C++语言的技术管理文档

（a.强制　　b.推荐　　c.允许）

a.

（1）缩进与空格的说明：缩进要使用 4 个空格。

（2）主动使用 const，避免使用宏

（3）规则：if、for、do、while、case、switch、default 等语句自占一行，且 if、for、do 、 while 等语句的执行语句部分无论多少都要加括号 {} 。

（4）除特殊情况外，所有代码都应置于命名空间中。命名空间名称应基于项目名及其路径保持唯一，禁止使用 using 指令，禁止使用内联命名空间

（5）将类的数据成员设为私有，除非它们是常量。这简化了不变量的推理，代价是必要时需编写简单的访问器。

（6） 避免使用裸指针，尽量使用智能指针，以便更好地管理内存，避免内存泄漏和悬空指针。

（7）遵循 RAII 原则（资源获取即初始化）。确保所有资源（如文件句柄、内存等）都在对象生命周期结束时自动释放。

（8）避免使用全局变量，全局变量难以跟踪和管理，容易造成依赖混乱。尽量通过函数参数或类成员传递数据。

（9）使用 enum class 而非传统 enum，以避免命名污染和隐式类型转换，增强类型安全性。

b.

（1）变量命名尽量使用与变量含义相关的英文单词，避免使用引起误解的词汇或模糊的缩写； 多个英文单词拼接而成的变量名单词与单词直接使用"_"进行分割。

（2）较短的单词可以省掉元音形成缩写，如：temp 可以写成 tmp;flag 可以写成 flg。

（3）值操作符、比较操作符、算术操作符、逻辑操作符、位域操作符，如“ =”、“ +=”、“ >=”、“ <=”、“+”、“ *”、“ %”、“ &&”、“ ||”、“ <<” 、“ ^” 等二元操作符 的前后应当加空格。

（4）一元操作符如“ !”、“ ~”、“ ++”、“ --”、“ &”等前后不加空格。

（5）像“ [ ] ”、“ .”、“ ->” 这类操作符前后不加空格。

（6）‘,’ 之后要留空格， 如 Function(x, y, z)。如果‘;’ 不是一行的结束符号， 其后也要留空格， 如 for (initialization; condition; update)

（7）用括号明确表达式的操作顺序，避免使用默认优先级。

（8）关键字之后要留空格。像 const、virtual、inline、case 等关键字之后至少要留一 个空格， 否则无法辨析关键字。像 if、for、while 等关键字之后应留一个空格再跟左括号‘（ ’，以突出关键字。

（9）函数之间要用空行分开， 相对独立的程序块之间必须要加空行

（10）函数名称要能表明函数的功能, 函数名之后不要留空格， 紧跟左括号’(’ ， 以与关键字区别。

（11）变量声明应尽可能靠近第一次使用处，避免一次性声明一组没有马上使用的变量；

（12）用空行将代码按照逻辑片断划分

（13）一行只写一个语句，如：不能写成：a = 2; b = 3; 正确写法是：

```
a = 2;
b =  3;
```

（14）程序块的分界符（如 C/C++语言的大括号‘{’和‘}’）应各独占一行并且 位于同一列，同时与引用它们的语句左对齐。
（15）要在适当的地方加上注释， 同时避免非必要的注释， 注释不要写的过多或过少， 一般控制在 20%~30%之间
（16）注释的主要目的应该是解释为什么这么做，而不是正在做什么。
（17）注释应与其描述的代码相近，对代码的注释应放在其上方或右方（对单条语句的 注释）相邻位置，不可放在下面，如放于上方则需与其上面的代码用空行隔开。
（18）注释符（包括/**/、//）与注释内容之间要用一个空格进行分隔
（19）只在函数短小（比如 10 行以内）时使用内联。
（20）包含头文件的名称和顺序： 相关头文件 > C 系统头文件 > C++标准库头文件 > 其他库头文件 > 本项目头文件
（21）把局部变量放进最小的作用域，并在声明时直接初始化。
（22）当元素可命名时，优先使用结构体而非 pair 或 tuple。
（23）优先使用组合而非继承。使用继承时，必须为公有继承。
（24）将相似的声明归类分组，并优先放置公开部分。
（25）在调用处的读者无需精确判断具体调用哪个重载就能理解代码意图时，才使用重载函数（含构造函数）。
（26）避免过多的继承层次结构，深度过深的继承树会导致代码复杂且难以维护。使用组合、接口或策略模式作为替代。
（27）避免在类的构造函数中执行复杂的逻辑或大量工作，构造函数应该尽量保持简单，以提高代码的可维护性和性能。
（28）尽量避免大量使用 goto 语句，goto 会导致代码逻辑难以跟踪和维护，使用结构化的控制流语句（如 if、for、while）代替。
（29）在使用多线程时，避免竞争条件。合理使用锁（如 std::mutex）和原子操作（如 std::atomic），避免死锁和数据不一致问题。

c.
（1）前置声明能不使用尽量不用，直接引入所需头文件。
（2）仅在常规语法（前置返回类型）不切实际或显著降低可读性时使用尾置返回类型。