几点类设计原则

1.即使是一个抽象基类，如果它有非静态数据成员，也应该给它提供一 个带参数的构造函数，来初始化它的数据成员。或许你可以通过其派生 类来初始化它的数据成员（假如 nostatic data member 为 publish 或 protected）,但这样做的后果则是破坏了数据的封装性，使类的维护和修 改更加困难。由此引申，类的 data member 应当被初始化，且只在其构造函 数或其 member function 中初始化。

2.不要将析构函数设计为纯虚的，这不是一个好的设计。将析构函数 设计为纯虚函数意味着，即使纯虚函数在语法上允许我们只声明而不定义 纯虚函数，但还是必须实现该纯虚析构函数，否则它所有的继承类都将遇 到链接错误。一个不能派生继承类的抽象类有什么存在的意义？必须定义 纯虚析构函数，而不能仅仅声明它的原因在于：每一个继承类的析构函数 会被编译器加以扩展，以静态调用方式其每一个基类的析构函数（假如有 的话，不论是显示的还是编译器合成的），所以只要任何一个基类的析构 函数缺乏定义，就会导致链接失败。矛盾就在这里，纯虚函数的语法，允 许只声明而不定义纯虚析构函数，而编译器则死脑筋的看到一个其基类的 析构函数声明，则去调用它的实体，而不管它有没有被定义。

3.真的必要的时候才使用虚函数，不要滥用虚函数。虚函数意味着不 小的成本，编译很可能给你的类带来膨胀效应：

- 每一个对象要多负担一个 word 的 vptr。
- 给每一个构造函数（不论是显示的还是编译器合成的），插入一些代 码来初始化 vptr，这些代码必须被放在所有基类构造函数的调用之后， 但需在任意用户代码之前。没有构造函数则需要合成，并插入代码。
- 合成一个拷贝构造函数和一个复制操作符（如果没有的话），并插入 对 vptr 的初始化代码，有的话也需要插入 vptr 的初始化代码。
- 意味着，如果具有 bitwise 语意，将不再具有，然后是变大的对象、没 有那么高效的构造函数，没有那么高效的复制控制。

4.不能决定一个虚函数是否需要 const ，那么就不要它。

5.决不在构造函数或析构函数中使用虚函数机制。在构造函数中，每次 调用虚函数会被决议为当前构造函数所对应类的虚函数实体，虚函数机制并 不起作用。当一个 base 类的构造函数含有对虚函数 vf()的调用，当其派生类 derived 的构造函数调用基类 base 的构造函数的时候，其中调用的虚函数 vf() 是 base 中的实体，而不是 derived 中的实体。这是由 vptr 初始化的位置决定的 ——在所有基类构造函数调用之后，在程序员供应的代码或是成员初始化队列 之前。因构造函数的调用顺序是：有根源到末端，由内而外，所以对象的构 造过程可以看成是，从构建一个最基础的对象开始，一步步构建成一个目标 对象。析构函数则有着与构造相反的顺序，因此在构造或析构函数中使用虚 函数机制，往往不是程序员的意图。若要在构造函数或析构函数中调用虚函 数，应当直接以静态方式调用，而不要通过虚函数机制。