C++技术管理规范手册

1 文件组织规范

1.1 文件命名与结构

a） **强制**：每个.cpp 文件必须有对应的同名.h 文件（入口文件如 main.cpp 除外）12

a） **强制**：头文件使用.h 或.hpp 后缀，实现文件使用.cpp 后缀（禁止使用.cc、.cxx 等非标准后缀）210

b） **推荐**：文件名采用小写蛇形命名法（如 network_manager.cpp），反映核心类名 28

a） **强制**：文件开头必须包含版权声明和版本信息（公司名称、许可证类型、创建日期）110

cpp

复制

下载

```cpp
/*
 * Copyright (c) 2025, YourCompany
 * SPDX-License-Identifier: Apache-2.0
 * Created: 2025-03-15
 * Last Modified: 2025-04-20
 */
```

1.2 包含顺序与依赖

a） **强制**：头文件包含顺序为标准库头文件、第三方库头文件、项目内头文件，每组之间空一行 210

b） **推荐**：使用前置声明减少头文件依赖（适用于指针或引用类型成员）14

a） **强制**：禁止在头文件中使用 using namespace（防止命名空间污染）48

c） **允许**：在.cpp 文件中使用 using 声明引入特定符号（如 using std::vector;）5

示例包含顺序：

cpp

复制

下载

```cpp
// myclass.cpp
#include "myclass.h"  // 优先位置

#include <vector>     // C++标准库
#include <sys/time.h> // C 系统库

#include <boost/asio.hpp> // 第三方库

#include "utils.h"    // 项目内头文件
```

2 头文件管理规范

2.1 自包含原则

a） **强制**：所有头文件必须自包含（可独立编译），不依赖其他头文件隐式包含12

a） **强制**：使用`#define`保护防止多重包含（格式：`PROJECT_PATH_FILE_H`）12

b） **推荐**：优先使用`#pragma once`（非标准但获主流编译器支持）2

cpp

复制

下载

```cpp
// network_manager.h
#ifndef PROJECT_NETWORK_MANAGER_H
#define PROJECT_NETWORK_MANAGER_H
// ... 内容 ...
#endif
```

2.2 头文件内容

a） **强制**：头文件只存放声明（类、函数原型、`extern`变量），禁止存放定义（内联函数除外）14b） **推荐**：头文件内函数声明按功能分组，每组加注释块分隔10

a） **强制**：禁止头文件循环依赖（可通过依赖图工具检测）1

3 命名规范

3.1 通用命名规则

a） **强制**：标识符使用英文命名，禁止拼音或缩写（通用缩写如 TCP 除外）28

b） **推荐**：名称应体现语义（如 `CalculateDistance()`而非 `CalcDist()`）25

命名规范对照表：

标识符类型	强制规则	示例	参考来源
类/结构体	大驼峰式	`NetworkManager`	28
函数/方法	小驼峰式	`calculateDistance()`	25
变量	小写蛇形	`connection_count`	28
常量	k+大驼峰	`kMaxRetryCount`	25
枚举成员	全大写蛇形	`CONNECTION_ACTIVE`	28
命名空间	小写蛇形	`network_layer`	2

4 类设计规范

4.1 类结构设计

a） **强制**：遵循单一职责原则（SRP），类行数不超过 500 行（LCOM 值>1 需重构）36

b） **推荐**：成员声明顺序：`public` → `protected` → `private`（每段内按类型分组）58

a） **强制**：明确禁用拷贝构造/赋值（使用`=delete`），除非明确需要 58

cpp

复制

下载

```cpp
class NetworkConnection {
public:
  NetworkConnection();
  virtual ~NetworkConnection() = default;

  // 禁用拷贝
```

```
    NetworkConnection(const NetworkConnection&) = delete;
    NetworkConnection& operator=(const NetworkConnection&) =
   delete;

    // 允许移动
    NetworkConnection(NetworkConnection&&) noexcept;
    NetworkConnection& operator=(NetworkConnection&&) noexce
pt;

    // 公有方法
    void connect();

protected:
    // 受保护成员

private:
    // 私有成员
};
```

4.2 继承与多态

b） **推荐**：优先使用组合而非继承（除非满足 Liskov 替换原则）56

a） **强制**：多态基类的析构函数必须为 `virtual`（或 `final` 类明确标注）48

c） **允许**：多重继承仅限接口类（纯虚类）58

5 函数设计规范

5.1 函数结构与参数

b） **推荐**：函数长度不超过 50 行（非硬性，但需保持高内聚）410

a） **强制**：参数顺序：输入参数在前，输出参数在后（输入用 `const&`或值传递）8

b） **推荐**：避免使用默认参数（改用函数重载）58

a） **强制**：`const` 正确性：不修改成员的方法必须声明为 `const`8

cpp

复制

下载

```
// 良好函数设计示例
bool FindDevice(const std::string& device_id, // 输入参数
```

```
                DeviceInfo* out_info) {      // 输出参数
  // ... 实现 ...
}

// 不良设计：默认参数+输入输出混合
bool FindDevice(std::string device_id, DeviceInfo& info, i
nt retry_count = 3);
```

6 资源管理规范

6.1 内存管理

a） **强制**：动态分配资源必须使用智能指针（std::unique_ptr/std::shared_ptr）35

b） **推荐**：工厂函数返回 unique_ptr（明确所有权转移）5

a） **强制**：禁止使用 new/delete（除与遗留代码交互的特殊情况）59

cpp

复制

下载

```
// 正确资源管理示例
auto buffer = std::make_unique<uint8_t[]>(1024); // 自动管
理内存

// 禁止的做法
uint8_t* buffer = new uint8_t[1024]; // 需手动释放
```

6.2 并发与线程安全

b） **推荐**：优先使用不可变对象（Immutable Object）简化多线程设计 39

a） **强制**：可修改的共享数据必须使用互斥锁保护（std::mutex）6

c） **允许**：性能关键区可使用原子操作（std::atomic）替代锁 9

7 格式与风格规范

7.1 代码布局

a） **强制**：缩进统一为 2 或 4 个空格（项目内一致），禁止使用 Tab10

b） **推荐**：控制行宽在 100 字符以内（80 字符为理想值）28

a） **强制**：操作符两侧加空格（除一元操作符）10

cpp

复制

下载

```cpp
// 良好格式示例
if (connection_count > kMaxConnections) {  // 操作符两侧空格
  ResetAllConnections();                   // 缩进2 空格
}

// 不良格式
if(connection_count>kMaxConnections){  // 缺少空格
ResetAllConnections();}                // 括号位置混乱
```

7.2 控制结构

a） **强制**：if/for/while 等必须使用大括号（即使只有一行）10

b） **推荐**：复杂条件表达式分行对齐（逻辑运算符在行首）10

cpp

复制

下载

```cpp
// 良好控制结构示例
for (const auto& device : devices) {
  if (device.is_active()
      && device.supports_protocol(Protocol::IPv6)  // 条件分行
      && !device.is_locked()) {
    device.send_packet(packet);
  }
}
```

8 注释规范

8.1 注释内容与位置

b） **推荐**：函数注释说明前置条件、后置条件及异常行为（使用 Doxygen 格式）3

a） **强制**：禁止无意义的重复注释（如`// set value` 对应 `SetValue()`）36

c） **允许**：复杂算法可加行内注释（解释为何这样做而非如何做）6

cpp

复制

下载

```cpp
/**
 * 建立安全连接
 * @param timeout_ms 超时时间(毫秒)，必须>0
 * @return 是否成功建立连接
 * @throws NetworkException 当证书验证失败时抛出
 */
bool EstablishSecureConnection(int timeout_ms);
```

9 现代C++特性

9.1 特性使用规范

b) **推荐**：优先使用 auto（需保证可读性，避免滥用）39

a) **强制**：Lamba 表达式必须显式捕获（[=]或[&]仅限简单局部作用域）5

c) **允许**：模板元编程（TMP）限于基础库开发 5

cpp

复制

下载

```cpp
// 现代C++特性良好示例
auto iter = std::find_if(devices.begin(), devices.end(),
                         [device_id](const Device& d) {  // 显式捕获
                           return d.id() == device_id;
                         });
```

10 规范执行与工具支持

本规范的实施需要结合自动化工具和人工审查：

静态分析工具：配置 Clang-Tidy 规则集（覆盖 80%以上规范条目）3

持续集成：将规范检查纳入 CI 流水线（阻断违规合并）6

代码审查：重点审查类设计、资源管理、线程安全等核心领域 9

度量指标：定期监控代码库的 LCOM 值、重复率、圈复杂度等质量指标 6

规范级别说明：

a) **强制**：必须遵守，违反将导致 CI 失败和代码驳回

b） **推荐**：建议遵守，需团队负责人批准豁免

c） **允许**：根据上下文选择使用，保持项目内一致