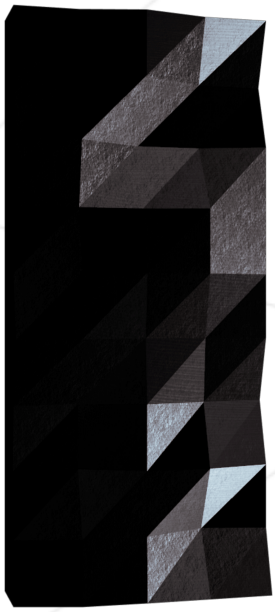

C++ - Módulo 07

C++ Plantillas

Resumen: Este documento contiene la evaluación del módulo 07 de los módulos C++ de 42.

Índice general

Capítulo I

Reglas Generales

- La declaración de una función en un header (excepto para los templates) o la inclusión de un header no protegido conllevará un 0 en el ejercicio.
- Salvo que se indique lo contrario, cualquier salida se mostrará en stdout y terminará con un newline.
- Los nombres de ficheros impuestos deben seguirse escrupulosamente, así como los nombres de clase, de función y de método.
- Recordatorio : ahora está codificando en `C++`, no en `C`. Por eso :
  - Las funciones siguientes están **PROHIBIDAS**, y su uso conllevará un 0: `*alloc`, `*printf` et `free`
  - Puede utilizar prácticamente toda la librería estándar. NO OBSTANTE, sería más inteligente intentar usar la versión para C++ que a lo que ya está acostumbrado en C, para no basarse en lo que ya ha asimilado. Y no está autorizado a utilizar la STL hasta que le llegue el momento de trabajar con ella (módulo 08). Eso significa que hasta entonces no se puede utilizar Vector/List/Map/etc... ni nada similar que requiera un include <algorithm>.
- El uso de una función o de una mecánica explícitamente prohibida será sancionado con un `0`
- Tenga también en cuenta que, a menos que se autorice de manera expresa, las palabras clave `using namespace` y `friend` están prohibidas. Su uso será castigado con un `0`.
- Los ficheros asociados a una clase se llamarán siempre `ClassName.cpp` y `ClassName.hpp`, a menos que se indique otra cosa.
- Tiene que leer los ejemplos en detalle. Pueden contener prerrequisitos no indicados en las instrucciones.
- No está permitido el uso de librerías externas, de las que forman parte `C++11`, `Boost`, ni ninguna de las herramientas que ese amigo suyo que es un figura le ha recomendado.
- Probablemente tenga que entregar muchos ficheros de clase, lo que le va a parecer repetitivo hasta que aprenda a hacer un script con su editor de código favorito.

- Lea cada ejercicio en su totalidad antes de empezar a resolverlo.
- El compilador es `clang++`
- Se compilará su código con los flags `-Wall -Wextra -Werror`
- Se debe poder incluir cada include con independencia de los demás include. Por lo tanto, un include debe incluir todas sus dependencias.
- No está obligado a respetar ninguna norma en `C++`. Puede utilizar el estilo que prefiera. Ahora bien, un código ilegible es un código que no se puede calificar.
- Importante: no va a ser calificado por un programa (a menos que el enunciado especifique lo contrario). Eso quiere decir que dispone cierto grado de libertad en el método que elija para resolver sus ejercicios.
- Tenga cuidado con las obligaciones, y no sea zángano; podría dejar escapar mucho de lo que los ejercicios le ofrecen.
- Si tiene ficheros adicionales, no es un problema. Puede decidir separar el código de lo que se le pide en varios ficheros, siempre que no haya moulinette.
- Aun cuando un enunciado sea corto, merece la pena dedicarle algo de tiempo, para asegurarse de que comprende bien lo que se espera de usted, y de que lo ha hecho de la mejor manera posible.

Capítulo II

Ejercicio 00: Algunas funciones

	Ejercicio : 00
Ejercicio 00: Algunas funciones	
Directorio de entrega : $ex00/$	
Ficheros a entregar : `whatever.cpp y un Makefile`	
Funciones autorizadas : `Ninguna`	

Escriba las plantillas (templates) de las siguientes funciones:

- `swap`: Intercambia los valores de dos argumentos. No devuelve nada.
- `min`: compara los valores de dos argumentos y devuelve el más pequeño. Si los argumentos son iguales, devuelve el segundo argumento.
- `max`: compara los valores de dos argumentos y devuelve el más grande. Si los argumentos son iguales, devuelve el segundo argumento.

Se puede llamar a estas funciones con cualquier tipo de argumento (no obstante, los dos deben ser del mismo tipo y tienen que poder soportar los operadores de comparación). Entregue un `main` que demuestre que su código funciona.

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;
}
```

Tendría que devolver (si lo tiene todo correcto):

```
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 : $ex01/$	
Ficheros a entregar : `iter.cpp y un Makefile`	
Funciones autorizadas : `Ninguna`	

Escriba la plantilla de función `iter` que reciba 3 parámetros y no devuelva nada. El primer parámetro corresponde a la dirección de una tabla, el segundo a su tamaño y el tercero a la función a la que hay que llamar en cada elemento de la tabla.
Entregue un `main` que demuestre que `iter` funciona con cualquier tipo de tabla o con una instancia de la plantilla de la función del tercer parámetro.

Capítulo IV

Ejercicio 02: Tabla

	Ejercicio : 02
Ejercicio 02: Tabla	
Directorio de entrega : $ex02/$	
Ficheros a entregar : `Array.hpp o Array.tpp o Array.h, y un Makefile`	
Funciones autorizadas : `Ninguna`	

Escriba la plantilla de clase `Array` que contenga elementos de tipo `T` y que autorice los comportamientos siguientes:

- Construcción sin parámetro: crea una tabla vacía
- Construcción con un `unsigned int n` como parámetro: crea una tabla de `n` elementos, inicializados por defecto. (Truco: Intente compilar `int * a = new int();` y muestre `a`)
- Construcción mediante copia y asignación. En ambos casos, el hecho de modificar una de las tablas después de la copia/asignación no debería afectar a la otra.
- NO utilice el operador `new[]` para la reserva de memoria. No tiene que hacer reservas preventivas de memoria. Su código debe acceder a memoria que no haya sido reservada.
- Se debe poder acceder a los elementos con el operador [].
- Si al acceder a un elemento con el operador [] resulta que el elemento se encuentra fuera de los límites, utilice una `std::exception`.
- Una función miembro `size` que devuelva el número de elementos de la tabla. Esta función ni recibe argumentos ni modifica la instancia de ningún modo.

Entregue un `main` que demuestre que su código funciona.