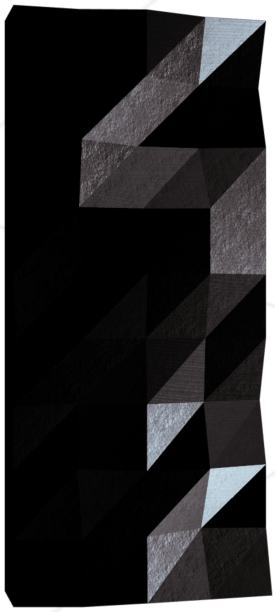

C++ - Módulo 07

c++ Templates

Resumen: Ejercicios del Módulo 07 de C++.

Índice general

Capítulo I

Instrucciones generales

Para los módulos de C++ utilizarás y aprenderás exclusivamente C++98. Tu objetivo es aprender las nociones de la programación orientada a objetos. Sabemos que las últimas versiones de C++ son muy diferentes en muchos aspectos, si quieres volverte un experto en C++ deberás aprender C++ moderno más adelante. Este es el principio de tu largo viaje por C++, es cosa tuya ir más allá después del common core.

- Cualquier función implementada en un header (excepto en el caso de templates), y cualquier header desprotegido, significa un 0 en el ejercicio.
- Todos los output deben ir al standard output, y deben terminar con un salto de línea, salvo que se indique lo contrario.
- Se deben seguir los nombres de archivos impuestos al pie de la letra, así como las clases, funciones y métodos.
- Recuerda: estás programando en `C++`, no en `C`. Por lo tanto:
    - Las siguientes funciones están PROHIBIDAS, y su uso será sancionado con un `0`, sin preguntas: `*alloc`, `*printf` y `free`.
    - Tienes permitido utilizar básicamente todo de la librería estándar. SIN EMBARGO, sería inteligente utilizar la versión de C++ de las funciones a las que estás acostumbrado en C, en lugar de simplemente seguir utilizando lo que sabes... En realidad, estás aprendiendo un lenguaje nuevo. Y NO, no tienes permitido utilizar STL hasta que debas hacerlo (es decir, el módulo 08). Esto significa que nada de vectors/ lists/maps/etc. O nada que requiera un "include <algorithm>" hasta entonces.
- De hecho, el uso de cualquier función o mecánica explícitamente prohibida será recompensada con un `0`, sin preguntas.
- Ten en cuenta que, salvo especificado de otro modo, las palabras de `C++ üsing namespace"` y `"friend"` están terminantemente prohibidas. Su uso será sancionado con `-42`, sin preguntas.
- Los archivos asociados con una clase se llamarán siempre `ClassName.hpp` y `ClassName.cpp`, salvo especificado de otro modo.
- Entrega en directorios denominados `ex00/`, `ex01/` ... `exn/`.

- Debes leer los ejemplos con cuidado. Pueden contener requisitos no tan obvios en la descripción del ejercicio.
- Dado que tienes permitido utilizar herramientas de `C++` que llevas aprendiendo desde el principio, no tienes permitido utilizar librerías externas. Y antes de que te lo preguntes, esto significa que ningún derivado de `C++11`, ni `Boost` o similares se permite.
- Puede que se requiera entregar un importante número de clases. Esto puede parecer tedioso, salvo que sepas instalar scripts en tu editor de texto favorito.
- Lee cuidadosamente los ejercicios POR COMPLETO antes de empezarlos. En serio, hazlo.
- El compilador a usar es `clang++`.
- Tu código debe compilar con las flags: `-Wall -Werror -Wextra`.
- Cada uno de tus `include` debe poder incluirse independientemente del resto. Los `include` deben contener obviamente los `include` de los que dependan.
- Por si te lo preguntas, no se requiere ningún estilo de código durante `C++`. Puedes utilizar una guía de estilos que te guste, sin limitaciones. Recuerda que si tu evaluador no es capaz de leer tu código, tampoco lo será de evaluarte.
- Algo importante: NO te evaluará un programa, salvo que el subject lo indique explícitamente. Por lo tanto, tienes cierta libertad en cómo hagas los ejercicios. Sin embargo, sé inteligente con los principios de cada ejercicio, y NO seas perezoso, te perderás MUCHO de lo que estos proyectos te pueden ofrecer.
- No es un problema real si tienes archivos adicionales a los que se te solicita, puedes elegir separar el código en más archivos de los que se te piden. Siéntete libre, siempre y cuando el resultado no lo evalúe un programa.
- Aunque el subject de un ejercicio sea corto, merece la pena gastar algo de tiempo para estar absolutamente seguro de que entiendes lo que se espera que entiendas, y que lo has hecho de la mejor forma posible.
- Por Odin, por Thor. Utiliza tu cerebro.

Capítulo II

Ejercicio 00: Algunas funciones

	Ejercicio: 00
Ejercicio 00: Algunas funciones	
Directorio de entrega: *ex*00/	
Archivos a entregar: `whatever.hpp, main.cpp y un Makefile`	
Funciones autorizadas: `Ninguna`	

Escribe las siguientes function templates:

- `swap`: intercambia los valores de dos argumentos. No devuelve nada.
- `min`: compara los dos argumentos y devuelve el más pequeño. En caso de que ambos sean iguales, devuelve el segundo.
- `max`: compara los dos argumentos y devuelve el más grande. En caso de que ambos sean iguales, devuelve el segundo.

Las templates deben estar definidas en headers. Debes entregar en el `main.cpp` las pruebas para tu programa. El archivo puede y será modificado durante tu evaluación. Estas funciones pueden utilizarse con cualquier tipo de argumento, siempre y cuando ambos tengan el mismo tipo y permitan utilizar todos los operadores de comparación. Utiliza el código suficiente como para demostrar que todo funciona como debe.

El siguiente código:

```
int     main( void ) {

    int a = 2;
    int b = 3;

    ::swap( a, b );
    std::cout << "a = " << a << ", b = " << b << std::endl;
    std::cout << "min( a, b ) = " << ::min( a, b ) << std::endl;
    std::cout << "max( a, b ) = " << ::max( a, b ) << std::endl;

    std::string c = "chaine1";
    std::string d = "chaine2";

    ::swap(c, d);
    std::cout << "c = " << c << ", d = " << d << std::endl;
    std::cout << "min( c, d ) = " << ::min( c, d ) << std::endl;
    std::cout << "max( c, d ) = " << ::max( c, d ) << std::endl;

    return 0;
}
```

Deberá imprimir lo siguiente en caso de que esté bien hecho:

```
a = 3, b = 2
min(a, b) = 2
max(a, b) = 3
c = chaine2, d = chaine1
min(c, d) = chaine1
max(c, d) = chaine2
```

Capítulo III

Ejercicio 01: Iter

	Ejercicio: 01
Ejercicio 01: Iter	
Directorio de entrega: $ex01/$	
Archivos a entregar: `iter.hpp, main.cpp y un Makefile`	
Funciones autorizadas: `Ninguna`	

Escribe una function template `iter` que acepte 3 parámetros y no devuelva nada. El primer parámetro es la dirección de un array, el segundo es la longitud del propio array y el tercero una función ejecutada para cada elemento del array.

Crea un programa que verifique el correcto funcionamiento de tu function template `iter` para cualquier tipo de array y/o con una function template instanciada como tercer parámetro.

Capítulo IV

Ejercicio 02: Array

	Ejercicio: 02
Ejercicio 02: Array	
Directorio de entrega: *ex*02/	
Archivos a entregar: `Array.hpp, main.cpp y un Makefile`	
Funciones autorizadas: `Ninguna`	

Escribe una class template `Array` que contenga elementos de tipo `T` y cumpla con el siguiente comportamiento:

- Construcción sin parámetros: crea un array vacío.
- Construcción con un `unsigned int n` como parámetro: crea un array de `n` elementos, iniciado por defecto. Pista: intenta compilar con `int *a = new int();`, luego muestra `*a`.
- Construcción por copia y por operador de asignación: en ambos casos, modificar uno de los dos arrays después de copiar/asignar no afectará al otro.
- TIENES que utilizar el operador `new[]` para la reserva de memoria. No debes reservar más o menos memoria de la necesaria. Tu código nunca debe acceder a memoria no reservada.
- Los elementos son accesibles utilizando el `operator[]`.
- Cuando accedas a un elemento utilizando `operator[]`, en caso de que esté fuera de límites una excepción `std::exception` debe lanzarse.
- Debe existir una member function `size` que devuelva el número de elementos en el array. Esta función no debe aceptar parámetros ni debe modificar la instancia de ninguna forma.

Crea un programa que permita validar cada uno de los puntos para verificar que tu class template funciona como se espera.