Programación 2

Examen de teoría (junio 2013)

4 de junio de 2013

Instrucciones

- **Duración: 3 horas**
- El fichero del primer ejercicio debe llamarse `ej1.cc`. Para el segundo problema es necesario entregar cuatro ficheros, llamados `Partida.cc, Partida.h, Jugador.cc, Jugador.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. **(5.5 puntos)**

   Debemos de realizar un programa para analizar los datos de los socios de un club deportivo. Para ello, contamos con un fichero con información de los socios y las actividades que practican.

   El fichero contiene información de las inscripciones de los socios a los distintas actividades[1]. Cada línea contiene el nombre de un socio y las actividades a las que está apuntado separadas por comas. Por ejemplo:

   ```
   Juan Lopez,judo,baloncesto
   Pedro Pomez,tenis,futbol
   Paco Pil,futbol,natacion
   Juan sin miedo,futbol
   Jesus Panizo,tenis,baloncesto,futbol
   ```

   El usuario ejecutará el programa pasándole como parámetro el nombre del fichero de las actividades. El programa devolverá como resultado una lista con las actividades y los socios que practican esa actividad. Un ejemplo de llamada sería `./ej1 actividades.txt`, y una salida con los datos anteriores:

   ```
   judo: Juan Lopez
   baloncesto: Juan Lopez, Jesus Panizo
   tenis: Pedro Pomez, Jesus Panizo
   futbol: Pedro Pomez, Paco Pil Pil, Juan sin miedo, Jesus Panizo
   natacion: Paco Pil Pil
   ```

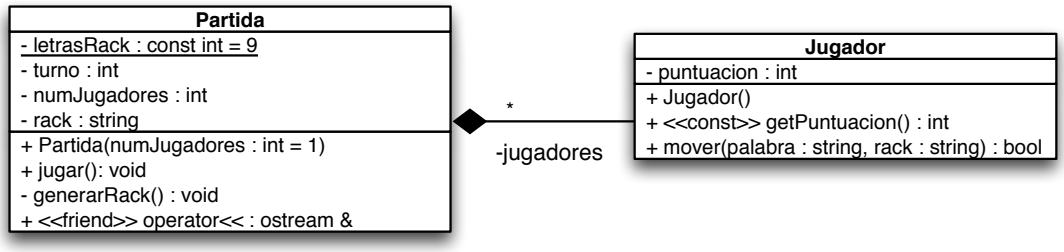
   Nota: Este programa debe implementarse usando programación procedimental (no orientada a objetos). Para resolverlo te hará falta crear un registro `Actividad` y hacer un vector STL de actividades (si no te sale con vectores, puedes usar arrays). Supondremos que el formato de ambos ficheros es siempre correcto. Debe comprobarse que el número de parámetros de entrada es correcto. También deben comprobarse los errores de apertura de fichero, pero no es necesario controlar los de lectura y escritura (fail).

2. **(4.5 puntos)**

   Queremos hacer un programa que nos permita jugar al juego Letras (basado en el programa Cifras y Letras). En este juego se genera aleatoriamente un rack de 9 letras (el mismo para todos), y los jugadores deben construir la palabra más larga[2] con las letras de este rack. Por cada letra de la palabra formada se sumará 1 punto a la puntuación del jugador. A continuación se muestra el diagrama de clases:

[1] El nombre de cada actividad debe cogerse del fichero, no hay siempre las mismas actividades.

[2] Siempre y cuando esté en un diccionario de palabras válidas, pero no lo comprobaremos en este ejercicio.



El método `mover` de la clase `Jugador` debe comprobar que todas las letras de la palabra están en el rack. Si es así, actualizará la puntuación del jugador sumando un punto por letra y devolverá `true`. En caso contrario, mostrará el mensaje `"Palabra incorrecta"` y devolverá `false` sin modificar la puntuación.

En la clase `Partida`, `generarRack` creará un rack aleatorio de 9 letras mayúsculas comprendidas entre la A y la Z[3]. Si el rack ya tenía algo se borrará su contenido, por lo que siempre se sobreescribe el rack anterior.

El método `jugar` debe generar un rack aleatorio llamando a `generarRack` y mostrarlo por pantalla con el mensaje `"Rack="`. A continuación, pedirá a los jugadores que introduzcan las palabras mediante el mensaje `"Jugador i="`, siendo `i` el número del jugador actual, y se comprobará que éstas sea válidas mediante el método `mover`. Cada vez que un jugador introduzca una palabra correcta se cambiará el turno, y tras cada ronda se generará un nuevo rack aleatorio. La partida terminará cuando algún jugador introduzca `"q"`.

Finalmente, el operador salida mostrará las puntuaciones totales de los jugadores.

Dado el siguiente fichero[4] `main.cc`:

```
#include <iostream>
#include <stdlib.h>
#include "Partida.h"
using namespace std;

int main()
{
   srand(2);

   Partida p(2);       // Creamos una partida para 2 jugadores
   p.jugar();          // Los jugadores introducen palabras hasta terminar la partida con "q"
   cout << p << endl;  // Se imprimen las puntuaciones finales
}
```

...a continuación se muestra un ejemplo de juego (en negrita lo que introduce el usuario):

```
Rack= EBGNHAMDH
Jugador 1= DAME
Jugador 2= GEMA
Rack= NUXBVZLUF
Jugador 1=ZULU
Jugador 2= UUUUU
Palabra incorrecta
Jugador 2=LUZ
Rack= PKKSNBVDS
Jugador 1= q
Puntuacion jugador 1= 8
Puntuacion jugador 2= 7
```

Ayuda: Puedes descargar el `makefile` de `http://www.dlsi.ua.es/~pertusa/exam/makefile` para compilar el programa.

[3]Para generar una letra aleatoria puedes usar `letra=(char)('A'+rand()%26)`. Las funciones `rand` y `srand` están en `stdlib.h`.
[4]Se ha elegido srand(2) para que la secuencia de letras aleatorias de este ejemplo contenga vocales.