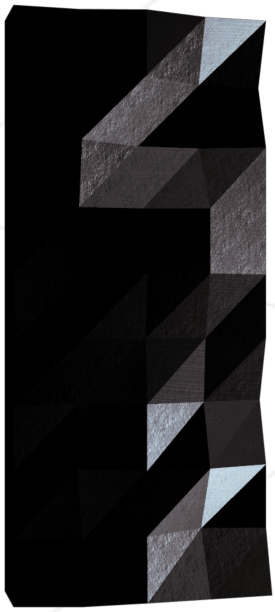

C++ - Module 07

C++ templates

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

Version: 10.0

Contents

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 message must end with a newline character and be 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 code your peer evaluators can't understand is code they can't grade. Do your best to write 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 the C++-ish versions of the C functions you are used to as much as possible.
- 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.

- 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 only in Modules 08 and 09.** That means: no **Containers** (vector/list/map, and so forth) and no **Algorithms** (anything that requires including 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!!!

Regarding the Makefile for C++ projects, the same rules as in C apply (see the Norm chapter about the Makefile).

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

AI Instructions

Context

This project is designed to help you discover the fundamental building blocks of your 42 training.

To properly anchor key knowledge and skills, it's essential to adopt a thoughtful approach to using AI tools and support.

True foundational learning requires genuine intellectual effort — through challenge, repetition, and peer-learning exchanges.

For a more complete overview of our stance on AI — as a learning tool, as part of the 42 training, and as an expectation in the job market — please refer to the dedicated FAQ on the intranet.

Main message

- Build strong foundations without shortcuts.
- Really develop tech & power skills.
- Experience real peer-learning, start learning how to learn and solve new problems.
- The learning journey is more important than the result.
- Learn about the risks associated with AI, and develop effective control practices and countermeasures to avoid common pitfalls.

Learner rules:

- You should apply reasoning to your assigned tasks, especially before turning to AI.

- You should not ask for direct answers to the AI.
- You should learn about 42 global approach on AI.

Phase outcomes:

Within this foundational phase, you will get the following outcomes:

- Get proper tech and coding foundations.
- Know why and how AI can be dangerous during this phase.

Comments and example:

- Yes, we know AI exists — and yes, it can solve your projects. But you're here to learn, not to prove that AI has learned. Don't waste your time (or ours) just to demonstrate that AI can solve the given problem.
- Learning at 42 isn't about knowing the answer — it's about developing the ability to find one. AI gives you the answer directly, but that prevents you from building your own reasoning. And reasoning takes time, effort, and involves failure. The path to success is not supposed to be easy.
- Keep in mind that during exams, AI is not available — no internet, no smartphones, etc. You'll quickly realise if you've relied too heavily on AI in your learning process.
- Peer learning exposes you to different ideas and approaches, improving your interpersonal skills and your ability to think divergently. That's far more valuable than just chatting with a bot. So don't be shy — talk, ask questions, and learn together!
- Yes, AI will be part of the curriculum — both as a learning tool and as a topic in itself. You'll even have the chance to build your own AI software. In order to learn more about our crescendo approach you'll go through in the documentation available on the intranet.

✓ Good practice:

> I'm stuck on a new concept. I ask someone nearby how they approached it. We talk for 10 minutes — and suddenly it clicks. I get it.

✗ Bad practice:

> I secretly use AI, copy some code that looks right. During peer evaluation, I can't explain anything. I fail. During the exam — no AI — I'm stuck again. I fail.

Chapter IV

Exercise 00: Start with a few functions

	Exercise: 00
Start with a few functions	
Directory: *ex*00/	
Files to Submit: `Makefile, main.cpp, whatever.{h, hpp}`	
Forbidden: `None`	

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.`

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 << "min( c, d ) = " << ::min( c, d ) << std::endl;
    std::cout << "max( c, d ) = " << ::max( c, d ) << std::endl;

    return 0;
}
```

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 V

Exercise 01: Iter

	Exercise: 01
Iter	
Directory: *ex*01/	
Files to Submit: `Makefile, main.cpp, iter.{h, hpp}`	
Forbidden: `None`	

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, passed as a `const` value.
- 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 VI

Exercise 02: Array

	Exercise: 02
Array	
Directory: *ex*02/	
Files to Submit: `Makefile, main.cpp, Array.{h, hpp}` and optional file: `Array.tpp`	
Forbidden: `None`	

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 VII

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.

During the evaluation, a brief **modification of the project** may occasionally be requested. This could involve a minor behavior change, a few lines of code to write or rewrite, or an easy-to-add feature.

While this step may **not be applicable to every project**, you must be prepared for it if it is mentioned in the evaluation guidelines.

This step is meant to verify your actual understanding of a specific part of the project. The modification can be performed in any development environment you choose (e.g., your usual setup), and it should be feasible within a few minutes — unless a specific timeframe is defined as part of the evaluation.
You can, for example, be asked to make a small update to a function or script, modify a display, or adjust a data structure to store new information, etc.

The details (scope, target, etc.) will be specified in the **evaluation guidelines** and may vary from one evaluation to another for the same project.

16D85ACC441674FBA2DF65190663F43A243E8FA5424E49143B520D3DF8AF68036E47
114F20A16827E1B16612137E59ECD492E468BC6CD109F65388DC57A58E8942585C8
D193B96732206