, se han agotado las posibilidades de encontrarlo. Siendo así, volvemos al principio.



## Practicar

## 1 Codificación del experimento nro. 4.

Pasamos a continuación a la fase de fabricación de nuestra sofisticada máquina de previsión meteorológica.

Todos nuestros planos (organigramas), apuntes y elementos necesarios para proceder a la práctica de este proyecto deberán de estar sobre nuestra mesa de trabajo.

Enciende tu ordenador y comienza escribiendo:

```
10 REM PROYECTO CONTESTADOR METEORO-  
LOGICO
```

Recuerda:

No olvides pulsar la tecla (ENTER), después de introducir una línea de programa.

La instrucción REM, sólo sirve para poner letreros que nos orienten dentro del programa, y no producen ninguna acción.

```
20 CLS
```

```
30 INPUT "USUARIO : ";USUA$
```

```
40 INPUT "DIA SEMANA : ";DIA$
```

Observa:

Las instrucciones 20, 30 y 40 corresponden a los símbolos segundo, tercero y cuarto, respectivamente.

Comenzaremos a continuación a crear las instrucciones correspondientes al bucle de búsqueda.

¿ De qué forma más sencilla podemos indicar un bucle en un programa ?

Recuerdas la instrucción FOR...NEXT. Aquí tienes una oportunidad de utilizar esta instrucción. Escribe:

```
50 REM BUCLE DE BUSQUEDAS
```

```
60 FOR NDA=1 TO 7
```

Observa:

La variable que utilizábamos para realizar la búsqueda la llamamos NDIA (Número de día). Como son 7 los días en los que vamos a buscar (cada uno de los que consta una semana), escribimos NDA=1 TO 7.

A partir de aquí se introducen las intrucciones que se repetirán y se encuentran dentro del bucle, como son:

```
70 READ DSE$,CON$
```

```
80 IF DIA$=DSE$ THEN GOTO 110
```

```
90 NEXT NDA
```

Recuerda:

La instrucción READ, va acompañada siempre de las DATAS, después procederemos a crearlas. En DSE\$, se carga el valor de la primera DATA, así como en CON\$ el valor del consejo que la acompaña.

Cada vez que el control del programa pasa por esta instrucción es leída una nueva pareja de datos,(DSE\$ y CON\$) de las DATAS que más adelante se crearán.

La instrucción NEXT NDIA, cierra el bucle, mandando siempre el control a la siguiente instrucción después de FOR...TO.

Hasta aquí nuestro programa tiene las siguientes instrucciones introducidas:

```
10 REM PROYECTO CONTESTADOR
    METEOROLOGICO
20 CLS
30 INPUT "USUARIO : ";USUA$
40 INPUT "DIA SEMANA : ";DIA$
50 REM BUCLE DE BUSQUEDAS
60 FOR NDA=1 TO 7
70 READ DSE$,CON$
80 IF DIA$=DSE$ THEN GOTO 110
90 NEXT NDA
```

Continúa ahora escribiendo:

```
100 GOTO 10
```

Recuerda:

En el caso de agotarse la búsqueda en el bucle, el programa continuaría en la siguiente instrucción después de NEXT NDIA. En este caso el día no ha sido encontrado, luego tal y como indica nuestro organigrama volveríamos al principio del programa.

```
110 PRINT "EL ";DIA$;" HARA ";CON$
```

```
120 END
```

Los símbolos séptimo y décimo quedan registrados de esta manera por medio de estas dos líneas.

Observa:

Tal y como indicamos en la línea 80 de nuestro programa, al coincidir las variables alfanuméricas DIA\$ y DSE\$ dábamos por encontrado el día de la semana a consultar por nuestro usuario y salíamos del bucle para indicar la información recogida en la variable CON\$. La línea 110 es la encargada de visualizar toda esta información.

Una vez indicada la información la línea 120 con la instrucción END indicará el final de la consulta.

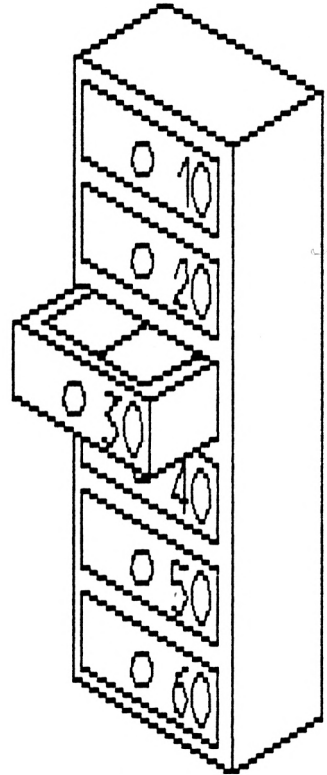
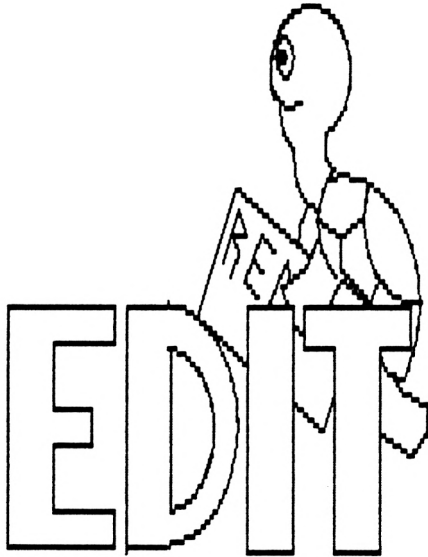
En este momento sólo nos queda por crear los datos con la información referente a cada uno de los días de la semana. Para ello y como recordarás utilizamos las instrucciones DATA.

Cada instrucción DATA va ligada a una instrucción READ. Puesto que pueden producirse siete lecturas con dos variables cada una de ellas (DSE\$ y CON\$), pasamos a crear 7 instrucciones DATA con dos campos separados por una coma: Uno para día (DSE\$) y otro para consejo (CON\$)

```
130 DATA "LUNES"," BUEN TIEMPO: IR CON CAMI-  
SETA"  
  
140 DATA "MARTES","EMPEORAMIENTO: CHAQUETA"  
  
150 DATA "MIERCOLES","NUBLADO: USAR CHUBAS-  
QUERO"  
  
160 DATA "JUEVES","FRESCO: IR ABRIGADO"  
  
170 DATA "VIERNES","BAJA TEMPERATURA: ABRI-  
GO"  
  
180 DATA "SABADO","LLUVIA: COGER PARAGUAS"  
  
190 DATA "DOMINGO","NIEVE: USAR BOTAS"
```

Con esta serie de instrucciones DATA queda completo nuestro proyecto. Realiza un listado utilizando el comando LIST y verifica que todas las "piezas" de tu máquina están bien colocadas, antes de ponerla en funcionamiento.

## 2 El comando EDIT.



Como su nombre indica, el comando EDIT, sirve para editar, es decir extrae la línea de programa que nosotros por cualquier causa determinada queremos modificar o corregir.

La forma de utilizar este comando es bien sencilla. Solamente es necesario indicar después de la palabra EDIT, la línea de programa a la que queremos acceder.



Veamos un ejemplo:

EDIT 50 (ENTER)

Al instante aparecerá en pantalla la línea 50 como sigue:

50 REM BUCLE DE BUSQUEDAS

En este momento podemos hacer cualquier modificación a la instrucción contenida en esta línea, sin necesidad de tener que teclearla completa.



## Preguntar

1

Explica en breves palabras la utilidad de la instrucción READ/DATA.

2

¿Cuál es la función de la instrucción REM?.

3

¿Cuándo nos encontramos con la siguiente línea en un programa, cuál es su cometido?.

```
10 REM INPUT ENTRADA EN$
```

4

Realiza un programa que cargue por medio de la utilización de la instrucción READ, los meses del año.

5

¿Está bien escrita la expresión:

```
10 READ UNO$,DOS$,TRES$
20 DATA 1,2,3
```

6

Escribe un programa que utilice un bucle que cargue 5 datos distintos en una instrucción READ/DATA.

## UNIDAD 9 TABLAS DE UNA DIMENSION



Conocer

1 ¿Qué es una tabla?.

Hasta ahora hemos estado trabajando con variables. Cada vez que necesitábamos crear algún elemento necesario para realizar alguno de nuestros experimentos, nos inventábamos el nombre de las variables que íbamos a utilizar.

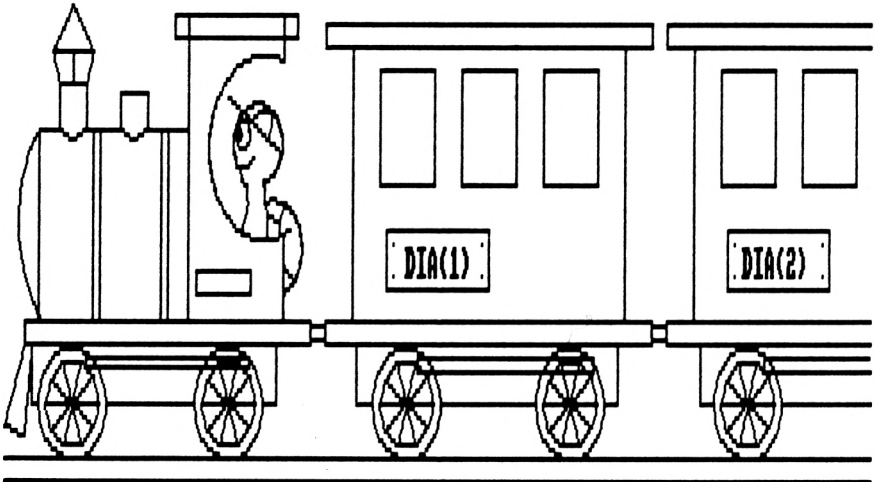
Existen programas muy largos y que tienen que utilizar un gran número de variables. Para realizar uno de estos programas tendríamos que estar inventándonos continuamente nombres para las distintas variables.

Imagina un programa en el que tuviésemos que realizar la predicción meteorológica de todo un año, y que cada uno de los días del año tuviese que estar contenido en una variable distinta. Podríamos inventarnos las variables llamadas: DIA1, DIA2, DIA3,... DIA 365. Como puedes apreciar utilizaríamos un número al final de la palabra DIA, de esta forma podríamos hacer distintas unas de otras sin mucho trabajo.

Una tabla, es una lista de variables que guardan una relación entre sí, y llevan asociado un número de índice. Al igual que la variable DIA1, lleva asociado al nombre el número uno, una tabla de variables se puede representar mediante un nombre y un número entre paréntesis, como por ejemplo: DIA(1). Con sólo cambiar el número que se encuentra dentro de el paréntesis, podemos tener variables distintas: DIA(1) es distinta a DIA(2), etc...

## 2 Dimensionamiento de tablas .

Cada vez que nos proponemos utilizar una variable a la que vaya a ir asociada un índice, tendremos que darle a nuestro ordenador la orden para la reserva en su memoria de la cantidad de elementos que ésta vaya a tener.



Así por ejemplo, si para los días del año, preveemos 365 días, la variable día contendrá la información desde el elemento DIA(1) hasta el elemento DIA(365). El ordenador, ha de estar informando y para ello utilizaremos la instrucción DIM.

La forma de representar esta instrucción es bien sencilla: DIM Nombre de la variable (nro. máximo de elementos). Así por ejemplo para nuestra variable DIA, indicaremos su dimensión de la siguiente forma: DIM DIA (365). De esta manera el ordenador reservará el espacio para los 365 datos que podrán ser introducidos dentro de cada uno de los elementos que componen la variable DIA(X).

### 3 El Comando AUTO.

Cuando realizamos la codificación de alguno de nuestros experimentos, nos es necesario ir tecleando cada una de las líneas de las que consta nuestro programa.

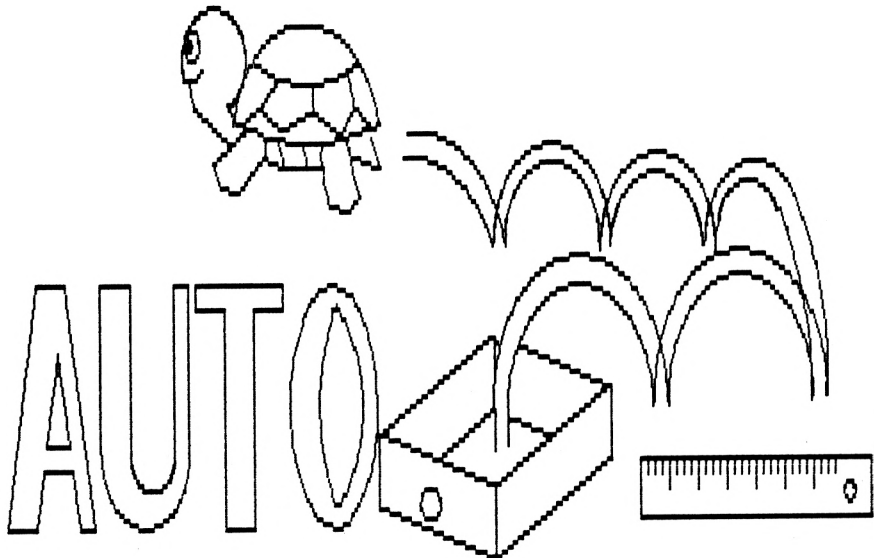
Cada vez que nosotros introducimos una nueva línea, iremos recordando el número de la anterior para continuar digitando la siguiente.

El comando AUTO, nos permite de una manera "automática" ir indicándonos en el momento de introducir un programa el número de la línea siguiente a codificar.

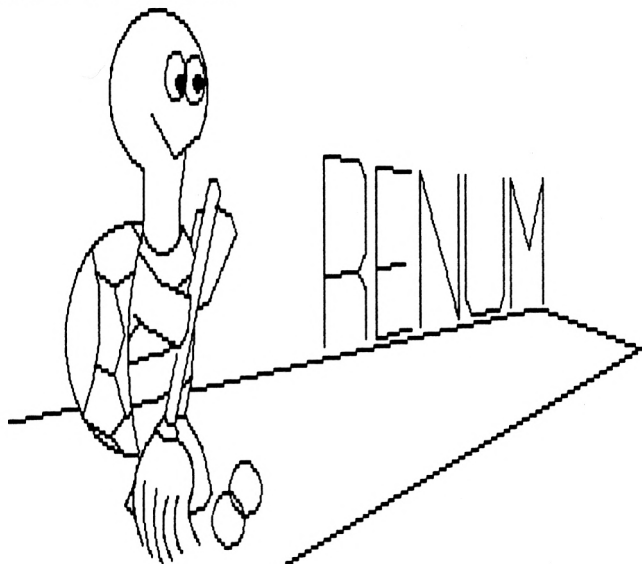
El formato de este comando es el siguiente:

AUTO, nro. primera línea, salto

Siendo nro. de la primera línea, la línea desde la que queremos partir para codificar y salto el nro. de líneas intermedias.



## 4 La instrucción RENUM.



¿Has pensado alguna vez en poner un poco de orden en los programas que has ido codificando?. En muchas ocasiones tus programas han ido aumentando en el número de líneas, a la vez que se han introducido otras nuevas entre la ya creadas. La numeración que tú en un principio habías pensado llevar, por ejemplo de diez en diez líneas no ha sido posible cumplirla.

La instrucción RENUM, sirve para poner un poco de orden en los programas ya codificados. Te permite "renumerar" las líneas de tus programas. Su formato es el siguiente:

RENUM, nro.nva.línea, anterior, incremento

Nro. nva. línea es la línea por la que queremos comenzar a reenumerar nuestro programa. Anterior se refiere a la antigua línea ya existente. El incremento marca las líneas de separación entre cada una de las reenumeradas.

Renumerar cualquiera de tus programas una vez acabados. Advertirás con que rapidez tu ordenador ordena todo tu programa y como los saltos y las referencias de las líneas reenumeradas son cambiadas para hacerlas coincidir con la nueva numeración.

#### 5 Experimento nro. 5. Proyecto ordenador.

Generalmente los ordenadores son llamados así por el "orden" que llevan el tratamiento de la información. Nuestro director de laboratorio tiene conocimiento de este tema y conoce la utilidad que venimos dando a nuestra máquina. Esta mañana ha estado contemplando al ordenador y nos ha comentado :

¿Este aparato funciona bien?

Nosotros, conocedores de el manejo y utilidad del mismo contestamos:

!Si, señor!.

Tras una conversación sobre la utilidad de nuestro ordenador, nuestro director nos ha pedido que demos si verdaderamente sirve para "ordenar". Nos ha encargado



que metamos en su memoria todos los utensilios que encontremos a nuestro paso y que le demos la orden de que los ordene alfabéticamente. Si verdaderamente el aparato es "ordenado", su utilidad quedará demostrada.

En nuestro laboratorio podemos encontrar una gran cantidad de cosas. Algunas se encuentran sobre la mesa, otras sin embargo se hayan repartidas indistintamente por los diferentes rincones de la habitación que ocupamos.

El experimento que vamos a realizar es un trabajo, y a éste le denominaremos "PROYECTO ORDENACION". Lo primero que haremos antes de poner manos a la obra será la de planificar las tareas que iremos desarrollando.

Lo primero que haremos será dejar muy claro lo que queremos hacer, es decir: coger los nombres de las cosas que tenemos en nuestra habitación y por medio de nuestro ordenador "ordenarlas" por orden alfabético.

Pongamonos por un momento a pensar la forma en que nosotros realizaríamos la ordenación alfabética de las cosas contenidas en nuestra habitación:

1 Cogeríamos un papel y un lápiz e iríamos apuntando todas las cosas en una lista.

2 Al terminar de apuntar todas, una forma de ordenar las cosas apuntadas sería leer el nombre de la primera cosa y compararlo con la siguiente.

3 En el caso de que el primer nombre fuese mayor en el orden alfabético lo cambiaríamos de lugar con el primero, es decir el primero lo pondríamos el segundo y el segundo el primero.

4 Compararíamos el segundo con el tercero, en el caso de ser mayor, repetiríamos la acción del 3, pero en este caso cambiándolo al lugar tercero, y el tercero al segundo.

5 Continuaríamos así sucesivamente hasta llegar al final de la lista, en cuyo caso volveríamos a comparar otra vez desde el principio

6 La última de las cosas a realizar sería el comprobar que en un momento dado, no hemos cambiado ningún nombre de cosa con el siguiente. Esto quiere decir que todos los nombres se encuentran ya ordenados.

Una vez realizada nuestra ordenación mental, procederemos a pasarla en forma de organigrama que más adelante podamos utilizar para codificar el programa que haga posible el "PROYECTO ORDENACION".

1 Primer símbolo.  
Símbolo de comienzo.

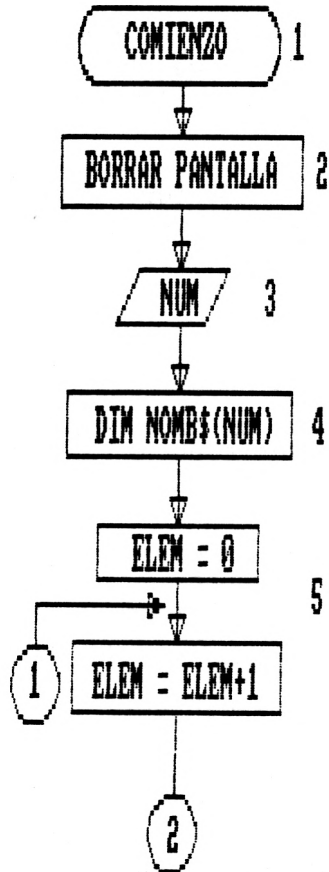
2 Segundo símbolo.  
Rectángulo. Borrado de pantalla.

3 Tercer símbolo.  
Romboide. Entrada del número de nombres de cosas a ordenar. Utilizamos como variable NUM.

4 Cuarto símbolo.  
Rectángulo. Lo utilizamos para incluir una instrucción DIM para dimensionar la tabla con NUM como número máximo de elementos a contener.

Comenzamos un bucle para aceptar los NUM nombres de cosas, para ir introduciéndolos en la tabla.

5 Quinto símbolo.  
Rectángulo. Se crea un contador para los elementos utilizando la variable numérica ELEM.

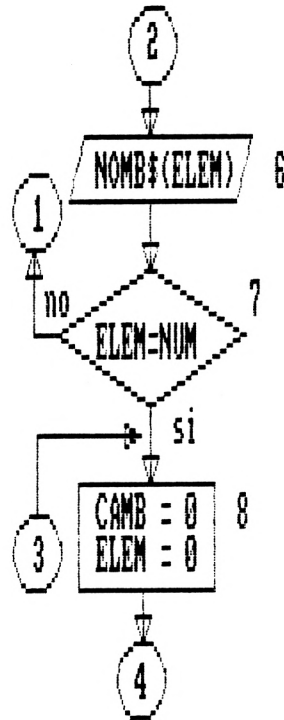


6 Sexto símbolo. Romboide. Entrada de datos. Se procede mediante la variable alfanumérica NOMB\$(ELEM), a aceptar el nombre de las cosas dentro del bucle.

7 Séptimo símbolo. Rombo. Control de final de bucle. Se compara la variable ELEM con NUM para comprobar si ésta es igual. En el caso de no serlo y por no haber llegado al final se devuelve el control al quinto símbolo (5).

A partir de este momento comenzamos la ordenación de la lista de nombres, de igual forma que actualizáramos en el caso de hacerlo manualmente.

8 Octavo Símbolo. Rectángulo. Lo utilizamos para dos variables a cero, una ELEM y otra llamada CAMB.

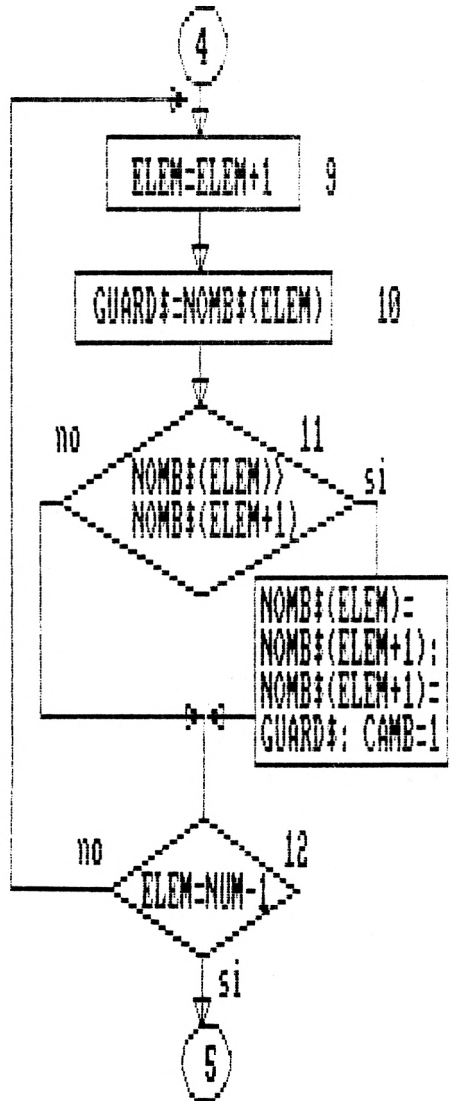


9 Noveno símbolo. Rectángulo. Creamos el contador para el bucle de ordenación. Utilizamos la variable ELEM de nuevo para ir aumentándola de uno en uno.

10 Décimo símbolo. Rectángulo. Guardamos en la variable GUAR\$ el valor alfanumérico de el elemento que corresponde según el contador de ELEM.

11 Onceavo símbolo. Rombo. Se procede a la comparación de las variables NOMB\$(ELEM) y NOMB\$(ELEM+1), en el caso de ser mayor NOMB\$(ELEM), se realiza el cambio entre un lugar y otro y utilizamos la variable GUAR\$ para no perder el valor que tenía NOMB\$(ELEM). Como ha habido un cambio, se hace igual a uno la variable CAMB.

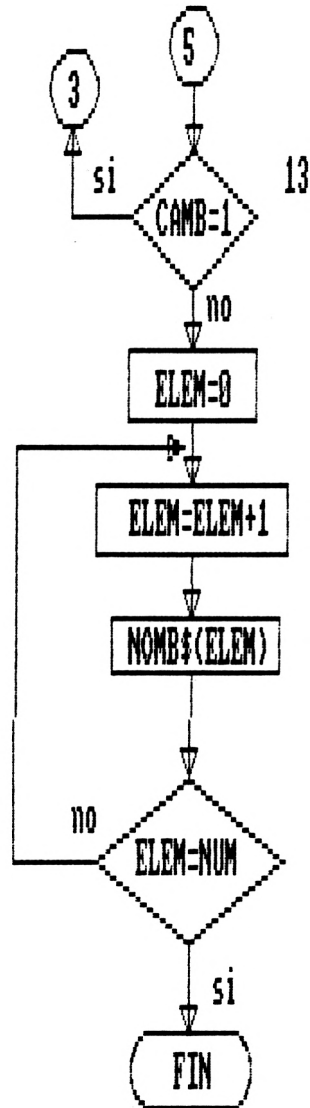
12 Doceavo símbolo. Rombo. Controla el final de el bucle de ordenación.



13 Treceavo símbolo. Rombo. Pregunta si el valor de la variable CAMB es igual a uno. En ese caso es que se ha producido algún cambio dentro del bucle de ordenación. Mandamos el control a empezar a dar otra búsqueda al comienzo del mismo. Ir al símbolo noveno (9) en el caso de CAMB=1.

Cuando la variable CAMB, es igual a cero, es porque no se a producido ningún cambio dentro del bucle, luego todos los nombres están ya en su orden. En este momento comenzamos un nuevo bucle para presentar en la pantalla todos los elementos de la tabla ordenados.

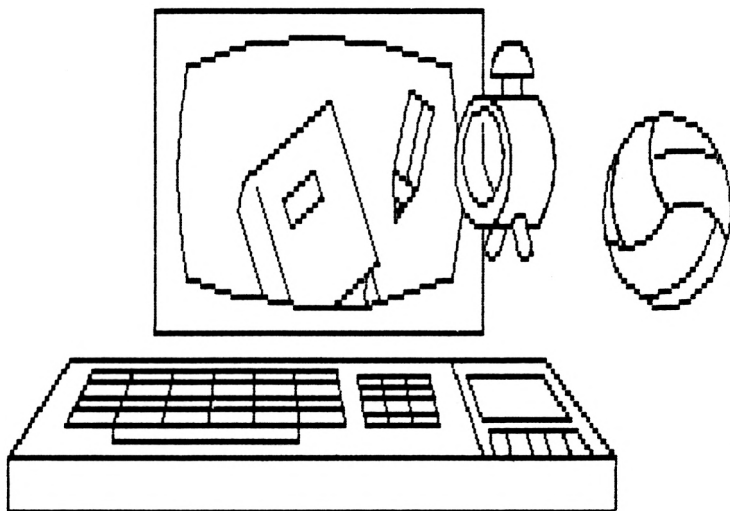
Antes de continuar repasa con detalle las acciones que se desarrollaron durante la ordenación.





## Practicar

## 1 Codificación del experimento nro. 5



El ordenador nos ha ayudado durante todos los experimentos realizados hasta ahora. Se han hecho de las más variadas pruebas: demostraciones de inteligencia, pruebas espaciales, fabricación de naves, etc...

El experimento que nos ocupa en la presente unidad es fácil y difícil a la vez. Demuestra que nuestro ordenador sin los datos y conocimientos que nosotros le aportamos no es capaz de trabajar por sí solo.

Generalmente los ordenadores realizan tareas como la que estamos analizando. Unas veces son las listas de los nombres de las personas de una ciudad, otras veces ordenan números correspondientes a listas de cosas para facilitar su búsqueda. Nosotros vamos a crear nuestro primer programa de ordenación y comprobaremos la gran rapidez con la que el ordenador trabaja para realizar este tipo de tareas.

Comencemos la codificación de nuestro "PROYECTO ORDENACION". Enciende el ordenador y escribe:

```
10 REM PROYECTO ORDENACION
```

```
20 CLS
```

```
30 INPUT "Número de cosas a ordenar ";NUM
```

Observa:

Comenzamos escribiendo una instrucción REM que indica el cometido del programa que viene a continuación.

El primer rectángulo y el romboide quedan codificados en las líneas 20 y 30 respectivamente.

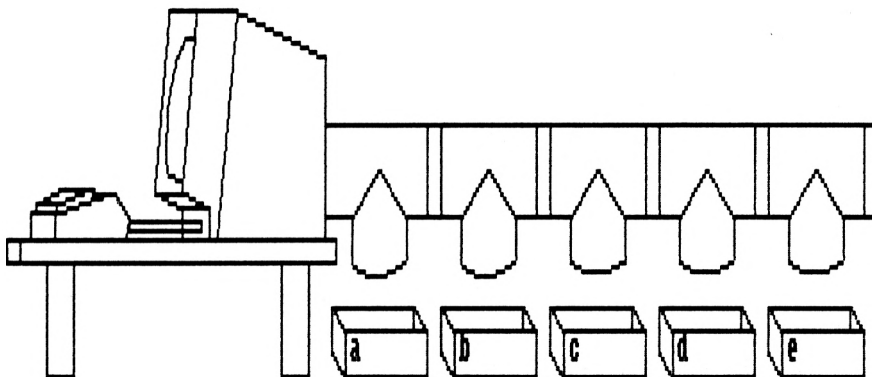
Continúa con:

```
40 DIM NOMB$(NUM)
```



Recuerda:

Tal y como te explicamos en el punto 2 de CONOCER, la instrucción DIM sirve para dimensionar una tabla. Aquí se quiere indicar que se van a utilizar el valor de NUM variables distintas, que es el número de cosas a ordenar indicados en la línea 30. Imagina que diéramos el valor 12, DIM permitiría reservar del espacio en la memoria para 12 variables alfanuméricas, pues DIM NOMB\$(NUM) sería lo mismo que escribir DIM NOMB\$(12). De esta manera no haría falta inventarnos 12 nombres de variables distintas a lo largo del programa, con variar el índice correspondiente a NOMB\$(-) nosotros podemos llamar al contenido de una u otra variable, también como veremos a continuación dar uno u otro valor a las mismas.



Las líneas siguientes de nuestro programa, van a permitirnos la introducción de los nombres de las cosas a ordenar de una forma muy sencilla. Escribe:

```
50 FOR ELEM=1 TO NUM
60 INPUT "Nombre : ";NOMB$(ELEM)
70 NEXT ELEM
```

Observa:

1 Las líneas 50, 60 y 70, corresponden al bucle que tenemos representado en nuestro organigrama con los símbolos 5 (principio del bucle), 6 (entrada de datos), y 7 (control de final). Como recordarás la instrucción FOR ... TO... NEXT, permite la repetición de una acción tantas veces como indica el valor de la variable representada detrás del FOR (en este caso ELEM=1), hasta el valor de la variable representada detrás del TO (NUM).

2 La repetición que produce este bucle, se ve inscrita dentro de él, se inscribe desde el FOR hasta la anotación NEXT. Las instrucciones en la línea 60 son las que se repetirán desde el valor de ELEM igual a 1 hasta que alcancen el valor contenido en la variable NUM.

3 Observa también que el valor de ELEM, irá tomando a lo largo de la repetición de las instrucciones que se encuentran dentro del bucle los valores desde 1 hasta NUM, esto quiere decir que la primera vez, la instrucción INPUT "Nombre : ";NOMB\$(ELEM), aceptará un dato para NOMB\$(1), puesto que ELEM será igual a 1, permitiendo introducir en NOMB\$(2) otro dato distinto y así

sucesivamente hasta llegar a el valor de NUM que marcará el final del último valor para ELEM.

Continuamos codificando nuestro programa. Entramos el bucle que nos permitirá realizar la ordenación:

```
80 CAMB=0
90 FOR ELEM=1 TO NUM
100 GUARD$=NOMB$(ELEM)
110 IF NOMB$(ELEM) > NOMB$(ELEM+1) THEN
NOMB$(ELEM)=NOMB$(ELEM+1):NOMB$(ELEM)=
GUARD$:CAMB=1
120 NEXT ELEM
```

Observa paso a paso:

1 En 80. CAMB, es una variable que lleva siempre al principio el valor cero, pero como veremos más adelante puede cambiar a 1.

2 En 90. GUARD\$, sirve para guardar el nombre recogido en la variable NOMB\$(ELEM), cada vez que ELEM va aumentando.

3 En 110. Como se explica en el símbolo 11 de el organigrama, se compara los valores de `NOMB$(LEM)` y el nombre siguiente que se encuentra en el índice `ELEM+1` de la variable `NOMB$(-)`.

4 Cuando la variable `NOMB$(ELEM)` es mayor que la variable `NOMB$(ELEM+1)`, se produce el cambio entre estas dos. Imagina una lista donde los dos primeros nombres son:

```
1 PERRO
2 CASA
```

Si `NOMB$(1)="PERRO"`, y `NOMB$(2)="CASA"`, se cumple que `NOMB$(ELEM)` para `ELEM=1`, es mayor que `NOMB$(ELEM+1)`. Hay que cambiar el elemento 1 por el dos es decir dejar la lista así:

```
1 CASA
2 PERRO
```

¿Cómo hacerlo? Muy fácil. Primero indicando que en la variable `NOMB$(1)` se introduzca el dato de la variable `NOMB$(2)`: `NOMB$(ELEM)=NOMB$(ELEM+1)`. Es decir, la lista se queda:

```
1 CASA
2 CASA
```

Segundo, introduciendo el dato de la variable NOMB\$(2) en la variable NOMB\$(1), pero ATENCION ya no lo contiene puesto que 1 y 2 tienen el mismo valor. Como hacerlo. Muy sencillo recuerda que en la variable GUARD\$ se "guardó" el valor de NOMB\$(1) por si acaso se necesitaba el dato de NOMB\$(ELEM). Con hacer NOMB\$(ELEM)=GUARD\$, NOMB\$(1) queda con el valor primero antes de hacer el cambio, es decir, con "PERRO". Así pues la lista quedaría como nos propusimos:

```
1 CASA
2 PERRO
```

5 En la misma instrucción se incluye el dar el valor 1 a CAMB. Es para señalar que se ha producido un cambio, es decir, que en la lista existía algún nombre desordenado y se ha tenido que hacer el cambio de datos de una variable a otra.

Continuamos codificando nuestro programa:

```
130 IF CAMB=1 THEN GOTO 80
```

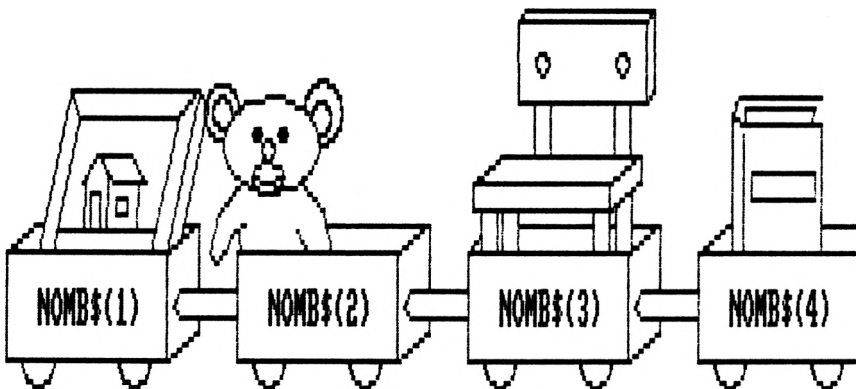
Observa que siempre que se ha producido un cambio dentro de nuestra lista de nombres volvemos a mandar el control de el programa a realizar una nueva comprobación para buscar algún otro nombre desordenado.

En el caso de que  $CAMB=0$ , nos indica que no ha sido realizado ningún cambio. Esto es porque todos los nombres están ordenados y no hay ninguno que sea menor y se encuentre detrás de otro mayor alfabéticamente.

Sólo nos queda sacar la lista ordenada por la pantalla. Para ello volvemos a utilizar un bucle escribiendo:

```
140 FOR ELEM=1 TO NUM
150 PRINT NOMB$(ELEM)
160 NEXT ELEM
```

Verifica la escritura de tu programa y realiza un listado completo de él. Modifica con el comando EDIT, todas aquellas instrucciones que estén mal introducidas.



Este es el listado completo de tu programa:

```
10 REM PROYECTO ORDENACION
20 CLS
30 INPUT "Número de cosas a ordenar ";NUM
40 DIM NOMB$(NUM)
50 FOR ELEM=1 TO NUM
60 INPUT "Nombre : ";NOMB$(ELEM)
70 NEXT ELEM
80 CAMB=0
90 FOR ELEM=1 TO NUM-1
100 GUARD$=NOMB$(ELEM)
110 IF NOMB$(ELEM) > NOMB$(ELEM+1) THEN
NOMB$(ELEM)=NOMB$(ELEM+1):NOMB$(ELEM+1)=
GUARD$:CAMB=1
120 NEXT ELEM
130 IF CAMB=1 THEN 80
```

```
140 FOR ELEM=1 TO NUM
150 PRINT NOMB$(ELEM)
160 NEXT ELEM
170 END
```

Realiza una prueba con la instrucción RENUM para cambiar el número de las líneas de instrucción de tu programa.





## Preguntar

1

Explica con breves palabras lo que es una tabla de datos, y pon un ejemplo.

2

¿Para qué se utiliza la instrucción DIM?

3

Inventa un problema en el que sea necesario utilizar una variable con subíndices.

4

Haz el organigrama de un programa que lea dentro de un bucle valores a una variable con subíndice.

5

Modifica el experimento nro. 5 para ordenar por orden alfabético inverso todos los datos introducidos.

6

Renumerar tu programa de uno en uno a partir de la línea 100.









# AMSTRAD



# Order your Aprisndo

Aprisndo is a powerful tool for managing your business. It offers a wide range of features and options to help you grow and succeed.

With Aprisndo, you can track your sales, manage your inventory, and analyze your performance. It's the only tool you need to run your business efficiently.

Order your Aprisndo today and see the difference it can make for your business. Contact us now to learn more.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

Call us at 1-800-APRISNDO for more information and to place your order.

Order your Aprisndo today and see the difference it can make for your business.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

Call us at 1-800-APRISNDO for more information and to place your order.

Order your Aprisndo today and see the difference it can make for your business.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

Call us at 1-800-APRISNDO for more information and to place your order.

Order your Aprisndo today and see the difference it can make for your business.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

Call us at 1-800-APRISNDO for more information and to place your order.



APRISNDO

Order your Aprisndo today and see the difference it can make for your business.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

Call us at 1-800-APRISNDO for more information and to place your order.

Order your Aprisndo today and see the difference it can make for your business.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

APRISNDO

Order your Aprisndo today and see the difference it can make for your business.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

Call us at 1-800-APRISNDO for more information and to place your order.

Order your Aprisndo today and see the difference it can make for your business.

Visit [www.aprisndo.com](http://www.aprisndo.com) to place your order and explore all the features and options available.

Don't miss out on this opportunity. Order your Aprisndo today and take your business to the next level.

Call us at 1-800-APRISNDO for more information and to place your order.

# AMSTRAD

# CPC



**MÉMOIRE ÉCRITE**  
**MEMORY ENGRAVED**  
**MEMORIA ESCRITA**



<https://acpc.me/>

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, et non commerciales.

[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.

[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.