Tecnologías de la Información

Universidad de Huelva

GRADO EN INGENIERÍA INFORMÁTICA

ESTRUCTURAS DE DATOS I

Práctica 1
Ficheros y Tablas Dinámicas

Un campo de golf de la costa de Huelva 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 en los torneos de golf que organiza habitualmente.

Cuando se abre la inscripción de un determinado torneo se genera un fichero binario, uno por cada torneo, con los datos de los jugadores que se van inscribiendo. Este fichero tendrá la siguiente estructura:

Nº Golfistas	Golfista 1	Golfista 2	…	Golfista n

Al inicio de cada fichero se almacena el número de golfistas que se han inscrito hasta ese momento en ese determinado torneo, siendo por tanto un dato de tipo entero. A continuación, se almacenan los datos de los *n* golfistas.

Para cada golfista se almacenará la siguiente información:

a) *Licencia federativa, imprescindible para poder jugar al golf*
b) *Hándicap del jugador*
c) *Nombre*
d) *Apellidos*
e) *Número de golpes dados al final del recorrido*
f) *Puntuación al final del torneo, obtenida según el número de golpes*

Los dos últimos datos solo se rellenarán cuando se haya celebrado el torneo.

El conteo en golf es muy sencillo, se cuentan los golpes que cada jugador ha necesitado para embocar la bola en cada uno de los 18 hoyos del recorrido. El número total de golpes necesarios en un hoyo determina la puntuación total en ese hoyo. Al final del recorrido se suman los golpes en todos los hoyos y se calcula la puntuación final del torneo.

Al mirar una tabla de clasificación de golf, puede que los resultados no parezcan muy claros. Junto al nombre de cada golfista, puede haber un número positivo o negativo. La puntuación puede ser +3, -4, -1 o cualquier otro número. También puede aparecer PAR.

En cada campo, hay distintas combinaciones de hoyos de par 3, 4 y 5, en función de la distancia entre la salida del hoyo, el tee, y el hoyo en sí, que se encuentra en el llamado green. Por ejemplo, puede que el 4.º hoyo en un campo sea un par 4 y, en otro, tenga menor distancia y sea un par 3.

Además, todos los campos tienen una puntuación de par total, el par del campo, que equivale a la suma de la puntuación de par de cada hoyo. Por eso hay cifras positivas y negativas en la tabla de clasificación. La puntuación -1 indica que un jugador está 1 golpe por debajo (o mejor) del par total. Si su puntuación es +4, está 4 golpes por encima (o peor) del par del campo. Y, si es 0, su puntuación coincide con el par.

El hándicap es la diferencia entre la puntuación del golfista, número de golpes totales, y el par del campo. Por ejemplo, si un jugador tiene un hándicap 28 en un campo de par 72 es porque necesita dar 100 golpes para completar el recorrido (100 sería la suma de 72 + 28).

En este sentido el golf es muy justo porque cada jugador juega con su hándicap, que determina su nivel de 0 a 54, que será mejor cuanto menor sea. Así a un jugador con hándicap 36 se le permiten dar 108 golpes en un campo de par 72, mientras que a un jugador de hándicap 0, tendría que dar 72.

Para no tener que calcular para cada jugador la puntuación de distinta forma y facilitar la obtención de la lista de clasificados, vamos a calcular siempre la puntuación como el número de golpes totales del jugador menos el par del campo, que en este caso es 72. Así tendremos puntuaciones negativas, positivas y PAR. Y el ganador será el que tenga menor puntuación.

Clase *Torneo*

La clase *Torneo* será usada para gestionar un fichero con los jugadores inscritos en cada torneo, y es la siguiente:

```
struct Golfista {
      cadena licencia;
      float handicap;
      cadena nombre;
      cadena apellidos;
      int golpes;
      int resultado;
};
```

```
class Torneo
{
    fstream fichero;
    int numGolfistas;
    cadena nomFichero;
    cadena nomTorneo;
public:
    ~Torneo();
    Torneo();
    int getNumGolfistas();
    void putNumGolfistas(int n);
    void getNomTorneo(cadena nombre);
    void getNomFichero(cadena nombre);
    void putNomTorneo(cadena nombre);
    void putNomFichero(cadena nombre);
    void crearFichero(char nombreFichero[]);
    void mostrar(float hdcp);
    Golfista consultar(int posicion);
    int buscar(cadena licencia);
    void insertar(Golfista g);
    void modificar(Golfista c, int posicion);
    void eliminar(int posicion);
    void Clasificar();
};
```

El método ***crearFichero*** del objeto Torneo será el método en el que se debe crear el fichero para comenzar a inscribir jugadores o abrir el fichero ya creado para seguir inscribiendo jugadores o gestionar las inscripciones (mostrarlas, modificarlas, eliminarles), así como simular la celebración del torneo y mostrar los resultados del torneo. Si el fichero nombrefichero no existe se procede a crear el fichero vacío (asignando y guardando el valor de *0* para el número de Golfistas). Si el fichero existe se abre para poder gestionarlo.

El método ***getNumGolfistas*** devuelve el número de golfistas del torneo.

El método ***putNumGolfistas*** permite establecer el número de golfistas del torneo.

El método ***getNomTorneo*** devuelve el nombre del torneo.

El método ***putNomTorneo*** permite establecer el nombre del torneo.

El método ***getNomFichero*** devuelve el nombre del fichero del torneo.

El método ***putNomFichero*** permite establecer el nombre del fichero del torneo.

El método ***mostrar*** se encarga de mostrar por pantalla los datos de todas las inscripciones de un determinado torneo que tengan el mismo hándicap pasado por parámetro. Si el valor pasado es -1 se mostrará por pantalla la información de todos los golfistas del fichero.

El método ***consultar*** devuelve el golfista cuya posición se pasa por parámetro. La posición del primer golfista en el fichero es la 1.

El método ***buscar*** devuelve la posición en el fichero del golfista cuya licencia se pasa como parámetro, si se encuentra, y en caso contrario devuelve el valor -1, para indicar que no existe ningún golfista con esa licencia en el fichero.

El método ***insertar*** realiza la inserción de los datos de un nuevo golfista, teniendo en cuenta que los golfistas deben continuar en el fichero ordenados por hándicap, de menor a mayor. Habrá que controlar que no se insertan golfistas con la misma licencia de los ya inscritos en un mismo torneo.

El método ***modificar*** se encarga de actualizar los datos de un golfista ya inscrito. Se pasarán los nuevos datos del golfista y la posición donde se encuentra. Si el golfista pasado no estuviera inscrito en el torneo, se mostraría un mensaje indicándolo. **Nota: No se admite en la modificación cambiar el hándicap del golfista.**

El método ***eliminar*** realiza la eliminación de los datos del golfista cuya posición se pasa por parámetro. Si la posición no existe, se mostraría un mensaje de error. Para eliminar una inscripción de un golfista 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 del torneo con los golfistas que se han inscrito. Su detalle se explica más adelante. Este método simulará la celebración del torneo y mostrará por pantalla la clasificación final con los datos de los golfistas, junto con el número de golpes y los resultados obtenidos.

Clase *Clasificacion*

Es usada para la simulación de la celebración de los torneos, como se describirá más adelante.

```
class Clasificacion {
    Jugador *elementos;
    int Jugadores;
    int tamano;
 public:
    Clasificacion();
    void anadirjugador(Jugador j);
    void eliminar(int i);
    Jugador consultar(int n);
    bool vacio();
    int numjugadores();
    void ordenar();
    ~Clasificacion();
}
```

La clasificación se realizará almacenando en una tabla dinámica (elementos) el resultado de cada golfista además de su índice en el fichero de inscripciones. Con la definición de la siguiente estructura:

```
struct Jugador {
    int indice;
    int resultado;
};
```

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

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

El método ***consultar*** permite obtener el Jugador 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 ***numjugadores*** devuelve el número de jugadores 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 una tabla de N objetos Torneo y, mostrando el siguiente menú con el número de torneos a cero, si no hay torneos abiertos, o igual al número de torneos abiertos, si los hay. Si no hay torneos abiertos se creará un fichero llamado TORNEOS.dat. Si hay torneos abiertos leeremos los torneos del fichero y volcaremos en la tabla de objetos Torneo la información de los torneos abiertos del fichero.

La estructura del fichero TORNEOS.dat será:

Torneo 1	Torneo 2	Torneo 3	…	Torneo N

Menú 1. Club de Golf

```
CLUB DE GOLF
----------------------------------------
Torneos: 0

        1. Listado Torneos Abiertos
        2. Alta Torneo
        3. Elegir Torneo
        4. Salir

 Indique la opcion deseada :
```

Las distintas opciones de este menú 1 CLUB DE GOLF se detallan a continuación:

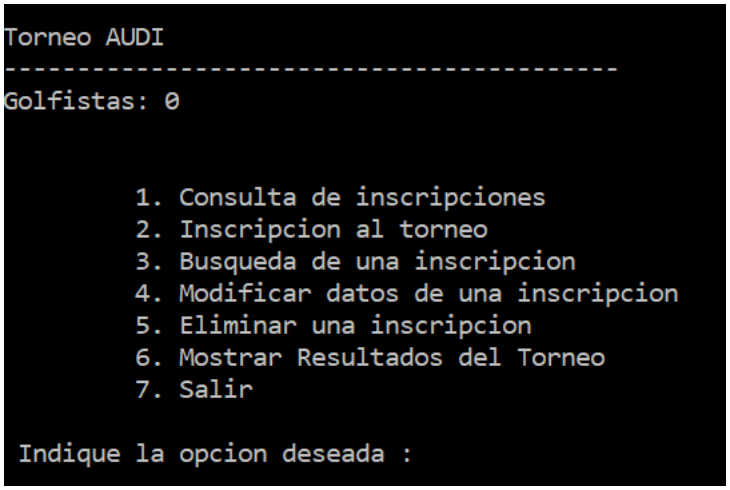
Opción 1.- Mostrará por pantalla los datos de los torneos que tienen abierta la inscripción a los golfistas. Si no hay ningún torneo en el que poder inscribirse, se mostrará un mensaje indicándolo.

Opción 2.- Pedirá los datos del nuevo torneo que abre la inscripción: nombre del torneo, nombre del fichero que almacenará la información del torneo y pondrá el número de golfistas inscritos en el torneo a 0. A continuación creará el fichero con dicho nombre con la estructura vista anteriormente, y actualizará el fichero TORNEOS.dat con la información del nuevo torneo.

Opción 3.- Permite elegir el torneo con el que se quiere operar: consultar, inscribir, buscar, modificar o eliminar golfistas. Al elegir el torneo se mostrará el menú 2 TORNEO, cuyas opciones se explicarán más adelante.

Opción 4.- Será la única opción que permite abandonar la aplicación.

Menú 2. Torneo



Las distintas opciones de este menú 2 TORNEO se detallan a continuación:

Opción 1.- Mostrará por pantalla los datos de las inscripciones de los golfistas con un determinado hándicap o de todas las inscripciones al torneo. Se solicitará un hándicap y se mostrarán los golfistas inscritos con ese hándicap y si no hay ninguno, se mostrará un mensaje indicándolo. En el caso de querer consultar todas las inscripciones al torneo se pondrá -1 en el valor del hándicap.

Opción 2.- Pedirá los datos del nuevo golfista a inscribir: licencia, hándicap, nombre y apellidos, y realizará su inserción manteniendo los golfistas ordenados por hándicap, de menor a mayor. Habrá que controlar que no se inscriben golfistas con el mismo número de licencia de los ya inscritos.

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

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

Opción 5.- Permite eliminar una inscripción al torneo. Pedirá la licencia del golfista y lo eliminará del fichero. Si la licencia no se encuentra, mostrará un mensaje indicándolo.

Opción 6.- Una vez simulado el torneo, mostrará por pantalla la clasificación final con los datos de los golfistas, junto con el número de golpes y el resultado obtenido.

Mostrar Clasificación

El método ***Clasificar*** va a simular que el torneo se ha celebrado y que los golfistas tienen su resultado final en función de un número de golpes aleatorio.

Una vez generados aleatoriamente los golpes de todos los participantes se creará un objeto Clasificación en cuyo atributo **elementos** (tabla dinámica) se almacenarán estructuras de tipo

Jugador, que contendrán únicamente de cada golfista del torneo: el resultado (el número de golpes que ha necesitado cada jugador en hacer el recorrido) y el índice que ocupa el golfista en el fichero.

```
struct Jugador {
    int indice;
    int resultado;
};
```

Cuando el objeto Clasificación ya esté creado, se ordenará la tabla dinámica de jugadores de menor a mayor resultado por el método de ordenación Burbuja. El código de dicho método se proporcionará de manera genérica y tendrá que adaptarlo a los datos que hay que ordenar.

A continuación, se mostrará por pantalla la clasificación del torneo con toda la información de los golfistas participantes en orden de menor a mayor resultado obtenido (negativo, positivo o PAR)

El fichero *base.cpp* se proporciona simplemente para recoger el código de la función ***ordenacionBurbuja***, y las definiciones de clases, 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.

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.