SCALE FOR PROJECT CPP MODULE 06 (HTTPS://PROJECTS.INTRA.42.FR/PROJECTS/CPP-MODULE-06)

You should evaluate 1 student in this team

Introduction

Please comply with 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 student or group whose work is evaluated the possible dysfunctions in their project. Take the time to discuss and debate the problems that may have been identified.

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

Guidelines

- Only grade the work that was turned in the Git repository of the evaluated student or group.

- Double-check that the Git repository belongs to the student(s). Ensure that the project is the one expected. Also, check that 'git clone' is used in an empty folder.

- Check carefully that no malicious aliases was used to fool you and make you evaluate something that is not the content of the official repository.

- To avoid any surprises and if applicable, review together any scripts used to facilitate the grading (scripts for testing or automation).

- If you have not completed the assignment you are going to evaluate, you have to read the entire subject prior to starting the evaluation process.

- Use the available flags to report an empty repository, a non-functioning program, a Norm error, cheating, and so forth.
In these cases, the evaluation process ends and the final grade is 0,

or -42 in case of cheating. However, except for cheating, student are strongly encouraged to review together the work that was turned in, in order to identify any mistakes that shouldn't be repeated in the future.

- 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.

Attachments

subject.pdf (https://github.com/rphlr/42-Subjects/)

Preliminary tests

If cheating is suspected, the evaluation stops here. Use the "Cheat" flag to report it. Take this decision calmly, wisely, and please, use this button with caution.

Prerequisites

The code must compile with c++ and the flags -Wall -Wextra -Werror
Don't forget this project has to follow the C++98 standard. Thus, C++11 (and later) 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 file (except for template functions).
- A Makefile compiles without the required flags and/or another compiler than c++.

Any of these means that you must flag the project with "Forbidden Function":

- Use of a "C" function (*alloc, *printf, free).
- Use of a function not allowed in the exercise guidelines.
- Use of "using namespace <ns_name>" or the "friend" keyword.
- Use of an external library, or features from versions other than C++98.

Yes No

Exercise 00: Conversion of scalar types

This exercise is about using the static_cast.

Scalar conversion

Did the student create a class with a private constructor, and static methods ?
Did the student use the static_cast to convert values?
Accept the use of implicit casts for promotion casts only.
Does the program work as required?

Anyway, please don't be too uncompromising towards the exercise's outputs if the spirit of the exercise is respected.
Even if this exercise is wrong, continue the evaluation process.

Yes No

Exercise 01: Serialization

This exercise is about using the reinterpret_cast.

Retyping of raw data

Does the program work as required?
Did the student create a class with a private constructor, and static methods ?
The reinterpret_cast<> should be used twice:

- First from data* to uintptr_t.
- Then, from uintptr_t to data*.

And the resulting data struct should be usable.

Yes No

Exercise 02: Identify real type

This exercise is about using the dynamic_cast.

Real type identification

Does the program work as required?
Check the code. Did the student use the dynamic_cast to identify the real type?
void identify(Base* p) should check if the cast return is NULL.
void identify(Base& p) should use a try and catch block to check if the cast failed.

(In case you're wondering, the header <typeinfo> must not appear anywhere.)

Yes No

Ratings

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

Ok

Outstanding project

Empty work

Incomplete work

W Invalid compilation

Cheat

Crash

Concerning situation

Leaks

l Forbidden function

Can’t support / explain code

Give this repository a star. ⭐

(https://github.com/rphlr/42-Evals)