Programación 2

Examen de teoría (junio 2018)

29 de mayo de 2018

Instrucciones

- **Duración: 3 horas**
- El fichero del primer problema debe llamarse `c3pbot.cc`. Para el segundo problema es necesario entregar cuatro ficheros, llamados `Youtuber.cc, Youtuber.h, Category.cc, Category.h`. Pon tu DNI y tu nombre en un comentario al principio de todos 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)** Año 2050. El destino ha querido que acabes siendo profesor de Programación 2. Cansado ya de muchos años contestando las mismas tutorías, has decidido programar un *chatbot* (robot conversacional) para que las conteste por ti. El programa, al que has decidido bautizar como C3PBot, deberá pedir al alumno que escriba su consulta por teclado mostrando el mensaje:

   `Hola, soy C3PBot.  Escribe tu pregunta a continuación:`

   Para contestar al alumno, el programa contará con la ayuda de un fichero de texto llamado `respuestas.txt` que contendrá una lista con preguntas frecuentes y sus correspondientes respuestas asociadas. Cada pregunta[1] vendrá en una línea diferente, seguida de una barra vertical separadora '`|`' y de la respuesta asociada. Ejemplo de contenido del fichero:

   ```
   cuando es el examen de julio|El 12 de julio a las 9:00h.
   en que aulas es el examen de junio|Todas las de la EPS I, de la L01 a la L27.
   cuando hay que entregar la practica 3|La fecha tope es el 23 de mayo.
   ...
   ```

   El programa deberá calcular la pregunta del fichero de texto que más se parece a la que ha formulado el alumno, devolviendo la correspondiente respuesta asociada. Para hacer este cálculo, primero deberás eliminar de la consulta del alumno todos los caracteres que no sean espacios en blanco, números o letras[2] (signos de puntuación, etc.), pasando las letras a minúsculas.[3]

   A continuación, deberás comprobar cuántas palabras hay en común entre la pregunta hecha por el alumno y cada una de las preguntas del fichero de texto, devolviendo una puntuación que dependerá del número total de caracteres que hay en las palabras en común. Por simplificar, asume que no va a haber palabras duplicadas en la pregunta del alumno, ni tampoco en las del fichero. Por ejemplo, si el alumno escribe `Creo que voy a suspender, ¿cuando es la recuperacion de julio?`, la cantidad de caracteres en común con `cuando es el examen de julio` será igual a 15: 6 (`cuando`) + 2 (`es`) + 2 (`de`) + 5 (`julio`), y con `en que aulas es el examen de junio` de 7: 3 (`que`) + 2 (`es`) + 2 (`de`).

[1] Las preguntas del fichero de texto solo contienen números, letras en minúsculas y espacios en blanco.

[2] Puedes usar la función `bool isalnum(char c)` de la librería `cctype` que devuelve `true` si el carácter `c` es letra o número y `false` en caso contrario.

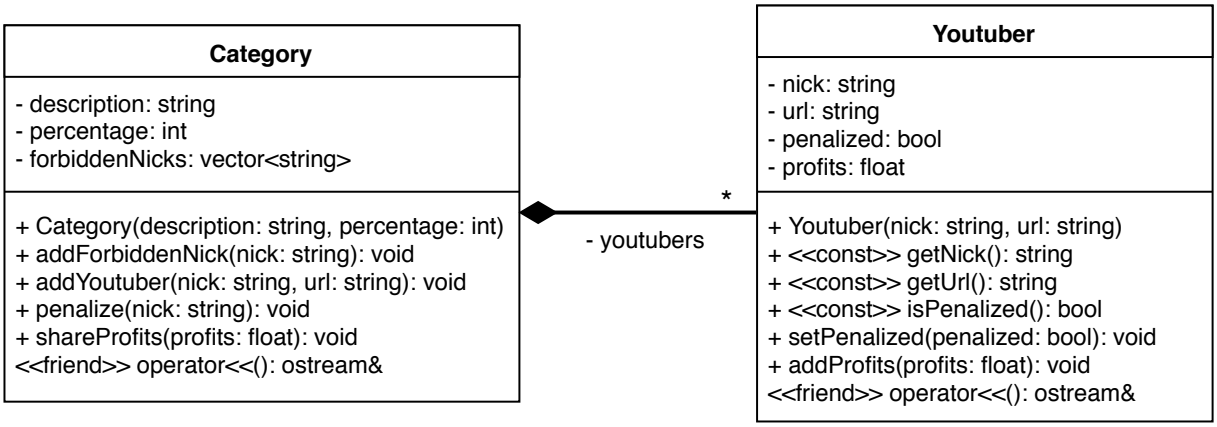
[3] Para pasar una letra a minúsculas puedes usar la función `char tolower(char c)` de la librería `cctype`. Si el parámetro `c` no es una letra mayúscula, `tolower` devuelve el mismo valor `c`.

La salida que proporcionará C3PBot será la respuesta asociada a la pregunta del fichero más similar a la pregunta del alumno, junto con un porcentaje de solapamiento entre ambas. Este valor se calculará como `total de caracteres solapados / total de caracteres de la consulta del alumno`. Para calcular el `total de caracteres de la consulta del alumno` solo se tiene en cuenta las letras y números de la pregunta del alumno (ni signos de puntuación, ni espacios en blanco, etc.). Ejemplo de salida para la pregunta anterior:

```
El 12 de julio a las 9:00h.  (30.6122%)
```

El valor `30.6122%` sale de dividir el número de caracteres solapados (15) por el total de caracteres de la pregunta de entrada (49). Una vez mostrada la salida, el programa terminará.

2. **(4.5 puntos)** Nuestra empresa de representación de youtubers necesita un programa para gestionar el reparto de beneficios (*profits*) entre nuestros representados. Para realizar el reparto se han establecido varias categorías según su fama, de acuerdo al siguiente modelo:



El constructor de la clase Youtuber recibe dos parámetros: `nick` y `url`. Para crear un youtuber se debe comprobar que ninguno de los dos campos estén vacíos. Para validar la url se debe comprobar además que la cadena recibida empieza por el texto "`http://www.youtube.com/channel/`" (pista: puedes usar el método `find` de la clase `string`). Si hay algún error en las validaciones se lanzará una excepción y no se creará el nuevo Youtuber. Los beneficios de un youtuber empezarán siendo cero, y se irán incrementando con cada llamada al método `addProfits`. Inicialmente, el youtuber no estará penalizado.

El total de beneficios se repartirá primero entre las distintas categorías de acuerdo al porcentaje establecido para cada una de ellas (p.ej. los más famosos se repartirán entre ellos el 70% de los beneficios). El constructor de la clase Category recibe un string con la descripción y un valor entero con el porcentaje a repartir entre sus miembros.

El método `addYoutuber` recibe el nick y la url y guardará el nuevo youtuber en la categoría si la información es correcta. Después de intentar crear el nuevo youtuber, se debe comprobar que el nick no esté prohibido para evitar conflictos con otros youtubers famosos. Para esto, se debe buscar si alguno de los nicks prohibidos del vector `forbiddenNicks` está contenido en el nuevo nick; por ejemplo, si "`Rubius`" es un nick prohibido, el nuevo nick "`FakeRubius`" no está permitido, pero sí lo estaría el nick "`Rubi`". En caso de que haya ocurrido algún error al crear el youtuber se debe mostrar el mensaje de error correspondiente ("`ERROR: DATA MISSING`", "`ERROR: WRONG URL <url>`" o "`ERROR: FORBIDDEN NICK <nick>`") y no añadir el youtuber. Por simplificar, no habrá nicks repetidos y no es necesario comprobarlo.

Un youtuber puede ser penalizado con el método `penalize` (p.ej. si ha publicado contenidos ilegales u ofensivos). Si un youtuber recibe dos penalizaciones deberá ser eliminado de la categoría mostrando el mensaje "`<nick> REMOVED`" y su nick pasará a formar parte de los nicks prohibidos, utilizando el método `addForbiddenNick`.

Finalmente, el método `shareProfits` se encargará de repartir el dinero recaudado entre los youtubers de cada categoría. Para esto calculará el porcentaje de los beneficios a repartir que corresponde a la categoría (el parámetro `profits` es el total de beneficios de todas las categorías juntas), y el dinero resultante se repartirá a partes iguales entre todos los miembros de la categoría.

Dado el siguiente `main.cc`, que puedes compilar con `g++ Youtuber.cc Category.cc main.cc -o ej2`:

```
#include "Category.h"

int main() {
    Category c1("Famous", 70);
    Category c2("Noobs", 30);

    c1.addForbiddenNick("Rubius");
    c1.addForbiddenNick("Auron");
    c1.addForbiddenNick("Dulceida");
    c1.addForbiddenNick("Adelita Power");

    c1.addYoutuber("ExpCaseros", "http://www.youtube.com/channel/expcaseros");
    c1.addYoutuber("Wismichu", "http://www.youtube.com/channel/wismichu");
    c1.addYoutuber("PewDiePie", "http://www.youtube.com/channel/pewdiepie");
    c1.addYoutuber("FakeRubius", "http://www.youtube.com/channel/fakerubius");

    c1.penalize("PewDiePie");

    c2.addYoutuber("Wannabe", "http://instagram.com/wannabe");
    c2.addYoutuber("Adelaida", "http://www.youtube.com/channel/adelaida");

    c1.penalize("PewDiePie");
    c1.addYoutuber("PewDiePie2", "http://www.youtube.com/channel/pewdiepie2");

    c1.shareProfits(12345.0);
    c2.shareProfits(12345.0);

    cout << c1;
    cout << c2;
}
```

La salida debe ser:

```
ERROR: FORBIDDEN NICK FakeRubius
ERROR: WRONG URL http://instagram.com/wannabe
PewDiePie REMOVED
ERROR: FORBIDDEN NICK PewDiePie2
---- Category: Famous ----
ExpCaseros, url=http://www.youtube.com/channel/expcaseros, profts=4320.75
Wismichu, url=http://www.youtube.com/channel/wismichu, profits=4320.75
---- Category: Noobs ----
Adelaida, url=http://www.youtube.com/channel/adelaida, profits=3703,5
```