Programación 2

Examen de teoría (junio 2016)

7 de junio de 2016

Instrucciones

- **Duración: 3 horas**
- El fichero del primer problema debe llamarse `juegodetronos.cc`. Para el segundo problema es necesario entregar cuatro ficheros, llamados `Fighter.cc, Fighter.h, Squadron.cc, Squadron.h`. Pon tu DNI y tu nombre en un comentario al principio de los ficheros fuente.
- La entrega se realizará como en las prácticas, a través del servidor del DLSI (`http://pracdlsi.dlsi.ua.es`), en el enlace `Programación 2`. Puedes realizar varias entregas, aunque sólo se corregirá la última.
- En la página web de la asignatura `http://www.dlsi.ua.es/asignaturas/p2` tienes disponibles algunos ficheros que te pueden servir de ayuda para resolver los problemas del examen, y el apartado `Reference` de la web `www.cplusplus.com`.

Problemas

1. **(5.5 puntos)**

   Basado en hechos reales. Ha empezado la sexta temporada de *Juego de Tronos* y, después de diez meses desde el último episodio, te das cuenta de que tienes serios problemas para recordar los nombres y las relaciones de parentesco, amistad y enemistad que se dan entre los distintos personajes que pueblan la serie.

   Por suerte para ti, un compañero de Programación 2 te ha proporcionado un fichero binario que contiene información sobre todos los personajes de la serie y sus relaciones. Cada registro del fichero contiene el nombre de dos personajes y el tipo de relación que existe entre ellos. Estas relaciones podrán ser de tres tipos: familiar (*relative*), enemigo (*enemy*) o amigo (*friend*). El formato de los registros del fichero es el siguiente:[1]

   - `nombre1`: vector de 40 caracteres
   - `relacion`: número entero cuyo valor puede ser `0` (si es familiar), `1` (si es enemigo) o `2` (si es amigo)
   - `nombre2`: vector de 40 caracteres

   Por ejemplo, un registro nos puede indicar que "Robb Stark" (`nombre1`) es enemigo (`relacion` vale `1`) de "Cersei Lannister" (`nombre2`). **Todas las relaciones son recíprocas**, es decir, que en este caso habría que considerar que Cersei Lannister también es enemiga de Robb Stark. Cada personaje puede aparecer en múltiples registros.

   Tu objetivo es realizar un programa que lea este fichero[2] y muestre por pantalla un resumen de la información leída para cada personaje. Un ejemplo de salida para Cersei Lannister sería el siguiente:

   ```
   Name: Cersei Lannister
   Relatives: Jaime Lannister, Tyrion Lannister, Joffrey Baratheon
   Enemies: Robb Stark
   Friends: Gregor Clegane, Harys Swyft
   ```

   Si para un personaje alguna relación no tiene valores (por ejemplo, Ramsay Bolton no tiene amigos) no se debe mostrar ese campo en la salida. El orden en el que se muestran los personajes no es relevante, pudiendo ser el mismo en el que han ido apareciendo en el fichero.

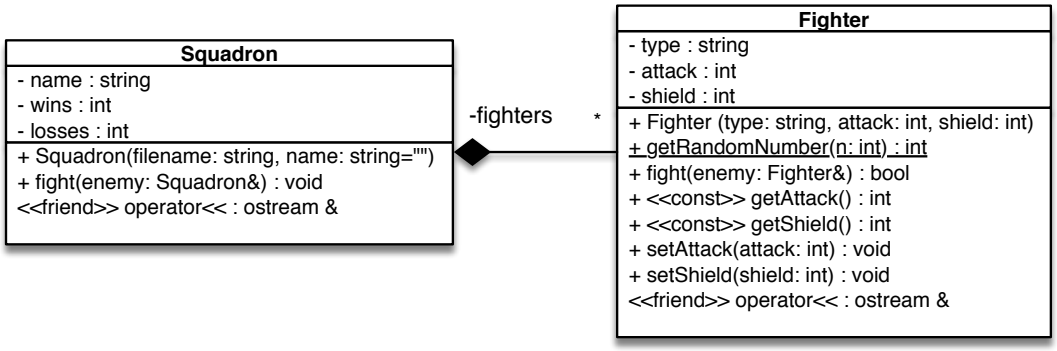
   El programa recibirá como único parámetro de entrada el nombre del fichero binario con la información de los personajes. Ejemplo de llamada:[3]

   ```
   ./juegodetronos personajes.dat
   ```

[1] Asumiremos que el formato del fichero de entrada es correcto. Tampoco hace falta comprobar si hay registros duplicados.
[2] Se debe comprobar que el fichero ha podido abrirse de manera correcta y mostrar error en caso contrario.
[3] Se deberá de comprobar que los argumentos sean correctos y mostrar un error por pantalla en caso contrario.

2. **(4.5 puntos)**

Queremos hacer un programa en el que puedan luchar dos escuadrones. Para ello tenemos el siguiente diagrama:



Un caza (`Fighter`) tiene un tipo, ataque y escudo. El constructor debe asignar a los atributos los valores que recibe, siempre y cuando el ataque y el escudo tengan valores entre 0 y 99. Si no es así, el constructor debe lanzar una excepción. El método `getRandomNumber` devolverá un número aleatorio entre 0 y $n-1$ con la instrucción `rand()%n` que requiere incluir la librería `cstdlib`. El caza actual atacará a un caza enemigo mediante el método `fight`. Para ello generaremos un número aleatorio n entre 0 y 99, y restaremos al escudo del enemigo el valor del ataque del caza actual multiplicado por n y dividido por 100. Si el escudo del caza enemigo queda con un valor igual o inferior a 0, `fight` debe devolver `true`, y si no `false`. El operador salida debe imprimir el caza en el formato que aparece al final, por ejemplo: `TIE-Fighter (a=50,s=4)`.

Un escuadrón (`Squadron`) tiene un nombre y almacena la cantidad de victorias y derrotas. El constructor debe abrir un fichero cuyo nombre recibe por parámetro, y que contiene una serie de líneas[4] en el formato `type attack shield`, por ejemplo "TIE-Fighter 10 4". Por cada línea de este fichero debemos construir un caza y añadirlo al conjunto de cazas del escuadrón, o mostrar el mensaje "Wrong fighter data" si los valores de ataque o escudo son incorrectos (en este caso el fichero se debe seguir leyendo). El método `fight` simula la lucha entre dos escuadrones. Para ello, debe extraer aleatoriamente, usando `getRandomNumber`, un caza de cada escuadrón y borrarlo[5], hacer que el caza del escuadrón actual ataque al caza enemigo, y si no lo destruye, que el caza enemigo ataque al caza actual. La lucha entre cazas acabará cuando alguno de los dos sea destruido, y el combate entre escuadrones terminará cuando alguno de los escuadrones esté vacío. Tras cada lucha entre dos cazas se actualizará el valor de combates ganados (`wins`) y perdidos (`losses`) en ambos escuadrones, y se guardará el caza superviviente al final de su escuadrón. El operador salida debe imprimir un escuadrón (nombre, `wins`, `losses` y sus cazas).

Dado el siguiente `main.cc` (que puedes compilar con `g++ Fighter.cc Squadron.cc main.cc -o ej2`):

```
#include "Squadron.h"
int main()
{
    Squadron a("first.txt","Imperial"), b("second.txt","Rebel");

    cout << "---" << endl << a << endl;

    a.fight(b);

    cout << "---" << endl << a << endl << b << endl;
}
```

...y los ficheros `first.txt` y `second.txt`, el programa debería imprimir:

```
Wrong fighter data
---
Imperial: Wins=0 Losses=0 TIE-Fighter (a=50,s=4) TIE-Bomber (a=6,s=60) TIE-Fighter (a=30,s=30)
---
Imperial: Wins=2 Losses=3
Rebel: Wins=3 Losses=2 Y-Wing (a=23,s=77) X-Wing (a=40,s=4)
```

[4] No hace falta comprobar el formato del fichero, se considerará que siempre es correcto, pero si no se puede abrir hay que mostrar un mensaje de error y no añadir ningún caza.

[5] Como en las prácticas 2 y 3.