老夏课堂 C++编程规范

目录

规范的代码可以促进团队合作，规范的代码有助于提升代码的可读性，注释规范自动生成文档。

1. 头文件规则

通常每一个.cpp 文件对应一个.h 文件，当然也有特列，main 函数入口 cpp 文件没有对应的.h，只有纯虚函数的接口类只有.h 文件没有.cpp。

1.1. #define 防止多重包含

所有头文件都应该使用#define 防止头文件被多重包含，这个多重包含指的是一个 cpp 编译过程中不被多次包含，如果有多个 cpp 都调用，那这个文件还是会被包含多次，所有头文件中不要做定义。

命名格式如果是普通的业务逻辑类的代码比如 x_msg_task.h 的文件就直接定义 X_MSG_TASK_H,如果是做类库或者公用的库，把项目名称加在前面。

实例 在 x_msg_task.h 头文件中：

```
#ifndef X_MSG_TASK_H
#define X_MSG_TASK_H
//你的声明代码
#endif
```

1.2. 减少头文件依赖

使用前置声明（forward declarations）尽量减少.h 文件中#include 的数

量。

当一个头文件被包含的同时也引入了一项新的依赖（dependency），只要该头文件被修改 ， 代码就要重新编译。如果你的头文件包含了其他头文件，这些头文件的任何改变也将导致那些包含了你的头文件的代码重新编译。因此，我们宁可尽量少包含头文件，尤其是那些包含 在其他头文件中的。

比如用到 XThread 类的指针，在头文件中可以不#include 类文件，而是直接用 class XThread;的声明,但如果类中定义的是实体对象，那就必须要引入头文件。

在.h 中尽量不引用头文件，在.cpp 中引用。

1.3. 头文件中不导入命名空间

头文件中不调用 using 导入命名空间，调用是直接写命名空间比如 std::string str;

命名空间导入在.cpp 中做，因为你没法确认你的.h 会被什么样的文件调用，会不会参数命名冲突。

2. 类规则

2.1. 构造函数（Constructor）的职责

构造函数中只进行那些没有实际意义的初始化就是非业务逻辑的初始化，尽量在 Init()方法集中初始化业务逻辑数据。

2.2. 结构体和类（Structs vs. Classes）

仅当只有数据时使用 struct，其它一概使用 class。类和结构体的成员变量使用不同的命名规则。

2.3. 继承（Inheritance）

优先使用组合，如果必须使用继承的话，只使用 public 继承。

2.4. 多重继承（Multiple Inheritance）

尽量不用，如果必须要用，多重继承中只有一个基类是有实现的，其他的都只能是接口类，不然内部对象空间分配容易产生问题。

2.5. 接口（Interface）

接口是指满足特定条件的类，这些类以 I 开头（非必需）。

只有纯虚函数（"=0"）和静态函数；

2.6. 声明次序（Declaration Order）

在类中使用声明次序如下：public:、protected:、private:

每一块中，声明次序一般如下：

构造函数；

析构函数；

成员函数，含静态成员函数；

数据成员，含静态数据成员。

3. 格式规则

3.1. 空格还是制表位（Spaces vs. Tabs）

只使用空格，每次缩进 4 个空格

3.2. 括号格式

左右括号都单独起一行

```
if (true)
{
    //
}
```

4. 命名约定

4.1. 通用命名规则（General Naming Rules）

函数命名、变量命名、文件命名应具有描述性，不要过度缩写，类型和

变量应该是名词，函数名可以用“命令性”动词。

4.2. 文件命名（File Names）

文件名要全部小写，可以包含下划线（_） my_useful.cpp。

4.3. Class 类型命名（Type Names）

类型命名每个单词以大写字母开头，不包含下划线：MyExcitingClass、MyExcitingEnum。所有类型命名——类、结构体、类型定义（typedef）、枚举——使用相同约定，

4.4. 变量命名（Variable Names）

变量名一律小写，单词间以下划线相连，类的成员变量以下划线结尾，如 my_exciting_local_variable、my_exciting_member_variable_。

4.5. 函数命名（Function Names）

普通函数：

函数名以大写字母开头，每个单词首字母大写，没有下划线：

AddTableEntry()

成员存取函数：

```
class MyClass {
public:
...
int num_entries() const { return num_entries_; }
void set_num_entries(int num_entries) { num_entries_ = num_entries; }
private:
int num_entries_;
};
```

4.6.宏、枚举命名

使用全部大写+下划线

枚举名称属于类型，因此大小写混合：UrlTableErrors。

```
enum UrlTableErrors
{
    OK = 0,
    ERROR_OUT_OF_MEMORY,ERROR_MALFORMED_INPUT,
};

#define PI_ROUNDED 3.0
```

5. Doxygen 注释规则

5.1. 文件注释

文件注释通常放在整个文件开头。

项目注释

```
//////////////////////////////////////////////////////////////////////////
/// @mainpage 项目注释
/// @author 作者
/// @version 版本
/// @date 2019 年 07 月 10 日
//////////////////////////////////////////////////////////////////////////
```

文件注释

```
//////////////////////////////////////////////////////////////////////////
/// @file 文件名
/// @brief 简介
/// @details 细节
//////////////////////////////////////////////////////////////////////////
```

5.2. 类定义注释

```
//////////////////////////////////////////////////////////////////////////
/// @brief 类的简单概述
/// @details 类的详细概述
//////////////////////////////////////////////////////////////////////////
```

5.3. 常量/变量的注释

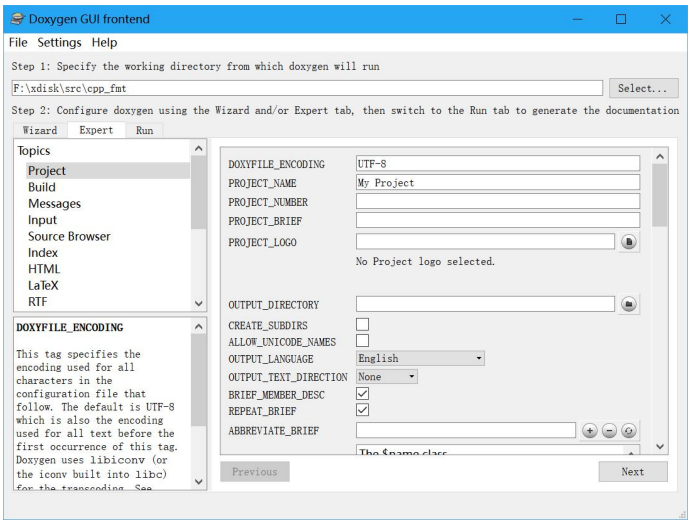
```
/// 代码前注释
常量/变量
常量/变量 ///< 代码后注释，一般是变量数量较多，并且名字短
```

5.4. 函数注释

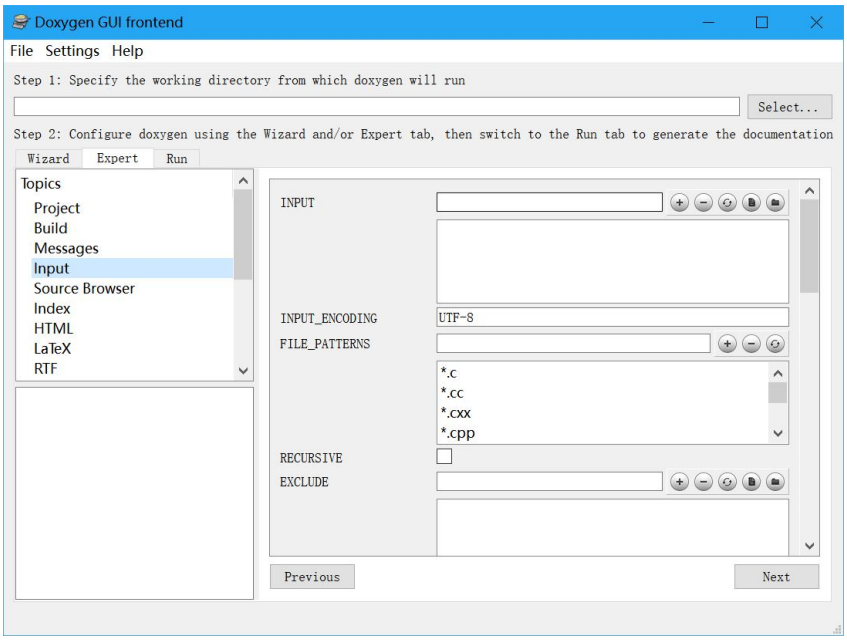
```
//////////////////////////////////////////////////////////////////////////
/// @brief 函数简介
///
/// @param 形参 参数说明
/// @param 形参 参数说明
/// @return 返回说明
//////////////////////////////////////////////////////////////////////////
```

5.5. Doxygen 设置使用

设定项目名称和项目语言



设定编码方式（INPUT_ENCODEING）vs 的项目一般设置为 GBK


Doxygen GUI frontend
File Settings Help
Step 1: Specify the working directory from which doxygen will run
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Topics
Project
Build
Messages
Input
Source Browser
Index
HTML
LaTeX
RTF
INPUT
INPUT_ENCODING
UTF-8
FILE_PATTERNS
*.c
*.cc
*.cxx
*.cpp
RECURSIVE
EXCLUDE
Previous
Next