BASIC

CURSO DE PROGRAMACION CON APLICACIONES DIDACTICAS.

Manuel Granados
Tomás Díez

Incluye apéndice de referencia de BASIC, IBM-PC y COMPATIBLES.

ra-ma

BASIC

CURSO DE PROGRAMACION CON APLICACIONES DIDACTICAS

MANUEL GRANADOS

Y

TOMAS DIEZ

BASIC. Curso de Programación con Aplicaciones Didácticas.

Editado por:
RA-MA Editorial
Ctra. de Canillas, 144
28043 MADRID. Telfs. (91) 200 97 46/47

I.S.B.N.: 84-86381-34-7
Depósito legal: M. 34.007-1987
Impreso en España.

Hoy en día y en casi todos los países, no se concibe que los ordenadores no estén introducidos en cualquier área de la actividad social y económica, ya en el mundo de la producción, de la cultura o del arte. Baste recordar que el «Financial Times» eligió como «hombre del año 82» al ordenador personal.

Al margen de la paradoja, la decisión supuso la aceptación por la cual la informática había salido del mundo científico y se había hecho popular y cotidiana.

Se puede decir que se ha iniciado una nueva revolución, comparable al nacimiento de la escritura o al de la imprenta. Por otra parte, la informática es pilar de base de la inteligencia artificial y la robótica, que producirán grandes modificaciones en nuestra vida. No es exagerado este pronóstico, pues en los últimos cien años los avances científicos y tecnológicos han crecido de forma exponencial.

Esperemos todos y deseemos que estos avances sirvan para conseguir, además de una vida cómoda, un mundo mejor, más justo y feliz.

El presente libro te puede ayudar a encarar el reto informático en el que estamos todos. Superemos el eslogan «o programas, o te programan». Para ello te proponemos una introducción al lenguaje más extendido, el BASIC que puedes utilizar en cualquier ordenador personal.

Cada tema consta de una exposición teórica con programas comentados. Al final hay una colección de programas propuestos, de dificultad creciente, a fin de que domines este lenguaje.

LOS AUTORES

INDICE

1. La informatización: su tratamiento e historia 9
2. El BASIC 19
3. Primeras operaciones y visualización 25
4. Diagramas de control 35
5. Transferencias de control 39
6. Iteracción mediante bucles 47
7. Funciones numéricas 59
8. Funciones con cadenas 69
9. Ficheros internos 77
10. Matrices 81
11. Subrutinas 97
12. Gráficos, color y sonido 103
13. El juego de caracteres ASCII 115
14. Grabación y carga de programas. Ficheros externos 131
15. Programas de aplicación didáctica 139

INTRODUCCION

El trabajo que se acompaña arranca de la iniciativa tomada en febrero del 83 por la cual nos propusimos dos profesores del Seminario de Física y Química alcanzar los conocimientos para el manejo operativo de microordenadores.

Tras una preparación intensiva (con asistencia a varios cursillos) y en buena medida autodidacta, impartimos el primer cursillo de programación BASIC para profesores interesados de nuestro Instituto en septiembre del 83, y comenzamos ese mismo curso a impartir clases de Informática para los alumnos como una asignatura optativa, tras la autorización del M.E.C. para 2.º BUP. Al año siguiente se extendería a 3.º BUP, y con el actual son ya cuatro cursos los que se imparten de esta materia.

MEMORIA DEL TRABAJO

El trabajo realizado ha sido la elaboración de un libro sacado de la experiencia de nuestras clases con los alumnos de Informática. Este libro trata del lenguaje de programación BASIC en un intento de aprovechamiento multidisciplinar, y consta de 15 capítulos (ver índice) que se caracterizan por:

— **en cuanto a su estructura:**

1. Sintaxis del BASIC y algunas rutinas en código máquina para procesador Z 80A.
2. Programas de carácter multidisciplinar a resolver por cada capítulo.
3. Un capítulo dedicado al tratamiento didáctico de Matemáticas, Física, Química, Biología e Inglés por ordenador.

— **en cuanto a su intencionalidad:**

1. Aprendizaje de un lenguaje de programación.
2. Aproximación al mundo de la Informática.
3. Resolución de supuestos que abarquen:
    - tratamiento de textos,
    - cálculo numérico,
    - dibujos y gráficos.

OBJETIVOS GENERALES

Frente al libro

1. Instrumento de consulta y ayuda de cara a la elaboración de programas.
2. Elaboración, dentro de un mismo contexto, de lenguaje BASIC, rutinas en código máquina y su aplicación a programas.

Frente al educando

1. Familiarización y uso por el alumno de esa formidable herramienta en que se ha convertido el ordenador.
2. Captación del interés del alumno por esta nueva tecnología, haciéndole ver:
   - sus limitaciones para destruir el mito,
   - sus posibilidades como usuario de esta tecnología.
3. Posibilitar caminos de análisis y estudios más completos y profundos sobre determinados temas.
4. Proposición de supuestos a resolver con fines específicos y de elección libre.
5. Reducir o suprimir en lo posible el tiempo dedicado a la rutina o a cálculos repetitivos.
6. Provocación de actitudes críticas del alumno frente al ordenador, impidiendo que sea un sujeto simplemente receptivo y pasivo.
7. Acentuación en el acercamiento entre alumno y profesor, creando un ambiente de concurrencia alrededor del fenómeno informático.

OBJETIVOS OPERATIVOS

Se ha distribuido el contenido de la obra en dos cursos. El primero, para 2.º BUP, abarca hasta el Cap. VIII mientras que para 3.º se realiza un repaso al comienzo del curso de lo aprendido en 2.º y posteriormente se desarrolla el resto del libro.

La enseñanza tiene como fin la realización de programas por parte de los alumnos, agrupados en equipo de 3.

Al final de los dos cursos el alumno ha adquirido la suficiente destreza en este lenguaje para realizar un trabajo «fin de curso», relativamente complejo y tutelado por el profesor.

RESULTADOS OBTENIDOS

Se ha observado que el alumno que posee algún microordenador en casa tiene una ventaja operativa respecto a sus compañeros.

Al margen de esta circunstancia, hemos de señalar que los alumnos han quedado capacitados para:

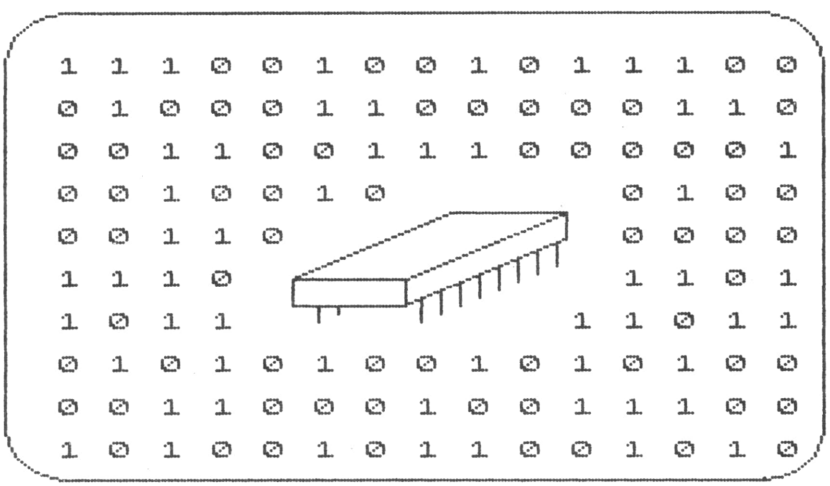
— abordar cualquier otro lenguaje de alto nivel, como Pascal,
— profundizar en un BASIC avanzado,
— conocer más a fondo el amplio mundo de la Informática y sus aplicaciones.

REALIZACION DE LA EXPERIENCIA

El desarrollo del trabajo constituye el libro que se acompaña, utilizado por nosotros desde el curso 84-85.

TEMA 1

LA INFORMACION: SU TRATAMIENTO E HISTORIA



ELEMENTO BASE DE UN ORDENADOR Y LA INFORMACION QUE PROCESA.

MEDIDA DE LA INFORMACION. EL BIT Y EL BYTE

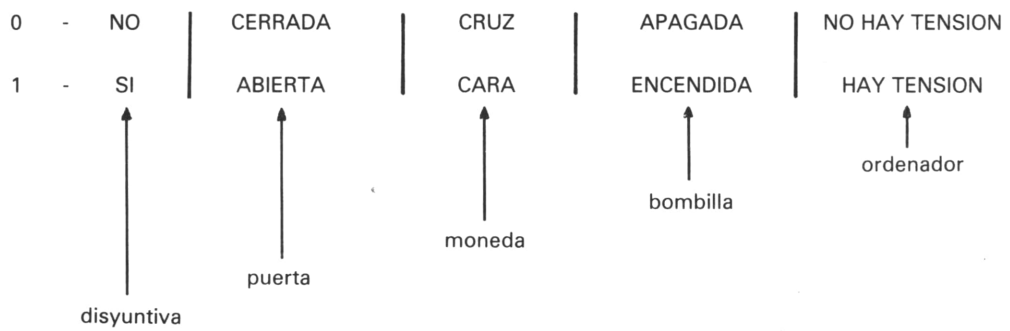
La información puede ser analógica o digital:

— Analógica: Si la información puede tomar un continuo de valores, como la que nos podría dar un termómetro o un disco al escuchar la música.

— Digital: Cuando la información es discreta, es decir, sólo toma ciertos valores, como la información que proporciona un dado o una ruleta.

Para clarificar la diferencia entre los dos conceptos, imagina una pelota en un determinado piso de un edificio. El piso en que se encuentra es información digital. La lanzamos por la ventana y las alturas por las que va pasando, que varían de forma continua, es información analógica.

La información digital binaria sería la que define solamente dos estados, que son mutuamente excluyentes y que definen todas las posibilidades. Se representarán numéricamente por el 0 y por el 1. Unos ejemplos podrían ser:



La medida matemática de la información es el logaritmo en base dos del inverso de la probabilidad del suceso

$$I = \log_2 \frac{1}{P(S)}$$

Así, la probabilidad de sacar cara o cruz en el lanzamiento de una moneda es 1/2, y, por tanto

$$I = \log_2 \frac{1}{1/2} = 1$$

A esta unidad se le da el nombre de bit (binary digit) y es la unidad mínima de información. Para un dado sería 2,585 bit, que es $\log_2 \frac{1}{1/6}$.

Con sólo tres bit, simbolizados con el 0 y el 1 podemos transmitir ocho informaciones, que corresponden a:

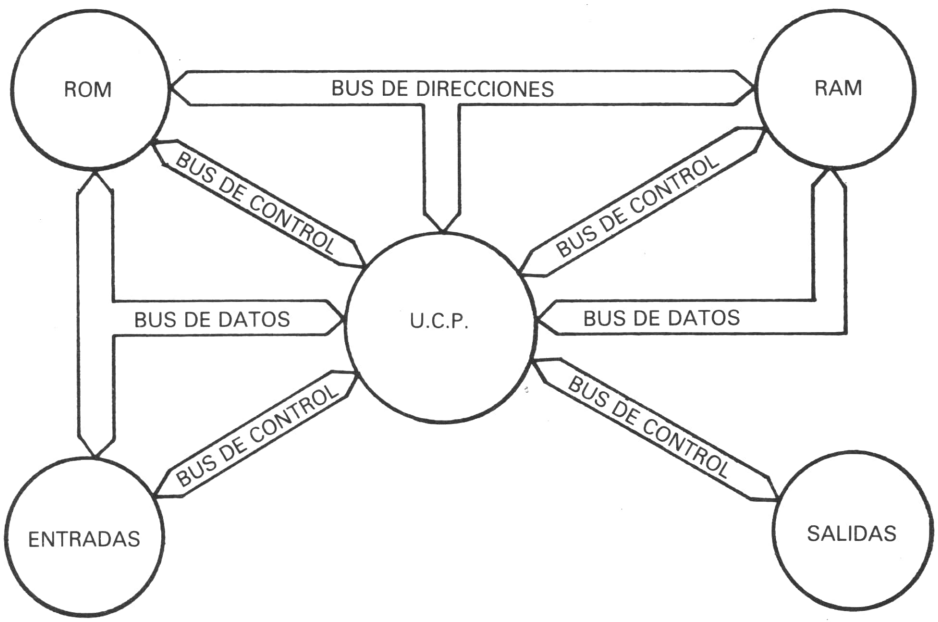
primera	000
segunda	001
tercera	010
cuarta	011
quinta	100
sexta	101
séptima	110
octava	111

El número total, son las variaciones con repetición de dos elementos (0 y 1), tomados de 3 en 3. La fórmula da 2^3. Por lo tanto, utilizando sólo tres bit se podrían transmitir ocho informaciones diferentes, letras o números. Como esto es claramente insuficiente, se utilizan agrupaciones de ocho bit, llamadas octetos o byte.

El número de posibilidades es entonces $2^8 = 256$, con las que podemos transmitir todo el abecedario, con mayúsculas, minúsculas, números del 0 al 9, símbolos especiales y caracteres gráficos, etc.

HARDWARE. ESTRUCTURA DE UN ORDENADOR

Esta palabra inglesa significa chatarra, y en informática hace mención al conjunto de elementos físicos del sistema: consta de microprocesador, memorias y periféricos. Estas partes están unidas físicamente entre sí mediante conductores llamados BUSES, que son los que transportan la información codificada en binario entre las distintas partes del sistema.



Vamos a ver cómo actúa cada parte por separado:

1. U.C.P. (UNIDAD CENTRAL DE PROCESO)

Se encarga de:
- recibir las instrucciones del programa y decodificarlas,
- llevar el control de registros,
- controlar las operaciones aritméticas y lógicas,
- llevar secuencialmente los diversos pasos del programa.

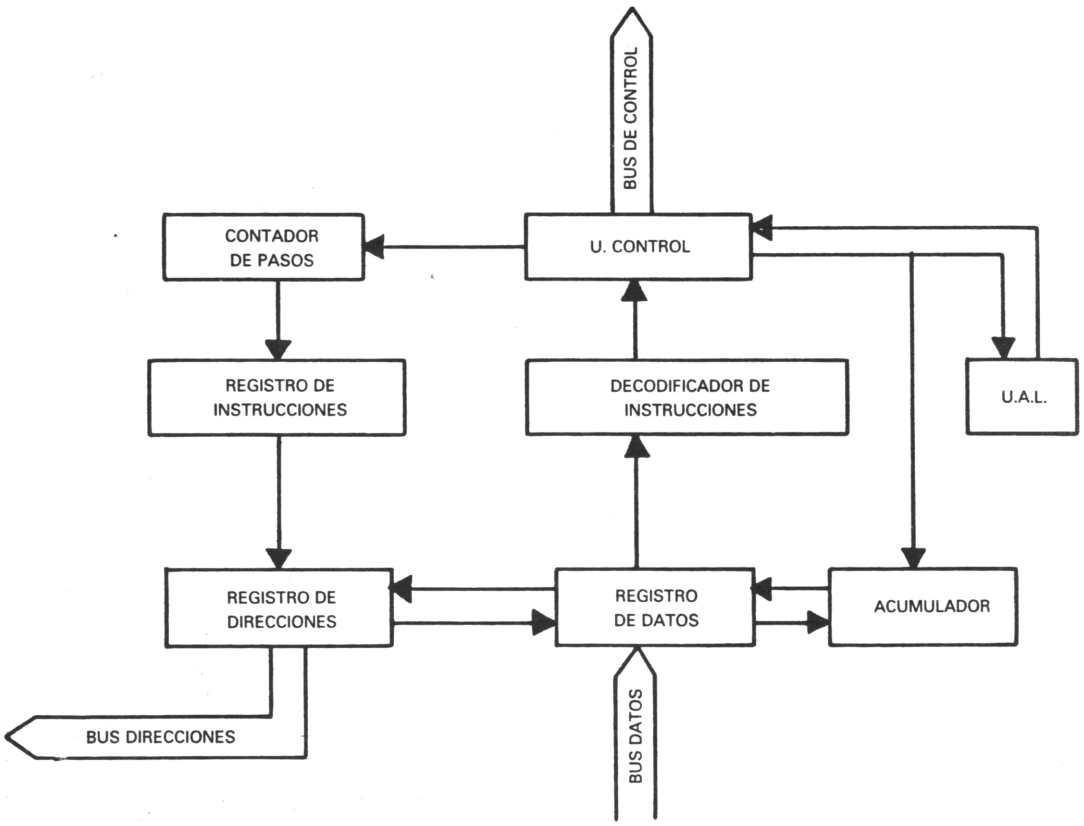
Si una línea de programa fuera LET C = A + B (es decir, el resultado de sumar A con B se almacena en la celda de memoria C), la secuencia de pasos que realiza la U.C.P. es:

PASO 1: Averigua la dirección de memoria donde está almacenado el valor de A. Se hace mediante el registro de direcciones y el bus de direcciones, que averigua el dato en la memoria ROM, y esta dirección entra por el bus de datos.

PASO 2: Averigua el valor de A. Para ello, la dirección hallada anteriormente sale por el bus de direcciones, yendo a la memoria RAM, y averiguando el valor de la variable A, que está allí almacenada. Entra a la U.C.P. por el bus de datos, pasando al acumulador.

PASO 3: Leyendo secuencialmente la orden A + B, ahora corresponde sumar. En la ROM se busca «sumar» (ADD) y un número codificado entra por el bus de datos, yendo al registro de instrucciones. Una vez decodificado, la unidad de control sabe que corresponde a ADD.

Sus partes se pueden esquematizar de la siguiente forma:



PASO 4: Búsqueda de la dirección de B, análogo al 1.

PASO 5: Averigua el valor de la variable B, análogo al paso 2. En cuanto el acumulador tiene los dos valores, los pasa a la U.A.L. (unidad aritmético-lógica), que los suma, por instrucción recibida por la unidad de control.

PASO 6: Almacena la suma en C. El registro de direcciones se encarga de buscar en la memoria una dirección no ocupada cuyo número viene dado por el bus de datos.

PASO 7: La unidad de control manda la dirección anterior y el valor de la suma a la RAM, para su almacenaje.

PASO 8: El contador de pasos sigue contando, y como ya se ha terminado la secuencia, el registro de instrucciones manda a por el número codificado de la instrucción que detiene la secuencia de operaciones, que una vez decodificado hace que la unidad de control detenga el proceso. Así, instrucción tras instrucción.

2. MEMORIAS ROM Y MEMORIAS RAM

ROM viene de Read Only Memory (memorias de solamente lectura). En estas memorias se encuentra el sistema operativo, es decir, cómo tiene que hacer cada cosa, y cuáles es capaz de hacer. Para un mismo microprocesador, según sea la capacidad y programación de la ROM, el or-

denador será más o menos sotisficado. Esta programación la realiza el fabricante y no se borra al desconectar el ordenador.

Así, por ejemplo, el Spectrum y el Amstrad llevan el mismo microprocesador, el Z80A, de ocho bits. Pero el primero tiene 16 K de memoria ROM y el otro 32 K (digamos que sabe hacer «el doble» de cosas).

PROM (Programmable Read Only Memory). Es semejante a la anterior, pero puede ser programada por el usuario, pero una sola vez (esta memoria no la dispone el ordenador personal).

EPROM (Erasable Programmable ROM). Se puede programar varias veces, y el borrado se hace mediante rayos ultravioleta.

EAROM (Electricaly Alterable ROM). Análoga a la anterior, donde tanto la grabación como el borrado se hace mediante impulsos eléctricos.

RAM significa Randon Access Memory (memorias de acceso arbitrario). En ellas se almacenan los datos e instrucciones de un programa. Su contenido se borra al desconectar el ordenador. Cada casilla o celda de memoria contiene un byte, es decir del 0 al 255 para ordenadores de ocho bits.

Cada celda de memoria tiene un número que la identifica por su posición, llamado «dirección». Si decimos, por ejemplo, que una memoria es de 32 K, o también 32 Kbytes, se entiende que su capacidad son 32 millares de bytes (exactamente 32×1024). O sea, cada K supone $2^{10}=1024$ celdas.

Supongamos una memoria de 64 K. El direccionamiento se hace con dos bytes ordenados, uno para la fila y otro para la columna. Como cada byte puede ir de 0 a 255 que son 256 posiciones, el «tablero de memorias» da:
256×256=65536 posiciones.

Las primeras direcciones están ocupadas por la ROM, y el resto es accesible por el usuario.

3. PERIFERICOS

También llamados unidades de entrada-salida, son los sentidos del ordenador, encargados de interrelacionarle con el exterior (usuario, otro ordenador, etc.).

Podemos clasificarlos según sea el flujo de información:

SOLO ENTRADA	SOLO SALIDA	E/S
Teclado	Monitor, TV	Disco
Cartucho	Sonido HIFI	Cassette
Joystick	Impresora	Modem
Lápiz óptico	Plotter	
Sensores	Relés	

SOFTWARE. LENGUAJES DE PROGRAMACION

El software o logical es lo que hace operativo al ordenador. Consiste en un conjunto de instrucciones que siguen una serie de reglas, atendiendo al lenguaje de programación utilizado. El conjunto de instrucciones que se introducen al ordenador para obtener un fin se llama programa.

Aprender a programar bien no es difícil, pero se requiere:

— dedicación,
— cierta imaginación,

— conocimiento de un lenguaje (este es el propósito del presente libro).

Por otra parte, no aprender a programar desemboca en:

— infrautilización del ordenador,
— dependencia,
— futuro «analfabetismo».

Podemos clasificar los lenguajes de programación en tres tipos:

— lenguaje máquina (LM),
— lenguaje ensamblador (LE),
— lenguajes de alto nivel (LAN).

LM: consiste en dar las instrucciones al ordenador directamente en su propio lenguaje (unos y ceros). Para hacer menos penosa la tarea se usa el código hexadecimal, en vez del binario. Esto se realiza de la forma siguiente:

Si cortamos por la mitad un byte nos quedan dos conjuntos de cuatro bit cada uno (nibble). Estos cuatro bit cubren del 0 al 15 decimal (16 números). La correspondencia es:

DECIMAL	NIBBLE	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Entonces, en vez de escribir un byte con ocho dígitos, es más cómodo una sucesión de dos nibble, o sea, dos números hexadecimales. Así, en parejas, es como se introducen en la máquina, una vez que está preparada para aceptarlos (programa objeto).

LE: en este lenguaje las instrucciones a ejecutar no se escriben en lenguaje binario, sino con unas pocas letras que da idea de la instrucción. Por ejemplo, ADD es sumar, y LD es cargar una variable. Estos códigos simplificados se llaman «mnemónicos» del sistema y dependen de cada microprocesador (programa fuente). Este programa, para ser ejecutado, necesita su traducción a código máquina, que la realiza el propio ordenador con un programa llamado «ensamblador». Este lenguaje es más asequible que el LM, permitiendo correcciones con facilidad.

El LM y el LE son lenguajes de bajo nivel.

LAN: se carazterizan por:

— ser más próximos al lenguaje humano,
— independientes del LM de cada aparato (cada aparato posee su compilador que traduce el LAN a LM),

— orientados a una actividad específica,
— diferente versatilidad y estructuración,
— sucesivas mejoras con el tiempo.

Se han desarrollado más de 1.000 lenguajes de alto nivel; sin embargo, sólo unos pocos han adquirido cierta importancia.

Se dice que un lenguaje es estructurado cuando el programa en ese lenguaje se desarrolla en módulos independientes, teniendo cada uno una finalidad específica y fusionándolos para dar el programa total. En estos lenguajes los errores son fácilmente localizados en su módulo y consiguientemente de fácil corrección. Si se quiere modificar un programa, se cambia el módulo correspondiente. Además, un módulo o procedimiento puede ser repetitivo, sin necesidad de volverlo a escribir, lo que provoca una gran velocidad de ejecución. Además, puede llamarse a sí mismo (recursividad).

Entre los no estructurados están:

FORTRAN (FORmula Translation: traducción de fórmulas). Nació a mediados de los 50 y ha tenido sucesivos perfeccionamientos. Los más usados son: Fortran IV y el 77. Están orientados al procesamiento de datos científicos.

COBOL (COmmon Business Oriented Language: Lenguaje común orientado a aplicaciones comerciales). Creado en 1960 para gestión de empresas. Maneja pocos cálculos, pero gran volumen de ficheros, siendo muy rígido y pesado.

BASIC (Beginner's Allpurpose Symbolic Instruction Code: código de instrucciones simbólicas de carácter general para principiantes). Su sencillez, rapidez de aprendizaje, vocabulario limitado, lenguaje próximo al humano y gran variedad de aplicaciones han hecho de este lenguaje el más utilizado y el de crecimiento más rápido desde su creación por John Kemeny y Thomas Kurtz en 1965.

Entre los estructurados, los mejores son:

PASCAL: Desarrollado por Niklaus Wirth en Zurich, a principios de los 70. Es sencillo, pero riguroso. Tiene bastante restricciones en sintaxis, por lo que es preferible para cálculos científicos. Como su compilador no es muy extenso, es adaptable a los microordenadores.

FORTH: Fue creado por Charles H. Moore a mediados de los 70 para control de procesos. Posteriormente se ha utilizado en gráficos por ordenador y aplicaciones de empresa. Es también fácilmente adaptable a los micros.

C: Creado en 1972 por Dennis Ritchie para profesionales de la programación, es muy poco estricto en sus reglas, pues presupone que el programador «sabe lo que está haciendo». Es de nivel medio, pudiendo manipular bits. Es transportable fácilmente de una máquina a otra que tenga distinto microprocesador, y tiene grandes aplicaciones en la construcción de modelos.

LOGO: Pretendiendo aplicar los resultados obtenidos por Piaget acerca del comportamiento cognoscitivo de los niños, Seymour Papert y sus colaboradores desarrollan en el MIT (Instituto Tecnológico Massachussets) durante 1969.

HISTORIA DE LA INFORMACION AUTOMATICA

Podemos clasificar esta historia en tres etapas:

1. De 1623 a 1889: calculadoras mecánicas.
2. De 1876 a 1946: calculadoras analógicas electromecánicas.
3. De 1946 a hoy: ordenadores digitales.

De forma cronológica, los hitos han sido:

2500 A.C.: se tiene conocimiento de la utilización del ábaco desde esta fecha, en distintas culturas, sobreviviendo su uso hasta nuestros días. Efectúa sumas y restas.

1623: Wilhelm Schickard construye la primera calculadora mecánica que se conoce.

1642: Pascal construye una sumadora.

1666: Samuel Morland construye una sumadora-restadora para monedas inglesas.

1671: Leibnitz realiza una máquina que multiplica por iteracción de sumas. En 1694 construye la calculadora universal (4 operaciones).

1806: Jacquard construye un telar automático controlado por una sucesión continua de tarjetas perforadas.

1791-1871: Charles Babbage construye una calculadora mecánica que utiliza un programa registrado en tarjetas perforadas, con unidad de entrada y salida, y con una estructura interna «Von Neumann». El primer programador fue la ayudante de Babbage, lady Ada Augusta, hiją de lord Byron, que llegó a decir: «la máquina no hace nada por sí sola, pero hará cualquier cosa si sabemos decirle cómo hacerlo».

1886: Hollerith, funcionario americano, introduce el sistema binario para realizar el conteo del censo de 1880. Construye el sistema compuesto de lectora eléctrica de fichas (rompe el acordeón de Jacquard para aislar cada item), una tabuladora y un clasificador. Se establece en 1896 y crea la industria que llegará a ser IBM.

1941: Konrad Zuse trabajó en la alemania nazi, aislado y sin apoyo. Reinventa: concepción global de Babbage, la coma flotante de Torres Quevedo, la aritmética binaria de Shannon.
Además construye el primer calculador programable.

1944: Howard Aiken y Thomas Watson, de Harvard e IBM, respectivamente, construyen el llamado Mark1, que es el último calculador de relés, es decir, electromecánico.

1945: se construye el primer ordenador electrónico decimal, el ENIAC. Pesaba 30 Tm. y tenía 18.000 válvulas.

1946: John Von Neuman propone almacenar en la memoria de ordenador la secuencia de instrucciones (programa). Este avance hace que este año sea considerado como el del nacimiento de la informática.

1951: Univac construye el primer ordenador con las ideas de Neuman. Los ordenadores de esta primera generación tienen un tiempo de ejecución de 1 ms.

1956: se descubre el transistor.

1958: se reduce el tiempo de ejecución de 1 μs, gracias a los transistores, que sustituyen a las válvulas.

1965: el tiempo de ejecución se reduce a 1 ns, con la sustitución de transistores por circuitos integrados.

1970: se están realizando proyectos de inteligencia artificial que culminarán con los ordenadores de la 5.ª generación. Los objetivos a lograr son: reconocimiento de formas y frases, hacer hipótesis o suposiciones, obtener conclusiones y aprendizaje y su aplicación práctica.

Generación	Tecnología	Hardware	Software
1.ª (46-54)	tubos de vacío memorias acústicas	aritmética de coma fija	lenguaje máquina lenguaje ensamblador
2.ª (58-64)	transistores núcleos de ferrita	aritmética de coma flotante registros de índice procesador E/S	lenguajes de alto nivel librería de subrutinas
3.ª (65-74)	circuitos integrados SSI, MSI	microprogramación pipelines memoria caché	multiprogramación multiproceso sistemas operativos memoria virtual
4.ª (75-80)	circuitos integrados LSI, VLSI	microprocesadores primera EPROM	nuevos lenguajes y métodos

EVOLUCION EN LOS DIVERSOS PROCESOS DE LA INFORMACION

Etapas / Proceso	1.ª	2.ª	3.ª	4.ª	5.ª
Reconocimiento	Lenguaje	Escritura	Símbolos	Señales analógicas: teléfono, radio, TV	Señales digitales
Transmisión	Recitadores Pinturas murales	Mensajeros	Señales ópticas	Telecomunicaciones	Terrestre Vía satélite
Tiempo aproximado en recorrer 40 km.	indefinido	un día	diez minutos	instantáneo	instantáneo
Procesamiento	SER HUMANO				Ordenador

NOTA: Cada etapa supone una mejora de la anterior, *sin excluirla.*

APLICACIONES DE LA INFORMATICA

En ciencia y tecnología:

- diseño de nuevos productos, construcciones y vehículo.
- comprobación de teorías físicas.
- predicción meteorólgica.
- aparatos de diagnóstico y tratamiento médico.
- estudios sociológicos y estadísticas.

En gestión

- actualización de cuentas corrientes, nóminas, inventarios.
- cajeros automáticos, tarjetas de crédito.
- mantenimiento y actualización de grandes bases de datos.

Control industrial:

- instrumentos de navegación.
- cohetes, armas.
- control de cualquier etapa en un proceso industrial.

Simulación de procesos:

- desarrollo de modelos matemáticos y físicos.
- desarrollo de modelos lógicos (sistemas expertos).

Tratamiento de la información:

- reconocimiento y tratamiento de voz e imágenes.
- traducción automática, procesadores de textos.

En la enseñanza:

- para aprender un lenguaje informático.
- como ayuda en la enseñanza de cualquier materia.

Generación de dibujos y figuras:

- aplicaciones tecnológicas y artísticas.
- composición y análisis de música.

En aparatos de uso corriente:

- videojuegos, juguetes, relojes digitales.
- programadores de electrodomésticos.
- tableros indicadores de coches.

PROFESIONES EN EL SOFTWARE

Analista de aplicaciones: Analiza de una manera global el problema a resolver con la informática.

Analista de sistemas: Trata de encauzar el problema teniendo en cuenta el ordenador de que se dispone y el sistema operativo que utiliza.

Programador: Realiza el programa en un determinado lenguaje, siguiendo las indicaciones del analista.

Operador: Controla el funcionamiento del ordenador a la hora de ejecutar el programa.

El usuario de un microordenador es todo esto a la vez y, como observamos, lo último que se realiza es el contacto físico con el aparato.

TEMA 2

EL BASIC

```
10> REM  Calculos geometricos
         Para cubos
20 INPUT "Arista? ";a
30 LET d=a*1.73
40 LET v=a*a*a
50 PRINT "  Diagonal = ";d
60 PRINT
70 PRINT "  Volumen = ";v

 run
Arista = 50
Diagonal = 86.5
Volumen = 125000
```

PROGRAMA N.º 6.

Todo lenguaje contiene un número de sentencias u órdenes, sirviendo cada una de ellas para un cometido concreto y muy elemental. Estas sentencias han de seguir unas normas generales:

— Orden riguroso,
— Sintaxis específica,
— Lógica formal.

En BASIC, el orden se consigue asignando a cada orden (o conjunto de órdenes) un número de línea (n.l.). El ordenador las ejecuta secuencialmente, por orden creciente.

La sintaxis es el modo de escribir el programa. Es decir, el formato de cada sentencia, y su conexión.

La lógica es la coherencia en la elaboración del programa, para conseguir el resultado buscado. Para ello te pueden ayudar los organigramas, explicados en el Cap IV.

ETAPAS DEL SOFTWARE

1.—ESCRIBIR EL PROGRAMA
2.—IMPLEMENTACION (TECLEADO)
3.—EJECUCION
4.—CORRECCION DE ERRORES
5.—MEJORA, SI ES POSIBLE

SENTENCIA REM

Como su propio nombre indica (REMark = comentario), no tiene otra finalidad que la de intercalar comentarios al programa, que sirven de seguimiento. Por tanto, en sí no es operativa, y el ordenador no hace nada cuando se la encuentra en la ejecución del programa.

Su formato es:

```
n.l.  REM (aspecto a comentar)
```

Esta es la única sentencia «inocente». Todas las que vienen son verdaderamente operativas.

VARIABLES. SENTENCIA LET

La memoria RAM se puede considerar formada por celdas individuales (semejante a las celdas de un panal), que pueden almacenar caracteres (letras, números o gráficos). A estas celdas se les llamará *variables* porque su contenido puede variar.

A las celdas que van a contener información se les asigna un nombre (usualmente de una o dos letras), que es propiamente la variable y mediante la sentencia LET se verifica su llenado.

LLENADO DE CELDAS

LET
INPUT
READ

Pueden ser:

— Numéricas, cuando guardan un número

Ej.:
```
n.l.  LET t=32
```

A la celda que guarda el número 32 le llamamos «t». En las líneas siguientes a ésta, siempre que te refieras a la variable t, estarás utilizando el contenido numérico que tenga.

— Alfanuméricas, cuando guardan texto o números (en este caso se pierde el carácter aritmético del número). Es preciso hacer la asignación entrecomillada.

Ej.:
```
n.l  LET A $="María"
n.l.  LET B $="Juan nació el 2-4- 1980"
n.l.  LET C $="32"
```

Observa la diferencia entre la variable t anterior y C $: con t se pueden hacer operaciones aritméticas, y con C $ no, pues el micro toma a «32» como palabra. En algunos micros no se necesita pulsar la sentencia LET, y así, se pondría:

```
n.l.   t=32
n.l.  C $="María"
```

VARIABLES	NOMBRES
NUMERICAS	A, IN, sto... J1, ZX5
ALFANUMERICAS	A$,ZC$... K3$, AF1$...

También se puede imprimir una expresión literal, encerrándola entre comillas.

Ej.: n.l. PRINT "Informática"

Ante esta forma del PRINT el micro actúa como si fotografiase lo que está entre comillas, y lo vertiese por la pantalla.

NO ES LO MISMO

PRINT 5+7

QUE

PRINT "5+7"

PROGRAMA COMENTADO

El ordenador te va a pedir el año actual, y te dirá cuántos han transcurrido desde el descubrimiento de América.

El siguiente programa te lo realiza:

```
10 LET a=1492
20 PRINT "DIGITA EL AÑO ACTUAL"
30 INPUT b
40 LET x=b-a
50 PRINT "Han transcurrido"
60 PRINT x
70 PRINT "años desde el descubrimiento de América"
```

La línea 10 asigna a la variable a el valor 1492 (al tratarse de un dato fijo, la asignación es preferible hacerla como LET). Sin embargo, como el año actual es variable, por línea 30 se le comunica este dato al ordenador.

Una vez asignada la resta a la variable x, la línea 60 permite su impresión por TV.

Esto corresponde a la primera etapa del software (ver pág. 19). Ahora te corresponde a ti cargarlo en tu micro (segunda etapa). Para ello, una vez tecleada la primera línea del programa, has de pulsar ENTER (o su equivalente en tu micro, y así queda almacenada en la memoria RAM.

Una vez hecho esto con todas las líneas del programa, pasamos a la tercera etapa del Soft. Es el momento de pulsar RUN. Con esta instrucción el micro ejecuta el programa.

Para la quinta etapa del soft habrá que esperar a que lleguemos al capítulo 3 º.

SENTENCIAS: INPUT, PRINT

INPUT sirve para introducir datos por el teclado, numéricos o no numéricos. Es la segunda forma de llenar una variable.

Si ponemos

```
30 INPUT x
```

cuando el ordenador llegue a la línea 30 esperará que le introduzcas un dato numérico, para luego proseguir con el resto del programa.

Si pulsamos

```
70 INPUT a $
```

el dato que esperaría que introduzcas sería alfanumérico.

PRINT saca en TV. aquel dato numérico y/o no numérico que se le ordene.

Esta sentencia tiene enormes posibilidades en BASIC, pudiéndolas clasificar en tres grupos:

— cuando no se especifica el lugar de impresión,
— cuando se especifica la posición en la línea,
— cuando se tiene acceso instantáneo a cualquier lugar de la pantalla.

Empezamos con el análisis del primer grupo

Puede realizar:

— impresión de constantes u operaciones entre ellas,
— impresión de variables u operaciones entre ellas.

```
GRUPO 1

   PRINT a
   PRINT X$
   PRINT "La Primavera"

GRUPO 2

   USAR SIGNOS , ; y TAB

GRUPO 3

USAR PRINT AT
```

Ej.: n.l. PRINT 7+5
n.l. PRINT a-b

En el primer caso imprimirá un 12 (al principio de la pantalla), y en el segundo imprimirá el resultado de la operación. Para ello es necesario que los valores a y b sean conocidos por el ordenador.

COMANDOS: NEW, LIST, EDIT, RUBOUT (DELETE)

NEW: Sirve para dejar en blanco todas las celdas de memoria RAM. Es necesario usarla antes de introducir un nuevo programa.

LIST: Si deseas ver el listado del programa, bien cuando lo estés introduciendo o una vez ejecutado (por ejemplo, para corregir un error), se usa el comando LIST, que tiene las siguientes posibilidades en los micros más completos:

— para ver el listado completo: pulsas LIST + ENTER,
— para ver una sola línea: pulsas LIST n.l. + ENTER,
— para ver el listado a partir de una línea determinada, pulsas LIST n.l. - + ENTER,
— para ver el listado comprendido entre dos números de línea: LIST n.l.-n.l. + ENTER.

Una vez localizada la línea en la que hubiera un error para corregirla:

1. Con las teclas ↑ o ↓, se lleva el cursor hasta la línea a corregir.
2. Pulsas EDIT, manteniendo previamente pulsada la tecla SHIFT. Así baja la línea a la parte inferior de la pantalla (lo que se denomina «editarla»). (Sólo para Spectrum).
3. Con las teclas → y ← correremos el cursor hasta situarlo a la derecha de la parte a corregir.
4. Pulsas SHIFT y a la vez RUBOUT (borrar). Así se irá borrando la parte deseada. En algunos micros es DELETE. (Si hubiera que intercalar algo, se puede hacer en cada momento en la posición que está el cursor. En algunos micros es necesario pulsar previamente INSERT.)
5. Una vez que la línea está modificada, con ENTER pasa corregida a la memoria.

En todos los aparatos, si se desea borrar un línea completa de programa, bastará pulsar su n.l. + ENTER. A veces te resultará más práctico hacer de nuevo una línea que corregirla.

PROGRAMAS

1. Las instrucciones que siguen tienen cada una de ellas una incorrección en el lenguaje BASIC. Detectar el error

```
10  LET x=2,05
15  PRINT k=12
20  LET x+y=2.7
30  INPUT x=5
40  LET 4=x
50  LET x=3y + z
60  PRINT LET a=7
```

2 Programa que calcule el volumen de una esfera, conocido su radio.

3. Hacer un programa para calcular superficies de rectángulos, conocidos el largo y el ancho.

4. Programa que calcule la media aritmética de cinco números cualesquiera.

5. Programa que te dé la longitud de una circunferencia y la superficie de su círculo, conocido el radio.

6. Programa que calcule la diagonal y el volumen de un cubo, sabiendo su arista.

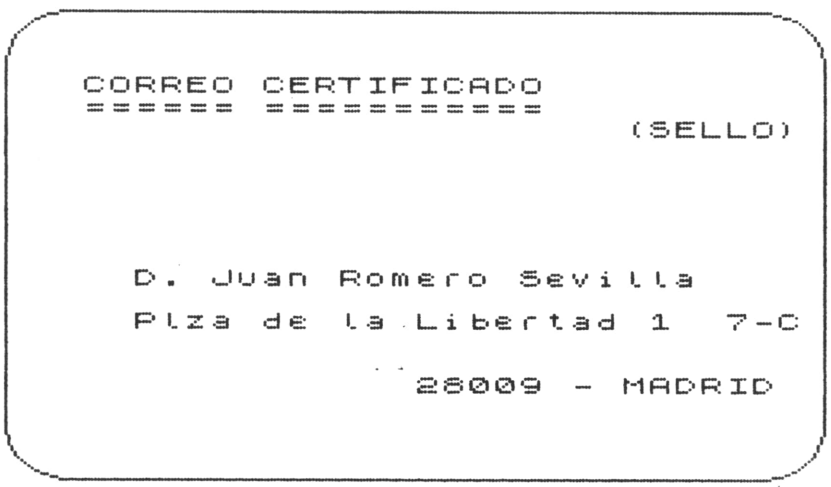
7. Programa que calcule la diagonal, superficie total y volumen de un paralelepípedo, conocidas sus tres dimensiones.

8. Dadas las dos bases y la altura de un trapecio rectángulo, hacer el programa que determine su área y el lado que falta.

9. Cierta moqueta está a 72 pesetas/m^2. Realizar el programa que dé el coste de la moqueta de una habitación rectangular cuyas dimensiones le debes dar al micro en cm. Hay un descuento del 12% por pronto pago, que también deberá aparecer en pantalla.

10. El ordenador pide los datos del D.N.I. a un individuo, y los imprime sacándolos por pantalla.

TEMA 3
PRIMERAS OPERACIONES Y VISUALIZACION



PROGRAMA N.º 9.

NOTACION DE NUMEROS

Han de observarse las siguientes normas:
- En los números decimales se utilizarán el (.) en vez de la coma.
- Si delante del (.) está el cero, éste se puede suprimir. Ej.: 0.25 = .25
- Notación científica en BASIC

NUMERO	NOTACION CIENTIFICA	BASIC
300000000	3.10^8	3E8
0.004	4.10^{-3}	4E-3

Es decir, el número que sigue a la E es el exponente del 10.

Delante de E debe aparecer siempre un número, aunque sea el 1. Ej.: 10^{15} debe escribirse 1E15 (E15 daría error).

El número que sigue a la E ha de ser entero (no decimal), y con signo, aunque si es positivo se suprime.

— Los extremos entre los que pueden trabajar los micros generalmente son:

10^{-38} ..... 10^{38}

— Si una variable sabemos que *sólo* va a utilizar números enteros, podremos poner a continuación del nombre de la variable el símbolo % (Ej. INPUT r%), pues de esta forma el micro no utilizará tanta memoria. Pero si hacemos esto, los extremos se reducen al intervalo

-32767 32767

OPERACIONES Y SU JERARQUIA

Al igual que una buena calculadora (no programable) los micros pueden hacer las siguientes operaciones, tanto con números como con variables:

— suma +
— resta -
— producto *
— cociente /
— potencia ↑ (compatibles **)
— raíz cuadrada SQR (recuerda que cualquier raíz es una potencia. Ej.: SQR(4)=4↑.5)
— exponenciación EXP } la base es el número e
— logaritmación LN }

La ventaja del ordenador sobre una calculadora es que puede hacer cálculos de forma iterativa (repeticiones sucesivas), hasta llegar a un resultado.

Vamos a ver cómo trabaja el micro en el caso siguiente:

```
n.l. LET x=2+3*5+3↑2
n.l. PRINT x
```

Cuando corras el programa, se verá en TV el valor 26. El micro ha efectuado las siguientes operaciones:

— la potencia 2+3*5+9
— el producto 2+15+9
— la suma =26

De manera que la jerarquía es la siguiente:

1. Paréntesis, teniendo preferencia los interiores.
2. Potencias.
3. Productos y cocientes.
4. Sumas y restas.

Si hacemos ahora:

```
n.l. LET x=((2+3)*5+3)↑2
n.l. PRINT x
```

el orden que sigue el micro es

— paréntesis interno (5*5+3)↑2
— producto (25+3)↑2
— paréntesis externo { producto (25+3)↑2 / suma (28)↑2
— potencia 28↑2 = 784

Veamos algunos ejemplos para que practiques:

```
20+60/2+4          Resultado: 54
(20+60)/2+4                   44
20+60/(2+4)                   30
(20+60)/(2+4)                 13.33...
```

Obsevación: Con respecto a las potencias A^B hay que tener en cuenta:
A↑B se puede realizar:
— si $A \geqq 0$, para todo valor numérico de B,
— si $A < 0$, sólo para valores enteros de B.
Si $A < 0$ y B es fraccionario, el micro no efectúa la potencia, y saca mensaje de error, aunque fuera una operación correcta bajo el punto de vista aritmético. Así,

(-8)↑3=-512
(-8)↑(-3)=-0.00195, aunque este valor lo sacará por TV en la forma -1.95E-3
(-8)↑(-3) que equivale a $\sqrt[3]{-8} = -2$ no lo efectuará el ordenador.
El Spectrum no efectúa ninguna potencia de base negativa (para el primer caso, deberías poner (-8)*(-8)*(-8), y lo mismo para variables.

CADENAS

Es un conjunto formado por caracteres cualesquiera y que deben ir entre comillas, como por ejemplo:

```
"28040-MADRID"
```

Si quieres asignar una cadena a una variable, ésta debe ser alfanumérica, y así pondríamos:

```
n.l.  LET x $= "28040-MADRID"
```

Observaciones

— El número de caracteres de una cadena no puede ser superior a 255.
— Los espacios en blanco han de considerarse como un carácter más. Así, x$ tiene 12 caracteres.
— Se pueden «sumar» cadenas, entendiendo que esta suma es una yuxtaposición. Así:

```
"Buenas tardes" + "," + "ordenador"
```

es igual a:

"Buenas tardes, ordenador"

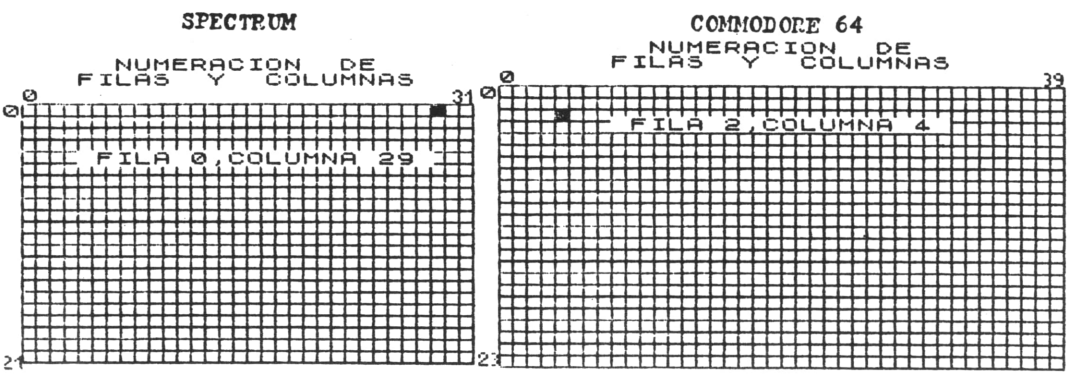
En el capítulo VIII veremos un amplio tratamiento de estas variables.

— No poseen valor numérico desde el punto de vista aritmético:

"1984" ≠ 1984

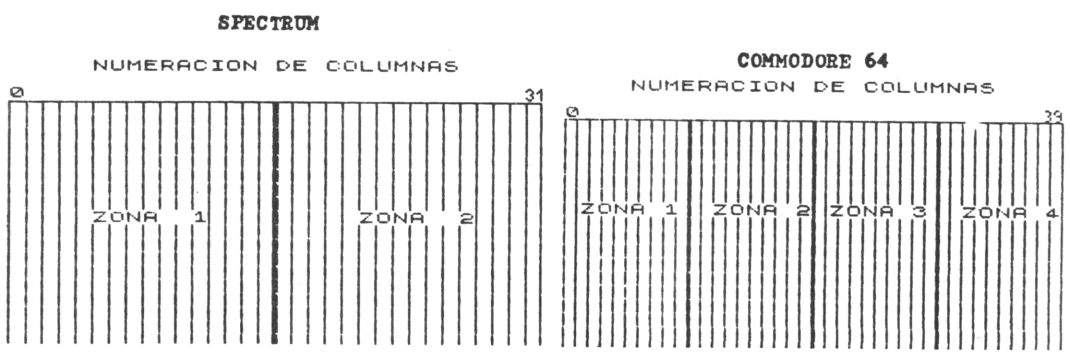
USO DEL (;) Y DE (,) EN LA INSTRUCCION PRINT

Según el tipo de micro, a efectos de impresión, la pantalla queda dividida en un mayor o menor número de filas y columnas invisibles. Por ejemplo, en el Spectrum hay 32 columnas, numeradas de la 0 a la 31 y, 22 filas, numerada de la 0 a la 21.



Cada carácter impreso en pantalla ocupa la intersección de una fila y una columna.

Por otra parte, la pantalla se puede considerar dividida en «zonas» verticales, que pueden ser 2, 3 ó 4 dependiendo del micro:



Los PC presentan 24 filas por 80 columnas dividiéndose en 5 zonas.

El (;) lo que hace es imprimir a continuación de lo que tenga impreso antes.

La (,) imprime en la siguiente «zona» de TV el siguiente carácter a imprimir. Te aconsejamos que introduzcas los siguientes ejemplos en tu micro, para que veas sus peculiaridades:

Ej. 1:
```
10 PRINT 1;2;
20 PRINT 3;4
30 PRINT 5,6;7,
40 PRINT 8
50 PRINT
60 PRINT 9,,10;11;
70 PRINT
80 PRINT 12
```

Ej. 2:
```
10 PRINT 1; "A";
20 PRINT 3; "BC"
30 PRINT -5, "DEF 1"; 7,
40 PRINT 8
50 PRINT
60 PRINT 9,,"GHIJK",-11;
70 PRINT
80 PRINT 12
```

La instrucción PRINT (sin más) puede hacer:

— Dejar una línea completa en blanco, como las líneas 50 de los ejemplos.

— Detrás de un (;) o (,) deja en blanco el resto de la línea correspondiente, como las líneas 70 de los ejemplos.

Observaciones

1. Por muchos (;) que se ponga, no se va a ir distanciando uno a uno el carácter a imprimir.
2. Si se ponen varias (,), por cada una salta a la siguiente zona de TV (fíjate en las líneas 60).
3. Algunos micros te dejan un espacio delante de los números para el signo; si es positivo no aparece, y si es negativo ocupa este espacio.

PRESENTACION DE TITULOS Y RESULTADOS. EL USO DE LAS "

En la ejecución de un programa interesa que aparezca el nombre de los datos a introducir, y que los resultados de la ejecución del mismo tengan una expresión literal que los identifique. Esto se tiene en cuenta en el programa siguiente:

```
10 REM Cálculo de volúmenes
15 PRINT "Te voy a calcular el volumen de una caja"
20 PRINT
25 PRINT "largo? ";
30 INPUT I
35 PRINT I
40 PRINT "fondo? ";
```

```
45 INPUT f
50 PRINT f
55 PRINT "alto? ";
60 INPUT a
65 PRINT a
70 PRINT
75 PRINT "Volumen= ";l*f*a
```

El interrogante de las líneas 25, 40 y 55 es conveniente ponerlo en el Spectrum (ya que en realidad se trata de una pregunta que te hace el micro), pero en otros ordenadores resultaría innecesario porque sale en pantalla automáticamente con la instrucción INPUT.

Se puede fusionar la estructura de las líneas 25 y 30 en una sola línea de programa:

```
25 INPUT "largo? ";l          INPUT fusionado →
```

Así las líneas 30 y 35 son innecesarias.

Observciones

— Con el INPUT fusionado, la pregunta aparece en una parte reservada de la pantalla, que se encuentra por debajo de la zona operativa en el Spectrum.

— Un INPUT fusionado admite la introducción de varios datos a sus variables correspondientes, siempre que éstas estén separadas por (,).

Ej.: `n.l. INPUT "Medidas de la caja? ";l,f,a`

También `n.l. INPUT "Nombre y edad? ";n $,e`

LA INSTRUCCION PRINT TAB

Tiene el formato

```
n.l. PRINT TAB (argumento);(escritura)
```

donde (argumento) puede ser una constante, una variable conocida o una operación con variables; (escritura) es lo que se desea imprimir.

El argumento especifica la columna donde comienza la impresión. Van numeradas de la 0 a la 31, en el Spectrum.

Observaciones

— Si el argumento fuese un número decimal, toma el entero más próximo.

— Si el argumento fuese mayor que 31, le resta esta cantidad tantas veces como sea necesario para que entre en pantalla.

— Dentro de una misma línea se pueden poner varios TAB, pero cuidando que las partes a imprimir no se superpongan, pues de lo contrario la impresión pasaría a la línea siguiente.

— Lo mismo sucedería si hay dos TAB en la misma línea de programa, pero el argumento (columna) del segundo es menor que el del primero.

— Algunos micros, al imprimir datos numéricos, reserva el primer espacio de escritura para el signo, omitiéndolo si fuese positivo.

Ejemplo: este programa dibuja una serie de 20 asteriscos en diagonal.

```
10  FOR x=1 O 20
20    PRINT TAB(x);"*"
30  NEXT x
```

LA INSTRUCCION PRINT AT

El formato es:

```
n.l. PRINT AT f,c; (expresión)
```

donde los parámetros f,c se refieren a la fila y columna, respectivamente, pudiendo ser: constantes, variables o fórmulas. La (expresión) puede ser:

— cadena, encerrada entre comillas,
— variable alfanumérica,
— variable numérica, u operaciones entre éstas,
— números, sin comillas, u operaciones con números.

Hay ordenadores que no utilizan esta instrucción. A cambio, tienen otra que posiciona el cursor en el origen (extremo superior izquierdo de la pantalla), y mediante la repetición de tantas líneas en blanco como filas se quieran bajar y la sentencia TAB se posiciona el cursor en la parte a imprimir.

Para PC, ver apéndice.

Observaciones

— Los valores de los parámetros debieran, en principio, estar comprendidos entre

$0 < f < 21 \qquad 0 < c < 31$

Pero sucede lo siguiente en el Spectrum.

Si c fuese negativo, hace un efecto de reflexión sobre el borde izquierdo, y se imprime. Análogamente, si f es negativo, la reflexión se hace sobre el borde superior. Sin embargo, no existe reflexión a la derecha, ni en la parte inferior de la pantalla.

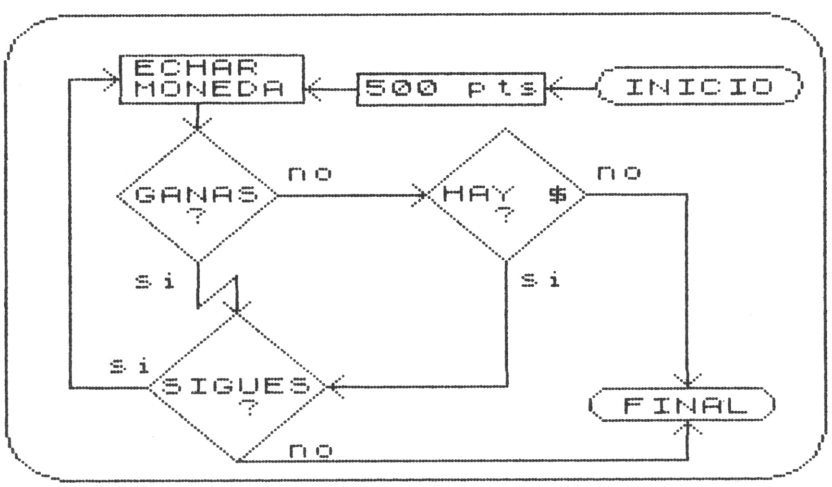
— Si los valores f, c no fuesen enteros, el ordenador los redondea para imprimir.

— Ten siempre presente que el origen de estos parámetros está en el ángulo superior izquierdo de la pantalla. Las filas se enumeran hacia abajo, y las columnas hacia la derecha.

Veamos un primer ejemplo:

```
10  REM PRACTICA CON EL PRINT AT
20  LET a$ ="HOY ES ":LET a=14
30  PRINT AT 0,0;a$ ;AT 3,7;a;AT 6,12;" ABRIL"; AT 11,22;1985
```

TEMA 4
DIAGRAMAS DE FLUJO



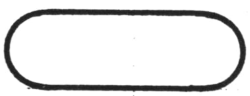
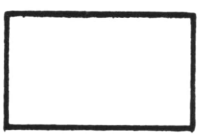
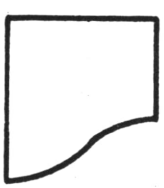
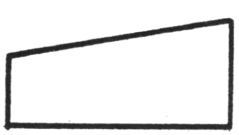
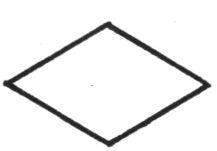
PROGRAMA N.º 7.

DIAGRAMAS DE FLUJO U ORGANIGRAMAS

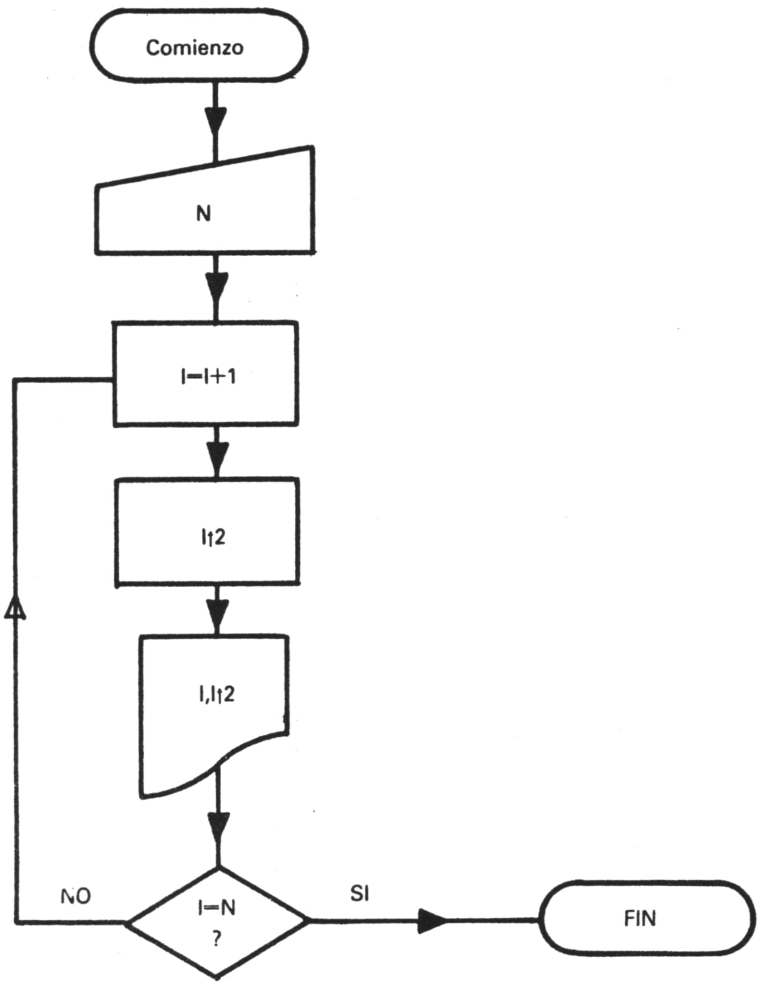
Son la presentación esquemática de cualquier manifestación o hecho desarrollado paso a paso y en orden mediante la utilización de símbolos.

Los utilizamos en Informática, pues nos pueden servir para la elaboración de programas. Hoy en día, con los lenguajes de alto nivel estructurados apenas se utiliza; sin embargo, para el principiante, o bien si el programa resulta bastante complicado, nos puede ser de utilidad y ayuda.

Los símbolos más utilizados son:

SIMBOLOS	SIGNIFICADO
	Comienzo, fin, interrupción.
	Proceso, tratamiento.
	Continuación a otra parte del organigrama.
	Impresión (TV o papel).
	Entrada de datos.
	Decisión lógica.

Imaginemos que queremos ver en TV una tabla de números y sus cuadrados correspondientes, hasta un número determinado, N. El organigrama asociado es:



Como 2.º ejemplo, ver el organigrama al principio del capítulo.

PROGRAMAS

Realiza los diagramas de flujo asociados a las siguientes situaciones:

1. Urge que se avise a Juan esta mañana. Le vas a llamar por teléfono cada cinco minutos hasta que te conteste; si llegan las doce, lo dejas. Cuando conteste, le dices que anule el pedido.

2. Juani, vete a la tienda, y si la ternera está a menos de 1.000 ptas./Kg. te traes dos kilos; si no, trae un kilo de rape, si estuviera fresco. De lo contrario trae dos bolsas de garbanzos. Y no te olvides de traer el pan y el periódico.

A**3.** Voy a mirar el aceite de mi coche; si llega a la marca me voy de viaje; si no, iré a por una lata de un litro a la gasolinera, se lo echaré y volveré a mirar el nivel.

4. Intento pasar a la Universidad este año. Pero mañana voy a una entrevista, y si me sale el trabajo dejo de estudiar. Estoy repitiendo COU, y sólo me quedan las matemáticas y la filosofía. Ayer hice un examen de filosofía, y si lo suspendo la dejo para septiembre. Si lo apruebo, seguiré con las dos, a ver si las saco en junio.

5. Quedan dos partidos de liga. El Barça tiene 25 puntos, el Madrid 24 y el Atletic 23. Hacer el organigrama de todos los posibles casos de puntuación final, sabiendo que se tienen que enfrentar en el último partido el Madrid y el Barcelona.

6. Pedro está jugando a las siete y media con unos amigos. Organigrama del juego de Pedro.

7. Vas a jugar a una máquina tragaperras con 500 ptas. en monedas de 25. Organigrama del juego.

8. Lo mismo que el anterior, pero suponiendo que tu objetivo consista en ganar 100 ptas.

9. Un ejecutivo va a la oficina andando si hace sol, y si llueve coge el coche. Trabaja toda la mañana, y come en la oficina, excepto los martes, que va a comer a casa de sus padres. Si hay trabajo atrasado se va otra vez a la oficina, y si no se va a hacer deporte (los L,X,V hace footing y los M,J, natación). Después va a casa a cenar y acostarse. Los fines de semana se va a la sierra.

TEMA 5
TRANSFERENCIAS DE CONTROL

PROGRAMA N.º 4.

SENTENCIA GOTO

Posibilidades del
GOTO

NUMERO DE LINEA
VARIABLE

Si ponemos, en un programa, p. ej.:

```
50  GOTO 250
```

el ordenador, cuando llega a la línea 50 salta, sin más, a la 250. Tiene utilidad para que un conjunto de instrucciones no sean ejecutadas en ese momento. Si metemos el programa:

```
10  REM volúmenes de cubos
20  PRINT "volúmenes de cubos"
30  INPUT  "arista? ";a
40  PRINT "ARISTA=";a,"VOLUMEN=";a * a * a
50  PRINT
60  GOTO 30
```

cuando el ordenador llegue a la línea 60, mandará el control a la línea 30, y te pedirá una nueva arista y así sucesiva e indefinidamente.

¿Cómo detendremos el proceso?

Para ello los micros disponen de un comando que interrumpe la ejecución de un programa en marcha. Es la tecla BREAK, que, tras pulsarla, además de detener el programa, nos da el mensaje de interrupción, con la última línea ejecutada.

En ordenadores compatibles, hay que pulsar simultáneamente las teclas CTRL y BREAK.

En el C-64, la interrupción es con la tecla RUN/STOP. Si deseamos continuar con el programa que se ha interrumpido (corregido o no), usaremos el comando CONT (continúa), y el ordenador proseguirá en la línea donde se detuvo.

Comentario: Si alguna vez ves, en un listado, por ej.:

```
500 GOTO 500
```

como el programa no termina nunca, no sacará el mensaje de terminación, con lo que la presentación en pantalla queda más «limpia». Para proseguir, con el BREAK.

NOTA: En PC, el programa se detiene si le mandas a un n.º de línea no existente.

SENTENCIA IF... THEN...

Tiene el formato general IF (A) THEN (Z) donde (A) es una comparación (en el sentido igual, mayor o menor) entre variables y/o constantes, y (Z) es la instrucción o grupo de instrucciones a ejecutar. Pero sólo se ejecutarán si la condición (A) es cierta. Si no lo fuese, el ordenador pasa a ejecutar la siguiente línea del programa.

Nota que (Z) puede ser *cualquier* instrucción, de las ya vistas o de las que veremos. Ejemplos:

```
n.l.  IF p=2 THEN GOTO 400
n.l.  IF a > b+c THEN PRINT "valor=";a
n.l.  IF z/3 < = x-1 THEN LET x=9
n.l.  IF c$="s" THEN INPUT "altura? ";h
n.l.  IF i$="3" THEN CLS: GOTO 5
n.l.  IF a=2 AND b=3 THEN a=b
n.l.  IF a=2 OR b=3 THEN a=a+1
n.l.  IF =2 AND (b=3 OR b=-3) THEN LET b=0
```

Observaciones

— Algunos micros tienen la sentencia complementaria ELSE, que opera de la siguiente forma:

```
n.l.  IF a>10 THEN GOTO 100 ELSE PRINT x+y
```

Si se verifica la condición, el micro iría a la línea 100; y si no, imprime el valor x+y.

— Si en la comparación (A) se utilizan combinaciones de operadores AND, OR debes agruparlos con paréntesis siguiendo las reglas del álgebra de Boole.

— Es importante tener muy en cuenta que si no se verifica la condición (A), no se ejecutará *ninguna* de las instrucciones (Z). Así, en el ejemplo 5.º, no se ejecutaría el borrado de pantalla (CLS) ni la transferencia de control a la línea 5.

— La condición (A) puede ser simplemente una variable sin más. Entonces la condición es cierta si esta variable es distinta de 0.

Ej.: n.l. IF m THEN GOTO 5

Utilizando NOT la condición es cierta si la variable es igual a 0.

```
10 LET Q=0
20 IF Q THEN PRINT "X"
```

NO IMPRIME

```
10 LET Q=5
20 IF Q THEN PRINT "X"
```

IMPRIME X

```
10 LET Q=0
20 IF NOT Q THEN PRINT "X"
```

IMPRIME X

OPERADORES LOGICOS

Sirven para comparar constantes, variables y fórmulas. Su aplicación entre variables alfanuméricas se verá en el Cap. VIII. Son:

OPERADOR	SIGNIFICADO
OR	o
AND	y
NOT	no (no es cierto)
=, <>	igual, distinto a
<, <=	menor, menor o igual que
>, >=	mayor, mayor o igual que

PRIORIDAD

NOT
AND
OR

Si usamos el operador OR relacionando dos premisas como por ej.:

```
a=7  OR  x > 5
```

basta que sea cierta *una* de las dos para que sea cierta la proposición total.

En cambio, usando el AND, como por ej.:

```
a=7  AND  x > 5
```

solamente será válida la proposición total si se verifican *las dos* premisas.

El operador NOT cambia radicalmente la comparación de manera que se obtiene la complementaria. Así:

```
NOT  q < 5
```

es equivalente a q >= 5.

Imagina que hay que comprobar si el valor de la variable h está en cualquiera de los dos intervalos que se señalan en la figura

la comparación sería:

```
(h >=-5 AND h <=-2) OR (h >=7 AND h <=11)
```

Como se ha alterado la prioridad AND/OR, es preciso utilizar paréntesis.

```
NOT (A=B) <==> A<>B
NOT (A<>B) <==> A=B
NOT (A AND B)<==>NOT A OR NOT B
NOT (A OR B)<==>NOT A AND NOT B
```

CONTADORES. EL DATO CLAVE

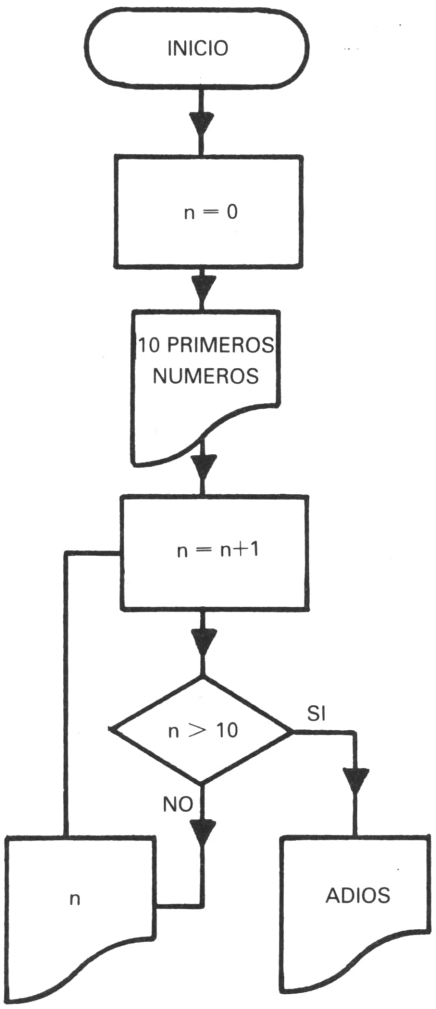
Un contador típico es de la forma LET n=n+1 (se puede usar cualquier variable).

Esta expresión no es correcta desde el punto de vista algebraico, pero sí en programación

Cuando el micro llega a esta orden, simplemente aumenta en 1 el valor que tenga la variable n en ese momento. Esta instrucción va a ser repetitiva, en tanto se cumpla una determinada condición. Si no, se sale del bucle.

Por ejemplo, para imprimir los diez primeros números naturales, se haría:

```
 5  LET n=0
10  PRINT "10 PRIMEROS NUMEROS"
15  LET n=n+1
20  IF n > 10 THEN GOTO 40
25  PRINT n
30  GOTO 15
40  PRINT "ADIOS"
```



En este ejemplo el contador (variable n) salta de uno en uno, pero no necesariamente tiene que ser así. En ocasiones puede interesar saltos diferentes, como:

```
n.l.  LET n=n-5
n.l.  LET p=p+2 . 5
```

Imagina un programa en el que el micro nos va pidiendo datos continuamente, para procesarlos, mediante INPUT. ¿Qué manera tiene el micro de saber que la lista de datos ha teminado?

Pues utilizando un valor o cadena (según el caso) que se le introduce como dato clave que debe ser conocido por el operador.

Si intentamos que el micro cuente el número de contestaciones afirmativas, negativas y abs-

tenciones a una pregunta de una encuesta, asociando al «SI» el 1, al «NO» el 2 y a la abstención el 0, el programa siguiente lo realiza:

```
  5 LET x=0:LET Y=0 Z=0
 10 PRINT "Recuento de votos"
 20 INPUT "Dato? ";a
 30 IF a=- 5 THEN GOTO 90
 40 IF a=1 THEN LET x=x+1
 50  IF a=2 THEN LET y=y+1
 60 IF a=0 THEN LET z=z+1
 70 REM SI a FUESE OTRO VALOR, QUEDA IGNORADO !
 80 GOTO 20
 90 PRINT "De un total de ";x+y+z;" votos"
100 PRINT "afirmativos= ";x
110 PRINT "negativos =";y
120 PRINT "abstenciones=";z
```

El que introduzca los datos deberá tener en cuenta que en este caso el dato clave es -5.

El dato clave puede usarse con otra vertiente: si se le plantean al usuario varias opciones a elegir, se decide por una de ellas asignándole un número previmente acordado.

Si por ej., el ordenador te propone elegir un juego, podría preguntar la opción del siguiente modo:

```
n.l.  PRINT "Quieres jugar a barcos (1) o a marcianos (2) ?"
n.l.  INPUT c
n.l.  IF c=1 THEN ...
n.l.  IF c=2 THEN ...
```

El 1 o el 2 hacen de dato clave para la opción elegida. Análogamente, se usa, para volver a ejecutar un programa, la «s» o la «n», para repetirlo o no, respectivamente:

```
n.l.  INPUT "Otra vez? (s/n) "; a $
n.l.  IF a $="s" THEN RUN
n.l.  IF a $="n" THEN STOP
n.l.   GOTO
```

PROGRAMAS

1. Realiza los programas que impriman en columna los números pares y positivos, para cada uno de los casos siguientes:

a) Sin limitación de números a imprimir.

b) De 22 a 222.

c) De 100 a otro valor que el micro conocerá mediante INPUT (este valor podrá ser mayor o menor que 100).

2. Programa que saque los múltiplos de 5, a partir del 30, con cada uno de los siguientes formatos:

a) En línea continua, separados por un guión.

b) Por «zonas» de pantalla.

3. Impresión en pantalla de una progresión aritmética indefinida, cuyo primer término y razón darás con INPUT al micro.

4. Lo mismo que el anterior, pero el ordenador preguntará además el número de términos a imprimir.

5. Lo mismo que 3 y 4, pero la sucesión deberá ser geométrica.

6. Impresión, en la primera «zona» de los veinte primeros términos de una progresión aritmética cualquiera, y en la segunda «zona», a elección del usuario, o bien el cuadrado, o bien la raíz de los términos.

7. Cierta moqueta está a 237 ptas./m^2. Se trata de averiguar el coste de enmoquetar una habitación rectangular cuyas dimensiones se introducen al micro en cm. Si el coste supera las 2.000 ptas., hay un descuento del 12%. Los gastos de transporte importan 1.000 ptas., y la instalación te la cobran a 50 ptas./m^2.

8. Se pretende contar y determinar el porcentaje de un determinado reemplazo militar en el que los soldados tengan una talla inferior a 1,60 y su peso sea superior a 70 kg. Se deben introducir los datos de cada soldado mediante INPUT. Se llamará la atención al operador si se introduce un dato no posible.

9. Introducidos 10 números mediante INPUT, el micro determina y saca en TV el menor y el mayor de ellos. Mejorarlo, no para 10, sino para un número indeterminado.

10. Un teatro dispone de 100 butacas de patio. Se representa «Romeo y Julieta» los días 15, 16 y 17 del mes. La taquilla está informatizada y funciona de la siguiente manera:

Tras pulsar una determinada tecla, te dice cuántas entradas quedan en ese momento para cada día. Solicitas las que quieres, con su fecha, y te dice el importe. Así sucesivamente hasta agotar las entradas, saliendo entonces el cartel de «NO HAY BILLETES». El precio de cada localidad son 2.000 ptas.

11. Programa que te permita ver en TV el presente, pretérito y futuro de un verbo *regular* de la primera declinación, cuando le introduzcas la raíz del verbo a conjugar y el tiempo deseado. Deben figurar los pronombres.

12. En cierta estación de esqui la señorita de la taquilla debe pasar por el teclado las peticiones para que salga impreso en el vale de entrada en pistas, la situación personal de cada esquiador. Las condiciones que figuran en el tablón son:

— una persona «DIA» completo 1.200 ptas.
— una persona «MEDIO» día 700 ptas.
— una persona «BONO» temporada 35.000 ptas.
— una persona «CURSILLO» semana 6.000 ptas.
— grupo de 4 o más, descuento del 10% en todos los casos.

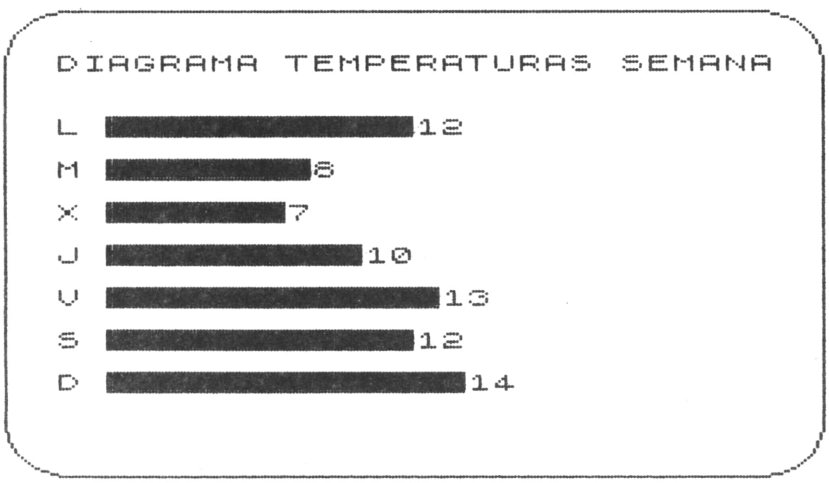
Realizar el programa que saque en TV la situación de cada solicitud.

13. Una vez que el ordenador sepa la aceleración y velocidad inicial de un movimiento rectilíneo, imprime en la primera «zona» el tiempo que transcurre (de seg. en seg.), y en la segunda el espacio recorrido (la aceleración puede ser negativa).

14. En un movimiento circular uniforme se le introduce al micro la velocidad angular. Se trata de saber el ángulo barrido, de seg. en seg., en radianes y en grados.

TEMA 6

ITERACCION MEDIANTE BUCLES

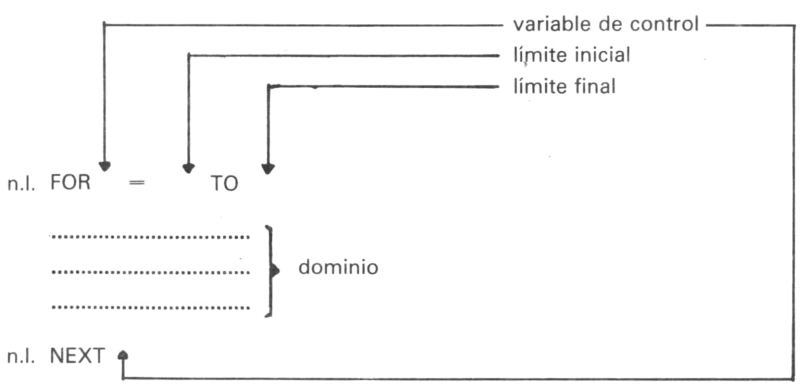


PROGRAMA N.º 19.

SENTENCIA FOR/NEXT

Sirve para que el micro repita un número determinado de veces una serie de instrucciones, a las que llamaremos «dominio» del bucle.

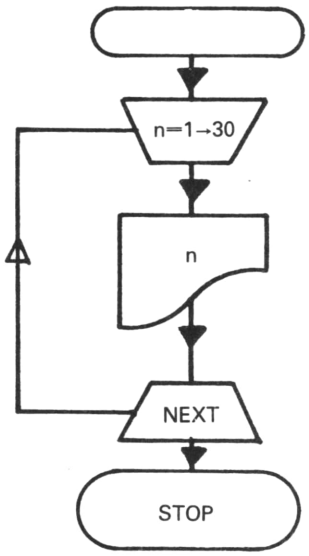
Su formato general es:



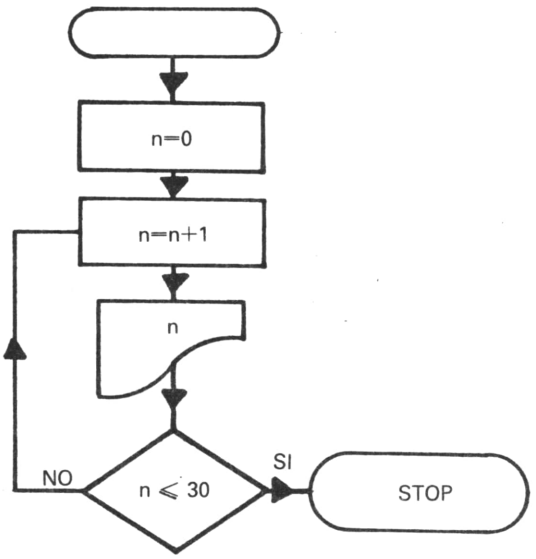
Su significado es: FOR (desde), TO (hasta), NEXT (siguiente)

Hasta aquí, para repetir la ejecución de unas instruccions se habían utilizado contadores y sentencias IF...THEN, en combinación. Pues bien, ahora, sin olvidarnos de ellas, empezaremos a utilizar esta nueva, que tiene, por así decirlo, mayor potencia.

Para comparar, veamos un programa que nos saca en TV los días del mes de mayo:




```
10 FOR n=1 TO 30
20 PRINT n
30 NEXT n
40 STOP
```




```
10 LET n=0
20 LET n=n+1
30 PRINT n
40 IF n < = 30 THEN GOTO 20
50 STOP
```

La sentencia FOR inicializa la variable de control al límite inicial. Se ejecuta una vez el dominio, y llegamos a la sentencia NEXT, que realiza dos tareas:

— Aumenta en 1 la variable de control.
— Compara si se ha sobrepasado el límite final.

Si no se ha sobrepasado, vuelve a ejecutar el dominio desde su primera línea.

En caso contrario, se ha terminado de realizar el bucle y el programa continúa en la línea siguiente al NEXT. Los límites, tanto inicial como final, pueden ser:

— constantes
— variables
— fórmulas

Ejemplos:

```
n.l. FOR h=4 TO z
n.l. FOR t=a-b TO 5 * a+3- b/2
```

Una vez más, te recordamos que el ordenador no puede ejecutar ninguna instrucción sin conocer el valor de las variables que entren en ella.

STEP: PASO POSITIVO, NEGATIVO Y DECIMAL

Hasta aquí hemos visto que la variable del bucle aumenta de uno en uno, pero hay ocasiones que interesa que aumente o disminuya una cantidad determinada. Para ello usamos la intrucción STEP (paso), a continuación del TO del bucle, con la cantidad requerida.

Advertencia: Si usas un paso negativo, el límite final ha de ser inferior al inicial.

STEP 1 NO ES NECESARIO PONERLO

```
10  FOR X=15 TO 1 STEP -3
20 PRINT X
30  NEXT X
```

```
(VISUALIZACION)
15
12
 9
 6
 3
```

```
10  FOR M=.75 TO 4 STEP .75
20  PRINT M
30  NEXT M
```

```
(VISUALIZACION)
0.75
1.5
2.25
3
3.75
```

Observaciones

— La habilidad del programador consiste en que la variable de control pueda ser usada dentro del dominio del bucle. Pero usarla no supone modificar su valor, lo cual puede acarrear problemas.

— Algunos micros no necesitan poner la variable del bucle en NEXT.

— Un bucle sencillo lo puedes poner en una sola línea de programa utilizando (:) para separar las instrucciones.

Vamos a ver lo fácil que es hacer la suma:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{99} + \frac{1}{100}$$

utilizando el siguiente bucle:

```
10  LET s=0:REM inicializa la suma a cero
20  FOR i=1 TO 100
30    LET s=s +1/i : REM va sumando cada fracción.
40  NEXT i
50  PRINT "La suma es: ";s
```

La razón de la línea 20 es que los denominadores van de uno en uno, y del 1 al 100. Entonces lo más lógico es usar un bucle cuya variable de control varíe de esta forma. La 30 va acumulando en la variable s la suma de las sucesivas fracciones. Si la 50 perteneciese al dominio del bucle, te imprimiría cada suma parcial; como aquí está después del bucle, sólo te imprime la suma final.

En el siguiente programa pretendemos que el micro imprima los múltiplos de un número dado con INPUT, y los cuadrados respectivos que no sobrepasen el número 1000:

```
10  INPUT "número? ";n
20  FOR i=n TO 1000 STEP n
30    IF i↑2 > 1000 THEN GOTO 60
40    PRINT i,i↑2
50  NEXT i
60  PRINT "Ejercicio terminado": STOP
```

PROPAGACION DE ERRORES

```
10  FOR N=0 TO 20 STEP P
20  IF N=INT N THEN PRINT N
30  NEXT N

  PARA P=0.1, 0.2 Y OTROS
  ESTE PROGRAMA NO FUNCIONA

ES DEBIDO AL ERROR COMETIDO
EN LA CODIFICACION BINARIA
DE UN NUMERO DECIMAL
```

BUCLES ATIPICOS

Son bucles que, por no ser operativos, debes procurar no ponerlos nunca en un programa. Pueden tener las formas siguientes, entre otras:

```
n.l.  FOR i=a TO a
n.l.    PRINT i+10
n.l.  NEXT i
```

Carece de sentido que los valores inicial y final de la variable de control sean los mismos. ¿Qué haría el ordenador en este caso?

```
n.l.  LET p=0
n.l.  FOR i=I TO 10 STEP p
n.l.    PRINT i
n.l.  NEXT i
```

Si el paso es nulo, el micro no incrementa el valor inicial de la variable control, por lo que el bucle no termina nunca.

```
n.l.  FOR i=10 TO 5 STEP p
n.l.    LET x= x+i
n.l.  NEXT i
```

Asegúrate que el valor de p sea negativo, pues de lo contrario no te realizará el bucle.

Hay un bucle atípico, que llamaremos «bucle en vacío» que sí resulta útil:

```
n.l.  FOR h=1 TO 200 : NEXT h
```

Al no existir aquí dominio del bucle, el ordenador va del FOR al NEXT 200 veces, sin hacer nada, lo que supone un tiempo de espera, que se usa para que el desarrollo de un programa sea más lento. Se puede regular el tiempo variando el valor final de la variable de control.

REGLAS A OBSERVAR EN BUCLES

1. SALIDAS

Se puede salir de un bucle antes de terminado, mediante sentencias adecuadas. La típica es:

```
IF ... THEN (A)

(A) PUEDE SER:
GOTO n.l. (sin vuelta)
GOSUB n.l. (con vuelta)
```

(aunque el bucle quede sin terminar no pasa nada)

2. VUELTA AL BUCLE

Una vez que has salido del bucle, se *puede* volver:

— al FOR, con lo que se vuelve a iniciar el bucle,

— al dominio, con lo que se seguirá ejecutando el bucle donde se dejó,

— al NEXT, con lo que se incrementa la variable de control en el PASO correspondiente, y el bucle continúa por el FOR.

3. ENTRADAS

La forma de ejecutar el dominio es comenzando por la sentencia FOR, que inicializa el bucle, de manera que queda prohibido transferir el control de un programa al dominio de un bucle sin pasar por esta sentencia.

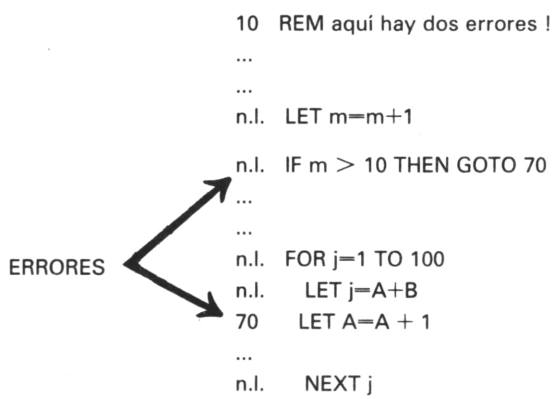
Si lo hicieras, cuando el micro llegase al NEXT, te daría el mensaje de error:

```
NEXT WITHOUT FOR (NEXT sin FOR)
```

Por ejemplo, es válido el siguiente programa:

```
10  LET h=0 : LET d=5
20  FOR x=1 TO 10
30    LET h=2 * h + d
40    PRINT x,h
50    IF h > 50 THEN GOTO 70
60  NEXT x
70  PRINT "FIN DEL EJERCICIO"
```

Pero no sería válido este otro:



BUCLES ANIDADOS

Entre las sentencias que constituyen el dominio de un bucle se puede encontrar la propia FOR ... NEXT pero con variable de control distinta. Esto constituye un «bucle interno».

EL DOMINIO DE UN BUCLE
PUEDE SER OTRO BUCLE!

Como el bucle interno se encuentra en el dominio del externo o principal, se ejecuta íntegramente por cada paso del bucle externo. De aquí que estos bucles tengan una regla adicional:

El primer bucle que se abre (con el FOR) es el último que se cierra (con el NEXT).

Aunque se requiere cierta práctica con los bucles simples para tener soltura con los anidados, verás que éstos son de grandísima utilidad.

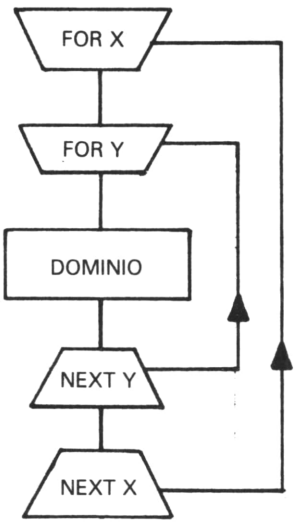
El siguiente programa nos permite ver en pantalla las tablas de multiplicar del 5 y del 6:

```
10  FOR i=5 TO 6
20    PRINT "TABLA DEL ";i
30    FOR j=I TO 10
40      PRINT i; "*";j;"=";i*j
50    NEXT j
60  NEXT i
```

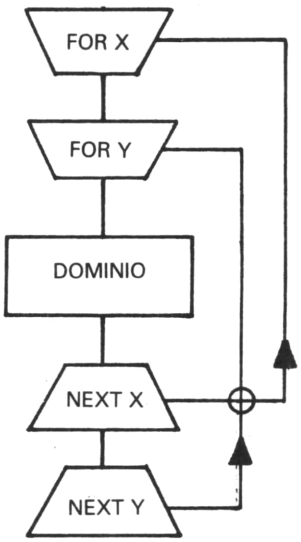
Corre este programa. ¿Te has fijado cómo trabaja? Por cada valor de la variable i (primer factor), se ejecuta el bucle entero de la j (segundo factor).

Observa la diferencia operativa de las líneas 20 y 40: mientras que la 20 se ejecuta solamente dos veces, la 40 lo hace 2 * 10 = 20 veces, por estar en el bucle interno.

BUCLE ANIDADO CORRECTO



BUCLE ANIDADO INCORRECTO



Otro ejemplo de bucle anidado serviría para hacer la suma siguiente:

$$S = \frac{1}{1+2} + \frac{2}{1+2+3} + \frac{3}{1+2+3+4} + \ldots + \frac{10}{1+2+\ldots+10+11}$$

```
10  REM suma con bucle anidado
20  LET s=0:REM inicializa la suma a cero
30  FOR n=1 TO 10
40    LET f=0:REM inicializa cada denominador
50    FOR d=1 TO n+1
60      LET f=f+d
70    NEXT d
80    LET s=s+n/f:REM suma el quebrado
90  NEXT n
100   PRINT "SUMA=";S
```

Observa la diferencia entre las líneas 20 y 40: el LET de la 20, como s tiene la suma total, se pone al principio (fuera de todo bucle); pero el de la 40, como f representa la suma de cada denominador, es preciso inicializarlo a cero para cada fracción, por lo que lo metemos en el bucle principal.

EL STOP COMO SENTENCIA Y COMO COMANDO. END

El STOP como sentencia actúa como la misma palabra indica: el programa se detiene cuando llega a esta orden.

El STOP no significa término de la ejecución de un programa, sino interrupción indefinida. Cuando quieras que prosiga el programa, usas el comando CONT.

Como sentencia STOP tiene utilidad para fraccionar la salida de datos, cuando sean muchos y no fuera suficiente con una pantalla (esto no es aplicable al Spectrum, ya que cuando se termina la pantalla, sale la pregunta «SCROLL»?). Pulsando entonces cualquier tecla, se ve la siguiente pantalla de datos.

Como comando, STOP se utiliza cuando el ordenador está esperando un dato y quieres detener el programa en ese momento. En esta situación no tiene efecto la tecla BREAK.

Hay ordenadores que utilizan como sentencia de final de programa END.

PROGRAMAS

1. ¿Qué imprimirían en TV los siguientes programas?

a)
```
2  FOR x=1 TO 10 : PRINT "x= ";x+3 : NEXT x
```

b)
```
 5  LET a=2
10  FOR b=1 TO 5
15    LET a=a+b
20    PRINT a
25  NEXT b
```

c)
```
10  LET y=20 : FOR z=1 TO 10 : LET y=y + z : NEXT z : PRINT
```

d)
```
40  LET n=2
45  FOR x=1 TO n : PRINT ".";: NEXT x
50  PRINT : LET n=n + 2
55  IF n=10 THEN STOP
60  GOTO 45
```

e)
```
20  FOR h=1 TO 10 STEP 3
25      PRINT h↑2,h↑.5
30  NEXT h: PRINT h
```

f)
```
50  FOR k=1 TO 10 STEP .5
55    PRINT k,k -k↑2
60  NEXT k
```

g)
```
35  FOR x=100 TO 1 STEP -3
45    IF x/2 -10 > 30 THEN PRINT x
50  NEXT x
```

2. Utilizando un bucle, haz que se imprima en TV cinco veces la frase «HOLA AMIGO, HAGAMOS CAMINO», de forma que estén:

a) seguidas,

b) una en cada fila,

c) igual que en b) pero con una línea en blanco entre cada dos frases.

3. Los programas 4, 6, 13 y 14 del capítulo anterior son susceptibles de mejora mediante el FOR/NEXT. Intentarlo.

4. El ordenador te pregunta cuántos alumnos hay en tu clase. Después te pedirá las notas de todos para una asignatura. Mediante un bucle, programa el cálculo de la nota media de la clase.

5. Calcular, mediante bucles, los valores de las siguientes expresiones:

a) $\frac{1}{3} + \frac{1}{3^2} + \frac{1}{3^3} + ... + \frac{1}{3^N}$

b) $\frac{1}{2} + \frac{3}{4} + \frac{5}{6} + \frac{7}{8} + ... + \frac{99}{100}$

c) $\frac{1 \cdot 2}{1+2} + \frac{2 \cdot 3}{2+3} + \frac{3 \cdot 4}{3+4} + ... + \frac{49 \cdot 50}{49+50}$

d) $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + ... + \frac{1}{99} - \frac{1}{100}$

6. Cálculo del factorial de un número natural, menor que 33, que has de dar al micro con INPUT.

7. El desarrollo:

$$1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + ... + \frac{1}{N!}$$

nos proporcionará una aproximación del número e (2,718281...), tanto mejor cuanto mayor sea N. Verifícalo con el micro.

8. Basándote en el programa 6, determina los siguientes números combinatorios:

a) $\binom{10}{5}$ b) $\binom{20}{16}$ c) $\binom{m}{n}$, ambos dados con INPUT.

9. Un señor decide dejar de fumar e invertir las 20.000 ptas. anuales que gastaba en tabaco en una libreta bancaria. Si el banco le da el 5% de interés anual, haz un programa que, en función del número de años dado con INPUT, permita conocer la cantidad que le devolvería el banco al cabo de ese tiempo. Hazlo para cada uno de los siguientes casos:

a) Si sólo deja de fumar el primer año.

b) Si deja de fumar para siempre.

10. Utilizando cualquier carácter que elijas del teclado, programa el dibujo de una barra horizontal de la longitud deseada por el operador:

11. Análogamente al anterior, programa el dibujo de una cruz centrada en la pantalla.

12. El cálculo del seno de un ángulo se puede realizar por el desarrollo de Mac Laurin:

$$\text{sen } x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots \qquad x \text{ en radianes}$$

Este cálculo tiene menos error cuantos más términos cojas. Se da con INPUT el número de términos a procesar y el ordenador realiza la suma.

13. Análogamente al anterior, se tiene:

a) $\log(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \ldots + (-1)^n \frac{x^{n-1}}{n-1}$

b) $(1+x)^y = 1 + \binom{y}{1} x + \binom{y}{2} x^2 + \binom{y}{3} x^3 + \ldots + \binom{y}{n-1} x^{n-1}$

14. En la Edad Media las vacas costaban cuatro doblones, los cerdos dos y las ovejas a 1/3 de doblón. Si un campesino compró 100 animales, con 100 doblones, haz un programa que averigüe cuántos compró de cada especie, si compró uno de cada al menos (hay tres posibles soluciones).

15. Descomponer el número 20 en dos sumandos tales que la suma de siete veces el cuadrado del primero más tres veces el cuadrado del segundo de un valor mínimo.

16. Siendo la suma de los catetos de un triángulo rectángulo 10 cm., hallar el de área máxima.

17. Un jardinero tiene que construir un jardín en forma de sector circular, que ha de tener un perímetro de 150 m. ¿Qué radio y qué ángulo debe tener para que su área sea máxima?

18. Dibujar en pantalla tantos barrotes horizontales como quiera el operador, y de la longuitud que desee. Considerar limitaciones según el micro.

19. Se trata de hacer un diagrama de barras horizontales que indique la temperatura media diaria de una determinada semana. El micro te pregunta las sucesivas temperaturas, y por cada una de ellas te dibujará una barra de longuitud proporcional, y a la derecha de la misma indicará la temperatura correspondiente.

20. Se pretende conocer los valores de las letras x, y, z que cumplen

$$\begin{array}{ccccccc} x & + & y & + & z & = & 14 \\ - & & - & & - & & \\ z & + & x & - & y & = & 6 \\ + & & + & & - & & \\ y & - & z & - & x & = & -6 \\ \| & & \| & & \| & & \\ 8 & & 0 & & -8 & & \end{array}$$

Estos valoresson números naturales, de 0 a 9, pudiéndose repetir. El micro los irá probando mediante los bucles anidados.

21. Un barco B se halla a 375 km. al este de otro barco A. Si B navega hacia el oeste a 20 km/h., y A hacia el sur a 15 km/h, haz el programa que te permita saber, de minuto en minuto, la distancia que los separa.

22. Una persona está a 100 m. de un río, y a 500 m. de un pueblo. El pueblo también dista 500 m. del río. El terreno es llano, y ha de ir al pueblo tras haberse bañado. Haz un programa que te permita saber en qué punto del río debe hacerlo para que el camino recorrido hasta el pueblo sea mínimo.

23. Queremos que salgan las iniciales de los cuatro puntos cardinales, cada uno de ellos en la correspondiente parte de la pantalla:

a) sólo con la función TAB,

b) sólo con la función AT.

24. El mismo programa que el anterior, con la función AT, de manera que las iniciales se vayan superponiendo, girando en sentido del reloj (utiliza un retardo en cada superposición).

25. Se trata de hacer el marco de la pantalla (22 * 32) a base de *un carácter* introducido mediante INPUT:

a) generado en sentido horario,

b) en sentido antihorario.

26. Se pretende llenar la pantalla con el signo + de las siguientes maneras:

a) Visualizado columna a columna, siendo la generación de abajo hacia arriba.

b) Visualizado fila a fila, generándose cada una de derecha a izquierda.

27. Haz los triángulos de números siguientes:

a)
```
1
2 2
3 3 3
...
```

b)
```
      1
    1 2 3
  1 2 3 4 5
...
```

28. Realiza el programa que visione en pantalla el símbolo:

```
      P
      P
      P
      P
    A   Z
  A       Z
A           Z
A               Z
```

29. Representa en pantalla, a gran tamaño y hechas con la misma letra, diversas letras del abecedario, como por ejemplo M,W,X.

30. Análogo al anterior, pero que sean letras del alfabeto griego y formadas por puntos.

31. Dibuja las siguientes banderas:

a) La bandera americana, con sólo 28 estrellas (asteriscos).

b) Una bandera «económica», compuesta por tres franjas verticales de 10 columnas cada una, en la que cada una de estas zonas esté formada por los símbolos del dólar, de la libra y de la peseta (P).

TEMA 7

FUNCIONES NUMERICAS

```
C=copas O=oros B=bastos E=esp.
==============================

CARTAS : E4, B2, C11, C5
         NO HAY JUEGO

CARTAS : C5, E5, O2, B5
         T R I O

CARTAS : O7, B10, E1, C10
         P A R E J A
```

PROGRAMA N.º 25.

FUNCIONES TRIGONOMETRICAS

Al igual que una calculadora (de hecho es una calculadora programable, con pantalla) el ordenador genera el resultado de las siguientes funciones trigonométricas:

SIN (x)
COS (x)
TAN (x)
ASN (m)
ACS (m)
ATN (m)

donde x es el argumento, y ha de ir siempre expresado en radianes (a diferencia de las calculadoras, los ordenadores trabajan en radianes), y m es el argumento, que está limitado de -1 a 1 en los dos primeros casos. Si tuvieras un ángulo x en grados, para pasarlo a radianes basta multiplicarlo por π y dividirlo por 180, ya que 1 rad = 57,3°.

FUNCIONES ALGEBRAICAS

SQR (x): calcula la raíz cuadrada del argumento, que, como sabes, debe ser positivo. Ahora bien, el resultado riguroso tiene los dos signos, pero sólo da el positivo.

EXP (x): calcula la exponencial de base el número e, y x como exponente. Puede ser $x < 0$.

LN (x): calcula el logaritmo neperiano del argumento. Siempre $x > 0$.

Si se pretende calcular logaritmos en cualquier otra base a, se ha de aplicar la fórmula del cambio de base

$$\log_a N = \frac{\text{Ln } N}{\text{Ln } a}$$

DEF FN Y FN

DEF FN, en su forma más simple, tiene el siguiente formato:

```
n.l.  DEF FN y(x) = función de x
```

donde y es el nombre asignado a la función (tiene que ser una letra), x es la variable independiente y función de x es la función a utilizar, escrita en BASIC.

Cuando el ordenador se encuentra con la instrucción FN y(a) siendo a un valor, va a la línea donde se encuentra DEF FN, calcula el valor de la función para a, y se lo asigna a FN.

Supongamos que queremos una tablilla de valores de la función $y = 3x^2 + 7x + 6$, para los valores enteros de x desde -4 a 4. El programa sería:

```
10  DEF FN y(x)=3*x*x+7*x+6
20  FOR x=-4 TO 4
30    PRINT x, FN y(x)
40  NEXT x
```

Observaciones

— DEF FN puede estar en cualquier línea, pero hay micros que obligan a ponerlo al principio.

— Es posible una definición más amplia, con varias variables, como por ej.:

```
DEF FN a(x,y,z) = x↑2+y↑2+z↑2
```

En la llamada con FN, harán falta, asimismo, tres argumentos, para las respectivas variables.

— La determinación aproximada de las soluciones de una ecuación se puede hacer por tanteo del modo siguiente:

a) Se define la función de x.

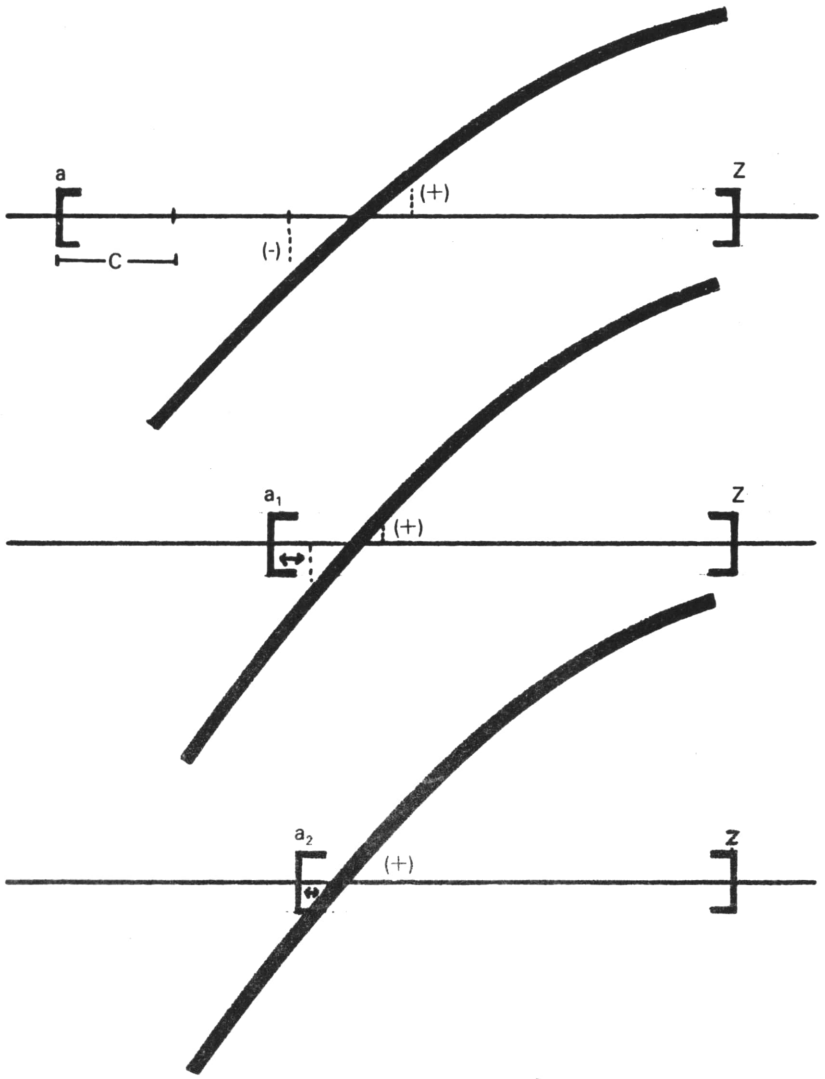
b) Se toma un intervalo suficientemente amplio donde pudieran estar las soluciones.

c) Se construye una tabla de valores, con un bucle de paso suficientemente pequeño.

d) Se observan los cambios de signo de la función, y los valores de x correspondientes al cambio acotan la solución.

e) Para más precisión se puede repetir el paso c, teniendo en cuenta los límites del intervalo acotado.

Observa las figuras siguientes:



EN ESTE METODO, EL LIMITE INFERIOR SE VA ACERCANDO A LA SOLUCION.

EL SUPERIOR ESTA FIJO.

PODRIA HACERSE TAMBIEN AL REVES

OTRAS FUNCIONES

INT (n): toma la parte entera del número n. Ejemplos:

```
INT 5.46 = 5
INT 0.092 = 0
INT -7.2 = -8
```

ABS (n): toma el valor de n, prescindiendo de su signo.

SGN (n): toma los siguientes valores:

```
n > 0 → SGN = 1
n = 0 → SGN = 0
n < 0 → SGN = -1
```

TRUNCAMIENTO Y REDONDEO

Como aplicaciones de la función INT, veremos:
a) Truncamiento de un número a p decimales.
b) Redondeo de un número a p decimales.

a) El ordenador trabaja hasta con 8 decimales, si se le deja. Si de un número N sólo quieres visualizar p decimales, bastará que uses la expresión:

```
INT (N * 10↑p)/ 10↑p
```

Observaciones

— El truncamiento de un número N siempre se efectúa por defecto.
— El número N queda en memoria inalterado, con todos sus decimales.

Ejemplo: si hacemos `LET N= 84.57386` el truncamiento a tres cifras decimales es

```
PRINT INT (N * 1000)/1000
```

imprimiendo 84.573.

b) En el truncamiento el ordenador no tiene en cuenta el valor de la primera cifra decimal ignorada. Según las reglas del redondeo de un número decimal, si ésta es 5 o mayor se debe incrementar en una unidad la última cifra decimal del número truncado. Si queremos entonces p decimales redondeados del número N, se utiliza la expresión:

```
INT (N * 10↑p + 0.5)/ 10↑p
```

En el ejemplo anterior, el redondeo a tres decimales daría el número más preciso de 84.574.

GENERACION DE NUMEROS ALEATORIOS

Números aleatorios son aquellos generados al azar, como por ejemplo tirando un dado.

RND es una función que da un número aleatorio comprendido entre 0 y 1 (exclusive), con 8 decimales. Ejemplo:

```
10  REM CUATRO NUMEROS ALEATORIOS
20  FOR i=I TO 4
30    LET a = RND
40    PRINT a
50  NEXT i
```

Nosotros, cuando lo hicimos, obtuvimos:

```
0.45634608
0.00315042
0.96583 214
0.60031042
```

Seguro que a ti te salen otros valores, debido a que el ordenador los genera al azar.

Supongamos que queremos simular el lanzamiento de un dado. Necesitamos que el micro dé un número entero entre 1 y 6, al azar. Fíjate en la secuencia seguida en la tabla siguiente:

RND		RND * 6	INT (RND * 6)	INT(RND * 6) + 1
0,———————	→	5,———————	0 5	1 6

Para simular 10 lanzamientos del dado, el programa sería:

```
10  FOR i=I TO 10
20    LET a= INT (RND * 6)+1
30    PRINT "lanzamiento ";i,a
40  NEXT i
```

Mediante un procedimiento parecido, se transforma el intervalo de toma de datos aleatorios (0,1) en el intervalo (a,b), con fórmula:

```
RND * (b-a) + a
```

(los límites a y/o b pueden ser negativos, con a < b)

Así, si quisiéramos la impresión de cuatro años, elegidos al azar entre los de este siglo, sería:

```
10  FOR i=I TO 4
20    LET an = INT(RND (1987-1900)) + 1900
30    PRINT an : NEXT i
```

Como último ejemplo, generamos 25 estrellas en la pantalla, repartidas al azar:

```
10  FOR i=I TO 25
20    LET x=RND * 21
30    LET y=RND * 31
40    PRINT AT x,y;"*" : NEXT i
```

PROGRAMAS

1. Obtener una tabla de valores de la función seno desde 0 grados a 360, donde el ángulo vaya incrementándose de 5 en 5 grados. Hacer lo mismo con el coseno y la tangente. Para este último caso debes evitar los ángulos de 90 y 270 grados, pues el infinito no existe para el micro. Sin embargo, deberá indicarlo.

2. Para comprobar la exactitud del micro, haz que salga en la primera zona de la pantalla los valores de la tangente, y en la segunda los valores del seno partido por el coseno. El ángulo barrido será desde -45° a 45°, de uno en uno.

3. Verifica las fórmulas siguientes:

a) $\text{sen}^2 \alpha + \cos^2 \alpha = 1$

b) $\text{tg}^2 \alpha = 1/\cos^2 \alpha - 1$

4. Se demuestra en Matemáticas que se pueden hacer las siguientes aproximaciones, que son tanto mejores cuanto menor sea el ángulo:

a) $\text{sen } \alpha \sim \alpha$,

b) $\text{sen } \alpha \sim \text{tg } \alpha$, donde α debe estar expresado en radianes.

Haz los programas que visualicen los valores de los dos miembros de la aproximación, en cada zona de la pantalla. Después de vistos los valores, el micro deberá imprimir el tanto por ciento de error, para cada ángulo α. El ángulo barrido será de 0 a 10°, con un incremento suficientemente pequeño. Comparar con el programa 12 del cap. anterior.

5. Obtener los límites siguientes, numéricamente

a) $\lim_{x \to \infty} x \text{ sen } \frac{a}{x}$ b) $\lim_{x \to \infty} \frac{x - \text{sen } x}{x + \text{sen } x}$

donde a es un parámetro a introducir por el usuario.

6. Obtener el límite de las sucesiones

a) $\sqrt{a}$, $\sqrt{a\sqrt{a}}$, $\sqrt{a\sqrt{a\sqrt{a}}}$ ...

b) $\sqrt{a}$, $\sqrt{a+\sqrt{a}}$, $\sqrt{a+\sqrt{a+\sqrt{a}}}$ ...

donde a es un número entero, introducido con INPUT.

7. Un campo de fútbol tiene 35 m. del córner al primer palo de la portería. Entre palo y palo hay 8 m. Un jugador se sitúa en la banda, a x metros del córner. Desde ese punto ve la portería bajo un ángulo α. Se trata de que el ordenador calcule α en función de x, o sea, que imprima una tabla con x a la izquierda y α (en grados) a la derecha. Empezar por x = 60, e ir disminuyendo de metro en metro. Se observa que hay una posición óptima para el tiro. ¿Cuál es?

8. Por un procedimiento parecido al anterior, deteminar el primer máximo y el primer mínimo de las funciones:

a) $\text{sen } \alpha + \text{tg } (\alpha/3)$,

b) $10 + \exp(-x/20) * \cos(x/5)$

9. Introducido un número real al ordenador, éste imprime dicho número y su raíz cuadrada, con el doble signo. Prevé la posibilidad de que el número sea negativo, en cuyo caso dará el número imaginario correspondiente.

10. Mediante la función SGN, y con el procedimiento teórico explicado, determina las soluciones de las siguientes ecuaciones:

a) $x^3 - 2x - 5 = 0$ (tiene una solución real y dos imaginarias).
b) $2x - 1n\ x - 4 = 0$
c) $2^x = 4x$
d) $e^x + e^{-x} - 4 = 0$
e) $x\ 1n\ x - 14 = 0$
f) $x^3 - 3x + 1 = 0$
g) $\text{sen}\ x = x - 1$

h) $x^{179} + \dfrac{163}{1 + x^2 + \text{sen}^2\ x} = 119$

i) $x^5 + x + 1 = 0$

11. Mediante el bucle FOR i= 1 TO 89 comprueba alguna de las igualdades siguientes:

1. $\text{tg}\ x + 1/\text{tg}\ x = 2/\text{sen}\ 2x$
2. $\text{tg}\ x/2 = \text{sen}\ x\ /\ (1 + \cos x)$
3. $\text{tg}\ x/2 = 1/\text{sen}\ x - 1/\text{tg}\ x$
4. $(2\ \text{sen}\ x - \text{sen}\ 2x)/\ (2\ \text{sen}\ x + \text{sen}\ 2x) = \text{tg}^2\ x/2$

Observación: Recuerda que el ángulo x debe estar expresado en radianes. En consecuencia, transforma la variable del bucle.

12. Obtén una tabla de valores que te permita deducir los límites:

a) $\dfrac{\sqrt{x+1} - 2}{x - 3}$, cuando $x \to 3$

b) $\dfrac{\sqrt{4 - x} - 2}{e^x - 1}$, cuando $x \to 0$

c) $\dfrac{\sqrt{x+2} - 2}{e^{x-2} - 1}$, cuando $x \to 2$

d) $x\ (5^{1/x} - 1)$, cuando $x \to +\infty$

con aproximación al límite tanto por la derecha como por la izquierda.

Observación:
Si la tabla de valores fuese oscilante, es que te has acercado demasiado al valor del límite; entonces elige para el valor inicial de la variable del bucle un número más alejado.

13. Comprueba, con ayuda de una tabla, los límites siguientes:

a) $\dfrac{1 - \cos x}{x^2}$ $x \to 0$ **(0,5)**

b) $\dfrac{\cos x - \cos a}{a - x}$ $x \to a$ **(sen a)**

(el parámetro a debe ser introducido mediante INPUT).

c) $\dfrac{\text{sen}^3 mx}{\text{sen } nx}$ $x \to 0$ $\left(\dfrac{\mathbf{m}}{\mathbf{n}}\right)$

(los parámetros m, n deben ser introducidos con INPUT).

d) $\dfrac{1 - \cos x}{(e^x - 1)^2}$ $x \to 0$ **(0,5)**

14. Análogamente a los anteriores, obtén los límites siguientes, cuya solución viene expresada en función del número e:

a) $\lim\limits_{x \to \infty} (1 + 1/n)^n$

b) $\lim\limits_{x \to \infty} \left(\dfrac{n^2 + 2}{n^2 - 2}\right)^{n^2}$

c) $\lim\limits_{x \to \infty} \left(\dfrac{n^2 + 1}{n^2 + n + 1}\right)^{2n - 1}$

15. Comprueba que el número e se puede expresar de las formas siguientes, tanto más aproximado cuantos más sumandos tomes:

a) $1 + 1/1! + 1/2! + 1/3! + 1/4! + ...$

b) $\dfrac{1}{1/2! - 1/3! + 1/4! - 1/5! + ...}$

c) $\dfrac{1}{1/2! - 3/4! - 5/6! - 7/8! - ...}$

16. Comprueba que, para 100 números aleatorios, se cumple aproximadamente:
a) que su media es 0,5,
b) que si tomamos con signo negativo la mitad de ellos, la media es 0,
c) que si los tomamos en el intervalo 1.000-2.000, su media es 1.500.

17. Supongamos una moneda equilibrada, la cual vamos a lanzar un determinado número de veces, dado por INPUT. Debe salir en TV los sucesivos lanzamientos (C, cara y + para cruz), y después las frecuencias relativas a cada caso. Se considerará que la moneda está defectuosa si una de las frecuencias fuese menor que el 30% en cuyo caso se repetirán los lanzamientos.

18. Programa por el que el ordenador te pregunte las tablas de multiplicar, de forma salteada, de manera que tras haber pedido 20 productos, el micro te conteste «BIEN» si no tuviste ningún fallo, «REGULAR» si hubo uno o dos, y «MAL» si hubo más de dos.

19. El ordenador genera un número aleatorio, comprendido entre 0 y 1.000, y tú debes descubrirlo. En cada intento, si no aciertas te dice si está por encima o por debajo; cuando finalmente lo aciertes, te dice cuántos intentos has necesitado.

20. Análogo al anterior, pero ahora el ordenador va a jugar contigo. Tras cada intento no acertado, el ordenador moverá el número-respuesta un máximo de 4 unidades, hacia arriba o hacia abajo.

21. Simular la obtención de los tres primeros gordos de la Lotería Nacional (los números son desde el 0 al 59.999). A un mismo número no le puede tocar dos premios.

22. Simular la obtención de los cinco números de una línea de cartón de bingo. Los números son del 1 al 99, sin poder repetirse.

23. Vamos a llamar a la baraja española de la forma siguiente: los palos, C, O, B y E, y los números de cada palo del 1 al 10. Simular la extracción de dos cartas de la baraja.

24. Ampliando el programa anterior, el ordenador examina las cartas. Si son del mismo palo, o de números consecutivos, lo indica.

25. De una baraja española se toman cuatro cartas. Tras la visualización, te indica si ha salido algún trío o pareja.

26. Se pretende que salgan en pantalla 50 estrellitas, de forma aleatoria. El ordenador cuenta el número de estrellas que salieron en un cuadrado de dimensiones 10 x 10, ubicado en el ángulo superior izquierdo de la pantalla.

27. Un dado está «cargado» de forma que la probabilidad de obtener las distintas caras es proporcional a la puntuación de la cara superior. Se lanza tal dado 20 veces. Se imprimen los resultados.

28. En el lanzamiento de dos dados equilibrados y al azar, calcular su suma. Tras hacer esto doscientas veces, el ordenador imprime la frecuencia relativa de las sumas 6,7 y 8. ¿Cuál es la más probable?

29. Una estadística oficial señala que en cierta población hay un 30% de enfermos sólo con gripe, un 20% que tienen sólo catarro, y un 10% con ambas enfermedades. Se realiza una encuesta telefónica entre 20 personas. El ordenador te imprime un posible resultado de la encuesta.

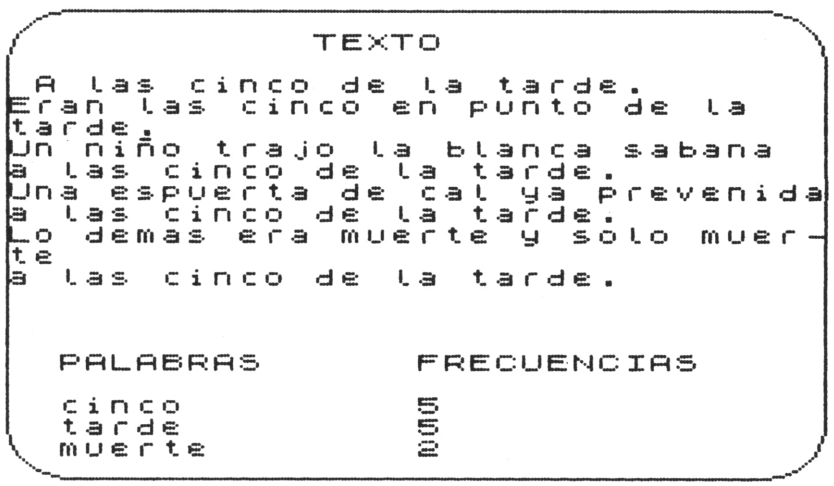
30. Tenemos una urna con dos bolas rojas, tres amarillas y cuatro negras. Simula la extracción de dos bolas en los siguientes casos:

a) con reposición de la primera,

b) sin reposición.

TEMA 8

FUNCIONES CON CADENAS



PROGRAMA N.º 15.

COMPARACION. CONCATENACION

Al igual que en las variables numéricas, podemos hacer comparaciones con las alfanuméricas o cadenas, con los operadores lógicos ya vistos en el Cap. V.

El alfabeto BASIC está formado, por este orden: (espacio), números del 0 al 9, mayúsculas, minúsculas.

El resto de los caracteres no se consideran ahora. Se dice que un carácter es menor que otro cuando está antes en el alfabeto BASIC. Así a < c, pero a > C. Si en la comparación los primeros caracteres son iguales, se prosigue hasta encontrar dos diferentes. Así:

"distinto" es mayor que "distante".
"AB-7" es menor que "AB- 9".

El programa que compara dos cadenas es:

```
10  INPUT a$,b$
20  IF a$ <b$ THEN PRINT a$;" menor que ";b$:GOTO 10
```

```
30 IF a$ >b$ THEN PRINT a$;"mayor que " ;b$:GOTO 10
40 PRINT "son cadenas iguales": GOTO 10
```

Se entiende por concatenación la yuxtaposición de cadenas.Se realiza con el operador +. Prueba como ejemplo:

```
10 REM concatenación de tres cadenas
20 LET a$="hoy sólo has dormido "
30 LET b$=" horas"
40 INPUt "cuántas horas has dormido esta noche? ";c$
50 PRINT a$ + c$ + b$
```

VALOR LOGICO DE UNA COMPARACION

Si una comparación, tanto numérica como alfanumérica es verdadera, el ordenador le asigna el valor lógico 1; en caso contrario, le asigna el 0 (el ordenador no acepta comparar una variable numérica con otra alfanumérica).

Si tenemos, inserta en un programa, la comparación (a>b), con a y b conocidas por el micro, se le asigna a esta expresión su valor lógico. Una aplicación interesante es la de direccionamiento múltiple. Si tenemos, por ejemplo:

```
n.l.  GOTO 200 * (a=b) + 300 * (a > b) + 700 * (a < b)
```

el control del programa irá:

```
a la 200 , si a=b
    300 ,   a > b
    700 ,   a < b
```

Es decir, el micro toma una comparación como una variable de tipo booleano, que sólo puede tomar como valores el 1 o el 0.

LONGITUD DE UNA CADENA

La sentencia LEN a$ calcula el número total de caracteres que tiene la cadena a$ (se incluyen signos de puntuación, espacios, etc.).

Ejemplo:

```
10 LET a$="INFORMATICA"
20 PRINT a$ ;" tiene ";LEN a$ ;" LETRAS"
```

SENTENCIAS: VAL, STR$

Si en una variable alfanumérica los únicos caracteres almacenados son números, ésta se

puede convertir en variable numérica, con la sentencia VAL. (En el Spectrum, si además de números hay otros caracteres en la cadena, la sentencia no funciona).

Ejemplo: Si tenemos a$="1985", podemos construir la variable

```
LET a= VAL a$
```

que adquiere el valor numérico de 1985.

La sentencia STR$ hace justamente lo contrario: convierte un número en una cadena. En el ej.:

```
10  LET d=12
20  LET a$ = STR$ (d)
```

aquí, en la variable d se guarda el valor numérico 12 y en la variable a$ se guarda la cadena «12», por lo que podríamos hacer d=12, pero no tiene sentido a$ = 12.

SENTENCIAS DE DESPIECE DE CADENAS

Si tenemos una cadena almacenada en la variable a$, podemos hacerle los siguientes cortes:

LEFT $ (a$,n): toma de a$ los n caracteres del lado izquierdo.

RIGHT $ (a$,n): toma de a$ los n caracteres del lado derecho.

MID$(a$,p,n): toma de a$ los n caracteres hacia la derecha desde aquel que está en la posición p de la cadena.

TL $ (a$): toma de a$ todos los caracteres excepto el primero.

Por ejemplo, en la cadena a$ = «HOY TENEMOS 18 GRADOS».

Vamos a hacer los siguientes despieces:

```
LEFT$(a$,8) → "HOY TENE"
RIGHT$(a$,6)→ "GRADOS"
MID$(a$,13,1)→ "18 GRA"
TL$(a$)→ "OY TENEMOS 18 GRADOS"
```

Hay ordenadores que no disponen de las sentencias anteriores, pero las suplen con la sentencia TO. El formato es:

```
a$ ( m TO n)
```

donde m y n son, respectivamente, las posiciones inicial y final de la subcadena obtenida:

- Si m=n, se obtiene el carácter de esta posición.
- Si m>n, se obtiene la cadena vacía.
- Si m>LEN a$(n.º caracteres de la cadena), el ordenador da error.
- Si se omite m, se toman los n primeros caracteres.
- Si se omite n, se toma toda la cadena, excepto los (m-1) primeros caracteres.
- Si se quiere los p últimos caracteres hay que poner:

```
a$ ( LEN a$ -p + 1 )
```

Haciendo a$="123456789" tendríamos:

```
a$ (2 TO 5) → "2345"
a$ (TO 5) → "12345"
a$ (7 TO) → "789"
a$ (4 TO 4) → "4"
a$ (5 TO 13) → error
```

Recuerda que, como siempre, m y n pueden ser:
— constantes,
— variables definidas,
— expresiones con variables.

SENTENCIA: INKEY$

Permite la entrada de un carácter y sólo uno, desde el teclado, sin necesidad de pulsar ENTER. Cualquier tecla pulsada se asigna a INKEY$, sólo si cuando pasa el programa por esta sentencia estás pulsando algo. Si no pulsas, se le asigna la cadena vacía.

En cualquier caso, INKEY$ no hace detener el programa, como lo haría INPUT. Por lo tanto, si te resulta necesario, puedes obligar al programa a volver a la misma línea, de forma continuada, como lo hacen las líneas 10 y 30 del siguiente programa. Utiliza el teclado como máquina de escribir, y la pantalla como papel:

```
 5 REM primer ejemplo de INKEY$
10 IF INKEY$ ="" THEN GOTO 10
20 PRINT INKEY$ ;
30 IF INKEY$<> "" THEN GOTO 30
40 GOTO 10
```

Comentario al programa:

línea 10: espera mientras no pulses una tecla, pero una vez pulsada asigna el carácter de la tecla a INKEY$, y va a la 20.

línea 20: imprime el carácter almacenado en INKEY$:
— Si pulsaste un espacio, deja un espacio en blanco.
— Si pulsaste un ENTER, pasa a la línea siguiente (retorno de carro).
— Si pulsaste un carácter equivocado, no es posible la corrección.

línea 30: espera que dejes de pulsar la tecla (prueba a suprimirla).

línea 40: manda repetir el proceso.

También podemos asignar INKEY$ a una variable alfanumérica, para posibles usos posteriores. Así, el programa anterior podíamos haberlo hecho de este modo:

```
 5 REM segundo ejemplo de INKEY$
10 IF INKEY$ ="" THEN GOTO 10
15 LET a$ =INKEY$
20 PRINT a$;
30 IF INKEY$<>"" THEN GOTO 30
40 GOTO 10
```

La versión de INKEY$ en otros micros es GET. Pero, así como en INKEY$ lo pulsado es considerado por el micro como una cadena (ya fuera número, letra u otro carácter), para GET hay dos versiones, una para valores numéricos, y otra para cadenas, que son, respectivamente:

— n.l. GET n
— n.l. GET n$

SENTENCIA: PAUSE

Realiza una pausa en la ejecución del programa. El formato es:

```
n.l.  PAUSE (argumento)
```

donde (argumento) puede ser, como siempre, una constante, una variable conocida u operaciones con variables.

Ej.: n.l. PAUSE 200

detiene el programa 200 unidades de tiempo. La unidad de tiempo utilizada es 1/50 segundos (para América es 1/60). Entonces:

```
200*1/50 = 4 segundos de pausa.
```

Si el argumento es 0, la pausa es indefinida.

NOTA: La pausa queda automáticamente eliminada al pulsar una tecla, que no queda almacenada en memoria.

Si quieres que la pausa sea ineliminable, la sustituyes por un bucle vacío:

```
n.l.  FOR e=1 TO 500:NEXT e
```

También podemos realizar el programa de la máquina de escribir usando el PAUSE 0:

```
10  REM PAUSE con INKEY$
20  PAUSE 0
30  PRINT INKEY$;
40  GOTO  20
```

El objeto del siguiente programa-ejemplo es seleccionar y almacenar en la variable a una de las seis posibilidades de un supuesto menú, que ha salido en pantalla:

```
100  PRINT AT 21,0: "pulsa del 1 al 6"
110  PAUSE 0
120  LET a$ =INKEY$
130  IF a$< "1" OR a$> "6" THEN CLS: GOTO 100
140  LET a=VAL a$
150  GOTO 1000*a
```

Comentario

Con las líneas 110 y 120 la tecla pulsada se almacena en la variable alfanumérica a$. La 130 analiza si la tecla pulsada esta fuera del intervalo 1-6, dentro de la ordenación del alfabeto BASIC, en cuyo caso vuelve a pedir el número, por no ser correcto. Cuando el número está dentro del intervalo pasa a la 140, que transforma a$ en su valor numérico, y se almacena en la variable a. La línea 150 manda el control del programa a aquella parte del mismo que realiza la opción elegida.

PROGRAMAS

1. El micro pregunta tu nombre y lo imprime en el centro de la pantalla, subrayándolo.

2. El micro pregunta tu nombre, y lo escribe dejando un espacio entra cada dos letras.

3. Asigna a 5 variables alfanuméricas las cadenas:
— «el futuro»,
— «la informática»,
— «en»,
— «está»,
— «aquí»,
Imprime, con estas variables, las siguientes frases:
— El futuro está aquí.
— El futuro está en la informática.
— La informática está aquí.

4. El ordenador analiza el INPUT de variable alfanumérica. Si es un número, lo imprime, junto con su cuadrado. Si no, da el mensaje «DATO NO COMPUTABLE».

5. Con la palabra LUISMANUEL haz el programa que imprima:
— los 4 primeros caracteres,
— los 6 últimos,
— los dos del medio,
— el mismo nombre, pero con un espacio entre ambos nombres simples,
— otro nombre, con estos nombres simples en orden inverso.

6. Utilizando las variables numéricas A=1,,B=2,,C=5,,D=8,,E=9, imprimir la fecha 5-1-1985, previa conversión de estas variables numéricas en otras alfanuméricas.

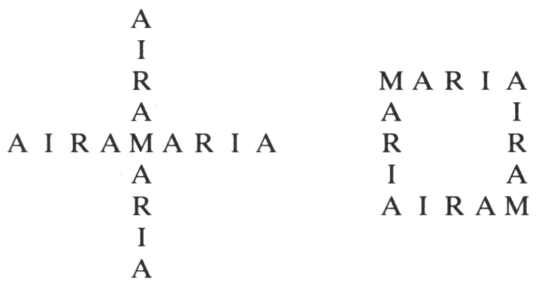
7. Se introduce el infinitivo de un verbo castellano al ordenador. Debe ser un verbo regular y de la 1.ª conjugación. El micro debe extraer la raíz e imprimir los tiempos presente y futuro.

8. Mediante INPUT de cadena, se introduce una fecha (por ejemplo, 25-7-1952). Se pasa a frase, del modo:
25 de julio de 1952.

9. Introducida una cadena, se imprime:
a) de arriba a abajo,
b) de derecha a izquierda.

10. El ordenador pregunta tu nombre. Con él traza:
a) una cruz,
b) un cuadrado.

Si por ejemplo fuese «MARIA», ha de salir, respectivamente:



11. Selecciona un párrafo de una obra literaria. Introducido al ordenador, éste te cuenta:
a) los signos de puntuación (, / . / : / ;),
b) las vocales,
c) espacios,
d) la letra «y».

12. Dada una frase al micro:
a) suprime los espacios en blanco,
b) donde hay un espacio te pone dos.

13. Dada una frase al micro, éste extrae:
a) aquellas palabras que empiecen por mayúscula,
b) los números.

14. Dado un párrafo al micro, éste detecta la existencia de una determinada palabra, que tu introduces con INPUT.

15. Análogo al anterior, pero contando cuántas veces aparece dicha palabra en el texto.

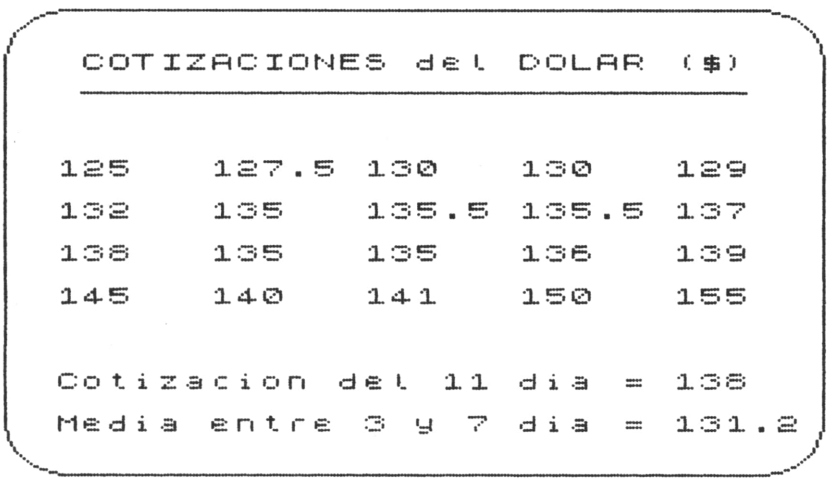
16. Se dibuja en el centro de la pantalla una estrella. Mediante las teclas de cursor muévela en la dirección correspondiente, usando la instrucción INKEY$:
a) deteniéndose en las paredes,
b) rebotando en ellas.

17. Una estrella se mueve con movimiento oscilante en la fila 10 de pantalla. El movimiento ha de ser en columnas consecutivas, y con un retardo, que será un bucle en vacío. El borrado de cada posición puede ser:
a) con CLS,
b) imprimiendo un espacio en blanco.

18. Amplía el programa anterior, de modo que quepan las siguientes posibilidades:
a) parada, al pulsar la «p»,
b) continuación de movimiento, al pulsar la «c»,
c) inversión instantánea del movimiento, al pulsar la «i».

19. Mediante la manipulación del tiempo de espera (bucle de espera) hacer que la estrella acelere (tecla «a») o bien frene (tecla «f»).

TEMA 9
FICHEROS INTERNOS



PROGRAMA N.º 4.

INTRODUCCION

En ocasiones nos encontramos que debemos manejar una inmensa cantidad de datos, que se han de introducir por el teclado con INPUT, o bien asignárselos con la instrucción LET a variables. Esta forma de operar es pesada y poco práctica.

En BASIC disponemos, en principio, de un conjunto de instrucciones que hacen este trabajo más eficiente. Son DATA, READ y RESTORE.

Los lugares donde se almacenan los datos se les llama archivos o ficheros, que pueden ser de dos tipos: internos y externos.

1. FICHERO INTERNO

Es aquel en que los datos se meten en líneas de programa con la instrucción DATA. Es decir, este fichero es parte del programa.

Estos ficheros son adecuados para un volumen pequeño o medio de datos a procesar, pues como estas líneas DATA se almacenan en la memoria RAM, a mayor volumen, menos memoria queda disponible.

2. *FICHERO EXTERNO*

Su soporte físico son cintas o discos (disquetes para los ordenadores personales), donde se puede llegar a almacenar gran cantidad de datos.

A su vez entre los ficheros de este tipo conviene diferenciar los de acceso secuencial y los de acceso directo.

Los secuenciales son bastante más lentos que los directos, pues para acceder a un dato a procesar el sistema operativo tiene que empezar leyendo los datos desde el primero, hasta encontrar el solicitado, mientras que los directos acceden al dato sin pasar por los anteriores.

En el capítulo XIV volveremos a hablar de este tipo de ficheros.

SENTENCIAS: READ, DATA

La sentencia READ tiene el formato:

```
READ (variables)
```

donde (variables) es una sucesión de variables:

- — numéricas,
- — alfanuméricas,
- — combinación de ambas

Ejemplos:

```
READ  a,b,x
READ  c$,h$,x$
READ  a$,b$,m,n,z$
```

Esta instrucción lee los datos de las líneas DATA por orden riguroso, y los asigna a las variables de la sentencia READ, por el orden que figuran.

La sentencia DATA tiene el formato:

```
DATA ( datos )
```

donde (datos) es una sucesión de números o cadenas (los datos de cadena, entre comillas).

Un ejemplo de líneas DATA que se corresponden con los ejemplos de READ son:

```
DATA  10,234,0
DATA  "PEDRO"," Alicia","Margarita"
DATA  "Gil","César",25,7172737,"C/Gran Vía,11"
```

Las líneas DATA se pueden colocar en cualquier parte del programa, ya que no son ejecutables. El ordenador sólo las utiliza cuando ejecuta una sentencia READ. Conviene agruparlas, por claridad, bien al principio o al final del programa.

Observaciones

- — Las líneas DATA deben contener, como mínimo, tantos datos como deba leer el pro-

grama (si hay más no pasa nada, no se leen). Si hay menos, se detiene el programa con el mensaje OUT OF DATA

— Los datos en las líneas DATA tienen que tener una estructura *idéntica* a la de lectura en la sentencia READ. Fíjate para ello en el tercer ejemplo de los READ y DATA anteriores. Así, lee y asigna:

```
a  a$ → "Gil"
a  b$ → "César"
a  m  → 25
a  n  → 7172737
a  z$ → "C/Gran Vía,11"
```

— Es interesante a veces el uso de una sola variable de lectura, mediante un bucle. Si, por ejemplo, se trata de leer y procesar 10 datos, sería:

```
  10  FOR n=1 TO 10
  20    READ a : PRINT a
  30    ......
  40    ......
  50  NEXT n
1000  DATA 1,48,32,3,79,2,11,0.7,- 4,2.3
```

En el ejemplo siguiente, el primer dato a leer, que se almacena en la variable n, nos dice precisamente el número de datos a leer. Con ellos se hará la suma.

```
  10  READ n
  20  LET s=0
  30  FOR i=I TO n
  40    READ x
  50    LET s=s+x
  60  NEXT i
  70  PRINT "SUMA=";s
 500  DATA 6
 510  DATA 4,50,8,- 12,0,2.5
```

SENTENCIA: RESTORE

Cada dato leído de una línea DATA no se puede volver a leer. Es como si se hubiese quedado inutilizado. Sin embargo, hay ocasiones en que necesitamos volver a hacer una relectura de datos. Para ello disponemos de la sentencia RESTORE, que hace utilizables de nuevo todos los datos, para una posterior orden READ. Basta ponerla en el lugar adecuado de programa.

Una variante de esta sentencia es

```
n.l.  RESTORE m
```

donde m es el número de línea DATA donde queremos empezar la lectura.

Con esta variante podemos hacer que el ordenador lea los datos de una línea dada, directamente, sin tener que haber leído los anteriores. Es como una sentencia GOTO, aplicable sólo a la sentencia READ de lectura de datos.

Un ejemplo del uso de esta sentencia es:

```
 10  REM lectura en la línea adecuada
 20  PRINT "TEMPERATURAS DE UNA SEMANA": PRINT
 30  INPUT "semana ? (1 -4 ) ";s
 40  IF s < 1 OR s > 4 THEN GOTO 30
 50  RESTORE 100 * s
 60  FOR i=I TO 7
 70    READ t
 80    PRINT "día ";i,t;" grados"
 90  NEXT i: GOTO 10
100  DATA 18,17,19,17,20,153,14
200  DATA  19,19,17,16,17,15,14
300  DATA  17,17,18,19,19,16,17
400  DATA  15,16,17,18,18,17,19
```

PROGRAMAS

1. Imagina que estamos en un observatorio metereológico en el que te encuentras encargado de recoger y procesar los datos de pluviometría. En el mes de mayo se obtuvieron los siguientes datos:

1.ª semana: 7,5,0,0,8,10,12.
2.ª semana: 12,13,2,0,0,0,0.7.
3.ª semana: 1,1.5,1.8,2,3,5,2.
4.ª semana: 4,10,15,15,7,7,5.

Se pretende realizar el programa que permita obtener la pluviosidad media de la primera semana, de las dos primeras, de las tres primeras y de las cuatro.

2. Con los mismos datos del anterior, obtener la pluviosidad media de la última semana, de las dos últimas, de las tres últimas y de las cuatro.

3. Con los mismos datos del anterior, realiza un programa que dé la media de dos semanas cualesquiera, a elección del usuario.

4. En los veinte días bursátiles de cierto mes, la cotización del dólar estuvo a 125,127.5,130,130,129,132,135,135.5,135.5,137,138,135,135,136,139 ,145,140,141,150,155, en pesetas.

Realiza un programa que permita las dos cosas siguientes:

a) Conocer la cotización de determinado día.

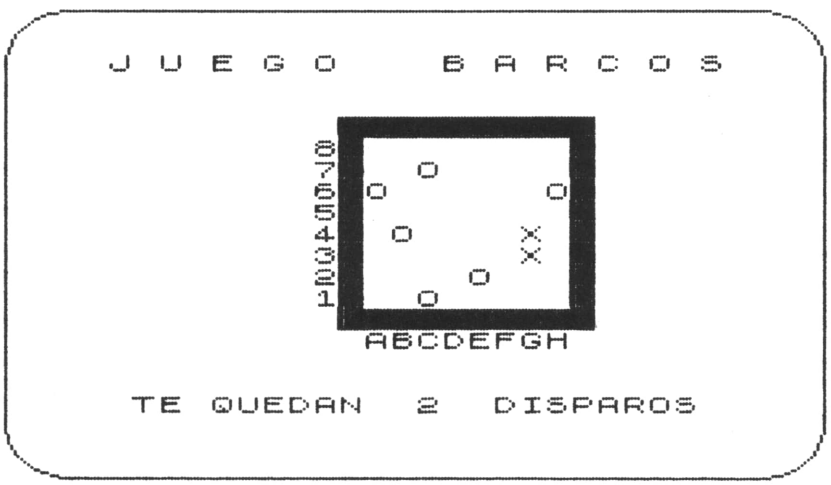
b) Obtener la media de las cotizaciones entre dos fechas (inclusive) que son introducidas por el usuario.

5. Se da la distribución del número de hermanos que tiene cada uno de los 33 alumnos de una clase. Calcular su desviación típica:

3 alumnos no tienen ningún hermano	10 alumnos tienen 3 hermanos
4 alumnos tienen 1 hermano	8 alumnos tienen 4 hermanos
7 alumnos tienen 2 hermanos	1 alumno tiene 5 hermanos

TEMA 10

MATRICES



PROGRAMA N.º 10

DEFINICION Y TIPOS. VARIABLES DE INDICE

Llamamos matriz a una agrupación de datos donde se conserve el orden de éstos. Estos datos pueden ser numéricos o alfanuméricos. Según sea la estructura de esta agrupación, se clasifican en:

- Monodimensionales, cuando los datos están distribuidos en línea.
- Bidimensionales, cuando están distribuidos en una red de filas y columnas.
- Tridimensionales, formados por filas, columnas y una tercera dimensión de fondo.
- Multidimensionales.

Un ejemplo de matriz monodimensional sería:

2 3 7 5

El ordenador almacena los cinco datos en sucesivas posiciones de la memoria. Cada uno de ellos es asignado a lo que llamaremos una variable de índice, siendo la matriz el conjunto de todas ellas.

CADA NUMERO OCUPA
5 BYTES DE MEMORIA

SENTENCIA: DIM

Con esta sentencia le mandamos al ordenador que reserve en memoria tanto espacio para los datos como se necesite. Estos espacios de memoria están inicialmente, es decir, cuando se ejecuta la instrucción DIM:

— con ceros, para matrices numéricas,
— blancos, para las alfanuméricas.

Y así continúan, hasta que son llenados con los datos, siguiendo los procedimientos que veremos a continuación.

MATRICES MONODIMENSIONALES NUMERICAS

El dimensionado de este tipo de matrices sigue el formato siguiente:

```
n.l.  DIM z(n)
```

donde z es el nombre asignado a la matriz y n es el número de posiciones de memoria que reserva el ordenador.

Para el ejemplo anterior hubiera sido necesario:

```
DIM z(4)
```

Una vez dimensionada, se procede a llenar la matriz. Lo más conveniente es usar un bucle para el llenado, a través de cada variable de índice. Estas variables tienen las características propias de todo tipo de variables. Su formato es z(i), refiriéndose a la variable de índice que ocupa el lugar i, en la matriz.

Los diversos procedimientos de carga de la matriz son:

a) Con LET.

Se le asigna a cada variable de índice el resultado de cierta operación:

```
n.l. DIM  a(12)
n.l. FOR  i=1 TO 12
n.l.      LET  a(i)=i*i-1
n.l. NEXT  i
```

En este bucle, al aumentar la variable del bucle, se van llenando las variables de índice (o sea la matriz), con la expresión descrita.

b) Con INPUT.

Las variables de índice se llenan con los números introducidos por teclado:

```
n.l. DIM  a(12)
n.l. FOR  i=1 TO 12
n.l.      INPUT "introduce número";a(i)
n.l. NEXT  i
```

c) Con READ.

Con esta instrucción, cada dato leído se introduce en la matriz. En este caso sería:

```
n.l. DIM  a(12)
n.l. FOR  i=1 TO 12
n.l.      READ a(i)
n.l. NEXT i
n.l. DATA   0,3,8,15,24,35,48,63,80,99,120,143
```

Una vez llenada la matriz, los datos están ahí, para su uso posterior. Si sólo queremos visualizarlos, se hace con un bucle análogo:

```
n.l. REM  visualización de la matriz
n.l. FOR i=1 TO 12
n.l.      PRINT "dato ";i,a(i)
n.l. NEXT i
```

Ejemplo:

Vamos a usar una matriz para almacenar y procesar (hacer la media) las temperaturas de una semana:

```
 10  REM suma de una matriz
 20  LET s=0
 30  DIM t(7)
 40  FOR i=1 TO 7
 50    READ t(i):NEXT i
 55  REM bucle de suma
 60  FOR i=1 TO 7
 70    LET s=s + t(i) : NEXT i
 80  REM visualización
 90  PRINT "TEMPERATURAS :"
100  FOR i=1 TO 12
110    PRINT t(i) :NEXT i
120  PRINT "La media es ";s/7
900  DATA 5,0,-2,0,-3.5,1,4
```

Intenta hacer este programa, pero con sólo un bucle.

Observaciones

— El número n, que indica cuántas posiciones de memoria se reservan, puede ser una variable.

— No se puede utilizar una variable de índice con un índice superior al de dimensionado.

— Como habrás observado en los bucles anteriores, el índice empieza en 1, aunque hay ordenadores cuyo índice empieza en 0.

— Como el índice puede ser el resultado de una fórmula, si diese un número decimal, el micro te lo redondea.

— No es necesario dimensionar una matriz si el número de variables de índice no supera 10.

— Es posible y compatible el uso de una variable que tenga el mismo nombre que la matriz (en el último ejemplo, se puede usar una variable llamada t, pues el ordenador distingue t de t(i)).

MATRICES BIDIMENSIONALES NUMERICAS

Recuerda el juego de los barcos, donde cada casilla está determinada por una fila y una columna. Si en cada casilla (celda de memoria) metiésemos un número, tendríamos una matriz bidimensional.

Una matriz con datos, de tres filas y cuatro columnas por ejemplo:

```
 7    1    -5    4      →    1.ª fila
 2    3     9   -6      →    2.ª fila
-1    0    24    7      →    3.ª fila

 ↓                      ↓

col 1                 col 4
```

El formato del dimensionado es:

```
n.l.  DIM y(f,c)
```

donde y es el nombre asignado a la matriz, f es el número de filas a reservar y c es el número de columnas a reservar.

En el ejemplo anterior, sería:

```
DIM y(3,4)
```

Estas matrices consumen mucha memoria, puesto que se reservan f×c variables.

El llenado de este tipo de matrices procede a hacerse con un bucle anidado, de manera que un bucle vaya barriendo las filas, y otro las columnas. Se puede hacer también con LET,INPUT o READ.

Vamos a ver el llenado de la matriz anterior:

```
10  DIM t(3,4)
15  REM llenado de cada fila
20  FOR f=1 TO 3
25    REM llenado de cada columna
30    FOR c=1 TO 4
40      PRINT "FILA ";f;" COLUMNA ";c,
50      INPUT t(f,c) : PRINT t(f,c)
60    NEXT c
70  NEXT f
```

Este bucle anidado carga la matriz fila a fila, es decir, que para un valor fijo de f(fila), se cargan las columnas.

Los datos están ahora correctamente guardados (si no te has equivocado en la introducción)

en su fila y columna adecuadas. Ahora bien, si intentamos visualizarlos con el siguiente programa:

```
100 FOR f=1 TO 3
110   FOR c=1 TO 4
120     PRINT t(f,c)
130   NEXT c
140 NEXT f
```

aparecerían los 12 datos en columna, no viéndose por tanto la matriz bidimensional.

Para poder visualizarla correctamente, debemos auxiliarnos de las sentencias de posicionado: TAB o AT.

Usando el TAB, con una separación entre columnas de cinco en cinco, sería:

```
 95 REM impresión de cada fila
100 FOR f=1 TO 3
105   REM impresión de cada columna en la fila
110   FOR c=1 TO 4
120     PRINT TAB(5*c);t(f,c)
130   NEXT c
140 PRINT:PRINT
150 NEXT f
```

Por la línea 120, los sucesivos datos se imprimen en la columna adecuada. Al terminar ésta, por la línea 140 se dejan dos líneas en blanco antes de imprimir la fila siguiente.

Ahora que hemos visto la visualización de la matriz, intenta su llenado con las instrucciones READ-DATA. Asimismo, intenta la impresión con la sentencia AT.

Dependiendo del tipo de matriz que sea, puede suceder que no entre en pantalla. Te sugerimos el troceado.

Con la matriz del ejemplo, vamos a hacer unos procesos sencillos:

1. LA SUMA DE CADA COLUMNA

El programa adecuado sería:

```
200 FOR c=1 TO 4
210   LET s=0
220   FOR f=1 TO 3
230     LET s=s + t(f,c)
240   NEXT f
250 PRINT "columna ";c,s
260  PRINT
270 NEXT c
```

2. LA SUMA DE TODOS LOS COMPONENTES DE LA MATRIZ

```
300  LET s=0
310  FOR c=1 TO 4
320    FOR f=1 TO 3
330      LET s=s + t(f,c)
340    NEXT f
350  NEXT c
360  PRINT "suma total=",s
```

3. IMPRESION DE UNA FILA ELEGIDA POR EL USUARIO

```
400  INPUT "fila deseada? ";f
410  IF f<1 OR f>3 THEN GOTO 400
420  CLS
430  FOR c=1 TO 4
440    PRINT TAB (5*c);t(f,c)
450  NEXT c
```

4. ALTERACION DEL CONTENIDO DE UNA DETERMINADA CASILLA

```
500  INPUT "casilla? (fila y columna) ";f,c
510  INPUT "nuevo dato? ";d
520  LET t(f,c)=d
530  INPUT "ya está. Algún cambio más? (s/n)";a$
540  IF a$ ="s" THEN GOTO 500
```

MATRICES TRIDIMENSIONALES NUMERICAS

Recuerda el cubo de Rubik, donde podemos localizar cada cubito elemental con tres coordenadas: fila, columna, y una tercera de fondo. Si cada cubito elemental(celda de memoria) contiene un dato, tendríamos una matriz tridimensional numérica. En este caso sería de dimensiones 3*3*3, pero, como es lógico, pueden ser otras.

El formato de dimensionado sería:

```
DIM M(a,b,c)
```

donde a, b y c representan las tres dimensiones.

El número de celdas de memoria que reserva el ordenador es a*b*c.

El llenado debe hacerse con tres instrucciones FOR -NEXT anidadas.

Como la visualización sólo es posible, como máximo, en dos dimensiones, sugerimos hacerla en «hojas» de pantalla.

MATRICES MULTIDIMENSIONALES NUMERICAS

Aumentando en una dimensión las matrices anteriores llegaríamos a las tetradimensionales, difíciles de imaginar. Por extensión, añadiendo nuevas dimensiones, tenemos las matrices n-dimensionales.

Una aplicación de interés es la ordenación de números. El siguiente programa ordena los datos, introducidos por teclado, en orden creciente:

```
 10 INPUT "cuántos números vas a ordenar?";n
 20 DIM l(n):DIM z(n)
 30 FOR i=1 TO n:INPUT z(i)
 40   LET l(i)=z(i):NEXT i
 50 FOR i=1 TO n-1
 60   FOR j=1 TO n-i
 70     IF l(j)<l(j+1) THEN GOTO 110
 80     LET x=l(j)
 90     LET l(j)=l(j+1)
100     LET l(j+1)=x
110   NEXT j
120 NEXT i
130 PRINT "NUMERO";TAB 10;"CRECIENDO";TAB 20;
        "DECRECIENDO" : PRINT
140 FOR i=1 TO n
150   PRINT z(i);TAB 14;l(i);TAB 24;l(n+1-i)
160 NEXT i
```

En la línea 30 se introducen los datos, llenándose la matriz z. En la 40 se hace un duplicado de la matriz l, que es la que se va a ordenar.

El procedimiento seguido es: dada una pareja de números, si están ordenados se dejan así, y si no lo están, se invierten sus posiciones en la matriz l. Esta inversión, cuando procede, se realiza en las líneas 80, 90 y 100, utilizando una variable auxiliar, x. La primera vez que se termina el bucle de la j, el número más grande está colocado al final de la lista. La segunda vez, el número más grande de los restantes está colocado en el penúltimo lugar. Y así sucesivamente.

Sean los números introducidos: 2-8-6-5-3-1, por este orden. Los sucesivos pasos realizados por el ordenador son:

```
PARA VARIABLE i=1
   PARA j=1      2 8
        j=2      2 6 8
        j=3      2 6 5 8
        j=4      2 6 5 3 8
        j=5      2 6 5 3 1 (8)

PARA VARIABLE i=2
   PARA j=1      2 6
        j=2      2 5 6
        j=3      2 5 3 6
        j=4      2 5 3 1 (6)
```

```
PARA VARIABLE i=3
    PARA j=1      2 5
          j=2     2 3 5
          j=3     2 3 1 (5)

PARA VARIABLE i=4
    PARA j=1      2 3
          j=2     2 1 (3)

PARA VARIABLE i=5
    PARA j=1      1 (2)
```

quedando finalmente la matriz l así:

```
1-2-3-5-6-8
```

Por el bucle de la 140 se imprimen la matriz z, que es la original, la matriz l de principio a fin, y la matriz l del fin al principio (quedando así también por orden decreciente).

MATRICES ALFANUMERICAS

Si necesitamos almacenar y procesar una lista de cadenas usaremos matrices alfanuméricas. Su tratamiento es análogo al de las numéricas, excepto en su dimensionado.

Este tipo de matrices también puede ser bidimensionales, tridimensionales...

DIMENSIONADO DE MATRICES ALFANUMERICAS

Cada carácter de una cadena ocupa una posición de memoria. Por lo tanto, el ordenador necesita saber, no solamente el número de cadenas a almacenar, sino también la longitud. Esto se pone de manifiesto en el formato de dimensionado, que es:

```
n.l.  DIM a$ (n,l)
```

donde n es el número de cadenas y l es el número de posiciones de memoria a reservar por cada cadena.

El ordenador, cuando pasa por esta instrucción reserva un total de n×l posiciones de memoria, inicializándolas todas con el carácter en blanco.

CADA CARACTER OCUPA
1 BYTE DE MEMORIA!

Ahora podemos llenar la matriz con cadenas, con la ayuda de un bucle. Luego, si es preciso, vendría un tratamiento de la matriz.

Ahora bien, cada vez que te refieras a una cadena de la lista sólo debes especificar la posición que ocupa (entre paréntesis), sin indicar la longitud. Por ejemplo, para leer los días de la semana y almacenarlos en la matriz S$, sería:

```
 10   DIM S$(7,9)
 20  FOR i=1 TO 7
 30    READ S$(i)
 40    PRINT S$(i)
 50  NEXT i
700  DATA "lunes", "martes", "miércoles", "jueves",
          "viernes", "sábado", "domingo"
```

Si en el dimensionado hubiésemos puesto:

```
DIM S$ (7)
```

el ordenador sólo guardaría el primer carácter de cada cadena.

Como lo normal es que la longitud de las cadenas sea diferente, hay que procurar que el dimensionado cubra la de mayor longitud. Ahora bien, como esto supone que habrá espacios de memoria que no se van a llenar (quedando inutilizados), se debe hacer una optimización en el dimensionado. Esto supone cortes en las cadenas más largas, pero compensa.

En el ejemplo anterior, con un dimensionado S$(7,9), la matriz alfanumérica queda:

l	u	n	e	s				
m	a	r	t	e	s			
m	i	e	r	c	o	l	e	s
j	u	e	v	e	s			
v	i	e	r	n	e	s		
s	a	b	a	d	o			
d	o	m	i	n	g	o		

Mientras que con un dimensionado DIM S$(7,3), queda:

l	u	n
m	a	r
m	i	e
j	u	e
v	i	e
s	a	b
d	o	m

Si imaginamos que las cadenas están agrupadas en filas y columnas, tendríamos una matriz alfanumérica bidimensional. Habría que dimensionarla con el formato siguiente:

```
n.l.  DIM B$(m,n,l)
```

donde m es el número de filas, n es el número de columnas y l es la longitud reservada a cada cadena.

Así, si nos imaginamos que hay una persona encargada por cada día de la semana de determinada tarea, y cuyos nombres son:

```
JUAN
ANDRES
ALEJANDRO
HORACIO
FRANCISCO
LUIS
ANTONIO
```

el dimensionado para almacenar días y nombres sería:

```
DIM S$(7,2,9)
```

reservando 9 posiciones para cada cadena.

La secretaría de un Instituto recibe una lista con los nombres de diez alumnos y su correspondiente número de matrícula. Necesita un programa que ordene alfabéticamente estos nombres, y los imprima con su número de matrícula. También debe poner el número de orden.

El programa podría ser el siguiente:

```
 10  DIM A$(10,17):DIM M(10)
 20  FOR i=1 TO 10:READ A$(i):READ M(i):NEXT i
 30  FOR j=1 TO 9
 40    FOR k=j+1 TO 10
 50      IF A$(j<A$(k) THEN GOTO 120
 55      REM        ordenación de cadenas
 60      X = A$(j)
 70      A$(j)=A$(k)
 80      A$(k)=X
 85      REM ordenación de números
 90        c=M(j)
100      M(j)=M(k)
110      M(k)=c
120    NEXT k
130  NEXT j
135  REM bucle de impresión
140  PRINT "N" ; TAB 4;"matrícula";TAB 22;"nombre"
150  FOR n=1 TO 32:PRINT "-";: NEXT n
160  FOR i=1 TO 10
170    PRINT i;TAB 6;M(i);TAB 14;A$(i)
180    PRINT : NEXT i
1000 DATA "Peláez,Alicia", 243,"Gómez,Juan",58,"Fernández,Luis",125,"Jiménez,Andrés",83
1050 DATA "Corrales,Ana",723,"Rey,Josefa",2,"Ruiz,Antonia",321, "Jaro,Pedro",579,"Delgado,Beatriz",
424,"Muñoz,Mercedes",624
```

Comentarios

El dimensionado se ha tomado para un máximo de 17 caracteres/nombre. Después de la lectura, empieza la ordenación en la línea 30, por un procedimiento análogo al visto anteriormente. Esta ordenación coloca la cadena menor al principio de la matriz A$, y así sucesivamente. Fíjate que al mismo tiempo que se va ordenando la matriz A se va ordenando la M.

Las líneas DATA están formadas por parejas nombre -matrícula, ya que éste es el formato del bucle de lectura.

Te proponemos que mejores el programa, de forma que admita un número N de alumnos. A su vez, esto se puede hacer de dos formas:

a) con N dado por INPUT,
b) con una lectura previa de datos, para averiguar cuántos hay.

PROGRAMAS

1. Introduce en una matriz los precios de cinco cereales a lo largo de tres años consecutivos. Realiza el programa que saque una tabla con el nombre de estos productos, sus respectivos precios, y la media durante ese período.

2. El ordenador genera y almacena 100 números de 1.000 al 7.000. Después de visualizarlos, te dice al mayor de ellos, el menor y la posición que ocupan en la lista.

3. Hallar la desviación típica de los 100 números de la tabla anterior.

4. El ordenador genera 50 números del -500 al +500. Te ha de contar cuántos han salido negativos, y cuántos positivos.

5. El ordenador genera ceros y unos, en total de ellos 125. La probabilidad de generar un cero ha de ser 0,4 y la de generar uno ha de ser 0,6. Calcula la frecuencia relativa de ceros y de unos, que se debe aproximar a las respectivas probabilidades.

6. El ordenador genera y almacena las temperaturas medias a lo largo de un año. Deberán estar comprendidas ente 10° bajo cero y 30°. Después te imprime cuántos días han estado con una temperatura media:

— entre -10 y 0,
— entre 0 y 10,
— entre 10 y 20,
— entre 20 y 30.

7. Se introducen por teclado dos matrices de dimensiones 3*3.

El ordenador:

a) las suma,
b) las resta,
c) las multiplica.

8. En las matrices anteriores, hallar sus determinantes.

9. El ordenador de una revista del motor tiene almacenados un determinado número de marcas, modelos y sus precios respectivos. Se pretende hacer un programa que permita responder a las siguientes preguntas:

a) marca y modelo del más caro,
b) marca y modelo del más barato,
c) precio medio de determinada marca,
d) modelos de una determinada marca,
e) modelos cuyo precio está entre dos límites.

10. Considera un tablero de 8*8 casillas, donde el ordenador sitúa aleatoriamente un barco de 4 casillas. Tienes 10 disparos para intentar hundirlo. El ordenador te dibuja en la posición de disparo una X si has acertado, o un 0 si has hecho agua. El disparo se especifica con las coordenadas A-H y 1-8. Cada vez que hagas un impacto, se te concede un disparo más. Si no logras hundir el barco, el ordenador te ofrece volver a intentarlo.

11. Vamos a simular un juego de bingo para dos jugadores. El ordenador pregunta el nombre de las dos personas y genera para cada una de ellas su respectivo cartón (15 números sin repetir, del 1 al 99). Asigna a cada jugador una zona de la pantalla, figurando en ella el nombre y números del cartón de cada jugador (en columna). Se simula la extracción de las bolas. Cada número acertado, deberá ser tachado en pantalla. El nombre del ganador es indicado claramente.

12. Se tiran dos dados 80 veces. El ordenador te va dando los resultados, y al final de las tiradas cuenta el número de veces que ha salido cada suma (de 2 a 12).

13. Amplía el programa anterior, de forma que salga el correspondiente diagrama de barras de la suma de dados.

14. Un juego de dados consiste en lo siguiente: el jugador tiene derecho a seguir tirando el dado mientras obtenga una puntuación igual o superior a la precedente. En otro caso, pasa el dado al siguiente jugador. Gana el jugador que obtenga mayor suma de las puntuaciones.

El ordenador pedirá número y nombre de jugadores. Después pide al primero de ellos que pulse una tecla, que simulará un lanzamiento. Sale en pantalla el resultado, y, cuando finaliza este

jugador, su suma total. Se repite el proceso con el jugador número 2, y así sucesivamente. Al final aparece el nombre del ganador.

15. Imagina que eres el último apostante en una carrera de seis caballos. El ordenador lee el nombre de éstos de una línea DATA, y asigna aleatoriamente a cada uno de ellos la cantidad apostada hasta ese momento, comprendida entre 5.000 y 100.000 pesetas. Imprime esto en pantalla y te pregunta tu caballo ganador y cuánto apuestas. Contigo se cierran las apuestas.

Se genera con igual probabilidad el caballo ganador, y si aciertas se te da el premio con el siguiente criterio: se divide todo lo apostado entre la cantidad total apostada al caballo ganador. El 70% de esta cantidad se multiplica por lo que apostaste, lo cual ya da la cantidad a recibir.

16. Análogo al anterior, en donde la probabilidad de ganar cada caballo viene dada por un pronóstico, que se introduce con INPUT para cada caballo (la suma de todas estas probabilidades debe dar la unidad).

17. Programa el juego de las siete y media entre tú y el ordenador.

18. Programa la generación y almacenaje de las 40 cartas de la baraja española, ordenadas.

19. En el program anterior, el ordenador elige cinco cartas. Analiza si hay pareja (parejas), trío o poker.

20. Tú juegas contra el ordenador a la carta más alta (ambas cartas son generadas por él).

21. Análogo al 17, pero las cartas deben almacenarse desordenadas.

22. Vamos a combinar los dos programasa anteriores de la forma siguiente: el ordenador te pide un número del 1 al 40, lo que va a equivaler a sacar una carta del mazo desordenado. Te muestra la carta correspondiente, y luego elige otra el ordenador. Gana la carta más alta.

23. El ordenador reparte seis cartas a tantos jugadores como desees.

24. En el programa anterior, el ordenador suma la puntuación obtenida por cada jugador, con el baremo siguiente:

AS=11 „ TRES=10 „ REY=4 „ CABALLO=3 „ SOTA=2

25. El ordenador va a rellenarte una quiniela de n columnas, siendo n un número dado por INPUT. Te genera la primera columna con igual probabilidad para el 1, la X y el 2, y te la muestra. Te pregunta si te gusta, y en caso afirmativo deberá almacenarla. Si no, genera otra. Y así con las n columnas.

Llega el domingo, y le introduces la quiniela ganadora. El ordenador te cuenta los aciertos de cada columna almacenada.

26. Igual que el anterior, donde el ordenador genere cada columna con tantos unos, equis y doses como desees (aproximadamente).

27. Igual que el 24, pero el pronóstico de cada partido se hace con el criterio siguiente: para cada partido, el micro te pregunta los puntos de cada equipo. El pronóstico se hará teniendo en cuenta estos números.

28. Igual que el anterior, pero el pronóstico se ajustará a otras variables que puedas imaginar (jugar en campo ajeno, goles obtenidos por cada equipo, número de veces que un equipo ha ganado al otro, etc.).

28. Vamos a situar las 27 celdas del cubo de Rubik números aleatorios, comprendidos entre el 50 y el 100. Realiza el programa que permite visualizar:

a) el contenido de esta matriz, por «pisos»,
b) la suma de las celdas de cada piso,
c) la suma de las celdas de cada diagonal (son 4 !).

30. Supón un patio de butacas de un minicine, formado por 20 filas, y 28 asientos por fila, separados por un pasillo central. Representaremos una butaca ocupada por un 1, y por un 0 si está vacía. En un determinado momento están ocupadas 200 butacas. Visualiza en pantalla con ceros y unos el estado del patio de butacas en este momento (el llenado se hará aleatoriamente).

31. Mejora el programa anterior, considerando que a partir de este momento te haces cargo de la taquilla. Se le dan al ordenador los datos siguientes: número de entradas y número de fila que quiere el cliente. Si hay sitio para ponerles juntos, se venden estas localidades, y se ve en pantalla la ocupación (con unos). Si no hay sitio en esa fila el ordenador solicita otra, y les coloca ahí. Y así sucesivamente. En cada momento aparecerá en pantalla el número de localidades que quedan por vender.

32. Vas a confeccionarte tu agenda telefónica, donde figura nombre, primer apellido y número de teléfono de las personas que conoces. Realiza el programa que ordene alfabéticamente tu agenda por el nombre o por el apellido, a voluntad.

33. Tienes ya hecha tu agenda telefónica. Quieres llamar a Luis, pero tienes varios amigos que se llaman Luis. El programa te visualiza todos éstos, con su correspondiente número telefónico. Si no hubiera ninguno, da un mensaje (Luis ha sido un ejemplo).

34. Dimensiona y llena completamente una matriz de 13*13 con los símbolos #,&,$, de forma aleatoria y equiprobable. Visualízala en pantalla tal como la llenó.

35. Vamos a confeccionar una historia ficticia de un individuo entresacando un dato al azar de cada una de las matrices siguientes, que llenarás previamente:

— signos del Zodíaco (12),
— provincias de la Comunidad Castilla-La Mancha (5),
— profesiones (las que quieras).

36. Utilizando una de las matrices del programa anterior, reordénala por palabras de menor a mayor longitud.

37. Almacenado un colectivo de individuos por su nombre, edad, estado civil e ingresos, realiza el programa que permita sacar en pantalla los individuos que tengan determinada edad o determinado estado civil.

38. Supón que en tu casa tienes un terminal del supermercado, y vas a hacer la compra de la semana desde tu casa. Inicialmente, el ordenador te presenta en pantalla los siguientes grupos:

1. legumbres y hortalizas,
2. carnes,
3. frutas,
4. vino s.

Una vez elegido el grupo, te presenta los siguientes productos de cada grupo, con su correspondiente precio:

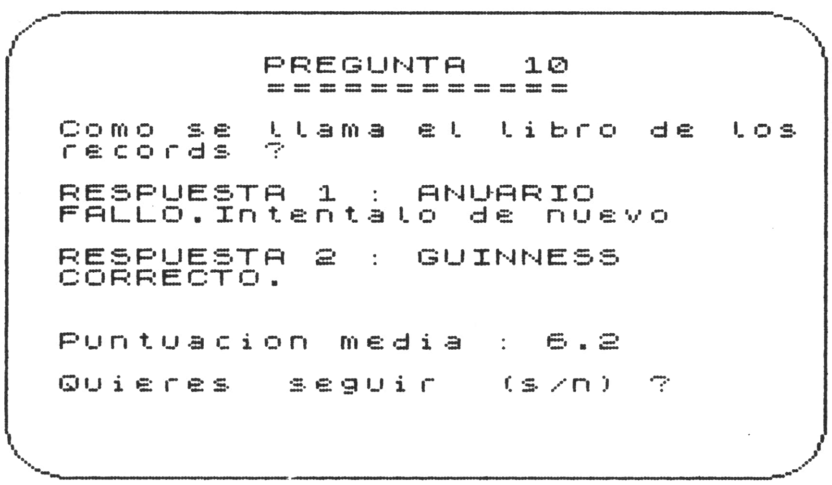
GRUPO 1: 1—lentejas, 150 ptas./Kg.
2—garbanzos, 130 ptas./Kg.
3—lechuga, 80 ptas./unidad.
4—tomates, 90 ptas./Kg.
GRUPO 2: 1—ternera, 850 ptas./Kg.
2—vacuno, 610 ptas./Kg.
3—cerdo, 440 ptas./Kg.
GRUPO 3: 1—fresas, 160 ptas./Kg.
2—manzanas, 80 ptas./Kg.
3—naranjas, 85 ptas./Kg.
4—plátanos, 120 ptas./Kg.
5—peras, 135 ptas./Kg.
GRUPO 4: 1—vino blanco, 65 ptas./litro.
5—vino tinto, 70 ptas./litro.

Eliges el correspondiente producto, y su cantidad. Después, el ordenador te da la posibilidad de elegir otra cosa. Cuando no quieras más, te hace la nota.

Inicialmente el ordenador del supermercado se identificará, y a su vez te pedirá el número de la tarjeta de crédito a la que se carga el importe de la compra. Si el importe sobrepasa 35.000 ptas., la compra no se puede realizar.

TEMA 11

SUBRUTINAS



PROGRAMA N.º 1.

INTRODUCCION. VENTAJAS

En ocasiones, el contenido de un determinado número de líneas juntas de programa se va a tener que repetir varias veces, a lo largo del listado. Para evitar esta repetición están las subrutinas. La subrutina es el paquete de líneas de programa que se repiten. Este paquete conviene numerarlo en una zona fuera del cuerpo principal del programa. Para la ejecución de la subrutina, existe un procedimiento de llamada, y retorno al programa principal.

Las ventajas que se derivan de su utilización son:

1.—Menor tiempo de introducción de programa.

2.—Ahorro de memoria RAM.

3.—Más fácil seguimiento del programa.

No obstante, la utilización de las subrutinas no es recomendable en todos los programas por razones que verás tú mismo.

SENTENCIAS: GOSUB- RETURN

La llamada a la subrutina, tantas veces como quieras, se realiza mediante la instrucción:

n.l. GOSUB (nº de línea del comienzo de subrutina)

Forzosamente, la subrutina deberá terminar con la sentencia RETURN, a secas. Cuando la ejecución del programa se encuentra con esta sentencia, el control se transfiere a la línea siguiente de aquel GOSUB.

Un esquema de utilización sería el siguiente:

```
  10 ...
  15 ...
  20 ...
  25 GOSUB 1000
  30 ...
  75 ...
  90 STOP
1000 ...
1010 ...
1020 RETURN
```

El siguiente programa está optimizado con la utilización de una subrutina. Tras introducir un número, analiza si es positivo, nulo o negativo. Esto se hace para 10 números.

Calcula su media y la analiza igualmente.

```
  10 LET s=0
  20 FOR i=1 TO 10
  30   INPUT a
  40   PRINT a,
  50   GOSUB    1000
  60   LET s=s+a
  70 NEXT i
  80 LET m=s/10
  90 LET a=m
 100 PRINT "MEDIA=";m,
 110 GOSUB    1000
 120 STOP
1000 IF a>0 THEN PRINT "ES POSITIVO"
1005 IF a=0 THEN PRINT "ES NULO"
1010 IF a<0 THEN PRINT "ES NEGATIVO"
1020 RETURN
```

El cambio de la línea 90 es preparatorio para la subrutina, ya que ésta sólo trabaja con la variable a. Observa que de no haber seguido este procedimiento hubieras necesitado repetir las líneas 1000-1005 y 1010 tras la obtención de la media m.

Observaciones

— Se puede acceder al paquete de la subrutina no solamente a través del comienzo de la misma (como sucede necesariamente en bucles), sino en cualquiera de sus líneas.

— Cada vez que uses una subrutina, debe ejecutarse el correspondiente RETURN; de lo contrario, el programa podría interceptar otro RETURN, mandando el control a una línea no deseada.

— Una subrutina puede llamar a otra (subrutinas encajadas). Como hay dos GOSUB, tiene que haber dos RETURN. Por extensión, análogo si son más de dos. El esquema sería:

```
  10 ...
  15 ...
  20 GOSUB 1000
  ...
  90 STOP
1000 ...
1005 GOSUB 2000
1010 ...
1015 IF x=a THEN GOSUB 2000
1020 ...
1030 RETURN
2000 ...
2100 ...
2200 RETURN
```

Vemos otro programa, utilizando subrutinas encajadas. Es un sencillo test de 5 preguntas de cultura general. Es de respuesta única, y al final te dice los aciertos.

```
  10 DATA "qué emperador reinaba cuando nació Cristo?", "Augusto"
  20 DATA "en qué año se llegó a la luna?", "1969"
  30 DATA "cuál es la provincia más grande de España?", "Badajoz"
  40 DATA  "qué científico español descubrió el ADN?", "Ochoa"
  50 DATA "qué nombre recibe la unidad de información más simple, en informática?", "bit"
 100 let s=0
 110 FOR i=1 TO 5
 120   READ p $:PRINT  p $
 130   READ r $
 140   INPUT a $
 150   IF a $=r $ THEN GOSUB 500:GOTO 200
 160   GOSUB 600
 200 NEXT i
 210 PRINT "PUNTUACION SOBRE 5: ";s
 220 STOP
 500 PRINT "acertaste !"
 510 LET s=s+1 : GOSUB 1000
 520 RETURN
 600 PRINT "No es correcto."
 610 PRINT "La respuesta es ";r $
 620 GOSUB 1000
```

```
 530  RETURN
1000  PRINT AT 15,0;"QUEDAN ";5-i;" preguntas"
1010  PRINT AT 18,9;"pulsa una tecla"
1020  PAUSE O:CLS
1030  RETURN
```

Observa que, tanto si aciertas como si no (rutinas 500 y 600, respectivamente), el programa pasa por la subrutina 1000.

DIRECCIONAMIENTOS MULTIPLES DE PROGRAMA

A veces interesa que el control del programa pase a una subrutima u otra, dependiendo del valor que tome una cierta variable. Para ello se utiliza la sentencia ON, cuyo formato es el siguiente:

```
n.l.  ON (variable)  GOSUB  n.l.,n.l.,n.l.,...
```

Como ejemplo, podríamos tener:

```
70  ON  x  GOSUB    900,435,1200,1234
```

Al llegar a la línea 70, la variable x ha de tener algún valor. Si es 1, el programa ejecuta la primera subrutina, que comienza en la línea 900. Si es 2, ejecuta la segunda subrutina, y así sucesivamente.

Observaciones

— Si x fuese un valor no entero, el ordenador toma su parte entera de cara a esta instrucción.

— Si el valor de x supera al número de subrutinas que siguen a GOSUB (4 en nuestro ejemplo) no se ejecuta ninguna de ellas y el programa pasa a la línea siguiente.

— Para ordenadores que no posean esta instrucción, se puede sustituir de dos formas distintas:

a) De forma análoga a como se hizo en la pág. 70 para el GOTO.

b) Almacenando en una lista ordenadamente según la variable.

En nuestro ejemplo, previo dimensionado de la matriz a,

```
  LET a(1)=900
  LET a(2)=435
  LET a(3)=1200
  LET a(4)=1234
70 IF x≥1 AND x≤4 THEN GOSUB a(x)
```

Todo lo dicho en este apartado es igualmente válido para la sentencia

```
ON x GOTO  n.l.,n.l.,n.l.,...
```

PROGRAMAS

1. Mejora el programa test de 5 preguntas de cultura general puesto como ejemplo comentado en este tema de modo que admita otra respuesta, caso de fallar la primera. La puntuación sería entonces:

— Acierto en la primera... 10 puntos.
— Acierto en el segundo intento... 4 puntos.

Al final debe dar la puntuación obtenida.

2. Se trata de llenar una fila o una columna (a voluntad) con un carácter elegido por el usuario. Asimismo se puede elegir el número de fila o columna a imprimir.

3. Una guía del ocio electrónica dispone de información sobre:

— cines,
— teatros,
— salas de fiesta.

Una vez elegido del menú el local deseado, aparece una información complementaria de transporte hacia ese local, que consta de:

— autobuses que conducen al mismo,
— línea de metro y parada correspondiente,
— teléfonos de radio- taxis.

4. Un periódico informatizado de ámbito nacional tiene las siguientes secciones y subsecciones:

A-POLITICA
— internacional
— nacional

B-ECONOMIA

C-DEPORTES
— fútbol
— actualidad deportiva

D-PASATIEMPOS
— TV
— juegos y crucigamas
— horóscopo

Se elige la información deseada, y se visualiza en pantalla, con posibilidad de volver al menú a su término.

5. Un grupo de turistas españoles viaja a un país cuya moneda nacional tiene «céntimos», que llamaremos centavos. El micro solicita (una sola vez) el nombre de esa moneda y el cambio oficial de la misma, en pesetas. Cada turista pulsa el dinero que desea cambiar, y el ordenador contestará cuántas monedas y centavos le corresponden.

6. La familia Martínez está compuesta de padre, madre, un niño, una niña y suegra. Su médico de cabecera va a introducir en su fichero del ordenador los datos médicos de cada componente de la familia, que son:

— fecha nacimiento,
— peso,

— grupo sanguíneo,
— enfermedades padecidas.

El programa debe ser tal que posibilite al médico la visualización de la ficha médica de cualquier miembro de la familia.

7. Mejora el programa anterior, de modo que visualice los miembros de la familia en la ordenación deseada:

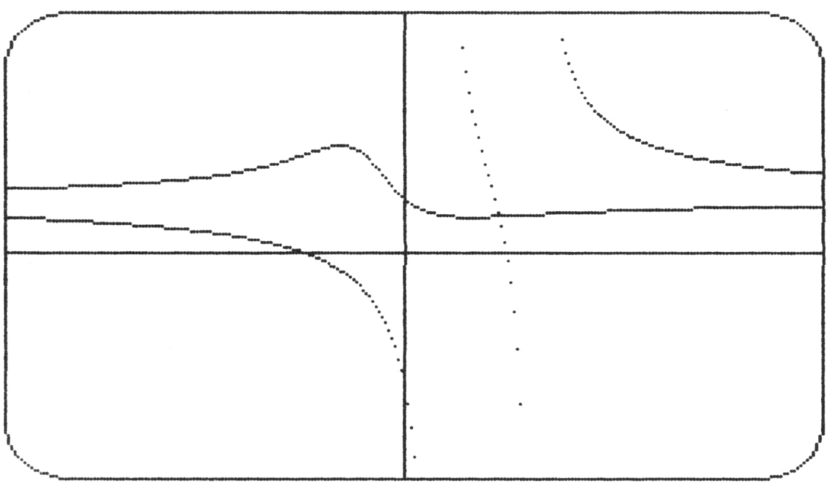
a) por edad, de mayor a menor,
b) por peso, de menor a mayor,
c) alfabéticamente.

8. Un empleado de un cine se coloca a la salida del mismo el día del estreno de una película para realizar la siguiente encuesta: edad, número de distrito donde vive (si es de fuera de la localidad no se procesará este dato), y si le gustó la película, le dejó indiferente o no le gustó. Tras la recogida de datos, se han de obtener los resultados siguientes:

1) N.º de espectadores comprendidos en cada uno de los siguientes bloques de edades: menores de 20 años, entre 20 y 40, mayores de 40 años.
2) De qué distrito de la ciudad han venido más espectadores, y de cuál menos.
3) La edad a la que tiene más aceptación la película entre los asistentes.

TEMA 12

GRAFICOS, COLOR Y SONIDO



PROGRAMA N.º 3-g, PARA a=1 y a=-2.5 (GRAFICA AMPLIADA).

ALTA RESOLUCION. EL PIXEL. SENTENCIA: PLOT

Imagina que el espacio que ocupa un carácter se divide en una cuadrícula de 8 * 8. Cada uno de estos nuevos cuadritos se llama «pixel». Vamos a ver instrucciones de dibujo que manejan estos pixel.

Para un ordenador que tuviera 22 filas y 32 columnas, el número de pixel disponibles sería:

22 * 8 = 176 en vertical, y
32 * 8 = 256 en horizontal.

Es decir, el número total de pixel es 45056.

La impresión de un pixel se realiza mediante la sentencia PLOT, que sigue el formato:

n.l. PLOT x,y ,donde

x e y son las coordenadas matemáticas del pixel, cuyo origen está en el ángulo inferior izquierdo de la pantalla. El primer pixel es el (0,0), por lo que la numeración es:

— coordenada x: de 0 a 255, inclusive.
— coordenada y: de 0 a 175, inclusive.

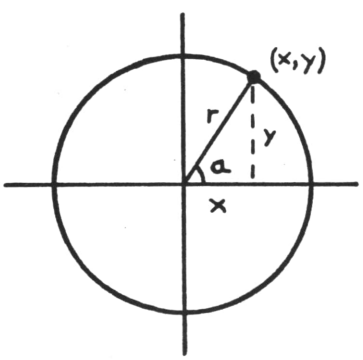
Como ejemplo, dibujemos, punto a punto, una recta vertical cuya abscisa es a elección:

```
10 INPUT "abscisa de la recta? ";x
20 FOR y=0 TO 175
30   PLOT x,y : NEXT y
```

También se puede dibujar una circunferencia PLOT a PLOT, de radio a elección:

```
10 INPUT "radio ? ";r:IF r>87 THEN GOTO 10
20 FOR a=0 TO 2*PI STEP PI/36
30   LET x=r * COS a: LET y=r*SIN a
40   PLOT x+127,y+87
50 NEXT a
```

La línea 20 hace el barrido del ángulo, de 5 en 5 grados. Mediante las razones trigonométricas de este ángulo (variable a) se obtienen las coordenadas x e y:



x= r*cos a
y= r*sen a

Si la línea 40 fuera PLOT x,y, sólo se observaría el primer cuadrante de circunferencia. Por eso hemos trasladado el centro de la circunferencia al punto 127,87.

Observación importante

En el Spectrum se da la particularidad de que si las coordenadas del pixel están fuera de la pantalla, hay dos posibilidades:

— Detención del programa, si el pixel está por encima o derecha de los límites de pixel.
— Rebote (dibujo simétrico), caso de la pared izquierda o inferior.

Prueba la primera posibilidad cambiando la línea 40 por `PLOT x+155,y+140`, con radio 50.
Prueba la segunda posibilidad cambiándola a `PLOT x+155,y+20`, con radio 50.
En el resto de micros simplemente no se ve el PLOT.

SENTENCIA: DRAW. POSIBILIDADES

Esta sentencia de dibujo, se usa bajo uno de los formatos siguientes:
primero: n.l. DRAW a,b
segundo: n.l. DRAW a,b,c

Con el primero se dibuja una línea recta que empieza en el último punto dibujado (con PLOT o DRAW) cuya proyección sobre el eje x es a, y sobre el y es b.

El siguiente programa dibuja un triángulo rectángulo:

```
10 PLOT 50,50
20 DRAW 0,100
30 DRAW 150,-100
40 DRAW -150,0
```

Dependiendo de la resolución del micro, te encontrarás que las líneas casi horizontales o casi verticales aparecen «escalonadas».

```
 1 REM RAYAS ALEATORIAS
 3 REM Y BORRADO CON INVERSE
 5 INPUT "NUMERO RAYAS?"; N
10 DIM M(N): DIM V(N)
15 LET M=50: LET V=50
20 FOR I=1 TO N
25 LET X=M: LET Y=V
30 LET M=INT (RDN*255)
35 LET V=INT (RDN*175)
40 LET M(I)=M: LET V(I)=V
45 PLOT M,V
50  DRAW X-M;Y-V
55 NEXT I
60 PAUSE 0
65 LET M=50: LE V=50
70 LET X=M: LET Y=V
75 FOR I=1 TO N
80 IF I>1 THEN LET X=M(I-1): LET Y=V(I-1)
85 PLOT M(1);V(I)
90 PLOT M(I),V(I)
95 DRAW INVERSE 1;X-M(I),Y-V(I)
100 NEXT I
```

15 RAYAS

Usando el segundo formato se dibuja un arco de circunferencia cuyo incremento según el eje x es a, según el eje y es b y c es el ángulo girado, que se debe expresar en radianes. Ahora bien, para unos valores dados de a y b, el giro puede hacerse en un sentido o en otro. Si c es mayor que O, se gira hacia la izquierda, y hacia la derecha si c es negativo.

El siguiente programa dibuja la media luna.

```
10  PLOT 50,50
20  DRAW 50,50,PI/3
30  DRAW -50,-50,-PI
```

Dibujemos ahora los álabes de una turbina, tantos como quieras:

```
10  INPUT "número de álabes ?"; n
20  FOR i=O TO 2 PI STEP 2*PI/n
30    PLOT 127,87 : DRAW 70*COS i,70*SIN i,PI/2
40  NEXT i
```

NOTA: En los PC es necesario disponer previamente PI=3,1416.

SENTENCIA CIRCLE

Dibuja circunferencias, bajo el formato:

```
n.l.    CIRCLE x,y,r        donde
```

x,y son las coordenadas del centro y r es el radio.

Vamos a ver dos ejemplos de construcción de circunferencias sucesivas:

```
10  REM CIRCUNFERENCIAS CONCENTRICAS
20  INPUT "número de circunferencias ? ";n
30  FOR i=1 TO n
40    CIRCLE 127,87,8*i
50  NEXT i:STOP
```

```
20  REM CIRCUNFERENCIAS CONTIGUAS
30  INPUT "número de circunferencias ? ";n
40  FOR i=1 TO n
50    CIRCLE 50+12+*i,60,20
60  NEXT 1
```

Observarás que si introduces un número muy grande, se detiene el dibujo al llegar al borde de la pantalla.

SENTENCIAS: INVERSE, OVER

Estas sentencias van detrás de una de las instrucciones PLOT,DRAW,CIRCLE, siguiéndole un parámetro cuyos valores pueden ser 0 ó 1. Cuando es 0, la sentencia no actúa, y cuando es 1, queda en estado operativo.

Estas sentencias, respectivamente, sirven para borrado e inversión de dibujo (negro ↔ → blanco), y tienen los formatos:

```
n.l.  DRAW( o PLOT o CIRCLE) INVERSE 1
n.l.  DRAW(o PLOT o CIRCLE) OVER 1
```

Como ejemplo comparativo, mete el siguiente programa y fíjate en la forma de operar de cada una de estas sentencias:

```
10  FOR i=10 TO 14
20    PRINT AT i,5; "■■■■■■■■"
30  NEXT i
40  PAUSE 0
50  PLOT 80,170:DRAW 0,-150
55  PAUSE 0
60  PLOT 130,170:DRAW INVERSE 1;0,-150
65 PAUSE 0
70  PLOT 180,170:DRAW OVER 1;0.- 150
```

Veamos otro ejemplo, donde el parámetro de INVERSE va variando a lo largo del programa. Es el dibujo de un cuadrado «respirando».

```
 10   DATA 3,3,31,31,127,87,1
 20  READ m,p,n,q,x,y,s
 30  INPUT "cuántas respiraciones ? ";h
 40  FOR v=1 TO 2*h
 50    FOR u=m TO n STEP s*2
 60      LET c=0:GOSUB 500
 70      PAUSE 8
 80      LET c=1:GOSUB 500
 90      LET x=x-s:LET y=y-s
100    NEXT u
110    LET m=u;LET n=p*(s=1) + q*(s=-1):LET s=-s
120  NEXT v
130  STOP
500  PLOT INVERSE e;x,y:DRAW INVERSE c;0.u
510  DRAW INVERSE c;u,0
520  DRAW INVERSE c;0,-u
530  DRAW INVERSE c;- u,0:RETURN
```

ATRIBUTOS DE PANTALLA: INK, PAPER, BRIGHT, FLASH. SENTENCIAS BORDER, INVERSE PARA CARACTERES

Las sentencias PAPER e INK, seguidas de un número que especifica el color (de 0 a 7 en el Spectrum) colorean, respectivamente, el fondo de la pantalla y la «tinta electrónica» de escritura de caracteres y pixel.

Por ejemplo, si pusiéramos:

```
n.l.   CLS:PAPER 7:INK 2
```

sobre fondo blanco escribiría con tinta roja.

También se puede colorear el borde o marco de la pantalla con la sentencia BORDER, asimismo, seguida del número de color.

En ordenadores como el Spectrum, se puede trabajar con dos tonos de brillo: normal (BRIGHT O) y el intenso (BRIGHT 1). Prueba:

```
10  PRINT AT 7,2; BRIGHT 1; "INFORMATICA"
```

La sentencia FLASH en su estado activado (FLASH 1) produce una especie de intermitencia intercambiando constantemente el color de la tinta por el papel, y viceverse. Prueba:

```
10  PRINT AT 7,2;FLASH 1;"INFORMATICA"
```

La sentencia INVERSE también sirve asociada a cadenas o variables alfanuméricas, para poder escribirlas con color invertido, es decir, con el color del fondo como tinta, y viceversa. Prueba:

```
10  PRINT AT 7,2;INVERSE 1;"INFORMATICA"
```

Observaciones

1. En todas estas sentencias, el parámetro puede ser variable, dentro de los límites establecidos.
2. En el Spectrum, cuando uses estas sentencias con sentencias de dibujo, ten en cuenta que actúan sobre el carácter.
3. La acción de las sentencias BRIGHT, FLASH, INVERSE se anula al cambiar de línea o al colocar dos puntos.

```
PRUEBA

PRINT AT 0,2; "NOMBRE"
    PAPER 5,,, "MERCEDES",,,

PRINT AT 2,18; "NOMBRE"; INK 5
,,, "LOPEZ",INK 3; "MERCEDES"

PRINT  "MERCEDES LOPEZ",,
PAPER 1;INK 6; "ALCALA,1"
```

SENTENCIAS: POINT, SCREEN $,ATTR

La sentencia SCREEN $ identifica un carácter en una posición de pantalla. POINT hace lo

mismo, pero referido a un pixel. Y ATTR investiga determinadas cualidades (atributos) de esa posición.

El formato del POINT es:

```
n.l.  IF POINT(x,y)=p THEN ...
```

donde x,y son las coordinadas del pixel a investigar y p es el parámetro que puede ser 0 ó 1. Si es 0, esa posición de pixel está vacía, y si es 1, está dibujada.

Como ejemplo de utilización, dibujemos un conjunto de puntos aleatorios, y luego el POINT los busca, dando su posición cuando encuentra uno de ellos.

```
10  INPUT "cuántos puntos quieres? ";n
20  FOR i=1 TO n
30    PLOT 100*RND+150,50*RND : NEXT i
40  PRINT "X Y"
50  FOR y=0 TO 50
60    FOR x=15TO 250
70      IF POINT(x,y)=1 THEN PRINT x;TAB 5;y
80    NEXT x
90  NEXT y
```

Este otro ejemplo simula el relleno de una botella hasta la altura que quieras (menor que 70):

```
10  INPUT "altura a llenar (< 70)";l
15  IF l>70  THEN GOTO 10
20  PLOT 12,70:DRAW 0,-14:DRAW -7,-7:DRAW 0,-45
30  DRAW 20,0:DRAW 0,45:DRAW -7,7:DRAW 0,14
40  FOR y=0 TO 1
50    FOR x=0 TO 50
60      IF POINT(x,y)=1 THEN PLOT x+1,y:IF POINT(x+2,y)=1 THEN GOTO 90
70    NEXT x
90  NEXT y
```

Si quisiéramos investigar los atributos (es decir, color de tinta, color de papel, brillo y flash) debemos usar la sentencia ATTR, cuyo formato es:

```
n.l.  IF ATTR (f,c)=número THEN ...
```

donde f,c son la fila y columna de la posición a investigar; número representa el conjunto de los atributos, obteniéndose con el siguiente criterio:

color de tinta	número de color (0-7)
color de papel	8 veces el n.º color
brillo activado	64
brillo desactivado	0
flash activado	128
flash	0

realizándose la suma de los valores correspondientes.

Por ejemplo, tinta negra sobre papel blanco, sin brillo y con flash sería la suma:

```
0+56+0+128= 184
```

Si insertamos en el programa anterior la línea `120 IF ATR(v,h)=184 THEN PRINT AT v,h; BRIGHT 1;a$:GOTO 90`

cada vez que la pelota se encuentre con el conjunto de atributos representados por el número 184 imprime la estrella, pero con brillo.

Te recalcamos que, al revés que con SCREEN $, el ATTR sólo investiga los atributos de la posición, independientemente del carácter que hubiera en dicha posición.

El formato de la sentencia SCREEN $ es:

```
n.l.  IF SCREEN $(f,c)=p$ THEN ...
```

donde f,c son la fila y columna del carácter a investigar en la pantalla y p$ es el carácter buscado (definido previamente).

Es semejante a la sentencia POINT, pero referida a caracteres. Prescinde del análisis de los atributos de ese carácter (color, brillo, etc.).

En el siguiente ejemplo, primero se dibuja un número de estrellas elegido por ti, distribuidas al azar por la pantalla. Después sale una pelota en la parte superior de la pantalla. Tras pulsar una tecla, avanza bajo un ángulo de 45°, rebotando en los extremos de la pantalla. En su recorrido por la pantalla, por la línea de programa 460 cada vez que se encuentre una estrella la hace relampaguear. El programa es:

```
 20  LET a$="*"
 30  INPUT "cuántas estrellas ? ";n
 40   FOR i=1 TO n
 50    PRINT AT RND*21,RND*31;a$:NEXT i
 60   LET x=1:LET y=1
 70   LET v=0:LET h=INT(RND*20)
 80   PRINT AT v,h;" "
 90   LET v=v+y:LET h=h+x
100  IF h=0 OR h=31 THEN LET x=- x
110  IF v=0 OR v=21 THEN LET y=-y
130  IF SCREEN $(v,h)=a$ THEN PRINT AT v,h;FLASH 1;a$:GOTO 90
140  PRINT AT v,h;"0"
150  PAUSE 0
160  GOTO 80
```

SENTENCIA: BEEP

Esta es la única sentencia de sonido que posee el Spectrum. El formato es:

```
n.l.  BEEP t,f
```

donde t es la duración del sonido, en segundos, y f es un número que da el tono del sonido (nota).

Las limitaciones de estos parámetros son:

```
0<t<10
-60<f<70
```

Para referencia a compatibles, véase el APENDICE.

Variando adecuadamente este par de parámetros puedes conseguir diversos efectos sonoros. Por ejemplo, para hacer escalas conviene saber que el DO central del piano se corresponde con una f igual a 0, y los sucesivos semitonos se obtienen aumentando la f de 1 en 1 (los anteriores, disminuyendo). Por lo tanto, si queremos variar la nota en una octava musical, variamos la f en 12 unidades.

El siguiente ejemplo da sonidos cada vez más graves y breves:

```
10  FOR n=40    TO c STEP -1
20          BEEP n/100,n : NEXT n
```

Un programa que combina dibujo con sonido es:

```
10  FOR x=0 TO 250
20  LET y=-0.01*x*x + 2.5*x
30  PLOT x,y
40  BEEP 0.01,y/3
50  NEXT x
```

PROGRAMAS

1. Dibuja los ejes cartesianos en el centro de la pantalla. Después traza, una a una, las seis funciones trigonométricas, con una amplitud suficientemente grande.

2. *Dibujo de cónicas.*—Te damos las ecuaciones de estas curvas en su forma más apta para la representación gráfica. Cuida que los valores de INPUT sean los adecuados para que la gráfica no se salga de la pantalla. Son:

CIRCUNFERENCIA: $x = R \cdot \cos \alpha$
$y = R \cdot \text{sen } \alpha$ con INPUT R y STEP adecuado.

ELIPSE: $x = a \cdot \cos \alpha$
$y = b \cdot \cos \alpha$ con INPUT a,b (STEP adecuado).

HIPERBOLA: $y = \pm\, b/a \sqrt{x^2 - a^2}$ con INPUT a, b

PARABOLA: $y = ax^2 + bx + c$ con INPUT, a,b,c

Si lo crees conveniente, traslada la figura.

3. Representa las funciones siguientes:

a) $y = \dfrac{x^2 + 4}{x^2 - 4}$

b) $y = \dfrac{x^2 - 4}{x^2 + 4}$

c) $y = \dfrac{x^2 - x + 2}{x}$

d) $y = \dfrac{\text{Ln } x}{x}$

e) $y = \dfrac{e^x}{x}$

f) $y = 80 \cdot \dfrac{\text{sen } x}{x}$

g) $y = \dfrac{x^2 + a}{x^2 + ax + 1}$, con INPUT a. Según que a valga:

a>2 „ a=2 „ 2>a>0 „ a=0 „ -2<a<0 „ a=-2 „ a<-2
salen siete gráficas diferentes.

4. Tomando u como parámetro, introduce las siguientes ecuaciones paramétricas, dibujando el micro la trayectoria:
a) x= 10∗u-120 „ y=80∗COS u / SQR u
u varía de 1 a 23, en intervalos de 0,1
b) x= 0.7∗u∗u -16∗u „ y= 40∗COS u
u varía de 0 a 29, de 0,1 en 0,1
c) x= 4∗u∗COS u + 20 „ y= 45∗SIN u
u varía de 0 a 12, con un paso de 0,1

5. La espiral tiene por ecuación, en polares, r=a∗t, (t en radianes). El programa para su dibujo es el siguiente:

```
10  INPUT "anchura canal ? ";a
20  FOR t=0 TO 10*PI STEP 0.2
30    LET r=a*t
40    LET x=r*COS t : LET y= r*SIN t
50    PLOT x+127,y+87
60  NEXT t
```

Basándote en la misma idea, construye las gráficas de:

espiral logarítmica → $r=A.e^{B.T.}$, con INPUT A,B

trifolio → r=A.cos 3t ,con INPUT A,B

cardioide → r=A(1 + cos t) ,con INPUT A

cicloide → x= A(t sen t)
y= A(1 - cos t) ,con INPUT A

6. Realiza la simulación de un tiro parabólico, con datos de entrada la velocidad inicial y ángulo de lanzamiento.

7. Una esfera de 4 unidades de PLOT va a moverse por la pantalla de izquierda a derecha, con una velocidad inicial nula, y una aceleración dada por INPUT.

8. Dibujar un prisma triangular, una pirámide, un tetraedro y un octaedro, sucesivamente.

9. Un sector circular queda definido con el radio, y el ángulo en radianes. Dibujar el sector que introduces, y dar el área.

10. Mejorar el programa anterior, dando la posibilidad de que la bisectriz apunte en una de las direcciones N, S, E, O.

11. Se trata de dibujar un triángulo equilátero de lado deseado. Además, la base ha de estar girada respecto a la horizontal un ángulo asimismo a voluntad.

12. Representar un polígono regular de lados cualesquiera (sugerencia: empieza dibujando uno horizontal).

13. Realizar el histograma circular de los datos de una estadística, por porcentajes. Debe hacerse de manera secuencial, es decir, tras dibujar un sector, se imprime la información correspondiente al mismo.

14. Realizar el histograma de barras verticales coloreadas, con la sentencia PRINT AT. La longitud de cada barra será proporcional a los datos.

15. El micro solicita el volumen de ventas de una empresa, mes a mes, de enero a diciembre. Utilizando un adecuado factor de escala, dibuja la gráfica «zig- zag» de estas ventas, colocando las iniciales de los meses en el eje horizontal. Al final debe aparecer una línea horizontal que indica la media anual de las ventas.

16. Dibujar tantas líneas aleatorias como se quiera sin levantar el «lápiz», y luego borrarlas en el mismo orden que fueron generadas.

17. Dibujar tantos rombos yuxtapuestos como se quiera. Idem para esferas.

18. Tienes n cartas en la mano, abiertas en abanico. Dibújalas.

20. Dibujar un sol con n rayos (INPUT n), siendo aleatoria la longitud de cada uno.

21. Dibuja una bobina recta y otra toroidal.

22. Introducida una palabra al ordenador, enmárcala de forma atractiva (sugerencia: como si estuviera en un pergamino, etc.).

23. Dibuja el símbolo de la radiactividad. Asimismo, el de peligro de muerte (calavera con dos tibias).

24. Simular el viaje de la tierra a la luna, de manera que figuren en pantalla la tierra y las sucesivas posiciones del cohete y de la luna, en su giro alrededor de la tierra. Se pretende que el cohete orbite a la luna, por lo que su trayectoria se asemeja a un 8. Has de prever que mientras el cohete viaja hacia la luna, ésta también avanza.

25. Haz una pared de cubitos utilizando un algoritmo, que presente perspectiva (la zona no visible, punteada).

26. Construye un muro de ladrillos, con número de filas deseado, tal como aparece en la fachada de los edificios.

27. A partir de un hexágono centrado en la pantalla, debe observarse la adición progresiva de hexágonos, hasta formar un panal.

28. Dibuja una carretera con una curva, de modo que tenga unos tramos de longitud aleatoria y adelantamiento prohibido. Se indicará mediante raya continua.

29. Coge el código de circulación actualizado y selecciona las señales que creas más frecuentes. Dibuja cada una de ellas en pantalla, con su significado. Cada una ocupará la pantalla, de manera que debe haber la posibilidad de ver la anterior, o pasar a la siguiente.

30. Dibuja el plano de un cruce cuatro caminos, con un seto central en la glorieta, y un paso de cebra para dos de las calles.

31. Mejora el anterior, añadiendo en dos esquinas opuestas un semáforo con los colores reales, ocupando cada uno de ellos un carácter. Deben estar en funcionamiento automático, con un tiempo de cambio dado por INPUT, y de modo que cuando uno está en rojo, el otro esté en verde.

32. Realiza, ayudado con un SCROLL, la secuencia de un adelantamiento. Los coches deben ser gráficos definidos (ver tema siguiente).

33. Dibuja en planta el plano de tu casa. Asigna una letra a cada habitación, de forma que al pulsarla aparezca información técnica sobre la misma, sin desaparición del plano.

34. Entre las columnas 5 y 25 sitúa un laberinto de líneas horizontales y verticales, construidas aleatoriamente con DRAW.

36. Refundiendo los dos programas anteriores, crea el juego que consista en pasar el PLOT a través del laberinto, de la zona izquierda a la derecha. Se debe usar la sentencia POINT para impedir atravesar las paredes. El micro contará el tiempo invertido.

TEMA 13

EL JUEGO DE CARACTERES ASCII

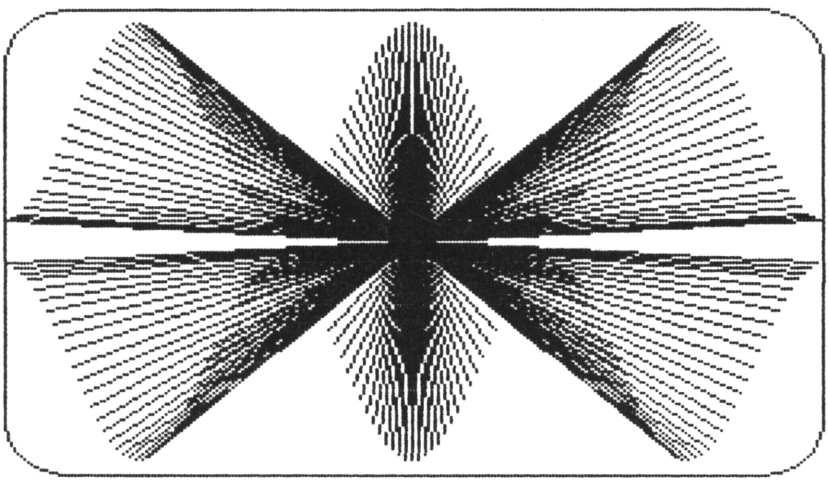


FIGURA DEL PROGRAMA EJEMPLO DE LA PAG. (121).

SENTENCIAS: ASC (CODE), CHR$

El juego de caracteres está formado por las letras del abecedario, números del 0 al 9, signos de puntuación, símbolos gráficos, etc.

El juego de caracteres más usado en el ASCII (American Standard Codes for Information Interchange): Código estándar americano para intercambio de información, en el que todos están especificados por un número de orden. El orden de los símbolos comunes a todos los ordenadores es: números, mayúsculas, minúsculas, símbolos gráficos.

Para ver el código de caracteres de tu micro, introduce el programa:

```
10  PRINT "n. orden","carácter"
20  FOR n=0 TO 255
30    PRINT n,CHR$(n)
40  NEXT n
```

El número máximo de caracteres ASCII sería 256, numerados del 0 al 255, si bien conviene visualizarlos a partir del número 32 (que es el espacio en blanco).

Se puede utilizar en un programa el CHR$ como instrucción, con el número de orden correspondiente. Por ejemplo,

```
PRINT CHR $ 8;
```

desplaza la posición de escritura un lugar a la izquierda en la pantalla.

La sentencia contraria a CHR $ es la ASC (en algunos micros CODE), cuyo formato es

```
n.l.  PRINT ASC a$
```

Esta sentencia da el número de código ASCII del primer carácter de la cadena a$.

Para comprobarlo, mete:

```
10  PRINT ASC("A")
20  PRINT   ASC("ANGELA")
```

El siguiente programa utiliza el teclado como máquina de escribir. Por la línea 20, sólo admite minúsculas, así como el espacio.

```
10  LET a$=INKEY $
20  IF ASC a$=32 OR (ASC a$>96 AND ASC a$<123)
    THEN PRINT a$;
30  IF INKEY $=a$ THEN GOTO 30
40  GOTO 10
```

¿Cuál es el efecto de suprimir la línea 30?

Para ver un empleo de CHR $ en el Spectrum, introduce en el programa anterior la línea:

```
25 IF   a$=CHR $ 8 THEN PRINT CHR $8;CHR $32;CHR $8;
```

Con esta línea, cada vez que pulses cursor a la izquierda:

— la posición de escritura se desplaza un lugar a la izquierda,
— imprime ahí un carácter en blanco,
— vuelve otra vez un lugar a la izquierda.

Esta secuencia permite corregir un error.

Otra utilidad de estas sentencias es para un menú de nueve opciones como máximo. Esquematizado, sería:

```
10  PRINT "OPCIONES 1 a 9. Pulsa la elegida "
20  LET a$=INKEY $
30  IF ASC a$<49 OR ASC a$>57 THEN GOTO 20
40  ...
```

Por último, un dibujo aleatorio de los símbolos gráficos del Spectrum es:

```
10  LET n= 128 + RND*15
20  PRINT CHR $ n;
30  GOTO 10
```

SENTENCIAS: POKE,PEEK

Las dos memorias RAM y ROM están direccionadas, es decir, cada celda de estas memorias

(donde se guarda un byte) se especifica mediante un número, llamado dirección. En el Spectrum, las direcciones:
16384 a 32767 son de memoria RAM, para el de 16 K.
16384 a 65535 son de memoria RAM, para el de 48 K.
0 a 16383 son de memoria ROM, no modificables.

La sentencia POKE sirve para cambiar el byte de una determinada dirección de la RAM. El formato es:

```
n.l.  POKE d,b
```

donde d es la dirección y b es el byte contenido, de -255 a 255.

Según sea la dirección tratada, se puede alterar alguna variable del sistema operativo, dependiendo también del valor b introducido, pudiendo dar lugar a diversos efectos. Algunos ejemplos son:
POKE 23572,32 : anula la sentencia EDIT.
POKE 23609,255: da un pitido cada vez que se pulsa una tecla. Disminuyendo el valor de b, se acorta el pitido.
POKE 23609,0 : se elimina el efecto anterior.
POKE 23658,8 : pone el cursor en modo mayúsculas.
POKE 23658,0 : vuelve el cursor al modo minúsculas.
POKE 23607,0 : después de la actuación de este POKE toda la escritura en pantalla aparecerá de forma «jeroglífica».
POKE 23607,60 : se vuelve a escritura normal.
POKE 23692,1 : ejecuta un SCROLL de una sola línea, una vez que la impresión haya llegado a la última línea.
POKE 23561,0 : el teclado no es repetitivo.
POKE 2365,n+2: borra n líneas últimas de la pantalla (es un CLS parcial). Es preciso poner a continuación INPUT"".
POKE 23613,PEEK 23730- 5: anula la tecla BREAK.
POKE 23613,PEEK 23730-3: activa el BREAK.

La sentencia PEEK proporciona el valor del byte (en decimal) contenido en una cierta dirección d.

El formato es:

```
n.l.  PRINT PEEK d
```

Otra utilización del PEEK es la expresión:

```
(PEEK 23672 + 256 * PEEK 23673 + 65536 * PEEK 23674)/50
```

que investiga el tiempo que lleva el ordenador conectado, en segundos. Entonces, para medir el tiempo que lleva la ejecución de determinado programa, podíamos hacer:

```
10  LET t1=(PEEK 23672+256*PEEK 23673+65536*PEEK 23674)/50
      ...
      ...
90  LET t2=(PEEK 23672+256*PEEK 23673+65536*PEEK 23674)/50
100  PRINT "tardo ";t2-t1;" segundos"
```

Como ejemplo de utilización conjunta del PEEK con el POKE, es posible hacer una línea 0 imborrable. Introduciendo las dos líneas siguientes:

```
1 LET w=PEEK 23637+256*PEEK 23638:POKE w,0:POKE w+1,0:STOP
2 REM ESTA LINEA ES IMBORRABLE
```

al detenerse el programa la línea 2 se convierte en la 0, que es imborrable (puedes suprimir ahora la línea 1).

NOTA: Estos códigos sólo son válidos para el microprocesador Z-80A.

SENTENCIA: USR

Esta sentencia sirve para el uso de rutinas muy particulares, como son:

A) Gráficos definidos por el usuario.

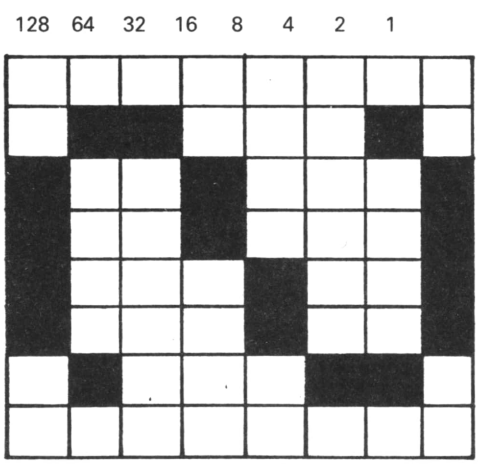
B) Llamada a subrutinas en código máquina.

En el primer caso, se usa conjuntamente por POKE, y en el segundo, con RANDOMIZE. Veamos cada uno de ellos por separado.

A) GRAFICOS DEFINIDOS POR EL USUARIO

Llamaremos gráfico definido por el usuario a un carácter o dibujo que vas a fabricar, que ocupará como máximo el espacio de un carácter de la pantalla. Como el espacio reservado en pantalla para un carácter ocupa 8 filas y 8 columnas *tamaño pixel*, para construir el gráfico deberás llenar los cuadritos que lo definen.

Una de las maneras es el siguiente procedimiento: Se asignan los números siguientes a cada columna del espacio ocupado por el carácter:



Después intenta el dibujo en la cuadrícula, preferiblemente a lápiz. Aquí hemos dibujado el símbolo de la corriente alterna. Cuando está terminado, se suma, para cada fila, los valores de las columnas utilizadas. Fíjate en el ejemplo. Estas sumas, ordenadas empezando por la fila supe-

rior, constituyen los valores de la línea DATA del programa. Estos números se introducen en la zona de memoria reservada a tal fin con las sentencias POKE USR, seguido de una letra de la A a la U, entrecomillada y con el cursor en forma gráfica.

El programa que imprimiría el símbolo anterior, sería:

```
 10 FOR i=0 To 7
 20   READ n
 30   POKE USR "A"+i,n
 40 NEXT i
 50 PRINT "A";" es el símbolo de la corriente alterna"
500 DATA 0,98,145,145,137,137,70,0
```

Tanto en la línea 30 como en la 50, antes de pulsar la A, debes poner el cursor en G (modo gráfico).

Una vez corrido el programa, cada vez que pulses la A, en modo gráfico, aparecerá el gráfico definido.

En el código ASCII están reservados para gráficos definidos las posiciones 144 a 164, o sea, 21. Corresponden de la letra A a la U, respectivamente.

Podemos hacer uso de esto para cargar varios gráficos, usando el mismo bucle. Por ejemplo, vamos a construir los cuatro subíndices 11,12,21,22 y se los asignaremos a la letra T. Sería de la forma siguiente:

```
 10 LET z$="T"
 20 LET s=143
 30 FOR j=1 TO 4
 40   LET s=s+1
 50   LET a$=CHR $(s)
 60   FOR i=0 TO 7
 70     READ n
 80     POKE USR a$ +i,n
 90   NEXT i
100 NEXT j
110 PRINT z$;CHR $ 144,z $;CHR $ 145,z $;CHR $ 146,z$;CHR $ 147
500 DATA 0,0,34,102,170,34,34,34:REM SUBINDICE 11
600 DATA 0,0,36,106,162,36,40,46:REM SUBINDICE 12
700 DATA 0,0,66,166,42,66,130,226:REM SUBINDICE 21
800 DATA 0,0,68,170,34,68,136,238:REM SUBINDICE 22
```

Observación

Dado que un carácter presenta una extensión muy pequeña, el dibujo que desees puedes repartirlo en varios gráficos, y conectarlos.

B) SUBRUTINAS EN CODIGO MAQUINA

Hay subrutinas en código máquina, que se introducen al micro mediante POKE a partir de una determinada dirección de comienzo, y cuya llamada se hace mediante RANDOMIZE USR (dirección de comienzo).

Para realizar estas subrutinas es necesario saber lenguaje máquina, pero te vamos a dar algunas ya realizadas. Por ejemplo, la subrutina que realiza un SCROLL en horizontal, de derecha a izquierda, es:

```
 1  REM SCROLL HORIZONTAL
 5  DATA 33,0,64,85,62,192,6,31,35,94,43,115,35,16,
        249, 114,35,61,32,242,201
10  FOR i=32000 TO 32020
20    READ n
30    POKE i,n
40  NEXT i
```

Mediante el bucle hemos introducido los valores de la línea DATA en las direcciones 32000 a 32020. Por tanto, como en este caso la dirección de comienzo es 32000, cada vez que queramos utilizar la subrutina, pondremos:

```
n.l.  RANDOMIZE USR 32000
```

TODAS LAS SUBRUTINAS
SON REUBICABLES

ES DECIR, PUEDES CAMBIAR
LA DIRECCION INICIAL.

DIRECCION INICIAL: 32.000

SI SE USA RND, LA LLAMADA A
LA SUBRUTINA DEBE HACERSE CON

LET Z=USR (dir)

EN VEZ DE RANDOMIZE USR (dir).

Vamos a aplicarlo a una cadena, de manera que salga por la derecha, letra a letra, y, después de atravesar la pantalla, desaparezca por la izquierda. Habría que añadir las líneas siguientes:

```
100  REM APARECE LA PALABRA POR LA DERECHA
110  INPUT "palabra ? "; $
120  FOR i=1 TO LEN a$
130    PRINT AT 10,31;a$(i)
140    PAUSE 15
150    RANDOMIZE USR 32000
160   NEXT i

200  REM DESPLAZAMIENTO POR LA PANTALLA
210  FOR i=1 TO 31
220    PAUSE 15
230    RANDOMIZE USR 32000
240  NEXT i
```

Esta subrutina ocupa 21 bytes, que han llenado las direcciones de memoria de las 32000 a la 32020. Sin embargo, puedes introducir la subrutina a partir de otra dirección inicial, de forma que, teniendo en cuenta su longitud de memoria, no sobrepases las direcciones.
65535, para el micro de 48 K
32767, para el de 16 K.

Hay una serie de subrutinas de interés que podrás utilizar en tus programas, que las vamos a dividir en tres apartados:
S1: subrutinas que actúan sobre toda la pantalla.
S2: subrutinas que actúan sobre un carácter o cadena.
S3: subrutinas que actúan pixel a pixel.

SUBRUTINAS TIPO S1

Te proponemos que veas el efecto de estas subrutinas mediante su introducción al micro. Veamos las siguientes:

```
  1 REM INVERSION DE LA PANTALLA
 10  FOR i=40000 TO 40017
 20    READ n:POKE i,n: NEXT i
 30  DATA 33,0,64,1,0,24,22,255,122,150,119,35,11,120,
          177,32,247,201
 90  REM APLICACION A DIBUJO
 50   FOR i=0 TO 175 STEP 4
 60    PLOT 0,0:DRAW 255,i: NEXT i
 90  REM LLAMADA A SUBRUTINA
100  PAUSE 0
110  RANDOMIZE USR 40000
120  GOTO 100
```

Por la línea 100, cada vez que pulses una tecla actúa la subrutina.

La siguiente es análoga a la del ejemplo SCROLL HORIZONTAL, que era un scroll de derecha a izquierda. Los siguientes valores de la línea DATA realizan el desplazamiento de izquierda a derecha:

```
DATA   33,255,87,22,0,62,192,6,31,43 ,94,35,115,43,
       16,249,114,43,61,32,242,201
```

Esta tercera subrutina sirve para almacenar en memoria el contenido de una pantalla en determinado momento. Puede admitir hasta cuatro pantallas diferentes.

```
 99  REM DIBUJO EN PANTALLA
100  LET p=1:FOR a=-127 TO 127
110  PLOT 127,93:DRAW a,p*(45+37*COS(3*a*PI/128))
120  LET p=-p:NEXT a
130  PRINT AT 21,4;"EFECTOS TRIDIMENSIONALES"
```

```
499  REM PREPARACION AL ALMACENAJE
500  LET s=32000:LET r=32050
510  RESTORE:FOR j=0 TO 11
520    READ h:POKE s+j,h:NEXT j
530  FOR j=0 TO 11
540    READ h:POKE r+j,h:NEXT j
550  DATA 17,232,128,33,0,64,1,0,27,237,176,201
560  DATA 17,0,64,33,232,128,1,0,27,237,176,201
600  GOSUB 1000: REM ALMACENAJE DEL DIBUJO DE LAS
                          LINEAS 100-130
899  REM VISUALIZACION CONTINUA
900  GOSUB 2000
910  GOTO 900
1000 INPUT "qué número de pantalla le asignas (1 a 4)?";p
1020 LET z=p*7000+26000
1030 LET a2=INT (z/256):LET a1=z-256*a2
1040 POKE 32001,a1:POKE 32002,a2
1050 RANDOMIZE USR 32000 : RETURN

2000 REM VISUALIZACION DE PANTALLA EN MEMORIA
2010 INPUT "pantalla a visualizar (1 a 4)?";p : CLS
2020 LET z=p*7000+26000
2030 LET a2=INT (z/256):LET a1=z-256*a2
2040 POKE 32054,a1:POKE 32055,a2
2050  RANDOMIZE USR 32050 : RETURN
```

En las líneas 100 a 130 se ha generado un dibujo para su almacenamiento y posterior visualización. El programa admite hasta 4. Por tanto, si quieres utilizar las otras, una vez generada la nueva pantalla, teclearás como comando GOSUB 1000 para su almacenamiento en memoria. Una vez almacenadas, el bucle continuo 900-910 visualiza la deseada.

Ampliemos las posibilidades de la subrutina anterior en dos aspectos:

— Almacenamiento de hasta 5 pantallas (cada una consume 6912 bytes de RAM).

— Posibilidad de superponer una pantalla almacenada en memoria con la imagen existente en ese momento.

```
  1 REM ALMACENAJE,VISUALIZACION Y SUPERPOSICION
 10>   FOR i=30900 TO 30946
 20     READ n:POKE i,n:NEXT i
 30  DATA 33,0,64,237,91,118,92,1,0,24,26,182,119,35,19,
          11,120,177,32,246,201 : REM datos de superposición
 40   DATA 237,91,118,92,33,0,64,1,0,27,237,176,201 :
        REM datos de almacenamiento
 50  DATA 237,107,118,92,17,0,64,1,0,27,237,176,201:
              REM datos de visualización.
 60  STOP : REM ahora se generan las pantallas
   ...
   ...
999 REM NUMERACION DE PANTALLA
```

```
1000  INPUT "pantalla ? (1-5) ";p
1010  LET pos= 30975 + 6912*(p-1)
1020  RANDOMIZE pos
1029  REM              OPCIONES
1030  INPUT "meter, sacar o montar?(1-2-3)";a$
1040  IF a$<"1" OR a$>"3" THEN GOTO 1030
1050  GOTO 1000  + 100*VAL a$
1099  REM SUBRUTINA ALMACENAJE
1100  RANDOMIZE USR 30921
1110  STOP : REM generación siguiente pantalla

1199  REM SUBRUTINA VISUALIZACION
1200  RANDOMIZE USR 30934
1210  GOTO 1000

1299  REM SUBRUTINA SUPERPOSICION
1300  POKE 30911,182
1310  RANDOMIZE USR 30900
1320  GOTO 1000
```

SOLO COINCIDENCIAS:

POKE 30911,166

COMPLEMENTARIO DE
COINCIDENCIAS:

POKE 30911,174

El programa empieza cargando en código máquina los datos de cada subrutina. Cuando se detenga en la línea 60 viene el momento de generar la primera pantalla. Se puede hacer:

— por comandos,

— por programa, que podría ir en la zona de puntos suspensivos.

Una vez generada, se le da GOTO 1000 para su numeración. Después el programa te pregunta (línea 1030) la opción a elegir:

1. Se almacena en memoria la imagen existente en pantalla.

2. Se visualiza la pantalla del número seleccionado en la línea 1000, borrando previamente la imagen que hubiera.

3. A la imagen existente en TV se superpone la pantalla seleccionada por la línea 1000.

Luego se generarían las sucesivas pantallas, hasta un máximo de 5, y una vez almacenadas puedes empezar a manejar su visualización- superposición.

Cada almacenaje en memoria se puede hacer con el color de tinta y papel que desees. Al seleccionar la opción 2 se visualizará instantáneamente tal cual. Ahora bien, al seleccionar la opción 3 se conserva el color de la primera pantalla. Es como si se superpusieran a ésta transparencias de las siguientes.

Como última subrutina del tipo S1 vamos a ver la que cambia los colores de tinta y fondo. El primero se hace mediante cambios en la dirección 23297, y el segundo en la dirección 23296:

```
  1 REM CAMBIO DE FONDO
 10 FOR i=40000 TO 40020
 20   READ n: POKE i,n:NEXT i
 30 DATA 33,0,88,1,0,3,237,91,0,91,126,163,178,119,
         35,11,120,177,32,246,201
 39 REM IMAGEN A TRATAR
 40  LET aS=" +++ INFORMATICA +++ "
 50 FOR i=1 TO 10
 60   PRINT:PRINT a$:NEXT i
 69 REM UTILIZACION
 70 LET c=0
 80 POKE 23296,0 : REM tinta fija
 90 PAUSE 20
100 POKE 23297,8*c: REM cambio de fondo
110 RANDOMIZE USR 40000
120 LET c=c+1:IF c>15 THEN LET c=0
130  GOTO 100
```

```
CAMBIAR TINTA:

POKE 23297,C
```

SUBRUTINAS TIPO S2

Esta primera subrutina tipo carácter a carácter coge un carácter y lo gira 180° respecto un eje horizontal que pasase por el centro del carácter:

```
  1 REM INVERSION VERTICAL
 10 FOR i=40000 TO 40019
 20   READ n:POKE i,n:NEXT i
 30 DATA 42,0,91,84,93,6,8,126,36,245,16,251,6,8,241,18,
         20,16,251,201
 70 INPUT "introduce cadena a invertir ";a$
 80 PRINT a$
 90 LET c=0:REM c simboliza la columna
100 POKE 23297,64
110 PAUSE 0
120 POKE 23296,c
130 RANDOMIZE USR 40000
140 LET c=c+1:IF c=256 THEN LET c=0
150 GOTO 110
```

La columna sobre la que actúa la subrutina está especificada en la línea 120 por medio de la

variable c. Por tanto, para invertir todos los caracteres de la cadena, esta variable debe incrementarse, lo que se hace en la 140.

Te vamos a dar los DATA de una subrutina análoga a la anterior, que realiza el giro con un eje vertical, centrado en cada carácter.Son:

```
DATA   42,0,91,62,8,6,8,203,30,203,17,16,250,113,36,61,32,243,201
```

Las subrutinas anteriores sólo actúan sobre la zona de la pantalla formada por las 8 primeras filas. Se debe a la limitación del parámetro c, que no puede sobrepasar el valor de 255.

Para poder acceder a toda la pantalla, habría que pokear:

```
POKE 23297,64 + 8*t ,con 0<t<2
POKE 23296,c ,con 0<c<255
```

valiendo t=0 para las primeras ocho líneas,t=1 para las 8 siguientes y t=2 para las 8 últimas. Para cada una de estas zonas, c va de 0 a 255.

Entonces, ampliando el programa anterior de modo que la subrutina acceda a toda la pantalla, tendríamos que poner, a partir de la línea 70:

```
 70  INPUT "carácter a invertir ? ";a$
 80  FOR i=1 TO 22*32
 90    PRINT a$;
100  NEXT i
110  PAUSE 0
200  IF t>2 THEN LET t=0
210  POKE 23297,64 + 8*t
300  POKE 23296,c
310   RANDOMIZE USR 40000
320  LET c=c+1:IF c=256 THEN LET c=0:LET t=t+1:GOTO 200
330  GOTO 300
```

Observación

En este último programa, como la subrutina actúa sobre tres zonas de ocho filas cada una (24), invierte también los caracteres escritos en zona reservada.

Esta tercera subrutina realiza un entintado, columna a columna, del color que desees y empezando por el lado izquierdo de la pantalla (forma una especie de telón por el lado izquierdo).

```
  1  REM TELON
 10  FOR i=40000 TO 40022
 20    READ n:POKE i,n:NEXT i
 30  DATA 33,255,90,58,0,91,14,24,6,31,43,94,35,115,
          43,16,249,119,43,13,32,242,201
 40  REM PREPARACION DE LA PANTALLA
 50  LET a$=" ... INFORMATICA ... "
 60  FOR i=1 TO 11:PRINT:PRINT a$:NEXT i
 70  PAUSE 0
 90  REM EJECUCION DE SUBRUTINA
```

```
100  FOR i=1 TO 10
110    FOR c=0 TO 7
120      PAUSE 20
130      POKE 23296,9*c
140      RANDOMIZE USR 40000
150    NEXT c
160  NEXT i
```

La línea 130 controla el color de cada barra según el valor de la variable c, que es el código de color. Por la línea 140 se llama a la subrutina, para que aparezca una nueva barra. Por la línea 110 se barren todos los colores, con lo que aparece un telón tipo «arco iris», siendo repetido 10 veces, debido al bucle de la línea 100.

La siguiente subrutina es análoga, pero ahora el telón parte por la derecha de la pantalla, con la diferencia que el telón es de un solo color (puedes hacerlo tipo arco iris).

```
  1  REM TELON POR LA DERECHA
 10  FOR i=40000 TO 40022
 20    READ n: POKE i,n : NEXT i
 30  DATA 33,0,88,58,0,91,14,24,6,31,35,94,43,115,35,16,
          249,119,35,13,32,242,201
 40  REM PREPARACION DE LA PANTALLA
 50  LET a$=" ... INFORMATICA ... "
 60   FOR i=1 TO 11:PRINT:PRINT a$ : NEXT i
 70  INPUT "color del telón ?";c
100  FOR i=1 TO 32
120    PAUSE 5
130    POKE 23296.9*c
140    RANDOMIZE USR 40000
150  NEXT i
```

El telón formado ha ocultado la imagen en pantalla. Para «descorrer» el telón y volver a visualizar la imagen oculta, añadir:

```
200  PAUSE 0
210  POKE 23296,56
220  FOR i=1 TO 32
230    RANDOMIZE USR 40000
240  NEXT i
```

Mejoras y complementos

1. Telón transparente: en las líneas 130, pokeando 8*c, el telón resultará transparente, por lo que no ocultará la imagen.
2. Opciones a los atributos tras el descorrimiento (ver función ATTR): en la línea 210, pokeando el valor de la fórmula de atributos explicada en la función ATTR, se logra una visualización de la pantalla con atributos diferentes de la original. Por ejemplo, pokeando:

```
128 + 8*1 + 6
```

sale tinta amarilla sobre fondo azul, con flash. Ver función ATTR.

Por último, te vamos a dar la subrutina de un telón bidireccional.

```
10  REM telón bidireccional
20  FOR i=60000 TO 60044
30    READ n: POKE i,n: NEXT i
40  DATA 42,0,91,35,34,0,91,43,6,24,58,4,91,17,32,0,
          119,25,16,252,42,2,91,43,34,2,91,35,6,24,58,
            4,91,119,25,16,252,201,195,144,234,25,16,252,201
50  LET a$=" ... INFORMATICA ... "
60  FOR i=TO 11:PRINT:PRINT a$: NEXT i
70  REM color del telón
80  INPUT "color del telón ? ";c
90  POKE 23300,9*c
100 POKE 23297,88:POKE 23299,88
110 REM cierre del telón
120 FOR i=0 TO 15
130   POKE 23296,i
140   POKE 23298,31- i
150   RANDOMIZE USR 60000
160 NEXT i
200 REM          apertura
210 PAUSE O
220 POKE 23300,56: Rem POKEAR fórmula atributos de ATTR
230 FOR i=15 0 STEP -1
240   POKE 23296,i
250   POKE 23298,31- i
260   RANDOMIZE USR 60000
270 NEXT i
```

Para telón transparente, hacer

```
90  POKE 23300, 8*c
```

SUBRUTINAS TIPO S3

Las siguientes subrutinas realizan un SCROLL de pantalla, pixel a pixel, es decir, a una velocidad moderada y dando una sensación de continuidad. La primera es un SCROLL hacia la izquierda:

```
10  REM DESPLAZAMIENTO A LA IZQUIERDA
20  FOR i=40000 TO 40016
30    READ n: POKE i,n: NEXT n
40  DATA 33,255,87,14,192,6,32,183,203,22,43,16,251,13,
          32,245,201
49  REM GRAFICA DEL SENO
```

```
 50  FOR i=1 TO 255
 60    PLOT i,80 + 70*SIN (i/20)
 70  NEXT i
 79  REM SIGUE EL SENO, Y LLAMADA A SUBRUTINA
 80  FOR i=256 TO 500
 90    RANDOMIZE USR 40000
100    PLOT 255,80 + 70*SIN (i/20)
110  NEXT i : STOP
```

Se pueden efectuar los cambios siguientes:

— El tercer valor de DATA (87), si lo sustituyes por 79, te realiza el desplazamiento de las 16 primeras filas.

— Si este valor lo sustituyes por 71, el desplazamiento es solamente de las 8 primeras líneas.

Para realizar la subrutina análoga, con desplazamiento contrario, la línea DATA correspondiente es

```
DATA 33,0,64,14,192,6,32,183,203,30,35,16,251,13,32,245,201
```

Para comprobar la rapidez del código máquina en comparación con el BASIC, te proponemos el siguiente ejercicio:

Consiste en realizar la imagen de cualquier figura respecto de un espejo plano (SIMETRIA ESPECULAR). El espejo será vertical y en el centro de la pantalla, y la figura estará en la mitad izquierda. Haz la que quieras aunque nosotros, para prueba, hemos elegido la siguiente:

```
 1  REM FIGURA DE PRUEBA EN LA MITAD IZQUIERDA
10  FOR i=1 TO 7
20    CIRCLE 5*i + 20,12*i + 10,12
30  NEXT i
40  FOR i=1 TO 130 STEP 10
50    PLOT 0,175
60    DRAW i,-160
70  NEXT i
80  PRINT AT 4,5; "SIMETRIA";AT 6,3;"ESPECULAR"
```

Ahora intenta dibujar su imagen, utilizando la instrucción POINT (para ello barre, pixel a pixel, todo el área izquierda, de forma que cuando encuentre un pixel activado, lo dibuje, simétrico, en la derecha). Cuando lo hayas logrado, te darás cuenta que lleva su tiempo, dado que el BASIC es lento.

Te vamos a dar la subrutina que realiza la misma simetría en código máquina.

```
9900  REM SIMETRIA ESPECULAR EN CODIGO MAQUINA
9910  FOR i=60000 TO 60043
9920    READ n:POKE i,n:NEXT i
9930  DATA 243,17,31,0,33,0,64,229,221,225,221,25,6,16,
           197,126,6,8,203,31,203,17,16,250,221,113,0,
           35,221,43,193,16,237,1,16,0,9,124,254,88,56,
           221,251,201
```

```
9940 PLOT 127,0:DRAW 0,175
9950 PRINT AT 21,0;«PULSA UNA TECLA»:PAUSE O
9960 RANDOMIZE USR 60000
```

Otra aplicación del USR se utiliza para averiguar la memoria gastada, así como la que queda disponible:

Memoria que queda: PRINT 65535 -USR 7962

Memoria gastada: como la memoria efectiva de uno de 48 K es 41474 bytes (el resto es para la memoria de vídeo), la gastada será 41474 menos la que queda.

PROGRAMAS

1. Introducida una frase, se deben contar:
— mayúsculas,
— minúsculas,
— comas,
— puntos.

2. Teniendo en cuenta que cada carácter lleva asociado un número según el código ASCII, idea un algoritmo de cifrado, de forma que transforme una frase en algo «indescifrable». Asimismo, conociendo la clave debe existir la posibilidad de descifrar el mensaje.

3. Análogo al anterior, pero de manera que el cifrado se hace sumando un número aleatorio al código ASCII de cada carácter del mensaje original. Para el descifrado, es necesario almacenar estos números aleatorios en una matriz.

4. Hacer un dibujo aleatorio con los símbolos gráficos (CHR$ 128 a CHR$ 143), de longitud aleatoria de 5 a 15 y que llene toda la pantalla (simulador de diseños de jerseys).

5. Diseña gráficos definidos para las letras griegas.

6. Diseña gráficos definidos para flechas (horizontales, verticales y diagonales).

7. Diseña el símbolo de la integral, ocupando tres caracteres en vertical. Asimismo, las diferenciales dt y dl (un carácter).

8. Diseña los símbolos matemáticos de:
— aproximadamente igual,
— el igual en vertical,
— equivale a,
— el factorial doble.

9. Dibuja, ocupando un carácter, un «comecocos».
a) con la boca abierta,
b) con la boca cerrada.

Mueve por la pantalla estos dos gráficos definidos de forma que se vayan «comiendo» una palabra.

10. Dibujar un hombre haciendo gimnasia (varias posiciones).

11. Dibujar los latidos de un corazón (más grande, más pequeños, más grande, etc.). Al pulsar el + debe acelerarse el movimiento, y retardarlo al pulsar el -.

12. Se necesitan las vocales acentuadas, así como la ñ, y la diéresis en la o y en la u.

13. Construye los caracteres ^{o}, ^{a}, así como los numeritos 2 y 3, que simbolizan el cuadrado y cubo.

14. Dados dos números enteros cualesquiera, averigua el tiempo que le lleva al ordenador ejecutar 1.000 veces su suma (sin imprimirla). Asimismo, calcúlalo para el resto de las operaciones aritméticas elementales.

15. ¿Cuántas veces más rápido es que el ordenador efectúe el producto 33*33 que efectuar la potencia 33 ↑ 2?

16. Construye un juego para dos jugadores que consiste en adivinar un número entero, de 0 a 1000, que el ordenador genera y guarda. Gana el que necesite menos tiempo para adivinarlo, dando el ordenador el ganador y el tiempo invertido por cada uno.

17. Para poner a prueba tu memoria haz el programa siguiente: Se trata de recordar un conjunto de números que el ordenador te va a mostrar previamente. En el comienzo del programa, el ordenador te pide cuántos números deseas que te muestre en la prueba (del 0 al 9). El ordenador los genera aleatoriamente, y te los muestra. Al pulsar una tecla, se borra la pantalla. En este momento has de introducir los números de la secuencia, de dígito en dígito. Al primer error, el ordenador te da el número de aciertos.

18. Análogo al anterior, pero en vez de dígitos, el ordenador te propone una secuencia de letras.

19. Utilizando la subrutina SCROLL HORIZONTAL tipo S1, simular un hombre corriendo sobre un fondo montañoso. El individuo deberás hacerlo con dos caracteres definidos que al superponerse den la sensación de movimiento. El fondo se genera aleatoriamente.

20. Análogo al anterior, pero con un movimiento más pausado (caminando), utilizando la subrutina de SCROLL tipo S3.

21. Genera dos pantallas: una es una cuadrícula de diez por diez, y otra es un conjunto de barcos adaptados a la cuadrícula (barcos propios), y la última es el conjunto de barcos enemigos. Utilizando la subrutina de superposición de pantallas, superponlas de diversas formas.

22. Se trata de observar polígonos inscritos en una circunferencia mediante el uso de la subrutina anterior. Para ello almacena en una pantalla la circunferencia, y en otras distintos polígonos regulares.

23. Usando la rutina de superposición, simular el giro constante de un disco alrededor de un diámetro vertical. Para ello has de generar y almacenar cuatro pantallas (circunferencia y tres elipses).

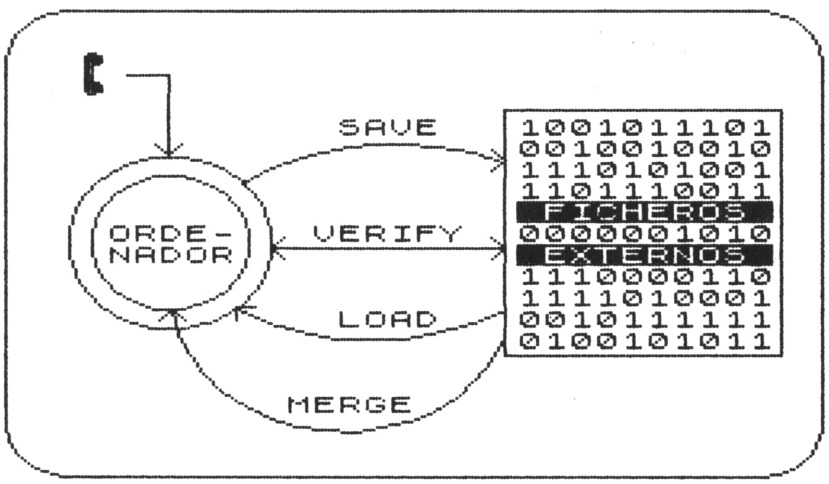
24. Se trata de realizar la imagen especular de una palabra. El ordenador pide tu nombre, y lo imprime en la primera fila. En la segunda dibuja mediante una línea, el imaginario espejo. Por último, en la tercera imprime la imagen del nombre (debes imprimir primero en la tercera línea el nombre normal, y luego usar la subrutina tipo S2).

25. Análogo al anterior, pero ahora el espejo está en posición vertical y a un carácter de la última letra del nombre. Antes de usar la subrutina correspondiente, debes preparar la pantalla de la forma siguiente (cojamos la palabra «PERAS»):

```
PERAS     SAREP
```

TEMA 14

GRABACION Y CARGA DE PROGRAMAS FICHEROS EXTERNOS



SENTENCIAS DE GRABACION Y CARGA.

FICHEROS BIDIRECCIONALES

CINTA
MODEM
DISCO

FICHEROS UNIDIRECCIONALES

MONITOR O TV
IMPRESORA
PLOTTER

GRABACION DE UN PROGRAMA EN CINTA. SENTENCIAS: SAVE,VERIFY

Existe la posibilidad de conservar el listado de un programa mediante la grabación del mismo en una cinta normal de cassette. Los cassette sencillos son los que mejor resultado dan, y si tu cassette es estéreo, has de ponerlo en conmutación MONO y los agudos al máximo.

Para ello se usa la sentencia SAVE, con el formato:

```
SAVE "(nombre del programa)"
```

donde (nombre del programa) no puede superar 10 caracteres.

El cassette debe prepararse del modo siguiente:

1. Conectar, mediante un cable, la salida MIC del micro con la entrada MIC del cassette.
2. Introducir la cinta, en la posición deseada del comienzo de grabación.
3. Anotar el número de vuelta.

Una vez hecho esto, se introduce SAVE «nombre» como comando, apareciendo en pantalla el mensaje: Start tape, then press any key («Pon en marcha la cinta y pulsa una tecla»).

Lo haces, y detienes la cinta cuando salga en pantalla el mensaje «OK».

Este programa, así grabado, necesita que se pulse el RUN para ejecutarlo, una vez cargado (del cassette al micro). Sin embargo, existe la posibilidad de autoejecución del programa a partir de una determinada línea, la que tú desees.

Para ello el comando sería:

```
SAVE «(nombre del programa)» LINE x
```

donde x es el número de línea donde comenzará la autoejecución.

Estas sentencias se usan normalmente como comando, pero pueden ir incorporadas al programa, como instrucciones, con su número de línea.

Para comprobar la bondad de la grabación, tenemos la sentencia VERIFY. Se siguen los pasos:

1. Rebobinar la cinta hacia el comienzo de la grabación.
2. Conectar, mediante un cable, la salida EAR del cassette con la entrada EAR del micro.
3. Pulsa VERIFY "".
4. Pon en marcha el cassette.

Si la verificación ha resultado positiva, saldrá el mensaje «OK». Entonces se puede considerar que el programa ha sido grabado correctamente.

CARGA DE UN PROGRAMA. SENTENCIAS: LOAD,MERGE

Para realizar la operación contraria, es decir, vuelco del programa de la cinta a la memoria RAM del ordenador, se usa la sentencia LOAD, bajo el formato:

```
LOAD "(nombre del programa a cargar)"
```

Si pusieras simplemente LOAD"", te cargaría el primer programa que encuentre en la cinta.

Los pasos a seguir son:

1. Conectar EAR con EAR.
2. Posicionar la cinta.
3. Pulsar LOAD"".
4. Poner en marcha la cinta.

Si el programa fue grabado en su momento con SAVE LINE, en cuanto esté cargado se autoejecuta. De lo contrario, si la carga ha sido correcta sale el «OK».

Cada cassette tiene un volumen máximo de salida. De ahí que tengas que hacer pruebas hasta conseguir el volumen adecuado para una carga correcta. Suele ser unos 3/4 del volumen máximo. En el mismo momento en que se encuentra con un error de carga, por cualquier motivo, el micro saca el mensaje:

```
Tape loading error
```

En este caso debes intentarlo otra vez, modificando el tono y/o el volumen.

La carga usando LOAD, borra de la memoria RAM todo el espacio reservado al BASIC. De ahí que se conserven, entre otras cosas, la memoria de pantalla y los caracteres gráficos del programa anterior. Si quieres borrado total de la memoria RAM, pulsa antes de la carga,

```
RANDOMIZE USR 0
```

Sin embargo, hay un procedimiento de carga de un programa que respeta todo lo que hay en memoria. Es usando la sentencia MERGE en vez de LOAD. El formato es:

```
MERGE "(nombre del programa a fusionar)"
```

Haciendo MERGE "" fusiona el primer programa que encuentre en la cinta.

Observación

La carga mediante MERGE no respeta las líneas del programa primitivo que coinciden en número de línea con el programa a fusionar, quedando las de este último.

Lo mismo sucede con las variables de igual nombre.

La carga de un programa con **MERGE** invalida el autorun **(SAVE LINE)**.

ALMACENAMIENTO DE PANTALLA EN CINTA

Si quieres conservar en cinta lo que figure en pantalla, basta seguir el procedimiento normal de grabación, ya explicado, usando el formato:

```
SAVE "(nombre de la pantalla)" SCREEN $
```

en forma de comando.

Para su carga, análogamente a lo explicado, con:

```
LOAD "nombre" SCREEN $
```

La aparición en pantalla durante la carga se hace por fragmentos progresivos, y al terminar aparecen los atributos.

ESTA VISUALIZACION ES
REUTILIZABLE EN EL PROGRAMA

GRABACION DE PANTALLAS
REUTILIZABLES

1. INTRODUCIR SUBRUTINA CON LOS
DATOS DE LA LINEA 50(PAG 122)
EN LA DIRECCION 58600.

2. GENERACION DE PANTALLA.
3. SAVE "PI" CODE 16384,6912.
4. LOAD "PI" CODE 58620.
5. RANDOMIZE 58620.
6. RANDOMIZE USR 58600.

CADA VEZ QUE SE DESEE
VISUALIZACION, TECLEAR
RANDOMIZE USR 58600.

GRABACION Y CARGA DE DATOS EN FORMA MATRICIAL

En cinta no solamente se puede almacenar un programa y sus variables, sino también un conjunto de datos contenidos en una matriz, ya sea numérica o alfanumérica. Vamos a dividir el proceso en dos fases:

FASE 1: *Llenado y grabación de la matriz*

Vamos a llenar dos matrices, una con cinco preguntas y otra con sus respuestas, para acoplarlas posteriormente a un programa que va a preguntarlas, y su evaluación correspondiente. Sería:

```
10 DIM p$(5,70): DIM r$(5)
20 FOR n=1 TO 5
30   READ p$(n), r$(n) : NEXT n
40 DATA "Los triángulos son de tres tipos: equilátero, isósceles y escaleno", "c", " Amstrong pisó la luna en
   1969", "c"," Las Filipinas son unas islas del hemisferio sur ","f"," La fórmula del ácido sulfúrico es H2SO4 ",
   "c"," Fleming, descubrió la penicilina, era americano ","f"
```

Una vez corrido el programa, podemos grabar el contenido de estas matrices en cinta. Para ello se utiliza el formato:

```
SAVE "(nombre)" DATA nombre de matriz ()
```

donde (nombre) es aquel por el cual el ordenador reconoce esta grabación, nombre de matriz es la letra asignada a la matriz y () es un par de paréntesis, escrito tal cual.

Esta sentencia, al igual que en casos anteriores, puede utilizarse:

— como comando,
— incorporada al programa.

En nuestro caso, podríamos poner:

```
50 SAVE "preguntas" DATA p$()
60 SAVE "respuestas" DATA r$()
```

Tras la grabación en cinta, puedes verificarla.

Observación

Fíjate que hemos grabado el contenido de las matrices exclusivamente, no hace falta grabar el programa.

FASE 2: *Programa de utilización*

Vamos a darte el programa que usa las anteriores matrices para un posible examen. Están incorporadas las sentencias de carga para las dos matrices, que siguen el formato:

```
LOAD "(nombre)" DATA nombre de la matriz ()
```

El programa sería:

```
 10  CLS:DIM p$(5,70):DIM r$(5)
 20   PRINT AT 12,0;"pulsa PLAY para cargar matrices"
 30  LOAD "preguntas" DATA p$()
 40  LOAD "respuestas" DATA r$()
 50  CLS:PRINT AT 10,10;FLASH 1;"PARA LA CINTA";AT
     16,7;"Y PULSA UNA TECLA":PAUSE 0
100  FOR n=1 TO 5
110    CLS:PRINT AT 6,5;"Pregunta ";n
120    PRINT:PRINT p$(n):GOSUB 500
130  NEXT n
200   CLS:PRINT AT 12,1;"De 5 preguntas has acertado ";x
220   STOP

500  PRINT AT 21,7;"cierto (c) falso (f)?"
210   POKE 23658,0:REM cursor en minúsculas
250  IF INKEYS=r$(n)THEN LET x=x+1 : RETURN
260  IF INKEYS="" THEN GOTO 250
270   RETURN
```

Este programa hay que grabarlo.

Si has seguido las fases 1 y 2, tendrás cargado en cinta las matrices y el programa de utilización, por este orden. Sin embargo, lo idóneo es:

— suprimir la línea 20 del programa de utilización,
— grabar este programa delante de la grabación de matrices, y con SAVE LINE 10.

GRABACION Y CARGA DE BYTES PARA CODIGO MAQUINA

Existe la posibilidad de grabar y recuperar los bytes contenidos en ciertas posiciones de memoria. Puede servir para aplicarlo a los bytes de las subrutinas del cap. XIII.

Por ejemplo, si queremos guardar en las posiciones de memoria de la 45000 a la 45050 el byte 33, sería:

```
10  FOR i=45000 TO 45050
20    POKE i,33
30  NEXT i
```

Después de correr el programa, se puede grabar el contenido de estas direcciones de memoria.

Para ello se utiliza, como comando o con número de línea, el formato:

```
SAVE "(nombre)" CODE d,l
```

donde (nombre) es aquel por el cual el ordenador reconocerá el conjunto de bytes, d es la dirección de comienzo y l es el número de bytes a grabar.

Para aplicarlo al programa anterior, sería:

```
SAVE "meter 33" CODE 45000,51
```

Para verificarlo, echa la cinta hacia atrás y teclea

```
VERIFY "" CODE
```

La carga de estos bytes se puede realizar de una de estas dos formas:

A) Con el formato LOAD "" CODE carga los bytes en las mismas direcciones en que estaban en el momento de la grabación.

B) Con el formato LOAD "" CODE n donde n es la dirección a partir de la cual comienzan los bytes a almacenarse en memoria. Por ejemplo, si ponemos:

```
LOAD "" CODE 50000
```

los bytes 33 (en total eran 51) comienzan a ubicarse a partir de la dirección 50000.

Observación

La carga en cualquiera de las dos modalidades respeta el programa BASIC mientras la carga de bytes no invada la zona de memoria en que está.

MODEM, DISCO E IMPRESORA

Los servicios de transmisión de datos entre ordenadores a distancia utilizan la red telefónica. Para ello se requiere transformar las señales digitales en analógicas, las cuales son válidas en la transmisión por el hilo telefónico (aunque existen tramos telefónicos de fibra óptica, donde la señal es digital). Este proceso se llama *modulación.*

Al llegar las señales al ordenador de destino hay que volver a convertirlas en digitales, es el proceso llamado demodulación. Al equipo *MO*dulador-*DEM*odulador, instalado uno en cada ordenador, se le llama «modem». La telemática es un aprovechamiento de este proceso.

Para almacenamiento y rápida lectura de grandes cantidades de datos se usan los disquettes (para micro y miniordenadores). Su memoria oscila, según tamaño y aparato, entre 150 Kbytes y 1 Mbyte (un millón de bytes). Para memorias mayores se usa el llamado disco duro.

Tanto discos como disquettes requieren, antes de su uso, la operación llamada formateo. Formatear es dividir, desde el punto de vista magnético, el disco en pistas (coronas circulares). A su vez, cada pista se divide en un determinado número de sectores. La unidad de disco chequea cada sector, y aquel que tenga algún defecto lo invalida, es decir, no lo utilizará. Al final, se comunican cuántos bytes quedan utilizables para el proceso de grabación.

El disco admite dos tipos de ficheros: secuencial y aleatorio o de acceso directo. En el secuencial, se van leyendo los sectores hasta encontrar el que tenga la información buscada. Consiguientemente, es más lento que el de acceso directo que, como su nombre indica, se posiciona directamente en el sector deseado.

La impresora es un periférico de gran utilidad, y relativamente barato, pues te permite pasar a papel:

— listados de programas,
— salida de resultados,
— figuras en pantalla.

A su vez tienen la posibilidad, que no existe en una máquina de escribir, de utilizar varios tipos de letra, además de caracteres especiales y programables.

Hay una gran gama en cuanto a colores, velocidades y técnicas de impresión.

TEMA 15

PROGRAMAS DE APLICACION DIDACTICA

Se exponen en este tema cinco programas, tratando cada uno de ellos un aspecto muy concreto de las siguientes materias:

— matemáticas,
— física,
— química,
— biología,
— inglés.

Con esto se pretende dar una idea de las posibilidades de ayuda que la informática nos proporciona en cualquier área de la enseñanza.

Cada uno de estos cinco programas son una simplificación de un programa didáctico ambicioso.Puedes elaborar unos programas semejantes, incluso de otras aplicaciones, si has asimilado el contenido de este libro. Con toda seguridad te serán útiles, en uno u otro aspecto.

MATEMATICAS

El programa te permite calcular las operaciones algebraicas de suma, resta y producto entre matrices (incluido un número por una matriz), indefinidamente, siempre que puedas ir asociando adecuadamente la expresión.

Por ejemplo, si se considera la operación ((A+B).c-D+E).F se introduce primero la matriz A, a la que se suma la B; al resultado se le multiplica por el número c; a este resultado se le resta la D, etc.

Características: Si se quiere hacer una operación entre matrices no compatibles (dimensionalmente), por las líneas 820 a 862 (ver listado adjunto) se rechaza la posibilidad de la operación, volviendo a solicitar una matriz hasta que ésta sea correcta.

En una operación entre matrices semejante a la anterior van saliendo en pantalla los resultados parciales. El programa vale para matrices de cualquier dimensión, y los resultados van saliendo fila a fila, guardando una línea en blanco entre cada dos filas de la matriz.

Para el ejemplo siguiente se han escogido matrices cuadradas de orden 3, mostrándose el resultado final de la operación:

((A+B).c-D+E).F

```
1   2   3
11  22  33      <-------------------  matriz A
4   5   6
```

```
0   20  10
30  40  0       <-------------------  matriz B
10  30  0
```

```
   c = 2        <-------------------  constante (matriz de dimensiones 1×1)
```

```
20   -10  0
20   0    30    <-------------------  matriz D
-2   40   12
```

```
1.5      -2.4     0
0        -42.25   4     <-----------  matriz E
5        0        3.4
```

```
2    0    1.6
3.8  5    9     <-------------------  matriz F
10   20   30
```

```
 La MATRIZ resultado es:
423.08   778       1218
834.65   1208.75   2034.95  <------  matriz ((A+B).c- D+E).F
218      218       428
```

Operaciones con MATRICES, de forma indefinida

```
  10 PRINT AT 16,0;"DATOS de la matriz"
  11 INPUT "N. de filas? ";j1: INPUT "N. de columnas? ";h1
  14 DIM a(j1,h1): FOR f=1 TO j1: FOR c=1 TO h1: PRINT AT 20,0;"FILA     ";f: PRI
NT "COLUMNA ";c: INPUT a(f,c): GO SUB 50: NEXT c: NEXT f: CLS
  19 PRINT AT 21,0;"OPERACION a realizar:": INPUT " +(1)  -(2)  *(3) ? ";nu: GO
SUB 50: IF nu<>1 AND nu<>2 AND nu<>3 THEN  GO TO 19
  20 LET co=0: LET m=0: PRINT AT 16,0;"DATOS de la siguiente matriz:": INPUT "N.
 de filas? ";j2: INPUT "N. de columnas? ";h2: GO SUB 50: GO TO (800+20*nu)
  25 DIM b(j2,h2): FOR f=1 TO j2: FOR c=1 TO h2: PRINT AT 20,0;"FILA     ";f: PRI
NT "COLUMNA ";c: INPUT b(f,c): GO SUB 50: NEXT c: NEXT f: CLS
  40 IF nu=1 OR nu=2 THEN  GO TO 500
  41 IF nu=3 THEN  GO TO 600
  50 POKE 23659,6: INPUT "": RETURN
 499 REM ********** SUMA ALGEBRAICA DE MATRICES
 500 LET j3=j1: LET h3=h1: GO SUB 800
 504 FOR f=1 TO j3: FOR c=1 TO h3: IF nu=2 THEN  LET b(f,c)=-b(f,c)
 506 LET c(f,c)=a(f,c)+b(f,c): GO SUB 810: NEXT c: NEXT f: GO TO 700
 599 REM ********** PRODUCTO DE MATRICES
 600 IF (j1=1 AND h1=1) OR (j2=1 AND h2=1) THEN  GO TO 650
 604 LET j3=j1: LET h3=h2: GO SUB 800
 605 FOR f=1 TO j3: FOR c=1 TO h3: LET k=-1
 610 LET k=k+1: LET d=1: LET c(f,c)=c(f,c)+a(f,d+k)*b(d+k,c): GO SUB 810
 615 IF k=h1-1 THEN  GO TO 625
 620 GO TO 610
 625 NEXT c: NEXT f: GO TO 700
 649 REM ********** PRODUCTO DE ESCALAR POR MATRIZ
 650 IF j2=1 THEN  IF h2=1 THEN  LET j3=j1: LET h3=h1: LET b=b(1,1)
 651 IF j1=1 THEN  IF h1=1 THEN  LET j3=j2: LET h3=h2: LET a=a(1,1)
 653 GO SUB 800
 655 FOR f=1 TO j3: FOR c=1 TO h3
 656 IF j1=1 THEN  IF h1=1 THEN  LET c(f,c)=a*b(f,c)
 657 IF j2=1 THEN  IF h2=1 THEN  LET c(f,c)=a(f,c)*b
 659 GO SUB 810: NEXT c: NEXT f
 699 REM *****IMPRESION MATRIZ RESULTADO*****
 700 FOR f=1 TO j3: FOR c=1 TO h3: LET p=m+1: IF c=1 THEN  LET p=0
 710 LET co=co+p: PRINT TAB co;c(f,c);: NEXT c: PRINT : PRINT : LET co=0: NEXT f
 750 LET j1=j3: LET h1=h3: DIM a(j1,h1): FOR f=1 TO j1: FOR c=1 TO h1: LET a(f,c
)=z(f,c): NEXT c: NEXT f
 779 REM ***** OPCION A CONTINUAR
 780 INPUT "MAS OPERACIONES (s/n)? ";a$: IF a$="s" THEN  GO TO 19
 782 IF a$="n" THEN  GO TO 1000
 784 GO TO 780
 799 REM ***** RUTINAS
 800 DIM c(j3,h3): DIM z(j3,h3): PRINT " La MATRIZ resultado es:": PRINT : RETUR
N
 810 LET z(f,c)=c(f,c): LET n=LEN (STR$ (c(f,c))): IF n>m THEN  LET m=n
 811 RETURN
 820 IF j1<>j2 THEN  PRINT AT 18,1;"NO SE PUEDEN SUMAR por no tenerel mismo nume
ro de filas.": PAUSE 300: GO SUB 50: GO TO 19
 822 IF h1<>h2 THEN  PRINT AT 18,1;"NO SE PUEDEN SUMAR por no tenerel mismo nume
ro de columnas.": PAUSE 300: GO SUB 50: GO TO 19
 840 IF j1<>j2 THEN  PRINT AT 18,0;"NO SE PUEDEN RESTAR por no tenerel mismo num
ero de filas.": PAUSE 300: GO SUB 50: GO TO 19
 842 IF h1<>h2 THEN  PRINT AT 18,0;"NO SE PUEDEN RESTAR por no tenerel mismo num
ero de columnas.": PAUSE 300: GO SUB 50: GO TO 19
 844 GO TO 25
 860 IF (j1=1 AND h1=1) OR (j2=1 AND h2=1) THEN  GO TO 25
 862 IF h1<>j2 THEN  PRINT AT 18,2;"NO SE PUEDEN MULTIPLICAR si notienen igual n
umero de columnas de la primera matriz con el de  filas de la segunda.": PAUSE 3
00: GO SUB 50: GO TO 19
 864 GO TO 25
```

Este programa:

a) simula el trazado de rayos y la obtención de la imagen, tanto para una lente convergente como divergente,

b) obtiene los resultados numéricos y cualitativos de la imagen.

Características: Según sea la distancia focal pulsada, la lente aparece en la posición más adecuada para ver el trazado de los rayos.

Las unidades de entrada son arbitrarias (m,cm...) de manera que los resultados se considerarán expresados en la misma unidad. La expresión o/i es la relación del tamaño objeto a tamaño imagen.

Para lentes convergentes, hay seis posibilidades:

— Objeto en el infinito.
— Objeto entre el infinito y centro de curvatura (doble de la distancia focal): ver fig. 1.
— Objeto en el centro de curvatura: ver fig. 4.
— Objeto entre el centro de curvatura y el foco: ver fig. 2.
— Objeto en el foco: ver fig. 5.
— Objeto entre el foco y la lente: ver fig. 3.

Para lentes divergentes, todas las posiciones del objeto conducen a un mismo caso de obtención de imagen (ver fig. 6):

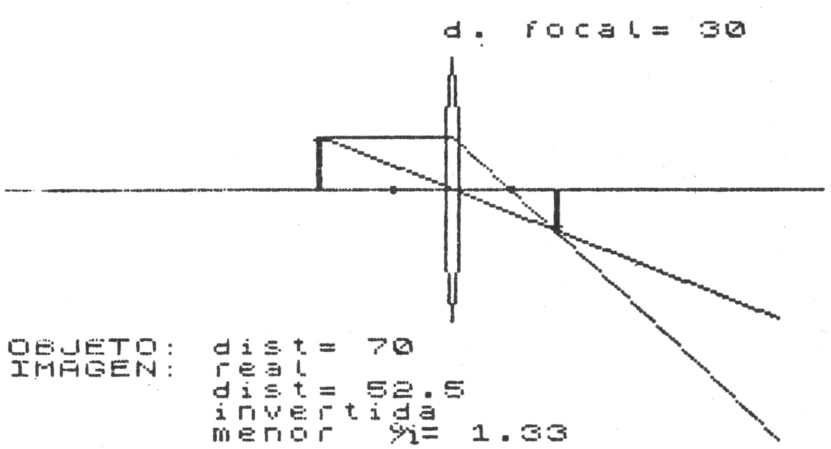


fig. 1

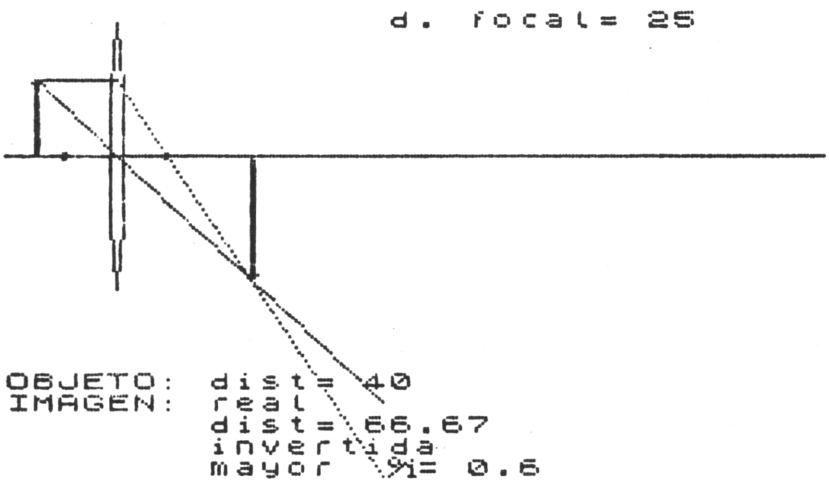

d. focal= 25
OBJETO: dist= 40
IMAGEN: real
dist= 66.67
invertida
mayor %= 0.6

fig. 2

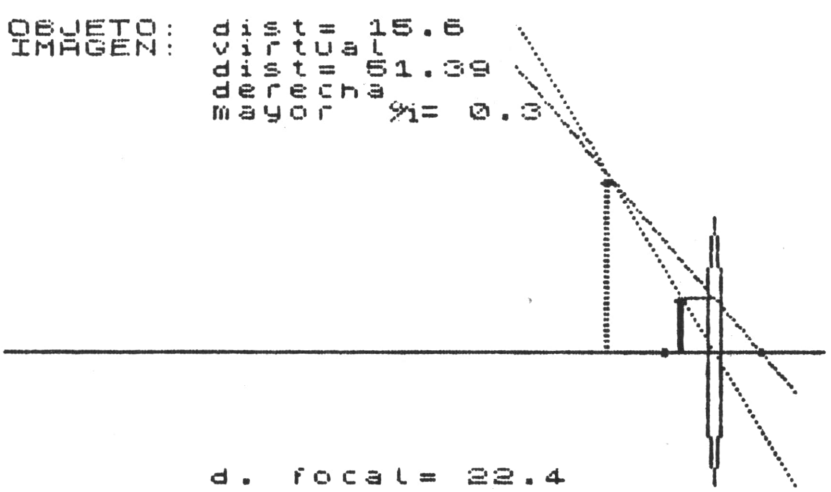

OBJETO: dist= 15.6
IMAGEN: virtual
dist= 51.39
derecha
mayor %= 0.3
d. focal= 22.4

fig. 3

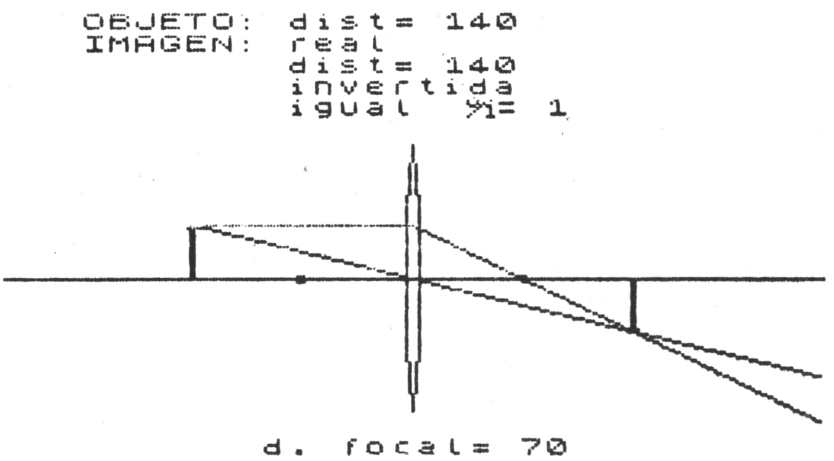

OBJETO: dist= 140
IMAGEN: real
dist= 140
invertida
igual Yi= 1
d. focal= 70

fig. 4

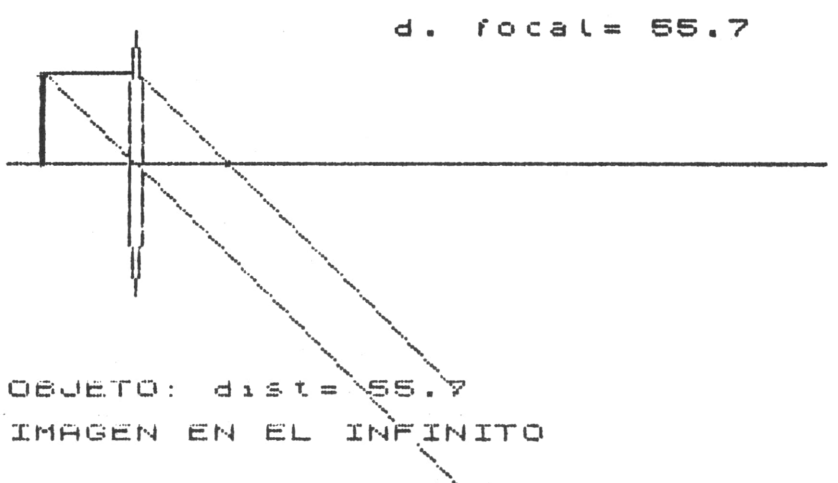

d. focal= 55.7
OBJETO: dist= 55.7
IMAGEN EN EL INFINITO

fig. 5

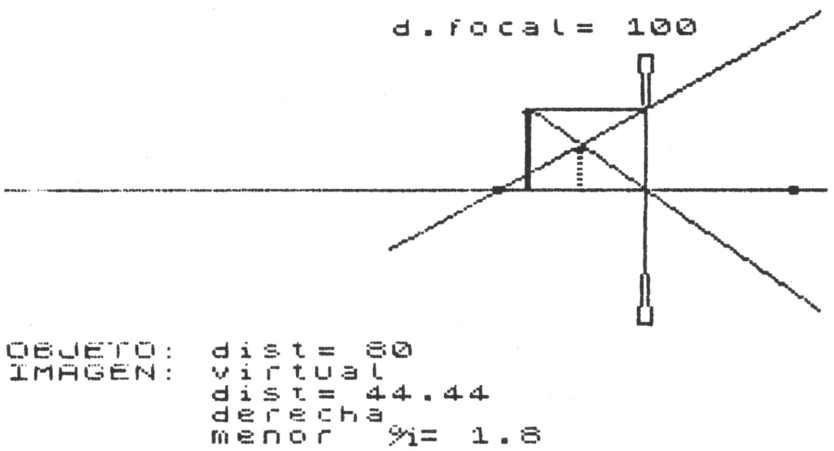


fig. 6

```
   1 RESTORE
   4 DATA 96,145,148,107,17,33,65,131
   5 DATA 0,0,62,0,0,62,0,128
   8 FOR a=0 TO 7: READ n: POKE USR " "+a,n: NEXT a
   9 FOR a=0 TO 7: READ n: POKE USR " "+a,n: NEXT a
  10 LET pa=35: LET t1=.3: LET n1=5: LET ti=.3: LET ni=30: LET t2=.01: LET n2=40
: LET t1=.5: LET n1=10: LET te=.1: LET ne=20: LET tf=.1: LET nf=0: LET tsg=1: LE
T nsg=50: LET tsf=.5: LET nsf=20: LET tsh=.5: LET nsh=35
 100 LET a$="": LET aa=0: LET a1=0: LET bb=0: LET y=-1
 105 BEEP tsf,nsf: PRINT #0;"CONVERGENTE(c)   DIVERGENTE(d)? ": PAUSE 0: LET z$=
INKEY$: INPUT "": IF z$="d" THEN  LET t=30: GO TO 600
 109 IF z$="c" THEN  LET t=20: GO TO 120
 110 GO TO 105
 119 REM *****LENTES CONVERGENTES*****
 120 BEEP tsf,nsf: INPUT "distancia focal (>=1 y <=1000)? ";f
 122 LET f1=f: LET ta=LN f/LN 10+.2: LET f=f/ta: IF f1<1 OR f1>1E3 THEN  GO TO 1
20
 125 BEEP tsf,nsf: INPUT "Distancia objeto / lente (>=1)? ";p
 127 LET p1=p: LET p=p/ta: IF p1<1 THEN  GO TO 125
 128 IF p>2*f THEN  LET a=150-f/ta: LET b=95: LET xx=17: LET yy=0: LET b$="menor
"
 130 IF p=2*f THEN  LET a=127: LET b=75: LET xx=0: LET yy=3: LET b$="igual"
 132 IF p>=f AND p<2*f THEN  LET t=20+f/2: LET a=25+f/ta: LET b=120: LET xx=17:
LET yy=0: LET b$="mayor"
 134 IF p<f THEN  LET a=230-f/ta: LET b=50: LET xx=0: LET yy=0: LET b$="mayor"
 136 IF p<>f THEN  LET x=(a*f-a*p-p*f)/(f-p): LET y=b+t+p*t/(f-p)
 138 IF y<0 OR y>175 THEN  LET a$="NO SALE EN TV"
 140 IF p=f THEN  LET x=1e30
 142 LET c=b+t: LET d=-t/p: LET e=-t/f: LET g=-a+p
 144 IF p<>f THEN  LET pp=f*p/(p-f)
 146 IF p<>f THEN  LET tt=t*pp/p
 148 GO SUB 166
 150 PAUSE pa: BEEP tsh,nsh: PRINT AT xx,yy;"OBJETO: dist= ";p1: PAUSE pa: IF a-
p>=1 THEN  GO SUB 188
 154 GO SUB 200
 156 IF p<f THEN  GO SUB 300
 158 GO TO 1000
```

```
166 BEEP t1,n1: PLOT a,b-50: DRAW 0,100,-10*PI/180: DRAW 0,-100,-10*PI/180
168 PAUSE pa: BEEP te,ne: PLOT 0,b: DRAW 255,0
172 IF a+f<254 THEN  PAUSE pa: BEEP tf,nf: CIRCLE a+f,b,1
173 IF a-f-1>1 THEN  PAUSE pa/4: BEEP tf,nf: CIRCLE a-f-1,b,1
174 IF p>2*f THEN  PRINT AT 2,17;"d. focal= ";f1
176 IF p>=f AND p<2*f THEN  PRINT AT 0,15;"d. focal= ";f1
177 IF p=2*f THEN  PRINT AT 20,10;"d. focal= ";f1
178 IF p<f THEN  PRINT AT 21,8;"d. focal= ";f1
180 RETURN
188 PAUSE pa: BEEP t1,n1: PLOT a-p,b: DRAW 0,t: DRAW -1,0: DRAW -1,-1: DRAW 3,0
: DRAW -1,1: DRAW -1,0: DRAW 0,-t: PAUSE pa: RETURN
200 LET ap=a-p: IF ap<=1 THEN  LET ap=1
205 FOR n=ap TO a: BEEP t2,n2: PLOT n,c: PLOT n,d*(n+g)+c: NEXT n
230 FOR n=a+1 TO 254: IF ABS (n-INT x)<=.5 THEN  LET aa=1: LET a1=1: GO SUB 400
235 LET h=e*(n-a)+c: LET i=d*(n+g)+c: IF h<0 OR i<0 THEN  GO TO 265
240 BEEP t2,n2: PLOT n,h: PLOT n,i: NEXT n: IF a1=1 THEN  GO TO 1000
265 IF aa=0 AND p>f THEN  LET a$="NO SALE EN TV": LET a1=1: GO SUB 400
266 IF p=f THEN  LET a1=1: FOR w=n2 TO 65 STEP .2: BEEP 1/(2*w),w: NEXT w: BEEP
tsn,nsn: PRINT AT 19,0; FLASH 1;"IMAGEN EN EL INFINITO"
267 IF a1=1 THEN  GO TO 1000
270 RETURN
300 FOR n=a TO 0 STEP -1: IF ABS (n-INT x)<=.5 THEN  LET bb=1: GO SUB 450
305 LET j=e*(n-a)+c: LET k=d*(n+g)+c: IF j>175 OR k>175 THEN  GO TO 335
310 BEEP t2,n2: PLOT n,j: PLOT n,k: NEXT n
335 IF bb=0 THEN  LET a$="NO SALE EN TV": GO SUB 450
 340 RETURN
 400 IF y>=0 THEN  IF x<=254 THEN  IF x>=1 THEN  BEEP ti,ni: PLOT x,b: DRAW 0,-b
+y: DRAW 1,0: DRAW 1,1: DRAW -3,0: DRAW 1,-1: DRAW 1,0: DRAW 0,b-y
 410 IF p<>f THEN  BEEP tsh,nsh: PRINT AT xx+1,yy;"IMAGEN: real  "+a$;AT xx+2,yy
+8;"dist= ";INT (100*ta*pp+.5)/100;AT xx+3,yy+8;"invertida";AT xx+4,yy+8;b$;"
  ";INT (100*t/tt+.5)/100: RETURN
 450 IF y<=175 THEN  IF x<=254 THEN  IF x>=1 THEN  BEEP ti,ni: FOR z=b TO y STEP
 2: BEEP ti,ni: PLOT x,z: PLOT x-1,z: NEXT z: DRAW -1,-1: DRAW 4,0: DRAW -1,1
 480 BEEP tsh,nsh: PRINT AT xx+1,yy;"IMAGEN: virtual  "+a$;AT xx+2,yy+8;"dist= "
;INT (100*ta*(a-x)+.5)/100;AT xx+3,yy+8;"derecha";AT xx+4,yy+8;b$;"      ";INT (1
00*t/(y-b)+.5)/100: RETURN
 599 REM *****LENTES DIVERGENTES*****
 600 BEEP tst,nst: INPUT "distancia focal (>=1 y <=1000)?  ";f
 602 LET f1=f: LET ta=LN f/LN 10+.2: LET f=f/ta: IF f1<1 OR f1>1E3 THEN  GO TO 6
00
 605 BEEP tst,nst: INPUT "Distancia objeto/lente (>=1)?   ";p
 607 LET p1=p: LET p=p/ta: IF p1<1 THEN  GO TO 605
 610 LET a=220-f/ta: LET b=95: LET xx=17: LET yy=0: LET a$="": LET b$="menor": L
ET c=b+t: LET d=-t/p: LET g=p-a: LET e=t/f
 615 LET x=(a*f+a*p-p*f)/(p+f): LET y=(t*f+b*p+f*b)/(p+f)
 620 BEEP t1,n1: PLOT a-2,b-50: DRAW 0,100,10*PI/180: DRAW 4,0: DRAW 0,-100,10*P
I/180: DRAW -4,0
 630 PAUSE pa: BEEP te,ne: PLOT 0,b: DRAW 255,0
 632 IF a+f<=254 THEN  PAUSE pa: BEEP tf,nf: CIRCLE a+f,b,1
 633 IF a-f-1>=1 THEN  PAUSE pa/4: BEEP tf,nf: CIRCLE a-f-1,b,1
 635 PRINT AT 2,15;"d.focal= ";f1
 645 PRINT AT xx,yy;"OBJETO: dist= ";p1: PAUSE pa: IF a-p>=1 THEN  GO SUB 188
 650 LET ap=a-p: IF ap<=1 THEN  LET ap=1
 655 FOR n=ap TO a: BEEP t2,n2: PLOT n,c: PLOT n,d*(n+g)+c: NEXT n
 680 FOR n=a+1 TO 254: LET h=e*(n-a)+c: LET i=d*(n+g)+c: IF i<0 OR h>175 THEN  G
O TO 710
 700 BEEP t2,n2: PLOT n,h: PLOT n,i: NEXT n
 710 PAUSE pa: FOR n=a TO 0 STEP -1: IF ABS (n-INT x)<=.5 THEN  LET a1=1: GO SUB
 450
 720 LET h=e*(n-a)+c: IF h<0 THEN  GO TO 730
 725 BEEP t2,n2: PLOT n,h: NEXT n
 730 IF a1=0 THEN  LET a$="NO SALE EN TV": GO SUB 450
3992 PRINT #0;"          PULSA UNA TECLA          ": PAUSE 0: INPUT "": RETURN
```

QUIMICA

El programa trata de reacciones químicas de recombinación de iones. Es decir, reacciones del tipo:

R1 + R2 → P1 + P2

donde R1,R2 son los reactivos y P1,P2 son los productos. Cada uno de ellos puede ser un ácido, una base o una sal.

El ordenador te plantea el miembro de la izquierda en la reacción anterior, y por recombinación de iones debes responder los dos productos. El programa consta de los cinco bloques siguientes (ver listado adjunto):

1. Almacena los cationes, su valencia, los aniones y su valencia. Se han elegido 43 cationes y 26 aniones.
2. Elige aleatoriamente (línea 45) el primer reactivo y luego (líneas 70 y 73), el segundo (ver fig. 7).
3. Solicita al usuario los productos de la reacción, que se pueden dar en cualquier orden. Hay que seguir la norma de escribir primero el catión y luego el anión.
4. Genera el verdadero producto Pl (a partir de la línea 120), y luego el P2 (a partir de la 160).
5. Comprueba la respuesta (líneas 205 y 210), dando la solución si has fallado (ver fig. 8).

Hay la posibilidad de continuar o no. Al final te dice cuántas reacciones has tenido bien.

En los ejemplos de las figuras, los reactivos han sido:

fig. 7: SAL + BASE =
fig. 8: SAL + SAL =
fig. 9: BASE + ACIDO =

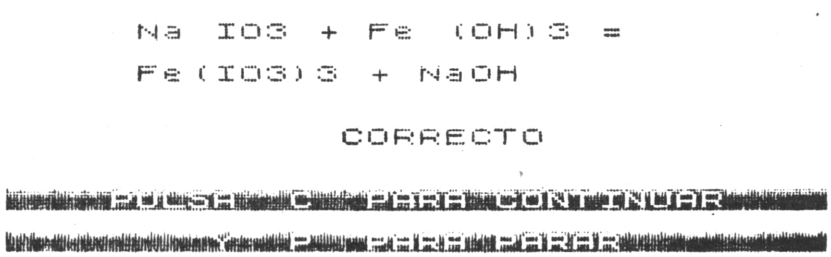


fig. 7

```
Li IO3 + Sn I2 =
LiI + Sn(IO3)3
   NO ES CORRECTO

LOS PRODUCTOS SON
Li I + Sn (IO3)2

PULSA C PARA CONTINUAR
Y P PARA PARAR
```

fig. 8

```
Ca (OH)2 + H3 PO4 =
Ca PO4 + H2O
   NO ES CORRECTO

LOS PRODUCTOS SON
Ca3 (PO4)2 + H2O

PULSA C PARA CONTINUAR
Y P PARA PARAR
```

fig. 9

```
  1 REM   REACCIONES  QUIMICAS
 10 REM      numero de iones
 13 LET nc=43: LET na=26: LET p=0: LET v=0: LET w=0
 15 DIM c$(nc,2): DIM a$(na,4): DIM c(nc): DIM a(na)
 20 CLS : BORDER 1: PAPER 6: INK 0: CLS
 30 FOR i=1 TO nc: READ c$(i),c(i): NEXT i
 35 FOR i=1 TO na: READ a$(i),a(i): NEXT i
 40 CLS : LET x$="": LET y$=""
 45 LET s1=INT (RND*nc)+1: LET t1=INT (RND*na)+1: IF c$(s1)="H " AND a$(t1)="OH
 " THEN  GO TO 45
 48 REM      primer  reactivo
 50 LET w=w+1: PRINT AT 5,5;c$(s1);: IF c$(s1,2)=" " THEN  PRINT CHR$ 8;
 53 IF a(t1)<>-1 AND -a(t1)<>c(s1) THEN  PRINT -a(t1);
 58 PRINT " ";: IF c(s1)<>1 AND -a(t1)<>c(s1) AND (a$(t1,4)<>" " OR a$(t1,3)<>"
") THEN  PRINT "(";: LET p=1
 60 IF c(s1)<>1 AND a$(t1)="OH  " THEN  PRINT "(";: LET p=1
 62 PRINT a$(t1);: IF a$(t1,4)=" " THEN  PRINT CHR$ 8;: IF a$(t1,3)=" " THEN  P
RINT CHR$ 8;: IF a$(t1,2)=" " THEN  PRINT CHR$ 8;
 64 IF p=1 THEN  PRINT ")";: LET p=0
 66 IF c(s1)<>1 AND -a(t1)<>c(s1) THEN  PRINT c(s1);
 68 PRINT " + ";
 69 REM      segundo  reactivo
 70 LET s2=INT (RND*nc)+1: IF c$(s1)=c$(s2) THEN  GO TO 70
 73 LET t2=INT (RND*na)+1: IF a$(t1)=a$(t2) THEN  GO TO 73
 75 IF c$(s2)="H " AND a$(t2)="OH  " THEN  GO TO 73
 77 IF c(s1)=4 AND a(t2)=-2 THEN  LET c(s1)=2: LET a(t2)=-1
 79 IF c(s2)=4 AND a(t1)=-2 THEN  LET c(s2)=2: LET a(t1)=-1
 81 PRINT c$(s2);: IF c$(s2,2)=" " THEN  PRINT CHR$ 8;
 83 IF a(t2)<>-1 AND -a(t2)<>c(s2) THEN  PRINT -a(t2);
 85 PRINT " ";: IF c(s2)<>1 AND -a(t2)<>c(s2) AND (a$(t2,4)<>" " OR a$(t2,3)<>"
") THEN  PRINT "(";: LET p=1
 88 IF c(s2)<>1 AND a$(t2)="OH  " THEN  PRINT "(";: LET p=1
 90 PRINT A$(t2);: IF a$(t2,4)=" " THEN  PRINT CHR$ 8;: IF a$(t2,3)=" " THEN  P
RINT CHR$ 8;: IF a$(t2,2)=" " THEN  PRINT CHR$ 8;
 93 IF p=1 THEN  PRINT ")";: LET p=0
 95 IF c(s2)<>1 AND -a(t2)<>c(s2) THEN  PRINT c(s2);
 98 PRINT " = ": PRINT
103 REM      respuestas
105 INPUT "PRIMER PRODUCTO ? ";P$: PRINT TAB 5;p$+" + ";
110 INPUT "SEGUNDO  PRODUCTO ? ";q$: PRINT q$
118 REM   generacion producto 1
120 LET m$=c$(s1,1): IF c$(s1,2)<>" " THEN  LET m$=m$+c$(s1,2)
126 IF a(t2)<>-1 AND -a(t2)<>c(s1) THEN  LET m$=m$+STR$ (-a(t2))
130 IF c(s1)<>1 AND -a(t2)<>c(s1) AND a$(t2,3)<>" " THEN  LET x$=x$+"(": LET p=
1
133 IF c(s1)<>1 AND a$(t2)="OH  " THEN  LET x$="(": LET p=1
135 LET x$=x$+a$(t2,1)
140 IF a$(t2,2)<>" " THEN  LET x$=x$+a$(t2,2): IF a$(t2,3)<>" " THEN  LET x$=x$
+a$(t2,3): IF a$(t2,4)<>" " THEN  LET x$=x$+a$(t2,4)
142 IF p=1 THEN  LET x$=x$+")": LET p=0
145 IF c(s1)<>1 AND -a(t2)<>c(s1) THEN  LET x$=x$+STR$ c(s1)
150 LET y$=m$+x$: LET w$=m$+" "+x$: IF w$="H OH" THEN  LET w$="H2O"
158 REM   generacion producto 2
160 LET n$=c$(s2,1): IF c$(s2,2)<>" " THEN  LET n$=n$+c$(s2,2)
166 IF a(t1)<>-1 AND -a(t1)<>c(s2) THEN  LET n$=n$+STR$ (-a(t1))
170 IF c(s2)<>1 AND -a(t1)<>c(s2) AND a$(t1,3)<>" " THEN  LET y$=y$+"(": LET p=
1
173 IF c(s2)<>1 AND a$(t1)="OH  " THEN  LET y$=y$+"(": LET p=1
175 LET y$=y$+a$(t1,1)
180 IF a$(t1,2)<>" " THEN  LET y$=y$+a$(t1,2): IF a$(t1,3)<>" " THEN  LET y$=y$
+a$(t1,3): IF a$(t1,4)<>" " THEN  LET y$=y$+a$(t1,4)
182 IF p=1 THEN  LET y$=y$+")": LET p=0
185 IF c(s2)<>1 AND -a(t1)<>c(s2) THEN  LET y$=y$+STR$ c(s2)
190 LET r$=n$+y$: LET s$=n$+" "+y$: IF s$="H OH" THEN  LET s$="H2O"
200 REM    comprobacion
205 IF (p$=v$ OR p$=w$) AND (q$=r$ OR q$=s$) THEN  GO TO 250
210 IF (p$=r$ OR p$=s$) AND (q$=v$ OR q$=w$) THEN  GO TO 250
220 PRINT : PRINT TAB 8; FLASH 1;"NO  ES  CORRECTO": PRINT
```

```
 230  PRINT : PRINT TAB 5;"LOS  PRODUCTOS  SON": PRINT
 240 PRINT TAB 5;w$+" + "+s$: GO TO 300
 250 PRINT : PRINT : PRINT TAB 13; FLASH 1;"CORRECTO": LET V=V+1
 300 PRINT : PRINT : PRINT  INVERSE 1;"    PULSA  C  PARA CONTINUAR";TAB 32
 310 PRINT : PRINT  INVERSE 1;"         Y  P  PARA PARAR";TAB 32
 330 IF INKEY$="c" THEN  GO TO 40
 340 IF INKEY$="p" THEN  CLS : PAPER 1: BORDER 1: INK 5: CLS : PRINT AT 9,11;"PU
NTUACION :": PRINT : PRINT " DE ";W;" REACCIONES,HAS ACERTADO ";V: STOP
 360 GO TO 330
 500 DATA "Li",1,"Na",1,"H",1,"K",1,"Cs",1,"Be",2,"Mg",2,"H",1,"Co",2,"Co",3,"H"
,1,"Pb",2,"Pb",4,"Ba",2,"Ca",2,"H",1,"Cu",1,"Cu",2,"V",3,"H",1,"V",5,"Zn",2,"H",
1,"Cr",2,"Cr",3,"Fe",2,"Fe",3,"H",1,"Ag",1,"Cd",2,"Al",3,"H",1,"Ni",2,"Ni",3,"H"
,1,"Au",1,"Au",3,"Hg",1,"H",1,"Hg",2,"Sn",2,"Sn",4,"H",1
 520 DATA "IO4",-1,"IO3",-1,"Cl",-1,"Br",-1,"OH",-1,"I",-1,"S",-2,"CO3",-2,"SiO3
",-2,"OH",-1,"NO3",-1,"NO2",-1,"PO4",-3,"PO3",-3,"OH",-1,"SO4",-2,"SO3",-2,"ClO4
",-1,"ClO3",-1,"OH",-1,"IO2",-1,"AsO4",-3,"AsO3",-3,"OH",-1,"BrO3",-1,"BrO2",-1
```

BIOLOGIA

En este programa se estudia el crecimiento en términos numéricos de un gen recesivo, A_2, frente a otro dominante, A_1. Vamos a suponer mutaciones recurrentes, o sea, sistemáticas, del tipo $A_1 \rightarrow A_2$, que pueden ser de dos tipos:

— Irreversibles, cuando solamente ocurren en el sentido $A_1 \rightarrow A_2$.

— Reversibles, cuando se da simultáneamente la mutación $A_2 \rightarrow A_1$.

Mutación irreversible.—Según los libros de Biología, si u es la velocidad de la mutación $A_1 \rightarrow A_2$ por generación,

$$u = \frac{\text{n.º de genes que mutan a } A_2}{\text{n.º de genes totales}}$$

y si q_0 es la frecuencia o porcentaje inicial del gen A_2, la frecuencia de este gen al cabo de n generaciones es:

$$q_n = 1 - \frac{1 - q_o}{e^{un}}$$

El programa pide los datos de u y de q_0 y dibuja el crecimiento del gen A_2 en función del número n de generaciones transcurridas. Véase la fig. 10, que son tres crecimientos a efectos de comparación, con q_0=0,1 siempre fijo, y u=1E-5,4E-5 y 1E-4, sucesivamente.

Mutaciones reversibles.—Sean las velocidades:

u, para la mutación $A_1 \rightarrow A_2$

v, para la mutación $A_2 \rightarrow A_1$

Entonces, si llamamos a

$\hat{q} = \frac{u}{u + v}$, la frecuencia del gen A_2 al cabo de n generaciones es:

$$q_n = \hat{q} + \frac{q_o - \hat{q}}{e^{(u+v)n}}$$

siendo, como antes, q_0 su frecuencia inicial.

El programa pide los datos de u,v y q, dibujando entonces la evolución del gen A_2. Se han elegido los dos ejemplos siguientes:

fig.	vel. u	vel. v	frec. q_o
11	2E-4	1E-4 4E-4 6E-4	0,05
12	0,5E-4 1E-4 2E-4	2E-4	0,15

Es instructivo ver la dependencia de la evolución de un gen respecto de cada uno de sus factores causantes, y con este programa se puede hacer fácilmente. Cada una de las partes del mismo está comentada en el listado adjunto.

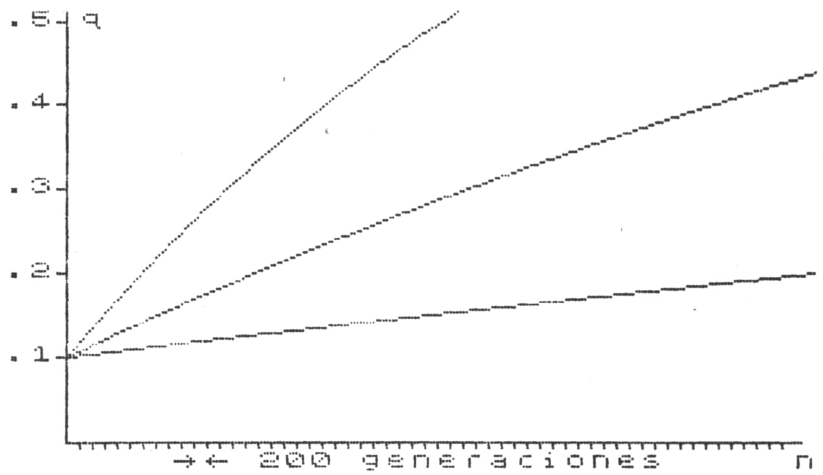


fig. 10

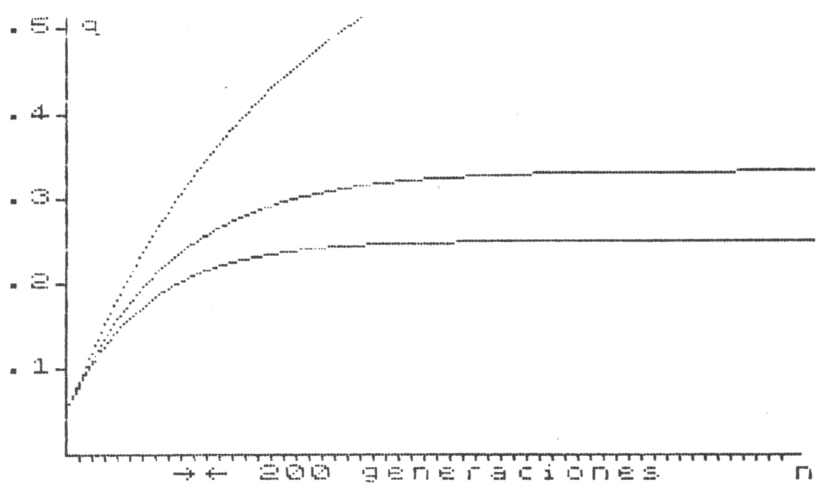


fig. 11

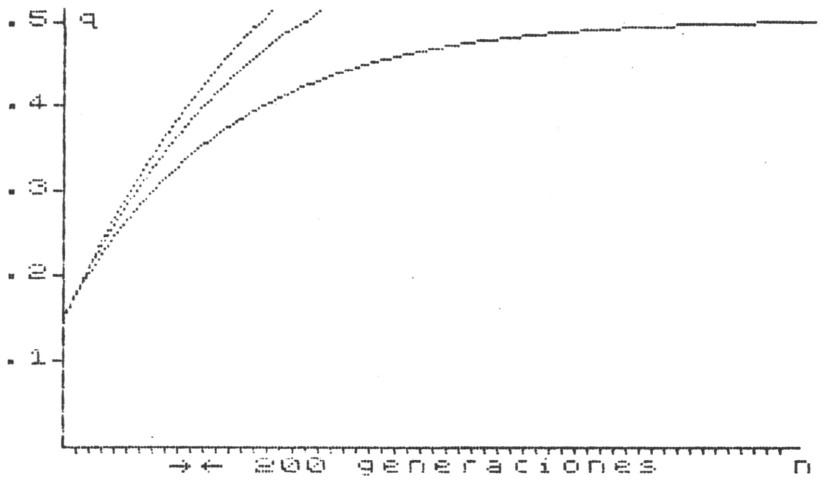


fig. 12

```
 10 REM     ***   CRECIMIENTO  DE  UN  GEN  RECESIVO   ***
 13
 20 BORDER 3: PAPER 1: INK 7: CLS
 30 PRINT AT 5,10;"TIPO MUTACION :",,,"1-irreversible",,"2-reversible"
 40 PRINT AT 16,9;"TECLEA OPCION"
 50 LET a$=INKEY$
 60 IF a$<>"1" AND a$<>"2" THEN  GO TO 50
 80 CLS : GO SUB 1000
100 GO TO 300*VAL a$
299
300 REM  ***  MUTACION  IRREVERSIBLE  ***
310 INPUT "vel. A1->A2?(0-0.001)";u
320 IF u>1E-3 THEN  GO TO 310
330 INPUT "frec. inicial.de A2?(0-0.3)";q
340 IF q>.3 THEN  GO TO 330
360 FOR n=0 TO 11700 STEP 50
370 LET qn=1-(1-q)/EXP (u*n)
380 IF qn*320>164 THEN  GO TO 420
390 PLOT n/50+20,qn*320+10
400 NEXT n
420 GO SUB 2000: GO TO 310
599
600 REM  ***   MUTACION  REVERSIBLE   ***
610 INPUT "veloc. A1->A2?(0-0.001)";u
620 IF u>1E-3 THEN  GO TO 610
630 INPUT "vel. A2->A1?(0-0.001)";v
640 IF v>1E-3 THEN  GO TO 630
660 INPUT "frec. inicial de A2?(0-0.3)";q
700 LET qb=u/(u+v)
710 FOR n=0 TO 11700 STEP 50
```

```
 720 LET qn=qb+(q-qb)/EXP ((u+v)*n)
 730 IF qn*320>164 THEN  GO TO 780
 750 PLOT n/50+20,qn*320+10
 760 NEXT n
 780 GO SUB 2000: GO TO 610
 999
1000 REM   ***   DIBUJO  DE  EJES   ***
1010 PLOT 20,175: DRAW 0,-165: DRAW 230,0
1030 FOR i=24 TO 244 STEP 4
1040 PLOT i,10: DRAW 0,-2: NEXT i
1050 PRINT AT 21,31;"n"
1060 FOR i=10+32 TO 170 STEP 32
1070 PLOT 20,i: DRAW -3,0: NEXT i
1080 PRINT AT 16,0;".1";AT 12,0;".2";AT 8,0;".3";AT 4,0;".4";AT 0,0;".5"
1090 PLOT 54,3: DRAW 6,0: DRAW -2,2: DRAW 2,-2: DRAW -2,-2
1110 PLOT 70,3: DRAW -6,0: DRAW 2,2: DRAW -2,-2: DRAW 2,-2
1120 PRINT AT 0,3;"q";AT 21,10;"200 generaciones"
1130 RETURN
1999
2000 REM     ***   MAS  GRAFICAS   ***
2010 INPUT "otra grafica?(s/n)";z$
2020 IF z$<>"S" AND z$<>"s" THEN  STOP
2050 RETURN
```

INGLES

Como último programa vamos a comentar uno que nos servirá como agenda de verbos irregulares en inglés, así como de entrenamiento en traducción.

Una vez almacenados los tres tiempos en inglés (presente, pasado y participio), así como su traducción, el programa presenta un menú de tres opciones (ver fig. 13):

1. **verbo regular o irregular**: hay que dar un infinitivo en inglés. El programa busca entonces entre los verbos almacenados, y si no lo encuentra lo toma como un verbo regular. Si está almacenado, da los tres tiempos (ver fig. 14).
2. **repaso** de pasado y participio: el ordenador elige al azar un verbo y un tiempo, que puede ser pasado o participio. Lo muestra en pantalla y pide el tiempo y el infinitivo en inglés (ver fig. 15).

El convenio es el siguiente: asignamos un 2 al pasado y un 3 al participio. Así, para dar la respuesta se teclea primero el número correspondiente y después, sin dejar espacios en blanco, el infinitivo. Osea, las respuestas deben empezar por un 2, por un 3 o por un 23 (para el caso que coincidan el pasado y el participio).

3. **traducción** al inglés: el ordenador elige cualquier tiempo de cualquier verbo. Solicita la traducción al inglés, dando el infinitivo en castellano (ver fig. 16).

Características: Las respuestas deben darse exactamente, por ser una exigencia en la comparación entre cadenas. Es decir, como sabes, no es lo mismo teclear «elegir» que «Elegir». Si has fallado en algo, el ordenador te da la respuesta correcta, indicando dónde está el fallo.

Por brevedad sólo se han almacenado 20 verbos (hay 175 verbos irregulares). Si queremos ampliar la lista no tienes más que cambiar la variable n en la línea 700. y procurar que ninguna cadena sobrepase 10 caracteres.

```
    1 REM   *** VERBOS IRREGULARES EN INGLES ***
   10 DATA "SER","BE","WAS","BEEN","GOLPEAR","BEAT","BEAT","BEATEN"
   20 DATA "CAMBIAR","BECOME","BECAME","BECOME","ATAR","BITE","BIT","BITTEN"
   30 DATA "SOPLAR","BLOW","BLEW","BLOWN","ROMPER","BREAK","BROKE","BROKEN"
   40 DATA "TRAER","BRING","BROUGHT","BROUGHT","EDIFICAR","BUILD","BUILT","BUILT"
   50 DATA "ARDER","BURN","BURNT","BURNT","COMPRAR","BUY","BOUGHT","BOUGHT"
   60 DATA "ELEGIR","CHOOSE","CHOSE","CHOSEN","TRATAR","DEAL","DEALT","DEALT"
   70 DATA "BEBER","DRINK","DRANK","DRUNK","COMER","EAT","ATE","EATEN"
   80 DATA "CAER","FALL","FELL","FALEN","SENTIR","FEEL","FELT","FELT"
   90 DATA "LUCHAR","FIGHT","FOUGHT","FOUGHT","PERDONAR","FORGIVE","FORGAVE","FOR
GIVEN"
  100 DATA "IR","GO","WENT","GONE","CRECER","GROW","GREW","GROWN"
  695 REM     ALMACENAMIENTO DE LOS VERBOS
  700 POKE 23658,8: LET n=20: REM  n ES NUMERO DE VERBOS A ALMACENAR
  720 DIM e$(n,10): DIM a$(n,3,10): DIM z$(3,10)
  740 FOR i=1 TO n
  760 READ e$(i): FOR j=1 TO 3: READ a$(i,j): NEXT j
  780 NEXT i
  800 LET p$="PRESENTE   PASADO    PARTICIPIO"
  810 LET r$=" ES UN VERBO REGULAR"
  820 LET Z$(1)="PRESENTE": LET Z$(2)="PASADO": LET Z$(3)="PARTICIPIO"
  900 CLS : PRINT AT 5,5;"OPCIONES:"
  910 PRINT : PRINT "1-verbo regular o irregular"
  920 PRINT : PRINT "2-repaso de pasado y participio"
  930 PRINT : PRINT "3-traduccion al ingles"
  950 INPUT "OPCION ? ";h
  960 IF h<>INT h OR h<1 OR h>3 THEN  GO TO 950
  990 CLS : GO SUB 1000*h: GO TO 900
  995 REM        VERBO  REGULAR  O  IRREGULAR
 1000 INPUT "INFINITIVO EN INGLES?";i$
 1020 IF i$="M" THEN  RETURN
 1040 LET j$=i$: FOR i=LEN i$ TO 9: LET j$=j$+" ": NEXT i
 1060 FOR i=1 TO n
 1080 IF a$(i,1)=j$ THEN  GO TO 1500
 1090 NEXT i
 1100 PRINT : PRINT i$+r$: GO TO 1000
 1500 PRINT : PRINT p$: PRINT a$(i,1);TAB 11;a$(i,2);TAB 21;a$(i,3)
 1600 GO TO 1000
 1995 REM        PASADO  Y  PARTICIPIO
 2000 LET f=INT (RND*n)+1: LET c=INT (RND*2)+2
 2020 PRINT : PRINT a$(f,c);":TIEMPO E INFINITIVO  EN  INGLES?  ";
 2040 INPUT i$: PRINT i$
 2050 IF i$="M" THEN  RETURN
 2060 IF i$(1 TO 2)="23" THEN  GO SUB 7000: GO TO 2000
 2080 IF i$(1)="2" THEN  GO SUB 5000: GO TO 2000
 2100 IF i$(1)="3" THEN  GO SUB 6000: GO TO 2000
 2120 PRINT  INVERSE 1;" DAME LA RESPUESTA SEGUN REGLAS ": GO TO 2020
 2995 REM        TRADUCION  AL  INGLES
 3000 LET f=INT (RND*n)+1: LET c=INT (RND*3)+1
 3020 PRINT : PRINT z$(c);" del verbo ";e$(f)
 3040 INPUT i$: PRINT TAB 7;i$;
 3060 IF i$="M" THEN  RETURN
 3080 LET j$=i$: FOR i=LEN i$ TO 9: LET j$=J$+" "
 3100 NEXT i
 3120 FOR i=1 TO n
 3140 IF j$=a$(f,c) THEN  PRINT TAB 20;"CORRECTO!": GO TO 3000
 3160 NEXT i
 3180 PRINT TAB 26;"FALLO.La respuesta es ";a$(f,c): GO TO 3000
 5000 LET j$=i$(2 TO ): FOR i=LEN i$ TO 10: LET j$=j$+" ": NEXT i
 5020 IF c=2 AND j$=a$(f,1) THEN  PRINT TAB 17;"CORRECTO": RETURN
 5040 IF c=3 AND j$=a$(f,1) THEN  PRINT : PRINT "MAL.Era el participio de ";a$(f,
1): RETURN
 5060 PRINT "MAL.El infinitivo es ";a$(f,1): RETURN
 5999
 6000 LET j$=i$(2 TO ): FOR i=LEN i$ TO 10: LET j$=j$+" ": NEXT i
 6020 IF c=3 AND j$=a$(f,1) THEN  PRINT TAB 17;"CORRECTO": RETURN
```

```
6040 IF c=2 AND j$=a$(f,1) THEN  PRINT "MAL.Era el pasado de ";a$(f,1): RETURN
6060 PRINT "MAL.El infinitivo es ";a$(f,1): RETURN
6999
7000 LET j$=i$(3 TO ): FOR i=LEN i$ TO 11: LET j$=j$+" ": NEXT i
7020 IF a$(f,2)=a$(f,3) AND j$=a$(f,1) THEN  PRINT TAB 17;"CORRECTO": RETURN
7040 IF a$(f,2)=a$(f,3) THEN  PRINT "FALLO en traduccion": PRINT "Es el verbo ";
a$(f,1): RETURN
7060 IF j$=a$(f,1) THEN  PRINT ";fallo en el tiempo": PRINT "El pasado es ";a$(f
,2): PRINT "Y el participio es ";a$(f,3): RETURN
7080 PRINT TAB 20; INVERSE 1;"TODO MAL": PRINT a$(f,c);"es el ";
7100 IF a$(f,2)=a$(f,3) THEN  PRINT "PASADO y PARTICIPIO de ";a$(f,1): RETURN
7150 PRINT z$(c);" de ";a$(f,1): RETURN
```

```
      OPCIONES:
1-verbo regular o irregular
2-repaso de pasado y participio
3-traduccion al ingles
```

fig. 13

```
LOVE ES UN VERBO REGULAR

PRESENTE    PASADO      PARTICIPIO
BURN        BURNT       BURNT

PRESENTE    PASADO      PARTICIPIO
DRINK       DRANK       DRUNK

PRINT ES UN VERBO REGULAR

PRESENTE    PASADO      PARTICIPIO
FIGHT       FOUGHT      FOUGHT

FORGET ES UN VERBO REGULAR
```

fig. 14

```
FORGAVE     :TIEMPO E INFINITIVO
 EN   INGLES?   3FORGIVE
MAL.Era el pasado de FORGIVE

BEATEN      :TIEMPO E INFINITIVO
 EN   INGLES?   23BEAT
:fallo en el tiempo
El pasado es BEAT
Y el participio es BEATEN

BROUGHT     :TIEMPO E INFINITIVO
 EN   INGLES?   23BRING
                CORRECTO

ATE         :TIEMPO E INFINITIVO
 EN   INGLES?   2EAT
                CORRECTO

FOUGHT      :TIEMPO E INFINITIVO
 EN   INGLES?
```

fig. 15

```
PARTICIPIO del verbo TRAER
      BROUGHT          CORRECTO!

PRESENTE   del verbo LUCHAR
      FIGHT            CORRECTO!

PARTICIPIO del verbo TRATAR
      DEAL                    FALLO.
La respuesta es DEALT

PRESENTE   del verbo COMER
      EAT              CORRECTO!

PARTICIPIO del verbo CAER
      FELL                    FALLO.
La respuesta es FALEN

PASADO     del verbo EDIFICAR
      BUILT            CORRECTO!
```

fig. 16

APENDICE

A fin de que sean trasladables las sentencias y, en consecuencia, los programas descritos en el presente libro, se dan las indicaciones y sugerencias adecuadas para ordenadores compatibles PC en los lenguajes GWBASIC y BASICA.

— La palabra clave LET no es necesario ponerla.

— No es necesaria la variable del bucle después de la sentencia NEXT (asume la de inicialización del bucle).

— Todas las variables se inicializan a 0 tras el RUN.

— Para fraccionamiento de cadenas, ver pág. 71.

— La sentencia DIM puede dimensionar varias matrices si éstas se separan por comas.

— Todas las funciones matemáticas y de cadena han de tener los argumentos siempre entre paréntesis.

— La opción de impresión dada por PRINT AT f,c;"HOLA" se sustituye por

LOCATE f,c:PRINT "HOLA"

— La coma de tabulación salta 14 espacios.

— La sentencia PAUSE es conveniente sustituirla por un bucle FOR...NEXT en vacío (1000 iteracciones $\simeq$ 1 seg).

— Es necesario especificar el modo de pantalla al inicio de programa:
para texto de 40 columnas: SCREEN 1,0,0
para texto de 80 columnas: SCREEN 2,0,0
para gráficos (320×200): SCREEN 1,0,0

— La sentencia PLOT tiene su equivalente por la PSET. Ahora bien, PLOT tiene el origen en el ángulo inferior izquierdo, mientras PSET lo tiene en el superior izquierdo; por ello,

PLOT x,y → PSET (x,175-y)

— La sentencia DRAW se sustituye por la LINE

DRAW a,b → LINE (x_1,y_1)-(x_2,y_2)

donde $a = x_2 - x_1$
$b = y_2 - y_1$

— La sentencia CIRCLE tiene varias expresiones:
para circunferencias completas:

CIRCLE (x,y),r

para arcos:

CIRCLE (x,y),r,c,p,f

donde x,y son las coordenadas del centro.
r es el tamaño del radio en pixels.
c es el color (0 a 3).
p es la posición del comienzo del dibujo, en radianes, y tomados a partir de las 3 horas del reloj.
f es la posición final del arco con las características del parámetro anterior.

— La sentencia de sonido BEEP se sustituye por SOUND:

BEEP t,f → SOUND f,t

donde, para compatibles PC, f es la frecuencia y va de 37 a 32767 Hz, t es el tiempo, de 0 a 65535 impulsos, siendo la duración de un impulso 1/18,2 seg. en el GWBASIC.

Todas las sentencias no comentadas aquí permanecen inalterables.

BASIC

CURSO DE PROGRAMACION CON APLICACIONES DIDACTICAS

El BASIC es un lenguaje de programación muy popular y adaptable a muchas aplicaciones.

Este libro, fruto de varios años de práctica de dos profesores de Instituto, cubre dos aspectos:

- La enseñanza de este lenguaje.
- Su aplicación a programas didácticos.

Este libro contiene programas comentados, proposición de programas a realizar por el lector, interesantes rutinas en código máquina para Z80—A y un apéndice de conversión de las sentencias necesarias para IBM y Compatibles.

ISBN: 84-86381-34-7

ra-ma