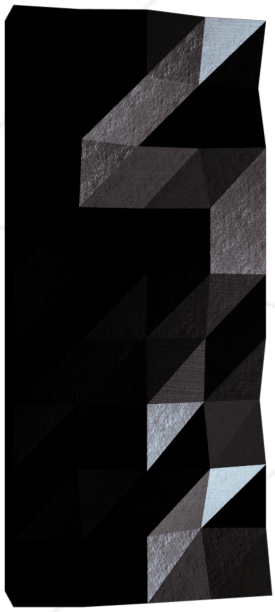

ft_containers

C++ containers, easy mode

Summary: This document is the subject for the project C++ containers of 42.

Contents

Chapter I

General rules

- Any function implemented in a header (except in the case of templates), and any unprotected header, means 0 to the exercise.
- Every output goes to the standard output, and will be ended by a newline, unless specified otherwise.
- The imposed filenames must be followed to the letter, as well as class names, function names and method names.
- Remember: You are coding in `C++` now, not in `C` anymore. Therefore:
  - The following functions are FORBIDDEN, and their use will be punished by a `0`, no questions asked: `*alloc`, `*printf` and `free`.
  - You are allowed to use basically everything in the standard library. HOWEVER, it would be smart to try and use the C++-ish versions of the functions you are used to in C, instead of just keeping to what you know, this is a new language after all. And NO, you are not allowed to use the STL until you actually are supposed to (that is, until module 08). That means no vectors/lists/maps/etc... or anything that requires an include <algorithm> until then.
- Actually, the use of any explicitly forbidden function or mechanic will be punished by a `0`, no questions asked.
- Also note that unless otherwise stated, the `C++` keywords `"using namespace"` and `"friend"` are forbidden. Their use will be punished by a `-42`, no questions asked.
- Files associated with a class will always be `ClassName.hpp` and `ClassName.cpp`, unless specified otherwise.
- Turn-in directories are `ex00/`, `ex01/`, ..., `exn/`.
- You must read the examples thoroughly. They can contain requirements that are not obvious in the exercise's description. If something seems ambiguous, you don't understand `C++` enough.
- Since you are allowed to use the `C++` tools you learned about since the beginning, you are not allowed to use any external library. And before you ask, that also means

no `C++11` and derivates, nor `Boost` or anything your awesomely skilled friend told you `C++` can't exist without.

- You may be required to turn in an important number of classes. This can seem tedious, unless you're able to script your favorite text editor.
- Read each exercise FULLY before starting it! Really, do it.
- The compiler to use is `clang++`.
- Your code has to be compiled with the following flags : `-Wall -Wextra -Werror`.
- Each of your includes must be able to be included independently from others. Includes must contains every other includes they are depending on, obviously.
- In case you're wondering, no coding style is enforced during in `C++`. You can use any style you like, no restrictions. But remember that a code your peer-evaluator can't read is a code she or he can't grade.
- Important stuff now : You will NOT be graded by a program, unless explictly stated in the subject. Therefore, you are afforded a certain amount of freedom in how you choose to do the exercises. However, be mindful of the constraints of each exercise, and DO NOT be lazy, you would miss a LOT of what they have to offer !
- It's not a problem to have some extraneous files in what you turn in, you may choose to separate your code in more files than what's asked of you. Feel free, as long as the result is not graded by a program.
- Even if the subject of an exercise is short, it's worth spending some time on it to be absolutely sure you understand what's expected of you, and that you did it in the best possible way.
- By Odin, by Thor! Use your brain!!!

Chapter II

Objectives

In this project you will implement the various containers types of the C++ standard template library. For each container, turn in the appropriately named class files. The namespace will always be `ft` and your containers will be tested using ft::<container>. Each class must be in Coplien form. As a reminder, we are coding in C++98, so any new feature of the containers `MUST NOT` be implemented, but every old feature (even) deprecated is expected.

Chapter III

Mandatory part

- Implement the following containers, and turn in the necessary files `<container>.hpp`
- You must also provide a `main.cpp` that tests everything for your evaluations.
- the member function get_allocator is not expected here, the remaining member functions are. Non-member overloads are expected aswell.
- if your container has an iterator system, you must reimplent them.
- You can use http://www.cplusplus.com/ as a reference of which to implement.
- You cannot implement any more public functions than the ones offered in the standard containers. Everything else must be private or protected. Each public function/variable must be justificated.
- For non-member overloads, the keyword `friend` is allowed. Each use of `friend` must be justified and will be checked during evaluation.

You must turn in the following containers and its associated functions:

- List
- Vector
- Map
- Stack
- Queue

Of course, STL is forbidden. You are however allowed to use the STD library.

Chapter IV

Bonus part

If you completely finished the mandatory part, you can try and turn in the bonuses.

As a bonus, you can turn in the following containers and their associated functions:

- Deque
- Set
- Multiset
- Multimap