ppp3_novice_template

English

这是一个为新手学习 《Programming: Principles and Practice Using C++ (3rd Edition)》(《程序设计: 使用 C++ 的原理与实践 (第 3 版)》) 提供的模板. 使用它不需要任何 C++ 或 CMake 经验.

软件需求

你可以通过 Windows/MacOS/Linux 上 VSCode 配置 C++: Clang + Clang-based Tools + CMake + Conan 安装所有软件.

- Git
- 一个支持 CMake 的 C++ IDE (最新版的 Visual Studio, Qt Creator, CLion 等)

下载和解压

如果已经下载, 请无视这部分.

1. 点击接近网页顶部的绿色 `Code` 按钮.
2. 点击弹出的 `Download ZIP` 按钮. 这会将本仓库最新版代码下载为一个 zip 压缩文件.
3. 解压该压缩文件到你放置代码的地方.

使用

1. 打开你的 IDE (最新版的 Visual Studio, Qt Creator, CLion 等) 或 *配置好的* 编辑器 (VSCode 等).
2. 在 IDE 中, `按文件夹` 或 `按 CMake 项目` 打开解压的文件夹.

如何添加新的程序

基本使用

用 cmake 项目学习 C++ 最好的一点是, 一个项目就能管理多个程序: 你不必新建一个项目来进行下一个练习.

在本模板中, 你按以下步骤简单添加一个程序:

1. 打开根目录下的 `CMakeLists.txt`.
2. 添加 `add_program(<program_name> <source_file1> [source_file2...])`. 例如,
    - `add_program(example_single src/example_single/main.cpp)` 添加了一个可执行程序名为 `example_single`, 其相关的代码文件有 `src/example_single/main.cpp`. 该代码文件中有一个 `int main()` 函数, 它作为程序运行的入口. 自此我们就能通过 cmake 从对应的代码生成该程序, 而后执行该程序.
    - `add_program(example_multiple src/example_multiple/main.cpp src/example_multiple/hello.cpp)` 添加了一个可执行程序名为 `example_multiple`, 其相关的代码文件有 `src/example_multiple/main.cpp` 和 `src/example_multiple/hello.cpp`. 这些代码文件中仅有一个 `int main()` 函数, 它作为程序运行的入口. 自此我们就能通过 cmake 从对应的代码生成该程

序, 而后执行该程序.

3. 通过某些按键或重新打开 IDE, 来重新配置本项目.

本书中使用的头文件已经默认可用, 你只需通过 `add_program` 添加程序, 就能任意地进行 `#include "PPP.h"` 或 `#include "PPPheaders.h"`.

强烈建议将源文件放进 `src` 文件夹内.

头文件

至于头文件 (`.h`, `.hpp` 等), 你可以简单将它和源文件放在一起. 之后源文件就能正确地 `#include "<header_file>"`. 例如, 在 `src/example_multiple` 文件夹中, `hello.cpp` 文件可以直接 `#include "hello.hpp"`.

如果你想要让一个头文件可以被任意位置的源文件包含, 你可以将它放入 `include` 文件夹. 例如, 在 `src/example_single` 文件夹中, `main.cpp` 文件可以 `#include "add.hpp"`, 而 `add.hpp` 是放在 `include` 文件夹里的.

`#include "PPP.h"` 报错?

当前 module 特性没有得到很好的支持. 如果你 `#include "PPP.h"` 报错, 应该使用 `#include "PPPheaders.h"` 来代替 `#include "PPP.h"`.

太长别看: 如果你是用 clang 18+ 作为编译器, 或者用最新版本的 Visual Studio 2022 作为 IDE, 你可能会运气很好, 成功 `#include "PPP.h"`. 但就目前情况来看, 你通常会失败, 而继续折腾下去会让你身心俱疲.

点击这里可查看 module 当前的工具支持情况. 此外,

对于使用 MacOS homebrew clang 的人: CMake 为 homebrew clang 提供 module 支持时存在一个问题, 需要你手动修复. 修复后, 你还需要在 cmake/detect_std_module.cmake 中删除 FIXME 部分的条件判断, 从而为本项目模板启用 module 支持.
对于使用 clangd 的人 (可能是使用 VSCode、Qt Creator、vim 等): 虽然 clangd 从 19 版本开始支持 module, 但它的支持还不是很完善.

安装 Qt

此处我提供三种方式来安装 Qt. 安装好后, 你将能直接在本项目模板中使用它.

(推荐) 下载 Qt 安装包并手动安装

1. 在 此处 下载 Qt 安装包.
2. 双击下载出来的 Qt 安装包从而安装 Qt.

更多细节见于 villevoutilainen/ProgrammingPrinciplesAndPracticeUsingQt.

使用 vcpkg

编辑 `CMakeLists.txt`, 在 `include(fetch_project_options)` 和 `project(cpp_novice LANGUAGES CXX)` 之间添加一行 `run_vcpkg()`, 即:

```cmake
cmake_minimum_required(VERSION 3.25)

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

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

重新打开你的 IDE. 如果运气好, Qt 将会自动安装.

使用 conan

1. 以某种方式安装 conan2.
2. 类似地, 在 `include(fetch_project_options)` 和 `project(cpp_novice LANGUAGES CXX)` 之间添加一行 `run_conan()`.
3. 重新打开你的 IDE.

如果运气好, Qt 将会自动安装.

安装其他第三方库

见于 请读我_安装第三方库.

参考资料

我基本是通过 *Modern CMake for C++* 学习 CMake 内容.

另外, 本仓库非常依赖于 aminya/project_options, 它大量改善了 CMake 的使用体验.

对于 conan 2.0, 官方文档 已经足够有用.

本仓库的细节见于 对配置文件的解释.