Programación 2

Examen de teoría (junio 2015)

5 de junio de 2015

Instrucciones

- **Duración: 3 horas**
- El fichero del primer problema debe llamarse `analizador.cc`. Para el segundo problema es necesario entregar cuatro ficheros, llamados `Scheduler.cc, Scheduler.h, Hoguera.cc, Hoguera.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 de ejemplo de entrada que te pueden servir de ayuda para evaluar los problemas del examen, y el apartado `Reference` de la web `www.cplusplus.com`, para hacer consultas sobre sintaxis.

Problemas

1. **(5.5 puntos)**

   Debemos realizar un programa para analizar los comentarios que se hacen de nuestros políticos en la red social Twitter. El programa recibirá como entrada tres parámetros. El primero de ellos será un fichero de texto con una lista de *tweets* (ya sabéis, mensajes en Twitter), cada uno en una línea. Por ejemplo:

   ```
   pedro @sanchezcastejon abre ronda sobre pactos y hablara con @pabloiglesias y @albertrivera
   es terrible que los pactos se hagan despues de las elecciones y no antes @agarzon
   @beatriztalegon habra que pedirle a @sanchezcastejon que acuerde con @ahorapodemos
   @albertrivera exige primarias a @marianorajoy para aceptar pactos
   ```

   El segundo parámetro será el nombre de un fichero de texto que contendrá una lista de cuentas de Twitter de políticos (empiezan por el carácter `@` y contienen una secuencia de una o más letras), cada una en una línea[1]. Por ejemplo:

   ```
   @pabloiglesias
   @sanchezcastejon
   @ritabarbera
   ```

   Por último, el tercer parámetro será una única palabra (que sólo contendrá letras). Por ejemplo: `pactos`

   El objetivo del programa es realizar una estadística que muestre, para cada una de las cuentas de políticos listados en el fichero, el número de *tweets* que mencionan a esa cuenta (*total*), el número de ellos que además incluyen la palabra pasada como parámetro (*encontrados*), y la proporción (*ratio*) de *tweets* que contienen la palabra sobre el total de *tweets* que mencionan esa cuenta (*ratio=encontrados/total*).

   Un ejemplo de llamada sería `analizador tweets.txt politicos.txt pactos`

   Un ejemplo de salida con los datos presentados arriba sería:

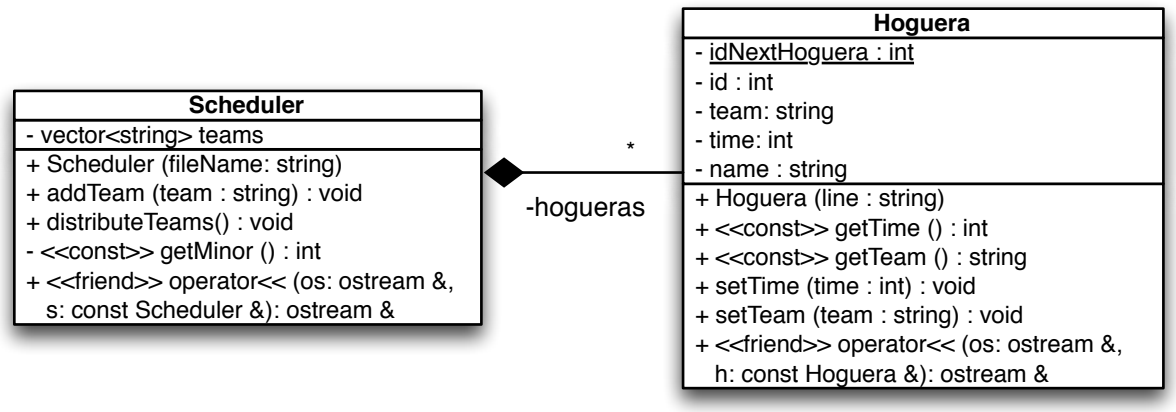
   ```
   @pabloiglesias (Total: 1; Encontrados: 1; Ratio: 100%)
   @sanchezcastejon (Total: 2; Encontrados: 1; Ratio: 50%)
   @ritabarbera (Total: 0; Encontrados: 0; Ratio: 0%)
   ```

   Notas: Si no se puede abrir alguno de los ficheros, el programa no debe continuar. Por simplificar se considera que todo el texto de entrada estará en minúsculas, y solamente tendrá palabras y menciones a usuarios. Puede haber *tweets* que no mencionen a ningún político de la lista y pueden haber políticos que no sean mencionados por ningún *tweet*. Las palabras no se tienen en cuenta si aparecen como subcadena de otra (es decir, si en el *tweet* aparece `compactos` no debemos considerar que se ha encontrado `pactos`).

[1] El formato del fichero será correcto, no es necesario comprobarlo.

2. **(4.5 puntos)**

Queremos hacer un programa que nos permita asignar distintos equipos de bomberos a las hogueras repartidas por Alicante.



Una hoguera tiene un nombre (`name`), un identificador único (`id`), el nombre del equipo de bomberos (`team`) asignado para apagarla, y el tiempo (`time`) que se tarda en hacerlo. El constructor crea una hoguera a partir de un string con el formato `name:time`, por ejemplo: `"Maissonnave:14"`, asignando al atributo `id` el valor de `idNextHoguera` (inicialmente 1), e incrementándolo para la siguiente hoguera que se cree. En el constructor se dejará el equipo inicialmente vacío. El operador salida debe imprimir la hoguera con el formato `name (id)=time`. Por ejemplo: `Maissonnave (2)=14`.

El organizador (`Scheduler`) asigna los equipos de bomberos (`teams`) a las distintas hogueras. Para ello, el constructor carga la información de las hogueras a partir de un fichero[2] cuyo nombre recibe por parámetro. Este fichero contendrá una hoguera por línea. El método `addTeam` simplemente añade un equipo al vector de equipos[3]. El método `distributeTeams` es el que se encarga de asignar las hogueras a los equipos de bomberos, asignando la hoguera de menor tiempo al primer equipo, después haciendo lo mismo para el segundo equipo, y así para el resto de hogueras. Para ello, debe usar la función `getMinor`, que devuelve la posición (en el vector) de la hoguera sin equipo asignado que tenga un tiempo menor. Esta función devolverá `-1` si todas las hogueras tienen ya equipo asignado. Finalmente, el operador salida debe imprimir la información de las hogueras agrupadas por equipos como se muestra en el siguiente ejemplo[4]. Dado el siguiente `main.cc`:

```
#include "Scheduler.h"

int main()
{
  Scheduler f("hogueras2015.txt");

  f.addTeam("Equipo A");
  f.addTeam("Equipo Benalua");
  f.addTeam("Equipo San Juan");

  f.distributeTeams();

  cout << f << endl;
}
```

...y el fichero `hogueras2015.txt`, el programa debería imprimir:

```
[ Equipo A ] : Carolinas Altas (3)=11 Oscar Espla (6)=15 Princesa Mercedes (7)=5
[ Equipo Benalua ] : Gran Via (4)=13 Benalua (5)=20 San Anton Alto (8)=6
[ Equipo San Juan ] : Altozano (1)=10 Maisonnave (2)=14
```

[2]Hay que comprobar que el fichero se pueda abrir correctamente, y en caso contrario mostrar un error. El formato del fichero será correcto, no es necesario comprobarlo.

[3]No hace falta comprobar que haya duplicados

[4]Si un equipo no tiene hogueras asignadas se mostrará solamente el nombre del equipo y el carácter ":", con el mismo formato que en el ejemplo.