TEMA 6
TABLAS NUMÉRICAS / VECTORES

1 VECTORES: DEFINICIÓN Y USO

1.1 Definición y uso

Una tabla suele estar formada por un conjunto de variables o elementos del mismo tipo. Para definir una tabla de una dimensión, hay que especificar el tipo de elementos, el nombre de la tabla y el número de elementos. Las tablas nos ofrecen una forma fácil y adecuada para definir y manejar tablas con muchos elementos del mismo tipo.
Como las tablas con elementos de tipo char tienen unas características especiales, en este tema sólo se tratarán las tablas numéricas, con elementos de tipo int y flota.

Las tablas de una única dimensión también se denominan **vectores**.

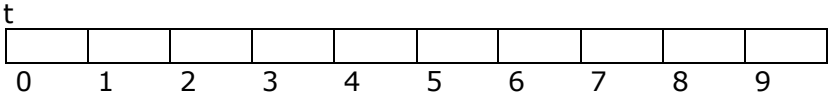
Cada elemento de un vector se identifica mediante un número conocido como **índice**. Si un vector se compone de N elementos, el índice del primer elemento será 0 y el del último, N-1.

1º ejemplo:

Supongamos que queremos definir un vector de nombre t y que tenga 10 elementos de tipo int. Deberíamos escribir lo siguiente:

```
int t[10];
```

Mediante esta definición, se crearía la siguiente tabla:



Cuando se define una tabla, inicialmente tendrá valores desconocidos.

Cada elemento de la tabla t se puede manejar como una variable de tipo int.

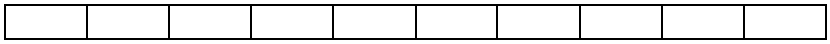
2º ejemplo:

En este ejemplo se explica como se manejan los vectores.

Supongamos que hemos definido una variable a de tipo int y una tabla t de 10 elementos de tipo int escribiendo lo siguiente:

```
int a;
int t[10];
```

t



Inicialmente tanto la variable a como los elementos de la tabla t tendrán un valor indefinido.

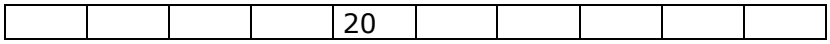
Supongamos ahora que al 4º elemento de la tabla t le queremos asignar el valor 20. Se haría de la siguiente manera:

```
t[4] = 20;
```

Como consecuencia, obtendríamos lo siguiente:

a

t



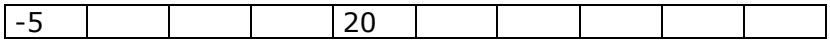
Ahora, para asignar el valor –5 al elemento de la posición 0 tendríamos que escribir lo siguiente:

```
t[0] = -5;
```

Como consecuencia, obtendríamos lo siguiente:

a

t



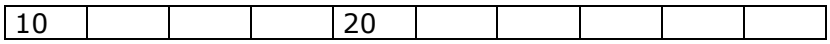
A continuación, en el elemento de la posición 0 guardaremos lo que hay en la posición 4 más lo que hay en la posición 0 multiplicado por dos:

```
t[0] = t[4] + (t[0] * 2);
```

Como consecuencia, obtendríamos lo siguiente:

a

t



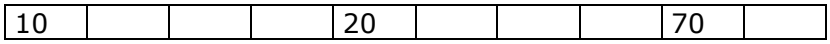
Seguidamente, a la variable a le asignaremos el valor 7 y luego en la posición a+1 de la tabla guardaremos el valor de la posición 0 multiplicado por a:

```
a = 7;
t[a + 1] = t[0] * a;
```

Como consecuencia, obtendríamos lo siguiente:

a

t



1.2 Vectores y funciones

Cuando un vector se pasa como argumento de una función, no se tiene que especificar el número de elementos del vector.

Una función puede cambiar los valores de los elementos de de un vector que sea argumento suyo.

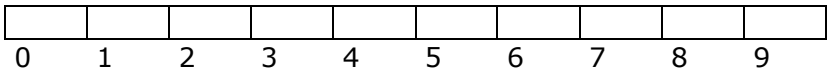
2 EJEMPLOS

2.1 Cómo rellenar la tabla t pidiendo números al usuario

Supongamos que hemos definido una tabla de 10 elementos y que ahora queremos rellenar esa tabla con números pidiendo al usuario que los introduzca por teclado. Existen dos formas de hacerlo:

int t[10];

t



En la primera versión al usuario se le pedirán los elementos uno a uno.

```
int i, num;

i = 0; /* la variable i se utilizará como índice. En cada momento, la
          variable i indicará la primera posición libre de la tabla.
          Además, también se utilizará para saber en cada momento cuántos
          números se han solicitado hasta entonces. */

 while (i < 10)
{
  printf("\nIntroduce un número entero para la posición %d: ", i);
  scanf("%d", &num);
  t[i] = num;
  i = i + 1;
}
```

En esta versión, para pedir cada elemento, se presentará un mensaje al usuario. Cuando teclee cada elemento tendrá que pulsar return. A continuación, se le solicitará el siguiente elemento, etcétera.

Al pedirle números al usuario, se vería lo siguiente en la pantalla:

Pantalla

```
Introduce un número entero para la posición 0: 10
Introduce un número entero para la posición 1: -8
Introduce un número entero para la posición 2: 29
Introduce un número entero para la posición 3: 7
Introduce un número entero para la posición 4: 20
Introduce un número entero para la posición 5: 34
Introduce un número entero para la posición 6: -8
Introduce un número entero para la posición 7: 22
Introduce un número entero para la posición 8: 1
Introduce un número entero para la posición 9: 19
```

La tabla quedaría así:

t

10	-8	29	7	20	34	-8	22	1	19
0	1	2	3	4	5	6	7	8	9

En la segunda versión, el usuario tendrá que teclear todos elementos uno detrás de otro (separados por un espacio en blanco).

```
  int i, num;

  i = 0; /* la variable i se utilizará como índice. En cada momento, la
            variable i indicará la primera posición libre de la tabla.
            Además, también se utilizará para saber en cada momento cuántos
            números se han solicitado hasta entonces. */

  printf("\nIntroduce los 10 elementos del la tabla separados por un espacio
y al final pulsa return: ");

   while (i < 10)
   {
    scanf("%d", &num);
    t[i] = num;
    i = i + 1;
   }
```

En esta version, al usuario se le mostrará un único mensaje. Después de teclear todos los elementos separados por un espacio, habrá que teclear return. Por lo tanto, en este caso sólo hay que pulsar return una única vez, al final.

Al pedirle los números al usuario, veríamos lo siguiente por pantalla:

Pantalla

```
Introduce los 10 elementos del la tabla separados por un espacio
y al final pulsa return: 10 -8 29 7 20 34 -8 22 1 19
```

La tabla quedaría así:

t

10	-8	29	7	20	34	-8	22	1	19
0	1	2	3	4	5	6	7	8	9

2.2 Función que suma los z primeros elementos de la tabla t

Esta funcion calculará y devolverá la suma de los elementos entre las posiciones 0 y z-1 de la tabla t.

Prototipo de la función:

int calcular_suma_elementos (int z, int t[]);

En el prototipo y la definición de la función no es necesario especificar el número de elementos del vector. Por eso se ha puesto t[]. De hecho, lo importante no es saber cuántos elementos tiene la tabla, sino cuántos elementos hay que sumar y eso nos lo dice el parámetro z.

Definición de la función:

```
int calcular_suma_elementos (int z, int t[])

 /* Esta función calcula y devuelve la suma de los z primeros elementos de la tabla
t */

 { /* llave de inicio de la función. */
   int suma, i;

   suma = 0; /* almacenará la suma de los elementos */

   i = 0; /* la variable i se utiliza como índice. De hecho, la tabla t ha de ser
             recorrida posición a posición y en cada momento, la variable i indicará en
             qué posición de la tabla estamos. */

    while (i < z)
    {
      suma = suma + t[i];
      i = i + 1;
    }
   return(suma);
 } /* Llave de fin de función. */
```

Uso de la función:

Supongamos que hemos definido una tabla de 10 elementos y que ya la hemos rellenado:

int b[10];

b

10	-8	29	7	20	34	-8	29	7	20
0	1	2	3	4	5	6	7	8	9

Si quisiéramos calcular la suma de los 4 primeros elementos de la tabla b y guardarlos en la tabla v, tendríamos que escribir lo siguiente:

v = calcular_suma_elementos (4, b);

Como consecuencia de esta instrucción, en la variable v se almacenaría el valor 38.

Como también se ha visto en este ejemplo, a la hora de utilizar una función no hay que especificar el número de elementos de las tablas que se pasan como argumento y ampoco hay que poner[].

2.3 Función que almacena los z primeros elementos de la tabla t1 incrementados en la tabla t2

Esta función almacenará loe elementos de la tabla t1 desde la posición 0 hasta la posición z-1 incrementados en las mismas posiciones de la tabla t2. Por ejemplo, en la 3º posiciñon de la tabla t2 se almacenará el valor de la 3º posición de la tabla t1 más 1.

Prototipo de la función:

void copiar_incrementando (int z, int t1[], int t2[]);

En este caso, los datos son z y t1 y el resultado es t2. Tener como resultado una tabla es como tener un grupo de resultados. Por eso, esta función es del 5º tipo.

Hay que recordar que en el prototitpo y en la definición no es necesario especificar el número de elementos de los vectores.

En este caso, la tabla t2 es el resultado de la función, pero a pesar de ser el resultado, no hay que poner * o &.

Definición de la función:

```
void copiar_incrementando (int z, int t1[], int t2[])

 /* Esta función copia los primeros z elementos de la tabla t1 incrementados
    en la tabla t2. */

 {
   int i;

   i = 0; /* la variable i se utiliza como índice de las tablas t1 y t2. */

   while (i < z)
    {
      t2[i] = t1[i] + 1;
      i = i + 1;
    }
 }
```

2.4 Función que incrementa los primeros z elementos de la tabla t

Esta función incrementa los valores de los elementos entre las posiciones 0 y z-1 de la tabla t.

- Prototipo de la función:

  ```
  void incrementar_tabla (int z, int t[]);
  ```

  En este caso, z es dato y t es, al mismo tiempo, dato y resultado. De hecho, a la función se le pasa la tabla t como dato y la función, tras incrementar los primeros z elementos de la tabla, la devuelve como resultado. Por lo tanto, el valor de t cambiará. Por eso esta función es de tipo 7.

- Definición de la función:

```
void incrementar_tabla (int z, int t[])

 /* Función que incrementa el valor de los primeros z elementos de la tabla t.
*/
```

```
{ /*
  int i;

  i = 0; /* La variable I se utiliza como índice de la tabla t. */

  while (i < z)
   {
    t[i] = t[i] + 1;
    i = i + 1;
   }
}
```

- Uso de la función:

  Supongamos que hemos definido una tabla de 15 elementos enteros y que utilizando la función *rellenar_tabla* hemos almacenado unos valores en las primeras 6 posiciones de la tabla b:

```
int b[15];
rellenar_tabla (6, b);
```

  Por tanto, por ejemplo nos encontraremos con la siguiente situación:

b

10	-8	29	7	20	34			...	
0	1	2	3	4	5	6	7	...	14

  Si quisiéramos incrementar el valor de las 4 primeras posiciones de la tabla b, tendríamos que escribir lo siguiente:

```
Incrementar_tabla (4, b);
```

  Y la tabla quedaría así:

b

11	**-7**	**30**	**8**	20	34			...	
0	1	2	3	4	5	6	7	...	14

  Como también se ha visto en este ejemplo, al utilizar una función, en las tablas que se pasan como argumento no hay que especificar número de elementos y tampoco hay que poner [].

3 EJERCICIOS RESUELTOS

3.1 Programa que presenta los elementos de una tabla del primero al último y del último al primero

Es programa de este ejercicio le tiene que pedir al usuario 10 números y guardarlos en una tabla de 10 elementos. Tras guardar los números en la tabla, los elementos se presentarán en el orden en el que están y en orden inverso.

```
#include <stdio.h>


void main()
{ /* llave de comienzo de function main */
  int tau[10];
  int i, num;
```

```
  system ("cls"); /* Para limpiar la pantalla. */

  /* Primero se rellena la tabla */

  i = 0; /* Índice de la tabla tau. */

  while (i < 10)
   {
     printf("\nTeclea un número entero para la posición %d: ", i);
     scanf("%d", &num);
     tau[i] = num;
     i = i + 1;
   }

  /* A continuación la tabla se mostrará tal cual */

  i = 0; /* Índice de la tabla tau. */

  printf("\n");

  while (i < 10)
   {
     printf("%d  ", tau[i]);
     i = i + 1;
   }


  i = 9; /* Índice de la tabla tau. */

  printf("\n");

  while (i >= 0)
   {
     printf("%d  ", tau[i]);
     i = i - 1;
   }

  printf("\Pulsa una tecla para finalizar.");
  getch(); /* El ordenador esperará a que pulsemos una tecla. */
}   /* fin de función main */
```

3.2 Programa que indica en qué posición de la tabla aparece un número y en total cuántas veces aparece

En este ejercicio, para empezar hay que pedir 10 números al usuario y guardarlos en una tabla de 10 elementos. Tras guardar los números, se pedirá otro número y se mostrará en qué posición de la tabla aparece ese número y cuántas veces aparece en total.

Funciones a utilizar:

- *calcular_número_apariciones*: función que, dados como argumentos los número enteros z y x y la tabla t, devuelve cuántas veces aparece el número x entre los primeros z elementos de la tabla t.

```
#include <stdio.h>

/* Mención o prototipo de la función que se utilizará en el programa. */

int calcular_numero_apariciones (int z, int x, int t[]);
```

```
/* Programa principal */

void main()
{ /* comienzo de programa principal */

  int tau[10];
  int num, i, num_apa;

  system ("cls"); /* Para limpiar la pantalla. */

  i = 0; /* índice de la tabla tau. Indica la primera posición libre de la
            tabla tau. Además también se utilizará para saber cuántos números
            se han pedido hasta entonces en cada momento. */

  printf("\nIntroduce 10 elementoz separados por espacios y después "
         "pulsa return: ");

  while (i < 10)
   {
     scanf("%d", &num);
     tau[i] = num;
     i = i + 1;
   }

  /* A continuación se pide un número */

  printf ("\nTeclea un número entero: ");
  scanf ("%d", &num);

  /* a continuación se estudian las apariciones */

  num_apa = calcula_numero_apariciones (10, num, tau);
  printf ("\nEl número %d aparece %d veces.", num, num_apa);

  /* Finalmente se muestran las posiciones */

  if (num_apa != 0)
   {
     i = 0;

     printf ("\nPosiciones: ");

     while (i < 10)
      {
        if (tau[i] == num)
         {
           printf ("%d  ", i);
         }
        i = i + 1;
      }
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /*
} /* fin de la function main */



                        /* Definición de la función */

int calcula_numero_apariciones (int z, int x, int t[])

 /* Esta función devuelve cuántas veces aparece el número x en las primeras
    z posiciones de la tabla t. */

 {
   int cont, i;
```

```
   i = 0; /* índice de la tabla t. */

   cont = 0; /* para contar las apariciones de x */

   while (i < z)
    {
      if (t[i] == x)
       {
         cont = cont + 1;
       }
      i = i + 1;
    }

   return (cont);
 }
```

3.3 Programa que dice si un número aparece en una tabla o no

Este ejercicio pide iniciamente al usuario que introduzca 10 números y los guarda en una tabla de 10 elementos. Tras guardar los elementos en la tabla, pedirá otro número y decidirá si aparece en la tabla o no.

Funciones a utilizar:

- *Aparece_o_no*: dados como argumentos los números enteros z y x, y la tabla t, la función devolverá 1 si el número x aparece entre los z primeros de la tabla t y 0 si no aparece. Esta función es de tipo 1.

```
#include <stdio.h>

/* Prototipo de la función que se utiliza en el programa. */

int aparece_o_no (int z, int x, int t[]);

/* Función principal */

void main()
{ /* Comienzo de la función principal */

  int tau[10];
  int num, i;

  system ("cls"); /* Para limpiar la pantalla. */

  i = 0; /* índice de la tabla tau. */

  printf("\nIntroduce los 10 elementos de la tabla separados por espacios"
         "y después pulsa return: ");

  while (i < 10)
   {
     scanf("%d", &num);
     tau[i] = num;
     i = i + 1;
   }

  /* A continuación se pide un número*/

  printf ("\nTeclea un número entero: ");
  scanf ("%d", &num);

  /* A contiinuación se estudian las apariciones */
```

```
  if (aparece_o_no (10, num, tau) == 1)
   {
     printf ("\nEn número %d por lo menos aparece una vez.", num);
   }
  else
   {
     printf ("\nEl número %d no aparece.", num);
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
} /* fin de la función principal */

                        /* Definición de la función */

int aparece_o_no (int z, int x, int t[])

 /* Esta función devuelve un 1 si el número x aparece al menos una vez en las
    z primeras posiciones de la tabla t, y 0 si no aparece. */

 {
   int encontrar, i;

   i = 0; /* índice de la tabla t. */

   encontrar = 0; /* cuando el valor de la variable encontrar es 0, quiere
                     decir que aún no hemos encontrado el valor x en la tabla.
                     Si en un momento dado la variable pasa a tener valor 1,
                     Significará que hemos encontrado x en la tabla t. */

   /* En esta función, cuando se encuentra el número x por primera vez, la
      búsqueda finalizará, porque es suficiente para saber que x aparece. */

   while ((i < z) && (encontrar == 0))
    {
      if (t[i] == x)
       {
         encontrar = 1;
       }
      i = i + 1;
    }

   return (encontrar);
 }
```

3.4 Programa que compara dos vectores con el mismo número de elementos

En este ejercicio se almacenan los elemento de los dos vectores en dos tablas y a continuación se comparan.
Ambos vectores tendrán el mismo númer de elementos pero este número se le pedirá al usuario al comienzo del programa. Aún así, el número máximo de elementos será 15.

Función a utilizar

- *comparar_tablas*: Dados como argumentos el número entero z y las tablas t1 y t2, la función devuelve el valor 1 si los primeros z elementos de la tabla t1 son iguales a los primeros z elementos de la tabla t2, si no, devolverá el valor 0.

```
#include <stdio.h>

/* Prototipo de la función que utiliza el programa */

int comparar_tablas (int z, int t1[], int t2[]);

/* Función principal */

void main()
{ /* Comienzo de la función principal */
  int vec1[15];
  int vec2[15];
  int numelem, comp, I, num;

  system ("cls"); /* Para limpiar la pantalla. */

  /* Primero se pregunta al usuario cuántos elementos tendrán los vectores.
     Pueden tener desde al menos 1 elemento hasta 15 */

  printf ("\nIndica el núm. de elementos (como mínimo 1 y como máximo 15): ");
  scanf ("%d", &numelem);

  while ((numelem < 1) || (numelem > 15))
   {
     printf ("\nEl número tecleado no es correcto.");
     printf ("\nIndica el núm. de elementos (como mínimo 1 y como máximo 15):
");
     scanf ("%d", &numelem);
   }


  /* A continuación se piden los elementos del primer vector */

  printf("\nIntroduce los %d elementos del primer vector separados por "
         "espacios y después pulsa return: ", numelem);
  i = 0;
  while (i < numelem)
   {
     scanf("%d", &num);
     vec1[i] = num;
     i = i + 1;
   }

  /* ahora se puden los elementos del segundo vector */

  printf("\nIntroduce los %d elementos del segundo vector separados por "
         "espacios y después pulsa return: ", numelem);
  i = 0;
  while (i < numelem)
   {
     scanf("%d", &num);
     vec2[i] = num;
     i = i + 1;
   }

  /* Para realizar la comparación se utiliza la función comparar_vectores */

  comp = comparar_vectores (numelem, vec1, vec2);

  /* Para finalizar se muestra el resultado de la comparación */

  if (comp == 1)
   {
     printf ("\nLos vectores son iguales.");
   }
  else if (comp == 0)
   {
```

```
     printf ("\nLos vectores no son iguales.");
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
} /* fin de la función principal */


                        /* Definición de la función */

int comparar_vectores (int z, int t1[], int t2[])

  /* Esta función devuelve un 1 si los primeros z elementos de las tablas t1 y
     t2 son iguales, y 0 si no lo son. */

 {
   int iguales, i;

   i = 0; /* índice de las tablas t1 y t2. */

   iguales = 1; /* mientras el valor de la variable iguales es 1, significa
                   que las posiciones comparadas hasta ahora son iguales.
                   Cuando el valor de la variable iguales pase a ser 0,
                   Significará que se han encontrado elementos que son
                   distintos en t1 y t2. */

   /* cuando se encuentren dos elementos distintos, no se continuará con la
      comparación, pues sabremos que las tablas ya son distitntas. */

   while ((i < z) && (iguales == 1))
    {
      if (t1[i] != t2[i])
       {
         iguales = 0;
       }
      i = i + 1;
    }

   return (iguales);
 }

                           /*******************/
```

3.5 Programa que indica si los elementos de un vector están ordenados en orden ascendente

En este ejercicio almacenamos los valores de un vector en una tabla y a continuación comprobamos si están ordenados de forma ascendente, mostrando un mensaje.

El vector puede tener como máximo 20 elementos, pero el número de elementos puede variar en cada ejecución. Por eso, al comienzo del programa, se deberá preguntar al usuario cuál es el número de elementos.

Función a utilizar:

- *decide_orden*: Dados como argumentos el número entero z y la tabla t, si los z primeros elementos de la tabla t están ordenados de forma ascendente, la función devolverá un 1, si están en orden descendente, devolverá 0 y si no, devolverá -1.

```
#include <stdio.h>

/* Prototipo de la función que utiliza el programa */

int decide_orden (int z, int t[]);

/* Función principal */

void main()
{ /* Comienzo de la función principal */

  int vec[20]; /* como el vector puede tener un máximo de 20 elementos, si
                  definimos 20 eementos sabremos que cualquier vector que
                  teclee el usuario cabrá bien en la tabla. */

  int numelem, orden, i, num;

  system ("cls"); /* Para limpiar la pantalla. */

  /* Pedir como máximo 20 elementos al usuario. */

  printf ("\nIndica el número de elementos(como mínimo 1 y máximo 20): ");
  scanf ("%d", &numelem);

  while ((numelem < 1) || (numelem > 20))
   {
     printf ("\nEl número tecleado no es correcto.");
     printf ("\nIndica el número de elementos(como mínimo 1 y máximo 20): ");
     scanf ("%d", &numelem);
   }

  /* A continuación se piden los elementos del vector */

  printf("\nIntroduce los %d elementos del vector separados por espacios y "
         "después pulsa return: ", numelem);
  i = 0;
  while (i < numelem)
   {
     scanf("%d", &num);
     vec[i] = num;
     i = i + 1;
   }

  /* Para saber si la tabla está ordenada, se utiliza la función decide_orden
*/

  orden = decide_orden (numelem, vec);

  /* Para finalizar, se le muestra al usuario la decisión */

  if (orden == 1)
   {
     printf ("\nLos elementos del vector están en orden ascendente.");
   }
  else
   {
     printf ("\nLos elementos del vector no están en orden ascendente.");
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
}   /* fin de la función principal */

                        /* Definición de la función */

int decide_orden (int z, int t[])

 /* Esta función devuelve un 1 si los z primeros elementos de la tabla t están
```

```
   en orden ascendente, 0 si están en orden descendente y -1 en otro caso. */

{
  int ascendente, descendente, i;

  i = 0; /* índice de la tabla t. */

  ascendente = 1;

/* Los elementos de la tabla se recorren y se comprueban de dos en dos
   Mientras las parejas comparadas estén en orden ascendente, se continua,
   Pero si se encuentra una pareja que no lo cumple, no se continua con la
   comparación. */                    →

/* Los primeros elementos a comparar son el 0 y el 1, lueo el 1 y el 2 y así
   hasta acabar con las posiciones z-2 y z-1. Por tanto, ha que recorrer las
   posiciones entre 0 y z-2. */


  while ((i < (z - 1)) && (ascendente == 1))
   {
     if (t[i] > t[I + 1])
      {
        ascendente = 0;
      }
     i = i + 1;
   }

  if (ascendente == 1)
   {
     return (1);
   }
  else /* si no está en orden ascendente, se comprueba si está en
          descendente. */
   {
     i = 0;
     descendente = 1;

     while ((i < (z - 1)) && (descendente == 1))
      {
        if (t[i] < t[i + 1])
         {
           descendente = 0;
         }
        i = i + 1;
      }

     if (descendente == 1)
      {
        return (0);
      }
     else /* Si tampoco está en descendente, se devuelve -1. */
      {
        return (-1);
      }
   }
}
```

3.6 Programa que encuentra y muestra los números primos de una tabla

El programa de este ejercicio para comenzar pide al usuario 20 números y los guarda en una tabla de 20 elementos. Tras ello, se recorrerá la tabla mostrando los elementos que son primos.

Función a utilizar

- *Es_primo*: dado como argumento un número ≥ 1, dice si ese número es primo o no. Si es primo, la función devuelve 1 y si no, devuelve 0.

```
#include <stdio.h>

/* Prototipo de la función que utiliza el programa */

int es_primo (int x);

/* Función principal */

void main()
{ /* Comienzo de la función principal */

  int tau[20];
  int i, num;

  system ("cls"); /* Para limpiar la pantalla. */

  /* Primero se rellena la tabla de núermos positivos */

  i = 0; /* índice de la tabla tau. */

  while (i < 20)
   {
     printf("\nTeclea para la posición %d un número entero >= 1: ", i);
     scanf("%d", &num);
     while (num< 1)
      {
        printf("\nEl número introducido no es válido.");
        printf("\nTeclea para la posición %d un número entero >= 1: ", i);
        scanf("%d", &num);
      }

     tau[i] = num;
     i = i + 1;
   }

                                                                          →
  /* Ahora se recorre la tabla para buscar los primos y mostrarlos */

  i = 0; /* índice de la tabla tau. */

  while (i < 20)
   {
     if (es_primo(tau[i]) == 1)
      {
        printf("\nEl elemento de la posición %d es primo: %d", i, tau[i]);
      }

     i = i + 1;
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
}   /* fin de la función principal */


                        /* Definición de la función */

int es_primo (int x)

 /* Dado como argumento un número x >= 1, si es primo, la función devuelve un
    1, y si no, devuelve un 0. */
```

```
{
  int num, cont;

  cont = 0; /* variable para contar los divisores de x */
  num= 1; /* variable para pasar uno a uno los números del 1 a x */

  while (num<= x)
   {
     if (x % num == 0)
      {
        cont = cont + 1;
      }
     num = num + 1;
   }

  if (cont == 2)
   {
     return (1);
   }
  else
   {
     return (0);
   }
}
```

3.7 Programa que ordena los elementos de un vector en orden ascendente

El programa de este ejercicio debe ordenar los elementos de un vector para que queden en orden ascendente.

El vector puede tener como máximo 20 elementos, pero este número de elementos puede variar en cada ejecución del programa. Por ellos, al comienzo del programa se deberá preguntar al usuario cuál es el número de elementos.

Funciónes a utilizar

- *ordenar_tabla*: Dados como argumentos los número enteros p1 y p2 y la tabla t, es la función que ordena de forma ascendente los elementos de la tabla t entre las posiciones p1 y p2. Esta función utilizará a su vez la función *posición_menor*.

- *posición_menor*: Dados como argumentos los número enteros p1 y p2 y la tabla t, es la función que devuelve la posición del número menor entre las posiciones p1 y p2. Si el número menor está repetido, devuelve la posición del que aparece primero.

```
#include <stdio.h>

/* Prototipos de las funciones que se utilizarán en el programa. */

int posición_menor (int p1, int p2, int t[]);
void ordenar_tabla (int p1, int p2, int t[]);

                          /* Función principal */

void main()
{ /* Comienzo de la función principal */

  int tau[20];

  int numelem, num, i;

  system ("cls"); /* Para limpiar la pantalla. */
```

```
  /* Preguntar al usuario el número de elementos (como máximo 20). */

  printf ("\nIndica el número de elementos (mínimo 1 y máximo 20): ");
  scanf ("%d", &numelem);

  while ((numelem < 1) || (numelem > 20))
   {
     printf ("\nEl número tecleado no es correcto.");
     printf ("\nIndica el número de elementos (mínimo 1 y máximo 20): ");
     scanf ("%d", &numelem);
   }

  /* A continuación se piden los elementos del vector*/

  printf("\nIntroduce los %d elementos del vector separados por espacios y "
         "después pulsa return: ", numelem);
  i = 0;
  while (i < numelem)
   {
     scanf("%d", &num);
     tau[i] = num;
     i = i + 1;
   }

  /* Se ordena la tabla de forma ascendente */

  ordenar_tabla (0, numelem - 1, tau);

  /* Para finalizar, se muestra la tabla ordenada */

  i = 0;

  while (i < numelem)
   {
     printf("%d  ", tau[i]);
     i = i + 1;
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
} /* fin de la función principal */

                        /* Definición de las funciones*/

void ordenar_tabla (int p1, int p2, int t[])

 /* Esta función ordena los elementos entre las posiciones p1 y p2 de la tabla
    t en orden ascendente. */

 {
   int posmen, i, aux;

   i = p1; /* índice de la tabla t. */

   while (i < p2)
    {
      posmen = posición_menor (i, p2, t);
      aux = t[i];
      t[i] = t[posmen];
      t[posmen] = aux;
      i = i + 1;
    }
 }

                              /*********************/
```

```
int posición_menor (int p1, int p2, int t[])

 /* Función que devuelve la posición del elemento más pequeño de la tabla t
entre las posiciones p1 y p2. */

 {
   int elmenor, posición, i;

   i = p1; /* índice de la tabla t. */

   elmenor = t[p1]; /* Almacena el menor valor de los recorridos hasta el
                       momento. */

   posición = p1; /* Almacena la posición del menor valor de los recorridos
                     hasta el momento. */


   while (i <= p2)
    {
      if (t[i] < elmenor)
       {
         elmenor = t[i];
         posición = i;
       }
      i = i + 1;
    }

   return (posición);
 }
```

3.8 Programa que suma dos polinomios que pueden ser de distinto grado

El programa de este ejercicio calcula y muestra el polinomio resultante de sumar dos polinomios que pueden ser de distinto grado.

Los dos polinomios iniciales y el que se obtiene como resultado de la suma se deben almacenar en tres tablas.

Antes de preguntar por los coeficientes de cada polinomio, el programa deberá preguntar al usuario el grado de cada polinomio.

El grado máximo de los polinomios puede ser 20, por lo tanto los polinomios tendrán un máximo de 21 coeficioentes y por lo tanto, se definirán tablas de 21 elementos.

Funciones a utilizar:

- *suma_polinomios*: dados como argumentos los grados de ambos polinomios (g1 eta g2) y las tablas (pol1 y pol2) que tienen los coeficientes de esos polinomios, la función devuelve los coeficientes (pol3) del nuevo polinomio resultante de la suma de los otro dos polinomios. Esta función utilizará las funciones *suma_tablas* y *copia_tabla*.

- *suma_tablas*: Dados como argumentos los número enteros p1 y p2 y las tablas t1, t2 y t3, la función suma los elementos de las tablas t1 y t2 posición a posición y deja el resultado en la misma posición de la tabla t3.

- *copia_tabla*: dados como argumentos los números enteros p1 y p2 y las tablas t1 y t2, la función copia los elementos de la tabla t1 desde la posición p1 hasta la p2 en las mismas posiciones de la tabla t2.

```
#include <stdio.h>


/* Prototipos de las funciones que se utilizarán en el programak. */

void suma_polinomios (int g1, float pol1[], int g2, float pol2[],
                      float pol3[]);
void suma_tablas (int p1, int p2, float t1[], float t2[], float t3[]);
void copia_tabla (int p1, int p2, float t1[], float t2[]);

/* Función principal */

void main()
{ /* Comienzo de la función principal */
  float poli1[21]; /* Si el máximo grado de los polinomios es 20
                      como mucho habrá 21 coeficientes. */

  float poli2[21];
  float poli3[21];
  int grado1, grado2, grado3, i;
  float coef;

  system ("cls"); /* Para limpiar la pantalla. */

  /* Para empezar se pide el grado del primer polinomio */

  printf ("\nIndica el grado del polinomio - "
          "debe estar entre ([0, %d]): ", 20);
  scanf ("%d", &grado1);

  while ((grado1 < 0) || (grado1 > 20))
   {
     printf ("\nEl número tecleado no es correcto.");
     printf ("\nIndica el grado del polinomio - "
           "debe estar entre ([0, %d]): ", 20);
     scanf ("%d", &grado1);
   }

  /* a continuación se piden los coeficientes del 1° polinomio */

  printf("\nAhora teclea los coeficientes del primer polinomio.");

   i = 0; /* indica la primera posición libre de la tabla. */

   while (i <= grado1)
    {
      printf("\nTeclea el coeficiente de grado %d: ", i);
      scanf("%d", &coef);
      poli1[i] = coef;
      i = i + 1;
    }

  /* Ahora se pide el grado del segundo polinomio */

  printf ("\nIndica el grado del segundo polinomio "
          "debe estar entre ([0, %d] ): ", 20);
  scanf ("%d", &grado2);

  while ((grado2 < 0) || (grado2 > z))
   {
     printf ("\nEl número tecleado no es correcto.");
     printf ("\nIndica el grado del segundo polinomio "
              "debe estar entre ([0, %d] ): ", 20);
     scanf ("%d", &grado2);
   }

  /* Y a continuación se piden los coeficientes del segundo polinimio */
```

```
  printf("\nAhora teclea los coeficientes del 2. polinomio.");

   i = 0; /* indica la primera posición libre de la tabla. */

   while (i <= grado2)
    {
      printf("\nTeclea el coeficiente de grado %d: ", i);
      scanf("%d", &coef);
      poli2[i] = coef;
      i = i + 1;
    }

  /* Como los polinomios pueden tener distinto grado, para comenzar se calcula
     el grado del nuevo polinomio. */

  if (grado1 >= grado2)
   {
     grado3 = grado1;
   }
  else
   {
     grado3 = grado2;
   }

  /* A continuación, se suman los polinomios */

  suma_polinomios (grado1, polinomioa1, grado2, polinomioa2,
                   polinomioa3);

  /* Para finalizar, se muesran los tres polinomios */

  system("cls");

  printf("\n1° polinomio: \n");

   i = 0;

   while (i <= grado1)
    {
      printf("%d  ", poli1[i]);
      i = i + 1;
    }

  printf("\n2° polinomio: \n");

   i = 0;

   while (i <= grado2)
    {
      printf("%d  ", poli2[i]);
      i = i + 1;
    }

  printf("\n3° polinomio: \n");

   i = 0;

   while (i <= grado3)
    {
      printf("%d  ", poli3[i]);
      i = i + 1;
    }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
} /* fin de la función principal */
```

```
                        /* Definición de las funciones*/

void suma_polinomios (int g1, float pol1[], int g2, float pol2[],
                      float pol3[])

 /* Esta función devuelve los coeficientes del polinomio (p3) que se obtiene
    de sumar los polinomios p1 y p2 de grados g1 y g2. */

 {
   if (g1 >= g2)
    {
      suma_tablas (0, g2, pol1, pol2, pol3);
      copia_tablas (g2 + 1, g1, pol1, pol3);
    }
   else
    {
      suma_tablas (0, g1, pol1, pol2, pol3);
      copia_tablas (g1 + 1, g2, pol2, pol3);
    }
 }
                             /********************/

void suma_tablas (int p1, int p2, float t1[], float t2[], float t3[])

 /* Esta función suma los elementos parejos de las tablas t1 y t2 entre las
    posiciones p1 y p2 almacenándolos en la misma posición de t3. */

 {
   int i;

   i = p1; /* indice de las tablas t1, t2 y t3. */

   while (i <= p2)
    {
      t3[i] = t1[i] + t2[i];
      i = i + 1;
    }
 }
                             /********************/

void copia_tabla (int p1, int p2, float t1[], float t2[])

 /* Esta función copia los elementos de la tabla t1 entre las posiciones p1 y
    p2 en las mismas posiciones de la tabla t2. */

 {
   int i;

   i = p1; /* índice de las tablas t1 y t2. */

   while (i <= p2)
    {
      t2[i] = t1[i];
      i = i + 1;
    }
 }
                             /********************/
```

3.9 Programa que muestra las calificaciones correspondientes a las notas de una clase de alumnos (la última nota negativa no se almacenará)

El programa de este ejercicio pide las notas de los alumnos, las guarda en una tabla ya continuación debe mostrar la calificación correspondiente a cada nota.Como última nota se

proporcionará un número negativo. Por tanto, cuando el programa reciba un número negativo, debe finalizar el proceso de petición de notas.

Como máximo se introducirán 100 notas positivas y **el último número negativo no se almacena**. En consecuencia, será suficiente con tener una tabla de 100 elementos para guardar las notas.

```
#include <stdio.h>


void main()
{ /* Comienzo de la función principal */
  float tablanotas[100];

  int numnotas, i;
  float unanota;

  system ("cls"); /* Para limpiar la pantalla. */

  /* Para comenzar, se piden las notas */

  i = 0; /* índice y contador del número de notas introducido. */

  printf("\nA continuación se pedirán las notas de una en una. La última "
         "debe ser negativa y como máximo pueden ser %d.", 100);
  printf("\nTeclea la  %dª nota: ", i);
  scanf("%d", &unanota);

  while (unanota >= 0)
   {
     tablanotas[i] = unanota;
     i = i + 1;
     printf("\nTeclea la  %dª nota: ", i);
     scanf("%d", &unanota);
   }

  numnotas = i;

                                                                    →
  /* Ahora se muestran las calificaciones */

  if (numnotas == 0)
   {
      printf ("\nNo se ha introducido ninguna nota.");
   }
  else
   {
     i = 0;
     while (i < numnotas)
      {
           if ((tablanotas[i] >= 0) && (tablanotas[i] < 5))
            {
              printf ("\nSuspenso.");
            }
           else if ((tablanotas[i] >= 5) && (tablanotas[i] < 7))
            {
              printf ("\nSuficiente.");
            }
           else if ((tablanotas[i] >= 7) && (tablanotas[i] < 9))
            {
              printf ("\nNotable.");
            }
           else if ((tablanotas[i] >= 9) && (tablanotas[i] < 10))
            {
              printf ("\nSobresaliente.");
            }
```

```
          else if (tablanotas[i] == 10)
           {
             printf ("\nMatrícula de Honor.");
           }

          i = i + 1;
      }
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
} /* fin de la función principal */
```

3.10 Programa que muestra las calificaciones correspondientes a las notas de una clase de alumnos (almacenando también la última nota negativa)

El programa de este ejercicio debe pedir las notas de los alumnos, guardarlas en una tabla y a continación mostrar la calificación correspondiente a cada nota.

Como última nota se introducirá un número negativo. Por lo tanto, cuando el programa reciba una número negativo, el proceso de petición de notas debe terminar.

Como máximo se introducirán 100 notas **y el último número negativo también se debe almacenar**. Por tanto, con tener una tabla de 101 elementos será suficiente.

```
#include <stdio.h>


void main()
{ /* Comienzo de la función principal */
  float tablanotas[101];
  int i;
  float unanota;

  system ("cls"); /* Para limpiar la pantalla. */

  /* Para comenzar se piden las notas.
     Se supone que el usuario sabe que como mucho hay que introducir 100 notas
     Y que el último número debe ser negativo. NO se harán comprobaciones.*/

  i = 0; /* índice de la tabla. */
  printf("\nA continuación se pedirán las notas una a una. La última debe "
         "negativa y como máximo se pueden introducir %d notas (sin contar "
         "el negativo.", 100);
  printf("\nTeclea la %dº nota: ", i);
  scanf("%d", &unanota);

  while (unanota >= 0)
   {
     tablanotas[i] = unanota;
     i = i + 1;
     printf("\nTeclea la %dº nota: ", i);
     scanf("%d", &unanota);
   }

  tablanotas[i] = unanota;                                              →

  /* A continuación se muestran las calificaciones. Como en este caso se ha
  guardado el último número negativo en la tabla, se continuará hasta que
  aparezca. */

  i = 0;

  if (tablanotas[i] < 0)
```

```
   {
     printf ("\nNo se ha introducido ninguna nota.");
   }
  else
   {
     while (tablanotas[i] >= 0)
      {
        if ((tablanotas[i] >= 0) && (tablanotas[i] < 5))
         {
           printf ("\nSuspenso.");
         }
        else if ((tablanotas[i] >= 5) && (tablanotas[i] < 7))
         {
           printf ("\nSuficiente.");
         }
        else if ((tablanotas[i] >= 7) && (tablanotas[i] < 9))
         {
           printf ("\nNotable.");
         }
        else if ((tablanotas[i] >= 9) && (tablanotas[i] < 10))
         {
           printf ("\nSobresaliente.");
         }
        else if (tablanotas[i] == 10)
         {
           printf ("\nMatrícula de Honor.");
         }

        i = i + 1;
      }
   }

  printf("\nPulsa una tecla para finalizar.");
  getch(); /* El ordenador espera a que punsemos una tecla.*/
}   /* fin de la función principal */
```