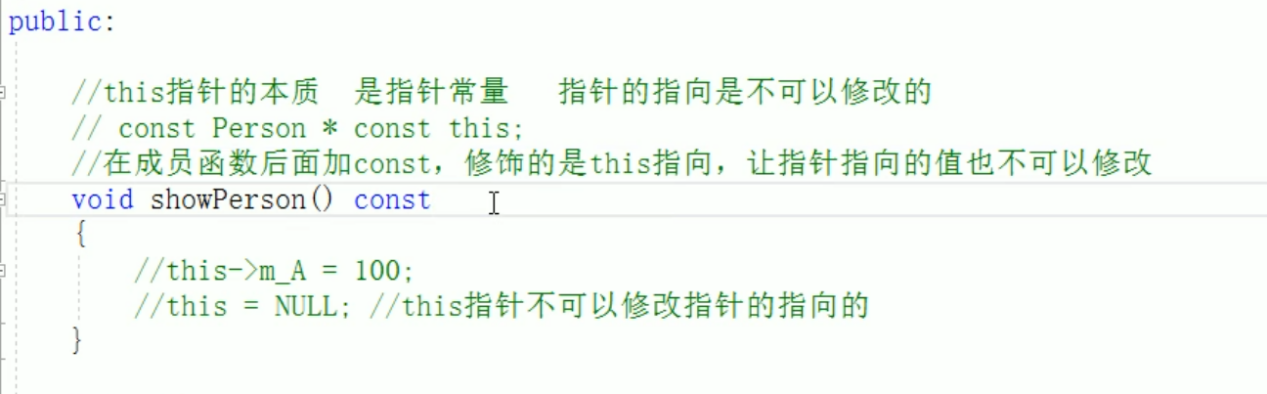
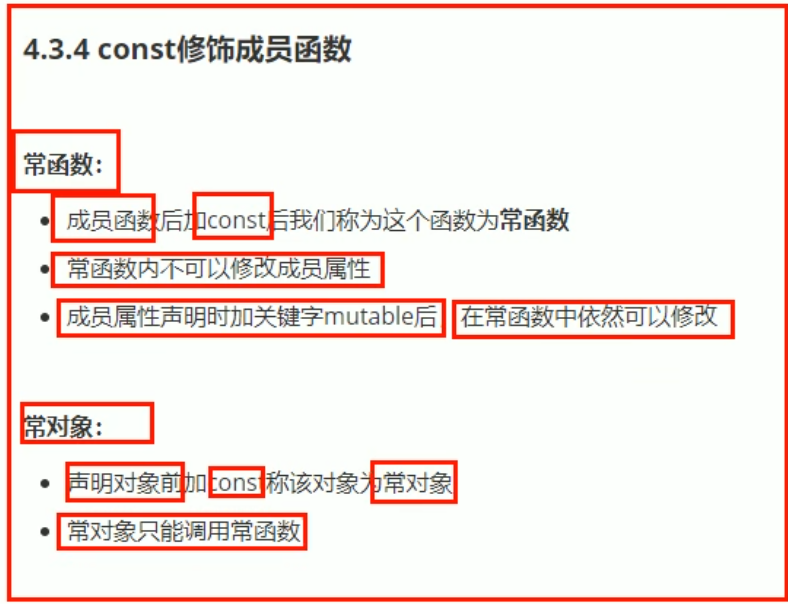
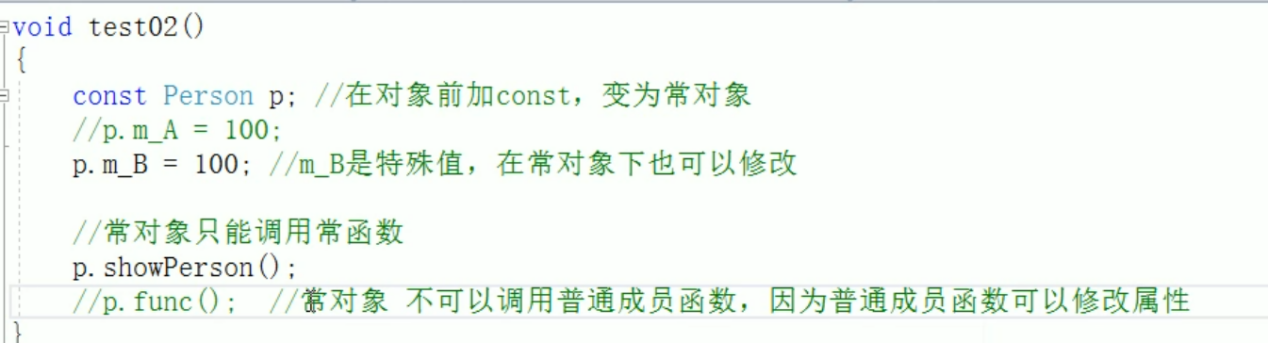
1. this指针与const修饰成员函数：

this指针的最大特性->指针的指向不可修改

而当const修饰成员函数又分为两种情况：一是在成员函数后面加const被称为常函数，效果如图；二是在声明对象前加const被称作常对象，而常对象只能调用常函数

1693824786802

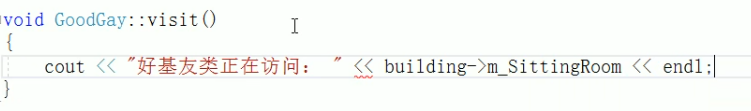
1. 友元：

友元的目的-让一个函数或者类访问另一个类中的私有成员

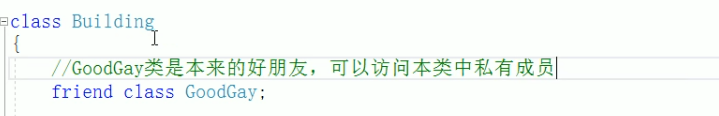
关键字为friend

以及三种实现：

1. 全局函数做友元

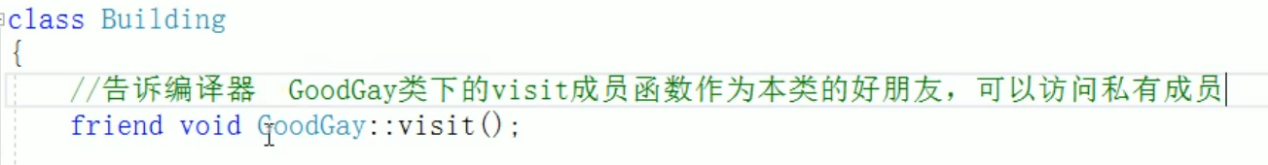


1. 类做友元



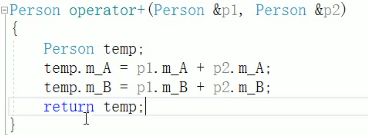


1. 成员函数做友元



1. 运算符重载

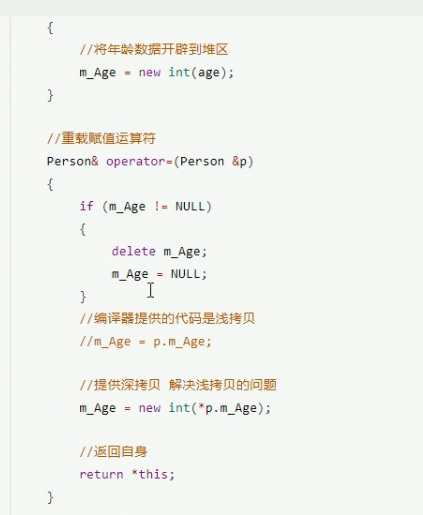
对算数运算符，赋值，自增运算符的重载方法：



//参数采用引用，而返回值直接返回是因为什么

--->将参数声明为引用是为了提高效率，如果按值传递p1，p2对象，代码功能都相同，但传递引用速度将更快，引用内存将更少

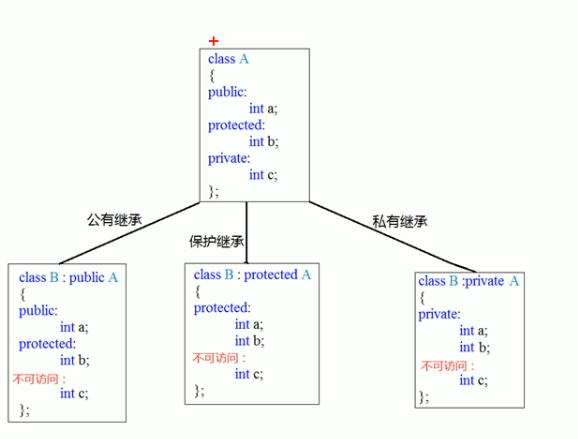
但返回值不能是引用，因为函数内部创建了一个temp的Person对象来表示另外两个对象之和，返回对象将创建对象的副本，然后调用函数可以使用它；但一旦返回的是引用的话，引用的将是temp对象，局部变量销毁后会被重新指向，即一个不存在的对象。

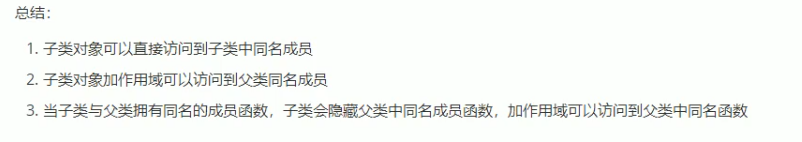




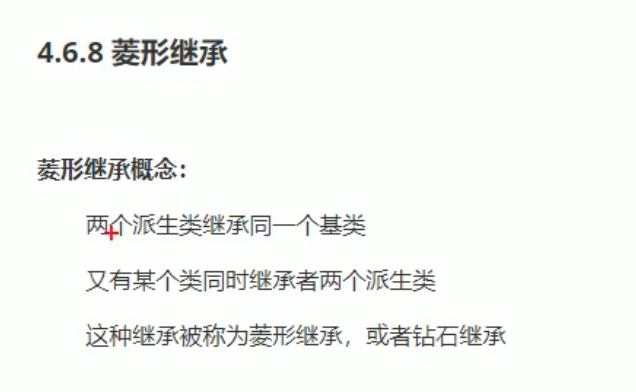
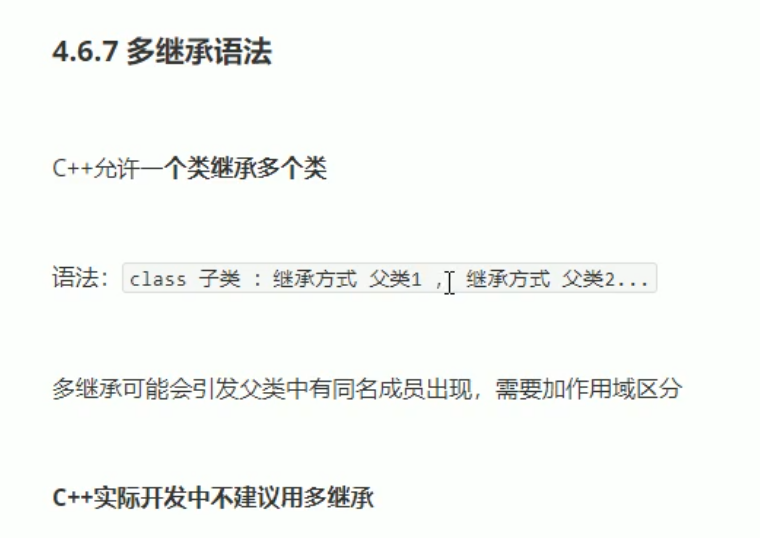
1. 继承：

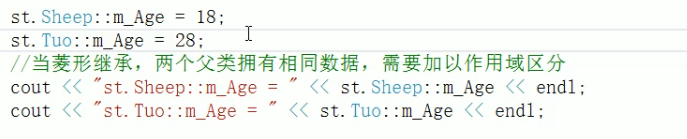
三种方式-公共继承，保护继承，私有继承











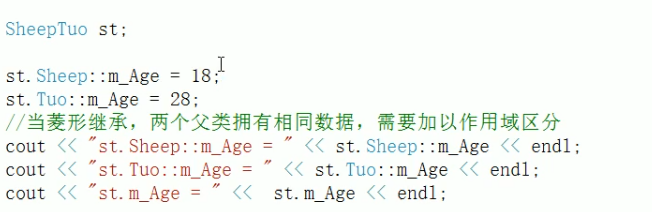
但对造成资源浪费（age存在两个空间存储），可以利用虚继承解决

->

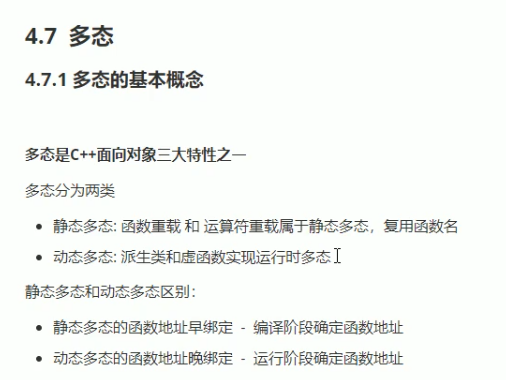


Virtual关键字同样可以解决指向不明的问题

->



5.多态



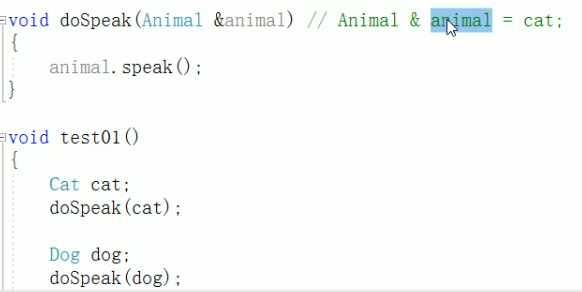
满足条件：

有继承关系

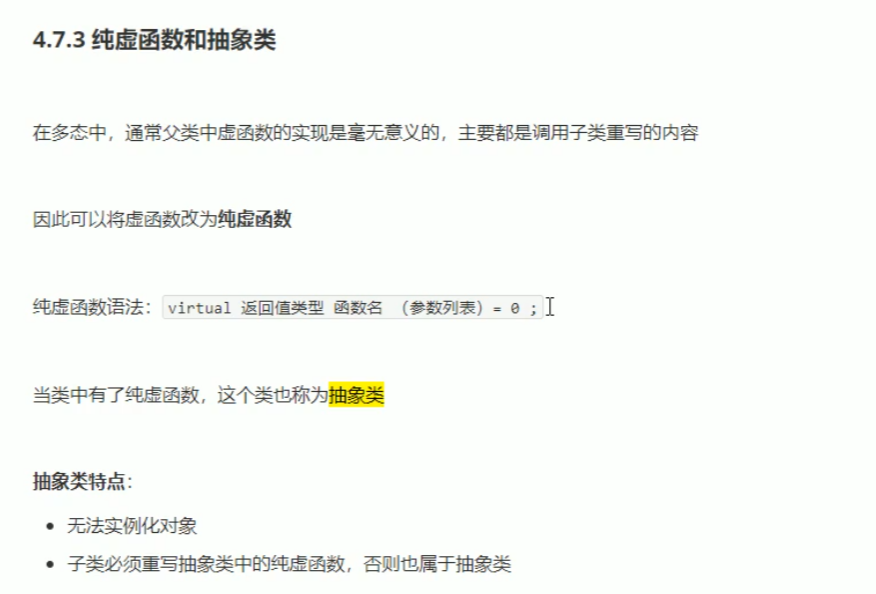
子类要重写（重写是需要函数返回值相同，函数名，参数列表完全相同）父类的虚函数（父类里需要加关键字vitrual，子类可加可不加）

动态多态的引用：

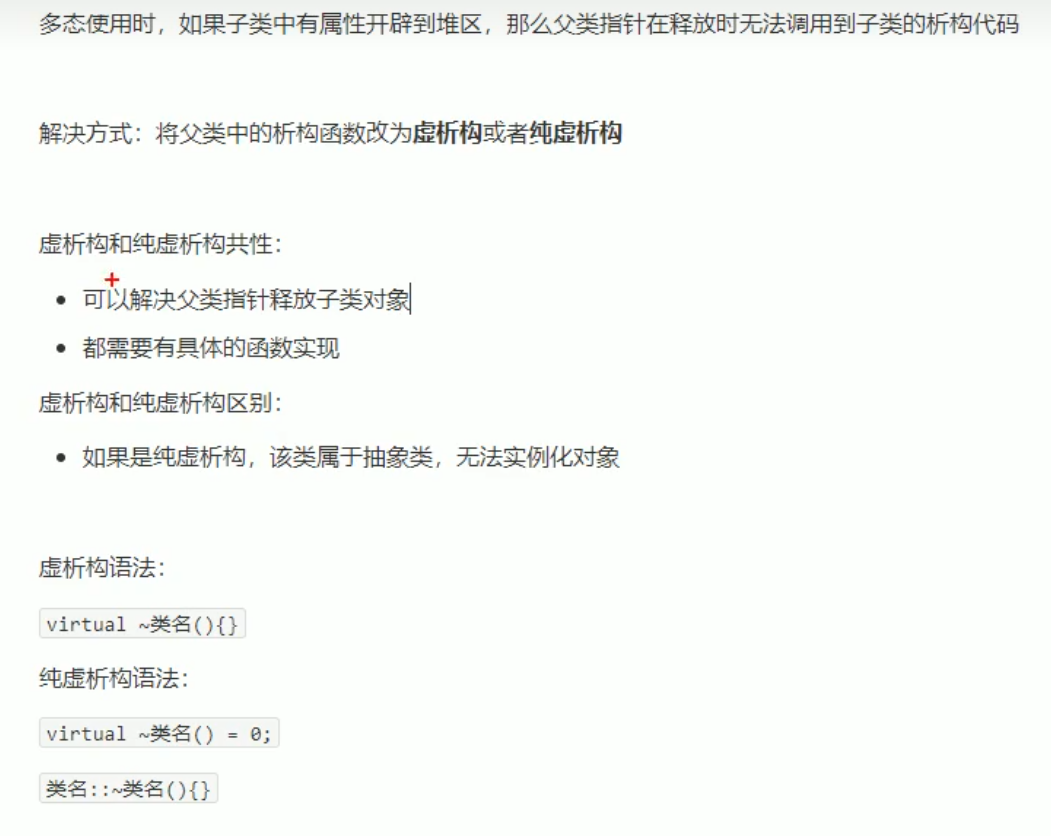
父类的指针或者引用 执行子类对象

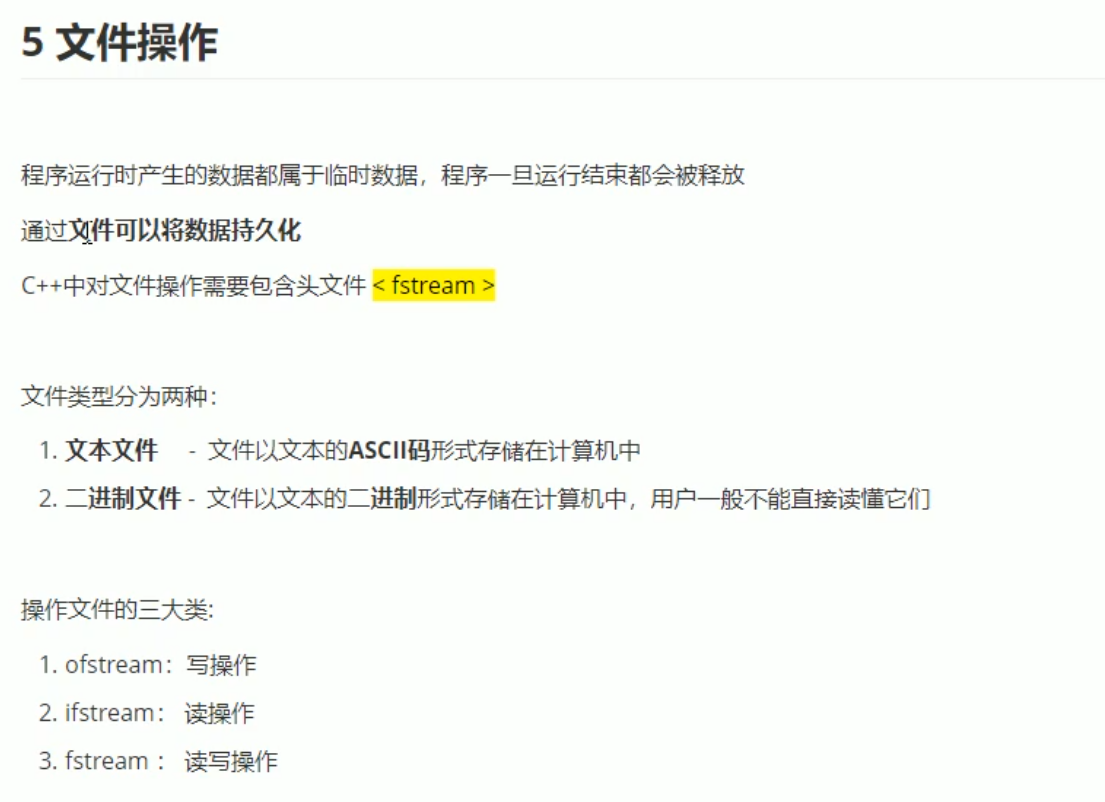


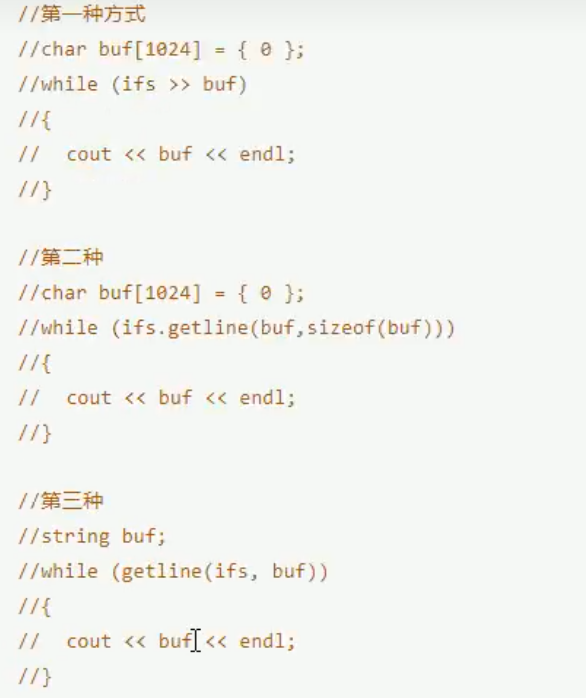
--->将输出cat/dog内定义的speak函数



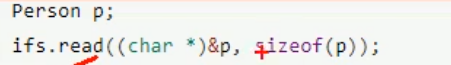
虚析构和纯虚析构



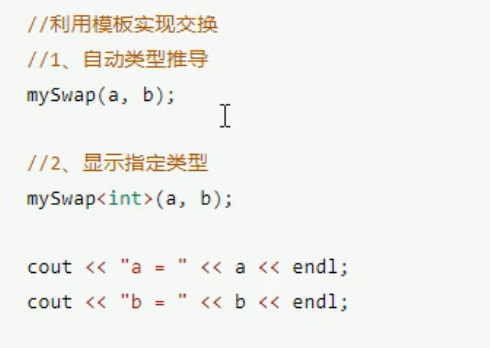


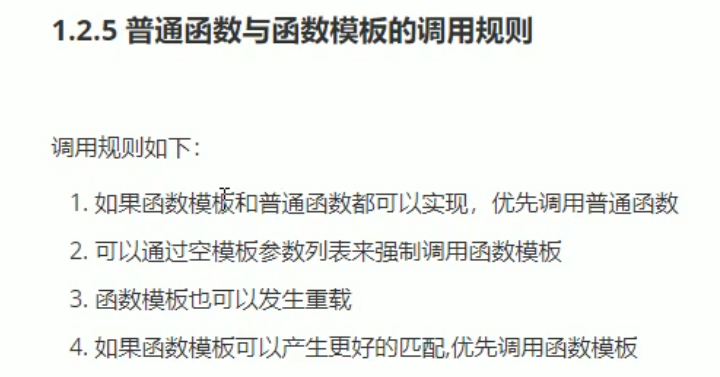


读文件：





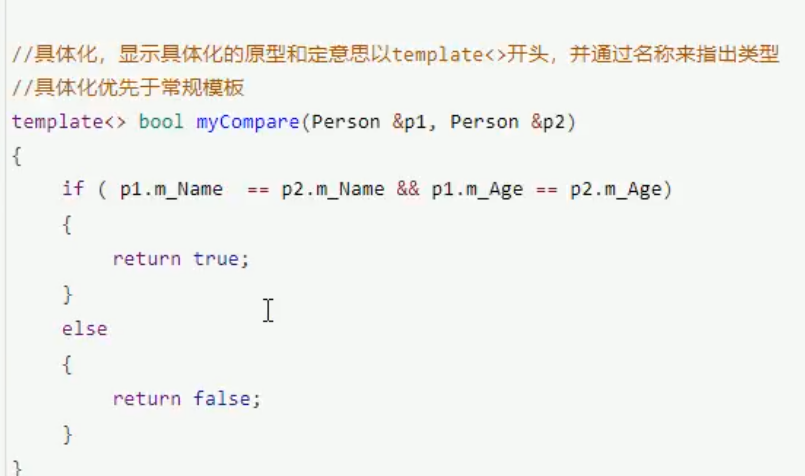


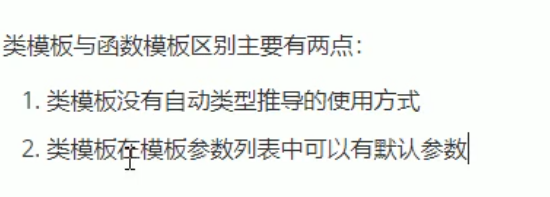


常规模板的局限性

像是传递数组无法满足交换判断等

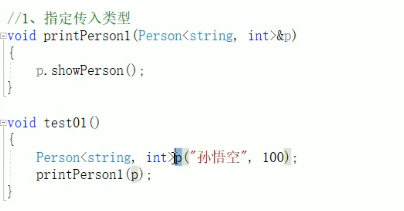
以及自定义类元素无法进行比较

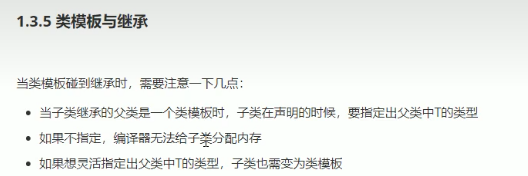


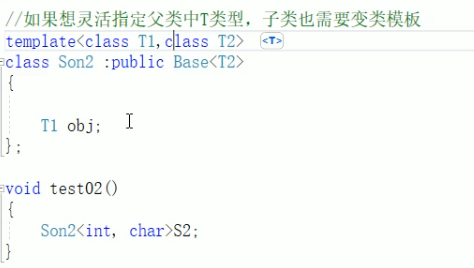


类模板对象做函数参数

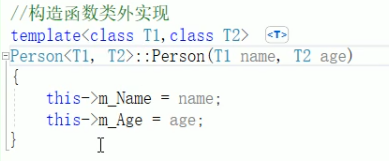
最常用的方式：

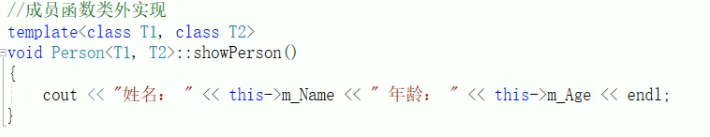


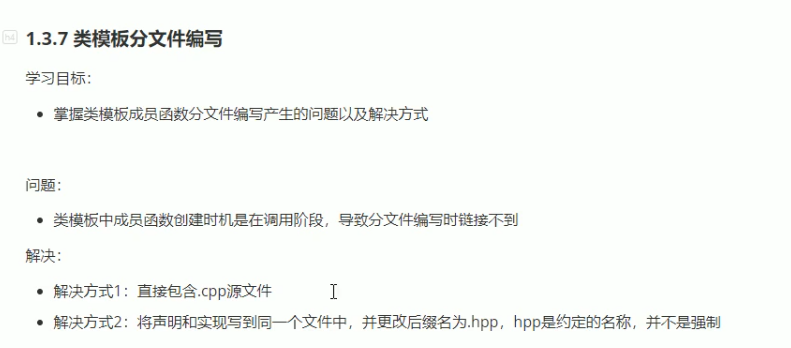




类外实现：







STL

