encouraged to continue to discuss your work (even if you have not finished it) to identify any issues that may have caused this failure and avoid repeating the same mistake in the future.

- Remember that for the duration of the defense, no segfault, no other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag.
You should never have to edit any file except the configuration file if it exists. If you want to edit a file, take the time to explicit the reasons with the evaluated student and make sure both of you are okay with this.

- You must also verify the absence of memory leaks. Any memory allocated on the heap must be properly freed before the end of execution.
You are allowed to use any of the different tools available on the computer, such as leaks, valgrind, or e_fence. In case of memory leaks, tick the appropriate flag.

Disclaimer

Please respect the following rules:

- Remain polite, courteous, respectful and constructive throughout the evaluation process. The well-being of the community depends on it.

- Identify with the person (or the group) evaluated the eventual dysfunctions of the work. Take the time to discuss and debate the problems you have identified.

- You must consider that there might be some difference in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade him/her as honestly as possible. The pedagogy is valid only and only if peer-evaluation is conducted seriously.

Guidelines

You must compile with clang++, with -Wall -Wextra -Werror
As a reminder, this project is in C++98
C++11 and later members functions or containers are NOT expected.

Any of these means you must not grade the exercise in question:
- A function is implemented in a header (except in a template)
- A Makefile compiles without flags and/or with something other than clang++

Any of these means that you must flag the project as Forbidden Function:
- Use of a "C" function (*alloc, *printf, free)
- Use of a function not allowed in the subject

- Use of "using namespace" or "friend"
- Use of an external library, or C++20 features
- Use of an already existing container, or any existing function, to implement another container

---

Attachments

 subject.pdf (https://cdn.intra.42.fr/pdf/pdf/27665/en.subject.pdf)

ex00

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise.

Class and attributes

There is a ClapTrap class present.
It has all the following private attributes ():
hitpoints
energy
name
Attack damage
Its attributes are initialized to the required values.

 Yes No

Member functions

The following member functions are present and work as specified:

- attack
- takeDamage - beRepaired

 Yes No

---

ex01

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise.

Class and attributes

There is a ScavTrap class present.

The ScavTrap inherits publicly from the ClapTrap class. It does not redeclare attributes.
The ClapTrap attributes are now protected instead of private.
Its attributes are initialized to the required values.

 Yes No

Member functions

The following member functions are present and work as specified:
- attack
- takeDamage (inherited)
- beRepaired (inherited)
The outputs of the constructor, destructor, and attack must be different from the ones in the ClapTrap.

 Yes No

Construction and destruction

There must be a constructor and a destructor for the ScavTrap with its specific messages, and it must be implemented so that it is called in the correct order when used, namely, if you create a ScavTrap it must first display the ClapTrap's message then the ScavTrap's, and if you delete it, it must display the ScavTrap's message first, then the ClapTrap's

 Yes No

Special features

There is a guardGate function that displays a small message on the standard output.
There is an attack function that displays a small message on the standard output different from the original "ClapTrap".

 Yes No

---

ex02

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise.

Parent class

There is a FragTrap class that
inherit publicly from ClapTrap.
Attributes must not be redeclared without reasons.

 Yes No

Construction and destruction

There must be a constructor and a destructor for the FragTrap with its specific messages, and it must be implemented so that it is called in the correct order when used, namely, if you create a FragTrap it must first display the ClapTrap's message then the FragTrap's, and if you delete it, it must display the FragTrap's message first, then the ClapTrap's

 Yes No

Special features

There is a highFivesGuys function that displays a small message on the standard output. There is an attack function that displays a small message on the standard output different from the original "ClapTrap".

 Yes No

---

ex03

As usual, there has to be a main function that contains enough tests to prove the program works as required. If there isn't, do not grade this exercise.

Ultimate C++ weird feature

There is a DiamondTrap class present.
It inherits from both the
FragTrap and the ScavTrap.
It sets the attributes to the appropriate values.
It uses virtual inheritance to avoid the pitfalls of diamond inheritance.

 Yes No

Choose wisely...

The DiamondTrap uses the attack method of the Scavtrap.
It has the special functions of both its parents.
The diamond trap has a private std::string name member.
The function whoAmI access to both name and clapTrap::name.

Yes No

Ratings

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

Ok Outstanding project

Empty work No author file Invalid compilation Norme Cheat Crash

Leaks Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation