Programación Orientada a Objetos.
Práctica 4. Curso 202-21

Sobre el código que has hecho hasta la fecha realiza los siguientes ejercicios:

1. Revisa las guardas de inclusión de todos tus archivos de cabecera para comprobar que están correctamente (a veces las copiamos y pegamos en otros archivos y podemos cometer errores) cada una debe tener el nombre de su archivo en mayúsculas + "_" + "H".
2. Añade a tu código comentarios antes de cada clase, método o función con una **breve y concisa** descripción. Usa comentarios tanto /* */ como //. El código fuente debe estar bien autodocumentado.
3. Revisa las funciones en línea que has usado. Revisa si debes pasar alguna función en línea a función normal o al revés. Importante: ¡usa siempre el mismo criterio!
4. Revisa las veces que has usado `const` en tu código, por ejemplo en la declaración de los observadores de las clases. ¿Has usado const en algún sitio más?. Recuerda que es conveniente pasar siempre los objetos como parámetros mediante referencias constantes, ¿recuerdas por qué? (esto es muy importante que se entienda bien). Las funciones que sean observadores deben ser const.
5. Añade el método `getEstadoRuleta()` a la clase ruleta que reciba como parámetro las **referencias** de datos en los que guardarás el número de jugadores que hay en un momento dado en la ruleta, la suma del dinero encima de la mesa (el de los jugadores más el de la banca), las veces que se ha lanzado la bola y lo que lleva ganado la banca (negativo si pierde). Recuerda: usa referencias.
6. Termina el programa principal en el que pruebas la ruleta. El programa debe ofrecer un menú con las siguientes opciones hasta que el usuario decida salir:
   a) Cargar los jugadores del fichero jugadores.txt
   b) Guardar los jugadores en jugadores.txt
   c) Ver el estado de la ruleta, el dinero de la banca y el de los jugadores.
   d) Hacer girar la ruleta mostrando el número que ha salido, los premios de cada jugador y lo que gana/pierde la banca
   e) Eliminar un jugador de la mesa
   f) Añadir un jugador a la mesa
   g) Salir del programa
   h) Si quisieras dotar al programa de una IGU (Interfaz Gráfica de Usuario) ¿por dónde comenzarías? ¿conoces alguna librería/biblioteca para el desarrollo de IGUs? Investiga este asunto y comparte con tu profesor y tus compañer@s.
7. **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 intentará 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.

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 esta 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) 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.
c) Operadores ++ y -- que incrementan y decrementan el valor del contador en 1. Tanto prefijo (++c) como sufijo (c++).
d) 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)
e) 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)
f) 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)
g) 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)
h) El método `undo(int n)` deshace las `n` últimas operaciones que se le han aplicado al contador. Por defecto `n` es igual a 1. Si `n` está fuera de rango devuelve false y no deshace nada (si está dentro del rango devuelve true).
i) Observador `get()` que devuelve el valor actual del contador.
j) Realiza también pruebas unitarias para la clase **Contador** en el fichero `contador_unittest.cc`

8. ¿Has usado sobrecarga de funciones en tu código a lo largo del curso?, ¿dónde?. ¿Y sobrecarga de operadores?, ¿dónde? ¿tienen utilidad? Comenta con tu profesor y compañer@s.
9. ¿Hay algún dato público en tus clases?, ¿por qué?.
10. ¿Has usado iniciadores base?¿dónde y por qué?. Cambia tu código si es necesario para usarlos. Suelen usarse bastante, son más cómodos ¿no te lo parece?. Comenta con tu profesor y compañer@s.
11. ¿Qué clases de las que has hecho son clases base y cuáles son clases derivadas? Las clases derivadas reutilizan las clases base “extendiéndolas” ¿estás de acuerdo?

12. Revisa los `Makefile` que has usado y analiza su contenido. No olvides nunca cómo se hacen los Makefiles, te servirán mucho en el futuro.
13. Revisa el fichero `ruleta_unittest.cc` y analiza su contenido. Termina de poner a punto todos los test de tu código. Recuerda que en el examen práctico se te proporcionará uno o varios tests y tendrás que hacer el código necesario para pasar dichos tests. Debes familiarizarte bien con ellos.
14. Comprueba si tu código pasa los tests proporcionados por el profesor.

El entregable de esta práctica es solamente la clase Contador. El resto de ejercicios también hay que hacerlos pero son o bien ejercicios teóricos de repaso, o bien para mejorar las prácticas anteriores.