Programación Orientada a Objetos
Práctica 8. Curso 2024-25

1.- Sobrecarga de operadores. La clase Contador gestiona un entero (valor_) cuyo valor puede estar entre un valor mínimo (min_) y un valor máximo (max_) que son pasados en el constructor. Si cualquier operación sobre el contador intentara llevar su valor por encima del límite superior, el contador toma el valor del límite superior. Si cualquier operación sobre el contador intentara llevar su valor por debajo del límite inferior, el contador toma el valor del límite inferior. Implementar las siguientes funciones de la clase *Contador* en los ficheros `contador.h y contador.cc`:

a) Constructor con 3 parámetros: valor inicial del contador (valor por defecto=0), valor mínimo del rango de valores (valor por defecto=0), y valor máximo del rango de valores (valor por defecto=1000). Si sólo se proporciona un parámetro, (el valor del contador), éste deberá estar entre el rango por defecto, y en caso contrario el valor del contador se pondrá a 0. Si se produce alguna otra condición de error (valor fuera de rango o límite mínimo pasado como parámetro mayor que el máximo) el contador oscilará entre 0 y 1000, y se pondrá un valor inicial al contador igual a 0.
b) Observador get() que devuelve el valor actual del contador.
c) Operador de asignación (=). Este operador debe tener 2 versiones sobrecargadas. Una que permita asignar al valor del contador un entero, y otra que permita asignar a un contador otro contador.
d) Operadores ++ y -- que incrementan y decrementan el valor del contador en 1. Tanto prefijo (++c) como sufijo (c++).
e) El operador + devuelve un contador cuyo valor es la suma entre un objeto de tipo Contador y un entero. Ejemplo: c + 10 (el valor máximo devuelto por la suma debe ser el que se estableció en el constructor)
f) El operador + que devuelve un contador con la suma entre un entero y un objeto de tipo Contador. Ejemplo: 10 + c (el valor máximo devuelto por la suma debe ser el que se estableció en el constructor)
g) Operador – que será análogo al anterior. Ejemplo: c – 10 (el valor mínimo devuelto por la suma debe ser el que se estableció en el constructor)
h) El operador - que será análogo al anterior. Ejemplo: 10 – c (el valor mínimo devuelto por la suma debe ser el que se estableció en el constructor)
i) Insertador y extractor propio. El insertador debe mostrar el valor del contador en pantalla. El extractor debe pedir el valor del contador por teclado y un mensaje indicando si el valor es correcto o no, en cuyo caso lo volverá a pedir hasta que sea correcto.

Compilar `contador.h` y `contador.cc` con el `Makefile` proporcionado, y realizar también los tests del fichero proporcionado: `contador-unittest.cc`

IMPORTANTE: Necesitamos de nuevo Google Test. Como ya lo tenemos de las prácticas anteriores en cualquiera de ellas en el directorio:

`marketplace/build/_deps/googletest-src/googletest`

lo que vamos a hacer es tomarlo de ahí.

Descarga los ficheros `Makefile` y `contador-unitest.cc` proporcionados al mismo directorio donde vas a hacer esta práctica.

Edita y busca en el interior de dicho `Makefile` la variable GTEST_DIR y la igualas al *path* donde tengas la carpeta `googletest` dentro de `marketplace/build`

Es decir, por ejemplo tienes que cambiar GTEST_DIR y que quede así:

```
GTEST_DIR = /home/i8u7lm/poo/p7/marketplace/build/_deps/googletest-src/googletest
```

cambiando el usuario 'i8u7lm' por tu usuario, y el resto del path por el path que lleve a ese directorio en una de tus prácticas anteriores, por ejemplo la 'p7'.

2.- Ordenación. Ir al enlace "Beginners guide to the std::sort() function" que hay bajo el enunciado de esta práctica en la web de la asignatura y entender como se ordena un vector de la STL, los #includes que hay que utilizar, la forma en que hay que invocar a la función sort(), etc. Después, realizar los siguientes ejercicios:

a) Hacer un programa que pida los elementos de un vector de enteros al usuario por teclado, los ordene y los muestre ordenados en pantalla (`ordena.cc`).
b) Investiga como hacer el mismo programa del punto anterior pero en orden descendente. Realiza un programa que solicite al usuario el orden en el que quiere ordenar el vector: ascendente o descendente (ordena2.cc).
c) Idem para un vector de reales (`ordena3.cc`).

3.- Plantillas / Templates. Repasar los apuntes y la presentación que hemos visto en clases de teoría sobre plantillas de función y plantillas de clase. Después, realizar los siguientes ejercicios:

a) Implementa en C++ un ejemplo usando la plantilla de función de los apuntes de C++ que se ha visto en clase de teoría. La función debe recibir un vector y su tamaño, y escribir su contenido en pantalla sea el vector de int, float o char. (`plantilla.cc`)
b) Implementa en C++ un ejemplo usando la plantilla de clase de los apuntes de C++ que se ha visto en clase de teoría. La plantilla **MiClase** que realiza una división de sus datos privados sean int, float, etc. (`plantilla2.cc`)