Tecnologías de la Información

Universidad de Huelva

GRADO EN INGENIERÍA INFORMÁTICA

ESTRUCTURAS DE DATOS I

Práctica 1
Ficheros y Tablas Dinámicas

Desde hace unos años se celebra la Maratón ciudad de Huelva, con una participación cada vez mayor. La organización de esta competición nos ha pedido la elaboración de un programa que les permita realizar las operaciones de inscripción y obtención de la clasificación final de manera sencilla.

Se realizan dos fases de inscripción, una para atletas profesionales a nivel mundial y otra segunda fase a nivel local para todo aquel corredor que quiera participar.

En la primera fase se genera un fichero binario con los datos de los profesionales inscritos y que servirá de base para continuar con la inscripción en la segunda fase. Este fichero tendrá la siguiente estructura:

Nº país 1	Atleta 1	Atleta 2	…	Atleta n	…	Nº país M	Atleta 1	…	Atleta z

Donde **Nº país 1** será el número de atletas inscritos de un país, y **Atleta 1 … Atleta n** los datos de los atletas inscritos de ese país. A continuación estarán los del segundo país y así hasta el último país del que haya algún atleta.

Para cada atleta se almacenará la siguiente información:

a) *Número de dorsal* asignado
b) *País*
c) *Nombre*
d) *Apellidos*
e) *Marca (*conseguida por el atleta al final de la Maratón)
f) *Posición (obtenida según esa marca)*

Los dos últimos datos solo se rellenarán cuando se haya celebrado la maratón.

Al principio de programa habrá que generar un nuevo fichero binario con los datos recibidos en la primera fase de inscripción y que nos permita inscribir a los corredores locales que quieran participar. La estructura de este nuevo fichero será:

Nº Atletas	Atleta 1	Atleta 2	…	Atleta n

Donde Nº Atletas será el número de atletas inscritos hasta ese momento, siendo por tanto un dato de tipo entero. A continuación van los datos de los atletas, cada uno con la siguiente estructura:

```
struct Atleta {
      int dorsal;
      cadena pais;
      cadena nombre;
      cadena apellidos;
      int marca;
      int posicion;
};
```

Clase *Maratón*

La clase *Maratón* será usada para gestionar este segundo fichero, y es la siguiente:

```
class Maraton {
        fstream fich; //fichero primera fase
        fstream fichero; //fichero segunda fase
        int numAtletas;

public:
        Maraton(char FicheroOrigen[],char FicheroDestino[]);
         ~Maraton();
         int getNumAtletas() { return numAtletas; };
          void consultar(cadena pais);
           void insertar(Atleta s);
           void modificar(int dor);
           void eliminar(int dor);
           void mostrarClasificacion();
};
```

El constructor del objeto Maraton será el método en el que se debe abrir el fichero origen recibido de la primera fase de inscripción, según la cadena pasada como parámetro (con el nombre del fichero físico). Si tal fichero no existiera se procedería a crear el fichero destino vacío (asignando y guardando el valor de *0* para el número de atletas). Si el fichero origen existe se crea el fichero destino (con el nombre del fichero físico pasado), con la estructura vista anteriormente, a partir de la información del fichero origen. Una vez generado el fichero destino todas las operaciones actuarán sobre este último fichero.

El método ***consultar*** se encarga de mostrar por pantalla los datos de todos las inscripciones de atletas de un determinado país (la cadena pasada como parámetro). Si dicha cadena solo tuviera el carácter * se mostraría la información de todos los atletas del fichero.

El método ***insertar*** realiza la inserción de los datos de un nuevo atleta, teniendo en cuenta que los atletas deben continuar en el fichero agrupados por países. Habrá que controlar que no se insertan atletas con el mismo dorsal de los ya inscritos.

El método ***modificar*** se encarga de actualizar los datos de un atleta ya inscrito, aquel cuyo dorsal se pasa por parámetro y cuyos datos se habrán pedido previamente. Si el dorsal del atleta pasado no estuviera inscrito ya en la maratón, se mostraría un mensaje indicándolo. El único campo que no se podrá modificar será el país de origen del atleta ya inscrito.

El método ***eliminar*** realiza la eliminación de los datos del atleta cuyo dorsal se pasa por parámetro. Si el dorsal del atleta pasado no estuviera inscrito ya en la maratón, se mostraría un mensaje al respecto. Para eliminar una inscripción de un atleta del fichero, se desplazan una posición a la izquierda todas las inscripciones a continuación de la eliminada (para no dejar huecos).

El método ***mostrarClasificacion*** se encarga de realizar una simulación de la celebración de la maratón con los atletas que se han inscrito en las dos fases de inscripción. Su detalle se explica más adelante. Este método una vez simulada la maratón, mostrará por pantalla la clasificación final con los datos de los atletas, junto con la marca y posición obtenida.

Clase *Clasificacion*

Es usada para la simulación de la maratón, como se describirá más adelante. Su definición es:

```
class Clasificacion {
    Corredor  *elementos; //elementos de la tabla
    int corredores;
    int tamano;
 public:
    Clasificacion();
    ~Clasificacion();
    void anadircorredor(Corredor a);
    void eliminar(int i);
    Corredor consultar(int i);
    bool vacio();
    int numcorredores();
};
```

La clasificación se realizará almacenando en una tabla dinámica (elementos) la marca y el dorsal de cada atleta además del índice de ese atleta en el fichero de inscripciones. Con la definición de la siguiente estructura:

```
struct Corredor {
    int indice;
    int dorsal;
    int marca;
};
```

El atributo tamano indica el tamaño de la tabla. Se puede dar la circunstancia de que la tabla esté dimensionada para 12 corredores y actualmente tenga ocupados 9 (más detalle en anadircorredor).

El método **anadircorredor** añade la estructura Corredor pasada como parámetro a la tabla de elementos del objeto Clasificación. Si dicha tabla estuviera llena habrá que redimensionar la tabla a un tamaño igual al anterior + SALTO (siendo **SALTO** una constante definida en el programa, con valor de 4, con el propósito de no tener que redimensionar la tabla con cada inserción, sino cada "SALTO" inserciones).

El método **eliminar**, eliminará de la tabla dinámica el Corredor que ocupe la posición i, pasada como parámetro, en la tabla.

El método **consultar** permite obtener el elemento Corredor que se encuentre en la tabla dinámica en la posición pasada.

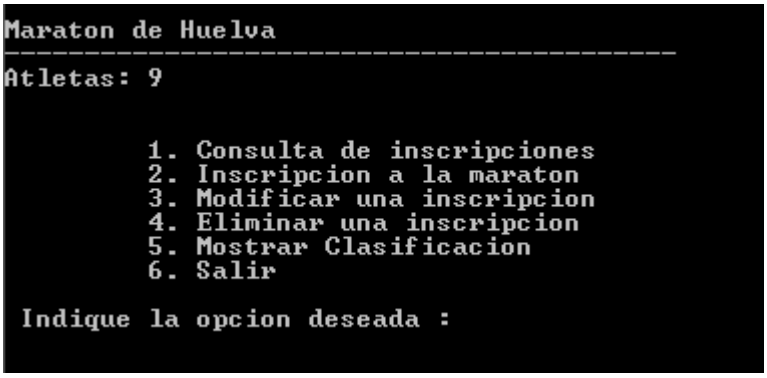
El método **vacio** devuelve verdadero si la tabla dinámica elementos está vacía o falso en caso contrario.

El método **numcorredores** devuelve el número de corredores en la tabla elementos.

Programa principal

El programa a desarrollar deberá comenzar creando un objeto *Maraton* a partir del fichero físico que se proporciona, "huelvapro.dat", y generando el fichero físico "huelva.dat" a partir de la información del anterior.

Luego mostrará el siguiente menú:



Las opciones 1 a 5 del menú se corresponden con las llamadas a los métodos de la clase *Maraton*.

Mostrar Clasificación

En el método ***mostrarClasificacion*** se va a simular que la maratón ya se ha celebrado y que los atletas han obtenido sus marcas al final de la carrera. Esas marcas se generarán aleatoriamente haciendo uso de la función marcas que se le proporcionará.

Una vez obtenidas las marcas de todos los participantes se creará el objeto Clasificación con el dorsal y la marca de cada corredor y el índice que ocupa su información en el fichero.

Cuando el objeto Clasificación ya esté creado, se ordenará la tabla dinámica de elementos por el método de ordenación Burbuja, cuyo código también se le proporcionará de manera genérica, pero tendrá que adaptarlo a los datos que hay que ordenar.

Y a continuación se mostrará por pantalla la clasificación de la maratón, todos los datos de los atletas participantes en orden de menor a mayor marca obtenida.

La marca es un valor de tipo entero, número de segundos transcurridos desde el comienzo de la carreta hasta la entrada en meta para cada uno de los atletas, pero para mostrar la marca por pantalla se implementará una función genérica que permita mostrar las marcas en formato: **horas: minutos: segundos**

Ficheros proporcionados

Se proporcionan los ficheros *huelvapro.dat* y *base.cpp*.

El fichero base.cpp se proporciona simplemente para recoger el código de la funciones marcas y ordenacionBurbuja, y de otras constantes y tipos que se reflejan en el enunciado. Deberá hacer uso de dicho código y, en su caso, moverlo al módulo que estime más conveniente.

El fichero *huelvapro.dat* contiene 9 atletas de 3 países distintos de ejemplo, con los siguientes datos:

```
4
Dorsal: 23
Pais: España
Nombre: Carlos
Apellidos: Ros Miñan

Dorsal: 34
Pais: España
Nombre: Ana
Apellidos: Martin Leal

Dorsal: 98
Pais: España
Nombre: Juan
Apellidos: Gil Orta

Dorsal: 65
Pais: España
Nombre: Sonia
Apellidos: Rios Pino

3
Dorsal: 12
Pais: Francia
Nombre: Jean
Apellidos: Reneau

Dorsal: 45
Pais: Francia
Nombre: Sophie
Apellidos: Moreau

Dorsal: 87
Pais: Francia
Nombre: Piere
Apellidos: Leblanc

2
Dorsal: 56
Pais: Italia
Nombre: Bruno
Apellidos: Basile

Dorsal: 28
Pais: Italia
Nombre: Paola
Apellidos: Milani
```

Los campos marca y posición tienen inicialmente un valor 0.