TP Labo – AlgoJukeBox

2do cuatrimestre 2012

Fecha de entrega: 19 de Septiembre de 2012 a las 18hs
Normas de entrega:

- **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 JTP's.
- Se deberá entregar un e-mail a `algo2.entregas@gmail.com` con asunto `Entrega TP0: [Alumno 1] - [Alumno 2]` contando con el listado de los integrantes en el cuerpo y, adjunto, el código fuente sin compilar comprimido.
- El día de la entrega deberán disponer de una copia del código que pueda ejecutarse en los laboratorios a fin de evaluar el correcto funcionamiento.
- Deben verificar que no se pierda memoría usando la herramienta *valgrind*.
- El resto de las normas son las contenidas en la página web de la materia.

1. Situación

Se desea comenzar a implementar un reproductor de música que operará de forma similar a una rocola. El sistema contendrá una colección de discos que se podrán recorrer en forma cíclica. Al ir recorriendo dichos discos habrá uno que será mostrado al usuario, quien podrá optar por seguir recorriendo la colección o reproducir dicho disco.

La colección no estará fija. Se debe permitir agregar o quitar discos durante el funcionamiento del sistema. Cuando un disco se ingresa, éste queda como disco actual.

A modo de ejemplo, en un sistema recién iniciado (sin discos), se ingresa: primero el Disco A, segundo el Disco B y por último el Disco C. El Disco C es el actual. Al recorrer la colección para ver el siguiente disco, deberá aparecer el Disco B. Al ver el siguiente disco, deberá aparecer el Disco A. Al repetir la operación, aparecerá el Disco C. Si el usuario elije retroceder al disco anterior, se volverá al Disco A.

El usuario también podrá elegir reproducir el disco actual. El sistema debe marcarlo a fin de saber cuál es el último disco mandado a reproducir. Notar que, en todo momento, a lo sumo hay un solo disco marcado.

2. Enunciado

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

Para este trabajo implementaremos una lista circular paramétrica, también conocida como Anillo. Está definida su interfase en el archivo `anillo.h`. En dicho archivo se detalla el comportamiento esperado de cada función. También se provee un archivo `tests.cpp` con ejemplos de tests que esperamos servirán de inspiración.

Para este trabajo tienen que completar la clase `Anillo` 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 `Anillo`. Además, para la entrega deben agregar sus propios tests en el archivo `tests.cpp`.

El objetivo es minimizar la cantidad de operaciones requeridas para las operaciones `siguiente`, `agregar`, `retroceder` y `marcar`. Por otra parte, todas las funciones deberían ser tan eficientes como sea posible. En el caso de que una operación sea implementada de una manera poco eficiente, se les pedirá que la implementen nuevamente. En caso de duda, recomendamos consultar a los docentes. Además, no deben reimplementar la misma funcionalidad repetidas veces, sino que deben utilizar funciones auxiliares en tales casos. Finalmente, las funciones deben ser concisas; funciones que implementen varias funcionalidades separables, deben ser separadas.

Implementar las funciones de test no implementadas en `tests.cpp`.

La implementación dada no debe perder memoria en ningún caso. Al momento de la corrección se hará el chequeo pertinente usando *valgrind*. Les sugerimos fuertemente que también lo usen durante el desarrollo, como mínimo en la parte de testing final.

3. Recomendaciones

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 Anillo.

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. Fíjense antes que algunos métodos son necesarios para todas las funciones de test.

A la hora de implementar sus propios tests, pueden usar las funciones auxiliares definidas en `aed2_tests.h`, si les resultan útiles.