```
//（2）使用初始化列表的优势
//除了必须用初始化列表的场合，我们用初始化列表还有什么其他目的？ 有，就是提高程序运行效率。
//对于类类型成员变量xobj放到初始化列表中能够比较明显的看到效率的提升
//但是如果是个简单类型的成员变量 比如 int m_test,其实放在初始化列表或者放在函数体里效率差别不大：
//提醒：
```

```
//（1）何时必须用成员初始化列表
//a)如果这个成员是个引用
//b)如果是个const类型成员
//c)如果你这个类是继承一个基类，并且基类中有构造函数，这个构造函数里边还有参数。
//d)如果你的成员变量类型是某个类类型，而这个类的构造函数带参数时；
```

```
//（3）初始化列表细节探究
//说明：
//（3.1）初始化列表中的代码可以看作是被编译器安插到构造函数体中的，只是这些代码有些特殊；
//（3.2）这些代码 是在任何用户自己的构造函数体代码之前被执行的。所以大家要区分开构造函数中的
        //用户代码 和 编译器插入的 初始化所属的代码。
//（3.3）这些列表中变量的初始化顺序是 定义顺序，而不是在初始化列表中的顺序。
//老师 不建议 在初始化列表中 进行  两个 都在初始化列表中出现的成员之间的初始化
```