

Práctica de vectores

Objetivo: Conocer el uso de vectores

Tiempo de realización: 2 semana

Cree un programas para probar las siguientes funciones. Puede que pare realizar algunos ejercicios, requiera de los anteriores.

NOTA: Los ejercicios 1 y 2 se crearán en un único archivo.

1) **void lee(float v[],int n);**

Esta función recibe un vector de flotantes y el tamaño del mismo, y rellena sus elementos con valores dados desde teclado.

2) **void imprime(float v[],int n);**

Esta función imprime los elementos del vector por pantalla

//Ejemplo:

```
#include <iostream>
```

```
using namespace std;
```

```
void lee(float v1[],int n)
```

```
{
```

```
    //put your code here
```

```
}
```

```
void imprime(float v1[],int n)
```

```
{
```

```
    //put your code here
```

```
}
```

```
int main(){
```

```
    float v[10];
```

```
    lee(v,10)
```

```
    imprime(v,10);
```

```
}
```

3) void fijaValorAleatorio(int v[],int n,int max);

Esta función pone todos los elementos del vector a valores aleatorio en el rango [0,max]

Ejemplo de generación de números aleatorios

```
#include <ctime> //para time
#include <cstdlib> //para srand y rand
#include <iostream> // para cout
using namespace std;
int main(){
    int secs=time(0);
    srand(secs); //inicializa el generador de numeros aleatorios
    for(int i=0;i<10;i++){
        int n=rand()%50; //genera un numero aleatorio entre 0 y 49
        cout<<n<<endl;
    }
}
```

Para los ejercicios a continuación, se deberá usar los ficheros de test .exe que hay en la web de moodle:

4) float max(float v1[],int n)

Esta función devuelve el valor máximo de un vector.

Principio de resolución:

La idea es utilizar una variable auxiliar (M), que inicialmente se le da el valor del primer elemento del vector (v[0]). Después, se recorre el vector desde la posición i=1 hasta el final. Para cada elemento v[i], se comprueba si es mayor que M. En caso afirmativo, se cambia el valor de M por el valor de v[i]. De esta manera, al final, M vale lo que el mayor valor del vector.

5) float min(float v1[],int n)

Esta función devuelve el valor mínimo de un vector.

6) bool soniguales(float v1[],float v2[],int n);

Esta función deberá indicar si los dos vectores son iguales. Dos vectores son iguales si todos sus elementos lo son. Es decir:

$$v1[i] == v2[i], \forall i = 0 \dots n-1$$

Se asumirá que v1 y v2 tienen tamaño n

7) void suma(float v1[],float v2[],float v3[],int n)

Esta función suma los elementos del vector v1 y v2 y los guarda en v3.

$$v3[i] = v1[i] + v2[i]$$

8) void invierte(float v1[],float v2[],int n)

Esta función guarda en v2 los elementos de v1 pero en orden inverso. El siguiente ejemplo muestra una lista de 6 elementos y el resultado de invertirla:

Antes de invertir: (7, 6, 3, 0, 1, 2)

Después de invertir: (2, 1, 0, 3, 6, 7)

9) int divisores(int val,int div[],int n);

Dado un número natural *val*, obtiene todos los divisores del mismo y los guarda en el vector *divisores*. El tamaño del vector es n, y si el numero tiene mas de n divisores solo se escribirán los n primeros. Es decir, no se deberá exceder el tamaño del vector.

Ejemplo:

Si val=4, **div**[0]=1,**div**[1]=2,**div**[2]=4 y se devuelve 3

Parámetros:

val: valor natural

divisores. vector de elementos donde se guarda la salida

n: tamaño del vector

Devuelve: número de divisores

10) void factoriales(int fact[],int n);

Guarda en el vector *fact* los factoriales de los números desde 0 a n-1. De forma que:

fact[0]=0!, **fact**[1]=1!, ..., **fact**[i]=i!, ..., **fact**[(n-1)]=(n-1)!

Parámetros:

fact: vector donde se guardarán los valores a calcular

n: número de elementos del vector

11) int sum_even(int v[],int n)

Retorna la suma de los valores pares del vector v.

Parámetros:

v: vector con los valores

n: número de elementos del vector v

Devuelve: suma de los valores pares del vector

Ejemplo:

Para el vector {1,3,3,4,6,1,9,8}. La funcion retorna 4+6+8=18

12) int find(int val,int v[],int n)

Retorna la el número de veces que aparece el valor *val* en el vector v

Parámetros:

val: valor a buscar

v: vector con los valores

n: número de elementos del vector v

Devuelve: el número de veces que aparece val en v

Ejemplo : para el vector

val=1

v=1,2,3,1,3,4,1

n=7

La función devuelve 3, que es el número de veces que aparece el valor 1 en el vector v

13) int find_indices(int val,int v[],int n,int indices[])

Determina las posiciones en que el valor val aparece en el vector v, y las guarda en el vector índices. Retorna la el número de veces que aparece el valor val en el vector v

Parámetros:

val: valor a buscar

v: vector con los valores

n: número de elementos del vector v y el vector índices

índices: vector con las posiciones en las que se localiza el valor val en v

Devuelve: el número de veces que aparece val en v

Ejemplo:

val=1

v[] = 1,1,2,3,4,1 (1 aparece en las posiciones 0,1,y 5 del vector v)

n=6

La función guardará en índices los valores

indices[]=0,1,5,?,?,?

y retorna el valor 3

14) void ordena_vector(int v[],int n)

La función ordena en forma ascendente los elementos del vector v utilizando el método de ordenación burbuja

15) bool find_in_sorted_vector(int val,int v[],int n)

La función indica si el elemento val se encuentra en el vector v, que está ordenado, en orden ascendente. **Se asume que el vector v[] YA ESTÁ ORDENADO. NO Tenéis que ordenarlo en la función!**