## a、强制规则（必须遵守）

1.头文件必须使用#pragma once或头文件保护宏（如#ifndef \_FOO\_H\_）

2.源文件（.cpp）必须包含对应的头文件（.h）作为第一条包含语句

3.类名、结构体名使用大驼峰（如class MyClass）

4.函数名使用小驼峰（如void doSomething()）

5.变量名使用蛇形命名法（如int my\_variable）

6.常量使用全大写 + 下划线（如const int MAX\_VALUE = 100）

7.使用 4 个空格缩进，禁止使用 Tab

8.每行代码不超过 100 个字符

9.类成员变量后加下划线（如int value\_;）

10.左大括号不换行（如if (condition) { ... }）

11.禁止使用using namespace std;（需显式指定命名空间）

12.禁止使用malloc/free，必须使用new/delete或智能指针

13.虚析构函数必须声明为virtual

14.禁止使用 C 风格强制类型转换（如(int)x），使用 C++ 风格（如static\_cast<int>(x)）

15.使用std::unique\_ptr或std::shared\_ptr管理动态内存

16.禁止返回栈上对象的引用或指针

17.禁止使用裸指针进行内存管理（特殊场景需审批）

18.禁止使用未初始化的变量

19.禁止使用goto语句

20.禁止使用register关键字

21.禁止使用volatile进行多线程同步

22.代码必须兼容 C++17 及以上标准

23.禁止使用平台特定的 API（如 Windows-only 函数）

## b、推荐规则（建议遵守）

1.头文件应自给自足（包含所有必要的依赖）

2.优先使用前置声明替代#include

3.避免设计过大的类（单一职责原则）

4.使用= delete显式禁用拷贝构造函数和赋值运算符

5.使用override关键字标记重写的虚函数

6.函数参数超过 3 个时使用结构体封装

7.使用const修饰不修改参数的函数

8.返回值优先使用值语义（如std::string而非const char\*）

9.优先使用 STL 容器（如std::vector、std::map）

10.使用范围 for 循环替代传统 for 循环

11.使用算法库（如std::find、std::transform）替代手动循环

12.使用异常处理不可恢复的错误

13.使用assert检查内部不变量

14.返回错误码时优先使用std::error\_code

15.避免不必要的拷贝（使用移动语义或引用）

16.使用std::atomic替代锁实现轻量级同步

17.避免递归函数（可能导致栈溢出）

18.使用 Doxygen 风格注释（如/\*\* ... \*/）

19.为公共 API 提供清晰的文档注释

20.解释复杂算法或非直观代码

## c、允许规则（酌情使用）

1.预处理指令 允许使用#ifdef进行条件编译（如平台特定代码） 允许使用#define定义简单常量（避免复杂宏）

2.模板编程 允许使用模板元编程（需确保编译时间可控） 允许使用概念（C++20）约束模板参数

3.并发编程 允许使用std::thread、std::async进行多线程编程 允许使用互斥锁（std::mutex）和条件变量（std::condition\_variable）

4.智能指针 允许使用std::weak\_ptr解决循环引用问题 允许使用std::make\_unique和std::make\_shared创建智能指针

5.RAII 原则 允许自定义 RAII 类管理资源（如文件句柄、网络连接）

6.内联函数 允许使用inline优化短小函数（避免滥用）

7.命名空间 允许使用嵌套命名空间组织代码

允许使用匿名命名空间封装文件私有代码