代码规范加强自查表（2023夏程序设计实践）	
文 件 结 构	
重要性	**审查项**
	头文件和定义文件的名称是否合理？
	头文件和定义文件的目录结构是否合理？
	版权和版本声明是否完整？
重要	**头文件是否使用了 ifndef/define/endif 预处理块？**
重要	**头文件中是否只存放“声明”而不存放“定义”**
程 序 的 版 式	
重要性	**审查项**
	空行是否得体？
	代码行内的空格是否得体？
	长行拆分是否得体？
重要	**“}” 是否独占一行并与“{”所在行对齐于同一列？**
重要	**一行代码是否只做一件事？如只定义一类变量，只写一条语句。**
	if、for、while、do等语句自占一行，不论执行语句多少都加“{}”。
	在定义变量（或参数）时，是否将修饰符*和＆紧靠变量名？
	注释是否清晰并且必要？
重要	**注释是否有错误或者可能导致误解？**
重要	**类结构的public，protected，private顺序是否在所有类的定义中保持一致？**
命 名 规 则	
重要性	**审查项**
	标识符是否直观且可以拼读？
	标识符的长度是否符合“min-length && max-information”原则？
重要	**程序中是否出现相同的局部变量和全部变量？**
	类名、函数名、变量和参数、常量的书写格式是否遵循一定的规则？
表 达 式 与 基 本 语 句	
重要性	**审查项**
重要	**如果代码行中的运算符比较多，是否已经用括号清楚地确定表达式的操作顺序？**
	是否编写太复杂或者多用途的复合表达式？
重要	**是否将复合表达式与“真正的数学表达式”混淆？**
重要	**是否用隐含错误或冗余的方式写if语句？例如 （1）将布尔变量直接与 TRUE、FALSE 或者 1、0 进行比较。 （2）将浮点变量用“==”或“！=”与任何数字比较。 （3）将指针变量用“==”或“！=”与 NULL比较。 （4）把“==”误写成“=”。**
	如果循环体内存在与循环次数无关的逻辑判断，并且循环次数很大，是否已经将逻辑判断移到循环体的外面？
重要	**case语句的结尾是否忘了加break？**
重要	**是否忘记写switch的default分支？**
重要	**使用goto语句时是否留下隐患？例如跳过了某些对象的构造、变量的初始化、重要的计算等。**
常 量	
重要性	**审查项**
	是否使用含义直观的常量来表示那些将在程序中多次出现的数字或字符串？
	在C++程序中，是否用const常量取代宏常量？
重要	**如果某一常量与其它常量密切相关，是否在定义中包含了这种关系？**
	是否误解了类中的const数据成员？const数据成员只在某个对象生存期内是常量，对于整个类而言却是可变的。
函 数 设 计	
重要性	**审查项**
	参数的书写是否完整？不要贪图省事只写参数的类型而省略参数名字。
	参数命名、顺序是否合理？
	参数的个数是否太多？
	是否使用类型和数目不确定的参数？
	是否省略了函数返回值的类型？
	函数名字与返回值类型在语义上是否冲突？
重要	**是否将正常值和错误标志混在一起返回？正常值应当通过地址传参获得，而错误标志用return语句返回。**
重要	**在函数体的“入口处”，是否用assert或其它方式对参数的有效性进行检查？**
重要	**是否滥用了assert？例如混淆非法情况与错误情况，后者是必然存在的并且是一定要作出处理的。**
重要	**return语句是否返回指向“栈内存”的“指针”或者“引用”？**
	是否用const提高函数的健壮性？const可保护函数的参数、返回值、函数体。“Use const whenever you need”
内 存 管 理	
重要性	**审查项**
重要	**用malloc或new申请内存之后，是否立即检查指针值是否为NULL？（防止使用指针值为NULL的内存）**
	是否忘记为数组和动态内存赋初值？（防止将未被初始化的内存作为右值使用）
重要	**数组或指针的下标是否越界？**
重要	**动态内存的申请与释放是否配对？（防止内存泄漏）**
重要	**是否修改“指向常量的指针”的内容？**

重要	是否出现野指针？例如 （1）指针变量没有被初始化就进行解引用。 （2）用free或delete释放了内存之后，忘记将指针设置为NULL，然后又访问这些内存。
重要	是否将malloc/free和new/delete混淆使用？
重要	malloc语句是否正确无误？例如字节数是否正确？类型转换是否正确？
重要	在创建与释放动态对象数组时，new/delete的语句是否正确无误（注意中括号）？
C++ 函 数 的 高 级 特 性	
重要性	**审查项**
	重载函数是否有二义性？
重要	是否混淆了成员函数的重载、覆盖与隐藏？
	运算符的重载是否符合制定的编程规范？
	是否滥用内联函数？例如函数体内的代码比较长，函数体内出现循环。
重要	是否用内联函数取代了宏代码？
类 的 构 造 函 数、析 构 函 数 和 赋 值 函 数	
重要性	**审查项**
重要	是否违背编程规范而让 C++ 编译器自动为类产生四个缺省的函数： （1）缺省的无参数构造函数；（2）缺省的拷贝构造函数；（3）缺省的析构函数；（4）缺省的赋值函数。
重要	构造函数中是否遗漏了某些初始化工作？
重要	是否正确地使用构造函数的初始化表？
重要	析构函数中是否遗漏了某些清除工作？
	是否错写、错用了拷贝构造函数和赋值函数？
重要	赋值函数一般分四个步骤： （1）检查自赋值； （2）释放原有内存资源； （3）分配新的内存资源，并复制内容； （4）返回 *this是否遗漏了重要步骤？
重要	是否正确地编写了派生类的构造函数、析构函数、赋值函数？注意事项： （1）派生类不可能继承基类的构造函数、析构函数、赋值函数。 （2）派生类的构造函数应在其初始化表里调用基类的构造函数。 （3）基类与派生类的析构函数应该为虚（即加virtual关键字）。 （4）在编写派生类的赋值函数时，注意不要忘记对基类的数据成员重新赋值。
类 的 高 级 特 性	
重要性	**审查项**
重要	是否违背了继承和组合的规则？ （1）若在逻辑上B是A的“一种”，且A的所有功能和属性对B而言都有意义，则允许B继承A的功能和属性。 （2）若在逻辑上A是B的“一部分”（a part of），则不允许B从A派生，而是要用A和其它东西组合出B。
其 它 常 见 问 题	
重要性	**审查项**
重要	数据类型问题： （1）变量的数据类型有错误吗？ （2）存在不同数据类型的赋值吗？ （3）存在不同数据类型的比较吗？
重要	变量值问题： （1）变量的初始化或缺省值有错误吗？ （2）变量发生上溢或下溢吗？ （3）变量的精度够吗？
重要	逻辑判断问题： （1）由于精度原因导致比较无效吗？ （2）表达式中的优先级有误吗？ （3）逻辑判断结果颠倒吗？
重要	循环问题： （1）循环终止条件不正确吗？ （2）无法正常终止（死循环）吗？ （3）错误地修改循环变量吗？ （4）存在误差累积吗？
重要	错误处理问题： （1）忘记进行错误处理吗？ （2）错误处理程序块一直没有机会被运行？ （3）错误处理程序块本身就有毛病吗？如报告的错误与实际错误不一致，处理方式不正确等等。 （4）错误处理程序块是“马后炮”吗？如在被它被调用之前程序已经出错。
重要	文件 I/O 问题： （1）对不存在的或者错误的文件进行操作吗？ （2）文件以不正确的方式打开吗？ （3）文件结束判断不正确吗？ （4）没有正确地关闭文件吗？