Fundamentos de Informática

Ejercicios resueltos de Programación en C.

Dirigido a Ingenieros Técnicos Industriales y Grados en Ingeniería

M. Carmen Ocariz Sanz
Montserrat Ferreira
Rosa Arruabarrena Santos
Olatz Ansa Osteriz

Escuela Universitaria Politécnica
Unibertsitate Eskola Politeknikoa

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

www.ehu.es
ISBN: 978-84-9860-438-2

ISBN: 978-84-9860-438-2

INDICE

EJERCICIOS CORTOS DE PROGRAMACIÓN

A continuación aparecen ejercicios sencillos de programación, escritos en lenguaje C. Al principio aparece una lista de veinte enunciados que puedes intentar resolverlos antes de consultar las soluciones que te aparecen después.
Para la verificación de las soluciones propuestas se ha empleado la versión 4.9.9.2 de Dev-C++, software libre que se puede descargar desde http://www.bloodshed.net/ .

ENUNCIADOS

1. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
 int s, j, k, i, l;
 s=0;
 for (j=1;j<=5;j++)
 { printf("%d", j );
   if (j%2 ==0) { s=s+j; }
 }
 printf("\n%d",s);
 i= 10;
 while (i>0)
   i=i-1;
 printf("\n%d",i);
 printf("\n\n");
 system ("PAUSE");
}
```

2. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
 int i, j;
 for (i=1; i<=10; i++)
 {   for (j=1; j<=10-i+1; j++)
     {  printf(" %d",j); }
     printf("\n");
 }
 printf("\n\n");
 system ("PAUSE");
}
```

3. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
 int i, j, sum;
 for (i=1; i<=3; i++)
 {   sum=0;
     for (j=1; j<=i; j++)
     {  sum=sum+j;
        if (i!=j) {printf("%d/%d + ",i,sum);}
        else {printf("%d/%d \n",i,sum);}
     }
 }
 printf("\n\n");
 system ("PAUSE");
}
```

4. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
#define maxFila 6
main()
{ int fila, columna, numero;
  numero=0;
  for (fila=1;fila<=maxFila; fila++)
  { columna=1;
    while (columna<=fila)
    { numero ++;
      printf("%5d", numero);
      columna ++;
    }
     printf("\n");
  }
  printf("\n\n");
  system ("PAUSE");
}
```

5. Escribe cuál es la salida del siguiente fragmento de programa:

```
main()
{ int Minimo=1, Maximo=5, i, dato=25;

  for(i=Maximo;Minimo<=i;i--)
  { printf("\n el que va el  %d es %d \n", i, dato-i); }
  printf("Está terminado");

  printf("\n\n");
  system ("PAUSE");
}
```

6. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
main()
{ int Minimo=1, Maximo=3, i,j;

  for(i=Minimo;i<=Maximo;i++)
    for(j=Minimo;j<=Maximo;j++)
      printf("i vale %d y j vale %d .\n", i, j);
  printf("\n\n");
  system ("PAUSE");
}
```

7. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
main()
{ int i,k;

  for (i=1; i<=5; i++)
  { for (k=1; k<=i; k++)
      printf("*");
    printf("\n");
  }
```

```
  for (i=4; 1<=i; i--)
  { for (k=1; k<=i; k++)
      printf("*");
    printf("\n");
  }

  printf("\n\n");
  system ("PAUSE");
}
```

8. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
main()
{int a,b,n,x,y;
 n=6;
 x=0;
 y=0;
 for (a=1; a<=n; a++)
 { if (a%2== 0)
   { for (b=a; b<n; b++) {x=x+1;}
     printf("\n%d >>> %d", a,x);
   }
   else
   { for (b=1; b<a; b++) {y=y+1;}
     printf("\n%d >>> %d", a,y);
   }
 }
 printf("\n\n");
 system ("PAUSE");
}
```

9. Escribe cuál es la salida del siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main()
{char cadena[]="este error es el grave?";
 int cont,i,desde, hasta, ind, tam;
 cont=0;
 tam= strlen(cadena);     // tamaño de la cadena
 for (ind=0; ind<tam; ind++)
 { if (cadena[ind]=='e')
   {cont++;}
 }
 printf("En la cadena '%s' hay %d  'e'\n",cadena, cont);
 for (i=1;i<=cont;i++)
 { printf("%d 'e' está: ",i);
   desde=0;
   hasta=0;
   while (desde<i)
   { if (cadena[hasta]=='e')
     {desde++;}
     hasta++;
   }
   for (ind=0; ind<hasta; ind++)
   {printf("%c", cadena[ind]);}
```

```
   printf("\n");
 }
 printf("\n\n");
 system ("PAUSE");
}
```

10. Di si los dos fragmentos de programa que están a continuación son equivalentes. Razona la respuesta.

    (Decir que son equivalentes significa que escribir uno u otro fragmento dentro de un programa da el mismo resultado.)

A.-	
`if (x<10) {x:=x+1;}` `if (x>=10) {x:=x-1;}`	`if (x<10) {x:=x+1;}` `else {x:=x-1;}`

B.-	

`for (i=10;i<=25; i++)`
`  printf("  %d \n",i);`
`printf("Ha terminado");`
	`i=10;`
	`while (i<=25)`
	`{ printf("  %d",i);`
	`  printf("Ha terminado \n ");`
	`}`

11. ¿Los dos fragmentos de programa, hacen lo mismo ?. Si es que si dí que hacen y si es que no ¿porqué no?:

A	`esta= 0;`
	`ind = 0;`
	`while ((ind<N) && (esta==0))`
	`{ if (x==tabla[ind])`
	`  {esta=1;}`
	`  ind=ind+1;`
	`}`
	`if (esta==1)`
	`{printf("%d está en la posición %d ", x, ind-1);}`
	`else`
	`{printf("%d no está", x);}`

B	`esta=0;`
	`for (ind=0;ind<N;ind++)`
	`{  if (x==tabla[ind])`
	`  {esta=1;}`
	`}`
	`if (esta==1)`
	`{printf("está en la posición %d  ", x, ind-1);}`
	`else`
	`{printf("%d no está", x);}`

12. ¿ Hacen lo mismo los dos fragmentos de programa?. Si es que si dí qué hacen y si es que no ¿porqué no?:

A	printf("Dame un número: ");
	scanf("%d", &n);
	cont=1;
	while(cont<=n)
	{ printf("%d \n", cont);
	cont=cont+2;
	}

B	printf("Dame un número: ");
	scanf("%d", &n);
	for(cont=1;cont<=n;cont++)
	{ if (cont%2==0)
	{printf("%d \n", cont);}
	}

13. Escribe qué hace el siguiente programa.

```
#include <stdio.h>
#include <stdlib.h>
main()
{ int x, y, num;
  printf("Dame un número: ");
  scanf("%d",&num);
  x=0;
  y=0;
  while (num!=0)
  { if (num%2==0)
    {y++;}
    x++;
    scanf("%d",&num);
  }
  printf("\Total de números: %d", x);
  printf("\nTotal de números pares: %d", y);

  printf("\n\n");
  system ("PAUSE");
}
```

14. El siguiente programa contiene errores, corrígelo:

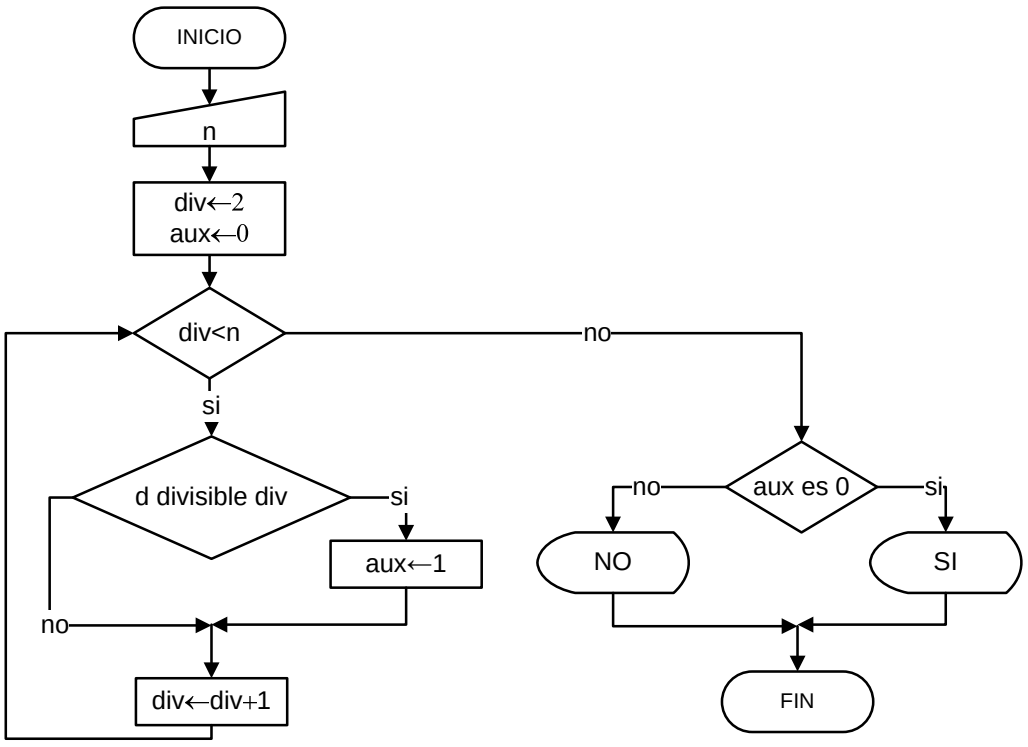
```
#define <stdio.h>
#define <stdlib.h>
#include Tope 10;

main()
{int tabla [tope], i, aux;
 for (i=0; i<Tope; i++)
 {   tabla[i]=i;
     i=i+1;
 }
 i=0;
 while (i<=Tope)
 { aux=tau[i];
   tabla[i]=tabla[i+1];
   tabla[i+1]=aux;
 }
```

```
 printf("\n\n");
 system ("PAUSE");
}
```

15. Sustituye las sentencias if por un único **switch** :

```
#include <stdio.h>
#include <stdlib.h>

main()
{ int n;
  printf("Escribe un número: ");
  scanf("%d", &n);
   if ((n==5) || (n==6) ) {n=n+4;}
   else if ((n>7) && (n<=10)) { n=n-5;}
   else if (n==7) {n=n+1;}
   else {n=n-1; }
  printf("Número: %d", n);

  printf("\n\n");
  system ("PAUSE");
}
```

16. Dí que hace el siguiente diagrama de flujo y a continuación escribe el correspondiente programa en C :



17. Realiza el diagrama de flujo y el programa en C que haga lo siguiente:

    a. Se teclearán dos números enteros por pantalla hasta que los dos sean menores que 50.

b. El más pequeño se irá incrementando de 5 en 5 y el más grande se decrementará de 2 en 2, se irán imprimiendo a la vez que se van generando. El programa terminará cuando los valores se crucen.

18. El siguiente programa intenta contar cuantos carácteres "**t**" seguidos de "**a**" hay en una frase que se pedirá por pantalla, la frase acabará con un punto. En el programa hay 5 errores, corrígelos. Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Escribe un frase(para acabar un punto): abba tkktajjtaiitaktaoi.
 – ta – aparece 4 veces
```

```
#include <stdio.h>
#include <stdlib.h>
main()
{ int antes, tot=0;
  char ahora;
  antes='z';
  printf("Escribe una frase, para acabar un punto '.': \n");
  scanf("%c", &ahora);
  while (ahora!='.')
    if (ahora=='a' || antes=='t') then {tot ++;}
    antes=ahora;
    scanf("%c", &orain);

  printf(" aparece veces  '-ta-'", tot);

  printf("\n\n");
  system ("PAUSE");
}
```

19. Haz un programa que pida un número entero y obtenga la suma de los cuadrados de los números de 4 en 4 que hay hasta llegar a él. Por ejemplo, si el dato de entrada es **10**, la solución vendrá dada por el resultado de la siguiente operación: $1^2 + 5^2 + 9^2$

20. ¿Qué hace el siguiente programa?

```
#include <stdio.h>
#include <stdlib.h>
main()
{ int x, n=0;
  for (x=45; 1<=x; x--)
    if (x<30) {n=n-x;}
    else {n=n+x;}
  printf("%d", n);
  printf("\n\n");
  system ("PAUSE");
}
```

SOLUCIONES PROPUESTAS

1.
```
12345
6
0
```

2.
```
 1 2 3 4 5 6 7 8 9 10
 1 2 3 4 5 6 7 8 9
 1 2 3 4 5 6 7 8
 1 2 3 4 5 6 7
 1 2 3 4 5 6
 1 2 3 4 5
 1 2 3 4
 1 2 3
 1 2
 1
```

3.
```
1/1
2/1 + 2/3
3/1 + 3/3 + 3/6
```

4.
```
     1
     2   3
     4   5    6
     7   8    9   10
    11  12   13   14   15
    16  17   18   19   20   21
```

5.
```
el que va el 5 es 20.
el que va el 4 es 21.
el que va el 3 es 22.
el que va el 2 es 23.
el que va el 1 es 20.
Está terminado.
```

6.
```
i vale 1 y j vale 1.
i vale 1 y j vale 2.
i vale 1 y j vale 3.
i vale 2 y j vale 1.
i vale 2 y j vale 2.
i vale 2 y j vale 3.
i vale 3 y j vale 1.
i vale 3 y j vale 2.
i vale 3 y j vale 3.
```

7.
```
*
**
***
****
*****
****
***
**
*
```

8.
```
1 === 1
2 >>> 5
3 === 4
4 >>> 8
5 === 9
6 >>> 9
```

9.
```
En la cadena ‘este error es el grave?’ hay 6 ‘e’.
1 ‘e’ está: e
2 ‘e’ está: este
3 ‘e’ está: este e
4 ‘e’ está: este error e
5 ‘e’ está: este error es e
6 ‘e’ está: este error es el grave
```

10. A) No son equivalentes, en el primer fragmento cuando x es 9 a x se le suma 1, con lo cual es 10, y al seguir en la siguiente instrucción como x es 10 se le resta 1 a la x, en el segundo fragmento si x es menor que 10 se le suma 1 a la x y en caso contrario se le resta 1 a la x.

    B) No son equivalentes, en el primer fragmento se escribe los números de 1 a 25 y al final Ha terminado, en el segundo fragmento como no se incrementa la i, comienza con 10 y escribe 10 Ha terminado de forma indefinida.

11. No tienen el mismo efecto, en los dos se intenta ver si un valor x está en una tabla de N elementos, en el primer fragmento termina de recorrer la tabla bien porque lo ha encontrado o porque se han acabado los elementos, en el segundo recorre siempre toda la tabla y además si lo ha encontrado siempre imprime que está en la posición ind-1, número de elementos de la tabla menos 1.

12. No tienen el mismo efecto, en los dos se solicita un número por pantalla, imprimiéndose los números impares hasta llegar al número en el primer fragmento y en el segundo fragmento los pares.

13. Se van pidiendo números por pantalla hasta que se teclee el número 0, en y se van sumando el número de pares que se introducen y en x el número de números que se introducen, al final se imprime x e y.

14.
```
#define <stdio.h>                 corregido #include <stdio.h>
#define <stdlib.h>                corr.  #include <stdio.h>
#include Tope 10;                 corr.  #define Tope 10
main (){
 int tabla [Tope], i, lag;

 for (i=1; i<Tope; i++)           corr.  for (i=0; i<Tope; i++)
 {   tau[i]=i;
```

```
    i=i+1;                          sobra
 }

 i=1;                               corr.  i=0;
 while (i<=Tope)          corr.  while (i<Tope)
 { lag=tau[i];
   tau[i]=tau[i+1];
   tau[i+1]=lag;                    falta  i=i+1;
 }

 printf("\n\n");
 system ("PAUSE");
}
```

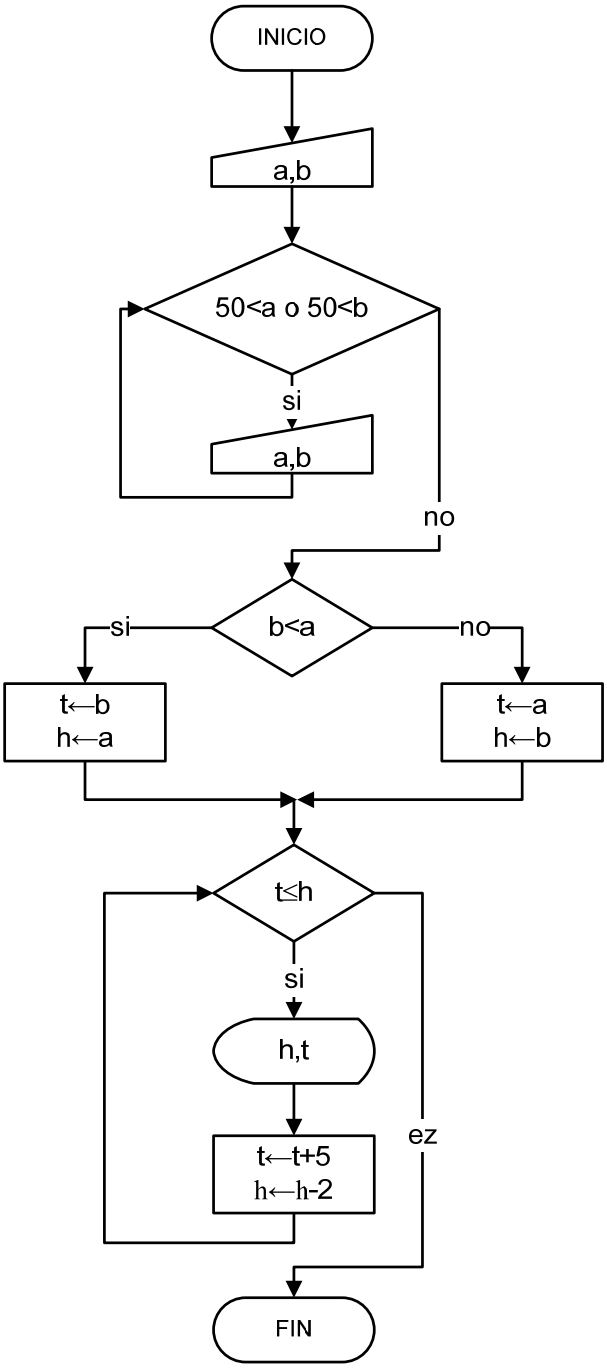
15.
```
 switch (n)
 {
  case 5: case 6: n=n+4; break;
  case 7: n=n+1; break;
  case 8: case 9: case 10: n=n-5; break;
  otherwise: n=n-1;
  }
```

16. Se imprime SI si un número tecleado previamente por pantalla es primo y NO si no lo es.

```
#include <stdio.h>
#include <stdlib.h>

main ()
{ int n, div,aux;
  printf("Dame un número: ");
  scanf("%d", &n);
  div=2;
  aux=0;
  while (div<n)
  { if (n%div==0)
    {aux=1;}
    div=div+1;
  }
  if (aux==0)
  {printf("SI");}
  else
  {printf("NO");}

  printf("\n\n");
  system ("PAUSE");
}
```

17.
```
#include <stdio.h>
#include <stdlib.h>

main ()
{ int a,b,t,h;
  printf("Dame dos números menores que 50 ");
  scanf("%d %d", &a, &b);
  while (50<a || 50<b)
```

```
  { printf("Dame dos números menores que 50 " );
    scanf("%d %d", &a, &b);
  }
  if (b<a)
  {t=b; h=a;}
  else
  {t=a; h=b;}
  while (t<=h)
  { printf("El más pequeño : %5d .El más grande: %5d\n",t,h);
    t=t+5;
    h=h-2;
  }

  printf("\n\n");
  system ("PAUSE");
}
```



18.
```
#include <stdio.h>
#include <stdlib.h>

main ()
{ int  tot=0;
  char ahora,antes;
  antes='z';
  printf("Escribe una frase, para acabar un punto '.': \n");
  scanf("%c", &ahora);
  while (ahora!='.')
  { if (ahora=='a' || antes=='t')
    {tot ++;}
    antes=ahora;
```

```
    scanf("%c", &ahora);
  }
  printf(" aparece %d veces '-ta-'", tot);

  printf("\n\n");
  system ("PAUSE");
}
```

19.
```
#include <stdio.h>
#include <stdlib.h>
main ()
{ int n, z;
  long sum;
  printf("Teclea un número: ");
  scanf("%d",&n);
  sum=0;
  z=1;
  while (z<=n)
  { sum= sum + z*z;
    z=z+4;
  }
  printf("Solución: %ld", sum);
  printf("\n\n");
  system ("PAUSE");
}
```

20. Empieza el programa dando el valor 45 a la x, entra en un bucle mientras x sea mayor o igual que 1, en el bucle si x es mayor o igual que 30 realiza n=n+x y si x es menor que 30 realiza n=n-x, en los dos resta 1 a x. Al final se imprime el valor 165.

EJERCICIOS LARGOS

A continuación se presentan 17 ejercicios largos estilo examen para programarlos en C, puedes intentar hacerlos, consultando a continuación el análisis y programación propuestos para cada uno de ellos.

Para la verificación de las soluciones propuestas se ha empleado la versión 4.9.9.2 de Dev-C++, software libre que se puede descargar desde http://www.bloodshed.net/ .

COMPONENTES ELECTRÓNICOS

En una empresa de montaje de aparatos electrónicos se desea un programa que calcule ciertas características de los productos de la empresa. La empresa monta 10 tipos diferentes de aparatos electrónicos, cada uno de los cuales incluye un número de componentes diferentes. En total la empresa utiliza 100 componentes electrónicos diferentes. Suponemos que cada aparato sólo tiene **una unidad** de cada uno de los componentes que lo forman.

Debes construir un programa que realice las siguientes funciones:

1. En primer lugar se procederá a la lectura de los datos que indican cuáles son los componentes de cada aparato electrónico. El usuario introducirá los datos relativos a los 10 aparatos, indicando primero el número de componentes que tiene.
2. A continuación, el programa deberá leer el número de unidades que se montan mensualmente de cada aparato.
3. Para realizar los pedidos correspondientes, se deben contabilizar el número de componentes que se necesitan cada mes y escribirlos en la pantalla.
4. El programa deberá identificar el componente eléctrico que más se utiliza.

Ejemplo: (en negrita los datos introducidos por el usuario)

Apartado 1)

```
Introduce los componentes de cada aparato
Número de componentes del aparato 0: 3
    Componente 1: 1
    Componente 2: 3
    Componente 3: 99
Número de componentes del aparato 1: 20
    Componente 1: 1
    Componente 2: 3
    ...
    Componente 20: 99
...
Número de componentes del aparato 9: 1
    Componente 1: 65
```

Apartado 2)

```
Introduce el número de aparatos montados mensualmente:
    Aparato 0: 100
    Aparato 2: 1000
    ...
    Aparato 9: 300
```

Apartado 3)

```
Número de unidades de cada componente utilizados mensualmente:
    Componente 0: 100
    ...
    Componente 99: 100
```

Apartado 4)

```
El componente electrónico 7 es el más utilizado, se necesitan
10000 unidades.
```

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *NumAparatos*=10. Es el número de aparatos distintos que se montan en la empresa.
2. *NumComponentes* =100. Es el número de componentes electrónicos que se utilizan para montar los diferentes aparatos.

Representación del las variables necesarias para la entrada

3. Necesitaremos una matriz de *NumAparatos* X *NumComponentes* posiciones para almacenar por cada aparato los componentes electrónicos que contiene; de manera que si *desglose*[*a*][*c*]=1 indica que el aparato a tiene el componente *c*, y *desglose*[*a*][*c*]=0 indica que el aparato *a* no contiene el componente electrónico *c*.

desglose	0	2	...	NumComponentes-1
0				
2				
...				
NumAparatos-1				

4. Necesitaremos un vector de *NumAparatos* posiciones para almacenar por cada aparato el número de montajes que se hacen del mismo al mes.

	0	2	...	NumAparatos-1
montajesAMes				

Representación de las varables necesarias para los resultados

5. Necesitaremos un vector de *NumComponentes* posiciones para almacenar por cada componente el número total del citado componente que se requiere para montar todos los aparatos pedidos de un mes.

	0	2	...	NumComponentes-1
totalesComp				

Pasos a seguir en la resolución del problema

1. Inicializar la matriz *desglose* a 0; ello indicará que cada aparato inicialmente no tiene componente alguno asociado. Ello se consigue asignando el valor 0 en cada una de las posiciones de la citada matriz, es decir, por ejemplo, recorriendo la matriz aparato por aparato y dentro de cada aparato componente por componente y asignando el valor 0 a cada par aparato-componente.
2. Leer ordenadamente la composición de todos los aparatos:
   a. Por cada aparato primeramente leer el número de componentes que lo contienen, sea éste *aTieneXComp*.

   b. Y posteriormente uno a uno leer y registrar en la matriz *desglose* los *aTieneXComp* identificadores de los componentes concretos del aparato. Para registrar que un aparato a tiene un componente *yComp* se almacena en la celda *desglose[a][yComp]* el valor 1.

3. Leer por cada aparato cuántos aparatos de cada tipo se van a montar en un mes, registrando dichos valores en el vector *montajesAMes*.

4. Calcular por cada tipo de componente cuántos componentes se van a emplear entre todos los montajes de aparatos, combinando para ello los datos de la matriz *desglose* y el vector *montajesAMes*.

   Para determinar el número de componentes de tipo *c* que se necesitarán en un mes y registrarlo en *totalesComp[c]*:

   a. Se inicializa a 0 el acumulador *totalesComp[c]*.

   b. Por cada tipo de aparato *a* se calcula el número de componentes de tipo *c* que se emplearán en un mes para montar todos los aparatos a: *desglose[a][c]* montajesAMes[a],* ya que si el aparato *a* no contiene el componente *c* entonces *desglose[a][c]* tendrá el valor 0. El número calculado de componentes *c* que se emplearán en los montajes de los aparatos tipos *a* se acumula en *totalesComp[c]*.

   Los dos pasos anteriores se repiten por cada tipo de componente *c* y se muestra en pantalla.

5. Para calcular el componente más empleado, basta con determinar en qué posición está el valor máximo del vector *totalesComp*:

   a. Inicialmente considerar que el primer valor de *totalesComp* es el máximo hasta el momento (*posMaxComp*).

   b. Con el resto de componentes (sea *c* el índice para denotar cada uno de ellos), se repite una y otra vez:

      i. Se comprueba si el número de componentes totales de tipo *c* es mayor que el máximo hasta el momento: *totalesComp[c]>totalesComp[posMaxComp]*.

      ii. Si es mayor, se actualiza la posición en la que se ubica el nuevo máximo; es decir, la variable *posMaxComp* con el componente actual *c*.

   c. Se imprime un mensaje en pantalla indicando que el componente *posMaxComp* es el componente más empleado, utilizándose un total de *totalesComp[posMaxComp]* unidades.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define NumAparatos 10
#define NumComponentes 100
main(){
  int desglose[NumAparatos][NumComponentes];
  int montajesAMes[NumAparatos];
  int totalesComp[NumComponentes];
  int a,c, totalComp, nc, posMaxComp;

// Inicializar desglose: Ningún componente por aparato
for (a = 0; a< NumAparatos; a++)
{   for( c = 0 ;c< NumComponentes ; c++)
    {   desglose[a][c]=0;}
}
// Lectura de los componentes de cada uno de los aparatos
printf(" Introduce los componentes de cada aparato");
for (a = 0 ;a< NumAparatos; a++)
{  printf(" Número de componentes del aparato %d :",a);
   scanf("%d",&totalComp);
   for( nc = 1 ;nc<= totalComp ; nc++)
   { printf(" Componente %d: ", nc);
     scanf ("%d",&c);
     desglose[a][c]=1;
     /*El aparato a tiene el componente c. Para identificar que a
     lo contiene se pone a 1 la celda [a][c] de la matriz
     desglose*/
   }
}
// Lectura de los aparatos montados mensualmente
printf("Introduce el número de aparatos montados mensualmente:\n");
for (a = 0 ; a< NumAparatos ; a++)
{  printf(" Aparato %d: ",a);
   scanf("%d",&montajesAMes[a]);
}
printf(" Número de componentes utilizados mensualmente:\n ");
```

```
for (c = 0; c<NumComponentes; c++)
{  totalesComp[c]=0;
   for (a=0 ; a<NumAparatos; a++)
   { totalesComp[c]=desglose[a][c]*montajesAMes[a]+totalesComp[c];}
   printf(" Componente %d : %d \n ",c,totalesComp[c]);
}
// El componente más empleado entre todos los montajes del mes.
posMaxComp= 0;
for (c= 1;c< NumComponentes ;c++)
{   if (totalesComp[c]>totalesComp[posMaxComp] )
       { posMaxComp=c; }
}
printf(" El componente electrónico %d es el más utilizado, se "
      "necesitan %d unidades.",posMaxComp,totalesComp[posMaxComp]);
printf("\n\n");
system ("PAUSE");
}
```

HÁBITOS DE CONSUMO: BEBIDAS

El Gobierno Vasco quiere hacer un estudio de hábitos de consumo de alcohol y bebida entre la juventud vasca. Para ello, ha escogido 100 pueblos donde llevará a cabo la consulta. El ayuntamiento de cada uno de estos pueblos recogerá el número de litros consumidos durante el pasado año de los 8 tipos de bebidas nombrados a continuación:

Tipos De Bebidas

0. Naturales: agua, leche, zumos, mostos, infusiones, ...
1. Gaseosos: Refrescos
2. Vinos
3. Cervezas
4. Whisky
5. Licores
6. Energéticos: Aquarius, Red Bull,...
7. Combinados: Cubatas, GinTonics,..

Mediante un programa en C, se te pide que realices un estudio que estará compuesto por los siguientes pasos:

1. Recoge y almacena la información, para cada pueblo, sobre los litros consumidos de los tipos de bebidas mencionados. Recuerda que 100 pueblos aportarán información a tu estudio. No obstante, la información no te la tienen por qué dar ordenada por tipo de bebida o número de litros y puede que de algún tipo de bebida no se tenga información.

   Así, será preciso introducir pares (tipo de bebida, litros) y para indicar que no hay más datos de un pueblo, bastará con introducir como tipo de bebida el valor -1. Ver ejemplo para cada uno de los apartados al final del enunciado.

2. Determina el "tipo de bebida" más consumida entre todos los pueblos; es decir, el tipo de bebida del que más litros se hayan bebido entre todos los pueblos.

3. Recoge y almacena cuáles de las clases de bebidas tienen alcohol. Para ello, te dan únicamente la lista de los "tipos de bebidas" que contienen alcohol acabada con un -1 para indicar que no hay más tipos de bebida que tengan alcohol. Por ejemplo: **7 2 5 4 3 -1**

   a. De los "tipos de bebidas" con alcohol, ¿cuál es el más consumido?

b. ¿En qué pueblo se bebe más alcohol en total?

Ejemplo: (En letra normal lo que el programa imprime y en negrita lo que el usuario teclea)

```
Para comenzar el estudio recogemos los datos por pueblos:
Por cada pueblo: Introduce pares (tipoBebida, litros). Para
acabar, el primer componente del último par tiene que tener
valor -1.
Pueblo 0:
    0  8
    4  7
    2  0
    3  50
    1  10
    -1  0
Pueblo 2:
    0  18
    1  15
    4  0
    2  6
    3  36
    -1  32
Pueblo 3:
    5  16
    ...
Pueblo 99:
    3  5
    1  12
    -1  9

El tipo de bebida más consumida es la: 3
{Suponiendo que lo que más se ha bebido en litros sean cervezas}

Ahora necesitamos que enumeres SÓLO los tipos de bebida CON
alcohol, -1 para acabar:     7    2    5    4    3    -1

El tipo de bebida con alcohol más consumida es la: 3

(Suponiendo que así sea en litros entre: "combinados","vinos",
"licores", "whiskies" y "cervezas".En general, el "tipo de
bebida" y "el tipo de bebida con alcohol" más consumidas no
tiene por qué coincidir).

El pueblo con mayor consumo de alcohol: X

(Suponiendo que entre "combinados", "vinos", "licores",
"whiskies" y "cervezas", X sea la población que más litros haya
vendido).
```

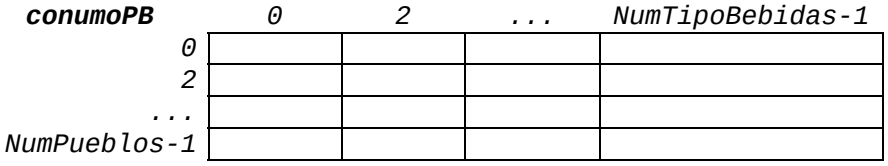
ANÁLISIS DEL PROBLEMA

Constantes del enunciado

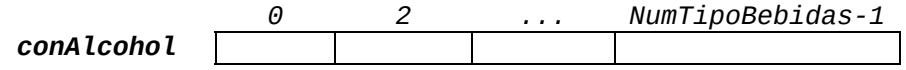
1. *NumPueblos*= 100. Es el número de pueblos con los que se va a hacer el estudio de hábitos de consumo.
2. *NumTipoBebidas*= 8. Es el número de tipos de bebidas.

Representación del las variables necesarias para la entrada

3. Necesitaremos una matriz de *NumPueblos X NumTipoBebidas* posiciones para almacenar por cada pueblo y por cada tipo de bebida los litros consumidos; de manera que si *consumpPB[i][j]=z* indica que el pueblo *i* ha consumido *z* litros del tipo de bebida *j*.

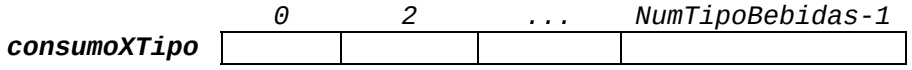


4. Necesitaremos un vector de *NumTipoBebidas* posiciones para almacenar si el tipo de bebida es o no alcohólica (1 para indicar que tiene alcohol y 0 en caso contrario).

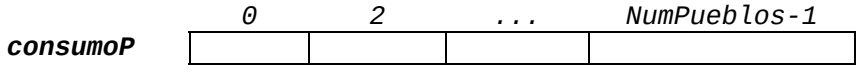


Representación de las varables necesarias para los resultados

5. Necesitaremos un vector de *NumTipoBebidas* posiciones para almacenar por cada tipo de bebida el número total de litros consumidos en los *NumPueblos* del País Vasco.



6. Necesitaremos un vector de *NumPueblos* posiciones para almacenar por cada pueblo el número total de litros alcohólicos consumidos entre todos los tipos de bebidas.



Pasos a seguir en la resolución del problema

1. Inicializar la matriz *consumoPB* a 0; ello indicará que cada pueblo, por cada tipo de bebida, aún no tiene litros consumidos asociados.
2. Leer ordenadamente por pueblos, pares que indican tipo de bebida y litros de la misma consumidos. En el supuesto de que se repita un par (pueblo - tipo de bebida), los litros consumidos del último par serán los

que prevalezcan (es decir, no se hace acumulación de litros consumidos por tipo de bebida).

Para ello, por cada pueblo *p*:

a. Leer un par de valores de tipo de bebida y litros consumidos (*tb*,*l*).

b. Mientras el tipo de bebida no sea -1 (*tp!=-1*): asignar al pueblo *p* en el tipo de bebida *tb* los *l* litros consumidos (*consumoPb[p][tb]=l*) y leer otro par de valores de tipo de bebida y litros consumidos.

c. Si el tipo de bebida es -1, indica que los datos de consumo del pueblo *p* ya han sido proporcionados.

3. Calcular por cada tipo de bebida los litros consumidos entre todos los pueblos y determinar, de esta forma, la bebida más consumida.

Por cada tipo de bebida *b*:

a. Inicializar a 0 los litros consumidos en su acumulador (*consumoXTipo[b]=0*).

b. Incrementar el contador de bebida tipo *b* con los litros consumidos en en todos los pueblos (*consumoXTipo[b]= consumoPB[p][b] + consumoXTipo[b]*, por cada pueblo *p*).

Una vez que se tienen calculados los litros consumidos por tipos de bebidas y recogidos en el vector *consumoXTipo*, se determina la bebida (índice del vector) con mayor contenido (mayor consumo de litros) y se imprime en pantalla. El proceso consiste en:

c. Inicialmente se considera que el mayor consumo de litros lo tiene la bebida 0 (*maxTB*).

d. Para el resto de bebidas se comprueba si los litros consumidos de la bebida actual *b*, supera los litros consumidos por la bebida líder hasta el momento *maxTB* (determinado mediante la comparación *consumoXTipo[b]>consumoXTipo[maxTB]*); en cuyo caso se registra la nueva bebida líder en consumo. En caso contrario, la bebida líder hasta el momento sigue manteniéndose líder, y no hay que hacer nada más.

e. Se imprime en pantalla un mensaje indicando cuál es la bebida de mayor consumo.

4. Leer los tipos de bebidas que tienen alcohol. Las operaciones a realizar para ello son:

a. Se inicializa a 0 el vector *conAlcohol*, indicando que inicialmente se consideran todas las bebidas no alcohólicas.

b. Se lee un tipo de bebida *tb*.

c. Mientras que el tipo de bebida no sea -1 (*tb!=-1*): se actualiza la bebida *tb* como alcohólica (*conAlcohol[tb]=1*) y se lee otro tipo de bebida.

d. Cuando el tipo de bebida es -1, la identificación de bebidas alcohólicas ha finalizado y con ello la lectura de los tipos de bebida con alcohol.

5. Calcular la bebida alcohólica más consumida. Ello se hace combinando los datos de los vectores *consumoXTipo* y *conAlcohol*.

   a. Se inicializa a 0 los litros máximo de alcohol consumidos (*maxAlcohol*)

   b. Así, por cada bebida *b*, y sólo si ésta tiene alcohol (*conAlcohol[b]==1*) se comprueba si los litros consumidos de la misma (*consumoXTipo[b]*) superan el mayor consumo hasta el momento (*maxAlcohol*) (ambas condiciones se comprueban mediante la comparación (*conAlcohol[b]==1and consumoXTipo[b]>=maxAlcohol)*); y si la comparación es cierta, se actualiza el nuevo máximo: la bebida alcohólica líder en consumo (identificada por la posición del vector) y los litros consumidos de la misma (*posMaxAlcohol=b* y *maxAlcohol=consumoXTipo[b])*.

   c. Se imprime en pantalla un mensaje indicando cuál es la bebida alcohólica con mayor consumo (*posMaxAlcohol*).

6. Calcular el pueblo con mayor consumo de alcohol. Se calcula de forma análoga pero, en lugar de por tipos de bebidas, por pueblos. Es decir,

   Por cada pueblo *p*:

   a. Inicializar a 0 los litros con alcohol consumidos en su acumulador (*consumoP[p]=0*).

   b. Incrementar dicho contador con los litros consumidos en el pueblo *p* de todas los tipos de bebidas alcohólicas (*consumoP[p]= consumoPB[p][b] + consumoP[p]*, por cada tipo de bebida *b* siendo *b* con alcohol (*conAlcohol[b]==1*)).

   Una vez que se tienen calculados los litros, con alcohol, consumidos por pueblos y recogidos en el vector *consumoP*, se determina el pueblo (índice del vector) con mayor contenido (mayor consumo de litros alcohólicos) y se imprime en pantalla. (El cálculo es análogo al recogido en el paso anterior pero ahora con el vector *consumoP,* en lugar de *consumoXTipo*).

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define NumPueblos 100
#define NumTipoBebidas 8
main (){
  int consumoPB[NumPueblos][NumTipoBebidas],consumoP[NumPueblos];
  int consumoXTipo[NumTipoBebidas],conAlcohol[NumTipoBebidas];
  int p, b, l, tb, tp, maxP, maxTB, maxAlcohol;
  int  posMaxAlcohol, posPMaxAlcohol;

//Inicializar ConsumoPB a 0 litros por pueblo-tipo bebida.
for (p = 0 ; p< NumPueblos ; p++)
   for (b = 0 ; b< NumTipoBebidas ;b++)
       consumoPB[p][b]=0;
//Lectura de los litros consumidos, ordenadamente por pueblos.
printf("Recogemos los datos por pueblos:\n");
printf("Por cada pueblo: Introduce pares (TipoBebida, litros).\n"
       "Para acabar, el primer componente del último par tiene"
       "que tener valor -1.\n");
for (p=0 ; p<NumPueblos; p=p+1)
{  printf("   Pueblo %d\n", p);
   scanf("%d %d",&tb,&l); //tipoBebida y litros de la misma en p
   while (tb!=-1)
   {  consumoPB[p][tb]= l;
      //Si el tb se ha repetido, prevalecerán los últimos litros}
      scanf("%d %d",&tb,&l);
   }
}
//Cálculo de la bebida más consumida entre todos los pueblos
for (b=0 ; b<NumTipoBebidas; b=b+1)
{  consumoXTipo[b]=0;
   for (p=0; p<NumPueblos ; p=p+1)
   { consumoXTipo[b]=consumoPB[p][b]+consumoXTipo[b];}
}
maxTB=0;        //Bebida tipo 0 inicialmente}
for (b=1; b< NumTipoBebidas; b=b+1)
{  if (consumoXTipo[b]>consumoXTipo[maxTB])
```

```
     { maxTB=b;}
}
printf("El tipo de bebida más consumida es la %d \n: ", maxTB);


/*Bebidas alcohólicas: obtención y algunas estadísticas.
  Iniciar todas las bebidas como NO alcohólicas (conAlcohol[x]=0
  Posteriormente, las que sean CON registrarlas a =1*/
for (b=0 ; b< NumTipoBebidas; b=b+1)
{ conAlcohol[b]=0; }
printf("Ahora necesitamos que enumeres SÓLO los tipos de bebida "
       " CON alcohol, -1 para acabar:\n ");
scanf("%d",&tb);
while (TB!=-1)
{  conAlcohol[tb]=1;
   scanf("%d",&tb);
}
maxAlcohol=0;      //0 litros inicialmente
for (b=0; b< NumTipoBebidas; b=b+1)
{   if (conAlcohol[b]==1 && consumoXTipo[b]>=MaxAlcohol)
    {   posMaxAlcohol=b;
        maxAlcohol=consumoXTipo[b];
    }
}
printf(" El tipo de bebida con alcohol más consumida "
       " es %d ",posMaxAlcohol);


//Pueblo con mayor consumo de alcohol
for (p=0;p< NumPueblos; p=p+1)
{  consumoP[p]=0;
   for (b=0; b< NumTipoBebidas; b=b+1)
   {  if (ConAlcohol[b]==1)
      { consumoP[p]=consumoPB[p][b]+consumoP[p]; }
      //Acumulación total del alcohol consumido en el pueblo p.
   }
}


maxAlcohol=0;       //0 litros inicialmente
for (p=0; p< NumPueblos; p=p+1)
```

```
{   if (consumoP[p]>maxAlcohol)
    {  posPMaxAlcohol=p;
       maxAlcohol=consumoP[p];
    }
}
printf(" El pueblo con mayor consumo de alcohol es: %d "
       " \n ",posPMaxAlcohol);
system ("PAUSE");
}
```

EMISORA DE RADIO

Una emisora de radio quiere hacer un concurso en el que proponen a sus oyentes los títulos de 10 canciones, a las que ha numerado del 0 al 9. Cada participante debe llamar para dar los números de 3 títulos por orden de preferencia decreciente.

Se le dan 3 puntos a la canción que el oyente nombra en 1er lugar, 2 puntos a la que indica en 2º lugar y 1 punto a la 3ª.

Se pide hacer un programa que realice las siguientes tareas:

7. Leer y almacenar los votos emitidos por cada oyente, suponemos que como máximo pueden concursar 100 oyentes. Y a medida que llaman los concursantes se les asigna un número de forma correlativa.

   La entrada de datos se realizará en forma de tríos, con los números de las canciones que vota cada oyente. Para finalizar se introducirá el valor -1 en el primer dato del trio, sin importar el valor de los otros dos datos.
   Ejemplo:

```
Oyente 0:  6  2  1
Oyente 1:  3  7  5
Oyente 2:  3  0  1
Oyente 3:  0  5  2
Oyente 4:  -1  0  0
```

8. Calcular los votos obtenidos por cada canción e indicar cuáles son la 1º y 2ª canción más votada.
   Según el ejemplo anterior:

```
Canción 0:    5 votos
Canción 1:    2 votos
Canción 2:    3 votos
Canción 3:    6 votos
Canción 4:    0 votos
Canción 5:    3 votos
Canción 6:    3 votos
Canción 7:    2 votos
Canción 8:    0 votos
Canción 9:    0 votos
1ª canción:  3
2ª canción:  0
```

9. Al final de la emisión se reparten puntos entre los oyentes que han concursado de la siguiente manera: 30 puntos si entre las 3 canciones votadas está la 1ª, 20 puntos si está la 2ª y 10 puntos suplementarios si han acertado los dos títulos más votados.

   Se desea obtener el número del oyente que más puntos ha obtenido, ya que se le dará un premio.
   En el ejemplo:

```
Oyente 0:  0 puntos
Oyente 1: 30 puntos
Oyente 2: 60 puntos
Oyente 3: 20 puntos
Ganador: el oyente número 2
```

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *NumOyentes* = 100. El número de máximo de oyentes que van a poder participar en el consurso.
2. *NumCanciones* =10. Número de canciones que participan en el concurso.
3. *NumVotos* =3 . Número de votos que puede emitir cada participante

Representación del las variables necesarias para la entrada

4. Necesitaremos una matriz de *NumOyentes X NumVotos* posiciones para almacenar, por cada oyente participante, los votos emitidos en el concurso de canciones; de manera que si *votosOyentesC[i][j]=X* indica que el *j*-ésimo voto del participante *i* es a la canción *X*.

votosOyentesC	0	2	...	NumVotos-1
0				
2				
...				
NumOyentes-1				

Representación de las varables necesarias para los resultados

5. Necesitaremos un vector de *NumCanciones* posiciones para acumular por cada canción los dados por los oyentes participantes.

	0	2	...	NumCanciones-1
votosXCancion				

Pasos a seguir en la resolución del problema

1. Leer los votos de los oyentes:
   a. Inicializar a 0 el contador de oyentes participantes en el concurso (*p*).
   b. Leer los tres votos del participante (*v1, v2, v3*).
   c. Mientras la tripleta de votos sea válida (primer voto distinto de -1)
      i. Registrar los votos conservando el orden de emisión (*votosOyentesC[p][0]*, *votosOyentesC[p][1]*, *votosOyentesC[p][2]*)
      ii. Incrementar en 1 el contador *p* de participantes
      iii. Leer otra tripleta (*v1, v2, v3*) de votos a canciones
   d. Determinar el número de participantes que ha habido en el concurso (*numParticipantes*), siendo éste *p-1* (debido a que la última tripleta leída al tener -1 no corresponde a ningún participante).
2. Calcular las dos canciones más votadas.

a. Primeramente, se calculan las puntuaciones de las canciones (*votosXCancion*). Para ello realizar los siguientes pasos:

   i. Inicializar los contadores de votos de las canciones a 0; es decir, el vector *votosXCancion* a 0.

   ii. Por cada oyente participante *p*, acumular sus tres votos emitidos (*votosOyentesC[p][j]*), sabiendo que a la primera canción obtiene 3 puntos, la segunda 2 y la tercera 1 punto. Estos incrementos se acumulan a los puntos que ya tienen registrados en el vector *votosXCancion*. Recalcar que han participado los oyentes del 0 a *numParticipantes*.

   iii. Por lo tanto, $numParticipantes \leq NumOyentes$.

   iv. Imprimir en pantalla la puntuación de cada una de las canciones.

b. Posteriormente, se calculan las dos canciones con puntuación más elevada.

   i. Se comparan los votos obtenidos por las canciones 0 y 1. La que tiene más puntos es la *primera* y la otra la *segunda*.

   ii. Con cada una de las canciones restantes *c*:

      (1) Si *c* ha obtenido más puntos que la máxima puntuada hasta el momento (*primera*), actualizar los dos máximos del siguiente orden: *segunda* con *primera* y *primera* con *c*.

      (2) Sino, si *c* ha obtenido más puntuación que *segunda*, actualizar sólo el segundo máximo (*segundo=c*).

   iii. Escribir en pantalla las dos canciones más votadas.

3. Calcular el oyente ganador.

   a. Se inicializa a 0 los puntos máximos actuales (*maxPuntos*).

   b. Por cada oyente participante

      i. Se determina si ha obtenido 0 puntos, 20 puntos, 30 puntos o bien 60 puntos.

      ii. Si la puntuación supera el máximo de puntos actual, actualizar el máximo de puntos actual (*maxPuntos*) y el participante que los ha obtenido (*oyenteGanador*).

   c. Imprimir en pantalla el oyente que más puntos ha conseguido.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define NumOyentes  100
#define NumCanciones  10
#define NumVotos  3
main (){
  int votosOyentesC[NumOyentes][NumVotos];
  int votosXcancion[NumCanciones];
  int p,numParticipantes,c,v,v1,v2,v3,primera,segunda;
  int votosP, maxPuntos, oyenteGanador;

//Comienzo del registro de votos de los oyentes participantes
printf(" Concurso LA CANCION DEL VERANO\n.");
printf(" Cada concursante votará a tres canciones, en orden de\n
");
printf("preferencia de a lo  sumo participarán NumOyentes\n");
printf(" Para acabar, introducir -1 en la primera canción votada
\n");
printf("de la tripleta.\n");
p=0;
printf(" Oyente %d: ",p);
scanf("%d %d %d", &v1,&v2,&v3);
while (v1!=-1)
{  votosOyentesC[p][0]=v1;
   votosOyentesC[p][1]=v2;
   votosOyentesC[p][2]=v3;
   p=p+1;  //Este será el siguiente participante.
   printf(" Oyente %d: ",p);
   scanf("%d %d %d", &v1,&v2,&v3);
}
numParticipantes=p-1;   /*La última tripleta tenía -1 en la primera
                          canción,luego su votación no cuenta*/

//Determinando la puntuación de cada canción
for (c=0 ; c<NumCanciones; c=c+1)
{   votosXcancion[c]=0;}
```

```
for (p=0; p<= numParticipantes; p=p+1)
{ for (v=0; v< NumVotos; v=v+1)
  {  votosXcancion[votosOyentesC[p][v]]=
     (NumVotos-v)+votosXcancion[votosOyentesC[p][v]];
     //la fórmula (NumVotos-v) genera los valores 3, 2 y 1
  }
}
for (c=0; c< NumCanciones; c=c+1)
{ printf("Canción %d: %d votos \n", c, votosXcancion[c]); }

// Determinamoslas dos canciones más votadas.
if (votosXcancion[0]>=votosXcancion[1])
{ primera=0; segunda=1; }
else
{ primera=1; segunda=0; }
for (c=2; c< NumCanciones ; c=c+1)
{  if (votosXcancion[c]>votosXcancion[primera])
   {   segunda=primera;
       primera=c;
   }
   else
   {  //Los votos de c no superan los de primera pero
      if (votosXcancion[c]>votosXcancion[segunda])
      {segunda= c; }
   }
} //fin del for
printf(" 1ª canción: %d \n", primera);
printf(" 2ª canción: %d \n ", segunda);

//Oyente participante que ha acertado las canciones más votadas
maxPuntos=0;     /*Inicialización de la puntuación más elevada
                   conseguida por los participantes*/
for (p=0; p<=numParticipantes ; p=p+1)
{   votosP=0;
    for (v=0; v< NumVotos; v=v+1)
    {  if (votosOyentesC[p][v] = primera)
```

```
        { votosP= 30 + votosP; }
         else
         {  if (votosOyentesC[p][v] = segunda)
            {  votosP= 20 + votosP;  }
         }
         if (votosP = 50)
         { votosP=60;  /*La puntuación debe ser 60 si ha votado
                        a la Primera y la Segunda más votadas*/
         }
    }
    printf(" Oyente participante %d: %d puntos ",p, votosP);
    if (votosP>maxPuntos)
    {  //Se actualiza el máximo con p y con sus respectivos puntos
       maxPuntos=votosP;
       oyenteGanador=p;
    }
}
printf(" El ganador es el participante: %d \n ", oyenteGanador);
system ("PAUSE");
}
```

GASOLINERA

En la gasolinera AMOREBIETA nos han pedido que realicemos un estudio sobre el uso de los distintos surtidores de combustible a lo largo de una semana. En la gasolinera hay 12 surtidores que cada noche se rellenan a 5000 litros.

A lo largo de cada día, el operario introduce tuplas de surtidor y litros solicitados. Si el tanque del que se ha solicitado el combustible tiene suficientes litros se suministrarán todos ellos, sino tan sólo se suministrarán los existentes emitiendo un mensaje en el que se indique los litros que se pueden suministrar. Al final de la jornada, el operario introducirá un –1, como número de surtidor, indicando el fin de datos de esa jornada laboral (como litros solicitados cualquier otra cantidad). Este proceso se repetirá para cada día de la semana.

Ejemplo**:**

```
Día 0:
    Surtidor, litros: 7   40
    Surtidor, litros: 11  300
    Surtidor, litros: 11  400
    Surtidor, litros: 0   36
    ...
    Surtidor, litros: 11  400
           Se llenará con 100 litros
    Surtidor, litros: 0   27
    Surtidor, litros: 11  300
           Se llenará con 0 litros
    Surtidor, litros: 2   57
    Surtidor, litros: -1  0

Día 1:
...

Día 6:
...
```

Además de los datos recogidos habrá que almacenar la siguiente información semanal:

1. Cada surtidor qué tipo de combustible almacena. Sabiendo que hay cinco tipos distintos de combustible codificados de la siguiente manera:

   0) EuroSuper'95

   1) Extra'98

   2) Star'97

   3) ExtraDiesel

   4) Gasoleo

Para cada surtidor se pedirá el tipo de combustible que almacena.

Ejemplo:

```
Introduce el tipo de combustible da cada surtidor:
    Surtidor 0: 0
    Surtidor 1: 1
    Surtidor 2: 0
    …
    Surtidor 11: 4
```

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
tipoSurtidores	0	1	0	1	2	2	3	0	3	0	4	4

2. La ganancia por litro y tipo de combustible se almacenará en otra tabla (la ganancia vendrá en Euros). Se pedirá por cada tipo de combustible la ganancia por litro.

   Ejemplo:

```
Dame la ganancia por litro de cada tipo de combustible:
       Tipo 0: 0.02
       Tipo 1: 0.03
       …
       Tipo 4: 0.02
```

	0	*1*	*2*	*3*	*4*
gananciaTipo	0.02	0.03	0.04	0.01	0.02

Se pide calcular lo siguiente:

a) Visualiza qué días y qué surtidores se han vaciado completamente.

b) Independientemente del tipo de combustible, ¿qué día se han vendido más litros?

c) ¿Qué tipo de combustible ha dado más beneficio esta semana? Y, ¿cuánto ha sido este beneficio?

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *NumSurtidores = 12*. Número de surtidores que tiene la gasolinera.
2. *DiasSemana= 7*. Número de días para los que se va ha realizar el estudio.
3. *NumCombustibles=5*. Número de diferentes combustibles que vende la gasolinera.

Representación del las variables necesarias para la entrada

4. Necesitaremos una matriz de *DiasSemana* x *NumSurtidores* posiciones, dónde se almacenara la cantidad de litros que le quedan por vender a cada surtidor. Al inicio de cada jornada laboral todos los surtidores tendrán 5000 litros por vender.

amorebieta	*0*	*1*	*...*	*NumSurtidores-1*
0				
1				
...				
DiasSemana-1				

5. Necesitaremos un vector de *NumSurtidores* posiciones para almacenar el tipo de combustible que vende cada surtidor.

	0	*1*	*...*	*NumSurtidores-1*
tipoSurtidor				

6. Necesitaremos un vector de *NumCombustibles* posiciones para almacenar la ganancia por litro de cada tipo de combustible.

	0	*1*	*...*	*NumCombustibles-1*
tipoGanancias				

Representación de las varables necesarias para los resultados

7. Necesitaremos un vector de *DiasSemana* posiciones para almacenar los litros vendidos cada día de la semana, teniendo en cuenta todos los tipos de combustible.

	0	1	...	DiasSemana-1
ventasDia				

8. Necesitamos un vector de *NumCombustibles* posiciones para almacenar los litros de cada tipo de combustible vendidos. Finalmente utilizaremos esta información para calcular las ganancias de cada tipo de combustible.

	0	*1*	*...*	*NumCombustibles-1*
ganancias				

Pasos a seguir en la resolución del problema

1. Inicializar la matriz *amorebieta* a 5000.
2. Para cada día:
   a. Leer la primera venta (*sur, lit*).
   b. Mientras el surtidor sea distinto de –1 hacer:
      i. Si tenemos suficientes litros en el surtidor entonces restar los litros que han pedido si no escribir un mensaje con los litros que se le dan y dejar a 0 el surtidor para ese día.
      ii. Leer siguiente venta (*sur, lit*).
3. Para cada surtidor leer el tipo de combustible que vende y almacenarlo en el vector *tipoSurtidor*.
4. Para cada tipo de combustible leer la ganancia que nos da cada litro vendido y almacenarlo en el vector *tipoGanancias*.
5. Recorrer la matriz *amorebieta* y escribir las posiciones dónde encontramos un 0, ya que esto indicará el día y el surtidor que ha vendido todo el combustible.
6. Para cada día, sumar lo que se ha vendido entre todos los surtidores. Para calcular lo que ha vendido un surtidor habrá que restarle a 5000 el valor almacenado en la matriz para ese día y surtidor. Los valores obtenidos se almacenarán en la vector *ventasDia*.
   a. Buscar y escribir la posición del máximo en el vector *ventasDia*.
7. Para cada surtidor:
   a. Contar cuantos litros ha vendido entre todos los días (recordad que para calcular los litros vendidos hay que restarle a 5000 el valor de la matriz) y almacenarlo en la variable *surtidor*.
   b. Mirar el tipo de combustible que vende el surtidor, en el vector t*ipoSurtidor,* y almacenarlo en la variable *combustible*.
   c. Acumular los litros vendidos (calculados en el paso 7-a, *surtidor*) en el vector *ganancias* en la posición que corresponda al tipo de combustible vendido (calculado en el paso 7-b, *combustible*).

      (*ganancias[combustible]=curtidor+ganancias[combustible]*)
8. Para cada tipo de combustible:
   a. Multiplicar el valor almacenado en el vector *ganancias* con el valor acumulador en el vector *tipoGanancias* y dejarlo en el vector *ganancias*.

9. Calcular el combustible que ha dado mayor ganancia, es decir, calcular la posición del máximo en el vector *ganancias*.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define DiasSemana 7
#define NumSurtidores 12
#define NumCombustibles 5
main(){
 int amorebieta[DiasSemana][NumSurtidores];
 int tipoSurtidor[NumSurtidores];
 float tipoGanancias[NumCombustibles], ganancias[NumCombustibles];
 int ventasDia[DiasSemana];
 int g,s, sur, lit, d, dMax, dMaxLitros, gMax, combustible;
 int diario,surtidor;

//Inicializar todos los surtidores de todos los días a 5000 litros
for (d=0; d < DiasSemana; d=d+1)
{   for (s=0; s< NumSurtidores; s=s+1)
    {  amorebieta[d][s]=5000;}
}
//Para cada día de la semana vamos a leer sus ventas
for (d=0 ; d< DiasSemana; d=d+1)
{  //Leer ventas del día d.
   printf(" Dia %d:\n ", d);
   printf("   Surtidor, litros: ");
   scanf("%d %d",&sur,&lit);
   while (sur!=-1) //Mientras no sea el fin de la jornada laboral
   {  //Comprobar si el surtidor tiene suficientes litros
      if  (amorebieta[d][sur]>=lit)
      { amorebieta[d][sur]=amorebieta[d][sur]-lit;}
      else
      {  //No hay suficientes litros, se dan los que hay
       printf("\t Se llenará con %d litros.\n",amorebieta[d][sur]);
       amorebieta[d][sur]=0 ;
      }
    printf("   Surtidor, litros: ");
    scanf("%d %d",&sur,&lit);
```

```
   } //fin del while
} //fin del for

//Lectura del tipo de combustible que tiene cada surtidor
printf(" Introduce el tipo de combustible que tiene cada "
       " surtidor:\n ");
for (s=0; s< NumSurtidores; s=s+1)
{  printf(" Surtidor %d: ", s);
   scanf("%d",&tipoSurtidor[s]);
}

//Ganancias por cada litro de combustible vendido
printf(" Dame ordenadamente la ganancia que supone la venta\n  ");
printf(" de cada litro de combustible:\n ");
for (g=0; g< NumCombustibles; g=g+1)
{  printf(" Tipo %d: ", g);
   scanf("%f",&tipoGanancias[g]);
}

//a) Indica qué días y qué surtidores se han vaciado completamente
for (d=0; d< DiasSemana; d=d+1)
{   for (s=0; s< NumSurtidores;s=s+1)
    {  if (amorebieta[d][s]==0)
       { printf(" El dia %d el surtidor %d se ha quedado sin "
                " combustible.\n ",d,s);
       }
    }
}

/*b) qué día se han vendido más litros
  Calcular para cada día los litros vendidos*/
for (d=0; d < DiasSemana; d=d+1)
{  ventasDia[d]=0;
   /*Los litros vendidos el día d son la suma de lo vendido por
todos los surtidores}
   for (s=0; s< NumSurtidores: s=s+1)
   {   ventasDia[d]= ventasDia[d] + (5000-amorebieta[d][s]);}
```

```
}

//Calcular el día que se ha vendido más
dMax=0;
for (d=0 ; d< DiasSemana; d=d+1)
{  if (ventasDia[d]> ventasDia[dMax])
   {  dMax=d;}
}
printf(" El día de la semana que se han vendido más litros es "
       " %d con %d litros \n',dmax, ventasDia[dMax]);

/*c) ¿Qué tipo de gasolina ha dado más beneficio esta semana?
¿Cuánto? */
for (s=0; s< NumSurtidores; s=s+1)
{ //Calcular los litros que ha vendido el surtidor s.
   surtidor=0;
   for (d=0 ; d< DiasSemana; d=d+1)
   { surtidor= surtidor + (5000-amorebieta[d][s]);}
   //Calcular el tipo de gasolina que vende el surtidor s.
   combustible=tipoSurtidor[s];
   //Acumular litros vendidos por tipo de combustible.
   ganancias[combustible]=ganancias[combustible]+ surtidor;
}
//Calcular ganancias de cada tipo de combustible.
for (g=0; g< NumCombustibles; g=g+1)
{ ganancias[g]= tipoGanancias[g] * ganancias[g];}

//Calcular el combustible que ha dado mayor ganancia.
gMax=0;
for (g=0 ; g< NumCombustibles; g=g+1)
 if (ganancias[g]>ganancias[gMax])
  { gMax=g;}
}
printf(" El tipo de combustible que ha dado mayor ganancias es "
     "%d con una ganancia de %7.2f euros\n",gMax,ganancias[gMax]);
system ("PAUSE");
}
```

TIENDA

Una pequeña tienda guarda información de las ventas de sus artículos a lo largo del año 2009. La tienda dispone de 100 artículos diferentes.

Se pide escribir un programa que realice las siguientes tareas:

1. Leer las ventas de los artículos (en unidades) a lo largo de los 12 meses del año.
2. Leer la lista de precios por artículo y obtener el mes que más ganancias ha tenido.
3. Por cada artículo leer el total de ganancias del periodo comprendido entre los meses de julio y diciembre del año 2008. A continuación comparar estos datos con los del mismo periodo del 2009 (leídos en el apartado 1). Indicar cuántos artículos han superado las ganancias en el año 2009 y cuales son estos artículos.

Ejemplo de ejecución:

☞ Para el apartado 1)

Introduce las ventas de los artículos de la siguiente forma (en negrita los datos tecleados por el usuario):

```
Artículo, mes , ventas : 1   5      800  (son unidades)
Artículo, mes , ventas : 0   4      700
Artículo, mes , ventas : 4   5      100
Artículo, mes , ventas : 6   5      800
Artículo, mes , ventas : 6   11     800
..............
..............
Artículo, mes , ventas : 1   0      800
Artículo, mes , ventas : -1  0      0
```

Tal y como se indica, los datos se introducen a través de tríos desordenados (Artículo, mes , ventas), además, sólo se introducen los datos de los meses y artículos que han tenido ventas. Las ventas indicarán el total de unidades vendidas de ese artículo en ese mes.

☞ Ejemplo del resultado a obtener en pantalla para el apartado 2)

```
El mes 3 es el mes que más ganancias ha tenido: 1000,50 euros.
```

Para realizar esto previamente se leerá un array con los precios, *en euros*, de cada artículo, de la siguiente forma (en negrita los datos tecleados por el usuario):

```
Precio del artículo 0: 123,50
Precio del artículo 1: 12,03
Precio del artículo 2: 1,73
............
```

```
...........
Precio del artículo 99: 129
```

☞ Ejemplo del resultado, para el apartado 3)

```
Los artículos que han superado las ganancias, en el periodo
comprendido entre los meses de Julio y Diciembre, respecto al
año 2008 son: 2, 5, 44, 99

Total artículos: 4
```

Previamente a esto se introducirá en un array las ganancias, *en euros*, realizadas de cada artículo en el año 2008, de la forma (en negrita los datos tecleados por el usuario):

```
Ganancias obtenidas entre los meses de Julio y Diciembre del
2008: (en euros)
Ventas del artículo  0:   970,52
Ventas del artículo  1:  8870,75
Ventas del artículo  2:  1170
...................
...................
Ventas del artículo 99:909870
```

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *NumMeses = 12*. Número de meses para los que se va ha realizar el estudio.
2. *NumArticulos = 100*. Número de artículos que vende la tienda.
3. *Julio = 7*. Es el mes a partir del cual nos piden que comparemos con las ventas del año 2008.

Representación del las variables necesarias para la entrada

4. Necesitaremos una matriz de *NumArticulos* x *NumMeses* posiciones, dónde se almacenarán las unidades de cada artículo vendidas cada mes.

ventas	0	1	...	NumMeses-1
0				
1				
...				
NumArticulos-1				

5. Necesitaremos un vector de *NumArticulos* posiciones dónde almacenaremos el precio de cada artículo.

precio	0	1	...	NumArticulos-1

6. Necesitaremos un vector de *NumArticulos* posiciones dónde almacenaremos las ganancias de cada artículo correspondientes a los meses de Julio – Diciembre del año 2008.

ventas2008	0	1	...	NumArticulos-1

Representación de las varables necesarias para los resultados

7. Necesitamos un vector de *NumMeses* posiciones, dónde almacenaremos las ganancias obtenidas cada mes teniendo en cuenta todos los artículos.

ventasMes	0	1	...	NumMeses-1

8. Necesitaremos un vector de *NumArticulos* posiciones, dónde almacenaremos las ganancias de cada artículo correspondientes a los meses de Junio – Diciembre del año 2009.

ventas2009	0	1	...	NumArticulos-1

Pasos a seguir en la resolución del problema

1. Inicializar la matriz *ventas* a 0.
2. Leer primera tripleta (*art, mes, vent*).
3. Mientras el primer valor de la tripleta sea distinta de -1, hacer:
   a. Almacenar las ventas en la matriz *ventas,* en la fila *art* y columna *mes.*
   b. Leer siguiente tripleta (*art, mes, vent*).
4. Para cada artículo leer su precio y almacenarlo en el vector *precio.*
5. Para cada mes:
   a. Inicializar las ganancias del mes a 0 (*ventasMes*).
   b. Para cada artículo (*art*) sumar a las ganancias del mes (*mes*). Para calcular las ganancias del mes, multiplicar las ventas del artículo en ese mes (almacenado en la matriz *ventas[art][mes]*) con el precio del artículo (almacenado en el vector *precio[art]*).
6. Calcular y escribir el mes de mayor ganancia, es decir, calcular la posición del máximo del vector *ventasMes* (calculado en el apartado 5).
7. Leer las ganancias de cada artículo correspondiente a los meses de Julio-Diciembre del año 2008 y almacenarlo en el vector *ventas2008*.
8. Vamos a calcular las ganancias de cada artículo correspondientes a los meses de Julio-diciembre del año 2009, para ello realizaremos los siguientes pasos:

   Para cada artículo:

   a. Inicializar ventas del artículo a 0 (*ventas2009*)
   b. Para cada mes entre los meses de Julio a Diciembre sumar la cantidad de artículos vendidos ese mes (almacenado en la matriz *ventas*) a los ya acumulados.
   c. Multiplicar la cantidad de artículos vendidos por el precio del artículo (almacenado en el vector *precio*).
9. Inicializar contador de artículos con mayor venta en 2009 a 0 (*num*).
10. Para cada artículo (*art*):
    a. Si las ventas del año 2009 (*ventas2009[art]*) son mayores que las ventas del año 2008 (*ventas2008[art]*) entonces escribir el número del artículo e incrementar contador (*num*) en 1.
11. Escribir contador (*num*) de artículos con mayor venta en 2009.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define NumArticulos 100
#define NumMeses 12
#define Julio 7
main(){
  int ventas[NumArticulos][NumMeses];
  int precio[NumArticulos], ventas2008[NumArticulos];
  int ventas2009[NumArticulos], ventasMes[NumMeses];
  int mVentaMax, art, mes, vent, num;

/*Inicializar a 0 la matriz Ventas para que todas las posiciones
tengan valores válidos tras la lectura*/
for (art=0; art< NumArticulos; art=art+1)
{  for (mes=0; mes< NumMeses; mes=mes+1)
   { ventas[art][mes]=0;}
}

//Introducción de los datos.
printf(" Venta de los articulos:\n ");
printf(" Articulo, Mes, Ventas: ");
scanf("%d %d %d",&art, &mes, &vent);
while (art!=-1)
{  ventas[art][mes]=vent;
   printf(" Articulo, Mes, Ventas: ");
   scanf("%d %d %d",&art, &mes, &vent);
}

//Lectura de los precios de los artículos.
printf(" Dame los precios de los artículos:\n");
for (art=0; art< NumArticulos; art=art+1)
{  printf(" Precio del articulo %d:",art);
   scanf("%d",&precio[art]);
}

//Cálculo de la ganancia mensual por producto.
for (mes=0; mes< NumMeses; mes=mes+1)
{ ventasMes[mes]=0;
  for (art=0; art< NumArticulos; art=art+1)
```

```
  {ventasMes[mes]= ventasMes[mes]+(ventas[art][mes]*precio[art]);}
}
//Posición del vector con máximo valor.
mVentaMax=0;
for (mes=0; mes< NumMeses; mes=mes+1)
{   if (ventasMes[mVentaMax]<ventasMes[mes])
    { mVentaMax = mes; }
}
printf(" El mes con mayor ganancia es %d, con unas ventas"
       " de %d euros.\n", mVentaMax,ventasMes[mVentaMax]);

//Lectura de ganancias por artículo entre Julio-Diciembre del 2008.
printf(" Ganancias obtenidas entre los meses de Julio y Diciembre"
       " del 2008:\n ");
for (art=0; art< NumArticulos; art=art+1)
{  printf(" Ventas del articulo %d: ", art);
   scanf("%d",&ventas2008[art]);
}
//Cálculo de las ganancias por artículo entre Julio-Dic. del 2009.
for (art=0; art< NumArticulos; art=art+1)
{  ventas2009[art]=0;
   for (mes=Julio; mes< NumMeses; mes=mes+1)
   { ventas2009[art]= ventas2009[art] + ventas[art][mes];}
   ventas2009[art]= ventas2009[art]*precio[art];
}
/*Comparar las ganancias del periodo de Julio-Diciembre del año
2008 y 2009 */
printf(" Los artículos que han superado las ganancias, en el"
       " periodo comprendido entre los meses de Julio y "
       " Diciembre, respecto al año 2008 son:");
num=0;
for (art=0; art< NumArticulos; art=art+1)
{ if (ventas2008[art]<ventas2009[art])
  {  printf(" %d, ",art);
     num=num+1;
  }
}
printf("\n Total artículos :%d \n ", num);
system ("PAUSE");
}
```

CENTRAL ELÉCTRICA

Una central eléctrica desea realizar un estudio sobre la cantidad de energía que suministra diariamente. Para ello, el programa informático que se ha de desarrollar deberá permitir recoger los datos de las potencias típicas, en megavatios, suministradas diariamente durante un período máximo de 52 semanas (1 año).

1. La entrada de datos se deberá programar de forma que, para cada semana, se introduzcan las potencias suministradas durante los 7 días.

   Cuando se desee finalizar la entrada de datos (por ejemplo, en el caso de realizar el estudio para un periodo inferior a las 52 semanas) se deberá introducir un –1, como dato de la potencia, en el primer día de la semana.

Ejemplo de entrada de datos:

```
Semana 0
    Potencia suministrada del día 0: 207
    Potencia suministrada del día 1: 301
    Potencia suministrada del día 2: 222
    Potencia suministrada del día 3: 302
    Potencia suministrada del día 4: 22
    Potencia suministrada del día 5: 167
    Potencia suministrada del día 6: 125

Semana 1
    Potencia suministrada del día 0: 367
    Potencia suministrada del día 1: 60
    Potencia suministrada del día 2: 120
    Potencia suministrada del día 3: 111
    Potencia suministrada del día 4: 301
    Potencia suministrada del día 5: 400
    Potencia suministrada del día 6: 434

Semana 2
    Potencia suministrada del día 0: 211
    Potencia suministrada del día 1: 72
    Potencia suministrada del día 2: 441
    Potencia suministrada del día 3: 102
    Potencia suministrada del día 4: 21
    Potencia suministrada del día 5: 203
    Potencia suministrada del día 6: 317

Semana 3
    Potencia suministrada del día 0: 401
    Potencia suministrada del día 1: 340
    Potencia suministrada del día 2: 161
    Potencia suministrada del día 3: 297
    Potencia suministrada del día 4: 441
    Potencia suministrada del día 5: 117
    Potencia suministrada del día 6: 206

Semana 4
    Potencia suministrada del día 0: -1
```

2. Para cada día de la semana calcular y escribir la potencia media e indicar cuál ha sido el día de mayor potencia media.
3. Calcular y escribir la potencia media de todo el periodo sometido a estudio.
4. Calcular y escribir el número de días, en los que la potencia suministrada ha sido superior a la potencia media de todo el periodo (calculado en el apartado anterior).
5. Debido al libre mercado, semanalmente se establecen los precios de ganancia por megavatio. Introducir en un array la ganancia (€/megavatio) para cada semana del periodo estudiado, y calcular y escribir la ganancia semanal y total de la central.

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

6. *NumDias = 7*. Número de días para los que se va ha realizar el estudio.
7. *NumSemanas= 52*. Número de semanas para las que se va ha realizar el estudio.

Representación del las variables necesarias para la entrada

8. Necesitaremos una matriz de *NumSemanas* X *NumDias* posiciones para almacenar las potencias consumidas cada día de la semana.

potencias	*0*	*1*	*...*	*NumDias-1*
0				
1				
...				
NumSemana-1s				

9. Necesitamos un vector de *NumSemanas* posiciones para almacenar el precio en euros de cada megavatio.

	0	*1*	*...*	*NumSemanas-1*
gananciaPS				

Representación de las varables necesarias para los resultados

10. Necesitaremos un vector de *NumDias* posiciones para almacenar la potencia media consumida cada día de la semana.

	0	*1*	*...*	*NumDias-1*
mediaD				

Pasos a seguir en la resolución del problema

1. Inicializar contador semanas a 0 (*semanas*)
2. Leer la potencia correspondiente al primer día de la primera semana (*valorLunes*)
3. Mientras potencia leída (*valorLunes*) sea distinta de –1 y el contador de semanas (*semanas*) sea menor a *numSemanas* hacer:
    a) Almacenar la potencia leída en la matriz *potencias,* fila *semana* y columna *0 (lunes).*
    b) Para el resto de los días de la semana (*dia*) leer la potencia y almacenarla en la matriz *potencias* fila *semana* y columna *dia.*
    c) Incrementar contador *semanas*
    d) Leer la potencia correspondiente al primer día de la semana (*valorLunes*)

4. Decrementar en 1 el contador *semanas*, para que indique el número de la última semana hasta la que se han introducido datos.
5. Calcular la potencia media de la semana y para cada día la potencia media consumida y almacenarlo en el vector *mediaD*, de la siguiente forma:
   a. Inicializar la potencia media de la semana a 0 (*potMedia*)
   b. Para cada día de la semana:
      i) Inicializar suma potencias del día a 0 (*mediaD[dia]=0*).
      ii) Para cada semana tratada acumular la potencia gastada ese día esa semana en la suma de potencias (*mediaD*).
      iii) Dividir la suma de las potencias consumidas entre las semanas tratadas.
      iv) Acumular la potencia del día a la potencia media de la semana.
6. Calcular la potencia media de la semana, para ello dividir la suma de las potencias medias diarias (*potMedia*) entre el número de días. Escribir el resultado por pantalla.
7. Calcular el día de mayor consumo de la semana. Para ello hay que buscar la posición del valor máximo en el vector *mediaD*.
8. Inicializar contador de días que superan la potencia media a 0 (*diasPM*)
9. Para cada día de la semana:
   a) Para cada semana tratada:
      i. Si la potencia consumida esa semana y ese día (*potencias[sem][dia]*) es mayor a la potencia media (*potMedia*) entonces incrementar contador (*diasPM*).
10. Escribir por pantalla el valor del contador *diasPM*.
11. Calcular la ganancia semanal y la ganancia de todo el periodo, para ello:

    Inicializar a 0 la ganancia total (*gananciaTotal*).

    Para cada semana:

    a. Ganancia semana es igual a 0 (*gananciaSemanal*)
    b. Para cada día acumular la potencia consumida y almacenarlo en la variable *gananciaSemanal*
    c. Multiplicar la ganancia semanal (*gananciaSemanal*) con el precio del megavatio por cada semana (almacenado en el vector *gananciaPS*), y escribir por pantalla el resultado obtenido.

   d. Acumular la ganancia semanal (*gananciaSemanal*) a la ganancia total (*gananciaTotal*)

12. Escribir por pantalla la ganancia total (*gananciaTotal*) de la central.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define NumDias 7
#define NumSemanas 52
main(){
  int potencias[NumSemanas][NumDias];
  float mediaD[NumDias],gananciaPS[NumSemanas];
  float potMedia, gananciaTotal,gananciaSemanal;
  int semanas,diasPM,valorLunes,dia,sem,dMax;

//Lectura de los datos.
semanas=0;
printf(" Semana %d \n ", semanas);
printf(" Potencia suministrada del dia 0: ");
scanf("%d",&valorLunes);
while ((valorLunes!=-1) && (semanas<NumSemanas))
{  potencias[semanas][0]=valorLunes;
   //Seguimos en el dia 1(martes).
   for (dia=1 ; dia< NumDias; dia=dia+1)
   {  printf(" Potencia suministrada del dia %d: ",dia);
      scanf("%d",&potencias[semanas][dia]);
   }
   semanas=semanas+1;
   printf(" Semana %d \n ", semanas);
   printf(" Potencia suministrada del dia 0: ");
   scanf("%d",&valorLunes);
}
semanas=semanas-1;  //Última semana que se ha completado.
printf("\n");

//Media de los lunes, de los martes,... y la media de la semana.
potMedia=0;
for (dia=0 ; dia< NumDias; dia=dia+1)
{  mediaD[dia]=0;
   for (sem=0; sem<=semanas; sem=sem+1)
   {  mediaD[dia]=mediaD[dia]+potencias[sem][dia];}
```

```
   mediaD[dia]=mediaD[dia]/(semanas+1);
   printf(" La media del dia %d es %6.2f \n", dia, mediaD[dia]);
   potMedia=potMedia+mediaD[dia];
}
potMedia=potMedia/NumDias;
printf(" Potencia media de todo el periodo:%6.2f \n ", potMedia);

// Calcular el día con mayor potencia.
dMax=0;
for (dia=1 ; dia< NumDias; dia=dia+1)
{   if (mediaD[dMax]<mediaD[dia])
    { dMax=dia; }
}
printf(" Día de la semana con media mayor: %d \n ",dMax);

// Numero de días que superan la potencia media.
diasPM=0;
for (dia=0 ; dia< NumDias; dia=dia+1)
{  for (sem=0; sem<=semanas; sem=sem+1)
   {  if (potencias[sem][dia]>potMedia)
      { diasPM=diasPM+1; }
   }
}
printf("Número de días que superan la potencia media:%d\n”,diasPM);


printf(" Introduce en una línea y separados por blancos la "
       " ganancia semanal por KW consumido: ");
for (sem=0; sem<=semanas; sem=sem+1)
{  scanf("%f",&gananciaPS[sem]); }


gananciaTotal=0;
for (sem=0; sem<=semanas; sem=sem+1)
{  gananciaSemanal=0;
   for (dia=0 ; dia< NumDias; dia=dia+1)
   {  gananciaSemanal=gananciaSemanal+potencias[sem][dia]; }
   gananciaSemanal=gananciaSemanal*gananciaPS[sem];
   printf(" Semana: %d ganancia : %6.2f \n" ,sem,gananciaSemanal);
   gananciaTotal=gananciaTotal+gananciaSemanal;
```

```
}
printf(" La ganancia total conseguida es:%6.2f \n", gananciaTotal);
system("PAUSE");
}
```

LADRÓN ELCA COMAYOR

El señor Elca Comayor ha decidido hacer un programa para ver como va su negocio. El negocio de nuestro amigo es un tanto curioso, pues se dedica al mundo del robo y está sumamente interesado en desarrollar un programa que le ayude a llevar sus cuentas.

Él quiere guardar los datos de las últimas 8 semanas. Hay que tener en cuenta que él puede trabajar todos los días de la semana, pues no cierra nunca su negocio. De esta manera, a medida que va haciendo las cuentas introduce tres datos en el programa: la semana (del 0 al 7), el día (del 0 al 6) y la cantidad que ha robado ese día. Cuando introduzca un -1 en el lugar que le corresponde a la semana el programa entenderá que la entrada de datos ha finalizado.

Ejemplo: (en negrita los datos tecleados por el usuario)

```
Introduce los datos (semana, día, cantidad): 3    5    300
Introduce los datos (semana, día, cantidad): 0    1    50
Introduce los datos (semana, día, cantidad): 5    6    179
Introduce los datos (semana, día, cantidad): 1    2    22
Introduce los datos (semana, día, cantidad): -1    0    0
```

Además de estos datos, nuestro amigo Elca Comayor introduce los datos de sus gastos de las últimas 8 semanas, esto es, cuánto ha gastado cada una de las semanas.

Ejemplo**:**

```
Dato de la semana 0: 77
Dato de la semana 1: 60
...........
...........
Dato de la semana 7: 99
```

Tras introducir todos estos datos, nuestro amigo el empresario quiere conocer los siguientes datos:

1. Cuánto ha robado cada semana y cuál ha sido la semana en la que más ha robado.
2. Cuánto ha robado por cada uno de los días de la semana, esto es, cuánto los lunes, martes, ... y cuál es el día de la semana que más roba, para poder dedicarle más tiempo a su negocio ese día.
3. Cuánto ha ganado cada semana y el total ganado en las 8 semanas, teniendo en cuenta los gastos.
4. Cuál ha sido el primer día en el que no ha cometido ningún robo.

Además, él sabe que si la policía le detiene, recibirá el castigo dependiendo de lo que ha robado:

```
Menos de 5.000 € → Realizará obras en la comunidad
Menos de 10.000€ → 3 meses en prisión
Menos de 20.000€ → 6 meses en prisión
Más de 20.000€ → Un año en prisión
```

A nuestro amigo Elca Comayor no le agrada nada la idea de tener que ir a prisión, por lo que le interesan mucho las consecuencias de su negocio. Por lo tanto, haz que el programa le diga a nuestro amigo cuál es la sanción que se le aplicará si le detienen.

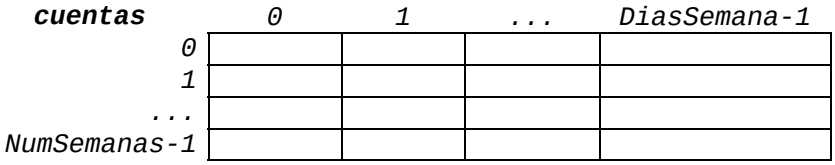
ANÁLISIS DEL PROBLEMA

Constantes del enunciado

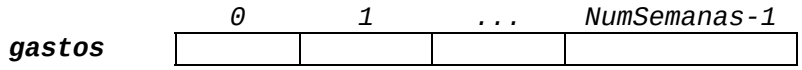
1. *DiasSemana* = 7. Es el número de días que tiene una semana.
2. *NumSemanas* = 8. Es el número de semanas que va a analizar el ladrón.

Representación de las variables necesarias para la entrada

3. Necesitaremos una matriz de *NumSemanas* x *DiasSemana* posiciones, a la que llamaremos *cuentas* dónde se guarden los euros robados cada día de la semana a lo largo de las 8 semanas.

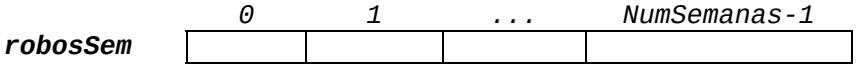


4. Necesitaremos un vector *Gastos*, de *NumSemanas* posiciones, para almacenar los euros que gasta el ladrón cada una de las semanas.

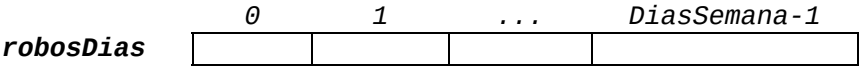


Representación de las variables necesarias para los resultados

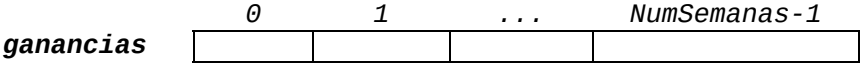
5. Necesitaremos un vector al que llamaremos *robosSem*, de *NumSemanas* posiciones, para almacenar la cantidad de dinero que roba cada una de las semanas.



6. Necesitaremos un vector r*obosDias*, de *DiasSemana* posiciones, para almacenar la cantidad de dinero que roba cada uno de los tipos de día.



7. Necesitaremos un vector *ganancias*, de *NumSemanas* posiciones, para almacenar la cantidad de dinero que gana por semana, esto es, después de hacer la diferencia entre lo que roba y lo que gasta.



Pasos a seguir en la resolución del problema

1. Inicializar la matriz *cuentas* a 0.
2. Leer la primera tripleta correspondiente al robo (*s, d, c*).
3. Mientras la semana sea distinta de -1:

   a. Añadir los datos en la matriz *cuentas*.
   b. Leer siguiente tripleta de robo (*s, d, c*).
4. Para cada semana:
   a. Leer y guardar el dato correspondiente al gasto de dicha semana.
5. Inicializar el vector *robosSem*.
6. Para cada semana:
   a. Calcular cuánto ha robado.
7. Para cada semana:
   a. Escribir por pantalla cuánto ha robado.
8. Calcular y escribir cuál de las semanas ha robado mayor cantidad.
9. Inicializar el vector *robosDias*.
10. Para cada día de la semana:
    a. Calcular cuánto ha robado.
11. Calcular y escribir cuál es el día que más roba.
12. Para cada semana:
    a. Calcular cuánto ha ganado, restando lo que ha robado menos lo que ha gastado.
13. Escribir cuánto ha ganado cada semana.
14. Calcular y escribir lo que ha ganado en total.
15. Inicializar la variable terminar a 0 (falso).
16. Mientras terminar sea 0 y no hayamos leído toda la matriz
    a. Leer la matriz por filas y buscar la primera posición que haya un 0.
    b. Si encontramos la posición en la que hay un 0 poner la variable terminar a 1 (verdad).
17. Escribir la primera posición (semana y día) que hay un 0.
18. Sumar la cantidad que ha robado.
19. Escribir el castigo que le impondrán si le detienen.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define DiasSemana 7
#define NumSemanas 8
main(){
 int cuentas[NumSemanas][DiasSemana];
 int robosSem[NumSemanas], gastos[NumSemanas];
 int ganancias [NumSemanas], robosDias[DiasSemana];
 int s, d, c,semMax, diaMax, gananciaTotal;
 int totalRobado, sem0, dia0,terminar;

//Inicializamos la matriz a 0.
for (s= 0; s< NumSemanas; s=s+1)
{   for (d = 0; d< DiasSemana; d=d+1)
    {  cuentas[s][d]= 0; }
}
//Vamos introduciendo lo que ha robado cada día.
printf(" Introduce los datos (semana, día, cantidad): ");
scanf("%d %d %d",&s,&d,&c);
while (s!= -1)
{  cuentas[s][d]=cuentas[s][d]+c;
   printf(" Introduce los datos (semana, día, cantidad): ");
   scanf("%d %d %d",&s,&d,&c);
}

//Introducimos los gastos de cada semana.
for (s= 0; s< NumSemanas; s=s+1)
{  printf(" Gastos de la semana %d: ", s);
   scanf("%d",&gastos[s]);
}

/*Cuánto ha robado cada semana y cuál ha sido la semana en la que
más ha robado.*/
//Inicializamos el vector RoboSem.
for (s= 0; s< NumSemanas; s=s+1)
   robosSem[s]=0;
```

```
//Calculamos cuánto ha robado cada semana.
for (s= 0; s< NumSemanas; s=s+1)
{  for (d = 0; d< DiasSemana; d=d+1)
   { robosSem[s]=robosSem[s]+cuentas[s][d]; }
  //Escribimos cuánto ha robado cada semana}
  printf(" La semana %d ha robado %d euros.",s,robosSem[s]);
}

//Calculamos qué semana ha robado más.
semMax=0;
for (s=1; s< NumSemanas; s=s+1)
{   if (robosSem[s] > robosSem[semMax])
    { semMax= s; }
}
//Escribimos en qué semana ha robado más.
printf(" La semana que más ha robado ha sido la semana %d"
       " con %d \n ", semMax, robosSem[semMax]);

/*Cuánto ha robado por cada uno de los días de la semana, y cuál es
el día de la semana que más roba*/
for (d = 0; d< DiasSemana; d=d+1)
{  robosDias[d]=0; }
//Calculamos cuánto ha robado por días de la semana.
for (d = 0; d< DiasSemana; d=d+1)
{  robosDias[d]=0; //Inicializamos el vector robosDias.
   for (s=0; s< NumSemanas; s=s+1)
   {  robosDias[d]= robosDias[d] + cuentas[s][d]; }
}
//Calculamos cuál es el día de la semana que más roba.
diaMax=0;
for (d = 1; d< DiasSemana; d=d+1)
{   if (robosDias[d] > robosDias[diaMax])
    {  diaMax = d; }
}
//Escribimos cuál es el día de la semana que más roba.
printf(" El día de la semana que más roba es el ", diaMax);
```

```
/*Cuánto ha ganado cada semana y el total ganado en las 8 semanas,
teniendo en cuenta los gastos*/
//Calculamos cuánto ha ganado cada semana.
for (s=0; s< NumSemanas; s=s+1)
{   ganancias[s]=robosSem[s] - gastos[s];
    printf(" La semana %d ha ganado %d euros.\n",s ,ganancias[s]);
}
//Sumamos lo que ha ganado en total.
gananciaTotal= 0;
for (s=0; s< NumSemanas; s=s+1)
   gananciaTotal= gananciaTotal + ganancias[s];
//Escribimos lo que ha ganado en las 8 semanas.
printf(" En total ha ganado %d euros.\n ", gananciaTotal);

//Cuál ha sido el primer día en el que no ha cometido ningún robo.
terminar= 0;
s=0;
while (terminar==0 && s< NumSemanas)
{ d=0;
   while (terminar==0 && d<DiasSemana)
   { if (cuentas[s][d]==0 )
      {  sem0=s;
         dia0=d;
         terminar = 1;
      }
      d= d + 1;
   }
   s= s + 1;
}
printf(" El primer día en no cometer un robo ha sido el %d"
       " en la semana %d.\n ", dia0, sem0);

//Calculamos el castigo dependiendo de lo que haya robado.
//Calculamos lo que ha robado en total.
totalRobado=0;
for (s=0; s< NumSemanas; s=s+1)
  totalRobado= totalRobado + robosSem[s];
```

```
//Qué castigo recibirá si le detiene la policía.
if (totalRobado< 5000 )
{ printf(" Tendrías que hacer obras en la comunidad. ");}
else if (totalRobado<10000 )
     { printf(" Tendrías que ir 3 meses a prisión. "); }
     else if (totalRobado<20000 )
           { printf(" Tendrías que ir 6 meses a prisión."); }
           else { printf(" Tendrías que ir 1 año a prisión."); }
system ("PAUSE");
}
```

CUMBRES DE MONTAÑA

Tenemos como datos de entrada una lista de 10 valores enteros que nos indican los metros correspondientes a la altura de 10 cumbres de montaña.

A continuación nos dan una lista de pares de datos, donde cada par está compuesto por dos valores enteros (el primero es un valor entre 0 y 6 y el segundo un valor entre 0 y 9). Dichos valores representan el nº del montañero y nº de cumbre que dicho montañero ha conseguido subir alguna vez. Tener en cuenta que un montañero podrá repetir la ascensión de una cumbre de montaña y que la entrada de datos terminará al introducise un -1 en el primer valor del par sin importar el valor del segundo dato.

Se pide:

1. Programar la entrada de datos (altura de las cumbres y la lista de pares terminada en -1) en los vectores y/o matrices que necesites para resolver el problema.
2. Devolver por pantalla la cumbre que menos montañeros hayan subido.
3. Devolver por pantalla el montañero que haya subido más metros.
4. Devolver por pantalla por cada montañero cual es la cumbre más alta que ha subido.

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *NumCumbres*=10. Número de cumbres que puden subir los montañeros.
2. *NumMont* = 7. Número de montañeros que forman parte del análisis.

Representación de las variables necesarias para la entrada

3. Necesitamos una matriz de *NumMont* x *NumCumbres* posiciones, dónde se almacenará el número de veces que ha subido cada montañero a cada una de las cumbres.

cumbres	*0*	*1*	*...*	*NumCumbres-1*
0				
1				
...				
NumMont-1				

4. Necesitaremos un vector de *NumCumbres* posiciones donde almacenaremos la altura de cada una de las cumbres.

	0	*1*	*...*	*NumCumbres-1*
alturas				

Representación de las variables necesarias para los resultados

5. Necesitaremos un vector de *NumCumbres* posiciones, donde almacenaremos el número de veces que ha sido subida cada una de las cumbres.

	0	*1*	*...*	*NumCumbres-1*
ascensiones				

6. Necesitamos un vector de *NumMont* posiciones, donde almacenaremos los metros que ha subido cada uno de los montañeros.

	0	*1*	*...*	*NumMont-1*
metSubidos				

Pasos a seguir en la resolución del problema

1. Para cada una de las cumbres:
   a. Leer y guardar la altura que tiene.
2. Inicializar la matriz *cumbres* a 0.
3. Leer el primer par (m, c).
4. Mientras el número de montañero sea distinto de -1, hacer :
   a. Almacenar la subida a la cumbre en la matriz *cumbres*.
   b. Leer siguiente par (m, c).

5. Inicializar el vector *ascensiones*.
6. Para cada cumbre:
    a. Sumar las ascensiones que han hecho todos los montañeros.
7. Calcular el mínimo número de ascensiones que ha tenido una cumbre.
8. Escribir las cumbres que hayan tenido el número mínimo de ascensiones.
9. Inicializar el vector *metSubidos*.
10. Para cada montañero:
    a. Calcular los metros que ha subido, para ello sumar al número de metros que ha subido el valor de multiplicar el número de ascensiones a la cumbre por la altura a dicha cumbre.
11. Calcular el máximo de metros que haya ascendido un montañero.
12. Escribir el montañero que haya subido el máximo de metros.
13. Para cada montañero :
    a. Calcular la cumbre más alta que ha subido.
    b. Si ha subido alguna montaña, escribir la cumbre más alta que ha subido, de lo contrario decir que no ha subido ninguna cumbre.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define NumCumbres 10
#define NumMont 7
main(){
  int cumbres[NumMont][ NumCumbres];
  int alturas[ NumCumbres], ascensiones[ NumCumbres];
  int metSubidos[NumMont];
  int m, c, min, max, cum;

// Introducimos las alturas de las cumbres.
printf("Introduce las alturas de las cumbres: ");
for (c=0;c< NumCumbres; c=c+1)
{  printf(" Montaña %d: ", c);
   scanf("%d",&alturas[c]);
}
//Inicializamos la matriz cumbres.
for (m=0;m< NumMont; m=m+1)
{   for (c=0 ;c<NumCumbres; c=c+1)
    {  cumbres[m][c]= 0;}
}
//Introducir cumbres que han subido los montañeros.
printf(" Introduce montañero y cumbre: ");
scanf("%d %d",&m, &c);
while (m!=-1)
{  cumbres[m][c]= cumbres[m][c] + 1;
   printf(" Introduce montañero y cumbre: ");
   scanf("%d %d",&m, &c);
}
//Escribir las cumbres menos visitadas.
//Calculamos cuántas veces ha sido visitada cada cumbre.
for (c=0 ;c<NumCumbres; c=c+1)
{ ascensiones[c]=0; //Inicializar el vector ascensiones.
  for (m=0;m< NumMont; m=m+1)
  { ascensiones[c]=ascensiones[c]+cumbres[m][c]; }
```

```
}
//Buscamos el número mínimo de ascensiones.
min=0;
for (c=1 ;c<NumCumbres; c=c+1)
{   if (ascensiones[c]< ascensiones[min])
    { min= c; }
}
/*Escribimos todas las cumbres que hayan tenido el número mínimo de
ascensiones.*/
printf("Las cumbres a las que menos montañeros han subido son:\n");
for (c=0 ;c<NumCumbres; c=c+1)
{   if (ascensiones[c] == ascensiones[min])
    { printf(" %d\n",c); }
}
//Escribir el montañero que haya subido más metros.
//Calculamos cuántos metros ha subido cada montañero.
for (m=0;m< NumMont; m=m+1)
{  metSubidos[m]=0; //Inicializamos el vector metSubidos}
   for (c=0 ;c<NumCumbres; c=c+1)
   {  metSubidos[m]=metSubidos[m] + alturas[c]*cumbres[m][c];}
}
//Calcular el máximo de metros que han subido los montañeros.
max=0;
for (m=1;m< NumMont; m=m+1)
{   if (metSubidos[m] > metSubidos[max])
    { max=m; }
}
//Escribimos los montañeros que hayan subido el máximo de metros.
printf(" El montañero que más metros ha subido es el: ");
for (m=0;m< NumMont; m=m+1)
{   if (metSubidos[max]==metSubidos[m])
    { printf(" %d ", m);}
}
printf("\n");
/*Escribir cuál es la cumbre más alta que ha subido cada
montañero*/
printf("La cumbre más alta que ha subido cada montañero es:\n ");
```

```
for (m=0;m< NumMont; m=m+1)
{  max=0; // Máxima altura subida por el montañero m.
   for (c=0 ;c<NumCumbres; c=c+1)
   {  if (cumbres[m][c]>0 && alturas[c]> max )
      {  max=alturas[c];
         cum=c;
      }
   }
   if (max == 0 )
   { printf(" El montañero %d no ha subido ninguna cumbre.\n",m); }
   else
   { printf(" La cumbre mas alta subida por el montañero %d es %d"
            " con una altura de  %d.\n",m,cum,max); }
}
printf("\n\n");
system ("PAUSE");
}
```

AUTOPISTA

La autopista Bilbao-Behobia desea que realices un programa informático que recoja mensualmente datos sobre los consumos de sus clientes habituales y realice los cálculos que abajo se te enumeran. El programa recogerá el número de veces que pasan los clientes habituales por los distintos peajes de la autopista, todo ello con vistas a poder realizarles una serie de descuentos y poder realizar también un estudio estadístico.

Esta autopista tiene actualmente 1000 clientes reconocidos como habituales a través de un contrato, y el número de peajes en este trayecto es de 15.

a) En un array tendremos que introducir los precios de cada uno de los 15 peajes. Para ello el programa le pedirá al usuario el precio de cada peaje de forma secuencial. Y después, se introducirán los datos de los clientes.

b) La forma de introducir los datos será mediante pares de datos que indiquen: número de cliente y número de peaje por el que ha pasado. Tenemos que tener en cuenta, que puede haber más de una entrada para el mismo cliente y número de peaje, con lo que habrá que acumular el número de veces que ha pasado por el mismo peaje.

Además, a la vez que introducimos estos datos, habrá que calcular en dos vectores, el número total de viajes que lleva realizados el cliente y el importe que va acumulando por todos sus tránsitos, teniendo en cuenta que al precio del peaje se le aplica un descuento en función del siguiente criterio:

- a los 8 primeros viajes un 25% sobre el precio del peaje
- del 9º al 20º viaje un 55% sobre el precio del peaje
- a partir del 20º un 75% sobre el precio del peaje

Ejemplo: (en negrita los datos introducidos por el usuario)

```
Introducir código de cliente y código de peaje (fin con -1 -1):
Cliente, peaje: 1   2
Cliente, peaje: 8   4
Cliente, peaje: 1   2
...
Cliente, peaje: -1   -1
```

1. Obtener cuál es el peaje más transitado por los clientes habituales.
2. Para cada cliente obtener el número de peaje que más utiliza.
3. Calcular para cada cliente el ahorro que ha obtenido este mes, por ser cliente habitual.
4. Listar por pantalla para cada cliente, el número total de viajes que ha realizado este mes, el importe total a pagar y el ahorro que ha obtenido. El listado tendrá que presentar el siguiente formato, ejemplo:

Numero cliente	Total viajes	Importe	Ahorro
0	20	120	30
1	10	79	15

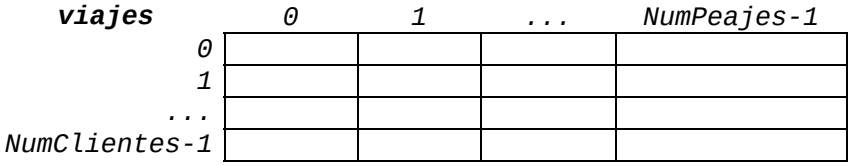
ANÁLISIS DEL PROBLEMA

Constantes del enunciado

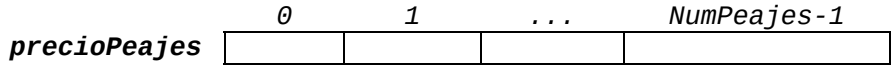
1. *NumPeajes* = 15. Número de peajes que se van a controlar en el programa.
2. *NumClientes* = 10. Número de clientes habituales.

Representación de las variables necesarias para la entrada

3. Necesitaremos una matriz de *NumClientes* x *NumPeajes* posiciones para almacenar los peajes por los que pasa cada uno de los clientes habituales.

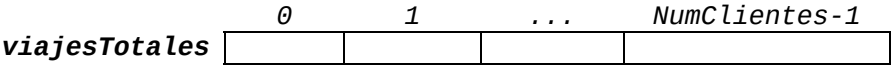


4. Necesitaremos un vector de *NumPeajes* posiciones para almacenar el precio en euros de cada peaje.

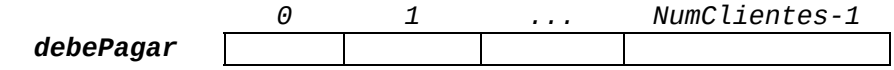


Representación de las variables necesarias para los resultados

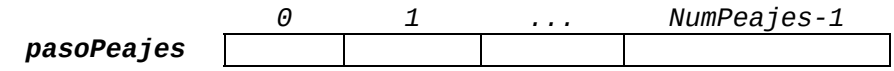
5. Necesitamos un vector de *NumClientes* posiciones para almacenar el número total de pasos por los peajes que tiene un cliente habitual.



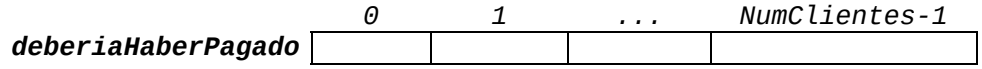
6. Necesitaremos un vector de *NumClientes* posiciones para almacenar los euros que tiene que pagar cada uno de los clientes habituales.



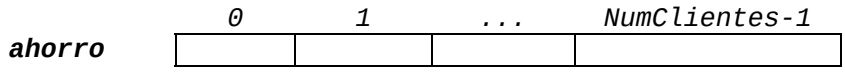
7. Necesitamos un vector de *NumPeajes* posiciones para almacenar el número de veces que se ha pasado por cada uno de los peajes.



8. Necesitaremos un vector de *NumClientes* posiciones para almacenar la cantidad de euros que debería pagar cada cliente habitual si no se le aplica ningún descuesto.



9. Necesitamos un vector de *NumClientes* posiciones para almacenar la cantidad de euros que ahorra cada uno de los clientes habituales.



Pasos a seguir en la resolución del problema

1. Inicializar la matriz *viajes* a 0
2. Para cada peaje:
   a. Leer y guardar el precio
3. Inicializar el vector *viajesTotales* a 0
4. Inicializar el vector *debePagar*
5. Leer el primer par de cliente y peaje (*c, p*)
6. Mientras el cliente y el peaje sean distinto de cero hacer:
   a. Guardar el viaje leído.
   b. Sumar al cliente el viaje leído.
   c. Calcular el precio que tiene que pagar el cliente teniendo en cuenta los datos introducidos hasta el momento.
   d. Leer el par (*c, p*)
7. Inicializar el vector *pasoPeajes*.
8. Para cada peaje:
   a. Calcular el número de veces que han pasado el conjunto de los clientes habituales por él.
9. Calcular el mayor valor del vector *pasoPeajes* para saber cuál ha sido el peaje por el que han pasado más clientes habituales.
10. Para cada cliente:
    a. Buscar cuál es el peaje por donde ha pasado más veces.
11. Inicializar el vector *deberiaHaberPagado*.
12. Para cada cliente:
    a. Calcular cuánto deberían haber pagado sin descuentos.
13. Para cada cliente :
    a. Calcular cuánto ha ahorrado.
14. Escribir cuánto ha ahorrado cada cliente.
15. Para cada cliente:

a. Escribir los viajes realizados.
b. Escribir lo que debe pagar.
c. Escribir lo que ha ahorrado.

CODIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define NumPeajes 15
#define NumClientes 10
main(){
  int viajes[NumClientes][NumPeajes];
  float precioPeajes[NumPeajes], debePagar[NumClientes];
  int viajesTotales[NumClientes],pasoPeajes[NumPeajes];
  float deberiaHaberPagado[NumClientes], ahorro[NumClientes];
  int c, p, maxPeaje;

//Inicializamos la matriz viajes a 0.
for (c=0; c<NumClientes; c=c+1)
{   for (p=0; p<NumPeajes; p=p+1)
    { viajes[c][p]=0; }
}
//Leer el precio de cada uno de los peajes.
for (p=0; p<NumPeajes; p=p+1)
{ printf(" Introduce el precio del peaje %d: ",p);
  scanf ("%f",&precioPeajes[p]);
}
//Introducción de datos.
//Inicializamos el vector viajesTotales y el vector debePagar a 0.
for (c=0; c<NumClientes; c=c+1)
{  viajesTotales[c]=0;
   debePagar[c]=0;
}
//Leemos y guardamos los viajes que ha hecho cada cliente.
printf(" Introduce el código del cliente y el peaje. "
       " (para terminar -1 -1):\n ");
printf(" Cliente,  Peaje: ");
scanf ("%d %d",&c,&p);
while (c >= 0 && p >= 0)
{  viajes[c][p]=viajes[c][p]+1;
   viajesTotales[c]=viajesTotales[c]+1;
```

```
   if (viajesTotales[c] <= 8 )
   { debePagar[c]=debePagar[c]+precioPeajes[p]*0.75;}
   else
   {  if (viajesTotales[c]<=20 )
      { debePagar[c]=debePagar[c]+precioPeajes[p]*0.45; }
      else
      { debePagar[c]=debePagar[c]+precioPeajes[p]*0.25; }
   }
  printf(" Cliente,  Peaje: ");
  scanf ("%d %d",&c,&p);
}
// Peaje más transitado.
// 1º calculamos el número de veces que se ha pasado por cada peaje
for (p=0; p<NumPeajes; p=p+1)
{  pasoPeajes[p]=0; //Inicializamos la posición a 0.
   for (c=0; c<NumClientes; c=c+1)
   {  pasoPeajes[p]= pasoPeajes[p] + viajes[c][p]; }
}
//Calculamos el peaje por el que más han pasado los clientes.
maxPeaje=0;
for (p=1; p<NumPeajes; p=p+1)
{  if (pasoPeajes[p]> pasoPeajes[maxPeaje] )
   { maxPeaje=p; }
}
printf(" El peaje más utilizado es el %d:\n", maxPeaje);

//Escribir el peaje que más utiliza cada uno de los clientes.
for (c=0; c<NumClientes; c=c+1)
{  maxPeaje=0;
   for (p=1; p<NumPeajes; p=p+1)
   {  if (viajes[c][p] > viajes[c][maxPeaje])
      {  maxPeaje=p; }
   }
   printf(" El peaje que más utiliza el cliente %d es "
          " el %d.\n ",c,maxPeaje);
}
```

```
/*Calcular el ahorro de este mes por ser cliente habitual.
Primero calculamos para cada cliente, cuánto debería haber pagado*/
for (c=0; c<NumClientes; c=c+1)
{  deberiaHaberPagado[c]=0;  //Inicializamos a 0.
   for (p=0; p<NumPeajes; p=p+1)
   { deberiaHaberPagado[c]=deberiaHaberPagado[c] +
                              precioPeajes[p]*viajes[c][p];
   }
}
//Restamos lo que deberían haber pagado con lo que han pagado.
for (c=0; c<NumClientes; c=c+1)
{ ahorro[c]= deberiaHaberPagado[c] - debePagar[c];
  //Escribimos cuánto ha ahorrado cada cliente}
  printf("El cliente %d ha ahorrado %5.2f euros.\n ",c,ahorro[c]);
}
//Escribir resultados.
printf("Cliente  Total Viajes  Importe  Ahorro \n");
for (c=0; c<NumClientes; c=c+1)
{ printf("%5d %10d %10.2f %8.2f \n",c,pasoPeajes[c],
          debePagar[c], ahorro[c]);
}
printf("\n\n");
system("PAUSE");
}
```

SALA DE JUEGOS

Una Sala de Juegos quiere que le realicemos un programa informático que le permita conocer en todo momento, cuál es la situación de los jugadores que tiene en la sala.

Para poder elaborar el programa nos informan de que, en un día pasan por la Sala de Juegos como máximo 1000 jugadores, y que en la sala hay 80 juegos.

El programa lo ejecutarán diariamente y, lo primero que le tiene que pedir el programa al usuario es, que introduzca para cada juego la cantidad, en euros, que se gana por cada punto, y estos datos se guardarán en un array que llamaremos *eurosJuego*. Después, aparecerá un menú que permitirá al usuario ir introduciendo para cada jugador los puntos que va obteniendo en cada juego en el que participa, así como calcular determinadas opciones que se detallan a continuación. Para ello a los jugadores cuando entran en la Sala de Juegos se les asigna un número del 0 al 999, y los juegos también están numerados del 0 al 79.

Ejemplo de lectura del array eurosJuego (datos introducidos por el usuario en negrita):

```
Dame la ganancia por puntos obtenido de cada juego:

    Juego nº0 : 6
    Juego nº1 : 1
    Juego nº3 : 4
    Juego nº3 : 6
    ............
    Juego nº29 : 15
    ............
    Juego nº79: 2
```

Ejemplo:

	0	*1*	*2*	*3*	.....	*29*	.....	*79*
eurosJuego	6	1	4	6	.....	15	.....	2

Al finalizar la entrada de estos datos el programa presentará en pantalla el siguiente menú. Al seleccionar una opción del menú, ésta se ejecutará dándonos el resultado y después volverá a aparecer el menú en pantalla, excepto al seleccionar la opción 5 ya que el programa terminará.

Ejemplo:

```
Menu sala de juegos:

    1. Introducir puntos de un jugador.
    2. Conocer los puntos que un jugador lleva conseguidos en
       un juego.
    3. Mostrar para un jugador los puntos conseguidos en cada
       juego que ha participado.
```

```
4.  Calcular los euros ganados por un jugador.
5.  Terminar.
Dame tu opción:
```

Al introducir cada una de las opciones se realizará lo indicado a continuación:

1. Introducir puntos de un jugador.

   El programa deberá pedir el nº del jugador y, después para cada juego en el que participe se introducirá el nº de juego y el nº de puntos obtenidos. Tener en cuenta al programar la entrada de datos que, un jugador puede sacar 0 puntos en un juego, y que si repite el juego los puntos se le tienen que acumular. Para finalizar la entrada de datos de un jugador se introducirá el par -1 -1. Hay que poder distinguir los juegos que no ha jugado un jugador y en los que ha conseguido 0 puntos.

   Programarlo de forma que en cualquier momento del día se puedan volver a introducir más datos para un mismo jugador.

   Ejemplo de ejecución*:*

```
Nº de jugador: 10
     Juego y puntos:  4   35
     Juego y puntos:  1   0
     Juego y puntos:  4   10
     Juego y puntos:  30  10
     Juego y puntos:  -1  -1
```

2. Conocer los puntos que un jugador lleva conseguidos en un juego.

   Ejemplo de ejecución*:*

```
Nº de jugador:   10
Nº de juego:      4

Total puntos obtenidos en este juego:   45
```

3. Mostrar para un jugador los puntos conseguidos en cada juego que ha participado.

   Ejemplo de ejecución:

```
Nº del jugador:   10
       Juego  1              Puntos : 0
       Juego  4              Puntos : 45
       Juego  30             Puntos : 10
```

4. Calcular los euros ganados por un jugador.

   Para poder calcularlo, se utilizará el array *eurosJuego* que se ha llenado al principio del día.

   Ejemplo de ejecución:

```
Nº de jugador:   10
Total euros ganados: 420
```

5. Terminar.

   Al finalizar el día, el usuario elegirá esta opción, que le permitirá terminar la ejecución del programa y además, el programa le deberá indicar la cantidad total de euros que la Sala de Juegos ha entregado ese día y número del juego que más euros ha entregado ese día.

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *MaxJugadores* = 1000. Es el número máximo de jugadores que pasan por la sala en un día.
2. *MaxJuegos* = 80. Es el número de juegos que hay en la sala.

Representación de las variables necesarias para la entrada

3. Necesitaremos una matriz de *MaxJugadores* x *MaxJuegos* posiciones, donde se almacenarán los puntos obtenidos por cada jugador en cada uno de los juegos en los que ha participado.

puntuacion	0	1	...	MaxJuegos-1
0				
1				
...				
MaxJugadores-1				

4. Necesitaremos un vector e*urosJuego***,** de *MaxJuegos* posiciones, para guardar en cada juego la cantidad, en euros, que se gana por cada punto que se obtenga en dicho juego.

	0	1	...	MaxJuegos-1
eurosJuego				

Representación de las variables necesarias para los resultados

5. Necesitamos un vector *entregadoJuego*, con *MaxJuegos* posiciones, para almacenar la cantidad total de euros que ha entregado la sala en cada uno de los juegos al finalizar el día.

	0	1	...	MaxJuegos-1
entregadoJuego				

Pasos a seguir en la resolución del problema

1. Llenar el vector *eurosJuego*.
2. Inicializar la matriz *puntuación* a –1, para poder distinguir los juegos no jugados de aquellos en los que si se ha jugado, pero que se ha obtenido 0 puntos.
3. Utilizar un bucle *do-while* para presentar en pantalla el menu con las 5 opciones que ofrece el programa, cada vez que se ejecute una de las opciones (de la 1 a la 4) se volverá a presentar el menú en pantalla y así sucesivamente hasta pulsar la opción 5. Terminar.
4. Para seleccionar la opción elegida por el usuario utilizar el *switch*.

   Opción 1. Introducir puntos de un jugador.

   a. Leer el número del jugador.

b. Leer un juego y sus puntos.

c. Mientras el juego y los puntos sean distintos de -1 hacer:

i. Si ya ha participado en ese juego entonces acumular los puntos en la matriz *puntuacion* sino introducir directamente los puntos en la matriz *puntuacion*.

ii. Leer siguiente juego y puntos de ese jugador.

Opción 2. Conocer los puntos que un jugador lleva conseguidos en un juego.

a. Leer número de jugador y de juego.

b. Si el valor que tiene la matriz *puntuacion*, en esta posición de jugador y juego, es distinto de –1 entonces indicar los puntos conseguidos, sino decir que no ha participado en el juego.

Opción 3. Mostrar para un jugador los puntos conseguidos en cada juego que ha participado.

a. Leer el número de jugador.

b. Para cada juego:

i. Si la puntuación del jugador en ese juego es distinta de –1 (ha jugado) entonces escribir la puntuación.

Opción 4. Calcular los euros ganados por un jugador.

a. Leer el número del jugador.

b. Para cada juego:

i. Si la puntuación es distinta de 0 entonces multiplicar los puntos, obtenidos en el juego, por los euros del vector *eurosJuego* y acumular el resultado en la variable *ganancia*.

c. Escribir el total ganado .

5. Al salir del menú (Opción 5), calcular los euros que ha entregado la sala en cada juego y el total de euros entregados ese día.

a. Para cada juego.

b. Recorrer todos los jugadores.

i. Si la puntuación es mayor que 0 entonces multiplicar la puntuación por los e*urosJuego* y acumular el resultado en el vector *entregadoJuego*.

c. Antes de cambiar de juego acumular en la variable *entregado* el total entregado en cada juego.

6. Obtener el juego que ha entregado más dinero, para ello hay que buscar la posición del máximo en el vector e*ntregadoJuego*.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define MaxJugadores 1000
#define MaxJuegos 80
main(){
 int puntuacion[MaxJugadores][MaxJuegos];
 float eurosJuego[MaxJuegos], entregadoJuego[MaxJuegos];
 int jugador, puntos, juego, opcion, jEntregaMax ;
 char continuar;
 float ganancia, entregado;


/* Leer ganancia por punto en cada juego y guardarlo en el vector
eurosJuego. */
printf(" Dame la ganancia por punto obtenido de cada juego:\n");
for (juego=0 ; juego< MaxJuegos; juego=juego+1)
{  printf(" Juego nº %d: ", juego);
   scanf("%f",&eurosJuego[juego]);
}
/* Inicializar la matriz Puntuación a -1 para poder distinguir
aquellos juegos jugados, y en los cuales se ha logrado 0 puntos, de
los que no se han jugado. */
for (jugador=0; jugador<MaxJugadores; jugador=jugador+1)
{  for (juego=0 ; juego< MaxJuegos; juego=juego+1)
   {     puntuacion[jugador][ juego]=-1; }
}
//Presentamos en pantalla el menú con las opciones.
do{
  system("CLS");  //limpiamos pantalla
  printf(" Las opciones del menú son: \n ");
  printf(" 1. Introducir puntos de un jugador. \n ");
  printf(" 2. Conocer los puntos que un jugador lleva "
             "conseguidos en un juego. \n ");
  printf(" 3. Mostrar para un jugador los puntos conseguidos en "
             "cada juego que ha participado. \n ");
  printf(" 4. Calcular los euros ganados por un jugador. \n ");
  printf(" 5. Terminar. \n ");
  printf(" Dame tu opción: ");
```

```
  scanf("%d",&opcion);
// Ejecuta la opción elegida.
  switch ( opcion )
  { case 1:
/* Introducir puntos de un jugador en cada uno de los juegos en los
que participa. */
      printf(" Nº de jugador: ");
      scanf("%d",&jugador);
      printf("    Juego y puntos: ");
      scanf("%d %d",&juego, &puntos);
      // Si el par leido es distinto de -1 -1 entra en el bucle.
      while (juego!=-1 && puntos!=-1)
      {
/* Si ya ha participado en este juego acumulamos los puntos. Si no,
si es la primera vez que participamos, introducimos directamente
los puntos*/
         if (puntuacion[jugador][juego]>=0 )
         { puntuacion[jugador][juego]=
                              puntuacion[jugador][juego]+puntos;
         }
         else {  puntuacion[jugador][juego]=puntos; }
      printf("    Juego y puntos: ");
      scanf("%d %d",&juego, &puntos);
      }  // fin del while
      break;  // fin de la opción 1.
   case 2:
/* Conocer los puntos que un jugador lleva conseguidos en un juego.
*/
      printf(" Nº de jugador: ");
      scanf("%d",&jugador);
      printf(" Nº de juego: ");
      scanf("%d",&juego);
/* Miramos el valor que tiene la tabla puntuación en esta posición
de jugador y juego. Si el valor es distinto de -1 indicamos los
puntos conseguidos si no, indicamos que no ha participado en ese
juego.*/
      if (puntuacion[jugador][juego]!= -1)
      { printf(" Total puntos obtenidos en este juego %d \n: ",
                  puntuacion[jugador][juego]);
```

```
      }
      else
      { printf(" El jugador %d no ha jugado al juego %d \n",
                 jugador,juego);
      }
      printf("\n");
      fflush(stdin);  //limpia el buffer de memoria
      printf(" Pulse cualquier tecla para continuar ...\n");
      scanf("%c",&continuar);
      break; //fin de la opción 2.
   case 3:
/* Mostrar para un jugador los puntos conseguidos en cada juego que
ha participado.*/
      printf(" Nº de jugador: ");
      scanf("%d",&jugador);
/* Para este jugador recorremos, en la matriz puntuación, todos los
juegos, y mostramos los puntos sólo si el valor es distinto de -1,
es decir, si ha jugado.*/
      for (juego=0; juego<MaxJuegos; juego=juego+1)
      { if (puntuacion[jugador][juego]!=-1 )
        { printf(" Juego: %d puntos: %d \n",juego,
                    puntuacion[jugador][juego]);
        }
      } // fin del for
      printf("\n");
      fflush(stdin);  //limpia el buffer de memoria
      printf(" Pulse cualquier tecla para continuar ...\n");
      scanf("%c",&continuar);
      break; //fin de la opción 3
   case 4:
      // Calcular los euros ganados por un jugador.
      printf(" Nº de jugador: ");
      scanf("%d",&jugador);
      ganancia=0;
      for (juego=0; juego<MaxJuegos; juego=juego+1)
      { if (puntuacion[jugador][juego]>=0 )
        { ganancia=ganancia+puntuacion[jugador][juego]*
                                          eurosJuego[juego];
```

```
         }
      } // fin del for.
      printf(" Total euros ganandos: %5.2f \n ", ganancia);
      printf("\n");
      fflush(stdin);  //limpia el buffer de memoria
      printf(" Pulse cualquier tecla para continuar ...\n");
      scanf("%c",&continuar);
      break; //fin de la opción 4
   case 5: break; //fin de la opción 4
   default: printf(" Opción incorrecta ");
      printf("\n");
      fflush(stdin);  //limpia el buffer de memoria
      printf(" Pulse cualquier tecla para continuar ...\n");
      scanf("%c",&continuar);
  }  //fin del switch
}while (opcion!=5);


/* Salimos del menú al finalizar el día. Calculamos la cantidad
total de euros entregados por la sala y el juego que mayor cantidad
de euros ha entregado este día. */
entregado=0;   /* en esta variable se acumulan el total de euros
                  obtenidos en los juego, durante este día */
for (juego=0; juego<MaxJuegos; juego=juego+1)
{  entregadoJuego[juego]=0;
   for (jugador=0; jugador<MaxJugadores; jugador=jugador+1)
   { if (puntuacion[jugador][juego]>0 )
     {  entregadoJuego[juego]=entregadoJuego[juego]+
                    puntuacion[jugador][juego]*eurosJuego[juego];
      }
    } // for anidado
    entregado=entregado+entregadoJuego[juego];
}  //for principal
printf("\n");
printf(" Total dinero entregado : %8.2f euros \n ", entregado);
printf("\n");


/* Calculamos el juego que ha entregado más dinero, para ello hay
que buscar la posición del máximo en el vector entregadoJuego */
```

```
jEntregaMax=0;
for (juego=1; juego<MaxJuegos; juego=juego+1)
{ if (entregadoJuego[jEntregaMax]<entregadoJuego[juego] )
  {  jEntregaMax=juego; }
}
printf("El juego que ha entregado mas dinero es el: %d con: %8.2f"
       " euros.\n", jEntregaMax, entregadoJuego[jEntregaMax]);
printf("\n\n");
system("PAUSE");
}
```

FAROLAS

Desde el ayuntamiento nos han pedido que hagamos un programa que les ayude a gestionar las farolas de cierta zona de la ciudad. Esta zona de la ciudad tiene 5 calles, y en cada una de ellas hay 15 farolas, numeradas del 0 al 14.

Con el fin de ahorrar, el ayuntamiento ha decidido que no se van a encender todas las farolas de cada calle, sino que en cada calle habrá un número máximo de farolas que pueden estar encendidas simultáneamente. (Nosotros almacenaremos estos máximos en un array llamado *maximoCalle*)

El programa que nos piden debe realizar las siguientes tareas:

1. Leer el nº máximo de farolas que pueden estar encendidas simultáneamente en cada calle. Ejemplo: (en negrita los datos introducidos por el usuario):

```
Introduce el nº máximo de farolas a encender de la calle 0: 2
Introduce el nº máximo de farolas a encender de la calle 1: 3
...
Introduce el nº máximo de farolas a encender de la calle 4: 15
```

2. Encender farolas. Avisar en caso de que se intente exceder el máximo permitido para esa calle. Ejemplo:

```
Introduce calle y farola (-1 -1 para terminar): 0 4
Introduce calle y farola (-1 -1 para terminar): 0 3
Introduce calle y farola (-1 -1 para terminar): 0 14
¡Error!En la calle 0 no puede haber más de 2 farolas encendidas.
Introduce calle y farola (-1 -1 para terminar): 2 4
Introduce calle y farola (-1 -1 para terminar): 4 11
Introduce calle y farola (-1 -1 para terminar):-1 -1
```

3. Obtener el nº de farolas encendidas en cada calle y el máximo para esa calle. Ejemplo:

```
En la calle 0 hay 2 farolas encendidas de un máximo de 2.
En la calle 1 hay 0 farolas encendidas de un máximo de 3.
...
En la calle 4 hay 1 farolas encendidas de un máximo de 15.
```

4. Para cada calle indica si es más seguro ir por el lado de la derecha o el de la izquierda. Para ello, consideraremos más seguro aquel lado que tenga más farolas encendidas. En caso de que el número de farolas encendidas en cada lado coincida, o estén todas apagadas, diremos que los dos lados son igual de seguros. Supón que las farolas con número par están en el lado derecho y las de número impar en el izquierdo. Ejemplo:

```
En la calle 0 da igual ir por la derecha o por la izquierda.
En la calle 1 da igual ir por la derecha o por la izquierda.
En la calle 2 es más seguro ir por la derecha.
```

```
...
```

5. Obtener el número de calles que tienen todas las farolas apagadas. Ejemplo:

```
El número de calles con todas las farolas apagadas es 2.
```

6. Clasificar las calles según los siguientes criterios:

   - Todas las farolas apagadas: **'A oscuras'.**
   - Si hay encendidas tantas como indica su máximo de calle y hay más encendidas que apagadas: **'Muy Luminosa'.**
   - Si hay encendidas tantas como indica su máximo de calle, pero hay más o igual nº de apagadas que encendidas: **'Luminosa'.**
   - En cualquier otro caso: **'Tenebrosa'.**

   Ejemplo:

```
La calle 0 es luminosa.
La calle 1 está a oscuras.
La calle 2 es tenebrosa.
...
```

7. Apagar todas las farolas.

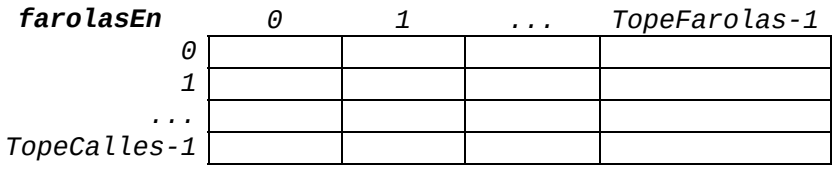
ANÁLISIS DEL PROBLEMA

Constantes del enunciado

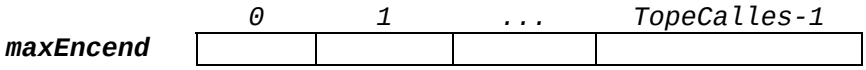
1. *TopeCalles* = 5. Número de calles que hay en la zona a estudiar.
2. *TopeFarolas* =15. Número de farolas que hay en cada calle.

Representación de las variables necesarias para la entrada

3. Necesitaremos la matriz *farolasEn,* de *TopeCalles* x *TopeFarolas* posiciones, donde se guardará un 0 para las farolas que estan apagadas y un 1 para las que están encendidas.

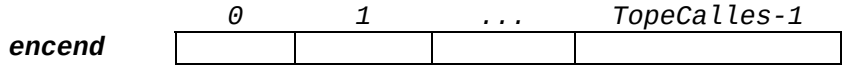


4. Necesitaremos el vector *maxEncen*, de *TopeCalles* posiciones, para guardar el número máximo de farolas que pueden estar encendidas simultaneamente en cada calle.



Representación de las variables necesarias para los resultados

5. Necesitamos el vector e*ncend*, de *TopeCalles* posiciones, para ir contando las farolas que se van encendiendo en cada calle y así no pasarnos del máximo permitido.



Pasos a seguir en la resolución del problema

1. Llenar el vector *maxEncend.*
2. Poner la matriz *farolasEn* a 0 (todas apagadas).
3. Poner a 0 el vector de farolas encendidas en cada calle (*encend*).
4. Encender farolas. Pedir la calle y la farola.
5. Mientras el par sea distinto de -1 hacer :
   a. Si el número de farolas encendidas en esa calle es menor que el máximo permitido entonces, encender la farola (un 1 en la matriz) y contarla como encendida, sino dar un mensaje de error (no se puede encender).
   b. Leer el siguiente par calle y farola.
6. Para cada calle indicar, el número de farolas encendidas y el máximo de esa calle.

7. Seguridad en las calles. Para cada calle hacer :
    a. Inicializar los contadores de farolas encendidas a la derecha y a la izquierda a 0.
    b. Para cada farola de esa calle hacer:
        i. Si el número de farola es impar y está encendida incrementar el contador de farolas a la izquierda , sino, si el número de farola es par y está encendida incrementar el contador de farolas a la derecha.
    c. Una vez recorridas todas las farolas de esa calle, comparar los contadores e indicar cómo es la calle.
8. Número de calles con todas las farolas apagadas. Inicializar el contador *todas* a 0. Para cada calle hacer:
    a. Si en el vector *encend* para esa calle el valor es 0, entonces incrementar el contador *todas*.
9. Clasificar las calles. Para cada calle hacer:
    a. Si el número de farolas encendidas es igual al máximo del la calle y el número de encendidas es mayor que el de apagadas entonces indicar que es muy luminosa.
    b. Si no, si el número de farolas encendidas es igual al máximo de la calle pero el número de encendidas es menor que el de apagadas entonces indicar que es luminosa.
    c. Si no, si están todas apagadas entonces indicar que está a oscuras.
    d. Y si no, indicar que es tenebrosa.
10. Apagar las farolas, es decir, poner la matriz *farolasEn* a 0.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define TopeFarolas 15
#define TopeCalles 5
main (){
  int farolasEn[TopeCalles][TopeFarolas];
  int maxEncend[TopeCalles], encend[TopeCalles];
  int f,c,izquierda,derecha,todas;


/* 1.Leemos el máximo número de farolas que se pueden encender en
cada calle.*/
for (c=0; c<TopeCalles; c=c+1)
{ printf(" Introduzca el máximo posibles de farolas encendidas "
         " de la calle %d: ",c);
   scanf("%d",&maxEncend[c]);
}
/* 2.Encender farolas. Partimos de que todas las farolas están
apagadas, es decir, ponemos la matriz farolasEn a 0. */
for (c=0; c<TopeCalles; c=c+1)
{   for (f=0; f<TopeFarolas; f=f+1)
    {  farolasEn[c][f]=0; }
}
/* Inicializamos a 0 el contador de farolas encendidas en cada
calle. */
for (c=0; c<TopeCalles; c=c+1)
{   encend[c]=0;  }
/* Encendemos las farolas deseadas, contando en el vector encend
las farolas que se van encendiendo en cada calle, para así poder
avisar si se pasan del máximo permitido. */
printf(" Introduzca calle y farola (-1 -1 para terminar): ");
scanf("%d %d",&c,&f);
while (c!=-1 && f!=-1)
{  if (encend[c]<maxEncend[c])
   {  farolasEn[c][f]=1;  //encendemos la farola
      encend[c]= encend[c]+1;
   }
   else
```

```
   {  printf("¡¡Error!! En la calle %d no puede haber más de %d
\n");
   printf(" farolas encendidas. \n",c, maxEncend[c]);
    }
   printf(" Introduzca calle y farola (-1 -1 para terminar): ");
   scanf("%d %d",&c,&f);
}  // fin del while


/* 3. Nº de farolas encendidas en cada calle y el máximo para esa
calle. */
for (c=0; c<TopeCalles; c=c+1)
{  printf(" En la calle %d hay %d farolas endendidas de un máximo"
          " de %d. \n", c, encend[c], maxEncend[c]);
}


//4. Seguridad en las calles.
for (c=0; c<TopeCalles; c=c+1)
{  izquierda=0;  derecha=0;
   for (f=0; f<TopeFarolas; f=f+1)
   { if (f%2!=0 && farolasEn[c][f]==1)
     { izquierda=izquierda+1; }
     else
     { if (f%2==0 && farolasEn[c][f]==1)
       { derecha=derecha+1; }
     }
   } //for anidado
   if (izquierda==derecha )
   {  printf(" En la calle %d da igual ir por la derecha o"
             " por la izquierda.\n", c);
   }
   else
   { if (izquierda > derecha )
     { printf(" En la calle %d es más seguro ir por la "
                 " izquierda.\n", c);
     }
     else
     { printf(" en la calle %d es más seguro ir por la "
```

```
                        " derecha.\n", c);
     }
   }
} //for principal
// 5. Número de calles con todas las farolas apagadas.
todas=0;
for (c=0; c<TopeCalles; c=c+1)
{ if (encend[c]==0 )
   { todas=todas+1; }
}
printf(" El número de calles con todas las farolas apagadas "
       " es %d.\n",todas);
//6. Clasificar las calles.
for (c=0; c<TopeCalles; c=c+1)
{  if (encend[c]==maxEncend[c] && encend[c]>TopeFarolas-encend[c])
   {  printf(" La calle %d es muy luminosa.\n",c); }
   else
   { if (encend[c]==maxEncend[c] &&
                         encend[c]<TopeFarolas-encend[c])
     {  printf(" La calle %d es   luminosa.\n"); }
     else
     {   if (encend[c]==0)
         {  printf(" La calle %d está a oscuras.\n",c); }
         else
         { printf(" La calle %d es tenebrosa.\n",c); }
     }
   }
} // fin del for
// 7- Apagar las farolas.
for (c=0; c<TopeCalles; c=c+1)
{   for (f=0; f<TopeFarolas; f=f+1)
   {   farolasEn[c][f]=0; }
}
system("PAUSE");
}
```

PUEBLOS DE GUIPÚZCOA

La Diputación nos ha dicho que en Guipúzcoa hay 85 pueblos y que nos va a proporcionar las distancias en kilómetros, que hay entre cada uno de los pueblos. También nos ha especificado cómo va a ser la entrada de datos de nuestro programa. El formato será el siguiente:

El programa solicitará al usuario la distancia desde un pueblo a todos los demás. Hay que tener en cuenta, que cada una de las combinaciones se pedirá una sola vez y que no se pedirá la distancia de un pueblo consigo mismo. Esto es, el programa **no** pedirá datos como estos:

- Si ha preguntado la distancia del pueblo 0 al pueblo 2, no se va a preguntar la distancia del pueblo 2 al pueblo 0.
- Tampoco se va a pedir, la distancia desde el pueblo 1 al pueblo 1.

Tenidas en cuenta estas restricciones la entrada de datos será así :

Ejemplo: (en negrita los datos introducidos por el usuario)

```
Lectura de la distancia entre los pueblos de Guipúzcoa:

Introduce la distancia desde el pueblo 0 a los siguientes
pueblos (en km):
    Pueblo 1: 25
    Pueblo 2: 68
    ....
    Pueblo 84:12

Introduce la distancia desde el pueblo 1 a los siguientes
pueblos (en km)):
    Pueblo 2: 105
    ....
    Pueblo 84: 26

.............................................................
```

Una vez hecha la lectura de datos, se presentará al usuario el siguiente Menú.

Ejemplo:

```
Menú del análisis de pueblos:

1. Ver tabla de distancias entre pueblos.
2. Mostrar qué 2 pueblos son los más alejados entre sí.
3. Dado un pueblo, mostrar cuál es el más lejano.
4. Mostrar el pueblo más céntrico.
5. Calcular el número de kilómetros que hace el cartero.
6. Salir.
Elige una opción:
```

Y una vez realizados los cálculos correspondientes a la opción elegida por el usuario, se volverá a presentar el Menú, excepto con la opción 6.

A continuación se explica qué es lo que hay que realizar en cada una de las opciones del Menú.

1. Escribir por pantalla los datos en forma de tabla. Ejemplo:

	0	1	2	3	...	83	84
0	—	*25*	*68*			*134*	*12.3*
1	*25*	—	*105.5*				*26.5*
2	*68*	*105.5*	—				
3				—			
...					—		
83	*134*					—	
84	*12.3*	*26.5*					—

2. ¿Cuáles son los pueblos más alejados el uno del otro?. Ejemplo:

```
Los dos pueblos más alejados entre sí son el 0 y el 83.
```

3. Dado un pueblo, ¿qué pueblo es el más lejano?. Ejemplo:

```
Introduce el número de un pueblo: 18
El pueblo más alejado del 18 es el 7.
```

4. ¿Cuál es el pueblo más céntrico?. (El pueblo más céntrico será aquel que sumadas las distancias desde ese pueblo a todos los demás, presente un valor más pequeño). Ejemplo:

```
El pueblo más céntrico es el 27.
```

5. Calcular los kilómetros que hace el cartero. Para ello, el usuario introducirá el pueblo en el que se encuentra, y luego la secuencia de los pueblos que recorre (terminando la secuencia con un -1) y el programa calculará el total de kilómetros. Suponemos que los pueblos se visitan en el orden en el que son introducidos. Ejemplo:

```
Introduce el pueblo en el que se encuentra el cartero: 25
Introduce los pueblos que recorre el cartero: 12  34  2  11  -1
El cartero realiza 356 km.
```

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *TopePueblos* = 85. Número de pueblos que hay en Guipúzcoa.

Representación de las variables necesarias para la entrada

2. Necesitaremos la matriz *distancia,* de *TopePueblos* X *TopePueblos* posiciones, donde se guardarán las distancias, en kilómetros, que hay entre cada uno de los pueblos.

distancia	*0*	*1*	*...*	*TopePueblos-1*
0				
1				
...				
TopePueblos-1				

Representación de las variables necesarias para los resultados

3. Necesitaremos el vector s*umaDistancias,* de *TopePueblos* posiciones, para guardar en él la suma de distancias de cada pueblo al resto de pueblos y así poder determinar cuál es el pueblo más céntrico.

	0	*1*	*...*	*TopePueblos-1*
sumaDistancias				

Pasos a seguir en la resolución del problema

1. Inicializar la diagonal de la matriz *distancia* a 0.
2. Llenar la matriz *distancia*, para ello solo es necesario pedir al usuario que introduzca los datos correspondientes a las celdas que están por encima de la diagonal y las celdas que están debajo de la diagonal se llenarán mediante programa. Así por ejemplo, al leer la distancia del pueblo 1 al 2 el programa deberá llenar tanto la celda [1], [2] como la [2],[1], al leer la distancia del pueblo 1 al 3 se llenarán las celdas [1],[3] y [3],[1] y así sucesivamente.
3. Utilizar un bucle *do- while* para presentar en pantalla el menú con las 6 opciones que ofrece el programa, cada vez que se ejecute una de las opciones (de la 1 a la 5) se volverá a presentarse el menú en pantalla y así sucesivamente hasta pulsar la opción 6. Salir.
4. Para seleccionar la opción elegida por el usuario utilizar el *switch.*

   Opción 1. Ver la tabla de distancias entre pueblos.

   a. Recorrer la matriz *distancia* con dos bucles for anidados. Si el pueblo origen y el destino son el mismo entonces escribir un guión, si no escribir la distancia.

   Opción 2. Mostrar los dos pueblos más alejados entre si.

a. Recorrer la matriz *distancia* buscando la distancia máxima e indicar a qué dos pueblos pertenece.

Opción 3. Dado un pueblo buscar el más alejado.

a. Pedir el pueblo origen.

b. Buscar, en la fila correspondiente a ese pueblo origen, la distancia máxima. De esta forma obtenemos el pueblo más alejado.

Opción 4. Buscar el pueblo más céntrico.

a. Crear el vector *sumaDistancias* en el que se suma la distancia de cada pueblo al resto de pueblos.

b. Buscar en el vector s*umaDistancia* la distancia más pequeña, que será la correspondiente al pueblo más céntrico.

Opción 5. Kilómetros que hace el cartero.

a. Leer el pueblo en el que está el cartero (*pO*)

b. Leer el pueblo destino (*pD*), e inicializar a *0* la variable *sumaKm*.

c. Mientras el pueblo destino sea distinto de *-1* acumular la distancia recorrida en s*umaKm*, hacer que el pueblo origen sea el pueblo en el que está actualmente el cartero y leer el siguiente pueblo destino.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define TopePueblos 85
main(){
  int distancia[TopePueblos][TopePueblos];
  int sumaDistancias[TopePueblos];
  int d,po,pd, opc;
  int masLejosO,masLejosD,centrico,sumaKm;
      /*d= distancia,  pO= pueblo origen,  pD= pueblo destino,
opc= opción del menú */
// Inicializar la diagonal de la matriz distancias a 0.
for (po=0; po<TopePueblos ;po=po+1)
{  distancia[po][po]=0;}
// Entrada de datos.
printf(" Lectura de la distancia entre los pueblos de "
       " Guipúzcoa: \n");
for (po=0; po<TopePueblos-1 ;po=po+1)
{   printf(" Introduce la distancia desde el pueblo %d \n",po);
      printf(" a los siguientes pueblos (en km):\n ");
   for (pd=po+1; po<TopePueblos ; po=po+1)
   {  printf("Pueblo %d : ",pd);
      scanf("%d",&d);
      distancia[po][pd]=d;
      distancia[pd][po]=d;
   }
}
do{
   printf(" Menú del análisis de pueblos.\n");
   printf("1.Ver tabla de distancias entre pueblos.\n");
   printf("2.Mostrar los 2 pueblos más alejados entre sí.\n");
   printf("3.Dado un pueblo, mostrar cuál es el más lejano.\n");
   printf("4.Mostrar el pueblo más céntrico.\n");
   printf("5.Calcular el nº de kilómetros que hace el cartero.\n");
   printf("6. Salir\n");
   printf("Elige una opción : ");
   scanf("%d",&opc);
```

```
   switch (opc )
   { case 1:
      //Salida por pantalla de la tabla de distancias entre pueblos.
      printf("   |");
      for (pd=0 ;pd < TopePueblos; pd=pd+1)
      {    printf("%4d",pd); }
      printf("\n"); //salto de línea
      printf("   |");
      for (pd=0 ;pd< TopePueblos ; pd=pd+1)
      {  printf("----"); }
      printf("\n"); //salto de línea
      for (po=0 ;po< TopePueblos ;po=po+1)
      { printf("%3d|",pd);
        for (pd=0 ; pd< TopePueblos ; pd=pd+1)
        { if (po==pd)
          { printf("%4s","-"); }
          else
          { printf("%4d ",distancia[po][pd]);}
        }  //for anidado
      printf("\n"); //salto de línea
      }  //for principal
     break; //opción 1
  case 2:  //Los dos pueblos más alejados el uno del otro.
      masLejosO=0; masLejosD=1;
      for (po=0; po < TopePueblos-1 ; po=po+1)
      {  for (pd=po+1 ;pd< TopePueblos ; pd=pd+1)
         { if (distancia[po][pd]>distancia[masLejosO][masLejosD])
           {  masLejosO=po;
              masLejosD=pd;
           } // fin del if
         }   //fin del for anidado
      }  //fin del for principal
printf("Los dos pueblos más alejados entre sí son "
       " el %d y el %d\n", masLejosO, masLejosD);
      break; //opción 2
   case 3:
```

```
     /* El pueblo destino más alejado del pueblo origen introducido
por el usuario. */
      printf("Introduce el número de  un pueblo: ");
      scanf("%d",&po);
      masLejosD=0;
      for (pd=1 ; pd< TopePueblos ; pd=pd+1)
      {  if (distancia[po][pd]>distancia[po][masLejosD] )
         {  masLejosD =pd;  }
      }
      printf("El pueblo más alajado del %d es %d\n",po,masLejosD);
      break;  //opción 3
  case 4:
     /* Buscamos el pueblo más céntrico. Primero sumamos la
distancia de cada pueblo al resto de pueblos */
      for (po=0; po< TopePueblos ; po=po+1)
      { sumaDistancias[po]=0;
        for (pd=0 ; pd< TopePueblos ; pd=pd+1)
        { sumaDistancias[po]=sumaDistancias[po]+distancia[po][pd];}
      }
      /* Buscamos el pueblo más céntrico, que será el que tenga la
suma de distancias más pequeña. */
      centrico=0;
      for (po=1; po< TopePueblos ; po=po+1)
      {   if (sumaDistancias[po]<sumaDistancias[centrico] )
          { centrico=po;  }
      }
      printf(" El pueblo más céntrico es el %d\n",centrico);
      break; //opción 4
   case 5: // kilómetros que hace el cartero
      printf(" Introduce el pueblo en el que está el cartero : ");
      scanf("%d",&po);
      printf(" Introduce los pueblos que recorre el cartero : ");
      scanf("%d",&pd);
      sumaKm=0;
      while (pd!=-1)
      {  sumaKm= sumaKm + distancia[po][pd];
         po=pd;
         scanf("%d",&pd);
```

```
      }
       printf(" El cartero realiza %d km.\n", sumaKm);
       } // fin del switch
   system("PAUSE"); //pausa hasta pulsar una tecla
   system("CLS");  //limpiar  pantalla
}while (opc!=6);  //fin del bucle

printf(" Fin del programa ");
system("PAUSE"); //pausa hasta pulsar una tecla
}
```

OLIMPIADAS

El Comité Olímpico Internacional nos ha solicitado que le realicemos un programa informático. Dicho programa, debe efectuar ciertos cálculos manipulando datos del volumen de participantes por país en diversas disciplinas olímpicas, de las olimpiadas Beijing'08, así como con los datos de la anterior edición, Atenas'04.

Olimpiadas Beijing'08 (Juegos Olímpicos de Pekín, China)

El subcomité de Pekín nos ha indicado que han intervenido deportistas de 203 países. Y aunque el número de disciplinas olímpicas reconocidas es de 37, no todos los países han tenido representación en todas y cada una de las mismas. Por lo tanto, el subcomité de Pekín únicamente nos va a proporcionar los datos de aquellas disciplinas deportivas, ordenados por países, en las que ha habido representación. El programa a elaborar deberá registrar los datos tal y como se muestra en el ejemplo siguiente.

Ejemplo de la entrada de datos de Pekin (*en **negrita** los datos introducidos por el usuario*):

```
Introduce los participantes de cada país en cada disciplina:
País 0:
   Disciplina y participantes: 3  100
   Disciplina y participantes: 6  23
   Disciplina y participantes:-1  1

País 1:
   Disciplina y participantes: 1  13
   Disciplina y participantes: 6  24
   Disciplina y participantes:-1  0
    ……….
    ……..
País 202:
   Disciplina y participantes: 12  13
   Disciplina y participantes: 34  24
   Disciplina y participantes: -1  3
```

Para indicar la finalización de la entrada de datos de un país se tecleará un -1 como número de disciplina sin importar el número de participantes.

Olimpiadas Atenas'2004 (Juegos Olímpicos de Atenas, Grecia)

El subcomité de Atenas nos ha indicado que igualmente en la edición del 2004 participaron 203 países, aunque no disponen de la distribución de los participantes por disciplinas sino únicamente disponen del volumen total de participantes por país. Eso sí, nos pueden suministrar dichas cifras siguiendo el mismo orden de países que nos ha proporcionado el subcomité olímpico de Pekín. Siguiendo las especificaciones indicadas, los datos de participación

en los juegos olímpicos de Atenas se solicitarán y almacenarán de la siguiente forma:

Ejemplo de la entrada de datos de Atenas: *(en **negrita** los datos introducidos por el usuario):*

```
Introducción datos Atenas 2004:
   Participantes del país 0:240
   Participantes del país 1:2402

   Participantes del país 202:3500
```

Una vez solicitados y almacenados los datos de participación por países de las últimas dos ediciones, el comité olímpico nos solicita que nuestro programa calcule:

Respecto a las Olimpiadas 2008 en Pekín:

1. Indicar cuál es, a nivel mundial, la disciplina deportiva con menor número de participantes y cuántos son. Ejemplo:

   ```
   La disciplina con menor número de participantes de estas
   Olimpiadas ha sido la 25, concretamente con 2 participantes.
   ```

2. Por cada país especificado (tecleado) por el usuario, se calculará la disciplina deportiva con mayor número de participantes. La solicitud del país y el consecuente cómputo se realizará hasta que el usuario introduzca un -1 como país. Además se controlará que el código del país pertenezca al rango [0 .. 202]. Ejemplo:

   ```
   País: 2
          Más deportistas en el deporte 19
   País: 302
          Código de país incorrecto. Especifique valor entre 0
          y 202 ó -1 para finalizar.
   País: 5
          Más deportistas en el deporte 1
   País: -1
   ```

En comparación con las Olimpiadas de Atenas 2004:

3. Listados de los países que han aumentado el volumen de participantes desde la edición anterior. Ejemplo:

   ```
   3   4   14 son los países con incremento en el volumen de
   participantes.

   En total, son 3 países.
   ```

   Si no hay ningún país que haya incrementado su número de participantes en Beijing'08 con respecto a Atenas'04, el mensaje a presentar en pantalla deberá adecuarse al siguiente:

   ```
   Ningún país ha incrementado su volumen de participantes
   desde la edición del 2004.
   ```

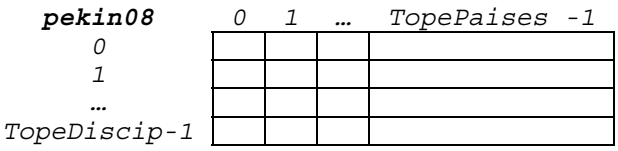
ANÁLISIS DEL PROBLEMA

Constantes del enunciado

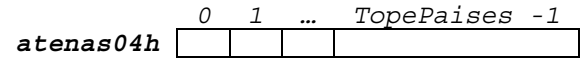
1. *TopeDiscip* = 37. Número de disciplinas olímpicas reconocidas.
2. *TopePaises = 203*. Número total de paises que intervienen en las olimpiadas .

Representación de las variables necesarias para la entrada

3. Necesitamos la matriz *pekin08* que tiene *TopeDiscip* X *Tope Paises* posiciones, donde se guardará por cada disciplina y país el número de participantes.

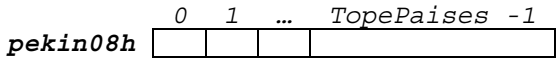


4. Necesitaremos un vector que le llamaremos *atenas04h* con *TopePaises* posiciones donde almacenaremos por cada país el número de participantes en los Juegos Olímpicas de Atenas de 2004.

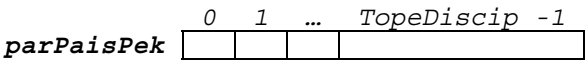


Representación de las variables necesarias para los resultados

5. Necesitamos un vector que le llamaremos *pekin08h* de *TopePaises* posiciones. En él guardaremos por cada país el total de participantes



6. Necesitaremos un vector que le llamaremos *parPaisPek* con *TopeDiscip* posiciones donde se guardará por cada disciplina el número de participantes.



Pasos a seguir en la resolución del problema

1. Pondremos a 0 la matriz *pekin08*, ya que puede haber paises y disciplinas en las que no hay participantes.
2. Rellenaremos la matriz *pekin08* con los datos que iremos pidiendo por pantalla. Se irá solicitando país a país de manera ordenada y por cada país la disciplina y el número de participantes, este par de datos no vendrá ordenado y para saber cuando se terminará de introducir datos de un país se introducirá un -1 en la disciplina

3. Para cada país se leerá el número de participantes que ha ha bido y se rellenará el vector *atenas04h*.
4. Para cada disciplina:
    a. Se pondrá una variable *sum* a 0 .
    b. Se acumulará recorriendo todos los países en sum el número de participantes.
    c. Cada vez que cambie de disciplina se almacenará en la posición correspondiente del vector *parPaisPek*.
5. Se calculará que valor es el mínimo del vector *parDisPek* (posición).
6. Se pedirá un número de país.
7. Y hasta que se teclee un -1 en el número de país,
    a. Se comprobará que el número de país esté en el rango ($0 \leq país \leq 203$)
    b. Para ese país se buscará en la columna correspondiente de la matriz *pekin08* la disciplina con menor número de participantes.
8. Se calculará en el vector *pekin08h* el número de participantes por cada país y se irá comparando cada elemento del vector con cada elemento del vector *atenas04h* para ir obteniendo los paises en los que ha aumentado el número de participantes. Utilizaremos una variable de nombre *tot* que la iniciaremos a 0, cada vez que encontremos un país que supera en el número de participantes sumaremos 1, de esta manera al final se podrá saber si no ha habido paises que han superado en el número de participantes.

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define TopeDiscip 37
#define TopePaises 203
main(){
  int pekin08[TopeDiscip][ TopePaises];
  int pekin08h[TopePaises], atenas04h[TopePaises];
  int parPaisPek [TopeDiscip];
  int pais, dep, pa, min, mejor, sum, tot;

/* Poner la matriz a o*/
for (pais =0; pais < TopePaises; pais++)
  for (dep =0; dep < TopeDiscip; dep ++)
    pekin08[dep][ pais]=0;

/* Registrar los datos de Pekin*/
printf("Introduce los datos de cada país en cada disciplina:\n");

for (pais =0; pais < TopePaises; pais ++)
{  printf(" País %d: \n", pais);
   printf("Disciplina y participante : ");
   scanf ("%d %d",& dep,&pa);
   while (dep!=-1)
   {  pekin08[dep][pais]=pa;
      printf("Disciplina y participante: ");
      scanf ("%d %d",& dep,&pa);
   }
}

printf("Introducción datos Atenas 2004:\n");
for (pais =0; pais < TopePaises; pais ++)
{   printf("      Participantes del país %d : ", pais);
   scanf ("%d",&atenas04h[pais]);
   // o    scanf("%d",&pa); atenas04h[pais]=pa;
}
```

```
/* Cálculo de la disciplina con menor número de participantes*/
for (dep =0; dep < TopePaises; dep ++)
{ sum=0;
  for (pais =0; pais <TopePaises; pais ++){sum=sum + pekin08[dep][
pais];}
  parPaisPek [dep] = sum;
}
min=0;
for (dep =1; dep < TopeDiscip; dep ++)
{  if (parPaisPek [dep] <
       parPaisPek [min])
   { min= dep; }
}


printf("La disciplina con menor número de participantes de\n");
 printf("estas Olimpiadas ha sido la %d concretamente con \n",min);
 printf("%d participantes \n", parPaisPek [min]);


/* Por cada país especificado (tecleado) por el usuario, se
calculará la disciplina deportiva con mayor número de
participantes, para finalizar pulsar -1 en el país */
printf("País: "); scanf("%d",& pais);
while (pais!=-1)
{  if ((0<= pais) && (pais <= TopePaises))/*Disciplina mejor?*/
   {  mejor=0;   /*En principio la 0.a */
      for (pais=1; pais< TopeDiscip; pais++)
      {  if (pekin08[dep][ pais] >
             pekin08[mejor][pais]){mejor= dep;} }
      printf ("Más deportistas en el deporte: %d\n",mejor);
   }
   else /*entrada errónea*/
   { printf("Código de país erróneo. Especifique valor entre 0 "
             " y 202 ó -1 para finalizar.\n");
   }


   printf("País: ");
   scanf("%d", & pais);
```

```
}
printf("\n\n");


/* Pekin vs Atenas*/
for (pais =0; pais < TopePaises; pais++)
{  sum=0;
   for (dep =1; dep< TopeDiscip; dep ++) {sum=sum + pekin08[dep][
pais];}
   pekin08h[pais]= sum;
}


/*  países que han aumentado el volumen de participantes desde la
edición anterior */
tot=0;
for (pais =0; pais < TopePaises; pais ++)
{ if (pekin08h[pais] > atenas04h[pais]){ printf ("%5d", pais);
tot++; } }
if (tot>0)
{ printf ("son los países con incremento en el volumen de"
"participantes .\n");
         printf ( " En total son  %d paises\n", tot);
}
else
{ printf("Ningún país ha incrementado su volumen de
participantes\n");
 printf("desde la edición del 2004 .\n");
}
printf("\n\n");
system("PAUSE");
}
```

VENTAS S. A.

La empresa Ventas S.A. quiere analizar cuál es su mejor zona de ventas. Para ello, nos pide un programa informático que le permita recoger y analizar los datos de las ventas que hacen sus 10 vendedores en las 20 zonas en las que trabajan.

La empresa vende 5 productos que tienen diferentes precios. Se deberá declarar en el programa un array constante, que contenga los precios de dichos productos, para poder calcular posteriormente el importe de las ventas. El array será el siguiente:

	0	1	2	3	4
precioProducto	20.5	50	15	5.25	150

En la entrada de datos el usuario introducirá para cada vendedor y zona lo siguiente: qué productos ha vendido y cuántas unidades de cada uno de ellos. De la siguiente forma:

```
Introduce nº de vendedor y nº de zona : 2  4
  Introduce el nº del producto vendido y nº de unidades: 2   10
  Introduce el nº del producto vendido y nº de unidades: 1   5
  Introduce el nº del producto vendido y nº de unidades: 3   20
  Introduce el nº del producto vendido y nº de unidades:-1  -1
Introduce nº de vendedor y nº de zona: 1   3
  Introduce el nº del producto vendido y nº de unidades: 5   50
  Introduce el nº del producto vendido y nº de unidades: 3   10
  Introduce el nº del producto vendido y nº de unidades:-1  -1
Introduce nº de vendedor y nº de zona: -1  -1
```

La entrada de datos finalizará cuando se introduzca un **-1** en el número de vendedor, sin importar el número que nos den en la zona. Y se cambiará de vendedor y zona cuando se introduzca un **-1** en el nº de producto, sin importar el nº de unidades. Con esta entrada de datos lo que nos interesa guardar en memoria, para hacer el estudio, es para cada vendedor y zona, el importe total de las ventas. Tener en cuenta que, puede ocurrir que algún vendedor no realice ninguna venta en alguna de las zonas.

El programa deberá realizar lo siguiente:

1. Mostrar en pantalla el importe total de las ventas realizadas por cada vendedor en cada zona, tal y como se indica en el ejemplo. **Ejemplo:**

```
Vendedor      Zona0    Zona1   Zona2 ..... Zona19
     0        200.5    150.00  567.00 ..... 7584.00
     1        560.25   486.00  968.00 .....  238.00
     ……       ……       ……      ……         ……        ……
     9        458.75   158.00  445.00 .....  868.00
```

2. Dada una zona, indicar cuál ha sido el mejor vendedor de esa zona. **Ejemplo:**

```
Dame zona: 3
El mejor vendedor de la zona 3 ha sido el vendedor número 2.
```

3. Indicar, qué vendedor y en qué zona ha obtenido el mayor importe de ventas. **Ejemplo:**

```
El mayor importe de ventas lo ha obtenido el vendedor 1 en la
zona 19.
```

4. Indicar para cada zona, qué vendedores no han conseguido realizar ninguna venta y cuántos son en total. **Ejemplo:**

```
Zona 0: 3  8  9
     Total 3 vendedores no han conseguido ventas.
Zona 1: Todos los vendedores han realizado alguna venta.
            ……
            ……
Zona 19: 3  4  5  7  8  9
     Total 6 vendedores no han conseguido ventas.
```

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *TopeVendedor* = 10. Número de vendedores de la empresa.
2. *TopeZonas = 20*. Número de zonas en las que trabajan los vendedores.

Representación de las variables necesarias para la entrada

3. Necesitaremos una matriz que le llamaremos *ventas*, de *TopeVendedor* X *TopeZonas* posiciones, para almacenar lo que ha vendido cada vendedor en cada zona.

ventas	0	1	…	TopeZonas -1
0				
1				
…				
TopeVendedor -1				

4. Necesitaremos un vector que le llamaremos *prodPrecio*, de 5 posiciones para almacenar en cada posición el precio de cada producto.

	0	1	…	4
prodPrecio	20.5	50		150

Pasos a seguir en la resolución del problema

1. Inicializar la matriz *ventas* a 0.
2. Leer el primer par vendedor, zona (*v,z*)
   a) Mientras el vendedor sea distinto de -1 hacer:
      a. Leer el par (*p,u*)
         i. Iniciar sum a 0
      b. Mientras el producto sea distinto de -1,
         i. Acumular en sum las unidades (*u*) por el precio del producto que está en el vector *prodPrecio*
         ii. Leer siguiente par (*p,u*)
      c. Almacenar las unidades acumuladas en la matriz *ventas*
3. Leer el siguiente par (*v,z*)
4. Mostrar en pantalla la matriz tal y cómo se indica en el enunciado
5. Por cada vendedor:
      a. Recorrer cada fila y mostrar las ventas en cada zona, que serán cada una de las posiciones de cada fila
6. Leer una zona :

   a. Recorrer en la matriz *ventas* la columna correspondiente a esa zona y buscar el elemento mayor

7. Suponer que el elemento que está en la primera posición de la matriz *ventas* es el mayor, para ello poner la posición de la fila y de la columna a 0, *vmax*=0 y *zmax*=0

   a. Recorrer toda la matriz y comparar siempre si *ventas[v][z] <ventas[vmax][zmax],*si ocurre tal condición se almacenan en *vmax y zmax* los nuevos valores de v y z, al finalizar en *vmax y zmax* estarán las posiciones del mejor vendedor y zona

8. Por cada zona:

   a. Inicializar cont a 0 para acumular el número de vendedores que no han conseguido ventas

   b. Recorrer en la matriz *ventas* cada columna para ver por cada vendedor en esa zona si no ha tenido ventas, si es así acumular en *cont* el número de vendedores que no han tenido ventas en esa zona.

   c. Si *cont* es 0 al terminar con la zona es que todos los vendedores han tenido ventas

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define TopeVendedor 10
#define TopeZonas 20

main(){
 float ventas [TopeVendedor] [TopeZonas];
 float prodPrecio [5] = {20.5, 50, 15, 5.25, 150};
 float sum ;
 int v, z, p, u, vmax, zmax, cont ;
 for(v=0 ; v<TopeVendedor ; v++)     // Inicializar la matriz a 0
 {  for(z=0 ; z<TopeZonas ; z++)
        {ventas[v][z]=0;}      //
 }
 /* Introducción de los datos  */
 printf("Introduce el nº de vendedor y nº de zona: ");
 scanf("%d %d", &v, &z);
 while (v!=-1)
 {  printf(" Introduce el nº del producto vendido y nº de unidades:
");
    scanf("%d %d", &p, &u);
    sum=0;
    while (p!=-1)       // repetir para cada producto y unidades
    {  sum=sum + u * prodPrecio[p];
       printf("Introduce el nº del producto vendido y nº de
unidades: ");
       scanf("%d %d", &p, &u);
    }
    ventas[v][z]= ventas[v][z]+ sum;
    printf("Introduce el nº de vendedor y nº de zona: ");
    scanf("%d %d", &v, &z);
 }

 /* Mostrar en pantala */
 printf("Vendedor / ");
 for(z=0 ; z<TopeZonas; z++)
```

```
 {  printf ("Zona%d\t", z);
 }
 printf("\n\n");


 for(v=0 ; v<TopeVendedor ; v++)
 {  printf("    %d\t",v);
    for(z=0 ; z<TopeZonas ; z++)
    { printf ("\t%5.2f", ventas[v][z]); }
    printf("\n");
 }


 /* Mejor vendedor  */
 printf("\nDame zona: "); scanf("%d",&z);
 vmax=0;       // suponemos que el de la primera posición es el
mejor
 for(v=0 ; v<TopeVendedor ; v++)
   if (ventas[vmax][z]<ventas[v][z]) {vmax=v;}
                   // si se encuentra uno mejor se sustituye


 printf(" El mejor vendedor de la zona %d ha sido el vendedor
número %d .\n\n", z,vmax);


 /* Buscar cuál ha sido el mejor vendedor y en que zona  */
 vmax=0;                 // Por defecto el que ocupa la primera
posición el mayor
 zmax=0;
 for (v=0 ; v<TopeVendedor ; v++)
  for(z=0 ; z<TopeZonas ; z++)
   { if (ventas[vmax][zmax]<ventas[v][z])
     { vmax=v; zmax=z;}   // si se encuentra alguno mayor se
almacena en vmax y zmax las nuevas posiciones
    }
 printf("El mayor importe de ventas lo ha obtenido el vendedor %d
en la zona %d .\n\n", vmax, zmax);


  /* Vendedores sin ventas  */
 for(z=0 ; z<TopeZonas ; z++)
 {cont=0;
  printf("\nZona %d: ", z);
```

```
  for (v=0 ; v<TopeVendedor ; v++)
  { if (ventas[v][z]==0)  { printf("  %d", v); cont++; }
  }
  if (cont!=0)
     {printf("\nTotal %d vendedores no han conseguido ventas.",
cont);}
  else {printf("\nTodos los vendedores han conseguido alguna
venta");
 }
 printf("\n\n");
 system("PAUSE");
}
```

ACCIDENTES DE CIRCULACIÓN

Tras las últimas vacaciones el número de accidentes de circulación ha sido tan elevado, que un total de 50 diputaciones forales han decido compartir los datos de los siniestros mortales sucedidos en dicho periodo, para extraer algunos resultados estadísticos.
Las 4 causas de siniestros mortales consideradas son: velocidad, alcohol, drogas y otros.
Una empresa teleoperadora registrará durante unos días las cifras que le irán suministrando los responsables de las distintas prosincias. La introducción de los datos se realizará mediante tripletas (prosincia, causa, núm. muertos). Puesto que puede suceder que alguno de los heridos muera, será necesario introducir en más de una ocasión tripletas con la misma prosincia y misma causa; en cuyo caso, habrá que incrementar con el nueso número suministrado la cifra ya almacenada. Para finalizar la introducción de los siniestros mortales, bastará con indicar -1 en alguno de los valores del triplete de entrada.
Ejemplo: (en negrita los datos de la teleoperadora)

```
Introducción de siniestros mortales (uno del triple -1 para
finalizar).
  Provincia   causa   n. muertos: 0  1  2
  Provincia   causa   n. muertos: 0  0  11
  Provincia   causa   n. muertos: 3  0  5
  Provincia   causa   n. muertos: 0  0  2
  ……….
  Provincia   causa   n. muertos: 0 -1  0
```

Una vez finalizado el registro del número de fallecidos, se registrará también por provincias el número total de kilómetros recorridos por bomberos y sanitarios en los siniestros considerados.
Ejemplo: (en negrita los datos de la teleoperadora)

```
Registro de los km recorridos en total por prosincia:

   Provincia 0: 340
   Provincia 1: 124
   Provincia 2: 678
   ........
   Provincia 49: 210
```

Con los datos almacenados se pide:

1) Calcular el número de muertos por provincia, y el número total de muertos tras las últimas vacaciones.

2) Determinar si la provincia con más víctimas mortales es la provincia con mayor kilometraje en asistencias. **Ejemplo**:

   ```
   La provincia con más víctimas mortales es la 35.
   ```

```
La provincia con mayor kilometraje en asistencias es 2 .
No coinciden las provincias.
```

3) Por provincia estudiada, determinar el promedio de kilómetros recorridos por cada víctima mortal. **Ejemplo**:

```
Provincia 0: 23.2km recorridos por víctima mortal.
Provincia 1: 12.7km recorridos por víctima mortal.
.............
```

4) Solicitar una de las causas de accidentes estudiadas y determinar el número de víctimas mortales originadas por la misma. Repetir el proceso en tanto en cuanto la causa indicada sea una de las estudiadas. **Ejemplo**:

```
Indique una causa de accidente: 1
En total hay registrados 57 muertos por la causa 1.

Indique una causa de accidente: 0
En total hay registrados 89 muertos por la causa 0.

Indique una causa de accidente: -7
```

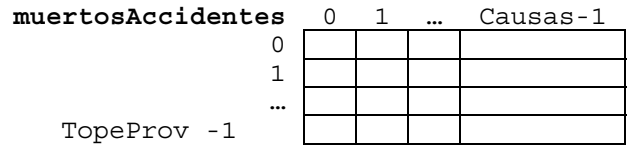
ANÁLISIS DEL PROBLEMA

Constantes del enunciado

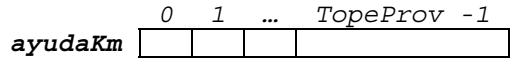
1. *TopeProv* = Número de provincias que intervienen en el estudio.
2. *Causas* = *4*. Número de causas de siniestros mortales.

Representación de las variables necesarias para la entrada

3. Necesitamos una matriz de *TopeProv* X *Causas* posiciones que le llamaremos *muertosAccidentes* donde se almacenará el número de muertos que ha habido en cada provincia por cada una de las causas.

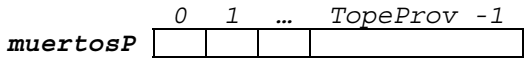


4. Necesitaremos un vector de *TopeProv -1* posiciones que le llamaremos *ayudaKm* donde almacenaremos el número total de kilómetros recorridos por bomberos y sanitarios en los siniestros considerados por provincia.

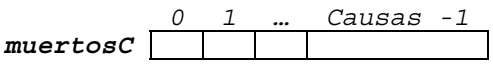


Representación de las variables necesarias para los resultados

5. Necesitamos un vector de *TopeProv* posiciones que le llamaremos *muertosP* donde almacenaremos el número total de muertos habidos por provincia.



6. Necesitaremos un vector de *Causas* posiciones que le llamaremos *muertosC* donde almacenaremos el número total de muertos que ha habido por cada una de las causas entre todas las provincias.



Pasos a seguir en la resolución del problema

1. Inicializar la matriz *muertosAccidentes* a 0.
2. Leer la primera tripleta (*p,c,m*)
3. Mientras los tres valores sean distintos de -1
   a. Acumularel número de muertos en la matriz *muertosAccidentes*
   b. Leer la siguiente tripleta (*p,c,m*)

4. Almacenar en el vector *ayudaKm* el número total de kilómetros recorridos por bomberos y sanitarios en los siniestros considerados por provincia.
5. Inicializar a 0 *tot* para calcular el total de muertos entre todas las provincias.
6. Por cada provincia:
    a. Calcular y almacenar en el vector *muertosP* el número de muertos habidos entre todas las causas, recorriendo la matriz *muertosAccidentes*
    b. Acumular en *tot* el total de muertos de la provincia
7. Calcular la provincia con más víctimas mortales, es decir, calcular la posición del máximo del vector *muertosP*, dejando la posición del máximo en *maxP*.
8. Calcular la provincia con más kilometraje en asistencias, es decir, calcular la posición del máximo del vector *ayudasKm*, dejando la posición del máximo en *KmmaxP*.
9. Comparar *MaxP* y *kmMaxP* , para determinar si la provincia con más víctimas mortales coincide caon la provincia con mayor kilometraje en asistencias.
10. Por cada provincia dividir cada elemento del vector *muertosP* por el elemento de la misma posición del vector *ayudasKm* para obtener el promedio de kilómetros recorridos por víctima mortal.
11. Por cada causa (*c*)
    a. Calcular y almacenar en el vector *muertosC* el número de muertos habidos entre todas las provincias, recorriendo la matriz *muertosAccidentes*
12. Leer un causa *c*
13. Mientras la causa ($0 \leq c < 4$), repetir:
    a. Mostrar el número de muertos registrados en el vector *muertosC* por esa causa
    b. Leer otra causa *c*

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define TopeProv 50
#define Causas 4
main (){
  int muertosAccidentes[TopeProv][Causas];
  int ayudaKm[TopeProv], muertosP[TopeProv];
  int muertosC[Causas];
  int pro,c, m, tot, maxP,kmMaxP;
  float bb;

  // Matriz a 0
  for (pro=0;pro<TopeProv; pro++)
     for (c=0;c<Causas;c++)
       muertosAccidentes[pro][c]=0;

  // Registrar muertos por provincia y causa
  printf("Introducción de siniestros mortales\n");
  printf("  (uno del triple -1 para finalizar)\n");
  printf("   Provincia Causa n. muertos:  ");
  scanf("%d %d %d", &pro, &c, &m);
  while (pro!=-1 &&c!=-1 && m!=-1)
  { muertosAccidentes[pro][c]=muertosAccidentes[pro][c]+m;
    printf("   Provincia Causa n. muertos:  ");
    scanf("%d %d %d", &pro, &c, &m);
  }

  // Registrar los kms recorridos por provincia
  for (pro=0;pro<TopeProv; pro++)
  { printf("%d Provincia: ", pro);
    scanf("%d", &ayudaKm[pro]);
   }

  // 1. Muertos por provincia y total
  tot=0;
```

```
  for (pro=0;pro<TopeProv; pro++)
  {  muertosP[pro]=0;
     for (c=0;c<Causas;c++)
        muertosP[pro]=muertosP[pro]+
                          muertosAccidentes[pro][c];
    printf("El total de muertos de la  provincia %d son %d.\n",pro,
muertosP[pro] ),
    tot=tot+muertosP[pro];
   }
   printf("El total de muertos %d.\n\n", tot);


   /* 2. Coinciden la provincia de más muertos con la de mayor km.
de asistencia?
    La provincia de más muertos */
   maxP=0;
   for (pro=1;pro<TopeProv; pro++)
      if (muertosP[maxP] < muertosP[pro]) {maxP=pro;}
   printf("La  provincia  con  más  víctimas  mortales  es  %d  .\n",
maxP);

   // La provincia con más kilometraje
   kmMaxP=0;
   for (pro=1;pro<TopeProv; pro++)
      if (ayudaKm[kmMaxP] < ayudaKm[pro]) {kmMaxP=pro;}
   printf("La provincias con mayor kilometraje es"
          " %d .\n", kmMaxP);

   if (maxP==kmMaxP)
      {printf("Coinciden las provincias.\n\n");}
   else {printf("No coinciden las provincias.\n\n");}

   //  3.  Determinar  el  promedio  de  kms.  recorridos  en  cada
provincia por cada víctima mortal
   for (pro=0;pro<TopeProv; pro++)
   {  bb=float(muertosP[pro])/ayudaKm[pro];
```

```
      printf(" Provincia %d: %.1f km recorridos por víctima mortal"
             " .\n", pro, bb);
   }
   printf("\n\n");

   // 4. Calcular por cada causa número de víctimas mortales entre
todas las provincias
   for (c=0;c<Causas;c++)
   {  muertosC[c]=0;
      for (pro=0;pro<TopeProv; pro++)
         muertosC[c]=muertosC[c]+
                         muertosAccidentes[pro][c];
   }

   printf("Indique una causa de accidente: ");
   scanf("%d", &c);
   while (0<=c &&c<Causas)
   {  printf("En total hay %d muertos por la causa %d.\n\n",
       muertosC[c], c);
       printf("Indique una causa de accidente: ");
       scanf("%d", &c);
   }

 printf("\n\n");
 system ("PAUSE");
}
```

EUSKADI IRRATIA

Desde "Euskadi Irratia" nos han llamado para realizar un programa informático que les permita gestionar la información sobre los colaboradores de los distintos programas de Radio. Analizando la información que tienen de las últimas temporadas radiofónicas, se han dado cuenta de que como máximo tienen **100** colaboradores por temporada y la programación de una temporada tiene como máximo **50** programas distintos. También es sabido que los colaboradores participan en más de un programa y puede ocurrir que alguno no participe en ninguno (por temas de enfermedad, porque se pensaba abrir una nueva sección en un programa que al final no ha salido, ...).
Para ello, hemos hablado con el responsable de la programación de "Euskadi Irratia" y nos ha pedido que el programa tenga una interfaz como la siguiente.

Inicialmente saldrá un menú con las siguientes opciones:

```
A) Introducir programación.
B) Añadir programa.
C) Añadir colaboradores.
D) Introducir la participación de los colaboradores en los
   distintos programas
E) Calcular lo que cobrará cada colaborador y lo que tiene que
   pagar "Euskadi Irratia" en total a todos los colaboradores
F) Qué programa gasta más en colaboradores.
G) En qué programas no ha participado ningún colaborador y
   cuantos son.
H) Terminar
```

Por cada una de las opciones el programa tendrá que hacer lo siguiente:

A) Introducir programación.
Cuando se elija esta opción será cuando se cambie de temporada, es decir la información que teníamos hasta ahora no sirve y nos van a dar otra (cambio de invierno a verano, por ejemplo). Los datos se introducirán en el siguiente formato (En **negrita** los datos introducidos por el usuario):

```
Dame la lista de programas de esta temporada (para terminar
teclear Fin):
    Programa  0: Goiz kronika
    Programa  1: Faktoria
    Programa  2: Aperitifa
    Programa  3: Goizak Gaur
    …
    Programa 38: Fin
```

B) Añadir programa.

Esta opción añade un nuevo programa a la programación, si no se ha llegado al máximo de programas posibles. Si se llega al máximo de programas el programa escribirá el siguiente mensaje:

"La lista de posibles programas está llena, póngase en contacto con el servicio de informática.

En caso contrario, se añadirá el nuevo programa:

```
Introduce programa 38: Esaidazu
```

C) Añadir colaboradores.
Cuando se elija esta opción se añadirán colaboradores, si no se ha llegado al máximo. En caso de que lleguemos al máximo daremos un mensaje de error diciendo que no podemos añadir más colaboradores, como por ejemplo:

"La lista de posibles colaboradores está llena, póngase en contacto con el servicio de informática."

Si se pueden añadir más colaboradores la comunicación seria de la siguiente manera (En **negrita** los datos introducidos por el usuario):

```
Introduce los colaboradores (para terminar teclea Fin):
    Colaborador : Iñigo Segurola
    Cobra por minuto (en euros): 5
    Colaborador : Andoni Egaña
    Cobra por minuto (en euros): 3.75
    Colaborador : Jesús Torquemada
    Cobra por minuto (en euros): 6.5
    …
    Colaborador: Fin
```

D) Introducir la participación de los colaboradores en los distintos programas.
Cada vez que se elija esta opción se añadirá a los colaboradores las nuevas participaciones que hayan tenido en los distintos programas. Se pedirá para cada programa la lista de colaboradores con el tiempo de intervención, como se ve en el siguiente ejemplo (En **negrita** los datos introducidos por el usuario):

```
Programa Goiz kronika:
    Dame colaborador y tiempo (en minutos): 2 150
    Dame colaborador y tiempo (en minutos): 1 120
    Dame colaborador y tiempo (en minutos): 25 150
    El colaborador 25 no existe.
    Dame colaborador y tiempo (en minutos): 2 150
    ...
    Dame colaborador y tiempo (en minutos): -1 0
Programa Factoria:
    Dame colaborador y tiempo (en minutos): 2 150
    ...
    Dame colaborador y tiempo (en minutos): -1 0
...
```

E) Calcular lo que cobrará cada colaborador y lo que tiene que pagar "***Euskadi Irratia***" en total a todos los colaboradores.

Teniendo en cuenta las participaciones realizadas hasta el momento calcular cuanto hay que pagar a cada colaborador y en gasto total de la radio en colaboradores. Si un colaborador no ha participado en ningún programa no aparecerá en la lista.

```
Andoni Egaña ha participado 120 minutos: 450 euros
Jesus Torquemada ha participado 450 minutos: 2925 euros
En total "Euskadi Irratia" gastará en colaboradores 3375 euros.
```

F) Qué programa gasta más en colaboradores.
Calcular, con los datos que tenemos hasta el momento, el programa que mas dinero ha gastado en colaboradores.

```
El programa que más gasta en colaboradores es Goiz Kronika con 2400 €.
```

G) En qué programas no ha participado ningún colaborador y cuántos son.
Indicar el nombre de los programas en los que no ha participado ningún colaborador y indica en cuántos casos ha pasado esto.

```
No ha participado ningún colaborador en los programas:
Goizak Gaur
Klasikoak EITBn
En total son 2 programas.
```

H) Terminar.
Al teclear esta opción el programa terminará.

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *TopeColab* = 100. Múmero máximo de colaboradores por programa.
2. *TopeProg* = 50. Número máximo de programas distintos en una misma temporada.
3. *TopeLong* = 70. Número máximo de caracteres que puede tener el nombre de un programa o de un colaborador.

Representación de las variables necesarias para la entrada

4. Necesitamos la matriz *nomProg*, de tipo caracter, para guardar los nombres de los programas. La matriz tiene *TopeProg* filas y *TopeLong* columnas, para poder guardar los nombre de los programa en cada una de las fila.

nomProg	0	1	...	TopeLong-1
0				
1				
...				
TopeProg-1				

5. Para guardar los nombres de los colaboradores utilizamos la matriz de caracteres *nomColab*. La matríz tiene *TopeColab* filas y *TopeLong* columnas.

nomColab	0	1	...	TopeLong-1
0				
1				
...				
TopeColab-1				

6. Para guardar los euros por minuto que cobra cada colaborador, utilizamos el vector *colabMin* con *TopeColab* posiciones.

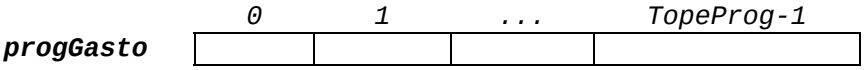
	0	1	...	TopeColab-1
colabMin				

7. Utilizamos la matriz *tiempo* de *TopeProg* X *TopeColab* posiciones, para guardar el tiempo (en minutos) de intervención de los colaboradores en los diferentes programas.

tiempo	0	1	...	TopeColab-1
0				
1				
...				
TopeProg-1				

Representación de las variables necesarias para los resultados

8. En el vector *progGasto* se guarda el gasto total que tiene cada programa en colaboradores.



Pasos a seguir en la resolución del problema.

1. Inicializamos en contador de colaboradores a 0 (*contColab)*.
2. Repetir:
   a. Escribir el menu en pantalla.
   b. Leer la opción elegida.
   c. Si se elige la opción 'A', entonces:
      i. Inicializamos el contador de programa a 0 (*contProg*).
      ii. Se inicializa la matriz *tiempo* a 0.
      iii. Se lee el nombre del programa.
      iv. Mientras no se introduzca el nombre "*Fin*", se guarda el nombre del programa en la matriz *nomProg*, se incrementa el contador de programa y se lee el siguiente nombre.
   d. Si se elige la opción 'B', entonces:
      i. Si el contador de programas *contProg* ha alcanzado el valor máximo *TopeProg*, entoneces damos un mensaje indicando que la lista está llena.
      ii. En caso contrario:
         1. se pide el nombre del programa y se guarda en el vector *nomProg*.
         2. se incrementa el contador de programas *contProg*.
   e. Si se elige la opción 'C', podemos añadir colaboradores.
      i. Si la lista de colaboradores está llena damos un mensaje indicandolo.
      ii. En caso contrario:
         1. Leemos el nombre del colaborador. Mientras no nos den el nombre "Fin" y mientras haya sitio en la lista de colaboradores guardamos el nombre del colaborador en la matriz *nomColab* y leemos en el vector *colabMin*, lo que cobra este colaborador por minuto. Incrementamos el contador de colaboradores y pedimos el siguiente nombre.
   f. Si se elige la opción 'D', se pueden añadir colaboradores en los distintos programas:

iii. Recorremos secuencialmente todos los programas.

1. Leemos el colaborador y el tiempo
2. Mientras no nos den el colaborador -1, y si además el colaborador existe, entonces acumulamos el tiempo del colaborador en ese programa, si no existe damos un mensaje indicandolo.
3. Leemos el siguiente colaborador y tiempo.

g. Si se elige la opción 'E', calculamos lo que cobra cada colaborador por el total de minutos trabajados y lo que gasta en total la radio en colaboradores:

i. Inicializamos a 0 lo que paga en total la radio.

ii. Recorremos secuencialmente todos los colaboradores,

1. Para cada colaborador recorremos todos los programas acumulando en la variable *min* el tiempo que ha participado.
2. Cuando teminamos de recorrer todos los programas, calculamos lo que gana en total el colaborador *ganancia,* multiplicando los minutos por lo que cobra por minuto.
3. Acumulamos la ganancia del colaborador al total que paga la radio.

h. Si se elige la opción 'F', calculamos cuál es el programa que mas ha gastado en colaboradores:

i. Para calcular el gasto total por programa *progGasto*, recorremos secuencialmente la matriz *tiempo* de la siguiente forma:

a. El bucle principal nos mueve por las filas que en este caso son los programas.

b. Para cada programa inicializamos el vector *progGasto* a 0 y recorremos todo las columnas o colaboradores, multiplicando el tiempo por lo que cobra el colaborador *colabMin* y acumulando el resustado en el vector *progGasto*.

ii. Buscamos el programa que mas ha gastado en el vector *progGasto*.

i. Si se elige la opción 'G', buscamos los programas en los que no ha tomado parte ningún colaborador:

i. Inicializamos a 0 el contador de programas *pCont*, el los que no ha participado ningún colaborador.

ii. Para cada programa contamos en *cCont*, cuántos colaboradores han participado mas de 0 minutos .

iii. Si el contador *cCont* se queda a 0, indica que en este programa no ha participado ningún colaborador, y por lo tanto mostramos por pantalla el nombre del programa y lo contamos en *pCont*.

j. Si se elige la opción 'H' se da un mensaje de FIN y se termina el programa.

k. Si la opción elegida no es ninguna de las anteriores se indicará que la opción es erronea y se mostrará otra ves el menu .

CÓDIGO C

```
#include <stdio.h>
#include <stdlib.h>
#define TopeColab 100
#define TopeProg 50
#define TopeLong 70
main (){
 //Matriz que guarda los nombres de los diferentes programas
 char nomProg[TopeProg][TopeLong];
 //Matriz que guarda los nombres de los diferentes colaboradores
 char nomColab[TopeColab][TopeLong];
 float progGasto[TopeProg], colabMin[TopeColab];
 int tiempo[TopeProg][TopeColab];
 int p, c, min, contProg, contColab,pmax, pCont,cCont;
 char opcion, nombre[TopeLong];
 float totalRadio,ganancia;
contColab=0; //inicializamos el contador de colaboradores
do {
  printf(" A) Introducir programación.\n");
  printf(" B) Añadir programa.\n");
  printf(" D) Añadir colaboradores en los distintos programas.\n");
  printf(" D)
  printf(" H) Terminar.\n");
  printf(" Elige una opción : ");
  fflush(stdin);
  scanf("%c",&opcion);
  switch(opcion){
  case 'A':
       //Introducir nueva programación.
       //Inicializamos la matriz de tiempo a 0
       for (p=0; p< TopeProg ;p++)
       {  for (c=0; c< TopeColab; c++)
          { tiempo[p][c]=0; }
       {
       //Inicializamos el contador de programas a 0
```

```
      contProg=0;
      //Leer la nueva programación
      printf(“ Dame la lista de programas de esta temporada. “
             “ (para teminar teclear Fin):\n”);
      printf(“ Programa %d”,contProg);
      fflush(stdin);
      gets(nombre);
      while (strcmp(nombre,“Fin“)!=0 && contProg<TopeProg)
      { strcpy(nomProg[contProg],nombre);
        contProg=contProg+1;
        printf(“ Programa %d”,contProg);
        fflush(stdin);
        gets(nombre);
      }
      break;
case ‘B’: //Añadir un programa nuevo
      if (contProg== TopeProg)
      {printf(“La lista de posibles programas está llena, póngase”
              “en contacto con el servicio de informática. “);
      }
      else
      { printf(“ Programa %d”,contProg);
        fflush(stdin);
        gets(nomProg[nombre]);
        contProg =contProg+1;
      }
      break;
case ‘C’: // Añadir colaboradores
      if (contColab == TopeColab)
      {printf(“La lista de posibles programas está llena, “
           “póngase en contacto con el servicio de informática.”);
      }
      else
      {printf(“Dame el colaborador(para teminar teclear Fin): “);
       fflush(stdin);
       gets(nombre);
       while (strcmp(nombre,”Fin”)!=0 && contColab<TopeColab)
```

```
        { strcpy(nomColab[contColab],nombre);
          printf(" Precio minuto (en euros): ");
          scanf("%f", &colabMin[contColab]);
          contColab=contColab+1;
          printf("Dame el colaborador ");
          fflush(stdin);
          gets(nombre);
          } // fin del while
        } // fin del else
        break;
    case 'D': // Añadir colaboradores en los distintos programas
        for (p=0; p<contProg ; p=p+1)
        { printf(" Programa %s:\n",nomProg[p]);
          printf(" \tDame colaborador y tiempo (en minutes): ");
          scanf("%d %d", &c, &min);
          while (c!= -1)
          { if (c<contColab)
            {tiempo[p][c]= tiempo[p][c]+min; }
            else
            { printf("El colaborador %d no esiste " ,c); }
            printf(" \tDame colaborador y tiempo (en minutes): ");
            scanf("%d %d", &c, &min);
          } // fin del while
        } //fin del for
        break;
    case 'E': /* Calculamos lo que cobra cada colaborador por el
        total de minutos trabajados y lo que gasta en total la
        radio*/
        totalRadio=0;
        for (c=0; c<contColab; c=c+1)
        { min=0;
          for (p=0; p < contProg; p=p+1)
          { min= min+tiempo[p][c]
          }
          ganancia= min*colabMin[c];
```

```
        totalRadio=totalRadio + ganancia;
        if (ganancia>0)
        { printf(" El colaborador %s ha participado %d minutos y "
              "ha ganado %8.2f euros.\n",nomColab[c],min,ganacia);
        } // fin del if
      } // fin del for principal
      printf (" Gasto total del los colaboradores de Euskadi"
             " rradia: %8.2f euros\n.", totalRadio);
      break;
  case 'F':
      //Cúal es el programa que más ha gastado en colaboradores.
      //Calculamos lo que ha gastado cada programa.
      for (p=0; p < contProg; p=p+1)
      { progGasto[p]=0;
        for (c=0; c<contColab; c=c+1)
        { progGasto[p]=progGasto[p]+ tiempo[p][c]*colabMin[c];}
      }
      //Buscamos el programa que mas ha gastado.
      pmax=0;
      for (p=1; p < contProg; p=p+1)
      { if (progGasto[pmax]<progGasto[p])
        { pmax=p; }
      }
      printf(" El programa que mas ha gastado en colaboradores es"
           " %s con %8.2f euros\n",nomProg[pmax],progGasto[pmax]);
      break;
  case 'G':
      /* En qué programas no ha tomado parte ningún colaborador y
       cuántos son.*/
      printf(" Programas en los que no ha tomado parte ningún "
             " colaborador:\n");
      pCont=0;
      for (p=0; p < contProg; p=p+1)
      { cCont=0;
        for (c=0; c<contColab; c=c+1)
```

```
          { if (tiempo[p][c]>0)
            {cCont=cCont+1;}
          }
          if (cCont==0)
          { printf("%s \n", nomProg[p]);
            pCont=pCont+1;
          }
        } // fin del for principal
        printf ("Total de programas %d",pCont);
        break;
  case 'H': // fin de programa
        printf ("FIN");
        break;
  default:
        printf ("Opción erronea.\n");
  }
} while( opcion!='H');
printf("\n\n");
system("PAUSE");
}
```

SAN SEBASTIAN

El Ayuntamiento de San Sebastián nos ha pedido que le realicemos un programa para conocer ciertos datos sobre la tamborrada infantil del día de San Sebastián. Dicho programa pedirá información sobre las diferentes compañías existentes, año de creación y puesto en el que ha salido cada año. Todas las compañías a partir de su año de creación han salido todos los años.
Nos ha comunicado que actualmente participan 50 compañías en el desfile y que tenemos los datos a partir del año 1980 hasta el 2006.
Al inicio, el programa deberá pedir los nombres y año de creación de las distintas compañías, de la siguiente forma (en negrita los datos introducidos por el usuario):

```
Introducción de los datos de las diferentes compañías infantiles:
        Nombre: Mundaiz
        Año: 1980
        Nombre: Santo Tomas Lizeoa
        Año: 1984
        ...
```

Posteriormente nos pedirá, para cada compañía, el puesto en el que ha salido cada uno de los años. Tened en cuenta que hay que pedir los datos de cada compañía a partir de su año de creación (en **negrita** los datos introducidos por el usuario):

```
Dame los puestos de la compañía Mundaiz:
        Año 1980: 3
        Año 1981: 7
        ...
        Año 2006: 1
Dame los puestos de la compañía Santo Tomas Lizeoa:
        Año 1984: 13
        Año 1985: 21
        ...
        Año 2006: 2

.............
```

El programa después de realizar la lectura de datos le presentará al usuario las siguientes opciones en un menú, hasta que el usuario elija "Salir". Si la opción dada por el usuario no es válida el programa dará el siguiente mensaje *"Opción incorrecta"* y volverá a presentar el menú.

Este es el menú que deberá escribir el programa:

```
Menú:
1.- Ver información
2.- Para los años dados por el usuario calcular cuantas
compañías han salido en el desfile
```

```
3.- Decir cuántas compañías han salido en último lugar el año de
su creación
4.- Salir
Dame tu opción: 2
```

Para cada opción el programa calculará lo siguiente:

1. *Ver información*. Como se muestra en el ejemplo, se escribirá en formato tabla las compañías y los puestos en los que ha salido cada una de ellas. Si una compañía no ha salido algún año escribir un guión. Ejemplo:

```
              1980   1981    1982    1983    1984    ....    2006
Mundaiz        3       7       1       3       5     ....      1
S.T. Lizeoa    -       -       -       -      13     ....      2
............
```

2. *Para los años dados por el usuario calcular cuantas compañías han salido en el desfile.* Al usuario se le pedirá una secuencia de años y para cada año el programa le dirá cuántas compañías han desfilado. La secuencia de años terminará cuando el usuario teclee un –1 como año. Ejemplo:

```
Para cada año que me des te escribiré las compañías que han
salido (para terminar teclea –1):
     Dame año: 1997
     Las compañías que han salido son 23
     Dame año: 2001
     Las compañías que han salido son 4
     Dame año: 1991
     Las compañías que han salido son 16
     Dame año: -1
```

3. *Decir cuántas compañías han salido en último lugar el año de su creación.* Es decir, cuenta cuántas compañías han salido el año de su creación en el **último** puesto.

4. *Salir*. Terminará el programa

Nota:

Para relacionar un año con la posición que ocupa en el array, se puede restar al año correspondiente el año inicial, Ej: 1991- 1980 = 11 en este ejemplo vemos que 1991 ocuparía la posición 11 y 1980 que es el primer año ocuparía la posición 0.

ANÁLISIS DEL PROBLEMA

Constantes del enunciado

1. *TopeCompanías* = 50. Número de compañias que participan en el desfile.
3. *AnoInicio* = 1980. Año de inicio de la tamborrada infantil
4. *UltimoAno* = 2006. Último año de la tamborrada infantil
5. *TopeAnos* = *UltimoAno* - *AnoInicio* +1
6. *Tamano* = 40. Máximo tamaño del nombre de las compañías

Representación de las variables necesarias para la entrada

7. Necesitaremos una matriz de caracteres donde se almacenarán los nombres de las distintas compañías .

nombres	*0*	*1*	*...*	*Tamano-1*
0				
1				
...				
TopeCcompanias-1				

8. Necesitamos un vector donde se almacenarán los años de creación de las distintas compañías.

	0	*1*	*...*	*TopeCompanias-1*
anoCreacion				

9. Necesitamos una matriz donde se almacenarán para cada compañía el puesto en el que ha salido cada uno de los años. Para relacionar un año con la posición que ocupa en el array, restaremos al año correspondiente el año inicial. Ej: 1991-1980 = 11, 1991 ocuparía la posición 11.

puestos	*0*	*1*	*...*	*TopeAnos-1*
0				
1				
...				
TopeCompanias-1				

Pasos a seguir en la resolución del problema

1. Inicializar a 0 la matriz puestos
2. Leer datos:
   a. Leer año de comienzo y nombre de la compañía y guardarlos en los vectores correspondientes: nombres y anoCreacion

b. Leer los puestos en los que ha salido cada compañia desde su año de creación y almacenarlos en la matriz puestos

3. Repetir
   a. Escribir menú
   b. Elegir opción
   c. Según la opción elegida:
      i. Ver información
         (1) Escribir los años (*De inicio a fin*)
         (2) Por cada compañía
            (a) Escribir el nombre de la compañia
            (b) Por cada año
               (i) Leer de la matriz puestos el puesto en el que ha salido cada año, si en una posición hay un 0 se escribirá un - .
      ii. Para cada año dado por el usuario calcular cuantas compañías han salido en el desfile
         (1) Leer año (*a*)
         (2) Mientras *a* es distinto de -1
            (a) *cont*=0
            (b) Leer la columna de ese año
               (i) Ver para cada compañía si tiene algún valor en ese año
            (c) Escribir el valor de *cont*
            (d) Leer año (*a*)
      iii. Calcular cuántas compañías han salido en último lugar el año de su creación
         (1) *ultimoPuesto*=0
         (2) Para cada compañía
            (a) Obtener el año de su creación del vector *anoCreacion*
            (b) Obtener la posición del año (columna) en la matriz a través de la fórmula: año inicio de la compañía-año inicial
            (c) Calcular el elemento mayor de esa columna

(d) Si la última compañía coincide coincide con la que estamos tratando es que es la última en su año de creación

(e) Sumamos 1 a *ultimoPuesto*

(3) Escribir el valor de *ultimoPuesto*

iv. Elegir opción

v. Si la opción no está entre 1 y 4 volver a escribir la opción hasta que la opción sea 4.

CÓDIGO C

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define AnoInicio 1980
#define AnoFin 2006
#define TopeCompanias 50
#define TopeAnos 27
#define Tamano 40

main(){
 int puestos[TopeCompanias][TopeAnos];
 int anoCreacion[TopeCompanias];
 char nombres[TopeCompanias][Tamano];
 int p, a, c, cont, opcion, puestoUltimo, cUltima, ultimaCom;

 /* Inicializamos a 0 la matriz */
 for (c=0; c< TopeCompanias; c=c+1) {
  for(a=0; a< TopeAnos;a=a+1) {
   puestos[c][a]=0;
  }
 }

 /* Leer año de comienzo y nombre de la compañía */
 printf("Introducción de los datos de las diferentes compañías
infantiles:\n");
 printf(":\n");
 for (c=0; c< TopeCompanias; c=c+1) {
   printf("Nombre: ");
   fflush(stdin);
   gets(nombres[c]);
   printf("Año: ");
   scanf("%d", &anoCreacion[c]);
 }
```

```
 /* Leer los puestos en los que ha salido cada compañia desde su
año de creación*/
 for (c=0; c< TopeCompanias; c=c+1) {
  printf("Dame los puestos de la compañía %s : \n", nombres[c]);
  for(a=anoCreacion[c]; a<=AnoFin; a=a +1) {
   printf("Año %d: ", a);
   scanf("%d", &puestos[c][a-AnoInicio]);
  }
 }


 do {
  /* Escribir menú */
  printf("Menú:\n");
  printf("\t1.- Ver información \n");
  printf("\t2.- Para los años dados por el usuario calcular "
         "cuantas compañías han salido en el desfile\n");
  printf("\t3.- Decir cuántas compañías han salido en último"
         "lugar el año de su creación\n");
  printf("\t4.- Salir\n");
  printf("\nDame tu opción: ");
  /* Elegir opción */
  scanf("%d", &opcion);
  /* Tratar opción*/
  switch (opcion) {
    case 1:
         /* Mostrar matriz en pantalla */
         /* Escribir años */
         printf("\t\t");
         for (a=AnoInicio; a<=AnoFin; a=a+1) {
          printf("%d\t", a);
         }
         printf("\n");
         /* Escribir los datos de cada compañia*/
         for (c=0; c<TopeCompanias; c=c+1) {
          printf("%-30s", nombres[c]);
          for(a=0; a<TopeAnos; a=a+1) {
           if (puestos[c][a]==0) {
```

```
            printf("-\t");
           }
           else {
                 printf("%d\t",puestos[c][a]);
                }
          }
          printf("\n");
        }
        break;

    case 2:
         /* Calcular para cada año cuantas compañias han salido en
el desfile*/
         printf("Para cada año que me des te escribiré las "
                "compañías que han salido\n");
         printf(" (para terminar teclea -1):\n");
         printf("Dame año: ");
         scanf("%d", &a);
         while (a != -1) {
          cont=0;
          for (c=0; c<TopeCompanias; c=c+1) {
           if (puestos[c][a-AnoInicio]!=0) {
            cont=cont+1;
           }
          }
          printf("Las compañías que han salido son %d\n\n", cont);
          printf("Dame año: ");
          scanf("%d", &a);
         }
         break;
    case 3:
         /* Calcular cuantas compañías han salido en último lugar
en su año de creación*/
         puestoUltimo=0;
         for (cUltima=0; cUltima<TopeCcompanias; cUltima=cUltima+1){
          a = anoCreacion [cUltima]-AnoInicio;
          ultimaCom=0;
```

```
          for (c=0; c<TopeCompanias; c=c+1) {
           if (puestos[c][a]>puestos[ultimaCom][a]) {
            ultimaCom = c;
           }
          }
          if (ultimaCom == cUltima) {
           puestoUltimo = puestoUltimo +1;
          }
         }
         printf("Número de compañías que han salido en último "
      "lugar el año de su creación total %d da\n", puestoUltimo);
         break;
    case 4:
         printf("\n\nTerminará el programa ...\n\n");
         break;
    default:
            printf ("\n\nEscribe bien la opción.\n\n");
  }
 } while (opcion!=4);

 printf("\n\n");
 system("PAUSE");
}
```