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 02

You should evaluate 1 student in this team

Git repository

```
git@vogsphere-v2-bg.1
```

Introduction

- Only grade the work that is in the student or group's GiT repository.

- Double-check that the GiT repository belongs to the student or the group. Ensure that the work is for the relevant project and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases were used to fool you and make you evaluate something other than the content of the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for this student to read the entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository, non-functioning program, a norm error, cheating etc. In these cases, the grading is over and the final grade is 0 (or -42 in case of cheating). However, with the exception of cheating, you are encouraged to continue to discuss your work (even if you have not finished it) in order 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 defence, 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

nim/ ner as honestly as possible. The peaagogy is valia 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 and C++20 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

Exercise 00: My First Canonical

This exercice introduces the notion of canonical class with a simple arithmetic exemple: the fixed point numbers.

Accessors

The Fixed class (or whatever its name) must provide accessors to
the raw value:
- int getRawBits(void) const;
- void setRawBits(
int const raw);
Are these member functions present and functional?

Yes | No

Canonical

A canonical class must provide at least:
- A default constructor
- A destructor
- A copy constructor
- An assignation operator
Are these elements present and functional?

Yes | No

Exercise 01: Towards a more useful fixed point class

Ex00 was a good start, but our class is still pretty useless being only able to represent the fixed point value 0.0.

Floating point constructor

Is it possible to construct an instance from a floating point value?

Yes | No

<< operator

Is there a << operator overload and is it functional?

Yes | No

Fixed point value to integer value

A member function "int toInt(void) const;" that converts the
fixed point value to an integer value must be present. Is it functional?

Yes | No

Fixed point value to floating point value

A member function \"float toFloat(void) const;\" that converts the fixed point value to a float value must be present. Is it functional?

Yes | No

Integer constructor

Is it possible to construct an instance from an integer value?

Yes | No

Exercise 02: Now we're talking

This exercice add comparison and arithmetic features to the class.

Division

The class must provide a division operator. Is it present and functional?

Yes | No

Addition and substraction

The class must provide addition and substraction operators. Are they present and functional?

Yes | No

Pre/post increment and pre/post decrement operators

The class must provide the pre-increment, post-increment, pre-decrement and post-decrement operators, that will increment or decrement the fixed point value from the smallest representable\\x0F such as 1 + \\x0F > 1. Are they present and are they functional?

Yes | No

Multiplication

The class must provide a multiplication operator. Is it present and functional?

Yes | No

Min and max

The class must provide 4 non member functions: min, max and their const overloads. Are they present and are they functional?

Yes | No

Comparison operators

The class must provide 6 comparison operators: \>, <, >=, <=, == and !=. Are they present and functional?

Yes | No

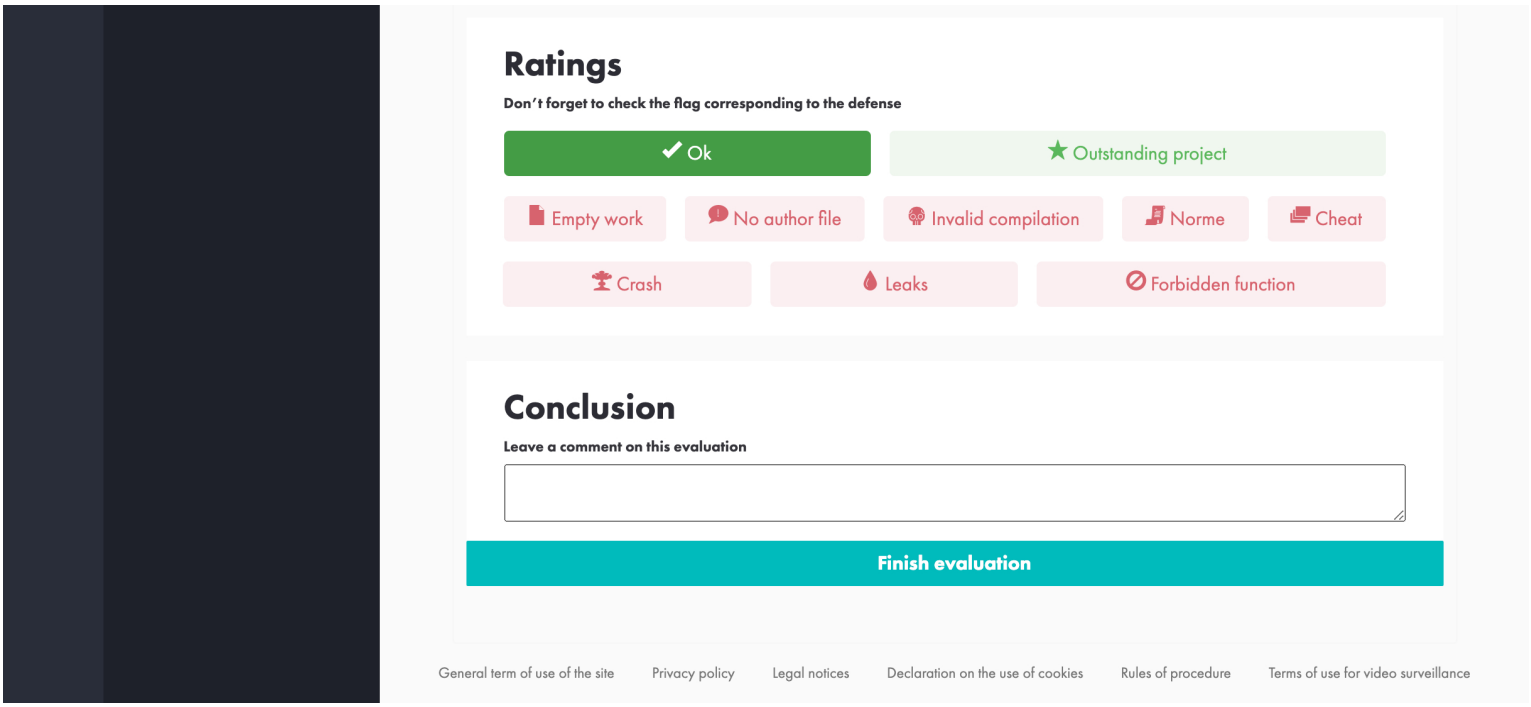
Exercise 03: Fixed point expressions

A small fixed point expressions interpreter.

Small fixed point expressions interpreter

Test the interpreter with more and more complex expressions. Start with values, then move to simple addtion, multiplications, then test complex expressions with parenthesis.

Rate it from 0 (failed) through 5 (excellent)


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
General term of use of the site
Privacy policy
Legal notices
Declaration on the use of cookies
Rules of procedure
Terms of use for video surveillance