Programación 2

Examen de teoría (junio 2017)

15 de junio de 2017

Instrucciones

- **Duración: 3 horas**
- El fichero del primer problema debe llamarse `buscador.cc`. Para el segundo problema es necesario entregar cuatro ficheros, llamados `Movie.cc, Movie.h, List.cc, List.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)**

   En este ejercicio deberás hacer un programa llamado `buscador` que permita localizar documentos relevantes para una determinada palabra, de manera similar a lo que hace Google cuando realizamos una búsqueda y nos devuelve páginas web relacionadas.

   El sistema utilizará dos ficheros[1]. El primero será un fichero de texto llamado `index.txt`. Este fichero contendrá un índice invertido que almacena, para cada palabra[2], qué documentos son relevantes y cuántas veces ha aparecido la palabra en ellos (lo que se conoce como *frecuencia*). Un ejemplo de fichero sería:

   ```
   big|0:3|3:5|4:2
   cat|1:9|5:5|14:3
   dog|2:25|4:12
   ...
   ```

   En cada línea, la barra vertical ('`|`') sirve para separar los distintos campos. En primer lugar aparece la palabra. A continuación aparecerá una secuencia de uno o más pares de números separados por dos puntos ('`:`'). El primer número indica el identificador de documento en el que aparece la palabra y el segundo cuántas veces aparece la palabra en ese documento (su frecuencia). Esta secuencia de pares de números está ordenada para cada palabra por el identificador de documento. En el ejemplo anterior, `cat` aparece en el documento `1` (`9` veces), en el documento `5` (`5` veces) y en el documento `14` (`3` veces).

   La información de los documentos se encuentra almacenada en un fichero binario, llamado `documents.bin`, que está compuesto de registros con el siguiente formato:

   ```
   char title [50]; // El título del documento
   char url[50]; // La URL donde localizar el texto completo del documento
   int length; // El número de palabras que contiene el texto completo del documento
   ```

   El documento con identificador `0` estará en el primer registro del fichero, el documento con identificador `1` en el segundo registro y así sucesivamente.

   Al llamar al programa se le pasará como parámetro por línea de comando una única palabra[3]. Si la palabra no existe en el fichero `index.txt` se deberá mostrar un mensaje de error y terminar. Ejemplo de llamada con la palabra `cat`:

[1] Se debe comprobar que se han podido abrir correctamente y emitir un mensaje de error en caso contrario. El formato de ambos ficheros es siempre correcto por lo que no hace falta comprobarlo.

[2] Las palabras del fichero están ordenadas alfabéticamente.

[3] En caso contrario se debe emitir un mensaje de error y finalizar el programa.

```
$ buscador cat
```

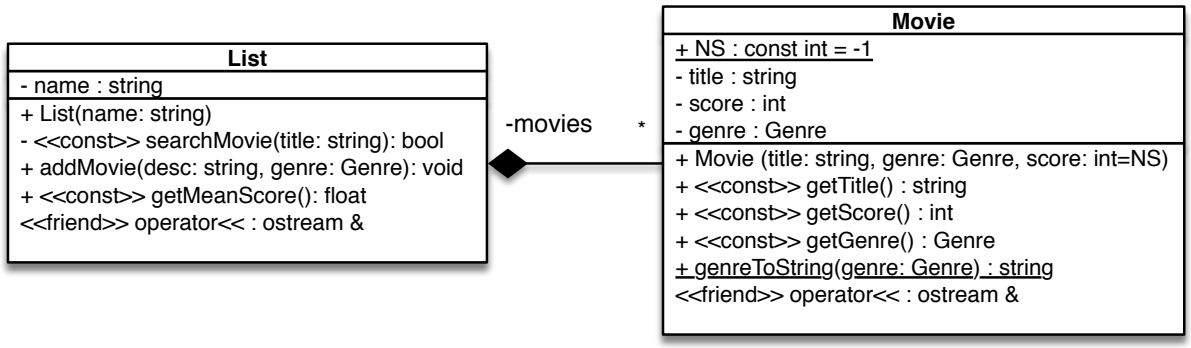
Si la palabra existe, se deberá extraer la lista de identificadores de documento en los que aparece y su correspondiente frecuencia. Con esa lista de identificadores se accederá al fichero binario `documents.bin` para obtener la información de cada documento.

Finalmente, de entre todos los documentos en los que aparece la palabra, se mostrará por pantalla únicamente la información de aquél en el que la palabra tenga una mayor *frecuencia relativa*. La frecuencia relativa se calcula como el número de veces que aparece la palabra en el documento (frecuencia), dividida por el número de palabras que contiene el documento (longitud). Por ejemplo, una palabra que aparezca 4 veces en un documento de 100 palabras, tendrá una frecuencia relativa en ese documento de $4/100 = 0.04$. La información a mostrar para el documento incluirá el título, la URL y la frecuencia relativa. Ejemplo de salida:

```
The cat in the hat (http://www.seussville.com/) [0.06]
```

2. **(4.5 puntos)**

Queremos hacer un programa para gestionar un servicio de vídeo online partiendo del siguiente diagrama:



Cada película (`Movie`) tiene un título, un género y una puntuación. Por defecto, la puntuación será NS (sin puntuar). El constructor debe lanzar una excepción si recibe una puntuación que no está entre -1 y 5. Al principio de `Movie.h` debemos declarar los posibles géneros: `enum Genre {Action, SciFi, Drama, Comedy};`

El método `genreToString` debe devolver el string correspondiente al género recibido (por ejemplo, `"Action"`). El operador salida imprimirá el título, la puntuación entre paréntesis, y el género, tal como puede verse en el ejemplo del final.

Una lista (`List`) tiene un nombre, y su constructor simplemente debe asignarlo con el valor recibido. El método `addMovie` recibe un `string` con el nombre de la película seguido de una coma y una puntuación, por ejemplo, `"A New Hope, 5"`. Como segundo parámetro le pasaremos el género. Este método debe separar el `string` recibido, crear una película con todos sus datos y añadirla al vector si se puede.

No se podrá añadir la película si el constructor de Movie lanza una excepción, en cuyo caso mostraremos el mensaje `Wrong movie` como puede verse en el ejemplo del final. Tampoco se añadirá si la película ya existía en el vector, aunque en este caso no hay que mostrar ningún mensaje. Para comprobar si una película está repetida podemos usar el método `searchMovie`, que debe compararla con su nombre y devolver `true` si existe.

El método `getMeanScore` debe devolver la puntuación media de todas las películas de la lista. El operador salida imprimirá el nombre de la lista, su puntuación media y todas las películas, como puede verse en el siguiente ejemplo. Dado este `main.cc`, que puedes compilar con `g++ Movie.cc List.cc main.cc -o ej2`:

```
int main() {
    List l("The best movies");

    l.addMovie("Brian's life, 4", Comedy);
    l.addMovie("A New Hope, 5", SciFi);
    l.addMovie("Sharknado, -10", SciFi);  // Excepcion
    l.addMovie("A New Hope, 5", SciFi); // Ya existe
    l.addMovie("Monster's Ball, 4",Drama);

    cout << l << endl;
}
```

La salida del programa debería ser la siguiente:

```
Wrong movie "Sharknado"
The best movies
4.33333
Brian's life (4) Comedy
A New Hope (5) SciFi
Monster's Ball (4) Drama
```