**C++ 编码规范**

**一、强制规则（必须遵守）**

1. **文件结构**  
   a. 头文件必须使用 #pragma once 或头文件保护宏（如 #ifndef FILE\_H）。  
   b. 源文件（.cpp）和头文件（.h）必须使用相同的基名（如 example.h 和 example.cpp）。
2. **命名规范**  
   a. 类名、结构体名使用大驼峰（如 MyClass）。  
   b. 函数名、变量名使用小驼峰（如 myFunction、myVariable）。  
   c. 常量使用全大写加下划线（如 MAX\_VALUE）。  
   d. 禁止使用单字符命名（如 i, j 除外）。
3. **代码格式**  
   a. 使用 4 个空格缩进，禁止使用 Tab。  
   b. 左花括号 { 必须位于行尾，右花括号 } 单独一行（K&R 风格）。  
   c. 每行代码长度不超过 120 个字符。
4. **语言特性**  
   a. 禁止使用 using namespace std; 或其他命名空间。  
   b. 使用 nullptr 代替 NULL。  
   c. 禁止使用 goto 语句。  
   d. 禁止使用 C 风格强制类型转换（如 (int)x），应使用 static\_cast、dynamic\_cast 等。
5. **内存管理**  
   a. 使用智能指针（std::unique\_ptr、std::shared\_ptr）代替原始指针。  
   b. 禁止使用 new 和 delete 手动管理内存（特殊情况需审批）。
6. **安全性**  
   a. 禁止使用未初始化的变量。  
   b. 禁止使用 gets() 等不安全的函数。  
   c. 函数参数需检查有效性（如空指针、越界索引）。
7. **头文件**  
   a. 头文件必须自我包含（即包含所有必要的依赖）。  
   b. 禁止在头文件中定义全局变量。

**二、推荐规则（建议遵守）**

1. **文件结构**  
   a. 头文件按功能分组（如 core/, utils/ 目录）。  
   b. 源文件包含对应的头文件作为第一个包含项。
2. **命名规范**  
   a. 类成员变量使用 m\_ 前缀（如 m\_value）。  
   b. 私有成员函数使用 \_ 前缀（如 \_validateInput()）。
3. **代码格式**  
   a. 函数参数较多时，每个参数单独一行并缩进。  
   b. 逻辑相关的代码块用空行分隔。
4. **语言特性**  
   a. 使用 const 修饰不修改成员变量的成员函数。  
   b. 使用 auto 简化复杂类型声明（如迭代器）。  
   c. 优先使用 for 范围遍历（如 for (auto& item : container)）。
5. **内存管理**  
   a. 使用 RAII（资源获取即初始化）原则管理资源。  
   b. 优先使用栈上分配（如 std::vector 代替动态数组）。
6. **安全性**  
   a. 使用 assert 检查内部逻辑假设。  
   b. 避免使用 reinterpret\_cast。
7. **注释**  
   a. 类和函数必须有文档注释（如 Doxygen 格式）。  
   b. 复杂逻辑需添加行注释解释意图。

**三、允许规则（可根据场景选择）**

1. **文件结构**  
   a. 小型项目可将相关类放在同一个文件中。  
   b. 允许使用 .cc 作为源文件扩展名。
2. **命名规范**  
   a. 允许使用缩写（如 numEntries 代替 numberOfEntries），但需保持一致性。
3. **代码格式**  
   a. 允许在条件判断中省略括号（如 if (condition)）。
4. **语言特性**  
   a. 允许使用 C++11 及以上特性（如 lambda 表达式、移动语义）。  
   b. 允许使用 enum class 代替传统枚举。
5. **内存管理**  
   a. 允许在性能敏感场景使用原始指针，但需确保安全。
6. **性能优化**  
   a. 允许使用内联函数（inline）优化频繁调用的小函数。
7. **测试**  
   a. 允许使用单元测试框架（如 Google Test）编写测试用例。