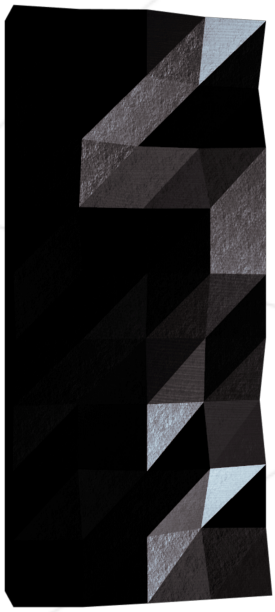

C++ - Módulo 06

C++ Casts

Resumen: Este documento contiene la evaluación del módulo 06 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

Reglas de la parte extra

- En estos ejercicios, la situación se tiene que resolver con un cast específico. La evaluación comprobará si ha elegido el cast adecuado.

Capítulo III

Ejercicio 00: Conversión escalar

	Ejercicio : 00
Ejercicio 00: Conversión escalar	
Directorio de entrega : $ex00/$	
Ficheros a entregar : `Los archivos que necesite + Makefile`	
Funciones autorizadas : `Una función que le permita convertir un string en un int, un float o un double. Le servirá de ayuda, pero no le hará todo el trabajo.`	

Escriba un programa que reciba como parámetro una cadena de caracteres que represente un valor literal de `C ++` (con su formato más habitual). Este valor debe pertenecer a alguno de los siguientes escalares: `char`, `int`, `float` o `double`. Solo se utilizará la notación decimal.

Ejemplos de valores literales `char`: `'c'`, `'a'`...
Para simplificar, consideraremos que no se pueden pasar caracteres no imprimibles a su programa. Si no se puede imprimir alguna conversión `char`, devuelva un mensaje de error.

Ejemplos de valores literales `int`: `0`, `-42`, `42`...

Ejemplos de valores literales `float`: `0.0f`, `-4.2f`, `4.2f`... También aceptará estos seudoliterales, ya sabe, para la ciencia: `-inff`, `+ inff` et `nanf`.

Ejemplos de valores literales `double`: `0.0`, `-4.2`, `4.2`... También aceptará estos seudoliterales, ya sabe, por diversión... : `-inf`, `+ inf` et `nan`.

El programa debe detectar el tipo del literal, adquirir el literal con el tipo adecuado (para que deje de ser una cadena) y convertirlo de forma **explícita** en los otros tres tipos y mostrar el resultado utilizando el mismo formato que se muestra a continuación. Si alguna de las conversiones carece de sentido o si desborda, indique que la conversión es imposible. Puede incluir cualquier header para gestionar los límites y los valores especiales.

Ejemplos:

```
./convert 0
char: Non displayable
int: 0
float: 0.0f
double: 0.0
./convert nan
char: impossible
int: impossible
float: nanf
double: nan
./convert 42.0f
char: '*'
int: 42
float: 42.0f
double: 42.0
```

Capítulo IV

Ejercicio 01: Serialización

	Ejercicio : 01
Ejercicio 01: Serialización	
Directorio de entrega : $ex01/$	
Ficheros a entregar : `Los archivos que necesite + Makefile`	
Funciones autorizadas : `Ninguna`	

Escriba la fonction "`void * serialize(void);`". Esta función devolverá la dirección sobre el heap de una secuencia de bytes que representan datos serializados. Los datos serializados corresponden a la concatenación de una tabla de 8 caracteres alfanuméricos, de un int aleatorio y de una segunda tabla de 8 caracteres alfanuméricos. Utilice lo que quiera para generar los valores aleatorios.

Escriba la fonction "`Data * deserialize(void * raw);`". Esta función debe deserializar los datos brutos en una estructura `Data` que se definirá así: "`struct Data {std::string s1; int n; std::string s2; };`". La memoria necesaria para la estructura se reservará en el heap.

Añada lo que sea preciso para crear un programa que demuestre que todo funciona.

Capítulo V

Ejercicio 02: Identifique el tipo verdadero

	Ejercicio : 02
Ejercicio 02: Identifique el tipo verdadero	
Directorio de entrega : $ex02/$	
Ficheros a entregar : `Los archivos que necesite + Makefile`	
Funciones autorizadas : `Ninguna`	

Cree una clase `Base` que posea únicamente un destructor público virtual. Cree tres clases vacías `A`, `B` y `C` que hereden públicamente de `Base`.

Escriba la función `"void identify_from_pointer(Base * p);"` que muestre `"A"`, `"B"` o `"C"` en función del verdadero tipo de `p`.

Escriba la función `"void identify_from_reference( Base & p);"` que muestre `"A"`, `"B"` o `"C"` en función del verdadero tipo de `p`.

Añada lo que sea preciso para crear un programa que demuestre que todo funciona. Está totalmente prohibido incluir `<typeinfo>`.