ppp2_novice_template

⚠️ **蓝色文字是可以点击的!**

⚠️ **蓝色文字是可以点击的!**

⚠️ **蓝色文字是可以点击的!**

English

这是一个为新手学习 《Programming: Principles and Practice Using C++ (2rd Edition)》(《程序设计: 使用 C++ 的原理与实践 (第 2 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 "std_lib_facilities.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.

安装 fltk

此处我提供两种方式来安装 fltk.

使用 vcpkg

⚠️ 请确保你的网络——尤其是**终端 (terminal)** 的网络——可以访问 github, 例如 "steam++工具箱" 中可以选择加速 github 访问. 如果使用科学工具, 请开启 "tun mode" 或 "增强模式" 之类的选项.

编辑 `CMakeLists.txt`, 在 `include(fetch_project_options)` 和 `project(cpp_starter 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(fix_msvc)
include(fetch_project_options)

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

以某种方式清除 cmake 缓存并重新配置你 IDE 中的 cmake (例如, 你也许可以删除 build 或 out 文件夹, 并重启软件). 如果运气好, fltk 将会自动安装.

使用 conan

⚠️ 请确保你的网络——尤其是**终端 (terminal)** 的网络——可以访问 github, 例如 "steam++工具箱" 中可以选择加速 github 访问. 如果使用科学工具, 请开启 "tun mode" 或 "增强模式" 之类的选项.

1. 以某种方式安装 conan2. 例如, 你可以从官网下载它, 或参考 我的 VSCode C++ 配置教程.
2. 类似地, 在 `include(fetch_project_options)` 和 `project(cpp_starter LANGUAGES CXX)` 之间添加一行 `run_conan()`.
3. 以某种方式清除 cmake 缓存并重新配置你 IDE 中的 cmake (例如, 你也许可以删除 build 或 out 文件夹, 并重启软件).

如果运气好, fltk 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; 我的 C++ 学习路径列于 学习大纲.

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.