Programación 2

Examen de teoría (julio 2012)

16 de julio de 2012

Instrucciones

- **Duración: 3 horas**
- El fichero del primer ejercicio debe llamarse `ej1.cc`. Para el segundo problema es necesario entregar cuatro ficheros, llamados `Paciente.cc, Paciente.h, Medicamento.cc, Medicamento.h`. Pon tu DNI y tu nombre en un comentario al principio de los ficheros fuente.
- La entrega se realizará del mismo modo que 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.

Problemas

1. **(5 puntos)**

   Tenemos un fichero de texto con datos sobre películas, que contiene en cada linea información sobre el título, director y año correspondientes. Un ejemplo de fichero sería el siguiente:

   ```
   title: La vida de Brian
   director: Terry Jones
   year: 1979
   title: Origen (Inception)
   director: Christopher Nolan
   year: 2010
   title: Metropolis
   director: Fritz Lang
   year: 1927
   ```

   Se trata de hacer un programa que lea un fichero que se le pase por parámetro (por ejemplo, `./ej1 pelis.txt`) e imprima el listado de películas, **ordenadas** descendentemente por año, en el siguiente formato:

   ```
   2010, Origen (Inception), Christopher Nolan
   1979, La vida de Brian, Terry Jones
   1927, Metropolis, Fritz Lang
   ```

   Se supone que habrá como máximo 100 películas en el fichero. En caso de que haya dos fechas iguales, el orden de ambas películas puede ser cualquiera.

   Para tener las películas ordenadas, se puede realizar una inserción ordenada (buscando en el vector la posición que corresponde al nuevo elemento, desplazando todos los elementos posteriores una posición hacia la derecha, y copiando el nuevo elemento en su posición correspondiente), o bien se pueden insertar todos los elementos y después aplicar un algoritmo de ordenación como el de la burbuja, por ejemplo.

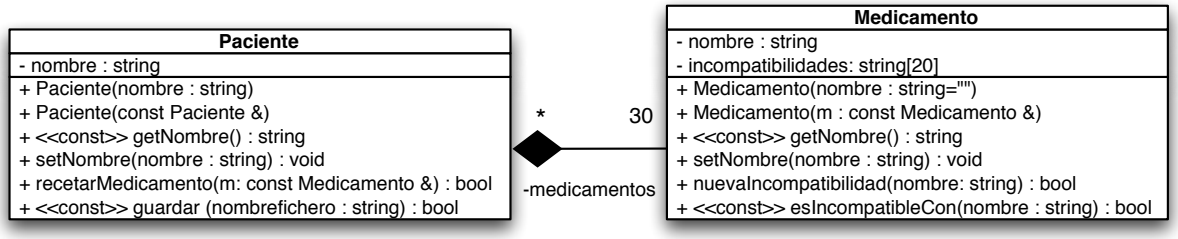
   Si se considera conveniente, puede usarse la función `int atoi(char c[])` de la librería `cstdlib` para transformar una cadena de caracteres en un número entero. Se supondrá que el formato del fichero será siempre correcto, y que contendrá 3 líneas por cada película. Hay que controlar los errores de argumentos del programa y de apertura de fichero, aunque no los de lectura/escritura.

   Nota: Este problema debe implementarse usando programación procedural (no orientada a objetos).

2. **(5 puntos)**

Queremos implementar[1] un programa que gestione las recetas de pacientes de un centro de salud. Para ello, debemos poder crear nuevos pacientes y recetarles medicamentos. El programa escribirá la información de un paciente (datos y recetas) en un fichero mediante el método `guardar`.

Hay ciertos medicamentos que son incompatibles entre sí. Los medicamentos incompatibles con uno dado se almacenarán en el vector `incompatibilidades`.



Por ejemplo, dado el siguiente fichero `main.cc`:

```
#include <iostream>
#include "Paciente.h"
#include "Medicamento.h"
using namespace std;

int main()
{
  Paciente p("Juan Lopez");

  Medicamento m("Diazepam");
  m.nuevaIncompatibilidad("Lanoxin");
  m.nuevaIncompatibilidad("Prozac");
  p.recetarMedicamento(m);

  Medicamento m2("Prozac");
  p.recetarMedicamento(m2);    // Incompatible, no se debe añadir (mostrar un error por pantalla)

  Medicamento m3("Salbutamol");
  p.recetarMedicamento(m3);

  p.guardar("prueba.txt");
}
```

El programa debería guardar en el fichero `prueba.txt`:

```
Paciente: Juan Lopez
1 Diazepam
2 Salbutamol
```

El método `nuevaIncompatibilidad` añade un nombre de medicamento al vector de incompatibilidades, y devuelve `true` si ha podido hacerlo (si hay menos de 20). El método `recetarMedicamento` añade un nuevo medicamento y devuelve `true` si ha podido (hay menos de 30 medicamentos y no hay incompatibilidades). El método `esIncompatibleCon` devuelve `true` si el nombre del medicamento que se le pasa está en la lista de incompatibilidades, y finalmente `guardar` debe devolver `true` si se ha podido abrir el fichero.

Ayuda: Puedes usar el siguiente `makefile` para compilar el programa:

```
main : main.o Paciente.o Medicamento.o
	g++ -o main main.o Paciente.o Medicamento.o
main.o: main.cc Paciente.h Medicamento.h
	g++ -c -g -Wall main.cc
Paciente.o: Paciente.cc Paciente.h Medicamento.h
	g++ -c -g -Wall Paciente.cc
Medicamento.o: Medicamento.cc Medicamento.h
	g++ -c -g -Wall Medicamento.cc
```

[1] La parte pública debe implementarse como aparece en el diagrama, aunque se pueden añadir nuevos métodos o atributos privados.