ppp3_novice_template

⚠️ **The blue text is clickable!**

⚠️ **The blue text is clickable!**

⚠️ **The blue text is clickable!**

中文 (额外地, 有全网最全的 VSCode 配置教程)

This is a template for novices learning *Programming: Principles and Practice Using C++ (3rd Edition)*. It requires no C++ or cmake experience.

Software Requirements

- Git
- a C++ IDE that supports CMake (latest Visual Studio, Qt Creator, CLion, etc.)

Download and unzip

1. Click the green `Code` button near the top of this page.
2. Click the `Download ZIP` button. This will download the latest repository as a zip file.
3. Unzip the downloaded zip file somewhere you are going to store your code.

Usage

1. Open your IDE (latest Visual Studio, Qt Creator, CLion, etc.) or *configured* Editors (VSCode with CMake Tools, etc.).
2. In your IDE, open this unzipped folder `as a folder` or `as a cmake project`.

How to add a new program?

Basics

The best thing about studying C++ with cmake is that a single project can manage multiple programs: you're not required to setup a new project in order to do the next exercise.

In this template, you can simply add a program by:

1. open `CMakeLists.txt` in the root folder.
2. add `add_program(<program_name> <source_file1> [source_file2...])`. For example,
    - `add_program(example_single src/example_single/main.cpp)` adds an executable named `example_single`, with its associated code file located at `src/example_single/main.cpp`. This code file contains an `int main()` function, which serves as the entry point for the program. After adding this line of `add_program`, we can use CMake to generate the program from the corresponding code and then execute the program.

    - `add_program(example_multiple src/example_multiple/main.cpp src/example_multiple/hello.cpp)` adds an executable named `example_multiple`, with its associated code files located at `src/example_multiple/main.cpp` and `src/example_multiple/hello.cpp`. Among these code files, there is only one `int main()` function, which serves as the entry point for the program. After adding this line of `add_program`, we can use CMake to generate the program from the corresponding code and then execute the program.
3. Reconfigure the project by using some button or reopening the IDE.

The headers used in book is configured correctly by default, just do the `add_program` step, then you can `#include "PPP.h"` or `#include "PPPheaders.h"` freely.

It's highly recommended to put your code inside `src` folder.

Headers

As for header files (`.h`, `.hpp`, etc.), you can simply put them together with source files. Then source files will be able to correctly `#include "<header_file>"`. For example, in `src/example_multiple` folder, `hello.cpp` can `#include "hello.hpp"` directly.

If you want to make a header file includable globally, you can put it inside `include` folder. For example, in `src/example_single` folder, `main.cpp` can `#include "add.hpp"` which is put inside `include` folder.

NOTE: In order to make Qt (in chapter 12-16) work correctly, you should also add header file paths in your `add_program` like source file paths.

`#include "PPP.h"` issues error?

Currently the module feature is not supported well. If your `#include "PPP.h"` issues error, you should use `#include "PPPheaders.h"` instead of `#include "PPP.h"`.

TLDR: If you're using clang 18+ as the compiler or the latest version of Visual Studio 2022 as the IDE, you might be lucky enough to successfully #include "PPP.h". But for now, you'll usually fail, and continuing to tinker with it will leave you physically and mentally exhausted.

click here to see current tools support for module. What's more,

- for MacOS homebrew clang users: There's a bug for homebrew clang with CMake's standard module library support, which requires your manual fix. After fix, you should remove the if condition in the FIXME part of `cmake/detect_std_module.cmake` file to enable module support for this project template.
- for clangd users (possibly using VSCode, Qt Creator, vim, etc.): Although clangd has supported module since 19, it hasn't supported it very well.

`QWidget: Must construct a QApplication before a QWidget`

In chapter 12-16, we use Qt as the graph library (see how to install it below).

However, the example code in the textbook fails and issues `QWidget: Must construct a QApplication before a QWidget`. Please check `src/example_gui/main.cpp` to learn how to fix it.

In a nutshell, you should put these lines of code around the example code:

```
int main() {
  using namespace Graph_lib;

  Application app;
  /* example code */
  app.gui_main();
}
```

Install Qt

Here I provide three ways to install Qt. After installation, You're able to use it in this project template directly.

(recommended) Download Qt installer and install it manually

1. Download Qt installer in this link.
2. Double click the downloaded Qt installer to install it.
3. Clear the CMake cache and reconfigure CMake in your IDE in some way (for example, you might delete the build or out folder and restart the software). Then you can use Qt in this project template.

For more details, see villevoutilainen/ProgrammingPrinciplesAndPracticeUsingQt.

Use vcpkg

⚠️ Please make sure the network, especially inside terminal, is accessible to github (for instance, in the "steam++ toolbox" app you can choose to speed up github access). If you're using a VPN, please enable "tun mode" or something alike.

Edit `CMakeLists.txt`, add a line `run_vcpkg()` between `include(fetch_project_options)` and `project(cpp_novice LANGUAGES CXX)`. That is:

```
cmake_minimum_required(VERSION 3.25)

list(APPEND CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake")
include(fix_msvc)
include(enable_cpp_module NO_POLICY_SCOPE)
include(cpp_novice_fetch_project_options)

run_vcpkg()
project(cpp_novice LANGUAGES CXX)
```

Clear the CMake cache and reconfigure CMake in your IDE in some way (for example, you might delete the build or out folder and restart the software). Then if you're lucky, the installation should have happened automatically.

Use conan

⚠️ Please make sure the network, especially inside terminal, is accessible to github (for instance, in the "steam++ toolbox" app you can choose to speed up github access). If you're using a VPN, please enable "tun mode" or something alike.

1. Install conan 2 somehow. For example, you can download it from the official website.
2. Similarly, add `run_conan()` between `include(fetch_project_options)` and `project(cpp_novice LANGUAGES CXX)`.
3. Clear the CMake cache and reconfigure CMake in your IDE in some way (for example, you might delete the build or out folder and restart the software).

If you're lucky, the installation should have happened automatically.

Install other third-party libraries

See README_install_thirdparty_libraries.

References

I learnt cmake mostly from *Modern CMake for C++*; my C++ learning map is listed in 学习大纲 (although the table of contents is in Chinese, almost all resources in it are in English).

What's more, this repository highly depends on aminya/project_options, which improves the CMake experience a lot.

For conan 2.0, the official documentation is helpful.

Details about this repository can be found in 对配置文件的解释.