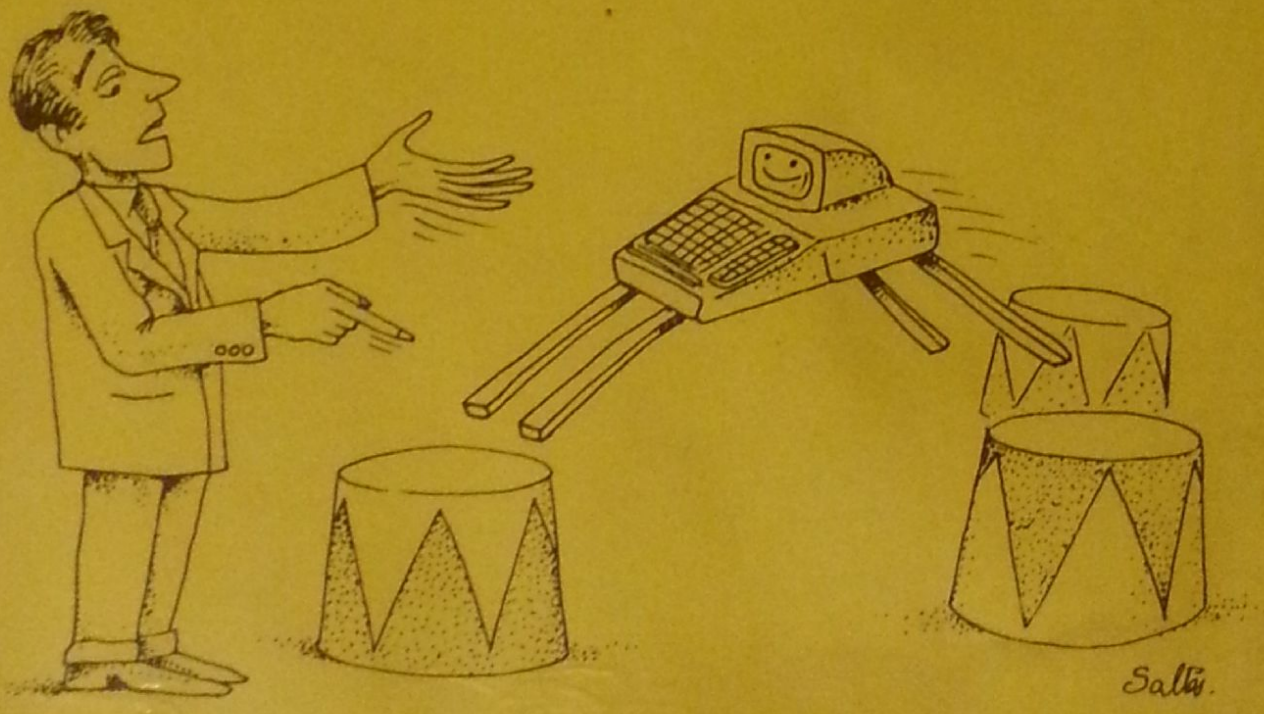
RICARDO AGUADO-MUÑOZ
AGUSTIN BLANCO
ENRIQUE RUBIALES
JAVIER ZABALA
RICARDO ZAMARREÑO

PROGRAMAS COMENTADOS DE

BASIC BASICO



De los mismos autores:

BASIC BASICO
CURSO DE PROGRAMACION

Es un libro de texto o divulgativo, fruto de la experiencia didáctica de tres años en el Instituto Piloto «Cardenal Herrera Oria».

Explica la programación en BASIC. Muchos de los ejercicios que propone se resuelven en el libro Programas Comentados de BASIC BASICO.

BASIC JUNIOR
INICIACION A LA PROGRAMACION

Está destinado a escolares y principiantes.

Es formativo y ameno. Tiene numerosos ejemplos divertidos.

BASIC JUNIOR te invita a jugar el apasionante juego de la programación de ordenadores.

PROGRAMAS EXPLICADOS
DE BASIC JUNIOR

Resuelve los cien programas propuestos en el libro BASIC JUNIOR. Iniciación a la Programación.

Además de ofrecer los cien listados, comprobados en ordenador, explica los pasos más importantes o de mayor dificultad de cada programa.

PROGRAMAS COMENTADOS DE

BASIC BASICO

RICARDO AGUADO-MUÑOZ
AGUSTIN BLANCO
ENRIQUE RUBIALES
JAVIER ZABALA
RICARDO ZAMARREÑO

DIRECCION DE LOS AUTORES:

JAVIER ZABALA
Vitrubio, 3
28006 MADRID

RICARDO ZAMARREÑO
Bristol, 10 - 9.° izda.
28028 MADRID

DISTRIBUYE:

GRUPO DISTRIBUIDOR EDITORIAL, S. A.
Don Ramón de la Cruz, 67
28001 MADRID
Tfno: 91-401 12 00

1.ª edición: diciembre 1983
2.ª edición: febrero 1984
3.ª edición: marzo 1984
4.ª edición: mayo 1984
5.ª edición: octubre 1984
6.ª edición: marzo 1985
7.ª edición: mayo 1985

Editan:
© Ricardo Aguado-Muñoz, Agustín Blanco, Enrique Rubiales, Javier Zabala y Ricardo Zamarreño
I.S.B.N.: 84-938-0330-3
Depósito legal: M-16167-1985
Imprime: Hijos de E. Minuesa, S. L. Ronda de Toledo, 24. 28005 Madrid.
Fotocompone: JALME, S. A. Juan de Olías, 12. 28020 Madrid.

INDICE

PRESENTACION

La colección de programas que te ofrecemos tiene un fin didáctico: que aprendas a programar ordenadores. Cada enunciado debe ser un desafío. Estás obligado a escribir un programa que funcione y a compararlo con el que te ofrecemos; y si no coincide con la solución propuesta, ¡tanto mejor!; y si, además, es más corto, más claro, más elegante y más eficaz que el nuestro, entonces alégrate y siente la íntima satisfacción del artista.

Porque, efectivamente, la programación de ordenadores, amén de ser una disciplina mental que desarrolla los hábitos de razonamiento lógico y la capacidad de análisis de los problemas, es también un arte; y, como tal, su plasmación en un ordenador refleja la manera de pensar, de razonar y, sobre todo, de imaginar del programador.

Frente a la visión terrorífica del *ordenador-que-nos-controla* y frente a esa imagen publicitaria del *ordenador-que-todo-lo-hace,* nosotros queremos insistir en el *ordenador como instrumento de expresión científica, artística y lúdica.* Y puesto que el ordenador va a entrar en las escuelas y en los hogares, queremos que nuestros niños y jóvenes dominen al ordenador, no que sean dominados por él. Pensando en ellos hemos escrito este libro. Si tú, amigo lector, ya no eres un chaval, haz como si lo fueras, de lo contrario no entrarás en el reino de la programación.

Con estos deseos hemos decidido recoger y editar la presente colección de *Programas comentados de BASIC BASICO.* Y, también, con la esperanza de que sea acogido por el público con la misma simpatía y benevolencia que nuestro anterior BASIC BASICO *Curso de programación,* al cual responden muchos de los programas aquí comentados.

LOS AUTORES

ADVERTENCIAS

PROGRAMAS "TODO TERRENO"

Hemos pretendido que los programas "funcionen" en cualquiera de los ordenadores existentes en el mercado. Esta pretensión de universalidad hace que algunos de ellos no sean los ideales para un aparato concreto. Apelamos a tu inteligencia y buena voluntad para que, utilizando las ventajas de *tu* ordenador, des soluciones más simples y elegantes a los problemas aquí propuestos.

REFERENCIAS AL LIBRO "BASIC BASICO"

Cada programa de este libro tiene un nombre. Muchos de ellos van seguidos de dos números entre paréntesis. Por ejemplo: TAQUILLA AUTOMATICA (4.24). Esto indica que el programa responde al ejercicio 4.24 del libro BASIC BASICO. CURSO DE PROGRAMACION.

MEJORA DE LOS PROGRAMAS

Hemos procurado presentar programas cortos y no repetir en exceso ciertas rutinas. Por esta razón, quizá haya programas que puedan ser mejorados por el lector con la simple adición de algunas líneas, que encontrará en otros programas o en el apartado de RUTINAS DE USO FRECUENTE.

CUADRO DE INSTRUCCIONES EMPLEADAS

Al final del libro presentamos una tabla, indicando qué instrucciones fundamentales se manejan en cada programa. Hemos omitido aquellas que, como PRINT e INPUT, son utilizadas en casi todos los programas.

RUTINAS

En las páginas 23, 24, 25 y 26, recogemos las RUTINAS DE USO FRECUENTE, que unas veces hemos utilizado y otras no, por no recargar los programas.

LONGITUD DE LOS PROGRAMAS

Por motivos didácticos obvios, hemos preferido programas cortos, en los que resalte más claramente la instrucción o construcción lógica que se pretende ilustrar. De todas maneras, también presentamos algunos programas largos, ya que al crecer la longitud, la complejidad crece más de lo que cabría esperar. Este tipo de programas presenta dificultades de estructuración que no se pueden observar en programas cortos.

ORDEN DE DIFICULTAD

Dentro de cada capítulo se ha procurado que los programas se presenten en un orden de dificultad creciente.

EXPLICACIONES

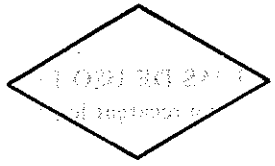
En todos los programas ofrecemos explicaciones suficientes para poder diseñarlos. Estas explicaciones van, unas veces, en lenguaje coloquial; otras, en forma de algoritmo; y otras, en un diagrama de flujo.

SIMBOLOS EN LOS DIAGRAMAS DE FLUJO

Los símbolos que hemos empleado en los diagramas de flujo son los siguientes:

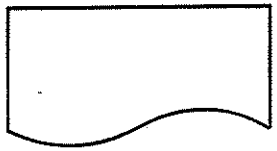
Principio y fin de algoritmo.

Operación o asignación de valor a una variable.

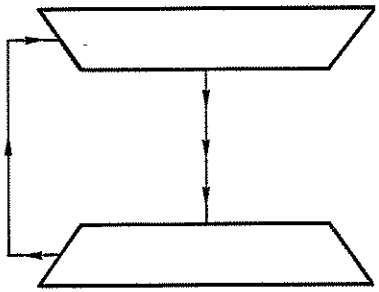


Comparación.

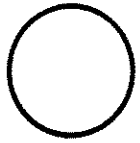
Entrada de valores.



Salida de valores (en Informática se suele emplear sólo para salida de valores a través de la impresora).



Bucle. La secuencia de instrucciones entre estos dos trapecios se repite un cierto número de veces.



Subrutinas o conexión entre segmentos de programa.

(BORRAR LA PANTALLA)

Así indicamos en los programas la instrucción que borra la pantalla y lleva el cursor a la primera posición de la primera línea. El lector tiene que sustituirla por alguna de estas líneas:

```
20 CLR
20 CLS
20 CLEAR
```

o por otra parecida, según el ordenador.

(CURSOR AL ORIGEN)

Así indicamos en algún programa la instrucción que lleva el cursor a la primera posición de la primera línea. Es innecesaria en aquellos ordenadores

que, mediante PRINT AT A,B o TAB(A,B), pueden escribir en cualquier lugar de la pantalla.

DIMENSIONES DE LA PANTALLA

El número de líneas de la pantalla y el de caracteres por línea varían de unos ordenadores a otros. También es variable el número de zonas verticales (dos, tres o cuatro) en que queda dividida la pantalla a efectos de impresión con comas. Casi todos los programas de este libro están pensados para una pantalla de 25 líneas y 40 caracteres, que se divide en cuatro zonas verticales. El lector deberá tener esto en cuenta para adaptar los programas a su ordenador, caso de ser necesario.

Los programas de gráficos en baja resolución están desarrollados pensando en 22 líneas de 32 lugares cada una, y ocho colores. En alta resolución se han considerado 256 puntos en horizontal y 176 en vertical.

TAB

Si el argumento de TAB es un número decimal, generalmente se trunca: TAB(16.8) da igual imagen que TAB(16).

PRINT AT X,Y; , TAB(X,Y);

Cualquiera de estas dos instrucciones permite escribir en la columna Y de la fila X. Si tu ordenador no posee ninguna de ellas, habrás de emplear la Subrutina de Uso Frecuente para bajar a la fila X, y luego TAB(Y).

GO TO, GO SUB

Estas instrucciones se pueden escribir como una sola palabra (GOTO, GOSUB) o como dos, según ordenadores.

IF-THEN-ELSE

Esta es una instrucción que aparece cada vez más en los BASIC ampliados, pero que no hemos empleado por no ser standard. Su uso puede simplificar bastantes de los programas propuestos; hazlo si tu ordenador la posee.

GET, INKEY$

En unos ordenadores la instrucción GET A$ toma un carácter único pulsado desde el teclado y lo guarda en la variable A$, pero *no espera* a que pulsemos dicho carácter. Por esta causa, en estos ordenadores, hemos de optar por una de estas dos formas de emplear la instrucción:

```
30 GET A$
40 IF A$="" THEN 30
```

o bien:

```
30 GET A$ : IF A$="" THEN 30
```

Otros ordenadores aceptan líneas como ésta:

```
30 A$=GET$
```

que espera hasta que pulsamos una tecla y guarda el carácter en A$.

Otros aún admiten la línea:

```
30 A$=INKEY$(200)
```

que espera un máximo de 200 unidades de tiempo para guardar en A$ el carácter pulsado desde el teclado; pasado este tiempo o pulsada una tecla, la ejecución continúa.

La línea

```
30 A$=INKEY$
```

no espera; equivale al GET A$ inicial.

Es de vital importancia que el lector conozca cómo funcionan estas instrucciones en su ordenador.

INICIALIZACION DE VARIABLES

Hemos inicializado siempre las variables, con asignaciones tales como A=0. En algunos ordenadores no es necesario hacerlo, pues "suponen" que las variables a las que aún no se ha asignado valor valen cero.

RUN

En algún ordenador no funcionará RUN 300, siendo 300 un número de línea, y ha de ser sustituido por GOTO 300 empleado como comando (orden directa al ordenador, que no va precedida por número de línea).

En general RUN 300 anula los valores de las variables existentes en ese momento en el ordenador. En cambio GOTO 300 conserva los valores de esas variables.

INPUT "CON ETIQUETA"

Hay ordenadores que no admiten líneas como ésta:

```
20 INPUT "NOMBRE";A$
```

En todo caso, se puede sustituir esta línea por estas dos:

```
15 PRINT "NOMBRE";
20 INPUT A$
```

que funcionan en cualquier ordenador; o, incluso, por:

```
15 PRINT "NOMBRE";
20 INPUT A$
25 PRINT A$
```

que hace aparecer en pantalla el valor de A$, si hubiese desaparecido tras el INPUT.

LINEAS CON MAS DE UNA INSTRUCCION

En ocasiones hemos puesto más de una instrucción en una misma línea, separadas por dos puntos:

```
30 A=0 :B=0 :PRINT D
```

No todos los ordenadores lo admiten.

Mucho cuidado con las líneas del tipo:

```
40 IF A=B THEN PRINT B : B=1
```

porque la instrucción B=1 sólo se llevará a efecto cuando A=B; y en:

```
40 IF B=A THEN 300 : B=1
```

la segunda instrucción no se llevará a efecto nunca.

RND, RANDOMIZE

Son muchos los programas de este libro que utilizan la función RND para producir números aleatorios. Las variantes más frecuentes, según los tipos de ordenador, son:

a) 30 RANDOMIZE
40 X=RND

En X se obtiene un número aleatorio entre 0 y 1, $0<X<1$.

b) 30 X=RND(K), siendo K un número positivo.

En X se obtiene un número aleatorio entre 0 y 1, $0<X<1$.

c) 30 X=RND(K), siendo K un entero positivo distinto de 1.

En X se obtienen aleatoriamente cualquiera de los enteros 1,2,···,K. Asi, por ejemplo, D=RND(6) simula el lanzamiento de un dado. En este tipo de ordenadores, X=RND(1) produce un número aleatorio, X, entre 0 y 1.

El lector deberá adaptar todos los programas que lleven RND, bien incluyendo una línea RANDOMIZE, si su ordenador es del tipo *a*, bien incluyendo como argumento de la función RND el número 1: RND(1), que es válido para los del tipo *b* y *c*.

SIN, COS, TAN

Las funciones trigonométricas tienen su argumento en radianes (un radián $\simeq 58$). Para pasar un ángulo de grados a radianes se multiplica el valor por π y se divide por 180.

TI, TIME

Los microordenadores tienen un reloj que mide el tiempo en 1/50 ó 1/100 de segundo. El tiempo transcurrido desde que se conectó el ordenador se encuentra en la variable TI (o en TIME), cuyo nombre es una palabra reservada, no puede ser empleada por el usuario.

En este libro hemos utilizado TI y suponemos que la unidad de tiempo es 1/50 de segundo.

ASC, CODE

La función ASC (en algunos ordenadores CODE) permite pasar de un carácter cualquiera, letra, número, separador o símbolo de operación, al correspondiente en el código ASCII.

CADENAS

Para la prolongación de cadenas por concatenación generalmente se utiliza el símbolo +, pero en algunos casos ha de ponerse &.

Para segmentarlas lo normal es emplear las funciones LEFT$, RIGHT$ y MID$, que funcionan así: si A$="CAROTA", entonces:

LEFT$(A$,4)="CARO" (toma los 4 caracteres de la izquierda).
RIGHT$(A$,4)="ROTA" (toma los 4 caracteres de la derecha).
MID$(A$,3,2)="RO" (toma 2 caracteres a partir de la posición 3).

Sin embargo, algunos ordenadores lograrían estos mismos resultados mediante las expresiones respectivas:

A$(1 TO 4)="CARO"=A$(TO 4)
A$(3 TO 6)="ROTA"=A$(3 TO)
A$(3 TO 4)="RO"

que, aún no siendo las usuales, son posiblemente más cómodas.

DIMENSIONADO

Hemos mantenido la denominación de listas y tablas, que ya empleamos en BASIC BASICO, para referirnos a las variables subindicadas. Se han dimensionado siempre con instrucciones como DIM L(4), DIM T(5,7), aunque en algunos ordenadores no es necesario hacerlo para dimensiones menores o iguales que 10.

Asimismo hemos supuesto que existe el índice cero para listas y tablas. Por eso hemos empleado en ocasiones variables tales como A(0), B(0,3) y C(0,0), aunque en general es una práctica de programación poco recomendable.

En algunos ordenadores el número total de elementos de una tabla no puede ser mayor de 256; comprueba si tu aparato es de ésos.

LISTAS Y TABLAS ALFANUMERICAS

En alguno de los ordenadores más baratos, las listas y tablas alfanuméricas no admiten más que un carácter en cada lugar. Así, aunque muchos ordenadores pueden admitir una lista, L$, de longitud cuatro, como por ejemplo:

$$L\$ = \begin{pmatrix} \text{CASA} \\ \text{PEDRO} \\ \text{TU} \\ \text{ROCIERO} \end{pmatrix}$$

L$(1)="CASA"
L$(2)="PEDRO"
L$(3)="TU"
L$(4)="ROCIERO"

hay algunos que sólo podrían aceptar:

L$(1)="C"
L$(2)="P"
L$(3)="T"
L$(4)="R"

La forma más obvia de salvar esta situación consiste en dimensionar una tabla de igual longitud que la lista, y de anchura igual a la longitud de la palabra más larga, DIM L$(4,7); entonces:

$$L\$ = \begin{pmatrix} C & A & S & A & & & \\ P & E & D & R & O & & \\ T & U & & & & & \\ R & O & C & I & E & R & O \end{pmatrix}$$

Ahora L$(2,5)="O", y L$(3,6)=" ".

Sin embargo, una vez que la tabla ha sido definida como tal, puedes manejarla como una lista, de forma que L$(1)="CASA ". El mayor problema que presenta este tratamiento es que "CASA"<> L$(1), ya que le faltan los tres espacios que tiene esta última.

ON GOTO, ON GOSUB

La línea 100:

100 ON A GOTO 1000, 2000, 3000
110 ...

transfiere la ejecución a las líneas 1000, 2000 ó 3000, según que el valor de la parte entera de la variable A sea 1, 2 ó 3. Si la parte entera de A es menor que uno o mayor que tres, entonces la ejecución pasaría a la línea 110. La instrucción ON-GOSUB es análoga.

En algunos ordenadores no existen estas instrucciones, pero es posible transferir la ejecución en la forma GOTO D, siendo D una variable. En este caso podemos simular la línea 100 de esta manera:

```
100 IF A>=1 AND A<=3 THEN GOTO INT(A)*1000
```

Si las líneas a las cuales se ha de transferir el control no pueden estar regularmente separadas, por ejemplo: 150, 1810, 3416, se pueden guardar ordenadamente estos valores en una lista:

```
L(1)=150    L(2)=1810    L(3)=3416
```

y sustituir la última versión de la línea 100 por:

```
100 IF A>=1 AND A<=3 THEN GOTO L(A)
```

PAUSE, WAIT

Si deseamos que la pausa que realiza el ordenador tenga duración fija, podemos poner:

```
PAUSE n
```

siendo n el número de fracciones de segundo (generalmente iguales a 1/50) que deseamos se detenga el funcionamiento. En otros ordenadores la instrucción análoga será:

```
WAIT n
```

Si el ordenador carece de estas instrucciones, pero está dotado de la variable TI, entonces podemos emplear la correspondiente Subrutina de Uso Frecuente.

PEEK, POKE

Estas instrucciones permiten conocer el contenido de la celda de memoria k: PEEK(k), o introducir un valor n en el lugar k de la memoria: POKE k,n. Emplear estas instrucciones, que no pertenecen al BASIC (lenguaje abstracto) sino a los BASICs (concreciones de ese lenguaje abstracto a máquinas determinadas), supone: *a*) conocer el lenguaje máquina del microprocesador en que está basado tu microordenador; *b*) conocer el mapa de memoria del microordenador. Cualquiera de estas razones evidencia la imposibilidad de tratar con generalidad estas instrucciones.

En algunos micros es necesario emplear POKE para simular la instrucción PRINT AT, cambiar los colores o generar sonidos.

RUTINAS DE USO FRECUENTE

Algunas de las rutinas que a continuación se exponen las hemos usado con frecuencia y otras en contadas ocasiones. También ponemos rutinas que nosotros no empleamos, pero de las que el lector puede servirse si lo considera oportuno.

PARADA DE TIEMPO FIJO (K unidades de tiempo)

```
1000 .....
1010 T=TI
1020 IF TI-T<K THEN 1020
1030 .....
```

Esta rutina se puede emplear si el ordenador dispone de un reloj interno que se pone a cero cuando el "micro" se conecta a la red.

La variable TI nos da, en cada instante, el tiempo transcurrido desde que se conectó el ordenador.

Si nos interesa que se detenga la ejecución K unidades de tiempo (cada unidad de tiempo suele ser 1/50 de segundo), entre las líneas 1000 y 1030 se intercalan las instrucciones 1010 y 1020.

PARADA DE TIEMPO FIJO (sin variable TI, K ciclos)

```
1000 FOR I=1 TO K
1010 NEXT I
```

Al ejecutar este ciclo, el ordenador "no hace nada" K veces, lo cual supone una espera.

PARADA DE CONTACTO

```
1000 GET A$
1010 IF A$="" THEN 1000
```

Está utilizada y explicada en el problema 1º.

Esta rutina es similar a:

```
1000 IF INKEY$=“” THEN 1000
```

PARADA DE CONTACTO CON TIEMPO ACOTADO

```
1000 T=TI
1010 GET A$
1020 IF A$=“” AND TI-T<K THEN 1010
```

PAUSA

```
1000 PAUSE K
```

El ordenador detiene la ejecución durante K · 1/50 segundos.

SI O NO

```
1000 PRINT “QUIERES VOLVER A EMPEZAR (S/N)?”
1010 GET R$ :IF R$=“” THEN 1010
1020 IF R$<>“S” AND R$<>“N” THEN 1010
1030 IF R$=“S” THEN (a la línea correspondiente)
1040 (BORRAR LA PANTALLA)
1050 PRINT “ADIOS”
1060 END
```

Si queremos poner las variables a cero pondremos:

```
1030 IF R$=“S” THEN RUN (número de la línea correspondiente)
```

En realidad la segunda instrucción de la línea 1010 es redundante con la línea 1020.

CICLO SIN FIN

```
1000 GOTO 1000
```

Esta instrucción tiene por objeto que la ejecución nunca se detenga. Se suele poner al final del programa para evitar los mensajes del tipo READY u OK, que pueden deteriorar la presentación en pantalla.

TRUNCAR A N CIFRAS DECIMALES

```
1000 B=INT(A*1EN)/1EN
```

Ver BASIC BASICO Curso de Programación, pág. 122.

REDONDEAR UN NUMERO A N DECIMALES

```
1000 B=INT(A*1EN+.5)/1EN
```

Ver BASIC BASICO Curso de Programación, pág. 137.

OBTENCION DE NUMEROS ALEATORIOS

1000 A=RND	Resultando	$0<A<1$	
1000 A=INT(N*RND)+1	Resultando	$1\leqslant A\leqslant N$	A entero
1000 N=A+(B-A)*RND	Resultando	$A<N<B$	

Este punto ha sido tratado para diversos ordenadores en las Advertencias Previas. Aquí exponemos la utilización concreta que se ha empleado en este libro.

BAJADA A LA LINEA X

```
1000 (CURSOR AL ORIGEN)
1010 FOR I=1 TO X
1020 PRINT
1030 NEXT I
```

CURSOR EN FILA X, COLUMNA Y

```
1000 (CURSOR AL ORIGEN)
1010 FOR I=1 TO X :PRINT :NEXT I
1020 PRINT TAB(Y);“los caracteres correspondientes”
```

BORRADO DE UN TROZO DE LINEA IMPRIMIENDO A CONTINUACION

Puesto el cursor en la posición deseada, se imprime encima de la parte de línea que queremos borrar, una cadena de tantos espacios en blanco como sea la longitud de dicho fragmento.

```
1000 PRINT TAB(Y);".....";
```


1. JUEGOS

1. ADIVINA UN NUMERO

Se trata de programar un inocente juego: El ordenador piensa un número del 1 al 100 y nosotros lo tenemos que adivinar.

Después de proponer nuestro número, el ordenador nos contestará con frases tales como "más alto", "más bajo", "has acertado después de 7 intentos", según que nuestro número sea menor, mayor o igual que el número secreto.

Unas instrucciones previas, impresas en la pantalla, vendrán bien para explicar el juego al que no lo conozca. Estas instrucciones permanecerán hasta que el jugador las haya leído; momento en el que, al pulsar una tecla cualquiera, se borrarán. Esto se consigue mediante la instrucción:

```
GET A$
```

que guarda en la variable de cadena A$ el carácter que en ese momento haya sido pulsado en el teclado.

Una secuencia típica de utilización del GET es la siguiente:

```
100  GET A$
110  IF A$="" THEN 100
120  ···  (sigue la ejecución)
```

Mientras no se pulse una tecla, la variable A$ estará vacía, es decir, la condición A$="" será cierta, y el programa volverá a la línea 100. Cuando se pulse una tecla cualquiera, la variable A$ se llenará, la condición A$="" no se cumplirá, y la ejecución del programa continuará en el paso 120.

Es claro que el ordenador debe elegir un número al azar del 1 al 100. Esto lo hace en la línea:

```
130  X=INT(100*RND+1)
```

```
10 REM ADIVINA UN NUMERO
20 ( BORRAR LA PANTALLA )
30 PRINT "TIENES QUE ADIVINAR UN NUMERO"
40 PRINT "COMPRENDIDO ENTRE 1 Y 100."
50 PRINT
60 PRINT "DESPUES DE ESCRIBIR TU NUMERO"
70 PRINT "PULSA LA TECLA RETURN."
80 PRINT
90 PRINT "PARA CONTINUAR PULSA UNA TECLA."
100 GET A$
110 IF A$="" THEN 100
120 ( BORRAR LA PANTALLA )
130 X=INT(100*RND+1)
140 I=0
150 PRINT
160 INPUT "ESCRIBE TU NUMERO";N
170 I=I+1
180 IF N<X THEN PRINT "MAS ALTO" : GOTO 150
190 IF N>X THEN PRINT "MAS BAJO" : GOTO 150
200 PRINT
210 PRINT "ENHORABUENA. ACERTASTE"
220 PRINT "DESPUES DE";I;"INTENTOS"
230 PRINT
240 PRINT "QUIERES VOLVER A JUGAR (S/N)?"
250 GET R$ : IF R$="" THEN 250
260 IF R$<>"S" AND R$<>"N" THEN 250
270 IF R$="S" THEN 120
280 ( BORRAR LA PANTALLA )
290 PRINT "ADIOS"
300 END
```


2. CARRETERA

La figura inferior muestra la simulación de una carretera con curvas: el coche es la «V» que, pulsando la tecla Q se desplaza un lugar hacia la izquierda, y pulsando la P un lugar hacia la derecha. Cuando el coche se sale de la carretera se termina la partida.

```
GRADO DE DIFICULTAD 1-3? 3
                    I==V==I
                     I=V===I
                      IV====I
                      I=V===I
                       I=V===I
                       I==V==I
                        I==V==I
                         I==V==I
                          I=V===I
                         I==V==I
                         I===V=I
                        I====VI
                       I====VI
                      I=====V

HAS RECORRIDO  1.3  KM SIN ACCIDENTES
```

La variable X indica la posición del comienzo de la cadena que es la carretera, y la variable Y, la posición del coche. La X se mueve a uno u otro lado aleatoriamente, según lo ordene la línea 150. Si se va a llegar a uno de los bordes laterales de la pantalla, se hace que la carretera «rebote» (líneas 160 y 170).

En la línea 120 se utilizan las funciones LEFT$ y RIGHT$ para imprimir cada vez la carretera y en ella el coche. Es la línea 130 la que contiene la condición de que si el coche se sale de la carretera, se termina el juego.

Fíjate en la variable A$ en la línea 50: necesita un espacio en blanco a cada lado. Observa también la línea 130. ¿Por qué utiliza el signo < =, y no pone simplemente =? Puedes probar a modificarlo y verás los errores en que se incurre.

```
10 REM CARRETERA
20 ( BORRAR LA PANTALLA )
30 PRINT : PRINT : PRINT
40 INPUT "GRADO DE DIFICULTAD 1-3";G
50 A$=" I=====I "
60 X=16 : Y=20
70 K=0
80 FOR I=1 TO 30*G : NEXT I
90 GET B$
100 IF B$="Q" THEN Y=Y-1
110 IF B$="P" THEN Y=Y+1
120 PRINT TAB(X);LEFT$(A$,Y-X);"V";RIGHT$(A$,X+8-Y)
130 IF Y-X<=1 OR Y-X>=7 THEN 190
140 K=K+1
150 X=X+INT(3*RND)-1
160 IF X=0 THEN X=2
170 IF X=32 THEN X=30
180 GOTO 80
190 PRINT : PRINT : PRINT
200 PRINT "HAS RECORRIDO ";K/10;" KM SIN ACCIDENTES"
210 END
```

Para utilizar este programa en tu microordenador probablemente tendrás que hacer algunas variaciones.

— Líneas 50 y 120: elegir para la carretera y el coche los símbolos más apropiados de entre los que disponga tu microordenador.

— Línea 60: centrar la carretera en tu pantalla. Para ello el valor inicial de la X es:

$$X = \frac{\text{n.º de espacios}}{2} - 4$$

Centrar el coche en la carretera: $Y = X + 4$.

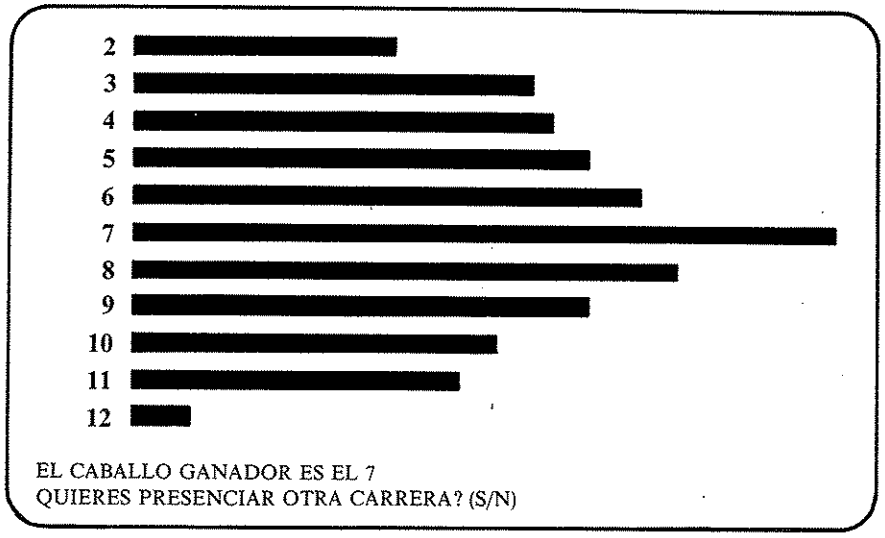
Ajustar el efecto de rebote por la derecha en la línea 170. Si A es el número de columnas de tu pantalla pondrás: IF X=A−8 THEN X=A−10.

— Línea 80: ajustar la expresión de la dificultad a la velocidad de ejecución de tu microordenador.

¡¡¡BUENA SUERTE!!!

3. CARRERAS DE CABALLOS (7.12)

Simular una carrera de once caballos numerados del 2 al 12. El avance de cada caballo se decide lanzando dos dados: la suma obtenida es el número del caballo que avanza un espacio. Se acaba la carrera cuando el primero llega a la meta. (Conviene representar el avance de los caballos por una especie de diagrama de barras.)



En primer lugar representaremos los caballos en la línea de salida dejando un espacio entre ellos. Comenzamos la carrera lanzando los dos dados:

$$X = INT(6*RND + 1) \qquad Y = INT(6*RND + 1)$$

La suma de los puntos se recoge en la variable S:

$$S = X + Y$$

A continuación se suma una unidad al contador de espacios del caballo S:

$$C(S) = C(S) + 1$$

Ahora tenemos que reflejar el avance del caballo S en la pantalla. Para ello situamos el cursor en el origen de la pantalla (rincón superior izquierdo) y descendemos hasta la línea de S, donde imprimimos, mediante la función TAB, un cuadradito ■ (o bien, el signo *) en la posición correspondiente a C(S).

La carrera se acaba en el instante en que uno de los caballos haya avanzado 36 espacios. Por tanto, si C(S) es menor que 36, hay que volver a echar los dados; en caso contrario se termina la carrera imprimiendo: EL CABALLO GANADOR ES EL S. Al final se pregunta si se quiere presenciar otra carrera. En caso afirmativo, la instrucción RUN 20 de la línea 230 pone todas las variables a cero y el programa recomienza a partir de la línea 20. Si la respuesta es negativa, el programa termina.

La impresión final del programa que ofrecemos ocupa 23 líneas de 40 caracteres. Tienes que tener en cuenta las dimensiones de la pantalla de tu ordenador y modificar el programa según aquéllas.

La técnica utilizada para representar el avance de los caballos puedes emplearla para construir diagramas de barras. Una variante de esta técnica de representación consiste en establecer las variables de cadena C$(S), donde puedes ir acumulando los cuadraditos ■ para construir las barras:

$$C\$(S) = C\$(S) + \text{"■"}$$

Al final imprimes las barras C$(S), para los distintos valores de S.

```
10 REM CARRERA DE CABALLOS
20 DIM C(12)
30 ( BORRAR LA PANTALLA )
40 FOR I=2 TO 12
50 PRINT : PRINT I
60 NEXT I
70 X=INT(6*RND+1)
80 Y=INT(6*RND+1)
90 S=X+Y
100 C(S)=C(S)+1
110 ( CURSOR AL ORIGEN )
120 FOR K=1 TO 2*S-3
130 PRINT
140 NEXT K
150 PRINT TAB(C(S)+3);"■"
160 IF C(S)<36 THEN 70
170 ( CURSOR AL ORIGEN )
```



```
180 FOR I=1 TO 21 : PRINT : NEXT I
190 PRINT "EL CABALLO GANADOR ES EL";S
200 PRINT "QUIERES PRESENCIAR OTRA CARRERA? (S/N)"
210 GET R$
220 IF R$<>"S" AND R$<>"N" THEN 210
230 IF R$="S" THEN RUN 20
240 ( BORRAR LA PANTALLA )
250 PRINT "ADIOS"
260 END
```

4. RULETA RUSA (INOFENSIVA)

Se toma un revólver, y tras colocar una bala en el tambor, se le da vueltas. Cada jugador realiza esa operación y simula disparar el revólver. Si "sale" la bala, "muere" y el juego acaba. En caso contrario, "vive" y toca el turno al siguiente jugador.

El número máximo de jugadores se ha fijado en 10, pero es posible jugar menos, ya que el número de jugadores se introduce mediante la variable N.

Numeramos los agujeros del tambor del 1 al 6 y en el 1 colocamos la bala. Por medio de un bucle hacemos disparar a todos los jugadores y dar vueltas al tambor. Si coincide el número tomado al azar con el 1, es que la bala hizo impacto y en otro caso, es que no lo hizo.

Presentación en pantalla:

```
CUANTAS PERSONAS? 3
NOMBRES DE LAS PERSONAS
? ENRIQUE
? AGUSTIN
? RICARDO
ENRIQUE !!CLIK!! HUBO SUERTE, ESTAS VIVO
AGUSTIN !!PUM!! NO HUBO SUERTE, ESTAS MUERTO
RICARDO !!CLIK!! HUBO SUERTE, ESTAS VIVO
```

```
10 REM RULETA RUSA (INOFENSIVA)
20 DIM A$(10)
30 PRINT "CUANTAS PERSONAS";
40 INPUT N
50 PRINT "NOMBRES DE LAS PERSONAS"
60 FOR I=1 TO N
70 INPUT A$(I)
80 NEXT I
90 B=1
100 REM VUELTAS AL TAMBOR
110 FOR I=1 TO N
120 P=INT(6*RND)+1
130 REM DISPAROS
```



```
140 PRINT A$(I);
150 IF B=P THEN 180
160 PRINT " !!CLIK!! HUBO SUERTE, ESTAS VIVO"
170 GOTO 190
180 PRINT " !!PUM!! NO HUBO SUERTE, ESTAS MUERTO"
190 NEXT I
200 END
```

5. TIRO AL PLATO

Se trata de programar un tiro al plato *muy particular: En la parte superior de la pantalla aparece, en posición aleatoria, una letra (el plato) elegida al azar. El jugador tiene un segundo y medio para pulsar (disparar) la tecla correspondiente a la letra. Si lo hace en menos de ese tiempo, rompe el plato, y en la pantalla aparece un efecto de flas e inmediatamente sale el plato siguiente. Si no acierta, aparece otro plato.*

El juego dura 10 segundos. Al final, el ordenador indica el número de platos rotos.

Vamos a hacer algunos comentarios al programa que ofrecemos:

Los microordenadores tienen un reloj que cuenta el tiempo en fracciones de segundo. En el programa hemos supuesto que nuestro ordenador mide el tiempo en 1/50 de segundo (en otros la unidad de tiempo es la centésima de segundo). El tiempo transcurrido desde que el ordenador ha sido conectado se encuentra en la variable TI, cuyo nombre está reservado (otros utilizan la variable TIME).

En la línea 80 se toma el tiempo inicial y se guarda en la variable T0. La línea 90, que se ejecuta varias veces, controla si el tiempo trascurrido es superior o igual a los 10 segundos, en cuyo caso termina el juego.

La línea 170 no permite que el plato esté más de un segundo y medio en el aire.

Las líneas 500-540 forman la subrutina de escritura de un carácter X$ en la posición D de la línea B (en aquellos ordenadores que posean la función PRINT AT D, B se puede prescindir de la subrutina). En la línea 140 se reclama la subrutina para situar el plato en la posición aleatoria (D, B). También se utiliza la subrutina en las líneas 220 y 250 para producir el efecto de flas cuando se rompe un plato.

Cada letra (plato) es elegida al azar del siguiente modo: en la línea 100 se produce un número aleatorio N comprendido entre 65 y 90. Este número es el código ASCII de la letra. En la línea 110 se obtiene la letra correspondiente a N y se guarda en la variable X$.


```
10 REM TIRO AL PLATO
20 ( BORRAR LA PANTALLA )
30 PRINT "CUANTOS PLATOS PUEDES ROMPER"
40 PRINT
50 PRINT "EN 10 SEGUNDOS ?"
60 FOR I=1 TO 2000 : NEXT I
70 C=0
80 T0=TI
90 IF TI-T0>=500 THEN 290
100 N=INT(26*RND+65)
110 X$=CHR$(N)
120 B=INT(10*RND+1)
130 D=INT(40*RND)
140 GOSUB 500
150 T1=TI
160 GET L$
170 IF TI-T1>=75 THEN 90
180 IF L$<>X$ THEN 160
190 C=C+1
200 FOR K=1 TO 3
210 X$="*"
220 GOSUB 500
230 FOR I=1 TO 20 : NEXT I
240 X$=" "
250 GOSUB 500
260 FOR I=1 TO 20 : NEXT I
270 NEXT K
280 GOTO 90
290 PRINT "HAS ROTO ";C; " PLATOS"
300 END
500 REM SUBRUTINA DE ESCRITURA
510 ( BORRAR LA PANTALLA )
520 FOR I=1 TO B : PRINT : NEXT I
530 PRINT TAB(D);X$
540 RETURN
```

6. SUBMARINO INMOVIL (7.22)

En un casillero 6 × 6 el ordenador sitúa al azar un submarino, fuera del alcance de nuestra vista y del sónar. Vamos lanzando cargas de profundidad en diversas casillas. Si una carga da al submarino, el ordenador contesta "HUNDIDO". Si da en una casilla contigua a la del submarino, el ordenador contestará "SUBMARINO CERCA". Si la carga da en otro lugar, la contestación es "AGUA".

Al hundir el submarino el ordenador ha de sacar en pantalla el número de cargas de profundidad utilizadas, así puede haber competición entre varios jugadores, ganando quien consiga el hundimiento con el menor número de ellas. (No es necesaria la representación gráfica.)

En las variables B1 y B2 están colocadas al azar las coordenadas del submarino. Para ello hacemos uso de la función RND.

En las variables X e Y están colocadas las coordenadas del lugar en que creemos nosotros que está situado el submarino. Introducimos en la variable U la distancia entre esas dos casillas. Mejor dicho, introducimos, por comodidad, la distancia al cuadrado. Si resulta ser 0, es que hemos HUNDIDO el submarino; si es mayor que 3, es "AGUA". Por exclusión, si no es ninguno de los dos, se imprimirá "SUBMARINO CERCA".

Presentación en pantalla:

```
INTRODUCE LAS COORDENADAS X,Y DE LA CARGA
? 2,2
EN LA CASILLA X= 2 Y= 2 AGUA
INTRODUCE LAS COORDENADAS X,Y DE LA CARGA
? 2,5
EN LA CASILLA X= 2 Y= 5 AGUA
INTRODUCE LAS COORDENADAS X,Y DE LA CARGA
? 5,2
EN LA CASILLA X= 5 Y= 2 AGUA
INTRODUCE LAS COORDENADAS X,Y DE LA CARGA
? 5,5
EN LA CASILLA X= 5 Y= 5 SUBMARINO CERCA
```

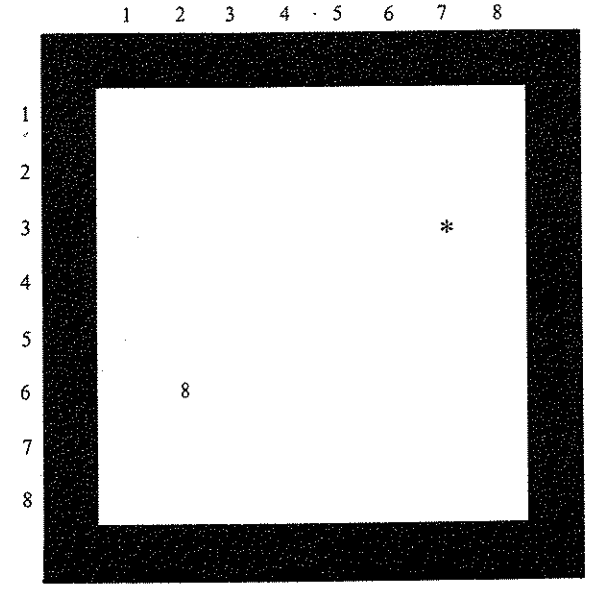


```
INTRODUCE LAS COORDENADAS X,Y DE LA CARGA
? 5,4
EN LA CASILLA X= 5 Y= 4 AGUA
INTRODUCE LAS COORDENADAS X,Y DE LA CARGA
? 4,6
EN LA CASILLA X= 4 Y= 6 HUNDIDO
HAS NECESITADO 6 CARGAS
```

```
10 REM SUBMARINO INMOVIL
20 B1=INT(6*RND)+1
30 B2=INT(6*RND)+1
40 I=0
50 PRINT "INTRODUCE LAS COORDENADAS X,Y DE LA C
ARGA"
60 INPUT X,Y
70 IF X>6 OR Y>6 THEN PRINT "EL CASILLERO ES 6
POR 6" : PRINT : GOTO 50
80 I=I+1
90 U=(B1-X)↑2+(B2-Y)↑2
100 IF U=0 THEN 160
110 IF U>3 THEN 140
120 PRINT "EN LA CASILLA X=";X;"Y=";Y;"SUBMARIN
O CERCA"
130 GOTO 50
140 PRINT "EN LA CASILLA X=";X;"Y=";Y;"AGUA"
150 GOTO 50
160 PRINT "EN LA CASILLA X=";X;"Y=";Y;"HUNDIDO"
170 PRINT "HAS NECESITADO";I;"CARGAS"
180 END
```

7. SUBMARINO

En un casillero de 8 × 8 el ordenador sitúa al azar un submarino, fuera del alcance de nuestra vista. Vamos lanzando cargas de profundidad en diversas casillas. Tras cada carga lanzada, el ordenador contesta con la distancia a la que se encuentra el submarino, medida en casillas horizontales y verticales. Si la distancia es cero, contesta "HUNDIDO" y muestra la posición del submarino (*). (Los buenos jugadores con tres cargas tienen suficiente para destruir el buque.)



LINEA, COLUMNA? 6, 2

En el programa que proponemos hemos establecido una subrutina (300-399) para dibujar el cuadro de batalla. Para ello hemos introducido en tres variables de cadena los siguientes motivos:

M1$ = "12345678"
M2$ = "■■■■■■■■■■" (10 cuadraditos negros)
M3$ = "■ ■" (8 espacios con un cuadradito negro a cada lado)

La impresión, en líneas sucesivas, de los motivos M1$, M2$, M3$ (ocho veces) y M2$, poniendo delante de cada M3$ el número correspondiente, produce el recuadro de la figura.

Con la función TAB se ajustan los motivos y se centra el dibujo.

En las líneas 110, 120 y 130 se cargan en la variable de índice D$(I) los caracteres *,1,2,3,4,5,6,7,8,9,A,B,C,D,E, que van a marcar la distancia D a la que se encuentra el submarino. Como para dicha marca sólo disponemos de un espacio, las distancias 10,11,12,13 y 14 son representadas por las letras A,B,C,D y E.

Comienza el juego a partir de la línea 140, que remite a la subrutina 300 de representación gráfica.

En la línea 150 el ordenador elige al azar las coordenadas del submarino.

Las líneas 160, 170 y 180 sirven para situar el cursor en la zona inferior de la pantalla, y preguntar allí las coordenadas de la zona del disparo.

Desde la 190 a la 220 se calcula la distancia D, y se imprime en la casilla correspondiente. En la 230, si la distancia no es cero, se repite el disparo, y si es cero, se imprime "HUNDIDO", borrando previamente con espacios la línea en la que hemos estado poniendo las coordenadas. Y el juego finaliza.

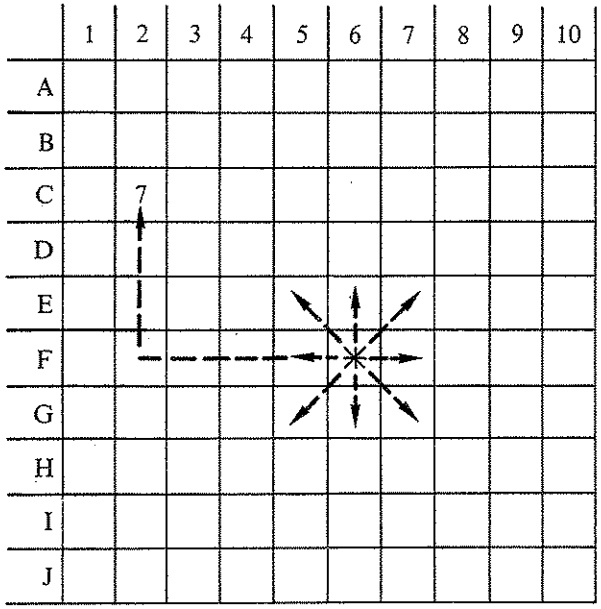
```
10 REM LA CAZA DEL SUBMARINO
20 GOSUB 300
30 PRINT : PRINT "ESTO ES LA CAZA DEL SUBMARINO"
40 PRINT : PRINT "SE LANZA UNA CARGA DE PROFUNDI
DAD"
50 PRINT "ESCRIBIENDO EL NUMERO DE LINEA Y EL DE
COLUMNA SEPARADOS POR COMA"
60 PRINT : PRINT "EN LA ZONA CORRESPONDIENTE APA
RECE"
70 PRINT "LA DISTANCIA A LA QUE SE HALLA EL"
80 PRINT "SUBMARINO MEDIDA EN CUADRADITOS  HORIZ
ONTALES Y VERTICALES"
90 PRINT : PRINT "PARA EMPEZAR EL JUEGO PULSA UN
A TECLA"
100 GET A$ : IF A$="" THEN 100
110 DIM D$(14)
120 FOR I=0 TO 14 : READ D$(I) : NEXT I
130 DATA *,1,2,3,4,5,6,7,8,9,A,B,C,D,E
140 GOSUB 300
150 X=INT(8*RND+1) : Y=INT(8*RND+1)
160 ( CURSOR AL ORIGEN )
```

```
170 FOR I=1 TO 20 : PRINT : NEXT I
180 INPUT "LINEA,COLUMNA";L,C
190 D=ABS(X-L)+ABS(Y-C)
200 ( CURSOR AL ORIGEN )
210 FOR I=1 TO L+1 : PRINT : NEXT I
220 PRINT TAB(C+12);D$(D)
230 IF D<>0 THEN 160
240 ( CURSOR AL ORIGEN )
250 FOR I=1 TO 20 : PRINT : NEXT I
260 PRINT "              HUNDIDO"
270 END
300 REM SUBRUTINA DE REPRESENTACION GRAFICA
310 M1$="12345678"
320 M2$="■■■■■■■■■■"
330 M3$="■        ■"
340 ( BORRAR LA PANTALLA )
350 PRINT TAB(13);M1$
360 PRINT TAB(12);M2$
370 FOR I=1 TO 8
380 PRINT TAB(9);I;TAB(12);M3$
390 NEXT I
395 PRINT TAB(12);M2$
399 RETURN
```


8. JUEGO DEL SUBMARINO MOVIL (7.23)

Variante del juego anterior en un casillero de 10 × 10. Después de cada carga lanzada la máquina contesta la distancia en casillas y, si la distancia es cero, responde "HUNDIDO EN 7 INTENTOS", por ejemplo. Pero en esta variante el submarino se mueve: cada vez que lanzamos una carga se desplaza aleatoriamente una casilla. (Si va hacia los bordes, rebotará en ellos.) Igual que antes, se trata de hundir el submarino con el menor número posible de cargas.

En las primeras líneas, hasta la 130, se representa el "campo de batalla", arriba números y a la izquierda letras. Las letras de la A a la J tienen los códigos 65 a 74.



DONDE DISPARAS (PRIMERO LA LETRA)? C,2

Después se sitúa el submarino, pero no se saca en la pantalla. Entonces el ordenador pregunta dónde disparamos. Es necesario un bucle previo, línea 190, para que la pregunta se formule siempre abajo, sin estropear el dibujo.

Supongamos que nuestra respuesta es C,2, y que el submarino está en F,6. En este juego se mide la suma de la distancia en horizontal y en vertical,

en este caso 7 (línea 220). Queremos que salga en la pantalla un 7 en la posición C,2. Es lo que hacen las líneas 230 a 250.

La variable K contabiliza el número de cargas lanzadas. Si la distancia D es cero, hemos hundido el submarino en K cargas. Si no, el submarino se mueve una casilla al azar, como muestra el dibujo. Puede también quedarse en el mismo sitio. El movimiento corre a cargo de las líneas 280 y 290, con las condiciones posteriores de que el submarino no se salga de la cuadrícula.

```
10 REM SUBMARINO MOVIL
20 ( BORRAR LA PANTALLA )
30 REM DIBUJO DE LAS COORDENADAS
40 PRINT SPC(5);
50 FOR I=1 TO 10
60 PRINT I;
70 NEXT I
80 ( CURSOR AL ORIGEN )
90 PRINT : K=0
100 FOR I=65 TO 74
110 PRINT
120 PRINT SPC(4);CHR$(I)
130 NEXT I
140 REM COORDENADAS DEL SUBMARINO
150 X=INT(10*RND)+1
160 Y=INT(10*RND)+1
170 REM CARGA DE PROFUNDIDAD
180 ( CURSOR AL ORIGEN )
190 FOR I=1 TO 22 : PRINT : NEXT I
200 INPUT "DONDE DISPARAS (PRIMERO LA LETRA)";A$,M
210 N=ASC(A$)-64
220 D=ABS(M-X)+ABS(N-Y)
230 ( CURSOR AL ORIGEN )
240 FOR I=1 TO 2*N : PRINT : NEXT I
250 PRINT SPC(2+3*M);D
260 K=K+1
270 IF D=0 THEN 350
280 X=X+INT(3*RND)-1
290 Y=Y+INT(3*RND)-1
300 IF X=0 THEN X=2
310 IF X=11 THEN X=9
320 IF Y=0 THEN Y=2
330 IF Y=11 THEN Y=9
340 GOTO 170
350 ( CURSOR AL ORIGEN )
360 FOR I=1 TO 23 : PRINT : NEXT I
370 PRINT "HUNDIDO EN ";K;"INTENTOS"
380 END
```


9. RULETA (6.25)

Haz un programa de simulación de una ruleta: tiene 36 números a los que se puede apostar, y el 0. Si sale el 0, todas las apuestas son para la casa. Si aciertas el número, el premio es tu apuesta multiplicada por 36. Además, los 36 números se clasifican en pares o impares, y pasa (19 o mayor) o falta; si ganas en estos casos, el premio es tu jugada multiplicada por dos. Empieza el juego con P pts. y ve haciendo apuestas. El ordenador debe decir cuánto dinero te queda, y no permitirte apostar más de lo que tienes. El juego se acaba si te quedas sin dinero.

En la variable Y se deposita el dinero que se apuesta a número; en la Y1 se deposita el dinero que se apuesta a par o impar y en la Y2 se deposita el dinero que se apuesta a pasa o falta.

En X se coloca el número que sale en la ruleta y en Z el número al que apostamos; en A$, si es par o impar; y en B$, si pasa o falta.

Si el número X es par, entonces en K$ se coloca una P y si es impar, se coloca una I. Si el número X es igual a 19 o mayor, en H$ se coloca una P y si es menor que 19, una F.

Para que un jugador no pueda apostar en cada caso más dinero del que tiene en ese momento, hemos recurrido a lo siguiente: en las líneas 110, 180 y 250 se resta al dinero disponible el dinero que se pretende apostar. A continuación comprobamos si lo que quedó es positivo o nulo. En caso contrario es que el jugador pretendía engañar a la ruleta. Le sacamos un mensaje y volvemos a poner en la variable P el dinero que tenía, para que haga una nueva apuesta.

A continuación, con diversas bifurcaciones condicionales, vamos considerando todas las posibles alternativas que nos indica el enunciado del problema. La única que parece algo extraña es aquella en que nadie acierta, ni los jugadores ni la banca. Para este caso hemos recurrido a un contador, de forma que si en él hay un 3, entonces perdemos todo lo apostado.

En cuanto a la terminación del juego, hemos considerado el caso de no tener más dinero; pero si queremos finalizar el juego sin haberlo agotado, tendremos que pulsar la tecla RUN/STOP.

Si no quieres apostar a alguna de las cosas, pon cero en la cantidad que te juegas.

Presentación en pantalla:

```
CON QUE CANTIDAD ENTRAS AL CASINO
? 10000

A QUE NUMERO APUESTAS
? 24
CUANTO DINERO
? 500
APUESTAS A PARES O IMPARES(P/I)
? I
CUANTO DINERO
? 3000
APUESTAS A PASA O FALTA(P/F)
? F
CUANTO DINERO
? 5500
HA SALIDO EL NUMERO 9
HAS GANADO 8000

TIENES, POR TANTO 18000
```

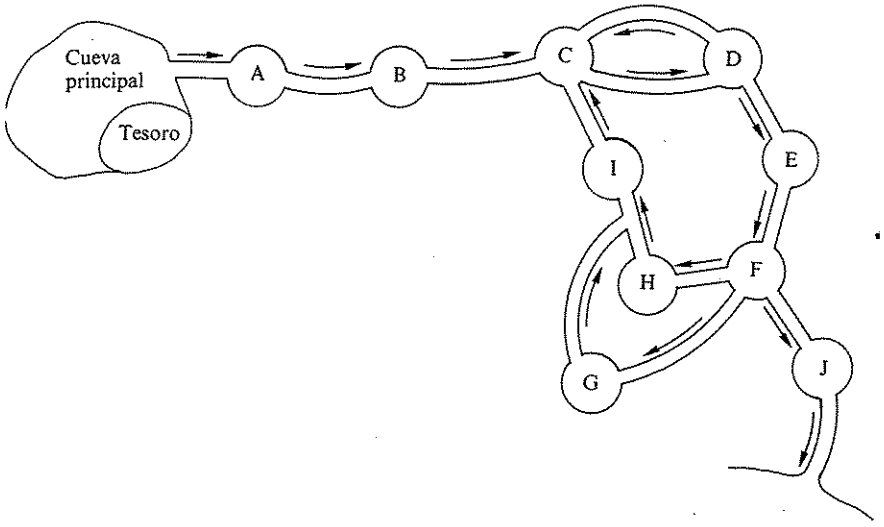
```
10 REM SIMULACION DE RULETA
20 PRINT "CON QUE CANTIDAD ENTRAS AL CASINO"
30 INPUT P
40 P1=P
50 PRINT : X=INT(37*RND) : I=0
60 IF P=0 THEN PRINT "SE ACABO EL JUEGO, NO TIENES DINERO" :
GOTO 460
70 PRINT "A QUE NUMERO APUESTAS"
80 INPUT Z
90 PRINT "CUANTO DINERO"
100 INPUT Y
110 P=P-Y : IF P>=0 THEN 140
120 PRINT "NO PUEDES APOSTAR MAS DINERO DEL QUE TIENES"
130 P=P+Y : GOTO 90
140 PRINT "APUESTAS A PARES O IMPARES(P/I)"
150 INPUT A$
160 PRINT "CUANTO DINERO"
170 INPUT Y1
180 P=P-Y1 : IF P>=0 THEN 210
190 PRINT "NO PUEDES APOSTAR MAS DINERO DEL QUE TIENES"
200 P=P+Y1 : GOTO 160
210 PRINT "APUESTAS A PASA O FALTA(P/F)"
220 INPUT B$
230 PRINT "CUANTO DINERO"
240 INPUT Y2
```



```
250 P=P-Y2 : IF P>=0 THEN 280
260 PRINT "NO PUEDES APOSTAR MAS DINERO DEL QUE TIENES"
270 P=P+Y2 : GOTO 230
280 PRINT "HA SALIDO EL NUMERO";X
290 IF X>=19 THEN H$="P" : GOTO 310
300 H$="F"
310 IF X/2=INT(X/2) THEN K$="P" : GOTO 330
320 K$="I"
330 IF X=0 THEN PRINT "HA GANADO LA BANCA" : GOTO 450
340 IF X=Z THEN P=P+37*Y : GOTO 360
350 I=I+1
360 IF A$=K$ THEN P=P+2*Y1 : GOTO 380
370 I=I+1
380 IF B$=H$ THEN P=P+2*Y2 : GOTO 400
390 I=I+1
400 IF I=3 THEN PRINT " NO HAS GANADO NADA" : GOTO 450
410 L=P-P1
420 IF L=0 THEN I=3 : GOTO 400
430 IF L<0 THEN PRINT "HAS PERDIDO";-L : GOTO 450
440 PRINT "HAS GANADO";L
450 PRINT : PRINT "TIENES, POR TANTO";P : GOTO 40
460 END
```

10. OGROS

Estás en poder de una tribu de ogros que habita un laberinto de cuevas (aunque tú no lo sabes, son unos bromistas impenitentes, y encima vegetarianos). Hoy celebran su Fiesta del Quede; en ella comunican a sus prisioneros que serán devorados a la mañana siguiente y luego, disimuladamente, les dejan posibilidad de escapar, distribuyéndose ellos por las cuevas según indican las iniciales de sus nombres.



Antes de huir haces un saquito con tus pantalones y robas monedas de oro, diamantes, brazaletes y amuletos. Luego huyes ...¡Has de saber que inmediatamente te encontrarás con la enorme

Arsaminiña, que se fingirá tu cómplice y te entregará un pesado amuleto y cinco grandes brazaletes, empujándote a seguir.

Bacilonrio te preguntará la edad y te dará un diamante más de los años que tienes, haciéndote también seguir; ya tambaleándote por el peso llegarás a

Carcajundia quien te dará tantas monedas como brazaletes lleves, quitándote tantos diamantes como amuletos tengas.

Dèmenciano cuenta el número de amuletos y diamantes que llevas y te envía por un túnel que te devuelve a Carcajundia, si tienes menos amuletos que diamantes y, en caso contrario, por otro que lleva a Eufemismondro.

Eufemismondro te da tres amuletos, pero no te deja descansar ni un momento.

Festibonia te enviará hacia Jocosondrón si tienes 10 amuletos o más; si tienes menos de cinco, a Guasononda; y en los demás casos a Histericundia.

Guasononda te quita tantos brazaletes como amuletos tienes y te hace huir por un corredor que conduce a Irrisondro.

Histericundia te da dos monedas y se pone a gritar como su nombre indica, con lo que te largas sin más.

Irrisondro te entrega un diamante por cada moneda que llevas, pero antes te ha quitado todos los diamantes que llevabas.

Jocosondrón te quita todos los amuletos, brazaletes, monedas y un diamante "que a los ogros nos da suerte", guardándolos para la siguiente Fiesta, y de un empellón te devuelve a la libertad!

Hemos de programar esta epopeya, con comentarios de los protagonistas mencionados, de forma que conozcamos la situación de nuestro tesoro en cada momento. Podemos prescindir de un diagrama de flujo formal, papel que puede desempeñar el plano de las cuevas. El programa pedido puede quedar así:

```
10 REM OGROS
20 (BORRAR LA PANTALLA)
30 PRINT "PUEDES HUIR DE LOS OGROS." :PRINT "HACES UN SACO CON LO
S PANTALONES."
40 PRINT : INPUT "CUANTOS AMULETOS COGES";AM
50 PRINT : INPUT "CUANTAS MONEDAS DE ORO COGES";MO
60 PRINT : INPUT "CUANTOS DIAMANTES";DI
70 PRINT : INPUT "CUANTOS BRAZALETES";BR
80 (BORRAR LA PANTALLA)
90 PRINT : PRINT : PRINT "ESCAPAS CON " : PRINT
100 PRINT TAB(12);AM;" AMULETOS" : PRINT
110 PRINT TAB(12);MO;" MONEDAS DE ORO" : PRINT
120 PRINT TAB(12);DI;" DIAMANTES" : PRINT
130 PRINT TAB(12);BR;" BRAZALETES" : PRINT
140 GET A$ : IF A$="" THEN 140
150 (BORRAR LA PANTALLA)
160 PRINT "        TE ENCUENTRAS CON ARSAMINIÑA QUE TE DA CINCO G
RANDES ";
170 PRINT "BRAZALETES Y TE CUELGA DEL CUELLO UN AMULETO CON UN
PEDRUSCO QUE ";
180 PRINT "PARECE UN ADOQUIN"
190 BR=BR+5 : AM=AM+1 : GOSUB 2000
200 PRINT "            ME LLAMAN BACILONDRIO." : PRINT
210 INPUT "    CUANTOS AÑOS TIENES, PEDAZO DE COSA PEQUEÑA";A : P
RINT
```

```
220 PRINT " CON QUE TIENES";A;"AÑOS, EH ? PUES TOMA ";A+1;"DIAMANTE
S Y VETE ";
230 PRINT "ANTES DE QUE CAMBIE DE OPINION"
240 DI=DI+A+1 : GOSUB 2000
250 PRINT : PRINT "            SOY CARCAJUNDIA." : PRINT
260 PRINT "   DEJAME VER CUANTOS BRAZALETES TIENES Y TE DARE MON
EDAS." : PRINT
268 BR=6
270 PRINT "          TIENES";BR;"BRAZALETES." : PRINT
280 PRINT "TOMA";BR;"MONEDAS Y TE QUITO";
290 IF DI<=AM THEN PRINT "TODOS LOS "; : DI=0 : GOTO 310
300 PRINT AM; : DI=DI-AM
310 PRINT "DIAMANTES." : MO=MO+BR
320 GOSUB 2000
330 PRINT : PRINT "    A MI ME LLAMAN DEMENCIANO, JUA, JUA " : PRINT
340 PRINT "      ... POR NADA EN PARTICULAR ... " : PRINT
350 PRINT TAB(12);"... JUA, JUA ..." : PRINT
360 PRINT "     A VER CUANTOS AMULETOS Y DIAMANTES TIENE MI CHIQU
ITITO ..."
370 IF AM>=DI THEN 440
380 PRINT : PRINT "      SOLO";AM;"AMULETOS Y";DI;"DIAMANTES." : PRINT
390 PRINT "  NO CREES EN LAS SUPERSTICIONES, EH ?" : PRINT
400 PRINT TAB(4);"... JUA, JUA ... ESO ME GUSTA ..." : PRINT
410 PRINT "       VE POR EL CAMINO DE LA IZQUIERDA Y SALDRAS DE LA
CUEVA."
420 GOSUB 2000
430 GOTO 250
440 PRINT "       VEO QUE ERES INTELIGENTE Y LLEVAS MAS AMULETOS Q
UE DIAMANTES"
450 PRINT TAB(4);"... JUA, JUA ... ESO ME GUSTA ..." : PRINT
460 PRINT "       VE POR EL CAMINO DE LA DERECHA Y SALDRAS DE LA C
UEVA."
470 GOSUB 2000
480 PRINT : PRINT " YO SOY EUFEMISMONDRO ... NO TEMAS ..." : PRINT
490 PRINT "  ENSEÑAME LO QUE LLEVAS EN EL SAQUITO" : PRINT
500 PRINT "        ... SOLO";AM;"AMULETOS ..." : PRINT
510 PRINT "   TOMA ESTOS QUE SON BIEN GRANDES ..." : PRINT : AM=AM+3
520 PRINT "   LOS MEJORES AMULETOS SON LOS QUE MAS PESAN." : PRINT
530 PRINT : PRINT : PRINT " ... Y AHORA ..." : PRINT
540 PRINT "          !! CORRE!!   !! CORRE !!"
550 GOSUB 2000
560 PRINT " HOLA SER DEL EXTERIOR, SOY FESTIBONIA. ME RECUERDAS ?" :
PRINT
570 PRINT "   VEO QUE TIENES"; : IF AM<5 THEN 650
580 PRINT " DEJAME CONTAR ..."; : IF AM>=10 THEN PRINT " DIEZ ..."; : GO
TO 700
590 PRINT AM;"AMULETOS." : PRINT
600 PRINT " NO SE COMO TE HAS ENTERADO, PERO ESE ES EL NUMERO Q
UE DA ";
610 PRINT "MEJOR SUERTE." : PRINT
620 PRINT "    SIGUE POR EL CAMINO DE LA DERECHA Y" : PRINT "PODRA
S ESCAPAR."
630 GOSUB 2000
640 GOTO 860
650 PRINT "AMULETOS." : PRINT
660 PRINT "!ESE ES EL NUMERO QUE PERMITE LA HUIDA!" : PRINT
670 PRINT "         VE POR EL CORREDOR CENTRAL."
680 GOSUB 2000
```



```
690 GOTO 750
700 PRINT TAB(6);AM;"AMULETOS ... MALA COSA ..." : PRINT
710 PRINT "   SAL POR EL CAMINO DE LA IZQUIERDA." : PRINT
720 PRINT "    ES MUY PELIGROSO, PERO LOS OTROS SON MORTALES."
730 GOSUB 2000
740 GOTO 1020
750 PRINT "ASI QUE FESTIBONIA TE DIJO QUE POR AQUI SE SALE ";
760 PRINT "... JE, JE ..." : PRINT
770 PRINT " Y LLEVAS";AM;"AMULETOS, EH ?" : PRINT
780 PRINT TAB(8);"TE DAN SUERTE ?"
790 PRINT " PUEDES QUEDARTELOS, PERO DAME ";
800 IF BR<=AM THEN PRINT "TODOS LOS BRAZALETES" : BR=0 : GOTO 820
810 PRINT AM;"BRAZALETES." : BR=BR-AM
820 PRINT : PRINT : PRINT "    SIGUE POR ESE CAMINO Y LLEGARAS A LA
GRAN VIA."
830 PRINT : PRINT "          !!! JA, JA, JA !!!"
840 GOSUB 2000
850 GOTO 940
860 PRINT "    SOY HISTERICUNDIA." : PRINT : PRINT
870 PRINT "          TOMA DOS MONEDAS DE ORO Y VETE "
880 PRINT " DE AQUI QUE ME PONES NERVIOSA." : MO=MO+2
890 FOR T=1 TO 500 : NEXT T
900 PRINT :PRINT :PRINT :PRINT :PRINT "       !!! QUE TE VAYAS !!!"
910 FOR T=1 TO 300 : NEXT T
920 PRINT : PRINT : PRINT "      !!!!! AAAAAAAAAAAAAAAA !!!!!"
930 GOSUB 2000
940 PRINT : PRINT "          YO SOY IRRISONDRO." : PRINT
950 PRINT " TRAE TODOS ESOS DIAMANTES Y MONEDAS." : PRINT
960 PRINT "  LOS DIAMANTES POR AHORA SON MIOS ..." : DI=0
970 PRINT "   ? A VER CUANTAS MONEDAS TIENES ?" : PRINT
980 PRINT : PRINT TAB(4);MO;" ... " ;
990 PRINT "PUES TOMA ";MO;"DIAMANTES Y" : PRINT "SIGUE POR ESE CAM
INO."
1000 DI=MO : GOSUB 2000
1010 GOTO 250
1020 PRINT : PRINT "    ! MARAVILLATE ANTE JOCOSONDRO !"
1030 FOR T=1 TO 300 : NEXT T
1040 PRINT : PRINT "       ! EL SUBLIME GUARDIAN !"
1050 FOR T=1 TO 800 : NEXT T
1060 PRINT : PRINT : PRINT TAB(20);"...QUE SOY YO."
1070 FOR T=1 TO 800 : NEXT T
1080 PRINT : PRINT : PRINT : PRINT"    PUEDES SALIR, PERO ANTES DEJA E
SOS"
1090 PRINT : PRINT " BRAZALETES, MONEDAS Y AMULETOS, Y "
1100 PRINT " TAMBIEN ESE DIAMANTE QUE A LOS OGROS"
1110 PRINT " NOS TRAE SUERTE."
1120 FOR T=1 TO 2500 : NEXT T
1130 PRINT : PRINT : PRINT "    SIGUE POR ESE CAMINO Y LLEGARAS A"
1140 PRINT "  UNA PARADA DE AUTOBUS. PASA CADA 10"
1150 PRINT "  MINUTOS."
1160 FOR T=1 TO 1800 : NEXT T
1170 PRINT : PRINT : PRINT "    ! AH ! ..."
1180 PRINT TAB(6);"NOSOTROS SOMOS VEGETARIANOS."
1190 DI=DI-1 : MO=0 : AM=0 : BR=0
1200 GOSUB 2000
1210 END
2000 GET A$ : IF A$="" THEN 2000
2010 PRINT : PRINT : PRINT"AHORA TIENES" : PRINT
```

```
2020 PRINT TAB(12);AM;" AMULETOS" : PRINT
2030 PRINT TAB(12);MO;" MONEDAS DE ORO" : PRINT
2040 PRINT TAB(12);DI;" DIAMANTES" : PRINT
2050 PRINT TAB(12);BR;" BRAZALETES"
2060 GET A$ : IF A$="" THEN 2060
2070 (BORRAR LA PANTALLA)
2080 RETURN
```


11. MASTER MIND

El ordenador va a elegir al azar cuatro colores de entre estos seis: amarillo, blanco, marrón, negro, rojo y verde. Tú tienes que adivinar qué colores son y qué posición ocupan. Los colores pueden repetirse.

La representación de cada color se hace con su inicial. Supongamos que el ordenador elige la sucesión de colores BAVV, a la que llamamos *palabra solución* y representamos por S$. Supongamos también que tú propones la palabra AMNV, que representamos por P$. En este caso, aciertas el color y posición de la V; también aciertas el color, pero no la posición, de la A.

Por cada acierto en color y posición, el ordenador escribirá un 1; y por cada color acertado, que no esté bien colocado, escribirá un 0. Veámoslo mejor en estos supuestos:

S$ = BAVV
P$ = AMNV 10
P$ = BVAV 1100

El programa que proponemos está dividido en cinco partes:

1) *Presentación del juego.*

En las líneas 100-150 damos unas breves explicaciones del juego.

2) *Generación de la palabra solución.*

En las líneas 200-220 generamos aleatoriamente la palabra solución S$, de la siguiente manera:

En la línea 210 establecemos la cadena de caracteres:

PAL$="ABMNRV"

formada por los colores disponibles. La variable PAL$ es como la paleta de un pintor en la que el ordenador va a mojar aleatoriamente. En efecto, en la línea 220 se generan cuatro números aleatorios del 1 al 6. Según sea el número generado, el ordenador escoge de la paleta el color correspondiente,

según el orden que ocupa en ella, y forma la palabra S$. Así, por ejemplo, si los cuatro números extraídos son 2, 1, 6, 6, la palabra solución es, precisamente, S$ = BAVV.

3) *Toma de palabra.*

Las líneas 300-395 constituyen la tercera parte del programa, donde, mediante INPUT, el ordenador toma nuestra palabra P$, al tiempo que controla su longitud y los errores de pulsación. Si ha habido equivocaciones, nos saca un mensaje de error y vuelve a pedirnos la palabra. Si la palabra es lícita, añade 1 al contador de intentos E.

4) *Comparación de palabras.*

En la cuarta parte, líneas 400-495, se compara la palabra solución S$ con la palabra propuesta por nosotros P$.

En la línea 420 se hace el desguace de ambas palabras en las letras correspondientes:

La palabra S$ = BAVV, da origen a S$(1) = B, S$(2) = A, S$(3) = V, S$(4) = V.

La palabra P$ = AMNV, da origen a P$(1) = A, P$(2) = M, P$(3) = N, P$(4) = V.

En las líneas 430-450, utilizando un ciclo FOR-NEXT, se cuentan las coincidencias en color y posición. El contador C se encarga de ello. Al mismo tiempo, en el caso de coincidencia, se cambian las letras coincidentes, por * en S$(I) y por + en P$(I), al objeto de evitar que se puedan contar como colores iguales en el recuento que sigue.

Las coincidencias en color, pero no en posición, se anotan en las líneas 460-495 utilizando un doble ciclo FOR-NEXT. La variable T es su contador. Cuando haya una coincidencia en color, es necesario cambiar por + la letra correspondiente de P$(J), para evitar contarla más de una vez.

Partimos de:

```
    1 2 3 4
S$: B A V V
P$: A M N V
```


Después del primer recuento, la situación es ésta:

```
S$: B A V *
P$: A M N +
```

Después del segundo recuento, tenemos:

```
S$: B A V *
P$: + M N +
```

5) *Calificación de la jugada.*

En esta última parte, líneas 500-590, se forma la palabra PI$, a base de 1 y 0, y se imprime al lado de P$. En el caso de que C sea 4, hemos acertado, y el juego finaliza. Si no, se nos pide otra palabra P$.

```
100 REM MASTER MIND
110 PRINT TAB(9);"M A S T E R  M I N D" : PRINT
120 PRINT "ELIJA 4 COLORES DE ENTRE LOS SIGUIENTES:" :
PRINT
130 PRINT "A=AMARILLO   B=BLANCO   M=MARRON"
140 PRINT "N=NEGRO      R=ROJO     V=VERDE" : PRINT
150 PRINT "CALIFICACION:1 COLOR Y POSICION, 0 COLOR" :
PRINT
200 REM ***GENERACION DE LA PALABRA SOLUCION***
210 PAL$="ABMNRV" : S$="" : E=0
220 FOR K=1 TO 4 : X=INT(6*RND+1) : S$=S$+MID$(PAL$,X,
1) : NEXT K
300 REM ***TOMA DE PALABRA***
310 INPUT P$
320 IF LEN(P$)<>4 THEN 370
330 FOR I=1 TO 4
340 P$(I)=MID$(P$,I,1)
350 IF P$(I)="A" OR P$(I)="B" OR P$(I)="M" THEN 390
360 IF P$(I)="N" OR P$(I)="R" OR P$(I)="V" THEN 390
370 ( CURSOR A LA LINEA ANTERIOR );
380 PRINT TAB(7);"SE EQUIVOCO. VUELVA A ESCRIBIR": GOT
O 310
390 NEXT I
395 E=E+1
400 REM***COMPARACION DE PALABRAS***
410 C=0 : T=0
420 FOR K=1 TO 4 : S$(K)=MID$(S$,K,1) : P$(K)=MID$(P$,
K,1) : NEXT K
```

```
430 FOR I=1 TO 4
440 IF S$(I)=P$(I) THEN C=C+1 : S$(I)="*" : P$(I)="+"
450 NEXT I
460 FOR I=1 TO 4
470 FOR J=1 TO 4
480 IF S$(I)=P$(J) THEN T=T+1 : P$(J)="+" : GOTO 495
490 NEXT J
495 NEXT I
500 REM***CALIFICACION DE LA JUGADA***
510 CA$="" : IF C=0 THEN 530
520 FOR K=1 TO C : CA$=CA$+"1" : NEXT K
530 IF T=0 THEN 550
540 FOR K=1 TO T : CA$=CA$+"0" : NEXT K
550 ( CURSOR A LA LINEA ANTERIOR );
560 PRINT TAB(7);CA$
570 IF C<>4 THEN 310
580 PRINT : PRINT "ACERTO VD. DESPUES DE";E;"INTENTOS"
590 END
```


12. LAS SIETE Y MEDIA (7.16)

Haz un programa para jugar a las siete y media contra el ordenador.

Suponemos conocidas las reglas del juego. El programa tiene tres partes:

1.ª Reparto de cartas a los jugadores. Por lo menos uno de los dos no se ha plantado todavía. Ocupa hasta la línea 370.

2.ª Las subrutinas:
   a) Subrutina 700: extracción de una carta.
   b) Subrutina 800: suma de las cartas de un jugador.
   c) Subrutina 1000: escritura en pantalla de las cartas.

3.ª Cuando ya se han plantado los dos jugadores. El ordenador "enseña" sus cartas, que tenía "tapadas", y analiza quién es el ganador. Ocupa desde la línea 380 hasta las subrutinas.

1.ª PARTE: REPARTO DE CARTAS

La variable S indica quién sale, quién recibe la primera carta. Es decisiva a la hora del recuento: si hay empate gana el que sea mano. La primera vez se pregunta mediante una instrucción INPUT quién sale; después el ordenador va alternando las salidas. Esto lo hace en la línea 640:

$$S = ABS(INT(S - 1.5)) + 1$$

Si S valía 1, pasa a valer 2, y viceversa, por lo que cambia el jugador que sale.

Las variables Z1 y Z2 valen 0 mientras el jugador respectivo pide cartas, y valen 1 cuando el jugador se planta. Entonces sólo se dan cartas al otro jugador, hasta que también se plante y se pase al recuento final.

2.ª PARTE: LAS SUBRUTINAS

a) Subrutina 700, de extracción de una carta. Se ha imaginado la baraja como una tabla A(10,4) donde la primera cifra es el número de la carta y la segunda el palo. Así:

A(8,2) es la Sota de Copas.

Al principio, la tabla A(10,4) está llena de ceros. Cada vez que el ordenador saca una carta, coloca un 1 en el lugar correspondiente de la tabla, y cuando lo hace el usuario, un 2. Esto sirve en primer lugar para no dar cartas repetidas. Si ya ha salido la Sota de Copas, no puede volver a salir (es la línea 720). Por otra parte esos valores, 1 y 2, se utilizan en la subrutina 1000.

b) Subrutina 800, de suma de las cartas de un jugador. Ahorra memoria ya que en el programa se acude a ella en cuatro ocasiones. Va guardando en las variables T(1) y T(2) la suma de las cartas del ordenador y del usuario respectivamente. Antes de cada jugada el ordenador consulta la variable T(1), y si es T(1) $\geqslant$ 5, se planta (líneas 220 y 230).

c) Subrutina 1000, de escritura en pantalla de las cartas. El ordenador no nos enseña sus cartas, pero sí las nuestras cada vez que nos toca. Así sabremos las cartas que tenemos y decidimos si seguimos o nos plantamos. Al final, en el recuento de cartas para analizar el ganador, se presentan en pantalla todas nuestras cartas y todas las cartas del ordenador.

3.ª PARTE: RECUENTO FINAL Y GANADOR

Se llega a este punto cuando los dos jugadores se han plantado. La máquina enseña primero todas las cartas del usuario (subrutina 1000) y al final su suma (variable T(2)). A continuación hace lo mismo con las suyas, que tenía tapadas. Luego señala el ganador, con todos los casos posibles (empate si los dos se han pasado, tiene en cuenta quién ha salido si las cartas dan la misma suma, etc.).

La línea 600 muestra en pantalla la puntuación hasta el momento.

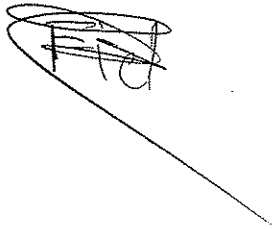
```
10 REM SIETE Y MEDIA
20 ( BORRAR LA PANTALLA )
30 DIM A(10,4),T(2)
40 P1=0 : P2=0
50 INPUT "QUIEN SALE: TU (1) O YO (2)";S
60 Z1=0 : Z2=0 : T(1)=0 : T(2)=0
70 FOR I=1 TO 10 : FOR J=1 TO 4
80 A(I,J)=0
90 NEXT J : NEXT I
100 IF S=1 THEN 150
110 PRINT "YO COJO UNA CARTA" : K=1
120 GOSUB 700
130 GOSUB 800
```



```
140 IF S=1 THEN 200
150 PRINT "TOMA UNA CARTA" : K=2
160 GOSUB 700
170 GOSUB 800
180 GOSUB 1000
190 IF S=1 THEN 110
200 IF K=1 OR Z1=1 THEN 270
210 REM JUEGA LA MAQUINA
220 K=1 : IF T(K)<5 THEN 240
230 Z1=1 : PRINT "ME PLANTO" : GOTO 270
240 PRINT "YO COJO OTRA CARTA"
250 GOSUB 700
260 GOSUB 800
270 REM JUEGA EL USUARIO
280 IF Z2=1 THEN 310
290 INPUT "QUIERES CARTA (1) O TE PLANTAS (2)";W
300 IF W=2 THEN Z2=1
310 IF Z1=1 AND Z2=1 THEN 380
320 IF Z2=1 THEN 210
330 PRINT "TOMA OTRA CARTA" : K=2
340 GOSUB 700
350 GOSUB 800
360 GOSUB 1000
370 GOTO 200
380 REM PRESENTACION Y COMPUTO FINAL
390 PRINT "TUS CARTAS" : K=2
400 FOR I=1 TO 10 : FOR J=1 TO 4
410 IF A(I,J)<>K THEN 430
420 GOSUB 1000
430 NEXT J : NEXT I
440 PRINT : PRINT "SUMAN";T(K)
450 IF K=1 THEN 490
460 IF T(2)>7.5 THEN PRINT "TE HAS PASADO"
470 PRINT : PRINT : PRINT "MIS CARTAS" : K=1
480 GOTO 400
490 IF T(1)>7.5 THEN PRINT "ME PASE"
500 IF T(1)>7.5 AND T(2)>7.5 THEN 590
510 IF T(1)<8 AND T(2)<8 THEN 550
520 IF T(1)>7.5 THEN 540
530 PRINT " TE GANO" : P1=P1+1 : GOTO 600
540 PRINT " ME GANAS" : P2=P2+1 : GOTO 600
550 IF T(1)>T(2) THEN 530
560 IF T(2)>T(1) THEN 540
570 PRINT "POR LA MANO";
580 ON S GOTO 540,530
590 PRINT "TABLAS"
600 PRINT : PRINT "PUNTUACION: YO";P1;"TU";P2
```

```
610 INPUT "OTRA PARTIDA (S/N)";A$
620 IF A$="N" THEN END
630 ( BORRAR LA PANTALLA )
640 S=ABS(INT(S-1.5))+1 : GOTO 60
700 REM EXTRACCION DE UNA CARTA
710 I=INT(10*RND)+1 : J=INT(4*RND)+1
720 IF A(I,J)<>0 THEN 710
730 A(I,J)=K
740 RETURN
800 REM PUNTUACION DE CADA JUGADOR
810 IF I<8 THEN T(K)=T(K)+I
820 IF I>=8 THEN T(K)=T(K)+.5
830 RETURN
1000 REM ESCRITURA DE LA CARTA
1010 IF I=1 THEN A$="AS" : GOTO 1070
1020 IF I<8 THEN A$=STR$(I) : GOTO 1070
1030 ON I-7 GOTO 1040,1050,1060
1040 A$="SOTA" : GOTO 1070
1050 A$="CABALLO" : GOTO 1070
1060 A$="REY"
1070 ON J GOTO 1080,1090,1100,1110
1080 B$=" DE OROS" : GOTO 1120
1090 B$=" DE COPAS" : GOTO 1120
1100 B$=" DE ESPADAS" : GOTO 1120
1110 B$=" DE BASTOS"
1120 PRINT A$+B$
1130 RETURN
```


2. SIMULACIONES



13. LOTERIA (6.24)

Simula la extracción de los tres primeros premios de la Lotería Nacional (números desde el 00000 al 59999). A un mismo número no le pueden tocar dos premios.

El programa es muy sencillo. Consiste en obtener un número aleatorio A comprendido entre 0 y 59999, que será el del primer premio.

A continuación se extrae otro número B, para el segundo premio. Se compara B con A, y si son iguales, se efectúa otra extracción para B. Si no lo son, se pasa a extraer el tercer número C, que a su vez se compara con los anteriores. En el caso de coincidencias, se efectúa una nueva extracción; y si no las hay, se pasa a imprimir los tres números.

```
10 REM LOTERIA
20 ( BORRAR LA PANTALLA )
30 A=INT(60000*RND)
40 B=INT(60000*RND)
50 IF B=A THEN 40
60 C=INT(60000*RND)
70 IF C=A OR C=B THEN 60
80 PRINT "EL PRIMER PREMIO ES ";A
90 PRINT
100 PRINT "EL SEGUNDO PREMIO ES ";B
110 PRINT
120 PRINT "EL TERCER PREMIO ES ";C
130 END
```

14. LANZAMIENTO DE UNA MONEDA (6.16)

Simular el lanzamiento de una moneda N veces. Al final interesa saber el número de lanzamientos y el número de caras y cruces. Durante los lanzamientos puedes hacer que cada vez que salga "cara" aparezca en la pantalla un carácter (C por ejemplo) y por cada "cruz" un signo +.

Simulamos la moneda mediante la función RND. En la línea 70 generamos aleatoriamente 0 ó 1; según salga uno u otro, imprimimos C ó +.

Hemos establecido un contador de caras en la segunda instrucción de la línea 80.

```
10 REM LANZAMIENTO DE UNA MONEDA
20 ( BORRAR LA PANTALLA )
30 K=0
40 INPUT "NUMERO DE LANZAMIENTOS";N
50 ( BORRAR LA PANTALLA )
60 FOR I=1 TO N
70 A=INT(2*RND)
80 IF A=0 THEN PRINT "C"; : K=K+1 : GOTO 100
90 PRINT "+";
100 NEXT I
110 FOR I=1 TO 5 : PRINT : NEXT I
120 PRINT "NUMERO DE LANZAMIENTOS";N
130 PRINT "NUMERO DE CARAS";K
140 PRINT "NUMERO DE CRUCES";N-K
150 END
```


15. ¡QUE QUINIELA!

Obtén los 14 signos de una quiniela de fútbol, teniendo en cuenta una probabilidad del 50% para el signo 1, del 30% para el signo X y del 20% para el signo 2.

La solución es equivalente al lanzamiento de un dado que tuviera diez caras. El signo 1, tiene una probabilidad de 5/10, el signo X de 3/10 y el signo 2 de 2/10. Por tanto, los valores del 1 al 5 son para el 1, del 6 al 8 para la X y el 9 y el 10 para el 2.

Presentación en pantalla:

```
PARTIDO 1 - SIGNO 1
PARTIDO 2 - SIGNO 1
PARTIDO 3 - SIGNO X
PARTIDO 4 - SIGNO 1
PARTIDO 5 - SIGNO X
PARTIDO 6 - SIGNO X
PARTIDO 7 - SIGNO 2
PARTIDO 8 - SIGNO 1
PARTIDO 9 - SIGNO X
PARTIDO 10 - SIGNO 1
PARTIDO 11 - SIGNO 1
PARTIDO 12 - SIGNO 1
PARTIDO 13 - SIGNO 1
PARTIDO 14 - SIGNO 1
```

```
10 REM ! QUE QUINIELA !
20 REM LOS CATORCE PARTIDOS
30 FOR I=1 TO 14
40 REM LANZAMIENTO DEL DADO DE 10 CARAS
50 U=INT(10*RND)+1
60 IF U<=5 THEN PRINT "PARTIDO";I;"- SIGNO 1"
70 IF U>5 AND U<9 THEN PRINT "PARTIDO";I;"- SIGNO X"
80 IF U>8 AND U<=10 THEN PRINT "PARTIDO";I;"- SIGNO 2"
90 NEXT I
100 END
```

16. QUINIELA MILLONARIA

Obtén los 14 signos de una quiniela de fútbol, teniendo en cuenta una probabilidad igual para cada uno de los signos 1, X y 2.

La solución, en este caso, se nos presenta como el lanzamiento de un dado de tres caras. Identificamos los números con los signos de la siguiente manera: el *uno* es el signo X, el *dos* es el signo 1 y el *tres* es el signo 2.

Presentación en pantalla:

```
PARTIDO 1 - SIGNO 2
PARTIDO 2 - SIGNO 1
PARTIDO 3 - SIGNO X
PARTIDO 4 - SIGNO X
PARTIDO 5 - SIGNO X
PARTIDO 6 - SIGNO 2
PARTIDO 7 - SIGNO 1
PARTIDO 8 - SIGNO 2
PARTIDO 9 - SIGNO X
PARTIDO 10 - SIGNO X
PARTIDO 11 - SIGNO 2
PARTIDO 12 - SIGNO 2
PARTIDO 13 - SIGNO 2
PARTIDO 14 - SIGNO 2
```

```
10 REM QUINIELA MILLONARIA
20 REM LOS CATORCE PARTIDOS
30 FOR I=1 TO 14
40 REM LANZAMIENTO DEL DADO DE 3 CARAS
50 U=INT(3*RND)+1
60 ON U GOTO 70,90,110
70 PRINT "PARTIDO";I;"- SIGNO X"
80 GOTO 120
90 PRINT "PARTIDO";I;"- SIGNO 1"
100 GOTO 120
110 PRINT "PARTIDO";I;"- SIGNO 2"
120 NEXT I
130 END
```


17. LA BAÑERA (4.9)

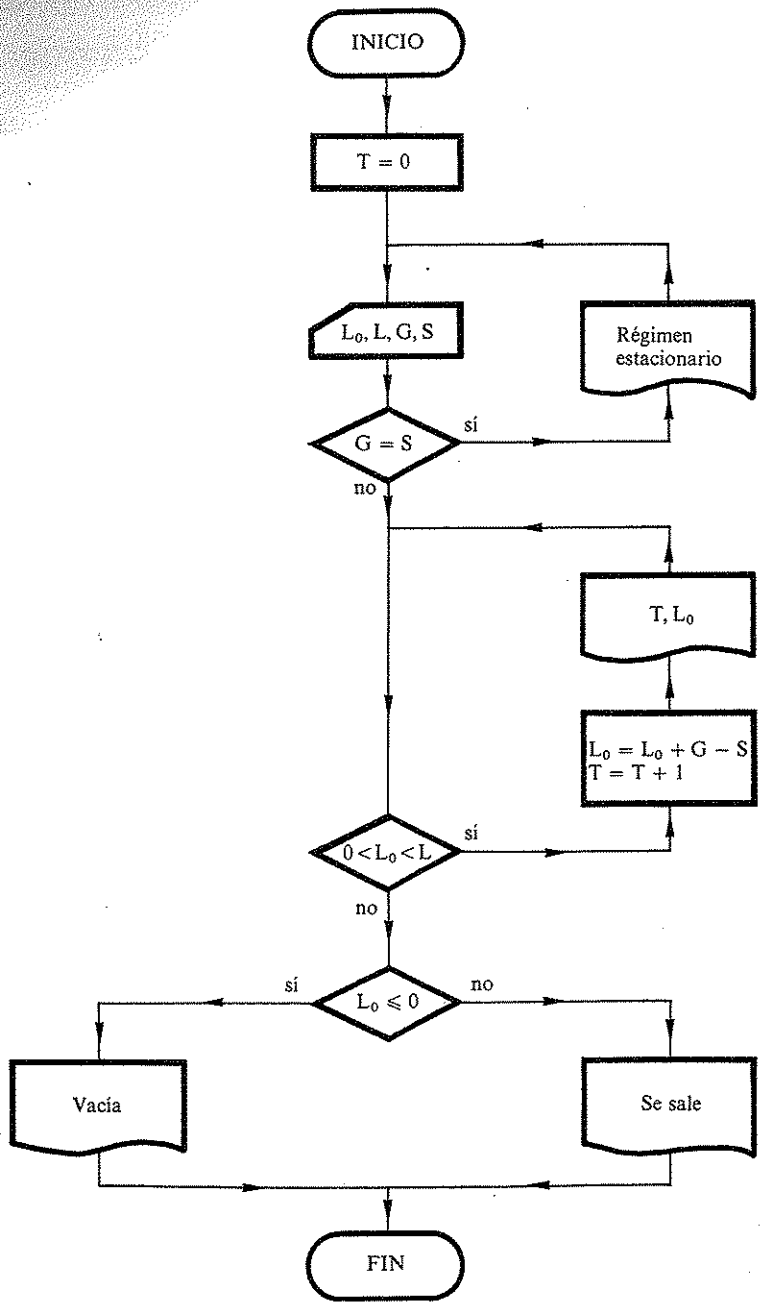
Una bañera tiene grifo y sumidero. Su capacidad es de L litros, el grifo echa G litros por minuto y el sumidero desagua S en el mismo tiempo. Si consideramos el proceso desde que la bañera tiene L_0 litros, ¿cuánto tiempo tarda en vaciarse o llenarse? (Si G y S son iguales ha de salir un aviso en pantalla.) Hemos de hacer un programa que nos dé la situación del sistema minuto a minuto.

Unas de las posibilidades nuevas que ofrecen los ordenadores es la de simular, rápida y sencillamente, situaciones reales en condiciones ventajosas. Resulta sencillo realizar este experimento midiendo el volumen de la bañera y el caudal del grifo, pero es problemática la alteración a voluntad del caudal del desagüe.

Para esta simulación empleamos el siguiente algoritmo:

1. Inicializar el tiempo, T
2. Hacer
   Entrada de datos
   hasta que G y S sean distintos
   hacer, mientras tanto,
   Escribir mensaje de aviso
   Fin de Hacer
3. Mientras $0 < L_0 < L$
   hacer
   $L_0 = L_0 + G - S$
   $T = T + 1$
   Escribir el estado del sistema
   Fin de Mientras
4. Si $L_0 \leqslant 0$
   entonces
   Escribir "Vacía a los";T;"minutos".
   si no
   Escribir "Rebosa a los";T;"minutos".
   Fin de Si
5. FIN

El diagrama de flujo asociado será:

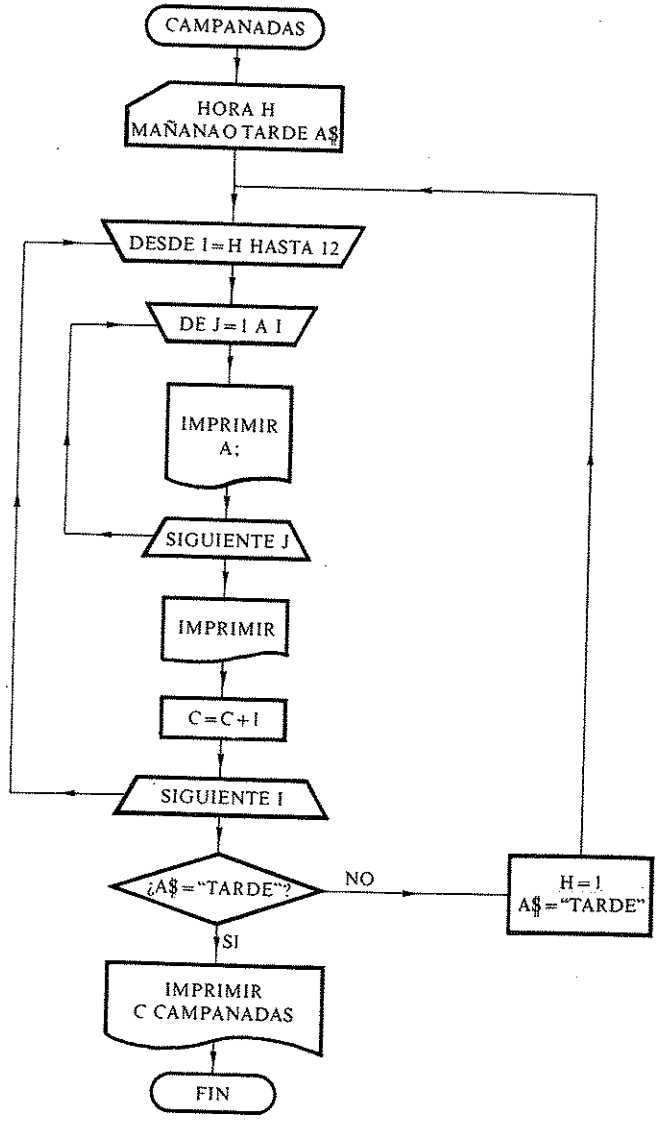


El programa resultante es:

```
10 REM BAÑERA
20 T=O
30 INPUT "CAPACIDAD DE LA BAÑERA (EN LITROS)";L
40 INPUT "CAUDAL DEL GRIFO (L./MIN.)";G
50 INPUT "CAPACIDAD DEL SUMIDERO (L./MIN.)";S
60 INPUT "CONTENIDO DE LA BAÑERA AL COMENZAR";L0
70 IF G=S THEN PRINT "DESAGUA LO MISMO QUE ENTRA. SITUACION ESTA
CIONARIA" :GOTO 30
100 IF NOT (0<L0 AND L0<L) THEN 200
110 L0=L0+G-S :T=T+1
120 PRINT TAB(2);T;"MIN";TAB(20);L0;"LITROS"
130 GOTO 100
200 PRINT :PRINT
210 IF L0<=0 THEN PRINT "VACIA A LOS";T;"MINUTOS." : GOTO 300
220 PRINT "REBOSA A LOS";T;"MINUTOS."
300 END
```

18. CAMPANADAS (4.11)

¿Cuántas campanadas da un relój desde la hora H hasta las doce de la noche?



El ordenador pregunta la hora (H) y a continuación comienza el bucle: deberá dar H campanadas, luego H + 1, y así hasta llegar a 12. Cada hora da I campanadas que se acumulan en la variable C.

Como capricho hemos anidado otro bucle que hace que a la hora I, en la que se dan I campanadas, se impriman I signos en una línea, en nuestro caso la letra A.

El bucle se acaba al llegar a las 12 horas. Viene ahora la consideración de si era por la mañana o por la tarde: si era por la tarde, se puede terminar el programa imprimiendo el valor de C; pero si era por la mañana, quedan todavía 12 horas por transcurrir. Hay que hacer H = 1 y reiterar el bucle anterior.

Una observación: para que esta reiteración se efectúe una sola vez, habrá que cambiar la variable A$ a "TARDE"; si no el proceso seguiría hasta el infinito. Dicho de otra forma: al llegar a las 12 del mediodía hay que colocar el reloj a la una (H = 1) de la tarde (A$ = "TARDE").

```
QUE HORA ES? 10
MAÑANA O TARDE? MAÑANA


 A  A  A  A  A  A  A  A  A  A
 A  A  A  A  A  A  A  A  A  A  A
 A  A  A  A  A  A  A  A  A  A  A  A


 A
 A  A
 A  A  A
 A  A  A  A
 A  A  A  A  A
 A  A  A  A  A  A
 A  A  A  A  A  A  A
 A  A  A  A  A  A  A  A
 A  A  A  A  A  A  A  A  A
 A  A  A  A  A  A  A  A  A  A
 A  A  A  A  A  A  A  A  A  A  A
 A  A  A  A  A  A  A  A  A  A  A  A

DA 111  CAMPANADAS
```

```
10 REM CAMPANADAS
20 PRINT : PRINT
30 INPUT "QUE HORA ES";H
40 INPUT "MAÑANA O TARDE";A$
50 C=0
60 PRINT : PRINT
70 FOR I=H TO 12
80 FOR J=1 TO I
90 PRINT " A ";
100 NEXT J
110 PRINT
120 C=C+I
130 NEXT I
140 IF A$="TARDE" THEN 180
150 A$="TARDE"
160 H=1
170 GOTO 60
180 PRINT : PRINT
190 PRINT "DA";C;" CAMPANADAS"
200 END
```


19. SUMA DE LOS PUNTOS DE UN DADO

Se arroja un dado tres veces consecutivas y se considera la suma de los puntos obtenidos. Esta varía entre 3 y 18. Hallar cuántas veces puede aparecer cada suma.

La suma 3 aparece una sola vez: cuando obtenemos (1,1,1); la suma 4 aparece tres veces: (1,1,2), (1,2,1) y (2,1,1); etc.

Podemos encargar al ordenador este trabajo de recuento. Para ello formará todas las ternas posibles (I,J,K), variando los índices de 1 a 6 con un triple ciclo FOR-NEXT. Cada vez que obtenga una terna (I,J,K), sumará sus elementos:

$$S = I + J + K$$

Para hacer el recuento de cada suma S, se utiliza la variable de índice F(S) (inicializada previamente en la línea 40):

$$F(S) = F(S) + 1$$

Una vez ejecutados los tres ciclos, encontramos la frecuencia de cada suma en la lista F(S); y se las puede imprimir con un simple ciclo FOR-NEXT.

```
10 REM FRECUENCIAS DE LAS SUMAS DE LOS PUNTOS DE UN DADO
20 DIM F(18)
30 ( BORRAR LA PANTALLA )
40 FOR S=3 TO 18 : F(S)=0 : NEXT S
50 FOR I=1 TO 6
60 FOR J=1 TO 6
70 FOR K=1 TO 6
80 S=I+J+K
90 F(S)=F(S)+1
100 NEXT K
110 NEXT J
120 NEXT I
130 FOR S=3 TO 18
140 PRINT "LA SUMA ";S;" APARECE ";F(S);" VECES"
150 NEXT S
160 END
```

20. FRECUENCIAS RELATIVAS DE UN DADO

Simula el lanzamiento de un dado N veces; cuenta las frecuencias de cada número utilizando una variable de índice e imprime las frecuencias relativas correspondientes. Te sugerimos repetir la simulación varias veces con valores crecientes de N y observar la estabilización de las frecuencias relativas.

Generamos N enteros aleatorios comprendidos entre 1 y 6 y los guardamos en la variable con índice F(X), que nos cuenta las frecuencias absolutas (instrucciones 50-80).

En la línea 200 calculamos las frecuencias relativas FR(X), redondeando los resultados a cuatro cifras decimales.

```
10 REM F.RELATIVAS DE UN DADO
20 DIM F(6)
30 ( BORRAR LA PANTALLA )
40 INPUT "CUANTAS TIRADAS";N
50 FOR I=1 TO N
60 X=INT(RND*6)+1
70 F(X)=F(X)+1
80 NEXT I
90 ( BORRAR LA PANTALLA )
100 PRINT TAB(9);"F.RELATIVAS DE UN DADO"
110 PRINT TAB(9);"======================"
120 PRINT
130 FOR X=1 TO 6
140 PRINT TAB(3+(X-1)*6);X;
150 NEXT X
160 PRINT
170 FOR I=1 TO 40 : PRINT "="; : NEXT I
180 PRINT
190 FOR X=1 TO 6
200 FR(X)=INT(F(X)/N*1E4+0.5)/1E4
210 PRINT TAB(2+6*(X-1));FR(X);
220 NEXT X
230 FOR I=1 TO 10 : PRINT : NEXT I
240 PRINT "QUIERES EFECTUAR OTRA TIRADA:(S/N)"
250 GET R$
260 IF R$<>"S" AND R$<>"N" THEN 250
270 IF R$="S" THEN 30
280 PRINT : PRINT "ADIOS."
290 END
```


21. ESTABILIZACION DE LAS FRECUENCIAS (6.19)

Vamos a simular de nuevo el lanzamientò de una moneda, pero esta vez el programa se parará cuando se estabilicen las frecuencias relativas. Después de cada lanzamiento se halla la frecuencia relativa de las caras. Cuando esa frecuencia difiera de la del lanzamiento anterior en menos de 0.01 el programa parará, indicando el número de lanzamientos, el de caras y cruces, y las dos últimas frecuencias relativas de caras redondeadas a 4 decimales. Piensa las dificultades que pueden presentarse en las primeras tiradas.

Si tiramos una moneda y calculamos la frecuencia relativa de las caras, (o de las cruces) observamos que a medida que aumenta el número de tiradas, dichas frecuencias tienden a estabilizarse; o dicho de otro modo, la diferencia entre dos frecuencias relativas consecutivas se aproxima a cero.

En las primeras tiradas puede ocurrir que dos frecuencias relativas consecutivas sean iguales. Por ejemplo, si en las tres primeras tiradas no ha salido ninguna cara, las frecuencias relativas son cero, y si han salido tres caras, las frecuencias son iguales a uno. En ambos casos, la diferencia entre dos frecuencias relativas es cero; cumplen la condición pedida, pero no hemos llegado a la estabilización porque la igualdad se romperá en las tiradas siguientes. Por tanto, consideramos que las frecuencias se estabilizan, con el grado de aproximación deseado, cuando una frecuencia difiera de la anterior en menos de 0.01, y ambas sean diferentes.

En nuestro programa la variable N recoge el número de tiradas y C el número de caras correspondiente a N tiradas.

Mediante un ciclo FOR-NEXT, líneas 40-90, simulamos dos lanzamientos. Si el número A es cero, contabilizamos una cara. Calculadas las frecuencias relativas FR(1) y FR(2), en la línea 100 hallamos el valor absoluto de su diferencia, que sometemos en la línea 110 a comparación con las condiciones exigidas. En caso de que se cumplan, redondeamos las frecuencias e imprimimos los resultados. Si no, hacemos FR(1) igual a FR(2) y volvemos al ciclo, generando un sólo número A, para lo cual a la variable I le asignamos el valor 2, que mantendrá hasta que finalice el problema.

```
10 REM ESTABILIZACION FR. RELATIVAS
20 ( BORRAR LA PANTALLA )
30 N=0 : C=0 : K=1 : DIM FR(2)
40 FOR I=K TO 2
50 A=INT(2*RND)
60 N=N+1
70 IF A=0 THEN C=C+1
80 FR(I)=C/N
90 NEXT I
100 D=ABS(FR(1)-FR(2))
110 IF D<.01 AND FR(1)<>FR(2) THEN 130
120 FR(1)=FR(2) : K=2 : GOTO 40
130 FR(1)=INT(FR(1)*1E4+.5)/1E4 : FR(2)=INT(FR(2)*
1E4+.5)/1E4
140 FOR I=1 TO 10 : PRINT : NEXT I
150 PRINT "HEMOS EFECTUADO";N;"LANZAMIENTOS"
160 PRINT : PRINT "HEMOS OBTENIDO";C;"CARAS Y";N-C
;"CRUCES"
170 PRINT : PRINT "LAS DOS ULTIMAS FRECUENCIAS SON
";FR(1);"Y";FR(2)
180 END
```


22. LA COLECCION DE CROMOS (7.20)

¿Como cuántos cromos hay que comprar para completar un álbum de cien?

Suponemos que los cromos se adquieren de uno en uno y que todos se presentan con igual probabilidad. Los cromos están numerados del 1 al 100; así que comprar un cromo es lo mismo que extraer al azar un número entre el 1 y el 100. Esto lo puedes conseguir mediante la instrucción:

$$N = INT(100 * RND + 1) \qquad (1)$$

El álbum formado por las casillas donde hay que pegar los cromos está representado por la variable de índice C(N), cuyos elementos constituyen la lista:

$$C(1),\ C(2),\ \cdots,\ C(100)$$

Inicialmente todos los elementos de esta lista valen cero (están sin cromo). Si el número N obtenido en la instrucción (1) es, por ejemplo, 23, sumaremos a la variable C(23) una unidad:

$$C(N)=C(N)+1$$

Ahora en la casilla 23 habrá, por lo menos, un cromo (exactamente uno, si antes no había ninguno).

La intuición nos dice que tenemos que extraer bastantes más de cien cromos, porque muchos los vamos a obtener repetidos; así que la instrucción (1) se tendrá que ejecutar muchas veces. Hay que detener el proceso cuando todas las casillas estén completas, o lo que es igual: cuando todas las variables C(1), ···, C(100) contengan valores distintos de 0. Así pues, una forma de terminar es comprobar que todos los números de la lista C(N) son no nulos. Hay una forma elegante de hacerlo: cada vez que obtengamos un *cromo nuevo* (que no haya aparecido antes) contaremos uno más en el contador de cromos nuevos:

$$IF\ C(N)=0\ THEN\ K=K+1$$

La compra se detiene cuando K es 100:

IF K = 100 THEN (hacia el final)

Es claro que hay que establecer también un contador A que lleve la cuenta del total de cromos adquiridos y, al final, imprimir el valor de A.

Puedes completar el programa imprimiendo también los valores de la lista C(1), ···, C(100) en forma de una tabla de 10 × 10.

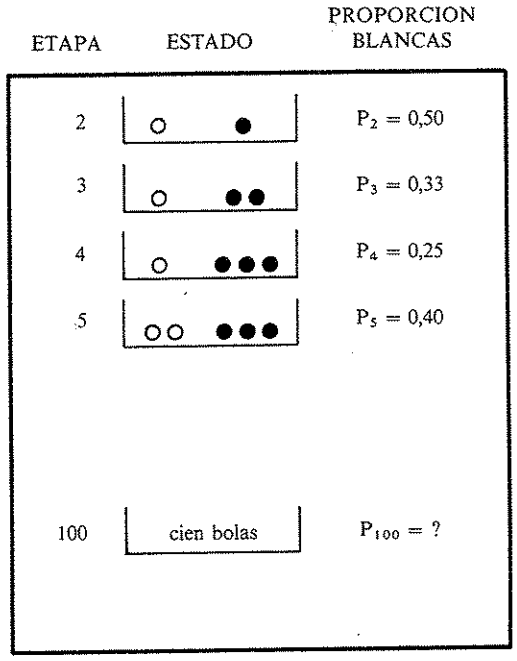
Así tendrás idea de cuántas veces se repite cada cromo.

```
10 REM COLECCION DE CROMOS
20 ( BORRAR LA PANTALLA )
30 PRINT "COMO CUANTOS CROMOS HAY QUE ADQUIRIR"
40 PRINT "PARA COMPLETAR UN ALBUM DE CIEN ?"
50 A=0 : K=0 : DIM C(100)
60 A=A+1
70 N=INT(100*RND+1)
80 IF C(N)=0 THEN K=K+1
90 C(N)=C(N)+1
100 IF K<100 THEN 60
110 PRINT "NUMERO DE CROMOS ADQUIRIDOS:";A
120 PRINT
130 REM ESCRITURA DE LA TABLA
140 FOR I=0 TO 9
150 FOR J=1 TO 10
160 PRINT TAB(4*(J-1));C(10*I+J);
170 NEXT J
180 PRINT
190 PRINT
200 NEXT I
210 END
```


23. EL DESTINO DE UNA URNA (7.21)

Se parte de una urna cuya composición inicial es una bola blanca y otra negra. Se extrae una bola al azar y se la devuelve a la urna acompañada de otra del mismo color. Se repite el proceso hasta que en la urna haya cien bolas.

Podemos considerar que la urna es un sistema que cambia de estado a lo largo de las distintas etapas. Desde la etapa inicial (etapa 2, porque en la urna hay dos bolas) hasta la etapa final (etapa 100) la urna atraviesa distintos estados, entendiendo por estado cada una de las composiciones posibles de la urna; así, por ejemplo, en la etapa 3 hay dos estados posibles: una blanca y dos negras o dos blancas y una negra.



En la etapa 100 hay 99 estados posibles. ¿A cuál de estos 99 estados llegará la urna? Imposible predecirlo: todos ellos son igualmente probables; es decir, la probabilidad de que la urna llegue a un estado final concreto es 1/99.

Aunque al principio no podamos predecir qué estado final alcanzará la urna, sí que lo podemos hacer (con cierta seguridad) cuando hayan transcurrido algunas etapas, porque se puede comprobar, mediante simulación, que la proporción de bolas blancas se estabiliza *a lo largo del proceso.*

Diseña un programa de modo que en cada etapa se imprima en pantalla la proporción de bolas blancas. Comprobarás el hecho notable de que dicha proporción se estabiliza a partir de una cierta etapa.

En el programa que ofrecemos, las variables T, B y P representan, respectivamente, el número total de bolas, el número de blancas y la proporción de blancas en cada etapa. Las variables B y P se inicializan en la línea 30. Observa que P también representa la probabilidad de extraer de la urna una bola blanca.

Las 98 extracciones se hacen por medio de un ciclo FOR-NEXT (Líneas 60-110), en el que T varía desde 3 hasta 100.

En cada etapa, se genera un número aleatorio que decide si sale blanca o negra; si este número es menor que P, la bola extraída es blanca y se suma 1 al contador B.

En la línea 80 se calcula la probabilidad P, y en la 90 se imprime T y la probabilidad redondeada a dos decimales.

En la línea 100 hemos colocado un ciclo FOR-NEXT, cuyo único propósito es retardar la extracción de la siguiente bola para que el proceso discurra lentamente y pueda ser observado en la pantalla.

```
10 REM EL DESTINO DE UNA URNA
20 ( BORRAR LA PANTALLA )
30 B=1 : P=0.50
40 PRINT "ETAPA","P.BLANCAS"
50 PRINT 2,0.50
60 FOR T=3 TO 100
70 IF RND<P THEN B=B+1
80 P=B/T
90 PRINT T,INT(100*P+0.5)/100
100 FOR I=1 TO 500 : NEXT I
110 NEXT T
120 END
```


24. ELECCIONES (7.10)

A unas elecciones se presentan N candidatos. Simular la votación y realizar el escrutinio. Muéstrese al final, la lista ordenada de los candidatos según los votos obtenidos.

La variable C$(I), (I = 1,2,3,...,N) recoge los nombres de los N candidatos y la variable C(I) el número de votos conseguido por el candidato I.

Introducidos el número de candidatos N, el de votantes V, y los nombres de los candidatos, se generan V números aleatorios enteros (de 1 a N), líneas 110-140 y se van contando los votos de cada candidato. En las líneas 160-230 se ordena la lista C(I) de mayor a menor, lo que permite ordenar simultáneamente C$(I). Ordenadas ambas listas, se presentan en pantalla, líneas 250-330.

```
10 REM ELECCIONES
20 ( BORRAR LA PANTALLA )
30 INPUT "CUANTOS CANDIDATOS SE PRESENTAN";N
40 PRINT : INPUT "CUANTOS CIUDADANOS VOTAN";V
50 DIM C$(N),C(N)
60 FOR I=1 TO N : C$(I)="" : C(I)=0 : NEXT I
70 FOR I=0 TO 39 : PRINT TAB(I);"="; : NEXT I
80 FOR I=1 TO N
90 PRINT : PRINT "NOMBRE DEL CANDIDATO";I; :
INPUT C$(I)
100 NEXT I
110 FOR I=1 TO V
120 A=INT(RND*N+1)
130 C(A)=C(A)+1
140 NEXT I
150 REM ORDENACION DE LA LISTA
160 FOR I=1 TO N-1
170 FOR J=1 TO N-I
180 IF C(J)>=C(J+1) THEN 220
190 X=C(J) : X$=C$(J)
200 C(J)=C(J+1) : C$(J)=C$(J+1)
210 C(J+1)=X : C$(J+1)=X$
220 NEXT J
230 NEXT I
240 REM LISTA ORDENADA
```

```
250 PRINT : PRINT
260 FOR I=1 TO N
270 PRINT C$(I);
280 FOR J=1 TO (25-LEN(C$(I)))
290 PRINT ".";
300 NEXT J
310 PRINT TAB(25);C(I);"VOTOS"
320 PRINT
330 NEXT I
340 END
```


25. **BARAJA** (7.15)

De una baraja española, el ordenador debe repartir cinco cartas a cada uno de los cuatro jugadores de una partida.

```
JUGADOR NUMERO 1 :
                    SEIS DE ESPADAS
                    CUATRO DE BASTOS
                    DOS DE OROS
                    TRES DE COPAS
                    CINCO DE BASTOS
JUGADOR NUMERO 2 :
                    DOS DE COPAS
                    CABALLO DE BASTOS
                    CUATRO DE COPAS
                    CABALLO DE COPAS
                    AS DE OROS
JUGADOR NUMERO 3 :
                    AS DE COPAS
                    TRES DE ESPADAS
                    DOS DE BASTOS
                    SEIS DE COPAS
                    CUATRO DE ESPADAS
JUGADOR NUMERO 4 :
                    CABALLO DE OROS
                    CINCO DE ESPADAS
                    REY DE BASTOS
                    DOS DE ESPADAS
                    REY DE ESPADAS
```

Hemos dividido el programa en tres partes: creación de la baraja, reparto de las cartas e impresión del reparto.

Creación de la baraja.

La baraja va a estar constituida por los elementos de la tabla B$(I,J); I = 1, 2, ···, 10; J = 1, 2, 3, 4:

AS DE OROS AS DE COPAS AS DE ESPADAS AS DE BASTOS
DOS DE OROS DOS DE COPAS DOS DE ESPADAS DOS DE BASTOS
... ...
REY DE OROS REY DE COPAS REY DE ESPADAS REY DE BASTOS

No es necesario teclear los cuarenta elementos: el propio programa crea la tabla a partir de dos líneas DATA:

150 DATA AS, DOS, TRES,..., SOTA, CABALLO, REY
160 DATA DE OROS, DE COPAS, DE ESPADAS, DE BASTOS

Los elementos de la primera DATA se guardan en la variable de índice N$(I) (líneas 40, 50 y 60); y los de la segunda se ponen en la variable de índice P$(J).

Con un doble ciclo (líneas 100-140) se construyen los cuarenta elementos de la baraja, utilizando para ello la instrucción:

B$(I,J)=N$(I)+" "+P$(J)

Reparto de las cartas.

Por medio de un doble ciclo (líneas 180-240) se reparten las cinco cartas a los cuatro jugadores, de la siguiente manera: se obtiene aleatoriamente un par de números (I,J), que marca la posición del naipe en la tabla B$(I,J); si el naipe no ha sido extraído previamente, se guarda en la tabla A$(L,M), L=1, 2, 3, 4, M=1, 2, 3, 4, 5; y se hace B$(I,J)="0" para indicar que ese naipe ya no se puede extraer. Si ocurriera que el naipe de la posición (I,J) ya hubiera sido extraído, entonces se prueba con otra pareja aleatoria. Claramente se ve que en la tabla A$(L,M) se van guardando las cartas correspondientes a los cuatro jugadores.

Impresión del reparto.

Las líneas 260-310 sirven para imprimir la tabla A$(L,M) en la pantalla, con lo cual podemos visualizar las cartas que han correspondido a cada jugador.

```
10 REM BARAJA
20 DIM N$(10),P$(4),B$(10,4),A$(4,5)
30 REM CREACION DE LA BARAJA
40 FOR I=1 TO 10
50 READ N$(I)
60 NEXT I
70 FOR J=1 TO 4
80 READ P$(J)
90 NEXT J
100 FOR I=1 TO 10
```



```
110 FOR J=1 TO 4
120 B$(I,J)=N$(I)+" "+P$(J)
130 NEXT J
140 NEXT I
150 DATA AS,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,SO
TA,CABALLO,REY
160 DATA DE OROS,DE COPAS,DE ESPADAS,DE BASTOS
170 REM REPARTO DE LAS CARTAS
180 FOR M=1 TO 5
190 FOR L=1 TO 4
200 I=INT(10*RND+1) : J=INT(4*RND+1)
210 IF B$(I,J)="0" THEN 200
220 A$(L,M)=B$(I,J) : B$(I,J)="0"
230 NEXT L
240 NEXT M
250 REM IMPRESION DEL REPARTO
260 FOR L=1 TO 4
270 PRINT "JUGADOR NUMERO";L;":"
280 FOR M=1 TO 5
290 PRINT TAB(18);A$(L,M)
300 NEXT M
310 NEXT L
320 GOTO 320
330 END
```

26. RUIDO DE UNA INFORMACION

Supongamos un canal por donde circula una información. El canal transmite únicamente "0" ó "1". A consecuencia de un ruido parásito, la emisión de un "0" es recibida como "1" y viceversa, siendo la probabilidad de perturbación de 0,1. Si deseamos comunicar una información importante, que se compone solamente de la señal "1", es interesante proteger esta información contra el ruido transmitiendo por ejemplo "1111111" en lugar de "1".

El receptor, debido a la perturbación, podrá recibir señales que contengan "1" y "0". Por ejemplo "1111111" se puede transformar en "1111001". La secuencia recibida se interpretará como "1", si se reciben más "1" que "0", y como "0", en caso contrario.

¿Cuál es la probabilidad de que la información se reciba correctamente?

EMISOR		RECEPTOR
"1111111"	CANAL	"1111001"

Simularemos la recepción de 1000 señales perturbadas. Como la probabilidad de que el "1" se reciba como "1" es 0,9 y como "0" 0,1, obtendremos una señal perturbada con las probabilidades indicadas extrayendo aleatoriamente de la variable de cadena A$="1111111110" un elemento. En la línea 70 generamos un número entero aleatorio $1 \leqslant P \leqslant 10$, y en la 80 seleccionamos de A$, mediante la función MID$, el elemento que ocupa el lugar P empezando por la izquierda. Repitiendo esta operación siete veces obtenemos una secuencia de "1" y "0".

En la línea 90 contabilizamos los unos que van saliendo y si ese número es mayor o igual a 4, la señal es "buena" y se registra en la variable F.

La probabilidad de que la señal sea recibida correctamente se obtiene dividiendo las señales favorables por las totales F/1000.


```
10 REM RUIDO DE UNA INFORMACION
20 ( BORRAR LA PANTALLA )
30 A$="1111111110" : F=0
40 FOR I=1 TO 1000
50 C=0
60 FOR J=1 TO 7
70 P=INT(10*RND)+1
80 S$=MID$(A$,P,1)
90 IF S$="1" THEN C=C+1
100 NEXT J
110 IF C>=4 THEN F=F+1
120 NEXT I
130 FOR I=1 TO 10 : PRINT : NEXT I
140 PRINT "HAN SALIDO";F;"SEÑALES FAVORABLES"
150 PRINT : PRINT
160 PRINT "LA PROBABILIDAD DE QUE LA SEÑAL" : PRINT
170 PRINT "SEA RECIBIDA CORRECTAMENTE ES";F/1000
180 END
```

27. DEFORMACION DE UN MENSAJE

Al transmitir un mensaje de una persona a otra, existe una pequeña probabilidad de alterar una palabra. Cuando la transmisión se hace a través de una larga cadena de personas, la probabilidad de que la última reciba el mensaje totalmente deformado es bastante grande.

Se trata de hacer un programa que simule, en una cadena de veinte personas, la transmisión del mensaje:

LA CARTA LLEGARA EL LUNES

Se supone que las siguientes palabras son intercambiables dentro de cada grupo: (CARTA, TARTA, MARTA), (LLEGARA, LLOVERA, LLAMARA), (EL, DEL, AL) y (LUNES, MARTES, MIERCOLES). Una palabra permanece invariable con probabilidad 0,94 y se cambia por una de las otras dos, ambas con probabilidad 0,03. La palabra LA permanece inalterable.

Después de ejecutar el programa, he aquí lo que han entendido cada una de las personas de la cadena:

```
MENSAJE INICIAL :
LA CARTA LLEGARA EL LUNES

LA CARTA LLEGARA EL LUNES
LA CARTA LLEGARA EL LUNES
LA TARTA LLOVERA EL LUNES
LA TARTA LLOVERA EL LUNES
LA TARTA LLOVERA EL LUNES
LA TARTA LLOVERA EL LUNES
LA TARTA LLOVERA AL LUNES
LA TARTA LLAMARA AL LUNES
LA TARTA LLAMARA AL LUNES
LA TARTA LLAMARA AL LUNES
LA TARTA LLAMARA AL LUNES
LA TARTA LLAMARA AL LUNES
LA TARTA LLAMARA DEL MARTES
LA TARTA LLAMARA DEL MARTES
LA TARTA LLAMARA DEL MARTES
LA TARTA LLAMARA DEL MARTES
LA MARTA LLAMARA DEL MARTES
```


Para diseñar el programa hemos considerado la tabla T$(I,J); I=0,1,2; J=1,2,3,4,5; de tres filas y cinco columnas, siguiente:

	1	2	3	4	5
0	LA	CARTA	LLEGARA	EL	LUNES
1	LA	TARTA	LLOVERA	DEL	MARTES
2	LA	MARTA	LLAMARA	AL	MIERCOLES

El estado del mensaje en cada etapa viene determinado por el conocimiento, para cada columna, del número de fila a la que pertenece la palabra que está en el mensaje. Así, por ejemplo, la sucesión de números 1,2,2,0,1 determina el mensaje:

LA MARTA LLAMARA EL MARTES

Si establecemos la lista I(1), I(2), I(3), I(4), I(5), (I(J), J = 1,2,3,4,5), el contenido de ella (0,1 ó 2) nos da el estado del mensaje.

Para pasar de una etapa a la siguiente, hacemos variar aleatoriamente el contenido de cada una de las variables I(J), de acuerdo con las probabilidades preestablecidas. Para ello sumamos a I(J) el número aleatorio N, que puede ser 0, 1 ó 2 con probabilidades 0,94, 0,03 y 0,03, respectivamente. Para evitar que la suma I(J)+N exceda de 2, tomamos el resto de la división por 3 (línea 230).

```
10 REM DEFORMACION DE MENSAJES
20 ( BORRAR LA PANTALLA )
30 DIM T$(2,5),I(5)
40 REM SE CARGA LA TABLA
50 FOR I=0 TO 2
60 FOR J=1 TO 5
70 READ T$(I,J)
80 NEXT J
90 NEXT I
100 REM SE INICIALIZAN EN 0 LOS PRIMEROS INDICES
110 FOR J=1 TO 5 : I(J)=0 : NEXT J
120 REM SE ESCRIBE EL MENSAJE INICIAL
130 PRINT "MENSAJE INICIAL :"
140 FOR J=1 TO 5 : PRINT T$(I(J),J)+" "; : NEXT J
150 PRINT : PRINT
160 REM SE GENERAN LOS MENSAJES SUCESIVOS
```

```
170 FOR K=1 TO 20
180 FOR J=1 TO 5
190 X=RND
200 IF X<0.94 THEN N=0 : GOTO 230
210 IF X<0.97 THEN N=1 : GOTO 230
220 N=2
230 I(J)=I(J)+N-INT((I(J)+N)/3)*3
240 PRINT T$(I(J),J)+" ";
250 NEXT J
260 PRINT
270 NEXT K
280 DATA LA, CARTA, LLEGARA, EL, LUNES
290 DATA LA, TARTA, LLOVERA, DEL, MARTES
300 DATA LA, MARTA, LLAMARA, AL, MIERCOLES
310 END
```


28. RUMORES

Deseamos estudiar la propagación de un rumor en una población de N habitantes. Supondremos que todo el que se entera del rumor lo cuenta a todas las personas que se encuentra, hasta que alguna de ellas ya lo conoce; desde ese momento dejará de contarlo. Queremos saber, tras cada unidad de tiempo, cuántas personas desconocen el rumor (D), cuántas lo propagan (P) y cuántas lo conocen pero no lo cuentan (C). También deseamos saber cuánto tiempo pasa antes de que se deje de hablar del asunto y cuánta gente no habrá oído nunca el rumor.

Podemos suponer la población dividida en tres clases, D, P y C, y expresar en una terna (D, P, C) el estado del sistema en cada momento; así, (600, 320, 80) significa que 600 personas desconocen el rumor, 320 lo propagan y 80 lo conocen pero no lo cuentan (callan).

Al paso de la primera categoría a la segunda lo llamaremos "contagio", y al paso de ésta a la tercera "inmunización". El contagio se produce por un contacto DP; el número máximo de posibles contactos de este tipo es D*P y, para un tiempo T pequeño, la disminución en D es proporcional a D*P con factor de proporcionalidad F (al que llamaremos "factor de contagio"); tras T los desconocedores disminuyen:

$$D = D - D*P*F$$

y, si no existiera inmunización los propagadores pasarían a ser:

$$P = P + D*P*F$$

La inmunización se produce:

a) Por contacto entre propagadores. Hay $\frac{P*(P-1)}{2}$ contactos de este tipo.

b) Por contacto entre propagador y ex-propagador. Hay P*C contactos de éstos.

Sumando ambos tipos de inmunización resultará:

$$P*C + \frac{P*(P-1)}{2}$$

Durante el tiempo T, el número de propagadores P aumentará lo mismo

que disminuya D, D*P*F, y disminuirá en F*(P*C+P*(P−1)/2). El cambio de estado viene dado por la correspondencia:

$$D \longrightarrow D - D*P*F$$
$$P \longrightarrow P + D*P*F - F*(P*C + P*(P-1)/2)$$
$$C \longrightarrow C + F*(P*C + P*(P-1)/2)$$

```
10 REM RUMOR
20 INPUT "POBLACION";N
30 INPUT "FACTOR DE CONTAGIO";F
40 T=0 : P=1 : D=N-1 : C=0
50 ( BORRAR LA PANTALLA )
60 PRINT "TIEMPO   DESCONOCEN   PROPAGAN    CALLAN"
70 PRINT TAB(1);T;TAB(10);INT(D);TAB(22);INT(P);TAB
(34);INT(C)
80 CO=D*P*F : REM CONTAGIO
90 I=((P*C)+(P*(P-1)/2))*F : REM INMUNIZACION
100 T=T+1 : D=D-CO : P=P+CO-I : C=C+I
110 PRINT TAB(1);T;TAB(10);INT(D);TAB(22);INT(P);TA
B(34);INT(C)
120 IF P>=1 THEN 80
130 END
```

Los resultados que nos proporcione el programa serán absurdos si F es grande. Cuanto mayor es una población, menor es la fracción de ésta a la que cada individuo tiene acceso directo y, consecuentemente, menor será el factor de contagio, F. Recomendamos tomar F aproximadamente igual al inverso de la población, o más pequeño, nunca mayor.

Teniendo en cuenta lo anterior, hemos ejecutado dos veces el programa, obteniendo:

```
POBLACION? 10000
FACTOR DE CONTAGIO? 1E-04
TIEMPO    DESCONOCEN    PROPAGAN     CALLAN
  0         9999          1           0
  1         9998          1           0
  2         9996          3           0
  3         9992          7           0
  4         9984          15          0
  5         9968          31          0
  6         9936          63          0
  7         9872          128         0
  8         9747          251         1
  9         9502          492         4
  10        9034          948         16
  11        8177          1759        63
```



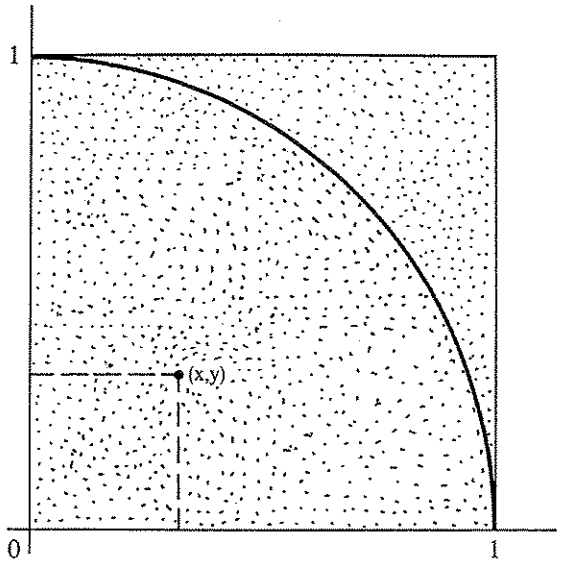
```
POBLACION? 10000
FACTOR DE CONTAGIO? 1E-04
TIEMPO    DESCONOCEN    PROPAGAN     CALLAN
  12        6738          3032        228
  13        4694          4547        758
  14        2559          5303        2136
  15        1202          4121        4675
  16        706           1840        7452
  17        576           429         8993
  18        551           58          9389
  19        546           6           9444
  20        546           0           9451
```

```
POBLACION? 100000
FACTOR DE CONTAGIO? 1E-05
TIEMPO    DESCONOCEN    PROPAGAN     CALLAN
  0         99999         1           0
  1         99998         1           0
  2         99996         3           0
  3         99992         7           0
  4         99984         15          0
  5         99968         31          0
  6         99936         63          0
  7         99872         127         0
  8         99744         255         0
  9         99489         510         0
  10        98982         1016        1
  11        97978         2016        6
  12        96000         3972        27
  13        92186         7706        107
  14        85082         14504       412
  15        72741         25734       1524
  16        54021         40750       5227
  17        32007         52331       15660
  18        15257         47193       37549
  19        8056          25537       66405
  20        5899          7376        86724
  21        5556          1157        93285
  22        5492          135         94372
  23        5485          14          94499
  24        5484          1           94513
  25        5484          0           94515
```

Para 100.000 habitantes, haciendo F=0,000001 resulta que el porcentaje de la población que nunca habrá oído el rumor pasa del 5% al 12%.

29. CALCULO DE π (6.29)

Se trata de calcular un valor aproximado de π, lanzando dardos sobre la diana representada en la figura.



La diana es un cuadrado de lado 1, en el que se ha inscrito un cuadrante de círculo.

Supongamos que los dardos se reparten uniformemente; entonces la probabilidad de que un dardo caiga en el cuadrante de círculo es:

$$p = \frac{\text{Area del cuadrante}}{\text{Area del cuadrado}} = \frac{\pi/4}{1} = \frac{\pi}{4}$$

Supongamos, ahora, que lanzamos N dardos sobre el cuadrado, y sea M el número de los que caen en el cuadrante. La frecuencia relativa de caída en el cuadrante M/N, será aproximadamente igual a $\pi/4$. Por tanto:

$$\pi \simeq \frac{4M}{N}$$

Si el número N es suficientemente grande, cabe esperar que 4M/N sea una buena aproximación de π.

En el programa, las coordenadas de cada dardo se generan aleatoriamente en la línea 60, y la línea 70 cuenta el número de aciertos en el cuadrante.

```
10 REM CALCULO DE PI
20 ( BORRAR LA PANTALLA )
30 M=0
40 INPUT "CUANTOS DARDOS LANZAS";N
50 FOR I=1 TO N
60 X=RND : Y=RND
70 IF X↑2+Y↑2<1 THEN M=M+1
80 NEXT I
90 (BORRAR LA PANTALLA )
100 PRINT "SI LANZAS";N;"DARDOS OBTENEMOS"
110 PRINT
120 PRINT "UN VALOR APROXIMADO DE PI=";4*M/N
130 END
```

30. LA AGUJA DE BUFFON

Si dibujamos en un papel una serie de líneas paralelas distantes entre sí d *unidades y dejamos caer sobre dichas líneas una aguja de la misma longitud* d, *entonces la probabilidad teórica de que la aguja corte a una de las paralelas es* $2/\pi$. *Simula varias series de lanzamientos y calcula valores aproximados de* π.

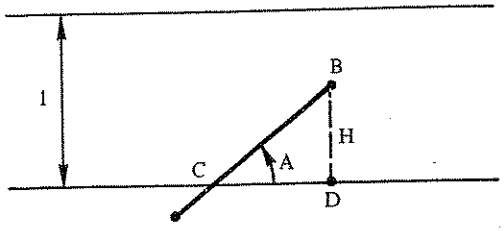


Fig. 1.

En la figura 1 hemos dibujado dos paralelas; la distancia entre ellas y la longitud de la aguja las hemos tomado, por comodidad, iguales a 1.

La posición de la aguja queda determinada por el ángulo A, comprendido entre 0 y 180 grados, y por la distancia BD = H, entre 0 y 1; si conocemos A y H, podemos calcular BC. En el triángulo rectángulo BDC, BC = H/SEN A. Si BC < 1, la aguja corta una paralela; en caso contrario, está contenida en la franja (fig. 2).

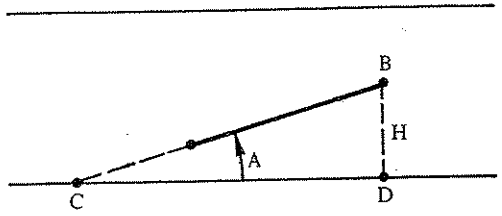


Fig. 2.

Si efectuamos N tiradas y calculamos los casos favorables, el cociente F/N es un valor aproximado de la probabilidad de que la aguja corte una línea.

Como $\frac{2}{\pi} \simeq \frac{F}{N}$, entonces $\pi \simeq \frac{2N}{F}$.

Es razonable pensar que cuanto mayor sea N, más precisos serán los resultados.

Simularemos el proceso diez veces, comenzando con N = 5 tiradas y duplicando en sucesivas simulaciones el número de tiradas de la anterior.

Los resultados los presentamos en forma de tabla:

TIRADAS	PROB.	V. APROX. DE π
5	0.6	3.3425
10	—	—
—	—	—

La tabla formada por las dos primeras columnas la guardamos en la variable T(I,J), (I = 1,2,...,10) y (J = 1,2); los valores de la tercera columna son los inversos de los de la segunda multiplicados por dos.

Para cada simulación generamos dos números aleatorios A y H, y calculamos BC, teniendo la precaución de pasar el ángulo a radianes. En la línea 70 se acumulan los casos favorables en la variable F y en la línea 90 se va cargando la variable T(I,J).

Como el lector comprobará, este método no da buenas aproximaciones del número π.

```
10 REM AGUJA DE BUFFON
20 DIM T(10,2) : N=5  : K=2 : F=0
30 FOR I=1 TO 10
40 FOR J=1 TO N
50 H=RND : A=180*RND
60 BC=H/SIN(3.14159265*A/180)
70 IF BC<1 THEN F=F+1
80 NEXT J
90 T(I,1)=N : T(I,2)=F/N
100 N=N*K : F=0
110 NEXT I
120 ( BORRAR LA PANTALLA )
130 PRINT TAB(4);"TIRADAS";TAB(14);"PROB.";TAB
(27);"V.APR.DE PI"
140 PRINT
150 FOR I=1 TO 10
160 PRINT TAB(5);T(I,1);TAB(13);T(I,2);TAB(26)
;2/T(I,2)
170 PRINT
180 NEXT I
190 END
```

31. CAZA DE BARCOS (6.28)

Cada vez que un submarino avista un barco enemigo sólo tiene tiempo de lanzar cuatro torpedos antes de ser localizado y perseguido. Cada torpedo tiene 1/3 de probabilidad de hundir el barco. Si un torpedo hunde un barco, en la pantalla aparecerá B HUNDIDO y no se lanzan más torpedos contra ese barco. Si no le da, aparecerá B NO y seguirá la serie de cuatro torpedos. Simula unos días de guerra en los que el submarino se encuentra diez barcos enemigos en total. Se sugiere que los resultados con cada barco aparezcan en una sola línea de pantalla.

El programa consta de los siguientes bloques:

1.° *Presentación e instrucciones.*

Subrutina 100.Líneas 100-150

2.° *Dibujo de la ventana del periscopio.*

Subrutina 200.Líneas 200-260

Para dibujar la ventana se consideran dos variables de cadena:

```
A1$="████████████████████████████"
A2$="█                          █"
```

Se imprimen en líneas sucesivas, primero A1$, después 15 veces A2$ y, finalmente, A1$.

3.° *Batalla.*

Subrutina 300.Líneas 300-680

En esta parte del programa necesitamos controlar con frecuencia la posición del cursor, lo que hacemos llevándolo al origen y desde allí colocándolo en la posición adecuada. También aparecen tres veces (líneas 320, 390 y 620) ciclos FOR-NEXT cuyo único objeto es retardar, décimas o segundos, la aparición de las imágenes que los siguen, para que la "película" no sea demasiado rápida.

Dentro del ciclo FOR-NEXT (líneas 300-540) se generan dos números aleatorios enteros X e Y, de modo que el barco dibujado en las líneas 350 y

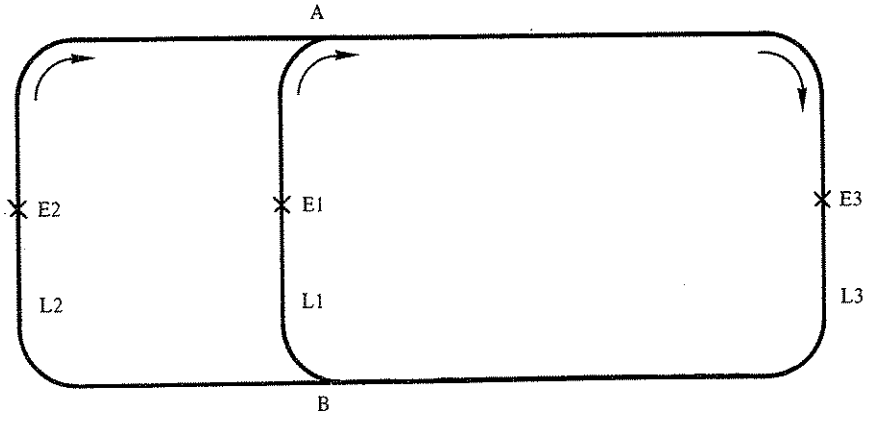
360 se sitúe dentro de la ventana. El ciclo 370-430 produce los disparos: si el número PR es cero, el barco ha sido tocado; un asterisco u otra señal simula el impacto y la subrutina 600 "borra" el barco del periscopio. A la derecha de la ventana se imprime B HUNDIDO. Si ninguno de los cuatro disparos es cero, la subrutina 600 borra el barco, se imprime B NO y un nuevo barco aparece aleatoriamente en el campo de visión del periscopio.

```
10 REM CAZA DE BARCOS
20 (BORRAR LA PANTALLA)
30 GOSUB 100 : REM INSTRUCCIONES
40 GOSUB 200 : REM DIBUJO VENTANA PERISCOPIO
50 GOSUB 300 : REM CAZA DE BARCOS
60 END
100 PRINT "EN LA VENTANA DEL PERISCOPIO APARECERA UN BARCO."
110 PRINT : PRINT "PARA CADA BARCO DISPONES DE CUATRO TORPEDOS."
120 PRINT : PRINT "EN TOTAL APARECERAN DIEZ BARCOS."
130 PRINT : PRINT "SI QUIERES COMENZAR PULSA UNA TECLA."
140 GET A$ : IF A$="" THEN 140
150 RETURN
200 (BORRAR LA PANTALLA)
210 A1$="████████████████"
220 A2$="█              █"
230 PRINT TAB(7);A1$
240 FOR I=1 TO 15 : PRINT TAB(7);A2$ : NEXT I
250 PRINT TAB(7);A1$
260 RETURN
300 FOR I=1 TO 10
310 (CURSOR AL ORIGEN)
320 FOR J=1 TO 500 : NEXT J
330 X=INT(12*RND+9) : Y=INT(11*RND+1)
340 FOR J=1 TO Y : PRINT : NEXT J
350 PRINT TAB(X);" __⌐L__ "
360 PRINT TAB(X);" \____/ "
370 FOR J=1 TO 4
380 (CURSOR AL ORIGEN)
390 FOR K=1 TO 1000 : NEXT K
400 PRINT "T";J
410 PR=INT(3*RND)
420 IF PR=0 THEN 480
430 NEXT J
440 GOSUB 600
450 FOR J=1 TO I : PRINT : NEXT J
460 PRINT TAB(28);"B";I;"NO"
470 GOTO 540
480 (CURSOR AL ORIGEN)
490 FOR J=1 TO Y+1 : PRINT : NEXT J
500 PRINT TAB(X+2);"*"
510 GOSUB 600
520 FOR J=1 TO I : PRINT : NEXT J
530 PRINT TAB(28);"B";I;"HUNDIDO"
540 NEXT I
550 RETURN
600 (CURSOR AL ORIGEN)
610 FOR J=1 TO Y : PRINT : NEXT J
```

```
620 FOR J=1 TO 1000 : NEXT J
630 PRINT TAB(X);"            "
640 PRINT TAB(X);"            "
650 (CURSOR AL ORIGEN)
660 PRINT "      "
670 (CURSOR AL ORIGEN)
680 RETURN
```


32. TRENES (6.56)

Dos trenes T1 y T2 tienen parte del recorrido común (L3) y parte distinto (L1 y L2), y circulan en un circuito cerrado en la misma dirección. Van con velocidad constante V1 y V2, y tienen una estación en la mitad de su trayecto común (E3) y otra en la mitad de su trayecto particular (E1 y E2). Salen a las 8 de la mañana y se pasan el día circulando, con paradas de un minuto en las estaciones. Como es natural no se pueden adelantar en el trayecto común, aunque sí pueden coincidir en E3. Vamos a hacer un programa que nos dé las horas a las que pasan los trenes por las agujas A y B. En cada vuelta el primero que pasa por A ha de ser el mismo que el primero que pase por B: si no, hay descarrilamiento. Cuando haya descarrilamiento sacaremos un mensaje en la pantalla señalando la hora del choque y terminaremos el programa.



Dando valores L1=60 V1=100
L2=80 V2=70
L3=100

aparece en la pantalla:

AGUJA A	AGUJA B
T1 8 18	
T2 8 34	
	T1 9 18
T1 9 56	
	T2 10 0
	T1 10 56
T2 11 10	
T1 11 34	

CHOQUE
A LAS 12 29

Interesa hallar el período de cada tren, el tiempo que tarda cada uno en dar una vuelta completa, les llamaremos P1 y P2.

T(1) es la hora a la que el tren 1 pasa por la aguja A
T(2) es la hora a la que el tren 2 pasa por la aguja A
T(3) es la hora a la que el tren 1 pasa por la aguja B
T(4) es la hora a la que el tren 2 pasa por la aguja B

El problema tiene dos aspectos: escribir correctamente los horarios y analizar cuándo se produce el choque. La escritura de los horarios tiene cierta dificultad. Hay que escribir cada vez el menor valor de T(I), pero puede corresponder al tren 1 o al 2, lo cual hay que recoger de alguna manera, y a la aguja A o a la B, lo que también ha de traducirse en la pantalla. Adjuntamos el organigrama de escritura de los horarios.

Hemos utilizado las siguientes fórmulas:

$$T=(V1*(T(1)-P1)-V2*(T(2)-P2))/(V1-V2)$$

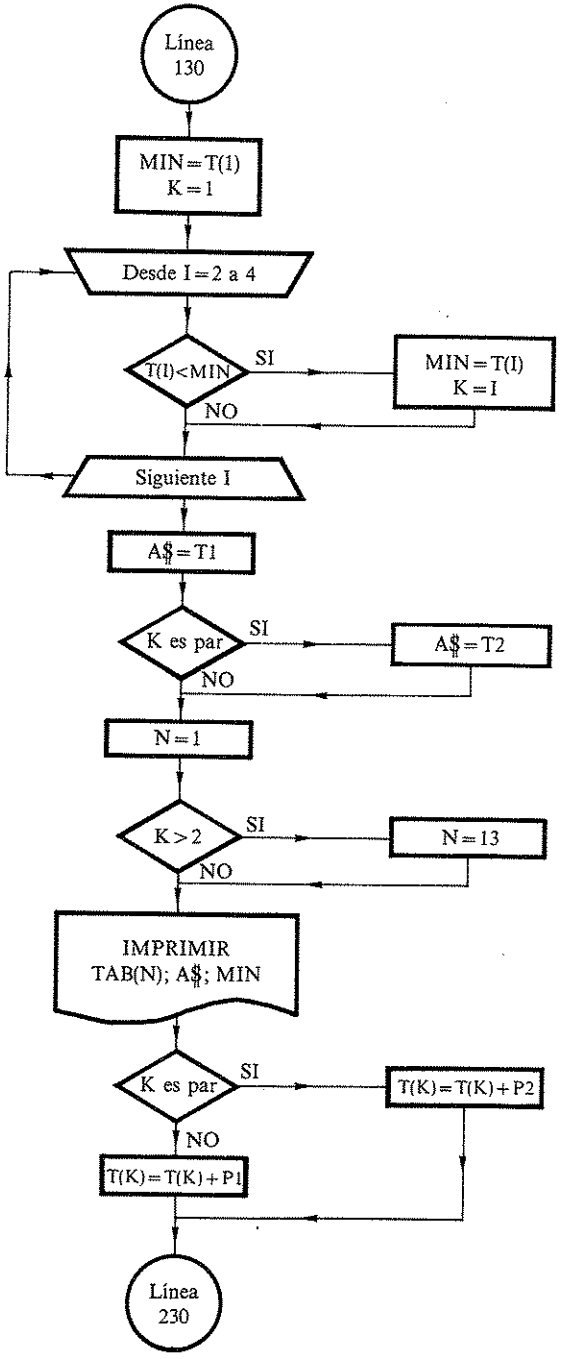
que da la hora del choque, y que se deduce de igualar las distancias recorridas desde el último paso por la aguja A:

$$(T-T(1)+P1)*V1=(T-T(2)+P2)*V2$$

La segunda es para truncar a minutos la parte decimal del tiempo:

$$INT((T-INT(T))*60)$$

Como detalle citaremos que si, en vez de truncar, se redondea, puede surgir un error, pues a las 12 horas 59,67 minutos se le asigna el valor 12 horas 60 minutos, que habría que corregir.



```
10 REM TRENES
20 ( BORRAR LA PANTALLA )
30 PRINT "DAR LAS 3 LONGITUDES EN KM."
40 INPUT L1,L2,L3
50 INPUT "VELOCIDADES EN KM/H";V1,V2
60 DIM T(4)
70 P1=(L1+L3)/V1+2/60 : P2=(L2+L3)/V2+2/60
80 T(1)=8+L1/V1/2
90 T(2)=8+L2/V2/2
100 T(3)=T(1)+L3/V1+1/60
110 T(4)=T(2)+L3/V2+1/60
120 PRINT : PRINT : PRINT TAB(1);"AGUJA A";TAB(13);
"AGUJA B"
130 REM ESCRITURA DE LOS HORARIOS
140 MIN=T(1) : K=1
150 FOR I=2 TO 4
160 IF T(I)<MIN THEN MIN=T(I) : K=I
170 NEXT I
180 A$="T1" : IF K/2=INT(K/2) THEN A$="T2"
190 N=1 : IF K>2 THEN N=13
200 PRINT TAB(N);A$;INT(MIN);INT((MIN-INT(MIN))*60)
210 IF K/2=INT(K/2) THEN T(K)=T(K)+P2 : GOTO 230
220 T(K)=T(K)+P1
230 REM COMPROBACION DEL CHOQUE
240 IF T(3)-P1=T(4)-P2 THEN T=T(3)-P1 : GOTO 320
250 IF T(1)<T(3) OR T(2)<T(4) THEN 130
260 IF T(1)-P1=T(2)-P2 THEN T=T(1)-P1 : GOTO 320
270 IF K=1 AND T(4)>T(3) THEN 300
280 IF K=2 AND T(4)<T(3) THEN 300
290 GOTO 130
300 T=(V1*(T(1)-P1)-V2*(T(2)-P2))/(V1-V2)
310 IF V1*(T-T(1)+P1)>L3/2 THEN T=T+1/60
320 PRINT : PRINT TAB(8);"CHOQUE"
330 PRINT : PRINT TAB(4);"A LAS";INT(T);INT((T-INT(
T))*60)
340 END
```

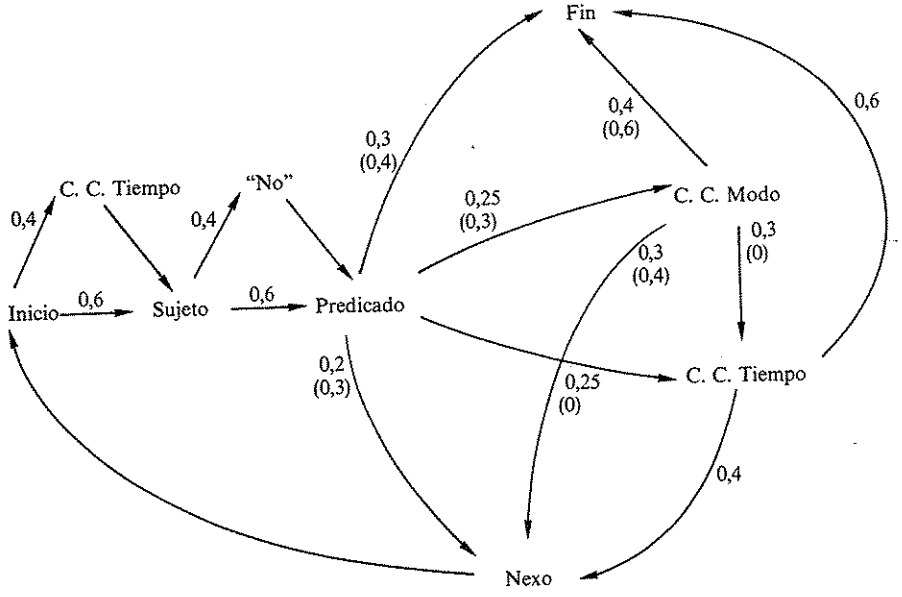

33. GENERADOR DE FRASES

Deseamos programar el ordenador de manera que sea capaz de generar una frase al azar, o dos frases encadenadas.

Aunque se recomienda emplear siempre los recursos de la programación estructurada, para este ejercicio recomendamos específicamente no hacerlo, y usar como guía el esquema que figura más abajo.

Para simplificar, consideramos frases en las que aparezcan sólo los elementos siguientes:

Sujeto.
Predicado, ampliación del concepto de verbo.
Complemento Circunstancial de Tiempo.
Complemento Circunstancial de Modo.
Nexo entre una frase y la siguiente, caso de haber dos.
La partícula "No".



El C. C. Tiempo podrá aparecer tras el Predicado sólo si no ha salido antes en la misma frase.

Llamemos nodos a los puntos de los que parten, o a los que llegan, flechas. En cada nodo del diagrama, el ordenador «elige su camino» al azar, generando números aleatorios. Tú puedes crear tu propio Generador añadiendo otros elementos de la frase, aquí no figuran todos, y alterando las probabilidades asociadas a los caminos de acuerdo con tus gustos.

Los caminos que salen del Predicado tienen asociados dos números, y lo mismo los que salen del C. C. de Modo; en ambos casos, el número sin paréntesis es la probabilidad del camino, si en esa frase no aparece el C. C. de Tiempo antes del Predicado; y el que lleva paréntesis es la probabilidad del camino, si el C. C. de Tiempo ya apareció.

Para los casos en que de un nodo salen más de dos caminos, Predicado y C. C. de Modo, hemos habilitado dos tablas, P y M, que tienen dos filas y tantas columnas como caminos salen del nodo en cuestión.

$$P = \begin{pmatrix} 0{,}3 & 0{,}55 & 0{,}8 & 1 \\ 0{,}4 & 0{,}7 & 0{,}7 & 1 \end{pmatrix} \qquad M = \begin{pmatrix} 0{,}4 & 0{,}7 & 1 \\ 0{,}6 & 0{,}6 & 1 \end{pmatrix}$$

En estas tablas aparece, en la primera fila, la probabilidad del primer camino, la suma de los dos primeros, de los tres primeros, etc.; y lo mismo para la segunda fila. Así, si un número al azar es superior al primer elemento de la fila, el programa no sigue el primer camino, pero sigue el segundo camino, si el número es inferior al segundo elemento de la fila, suma de las probabilidades de los dos primeros caminos.

Cuántos y cuáles son los Sujetos, Predicados, etc., es algo que podemos decirle al ordenador mediante instrucciones DATA y READ o INPUT; en el listado que figura a continuación hemos optado por la primera posibilidad, leyéndose estos elementos en listas T$, C. C. de Tiempo, S$, Sujetos, P$, Predicados, M$, C. C. de Modo y N$, Nexos.

Empleamos las variables de control F, SP, TP, PP y L. La variable F indica si estamos en la frase primera o en la segunda (en este caso, a partir del nodo Predicado, aunque el número aleatorio señale el camino que lleva al Nexo, se irá al Final); SP, TP y PP indican el Sujeto, Tiempo y Predicado empleados en la primera frase, para evitar su repetición en la segunda, caso de haber segunda. Por último, L vale 1 si no apareció Tiempo antes del Predicado, y 2 si apareció, con su ayuda tomamos probabilidades (elegimos fila) empleando las tablas P y M.


```
10 REM GENERADOR DE FRASES
20 DIM P(2,4),M(2,3)
30 FOR I=1 TO 2
40 FOR J=1 TO 4 : READ P(I,J) : NEXT J
50 NEXT I
60 FOR I=1 TO 2
70 FOR J=1 TO 3 : READ M(I,J) : NEXT J
80 NEXT I
90 READ T : DIM T$(T)
100 FOR I=1 TO T : READ T$(I) : NEXT I
110 READ S : DIM S$(S)
120 FOR I=1 TO S : READ S$(I) : NEXT I
130 READ P : DIM P$(P)
140 FOR I=1 TO P : READ P$(I) : NEXT I
150 READ M : DIM M$(M)
160 FOR I=1 TO M : READ M$(I) : NEXT I
170 READ N : DIM N$(N)
180 FOR I=1 TO N : READ N$(I) : NEXT I
190 F=0 : SP=0 : TP=0 : PP=0 : L=1
200 ( BORRAR LA PANTALLA )
210 F=F+1 : K=RND
220 IF K>.4 THEN 250
230 K=INT(T*RND+1) : IF K=TP THEN 230
240 PRINT T$(K) : TP=K : L=2
250 K=INT(S*RND+1) : IF K=SP THEN 250
260 PRINT S$(K) : SP=K
270 IF RND<.3 THEN PRINT "NO"
280 K=INT(P*RND+1) : IF K=PP THEN 280
290 PRINT P$(K) : PP=K
300 K=RND : IF K<P(L,1) THEN 400
310 IF K<P(L,2) THEN PRINT M$(INT(M*RND)+1) : GOTO 350
320 IF K<P(L,3) THEN PRINT T$(INT(T*RND)+1) : GOTO 380
330 IF F=2 THEN 400
340 PRINT N$(INT(N*RND)+1) : GOTO 210
350 K=RND : IF K<M(L,1) THEN 400
360 IF K<M(L,2) THEN 320
370 GOTO 330
380 IF RND<.6 THEN 400
390 GOTO 330
400 PRINT "SIGO HABLANDO (S/N)?"
410 GET H$ : IF H$<>"S" AND H$<>"N" THEN 410
420 IF H$="S" THEN 190
1000 DATA .3,.55,.8,1,.4,.7,.7,1
1010 DATA .4,.7,1,.6,.6,1
1020 DATA 3, LOS JUEVES, POR LA NOCHE, CUANDO LLUEVE
1030 DATA 3, CARLOS, MI TIA PACA, TU VECINO
```

```
1040 DATA 4, RONCA, SALTA, DA VOLTERETAS, COME PERAS
1050 DATA 5, ALOCADAMENTE, SIN PARAR, A LO TONTO, A TO
DA PASTILLA, EN BAÑADOR
1060 DATA 3, ADEMAS, AUNQUE, SIN EMBARGO
2000 END
```


3. TEXTOS

34. SUPRESION DE ESPACIOS (6.51)

Haz un programa que suprima todos los espacios en blanco de una cadena.

El código ASCII, ya usado en problemas anteriores, asocia a todo carácter un número natural. Al espacio en blanco le corresponde el código 32.

Una función ligada a este código es la función ASC, que aplicada a un carácter nos da su código. Por ejemplo, si B$ es el espacio vacío, entonces ASC(B$) = 32.

Con un ciclo FOR-NEXT, líneas 60-110, y la función MID$ seleccionamos cada carácter B$ de la cadena. En la línea 80 obtenemos su código; si este código es 32, tenemos un espacio en blanco y pasamos a estudiar el siguiente carácter. En caso contrario imprimimos el carácter, línea 100.

El punto y coma (;) con el que acabamos dicha línea nos asegura que todos los carácteres se imprimen unos a continuación de otros.

```
10 REM SUPRESION ESPACIOS EN BLANCO
20 ( BORRAR LA PANTALLA )
30 INPUT "ESCRIBE LA CADENA";C$
40 ( BORRAR LA PANTALLA )
50 L=LEN(C$)
60 FOR I=1 TO L
70 B$=MID$(C$,I,1)
80 B=ASC(B$)
90 IF B=32 THEN 110
100 PRINT B$;
110 NEXT I
120 END
```

35. VOCALES QUE HAY EN UNA FRASE

Haz un programa que, tras pedir una frase, cuente el número de vocales que aparecen en ella. Para lo cual puedes servirte de la variable V$ donde pones las vocales.

En el programa que vamos a considerar, tenemos tres variables. Dos de cadena: A$, donde introducimos la frase, y V$, que tiene las cinco vocales. La tercera, F(J), corresponde a una lista que nos lleva la cuenta de cada vocal que aparece en la frase.

Cada carácter de la frase se compara sucesivamente con cada una de las vocales de V$. Tan pronto haya una coincidencia, se cuenta ésta y se pasa a comparar el siguiente carácter de la frase.

Presentación en pantalla:

```
INTRODUCE LA FRASE
? EL PEZ GRANDE SE COME AL CHICO
VOCAL          FRECUENCIA
-----          ----------
  A                2
  E                5
  I                1
  O                2
  U                0
```

```
10 REM VOCALES QUE HAY EN UNA FRASE
20 DIM F(5)
30 FOR I=1 TO 5
40 F(I)=0
50 NEXT I
60 PRINT "INTRODUCE LA FRASE"
70 INPUT A$
80 V$="AEIOU"
90 FOR I=1 TO LEN(A$)
100 FOR J=1 TO 5
```



```
110 IF MID$(A$,I,1)=MID$(V$,J,1) THEN F(J)=F(J)+1 :
J=6
120 NEXT J
130 NEXT I
140 PRINT "VOCAL","FRECUENCIA"
150 PRINT "-----","----------"
160 FOR J=1 TO 5
170 PRINT TAB(2);MID$(V$,J,1),SPC(3);F(J)
180 NEXT J
190 END
```

36. FRECUENCIA DE UN CARACTER (6.49)

Necesitamos un programa que pida una frase. Luego ha de preguntar qué carácter deseas buscar; examinará la cadena y cada vez que encuentre el carácter señalará en qué posición está. Al final sacará el número total de veces que lo ha encontrado.

La frase se introduce en la variable A$. El carácter que queremos buscar en la variable B$.

Realizamos una comparación de cada carácter de la frase, mediante la función MID$, con el carácter deseado. Si es cierta la comparación, se aumenta el contador en uno y se escribe en qué posición está situado. Si no es cierta la comparación, se pasa al siguiente carácter de la frase y así sucesivamente hasta que se acaben los caracteres.

Presentación en pantalla:

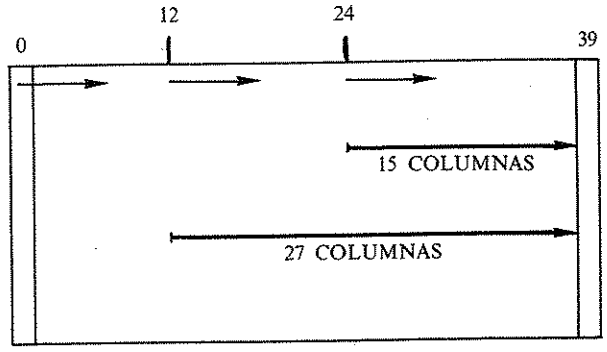
```
INTRODUCE LA FRASE
? LOS ANIMALES SON SERES QUE A VECES PARECEN INTELIGENTES

QUE CARACTER DESEAS BUSCAR
? E
ESTA EN LA POSICION 11
ESTA EN LA POSICION 19
ESTA EN LA POSICION 21
ESTA EN LA POSICION 26
ESTA EN LA POSICION 31
ESTA EN LA POSICION 33
ESTA EN LA POSICION 39
ESTA EN LA POSICION 41
ESTA EN LA POSICION 47
ESTA EN LA POSICION 51
ESTA EN LA POSICION 54

EL NUMERO DE VECES QUE APARECIO EL CARACTER E ES 11
```



```
10 REM FRECUENCIA DE UN CARACTER EN UNA FRASE
20 PRINT "INTRODUCE LA FRASE"
30 INPUT A$
40 A=0
50 PRINT
60 PRINT "QUE CARACTER DESEAS BUSCAR"
70 INPUT B$
80 FOR I=1 TO LEN(A$)
90 IF MID$(A$,I,1)<>B$ THEN 120
100 A=A+1
110 PRINT "ESTA EN LA POSICION";I
120 NEXT I
130 PRINT
140 PRINT "EL NUMERO DE VECES QUE APARECIO EL CA
RACTER ";B$;" ES";A
150 END
```

37. FLECHAZO AL ORDENADOR (6.27)

Flechazo al ordenador: la máquina te pregunta el nombre. A continuación lo imprime 15 veces, una en cada línea, situándolo aleatoriamente a la izquierda, derecha o centro de la pantalla, y añadiéndole signos de admiración (!), también en número aleatorio, comprendidos entre 1 y 5. (Es una combinación de RND y TAB, y si quieres que el nombre no se salga de una línea para escribirse en la siguiente, has de tener en cuenta su longitud.)



Suponemos que la pantalla del monitor tiene 40 columnas numeradas del 0 al 39. La posición de la izquierda comienza en la columna 0, la central en la 12 y la posición derecha en la columna 24. En el caso de que tu pantalla tenga dimensiones diferentes, se han de modificar las líneas 60, 90, 100, 120 y 180.

Generamos dos números aleatorios X e Y. El primero toma los valores 1, 2 ó 3 y asigna la posición izquierda, central o derecha respectivamente a la secuencia de signos de admiración y nombre. El segundo indica el número de dichos signos que preceden y siguen al nombre.

Si L es la longitud del nombre N$, la longitud de toda la cadena es 2*Y+L.

La posición límite del cursor en el último signo de admiración debe ser la 38, para que pase a ocupar a continuación la 39; entonces la instrucción PRINT de la línea 160 sitúa el cursor en la línea siguiente. Por el contrario,

si imprimimos en la posición 39, el cursor pasa automáticamente a la línea siguiente y, debido a la instrucción PRINT, la próxima cadena se imprime dos filas más abajo.

Cuando el nombre tiene más de 29 caracteres, la impresión completa puede invadir la línea siguiente. Para evitar esto la línea 60 rechaza los nombres con más de esos caracteres.

Si X=2 y 2*Y+L es mayor que 27, el cursor pasa a la línea siguiente; para evitarlo debemos desplazar la cadena a la izquierda tantos lugares como exceda a 27.

En el caso de que X=3 y 2*Y+L sea mayor que 15, por la misma razón, desplazamos la cadena a la izquierda (2*Y+L)−15 lugares.

```
10 REM FLECHAZO AL ORDENADOR
20 ( BORRAR LA PANTALLA )
30 INPUT "NOMBRE";N$
40 ( BORRAR LA PANTALLA )
50 L=LEN(N$)
60 IF L>29 THEN PRINT "NOMBRE DEMASIADO LARGO, CAM
BIA DE ENAMORADO/A" : GOTO 20
70 FOR I=1 TO 15
80 X=INT(3*RND)+1 : Y=INT(5*RND)+1
90 IF X=2 AND 2*Y+L>27 THEN A=27 : GOTO 170
100 IF X=3 AND 2*Y+L>15 THEN A=15 : GOTO 170
110 FOR J=1 TO Y
120 PRINT TAB((X-1)*12+(J-1));"!";
130 NEXT J
140 PRINT N$;
150 FOR J=1 TO Y : PRINT "!"; : NEXT J
160 PRINT : GOTO 210
170 FOR J=1 TO Y
180 PRINT TAB((X-1)*12+(J-1)-(2*Y+L-A));"!";
190 NEXT J
200 GOTO 140
210 NEXT I
220 END
```

38. FRECUENCIA DE UNA PALABRA

Queremos contar cuántas veces aparece una palabra en una frase. Si, por ejemplo, buscamos la palabra "sol", es obvio que las palabras "sola", o "solar", no nos sirven: la palabra "sol" debe tener a izquierda y derecha un espacio en blanco, o un signo de puntuación.

Planteamos el programa utilizando el código ASCII. En este código cada carácter (números, letras, signos de puntuación, etc.) tiene asociado un número. Concretamente, a las letras, les corresponden números enteros consecutivos desde 65 a 90.

CARACTER:	A B C D	Y Z
CODIGO:	65 66 67 68	89 90

Una función asociada a este código es la función ASC que aplicada a un carácter nos da su código; ASC(D)=68.

Si mediante la función MID$ seleccionamos un carácter de una cadena, y a ese carácter aplicamos la función ASC, obtendremos un número comprendido entre 65 y 90, ambos inclusive, si es una letra. Esta técnica nos permitirá aislar las palabras.

En la variable F$ guardamos la frase, en P$ la palabra cuya frecuencia se busca y en N el número de veces que aparece dicha palabra.

Queremos contar, por ejemplo, cuántas veces aparece la palabra "dos" en la frase:

"Dos y dos son cuatro"

Esta cadena tiene una longitud de 21 caracteres incluyendo los espacios vacíos y el punto final.

Establecemos una cadena de caracteres A$, que inicialmente está vacía. En ella vamos depositando las letras de la frase hasta llegar a un carácter que no sea letra. En ese momento en A$ tenemos una palabra, formada exclusivamente por letras, que comparamos con la palabra pedida P$. Si es

la misma la contabilizamos en N y vaciamos A$, y si no lo es, solamente vaciamos A$. En ambos casos continuamos buscando la siguiente palabra, hasta finalizar la frase.

El análisis de la última palabra exige algunos comentarios. Si, en nuestro ejemplo, acabamos la cadena en la última letra, la O, al llegar a ella, como su código cumple las condiciones impuestas, pasa a la línea 150 y de aquí, por haberse acabado la cadena, a 160, 170 ó 180. No ocurre nada si la última palabra no es la buscada. Si lo es, no es contabilizada. Para subsanar este posible error hemos de acabar la cadena en un signo de puntuación.

```
10 REM FRECUENCIA DE UNA PALABRA
20 ( BORRAR LA PANTALLA )
30 INPUT "ESCRIBE LA FRASE";F$
40 PRINT
50 INPUT "PALABRA SELECCIONADA";P$
60 ( BORRAR LA PANTALLA )
70 L=LEN(F$)
80 A$="" : N=0
90 FOR I=1 TO L
100 C$=MID$(F$,I,1)
110 C=ASC(C$)
120 IF C>=65 AND C<=90 THEN A$=A$+C$ : GOTO 150
130 IF A$=P$ THEN N=N+1
140 A$=""
150 NEXT I
160 IF N=0 THEN PRINT P$;" NO APARECE EN LA FRA
SE" : GOTO 190
170 IF N=1 THEN PRINT P$;" APARECE UNA VEZ" : G
OTO 190
180 PRINT P$;" APARECE";N;"VECES"
190 END
```

39. CONJUGACION

Haz un programa que te pida el infinitivo de un verbo de la primera conjugación, y a continuación escriba su presente y su pretérito imperfecto de subjuntivo.

Yo ame, tú ames, él ame, ...; yo amara, tú amaras, él amara, ... ¡Repasar la gramática nunca viene mal!

Para diseñar el programa hemos considerado la tabla C$(I,J); I=0, 1, 2, ..., 6; J=1, 2, 3; de 7 filas y 3 columnas, siguiente:

PRONOMBRES	PRESENTE DE SUB.	PRET. IMPERF. DE SUBJ.
YO	E	ARA
TU	ES	ARAS
EL	E	ARA
NOSOTROS	EMOS	ARAMOS
VOSOTROS	EIS	ARAIS
ELLOS	EN	ARAN

En la primera columna están los pronombres personales, en la segunda las terminaciones del presente de subjuntivo y en la tercera las del pretérito imperfecto de subjuntivo (de las dos formas posibles de este tiempo hemos puesto sólo la primera).

Las líneas 50-90 cargan la tabla por columnas, utilizando las instrucciones DATA del final del programa. En B$(0,2) y B$(0,3) se guardan las cadenas de caracteres "PRESENTE DE SUB." y "PRET. IMPERF. DE SUBJ.", que más tarde se imprimirán encabezando el tiempo correspondiente.

Con las líneas 110-130 el ordenador pregunta el infinitivo del verbo, V$, y halla su radical, R$, utilizando la función LEFT$.

En las líneas 150-230 se construyen e imprimen los tiempos del verbo, añadiendo al radical la terminación y anteponiendo el pronombre personal correspondiente.


```
10 REM CONJUGACION DE UN VERBO
20 ( BORRAR LA PANTALLA )
30 DIM C$(6,3)
40 REM SE CARGA LA TABLA
50 FOR J=1 TO 3
60 FOR I=0 TO 6
70 READ C$(I,J)
80 NEXT I
90 NEXT J
100 REM SE OBTIENE EL RADICAL DEL VERBO
110 INPUT "INFINITIVO DEL VERBO";V$
120 L=LEN(V$)
130 R$=LEFT$(V$,L-2)
140 REM SE IMPRIME LA CONJUGACION
150 PRINT : PRINT
160 FOR J=2 TO 3
170 PRINT C$(0,J)
180 PRINT
190 FOR I=1 TO 6
200 PRINT C$(I,1)+" "+R$+C$(I,J)
210 NEXT I
220 PRINT : PRINT
230 NEXT J
240 DATA PRONOMBRES,YO,TU,EL,NOSOTROS,VOSOTROS,
ELLOS
250 DATA PRESENTE DE SUB.,E,ES,E,EMOS,EIS
,EN
260 DATA PRET.IMPERF.DE SUBJ.,ARA,ARAS,ARA,ARAM
OS,ARAIS,ARAN
270 END
```

40. MENSAJE SECRETO

Una de las maneras más inocentes de cifrar un mensaje consiste en sustituir unas letras por otras. Haz un programa que sirva para cifrar y descifrar mensajes.

El lenguaje BASIC tiene dos funciones relacionadas con el código ASCII: la función ASC, que aplicada a un carácter nos da su código numérico; y la función CHR$, que aplicada a un número nos da el carácter correspondiente. Así, por ejemplo, ASC(A) es 65 y CHR$(66) es B.

Una manera de hacer las sustituciones en un mensaje consiste en cambiar cada letra por el carácter cuyo código ASCII excede en tres unidades al de aquélla. La letra A, por ejemplo, será sustituida por la D, según el esquema:

$$A \xrightarrow{ASC} 65 \xrightarrow{+3} 68 \xrightarrow{CHR\$} D$$

Aquí la clave es sumar tres.

Para descifrar el mensaje hay que proceder al revés:

$$D \xrightarrow{ASC} 68 \xrightarrow{-3} 65 \xrightarrow{CHR\$} A$$

Teniendo en cuenta que los procesos de cifrado y de descifrado son prácticamente iguales, salvo que en un caso se suma 3 y en el otro se resta, un mismo programa nos puede servir para cifrar y descifrar. En efecto, la instrucción:

C=C+S*3

que cambia el código de las letras, sirve para cifrar cuando S vale 1, y para descifrar cuando S vale −1.

Al correr el programa aparece en la pantalla:

CLAVE ?

Debemos contestar con un número no muy alto, 3 por ejemplo, que el ordenador guarda en la variable X.

A continuación se nos pregunta:

CIFRAR (1) O DESCIFRAR (−1) ?

Si pulsamos la tecla 1, aparece en la pantalla algo parecido a esto:

MENSAJE
? HOY A LAS CINCO
MENSAJE CIFRADO:
KR\ #D#ODV#FLQFR

Si pulsamos −1, obtenemos algo así:

MENSAJE CIFRADO
? KR\ #D#ODV#FLQFR
MENSAJE DESCIFRADO:
HOY A LAS CINCO

En la línea 30 asignamos a las variables P$(1), P$(2) y P$(3) las cadenas de caracteres "DESCIFRADO", "CIFRADO" y "" (cadena vacía), que sirven para adjetivar convenientemente a la palabra MENSAJE, según los valores de S (líneas 100 y 120).

```
10 REM MENSAJE SECRETO
20 DIM P$(3)
30 P$(1)="DESCIFRADO" : P$(2)="CIFRADO" : P$(3)=""
40 ( BORRAR LA PANTALLA )
50 INPUT "CLAVE";X
60 ( BORRAR LA PANTALLA )
70 PRINT
80 INPUT "CIFRAR (1) O DESCIFRAR (-1)";S
90 PRINT
100 PRINT "MENSAJE ";P$((S+5)/2)
110 INPUT M$
120 MC$=""
130 N=LEN(M$)
140 FOR I=1 TO N
150 L$=MID$(M$,I,1)
160 C=ASC(L$)
170 C=C+S*X
```

```
180 L$=CHR$(C)
190 MC$=MC$+L$
200 NEXT I
210 PRINT
220 PRINT "MENSAJE ";P$((S+3)/2);":"
230 PRINT MC$
240 GOTO 70
250 END
```

(Modifica el programa introduciendo claves más difíciles de descubrir.)

41. MORSE

Hemos de preparar un programa para la transmisión automática de mensajes en código Morse. Los caracteres admitidos son los que aparecen, junto con su código, en la tabla. Además, deseamos despreocuparnos de la transmisión de los códigos de final de mensaje y principio de mensaje.

Carácter	Morse	ASC	Carácter	Morse	ASC	Carácter	Morse	ASC
0	-----	48	D	-..	68	Q	--.-	81
1	.----	49	E	.	69	R	.-.	82
2	..---	50	F	..-.	70	S	...	83
3	...--	51	G	--.	71	T	-	84
4	....-	52	H	....	72	U	..-	85
5	.....	53	I	..	73	V	...-	86
6	-....	54	J	.---	74	W	.--	87
7	--...	55	K	-.-	75	X	-..-	88
8	---..	56	L	.-..	76	Y	-.--	89
9	----.	57	M	--	77	Z	--..	90
A	.-	65	N	-.	78	Punto	.-.-.-	46
B	-...	66	O	---	79	?	..--..	63
C	-.-.	67	P	.--.	80	Comienzo	-.-.-	
						Fin	...-.-	

Almacenamos los códigos ordenados en líneas DATA, de manera que el ordenador los pueda leer y formar con ellos la lista siguiente:

(-----, .----, ..., ----., .-, ..., --.., .-.-.-, ..--.., -.-.-, ...-.-)
 Ø 1 ... 9 A ... Z . ? Princ. Fin

Como no admitiremos más símbolos que los que aparecen en las columnas de caracteres de la lista, lo primero que haremos, tras formar la lista, será comprobar si todos los caracteres del mensaje son admisibles o no. En caso afirmativo, pasamos a emitir el mensaje: tras el "principio de mensaje" hemos de estudiar, para todos los caracteres a emitir, si son números (ocupan en la lista el lugar correspondiente a su código ASCII menos 47), letras (ocupan el lugar correspondiente a su código ASCII menos 55), punto (lugar 37) o interrogante (lugar 38).

El programa puede quedar así:

```
10 REM MORSE
20 DIM C$(40)
30 FOR I=1 TO 40 : READ C$(I) : NEXT I
40 INPUT "CONTENIDO DEL MENSAJE";M$
50 FOR I=1 TO LEN(M$)
60 CO=ASC(MID$(M$,I,1))
70 IF (48<=COANDCO<=57) OR (65<=COANDCO<=90) OR CO
=32 OR CO=46 OR CO=63 THEN 90
80 PRINT "EL SIMBOLO ";MID$(M$,I,1);" NO SE PUEDE
USAR" : GOTO 40
90 NEXT I
100 ( BORRAR LA PANTALLA )
110 PRINT : PRINT : PRINT C$(39);" ";
120 FOR I=1 TO LEN(M$)
130 CO=ASC(MID$(M$,I,1))
140 IF CO=32 THEN PRINT " "; : GOTO 190
150 IF CO=46 THEN PRINT C$(37);" "; : GOTO 190
160 IF CO=63 THEN PRINT C$(38);" "; : GOTO 190
170 IF 48<=CO AND CO<=57 THEN PRINT C$(CO-47); : G
OTO 190
180 IF 65<=CO AND CO<=90 THEN PRINT C$(CO-54);
190 NEXT I
200 PRINT " ";C$(40) : GOTO 300
210 DATA -----,.----,..---,...--,....-,.....,-....
,--...
220 DATA ---..,----.,.-,-...,-.-.,-..,.,...-.,--.,.
...,..
230 DATA .---,-.-,.-..,--,-.,---,.--.,--.-,.-.,...
,-,..-
240 DATA ...-,.--,-..-,-.--,--..,.-.-.-,..--..,-.-
.-,...-.-
300 END
```


42. POESIA ALEATORIA

Deseamos un programa que reordene al azar los versos de una poesía cualquiera.

Este programa deberá constar de las siguientes partes:

— introducción de versos;
— rectificación de éstos, si es necesaria;
— exhibición de la poesía reordenada; en tanto se desee, deberá repetirse esta parte.

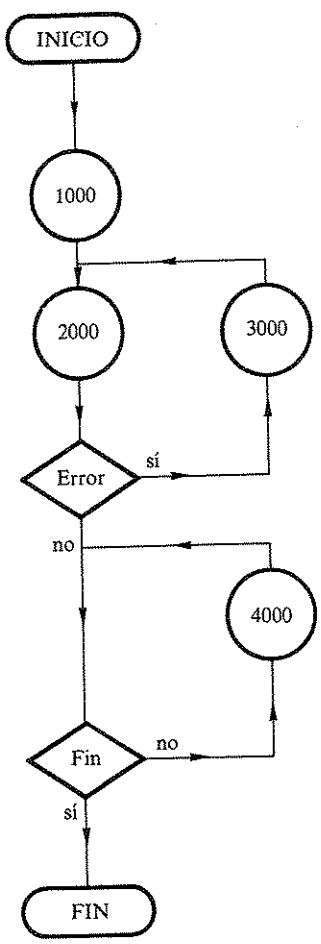
En lenguaje algorítmico se pueden expresar estas condiciones de la siguiente manera:

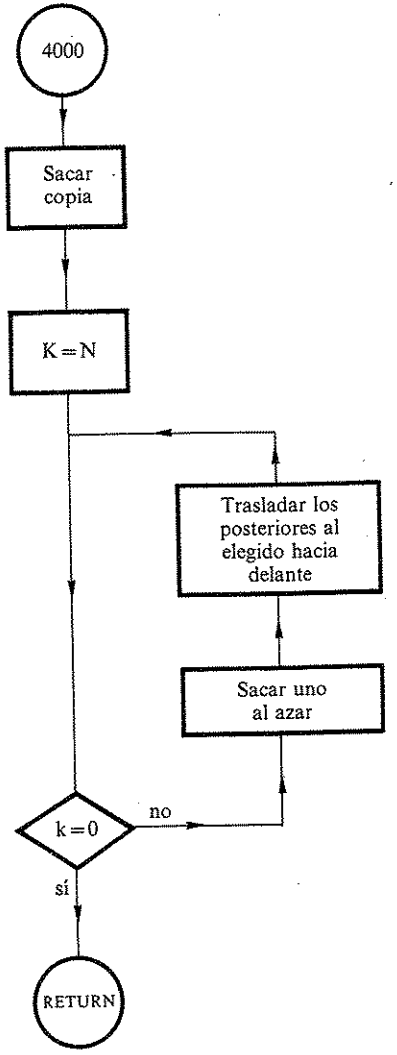
Entrar versos
Repetir la exhibición de los versos
hasta que estén todos bien
haciendo mientras las rectificaciones oportunas
Fin de Repetir
Mientras se desee,
hacer exhibición aleatoria de la poesía
Fin de Mientras
FIN

Para que la estructura del programa sea más sencilla, podemos recurrir al empleo de subrutinas, asignándoles las siguientes direcciones:

1000 a la rutina de entrada de versos,
2000 a la rutina de exhibición de la poesía en su orden,
3000 a la rutina de corrección de errores y
4000 a la rutina de exhibición aleatoria.

Los diagramas de flujo del programa y de la subrutina 4000, única no trivial, serán:





En la subrutina 4000 comenzamos sacando una copia de la poesía introducida en la memoria y desde ese momento, y mientras queden versos en la copia, escribimos un verso cualquiera de la copia y adelantamos todos los siguientes. Si la copia tiene seis versos, por ejemplo, escogemos uno entre el primero y el sexto; supongamos que es el cuarto; lo escribimos y adelantamos todos los versos posteriores a éste, quinto y sexto, a las posiciones cuarta y quinta respectivamente; luego sacamos otro entre el primero y el quinto, y así sucesivamente.


```
10 REM POESIA ALEATORIA
100 REM PROGRAMA PRINCIPAL
110 GOSUB 1000
120 GOSUB 2000
130 INPUT "?HAY ERRORES";C$
140 IF C$<>"SI" AND C$<>"NO" THEN GOTO 130
150 IF C$="NO" THEN GOTO 180
160 GOSUB 3000
170 GOTO 120
180 INPUT "?FINALIZO";C$
190 IF C$<>"SI" AND C$<>"NO" THEN GOTO 180
200 IF C$="SI" THEN GOTO 230
210 GOSUB 4000
220 GOTO 180
230 END
1000 REM RUTINA DE ENTRADA
1010 INPUT "NUMERO DE VERSOS";N
1020 DIM V$(N),R$(N)
1100 FOR I=1 TO N
1110 PRINT "VERSO ";I; : INPUT V$(I)
1120 NEXT I
1140 RETURN
2000 REM EXHIBICION EN SU ORDEN
2010 ( BORRAR LA PANTALLA )
2100 FOR I=1 TO N : PRINT I;TAB(4);V$(I) : NEXT I
2110 RETURN
3000 REM CORRECCION
3100 INPUT "NUMERO DEL VERSO A RECTIFICAR (100 PA
RA NINGUNO)";I
3110 IF I=100 THEN GOTO 3140
3120 PRINT "NUEVA VERSION";
3130 INPUT V$(I) : GOTO 3100
3140 RETURN
4000 REM EXHIBICION REORDENADA
4010 ( BORRAR LA PANTALLA )
4020 FOR I=1 TO N : R$(I)=V$(I) : NEXT I
4030 K=N
4100 IF K=0 THEN GOTO 4150
4110 I=1+INT(K*RND) : PRINT R$(I)
4120 IF I=K THEN 4140
4130 FOR J=I+1 TO K: R$(J-1)=R$(J) : NEXT J
4140 K=K-1 : GOTO 4100
4150 RETURN
```

43. GIRO DE LETRAS (6.41)

Escribir en la pantalla, como si fueran los vértices de un cuadrado, las siguientes letras:

A B

D C

Hacer un programa que gire las letras en el sentido de las agujas del reloj, si pulsamos R; y en sentido contrario, si pulsamos C.

Colocamos las letras A, B, C y D en las variables A$, B$, C$ y D$ respectivamente. Para imprimirlas en la pantalla, de forma que estén centradas, nos valemos de la función TAB.

Antes del giro colocamos las letras en las variables auxiliares A1$, B1$, C1$ y D1$, que es lo que nos va a permitir realizar a continuación el giro en un sentido u otro, según la letra que pulsemos, pues esos valores irán a las variables primitivas, pero ya giradas.

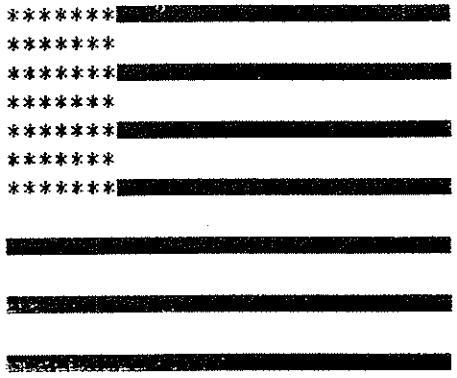
El programa no tiene fin, por lo que es necesario para terminar pulsar la tecla correspondiente a interrupción de la ejecución del programa.

```
10 REM "GIRO" DE LETRAS
20 A$="A" : B$="B" : C$="C" : D$="D"
30 REM PRESENTACION DE LAS CUATRO LETRAS
40 PRINT TAB(18);A$;TAB(22);B$
50 PRINT : PRINT
60 PRINT TAB(18);D$;TAB(22);C$ : PRINT
70 PRINT "PARA ";CHR$(34);"GIRAR";CHR$(34);" EN EL SENTIDO DE LAS AGUJAS"
80 PRINT : PRINT "DEL RELOJ PULSE R Y EN SENTIDO CONTRARIO"
90 PRINT : PRINT "PULSE C" : PRINT
100 INPUT G$
110 IF G$<>"R" AND G$<>"C" THEN 70
120 A1$=A$ : B1$=B$ : C1$=C$ : D1$=D$
130 IF G$="R" THEN 160
140 D$=A1$ : C$=D1$ : B$=C1$ : A$=B1$
150 (BORRAR LA PANTALLA) : GOTO 40
160 B$=A1$ : C$=B1$ : D$=C1$ : A$=D1$
170 (BORRAR LA PANTALLA) : GOTO 40
180 END
```


4. GRAFICOS

44. BANDERA U.S.A.

Haz un programa para dibujar en la pantalla la bandera de los EE.UU.



Hay muchas maneras de programar el dibujo de esta bandera. Nosotros hemos elegido una que conlleva el empleo de subrutinas; no porque creamos que es el mejor modo de hacerlo, sino por mostrar un ejemplo de empleo de subrutinas a varios niveles.

El diseño del programa está basado en la observación de que en cada fila hay siete estrellas y que las barras están formadas por catorce o por veintiún cuadraditos.

La subrutina llamada ESTRELLAS imprime 7 estrellas en línea. La subrutina BARRA imprime una barra horizontal de 7 cuadraditos.

La subrutina ESTRELLAS Y BARRAS utiliza las dos anteriores para imprimir una línea de 7 estrellas y 14 cuadraditos. La subrutina BARRAS Y BARRAS imprime una barra de 21 cuadraditos, utilizando para ello la subrutina BARRA.

El programa principal (líneas 100-220) combina las subrutinas para imprimir, línea a línea, la bandera.

Si tu ordenador no dispone del signo ■, puedes sustituirlo por el signo =.

(Esperamos que la pérdida de una estrella no sea motivo de ninguna nota de protesta diplomática.)

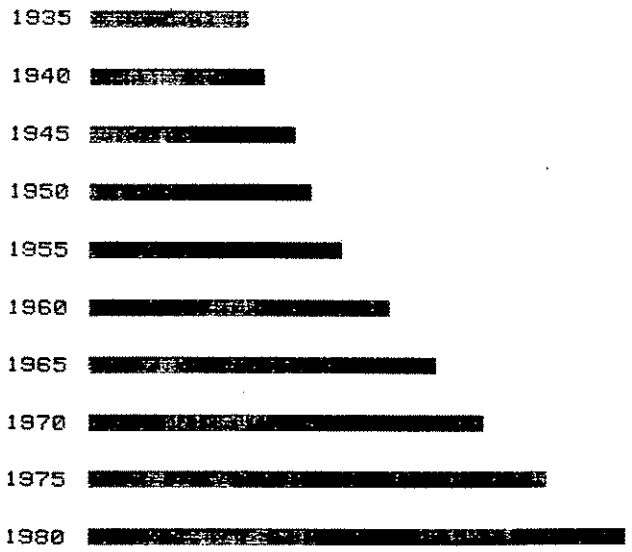
```
100 REM BANDERA U.S.A.
110 ( BORRAR LA PANTALLA )
120 FOR FILA=1 TO 3
130 GOSUB 700 : REM ESTRELLAS Y BARRAS
140 GOSUB 500 : REM ESTRELLAS
150 PRINT
160 NEXT FILA
170 GOSUB 700 : REM ESTRELLAS Y BARRAS
180 FOR FILA=5 TO 7
190 PRINT
200 GOSUB 800 : REM BARRAS Y BARRAS
210 NEXT FILA
220 END
500 REM SUBRUTINA ESTRELLAS
510 FOR E=1 TO 7
520 PRINT "*";
530 NEXT E
540 RETURN
600 REM SUBRUTINA BARRA
610 FOR C=1 TO 7
620 PRINT "■";
630 NEXT C
640 RETURN
700 REM SUBRUTINA ESTRELLAS Y BARRAS
710 GOSUB 500 : REM ESTRELLAS
720 GOSUB 600 : REM BARRA
730 GOSUB 600 : REM BARRA
740 PRINT
750 RETURN
800 REM SUBRUTINA BARRAS Y BARRAS
810 GOSUB 600 : REM BARRA
820 GOSUB 600 : REM BARRA
830 GOSUB 600 : REM BARRA
840 PRINT
850 RETURN
```


45. EVOLUCION DE LA POBLACION (6.15)

La población de una ciudad ha evolucionado de la siguiente forma:

1935	*1238400*	*1960 .....*	*2362057*
1940	*1401107*	*1965 .....*	*2716366*
1945	*1580880*	*1970 .....*	*3123821*
1950	*1786055*	*1975 .....*	*3592394*
1955	*2053963*	*1980 .....*	*4237525*

Queremos representar esta evolución mediante un diagrama de barras horizontales. Para ello deberás utilizar el signo ■. *Tras escribir el año y espacio en blanco, pondrás las barras de longitud proporcional a la población. Has de cuidar el problema de escalas: dispones de 34 casillas que has de aprovechar al máximo. Por ejemplo, a 4237525 que es la cifra más alta, le puedes hacer corresponder las 34 casillas, o un número de espacios próximo a 34.*



El año y la población están recogidos en las variables con índice A(I) y P(I) respectivamente, donde $1 \leqslant I \leqslant 10$.

Para elegir la escala, asociamos al valor máximo que designamos por M, las 34 casillas; luego a un número N de habitantes le hacemos corresponder:

$$N*34/M$$

Como generalmente saldrá un número decimal es necesario redondear las unidades mediante la sentencia:

$$S=INT(N*34/M+0.5)$$

El número de casillas asociado a la población P(I) se guardará en la variable C(I). En el ciclo 130-160 leemos los años y las poblaciones correspondientes, y a la vez determinamos el valor máximo de la población, mediante la instrucción de la línea 150, valor que guardamos en la variable M.

En la línea 190 se imprime el año A(I) y en la 200 se hace el cambio de escala.

A continuación imprimimos la barra correspondiente a la población I con las instrucciones de las líneas 210-230.

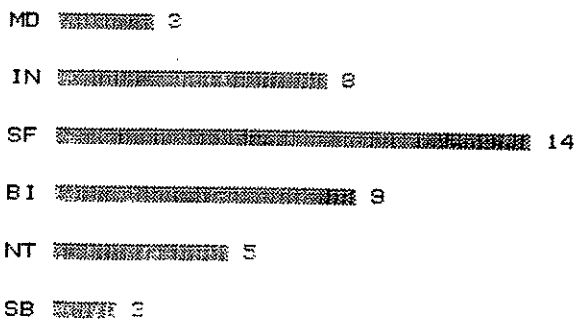
```
10 REM EVOLUCION DE LA POBLACION
20 ( BORRAR LA PANTALLA )
30 PRINT "REPRESENTACION DE LA EVOLUCION"
40 PRINT "DE LA POBLACION DE UNA CIUDAD"
50 PRINT "MEDIANTE DIAGRAMA DE BARRAS"
60 PRINT : PRINT
70 PRINT "PARA VER EL DIAGRAMA PULSE"
80 PRINT "UNA TECLA"
90 GET A$
100 IF A$="" THEN 90
110 ( BORRAR LA PANTALLA )
120 M=0
130 FOR I=1 TO 10
140 READ A(I),P(I)
150 IF P(I)>M THEN M=P(I)
160 NEXT I
170 REM REPRESENTACION DIAGRAMA
180 FOR I=1 TO 10
190 PRINT : PRINT : PRINT A(I);
200 C(I)=INT(P(I)*34/M+0.5)
210 FOR J=1 TO C(I)
220 PRINT "■";
230 NEXT J
```



```
240 NEXT I
250 DATA 1935,1238400,1940,1401107
260 DATA 1945,1580880,1950,1786055
270 DATA 1955,2053963,1960,2362057
280 DATA 1965,2716366,1970,3123821
290 DATA 1975,3592394,1980,4237525
300 END
```

46. DIAGRAMA DE BARRAS HORIZONTALES

Programa un diagrama de barras horizontales para representar distintas frecuencias.



El número de barras del diagrama se lo daremos al ordenador mediante INPUT. Una vez establecido este número B, doce como mucho, introduciremos por parejas el nombre N$(I) y la frecuencia F(I) correspondientes a cada barra.

De los cuarenta caracteres que tiene cada línea de pantalla, reservamos los dos primeros para el nombre de la barra, el tercero para un espacio, y a continuación dibujamos la barra, con 30 caracteres como máximo; por último, escribimos las cifras de la frecuencia separadas de la barra por un espacio.

La frecuencia mayor va a ser representada en el dibujo por 30 caracteres; por tanto, tendremos que reducir (o ampliar) todas las frecuencias de modo que la mayor de ellas sea 30. Esto se puede hacer con la instrucción:

```
FR(I)=F(I)/MAX*30
```

En donde MAX representa la frecuencia máxima, que previamente hay que obtener.

Como, previsiblemente, los valores FR(I) no serán enteros, los redondearemos al entero más próximo. Así:

```
FR(I)=INT(F(I)/MAX*30+0.5)
```

Esta será la instrucción que realmente figure en el programa.

Del nombre de cada barra imprimiremos los dos primeros caracteres más un espacio. Podemos hacerlo así:

```
PRINT LEFT$(N$(I),2);"  ";
```

Para prever la posibilidad de que algún nombre tuviera inicialmente un solo carácter, en cuyo caso la barra correspondiente quedaría corrida un lugar a la izquierda, añadimos un espacio a todos los nombres N$(I). La instrucción que va a figurar en el programa es:

```
PRINT LEFT$(N$(I)+"  ",2);"  ";
```

La impresión de cada barra la hacemos mediante el ciclo:

```
FOR J=1 TO FR(I)
PRINT "■";              (1)
NEXT J
```

Hay que tener previsto un caso excepcional: cuando la frecuencia FR(I) es cero. En efecto: las instrucciones interiores de un ciclo FOR-NEXT se ejecutan por lo menos una vez. En consecuencia, el ciclo:

```
FOR J=1 TO 0
PRINT "■";
NEXT J
```

imprimirá un cuadradito, en lugar de dejar la línea en blanco.

Para obviar este inconveniente, una instrucción del tipo

```
IF FR(I)=0 THEN ...
```

debe preceder al ciclo (1), de modo que éste pueda saltarse en el caso de que FR(I) sea cero.

El ordenador anuncia el fin de la ejecución de un programa imprimiendo en la pantalla la palabra READY. En ocasiones interesa que no aparezca esta impresión para no estropear lo escrito en la pantalla. Una forma de evitar READY es con una línea como ésta:

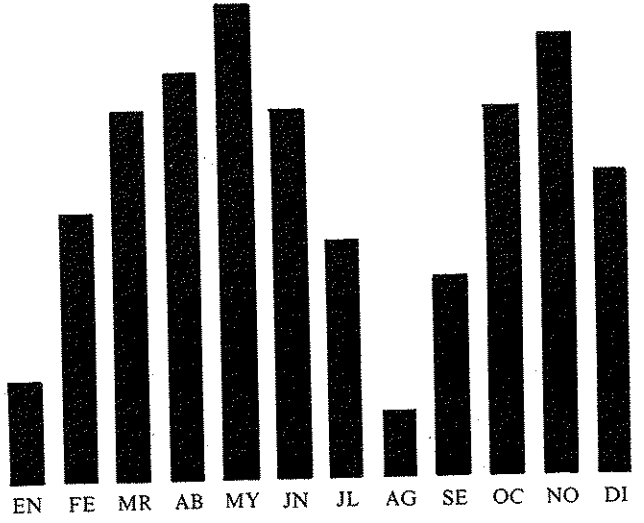
```
300 GOTO 300
```

con lo cual la ejecución se mete en un ciclo sin fin. Para salir de él se pulsa la tecla STOP.

```
10 REM DIAGRAMA DE BARRAS HORIZONTALES
20 INPUT "NUMERO DE BARRAS(12 COMO MAXIMO)";B
30 DIM F(B),FR(B),N$(B)
40 PRINT
50 PRINT "ESCRIBE EL NOMBRE DE LA BARRA Y"
60 PRINT
70 PRINT "LA FRECUENCIA SEPARADOS POR UNA COMA" :
PRINT
80 FOR I=1 TO B
90 INPUT N$(I),F(I)
100 NEXT I
110 MAX=F(1)
120 FOR I=1 TO B
130 IF MAX < F(I) THEN MAX=F(I)
140 NEXT I
150 FOR I=1 TO B
160 FR(I)=INT(F(I)/MAX*30+0.5)
170 NEXT I
180 ( BORRAR LA PANTALLA )
190 FOR I=1 TO B
200 PRINT LEFT$(N$(I),2);" ";
210 IF FR(I)=0 THEN 250
220 FOR J=1 TO FR(I)
230 PRINT "■";
240 NEXT J
250 PRINT F(I)
260 PRINT
270 NEXT I
280 GOTO 280
290 END
```


47. DIAGRAMA DE BARRAS VERTICALES

Programa un diagrama de barras verticales para representar las ventas mensuales de una empresa.



Vamos a resolver este problema concreto con un programa muy general, que servirá para representar un diagrama de barras en un caso cualquiera. El programa está preparado para que pueda tener salida por la impresora, si así lo deseas.

El número máximo de barras queda establecido en 12 y la altura máxima de las barras en 20.

El número B, de barras que vamos a dibujar, se introduce al principio con INPUT; así como los nombres N$(J) de las barras y las frecuencias F(J) correspondientes. Al tiempo se va obteniendo el valor máximo de las frecuencias.

Para llegar a imprimir el diagrama te proponemos la siguiente idea:

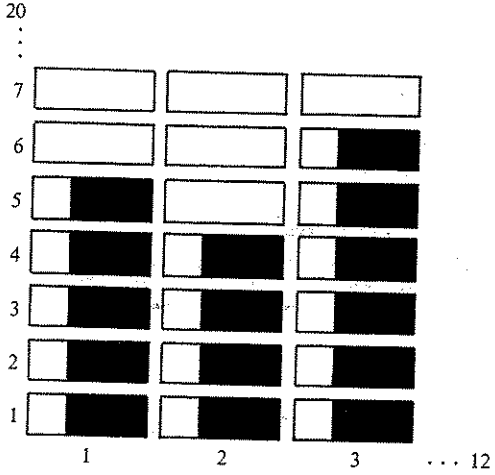
Establecer una tabla de cadenas de caracteres T$(I,J), de 20 filas y B columnas (I = 1,2,...,20; J = 1,2,...,B):

T$(20,1) T$(20,2) . . . T$(20,B)
. . . . . .
. . . . . .
. . . . . .
T$(2,1) T$(2,2) . . . T$(2,B)
T$(1,1) T$(1,2) . . . T$(1,B)

Los elementos de la tabla van a ser uno de estos dos motivos:

Ladrillo blanco: LB$ = "□□□" (tres espacios)
Ladrillo negro: LN$ = "□■■" (un espacio y dos cuadraditos negros)

Un diagrama de barras es como un muro construido con estos ladrillos, con la condición de que nunca haya un ladrillo negro sobre un ladrillo blanco.



Al ejecutar el programa, primero se rellena toda la tabla de ladrillos blancos y después, en cada columna, se cambian por ladrillos negros hasta la altura correspondiente. A la hora de imprimir la tabla, lo haremos por filas, de arriba abajo, con lo que quedará dibujado el diagrama. Debajo imprimiremos las dos primeras letras del nombre de cada barra.

Para cada columna J, la pila correspondiente de ladrillos negros se forma con el ciclo:

```
FOR I=0 TO FR(J)
T$(I,J)=LN$
NEXT I
```


siendo FR(J) la frecuencia reducida (o ampliada), de modo que la barra más alta tenga siempre 20 de altura.

Observa que el ciclo se inicia siempre por I=0, cuyos elementos serían como los cimientos del muro. Sin embargo, la impresión de la tabla se hace a partir de I=1, con lo cual los cimientos no figurarán en el dibujo. ¿Por qué todo este lío? Por que el ciclo FOR-NEXT se ejecuta al menos una vez, y si la frecuencia FR(J) fuese cero y el índice I empezara en 1, entonces el ciclo:

```
FOR I=1 TO 0
T$(I,J)=LN$
NEXT I
```

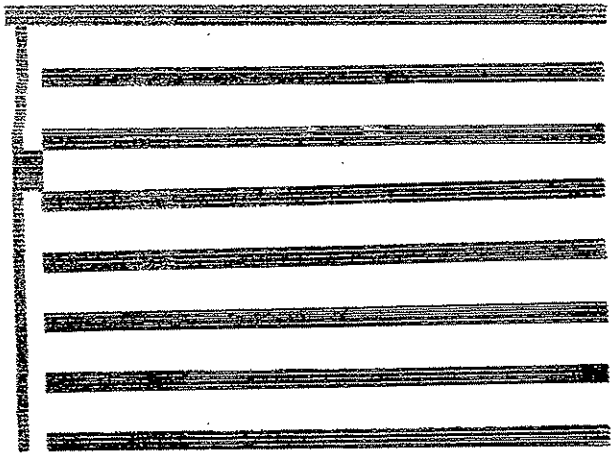
daría lugar a una barra vertical de altura 1, y no debería haber barra.

```
10 REM DIAGRAMA DE BARRAS VERTICALES
20 INPUT "NUMERO DE BARRAS ( 12 COMO MAXIMO )";B
30 DIM T$(20,B),N$(B),F(B),FR(B)
40 MAX=0
50 LB$="□□□" : LN$="□■■"
60 FOR I=1 TO 20
70 FOR J=1 TO B
80 T$(I,J)=LB$
90 NEXT J
100 NEXT I
110 FOR J=1 TO B
120 PRINT
130 INPUT "NOMBRE DE LA BARRA";N$(J)
140 PRINT "VALOR DE ";N$(J);
150 INPUT F(J)
160 IF MAX<F(J) THEN MAX=F(J)
170 NEXT J
180 FOR J=1 TO B
190 FR(J)=INT(F(J)/MAX*20+0.5)
200 NEXT J
210 FOR J=1 TO B
220 FOR I=0 TO FR(J)
230 T$(I,J)=LN$
240 NEXT I
250 NEXT J
260 ( BORRAR LA PANTALLA )
270 FOR I=20 TO 1 STEP -1
280 FOR J=1 TO B
290 PRINT T$(I,J);
```

```
300 NEXT J
310 PRINT
320 NEXT I
330 PRINT
340 FOR J=1 TO B
350 PRINT " "+LEFT$(N$(J)+" ",2);
360 NEXT J
370 PRINT : PRINT
380 PRINT "FRECUENCIA MAXIMA:";MAX
390 GOTO 390
400 END
```


48. ASCENSOR

Dibuja un esquema de la sección de un edificio en que aparezca el hueco del ascensor, y programa el movimiento de éste.



El programa que presentamos consta de dos partes claramente diferenciadas: el dibujo de la sección del edificio (hasta la línea 80), que emplea la subrutina 500 para dibujar el suelo de las plantas; y las líneas 110 a 140, en las cuales se comprueba si se ha pulsado la tecla de subida o de bajada; en caso afirmativo se envía a las subrutinas correspondientes, que mueven el ascensor mientras pueden o hasta que llega una orden de parada.

La variable P controla el "piso" (la fila) en que se encuentra el ascensor.

```
10 REM ASCENSOR
20 REM EDIFICIO
30 PRINT "■■"; : GOSUB 500
40 FOR P=6 TO 0 STEP -1
50 PRINT : PRINT "▌" : PRINT "▌" : PRINT " ▌"; : GO
SUB 500
60 NEXT P
80 PRINT AT 19,1;"■";AT 20,1;"■"
```

```
100 LET P=19 : LET T$=INKEY$ : IF T$<>"S" THEN GO
TO 100
110 LET T$=INKEY$
120 IF T$="S" THEN GOSUB 300
130 IF T$="B" THEN GOSUB 400
140 GOTO 110
199 STOP
300 REM SUBIDA
310 LET P=P-1
320 PRINT AT P,1;"■";AT P+1,1;"■";AT P+2,1;" "
325 FOR I=1 TO 50 : NEXT I
330 IF INKEY$<>"P" AND P>1 THEN GOTO 310
340 RETURN
400 REM BAJADA
410 LET P=P+1
420 PRINT AT P-1,1;" ";AT P,1;"■";AT P+1,1;"■"
425 FOR I=1 TO 50 : NEXT I
430 IF INKEY$<>"P" AND P<20 THEN GOTO 410
440 RETURN
500 REM PISO
510 FOR C=2 TO 30
520 PRINT "■";
530 NEXT C
540 RETURN
```


49. REPRESENTACION GRAFICA DE FUNCIONES

Vamos a elaborar un programa para representar funciones en la pantalla sin disponer de alta resolución. Lo hacemos pensando en una pantalla de 25 filas por 40 columnas, pero de forma que el programa sea fácilmente trasportable a otras dimensiones.

La idea básica es definir la pantalla como una tabla A$(I,J) de caracteres, que serán blancos, asteriscos o los elementos de los ejes. Primero se meterán en los lugares precisos de la tabla los caracteres correspondientes, y al final se imprimirá la tabla entera, apareciendo la gráfica.

Dimensionamos A$(23,38) porque el cero también cuenta y porque no se puede llegar al final de cada fila, al elemento 39: si no, se imprimiría una línea en blanco. Algo parecido suele suceder con el 24, que puede hacer perder la fila superior.

Una de las partes importantes del programa son las denominadas VARIABLES AUXILIARES (líneas 190 a 290):

- IN: es el valor del incremento de la X, la amplitud total del intervalo dividida entre 38.
- M1: es el valor de la X más próximo a cero. Si la X pasa de positiva a negativa, M1 servirá para señalar la columna en la que dibujaremos el eje OY.
- M2: valor de la Y más próximo a cero. Si la función cambia de signo, M2 servirá para dibujar el eje OX.
- MIN: valor mínimo de la función.
- MAX: valor máximo de la función.

Estos últimos sirven para centrar la escala de las ordenadas: situaremos a la mitad el valor YM=(MIN+MAX)/2. Después, y para no distorsionar la escala, tomaremos como incremento en el eje OY el mismo que en el eje OX, de forma que el valor mínimo admitido para la Y será Y1=YM−11·IN y el máximo Y2=YM+11·IN.

Después se representan los ejes, si es que el intervalo de curva a representar los corta. El eje OY se representa en la columna (M1−X1)/IN y el eje OX en la fila (M2−Y1)/IN. En la intersección se representa una cruz.

Llega por fin la situación de los asteriscos que representarán la función. Desde I=X1 hasta X2, tomando por incremento IN, se van situando los asteriscos en la posición

$$((FNY(I)-Y1)/IN,(I-X1)/IN)$$

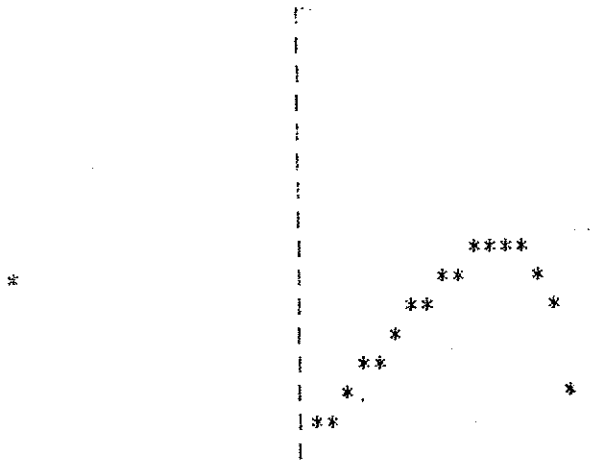
Como los índices han de ser enteros, cada vez redondearemos sus valores.

Las líneas que siguen a la 460 se encargan de llevar a la pantalla los valores de la tabla.

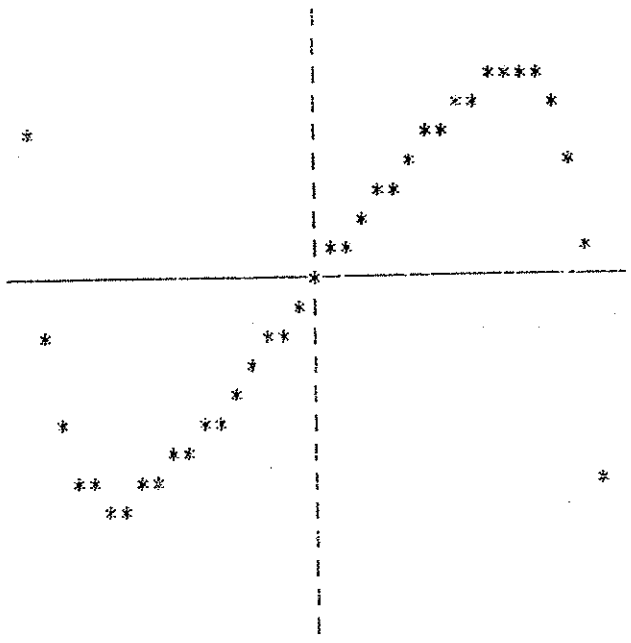
La ejecución de este programa puede traer algunos quebraderos de cabeza. Para muestra un botón: queremos representar la función

$$y = \frac{x^3 - 3x}{x^2 - 4}$$

en el intervalo [−1.9,1.9]. Aunque la curva es una S tumbada, en nuestra máquina hemos obtenido la siguiente representación:



Sin embargo, tomando el intervalo [−1.9,1.91] la representación es la correcta.



El problema es el siguiente: en el bucle de las líneas 220 a 270 podríamos escribir:

220 FOR I=−1.9 TO 1.9 STEP .1

Pero de hecho nuestra máquina no llega a tomar el valor 1.9, porque aparecen unos decimales que exceden de ese valor. Esto descentra la función; en efecto:

f(−1,9)=2,97	f(1,9)=−2,97
f(−1,8)=0,568	f(1,8)=−0,568

(el máximo y mínimo no llegan en valor absoluto a 0,8).

Si se prescinde de f(1,9)=−2,97, la función está comprendida entre −0,568 y 2,97. El punto medio, el centro de la representación, es YM=1,2. Como el incremento es de 0,1, se representan todos los valores de la Y comprendidos entre Y1 e Y2:

$$Y1 = 1{,}2 - 11 \cdot 0{,}1 = 0{,}1$$
$$Y2 = 1{,}2 + 11 \cdot 0{,}1 = 2{,}3$$

lo que explica la aparente representación incorrecta. Probando y eligiendo bien los extremos del intervalo de representación se solucionan todos los problemas.

```
10 REM REPRESENTACION DE FUNCIONES
20 (BORRAR LA PANTALLA)
30 PRINT : PRINT "TECLEA LA FUNCION PRECEDIDA DE"
40 PRINT : PRINT "        100 DEF FNY(X)=..."
50 PRINT : PRINT "DESPUES ESCRIBE RUN 100"
60 STOP
110 (BORRAR LA PANTALLA)
120 DIM A$(23,38)
130 PRINT : PRINT "INTERVALO DE REPRESENTACION"
140 INPUT "(X1<X2)";X1,X2
150 (BORRAR LA PANTALLA)
160 FOR I=0 TO 23 : FOR J=0 TO 38
170 A$(I,J)=" "
180 NEXT J : NEXT I
190 REM VARIABLES AUXILIARES
200 M1=X1 : M2=FNY(X1) : MIN=FNY(X1) : MAX=FNY(X1)
210 IN=(X2-X1)/38
220 FOR I=X1 TO X2 STEP IN
230 IF ABS(I)<ABS(M1) THEN M1=I
240 IF ABS(FNY(I))<ABS(M2) THEN M2=FNY(I)
250 IF FNY(I)<MIN THEN MIN=FNY(I)
260 IF FNY(I)>MAX THEN MAX=FNY(I)
270 NEXT I
280 YM=(MIN+MAX)/2
290 Y1=YM-11*IN : Y2=YM+11*IN
300 REM REPRESENTACION DE LOS EJES
310 IF SGN(X1)=SGN(X2) THEN 350
320 FOR J=0 TO 23
330 A$(J,INT((M1-X1)/IN+.5))="|"
340 NEXT J
350 IF SGN(Y1)=SGN(Y2) THEN 390
360 FOR I=0 TO 38
370 A$(INT((M2-Y1)/IN+.5),I)="-"
380 NEXT I
390 IF SGN(X1)=SGN(X2) OR SGN(Y1)=SGN(Y2) THEN 410
400 A$(INT((M2-Y1)/IN+.5),INT((M1-X1)/IN+.5))="+"
410 REM REPRESENTACION DE LA FUNCION
420 FOR I=X1 TO X2 STEP IN
430 IF FNY(I)<Y1 OR FNY(I)>Y2 THEN 450
440 A$(INT((FNY(I)-Y1)/IN+.5),INT((I-X1)/IN+.5))="*"
450 NEXT I
460 FOR I=23 TO 0 STEP -1 : FOR J=0 TO 38
470 PRINT A$(I,J);
480 NEXT J
490 PRINT
500 NEXT I
510 GOTO 510
520 END
```


50. RECTA ENTRE 2 PUNTOS (6.42)

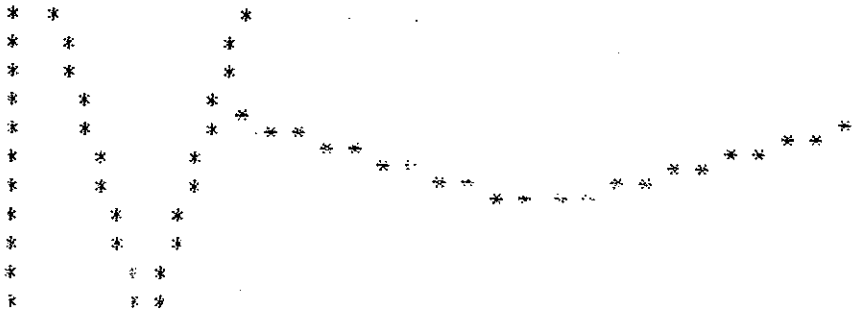
Hacer un programa que, dados dos puntos en la pantalla por sus coordenadas (X columna, Y fila), trace una línea de asteriscos entre ambos.

Supondremos que la pantalla es de 25 filas por 40 columnas, numeradas del 0 al 24 y del 0 al 39. Es peligroso trabajar con el 39 pues, tras imprimir ahí, el cursor pasa a la fila siguiente. Tomamos entonces las columnas del 0 al 38 y, por analogía, las filas del 0 al 23. Si tu microordenador tiene otras dimensiones de pantalla has de corregir las líneas 50, 70, 110 y 120.

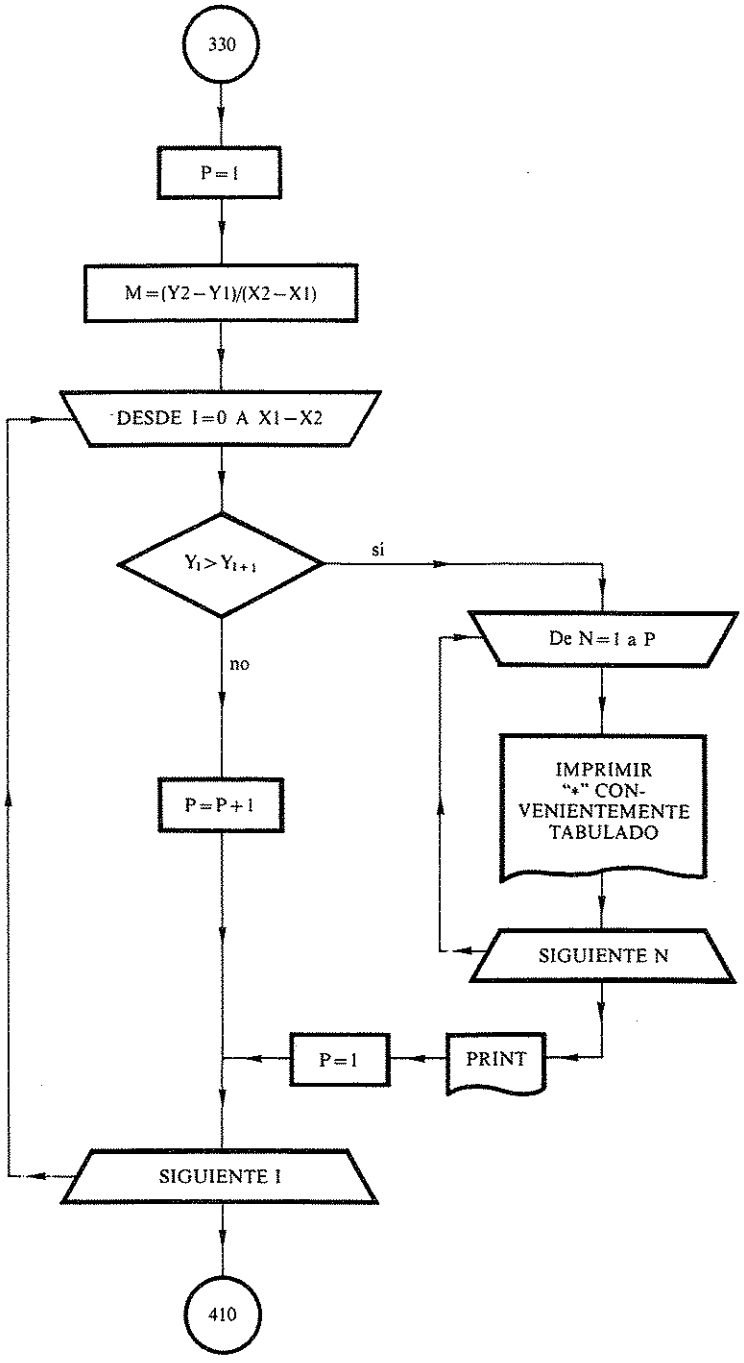
Hasta la línea 70 es la entrada de datos. Después hacemos que (X1, Y1) sea el punto más alto de los dos dados, borramos la pantalla y bajamos el cursor hasta la fila Y1.

Según la pendiente, las rectas se clasifican en los 4 siguientes tipos que se resuelven sucesivamente:

$m=\infty$ $\quad |m|\geq 1$ $\quad -1<m\leq 0$ $\quad 0<m<1$



Los tres primeros tipos son sencillos, se resuelven en pocas líneas, que entenderás fácilmente. En el último no hemos querido utilizar variables de índice, y entonces tiene cierto interés. Todo el problema es que se imprime de arriba a abajo, pero en este caso había que ir de derecha a izquierda, pues puede haber varios asteriscos en una línea. Se recoge el organigrama parcial de este caso, en el que P es el número de asteriscos que se imprimen en una misma fila y M es la pendiente.




```
10 REM RECTA ENTRE DOS PUNTOS
20 (BORRAR LA PANTALLA)
30 PRINT "DAR LOS PUNTOS"
40 INPUT "PUNTO 1 (X,Y)";X1,Y1
50 IF X1>38 OR X1<0 OR Y1>23 OR Y1<0 THEN 40
60 INPUT "PUNTO 2 (X,Y)";X2,Y2
70 IF X2>38 OR X2<0 OR Y2>23 OR Y2<0 THEN 60
80 IF Y2<=Y1 THEN 100
90 X=X1 : Y=Y1 : X1=X2 : Y1=Y2 : X2=X : Y2=Y
100 (BORRAR LA PANTALLA)
110 IF Y1=23 THEN 130
120 FOR I=1 TO 23-Y1 : PRINT : NEXT I
130 IF X1<>X2 THEN 180
140 FOR I=1 TO Y1-Y2+1
150 PRINT TAB(X1);"*"
160 NEXT I
170 GOTO 170
180 IF ABS((Y1-Y2)/(X1-X2))<1 THEN 230
190 FOR I=Y1 TO Y2 STEP -1
200 PRINT TAB(X1+(X2-X1)/(Y1-Y2)*(Y1-I));"*"
210 NEXT I
220 GOTO 220
230 IF X1>X2 THEN 310
240 K=0
250 FOR I=0 TO X2-X1
260 PRINT TAB(X1+I);"*";
270 Y=-(Y2-Y1)/(X2-X1)*(I+1)
280 IF INT(Y+.5)>K THEN PRINT : K=K+1
290 NEXT I
300 GOTO 300
310 P=1 : M=(Y2-Y1)/(X2-X1)
320 FOR I=0 TO X1-X2
330 IF INT(Y1-M*I+.5)>INT(Y1-M*(I+1)+.5) THEN 350
340 P=P+1 : GOTO 390
350 FOR N=1 TO P
360 PRINT TAB(X1-I+N-1);"*";
370 NEXT N
380 PRINT : P=1
390 NEXT I
400 GOTO 400
410 END
```

5. COLOR Y ALTA RESOLUCION

COLOR Y ALTA RESOLUCION

Presentamos a continuación una serie de programas que realizan dibujos en baja y en alta resolución. En baja resolución se considera que la pantalla está formada por un cierto número de filas (generalmente entre 22 y 24) y de columnas (entre 22 y 80, aunque en los micros lo más normal es que sean 22, 32 ó 40). Así queda la pantalla dividida, por una retícula, en una cierta cantidad de pequeños rectángulos, en cada uno de los cuales se puede dibujar una letra, un número o algún carácter especial. De este modo se realizan los dibujos en baja resolución; los caracteres se sitúan en pantalla gracias a dos coordenadas: su fila (se cuenta siempre de arriba a abajo) y su columna (de izquierda a derecha).

En muchos microordenadores, cada vez en más, podemos dibujar también en alta resolución; la pantalla se puede considerar dividida en un número mayor de lugares (68×48, 256×176, 256×192, 640×320, etc.; el primer número se refiere a los puntos en horizontal y el segundo a los puntos en vertical). Conviene tener en cuenta que, en algunos ordenadores, los valores máximos de las coordenadas de los puntos y el número de puntos no coinciden: puede suceder que al pintar en pantalla el punto de coordenadas (1023, 1023) tengamos la misma imagen que al pintar el (1022, 1023), por ejemplo.

Una particularidad a tener en cuenta es que, en bastantes micros, el punto (10,30) está más abajo que el (10,20), porque la segunda coordenada crece desde arriba hacia abajo. Nosotros hemos optado por tomar por origen de coordenadas el vértice inferior izquierdo, y no el superior. Si tu ordenador sitúa el origen arriba, debes realizar siempre un cambio de coordenadas para no verlo todo del revés (en nuestros programas o en los que tú inventes); este cambio, que afecta sólo a la segunda coordenada, para una pantalla de 256×192 puntos será:

$$(x,y) \rightarrow (x,192-y)$$

que en BASIC se expresa:

$$Y=192-Y$$

Para dibujar en color, sea en baja o alta resolución, debes consultar con el mayor detenimiento el manual de tu aparato. Hay ordenadores que no permiten dibujar en alta resolución con todos los colores al tiempo, pero si tu micro carece de esta limitación, puedes mejorar espectacularmente los dibujos que te sugerimos. Buena suerte.

51. BANDERA DE FRANJAS HORIZONTALES

Deseamos dibujar banderas cuyos colores estén dispuestos formando bandas horizontales.

Para definir una bandera debemos especificar cuántas franjas tiene, y cuál es el color y ancho de cada una. Estos valores, color y ancho de cada franja, los guardamos en las columnas primera y segunda de una tabla B, que tendrá tantas filas como franjas tenga la bandera con que se corresponde.

Una vez dados los valores de B, el ordenador va a dibujar cada una de las franjas (línea 90), para ello se prepara a hacer cuadraditos del color correspondiente (línea 100), cosa que tendrá que hacer en tantas filas como indique la anchura de la banda (línea 110). Las líneas 120 a 140 pintan cada una de esas líneas de cuadraditos.

```
10 REM BANDERA DE FRANJAS HORIZONTALES
20 INPUT "NUMERO DE FRANJAS";NU
30 DIM B(NU,2)
40 FOR N=1 TO NU
50 PRINT "COLOR ";N; : INPUT B(N,1)
60 INPUT "ANCHURA ";B(N,2)
70 NEXT N
80 ( BORRAR LA PANTALLA )
90 FOR N=1 TO NU
100 PAPER B(N,1)
110 FOR I=1 TO B(N,2)
120 FOR J=1 TO 30
130 PRINT " ";          (cada espacio —" "— será del color B(N,1)).
140 NEXT J
150 PRINT
160 NEXT I
170 NEXT N
180 END
```


52. BANDERA DE FRANJAS VERTICALES

Si ahora queremos que los colores formen bandas verticales, el problema cambia un poco. Vamos, de todas maneras, a utilizar la misma tabla B del ejercicio anterior, y almacenaremos en ella el color de cada franja y su anchura, en B(N,1) y B(N,2) respectivamente. La variable NU seguirá siendo el número de franjas.

Supondremos que la bandera ha de cubrir 22 filas de pantalla. En cada una de ellas el ordenador debe imprimir una tira horizontal de cuadraditos del primer color B(1,1), de B(1,2) cuadraditos de largo; luego, en la misma fila, otra de color B(2,1) y de B(2,2) cuadrados de largo, y así hasta completar la primera tira horizontal. Luego se pasa a la segunda (línea 160) y se repite el proceso hasta llegar a la fila 22.

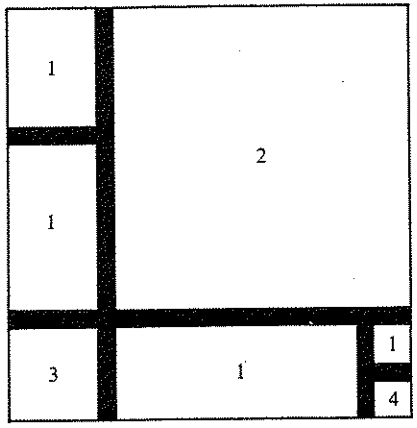
```
10 REM BANDERA VERTICAL
20 INPUT "NUMERO DE FRANJAS ";NU
30 DIM B(NU,2)
40 FOR N=1 TO NU
50 PRINT "COLOR ";N; : INPUT B(N,1)
60 PRINT : INPUT "ANCHURA ";B(N,2)
70 NEXT N
80 ( BORRAR LA PANTALLA )
90 FOR F=0 TO 21
100 FOR N=1 TO NU
110 PAPER B(N,1)
120 FOR I=1 TO B(N,2)
130 PRINT " ";
140 NEXT I
150 NEXT N
160 PRINT
170 NEXT F
180 END
```

53. MONDRIAN

Piet Mondrian ha sido un pintor cuyos cuadros más característicos son de motivos geométricos. Vamos a decirte cómo reproducir en la pantalla de tu ordenador una de sus obras más conocidas. Deseamos que tus diseños alcancen tanta fama como los suyos.

Fondo: Negro

1. *Blanco*
2. *Rojo*
3. *Azul*
4. *Amarillo*



Lo mismo que ocurría con las franjas de las banderas, a cada rectángulo parcial conviene que le asignemos un número, su color, y otros dos relacionados con su largo y ancho, medidos en cuadraditos elementales; pero además le adjudicaremos otros dos, los correspondientes a la fila y columna en que se sitúa su vértice superior izquierdo.

Todos estos valores los almacenamos en líneas DATA.

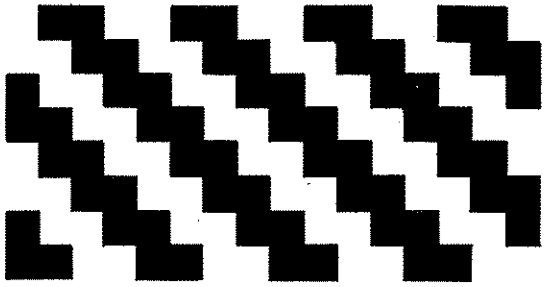
El programa es muy sencillo. Tras poner la pantalla en negro (línea 30), se repite la misma tarea para cada uno de los siete rectángulos: se leen los cinco valores asociados al rectángulo (línea 50); luego el ordenador se prepara a pintar en el color preciso (línea 60), y hace A(2) filas de anchura A(3), comenzando con el cuadradito de "coordenadas" A(4), A(5) (A(4) es la fila en que está el vértice superior izquierdo, y A(5) es la columna de ese mismo vértice).


```
10 REM MONDRIAN
20 DIM A(5)
30 PAPER 0 : CLS
40 FOR I=1 TO 7
50 FOR J=1 TO 5 : READ A(J) :NEXT J
60 PAPER A(1)
70 FOR F=0 TO A(2)
80 FOR C=1 TO A(3)
90 PRINT AT A(4)+F,A(5)+C;" "
100 NEXT C
110 NEXT F
120 NEXT I
200 DATA 7,6,5,0,5
210 DATA 7,7,5,8,5
220 DATA 5,4,5,17,5
230 DATA 2,15,16,0,11
240 DATA 7,4,13,17,11
250 DATA 7,1,2,17,25
260 DATA 6,1,2,20,25
300 END
```

54. ESPIGUILLA

Haz un programa que llene la pantalla de tejido de espiguilla, de colores a elegir.

Se llama tejido de espiguilla a aquel en que los hilos se cruzan dando lugar a un dibujo como el de la figura.



Para fijar ideas vamos a suponer una pantalla de veintidos filas y treinta y dos columnas. Deben aparecer dos tipos de filas: las que tienen un cuadrado en cada extremo y todo lo demás son rectángulos, pares de cuadrados de igual color, y las formadas totalmente por rectángulos.

En la figura hay cuatro clases de filas, ya que se puede comenzar pintando primero con el color más claro, las dos primeras filas, o con el más oscuro, filas tercera y cuarta. Sin embargo, como las filas primera y tercera, por un lado, y la segunda y cuarta, por el otro, tienen la misma estructura, podemos pedirle al ordenador que, tras dibujar una fila de cada tipo, permute los colores. Un algoritmo que describe este proceso es el siguiente:

1. Introducir los colores elegidos
2. Inicializar
3. Borrar la pantalla
4.1 Hacer 11 veces (contador N)

4.2 Si N es impar
entonces
Hacer P=P1 y S=S1
si no
Hacer P=S1 y S=P1
Fin de Si
4.3 Dibujar fila del Tipo 1
4.4 Dibujar fila del Tipo 2
4.5 Siguiente vez
5 FIN

Para fila de Tipo 1 y para fila de Tipo 2 podemos preparar subrutinas de acuerdo con los siguientes algoritmos:

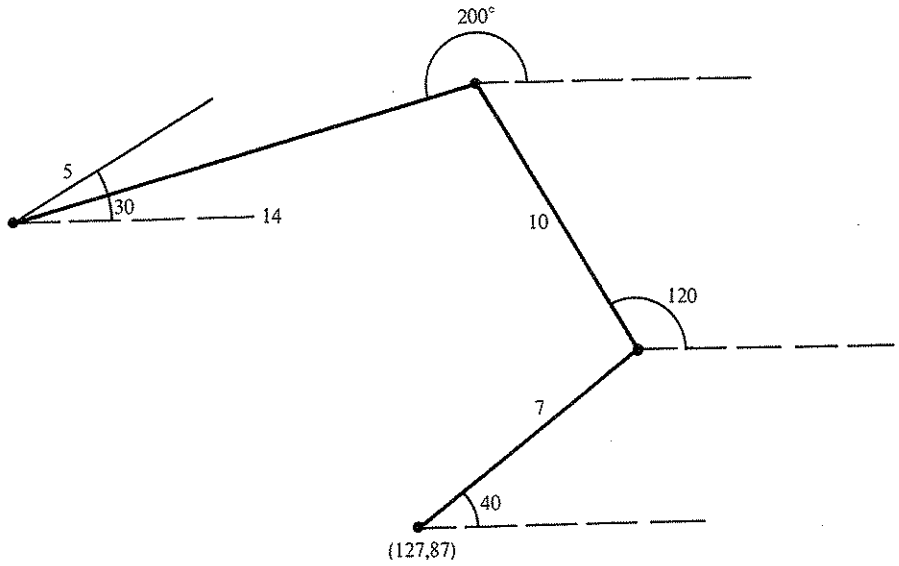
4.3 Dibujar un cuadrado con el color P
Hacer 7 veces
Dibujar 2 cuadrados de color S
Dibujar 2 cuadrados de color P
Siguiente vez
Dibujar 2 cuadrados de color S
Dibujar 1 cuadrado de color P
Fin de la Subrutina
4.4 Hacer 8 veces
Dibujar 2 cuadrados de color P
Dibujar 2 cuadrados de color S
Siguiente vez
Fin de la Subrutina

Al traducir estos algoritmos al BASIC tenemos el programa:

```
10 REM ESPIGUILLA
20 INPUT "COLORES";P1,S1
30 ( BORRAR LA PANTALLA )
40 P=P1 : S=S1
50 FOR F=0 TO 10
60 GOSUB 200 : REM FILA TIPO 1
70 GOSUB 300 : REM FILA TIPO 2
80 IF (F/2-INT(F/2))=0 THEN P=S1 : S=P1 : GOTO 100
90 P=P1 : S=S1
100 NEXT F
110 GOTO 110
120 END
```

```
200 REM FILA TIPO 1 ( EMPIEZA CON UN CUADRADO )
210 PAPER P : PRINT " ";
220 FOR I=1 TO 7
230 PAPER S : PRINT "  ";
240 PAPER P : PRINT "  ";
250 NEXT I
260 PAPER S : PRINT "  ";
270 PAPER P : PRINT " "
280 RETURN
300 REM FILA TIPO 2 (TODO RECTANGULOS )
310 FOR I=1 TO 8
320 PAPER P : PRINT "  ";
330 PAPER S : PRINT "  ";
340 NEXT I
350 RETURN
```


55. LA HORMIGA MAREADA



Si coges una hormiga y le lees con voz monótona, media hora es bastante, la guía telefónica, al dejarla en el suelo es incapaz de seguir un camino recto más de veinte pasos; es más, el rumbo tras esos pasos, los que sean, es totalmente impredictible (la situación es muy parecida a la que se produce cuando un neutrón a alta velocidad intenta atravesar una pared metálica y choca con los átomos que la forman).

Para evitar torturar a ninguna hormiga, hemos simulado su camino en la pantalla de un ordenador. Situamos la hormiga en el punto de coordenadas 127,87 (TX y TY) y el tiempo en cero minutos (líneas 30 y 40): generamos aleatoriamente el número de pasos que va a recorrer en línea recta, R, y el ángulo A de esa recta con la horizontal (línea 50). Ese desplazamiento se traduce en un incremento horizontal, X, y otro vertical, Y; la hormiga en el minuto T+1 ha pasado a estar en TX+X,TY+Y (línea 70). Si no se ha salido de la pantalla (su posición sigue estando entre 0 y 255 en horizontal y entre 0 y 175 en vertical (línea 80)), se dibuja el segmento que va de la posición anterior a la nueva (línea 90), y se repite el proceso; si se salió, nos indicará cuántos minutos ha tardado.

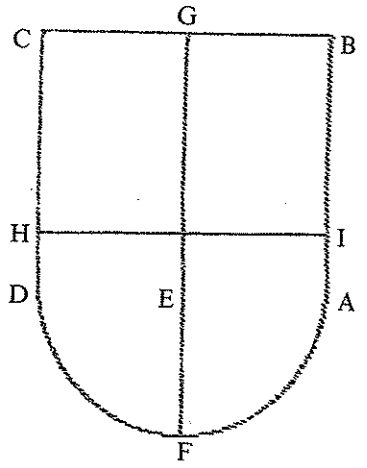
```
10 REM HORMIGA MAREADA
20 ( BORRAR LA PANTALLA )
30 PLOT 127,87
40 T=0 : TX=127 : TY=87 : PI=3.1416
50 R=20*RND : A=2*PI*RND
60 X=R*COS(A) : Y=R*SIN(A)
70 TX=TX+X : TY=TY+Y : T=T+1
80 IF (TX<0) OR (TX>255) OR (TY<0) OR (TY>175) TH
EN 110
90 DRAW X,Y
100 GOTO 50
110 PRINT "HA TARDADO ";T;" MINUTOS EN SALIR"
120 END
```


56. ESCUDO

Programa el dibujo de un escudo cuartelado.

Se llaman cuartelados los escudos divididos en cuatro partes, como indica la figura.

Los puntos tienen las coordenadas siguientes:



A(185,65)
B(185,165)
C(65,165)
D(65,65)
E(125,65)
F(125,5)
G(125,165)
H(65,85)
I (185,85)

Es muy fácil unir mediante segmentos rectilíneos los puntos que se indican. La única excepción se encuentra en la parte inferior, donde los puntos D, F y A se unen empleando una semicircunferencia de centro en E.

```
10 REM ESCUDO CUARTELADO - LINEA
20 ( BORRAR LA PANTALLA )
30 PLOT 185,65
40 DRAW 0,100
50 DRAW -120,0
60 DRAW 0,-100
70 FOR A=3.1416 TO 2*3.1416 STEP 0.01
80 PLOT 125+60*COS(A),65+60*SIN(A)
90 NEXT A
100 PLOT 125,5
110 DRAW 0,160
120 PLOT 65,85
130 DRAW 120,0
140 END
```

Este programa nos da el dibujo en línea, marcando sólo el borde del escudo y la separación entre los cuarteles.

Si queremos el dibujo en color, hemos de elegir el color del fondo de la pantalla y el de las líneas que dibujamos, llenando los cuarteles con estas líneas, de manera que queden pintados del color que deseamos. Un programa que dibuja según hemos indicado es el siguiente:

```
10 REM ESCUDO CUARTELADO - COLOR
20 INPUT "COLORES ";C1,C2
30 ( BORRAR LA PANTALLA )
40 FOR L=165 TO 86 STEP -1
50 INK C1
60 PLOT 65,L : DRAW 60,0
70 INK C2
80 PLOT 125,L : DRAW 60,0
90 NEXT L
100 FOR L=85 TO 65 STEP -1
110 INK C2
120 PLOT 65,L : DRAW 60,0
130 INK C1
140 PLOT 125,L : DRAW 60,0
150 NEXT L
160 FOR L=64 TO 5 STEP -1
170 M=60*COS(ASN((64-L)/60))
180 INK C2
190 PLOT 125-M,L : DRAW M,0
200 INK C1
210 PLOT 125,L : DRAW M,0
220 NEXT L
230 END
```

En algunos ordenadores existe una instrucción que permite cubrir con un color una zona cerrada por una curva. Si tu ordenador es de ésos y, además, esta orden es compatible con la alta resolución (lo habitual es que esto no suceda), entonces podrías colorear fácilmente los cuarteles prolongando el programa primero.

(*) La instrucción DRAW 5,6 dibuja un segmento que va, desde el punto en que nos encontremos, hasta otro situado cinco puntos a la derecha y 6 más arriba. Mientras que DRAW −3,−2 dibuja un segmento que va, desde donde nos encontremos, hasta otro punto situado tres puntos a la izquierda y dos más abajo.

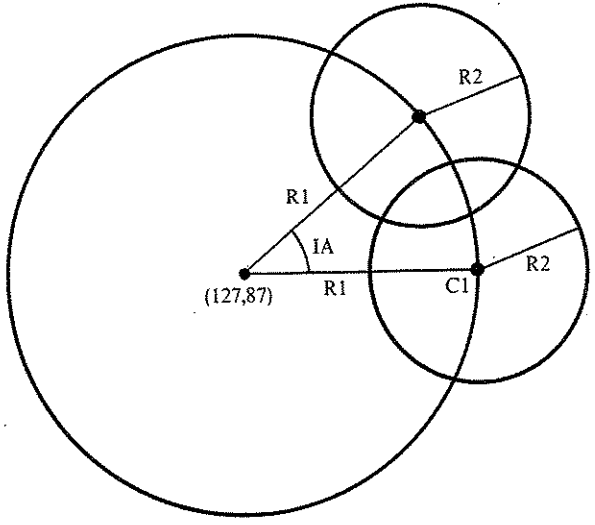
57. FLOR DE CIRCUNFERENCIAS

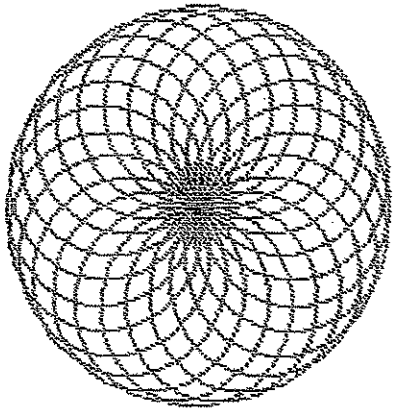
Dibuja flores con circunferencias.

Es muy fácil hacerlo empleando la orden CIRCLE, o una análoga, presente en muchos micros.

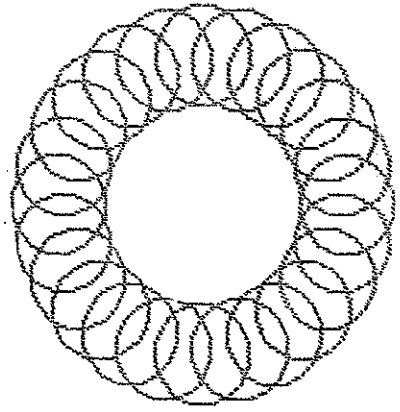
Seguramente lo más simple es suponer una circunferencia imaginaria, cuyo centro va a ser para nosotros el de la pantalla, y cuyo radio variaremos cada vez que ejecutemos el programa. Podemos situar sobre esta circunferencia imaginaria un cierto número de puntos, con la condición de que cada uno diste del anterior lo mismo que del siguiente. A continuación tomamos estos puntos como centros de otras tantas circunferencias, los pétalos de la flor, cuyo radio también habrá que fijar.

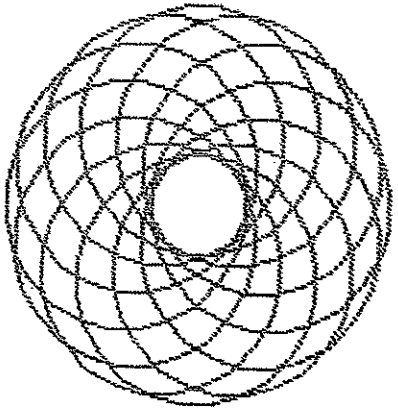
Si los centros de los pétalos están uniformemente distribuidos sobre la circunferencia, y son N, cada uno distará del anterior un arco igual a toda la circunferencia partido por N (IA = 2*π/N, líneas 50 y 90). Esto quiere decir que, si al radio de la circunferencia imaginaria que va al centro de un pétalo, le giramos IA, entonces tenemos el radio que va al centro del pétalo siguiente.





```
10 REM FLOR DE CIRCUNFERENCIAS
20 INPUT "NUMERO DE CIRCUNFERENCIAS";N
30 INPUT "RADIO PRINCIPAL";R1
40 INPUT "RADIO DE LOS PETALOS";R2
50 IA=2*3.1416/N : AN=0
60 ( BORRAR LA PANTALLA )
70 FOR P=1 TO N
80 A=127+R1*COS(AN) : B=87+R1*SIN(AN)
90 CIRCLE A,B,R2
100 AN=AN+IA
110 NEXT P
120 END
```





58. HAZ DE RECTAS

Llamamos haz de rectas de vértice en el punto (C1,C2), al conjunto de todas las rectas del plano que pasan por este punto. ¿Cómo representar un haz de rectas en la pantalla de un ordenador?

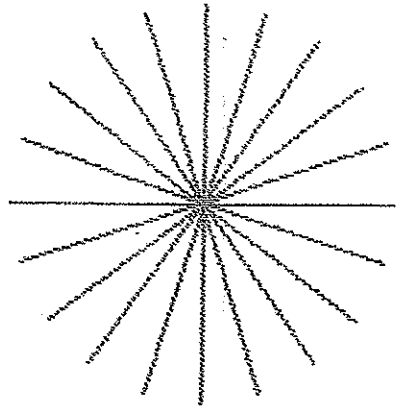
La contestación es muy sencilla: no se puede. No se puede porque el número de rectas de un haz es infinito y representaríamos todo el plano; además, cada una es infinita y no cabe en la pantalla.

Lo que sí podemos hacer es dibujar trozos de algunas de las rectas del haz. En particular, vamos a dibujar segmentos en cuya mitad esté el vértice del haz. También les vamos a pedir que los ángulos que formen las rectas contiguas sean iguales.

Queda claro que los segmentos que cumplen estas condiciones son diámetros, igualmente separados unos de otros, de una circunferencia imaginaria de centro en el vértice del haz. Si, por ejemplo, la longitud de los segmentos es 10, la circunferencia tendrá radio 5 (línea 30).

Vamos a buscar N puntos, tantos como rayos del haz queramos representar, uniformemente separados sobre media circunferencia (línea 80) y los uniremos con los diametralmente opuestos (línea 90).

La línea 110 muestra cómo se aumenta el ángulo que forma con la horizontal, el radio de uno de los puntos empleados; así obtenemos el punto siguiente.

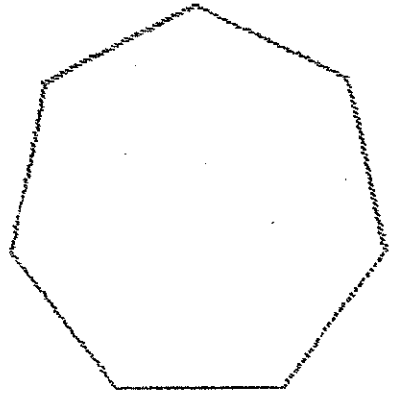


```
10 REM HAZ DE RECTAS
20 INPUT "COORDENADAS DEL CENTRO";C1,C2
30 INPUT "LONGITUD DE LOS SEGMENTOS ( OJO NO TE SAL
GAS )";R : R=R/2
40 INPUT "NUMERO DE SEGMENTOS";N
50 INPUT "ANGULO QUE FORMAN EL PRIMERO Y LA HORIZON
TAL";A : A=A*3.1416/180
60 IA=3.1416/N
70 ( BORRAR LA PANTALLA )
80 FOR I=1 TO N
90 X1=C1+R*COS(A) : Y1=C2+R*SIN(A)
100 X2=C1-R*COS(A) : Y2=C2-R*SIN(A)
110 PLOT X1,Y1 : DRAW X2-X1,Y2-Y1
120 A=A+IA
130 NEXT I
140 END
```


59. POLIGONO REGULAR

Deseamos trazar polígonos regulares de cualquier número de lados, con centro en cualquier lugar de la pantalla, de radio (segmento que une el centro con un vértice) arbitrario y de manera que se pueda elegir también el ángulo que forman la horizontal y el primer radio.

Igual que en Flor de Circunferencias, se trata de tomar puntos uniformemente separados sobre una circunferencia auxiliar. En la línea 60 se obtiene el arco de separación de los vértices. En la 70 se calculan las coordenadas del primer vértice, que no tiene por qué estar sobre un radio horizontal. Ya sólo falta unir los vértices del polígono. Como estamos sobre una circunferencia, el primer vértice es también el siguiente al último.



```
10 REM POLIGONO REGULAR
20 INPUT "NUMERO DE LADOS";N
30 INPUT "COORDENADAS DEL CENTRO";C1,C2
40 INPUT "RADIO";R
50 INPUT "ANGULO DEL PRIMER RADIO Y LA HORIZONTAL";
A : A=A*3.1416/180
60 IA=2*3.1416/N
70 X1=C1+R*COS(A) : Y1=C2+R*SIN(A)
80 ( BORRAR LA PANTALLA )
```

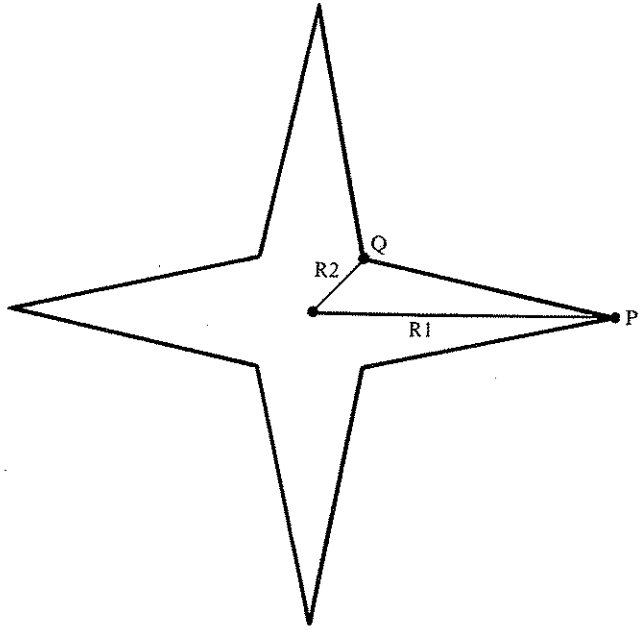
```
90 PLOT X1,Y1
100 FOR I=A+IA TO A+2*3.1416+IA STEP IA
110 X=C1+R*COS(I) : Y=C2+R*SIN(I)
120 DRAW X-X1,Y-Y1
130 X1=X : Y1=Y
140 NEXT I
150 END
```


60. ESTRELLA

Prepara el ordenador para dibujar estrellas de cualquier tamaño, en cualquier lugar, y con cualquier número de puntas. Además, el radio que va del centro de la estrella a la primera punta ha de poder fijarse a voluntad.

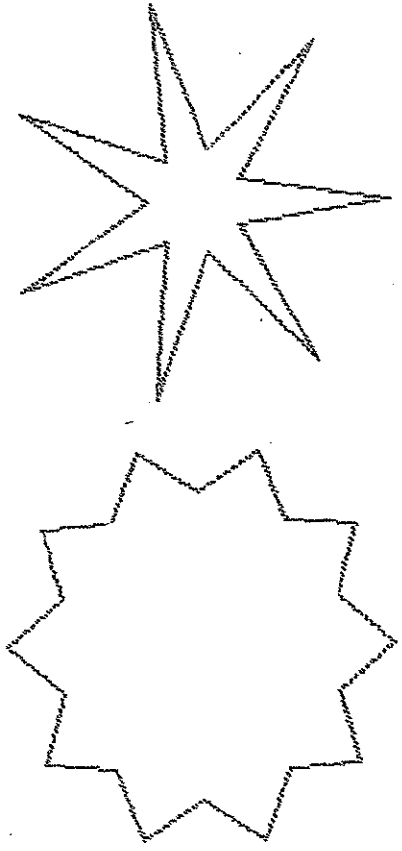
Supongamos que ya hemos elegido dónde queremos situar el centro y a qué distancia de éste han de estar las puntas del tipo P y las del tipo Q. A estas distancias las llamaremos R1 y R2, respectivamente.

Al punto P le podemos llamar punta exterior y a Q punta interior. Resulta que cada punta interior está en la bisectriz del ángulo determinado por los radios que van a dos puntas externas consecutivas. Si las puntas externas están homogéneamente separadas, las internas también lo están.



El problema es totalmente análogo al de dibujar polígonos, con la salvedad de que ahora los puntos que hemos de unir están situados alternadamente sobre dos circunferencias concéntricas.

Se dibuja la primera punta exterior de la estrella, y luego se aumenta en π/N el ángulo del radio que sitúa la punta (π/N es lo mismo que 2π/2N, ten en cuenta que hay N vértices interiores y N exteriores). Se une la primera punta con la primera interior, o al revés (líneas 140 y 170); y así sucesivamente, siempre recordando cuál era la anterior (líneas 150 y 180).



```
10 REM ESTRELLA
20 INPUT "NUMERO DE PUNTAS";N
30 INPUT "COORDENADAS DEL CENTRO";C1,C2
40 INPUT "RADIO MAYOR";R1
50 INPUT "RADIO MENOR";R2
60 INPUT "ANGULO QUE FORMAN EL RADIO DE LA PRIMERA
PUNTA Y LA HORIZONTAL";A
70 A=A*3.1416/180 : IA=3.1416/N
```



```
80 ( BORRAR LA PANTALLA )
90 X=C1+R1*COS(A) : Y=C2+R1*SIN(A)
100 PLOT X,Y
110 FOR I=1 TO N
120 A=A+IA
130 X1=C1+R2*COS(A) : Y1=C2+R2*SIN(A)
140 DRAW X1-X,Y1-Y : A=A+IA
150 X=X1 : Y=Y1
160 X1=C1+R1*COS(A) : Y1=C2+R1*SIN(A)
170 DRAW X1-X,Y1-Y
180 X=X1 : Y=Y1
190 NEXT I
200 END
```

61. CLAVOS Y CUERDAS

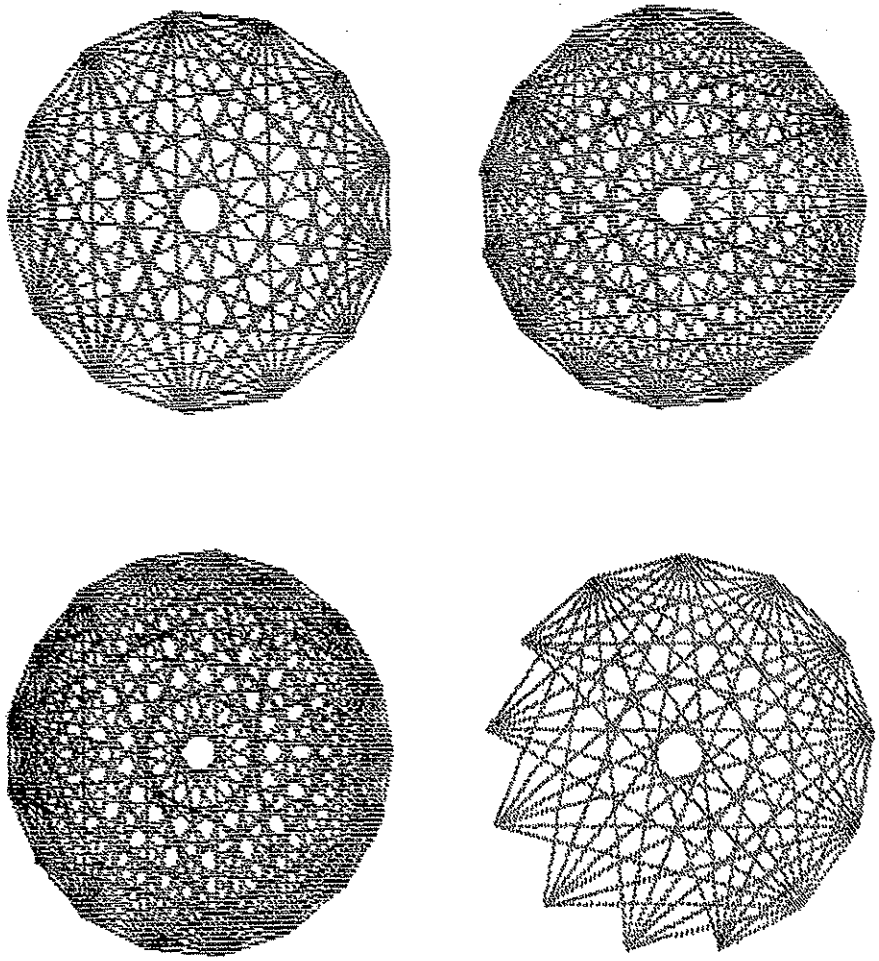
Hay unos cuadros de tipo geométrico que se fabrican con clavos y cuerdas. Los clavos se reparten homogéneamente sobre una circunferencia imaginaria, y se une cada uno de ellos con todos los demás empleando cuerdas finas. Prepara tu ordenador para que haga dibujos de este tipo.

El ejercicio es muy fácil y sin embargo te va a permitir impresionar a todos tus conocidos. Es igual que el del Polígono, la única diferencia estriba en que desde cada vértice (línea 60) has de trazar segmentos a todos los demás (línea 80).

Te recomendamos usar un número impar de vértices. Mira qué pasa si no lo haces.

Si quieres puedes interrumpir el dibujo antes de que esté completo (ver figura), a veces el resultado merece la pena.

```
10 REM CLAVOS Y CUERDAS
20 INPUT "NUMERO DE VERTICES";N
30 INPUT "RADIO";R
40 IA=2*3.1416/N
50 A=0
60 FOR V=1 TO N
70 FOR I=1 TO N-1
80 X1=127+R*COS(A) : Y1=87+R*SIN(A)
90 PLOT X1,Y1
100 X2=127+R*COS(A+(IA*I)) : Y2=87+R*SIN(A+(IA*I))
110 DRAW X2-X1,Y2-Y1
120 NEXT I
130 A=A+IA
140 NEXT V
150 END
```

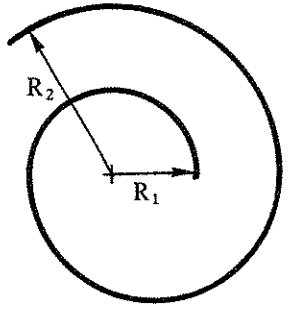



62. ESPIRAL

Programar la espiral de Arquímedes.

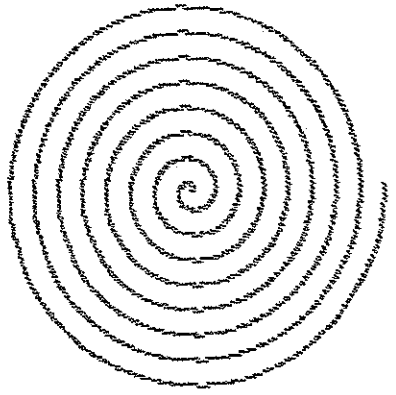
Hay varios tipos de espirales. La más conocida es la de Arquímedes: cada vuelta que da la curva alrededor de su centro hace que los puntos se alejen una cantidad constante P, a la que llamamos paso. Si con un ángulo de 45 un punto de la espiral dista 4 unidades del centro, y el paso es 9, con un ángulo de 405 el punto correspondiente distará 4+9 del centro, y con 765 distará 4+9+9.

Vamos a representar los puntos de la espiral desde que distan R1 del centro, hasta que distan R2. Suponemos el radio inicial horizontal, pero ¿qué ángulo formará con R1 el radio final, R2? Muy sencillo: si en 360 (o 2π radianes) el radio de la espiral pasa de R1 a R1+P, entonces para aumentar de R1 a R2 necesitará un ángulo de $\frac{360*(R2-R1)}{P}$ (o mejor, $\frac{2\pi*(R2-R1)}{P}$ radianes), al que hemos llamado ángulo total AN.



Le pedimos al ordenador que en cada giro total dibuje 4*(R1+R2) puntos. Si te parecen muchos, puedes disminuir el denominador de la fórmula que calcula IA, incremento del ángulo (línea 50), y al disminuir el denominador aumentan los saltos de un punto al siguiente, disminuyendo el número de puntos dibujados.

El primer punto que hemos dibujado está a la derecha del centro, a distancia R1. A partir de éste, hemos de considerar que si en una vuelta (2π radianes) R aumenta en P, en IA radianes (diferencia entre el ángulo asociado al radio de un punto y el asociado al radio del siguiente) R ha de aumentar proporcionalmente en $\frac{IA*P}{2\pi}$ (línea 90).

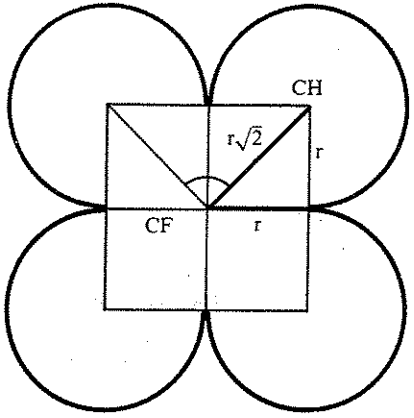


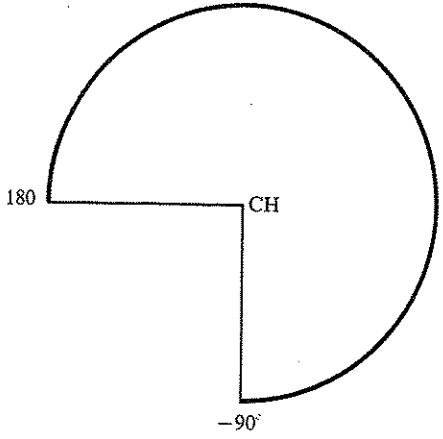
```
10 REM ESPIRAL
20 INPUT "COORDENADAS DEL CENTRO";C1,C2
30 INPUT "RADIOS INICIAL Y FINAL";R1,R2
40 INPUT "PASO";P
50 R=R1 : AN=2*3.1416*(R2-R1)/P : IA=3.1416/(2*(R1+
R2))
60 X=C1+R : Y=C2
70 ( BORRAR LA PANTALLA )
80 PLOT X,Y
90 FOR T=0 TO AN STEP IA
100 R=R+IA*P/(2*3.1416) : X=C1+R*COS(T) : Y=C2+R*SI
N(T)
110 PLOT X,Y
120 NEXT T
130 END
```

63. TREBOL

Prepara un programa que permita dibujar tréboles de cuatro hojas.

Podemos considerar un trébol de cuatro hojas constituido por cuatro arcos, cada uno de ellos igual a tres cuadrantes de circunferencia.

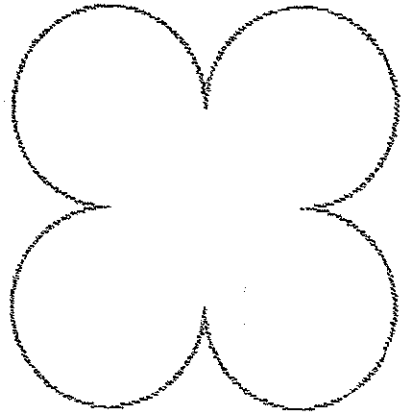




Las hojas tienen como principio y fin los siguientes ángulos:

Pétalo	Principio	Fin
1	−90°	180
2	0	270
3	90°	360°
4	180	450°

Resulta que si a los ángulos del principio y del fin de una hoja les sumamos 90 , tenemos los de la siguiente. Si giramos el centro de la hoja 90° alrededor del centro del trébol, tenemos el centro de la hoja siguiente. Así pues, podemos representar las cuatro hojas empleando una sola expresión.



```
10 REM TREBOL
20 LET PI=3.1416 : LET A=-PI/2 : LET B=PI : LET CH=PI/4
30 (BORRAR LA PANTALLA)
40 FOR H=1 TO 4
50 LET C1=127+40*SQR(2)*COS(CH) : LET C2=87+40*SQR(2)*SIN(CH)
60 FOR T=A TO B STEP 0.02
70 PLOT C1+40*COS(T),C2+40*SIN(T)
80 NEXT T
90 LET CH=CH+PI/2 : LET A=A+PI/2 : LET B=B+PI/2
100 NEXT H
110 END
```

6. GESTION

64. FACTURA

En una tienda especializada en la venta e instalación de moquetas desean una rutina de facturación. Cobran las ventas por metros cuadrados, según la calidad elegida, más un extra de 200 ptas/m² por longitud del borde de la pieza encargada, que ha de rematarse para evitar que se deshaga, más 150 ptas/m² por instalación. Además, como reclamo, hacen un 10% de descuento sobre el total.

Hemos preparado un programa que exhibe la factura en la pantalla del ordenador, como indica la figura.

```
                    FACTURA
-------------------------------------------

PRECIO M2.    380

DIMENSIONES   5 X 4.2

PRECIO                      7980

REMATE                      3680

INSTALACION                 3150
                          -------

TOTAL                      14810

DESCUENTO                   1481
                          -------

A PAGAR                    13329

              GRACIAS POR SU COMPRA
```

Para obtener esta factura hemos recurrido al siguiente programa:

```
10 REM FACTURA
20 INPUT "PRECIO/M2";P
30 INPUT "DIMENSIONES";L,A
100 ( BORRAR LA PANTALLA )
110 PRINT TAB(13);"FACTURA"
```

```
120 PRINT "-------------------------------------------"
130 PRINT : PRINT "PRECIO M2.";TAB(12);P
140 PRINT : PRINT "DIMENSIONES";TAB(12);L;"X";A
150 PR=P*L*A
160 PRINT : PRINT "PRECIO";TAB(28-LEN(STR$(PR)));PR
170 RE=2*(L+A)*200
180 PRINT : PRINT "REMATE";TAB(28-LEN(STR$(RE)));RE
190 I=L*A*150
200 PRINT : PRINT "INSTALACION";TAB(28-LEN(STR$(I))
);I
210 PRINT TAB(20);"-------------------"
220 T=PR+RE+I
230 PRINT : PRINT "TOTAL";TAB(28-LEN(STR$(T)));T
240 D=T*0.1
250 PRINT : PRINT "DESCUENTO";TAB(28-LEN(STR$(D)));D
260 PRINT TAB(20);"-------------------"
270 PA=T-D
280 PRINT : PRINT "A PAGAR";TAB(28-LEN(STR$(PA)));PA
290 PRINT : PRINT TAB(5);"GRACIAS POR SU COMPRA"
300 GOTO 300
310 END
```


65. CAMBIO DE DIVISAS (8.6)

Haz un programa sobre cambio de moneda extranjera, que pueda facilitar la tarea del empleado bancario encargado de este departamento.

Hemos introducido en líneas DATA los nombres de las principales divisas y sus cambios comprador y vendedor.

El programa comienza volcando en tres variables de índice todos los datos: A$(I) es el nombre de la divisa, C(I) es el cambio comprador y V(I) el cambio vendedor. Se aprovecha el propio bucle de lectura para presentarnos los nombres de las divisas.

Después, a través de INPUT, se pregunta la divisa que se desea cambiar y si se trata de comprarla o de venderla. En las líneas 130 y 140 se calcula el caso de comprar divisas y en las líneas 160 a 180 el caso de la venta.

Este programa exige que todos los días que se quiera utilizar se actualicen los datos de las líneas 190 a 270. Una observación: que las palabras no lleven a confusión. Cuando se dice cambio comprador, es que el Banco compra (nosotros vendemos), y de igual manera el cambio vendedor lo es para el Banco (para nosotros es una compra). Por eso en la línea 140 aparece la variable V(N), y en la 180 la C(N), y no al revés.

```
10 REM CAMBIO DE DIVISAS
20 (BORRAR LA PANTALLA)
30 DIM A$(17),C(17),V(17)
40 FOR I=1 TO 17
50 READ A$(I),C(I),V(I)
60 PRINT I;A$(I)
70 NEXT I
80 PRINT
90 INPUT "QUE DIVISA QUIERES (EL NUMERO)";N
100 (BORRAR LA PANTALLA)
110 INPUT "COMPRAS DIVISAS (1) O LAS VENDES (2)";Z
120 IF Z=2 THEN 160
130 INPUT "CUANTAS PESETAS QUIERES CAMBIAR";X
140 PRINT "POR";X;"PESETAS TE DOY";X/V(N);A$(N)
150 GOTO 280
160 PRINT "CUANTOS/AS ";A$(N);" QUIERES VENDER";
170 INPUT X
180 PRINT "POR";X;A$(N);" TE DOY";X*C(N);"PESETAS"
190 DATA DOLARES U.S.A.,151.77,152.12,DOLARES CAN,122.83,123.27
200 DATA FRANCOS F,18.76,18.81,LIBRAS EST,224.50,225.64
```

```
210 DATA LIBRAS IRL;178.33,179.36,FRANCOS S,69.75,70.07
220 DATA FRANCOS B,2.818,2.830,MARCOS,56.43,56.67
230 DATA LIRAS,.0953,.0956,FLORINES,50.48,50.68
240 DATA CORONAS S,19.25,19.32,CORONAS D,15.70,15.76
250 DATA CORONAS N,20.21,20.28,MARCOS FIN,26.47,26.58
260 DATA CHELINES A,8.013,8.058,ESCUDOS,1.233,1.238
270 DATA YENS,.620,.623
280 END
```


66. EL RECIBO DE LA LUZ (4.30)

¿Conoces las operaciones mediante las cuales las empresas eléctricas calculan el recibo de la luz de tu casa? Este programa calcula el recibo de la electricidad para la tarifa de "alumbrado doméstico" (A2), con los precios y disposiciones vigentes en 1983.

En las líneas 20 y 30 se asignan las tarifas actuales a las variables del programa:

PT: factor de potencia. Importe por cada Kw contratado.
K1: precio del Kwh del primer bloque.
K2: precio del Kwh del segundo bloque.
EM: alquiler de los equipos de medida.
OC: impuesto municipal del 1,5% sobre los tres primeros apartados.
TE: I.G.T.E. del 1,57% sobre los tres primeros apartados.
CA: canon de 0,27 pesetas por Kwh consumido.

Con estos precios, los únicos datos necesarios para hacer la factura son la potencia contratada y el consumo en el período bimensual; y se ha obtenido el siguiente resultado:

```
POTENCIA CONTRATADA Y CONSUMO DE ENERGIA
? 5.5 , 1568


     TARI    CONSUMOS
     A2          1568

T.POT    5.5 X 402     2211
KWH1     990 X 6.63    6564
KWH2     578 X 5.67    3277
E.MEDIDA                134
OTROS CONCEPTOS FACT.   181
I.G.T.E.                183
CANON    1568 X .27     423

IMPORTE TOTAL 12973
```

La primera cantidad, 2211, se calcula en la línea 60. Viene a continuación la distinción de los dos bloques: el primer bloque incluye hasta un máximo de 180 horas de utilización, es decir, la potencia contratada multiplicada por 180, que en este caso resulta 990 Kwh. Si el consumo no llega a 990, se factura todo en el primer bloque; si excede ese valor, el exceso se factura en el segundo bloque con una tarifa algo más reducida. En este caso 578 Kwh se facturan en el segundo bloque.

A continuación se factura el alquiler del contador, que es una cantidad constante, y los impuestos, descritos ya al explicar las variables OC, TE y CA.

El programa utiliza al final, en las líneas de impresión de resultados, la función TAB, así como el redondeo a pesetas sin céntimos. El argumento de la función TAB en las líneas 180 a 260 parece complicado. No lo es tanto, tiene por objeto que las cifras de los distintos conceptos se alineen por la derecha, a la columna 26, con independencia de que se trate de cifras de 3, 4 u otro número de dígitos.

```
10 REM RECIBO DE LA LUZ
20 PT=402 : K1=6.63 : K2=5.67
30 EM=134 : OC=.015 : TE=.0157 : CA=.27
40 PRINT "POTENCIA CONTRATADA Y CONSUMO DE ENERGIA"
50 INPUT PC,E
60 PO=PC*PT
70 A=PC*180
80 IF E<=A THEN 120
90 E1=A
100 E2=E-E1
110 GOTO 130
120 E1=E
130 REM REDONDEO E IMPRESION DE RESULTADOS
140 PRINT : PRINT : PRINT
150 PRINT TAB(5);"TARI";TAB(12);"CONSUMOS"
160 PRINT TAB(5);"A2";TAB(15);E
170 PRINT : PRINT
180 PRINT "T.POT";TAB(7);PC;"X";PT;TAB(26-LEN(STR$(INT(PO+.5))));INT(PO+.5)
190 PRINT "KWH1";TAB(7);E1;"X";K1;TAB(26-LEN(STR$(INT(E1*K1+.5))));INT(E1*K1+
.5)
200 IF E2=0 THEN PRINT : GOTO 220
210 PRINT"KWH2";TAB(7);E2;"X";K2;TAB(26-LEN(STR$(INT(E2*K2+.5))));INT(E2*K2+
.5)
220 PRINT "E.MEDIDA";TAB(26-LEN(STR$(EM)));EM
230 Z=INT(PO+.5)+INT(E1*K1+.5)+INT(E2*K2+.5)
240 PRINT "OTROS CONCEPTOS FACT.";TAB(26-LEN(STR$(INT(Z*OC+.5))));INT(Z*
OC+.5)
250 PRINT "I.G.T.E.";TAB(26-LEN(STR$(INT(Z*TE+.5))));INT(Z*TE+.5)
260 PRINT "CANON";TAB(7);E;"X";CA;TAB(26-LEN(STR$(INT(E*CA+.5))));INT(E*CA+
.5)
270 PRINT : PRINT
280 PRINT "IMPORTE TOTAL";Z+EM+INT(Z*OC+.5)+INT(Z*TE+.5)+INT(E*CA+.5)
290 END
```


67. ANUALIDAD CONSTANTE

Haz un programa que calcule la anualidad constante para amortizar un crédito, abonando también sus intereses.

Supongamos que José García obtiene un crédito de 100.000 pesetas a amortizar en 5 años y con un 15% de interés. Las cantidades que deberá reintegrar cada año en concepto de amortizaciones e intereses aparecen en el siguiente cuadro:

AÑO	AMORTIZACION	INTERES	TOTAL	CREDITO POR AMORTIZAR
1	20.000	15.000	35.000	80.000
2	20.000	12.000	32.000	60.000
3	20.000	9.000	29.000	40.000
4	20.000	6.000	26.000	20.000
5	20.000	3.000	23.000	—
	100.000	45.000	145.000	

El interés del primer año es sobre 100.000 pesetas, pero en los años sucesivos va disminuyendo a medida que se amortiza el capital. Esto produce una "injusticia" pues al principio, cuando acaba de obtener el crédito, es cuando más debe pagar, mientras que en el quinto año, con la peseta muy devaluada, el pago es mucho menor.

Vamos a obtener una fórmula que haga que José García pague todos los años la misma cantidad: una anualidad constante. Para ello, y dado que los intereses varían, habrá que hacer variar las amortizaciones.

Si se toma el interés en tanto por uno, la anualidad constante tiene como expresión:

$$A = C \cdot I \cdot \frac{(1 + I)^n}{(1 + I)^n - 1}$$

Siendo I el interés, n el número de años y C el crédito.

El proceso de cálculo es el siguiente: a la anualidad constante (A) se le resta el interés del primer año ($IN = C \cdot I$) y se obtiene la amortización del primer año ($AN = A - IN$).

El segundo año el interés no se calcula sobre C, sino sobre $CN = C - AN$, pues una parte ya se ha amortizado. Ahora $IN = CN \cdot I$. De nuevo la amortización es la diferencia entre la anualidad constante y el interés ($AN = A - IN$). Así debe continuarse hasta el último año, en el que $CN - AN$ debe resultar cero, pues se amortiza definitivamente el crédito. En el ejemplo citado se obtendría el siguiente cuadro:

AÑO	AMORTIZACION	INTERES	TOTAL	CAPITAL
1	14831.56	15000	29831.56	85168.44
2	17056.29	12775.27	29831.56	68112.16
3	19614.73	10216.82	29831.55	48497.42
4	22556.94	7274.61	29831.55	25940.48
5	25940.48	3891.07	29831.55	0
	100000	49157.77	149157.77	

En él se observa que la suma de todas las amortizaciones es igual al crédito obtenido, y así el capital final es cero. Podemos apreciar cómo se ajustan las amortizaciones e intereses de cada año para que la cantidad anual abonada sea constante.

El siguiente programa proporciona el cuadro general con las tres variables apuntadas: interés, número de años y crédito. Para mayor sencillez no imprime la suma final.

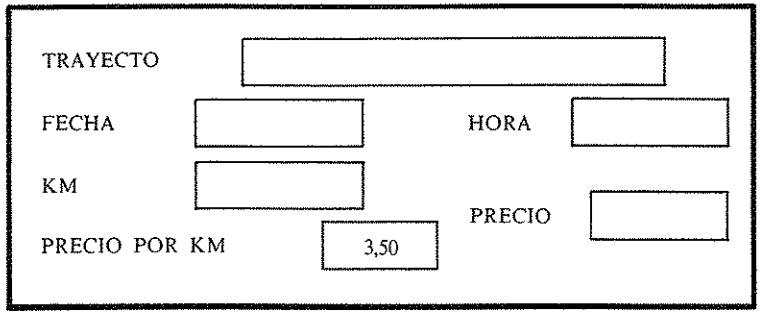
```
10 REM CALCULO DE LA ANUALIDAD CONSTANTE
20 PRINT TAB(5);"PETICION DE DATOS"
30 PRINT : PRINT
40 INPUT "IMPORTE DEL CREDITO";C
50 INPUT "CUANTOS AÑOS";N
60 INPUT "INTERES EN TANTO POR CIENTO";I
70 I=I/100
80 A=C*I*(1+I)↑N/((1+I)↑N-1)
90 REM TITULARES
100 PRINT "AÑO";TAB(5);"AMORTIZ";TAB(15);"INTERES";
110 PRINT TAB(25);"TOTAL";TAB(34);"CAPITAL"
```



```
120 PRINT
130 REM PROCESO ITERATIVO
140 CN=C
150 FOR K=1 TO N
160 IN=CN*I
170 AN=A-IN
180 CN=CN-AN
190 REM REDONDEO E IMPRESION
200 X=INT(AN*100+.5)/100
210 Y=INT(IN*100+.5)/100
220 Z=INT(CN*100+.5)/100
230 PRINT K;TAB(3);X;TAB(13);Y;TAB(23);X+Y;TAB(33);Z
240 NEXT K
250 END
```

68. VENTA DE BILLETES (6.38)

Una empresa de transporte de viajeros por carretera sirve la línea Madrid-Toledo, con paradas intermedias en Getafe, Parla, Torrejón, Illescas, Cabañas, Yuncos y Olías. El precio por km es de 3,50 pesetas, y además los viajeros que utilicen las estaciones de Madrid o Toledo deberán abonar un canon de 5 pesetas. Supondremos que no hay limitación de plazas. Las salidas serán cada hora y media, desde las 8 h a las 21,30, y simultáneamente desde los dos extremos de la línea. La empresa necesita un programa que, tras preguntar el trayecto, día y hora, imprima el siguiente billete:



El siguiente algoritmo lineal resuelve el problema:

Inicializar las variables
Dibujar la pantalla del vendedor
Introducir los datos del billete, cuidando que no coincidan origen y extremo
Calcular la distancia y el precio
Imprimir el billete
Fin

Veamos cómo se traduce en este caso.

Para el empleado que despache los billetes, lo más cómodo será tener en pantalla los nombres de las poblaciones en las cuales la línea tiene parada, un número al lado de cada uno, y pulsar los números correspondientes en lugar de escribir dos nombres cada vez que le pidan un billete. Como además, el

precio del billete depende de la distancia recorrida, hemos dispuesto en instrucciones DATA los nombres de las poblaciones y, entre ellas, las distancias que las separan. Así el ordenador puede formar la tabla T$, como sigue:

MADRID	13
GETAFE	9
PARLA	5
TORREJON	8
ILLESCAS	4
YUNCOS	11
CABAÑAS	8
OLIAS	12
TOLEDO	0

En ella figura, al lado de cada población, la distancia que media entre ésta y la siguiente. En pantalla aparecerá la columna de poblaciones seguida de una columna de números. Cuando alguien pida un billete, el vendedor pulsará los números correspondientes a origen y destino, comprobando el ordenador si la petición es absurda. L1 y L2 son los nombres de las variables que "recuerdan" estos valores. Posteriormente se indican el día, D$, y la hora, H$.

Luego se calculan la distancia entre origen y destino, DI, y el precio provisional, PR. Para conocer el definitivo habrá que saber si origen, destino o ambos coinciden con lugares en los cuales haya que pagar el canon de estación. La variable L es utilizada como variable auxiliar para el cálculo de la distancia.

Hemos supuesto que el lector carece de impresora; por ello la subrutina 500 escribe en pantalla el billete. Si el amable lector fuese, además, afortunado poseedor de una flamante impresora, podría conservar en pantalla la lista de poblaciones e imprimir simultáneamente el billete.

```
10 REM VENTA DE BILLETES
20 DATA MADRID,13,GETAFE,9,PARLA,5,TORREJON,8,ILLESC
AS,4
30 DATA YUNCOS,11,CABAÑAS,8,OLIAS,12,TOLEDO,0
100 GOSUB 200 : REM INICIALIZAR
110 GOSUB 300 : REM PANTALLA
120 GOSUB 400 : REM DISTANCIA Y PRECIO
130 GOSUB 500 : REM BILLETE
140 END
200 DIM T$(9,2)
210 FOR I=1 TO 9
220 FOR J=1 TO 2 : READ T$(I,J) : NEXT J
```

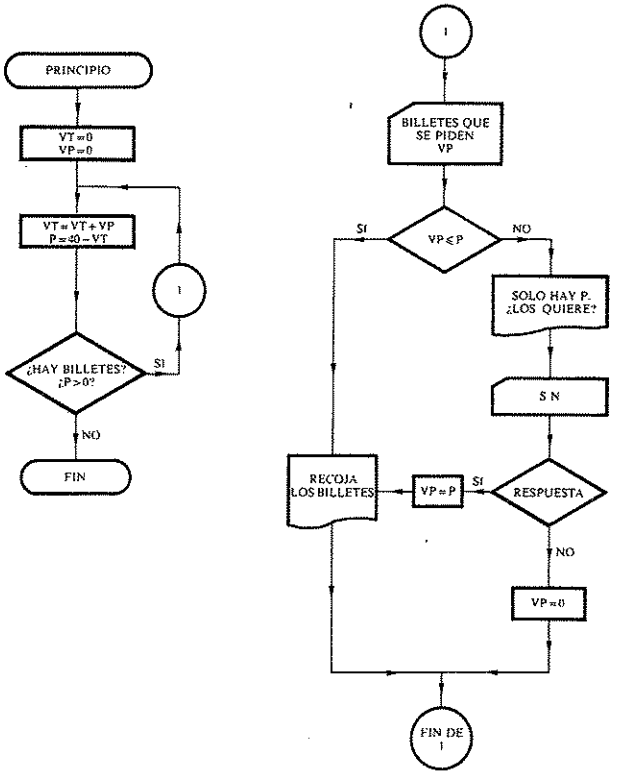
```
230 NEXT I
240 PR=0 : DI=0
250 RETURN
300 ( BORRAR LA PANTALLA )
310 PRINT : PRINT
320 FOR I=1 TO 9 : PRINT TAB(2);T$(I,1);TAB(12);I :
PRINT : NEXT I
330 PRINT : PRINT
340 INPUT "ORIGEN ";L1
350 IF (L1<1) OR (L1>9) THEN PRINT "ERROR" : GOTO 340
360 INPUT "DESTINO ";L2
370 IF (L2<1) OR (L2>9) OR (L2=L1) THEN PRINT "ERROR
" : GOTO 360
380 INPUT "DIA ";D$ : INPUT "HORA ";H$
390 RETURN
400 REM DISTANCIA
410 IF L1>L2 THEN L=L1 : L1=L2 : L2=L
420 FOR L=L1 TO L2 : DI=DI+VAL(T$(L,2)) : NEXT L
430 REM PRECIO
440 PR=DI*3.5
450 IF L1=1 THEN PR=PR+5
460 IF L2=9 THEN PR=PR+5
470 RETURN
500 ( BORRAR LA PANTALLA )
510 PRINT : PRINT
520 PRINT TAB(3);"TRAYECTO";TAB(12);T$(L1,1);TAB(21)
;"*";T$(L2,1)
530 PRINT : PRINT TAB(3);"FECHA";TAB(9);D$;TAB(20);"
HORA";TAB(25);H$ : PRINT
540 PRINT TAB(3);"KMS";TAB(7);DI;TAB(17);"PRECIO";TA
B(24);PR
550 PRINT : PRINT TAB(6);"PRECIO POR KM 3.50"
560 RETURN
```


69. TAQUILLA AUTOMATICA (4.24)

Un ordenador controla el número de plazas libres en un autobús de 40 plazas. A medida que la gente va pidiendo uno o varios billetes quedan menos plazas libres. Si, por ejemplo, alguien pide cinco plazas y sólo quedan tres, contesta con el siguiente mensaje: "No tengo tantas plazas. Sólo me quedan tres. ¿Le interesan?" Cuando el autobús esté completo saca el mensaje de "COMPLETO" y se termina el programa.

En la variable VT recogemos la venta total y en VP la última venta parcial. El número de billetes existentes en cada momento es P = 40 − VT.

El siguiente organigrama ilustra el proceso.



```
10 REM TAQUILLA AUTOMATICA
20 VT=0 : VP=0
30 ( BORRAR LA PANTALLA )
40 VT=VT+VP : P=40-VT
50 FOR I=1 TO 10 : PRINT : NEXT I
60 PRINT "TAQUILLA AUTOMATICA"
70 PRINT : PRINT "SI QUIERE COMPRAR BILLETES PULSE
UNA TECLA"
80 GET A$ : IF A$="" THEN 80
90 ( BORRAR LA PANTALLA )
100 IF P=0 THEN PRINT "COMPLETO. LO SIENTO" : GOTO
210
110 INPUT "CUANTOS BILLETES DESEA";VP
120 IF VP<=P THEN 190
130 PRINT : PRINT "NO HAY TANTOS, SOLO HAY";P
140 PRINT : PRINT "LE INTERESAN.(S/N)?"
150 GET R$
160 IF R$<>"S" AND R$<>"N" THEN 150
170 IF R$="S" THEN VP=P : GOTO 190
180 PRINT : PRINT "LO SIENTO. ADIOS" : VP=0 : GOTO
200
190 PRINT : PRINT "RECOJA LOS BILLETES"
200 FOR I=1 TO 3000 : NEXT I : GOTO 30
210 END
```


70. HIPER

Un almacén con departamentos de ferretería, droguería y bebidas, tiene un servicio de venta a distancia para abonados. Para simplificar, vamos a suponer que en ninguno de sus departamentos se venden más de cuatro artículos. Cuando un cliente intenta comprar algo desde su domicilio, su sistema TV-microordenador-teléfono se conecta con el ordenador del almacén; el cliente teclea su nombre en su microordenador y el ordenador del almacén comprueba si está abonado; en caso afirmativo, exhibe la lista de productos del departamento que se pida, atiende el pedido, si hay existencias suficientes, y actualiza la lista de éstas.

Como los departamentos son tres, se puede pensar en tener tres tablas alfanuméricas distintas, de forma que en la fila cero de cada una ponga el nombre del departamento y en las siguientes los artículos y su cantidad. Sin embargo, las hojas de papel se suelen agrupar en cuadernillos; nada más fácil que hacer lo mismo con estas tablas: la página uno será la de Ferretería, la dos de Droguería y la tres de Bebidas. A esta tabla la llamaremos tabla tridimensional, y sus dimensiones serán 5 (filas), 2 (columnas) y 3 (páginas).

FERRETERIA	
TORNILLOS	52
CLAVOS	61
TUERCAS	50
GRIFOS	14

DROGUERIA	
FERRETERIA	23
TORNILLOS	52
CLAVOS	61
TUERCAS	50
GRIFOS	14

BEBIDAS	
	19
	27
	30
	X
	21
	34
	14

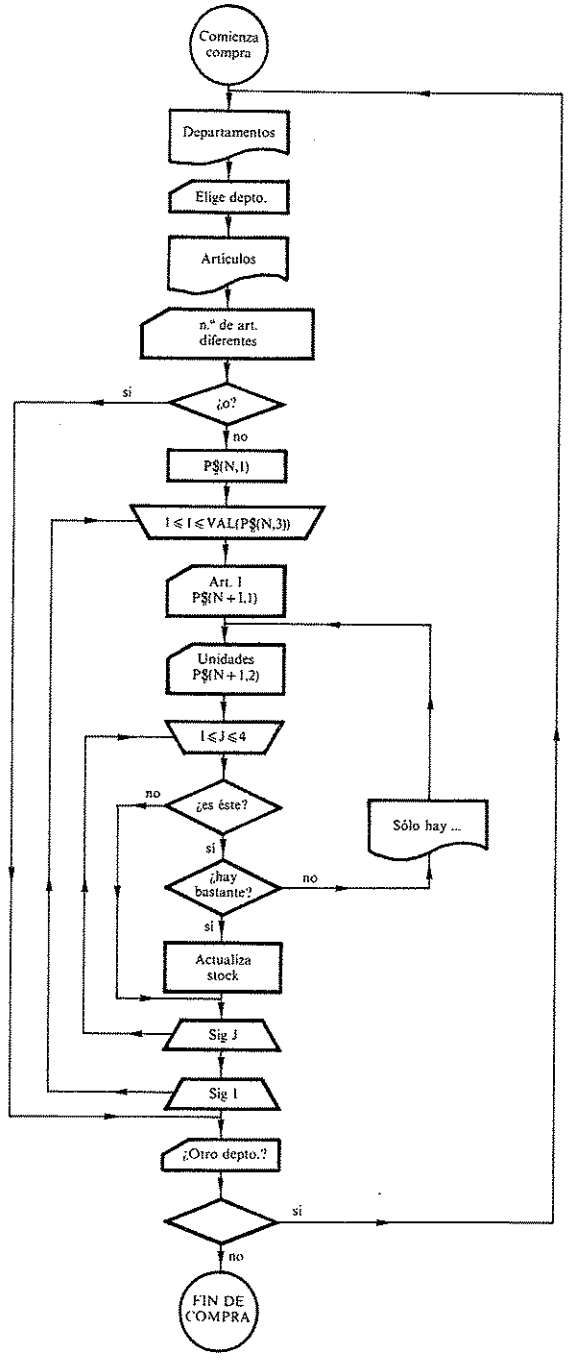
Cuando llame un cliente, el ordenador comprobará si figura en el fichero de abonados y, en caso afirmativo, atenderá su pedido. En la pantalla de TV del usuario aparecerá primero la lista de departamentos y luego la de artículos del departamento solicitado. Los pedidos se estructuran como una tabla de tres

columnas y número suficiente de filas, P$(50,3): o bien en la primera columna figura el nombre de un departamento y en la tercera columna el número de artículos de ese departamento que hay en el pedido, o bien en la primera el nombre de un artículo del departamento citado y en la segunda el número de unidades que se solicitan. La variable N nos indica en cada momento en qué fila nos encontramos. Esta tabla aparecerá en la pantalla al final de la comunicación.

P$

1	FERRETERIA		2
2	TORNILLOS	10	
3	TUERCAS	10	
4	DROGUERIA		3
.	.	.	.
.	.	.	.
.	.	.	.
50			

Desarrollamos con detenimiento el diagrama de flujo del proceso de petición de departamento y de artículos, único que tiene complicación.



```
10 REM VENTA A DISTANCIA
20 DATA FERRETERIA," ",TORNILLOS,52,CLAVOS,61,TUERCA
S,50,GRIFOS,14
30 DATA DROGUERIA," ",JABON,23,PERFUME,21,INSECTICID
A,34,TINTE,14
40 DATA BEBIDAS," ",ESPUMOSO,19,REFRESCO,27,GASEOSA,
30,X,X
50 DIM A$(4,2,3) : DIM P$(50,3)
100 FOR P=1 TO 3
110 FOR F=0 TO 4
120 FOR C=1 TO 2 :READ A$(F,C,P) : NEXT C
130 NEXT F
140 NEXT P
200 OPEN#2
210 ( BORRAR LA PANTALLA )
220 INPUT "NOMBRE ";N$
230 INPUT#2,S$ : IFS$=N$ THEN CLOSE#2 : GOTO 300
240 IF S$="FIN DE FICHERO" THEN CLOSE#2
250 PRINT " PARA ABONARSE A NUESTRO SERVICIO LLAME A
L TFNO. 8642."
260 FOR I=1 TO 3000 : NEXT I
270 GOTO 200
300 N=1
310 ( BORRAR LA PANTALLA )
320 PRINT TAB(10);"DEPARTAMENTOS" : PRINT
330 FOR P=1 TO 3
340 PRINT TAB(2);A$(0,1,P);
350 FOR I=2+LEN(A$(0,1,P)) TO 25 : PRINT "."; : NEXT
I
360 PRINT P
370 NEXT P
380 PRINT : PRINT : INPUT "QUE DEPARTAMENTO DESEA";P
390 IF (P<1) OR (P>3) THEN 380
400 ( BORRAR LA PANTALLA )
410 PRINT TAB(10);A$(0,1,P) : PRINT
420 FOR F=1 TO 4 : PRINT TAB(2);F;".....";A$(F,1,P)
: NEXT F :PRINT
430 INPUT "NO. DE ARTICULOS DIFERENTES DE ESTE DEPTO
. QUE DESEA";P$(N,3)
440 IF VAL(P$(N,3))=0 THEN 530
450 P$(N,1)=A$(0,1,P)
460 FOR I=1 TO VAL(P$(N,3))
470 PRINT "ART.";I; : INPUT P$(N+I,1) : INPUT "UNIDA
DES";P$(N+I,2)
480 FOR J=1 TO 4 :IF P$(N+I,1)<>A$(J,1,P) THEN 520
490 IF VAL(P$(N+I,2))>VAL(A$(J,2,P)) THEN PRINT "SOL
O HAY ";A$(J,2,P):GOTO 470
```



```
500 A$(J,2,P)=STR$(VAL(A$(J,2,P))-VAL(P$(N+I,2))) :
GOTO 520
510 NEXT J
520 NEXT I
530 INPUT "DESEA COMPRAR EN OTRO DEPTO. ";C$ : IF C$
="SI" THEN N=N+I-1:GOTO 310
600 ( BORRAR LA PANTALLA )
610 FOR K=1 TO N+I
620 FOR J=1 TO 3 : PRINT TAB(10*J);P$(K,J); : NEXT J
630 PRINT
640 NEXT K
650 PRINT "MUCHAS GRACIAS, SR. ";S$
700 END
```

71. ORDENACION ALFABETICA (7.17)

Ordena alfabéticamente, por el método de intercambio, una lista de nombres.

Los nombres van a estar asignados a la variable con índice A$(I). Mediante la variable N damos el número de nombres que deseamos introducir, con un máximo de 100.

El método que utilizaremos para alfabetizar la lista de nombres es el de intercambio, que consiste en comparar dos nombres. Si están bien ordenados, se pasa al siguiente nombre y si no, mediante una variable auxiliar, realizamos el intercambio y luego pasamos al siguiente. Para realizar la comparación utilizamos dos bucles anidados, uno exterior que nos recorre desde el primero hasta el penúltimo y el interior, que nos sirve para realizar las comparaciones.

```
10 REM ORDENACION ALFABETICA
20 DIM A$(100)
30 PRINT "CUANTOS NOMBRES";
40 INPUT N
50 PRINT "INTRODUCE LOS NOMBRES"
60 FOR I=1 TO N
70 INPUT A$(I)
80 NEXT I
90 FOR I=1 TO N-1
100 FOR J=1 TO N-1
110 IF A$(J)<=A$(J+1) THEN 150
120 X$=A$(J)
130 A$(J)=A$(J+1)
140 A$(J+1)=X$
150 NEXT J
160 NEXT I
170 PRINT "NOMBRES ORDENADOS ALFABETICAMENTE"
180 FOR I=1 TO N
190 PRINT A$(I)
200 NEXT I
210 END
```


72. ORDENACION POR INSERCION

Vamos a ordenar de menor a mayor los números de una lista de N elementos.

El método de inserción se usa generalmente cuando se quieren ordenar los nombres de una lista alfabéticamente.

Con un ejemplo entenderemos mejor el procedimiento. Sea una lista de siete elementos:

3 7 4 1 5 9 2

En una variable con índice L(I), guardamos la lista.

L(1)	L(2)	L(3)	L(4)	L(5)	L(6)	L(7)
3	7	4	1	5	9	2

La ordenación de la lista requiere los siguientes pasos:

1.° Se compara el primero con el segundo, que en este caso están ordenados.

2.° Se toma L(3) y se compara con los anteriores; al ser L(1)<L(3), no se alteran sus posiciones; y como L(2)>L(3) hay que ordenarlos. Para ello hay que guardar L(3) en la variable X y poner en L(3) el contenido de L(2).

```
X=L(3)
L(3)=L(2)
L(2)=X
```

Efectuado el cambio, la nueva lista es:

L(1)	L(2)	L(3)	L(4)	L(5)	L(6)	L(7)
3	4	7	1	5	9	2
elementos ordenados			elementos aún no ordenados			

3.º A continuación se compara L(4) con los anteriores; como L(4) es menor que L(1), habrá que poner L(4) en lugar de L(1) y los elementos L(1), L(2), L(3) pasarán a ocupar las posiciones L(2), L(3), L(4) respectivamente.

$$\begin{aligned} X &= L(4) \\ L(4) &= L(3) \\ L(3) &= L(2) \\ L(2) &= L(1) \\ L(1) &= X \end{aligned} \qquad (1)$$

Tendríamos la lista:

L(1)	L(2)	L(3)	L(4)	L(5)	L(6)	L(7)
1	3	4	7	5	9	2
elementos ordenados				elementos aún no ordenados		

En todo momento la lista se compone de dos partes; una parte ordenada, la de la izquierda, que va creciendo a expensas de la parte desordenada.

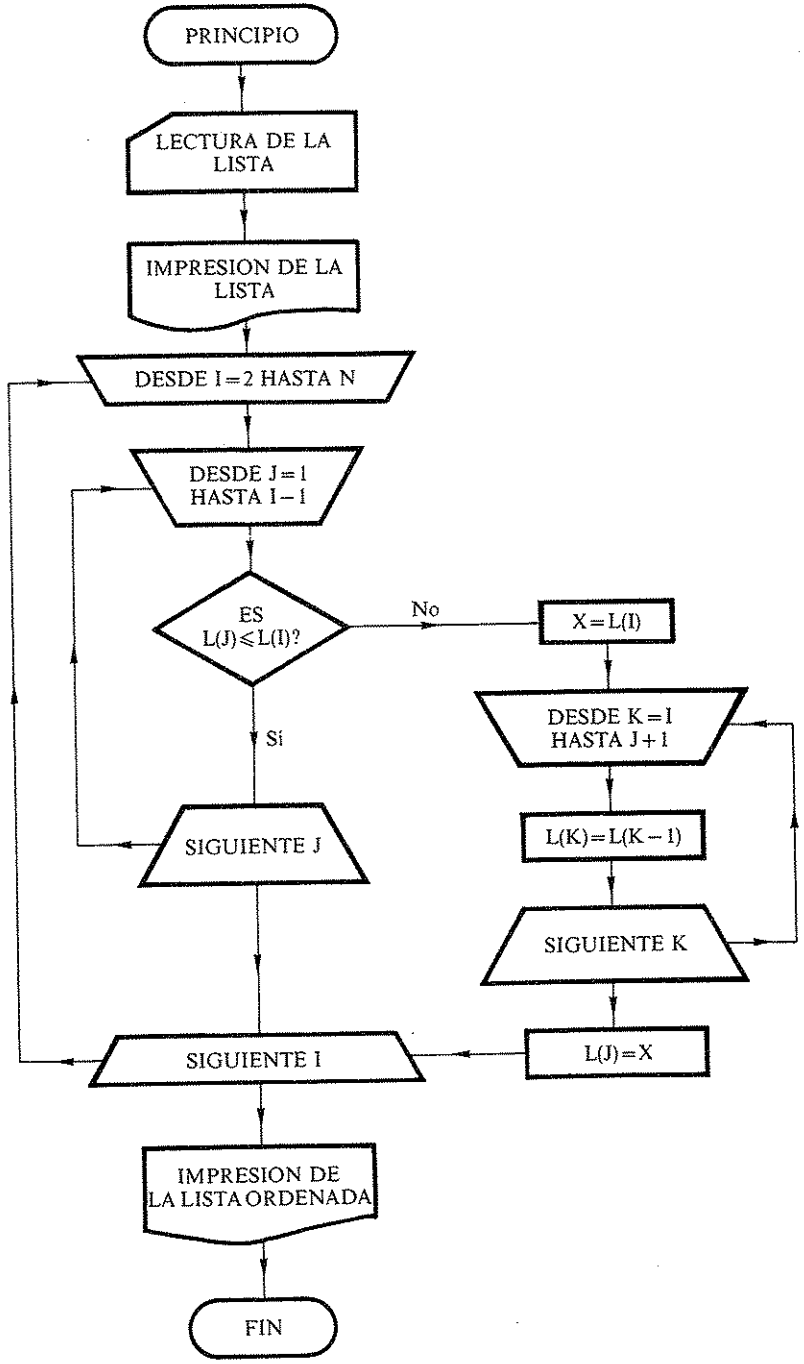
Los pasos 4.º y siguientes consisten en comparar los elementos L(5), L(6)... con los anteriores, efectuando los cambios pertinentes.

El proceso se puede representar brevemente:

a) Comparar cada elemento, a partir de L(2), con todos los anteriores.

b) Si L(J)<L(I), siendo J<I, pasamos al siguiente J; en caso contrario, tenemos que poner L(I) antes de L(J) haciendo cambios como los expresados en (1), que dada su estructura se pueden resumir en un ciclo FOR-NEXT.

Recogemos estas ideas en el organigrama adjunto.



```
10 REM ORDENACION POR INSERCION
20 ( BORRAR LA PANTALLA )
30 INPUT "CUANTOS ELEMENTOS TIENE LA LISTA";N
40 DIM L(N)
50 FOR I=1TO N
60 INPUT L(I)
70 NEXT I
80 ( BORRAR LA PANTALLA )
90 PRINT "LISTA DESORDENADA"
100 PRINT
110 FOR I=1 TO N
120 PRINT L(I);
130 NEXT I
140 PRINT : PRINT : PRINT
150 FOR I=2 TO N
160 FOR J=1 TO I-1
170 IF L(J)<=L(I) THEN 240
180 X=L(I)
190 FOR K=I TO J+1 STEP -1
200 L(K)=L(K-1)
210 NEXT K
220 L(J)=X
230 GOTO 250
240 NEXT J
250 NEXT I
260 PRINT "LISTA ORDENADA"
270 PRINT
280 FOR I=1 TO N
290 PRINT L(I);
300 NEXT I
310 END
```


73. ORDENACION POR EDADES (7.7)

Construye un programa que pida el nombre y la fecha de nacimiento de un cierto número de personas (número menor que mil). Que la máquina, después de tratar estos datos, muestre los nombres y fechas de nacimiento, pero ordenados por edades, de más viejo a más joven. Te conviene escribir las fechas de nacimiento con un sólo número, así: 16 de marzo de 1965 = 19650316.

En la primera parte del programa, hasta la línea 130, se establece un bucle de entrada de datos. El bucle es capaz de leer los datos de hasta 1000 personas. La forma de salir del bucle, de indicar que hemos terminado la introducción de datos, es teclear el nombre ficticio "ZZZ"; de esta manera no hay que contar de antemano el número de datos a introducir.

Previamente hay que dimensionar las variables A$(I) y N(I) que recibirán los datos, y presentar en la pantalla las normas de funcionamiento del programa.

Son los bucles anidados de las líneas 140 a 200 los encargados de la ordenación. Ordenan las fechas de nacimiento y paralelamente los nombres respectivos. Hemos utilizado el método de ordenación por intercambio. Te aconsejamos que escribas en unas octavillas las fechas de unas pocas personas y que ejecutes manualmente el programa con los papelitos. (Para una explicación detenida del método de ordenación ver BASIC BASICO, Curso de Programación, pág. 158.)

```
10 REM ORDENACION POR EDADES
20 (BORRAR LA PANTALLA)
30 DIM A$(1000),N(1000)
40 PRINT : PRINT "INTRODUCIR LOS NOMBRES Y APELLIDOS DE"
50 PRINT "LAS PERSONAS."
60 PRINT "DAR LA FECHA DE NACIMIENTO EN LA FORMA"
70 PRINT "AÑO MES DIA"
80 PRINT "PARA TERMINAR DAR EL NOMBRE FICTICIO ZZZ"
90 FOR I=1 TO 1000
100 INPUT "NOMBRE";A$(I)
110 IF A$(I)="ZZZ" THEN 140
120 INPUT "FECHA";N(I)
130 NEXT I
140 FOR J=1 TO I-2
150 FOR K=1 TO I-1-J
160 IF N(K)<=N(K+1) THEN 190
170 X=N(K) : N(K)=N(K+1) : N(K+1)=X
```

```
180 B$=A$(K) : A$(K)=A$(K+1) : A$(K+1)=B$
190 NEXT K
200 NEXT J
210 REM ESCRITURA DE LA LISTA
220 FOR J=1 TO I-1
230 PRINT A$(J);TAB(25);N(J)
240 NEXT J
250 END
```


74. GEOGRAFIA ECONOMICA (9.24)

Dada la información que figura en la tabla, hemos de preparar un programa que nos proporcione la lista de países ordenados de mayor a menor de acuerdo con uno de los siguientes criterios:

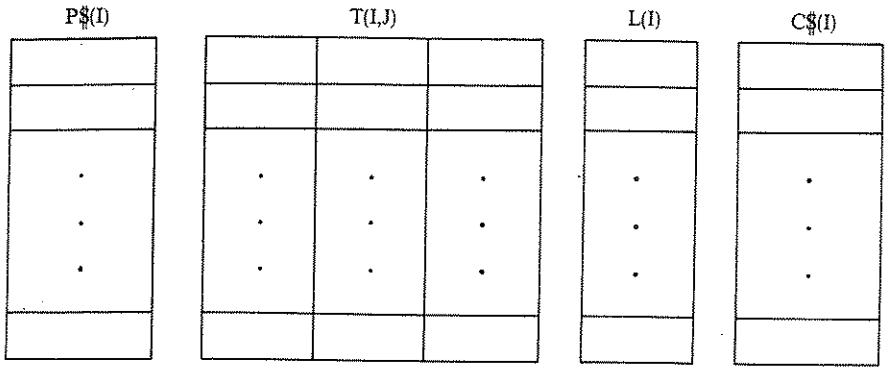
a) *Habitantes por coche.*

b) *Habitantes por km^2.*

c) *Coches por km^2.*

PAIS	COCHES	POBLACION	SUPERFICIE
GRAN BRETAÑA	$13{,}6 \cdot 10^6$	$55{,}9 \cdot 10^6$	$244 \cdot 10^3$ km^2
BELGICA	$2{,}3 \cdot 10^6$	$9{,}8 \cdot 10^6$	$30{,}5 \cdot 10^3$ km^2
DINAMARCA	$1{,}2 \cdot 10^6$	$5 \cdot 10^6$	$43{,}1 \cdot 10^3$ km^2
FRANCIA	$13{,}4 \cdot 10^6$	$52{,}1 \cdot 10^6$	$547 \cdot 10^3$ km^2
ALEMANIA FEDERAL ..	$15{,}6 \cdot 10^6$	$62 \cdot 10^6$	$248{,}6 \cdot 10^3$ km^2
IRLANDA	$0{,}4 \cdot 10^6$	$3 \cdot 10^6$	$70{,}3 \cdot 10^3$ km^2
ITALIA	$12{,}5 \cdot 10^6$	$54{,}9 \cdot 10^6$	$301{,}2 \cdot 10^3$ km^2
LUXEMBURGO	$0{,}1 \cdot 10^6$	$0{,}4 \cdot 10^6$	$2{,}6 \cdot 10^3$ km^2
HOLANDA	$2{,}8 \cdot 10^6$	$13{,}4 \cdot 10^6$	$40{,}8 \cdot 10^3$ km^2

Almacenaremos en una lista, P$(I), los nombres de todos los países que figuran en la primera columna de la tabla. El resto de los valores de ésta los guardamos en una tabla numérica, T(I,J). Como las posibles ordenaciones no serán por los valores de ninguna de las columnas, sino según los cocientes de los valores de dos de ellas, confeccionamos con esos valores una lista, L(I). Una vez formada L, se ordena y se exhiben los nombres de P$ reordenados de acuerdo con el orden de L; para evitar cambiar el orden de P$(I) cada vez, mantenemos ésta invariante y reordenamos una copia suya, C$(I). Los datos, por tanto, quedan organizados de la siguiente manera:



Vamos a utilizar el siguiente algoritmo:

1. Leer P$ y T
2. Elegir ordenación, E (Hab/km^2, Hab/coche, Coche/km^2)
3. Según sea E
   3.1. Dar valores a A y B (para formar L con los valores presentes en las columnas A y B de T)
   Fin de Según
4. Formar C$ y L
5. Reordenar L y C$ (según L)
6. Exhibir C$ y L
7. FIN

El programa quedará como sigue:

```
10 REM GEOGRAFIA ECONOMICA
100 REM INICIALIZAR
110 READ N : DIM P$(N),T(N,3),L(N),C$(N)
120 FOR I=1 TO N : READ P$(I)
130 FOR J=1 TO 3 : READ T(I,J) : NEXT J
140 NEXT I
200 REM MENU
210 ( BORRAR LA PANTALLA )
220 PRINT : PRINT
230 PRINT "ELIGE DE ACUERDO CON EL NUMERO" : PRINT
: PRINT
240 PRINT "HAB/COCHE.............1" : PRINT
250 PRINT "HAB/KM2...............2" : PRINT
260 PRINT "COCHES/KM2............3" : PRINT
```



```
270 INPUT E : IF (E<1) OR (E>3) THEN PRINT "RECTIFI
CA" : GOTO 270
300 ON E GOSUB 1100,1200,1300
400 REM LISTA L, QUE PERMITE ORDENAR
410 FOR I=1 TO N : C$(I)=P$(I) : L(I)=T(I,A)/T(I,B)
: NEXT I
500 REM ORDENACION DE L Y DE LOS PAISES
510 FOR I=1 TO N-1
520 FOR J=1 TO N-I : IF L(J)>=L(J+1) THEN550
530 X=L(J) : L(J)=L(J+1) : L(J+1)=X
540 X$=C$(J) : C$(J)=C$(J+1) : C$(J+1)=X$
550 NEXT J
560 NEXT I
600 REM EXHIBICION
610 ( BORRAR LA PANTALLA )
620 PRINT : PRINT TAB(3);"PAIS";TAB(22);E$ : PRINT
630 FOR I=1 TO N
640 PRINT TAB(1);C$(I);TAB(20);L(I) : PRINT
650 NEXT I
660 GOTO 2000
1090 REM VALORES PARA FORMAR L
1100 A=2 : B=1 : E$="HAB/COCHE" : RETURN
1200 A=2 : B=3 : E$="HAB/KM2" : RETURN
1300 A=1 : B=3 : E$="COCHES/KM2" : RETURN
1500 DATA 9,GRAN BRETAÑA,13500000,55900000,244000
1510 DATA BELGICA,2300000,9800000,30500
1520 DATA DINAMARCA,1200000,5000000,43100
1530 DATA FRANCIA,13400000,52100000,547000
1540 DATA ALEMANIA FEDERAL,15600000,62000000,2486
00
1550 DATA IRLANDA,400000,3000000,70300
1560 DATA ITALIA,12500000,54900000,301200
1570 DATA LUXEMBURGO,100000,400000,2600
1580 DATA HOLANDA,2800000,13400000,40800
2000 END
```

75. CREACION DE UN FICHERO

Haz un programa para almacenar en un fichero los datos de la siguiente tabla.

LUGAR	EMPRESA	VENTAS EN $\$ \cdot 10^6$	ACTIVIDAD
1	Exxon	48.630	Petróleo
2	General Motors	47.181	Automoción
3	Ford	28.839	Automoción
4	Texaco	26.451	Petróleo
5	Mobil Oil	26.062	Petróleo
6	Standard Oil of California	19.434	Petróleo
7	Gulf Oil	16.451	Petróleo
8	I.B.M.	16.304	Electrónica
9	General Electric	15.697	Electrónica
10	Chrysler	15.537	Automoción
11	I.T.T.	11.764	Teléfonos
12	Standard Oil	11.532	Petróleo
13	Shell Oil	9.229	Petróleo
14	U.S. Steel	8.604	Acero
15	Atlantic Richfield	8.462	Petróleo
16	Du Pont	8.361	Químicas
17	Continental Oil	7.957	Petróleo
18	Western Electric	6.930	Electrónica
19	Procter & Gamble	6.512	Prod. de Consumo
20	Tenneco	6.389	Petróleo

Podemos considerar al ordenador integrado por una serie de órganos interconectados:

— la UAL, Unidad Aritmético-Lógica, conjunto de circuitos donde se efectúan las operaciones lógicas y numéricas;

— los órganos de entrada y salida, que le permiten "escuchar" y "hablar";

— la memoria, parte fija (ROM, donde conserva sus conocimientos permanentes, en cierta medida equivalente a nuestra memoria) y parte volátil (RAM, que se borra al apagar y es equivalente a las notas que nosotros tomamos en una cuartilla, que nos permiten recordar instrucciones, direcciones y hacer operaciones, y que tiramos al terminar la jornada de trabajo).

¿Dónde guardar la información que no debemos perder, de una manera que nos permita su fácil recuperación y utilización? ¡En un fichero, claro!

En una oficina un fichero es un armario utilizado para guardar ordenadamente la información. En Informática un fichero es una estructura que permite guardar ordenadamente, y de manera permanente, información. El fichero se crea en la memoria del ordenador, y se pasa, de una vez o en partes, a un soporte material permanente (antes tarjetas o cintas de papel perforadas, ahora casi siempre soporte magnético o tecnología más avanzada, lasser o memoria de burbujas). Este soporte no suele residir en el ordenador sino en un periférico especializado (magnetófono de cassette, unidad de cinta o de disco).

La creación de un fichero supone introducir en el ordenador gran cantidad de datos sin ningún error, ni en el valor de los datos ni en su orden. La mejor manera de lograrlo nos parece que consiste en almacenar transitoriamente la información en líneas DATA, de forma que podamos corregir con mayor facilidad los inevitables errores de transcripción.

El programa que proponemos crea un fichero secuencial. Este tipo de ficheros, el más sencillo, se concibe como una sucesión de valores (todos números o todos caracteres; no es recomendable utilizar ficheros mixtos) que se crea en la memoria volátil y se "vuelca", secuencialmente, en el periférico especializado. Lo mismo que con el fichero-armario, antes de poder introducir o extraer información del fichero-magnético hay que abrirlo, y al final habrá que cerrarlo, empleando instrucciones específicas.

```
10 REM FICHERO SECUENCIAL
20 OPEN#3 : REM ABRIR FICHERO 3
30 FOR I=1 TO 3
40 READ D$ : REM LEER DATO
50 PRINT#3,D$ : REM ESCRIBIR EN EL FICHERO 3
60 NEXT I
70 IF D$="ZZZ" THEN 9000 : REM SEÑAL DE FIN DE FICHE
RO
```

```
80 GOTO 30
100 DATA EXXON,48630,PETROLEO,GENERAL MOTORS,47181,A
UTOMOCION
110 DATA FORD,28839,AUTOMOCION,TEXACO,26451,PETROLEO
120 DATA MOBIL OIL,26062,PETROLEO,STANDARD OIL OF CA
LIFORNIA,19434,PETROLEO
130 DATA GULF OIL,16451,PETROLEO,IBM,16304,ELETRONIC
A
140 DATA GENERAL ELECTRIC,15697,ELETRONICA,CHRYSLER,
15537,AUTOMOCION
150 DATA ITT,11764,TELEFONOS,STANDARD OIL,11532,PETR
OLEO
160 DATA SHELL OIL,9229,PETROLEO,US STEEL,8604,ACERO
170 DATA ATLANTIC RICHFIELD,8462,PETROLEO,DU PONT,83
61,QUIMICAS
180 DATA CONTINENTAL OIL,7957,PETROLEO,WESTERN ELECT
RIC,6930,ELECTRONICA
190 DATA PROCTER&GAMBLE,6512,CONSUMO,TENNECO,6389,PE
TROLEO
2000 DATAZZZ,ZZZ,ZZZ
9000 CLOSE 3
9010 END
```

En muchos ordenadores cuando se abre un fichero hay que indicar, además del número, si se abre para leer o para escribir, si va a ser de tipo secuencial o no, en qué periférico va a residir, etc. Antes de intentar hacer funcionar este programa haz las modificaciones que indique el manual de tu equipo.

El fichero secuencial, que es el más sencillo, no es el más útil ya que para acceder a un dato hay que leer todos los anteriores; pero la creación y utilización de ficheros de acceso directo a los datos depende más aún del sistema informático concreto que estemos empleando, y no son posibles más que utilizando discos.

76. SECTORES INDUSTRIALES

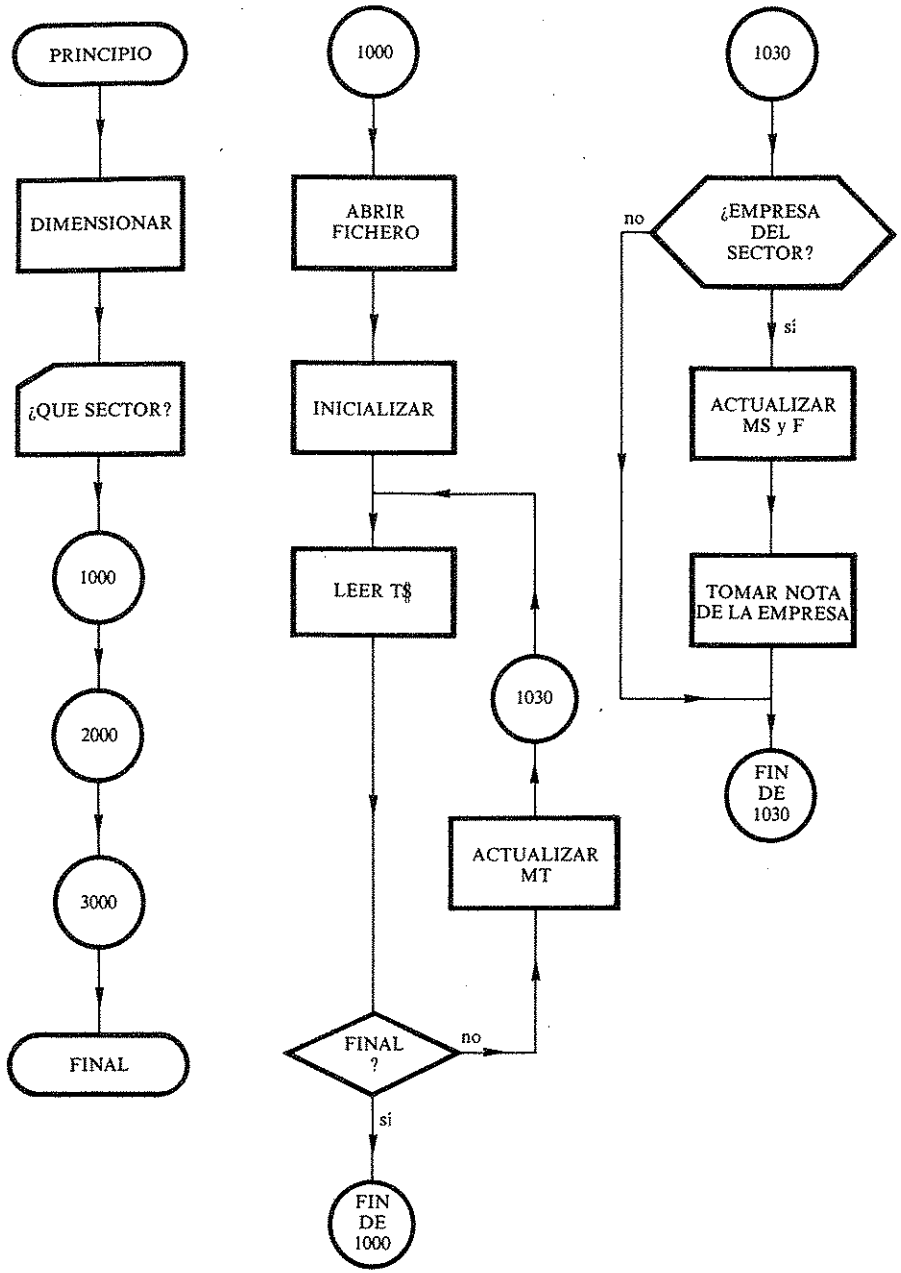
Queremos obtener un programa que, a partir de los datos del fichero del ejercicio anterior, nos proporcione los nombres de las empresas del sector que indiquemos y los porcentajes de cada una en el movimiento económico de este sector y en el total.

Este programa ha de leer el fichero creado en el ejercicio anterior, ir "tomando nota" de las ventas de todas las empresas, sean del sector que sean, en MT; aislar las del sector pedido y anotar en MS el movimiento de éstas. Todo ello lo puede hacer la misma subrutina (líneas 1000 a 1070). Otra ordenará por importancia económica las empresas del sector, si no lo estuvieran. Por último, otra dará la lista y los porcentajes pedidos.

Un desarrollo parcial del diagrama de flujo, que atendiera a las partes del programa con mayor dificultad, sería el que figura en la página siguiente.

No desarrollamos la tercera rutina, que es lineal, ni la segunda, que es una generalización de la ordenación de listas y aparece tratada en otro lugar.

La variable MT representará el movimiento económico total, y MS el movimiento sectorial. S$ será el sector que nos interesa; Z$ la señal de fin de fichero; F el número de empresas en ese sector. Estas empresas, su nombre y ventas las guardaremos en L$ y previamente habrán sido recogidas del fichero por T$, donde figura, además, el sector a que pertenecen. X$ e Y$ son variables auxiliares para la representación final.



El programa quedará así:

```
10 REM ESTUDIO DE LA INDUSTRIA POR SECTORES
20 DIM T$(3),L$(20,2)
30 INPUT "QUE SECTOR LE INTERESA";S$
40 GOSUB 1000 : REM LEER FICHERO
50 GOSUB 2000 : REM ORDENAR EMPRESAS SECTOR
60 GOSUB 3000 : REM EXHIBIR
70 GOTO 4000 : REM FINAL
1000 OPEN#1 : Z$="ZZZ" : MT=0 : MS=0 : F=0
1010 FOR I=1 TO 3 : INPUT#1,T$(I) : NEXT I
1020 IF T$(3)=Z$ THEN 1070
1030 MT=MT+VAL(T$(2)) : IF T$(3)<>S$ THEN 1010
1040 F=F+1 : MS=MS+VAL(T$(2))
1050 FOR I=1 TO 2 : L$(F,I)=T$(I) : NEXT I
1060 GOTO 1010
1070 CLOSE 1
1080 RETURN
2000 FOR I=1 TO F-1
2010 FOR J=1 TO F-I : IF VAL(L$(J,2))>=VAL(L$(J+1,2)
) THEN 2050
2020 X$=L$(J,1) : Y$=L$(J,2)
2030 L$(J,1)=L$(J+1,1) : L$(J,2)=L$(J+1,2)
2040 L$(J+1,1)=X$ : L$(J+1,2)=Y$
2045 PRINT"I=";I;"   J=";J
2050 NEXT J
2060 NEXT I
2070 RETURN
3000 ( BORRAR LA PANTALLA )
3010 PRINT TAB(6);"SECTOR ";S$ : PRINT
3020 PRINT TAB(2);"EMPRESA";TAB(17);"SECTOR";TAB(33)
;"TOTAL"
3030 FOR I=1 TO 38 : PRINT"*"; : NEXT I : PRINT
3040 FOR I=1 TO F
3050 K=5 : V=VAL(L$(I,2))*100/MS : IF V<10 THEN K=4
3060 X$=LEFT$(STR$(V),K)
3070 L=5 : V=VAL(L$(I,2))*100/MT : IF V<10 THEN L=4
3080 Y$=LEFT$(STR$(V),L)
3090 PRINT TAB(0);L$(I,1);TAB(21-K);X$;"%";TAB(37-L)
;Y$;"%"
3100 NEXT I
3110 RETURN
4000 END
```

7. MATEMATICAS

77. TABLAS DE SUMAR

Queremos que aparezca en la pantalla la tabla de sumar de un número previamente elegido, y que después, el ordenador nos la pregunte.

Primero, mediante INPUT, el ordenador nos pedirá el número de la tabla que queremos aprender. Después escribirá la tabla.

Pasado algún tiempo, borrará la pantalla y empezará a preguntarnos la tabla salteada. Al final nos dirá el número de fallos y nos preguntará si queremos continuar.

```
10 REM TABLAS DE SUMAR
20 ( BORRAR LA PANTALLA )
30 F=0 : V=0
40 INPUT "QUIERO ESTUDIAR LA TABLA DEL";N
50 PRINT : PRINT
60 FOR I=1 TO 10
70 PRINT N;"+";I;"=";N+I
80 PRINT
90 NEXT I
100 FOR J=1 TO 5000 : NEXT J
110 ( BORRAR LA PANTALLA )
120 FOR J=1 TO 30
130 I=INT(10*RND)+1
140 PRINT N;"+";I;"=";
150 INPUT S
160 IF S=N+I THEN 210
170 F=F+1
180 PRINT "NO"
190 V=V+1 : IF V=3 THEN V=0 : GOTO 50
200 GOTO 140
210 NEXT J
220 PRINT "HAS TENIDO";F;"FALLOS"
230 PRINT "QUIERES CONTINUAR? (S/N)"
240 GET A$
250 IF A$<>"S" AND A$<>"N" THEN 240
260 IF A$="S" THEN RUN 20
270 ( BORRAR LA PANTALLA )
280 PRINT "! ADIOS !"
290 END
```

78. MULTIPLICAR SUMANDO (4.7)

Imaginando que el ordenador no sabe multiplicar, haz un programa de multiplicación de números enteros por adiciones sucesivas.

Si queremos la multiplicación del número F1, por ejemplo 6, por el número F2, por ejemplo 9, observemos las siguientes sumas:

$$\left.\begin{array}{r} 0+6=6 \\ 6+6=12 \\ 12+6=18 \\ 18+6=24 \\ 24+6=30 \\ 30+6=36 \\ 36+6=42 \\ 42+6=48 \\ 48+6=54 \end{array}\right\}$$

Estas sumas se realizan tantas veces como indica el segundo factor, de modo que el producto es la última suma.

Para la repetición de las sumas sucesivas nos valemos de un contador, que dará por concluidas las sumas al ser igual al segundo factor.

```
10 REM MULTIPLICAR SUMANDO 1
20 PRINT "PRIMER FACTOR";
30 INPUT F1
40 PRINT "SEGUNDO FACTOR";
50 INPUT F2
60 I=0
70 S=0
80 IF I=F2 THEN 120
90 S=S+F1
100 I=I+1
110 GOTO 80
120 PRINT "EL PRODUCTO DE";F1;"POR";F2;"ES";S
130 END
```

Utilizando la instrucción FOR-NEXT el programa queda más sencillo.

```
10 REM MULTIPLICAR SUMANDO 2
20 PRINT "PRIMER FACTOR";
30 INPUT F1
40 PRINT "SEGUNDO FACTOR";
```



```
50 INPUT F2
60 S=0
70 FOR I=1 TO F2
80 S=S+F1
90 NEXT I
100 PRINT "EL PRODUCTO DE";F1;"POR";F2;"ES";S
110 END
```

79. DIVIDIR RESTANDO (4.8)

Suponiendo que el ordenador no sabe dividir, haz un programa de división de dos números por sustracciones sucesivas (división entera que al final dé el cociente sin decimales y el resto).

Queremos efectuar la división entera del número D por el número D1, calculando el cociente y el resto sin utilizar la tecla /.

Observemos qué ocurre si tomamos como ejemplo 25 y 6.

$25-6=19$
$19-6=13$
$13-6=7$
$7-6=1$

Las cuatro sustracciones nos dan el cociente 4, y la última, el resto 1.

El algoritmo que realizamos consiste en repetir las sustracciones sucesivas hasta que la última diferencia sea menor que el divisor.

```
10 REM DIVIDIR RESTANDO 1
20 PRINT "DIVIDENDO";
30 INPUT D
40 PRINT "DIVISOR";
50 INPUT D1
60 REM SE INICIALIZA CONTADOR
70 I=0
80 A=D
90 REM RESTAS SUCESIVAS
100 IF A<D1 THEN 140
110 A=A-D1
120 I=I+1
130 GOTO 100
140 PRINT "EL COCIENTE DE DIVIDIR";D;"POR";D1;"ES";I
150 PRINT
160 PRINT "EL RESTO ES";A
170 END
```

En este caso no se aprecia ventaja con la utilización de la instrucción FOR-NEXT:

```
10 REM DIVIDIR RESTANDO 2
20 PRINT "DIVIDENDO";
30 INPUT D
40 PRINT "DIVISOR";
50 INPUT D1
60 REM SE INICIALIZA CONTADOR
70 A=D
80 FOR I=0 TO D1
90 IF A<D1 THEN 120
100 A=A-D1
110 NEXT I
120 PRINT "EL COCIENTE DE DIVIDIR";D;"POR";D1;"ES";I
130 PRINT
140 PRINT "EL RESTO ES";A
150 END
```

80. SUMA DE POLINOMIOS (8.12)

Prepara un programa que realice la suma de dos polinomios. Los grados de los polinomios no pueden ser mayores de 10.

En principio, se supone que ambos polinomios son de grado 10 y no todos sus coeficientes son 0.

Veamos un ejemplo. Sean los polinomios:

$$P(x)=2x^3+3x^2-x+3 \quad y \quad Q(x)=2x-8$$

El grado de P(x) es 3 e irá en la variable N. El grado de Q(x) es 1 e irá en la variable M. Los coeficientes de P(x) son: A(0)=3, A(1)=−1, A(2)=3, A(3)=2 y los restantes son cero. Los coeficientes de Q(x), son: B(0)=−8, B(1)=2 y los restantes cero.

La suma es:

A(0)+B(0)=−5, que es el coeficiente de grado 0.
A(1)+B(1)= 1, que es el coeficiente de grado 1.
A(2)+B(2)= 3, que es el coeficiente de grado 2.
A(3)+B(3)= 2, que es el coeficiente de grado 3.

La suma de los demás coeficientes da cero por lo que no aparecen en el resultado final.

```
10 REM SUMA DE POLINOMIOS
20 DIM A(10),B(10)
30 REM TODOS LOS COEFICIENTES EN PRINCIPIO SON CERO
40 FOR I=0 TO 10
50 A(I)=0 : B(I)=0
60 NEXT I
70 PRINT "GRADO DEL PRIMER POLINOMIO";
80 INPUT N
90 IF N>10 THEN 70
100 PRINT : PRINT "COEFICIENTES DEL PRIMER POLINOMIO (DE MAYOR A M
ENOR)"
110 FOR I=N TO 0 STEP -1
120 INPUT A(I)
130 NEXT I
140 PRINT : PRINT "GRADO SEGUNDO POLINOMIO";
```



```
150 INPUT M
160 IF M>10 THEN 140
170 PRINT : PRINT "COEFICIENTES DEL SEGUNDO POLINOMIO (DE MAYOR A
MENOR)"
180 FOR I=M TO 0 STEP -1
190 INPUT B(I)
200 NEXT I
210 IF N<=M THEN K=M : GOTO 230
220 K=N
230 REM POLINOMIO SUMA
240 PRINT : PRINT "LA SUMA ES:" : PRINT
250 FOR I=K TO 0 STEP -1
260 PRINT "EL COEFICIENTE DE GRADO";I;"ES";A(I)+B(I)
270 NEXT I
280 END
```

81. SUMA DE FILAS Y COLUMNAS EN UNA TABLA (7.5)

Dada una tabla de 13 × 13 números, tomar cada fila y sumar todos sus elementos y lo mismo con las columnas. Hacer al final la suma de los totales de las filas y la de los totales de las columnas y comprobar que ambas coinciden.

Aunque el enunciado del ejercicio nos dice 13 filas y 13 columnas, vamos a considerar N filas y M columnas, de modo que un INPUT nos permita introducir las filas y columnas que deseemos. En nuestro ejercicio hasta 13 filas y 13 columnas.

La tabla utiliza la variable con dos índices A(I,J). La variable S(I) contiene la suma de cada una de las filas y S1(J) contiene la suma de cada una de las columnas. Las variables T y T1 contienen, respectivamente, la suma total de las sumas de cada fila y de las columnas.

```
10 REM SUMA DE FILAS Y COLUMNAS DE UNA TABLA
20 DIM A(13,13),S(13),S1(13)
30 PRINT "CUANTAS FILAS";
40 INPUT N
50 PRINT "CUANTAS COLUMNAS";
60 INPUT M
70 PRINT "ENTRADA DE LOS ELEMENTOS DE LA TABLA"
80 FOR I=1 TO N
90 FOR J=1 TO M
100 PRINT "FILA";I;"COLUMNA";J
110 INPUT A(I,J)
120 NEXT J
130 NEXT I
140 PRINT "ELEMENTOS DE LA TABLA"
150 FOR I=1 TO N
160 FOR J=1 TO M
170 PRINT A(I,J);
180 NEXT J
190 PRINT
200 NEXT I
210 PRINT
220 FOR I=1 TO N
230 S(I)=0
240 FOR J=1 TO M
250 S(I)=S(I)+A(I,J)
260 NEXT J
270 PRINT "LA SUMA DE LA FILA";I;"ES";S(I)
280 NEXT I
```



```
290 PRINT
300 FOR J=1 TO M
310 S1(J)=0
320 FOR I=1 TO N
330 S1(J)=S1(J)+A(I,J)
340 NEXT I
350 PRINT "LA SUMA DE LA COLUMNA";J;"ES";S1(J)
360 NEXT J
370 PRINT : T=0
380 FOR I=1 TO N : T=T+S(I) : NEXT I
390 PRINT "LA SUMA DE LOS TOTALES DE LAS FILAS ES";T
400 PRINT : T1=0
410 FOR J=1 TO M : T1=T1+S1(J) : NEXT J
420 PRINT "LA SUMA DE LOS TOTALES DE LAS COLUMNAS ES";T1
430 END
```

82. CONVERSION DE ANGULOS

Se trata de hacer un programa que convierta los radianes en grados, minutos y segundos; y recíprocamente, los grados, minutos y segundos en radianes.

El programa es sencillo y lo puedes entender fácilmente.

```
10 REM CONVERSION DE ANGULOS
20 (BORRAR LA PANTALLA)
30 PRINT "DE GRADOS A RADIANES (1) O AL REVES (2)"
40 INPUT Z
50 IF Z=2 THEN 110
60 PRINT "DAR LOS GRADOS MINUTOS Y SEGUNDOS"
70 INPUT "SEPARADOS POR COMAS (,)";G,M,S
80 G=G+M/60+S/3600 : R=G*3.14159265/180
90 PRINT INT(R*1000+.5)/1000;"RADIANES"
100 GOTO 170
110 INPUT "ANGULO EN RADIANES";R
120 G=R*180/3.14159265
130 G1=INT(G) : G=G-G1
140 M=G*60 : M1=INT(M)
150 S=(M-M1)*60 : S1=INT(S*1000+.5)/1000
160 PRINT G1;"GRADOS";M1;"MINUTOS";S1;"SEGUNDOS"
170 END
```


83. CONVERSION DE LONGITUDES (4.22)

Interesa hacer un programa de conversión de longitudes para los ingleses. El ordenador pedirá la longitud en metros, y la convertirá a yardas, pies y pulgadas. Una yarda tiene 0,9144 m, y está dividida en 3 pies y cada pie en 12 pulgadas (no utilices la función INT, sino el algoritmo de "dividir restando").

Tras leer los datos e inicializar las variables estamos en la línea 60. En ella y en las dos siguientes se cuenta el número de yardas. Cada vez que sea L >=0,9144 se le resta ese valor y se cuenta una yarda más.

Cuando es L <0,9144 ya están contadas todas las yardas y hay que pasar a contar los pies por un proceso idéntico. Son las líneas 100 a 130.

Al final las pulgadas son el cociente de la longitud restante y la longitud de una pulgada.

```
10 REM CONVERSION DE LONGITUDES
20 (BORRAR LA PANTALLA)
30 INPUT "LONGITUD EN M.";L
40 YA=.9144 : PI=YA/3 : PU=PI/12
50 N1=0 : N2=0
60 IF L<YA THEN 100
70 N1=N1+1
80 L=L-YA
90 GOTO 60
100 IF L<PI THEN 140
110 N2=N2+1
120 L=L-PI
130 GOTO 100
140 N3=L/PU
150 PRINT : PRINT N1;"YARDAS";N2;"PIES";N3;"PULGADAS"
160 END
```

Utilizando la función INT el programa se simplifica. Basta realizar dos veces la división entera, hallando el cociente entero y el resto. Al final, el último resto pasado a pulgadas se redondea a tres decimales.

```
10 REM CONVERSION DE LONGITUDES
20 (BORRAR LA PANTALLA)
30 INPUT "LONGITUD EN M.";L
40 YA=.9144 : PI=YA/3 : PU=PI/12
50 N1=INT(L/YA)
60 L=L-N1*YA
```

```
70 N2=INT(L/PI)
80 L=L-N2*PI
90 N3=L/PU
100 N3=INT(N3*1000+.5)/1000
110 PRINT : PRINT N1;"YARDAS";N2;"PIES";N3;"PULGADAS"
120 END
```


84. SUMAS PARA PEPITO

Programar la aparición, en la pantalla, de una suma de dos números de cuatro cifras, que Pepito deberá efectuar pulsando las teclas correspondientes.

Debes conseguir que en el centro de la pantalla aparezca, de modo aleatorio, una suma como ésta:

```
  3742
+ 9315
------
```

Para empezar a sumar, Pepito deberá pulsar la tecla del 7, el cual se colocará en el sitio correspondiente de la suma. Si pulsara cualquier otra tecla, no se reflejaría en la pantalla en espera de pulsar la correcta. Y así hasta que Pepito acabe la suma.

El primer problema que se presenta al programar es el de cómo conseguir números aleatorios de cuatro cifras. La instrucción

$$A = INT(9000 \times RND) + 1000$$

asigna a la variable A valores enteros que van desde 1000 a 9999.

De igual modo obtenemos el número B. La suma de los dos se guardan en S. Esta suma será un número de cuatro o cinco cifras. Con objeto de desguazar el número S, esto es, descomponerlo en sus cifras, lo convertimos en cadena de caracteres:

$$S\$ = STR\$(S)$$

Ahora, mediante la función MID$ obtenemos, una a una y de derecha a izquierda, las cifras de S$. La cifra que ocupa el lugar I, empezando por la derecha, se puede obtener así:

$$S\$(I) = MID\$(S\$, L - I + 1, 1)$$

En donde L = LEN(S$) es el número de cifras de la suma: 4 ó 5, según los casos.

La cifra que propone Pepito se guarda en A$, mediante GET, y se coloca en su sitio únicamente en el caso de que sea correcta. Para averiguarlo hay que comparar A$ con la cifra correspondiente, que se encuentra en S$(I).

```
10 REM SUMAS PARA PEPITO
20 (BORRAR LA PANTALLA)
30 PRINT "REALIZA LA SIGUIENTE SUMA"
40 PRINT "PULSANDO LAS TECLAS CORRESPONDIENTES"
50 A=INT(9000*RND+1000)
60 B=INT(9000*RND+1000)
70 S=A+B
80 S$=STR$(S)
90 L=LEN(S$)
100 FOR I=1 TO L
110 S$(I)=MID$(S$,L-I+1,1)
120 NEXT I
130 (CURSOR AL ORIGEN)
140 FOR I=1 TO 10 : PRINT : NEXT I
150 PRINT TAB(17);STR$(A)
160 PRINT TAB(16);"+";STR$(B)
170 PRINT TAB(16);"-------"
180 FOR I=1 TO L-1
190 GET A$
200 IF A$<>S$(I) THEN 190
210 PRINT TAB(22-I);A$
220 (CURSOR A LA LINEA ANTERIOR);
230 NEXT I
240 PRINT : PRINT : PRINT
250 PRINT "QUIERES HACER OTRA SUMA (S/N)?"
260 GET A$
270 IF A$<>"N" AND A$<>"S" THEN 260
280 IF A$="S" THEN 20
290 PRINT : PRINT TAB(10);"ADIOS"
300 END
```

Observación:

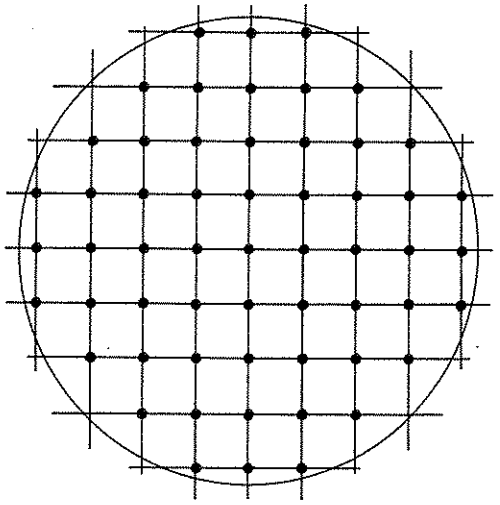
En muchos ordenadores la cadena S$ lleva en ella un espacio a la izquierda. Esto hace que la longitud real de S$ exceda en una unidad al número de cifras de S. Por esta razón, en la línea 180 figura L − 1 y no L, como a primera vista parece que debiera ser.

Desde luego se puede poner L en la línea 180; pero, en este caso, Pepito tiene que acabar la suma pulsando la *tecla espacio.*

85. PUNTOS EN UN CIRCULO

Escribir un programa que calcule los puntos de coordenadas enteras que están en el interior de la circunferencia de ecuación $x^2+y^2=18$.

El radio de la circunferencia es $\sqrt{18}$. Como se observa en la figura, las coordenadas enteras de los puntos interiores varían entre -4 y 4.



Por tanto, mediante dos bucles anidados, uno para la x y otro para la y, podemos "barrer" todos los puntos de coordenadas enteras que están en el interior de la circunferencia y llevar la cuenta, mediante la variable contador C.

Es evidente que la cuenta, en el ejemplo propuesto, se podría haber hecho a mano; sin embargo, proponemos al lector que lo intente con la circunferencia de ecuación $x^2+y^2=1000$.

```
10 REM PUNTOS DE COORDENADAS ENTERAS INTERIORES A UNA CIRCUNFE
RENCIA
20 C=0
30 FOR X=-4 TO 4
40 FOR Y=-4 TO 4
50 IF X↑2+Y↑2>=18 THEN 70
60 C=C+1
70 NEXT Y
80 NEXT X
90 PRINT "EL NUMERO DE PUNTOS DE COORDENADAS ENTERAS ES";C
100 END
```


86. NUMEROS PRIMOS (6.3)

Obtener todos los números primos menores que 342.

En el bucle de las líneas 30 a 100 y variable N, se analizan todos los números desde el 2 al 342. El bucle anidado de las líneas 40 a 80 estudia, para cada N, si tiene algún divisor.

El número I será divisor de N, si el resto de dividir N entre I es cero.

$$R = N - I*INT(N/I)$$

Si N tiene algún divisor, la línea 70 envía a estudiar el siguiente N. Si no, al terminar el examen de los divisores, la línea 90 hace imprimir N, que es primo.

La línea 50 se dirige a abreviar la ejecución del programa. En efecto: al estudiar los divisores de 337 para ver si es primo, basta llegar al primer divisor que cumpla $I*I > 337$, en nuestro caso 19. De faltar esta línea el bucle de variable I continuaría hasta $I = 337$, resultando mucho más lento.

```
10 REM NUMEROS PRIMOS
20 (BORRAR LA PANTALLA)
30 FOR N=2 TO 342
40 FOR I=2 TO N
50 IF I*I>N THEN 90
60 R=N-I*INT(N/I)
70 IF R=0 THEN 100
80 NEXT I
90 PRINT N
100 NEXT N
110 END
```

87. DESCOMPOSICION EN FACTORES PRIMOS (6.4)

Descomponer un número en sus factores primos.

En este ejercicio utilizaremos el mismo algoritmo que los alumnos de E.G.B.

276	2
138	2
69	3
23	23
1	

El núcleo del programa son las líneas 50 a 90. Comenzando por los números D más bajos, se analiza si dividen a N. En caso afirmativo se imprimen N y el divisor, y, para continuar el trabajo siguiendo el mismo algoritmo, se sustituye N por N/D.

Fíjate en que, como en este ejemplo, si 2 es un divisor, hay que volverlo a probar ya que 276/2=138, que también puede ser divisible por 2. De ahí la línea 100.

No se corre ningún riesgo al probar divisores no primos. En efecto, el ordenador siempre dirá que el número N no es divisible por un número compuesto. Pensemos, por ejemplo, que antes de llegar a probar el divisor 4, al número N se le han quitado todos los factores 2 que contenía y, por tanto, el nuevo número N no será divisible por 4.

La línea 60 va dirigida a la brevedad en la ejecución. Si en un momento $D^2 > N$, con todas las divisiones previas realizadas, es que N es primo, y procede acabar el programa. De no existir esta línea, tantearía todos los divisores hasta llegar a 23, mientras que aquí se para en el 5.

```
10 REM DESCOMPOSICION EN FACTORES PRIMOS
20 (BORRAR LA PANTALLA)
30 INPUT "DAR EL NUMERO";N
40 D=1
50 D=D+1
60 IF D*D>N THEN 110
70 IF N-D*INT(N/D)<>0 THEN 50
80 PRINT N,D
90 N=N/D
100 GOTO 60
110 PRINT N,N
120 PRINT 1
130 END
```


88. MAXIMO COMUN DIVISOR (6.5)

Programa el algoritmo de Euclides para el cálculo del máximo común divisor.

El algoritmo de Euclides para el cálculo del máximo común divisor de dos números A y B está basado en la siguiente propiedad:

$$MCD(A,B)=MCD(B,R)$$

en donde R es el resto de la división entera de A por B.

De esta propiedad se deduce que si el resto R es cero, el máximo común divisor es B.

Si el resto no es cero, se efectúa una nueva división llamando A a B y B a R.

Después de un número finito de pasos se conseguirá resto cero. Entonces el máximo común divisor estará en B.

Ilustremos el algoritmo con un caso concreto:

$$MCD(124,14)=?$$

124	14	14	12	12	2
12	8	2	1	0	6

$$MCD(124,14)=2$$

```
10 REM MAXIMO COMUN DIVISOR DE A Y B
20 INPUT "A=";A
30 INPUT "B=";B
40 R=A-B*INT(A/B)
50 IF R=0 THEN 90
60 A=B
70 B=R
80 GOTO 40
90 PRINT "MCD=";B
100 END
```

Otro programa puede ser el siguiente:

```
10 REM MAXIMO COMUN DIVISOR DE A Y B
20 INPUT "A, B";A,B
30 R=A-B*INT(A/B)
40 IF R<>0 THEN A=B : B=R : GOTO 30
50 PRINT "M.C.D. =";B
60 END
```


89. TABLA DE VALORES DE UNA FUNCION (6.43)

Hacer un programa que dé una tabla de valores de cualquier función entre dos números y para un incremento dado.

Mediante la instrucción DEF FNY(X) de la línea 100 definimos la función que deseamos. En nuestro programa hemos considerado la función DEF FNY(X)=X↑2, que puede ser sustituida por cualquier otra.

El programa se puede considerar dividido en tres bloques:

1.° *Presentación.* Líneas 30-90 en las que se dan las instrucciones para la introducción de la función y para la continuación del programa.

2.° *Entrada de datos.* Líneas 120 y 130; las variables A y B recogen los extremos del intervalo y la variable C el incremento.

3.° *Construcción de la tabla.* Líneas 150-200. Mediante un ciclo FOR-NEXT calculamos los valores de la función en los distintos puntos del intervalo [A,B]. Para que la tabla tenga una mejor presentación redondeamos los valores de la variable X y los de la función FNY(X), a cuatro decimales. La función TAB nos permite situar adecuadamente la tabla en la pantalla. Nuestra presentación está pensada para una pantalla de 40 columnas.

```
10 REM TABLA DE VALORES DE UNA FUNCION
20 (BORRAR LA PANTALLA)
30 PRINT "TABLA DE VALORES DE UNA FUNCION"
40 PRINT "==========================="
50 PRINT : PRINT "SI NO HAS DEFINIDO AUN LA FUNCION
60 PRINT "HAZLO AHORA ESCRIBIENDO PREVIAMENTE"
70 PRINT "      100DEF FNY(X)=....."
80 PRINT : PRINT "SI YA LA HAS DEFINIDO TECLEA RUN 100"
90 STOP
100 DEF FNY(X)=X↑2
110 (BORRAR LA PANTALLA)
120 INPUT "EXTREMOS DEL INTERVALO";A,B
130 PRINT : INPUT "INCREMENTO";C
140 (BORRAR LA PANTALLA)
150 PRINT "VALORES DE X";TAB(18);"VALORES DE LA FUNCION"
160 PRINT "==========";TAB(18);"==================="
170 FOR X=A TO B STEP C
180 Y1=INT(FNY(X)*1E4+.5)/1E4
190 PRINT TAB(4);INT(X*1E4+.5)/1E4;TAB(25);Y1
200 NEXT X
210 END
```

90. CUADRADO PERFECTO (6.9)

Sabiendo que la solución es única, hallar un número de cuatro cifras, de la forma aabb, que sea cuadrado perfecto.

Todo número que sea cuadrado perfecto ha de acabar en 0, 1, 4, 5, 6 ó 9, pero gracias a la rapidez de cálculo del ordenador podemos prescindir de esta restricción.

Tendremos que comprobar números de cuatro cifras, desde 1100 hasta 9999, que cumplan la condición indicada.

Un número es cuadrado perfecto si verifica:

$$INT(N\uparrow.5) = N\uparrow.5$$

o bien

$$(INT(N\uparrow.5))\uparrow 2 = N$$

Estas igualdades pueden no ser útiles porque el ordenador no siempre utiliza valores exactos, sino que a veces utiliza valores aproximados; por ejemplo:

$$\sqrt{6400} = 80.000001$$

y de igual modo:

$$80^2 = 6400.000001$$

Una posible solución es redondear las unidades de la raíz, que efectuamos en la línea 140.

```
10 REM CUADRADO PERFECTO
20 (BORRAR LA PANTALLA)
30 PRINT "HAY QUE CALCULAR UN NUMERO"
40 PRINT "DE LA FORMA AABB QUE SEA"
50 PRINT "CUADRADO PERFECTO"
60 PRINT "SI DESEA QUE COMIENCE PULSE"
70 PRINT "UNA TECLA"
80 GET A$
90 IF A$="" THEN 80
100 (BORRAR LA PANTALLA)
110 FOR A=1 TO 9
120 FOR B=0 TO 9
130 N=1000*A+100*A+10*B+B
140 R=INT(N↑0.5+0.5)
150 IF R*R=N THEN 180
160 NEXT B
170 NEXT A
180 FOR I=1 TO 10 : PRINT : NEXT I
190 PRINT"EL NUMERO PEDIDO ES";N
200 END
```

El problema admite otro punto de vista. Los números de cuatro cifras que sean cuadrados perfectos tienen raíces comprendidas entre 32 y 99. Podemos elevar al cuadrado esos números N y comprobar si son de la forma aabb. Para realizar tal comprobación transformamos cada número en la cadena N$=STR$(N). En algunos ordenadores la cadena N$ aparece con un carácter más que el número debido al signo. Por ejemplo, si N=2132, entonces N$="+2132" tiene 5 caracteres.

Mediante la función MID$ comprobamos si los caracteres segundo y tercero son iguales, así como los de lugar cuarto y quinto.

```
10 REM CUADRADO PERFECTO (2)
20 FOR R=32 TO 99
30 N=R*R
40 N$=STR$(N)
50 IF (MID$(N$,2,1)=MID$(N$,3,1)) AND (MID$(N$,4,1)=MID$(N$,5,1)) THEN 70
60 NEXT R
70 PRINT"EL NUMERO ES";N
80 END
```

91. CAPICUA (6.11)

Sabiendo que la respuesta es única, hallar un número capicúa de seis cifras que sea cuadrado perfecto.

Como el ordenador opera muy deprisa le haremos estudiar todos los capicúas desde 100001 hasta 999999 y comprobar si son cuadrados perfectos, y en caso afirmativo, le pediremos que lo imprima y que termine el programa.

Las primeras líneas son tres bucles encajados de manera que N va tomando los valores de todos los capicúas.

Tenemos la misma dificultad que en el problema anterior: no se puede poner como condición de la sentencia IF-THEN la siguiente:

$$\text{INT}(N\uparrow.5) = N\uparrow.5 \qquad (1)$$

ni tampoco

$$(\text{INT}(N\uparrow.5))\uparrow 2 = N \qquad (2)$$

por los mismos motivos que antes. Lo resolveremos también de la misma manera: utilizaremos una condición semejante a la (2) pero redondeando la $\sqrt{N}$ a las unidades, sin decimales, en la línea 70.

```
10 REM CAPICUA
20 FOR A=1 TO 9
30 FOR B=0 TO 9
40 FOR C=0 TO 9
50 N=100001*A+10010*B+1100*C
60 R=SQR(N)
70 IF INT(R+.5)*INT(R+.5)=N THEN 110
80 NEXT C
90 NEXT B
100 NEXT A
110 PRINT "EL NUMERO ES";N
120 END
```

El programa admite otro enfoque. Los números de seis cifras tienen raíces cuadradas comprendidas entre 315 y 1000. Podemos elevar al cuadrado estos valores y comprobar cada vez si el cuadrado es capicúa o no.

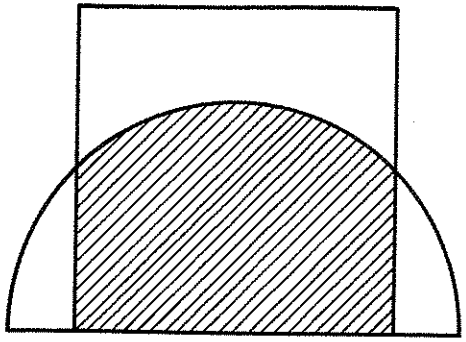
Es lo que hemos hecho en la segunda versión del programa. El análisis sobre si el cuadrado es capicúa o no se hace en las líneas 50 y 60.

Cronometra qué programa es más eficiente. A nosotros el primer programa nos ha dado un tiempo de ejecución de 53,8 segundos, mientras que el segundo sólo 18,8 segundos: la tercera parte.

```
10 REM CAPICUA 2
20 FOR R=315 TO 1000
30 N=R*R
40 A$=STR$(N)
50 B$=MID$(A$,4,1)+MID$(A$,3,1)+MID$(A$,2,1)
60 IF B$=RIGHT$(A$,3) THEN 80
70 NEXT R
80 PRINT "EL NUMERO ES";N
90 END
```

92. LA VACA Y EL PRADO (6.31)

Un campesino tiene un prado cuadrado de lado L y una vaca que pasta en él. Ata la vaca con una cuerda al centro de uno de los lados. Quiere saber la longitud R que ha de dar a la cuerda, para que la vaca alcance exactamente la mitad de la superficie del prado y no se indigeste.



Haz que R vaya creciendo desde R = L/2 hasta que la superficie de pasto sea $L^2/2$, con incrementos de R iguales a L/1000.

La fórmula de la superficie de pasto es:

S = 3.14159*R↑2/2 + L/2*(R↑2 − L↑2/4)↑.5 − ATN((R↑2 − L↑2/4)↑.5*2/L)*R↑2

Como el ordenador calcula los valores de modo aproximado, hay que tomar diversas precauciones, para que el programa sea válido.

En primer lugar, en la función de la superficie S, aparece una raíz cuadrada $\sqrt{R^2 - L^2/4}$. El primer valor de R es L/2 y el radicando es cero. No tendría que haber dificultades, pero de hecho sí pueden existir, porque el ordenador quizá obtenga para $R^2 - L^2/4$ un valor muy pequeño, pero negativo. Evitamos esto, tomando el valor absoluto del radicando.

Por otra parte, no podemos imponer la condición de que la superficie S, para un cierto valor de R, sea $L^2/2$, porque en general no serán números exactamente iguales. Tampoco es correcto poner

$$ABS(S - L^2/2) < 0.001$$

porque a priori no sabemos si la diferencia puede ser de ese orden de magnitud.

Por ejemplo: Si L = 35, $L^2/2 = 612.5$

Si R = 20.3699997 ..... S = 611.269678
Si R = 20.4049997 ..... S = 612.742523

En ambos casos $L^2/2$ difiere de S en una cantidad mayor que una milésima. Salvamos estos inconvenientes calculando el valor de R que hace mínimo ABS($L^2/2 - S$).

Para ello tomamos como valor mínimo inicial, fuera del ciclo FOR-NEXT, un número que sea mayor que todas las diferencias. Uno posible es L↑2, que guardamos en la variable MIN. Cuando se ha recorrido el ciclo, en MIN se encuentra el mínimo y en X el valor de R que hace D mínimo.

```
10 REM LA VACA Y EL PRADO
20 (BORRAR LA PANTALLA)
30 PI=3.1416
40 INPUT "LADO DEL CUADRADO=";L
50 DEF FNY(R)=PI*R↑2/2+L/2*(ABS(R↑2-L↑2/4))↑.5-ATN((ABS(R↑2-L↑2/4))↑.5*2/L)*
R↑2
60 MIN=L↑2
70 FOR R=L/2 TO L STEP L/1000
80 S=FNY(R)
90 D=ABS(L↑2/2-S)
100 IF MIN>D THEN MIN=D : X=R
110 NEXT R
120 PRINT : PRINT "EL RADIO MIDE";X
130 END
```

93. AREA DE UN POLIGONO

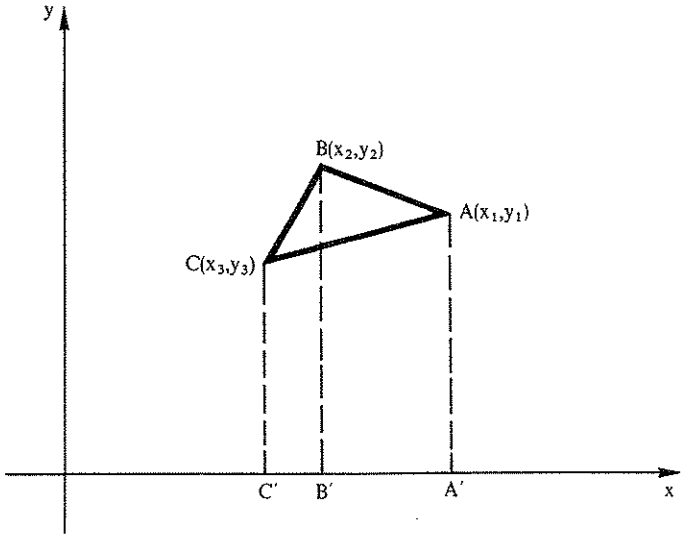
Realiza un programa que calcule el área de un polígono, por el método de los trapecios, conociendo las coordenadas de los vértices.

La fórmula que nos da el área del polígono por el método de los trapecios es:

$$A = 1/2[(x_1 - x_2)(y_1 + y_2) + (x_2 - x_3)(y_2 + y_3) + \cdots + (x_n - x_1)(y_n + y_1)],$$

donde $(x_1,y_1), (x_2,y_2), \cdots, (x_n,y_n)$ son las coordenadas de los vértices del polígono.

En la figura, para tres puntos, deducimos la fórmula que hemos puesto anteriormente.



Area del trapecio CC'B'B:

$$A_1 = 1/2(y_2 + y_3) \cdot (x_2 - x_3)$$

Area del trapecio BB'A'A:

$$A_2 = 1/2(y_1 + y_2) \cdot (x_1 - x_2)$$

Area del trapecio CC'A'A:

$$A_3 = 1/2(y_1 + y_3) \cdot (x_1 - x_3)$$

Area del triángulo ABC:

$$A_T = A_1 + A_2 - A_3 = 1/2(y_2 + y_3) \cdot (x_2 - x_3) + 1/2(y_1 + y_2) \cdot (x_1 - x_2) - \\ - 1/2(y_1 + y_3) \cdot (x_1 - x_3)$$

$$A_T = 1/2[(x_1 - x_2) \cdot (y_1 + y_2) + (x_2 - x_3) \cdot (y_2 + y_3) + (x_3 - x_1) \cdot (y_3 + y_1)]$$

Para la realización del programa hemos supuesto que el polígono no puede tener más de 20 vértices.

Las coordenadas se introducen mediante un INPUT y se utilizan las variables con índice X(I) e Y(I).

Para el cálculo del área A, se empieza tomando el valor cero y luego se acumulan los factores hasta llegar al penúltimo, ya que el último se suma fuera del bucle.

Como el área es un número positivo y nos puede salir negativo (áreas orientadas), hemos recurrido a la función valor absoluto (ABS).

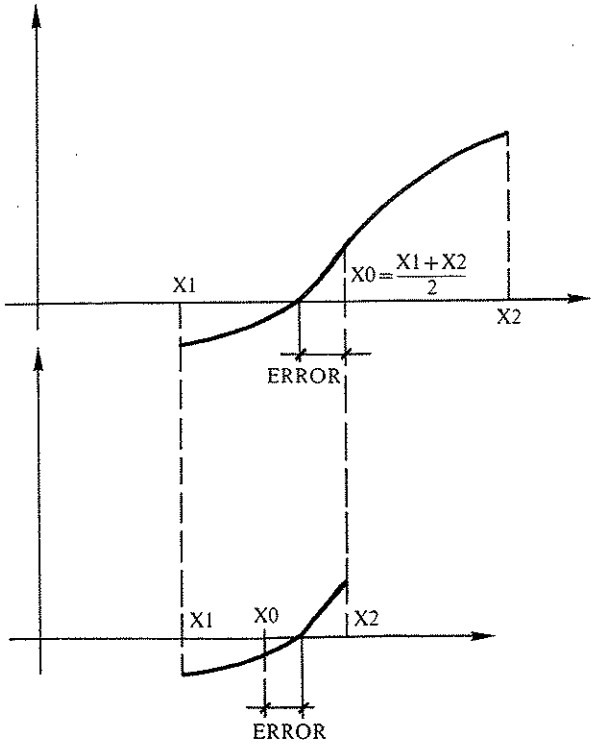
```
10 REM AREA DE UN POLIGONO
20 DIM X(20),Y(20)
30 PRINT "CUANTOS VERTICES (20 COMO MAXIMO)";
40 INPUT N
50 PRINT "INTRODUCE LAS COORDENADAS DE LOS VERTICES"
60 FOR I=1 TO N
70 PRINT "VERTICE";I;
80 INPUT X(I),Y(I)
90 NEXT I
100 REM CALCULO DEL AREA MEDIANTE LA FORMULA
110 A=0
120 FOR I=1 TO N-1
130 A=A+(X(I)-X(I+1))*(Y(I)+Y(I+1))
140 NEXT I
150 A=A+(X(N)-X(1))*(Y(N)+Y(1))
160 PRINT "EL AREA ES";ABS(A)/2
170 END
```

94. METODO DE BIPARTICION (6.45)

Si una función continua en un intervalo toma valores de distinto signo en sus extremos, la función se anula en algún punto intermedio (Teorema de Bolzano). Vamos a hallar por bipartición del intervalo ese punto (al menos uno) en que se anula.

En general lo que se calcula es un valor aproximado, con un error menor que un número E fijado previamente.

Supuesto que se verifica la condición expuesta en el intervalo [X1,X2], el procedimiento es sencillo.



En la figura se representa una función Y=F(X), siendo F(X1) negativo y F(X2) positivo. Tomamos el punto medio del intervalo [X1,X2] y vemos el

valor de la función en ese punto F(X0). Si es cero, hemos encontrado una raíz y el problema termina. Si F(X0) no es cero, entonces, de entre los dos segmentos en que se ha dividido el intervalo, elegimos aquél en que la función toma valores de signo contrario en sus extremos. Y así sucesivamente, hasta que la longitud del intervalo sea menor que 2E. Tomando ahora, X0 como valor aproximado de la raíz, cometemos un error menor que E.

Nuestro programa está planteado para que introduzcamos un intervalo [X1,X2] sin tener en cuenta si en su extremos la función toma valores de signo opuesto. Ese trabajo se lo dejamos al ordenador que lo realiza mediante la subrutina 300.

El programa está dividido en las siguientes partes:

1.ª *Presentación, instrucciones y entrada de datos* (líneas 10-110).

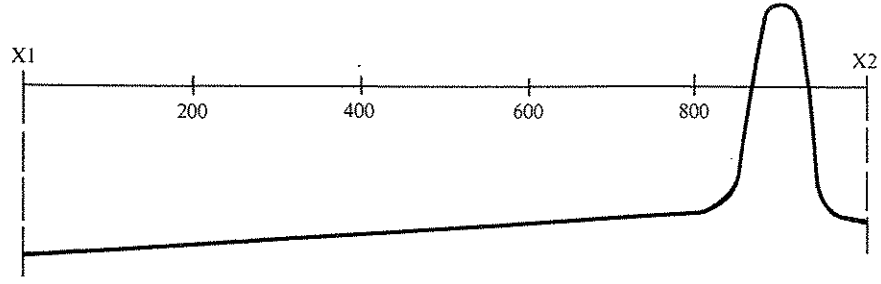
2.ª *Se comprueba si alguno de los extremos es raíz de la ecuación.* Líneas 150-160.

En caso de que ninguno de los extremos sea raíz:

3.ª *Estudio del cambio de signo* (subrutina 300).

Si los extremos cumplen la condición del Teorema de Bolzano, nos salimos de la subrutina y el programa continúa. En caso contrario hay que buscar un punto C en el interior del intervalo [X1,X2] tal que el intervalo [X1,C] sea válido.

Para ello se podría dividir [X1,X2] en un número elevado de partes, 100 ó 1000 y ver los valores de la función en los puntos intermedios. Pero este procedimiento es menos rentable que obtener números aleatorios comprendidos entre X1 y X2. En efecto, en la figura, la función cambia de signo entre las divisiones 850-950. Habría que comprobar como mínimo los 850 primeros puntos para observar un cambio de signo. En cambio la probabilidad de obtener un número aleatorio en ese intervalo es 1/10.



Si hemos obtenido el intervalo adecuado:

4.ª *Se busca la raíz en ese intervalo* (líneas 180-250).

```
10 REM METODO DE BISECCION
20 (BORRAR LA PANTALLA)
30 FOR I=1 TO 6 : PRINT : NEXT I
40 PRINT "METODO DE BISECCION"
50 PRINT : PRINT : PRINT "          PULSE LIST 130"
60 PRINT : PRINT : PRINT "          PONGA LA FUNCION"
70 PRINT : PRINT : PRINT "          PULSE RUN 90"
80 STOP
90 (BORRAR LA PANTALLA)
100 INPUT "EXTREMOS DEL INTERVALO";X1,X2
110 PRINT : INPUT "ERROR MENOR QUE";E
120 FOR I=1 TO 10 : PRINT : NEXT I
130 DEF FNY(X)=...
140 Y1=FNY(X1) : Y2=FNY(X2)
150 IF Y1=0 THEN PRINT "HAY UNA RAIZ EN";X1 : GOTO 250
160 IF Y2=0 THEN PRINT "HAY UNA RAIZ EN";X2 : GOTO 250
170 GOSUB 300
180 X0=(X1+X2)/2
190 Y0=FNY(X0)
200 IF Y0=0 THEN PRINT "HAY UNA RAIZ EN";X0 : GOTO 250
210 IF ABS(X2-X1)<2*E THEN PRINT X0;"ES RAIZ CON ERROR MENOR QUE";E
: GOTO 250
220 IF SGN(FNY(X1))=SGN(Y0) THEN 240
230 X2=X0 : GOTO 180
240 X1=X0 : GOTO 180
250 END
300 REM ESTUDIO DEL CAMBIO DE SIGNO
310 IF SGN(Y1)<>SGN(Y2) THEN 400
320 L=X2-X1
330 FOR I=X1 TO 1000
340 C=X1+L*RND
350 IF SGN(Y1)<>SGN(FNY(C)) THEN 390
360 NEXT I
370 PRINT "NO SE ENCUENTRA CAMBIO DE SIGNO"
380 PRINT "QUIZA NO HAYA RAICES" : GOTO 250
390 X2=C
400 RETURN
```

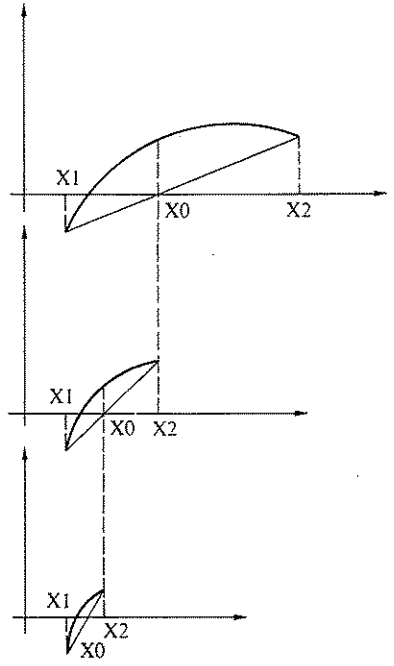

95. METODO DE LA SECANTE

En el programa anterior hemos enunciado el Teorema de Bolzano.

Localizado un intervalo de este tipo, podemos aproximar una raíz de la ecuación $F(x)=0$, por varios métodos; uno de ellos es el de la secante, que pasamos a describir:

Dada la gráfica de la función y el intervalo, se unen mediante un segmento los extremos del arco (una secante a la curva) y se toma como valor aproximado de la raíz el punto de corte de la secante con el eje de las X, según la fórmula:

$$X0 = \frac{X1 \cdot F(X2) - X2 \cdot F(X1)}{F(X2) - F(X1)}$$



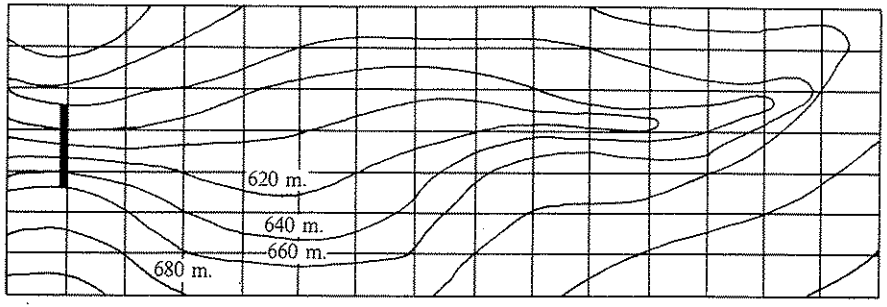
Salvada la excepción de que X0 sea la raíz exacta, la raíz quedará a izquierda o derecha de X0, en nuestro gráfico a la izquierda. Prescindimos entonces del intervalo (X0,X2], llamamos X2 al punto X0 y volvemos a trazar una secante. Se obtienen sucesivamente valores de X0 más próximos a la raíz.

El programa tiene una introducción para que el usuario defina la función en la línea 100. Después se realiza el proceso descrito. El organigrama sería semejante al del método de bipartición del libro BASIC BASICO. El núcleo del programa es un bucle en el que por N veces (el número de iteraciones) se halla el punto X0, intersección de la secante con el eje de las X, y se toma como nuevo intervalo el que contenga la raíz.

```
10 REM RAICES REALES DE UNA ECUACION
20 REM METODO DE LA SECANTE
30 (BORRAR LA PANTALLA)
40 PRINT : PRINT "TECLEA LA FUNCION DE X PRECEDIDA DE"
50 PRINT : PRINT "       100 DEF FNY (X)=..."
60 PRINT : PRINT "DESPUES ESCRIBE RUN 100"
70 STOP
110 (BORRAR LA PANTALLA)
120 PRINT
130 INPUT "INTERVALO (X1<X2)";X1,X2
140 IF SGN(FNY(X1))<> SGN(FNY(X2)) THEN 170
150 PRINT : PRINT "NO ES SEGURO QUE LA FUNCION TENGA RAICES"
160 PRINT : PRINT "EN ESE INTERVALO" : PRINT : GOTO 120
170 PRINT
180 INPUT "NUMERO DE ITERACIONES";N
190 PRINT : PRINT "APROXIMACION";TAB(16);"VALORES DE LA FUNCION"
200 PRINT "==========================" : PRINT
210 FOR I=1 TO N
220 X0=(X1*FNY(X2)-X2*FNY(X1))/(FNY(X2)-FNY(X1))
230 IF FNY(X0)=0 THEN I=N : GOTO 260
240 IF SGN(FNY(X0))=SGN(FNY(X1)) THEN X1=X0 : GOTO 260
250 X2=X0
260 PRINT X0;TAB(16);FNY(X0)
270 NEXT I
280 PRINT : PRINT "UNA RAIZ APROXIMADA ES ";X0
290 END
```


96. PANTANO

Se ha construido una presa en la boca de una cerrada para formar un embalse. Las curvas de nivel de la zona que cubrirán las aguas son las de la figura. El borde superior de la presa empalma con la curva de cota 660 m. (el agua podrá llegar hasta esa altura).



Con el fin de conocer el volumen de agua embalsada en cada momento, se ha pintado en la pared interior de la presa una escala numerada que indica la distancia desde el borde superior hasta la superficie del agua. Hemos de preparar un programa que nos permita conocer el volumen de agua embalsada a partir de la distancia mencionada.

Dibujamos una cuadrícula sobre el mapa de la figura, de forma que una de las líneas verticales de la cuadrícula coincida con la presa, que suponemos recta. Necesitamos conocer no sólo el dato que puede variar, la altura del agua, sino también otro que se mantendrá siempre constante: la cota media de cada uno de los cuadrados. Para los que están detrás de la presa y que tienen cota media inferior al nivel del agua en ese momento, la diferencia entre estas dos alturas, cota media y altura del agua, multiplicada por la superficie del rectángulo nos da el volumen de agua sobre él.

En la siguiente tabla de valores representamos la cuadrícula y la cota media de cada uno de los rectángulos.

715	692	681	673	668	666	664	665	672	675	681	683	679	682
674	681	663	651	643	636	635	644	647	650	659	668	674	685
657	650	637	629	622	609	618	623	627	629	631	636	657	686
630	622	618	612	608	617	631	642	637	652	659	679	701	698
652	639	625	620	623	632	649	667	677	681	686	692	704	711
679	658	641	632	637	645	663	679	684	689	697	709	714	716
692	681	674	651	654	662	671	686	692	712	714	715	717	721

Hemos utilizado el algoritmo que sigue para obtener el programa:

Entrar la distancia del agua al borde, DB
Inicializar la cota de la superficie del agua y el volumen, CS y V
Para todas las filas de la cuadrícula
  Para todas las columnas
    Leer la cota del rectángulo, C
    Si la cota es inferior a la altura del agua, C < CS
    entonces aumentar V en la diferencia de estas alturas
    Fin de Si
  Siguiente columna
Siguiente fila
Multiplicar V por la superficie del rectángulo
Escribir el volumen
Fin

Suponemos que la superficie de los rectángulos es de 10.000 m^2 (100 m. de largo por 100 de ancho). Depositamos ordenadamente los valores de la tabla en líneas DATA con lo cual la realización y ejecución del programa es inmediata.

```
10 REM PANTANO
20 DATA 715,692,681,673,668,666,664,665,672,675,681,683,679,682
30 DATA 674,681,663,651,643,636,635,644,647,650,659,668,674,685
40 DATA 657,650,637,629,622,609,618,623,627,629,631,636,657,686
50 DATA 630,622,618,612,608,617,631,642,637,652,659,679,701,698
60 DATA 652,639,625,620,623,632,649,667,677,681,686,692,704,711
70 DATA 679,658,641,632,637,645,663,679,684,689,697,709,714,716
80 DATA 692,681,674,651,654,662,671,686,692,712,714,715,717,721
100 INPUT "DISTANCIA DEL AGUA AL BORDE ";DB
```



```
110 CS=660-DB : V=0
120 FOR I=1 TO 7
130 FOR J=1 TO 14
140 READ C : IF C<CS THEN V=V+(CS-C)
150 NEXT J
160 NEXT I
170 V=V*100*100
180 PRINT "EL VOLUMEN EMBALSADO ES ";V;" M3"
190 END
```

97. LAGO

Deseamos tener una idea aproximada del relieve del fondo de un lago (véase figura 1). Haz un programa que nos dé una representación tipo mapa (véase figura 2), con su profundidad en cada sitio.

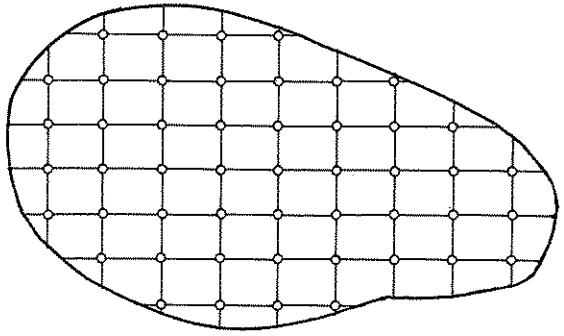


Fig. 1

Para lograrlo procedemos en forma parecida a como hicimos en el ejemplo anterior: una embarcación sigue las líneas horizontales y efectúa medidas (mediante sónar) de la profundidad del fondo en los puntos señalados. Introducimos los resultados de estas medidas en líneas DATA, precedidas por otro valor que indica la posición en que se realizó la primera medida de cada serie, desde el punto que está más al Oeste de todos. Cada tanda de medidas va seguida de un -1, que permite distinguirla de la siguiente.

```
10 REM LAGO
20 (BORRAR LA PANTALLA)
30 FOR I=1 TO 7
40 READ PF,P : PRINT TAB(PF-LEN(STR$(P)));P;":";
50 READ P : IF P<>-1 THEN PF=PF+3 : PRINT TAB(PF-LEN(STR$(P)));P;":":
GOTO 50
60 PRINT : PRINT
70 NEXT I
100 DATA 7,2,7,9,3,-1
110 DATA 4,2,6,12,20,12,8,3,-1
```



```
120 DATA 4,3,14,20,23,19,12,7,2,-1
130 DATA 4,4,17,32,35,29,26,15,9,1,-1
140 DATA 4,3,10,18,26,20,16,11,7,3,-1
150 DATA 7,4,8,11,10,8,6,5,2,-1
160 DATA 10,2,4,3,2,-1
200 END
```

Ejecutando el programa tenemos la siguiente imagen:

```
   2: 7: 9: 3:
 2: 6:12:20:12: 8: 5:
 3:14:20:23:19:12: 7: 2:
 4:17:32:35:29:26:15: 9: 1:
 3:10:18:26:20:16:11: 7: 3:
   4: 6:11:10: 6: 5: 5: 2:
      2: 4: 3: 2:
```

Fig. 2

Este programa funciona correctamente en ordenadores que no reservan lugar en la pantalla para el signo de los números positivos; pero si tu ordenador deja un espacio libre delante de los números y posiblemente otro detrás, has de efectuar las rectificaciones oportunas en las líneas 40 y 50.

98. DIA DE LA SEMANA

Realiza un programa que nos dé el día de la semana en que cae una determinada fecha (día, mes y año).

Hay que tener en cuenta si el año es juliano o gregoriano, y también si los meses son los de enero y febrero o los restantes del año, para aplicar la fórmula:

$$N = D + 2*M + INT(3*(M+1)/5) + A + INT(A/4) - G.$$

Para las fechas comprendidas desde el inicio de la era cristiana hasta el 4 de octubre de 1582, que corresponden al calendario juliano, G es cero. Para las posteriores al 15 de octubre de 1582 (calendario gregoriano) G es INT(A/100)−INT(A/400)−2. No se admiten fechas intermedias entre el 4 y el 15 de octubre de ese año.

En caso de un día de enero o febrero, M será M+12 y A tiene que ser A−1.

La variable N son los días transcurridos desde el principio de la era cristiana hasta la fecha considerada, y la fórmula:

$$R = N - 7*INT(N/7) + 1$$

nos dice que si R=1, el día es sábado; si R=2, domingo, etc.

Como las fórmulas anteriormente presentadas son complicadas de explicar, a continuación se incluye una breve bibliografía, que se puede consultar en relación con este tema.

1. Santiago Thio de Pol.: *Primos o algunos dislates sobre números.* Edit. Alhambra.
2. *Diccionario Enciclopédico Salvat Universal.* Tomo V.
3. *Enciclopedia Salvat de la Ciencia y de la Tecnología.* Tomo II.
4. B. A. Vorontsov-Veliaminov.: *Problemas y ejercicios prácticos de astronomía.* Edit. Mir.


```
10 REM DIA DE LA SEMANA
20 PRINT "DIA,MES (EN NUMERO),AÑO (LAS 4 CIFRAS)"
30 INPUT D,M,A
40 REM AÑO JULIANO (HASTA 4-10-1582) O GREGORIANO (DESDE 15-10-1582)
50 IF A=1582 AND M=10 AND D<5 THEN 80
60 IF A=1582 AND(M>10 OR(M=10 AND D>14)) THEN 100
70 IF A>1582 THEN 100
80 G=0
90 GOTO 110
100 G=INT(A/100)-INT(A/400)-2
110 REM SI SON LOS MESES DE ENERO O FEBRERO
120 IF M>2 THEN 150
130 M=M+12
140 A=A-1
150 REM LOS RESTANTES MESES
160 N=D+2*M+INT(3*(M+1)/5)+A+INT(A/4)-G
170 R=N-7*INT(N/7)+1
180 ON R GOTO 200,250,300,350,400,450,500
200 PRINT "SABADO" : GOTO 510
250 PRINT "DOMINGO" : GOTO 510
300 PRINT "LUNES" : GOTO 510
350 PRINT "MARTES" : GOTO 510
400 PRINT "MIERCOLES" : GOTO 510
450 PRINT "JUEVES" : GOTO 510
500 PRINT "VIERNES"
510 END
```

99. VIAJE RELATIVISTA

Nos preparamos para realizar un viaje a velocidades cercanas a la de la luz (300.000 km/s). Según la teoría de la relatividad, el tiempo que para un observador inmóvil dura t para otro que se desplaza a velocidad v, respecto del anterior, se contrae de acuerdo con la expresión $t' = t\sqrt{1-(v/c)^2}$, *siendo c la velocidad de la luz.*

Indicamos al ordenador la distancia que vamos a viajar y deseamos ver en pantalla una tabla en que figuren los días (en tiempo local de la nave) en que se transformará un año terrestre y el tiempo que tardaremos en realizar el viaje (en años de la nave). Y esto para dos velocidades dadas y otras nueve intermedias, todas equidistantes entre sí.

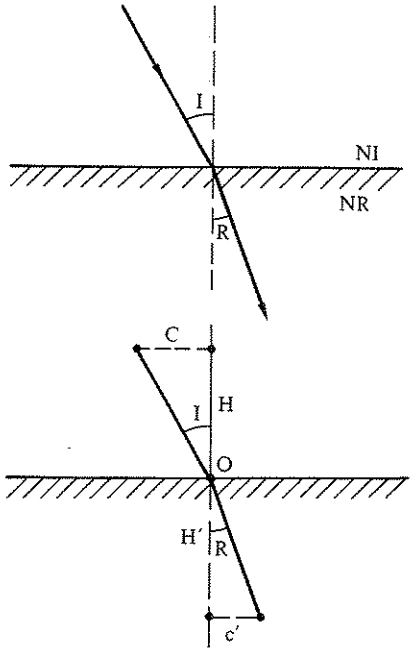
El problema se puede resolver de la siguiente forma:

Introducir distancia y velocidades, D, V1, V2
Calcular el incremento de la velocidad, IV
Para velocidades de V1 a V2, con incremento IV
  Calcular días de la nave equivalentes a un año terrestre
  Calcular tiempo de viaje, TV
  Exhibir V, T, TV
Siguiente velocidad
FIN

```
10 REM VIAJE RELATIVISTA
20 C=300000
30 INPUT "DISTANCIA A QUE HEMOS DE VIAJAR (EN AÑOS LUZ)";D
40 PRINT "VALORES EXTREMOS DE LA VELOCIDAD" : INPUT V1,V2
50 IV=(V2-V1)/10
100 (BORRAR LA PANTALLA)
110 PRINT "DISTANCIA";D;"AÑOS LUZ" : PRINT
120 PRINT "VELOCIDAD";TAB(12);"DIAS/AÑO";TAB(23);"T.VIAJE"
130 FOR N=0 TO 31 : PRINT "_"; : NEXT N
140 PRINT
200 FOR V=V1 TO V2 STEP IV
210 T=365*SQR(1-(V/C)↑2)
220 R$=MID$(STR$(T-INT(T)),2,4)
230 T=INT(T)+VAL(R$)
240 TV=D*SQR(1-(V/C)↑2)
250 R$=MID$(STR$(TV-INT(TV)),2,4)
260 TV=INT(TV)+VAL(R$)
270 PRINT V;TAB(13);T;TAB(22);TV
280 NEXT V
290 END
```


100. REFRACCION

Representar gráficamente la trayectoria de un rayo luminoso refractado, si conocemos el ángulo de incidencia y los índices de refracción.



La figura ilustra el camino seguido por el rayo incidente y el refractado para ángulos I y R, respectivamente. Para representar este camino utilizando la función TAB hemos de considerar, en cada momento, a qué altura (H) sobre la superficie nos encontramos y situarnos a la distancia conveniente de la vertical de O. Conocido el ángulo I, resulta:

$$\frac{C}{H} = TAN(I), \quad \text{luego} \quad C = H*TAN(I)$$

Como NI*SIN(I) = NR*SIN(R), para el segundo medio resulta:

$$SIN(R) = \frac{NI*SIN(I)}{NR}$$

y, por tanto,

$$R = ARC\ SIN(NI*SIN(I)/NR)$$

Para obtener la separación horizontal de la vertical trazada por O, hemos de considerar:

$$TAN(R) = \frac{C'}{H'}$$

luego

$$C' = H'*TAN(R) = H'*TAN(ARC\ SIN(NI*SIN(I)/NR))$$

Si tu ordenador no dispone de la función ARC SIN, puedes introducirla tú, ya que:

$$\text{arc sen } x = \text{arc tg } \frac{x}{\sqrt{1-x^2}}$$

Así pues, siendo, las variables:

I = ángulo de incidencia.
R = ángulo de refracción.
NI = índice de refracción en el primer medio.
NR = índice de refracción en el segundo medio.
H = distancia a O en vertical.
C = distancia a O en horizontal.

el programa puede adoptar la forma.

```
10 REM REFRACCION
20 INPUT "INTRODUZCA LOS INDICES DE REFRACCION";NI,NR
30 INPUT "ANGULO FORMADO POR EL RAYO INCIDENTE Y LA VERTICAL";I
:I=I*3.1416/180
40 (BORRAR LA PANTALLA)
50 FOR H=10 TO 1 STEP -1 : PRINT TAB(20-H*TAN(I));"*" : NEXT H
60 FOR C=0 TO 19 : PRINT TAB(C);"_"; : NEXT C : PRINT "*";
70 FOR C=21 TO 38 : PRINT TAB(C);"_"; : NEXT C : PRINT
80 A=NI*SIN(I)/NR
90 FOR H=1 TO 10 : R=ARC(SIN(A)) : PRINT TAB(20+H*TAN(R));"*" : NEXT H
100 END
```

Para una representación correcta a cargo de las líneas 50 y 90, se requiere que tanto el ángulo de incidencia como el de refracción no excedan los 45°.

101. TABLA Y SU TRASPUESTA (7.8)

Dada una tabla de cinco filas y tres columnas, escribir dicha tabla y su traspuesta.

Inicialmente los datos están depositados en líneas DATA de donde los lee el programa utilizando la instrucción READ y la variable A(I,J).

La lectura se hace por filas y posteriormente se escribe la tabla inicial. Para trasponer, lo que se hace es cambiar filas por columnas y éstas por aquéllas, de modo que se fija cada columna y se imprime como fila.

Presentación en pantalla:

```
TABLA INICIAL:
----- ---- ---
 1  2  3
 4  5  6
 7  8  9
 3  1  0
 1  2  1

TABLA TRASPUESTA:
-----  -- -- -- --
 1  4  7  0  1
 2  5  8  1  2
 3  6  9  0  1
```

```
10 REM TABLA Y SU TRASPUESTA
20 DIM A(5,3)
30 REM LECTURA DE LA TABLA
40 FOR I=1 TO 5
50 FOR J=1 TO 3
60 READ A(I,J)
70 NEXT J
80 NEXT I
90 REM IMPRESION DE LA TABLA
100 PRINT "TABLA INICIAL:"
110 PRINT "----- ---------"
120 FOR I=1 TO 5
130 FOR J=1 TO 3
140 PRINT A(I,J);
150 NEXT J
```

```
160 PRINT
170 NEXT I
180 PRINT
190 REM IMPRESION DE LA TRASPUESTA
200 PRINT "TABLA TRASPUESTA:"
210 PRINT "- - - - - - - - - - - - - - -"
220 FOR J=1 TO 3
230 FOR I=1 TO 5
240 PRINT A(I,J);
250 NEXT I
260 PRINT
270 NEXT J
280 REM ELEMENTOS DE LA TABLA
290 DATA 1,2,3,4,5,6,7,8,9
300 DATA 0,1,0,1,2,1
310 END
```


102. MULTIPLICACION DE DOS TABLAS

Prepara un programa que nos permita multiplicar dos tablas, una de 3 filas y 4 columnas, y la otra de 4 filas y 2 columnas.

En la variable con dos índices A(I,J) introducimos los elementos de la primera tabla, en B(I,J) los de la segunda y en C(I,J) los de la tercera.

Para multiplicarlas, hay que tener en cuenta que el número de columnas de la primera tabla debe ser igual al número de filas de la segunda, lo que se cumple en este ejercicio.

Para entender cómo se realiza esta multiplicación ponemos un ejemplo: fijamos la segunda fila de la primera tabla y la primera columna de la segunda, con lo que resulta:

$$C(2,1) = A(2,1) \cdot B(1,1) + A(2,2) \cdot B(2,1) + A(2,3) \cdot B(3,1) + A(2,4) \cdot B(4,1)$$

Presentación en Pantalla

```
ELEMENTOS DE LA PRIMERA TABLA
-1  0  5  2
 6  5 -2  3
 0  1  2  3

ELEMENTOS DE LA SEGUNDA TABLA
 0  3
 1  4
-5  2
 1  6

LA TABLA PRODUCTO ES
-20  31
 18  52
-6  26
```

```
10 REM MULTIPLICACION DE DOS TABLAS
20 DIM A(3,4),B(4,2),C(3,2)
30 PRINT "ELEMENTOS DE LA PRIMERA TABLA"
40 FOR I=1 TO 3
50 FOR J=1 TO 4
```

```
60 READ A(I,J)
70 PRINT A(I,J);
80 NEXT J
90 PRINT
100 NEXT I
110 PRINT : PRINT "ELEMENTOS DE LA SEGUNDA TABLA"
120 FOR I=1 TO 4
130 FOR J=1 TO 2
140 READ B(I,J)
150 PRINT B(I,J);
160 NEXT J
170 PRINT
180 NEXT I
190 FOR I=1 TO 3
200 FOR J=1 TO 2
210 C(I,J)=0
220 FOR L=1 TO 4
230 C(I,J)=C(I,J)+A(I,L)*B(L,J)
240 NEXT L
250 NEXT J
260 NEXT I
270 PRINT : PRINT "LA TABLA PRODUCTO ES"
280 FOR I=1 TO 3
290 FOR J=1 TO 2
300 PRINT C(I,J);
310 NEXT J
320 PRINT
330 NEXT I
340 REM DATOS PRIMERA TABLA
350 DATA -1,3,5,2,6,5,-2,3,0,1,2,3
360 REM DATOS SEGUNDA TABLA
370 DATA 0,3,1,4,-5,2,1,6
380 END
```


103. TRIANGULO DE TARTAGLIA

Prepara un programa para obtener el triángulo de Tartaglia.

Si observamos el triángulo de Tartaglia, también conocido con el nombre de triángulo de Pascal:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
. . . . . .
```

vemos que un elemento central de la línea 4, por ejemplo el 3, se obtiene sumando los dos elementos de la línea que está por encima, que en este caso es la línea 3 (3 = 1 + 2).

Nuestro triángulo de Pascal tendrá como máximo 20 líneas, pues así se ha dimensionado la variable con dos índices C(I,J).

El primer uno se asigna desde el principio, pues se toma C(0,0) = 1; y los de los lados se obtienen en las sucesivas ejecuciones de la línea 90. Los restantes elementos se construyen mediante otro bucle, de variable N, anidado con el anterior, con la expresión C(M,N) = C(M − 1,N − 1) + C(M − 1,N). Por ejemplo, sería:

$$C(4,2) = C(3,1) + C(3,2)$$

Presentación en pantalla:

```
HASTA QUE LINEA? 9
 1
 1  1
 1  2  1
 1  3  3  1
 1  4  6  4  1
 1  5  10  10  5  1
 1  6  15  20  15  6  1
 1  7  21  35  35  21  7  1
 1  8  28  56  70  56  28  8  1
```

```
10 REM TRIANGULO DE TARTAGLIA
20 DIM C(20,20)
30 PRINT "HASTA LA LINEA";
40 INPUT L
50 REM CALCULO DE LOS ELEMENTOS
60 C(0,0)=1
70 FOR M=1 TO L-1
80 REM LOS ELEMENTOS DE LOS EXTREMOS
90 C(M,0)=1 : C(M,M)=1
100 REM LOS RESTANTES ELEMENTOS
110 FOR N=1 TO M-1
120 C(M,N)=C(M-1,N-1)+C(M-1,N)
130 NEXT N
140 NEXT M
150 REM ESCRITURA DE LAS LINEAS ELEGIDAS
160 FOR M=0 TO L-1
170 FOR N=0 TO M
180 PRINT C(M,N);
190 NEXT N
200 PRINT
210 NEXT M
220 END
```


104. VALORES APROXIMADOS DEL SENO (5.11)

El valor de sen(x) puede aproximarse mediante la siguiente función:

$$sen(x) \simeq x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \qquad (x \text{ en radianes})$$

Haz el programa que presente una tabla de la función seno, de 0° a 45° con valores de grado en grado. Compara estos resultados con los que se obtienen utilizando la función SIN(X) del ordenador.

La variable I va a recorrer los grados de uno en uno, desde 0 a 45. Como X tiene que ir en radianes, lo primero que hay que hacer es convertir I grados en X radianes. Esto lo puedes conseguir mediante la instrucción:

```
X=I*3.14159265/180
```

Si observas atentamente la fórmula del enunciado, verás que se puede pasar de un sumando a otro multiplicando el precedente por X^2 y dividiéndolo por el producto de dos números naturales consecutivos, haciendo abstracción de los signos, evidentemente. Esto es:

$$X \cdot \frac{X^2}{2 \cdot 3} = \frac{X^3}{3!}; \qquad \frac{X^3}{3!} \cdot \frac{X^2}{4 \cdot 5} = \frac{X^5}{5!}; \qquad \frac{X^5}{5!} \cdot \frac{X^2}{6 \cdot 7} = \frac{X^7}{7!}$$

Más tarde habrá que sumar estos términos cambiándolos alternativamente de signo.

Si al primer término lo llamamos A=X, los tres siguientes los podemos calcular con la fórmula:

```
A=-A*X*X/(2*J*(2*J+1))      para      J=1, 2, 3
```

A medida que vayamos obteniendo estos valores de A, los iremos acumulando en la variable S, con el signo que les corresponda:

```
S=S+A
```

Si previamente hemos inicializado la variable S, con S = X, al final tendremos en S un valor aproximado de sen(X).

```
10 REM VALOR APROXIMADO DEL SENO
20 PRINT "ANGULO";TAB(7);"V. APROXIMADO";TAB(25);"FUNCION SIN"
30 FOR I=0 TO 45
40 X=I*3.14159265/180
50 S=X : A=X
60 FOR J=1 TO 3
70 A=-A*X*X/(2*J*(2*J+1))
80 S=S+A
90 NEXT J
100 PRINT TAB(2);I;TAB(6);S;TAB(24);SIN(X)
110 NEXT I
120 END
```


105. VARIABLES ALEATORIAS BIDIMENSIONALES

Pretendemos obtener un programa que nos facilite los parámetros que caracterizan las distribuciones marginales de una distribución bidimensional (media, varianza y desviación típica) y los específicos de la bidimensional (covarianza y coeficiente de correlación) además de las ecuaciones de ambas rectas de regresión.

Si la distribución bidimensional responde a los datos:

$x_1, x_2, \cdots, x_n$ (primera distribución marginal)
$y_1, y_2, \cdots, y_n$ (segunda distribución marginal)

las fórmulas asociadas a los parámetros mencionados son:

$$m_x = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

$$s_x^2 = \frac{(x_1 - m_x)^2 + (x_2 - m_x)^2 + \cdots + (x_n - m_x)^2}{n}$$

$$s_x = \sqrt{s_x^2}$$

$$m_y = \frac{y_1 + y_2 + \cdots + y_n}{n}$$

$$s_y^2 = \frac{(y_1 - m_y)^2 + (y_2 - m_y)^2 + \cdots + (y_n - m_y)^2}{n}$$

$$s_y = \sqrt{s_y^2}$$

para las distribuciones marginales; para la conjunta, tenemos la covarianza y el coeficiente de correlación:

$$co = s_{xy} = \frac{(x_1 - m_x)(y_1 - m_y) + (x_2 - m_x)(y_2 - m_y) + \cdots + (x_n - m_x)(y_n - m_y)}{n}$$

$$r = \frac{s_{xy}}{s_x \cdot s_y}$$

y las rectas de regresión son:

$$y - m_y = \frac{co}{s_x^2}(x - m_x) \qquad y - m_y = \frac{s_y^2}{co}(x - m_x)$$

A la vista de estas fórmulas, resulta evidente que la primera parte del programa ha de repetir cálculos idénticos con datos diferentes. Podemos ahorrarnos trabajo almacenando los datos en una tabla, de la siguiente manera,

$$D = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{pmatrix}$$

y la media y la varianza de las distribuciones marginales en otra:

$$P = \begin{pmatrix} m_x & s_x^2 \\ m_y & s_y^2 \end{pmatrix}$$

```
10 REM DISTRIBUCION BIDIMENSIONAL
20 INPUT "NUMERO DE PARES DE VALORES";N
30 DIM D(2,N),P(2,2)
40 FOR F=1 TO 2
50 FOR C=1 TO N : READ D(F,C) : NEXT C
60 NEXT F
100 FOR F=1 TO 2 : M=0 : V=0
110 FOR C=1 TO N : M=M+D(F,C) : NEXT C
120 P(F,1)=M/N
130 FOR C=1 TO N : V=V+(ABS(D(F,C)-P(F,1)))↑2 : NEXT C
140 P(F,2)=V/N
150 NEXT F
200 (BORRAR LA PANTALLA)
210 FOR F=1 TO 2
220 ON F GOTO 230,240
230 PRINT "PRIMERA VARIABLE" : GOTO 250
240 PRINT "SEGUNDA VARIABLE"
250 PRINT : PRINT "MEDIA=";P(F,1),"VARIANZA=";P(F,2)
260 PRINT "DESV. TIPICA=";SQR(P(F,2)) : PRINT
270 NEXT F
300 CO=0
310 FOR C=1 TO N : CO=CO+(D(1,C)-P(1,1))*(D(2,C)-P(2,1)) : NEXT C
320 CO=CO/N : PRINT : PRINT "COVARIANZA=";CO
330 PRINT "COEF. DE CORRELACION=";CO/SQR(P(1,2)*P(2,2))
340 PRINT : PRINT "RECTAS DE REGRESION" : PRINT
350 PRINT "Y-(";P(2,1);")=";CO/P(1,2);"(X-(";P(1,1);"))"
360 PRINT "Y-(";P(2,1);")=";P(2,2)/CO;"(X-(";P(1,1);"))"
400 DATA 6,5,6,4,5,2,5,3,1,3,4,2,3,6,2,6,3,4,4,5,2,2,1,4,3,1,6,2,4,4
410 DATA 4,3,2,2,3,1,2,3,2,2,4,3,4,2,1,4,3,3,2,3,1,2,3,2,2,1,4,3,3,2
420 END
```


106. SISTEMAS DE ECUACIONES

Resolver un sistema de ecuaciones lineales por el método de diagonalización.

En el ejercicio 7.24 del libro BASIC BASICO se describe la resolución de sistemas de ecuaciones lineales por el método de triangulación o de Gauss. El método de diagonalización da un paso más que clarifica y acorta el programa.

Sea el sistema:

$$\begin{aligned} A(1,1)\cdot X1 + A(1,2)\cdot X2 + A(1,3)\cdot X3 &= A(1,4) \\ A(2,1)\cdot X1 + A(2,2)\cdot X2 + A(2,3)\cdot X3 &= A(2,4) \\ A(3,1)\cdot X1 + A(3,2)\cdot X2 + A(3,3)\cdot X3 &= A(3,4) \end{aligned}$$

donde A(I,J) son números.

Si A(1,1) es distinto de cero, se divide la primera ecuación por A(1,1). Después se resta a las demás la primera multiplicada por A(I,1). Queda un sistema equivalente pero de la forma:

$$\begin{aligned} X1 + B(1,2)\cdot X2 + B(1,3)\cdot X3 &= B(1,4) \\ B(2,2)\cdot X2 + B(2,3)\cdot X3 &= B(2,4) \\ B(3,2)\cdot X2 + B(3,3)\cdot X3 &= B(3,4) \end{aligned}$$

Si A(1,1) es igual a cero, al no poder dividir por cero tomamos en vez de la primera ecuación la segunda: intercambiamos las dos ecuaciones. Pero si A(2,1) también resulta nulo habrá que tomar A(3,1) y el intercambio sería con la tercera. Eso sí, uno de los tres ha de ser distinto de cero, pues de lo contrario la X1 queda indeterminada y se trataría en realidad de un sistema de tres ecuaciones con dos incógnitas.

El paso siguiente consiste en dividir la segunda ecuación por B(2,2), si es distinto de cero. Después se resta a las demás la segunda multiplicada por B(I,2). El sistema resultante es:

$$\begin{aligned} X1 \qquad\qquad + C(1,3)\cdot X3 &= C(1,4) \\ X2 + C(2,3)\cdot X3 &= C(2,4) \\ C(3,3)\cdot X3 &= C(3,4) \end{aligned}$$

Si B(2,2) es igual a cero, intercambiamos la segunda ecuación con la tercera. Si ahora B(3,2) es también nulo, el sistema no tiene ninguna solución o tiene más de una. Sin distinguir ambos casos nuestro programa imprime: "ESTE SISTEMA NO TIENE SOLUCION UNICA" y se finaliza la ejecución.

Ahora se divide la tercera ecuación por C(3,3), que si fuera nulo, revelaría un sistema con más de una solución. Después se resta a cada ecuación la tercera multiplicada por C(I,3). El nuevo sistema es:

$$\begin{aligned} X1 &= D(1,4) \\ X2 &= D(2,4) \\ X3 &= D(3,4) \end{aligned}$$

que refleja directamente el valor de las incógnitas.

El programa emplea las primeras líneas, hasta la 150, en la entrada de datos: los coeficientes y términos independientes.

En las líneas 160 a 270 se buscan los coeficientes de las diagonales no nulos. Si es preciso, se intercambian dos ecuaciones, líneas 240 a 270. Si no es posible obtener un determinado coeficiente de la diagonal no nulo, se escribe el mensaje antes citado y se finaliza la ejecución.

En las líneas 280 a 410 se realiza la diagonalización en dos pasos: primero se divide la ecuación I por A(I,I) y después se les resta a todas las demás multiplicando por A(J,I).

La diferencia esencial con el método de Gauss es que éste en vez de utilizar las líneas:

```
340 FOR J=1 TO E
350 IF I=J THEN 400
```

utilizaría para la triangulación las siguientes:

```
340 IF I=E THEN 410
350 FOR J=I+1 TO E
```

Pero el método de diagonalización, al llegar a este punto ya está en condiciones de presentar los resultados, mientras que el Gauss requiere continuar el trabajo.


```
10 REM SISTEMA DE ECUACIONES
20 (BORRAR LA PANTALLA)
30 INPUT "NUMERO DE ECUACIONES";E
40 DIM A(E,E+1)
50 REM ENTRADA DE DATOS
60 PRINT : PRINT "COEFICIENTES"
70 FOR I=1 TO E
80 PRINT : PRINT "ECUACION";I
90 FOR J=1 TO E+1
100 IF J=E+1 THEN 120
110 PRINT "     COEFICIENTE";J; : GOTO 130
120 PRINT "CONSTANTE";
130 INPUT A(I,J)
140 NEXT J
150 NEXT I
160 REM BUSQUEDA DE UNA ECUACION DE CO-
170 REM EFICIENTE I,I NO NULO
180 FOR I=1 TO E
190 FOR J=1 TO E
200 IF A(J,I)<>0 THEN 240
210 NEXT J
220 PRINT "ESTE SISTEMA NO TIENE SOLU-"
230 PRINT "CION UNICA" : GOTO 470
240 REM INTERCAMBIO DE ECUACIONES
250 FOR K=1 TO E+1
260 X=A(I,K) : A(I,K)=A(J,K) : A(J,K)=X
270 NEXT K
280 REM DIAGONALIZACION
290 Y=1/A(I,I)
300 FOR K=1 TO E+1
310 A(I,K)=A(I,K)*Y
320 NEXT K
330 REM SUSTRACCION DE 2 ECUACIONES
340 FOR J=1 TO E
350 IF I=J THEN 400
360 Y=-A(J,I)
370 FOR K=1 TO E+1
380 A(J,K)=A(J,K)+Y*A(I,K)
390 NEXT K
400 NEXT J
410 NEXT I
420 REM RESULTADOS
430 PRINT
440 FOR I=1 TO E
450 PRINT "X";I;"=";INT(A(I,E+1)*1000+.5)/1000
460 NEXT I
470 END
```

107. INTERPOLACION (9.15)

Con objeto de presentar un texto de más fácil lectura, utilizamos indistintamente los conceptos de polinomio, función polinómica y curva asociada a la anterior.

Tenemos un conjunto de puntos (x_i,y_i) por los que pasa una función. Haz un programa capaz de interpolar, para cualquier valor de x el correspondiente de la y, según el método de Lagrange.

Se define como polinomio de Lagrange el polinomio de grado n:

$$P(x) = Y_0 \cdot P_0(x) + Y_1 \cdot P_1(x) + \cdots + Y_n \cdot P_n(x).$$

Donde:

$$P_0(x) = \frac{(x - x_1) \cdot (x - x_2) \cdots (x - x_n)}{(x_0 - x_1) \cdot (x_0 - x_2) \cdots (x_0 - x_n)}$$

$$P_1(x) = \frac{(x - x_0) \cdot (x - x_2) \cdots (x - x_n)}{(x_1 - x_0) \cdot (x_1 - x_2) \cdots (x_1 - x_n)}$$

. . . . . .

$$P_n(x) = \frac{(x - x_0) \cdot (x - x_1) \cdots (x - x_{n-1})}{(x_n - x_0) \cdot (x_n - x_1) \cdots (x_n - x_{n-1})}$$

Este polinomio que pasa por los puntos (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) es el que utilizamos para interpolar.

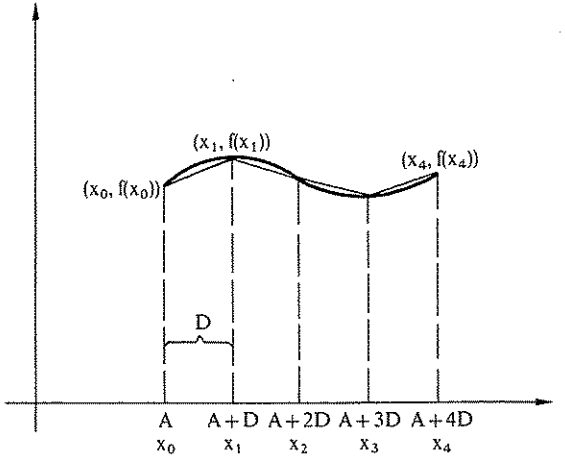
En el programa admitimos como máximo 20 puntos, siendo la abscisa la variable con índice X(I) y la ordenada, la variable con índice Y(I). Después de haber introducido los puntos, damos el valor de x, que queremos interpolar y a continuación calculamos todos los productos del numerador, que colocamos en la variable U y después, todos los del denominador, que colocamos en la variable V y a la vez realizamos el producto del cociente U/V por las ordenadas correspondientes. Al final calculamos Y y preguntamos si se desea interpolar nuevamente o no.


```
10 REM INTERPOLACION POR EL METODO DE LAGRANGE
20 DIM X(20),Y(20)
30 PRINT "CUANTOS PUNTOS";
40 INPUT N
50 PRINT "INTRODUCE LAS COORDENADAS DE LOS PUNTOS"
60 FOR I=0 TO N-1
70 INPUT X(I),Y(I)
80 NEXT I
90 PRINT "INTRODUCE EL VALOR DE X";
100 INPUT X
110 REM CALCULO A PARTIR DE LA FORMULA
120 P=0
130 FOR L=0 TO N-1
140 U=1 : V=1
150 FOR I=0 TO N-1
160 IF L=I THEN 180
170 U=(X-X(I))*U : V=(X(L)-X(I))*V
180 NEXT I
190 P=Y(L)*U/V+P
200 NEXT L
210 PRINT "EL VALOR DE Y ES";P : PRINT
220 PRINT "QUIERES INTERPOLAR OTRO VALOR DE X (S/N)";
230 INPUT A$
240 IF A$="S" THEN 90
250 IF A$="N" THEN END
260 GOTO 220
```

108. INTEGRACION NUMERICA

Se desea obtener una buena aproximación de la integral definida de una función f(x) en un intervalo [A,B].

Para ello vamos a sustituir la función por una poligonal de la siguiente manera: se divide el intervalo [A,B] en N partes iguales (en nuestro dibujo en 4 partes). Así se obtienen los puntos $x_0, x_1, \cdots, x_4$. Si unimos con líneas rectas los puntos $(x_0, f(x_0))$, $(x_1, f(x_1))$, $\cdots$, $(x_4, f(x_4))$, según muestra la figura, resulta una poligonal.



La integral definida es el área limitada por la curva y el eje de las X en el intervalo fijado (omitimos la cuestión del signo de la función, que no afecta a este procedimiento). Vamos a aproximar este área mediante la suma de las áreas de los cuatro trapecios que forman la poligonal y las verticales.

Siendo D=(B−A)/4, el área tiene la siguiente expresión:

$$S = \frac{D\cdot(f(A)+f(A+D))}{2} + \frac{D\cdot(f(A+D)+f(A+2D))}{2} +$$

$$+ \frac{D\cdot(f(A+2D)+f(A+3D))}{2} + \frac{D\cdot(f(A+3D)+f(A+4D))}{2}$$

Es decir:

$$S = D\cdot\left(\frac{f(A)}{2} + f(A+D) + f(A+2D) + f(A+3D) + \frac{f(A+4D)}{2}\right)$$

Cuanto mayor sea el número de partes en que dividimos el intervalo, más se aproximará la poligonal a la curva, y la suma calculada a la integral definida. No hay que olvidar sin embargo, que si el número de divisiones es grande, los errores de cálculo por redondeo o truncamiento de decimales pueden ser apreciables.

En el programa las primeras líneas se destinan a definir la función. El cálculo de la integral comprende las líneas 170 a 200. Primero se efectúa la suma de $f(x_i)$ en el bucle y después la semisuma de la función en los extremos y el producto por la amplitud del intervalo.

```
10 REM INTEGRACION NUMERICA
20 REM METODO DE LOS TRAPECIOS
30 (BORRAR LA PANTALLA)
40 PRINT : PRINT "TECLEA LA FUNCION PRECEDIDA DE"
50 PRINT : PRINT "        100 DEF FNY(X)=..."
60 PRINT : PRINT "DESPUES ESCRIBE RUN 100"
70 STOP
110 (BORRAR LA PANTALLA)
120 PRINT "EXTREMOS DE INTEGRACION" : PRINT
130 INPUT "(A<B)";A,B
140 PRINT
150 INPUT "NUMERO DE DIVISIONES";N
160 S=0 : D=(B-A)/N
170 FOR I=1 TO N-1
180 S=S+FNY(A+I*D)
190 NEXT I
200 S=D*(S+(FNY(A)+FNY(B))/2)
210 PRINT : PRINT "LA INTEGRAL VALE ";S
220 END
```

		Pg.	B.B.	GOTO	IF- -THEN	FOR- -NEXT STEP	DATA READ RES- TORE	GOSUB RE- TURN	ON- GOTO ON- GOSUB RE- TURN	RND	TAB SPC	INT	FUN- CIONES MATE- MATI- CAS	LISTAS L(I)	TABLAS T(I, J)	GET INKEY$	DEF FN	OR AND NOT	MID$ RIGHT$ LEFT$ LEN	STR$ VAL	ASC CHR$	TIME	COLOR INK	PLOT DRAW	CIRCLE
1	Adivina un número	29		•	•					•		•				•		•							
2	Carretera	31		•	•	•				•	•	•				•		•	•						
3	Carreras de caballos	33	7.12		•	•				•	•	•		•		•		•							
4	Ruleta rusa	36		•	•	•				•		•		•											
5	Tiro al plato	38		•	•	•		•		•	•	•				•					•	•			
6	Submarino inmóvil	40	7.22	•	•					•		•						•							
7	Submarino	42			•	•	•	•		•	•	•	•	•		•									
8	Juego del submarino móvil	45	7.23	•	•	•				•	•	•	•								•				
9	Ruleta	47	6.25	•	•					•		•													
10	Ogros	50		•	•	•		•			•					•									
11	Master mind	55		•	•	•				•	•	•						•	•						
12	Las siete y media	59	7.16	•	•	•		•	•	•		•	•	•	•			•		•					
13	Lotería	65	6.24		•					•		•						•							
14	Lanzamiento de una moneda	66	6.16	•	•	•				•		•													
15	¡Qué quiniela!	67			•	•				•		•						•							
16	Quiniela millonaria	68		•		•			•	•		•													
17	La bañera	69	4.9	•	•						•							•							
18	Campanadas	72	4.11	•	•	•																			
19	Suma de los puntos de un dado	75				•								•											
20	Frecuencias relativas de un dado	76	7.2		•	•				•	•	•		•		•		•							
21	Estabilización de las frecuencias	77	6.19	•	•	•				•		•	•	•				•							
22	La colección de cromos	79	7.20		•	•				•		•		•											
23	El destino de una urna	81	7.21		•	•				•		•													
24	Elecciones	83	7.10		•	•				•	•	•		•											
25	Baraja	85	7.15		•	•	•			•	•	•		•	•										
26	Ruido de una información	88			•	•				•		•							•						
27	Deformación de un mensaje	90		•	•	•	•			•		•		•	•										
28	Rumores	93			•						•	•													
29	Cálculo de π	96	6.29		•	•				•															
30	La aguja de Buffon	98			•	•				•	•		•		•										
31	Caza de barcos	100	6.28	•	•	•		•		•	•	•				•									
32	Trenes	103	6.56	•	•						•	•		•				•							
33	Generador de frases	107		•	•	•	•			•		•			•										
34	Supresión de espacios	113	6.51		•	•													•		•				
35	Vocales que hay en una frase	114			•	•								•					•						
36	Frecuencia de un carácter	116	6.49		•	•													•						
37	Flechazo al ordenador	118	6.27	•	•	•				•	•	•						•	•						
38	Frecuencia de una palabra	120	6.50	•	•	•												•	•		•				
39	Conjugación	122	6.54			•	•			•					•				•						
40	Mensaje secreto	124		•		•								•					•		•				
41	Morse	127	8.21	•	•	•	•							•				•	•		•				
42	Poesía aleatoria	129		•	•	•		•		•	•	•		•				•							

		Pg.	B.B.	GOTO	IF- -THEN	FOR- -NEXT STEP	DATA READ RES- TORE	GOSUB RE- TURN	ON- GOTO ON- GOSUB RE- TURN
43	Giro de letras	132	6.41		•				
44	Bandera U.S.A.	135				•		•	
45	Evolución de la población	137	6.15		•	•	•		
46	Diagrama de barras horizontales	140		•	•	•			
47	Diagrama de barras verticales	143		•	•	•			
48	Ascensor	147		•	•	•		•	
49	Representación gráfica de funciones	149		•	•	•			
50	Recta entre 2 puntos	153	6.42	•	•	•			
51	Bandera de franjas horizontales	160				•			
52	Bandera de franjas verticales	161				•			
53	Mondrian	162				•	•		
54	Espiguilla	164		•		•		•	
55	La hormiga mareada	167		•	•				
56	Escudo	169				•			
57	Flor de circunferencias	171				•			
58	Haz de rectas	173				•			
59	Polígono regular	175				•			
60	Estrella	177				•			
61	Clavos y cuerdas	180				•			
62	Espiral	182				•			
63	Trébol	184				•			
64	Factura	189		•					
65	Cambio de divisas	191	8.6	•	•	•	•		
66	El recibo de la luz	193	4.30	•	•				
67	Anualidad constante	195				•			
68	Venta de billetes	198	6.38	•	•	•	•	•	
69	Taquilla automática	201	4.24	•	•	•			
70	Hiper	203	7.19	•	•	•	•		
71	Ordenación alfabética	208	7.17		•	•			
72	Ordenación por inserción	209		•	•	•			
73	Ordenación por edades	213	7.7		•	•			
74	Geografía económica	215	9.24	•	•	•	•		•
75	Creación de un fichero	218		•	•		•		
76	Sectores industriales	221		•	•	•	•		
77	Tablas de sumar	227		•	•	•			
78	Multiplicar sumando	228	4.7	•	•	•			
79	Dividir restando	230	4.8	•	•	•			
80	Suma de polinomios	232	8.12	•	•	•			
81	Suma de filas y columnas en una tabla	234	7.5			•			
82	Conversión de ángulos	236		•	•				
83	Conversión de longitudes	237	4.22	•	•				
84	Sumas para Pepito	239				•			

	RND	TAB SPC	INT	FUN- CIONES MATE- MATI- CAS	LISTAS L(I)	TABLAS T(I, J)	GET INKEY$	DEF FN	OR AND NOT	MID$ RIGHT$ LEFT$ LEN	STR$ VAL	ASC CHR$	TIME	COLOR INK	PLOT DRAW	CIRCLE
43		•							•							
44																
45			•				•									
46			•		•					•						
47			•		•	•				•						
48							•									
49			•	•		•		•	•							
50		•	•	•					•							
51						•								•		
52						•								•		
53					•									•		
54			•											•		
55	•			•					•						•	
56				•										•	•	
57				•												•
58				•											•	
59				•											•	
60				•											•	
61				•											•	
62				•											•	
63				•											•	
64		•								•	•					
65		•			•						•					
66		•	•							•	•					
67		•	•													
68		•				•			•		•					
69							•									
70		•				•			•	•	•					
71					•											
72					•											
73					•											
74		•			•	•			•							
75																
76																
77							•									
78																
79																
80					•											
81					•	•										
82			•													
83			•													
84		•	•				•			•	•					

		Pg.	B.B.	GOTO	IF- -THEN	FOR- -NEXT STEP	DATA READ RES- TORE	GOSUB RE- TURN	ON- GOTO ON- GOSUB RE- TURN
85	Puntos en un círculo	241			•	•			
86	Números primos	243	6.3		•	•			
87	Descomposición en factores primos	244	6.4	•	•	•			
88	Máximo común divisor	245	6.5	•	•				
89	Tabla de valores de una función	247	6.43			•			
90	Cuadrado perfecto	248	6.9		•	•			
91	Capicúa	250	6.11		•	•			
92	La vaca y el prado	252	6.31		•	•			
93	Area de un polígono	254				•			
94	Método de bipartición	256	6.45	•	•	•		•	
95	Método de la secante	259		•	•	•			
96	Pantano	261			•	•	•		
97	Lago	264		•	•	•	•		
98	Día de la semana	266		•	•				•
99	Viaje relativista	268				•			
100	Refracción	269				•			
101	Tabla y su traspuesta	271	7.8			•	•		
102	Multiplicación de dos tablas	273				•	•		
103	Triángulo de Tartaglia	275				•			
104	Valores aproximados del seno	277	5.11			•			
105	Variables aleatorias bidimensionales	279		•		•	•		•
106	Sistemas de ecuaciones	281		•	•	•			
107	Interpolación	284		•	•	•			
108	Integración numérica	286				•			

	RND	TAB SPC	INT	FUN- CIONES MATE- MATI- CAS	LISTAS L(I)	TABLAS T(I, J)	GET INKEYS	DEF FN	OR AND NOT	MIDs RIGHTs LEFTs LEN	STRs VAL	ASC CHRs	TIME	COLOR INK	PLOT DRAW	CIRCLE
85			•													
86			•													
87			•													
88			•													
89		•	•					•								
90			•				•		•	•	•					
91			•	•						•	•					
92				•				•								
93					•											
94	•			•				•								
95		•		•				•								
96																
97		•									•					
98			•						•							
99		•	•	•						•	•					
100		•		•												
101						•										
102						•										
103						•										
104		•		•												
105				•		•										
106						•										
107					•											
108								•								

De los mismos autores:

BASIC BASICO. CURSO DE PROGRAMACION

PROGRAMAS COMENTADOS DE **BASIC BASICO**.

BASIC JUNIOR. INICIACION A LA PROGRAMACION