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 en la provincia de Huelva la prueba de mountain bike "Huelva extrema", con un recorrido aproximado de 180km. 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 ciclistas profesionales a nivel mundial y otra segunda fase a nivel local para todo aquel ciclista 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	Cicl. 1	Cicl. 2	…	Cicl. n	…	Nº país M	Cicl. 1	…	Cicl. z

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

Para cada ciclista se almacenará la siguiente información:

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

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

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

Nº Ciclistas	Cicl. 1	Cicl. 2	…	Cicl. n

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

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

Clase *Prueba*

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

```
class Prueba {
        fstream fich; //fichero primera fase
        fstream fichero; //fichero segunda fase
        int numCiclistas;

public:
         Prueba(char FicheroOrigen[],char FicheroDestino[]);
          ~Prueba();
         int getNumCiclistas();
         void mostrar(cadena pais);
         Ciclista consultar(int posicion);
         int buscar(int dorsal);
         void insertar(Ciclista c);
         void modificar(Ciclista c, int posicion);
         void eliminar(int posicion);
         void Clasificacion();
};
```

El constructor del objeto Prueba 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 Ciclistas). 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 **getNumCiclistas** devuelve el número de ciclistas en la prueba.

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

El método ***consultar*** devuelve el ciclista cuya posición se pasa por parámetro.

El método ***buscar*** devuelve la posición en el fichero del ciclista cuyo dorsal se pasa como parámetro, si se encuentra en él, y en caso contrario devuelve el valor -1, para indicar que no existe ningún ciclista con ese dorsal en el fichero.

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

El método ***modificar*** se encarga de actualizar los datos de un ciclista ya inscrito. Se pasarán los nuevos datos del ciclista y la posición donde se encuentra. Si el dorsal del ciclista pasado

no estuviera inscrito ya en la prueba, se mostraría un mensaje indicándolo. Nota: No se admite en la modificación cambiar el nombre del país.

El método ***eliminar*** realiza la eliminación de los datos del ciclista cuya posición se pasa por parámetro. Si la posición no existe, se mostraría un mensaje de error. Para eliminar una inscripción de un ciclista del fichero, se desplazan una posición a la izquierda todas las inscripciones a continuación de la eliminada (para no dejar huecos).

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

Clase *Clasificacion*

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

```
class Clasificacion {
    Participante  *elementos; //elementos de la tabla
    int participantes;
    int tamano;
 public:
    Clasificacion();
    ~Clasificacion();
    void anadirparticipante(Participante a);
    void eliminar(int i);
    Participante consultar(int i);
    bool vacio();
    int numpartipantes();
    void ordenar();
};
```

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

```
struct Participante {
    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 participantes y actualmente tenga ocupados 9 (más detalle en anadirparticipante).

El método **anadirparticipante** añade la estructura Participante 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 Participante que ocupe la posición i, pasada como parámetro, en la tabla.

El método **consultar** permite obtener el elemento Participante 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 **numparticipantes** devuelve el número de participantes en la tabla elementos.

El método **ordenar** ordena la tabla dinámica elementos por el algoritmo burbuja.

Programa principal

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

Luego mostrará el siguiente menú:

```
Huelva Extrema
----------------------------------------
Ciclistas: 9


        1. Consulta de inscripciones
        2. Inscripcion a la prueba
        3. Busqueda de una inscripcion
        4. Modificar datos de una inscripcion
        5. Eliminar una inscripcion
        6. Mostrar Clasificacion
        7. Salir

 Indique la opcion deseada :
```

Las distintas opciones del menú se detallan a continuación:

Opción 1.- Mostrará por pantalla los datos de las inscripciones de un determinado país o de todas las inscripciones a la prueba. Se solicitará el nombre de un país, si se encuentra se mostrarán los ciclistas inscritos de ese país y si no se mostrará un mensaje indicándolo. En el caso de querer consultar los datos de todas las inscripciones se pondrá * en el nombre del país.

Opción 2.- Pedirá los datos del nuevo ciclista a inscribir y realizará su inserción manteniendo los ciclistas agrupados por países. Habrá que controlar que no se insertan ciclistas con el mismo dorsal de los ya inscritos.

Opción 3.- Mostrará los datos de un ciclista, solicitando su dorsal. Si el ciclista no se encuentra inscrito, mostrará un mensaje indicándolo.

Opción 4.- Permite modificar los datos de una inscripción, a excepción del país. Pedirá los nuevos datos del ciclista y realizará la modificación en el fichero. Si el dorsal no se encuentra, mostrará un mensaje indicándolo.

Opción 5.- Permite eliminar una inscripción a la prueba. Pedirá el dorsal del ciclista y lo eliminará del fichero. Si el dorsal no se encuentra, mostrará un mensaje indicándolo.

Opción 6.- Una vez simulada la prueba, mostrará por pantalla la clasificación final con los datos de los ciclistas, junto con la marca y posición obtenida.

Mostrar Clasificación

En el método ***Clasificacion*** se va a simular que la prueba ya se ha celebrado y que los ciclistas 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 participante y el índice que ocupa su información en el fichero.

Cuando el objeto Clasificación ya esté creado, se ordenará la tabla dinámica de participantes 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 prueba, todos los datos de los ciclistas 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 carrera hasta la entrada en meta para cada uno de los ciclistas, 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 *huelvaextremapro.dat* y *base.cpp*.

El fichero base.cpp se proporciona simplemente para recoger el código de las 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 *huelvaextremapro.dat* contiene 9 ciclistas 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.

Para la implementación de la práctica será obligatorio el uso de Diseño Modular, además de Orientación a Objetos.

La práctica tendrá que estar terminada antes de la primera prueba práctica.