Taller de C++ 1: Memoria dinámica
Implementación de una pila

Normativa

Límite de entrega: Miércoles 31 de agosto de 2016 a las 12:00 hs.
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://www.dc.uba.ar/materias/aed2/2016/2c/cursada`)

Enunciado

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

Se pide:

1. Implementar en C++ la clase Pila cuya interfaz será entregada por la cátedra en el archivo Pila.h adjunto. Dicha interfaz está basada en el TAD Pila provisto con anterioridad en el apunte de TADs Básicos.
2. No deben agregar ningún método o variable nueva, sólo completar los que se discrimina en el archivo dado. Pueden agregar funciones auxiliares en la parte privada pero nada en la parte pÃžblica de la clase.
3. Para la entrega por mail, adjuntar Ãžnicamente los archivos Pila.cpp y Pila.h.

Normas de entrega adicionales

- Este taller se realiza de forma **individual** y con entrega digital.
- Al finalizar la implementación se deberá enviar el archivo con la soluciÃşn a la siguiente dirección de correo electrónico: algo2.dc+taller1@dc.uba.ar. El asunto deberÃą indicar su libreta universitaria en formato XXX/YY.
- El trabajo se evaluará automáticamente utilizando para ello una batería de tests que no serán provistos a los alumnos. Si el trabajo resulta aprobado se reenviará, también de forma automática, un correo respuesta informando esta situación. En caso negativo la respuesta indicará que tipo de problema no pudo resolver el código en cuestión. El sistema deberÃıa responder en un tiempo razonable si el TP fue aprobado o no. Si no recibe confirmaciÃşn de la recepciÃşn de su TP luego de 30 min., por favor informe su situaciÃşn a la lista de docentes. Se pueden hacer tantas entregas como se necesiten, hasta la fecha lÃımite de entrega.
- Durante la clase del miércoles 31 de octubre, de forma aleatoria se elegirán alumnos para que brinden un breve **coloquio** acerca del trabajo realizado en el taller. Los coloquios se realizarán en todos los talleres hasta completar la cantidad total de alumnos. El día del coloquio se deberá disponer de una copia del código que pueda ejecutarse en los laboratorios.
- La implementación dada no debe perder memoria en ningún caso. Se recomienda la verificación mediante el uso de *valgrind*.
- EstÃą prohibido utilizar la biblioteca estÃąndar de C++ (STL) con la excepciÃşn de las siguientes librerÃıas: **iostream**, **string** y **cassert**.
- Se admite el estÃąndar de C++ hasta el aÃño 2003. EstÃą prohibido usar las funcionalidades que agrega el estÃąndar C++11 o posteriores.

Recomendaciones

- Respecto de los archivos provistos, si intentan compilar `tests.cpp` podrÃąn hacerlo, pero no podrÃąn linkearlo y generar un binario porque, por supuesto, va a faltar la implementaciÃşn de todos los mÃ©todos de la clase testeada.
- Una sugerencia para empezar es dejar la implementaciÃşn de todos los mÃ©todos necesarios escrita, pero vacÃıa, de manera de poder compilar. Pueden comentar todos los tests que requieran mÃ©todos aÃžn no implementados de manera de poder usar la aplicaciÃşn para el testing a medida que van implementando. Tengan en cuenta que algunos mÃ©todos pueden ser necesarios para muchas de las funciones de test, por lo tanto, es aconsejable empezar por esos test.

- AdemÃąs de los casos de test provistos por la cÃątedra les recomendamos fuertemente que realicen sus propios tests.
- Dado que su implementaciÃşn no debe perder memoria, es una buena prÃąctica utilizar la herramienta *valgrind* durante el desarrollo, como mÃınimo en la parte de testing final.
- Se sugiere chequear las precondiciones con **assert** para facilitar la depuraciÃşn de errores.
- Se sugiere repasar el capÃıtulo 10 del Cormen (ver bibliografÃıa en la pÃągina de la materia).