**Capítulo I**

Introducción

C++ es un lenguaje de programación de propósito general creado por Bjarne Stroustrup como una extensión del lenguaje de programación C, o «C con Clases» (fuente: Wikipedia).

El objetivo de estos módulos es introducirte en la Programación Orientada a Objetos.

Este será el punto de partida de tu viaje en C++. Se recomiendan muchos lenguajes

para aprender POO. Decidimos elegir C++ ya que deriva de tu viejo amigo C.

Debido a que este es un lenguaje complejo, y con el fin de mantener las cosas simples, su código se

cumplirá con el estándar C++98.

Somos conscientes de que el C++ moderno es muy diferente en muchos aspectos. Así que si quieres

convertirte en un desarrollador C++ competente, depende de ti ir más allá después de los 42 Common

¡Core!

Descubrirás nuevos conceptos paso a paso. Los ejercicios aumentarán progresivamente

en complejidad.

**Capítulo II**

Reglas generales

Compilación

• Compila tu código con C++ y los indicadores -Wall -Wextra -Werror

• Tu código debería compilar incluso si añades el indicador -std=c++98

Formato y convenciones de nomenclatura

• Los directorios de ejercicios se nombrarán de la siguiente manera: ex00, ex01, ..., exn

• Nombra tus archivos, clases, funciones, funciones miembro y atributos según lo requerido en las directrices.

• Escribe los nombres de clase en formato Mayúsculas y minúsculas. Los archivos que contienen código de clase siempre se nombrarán según el nombre de la clase. Por ejemplo: ClassName.hpp/ClassName.h, ClassName.cpp o ClassName.tpp. Si tienes un archivo de encabezado que contiene la definición de la clase "BrickWall" que representa un muro de ladrillos, su nombre será BrickWall.hpp.

• A menos que se especifique lo contrario, todos los mensajes de salida deben terminar con un carácter de nueva línea y mostrarse en la salida estándar.

• ¡Adiós Norminette! No se impone ningún estilo de codificación en los módulos de C++. Puedes seguir tu estilo favorito. Pero recuerda que el código que tus compañeros evaluadores no pueden entender es código que no pueden calificar. Haz lo posible por escribir un código limpio y legible.

Permitido/Prohibido

Ya no estás programando en C. ¡Hora de C++! Por lo tanto:

• Puedes usar casi todo de la biblioteca estándar. Por lo tanto, en lugar de limitarte a lo que ya sabes, sería inteligente usar, en la medida de lo posible, las versiones similares a C++ de las funciones de C que usas.

• Sin embargo, no puedes usar ninguna otra biblioteca externa. Esto significa que C++11 (y sus derivados) y las bibliotecas Boost están prohibidas. Las siguientes funciones también están prohibidas: \*printf(), \*alloc() y free(). Si las usas, tu calificación será 0 y listo.

Tenga en cuenta que, a menos que se indique explícitamente lo contrario, está prohibido usar las palabras clave <ns\_name> y friend en el espacio de nombres. De lo contrario, su calificación será de -42.

Solo se permite usar la STL en los Módulos 08 y 09. Esto significa:

No se permiten contenedores (vector, lista, mapa, etc.) ni algoritmos (cualquier cosa que requiera incluir la cabecera <algorithm>) hasta entonces. De lo contrario, su calificación será de -42.

Algunos requisitos de diseño

Las fugas de memoria también ocurren en C++. Al asignar memoria (usando la palabra clave new), debe evitarlas.

Del Módulo 02 al Módulo 09, sus clases deben diseñarse en la forma canónica ortodoxa, a menos que se indique explícitamente lo contrario.

Cualquier implementación de función incluida en un archivo de cabecera (excepto las plantillas de función)

significa un 0 en el ejercicio.

Debería poder usar cada una de sus cabeceras independientemente de las demás. Por lo tanto, deben incluir todas las dependencias necesarias. Sin embargo, debes evitar el problema de la doble inclusión añadiendo protecciones de inclusión. De lo contrario, tu calificación será 0.

Léeme

• Puedes añadir archivos adicionales si lo necesitas (por ejemplo, para dividir tu código). Como estas tareas no son verificadas por un programa, puedes hacerlo siempre que entregues los archivos obligatorios.

• A veces, las pautas de un ejercicio parecen cortas, pero los ejemplos pueden mostrar requisitos que no están explícitamente escritos en las instrucciones.

• ¡Lee cada módulo completo antes de empezar! ¡En serio, hazlo!

• ¡Por Odín, por Thor! ¡Usa tu cerebro!

En cuanto al Makefile para proyectos de C++, se aplican las mismas reglas que en C (consulta el capítulo de Normas sobre el Makefile).

Tendrás que implementar muchas clases. Esto puede parecer tedioso, a menos que puedas programar tu editor de texto favorito.

Tienes cierta libertad para completar los ejercicios. Sin embargo, sigue las reglas obligatorias y no seas perezoso. ¡Te perderás mucha información útil! No dudes en leer sobre conceptos teóricos.

**Capítulo III**

Ejercicio 00: Megáfono

Ejercicio: 00

Megáfono

Directorio de entrega: ex00/

Archivos a entregar: Makefile, megaphone.cpp

Funciones prohibidas: Ninguna

Para asegurarte de que todos estén despiertos, escribe un programa que se comporte así:

$>./megaphone "shhhhh... Creo que los estudiantes están dormidos..."

SHHHHH... Creo que los estudiantes están dormidos...

$>./megaphone ¡Maldición! "Lo siento, estudiantes, pensé que esto estaba apagado."

¡Maldición! Lo siento, estudiantes, pensé que esto estaba apagado.

$>./megaphone

\* RUIDO DE RETROALIMENTACIÓN FUERTE E INSOPORTABLE \*

$>

Resuelve los ejercicios en C++.

**Capítulo IV**

Ejercicio 01: Mi Genial Agenda Telefónica

Ejercicio: 01

Mi Genial Agenda Telefónica

Directorio de entrega: ex01/

Archivos a entregar: Makefile, \*.cpp, \*.{h, hpp}

Funciones prohibidas: Ninguna

¡Bienvenidos a los 80 y a su increíble tecnología! Escribe un programa que se comporte como un software de agenda telefónica de baja calidad.

Debes implementar dos clases:

• Agenda Telefónica

◦ Contiene un array de contactos.

◦ Puede almacenar un máximo de 8 contactos. Si el usuario intenta añadir un noveno contacto, reemplaza el más antiguo por el nuevo.

◦ Ten en cuenta que la asignación dinámica está prohibida.

• Contacto

◦ Representa un contacto de la agenda telefónica.

En tu código, la agenda telefónica debe instanciarse como una instancia de la clase Agenda Telefónica. Lo mismo ocurre con los contactos. Cada una de ellas debe instanciarse como una instancia de la clase Contacto. Puedes diseñar las clases como quieras, pero recuerda que todo lo que se use siempre dentro de una clase es privado, y todo lo que se pueda usar fuera de una clase es público. No olvides ver los vídeos de la intranet.

Al iniciar el programa, la agenda telefónica está vacía y se le solicita al usuario que introduzca uno de tres comandos. El programa solo acepta AGREGAR, BUSCAR y SALIR.

• AGREGAR: guardar un nuevo contacto

◦ Si el usuario introduce este comando, se le solicita que introduzca la información del nuevo contacto campo por campo. Una vez completados todos los campos, añada el contacto a la agenda telefónica.

◦ Los campos del contacto son: nombre, apellido, apodo, número de teléfono y secreto más oscuro. Un contacto guardado no puede tener campos vacíos.

• BUSCAR: mostrar un contacto específico

◦ Mostrar los contactos guardados como una lista de 4 columnas: índice, nombre, apellido y apodo.

◦ Cada columna debe tener 10 caracteres de ancho. Se separan con una barra vertical ('|'). El texto debe estar alineado a la derecha. Si el texto es más largo que la columna, debe truncarse y el último carácter visible debe reemplazarse por un punto (’.’).

◦ Luego, vuelva a solicitar al usuario el índice de la entrada que se mostrará. Si el índice está fuera de rango o es incorrecto, defina un comportamiento relevante. De lo contrario, muestre la información de contacto, un campo por línea.

• SALIR

◦ ¡El programa se cierra y los contactos se pierden para siempre!

• Cualquier otra entrada se descarta.

Una vez ejecutado correctamente un comando, el programa espera otro. Se detiene cuando el usuario ingresa SALIR.

Asigne un nombre relevante a su ejecutable.

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

<http://www.cplusplus.com/reference/iomanip/>

**Capítulo V**

Ejercicio 02: El trabajo de tus sueños

Ejercicio: 02

El trabajo de tus sueños

Directorio de entrega: ex02/

Archivos a entregar: Makefile, Account.cpp, Account.hpp, tests.cpp

Funciones prohibidas: Ninguna

Account.hpp, tests.cpp y el archivo de registro están disponibles para descargar en la página de intranet del módulo.

Hoy es tu primer día en GlobalBanksters United. Tras aprobar las pruebas de selección (gracias a algunos trucos de Microsoft Office que te enseñó un amigo), te uniste al equipo de desarrollo. También sabes que el reclutador quedó impresionado con la rapidez con la que instalaste Adobe Reader. Ese pequeño extra marcó la diferencia y te ayudó a vencer a todos tus oponentes (es decir, a los demás solicitantes): ¡lo lograste!

En fin, tu gerente te acaba de dar trabajo. Tu primera tarea es recrear un archivo perdido. Algo salió mal y se eliminó un archivo fuente por error. Desafortunadamente, tus compañeros no saben qué es Git y usan memorias USB para compartir código. En este punto, tendría sentido irte de inmediato. Sin embargo, decides quedarte. ¡Reto aceptado!

Tus compañeros desarrolladores te dan un montón de archivos. Al compilar tests.cpp, descubres que el archivo que falta es Account.cpp. Por suerte, el archivo de encabezado Account.hpp se guardó.

También hay un archivo de registro. Quizás podrías usarlo para entender cómo se implementó la clase Account.

Empiezas a recrear el archivo Account.cpp. En solo unos minutos, escribes unas pocas líneas de C++ puro y genial. Tras un par de compilaciones fallidas, tu programa pasa las pruebas. Su salida coincide perfectamente con la guardada en el archivo de registro (excepto por las marcas de tiempo, que obviamente serán diferentes, ya que las pruebas guardadas en el archivo de registro se ejecutaron antes de que te contrataran).

¡Caramba, eres impresionante!

El orden en que se llaman los destructores puede variar según tu compilador o sistema operativo. Por lo tanto, tus destructores pueden llamarse en orden inverso. Puedes aprobar este módulo sin realizar el ejercicio 02.