Trabajo práctico Labo: Implementación en C++

"CorrePocoyo"

Normativa

Límite de entrega: Miércoles 22 de abril de 2015 a las 12:00 hs. (TM) o las 20:00 hs. (TT)) hs.
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://www.dc.uba.ar/materias/aed2/2014/1c/informacion`)

1. Normas de entrega adicionales

- **Este TP se realiza de a dos.** No es necesario informar el grupo.
- **La entrega es presencial**; si alguien no puede asistir, con justificativo debe comunicarse con los JTPs.
- El día de la entrega deberán disponer de una copia del código que pueda ejecutarse en los laboratorios.
- Para aprobar, además del correcto funcionamiento del TP deben estar presentes ambos integrantes y responder las preguntas del corrector.

2. Enunciado

Pocoyó [1] no necesita presentaciones. En este episodio se encuentra corriendo la carrera de su vida [2]. Para esto nos interesa poder al principio agregar a todos los participantes de la carrera.

Luego, como en toda carrera, podemos indicar un elemento y decir que éste "pasa" al que está inmediatamente delante de él.

En este tipo de carreras suele darse también que un nuevo competidor se incorpore a la carrera. Ingresará en cualquier lugar. Para eso le indicaremos delante de quién se estará colocando. De no hacerlo, lo hará detrás del último.

Otras veces sucede que un competidor se agota y deja la carrera.

A su vez la cámara estará siguiendo a uno de los competidores. Podremos pues "avanzar" o retroceder la misma en cuanto a qué competidor está filmando.

Se pide:

1. Implementar en C++ la clase paramétrica[3] `CorrePocoyo<T>`, cuya interfase se provee en el archivo `.h` adjunto, junto con la implementación de todos los métodos públicos que en ella aparecen.
   No pueden agregar nada público. Sí pueden, y deben, agregar cosas privadas, en particular los campos que les parezcan pertinentes. También pueden agregar funciones auxiliares, tanto de instancia como estáticas, clases auxiliares, etc., pero nada en la parte pública de la clase.
2. Implementar las funciones de test no implementadas en `tests.cpp`. El correcto funcionamiento de los test **no es garantía** de aprobación.
3. La implementación dada no debe perder memoria en ningún caso. Al momento de la corrección se hará el chequeo pertinente usando *valgrind*.

[1] http://en.wikipedia.org/wiki/Pocoyo
[2] https://www.youtube.com/watch?v=zFkNEWLpKyM
[3] Para poder contar con diversos autos :-)

3. Recomendaciones

El objetivo de este trabajo práctico es familiarizarse con el lenguaje C++ y las características del mismo que se usarán en esta materia (templates, memoria dinámica, etc.), de manera de llegar mejor preparados a afrontar un desarrollo más grande y complicado como el TP3.

Respecto de los archivos provistos, si intentan compilar `tests.cpp` podrán hacerlo, pero no podrán linkearlo y generar un binario porque, por supuesto, va a faltar la implementación de todos los métodos de la clase testeada.

Una sugerencia para empezar es dejar la implementación de todos los métodos necesarios escrita, pero vacía, de manera de poder compilar. Pueden comentar todos los tests que requieran métodos aún no implementados de manera de poder usar la aplicación para el testing a medida que van implementando. Tengan en cuenta que algunos métodos pueden ser necesarios para muchas de las funciones de test, por lo tanto, es aconsejable empezar por esos test.

Además de los casos de test provistos por la cátedra les recomendamos fuertemente que realicen sus propios tests.

Dado que su implementación no debe perder memoria, es una buena práctica utilizar la herramienta *valgrind* durante el desarrollo, como mínimo en la parte de testing final.