SCALE FOR PROJECT PISCINE CPP (/PROJECTS/PISCINE-CPP) / DAY 07 (/PROJECTS/42-PISCINE-C-FORMATION-PISCINE-CPP-DAY-07)

You should evaluate 1 student in this team

★

Git repository

vogsphere@vgs.42.us.or

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".

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++
- A class is not in Coplien's form

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)
- Use of "C" legacy cast

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/2059/d07.en.pdf)

Exercice 00: A few functions

In this exercice, the student must write 3 simple function templates: swap, min and max.

Simple types

Refer to the subject for the expected output with simple types, such as int.

☑ Yes ☒ No

Complex types

Does the functions also work with complex types such as ?

```
class Awesome
{
  public:
    Awesome( int n ) : _n( n ) {}

    bool operator==( Awesome const & rhs ) { return (this->_n == rhs._n); }
    bool operator!=( Awesome const & rhs ) { return (this->_n != rhs._n); }
    bool operator>( Awesome const & rhs ) { return (this->_n > rhs._n); }
    bool operator<( Awesome const & rhs ) { return (this->_n < rhs._n); }
    bool operator>=( Awesome const & rhs ) { return (this->_n >= rhs._n); }
    bool operator<=( Awesome const & rhs ) { return (this->_n <= rhs._n); }

  private:
    int _n;
};
```

?

☑ Yes ☒ No

Exercice 01: Iter

The aim of this exercice is to write a generic iteration function through arrays.

Does it work ???

Test the following code with the student's iter:

```
class Awesome
{
  public:
        Awesome( void ) : _n( 42 ) { return; }
        int get( void ) const { return this->_n; }

  private:
        int _n;
};

std::ostream & operator<<( std::ostream & o, Awesome const & rhs ) { o << rh

template< typename T >
void print( T const & x ) { std::cout << x << std::endl; return; }

int main()
{
    int tab[] = { 0, 1, 2, 3, 4 };     // <--- J'ai jamais compris pourquoi
    Awesome tab2[5];
    iter( tab, 5, print );
    iter( tab2, 5, print );
    return 0;
}
```

If everything went well, it should display:
0
1
2
3
4
42
42
42
42
42

Yes No

Exercise 02: Array

In this exercice, the student must write a class template that behaves like an array. If the inner allocation of the actual array does not come from a use of new[], don't grade this exercice. Ask the student to proove her/his work with arrays of simple and complex types before grading.

Constructors

Is it possible to create an empty array and an array of a specific size ?

Yes No

Access

Elements must be accessible for reading and writing through the operator[] (or just for reading if the instance is const).
Access to an element out of the limits must throw an std::exception.

Yes No

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

Finish evaluation