

C++ - Module 07
C++ templates

 $Summary: \\ This \ document \ contains \ the \ exercises \ of \ Module \ 07 \ from \ C++ \ modules.$ 

Version: 9.3

# Contents

Ι	Introduction	2
II	General rules	3
III	Exercise 00: Start with a few functions	5
IV	Exercise 01: Iter	7
V	Exercise 02: Array	8
$\mathbf{VI}$	Submission and peer-evaluation	9

### Chapter I

### Introduction

C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language, or "C with Classes" (source: Wikipedia).

The goal of these modules is to introduce you to **Object-Oriented Programming**. This will be the starting point of your C++ journey. Many languages are recommended to learn OOP. We decided to choose C++ since it's derived from your old friend C. Because this is a complex language, and in order to keep things simple, your code will comply with the C++98 standard.

We are aware modern C++ is way different in a lot of aspects. So if you want to become a proficient C++ developer, it's up to you to go further after the 42 Common Core!

### Chapter II

### General rules

#### Compiling

- Compile your code with c++ and the flags -Wall -Wextra -Werror
- Your code should still compile if you add the flag -std=c++98

#### Formatting and naming conventions

- The exercise directories will be named this way: ex00, ex01, ..., exn
- Name your files, classes, functions, member functions and attributes as required in the guidelines.
- Write class names in **UpperCamelCase** format. Files containing class code will always be named according to the class name. For instance: ClassName.hpp/ClassName.h, ClassName.cpp, or ClassName.tpp. Then, if you have a header file containing the definition of a class "BrickWall" standing for a brick wall, its name will be BrickWall.hpp.
- Unless specified otherwise, every output messages must be ended by a new-line character and displayed to the standard output.
- Goodbye Norminette! No coding style is enforced in the C++ modules. You can follow your favorite one. But keep in mind that a code your peer-evaluators can't understand is a code they can't grade. Do your best to write a clean and readable code.

#### Allowed/Forbidden

You are not coding in C anymore. Time to C++! Therefore:

- You are allowed to use almost everything from the standard library. Thus, instead of sticking to what you already know, it would be smart to use as much as possible the C++-ish versions of the C functions you are used to.
- However, you can't use any other external library. It means C++11 (and derived forms) and Boost libraries are forbidden. The following functions are forbidden too: \*printf(), \*alloc() and free(). If you use them, your grade will be 0 and that's it.

C++ - Module 07 C++ templates

• Note that unless explicitly stated otherwise, the using namespace <ns\_name> and friend keywords are forbidden. Otherwise, your grade will be -42.

• You are allowed to use the STL in the Module 08 and 09 only. That means: no Containers (vector/list/map/and so forth) and no Algorithms (anything that requires to include the <algorithm> header) until then. Otherwise, your grade will be -42.

#### A few design requirements

- Memory leakage occurs in C++ too. When you allocate memory (by using the new keyword), you must avoid memory leaks.
- From Module 02 to Module 09, your classes must be designed in the **Orthodox** Canonical Form, except when explicitly stated otherwise.
- Any function implementation put in a header file (except for function templates) means 0 to the exercise.
- You should be able to use each of your headers independently from others. Thus, they must include all the dependencies they need. However, you must avoid the problem of double inclusion by adding **include guards**. Otherwise, your grade will be 0.

#### Read me

- You can add some additional files if you need to (i.e., to split your code). As these assignments are not verified by a program, feel free to do so as long as you turn in the mandatory files.
- Sometimes, the guidelines of an exercise look short but the examples can show requirements that are not explicitly written in the instructions.
- Read each module completely before starting! Really, do it.
- By Odin, by Thor! Use your brain!!!



You will have to implement a lot of classes. This can seem tedious, unless you're able to script your favorite text editor.



You are given a certain amount of freedom to complete the exercises. However, follow the mandatory rules and don't be lazy. You would miss a lot of useful information! Do not hesitate to read about theoretical concepts.

### Chapter III

# Exercise 00: Start with a few functions



Implement the following function templates:

- swap: Swaps the values of two given parameters. Does not return anything.
- min: Compares the two values passed as parameters and returns the smallest one. If they are equal, it returns the second one.
- max: Compares the two values passed as parameters and returns the greatest one. If they are equal, it returns the second one.

These functions can be called with any type of argument. The only requirement is that the two arguments must have the same type and must support all the comparison operators.



Templates must be defined in the header files.

C++ - Module 07 C++ templates

Running the following code:

```
int main( void ) {
    int a = 2;
    int b = 3;

    ::swap( a, b );
    std::cout << "a = " << a << ", b = " << b << std::endl;
    std::cout << "min( a, b ) = " << ::min( a, b ) << std::endl;
    std::cout << "max( a, b ) = " << ::max( a, b ) << std::endl;

    std::string c = "chaine1";
    std::string d = "chaine2";

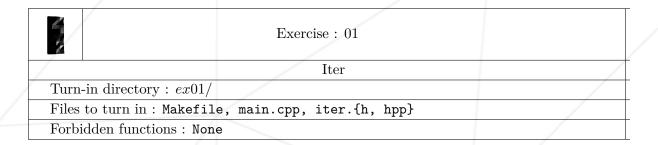
    ::swap(c, d);
    std::cout << "c = " << c << ", d = " << d << std::endl;
    std::cout << "c = " << c << ", d = " << d << std::endl;
    std::cout << "min( c, d ) = " << ::min( c, d ) << std::endl;
    std::cout << "max( c, d ) = " << ::max( c, d ) << std::endl;
    return 0;
}</pre>
```

Should output:

```
a = 3, b = 2
min(a, b) = 2
max(a, b) = 3
c = chaine2, d = chaine1
min(c, d) = chaine1
max(c, d) = chaine2
```

### Chapter IV

### Exercise 01: Iter



Implement a function template iter that takes 3 parameters and returns nothing.

- The first parameter is the address of an array.
- The second one is the length of the array.
- The third one is a function that will be called on every element of the array.

Submit a main.cpp file that contains your tests. Provide enough code to generate a test executable.

Your iter function template must work with any type of array. The third parameter can be an instantiated function template.

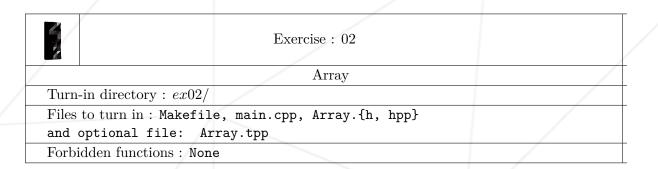
The function passed as the third parameter may take its argument by const reference or non-const reference, depending on the context.



Think carefully about how to support both const and non-const elements in your iter function.

### Chapter V

### Exercise 02: Array



Develop a class template **Array** that contains elements of type T and that implements the following behavior and functions:

- Construction with no parameter: Creates an empty array.
- Construction with an unsigned int n as a parameter: Creates an array of n elements initialized by default.

Tip: Try to compile int \* a = new int(); then display \*a.

- Construction by copy and assignment operator. In both cases, modifying either the original array or its copy after copying musn't affect the other array.
- You MUST use the operator new[] to allocate memory. Preventive allocation (allocating memory in advance) is forbidden. Your program must never access non-allocated memory.
- Elements can be accessed through the subscript operator: [].
- When accessing an element with the [] operator, if its index is out of bounds, an std::exception is thrown.
- A member function size() that returns the number of elements in the array. This member function takes no parameters and must not modify the current instance.

As usual, ensure everything works as expected and turn in a main.cpp file that contains your tests.

## Chapter VI

## Submission and peer-evaluation

Turn in your assignment in your Git repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.



16D85ACC441674FBA2DF65190663F43A243E8FA5424E49143B520D3DF8AF68036E47 114F20A16827E1B16612137E59ECD492E468BC6CD109F65388DC57A58E8942585C8 D193B96732206