Programación Orientada a Objetos. Curso 2022-23
Práctica 1. Primer contacto con C++

NOTA: en tu cuenta de la UCO, crea una carpeta poo para las prácticas de esta asignatura, y mete todos los ejercicios de esta práctica 1 en el subdirectorio poo/p1.

1.- Comencemos con el típico programa "hello world" en C++ que escribiremos en el fichero:

`hello-world.cc`

En C++, para escribir en pantalla la cadena "hello world" deberás usar el objeto:

`cout`

Para usar el objeto `cout` tendrás que incluir la correspondiente cabecera:

```
#include <iostream>  // fíjate que no termina en .h
        // Doble barra es un comentario hasta final de línea
```

Como el objeto `cout` está dentro del espacio de nombres `std` debes utilizarlo con el operador `::` (*scope resolution operator*) así:

`std::cout`

Y para enviarle la cadena a visualizar en pantalla se utiliza el operador de inserción (*insertion operator*):

`<<`

Por tanto la línea quedará:

```
std::cout << "hello world\n";
```

En el ejercicio 2 entenderás mejor el concepto de espacio de nombres.

Para *autodocumentar* mejor tus ficheros de código, recuerda a partir de ahora incluir al inicio de cada fichero una cabecera de 2 líneas con el nombre del archivo y un breve comentario descriptivo (mejor si es en inglés). Por ejemplo, en este caso sería:

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

Completa el código en el fichero `hello-world.cc` que contendrá la función `main()` y compila el programa con `g++` corrigiendo posibles errores.

NOTA: la extensión habitual para archivos con código C++ es ".cc", pero hay otras como ".C", ".cpp", ".++", ".c++", ".cxx" y otras. **Nosotros usaremos la extensión ".cc" para el código y ".h" para las cabeceras.**

2.- Los espacios de nombres (namespaces) en C++.

Un espacio de nombres (namespace) es un bloque de código entre llaves con definiciones en su interior. Estas definiciones quedan asociadas al nombre de dicho *namespace*. Dos espacios de nombres diferentes pueden definir el mismo identificador sin que exista conflicto. Como en este ejemplo:

```
namespace n1{
int a; // Esta es la variable n1::a
```

```
int b; // Esta es la variable n1::b
}

namespace n2{
float a;  // Esta es la variable n2::a
float c;  // Esta es la variable n2::c
}
```

Así, la variable `n1::a` será diferente a la variable `n2::a`, y podemos hacer:

```
#include <iostream>
int main(void)
{
int a=55;
n1::a=0;
n2::a=2.3;
std::cout<< “n1::a= “ << n1::a << “\n”;
std::cout<< “n2::a= ” << n2::a << “\n”;
std::cout<< “a= ” << a << “\n”;
}
```

De esta forma, podemos crear un espacio de nombres sin preocuparnos de que otro programador utilice los mismos identificadores que nosotros. Mientras los nombres de los propios espacios de nombres sean diferentes, no habrá problema. Ten en cuenta que esto es muy útil en grandes proyectos software.

Como ejemplo escribe un programa que se llame `ns.cc` que declare los dos espacios de nombres anteriores y que en su función `main()` ejecute el código del ejemplo anterior.

Reflexiona y comenta con tus compañeros/as este código. Debes entender bien el concepto de espacio de nombres, y que

```
n1::a
n2::a
  a
```

son tres variables con el mismo nombre ‘a’, pero diferentes y perfectamente usables sin conflicto gracias a los espacios de nombres.

“std” es un gran espacio de nombres ya definido internamente por C++ y donde se encuentran montones de objetos y clases que usaremos en nuestros programas.

Una vez hayas compilado y ejecutado el ejemplo anterior cambia el programa `ns.cc` y añade (en la siguiente línea a *#include <iostream>*) la siguiente instrucción para utilizar `std` como namespace por defecto:

```
using namespace std;
```

Cambia la función `main()` y quita ‘std::’ de todas las apariciones de ‘std::cout’ y deja únicamente ‘cout’.

Observa que el programa compila y no es necesario poner `std::` delante del objeto `cout` (aunque siempre es preferible usar la forma larga `std::cout`).

3.- Hacer un programa en C++ (`guess.cc`) que genere un número aleatorio entre 1 y 10, y solicite al usuario un número por teclado para posteriormente adivinarlo indicando al usuario cada vez si el número generado es menor, mayor o es correcto.

Usar para la entrada por teclado el objeto de entrada estandar

`cin`

para lo que tendrás que incluir la misma cabecera que con `cout` del ejercicio anterior y usar el operador de extracción (*extraction operator*):

`>>`

En el siguiente código leemos una variable entera desde el teclado y la almacenamos en “i”:

```
std::cout << “Introduce un número: ”;
std::cin >> i;
```

En C, la función rand() de cstdlib genera un número aleatorio entre 0 y el valor RAND_MAX (valor máximo que depende de cada compilador/implementación). No es exactamente aleatorio al 100% ya que utiliza un valor de partida (la semilla) para generar el número aleatorio, de ahí que en realidad se llame número pseudoaleatorio.
Si usamos rand() sin más, la semilla es siempre 1 y los valores que van saliendo en sucesivas llamadas a rand() son siempre la misma secuencia.
Un truco que se suele utilizar para que dicha secuencia de números pseudoaleatorios sea diferente cada vez que ejecutemos el programa es poner una semilla distinta al principio del programa. Esto se hace con la función srand() y el truco es usar como semilla el número de segundos del reloj del sistema. La función time(NULL) nos devuelve un entero con el número de segundos que han transcurrido desde el 1 de enero de 1970 y como cambia cada segundo, el valor de la semilla irá cambiando cada vez que ejecutemos el programa, por ello ponemos al inicio del programa:

```
srand(time(NULL)); // time(NULL) returns “UNIX Epoch”, this is seconds
                   // since 00:00:00 1 January 1970
```

IMPORTANTE: la semilla se fija solo una vez y al principio del programa por tanto, solo se pone una única vez srand(time(NULL)) al principio del programa.

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

```
#include <cstdlib> // Fíjate que no llevan la terminación .h
#include <ctime>   // Fíjate que no llevan la terminación .h
```

Fíjate en que son cabeceras también usadas en C, que ahora en C++ se usan anteponiendo una “c” delante y sin la terminación “.h”.

A partir de ahora usa siempre los objetos `cin` y `cout` en tus programas para entrada/salida teclado/pantalla.

Los objetos `cin` y `cout` son técnicamente “*streams*” o “flujos de datos”. Un *stream* es una secuenciación de datos, en este caso de caracteres. El stream `cin` está asociado a la entrada (el teclado) y `cout` a la salida (la pantalla).