强制

1.代码风格一致性  
整个项目必须保持一致的代码风格，使用.clang-format等工具统一格式化。

2.编译警告   
项目编译时应开启常见的编译警告，并尽可能将警告视为错误处理。

3.代码审查   
所有代码必须经过代码审查才能合入主分支。

4.头文件保护  
所有头文件必须使用#pragma once或标准的#ifndef保护。

5.禁止使用宏进行常量定义或函数定义  
使用const、constexpr、enum或内联函数代替宏。

6.使用命名空间  
除了在头文件中禁止使用using namespace在全局作用域，以避免命名污染。

7.明确的构造函数  
单参数构造函数必须使用explicit关键字。

8.使用override关键字   
重写虚函数时，必须使用override关键字。

9.使用智能指针

使用std::unique\_ptr、std::shared\_ptr等管理动态内存，避免手动new/delete。

10.禁止使用裸指针管理内存  
禁止使用裸指针管理内存，只能用于观察的场景。

11.资源获取即初始化  
所有资源都应当用对象来管理，利用构造函数获取，析构函数释放。

12.使用nullptr   
使用nullptr代替NULL或0表示空指针。

13.避免使用旧式类型转换   
使用static\_cast、const\_cast、dynamic\_cast、reinterpret\_cast代替旧式类型转换。

14.使用标准库  
优先使用C++标准库而非操作系统特定的API，除非有特殊需求。

15.线程安全   
必须明确类型是否线程安全。如果类型不是线程安全的，需要说明如何安全使用。

16.避免不必要的拷贝  
使用引用传递代替值传递，特别是对于大型对象。

17.断言  
使用assert进行程序内部错误的检查，在发行版中会被关闭。

18.不要忽略错误  
错误必须被处理，不能忽略。

19.禁止使用goto  
禁止使用goto语句。

20.魔法数字  
禁止在代码中直接使用魔法数字，应当使用命名常量。

21.初始化  
所有变量都应当初始化。

22.禁止使用过时特性  
禁止使用auto\_ptr、register、bool类型与非bool类型比较等过时特性。

推荐

23.单元测试  
新功能应当包含单元测试，修改旧代码也应更新对应的单元测试。

24.代码注释  
注释应当解释为什么而不是是什么，除非是什么并不明显。

25.包含头文件顺序  
头文件包含顺序：相关头文件、C库、C++库、其他库的头文件、项目内的头文件。

26.禁止使用C风格的字符串和函数  
使用std::string代替C风格字符串，使用C++的输入输出流代替C的输入输出函数。

27.禁用拷贝构造和赋值  
如果不希望类被拷贝，则应该明确禁用。

28.成员变量访问权限  
成员变量应设为私有，通过访问函数进行访问。如果只需要访问，提供公有getter而不提供setter。

29.虚函数的使用   
如果一个类被继承，且某函数希望被重写，应该声明为虚函数。析构函数如果需要被继承，则必须是虚函数。

30.类的小型化  
类应该尽量小，单一职责。

31.默认成员初始值  
在类定义中为成员变量提供默认初始值。

32.使用基于范围的for循环  
在遍历容器时，推荐使用基于范围的for循环。

33.使用auto关键字   
在类型明显或者冗长时使用auto，但不要用于基本类型以免降低可读性。

34.移动语义  
对于需要转移所有权的场景，使用移动语义。

35.使用constexpr  
在编译时能计算的常量，使用constexpr。

36.使用标准库线程   
使用std::thread、std::mutex、std::atomic等标准库设施编写多线程程序。

37.避免使用原始锁  
使用std::lock\_guard或std::unique\_lock管理锁，避免手动lock/unlock。

38.减少锁的作用域  
尽量减少锁的作用域，避免在持有锁的情况下调用用户代码。

39.内联函数   
内联函数应谨慎使用，通常只用于非常小的函数。

40.前置自增/自减   
对于迭代器和复杂类型的自增/自减，使用前置以避免临时对象。

41.枚举  
使用枚举代替宏定义常量，使用enum class代替旧式枚举。

42.类型别名  
使用using代替typedef定义类型别名。

43.变量作用域  
尽量缩小变量作用域，在需要时定义。

44.简化代码  
代码应尽可能简单、直接，避免过于复杂的表达式和嵌套。

允许

45.多重继承   
多重继承只允许用于接口类。

46.lambda表达式   
在需要局部函数对象时，可以使用lambda表达式，但避免过于复杂。

47.使用异常处理错误  
允许使用异常，但是注意不要在构造函数中抛出异常后导致资源泄漏

48.错误返回码  
也可以使用错误返回码，但不能忽略错误返回码。