SCALE FOR PROJECT PISCINE CPP (/PROJECTS/PISCINE-CPP) / DAY 02 (/PROJECTS/42-PISCINE-C-FORMATION-PISCINE-CPP-DAY-02)

Introduction

The subject of this project is rather vague and leaves a lot to the user's choice. This is INTENDED. The questions in this grading scale, however, are very focused and concentrate on what we think is the core of each exercise, what we want you to grasp. So we would like you to do the same : You can and should tolerate moderate deviations in filenames, function names, etc ... as long as the exercise basically works as intended. Of course, in case the student you are grading really strayed too far, you should not grade the exercise in question at all. We leave it to your good judgement to determine what constitutes "straying too far".

The usual obvious rules apply : Only grade what's on the git repository of the student, don't be a dick, and basically be the grader you would like to have grading you.

Do NOT stop grading when an exercise is wrong.

Guidelines

You must compile with clang++, with -Wall -Wextra -Werror

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 Cheat:
- Use of a "C" function (*alloc, *printf, free)
- Use of a function not allowed in the subject
- Use of "using namespace" or "friend" (Unless explictly allowed in the subject)
- Use of an external library, or C++11 features (Unless explictly allowed in the subject)

Attachments

Subject (/uploads/document/document/1030/d02.en.pdf)

Exercise 00: My First Canonical

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

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

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

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 an integer 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: Food for thought

There are no points related to this exercise. It's just about thinking. Take a couple minutes to have ask the following question to the student you're evaluating : What do you think of using a namespace instead of a class to represent fixed point values ? I mean, with a scoped typedef on int, we could write scoped functions and save\r\nthe cost of instanciations, but at the same time, we'd loose some handy syntax. What's your point of view ? The bool selector below has no incidence on grade.

Casual chat

See above.

Yes No

Exercise 04: 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

Empty work | Incomplete work | No author file | Invalid compilation

Norme | Cheat | Crash | Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!