ppp2_novice_template

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++ (2rd 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_code(<program_name> <source_file1> [source_file2...])` (for example, `add_code(example_single src/example_single/main.cpp)`).
3. Reconfigure the project by using some button or reopening the IDE.

The headers used in the book are configured correctly by default, just do the `add_code` step, then you can `#include "std_lib_facilities.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.

Install fltk

Here I provide two ways to install fltk.

Use vcpkg

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

```cmake
cmake_minimum_required(VERSION 3.25)

list(APPEND CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake")
include(fetch_project_options)

run_vcpkg()
project(cpp_starter LANGUAGES CXX)
```

Reopen your IDE. Then if you're lucky, the installation should have happened automatically.

Use conan

1. Install conan 2 somehow.
2. Similarly, add `run_conan()` between `include(fetch_project_options)` and `project(cpp_starter LANGUAGES CXX)`.
3. Reopen your IDE.

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++*.

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.