LABORATORIO 8
TABLAS NUMÉRICAS - ARRAYS

1 OBJETIVOS

Con esta práctica, se pretende:

- Perfeccionar en uso de tablas numéricas.
- Utilizar funciones con tablas numéricas: paso de parámetros, llamadas a función etc...

2 MOTIVACIÓN

- Supongamos que queremos cargar diferentes tablas numéricas del mismo tamaño y condición, no hace falta repetir el código tantas veces como tablas a cargar, sino que se llamará a la función correspondiente pasándole como parámetro la tabla que queremos cargar.
- Lo mismo a la hora de visualizar, o en todas aquellas tareas repetitivas que realizamos con las tablas, y que posteriormente se pueden utilizar en el mismo programa u otros.

2.1 El uso de tablas numéricas en funciones en C

A la hora de usar tablas númericas como parámetros en una función hay que tener en cuenta tres cosas: declaración de la función , la definición de la función y la llamada a la función.
Ejm: función que inializa a cero un tabla de tamaño N.

- Declaración:

```
void inicializar_matriz (int limite, int vec[]);
```

- Definición:

```
void inicializar_matriz (int limite, int vec[])
{
      int i;

      for (i=0;i<limite;i++){
            vec[i]=0;
      }
}
```

- Llamada:

```
void main()
{
   /*número de elementos de una tabla */
   int const NELE=10;

   /*Declaración o definición de una tabla de enteros */
   int notas_alum[NELE];
```

```
        inicializar_matriz (NELE, notas_alum);
        ...
    }
```

3 EJERCICIOS

3.1 Ejercicio 1

3.1.1 Objetivo:

El objetivo del ejercicio es probar funciones con tablas y poder identificar los diferentes elementos (declaración, definición y función) y comprobar su funcionamiento.

3.1.2 Ayuda:

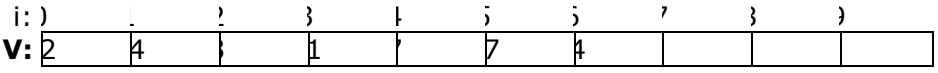
Se da resuelto el enunciado para tomarlo de guía para el resto de ejercicios.

```
#include <stdio.h>
#include <stdlib.h>
#define N 10

int buscar_posicion_en_tabla(int num, int tabla[]);

void main()
{
 int tabla [N],i=0,posicion,num;
 printf("\nDame 10 numeros para almacenar en una tabla.\n");

 for (i=0;i<N;i++)
 {
  printf("\nEl %d:",i);
  scanf("%d",&num);
  tabla[i]=num;
 };
 printf("\nDame un numero");
 printf(" y te dire la primera posicion de dicho numero dentro de la
tabla:");
 scanf("%d",&num);
 posicion=buscar_posicion_en_tabla(num,tabla);
 printf("\nEl numero %d esta en la posicion %d.\n",num,posicion);

 system("PAUSE");
}

int buscar_posicion_en_tabla(int num, int tabla[])
{
 int encontrado=0,i=0;
 while((i<9)&&(encontrado==0))
  {
   if (tabla[i]==num) encontrado=1;
   i++;
  }
 if (encontrado==0) return (-1);
 else return (i-1);
```

3.1.3 Enunciado 1:

Hacer un programa que lea por teclado los elementos de una tabla (no necesariamente ordenada) de tamaño *N*. Después pedirá al usuario un número y calculará la primera posición de dicho número dentro de la tabla. En caso de que el número no se encuentre en el vector, se visualizará un –1.

Ejemplo (suponiendo que *N* vale 10):




```
DAME UN NUMERO: 17
El número 17 está en la posición 5
```

3.2 Ejercicio 2

3.2.1 Objetivo:

El objetivo del ejercicio 2 es poner en práctica los conocimiento adquiridos en el ejercicio anterior.

3.2.2 Ayuda para el enunciado 2:

Utilizar el ejercicio 1.

3.2.3 Enunciado 2:

Igual que el ejercicio anterior, pero con dos notables diferencias:

- La tabla podrá no llenarse, dado que si el usuario teclea un número negativo no se seguirán leyendo más valores.
- El usuario introducirá los elementos de la tabla ordenados de mayor a menor (no es necesario comprobarlo).

3.3 Ejercicio 3

3.3.1 Objetivo:

Dada la especificación de una función o interfaz de una función realizar un programa que la implente y use.

3.3.2 Ayuda para el enunciado 3:

Para realizar este programa utilizar la siguiente interfaz de función:

void insertar_en_tabla(int posicion, int dimension, int tabla[]) donde posición es la posición a partir del cual se buscará el elemento menor y una vez encontrado se colocará en dicha posición después de haber desplazado todos los elementos a la derecha. Dimensión es el número de elementos introducidos en la tabla y tabla es la tabla a modificar. Esta función se encargará de buscar el elemento menor desde la posición x hasta el final y después de desplazar todos los elementos a la derecha, insertar el elemento menor en la posición x.

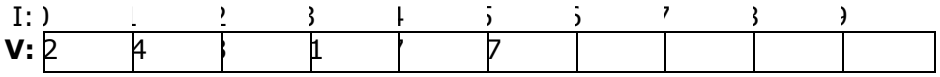
3.3.3 Enunciado 3:

Realizar un programa que solicite números y los vaya introduciendo en una tabla de dimensión *N(Para este programa suponer que N=10)*; la entrada de elementos finaliza

cuando el usuario teclea –1 o la tabla se llena (¡¡CUIDADO!! nunca se dejará que se llene entera, de forma que siempre haya al menos un hueco libre para la inserción). El programa deberá contar la cantidad de números introducidos (excluyendo el –1) y modificar la posición *x* de la tabla (indicada por el usuario), introduciendo en ella el menor elemento encontrado en la tabla a partir de la posición *x* hasta el final. Antes de introducir el menor en la posición x desplazará el resto de elementos a la derecha.

Ejemplo:

DAME HASTA 10 NUMEROS (NEGATIVO PARA FINALIZAR):
12 14 3 11 7 17 –1



DAME LA POSICION X: 3
Una vez realizada la modificación, el array queda así:
12 14 3 **7** 11 7 17

3.4 Ejercicio 4

3.4.1 Objetivo:

El objetivo del ejercicio 4 es crear la interfaz de una función y además realizar un programa que la implente y use.

3.4.2 Ayuda para el enunciado 4:

Para realizar este programa utiliza la función es_capicua, que nos dirá si el contenido de la tabla es capicua o no.

3.4.3 Ejercicio 4

Hacer un programa que, dado un número entero de como mucho N dígitos(suponiendo que *N* sea 5), decida si es capicúa o no.

Ejemplo:

```
DAME UN NUMERO DE HASTA 5 DIGITOS: 1231
El número 1231 no es capicúa.
```