Programación Orientada a Objetos. Curso 2014-2015
Práctica 1

1.- Hacer el típico programa "hello world" en C++. Usar el objeto de la salida estándar "cout", para el que tendrás que incluir la correspondiente cabecera

```
#include <iostream>
```

y tendrás que indicar que vas a usar el espacio de nombres std:

```
using namespace std;
```

Recuerda a partir de ahora incluir en cada fichero de código fuente que hagas una cabecera del tipo:

```
// holamundo.cc
// A program that prints the inmortal saying "hello world"
```

Escribe todo el código en el mismo fichero fuente (holamundo.cc) que contendrá la función main() y compila el programa con g++.

2.- Hacer un programa que genere un número aleatorio entre 1 y 10, y solicite al usuario un número para posteriormente adivinarlo indicando al usuario si el número generado es menor, mayor o es correcto. Usar para la entrada el objeto de entrada estandar "cin" para lo que tendrás que incluir lo mismo que con "cout" del ejercicio anterior.
Recuerda que en la generación de números aleatorios la semilla se puede establecer mediante:

```
srand(time(NULL))
```

y que tendrás que incluir las cabeceras cstdlib y ctime para srand() y time() respectivamente:

```
#include <cstdlib>
#include <ctime>
```

A partir de ahora usar siempre los objetos cin y cout para entrada/salida teclado/pantalla.

3.- (Lee todo el enunciado del ejercicio antes de empezar) La clase *Dados* representa el lanzamiento de dos dados. Tal y como hemos visto en la clase de teoría, declarar la clase *Dados* en el fichero dados.h y el cuerpo de dicha clase en el fichero dados.cc
Operaciones de la clase *Dados*:

- lanzamiento: obtiene un nuevo valor aleatorio para los dos dados
- getDado1: devuelve el valor del primer dado
- getDado2: devuelve el valor del segundo dado
- setDado1: asigna un valor (que se pasa como parámetro) al primer dado, y devuelve true si la operación es correcta, y false si el valor pasado no está entre 1 y 6
- setDado2: igual a la anterior para el segundo dado
- suma: devuelve el valor de la suma de los dos dados
- Un constructor que inicie la semilla de la generación de números aleatorios y de un valor inicial a los dados igual a 1. Un constructor es un método público de la clase que se llama exactamente igual que la clase, que no devuelve nada (ni siquiera void) y que se ejecuta automáticamente al declarar un objeto de esa clase.

Crear un fichero juego.cc con la función main() que declare un objeto de la clase *Dados*, y que permita invocar todas y cada una de las operaciones de la clase.

Analizar todos los #includes necesarios para el programa y **usar guardas de inclusión siempre** en nuestros ficheros .h

Usar la guía de estilo que se ha dejado en el moodle de la asignatura para nombrar, clases, funciones, variables, etc. (Google C++ Style Guide).

Intentar acceder directamente a los datos privados de la clase *Dados* desde la función main(). Por ejemplo añadiendo al final de la función main un cout que saque en pantalla el valor del primer dado mediante el objeto de tipo Dados, el operador punto y el nombre del dato privado. Compilar el programa y analizar el error que indica el compilador.

4.- Introducción a las pruebas unitarias (**UNIT TESTING**). Introducción al desarrollo mediante la elaboración de tests o desarrollo guiado por tests o Test-Driven Development (TDD):.

Comenta la siguiente figura:

Refactoring: "*Code refactoring is the process of restructuring existing computer code – changing the factoring – without changing its external behavior. Refactoring improves nonfunctional attributes of the software. Advantages include improved code readability and reduced complexity to improve source code maintainability, and create a more expressive internal architecture or object model to improve extensibility."* (http://en.wikipedia.org/wiki/Code_refactoring)

Nosotros vamos a usar **Google Test**, un framework considerado de tipo **xUnit.**
xUnit es una familia de frameworks para pruebas unitarias (unit testing) que comparten características comunes que veremos en esta práctica.

Ejercicio:

a) En el moodle de la asignatura hay un enlace a googletest. Descargar de la sección "Downloads" el arhivo gtest-1.7.0.zip y descomprimir en vuestro home (se habrá creado el directorio gtest-1.7.0)
b) Descargar del moodle de la asignatura el archivo dados_unittest.zip y descomprimirlo en el directorio donde tenéis la clase *Dados*
c) Adaptar el Makefile a vuestro proyecto y hacer:

```
make
```

d) Abrir el fichero dados_unittest.cc y analizar cada uno de los test que se han incluido en él. Para ello se puede consultar el ejemplo en la documentación de googletest a través del enlace proporcionado a su documentación.
e) Ejecutar los tests sobre la clase *Dados* mediante el fichero ejecutable creado con el makefile, ejecutando:

```
./dados_unittest
```

f) Analizar cada línea de la salida de la ejecución de los test.
g) Vamos a utilizar este tipo de test a lo largo de todo el curso. Para familiarizarnos mejor con

ellos podemos introducir algún error en cada una de las operaciones de la clase dados y ejecutar los test. Ir corrigiendo cada error, uno a uno y ejecutando los tests en cada corrección, hasta que se corrijan todos los errores.

h) Añade un test para la clase *Dados* al fichero dados_unittest.cc para que compruebe el buen funcionamiento de una operación que se llame "diferencia" y que devuelva un entero con la distancia entre el menor y el mayor valor de los almacenados en los dados.
i) Después de añadir el test escribe el código de dicha operación dentro de la clase de forma que pasen los test que has creado para ella.
j) Por último limpia y optimiza el código de toda la clase Dados para que quede al máximo nivel de calidad, autodocumentación, etc. Y vuelve a ejecutar los test para confirmar que todo ha quedado correcto.
k) Actualiza el fichero juego.cc del apartado anterior para que incluya la ejecución de la operación "diferencia".

5.- En unit testing, el *test runner* es el programa que ejecuta los diferentes tests diseñados y muestra los resultados (en este caso el programa `dados_unittest.cc`). Añade los siguientes test unidad a dicho fichero para posteriormente modificar la clase *Dados* para que los cumpla:

a) Cada vez que se invoque al método setDado1() o setDado2() se considerará realizado un nuevo lanzamiento. El método getLanzamientos1() devolverá el número de veces que se ha realizado el lanzamiento del dado 1. El método getLanzamientos2() será igual para el dado 2. Añade un test que compruebe que después de declarar un objeto estos métodos deben devolver 0. Y que después de un número determinado de lanzamientos *n*, el método debe devolver *n*.
b) Los métodos getMedia1() y getMedia2() devolverán la media de los valores que van saliendo en el dado 1 y 2 respectivamente. Los valores iniciales (igual a 1) de los dados que se fijan en el constructor no deben ser considerados para la media. Si se pide la media antes del primer lanzamiento, ésta debe ser cero (añadir un test que compruebe esto).
c) Si se usa setDado1() o setDado2() el valor asignado debe ser considerado para la media (añadir un test que compruebe esto).
d) El método getUltimos1() recibe un vector de enteros y le asigna los 5 últimos valores obtenidos para el primer dado. El método getUltimos2() se comporta de forma análoga para el segundo dado. Modifica la clase *Dados* para que pueda ir almacenando los 5 últimos lanzamientos de esta forma y añade los tests que comprueben su funcionamiento correcto.

6.- Echa un vistazo y estudia todo el material y todos los enlaces que se han dejado en el moodle de la asignatura correspondientes a esta práctica. Verás que también hay una sección con material adicional y enlaces de interés en el moodle de la asignatura que será interesante que también revises de vez en cuando.