Trabajo práctico Labo: Implementación en C++

"Messinería"

Normativa

Fecha de entrega: Miércoles 23 de abril de 2014

Normas de entrega: Las contenidas en la página web de la materia.

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

El ídolo del fútbol mundial, Lionel Messi©, está teniendo cada vez más influencia en la vida de las personas. Gracias a sus continuas y maravillosas actuaciones, está logrando que mucha gente se fanatice y se obsesione por él. Debido a estos acontecimientos, nació la Messineria.

Los adeptos a la Messineria alrededor del mundo se reúnen en sectas clandestinas para realizar alabanzas a su ídolo.

El rito sectario que profesan los acólitos consiste en arrodillarse uno al lado del otro (formando un círculo) alrededor de un monolito con la imagen de Messi[1], alabándolo en orden.

Cada vez que "Lío" hace un gol, un adepto nuevo se incorpora. Además, en dicha secta puede haber un único adepto distinto a los demás, denominado como "El Elegido". Si ya tenemos al Elegido el nuevo adepto deberá arrodillarse a la izquierda de él. En caso contrario, se arrodillara a la izquierda del que esté alabando.

De forma análoga, cada vez que Cristiano Ronaldo® hace un gol, un adepto se desliga de la secta.

Cuando un adepto decide que su turno de alabanza finalizó, le cede el turno al que está arrodillado a la derecha de él.

La facultad que lo distingue al Elegido de todos los demás es la de poder interrumpir el turno de cualquier otro con el fin de tomar el control de alabar. Sin embargo, cuando el Elegido decide ceder el turno, le tocará a quien había sido interrumpido.

Algunas veces sucede que los adeptos se olvidan de pedir algo más en sus alabanzas. Según las reglas de la secta se permite que un adepto que acaba de ceder su turno, vuelva a pedirlo. Dicho turno será otorgado en ese instante al inmediato anterior. Aquí no se tienen en cuenta las interrupciones del Elegido.

Se pide:

1. Implementar en C++ la clase paramétrica[2] `Messineria<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://goo.gl/QruQqH

[2] Para poder contar con Messinerías en distintos idiomas :-)

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.