Remember that the quality of the defenses, hence the quality of the of the school on the labor market depends on you. The remote defences during the Covid crisis allows more flexibility so you can progress into your curriculum, but also brings more risks of cheat, injustice, laziness, that will harm everyone's skills development. We do count on your maturity and wisdom during these remote defenses for the benefits of the entire community.

SCALE FOR PROJECT CPP MODULE 01 (HTTPS://PROJECTS.INTRA.42.FR/PROJECTS/CPP-MODULE-01)

You should evaluate 1 student in this team

Git repository

```
git@vogsphere-v2.42madrid.com:vogsphere/intra-uuid-59bd5f74-f043-4a5(
```

Introduction

Por favor, respete las siguientes reglas:
- Durante el proceso de evaluación, sea educado, cortés, respetuoso y constructivo. El bienestar de la comunidad se basa en ello.

- Identifique con la persona (o el grupo) evaluada los posibles fallos del trabajo. Tómese su tiempo para hablar y debata sobre los problemas encontrados.

- Tenga en cuenta que pueden existir pequeñas diferencias de interpretación entre las instrucciones del proyecto, su alcance y sus funcionalidades. Mantenga la mente abierta y sea lo más justo posible a la hora de calificar. La pedagogía funciona únicamente si se realiza evaluación entre pares de forma seria.

Disclaimer

- Califique únicamente el contenido del directorio git clonado del estudiante o del grupo.

- Compruebe que el directorio git pertenece al estudiante o al grupo, que el proyecto es el correcto y que ha utilizado "git clone" sobre un directorio vacío.

- Preste mucha atención a que no se haya utilizado ningún alias para engañarle y asegúrese de que está corrigiendo la entrega oficial.

- Para evitar cualquier tipo de sorpresa, compruebe con el evaluado los posibles scripts utilizados para facilitar la evaluación.

- Si como evaluador no ha realizado el proyecto en curso, tendrá que leer todo el enunciado antes de empezar la evaluación.

- Utilice los flags disponibles para notificar una entrega vacía, un programa que no funcione, un error de norma, un problema de trampas... En estos casos, se termina la evaluación y la nota es 0 (o -42, en caso de trampas). No obstante, salvo en caso de trampas, se le anima a seguir comentando el trabajo entregado (aunque esté incompleto) para identificar las causas del fracaso y evitar que se vuelvan a reproducir en el futuro.

- Durante la evaluación, no se permitirá ningún segfault ni ninguna

parada del programa (inesperada, prematura o incontrolada). En esos casos,
la nota final es 0. Utilice el flag adecuado.
No debería tener que editar ningún archivo, salvo algún archivo de configuración,
en caso de que exista. Si desea modificar algún archivo, tendrá que explicar
claramente las razones del cambio y estar de acuerdo con el estudiante evaluado antes de
hacer nada.

- También le corresponde verificar que no existen fugas de memoria. Cualquier memoria reservada
en el heap deberá ser liberada correctamente antes de que finalice la ejecución.
Para eso, puede utilizar las diferentes herramientas que se encuentran en el ordenador,
como leaks, valgrind o también e_fence. En caso de fuga de memoria, marque el flag apropiado.

Guidelines

Debe compilar con clang++, con -Wall -Wextra -Werror
Le recordamos que se trata de un proyecto en C++98 y que no se puede utilizar ninguna función/container de las versiones posteriores.
Si los utiliza, tendrá un 0.

Estos elementos significan que no debe calificar el ejercicio solicitado:
- Se implementa una función en un header (salvo en los templates)
- Un Makefile compila sin los flags y/o con algo que no sea clang++

Estos elementos significan que debe ponerle el tag "Función Prohibida" al proyecto:
- El uso de una función "C" (*alloc, *printf, free)
- El uso de una función prohibida por el proyecto
- El uso de "using namespace" o de "friend"
- El uso de una librería externa o de las características de las versiones
posteriores a C++98

Attachments

☐ subject.pdf (https://cdn.intra.42.fr/pdf/pdf/30449/es.subject.pdf)

Ejercicio 00: BraiiiiiiinnnzzzZ

El objetivo de este ejercicio es entender cómo reservar memoria en C++.

Makefile y main

Debe haber un Makefile que compila utilizando las flags apropiadas.
Debe haber un main para probar el ejercicio.

☐ Yes ☐ No

Clase Zombie

Hay una clase Zombie.
Tiene un atributo nombre privado.
Tiene al menos un constructor.
Tiene una función miembro announce(void) que imprime: " BraiiiiiiinnnzzzZ..."
El destructor imprime un mensaje que incluye el nombre del zombie.

☐ Yes ☐ No

newZombie

Hay una función newZombie prototipada así: [Zombie* newZombie(std::string name);]
Debe reservar un Zombie en el heap y devolverlo.
Lo ideal sería que llamara al constructor que acepta una string e inicializa el nombre.

El ejercicio debe validarse si el Zombie puede anunciarse a sí mismo con el nombre pasado a la función.
El main contiene tests para demostrarlo.
El zombie se elimina correctamente antes de terminar el programa.

☐ Yes ☐ No

---

randomChump

Hay una función randomChump prototipada así: [void randomChump(std::string name);]
Debe crear un Zombie en el stack, y hacer que se anuncie a sí mismo.
Lo ideal sería que el zombie estuviera localizado en el stack (para que se elimine implícitamente al final de la función). Puede estar también reservado en el heap y ser eliminado explícitamente.
El estudiante debe justificar su elección.
El main debe contener las pruebas para demostrarlo.

☐ Yes ☐ No

Ejercicio 01: ¡Másss cerebross!

El objetivo de este ejercicio es reservar un número de objetos a la vez utilizando new[], iniciarlos, y eliminarlos propiamente.

Makefile and main

Hay un Makefile que compila utilizando las flags apropiadas.
Hay un main para probar el ejercicio.

☐ Yes ☐ No

---

zombieHorde

La clase Zombie tiene un constructor por defecto.
Hay una función zombieHorde prototipada así: [Zombie* zombieHorde(int N, std::string name);]
Reserva espacio en el heap para N zombies utilizando new[].
Después de la reserva, hay una inicialización de los objetos para establecer su nombre.
Devuelve un puntero al primer zombie.
Hay suficientes pruebas en el main para demostrar los puntos anteriores.
Por ejemplo; llamar a announce() en todos los zombies.
Por último, todos los zombies deben eliminarse a la vez en el main.

☐ Yes ☐ No

Ejercicio 02: HI THIS IS BRAIN

¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias! ¡Desmitifica las referencias!

Makefile y main

Hay un Makefile que compila utilizando las flags apropiadas.
Hay un main que prueba el ejercicio.

☐ Yes ☐ No

HI THIS IS BRAIN

Hay una string que contiene "HI THIS IS BRAIN".
stringPTR apunta a la string.
stringREF toma una referencia a la string.
La dirección de la string se muestra utilizando la variable de la string,
su stringPTR y su stringREF.
La string se muestra utilizando el stringPTR y la stringREF.

☐ Yes ☐ No

Ejercicio 03: Violencia innecesaria

El objetivo de este ejercicio es entender las pequeñas diferencias que presentan los punteros y las referencias y hacer un mejor uso de ellos en función del ciclo de vida del objeto que vamos a utilizar.

Makefile y main

Hay un Makefile que compila utilizando las flags apropiadas.
Hay un main que prueba el ejercicio.

☐ Yes ☐ No

Weapon

Hay una clase Weapon que tiene una string type, un getType y un setType.
La función getType devuelve una referencia constante a la string type.

☐ Yes ☐ No

HumanA y HumanB

HumanA puede tener una referencia o un puntero a Weapon.
Lo ideal es que esté implementado como una referencia, dado que
Weapon existe desde que se crea hasta que se destruye, sin cambios.
HumanB tiene que tener un puntero a Weapon porque el valor no se
establece al crearse, y Weapon puede ser NULL.

☐ Yes ☐ No

Ejercicio 04: Sed es para perdedores

Tras hacer este ejercicio deberías sentirte cómodo con ifstream y ofstream.

Makefile y main

Hay un Makefile que compila utilizando las flags apropiadas.
Hay un main que prueba el ejercicio.

☐ Yes ☐ No

Sed es para perdedores

Hay una función replace que funciona como se especifica en el subject.
La gestión de errores debe ser buena; intenta pasar un archivo inexistente,
cambiar los permisos, mandarlo vacío, etc.
Si encuentras un error que no se gestiona, y es algo común, no des puntos
a este ejercicio.
El programa debe leer del archivo utilizando un ifstream o equivalente,
y escribir utilizando un ofstream o equivalente.

La implementación de la función debe hacerse utilizando funciones de std::string, no leyendo la string carácter por carácter.
Esto ya no es C.

☐ Yes ☐ No

Ejercicio 05: Karen 2.0

El objetivo de este ejercicio es hacerte utilizar punteros a a funciones miembro de clases. Es también un buen momento para introducir los niveles de log.

Makefile y main

Hay un Makefile que compila utilizando las flags apropiadas.
Hay un main que prueba el ejercicio.

☐ Yes ☐ No

Querida Karen

Hay una clase Karen con al menos las 5 funciones solicitadas en el enunciado.
La función complain() ejecuta las otras funciones utilizando un puntero a ellas.
Lo ideal es que el estudiante evaluado haya implementado un sistema de emparejamiento de las distintas strings en función del nivel de log a la función miembro correspondiente.
Si la implementación es diferente pero el ejercicio funciona puedes darlo como válido.
Lo único que no se permite es el uso de if/elseif/else.
El estudiante puede elegir cambiar el mensaje de Karen o dejar el dado en el enunciado. Ambas son opciones válidas.

☐ Yes ☐ No

Ejercicio 06: Vamos a filtrar a Karen

Ahora que habéis crecido como programadores, deberéis empezar a utilizar nuevos tipos de instrucciones, sentencias, bucles, etc. El objetivo de este ejercicio es hacerte descubrir el SWITCH.

Makefile y main

Hay un Makefile que compila utilizando las flags apropiadas.
Hay un main que prueba el ejercicio.

☐ Yes ☐ No

Apaga a Karen

El programa karenFilter acepta un único parámetro que se corresponde al nivel de log: "DEBUG", "INFO", "WARNING" y "ERROR". Después de esto, solo mostrará los mensajes con el mismo nivel o superior (debug > info > warning > error). Esto debe implementarse utilizando una sentencia SWITCH. De nuevo, no se toleran if/elseif/else ya.
El switch debe tener un caso por defecto.

☐ Yes ☐ No

Ratings

Don't forget to check the flag corresponding to the defense

☐ Ok

☐ Empty work ☐ Incomplete work W Invalid compilation ☐ Cheat d Crash l Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation