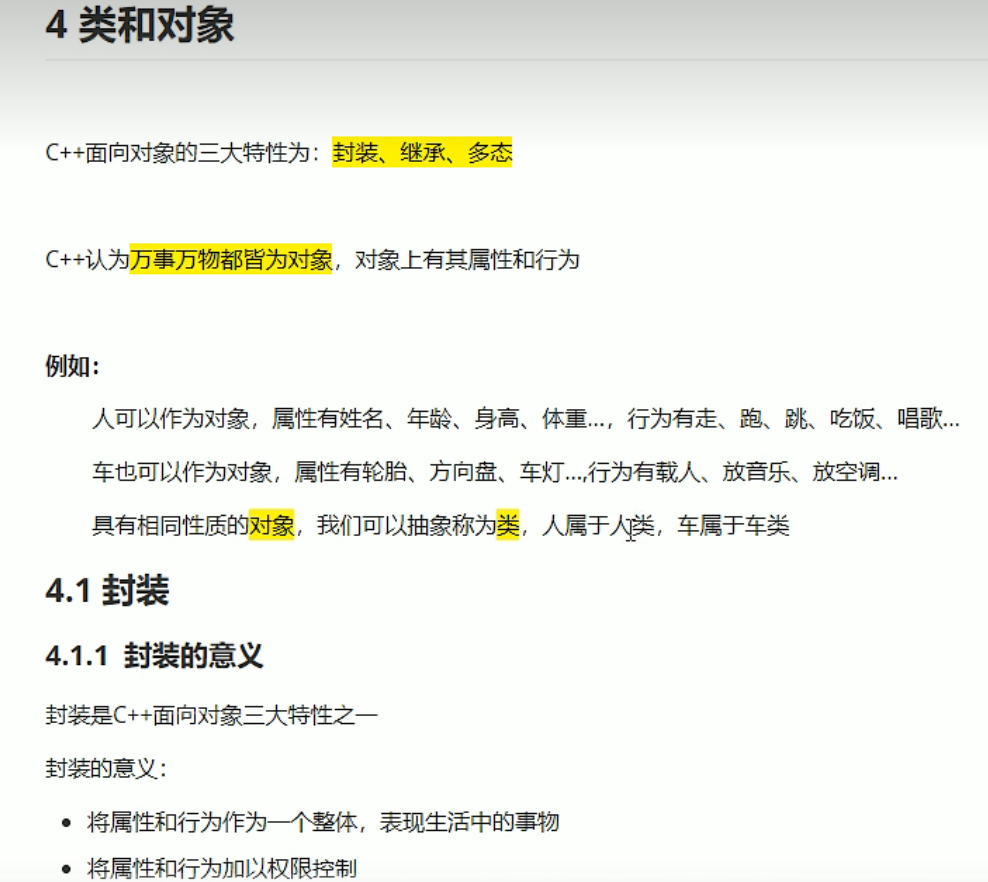
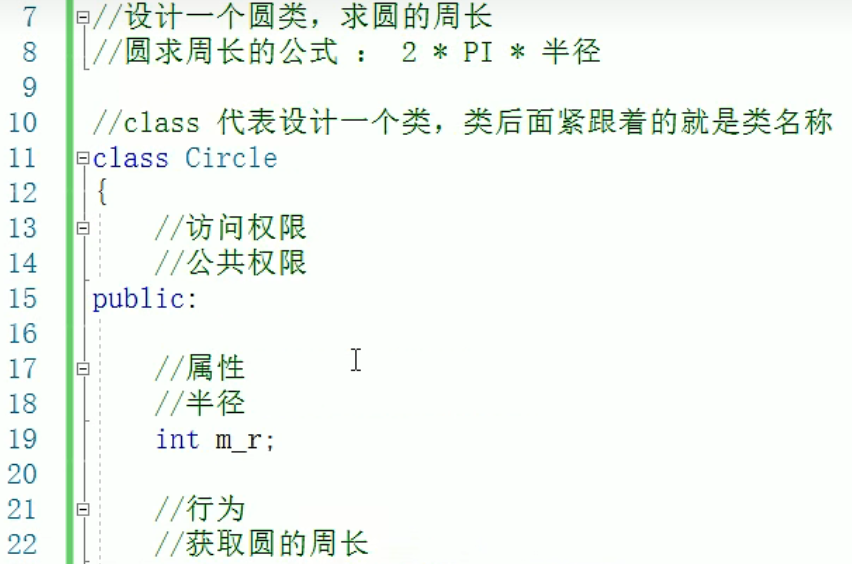
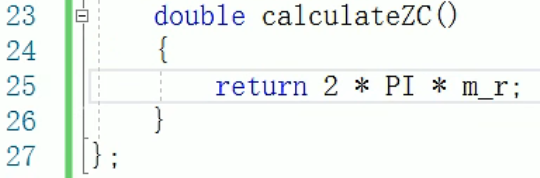
**类和对象**

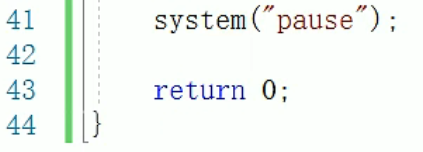
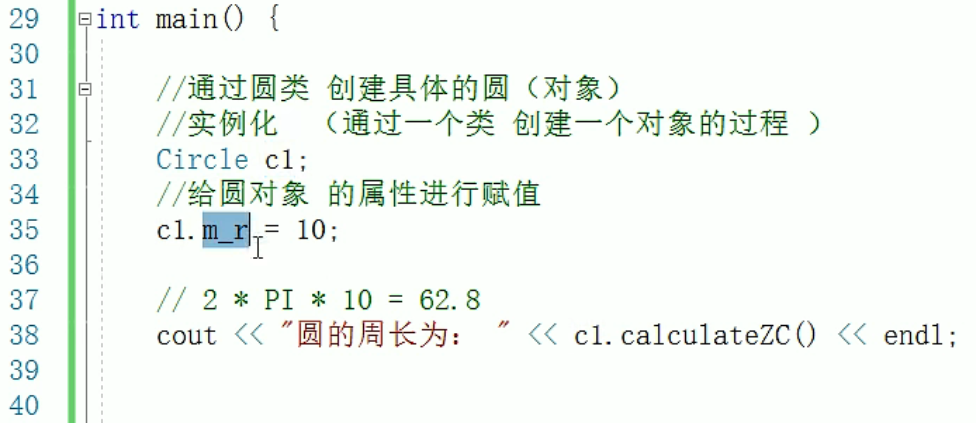
1.三大特性：封装 继承 多态



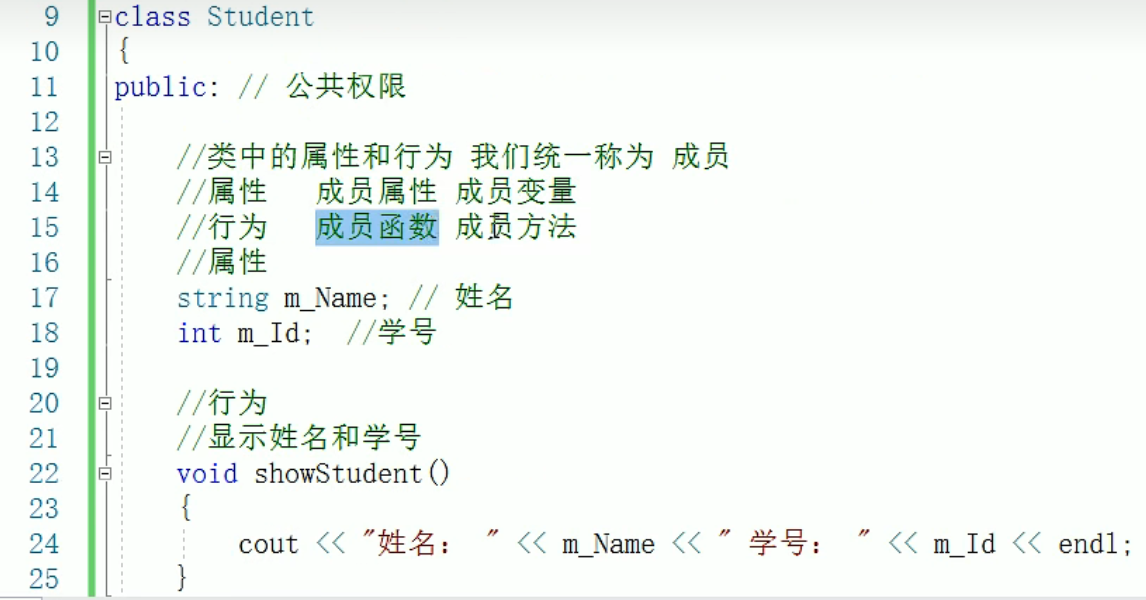
**简单写个例子1：**

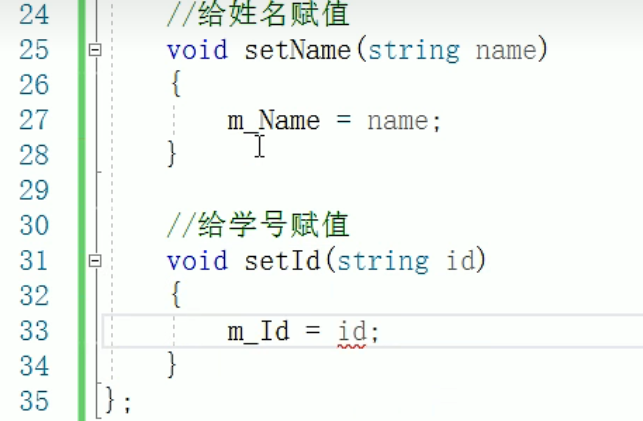


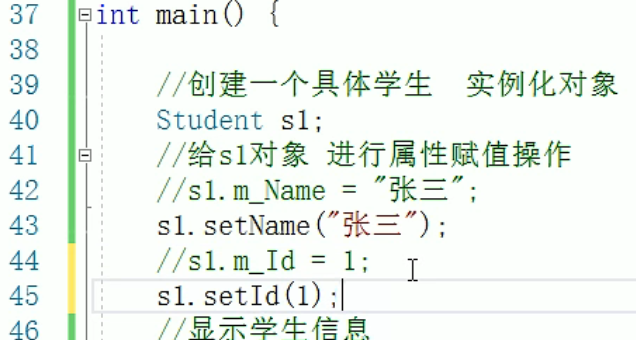




**例子2：**







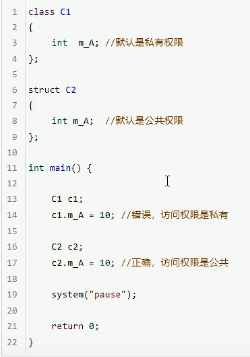
2.访问权限：公共 保护 私有



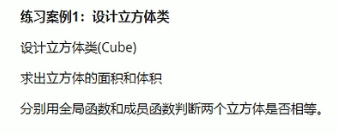
3.Struct 与 Class的区别

Struct 成员默认为公有权限

Class 成员默认为私有权限

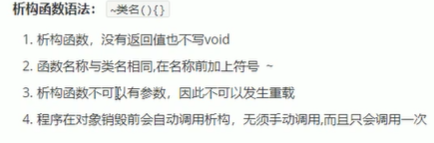
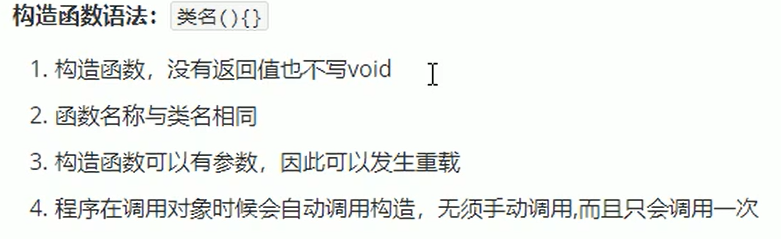


封装练习：



4. 构造函数和析构函数

4.1 构造函数的语法

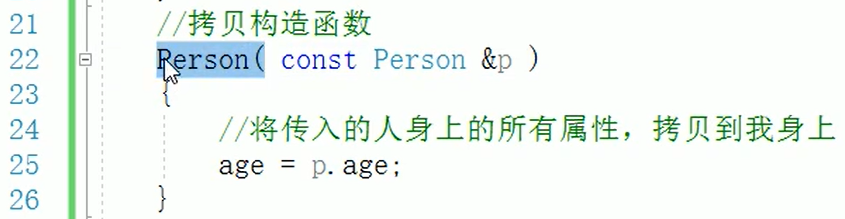


4.2构造函数的分类：

（1）按照参数分类



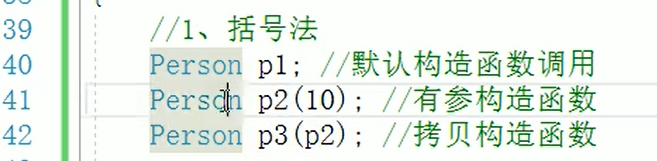
（2）按照类型：普通构造 和 拷贝构造



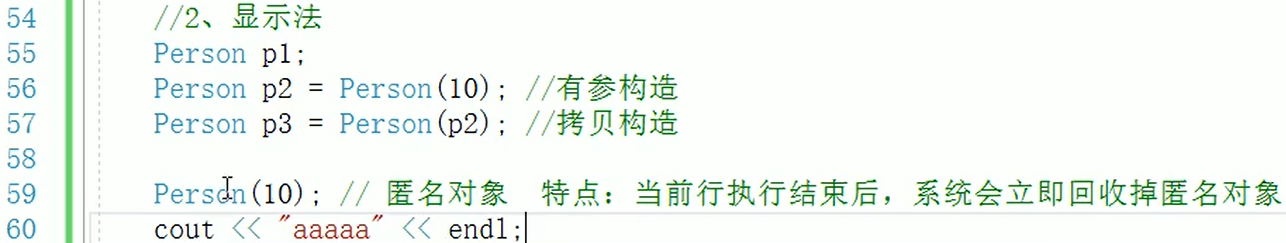
4.3构造函数的调用

（1）括号法(推荐使用)

注意： 默认构造的调用不加‘（）’，会被编译器认为是声明



（2）显示法



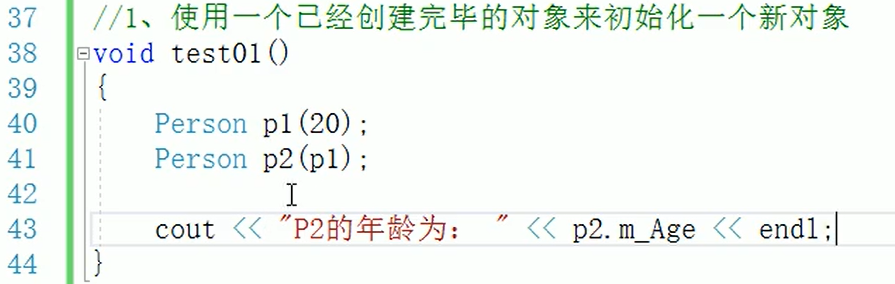


（3）隐式转换法



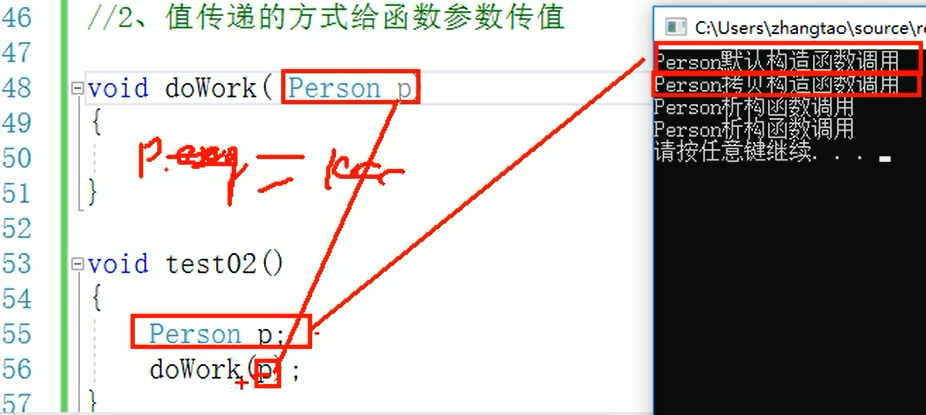
4.4 构造函数的调用时机

（1）使用已创建完毕的对象来初始化一个新对象

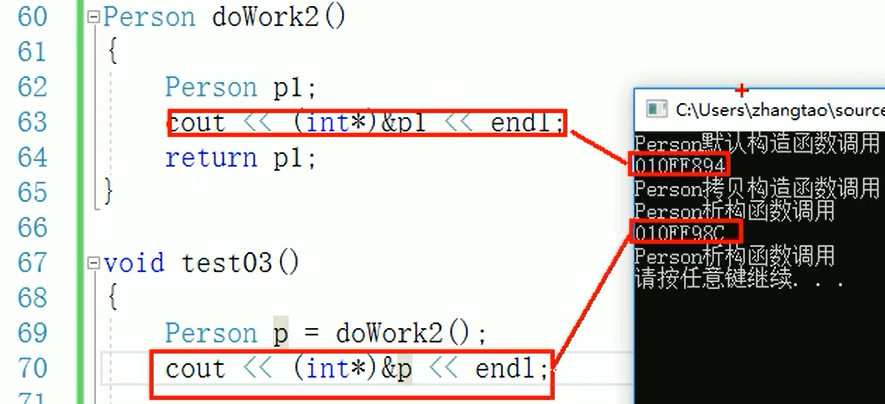


（2）值传递的方式给函数参数传值

传值，也是一份拷贝，修改上面的p（拷贝）不会影响下面的p

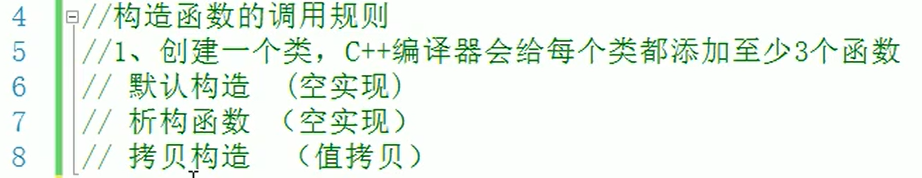


（3）值方式返回局部对象



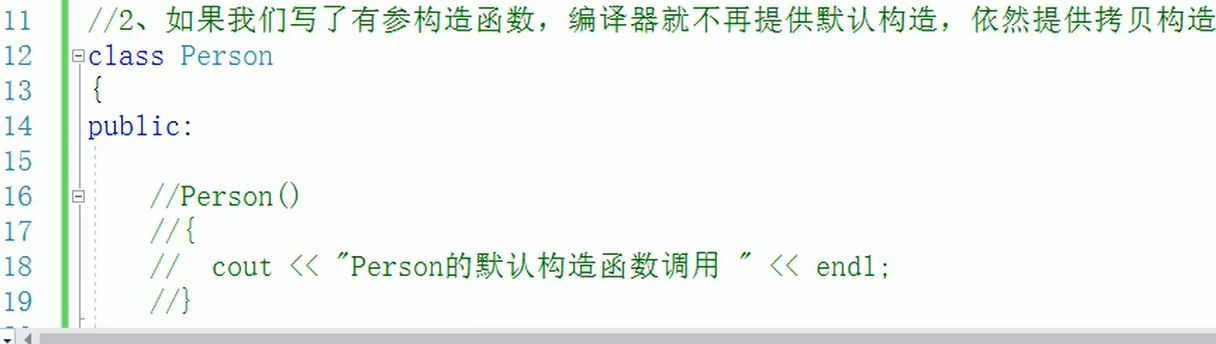
4.5 构造函数的调用规则

（1）默认情况下，C++会至少提供三个下面三个函数

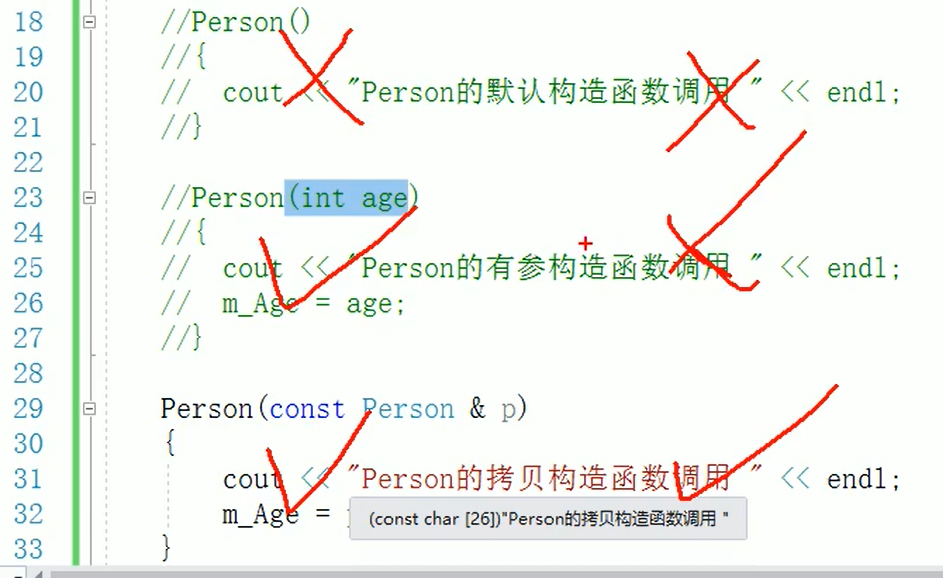


（2）如果写了有参构造，默认构造不再提供，但依然提拷贝构造

如果写了拷贝构造，则不在提供普通构造函数

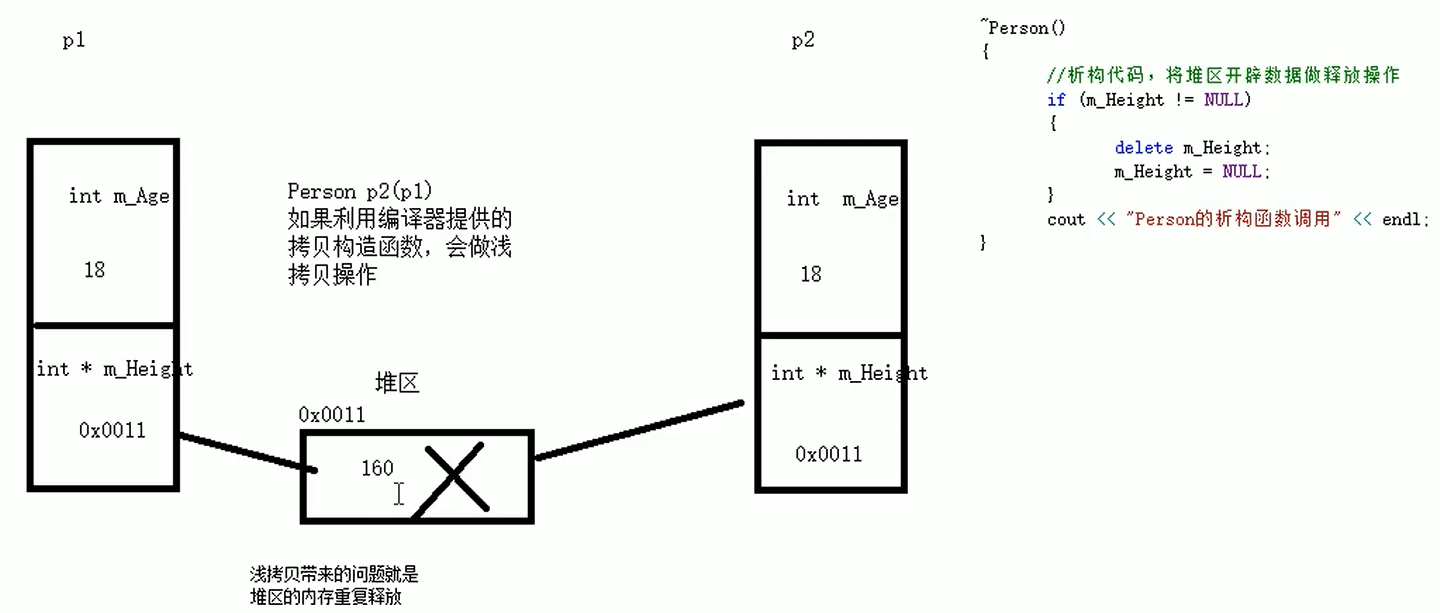


总结：

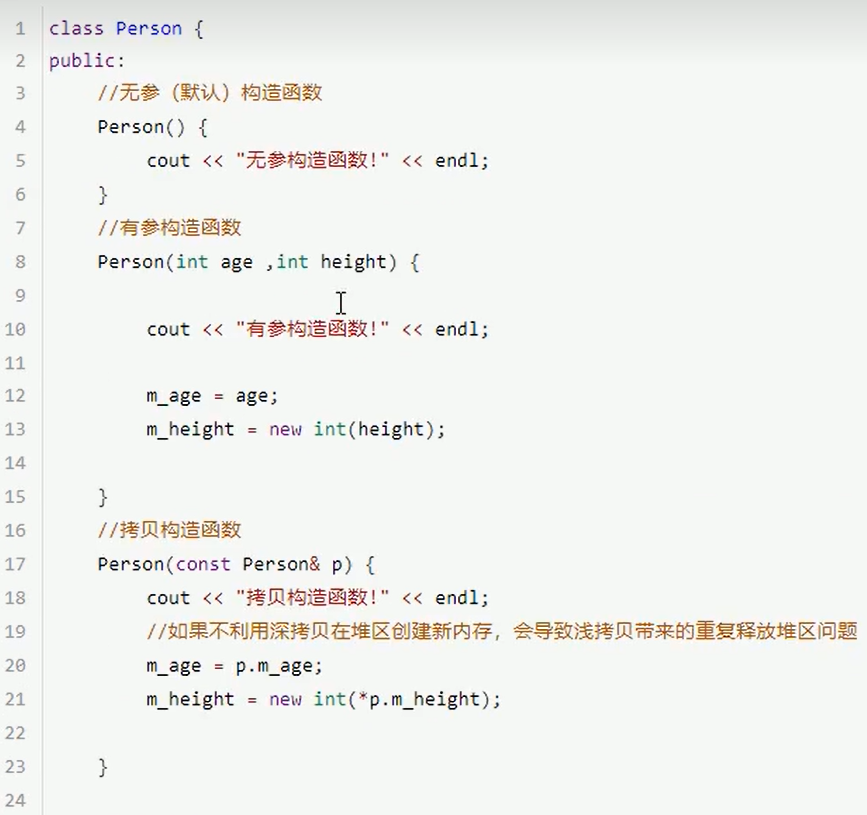


4.6 深拷贝与浅拷贝

（1）浅拷贝(编译器提供的默认拷贝构造)的问题：堆去的内存重复释放



（2）深拷贝：自己写拷贝构造函数

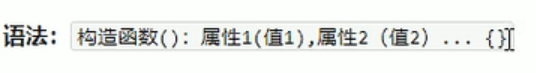
例子：  



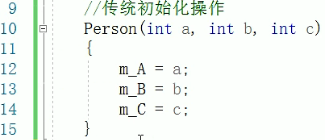

总结：如果属性又在堆区开辟的，一定要自己提供拷贝构造函数，防止浅拷贝带来的问题

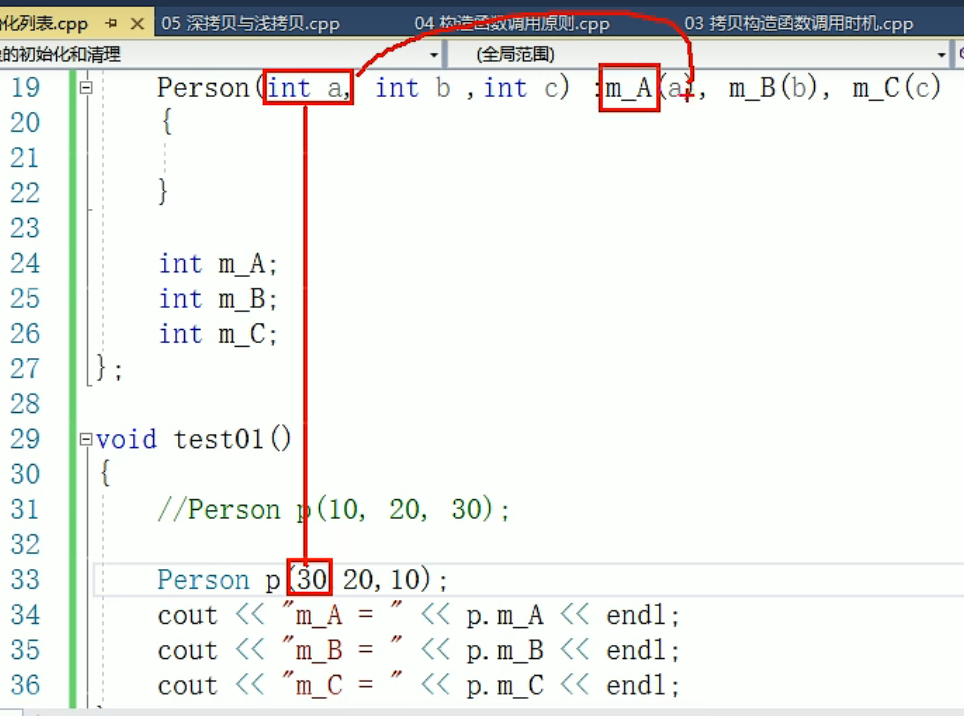
**4.7 初始化列表**

**（1）语法**



**（2）例子**



**.**

**4.8 类对象作为类成员**

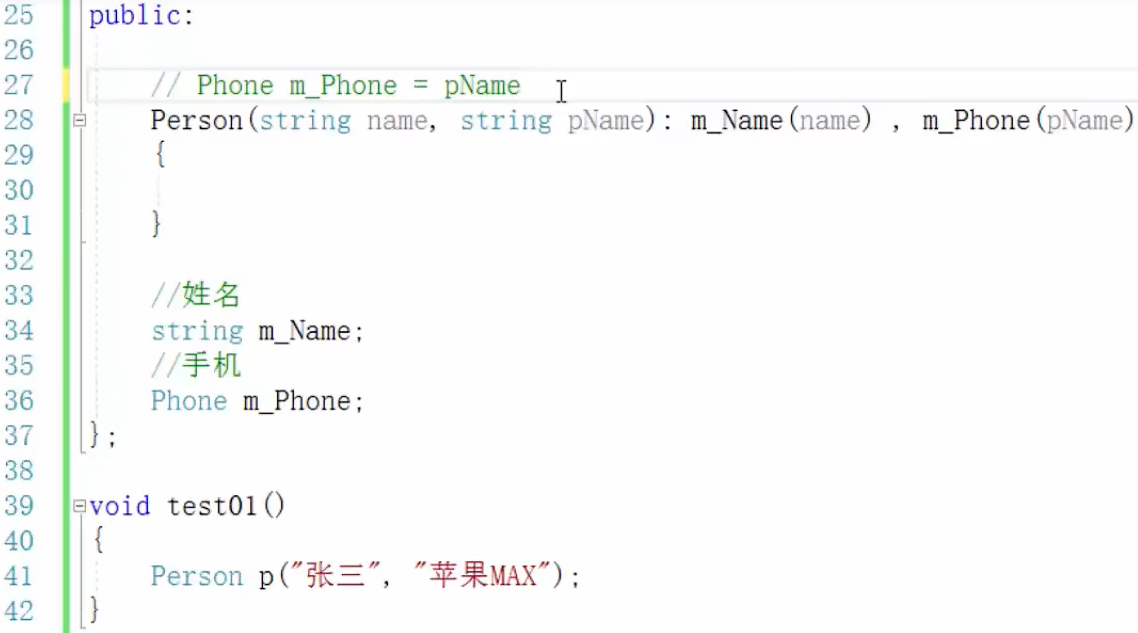
（1）类作为其他类的对象时，构造、造析构的先后顺序

先调用对象成员的构造，在调用本类构造；

析构的顺序则相反

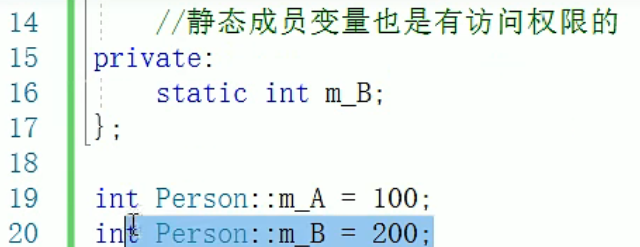
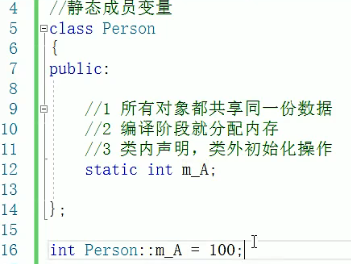


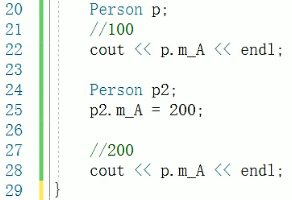




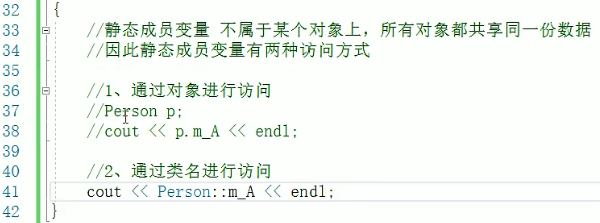
**4.9 静态成员与静态函数**

（1）静态成员特点

私有的是访问不到的



（2）静态成员访问方式



（3）静态成员函数：程序共享同一个函数；只能调用静态变量；

