1. 代码风格规范

代码风格规范的原则是：简单、易读、无二义性

| **风格** | **规范** |
| --- | --- |
| 缩进 | 4个空格，距离从可读性来说，正好 |
| 行宽 | 以前规定的80字符太小，现在可以限定为字符 |
| 括号 | 用括号清楚的表示逻辑优先级 |
| 断行与空白的{}行 | 每个“{”，“}”都独占一行，结构更清晰 |
| 分行 | 不要把多条语句、多个变量定义放在一行 |
| 命名 | （1）在变量名中不要提到类型或其他语法方面的描述 （2）避免过多的描述 （3）如果信息可从上下文中得到，则此类信息不必写入变量名中 （4）避免可要可不要的修饰词 |
| 下划线 | 分隔变量名字中的作用域标注和变量的语义 |
| 大小写 | 多个单词组成变量名，大小写适配得到，清晰易读 |
| 注释 | 解释程序做什么，为什么这样做 |

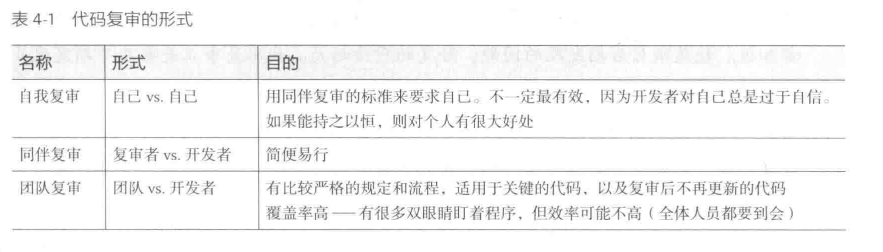
2、代码设计规范

代码设计规范不光是程序书写的格式问题，而且牵涉到程序设计，模块之间的关系，设计模式等方面，这里主要讨论通用的原则。

| **风格** | **规范** |
| --- | --- |
| 函数 | 只做一件事，并且要做好 |
| goto | 函数最好有单一出口，为达目的，可使用goto |
| 错误处理（参数处理） | 在Debug版本中，所有参数都要验证其正确性 |
| 错误处理（断言） | 检验正确性时，可以使用断言 |
| 类 | （1）使用类来封装面向对象的概念和多态 （2）避免传递类型实体的值，应该用指针 （3）对于有显式的构造和析构函数的类，不要创建全局的实体 （4）仅在必要时，使用“类” |
| class vs. struct | 如果只是数据的封装，用struct即可 |
| 公共/保护/私有成员 | 按照此顺序来说明类中的成员 |
| 数据成员 | （1）数据类型的成员用m\_name说明 （2）不要使用公共的数据成员，要用inline访问函数，这样才可兼顾封装和效率 |
| 虚函数 | （1）实现多态 （2）仅在很有必要时，使用虚函数 （3）若一类型实现多态，在基类中的析构函数应该是虚函数 |
| 构造函数 | （1）不做复杂操作 （2）不应返回错误 |
| new 和 delete | （1）尽量实现自己的new/delete （2）检查new的返回值 （3）释放指针时不用检查NULL |
| 运算符 | （1）无需自定义操作符 （2）不要做标准语义之外的动作 （3）运算符的实现必须非常有效率 （4）当打不定主意时，用成员函数 |
| 异常 | （1）不要将异常作为处理程序主要流程 （2）了解异常及处理异常的花销 （3）使用时，注意在什么地方清理数据 （4）异常不能跨过DLL或进程的边界来传递信息 |
| 类型继承 | （1）仅在必要时使用 （2）用const标注只读的参数 （3）用const标注不改变数据的函数 |

3、代码复审

正确定义：看代码是否在代码规范的框架正确地解决了问题。



基本手段：同伴复审

复审目的：

①找出代码错误

②发现逻辑错误，程序可以编译通过，但逻辑是错的

③发现算法错误，如使用的算法不够优化

④发现潜在错误和回归性错误

⑤发现可能需要改进的地方

⑥互相教育开发人员，传授经验