TALLER 2 ORDENAMIENTO REGISTRO, ORDENAMIENTO Y BÚSQUEDA DE TIEMPOS DEPORTIVOS

Situación problema

Un entrenador necesita un programa en lenguaje C para registrar los tiempos obtenidos por los deportistas en una prueba de velocidad.

Cada deportista realiza una prueba y su resultado se registra en **segundos**, por ejemplo:

```
11.25
10.87
12.14
10.52
11.03
```

En una prueba de velocidad, **el menor tiempo representa el mejor resultado**.

El programa debe permitir registrar los tiempos, ordenarlos de menor a mayor y buscar un tiempo específico.

Objetivo

Desarrollar un programa en C que permita:

1. Registrar los tiempos de un grupo de deportistas.
2. Mostrar los tiempos registrados.
3. Ordenar los tiempos de menor a mayor.
4. Buscar un tiempo determinado dentro del vector.
5. Informar la posición en la que se encuentra el tiempo buscado o indicar que no existe.

Requisitos del programa

El programa debe utilizar un vector de tipo `float` o `double`.

El usuario debe indicar inicialmente cuántos deportistas participarán.

Ejemplo:

```
Cantidad de deportistas: 6
```

Posteriormente se deben ingresar los tiempos:

```
Tiempo del deportista 1: 11.25
Tiempo del deportista 2: 10.87
Tiempo del deportista 3: 12.14
Tiempo del deportista 4: 10.52
Tiempo del deportista 5: 11.03
Tiempo del deportista 6: 10.95
```

Menú del programa

El programa debe presentar un menú similar al siguiente:

```
========== PRUEBA DE VELOCIDAD ==========

1. Registrar tiempos
2. Mostrar tiempos
3. Ordenar tiempos
4. Buscar un tiempo
5. Salir

Seleccione una opción:
```

El menú debe repetirse hasta que el usuario seleccione la opción **5**.

Funcionamiento de cada opción

Opción 1. Registrar tiempos

Solicitar la cantidad de deportistas y almacenar sus tiempos en un vector.

Ejemplo:

```
Ingrese cantidad de deportistas: 5

Tiempo 1: 11.24
Tiempo 2: 10.85
Tiempo 3: 12.10
Tiempo 4: 10.64
Tiempo 5: 11.03
```

Opción 2. Mostrar tiempos

Mostrar los tiempos registrados en el orden en que fueron ingresados.

Ejemplo:

```
Tiempos registrados en el caso que no estén ordenados aun:

11.24
10.85
12.10
10.64
11.03
```

Opción 3. Ordenar tiempos

Ordenar el vector **de menor a mayor**, porque en una prueba de velocidad el menor tiempo representa el mejor desempeño.

Resultado:

```
10.64
10.85
11.03
11.24
12.10
```

Para esta opción el estudiante debe implementar **uno de los siguientes algoritmos**:

- Bubble Sort
- Selection Sort
- Insertion Sort

El algoritmo utilizado debe estar implementado mediante una función. Debe justificar porque la escogio y por que es la mejor opción o es indiferente cualquiera de los métodos de ordenamiento

Ejemplo de estructura:

```c
void ordenar(float tiempos[], int n)
{
    // algoritmo seleccionado
}
```

No se permite utilizar funciones de ordenamiento predefinidas.

Opción 4. Buscar un tiempo

El usuario debe ingresar un tiempo que desea consultar.

Ejemplo:

```
Ingrese el tiempo que desea buscar: 10.85
```

Si el tiempo existe:

```
Tiempo encontrado en la posicion 1.
```

Si no existe:

```
Tiempo no encontrado.
```

Para esta opción se debe utilizar **búsqueda binaria**.

Condición

La búsqueda binaria debe realizarse **después de ordenar el vector**.

El programa debería **verifica**r que los tiempos hayan sido ordenados antes de realizar la búsqueda.

Ejemplo de ejecución

```
========== PRUEBA DE VELOCIDAD ==========

1. Registrar tiempos
2. Mostrar tiempos
3. Ordenar tiempos
4. Buscar un tiempo
5. Salir

Seleccione una opción: 1

Cantidad de deportistas: 6

Tiempo 1: 11.25
Tiempo 2: 10.87
Tiempo 3: 12.14
Tiempo 4: 10.52
Tiempo 5: 11.03
Tiempo 6: 10.95
```

El usuario selecciona: 3

Resultado:

```
Tiempos ordenados:

10.52
10.87
10.95
11.03
11.25
12.14
```

Posteriormente:

```
Seleccione una opción: 4

Ingrese el tiempo a buscar: 11.03

Tiempo encontrado en la posición 4.
```

Funciones mínimas sugeridas

El programa debe organizarse mediante funciones.

Se sugiere utilizar:

```
void registrar(float tiempos[], int n);

void mostrar(float tiempos[], int n);

void ordenar(float tiempos[], int n);

int buscar(float tiempos[], int n, float buscado);
```

El estudiante debe decidir cómo implementar internamente cada función.

Condiciones del taller

El programa debe cumplir las siguientes condiciones:

- Utilizar un vector.
- Utilizar un menú.
- Utilizar funciones.
- Utilizar ciclos.
- Utilizar un algoritmo de ordenamiento estudiado en clase.

- Utilizar búsqueda binaria.
- No utilizar funciones de ordenamiento de bibliotecas.
- La búsqueda binaria debe ejecutarse sobre los tiempos ordenados.
- Mostrar mensajes claros al usuario.

Preguntas de análisis

Antes de programar, responda:

1.¿Por qué los tiempos deben ordenarse de menor a mayor y no de mayor a menor?

2. ¿Por qué la búsqueda binaria no debe realizarse sobre el vector original si los tiempos están desordenados?

3. ¿Qué algoritmo de ordenamiento utilizará y por qué?

4.¿Qué debería ocurrir si el usuario intenta buscar un tiempo antes de ordenar el vector?

5.¿Qué debe devolver la función de búsqueda cuando el tiempo no existe?

6. Si dos deportistas tienen el mismo tiempo, ¿debe el programa permitirlo? Explique.

Rubrica trabajo:

Pregunta de análisis: (20)

```
1. Registrar tiempos (10)
2. Mostrar tiempos (10)
3. Ordenar tiempos (30)
4. Buscar un tiempo (30)
```

Rubrica General:

Trabajo:40%

video sustentación con cámara y todos las miembros de grupo interviniendo:60%.

Nota aclaratoria: tiempo de video: 15 minutos. penalización de 20 puntos si se pasa del tiempo o no entrega el documento en formato pdf. El documento debe entregarse en plataforma, con el código separado con extensión .c; si por motivos no se logra entregar la solución por correo será penalizada de 20 puntos por cada 5 minutos retrasados sobre el total de la nota. Se deben ejecutar los códigos y explicar en el video.

cantidad de integrantes: maximo=4 personas