Metodología de la Programación (2020/21)
1º GII / GII-M / GII-ADE
Conv. Ordinaria - 25 de Junio de 2021

Normas para la realización del examen: **Duración: 3 horas**

- **El único material permitido durante la realización del examen es un lápiz o bolígrafo (azul o negro).**
- **Debe disponer de un documento oficial que acredite su identidad a disposición del profesor.**
- **No olvide escribir su nombre completo y grupo en todos y cada uno de los folios que entregue.**

En este ejercicio, vamos a considerar la gestión de la información de un videojuego de estrategia basado en batallas de defensa de torres y en el que los jugadores se organizan en clanes. Cada jugador libra batallas individuales contra rivales que se le asignan de forma aleatoria. Las batallas ganadas otorgan trofeos que le permiten subir de nivel y obtener recompensas para mejorar sus mazos de guerra, de la misma forma que las batallas perdidas le harán perder trofeos. Vamos a concentrarnos en una versión muy simplificada de la gestión de este tipo de juegos: la relativa a la organización de jugadores en clanes y la de los historiales de batalla de los miembros del clan.

Se propone la siguiente estructura básica para las clases que permiten modelizar un clan de jugadores:

```
class Clan{
private:
  int id;
  string nombre;
  Jugador * jugadores;
  int numJugadores;
};
```

```
class Jugador{
private:
  int id;
  string nombre;
  int trofeos;
  Historico historial;
};
```

```
class Historico{.
private:
  int num_batallas;
  Batalla * batallas;
};
```

```
class Batalla{
private:
  int rival;
  int trofeos;
};
```

Suponga que dispone de la implementación en la clase **Batalla** de los siguientes métodos públicos:

```
Batalla();
Batalla(int el_rival, int los_trofeos);
int getRival() const;
int getTrofeos() const;
void setRival(int el_rival);
void setTrofeos(int los_trofeos);
```

y de los operadores de inserción y extracción de flujos (<< y >>).

Suponga que dispone de la implementación en la clase **Historico** de los siguientes métodos públicos:

```
Historico();
Historico(const Historico & otro);
~Historico();
int numBatallas() const;
Historico & operator=(const Historico & otro);
Batalla & operator[](int indice);
const Batalla & operator[](int indice) const;
```

y de los operadores de inserción y extracción de flujos (<< y >>).

Suponga que dispone de la implementación en la clase **Jugador** de las versiones constante y no constante del operador [] y de los operadores de inserción y extracción de flujos (<< y >>).

Suponga que dispone de la implementación en la clase **Clan** del siguiente métodos público:

```
int getNumeroJugadores() const;
```

◁ Ejercicio 1 ▷ Métodos básicos de las clases Clan y Jugador [1.25 puntos]

Defina los siguientes métodos para las clases que se indican:

1. *(0.25 puntos)* **Constructor** de la clase **Jugador** que recibe como argumentos un identificador, un nombre, un número de trofeos y un historial (un objeto de la clase **Historico**).
2. *(0.25 puntos)* **Destructor** de las clases Clan y Jugador: indique si es necesario y, en caso afirmativo, impleméntelo.
3. *(0.5 puntos)* **Constructor de copia** y **operador de asignación** de las clases Clan y Jugador: indique si son necesarios y, en caso afirmativo, impleméntelos.
4. *(0.25 puntos)* Clase Clan: **Constructor sin argumentos** (crea un objeto que representa un *clan vacío, sin jugadores*, con `id = 0` y `nombre = ""`. Clase Jugador: indique si es *necesario (o sólo recomendable)* implementar el **constructor sin argumentos**. En caso afirmativo, haga la implementación.

◁ Ejercicio 2 ▷ Sobrecarga de operadores [2 puntos]

1. *(1 punto)* Sobrecargue el operador += para la clase **Clan**. Debe desarrollar dos sobrecargas:
   (a) *(0.5 puntos)* El objetivo es incorporar un nuevo jugador al clan. El jugador se añade **al final** del array de jugadores. Tenga en cuenta que el clan puede estar vacío.
   (b) *(0.5 puntos)* El objetivo es realizar la anexión de todos los jugadores de *OTRO* clan. Los jugadores incorporados se añaden **al final** del array de jugadores. Entre las posibles implementaciones del operador, elija una que resulte eficiente. El identificador y el nombre del clan destino, al que se incorporan los jugadores, no cambian.
2. *(0.5 puntos)* Sobrecargue el operador += para la clase **Jugador**. El objetivo es incorporar al historial de un jugador una nueva batalla. La nueva batalla se incorpora **al principio** del historial, de forma que las batallas más recientes se encuentren al principio del vector de batallas.
3. *(0.5 puntos)* Sobrecargue el operador [ ] (versiones constante y no constante) en las clases **Clan** y **Jugador**. En la clase **Clan** debe devolver el i-ésimo jugador del clan. En la clase **Jugador**, devolverá la i-ésima batalla del historial del jugador. No es necesario comprobar si el índice del elemento a devolver es correcto.

◁ Ejercicio 3 ▷ Métodos y funciones para E/S [1 punto]

1. *(0.5 puntos)* Sobrecargue los operadores << y >> para la clase **Clan**. Deberá seguirse el formato de la siguiente figura.

```
1 Men's Clan
3
1001 A Bad Man
5032
3
1100 25
1101 28
1102 -27
1002 A Good Man
4800
3
1103 23
1105 32
1104 -31
1003 A Quiet Man
6023
4
1106 -28
1107 31
1108 29
1110 26
```

El formato, como se puede ver, contiene:

- El número que representa el identificador del clan, un espacio en blanco y el nombre del clan (y salto de línea).
- Un número entero que indica el número de jugadores del clan (y salto de línea).
- Para cada jugador del clan:
  - Un número entero indicando el identificador del jugador, un espacio en blanco y el nombre del jugador (y salto de línea).
  - Un número entero indicando el número de trofeos del jugador (y salto de línea).
  - Un número entero indicando el número de batallas del historial del jugador (y salto de línea).
  - Una línea por cada batalla conteniendo dos enteros separados por un espacio en blanco, correspondientes al identificador del rival y los trofeos obtenidos (y salto de línea).

2 *(0.5 puntos)* Implemente un constructor de la clase **Clan** con un argumento que indica el nombre de un fichero de texto que contiene la información completa de un clan. La primera línea del archivo contiene la cadena **"FICHEROCLAN"**. A continuación aparecería la información del clan, con el formato comentado anteriormente. Si la primera línea del fichero no contiene la cadena indicada, podemos deducir que no se trata de un fichero con el formato adecuado.

◁ Ejercicio 4 ▷ Métodos de cálculo [1.25 puntos]

1. *(0.5 puntos)* Implemente el método **eficiencia** de la clase Jugador, que calcula la media de trofeos obtenidos en las batallas del historial. Por convención, la eficiencia de un jugador sin batallas es 0.
2. *(0.75 puntos)* Implemente el método **eficiencia** de la clase Clan, que calcula la media de trofeos obtenidos por los jugadores del clan. Por convención, la eficiencia de un clan vacío (sin jugadores) es 0.

◁ Ejercicio 5 ▷ Aplicación: encontrar jugadores máster [0.5 puntos]

Escriba un programa *completo* que reciba como argumentos un valor real (valor umbral de eficiencia) y un número indeterminado de nombres de archivos de texto (ficheros con la descripción de un clan). Se considerará que un jugador es *máster* si su valor de eficiencia es mayor o igual que el umbral proporcionado. El programa mostrará, para cada uno de los jugadores *máster*, sus datos (eficiencia, identificador, nombre y número de trofeos). También mostrará en pantalla su historial de batallas, indicando para cada batalla el identificador del rival y los trofeos obtenidos.

Nota: puede convertir una cadena tipo C a un valor double mediante la función **atof**, que recibe como argumento una cadena tipo C (por ejemplo "20.5") y devuelve el valor real asociado.