C++ 代码规范文档

# 一、目的与范围

本规范旨在统一团队的 C++ 编程风格，提升代码的可读性、可维护性与可扩展性。

# 二、命名规范

## 变量命名

使用 camelCase，成员变量加前缀 m\_，全局变量加前缀 g\_  
示例：  
 int userAge;  
 int m\_userId;  
 int g\_maxUserCount;

## 函数命名

使用 camelCase，函数名应为动词+名词形式  
示例：  
 void loadUserData();  
 bool isLoggedIn();

## 类命名

使用 PascalCase，如 UserManager  
示例：  
 class UserManager;  
 class FileHandler;

## 枚举命名

枚举类型使用 PascalCase，枚举值加前缀  
示例：  
 enum class Status { STATUS\_OK, STATUS\_ERROR };  
 enum class LogLevel { LEVEL\_INFO, LEVEL\_WARNING };

## 常量命名

使用全大写加下划线，如 MAX\_LENGTH  
示例：  
 const int MAX\_BUFFER\_SIZE = 1024;  
 const double PI = 3.14159;

## 命名空间

使用项目或模块名，不使用缩写  
示例：  
 namespace ecommerce {  
 class CartManager;  
 }

# 三、代码风格

- 缩进：使用4个空格，不使用制表符  
- 空格：操作符前后添加空格  
- 大括号风格：使用 K&R 风格

void foo() {

if (condition) {

doSomething();

} else {

doSomethingElse();

}

}  
- 每行最大长度：不超过 100 字符  
- 注释：使用 // 进行单行注释，使用 /\* \*/ 进行多行注释，重要模块使用文档注释

# 四、设计与架构建议

- 使用 include guard  
- 引用顺序：先项目头文件，再第三方库，最后系统库  
- 类设计：成员变量私有化，提供 getter/setter  
- 资源管理：推荐使用智能指针  
- 禁止使用：goto、裸指针（尽量避免）

# 五、错误处理与异常

- 正常流程使用返回值，异常情况可使用异常抛出  
- 建议使用自定义异常类  
- 避免在析构函数中抛出异常

# 六、现代 C++ 特性

- 使用智能指针替代裸指针  
- 合理使用 auto 提高可读性  
- 使用范围 for 循环代替传统循环  
- 使用 lambda 表达式简化回调逻辑, 超过 **15行**，拆成普通函数  
- 使用 nullptr 替代 NULL

# 七、文件组织

- 每个类一个 .h/.cpp 文件  
- 头文件仅包含必要内容  
- 避免循环依赖，使用前向声明

# 八、文档与注释格式

- 采用 Doxygen 风格注释（/// 或 /\*\* \*/）  
- 类与函数必须有说明注释  
- 注释语言统一使用英文

/\*\*

\* @class ClassName

\* @brief 简短描述类的功能

\*

\* 详细描述类的作用、设计思路或使用场景。

\* 可以包含成员变量和主要方法的说明。

\*/

/\*\*

\* @brief 函数简短描述

\*

\* 函数的详细说明，包括功能、算法思路或注意事项。

\*

\* @param param1 参数1说明

\* @param param2 参数2说明

\* @return 返回值说明

\*

\* @exception 可能抛出的异常说明（如果有）

\*/