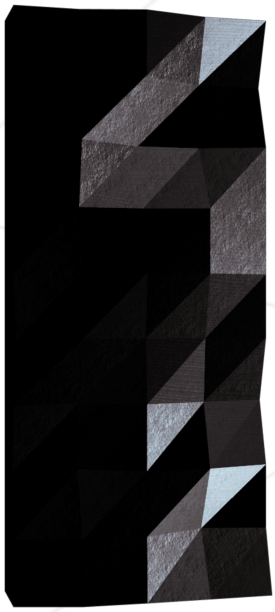

C++ - Módulo 00

Namespaces, class, funciones miembro, stdio streams, lista de inicialización, static, const, y montones de cosas basicas

Resumen:
Este documento contiene los ejercicios del módulo 00 de C++.

Versión: 8

Índice general

Capítulo I

Introducción

C++ Es un lenguaje de programación de propósito general creado por Bjarne Stroustrup como una extension del lenguaje de programación C, o Ç con clases"(fuente: Wikipedia).

El objetivo de estos módulos es presentarte la **Programación orientada a objetos**. Este será el puto de partida de tu viaje en C++. Hay muchas opciones para aprender POO. Decidimos elegir C++ ya que se deriva de tu viejo amigo C y para mantener las cosas simples, debido a la complejidad del lenguaje tu código deberá compilar con el estándar C++98.

Somos consciente, el lenguaje C++ es moderno y diferente en muchos aspectos. Si quieres convertirte en un especialista en este lenguaje, ¡dependerá de ti ir más allá del 42 Common Core!

Poco a poco irás descubriendo nuevos conceptos. Los ejercicios aumentarán su dificultad progresivamente.

Capítulo II

Reglas generales

Compilando

- Compila tu código con `c++` y los flags `-Wall -Wextra -Werror`
- Tu código deberá compilar si además agregas el flag `-std=c++98`

Formato y convención de nomenclatura

- Los directorios de los ejercicios serán nombrados de la siguiente manera: `ex00, ex01, ... , exn`
- Nombra tus ficheros, clases, funciones, funciones miembros y atributos como lo requieran las reglas.
- Escribe los nombres de las clases con el formato **UpperCamelCase**. Los archivos que contengan código de clase deben ser nombrados de acuerdo con el nombre de la clase. Por ejemplo:
  `ClassName.hpp`/`ClassName.h`, `ClassName.cpp`, o `ClassName.tpp`. Entonces, si tu tienes un archivo header que contiene la definición de la clase "BrickWall"que representa una pared de ladrillos, su nombre será `BrickWall.hpp`.
- Si no se especifica lo contrario, todos los mensajes de salida deben terminar con un salto de linea y mostrarse en la salida estándar.
- *¡Adiós Norminette!* No se aplica ningún estilo de codificación en los módulos de C++. Puedes seguir a tu favorito. Pero ten en cuenta que un código que tus evaluadores no puede entender es un código que no pueden calificar. Haz tu mejor esfuerzo para escribir un código limpio y legible.

Permitido/Prohibido

No estas codificando en C. ¡llegó la hora de C++! Por lo tanto:

- Se te permite utilizar casi todo de la biblioteca estándar. De este modo, en lugar de seguir aferrado a lo que ya sabes sería inteligente usar, tanto como sea posible, las versiones de C++ de las funciones de C a las que estás acostumbrado.

- Sin embargo, no puedes usar ninguna otra biblioteca externa. Esto significa que ni C++11 ni ninguno de sus derivados, así como las librerías `Boost` están permitidos. Las siguientes funciones también están prohibidas: `*printf()`, `*alloc()` y `free()`. Si los usas, tu calificación sera un 0 definitivo.
- Ten en cuenta que, a menos que se indique explícitamente lo contrario, el `using namespace <ns_name>` y la palabra reservada `friend` están prohibidas. De lo contrario, tu calificación será -42.
- **Podrás usar la librerías STL unicamente en el modulo 08.** Esto significa: No podrás utilizar **Contenedores** (vector/list/map/etcétera) ni **Algorítmos** (o cualquier cosa que requiera incluir el header `<algorithm>`) hasta ese momento. De lo contrario, tu calificación será un -42.

Algunos requisitos de diseños

- Aunque no lo creas se producen fugas de memoria en C++. Cuando reservas memoria (utilizando la palabra reservada `new`), debes evitar **memory leaks**.
- Desde el Modulo 02 al Modulo 08, Tus clases deben estar diseñadas en la **forma canónica ortodoxa, excepto cuando expresamente se indique lo contrario**.
- Cualquier implementación de una función en un archivo header (excepto para funciones de tipo template) significaría un 0 para el ejercicio.
- Deberías poder usar cada uno de tus headers independientemente de los demás. Por lo tanto, deben incluir todas las dependencias que necesitan. No obstante, debes evitar el problema de la doble inclusión agregando **include guards**. De lo contrario, tu calificación sera un 0.

Léeme

- Puedes agregar algunos archivos adicionales si necesitas (p.ej. para organizar tu código). Como estos ejercicios no serán verificados por un programa, siéntete libre de hacerlo siempre y cuando entregues los archivos obligatorios.
- a veces, los requerimientos de un ejercicio parecen breves, pero los ejemplos pueden mostrar requisitos que no están escritos explícitamente en las instrucciones.
- ¡Lee cada modulo completamente antes de empezarlos! De verdad, hazlo.
- ¡Por Odin!, ¡Por Thor! ¡¡¡Usa tu cerebro!!!

Tendrás que implementar muchas clases. Esto puede ser tedioso, a no ser que domines los scripts en tu editor de texto favorito.

Tienes cierta libertad para completar los ejercicios. Sin embargo, Sigues las reglas obligatorias y no seas un sangana/o. ¡Te perderías mucha infomación útil! No dudes en leer sobre conceptos teóricos.

Capítulo III

Ejercicio 00: Megáfono

	Ejercicio: 00
Megáfono	
Directorio de entrega: *ex*00/	
Archivos de entrega: `Makefile, megaphone.cpp`	
Funciones prohibidas: `Ninguna`	

Para asegurarnos de que todo el mundo está despierto, escribe un programa que tengas el siguiente comportamiento:

```
$>./megaphone "shhhhh... I think the students are asleep..."
SHHHHH... I THINK THE STUDENTS ARE ASLEEP...
$>./megaphone Damnit " ! " "Sorry students, I thought this thing was off."
DAMNIT ! SORRY STUDENTS, I THOUGHT THIS THING WAS OFF.
$>./megaphone
* LOUD AND UNBEARABLE FEEDBACK NOISE *
$>
```

`Resuelve el ejercicio al estilo c++.`

Capítulo IV

Ejercicio 01: Mi extraordinaria agenda

	Ejercicio: 01
Mi extraordinaria agenda	
Directorio de entrega: $ex01/$	
Archivos de entrega: `Makefile, *.cpp, *.{h, hpp}`	
Funciones prohibidas: `Ninguna`	

¡Bienvenido a los años 80 y su sorprendente tecnología! Escribe un programa que se comporte como una apestosa pero increíble agenda.

Deberás implementar dos clases:

- **PhoneBook**
  - Tiene un array de contactos.
  - Puede almacenar un máximo de **8 contactos**. Si el usuario intenta agregar un noveno contacto, reemplace el más antiguo por el nuevo.
  - Ten en cuenta que la reserva de memoria dinámica está prohibida
- **Contact**
  - Soporte para una agenda

En tu código, la agenda debe ser declarada como una instancia de la clase **PhoneBook**. Lo mismo para los contactos. Cada uno de ellos debe ser declarado como una instancia de la clase **Contact**. Siente libre de diseñar las clases como desees, pero ten en cuenta que cualquier cosa que se utilice dentro de una clase es privada, y que cualquier cosa que pueda usarse fuera de una clase es publica.

No olvides ver los videos de la intranet.

Al iniciar el programa, la agenda estará vacía y se le solicitara al usuario que ingrese uno de los siguientes comandos: ADD, SEARCH y EXIT. El programa unicamente aceptará estos.

- **ADD**: Agrega un nuevo contacto
  - Si el usuario introduce este comando, se le solicitará rellenar la información del nuevo contacto un campo a la vez. Una vez completados todos los campos, agregarás el nuevo contacto en la agenda.
  - Los campos son: first name, last name, nickname, phone number, y darkest secret. Ninguno de los campos puede quedar vacío.
- **SEARCH**: Muestra un contacto especifico
  - Muestra los contactos guardado en una lista de **4 columnas**: index, first name, last name and nickname.
  - Cada columna debe ser de una longitud de **10 caracteres**, separadas por el carácter pipe ('|'). El texto debe estar alineado a la derecha. Si el texto es más largo que la longitud de la columna, este debe ser truncado y el ultimo carácter imprimible es reemplazado por un punto ('.').
  - Por último, el programa pedirá al usuario introducir el indice de la entrada a buscar y mostrará la información del contacto. Si el indice esta fuera del rango o es incorrecto, trátalo adecuadamente. En cualquier otro caso, muestra la información del contacto, un campo por línea.
- **EXIT**
  - El programa termina y los contactos se pierden para siempre!
- **Cualquier otro comando será descartado.**

Una vez que finalice correctamente la ejecución de un comando, El programa quedará a la espera de otro. El programa finaliza cuando el usuario ingresa EXIT.

Dale un nombre relevante a tu programa.

http://www.cplusplus.com/reference/string/string/ y por supuesto
http://www.cplusplus.com/reference/iomanip/

Capítulo V

Exercise 02: El trabajo de tus sueños

	Ejercicio: 02
El trabajo de tus sueños	
Directorio de entrega: *ex*02/	
Archivos de entrega: `Makefile, Account.cpp, Account.hpp, tests.cpp`	
Funciones prohibidas: `Ninguna`	

`Account.hpp, tests.cpp y el fichero log están disponibles para ser descargados en la pagina de la intranet de este modulo.`

Es tu primer día trabajando para *GlobalBanksters United*. Has pasado todas las pruebas técnicas para el equipo de desarrollo gracias a unos trucos de *Microsoft Office* que un amigo te enseñó. También sabes que el reclutador flipó con lo rápido que instalaste *Adobe Reader*. Ese pequeño extra marcó la diferencia y que fuera de combate a tus adversarios. Lo has conseguido!

Tu superior te ha dado la primera tarea: Recuperar un archivo perdido. Algo salió mal y un archivo fuente fue borrado por accidente. Desafortunadamente, ninguno de tus compañeros sabe utilizar `Git` y en su lugar utilizan un USB para compartir código. Lo mejor sería abandonar el barco, pero a estas alturas, decides quedarte. ¡Ánimo valiente!

Tus compañeros te dan un montón de archivos. Al compilar el `tests.cpp` confirma que el archivo `Account.cpp` falta. Por suerte, la cabecera `Account.hpp` se salvó. También hay un archivo de log. Quizá puedas utilizar todo esto para reconstruir cómo estaba implementada la clase `Account`.

Empiezas a crear un archivo `Account.cpp` y a desplegar tus habilidades con unas maravillosas líneas de puro C++. Después de unos cuantos fallos de compilación, consigues pasar todos los test y replicar el archivo log, bueno, salvo por las marcas temporales que, lógicamente, eran anteriores a tu contratación. Has salvado el día.

¡Vaya, eres impresionante!

El orden en que se llaman los destructores puede diferir dependiendo de su compilador o sistema operativo. Entonces, tus destructores pueden ser llamados en orden inverso.

Puedes superar este modulo sin hacer el ejercicio 02.