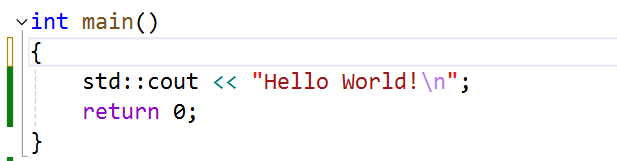
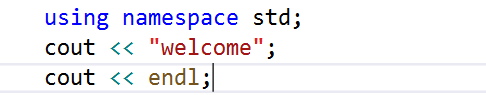
**Std::cout是标准的写法**

****

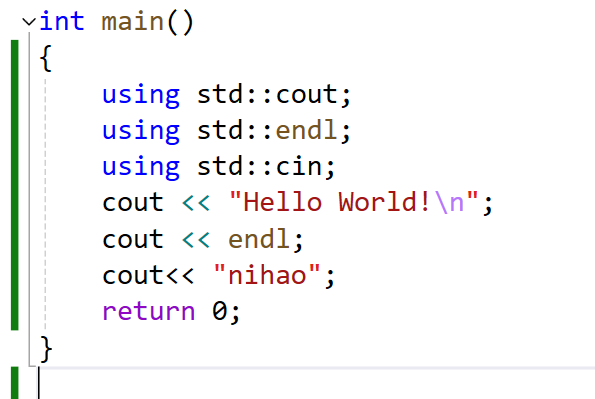
**如果没有用std::cout，需要在前面用using xxxxx std**

**来防止不同头文件中的函数使用错误**

****

**或者在前面用这种方式来声明，这种方式是最稳定的一种方式，会保证不会出错。**

**Cin是输入 cin>>**



**换行有两种**

**1.cout<<endl cout输出默认10进制**

**2.”\n”**

**输出具体的数**

****

**数学运算的头文件**

****

**构建函数**

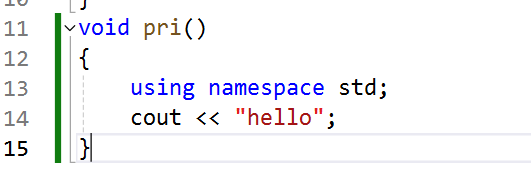
**Type functionname ( )**

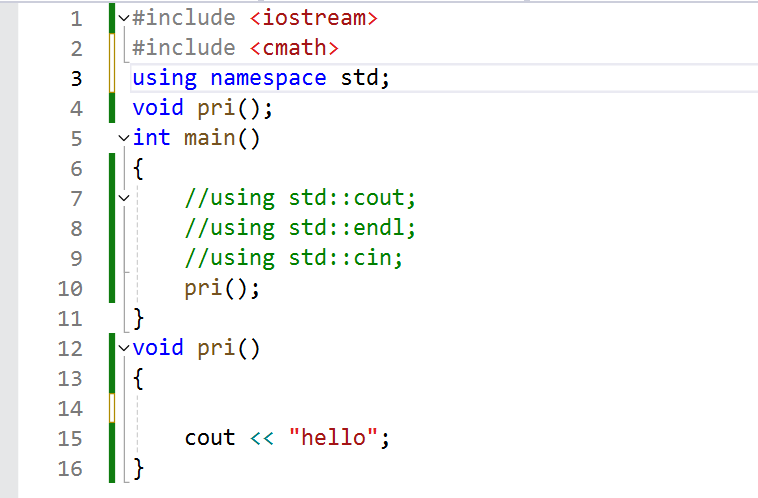
**{**

**}**

**两种方法**

1. **在函数构建内部加入using，只针对这个函数**
2. **在函数声明前面加上using，会让在声明后的所有函数都可以用**

****

****

**Sizeof的调用要加上#include <climits>头文件**

**类型转换，低精度向高精度转换不会变化本身的值，而高精度向低精度转换其结果是不确定的**

**数组的空数组要写’\0’,数组中的内容要写成单引号**

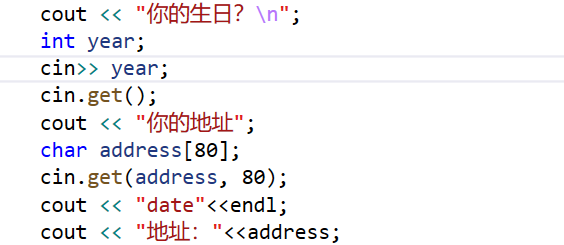
**计算数组长度的要包含的头文件**

**用strlen()**

****

**两个Cin是用空格作为分割，若你要用回车来区分两个输入就不能用cin**

**可以用cin.get(数组名，数组容量)和cin.getline(),这两个都是针对数组。Cin后面可以用cin.get()来吃掉回车符号，让后再次输入。**

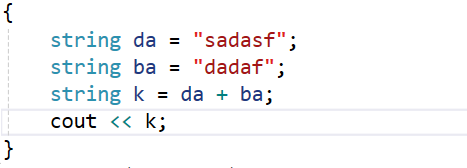
****

**String类**

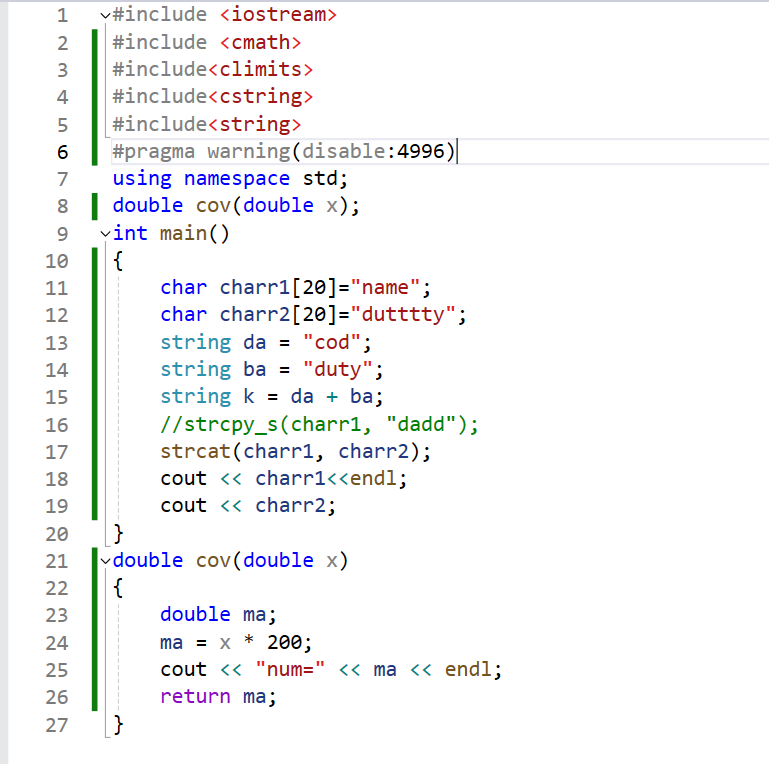
**String相比于数组而言更见简便，可以自动处理string的大小。**

**数组直接的数不可以赋值给另一个，而string可以。**

**String可以直接用 + 来进行相加**

****

**删除警告可以用pragma warning（）**

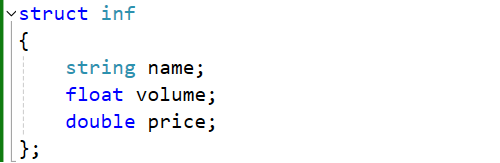
****

**Strcat\_s()添加函数，把2加到1**

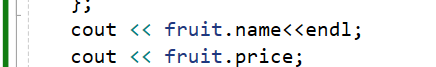
**Strcpy\_s()拷贝函数，把2拷贝到1**

****

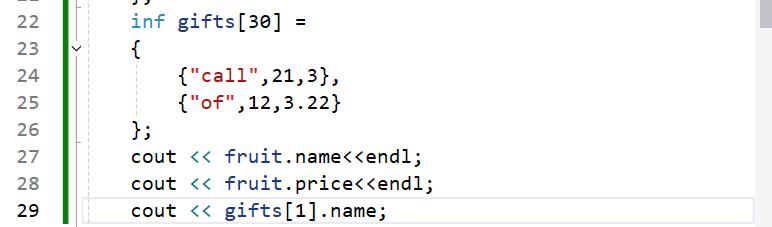
**结构体的构造，结构体可以赋值**

****

**结构体名字.项目 其对应的类型便是其项目的类型**

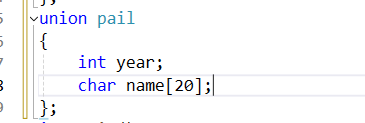
****

**结构数组 struct**

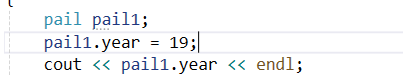
****

**共用体 union**

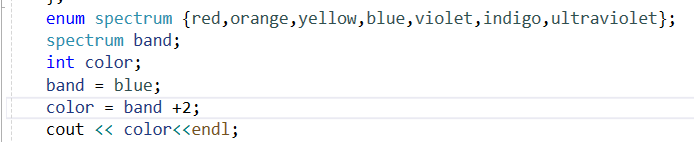
**共用体是对int或double或long，这三个里面的一种进行存储，结构体可以同时存储上面三种。**

****

**在调用的时候只能这样，不能像结构体一样直接大括号里面调用。**

****

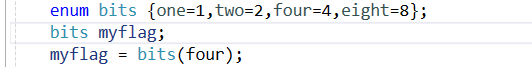
**枚举 enum**

****

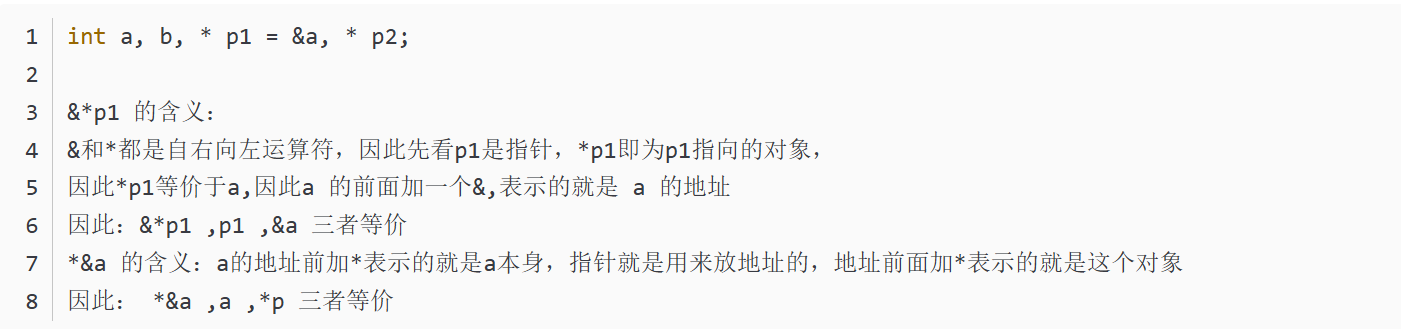
**枚举其中对应的是int整形，各类颜色代表0，1，2，这中整形。**

**可以定义一个整型在等号左边进行加减，但是不能定义枚举在左边进行加减。**

**枚举可以进行自己定义里面的数值**

****

**指针**

****

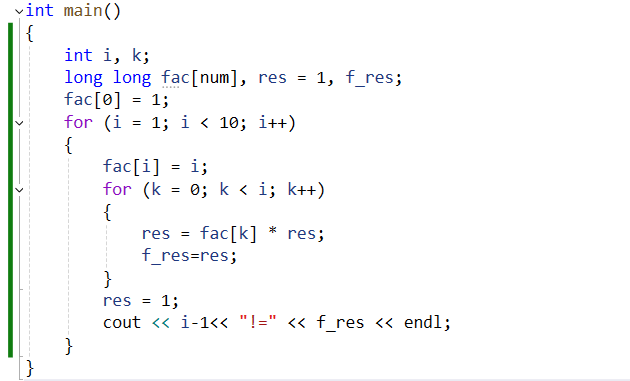
**< 和 > 的优先级比 << 要低所以要用括号括起来，来正确运行**

****

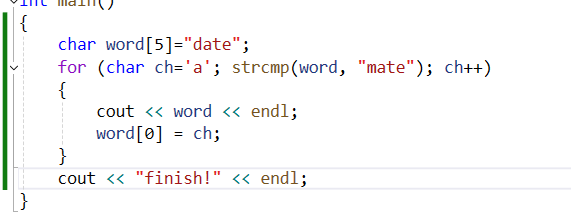
**Cout.setf(ios\_base::boolalpha)可以将bool值0，1变成ture和false**

****

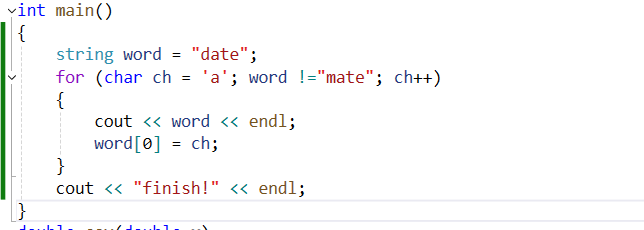
**For循环阶乘计算**

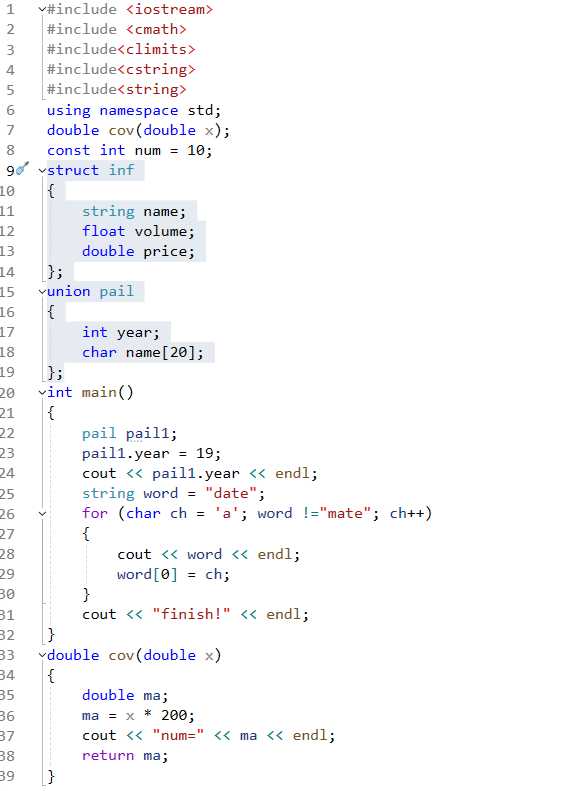
****

**通过strcmp（名称，比较字符）来进行比较运算，当第一个与第二个相等则停止**

****

**或者用string来进行比较**

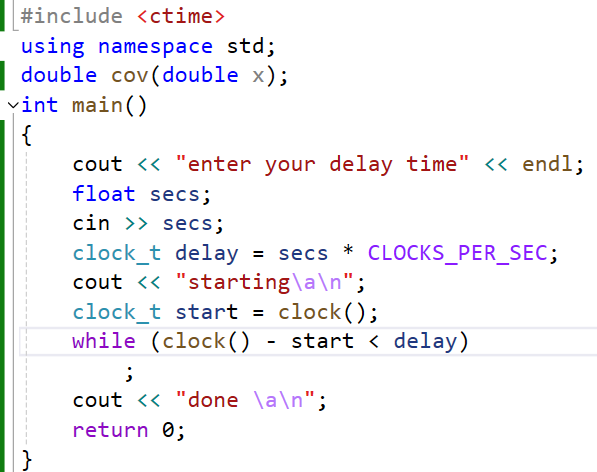
****

****

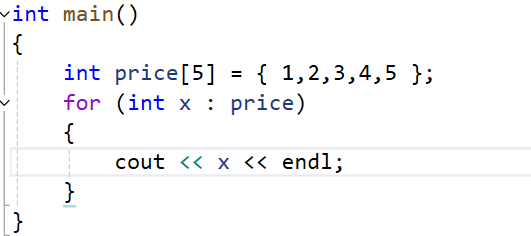
**类型别名**

1. **#define 新名字 int/char/double**
2. **typedef char byte**

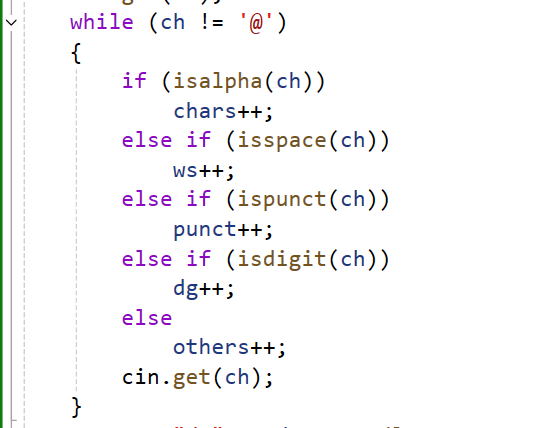
**延时程序**

****

**For循环遍历数组**

****

**检测数组内的字符个数**

****

**？：运算符 A? B: C**

**如果A条件成立，则A的结果是B**

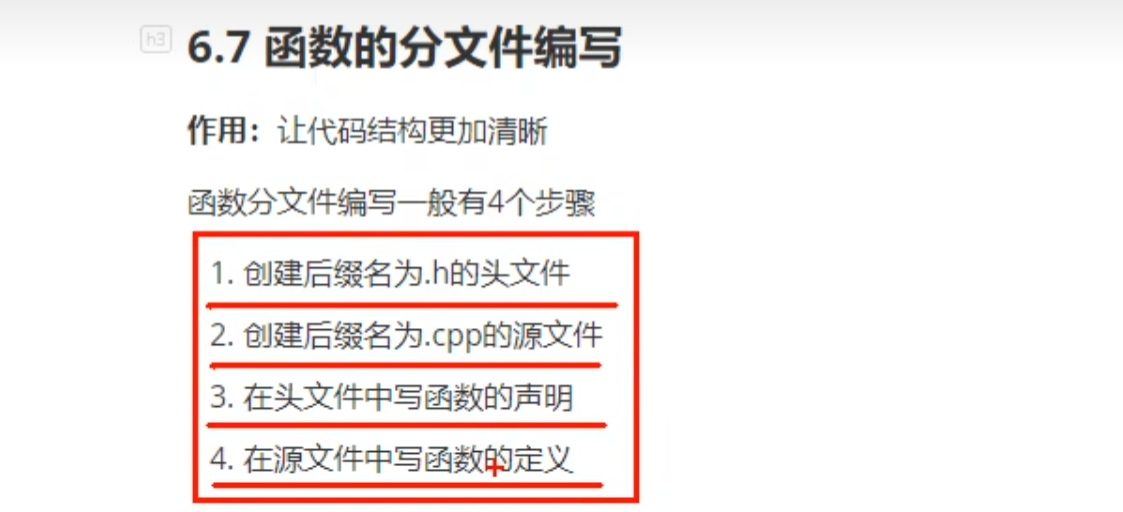
**若A的条件不成立,A的结果是B**

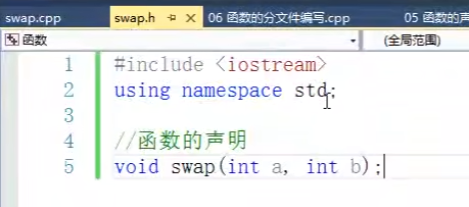
**Switch语句**

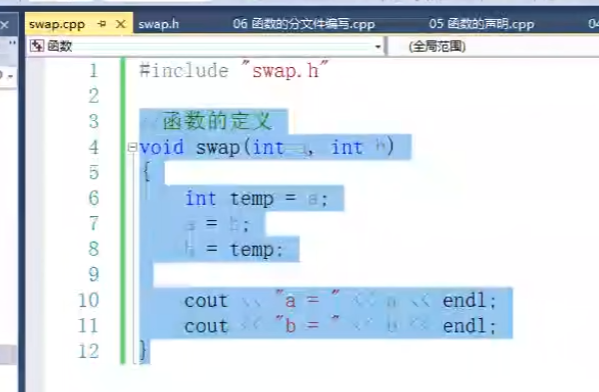
**Switch语句用于多个选择，switch可以和枚举一起使用。**

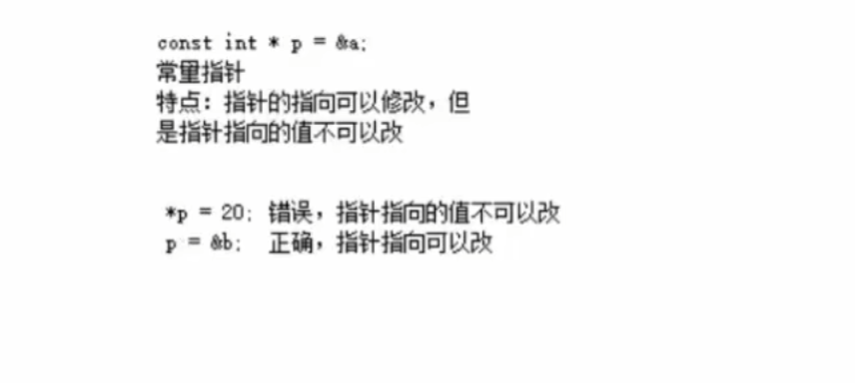
**Switch只能适用于具体数，不能适用于范围值**

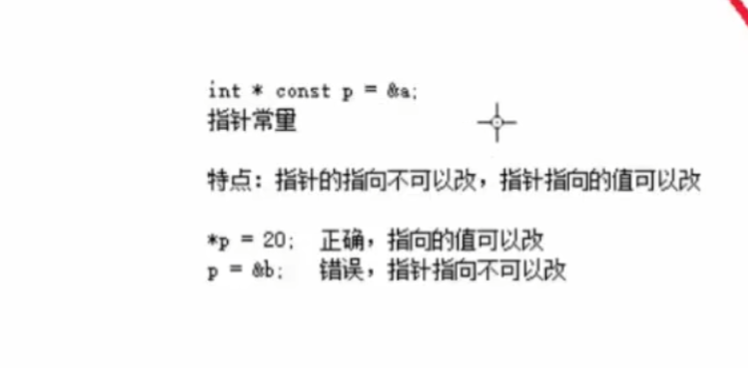
**数组的大小** nums.size();

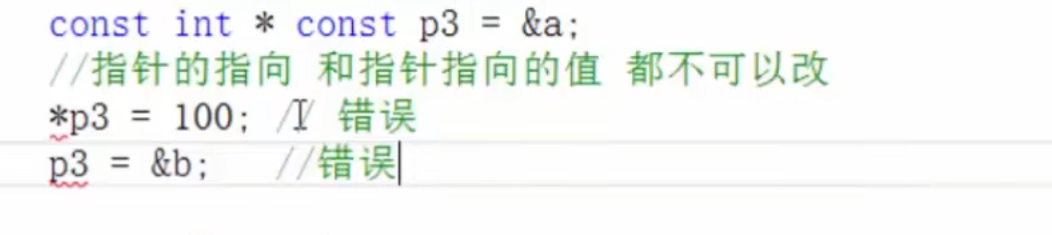
****

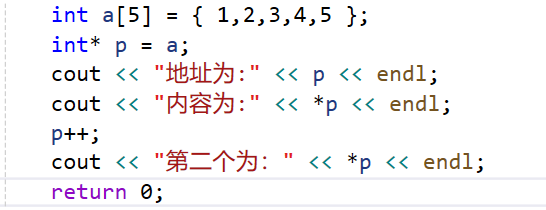
****

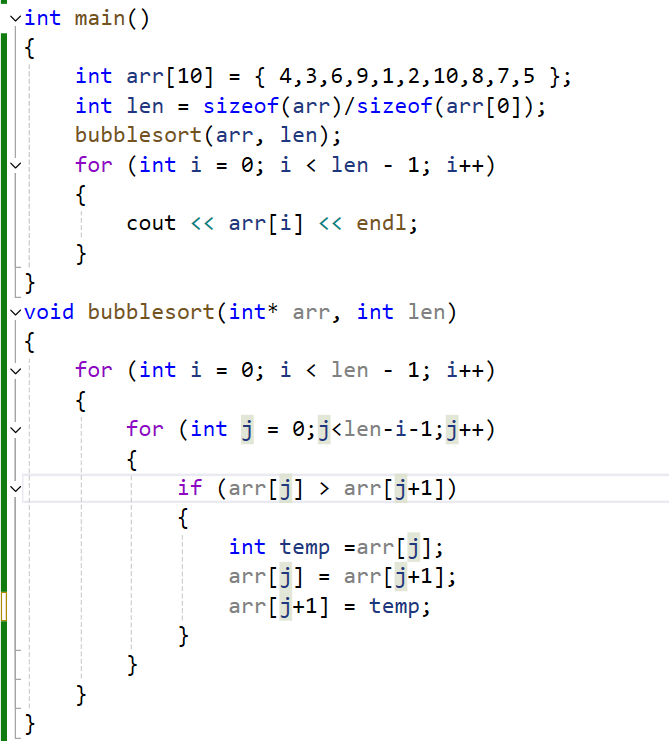
****

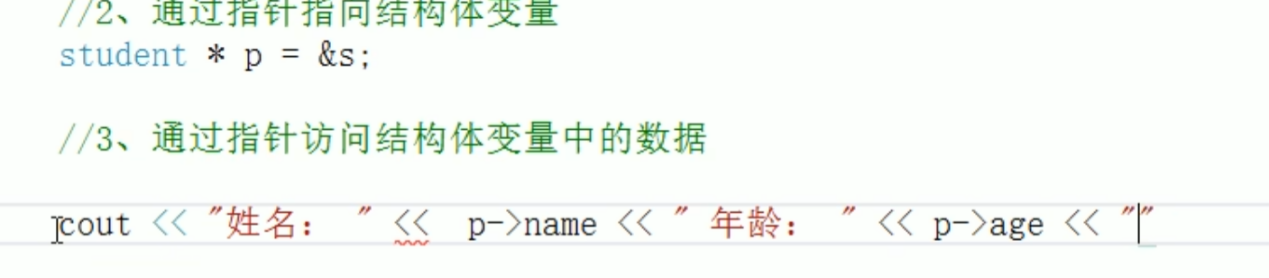
****

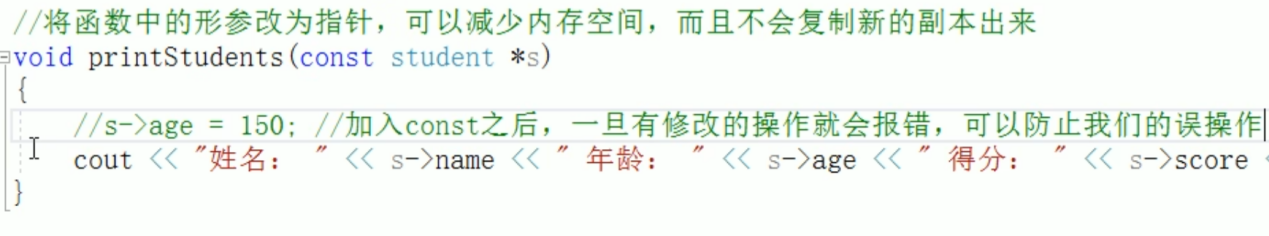
****

****

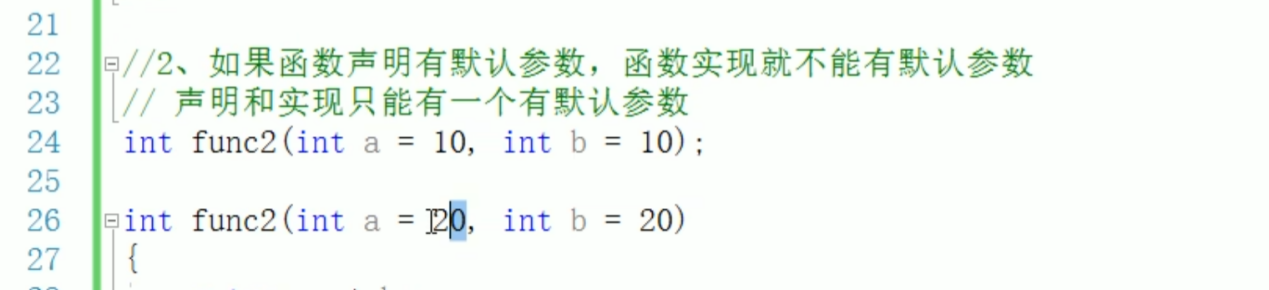
****

****

****

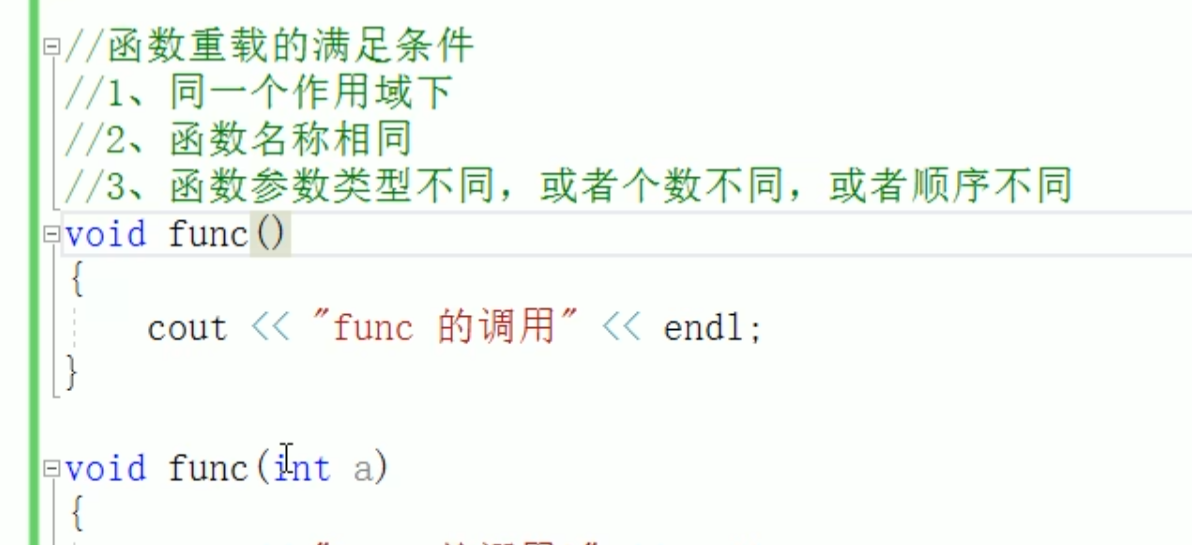
****

**函数的默认参数**

****

**函数的默认值如果有传入则就是传入值，没有值则就是默认值。**

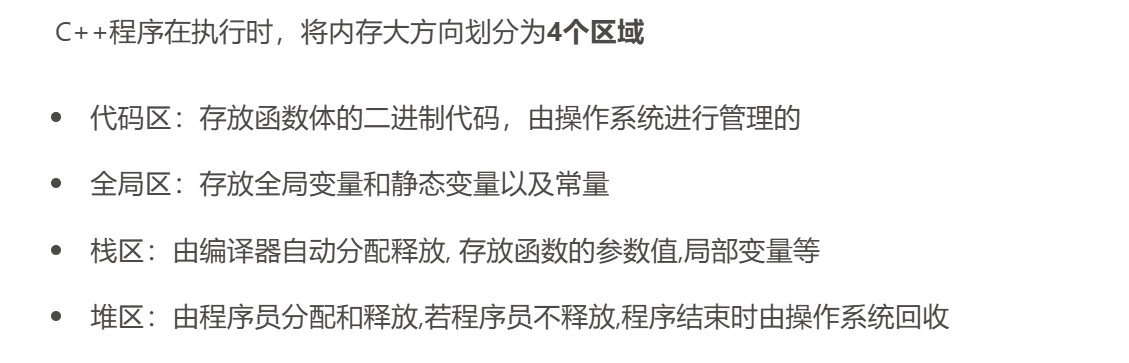
**函数重载**

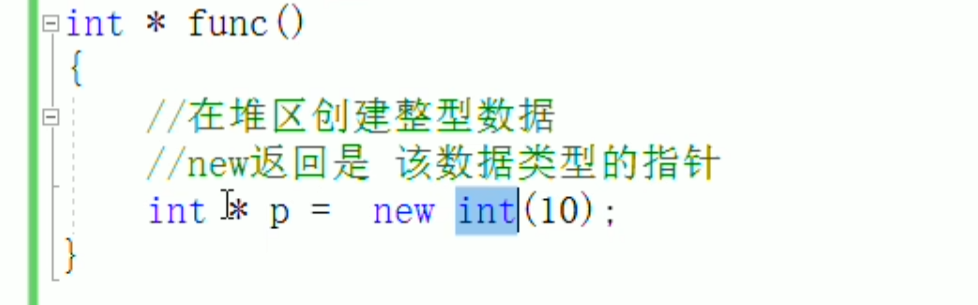
****

**引用也可以重载**

****

**内存分区模型**

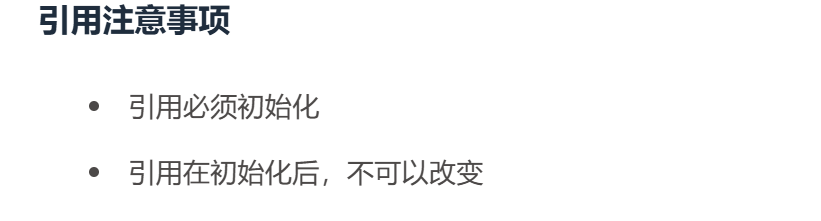
****

****

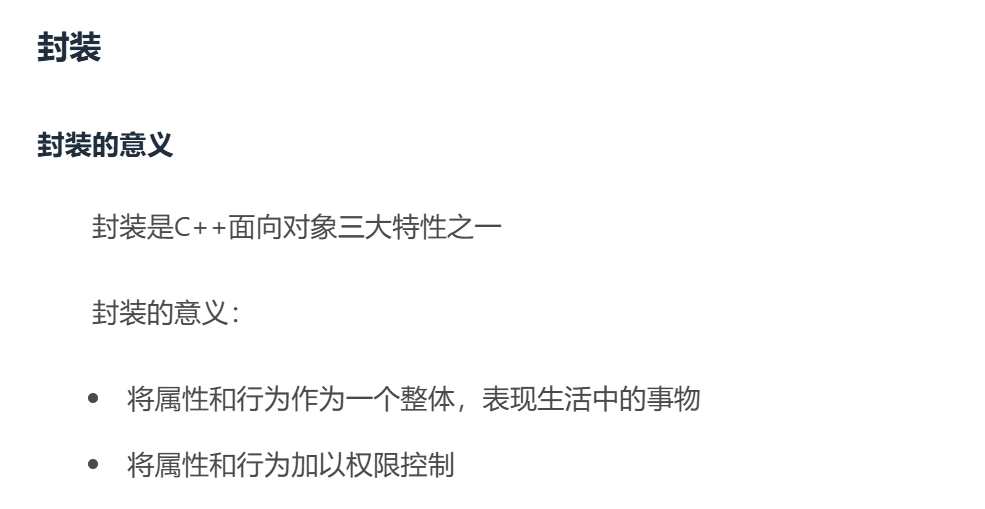
****

****

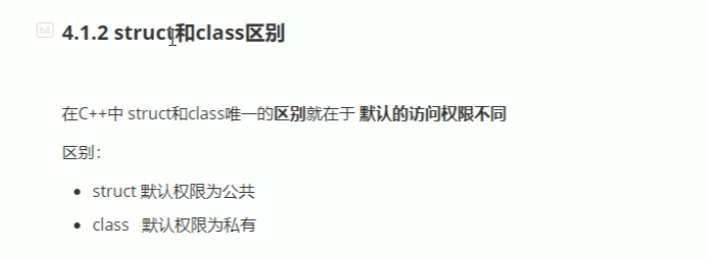
**当别名改变时候，其原名的数值也会改变**

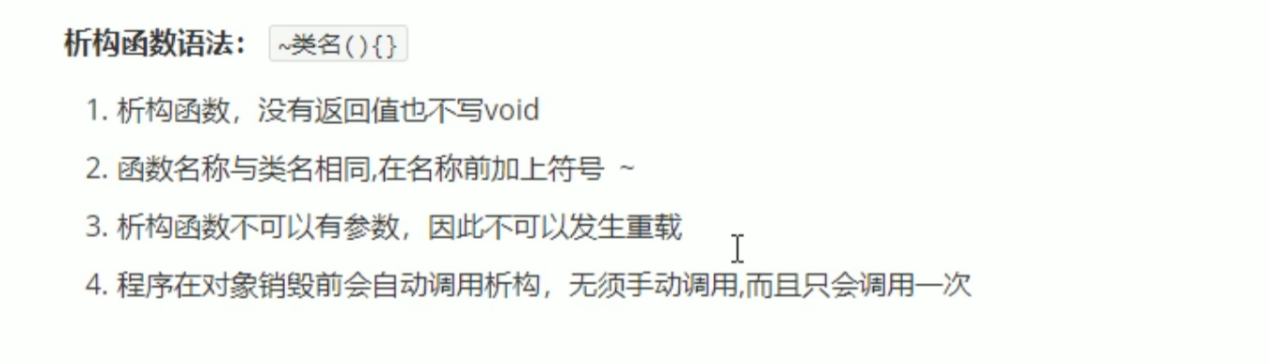
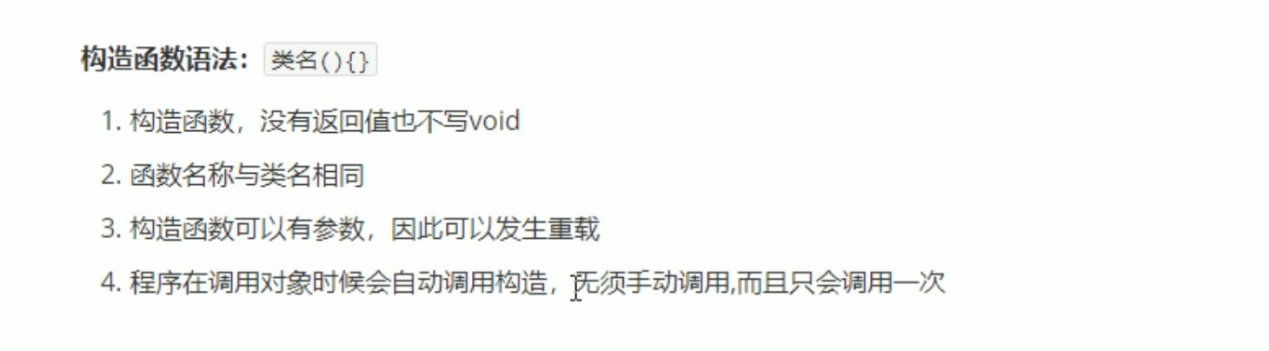
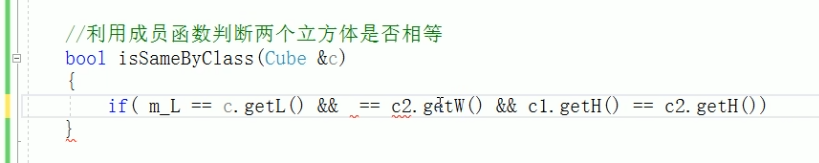
****

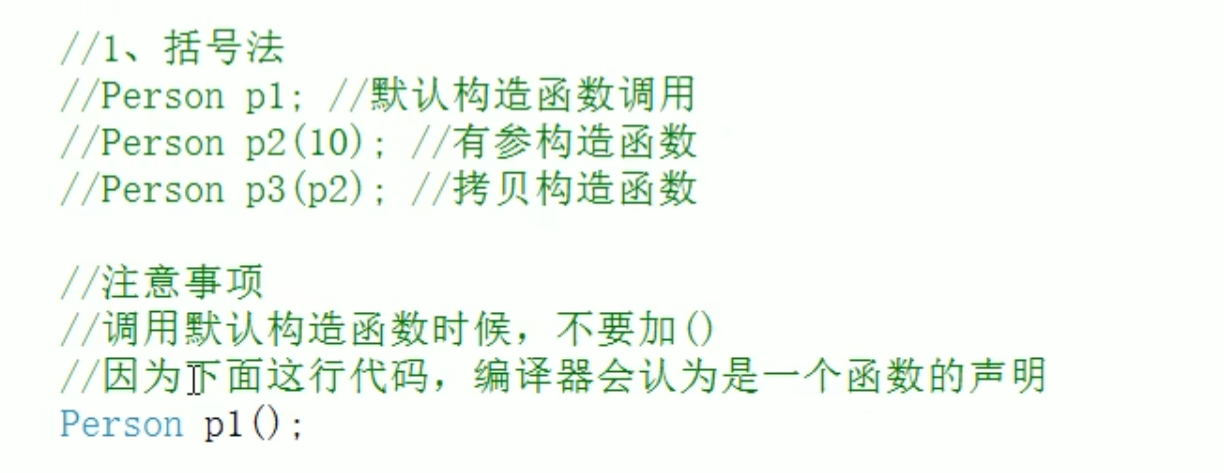
**通过引用参数产生的效果同按地址传递是一样的。引用的语法更清楚简单**

****

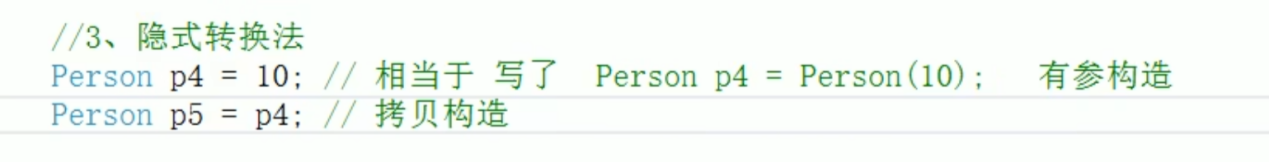
****

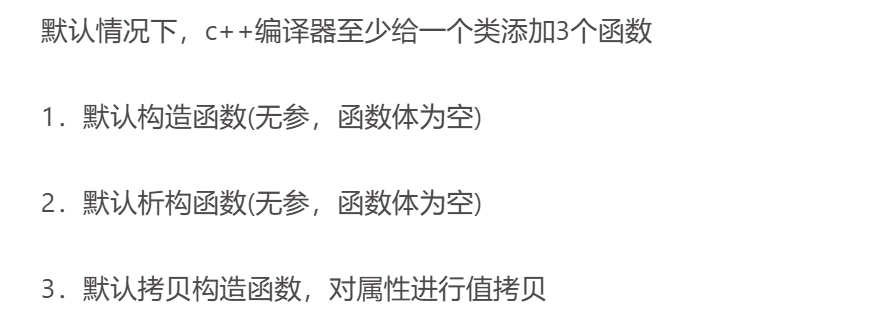
****

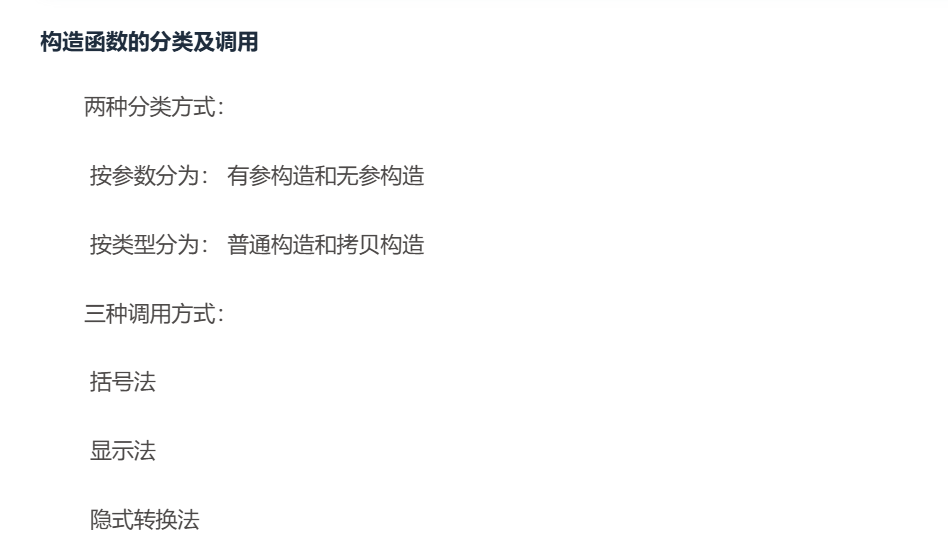
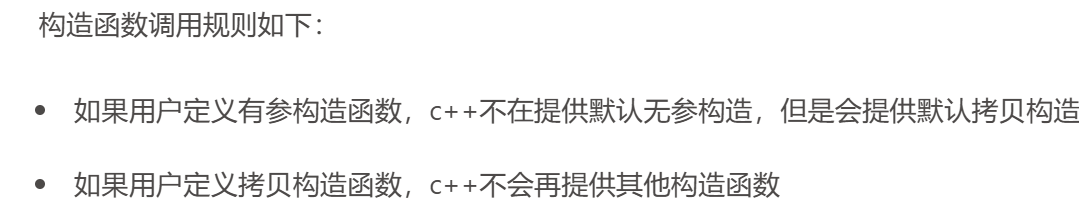
****

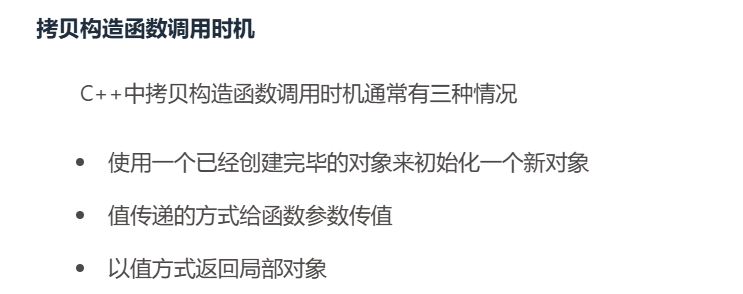
****

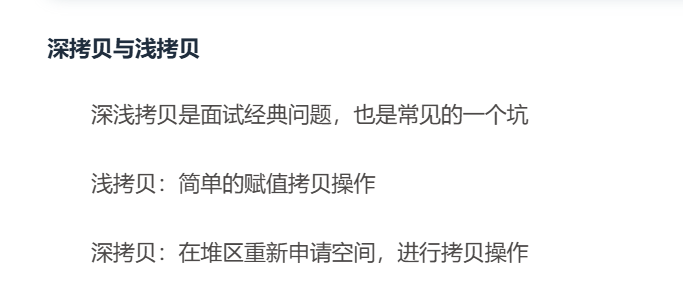
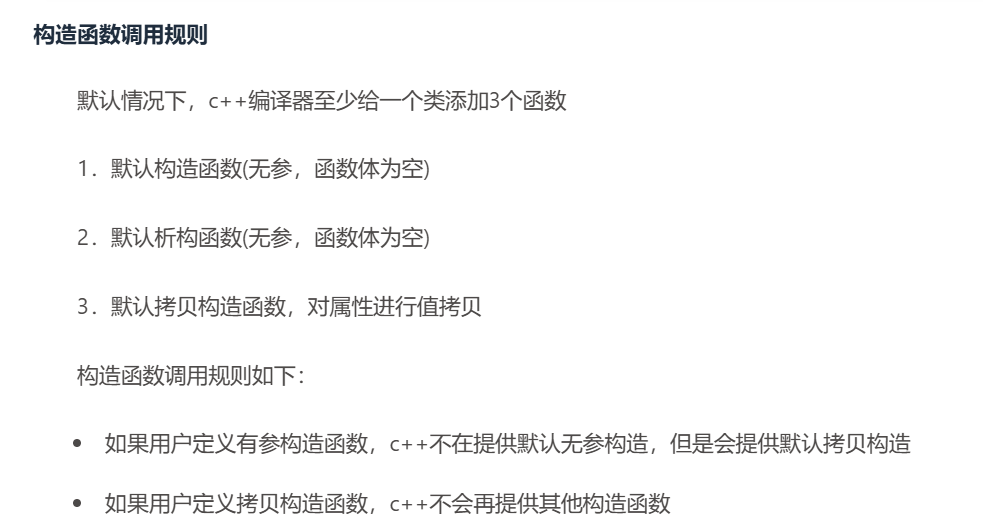
****

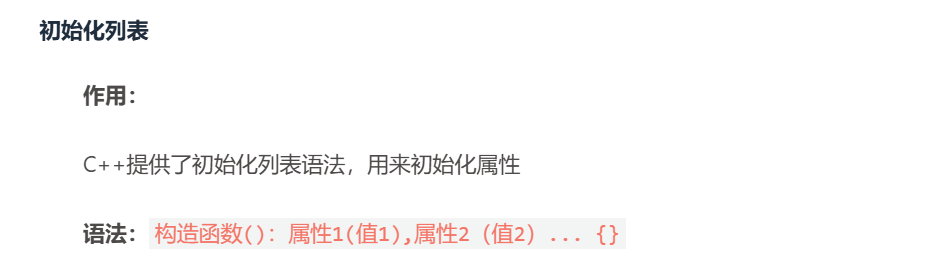
****

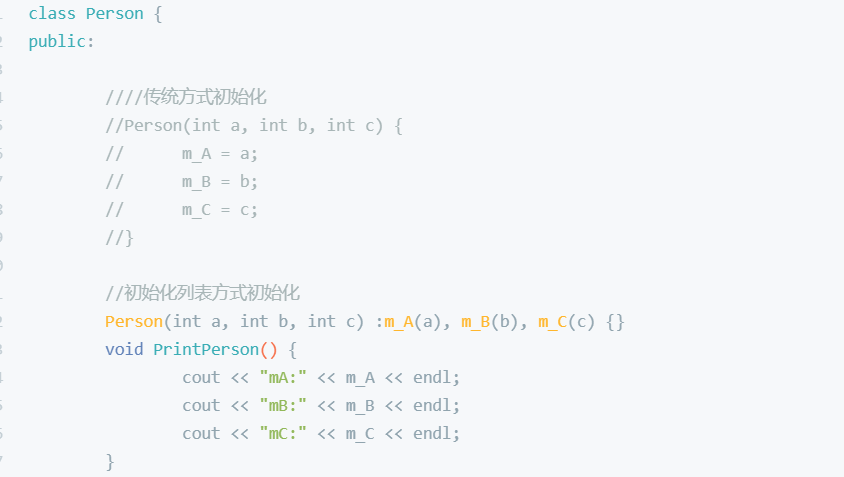
****

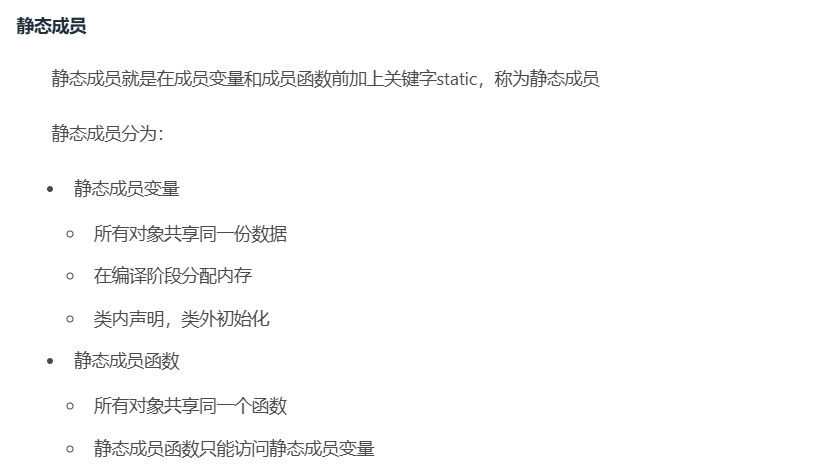
****

****

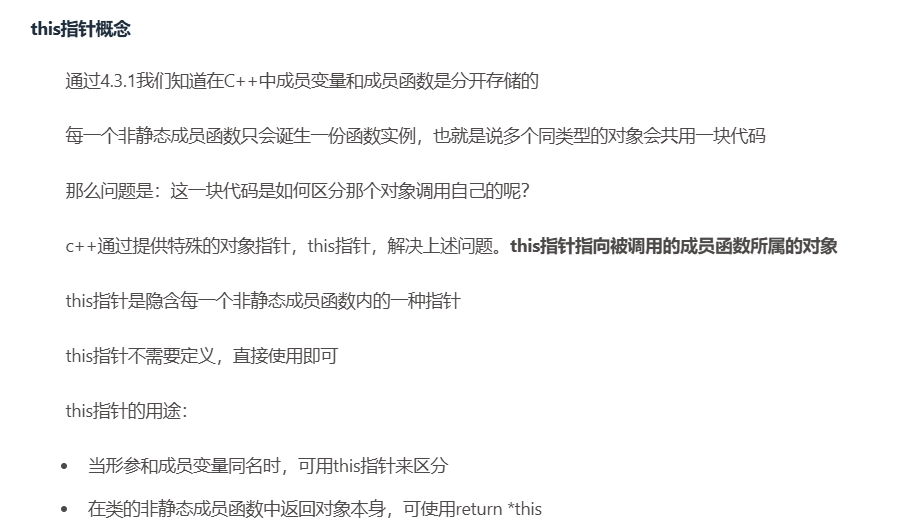
****

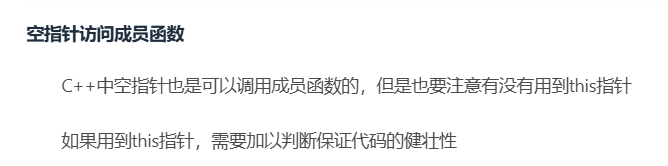
****

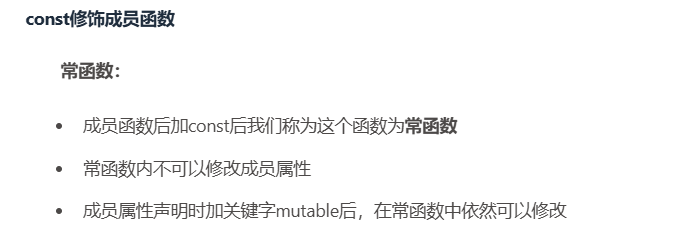
****

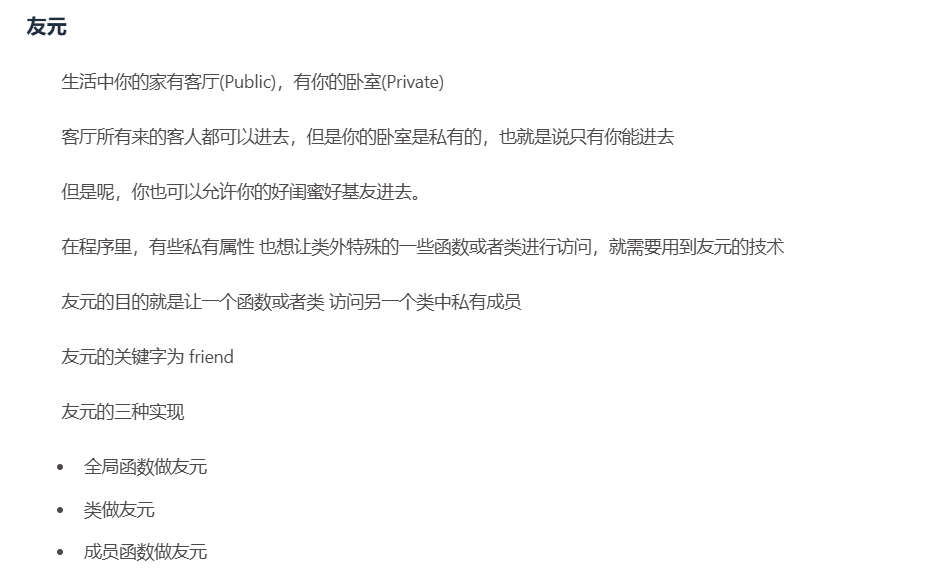
****

****

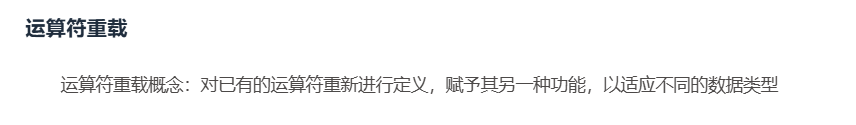
****

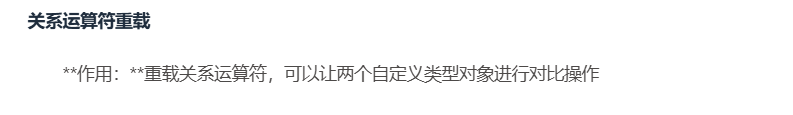
****

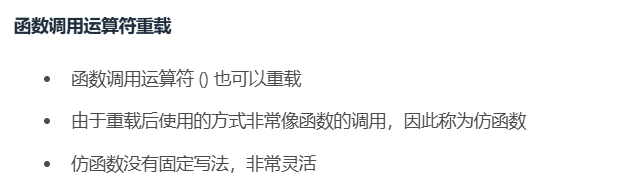
****

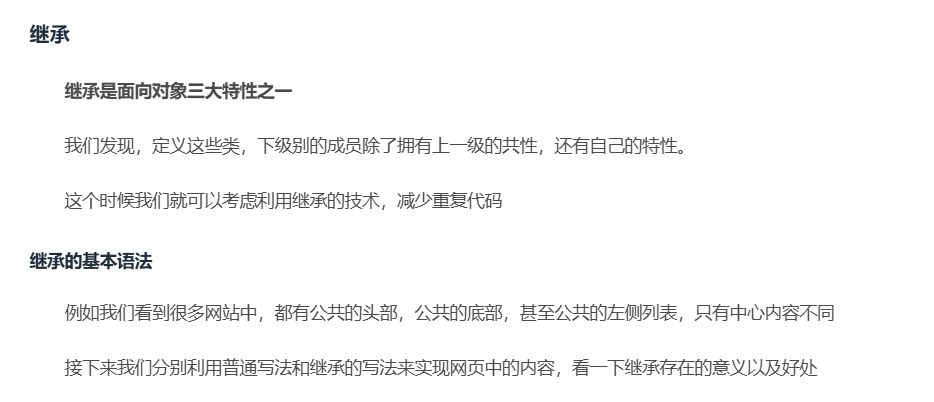
****

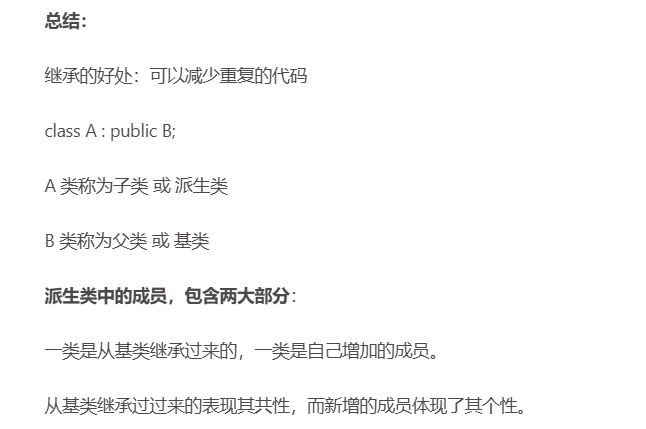
**有类做友元，全局函数做友元和成员函数做友元**

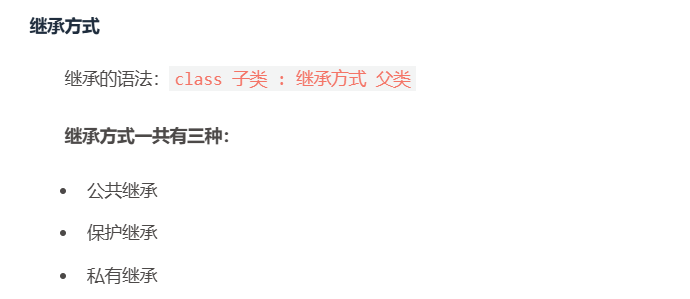
****

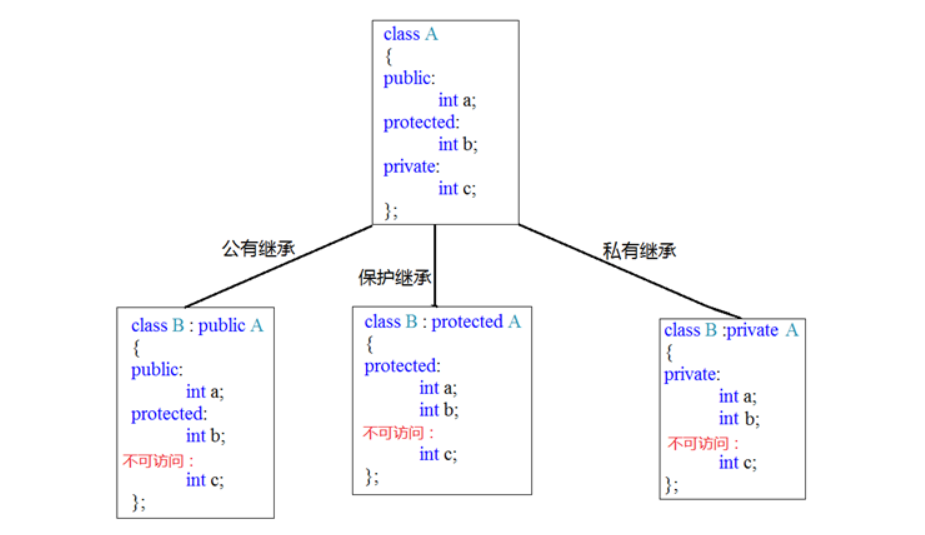
****

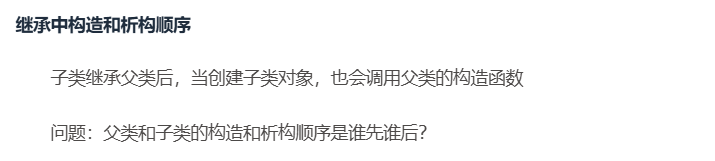
****

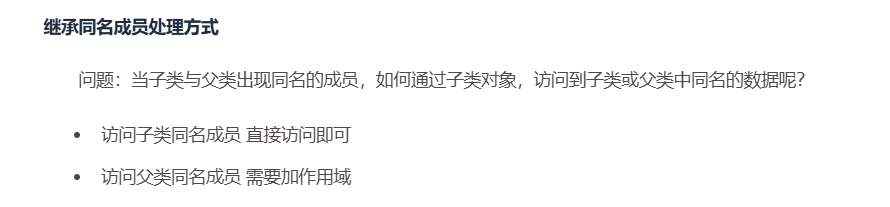
****

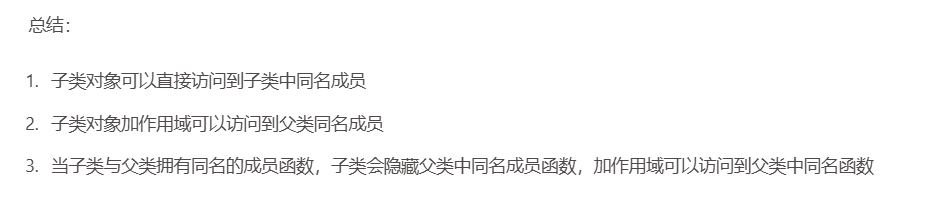
****

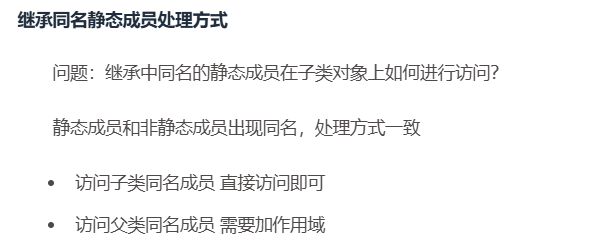
****

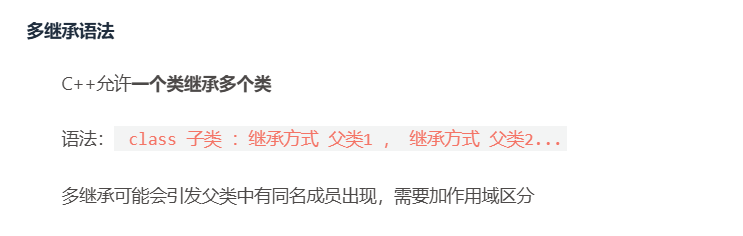
****

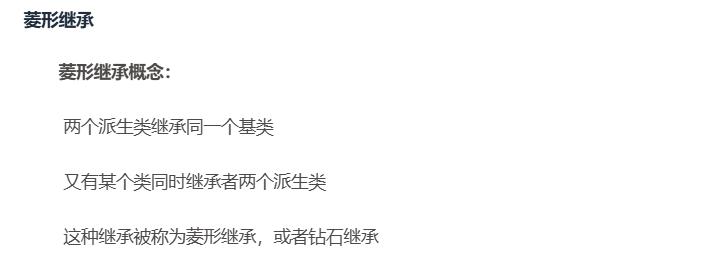
****

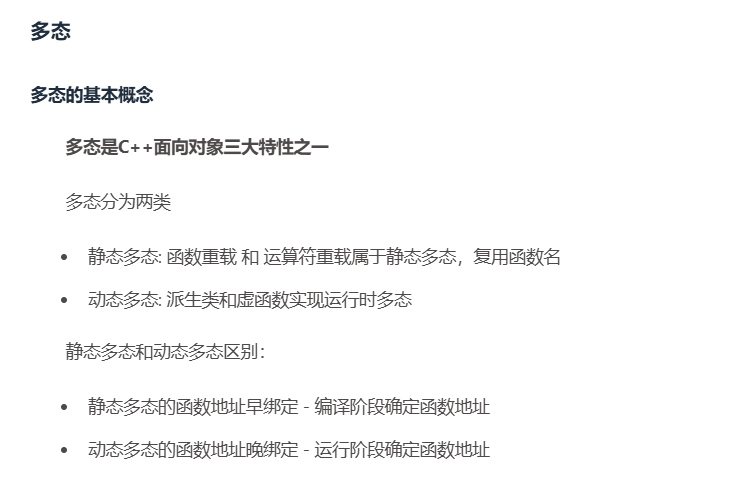
****

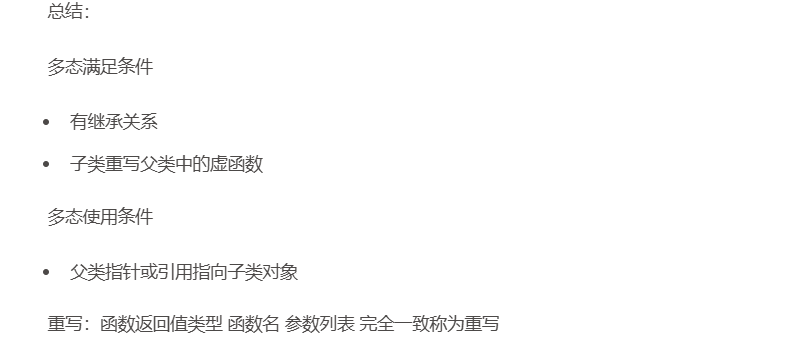
****

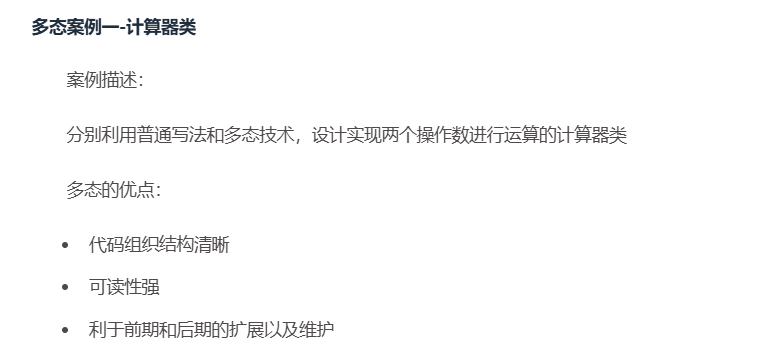
****

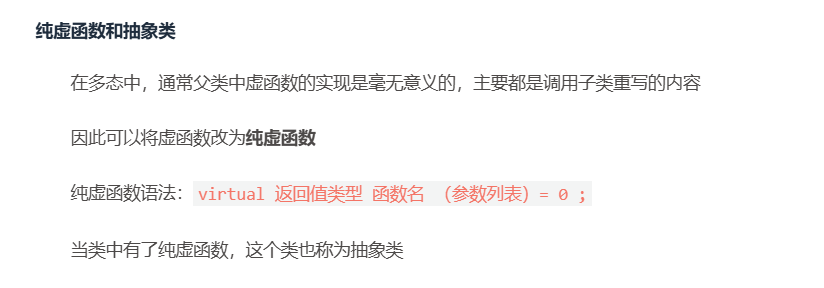
****

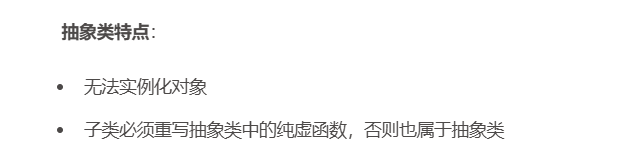
****

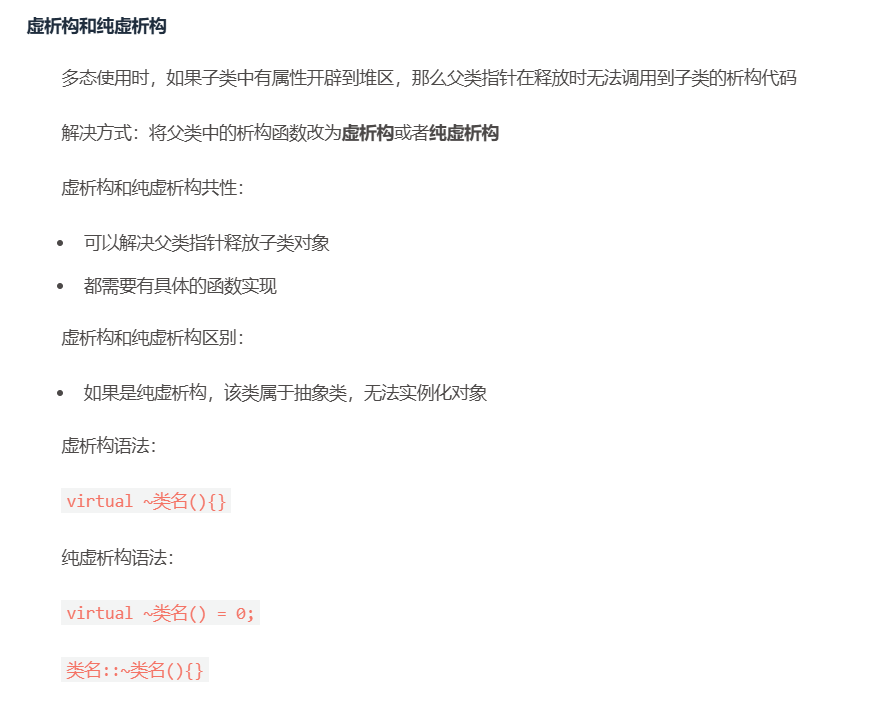
****

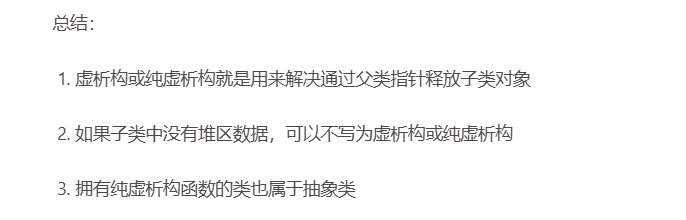
****

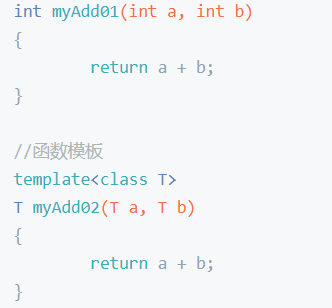
****

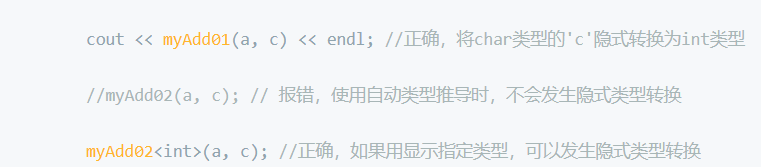
****

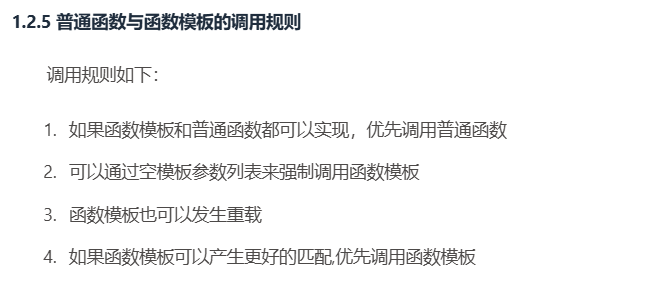
****

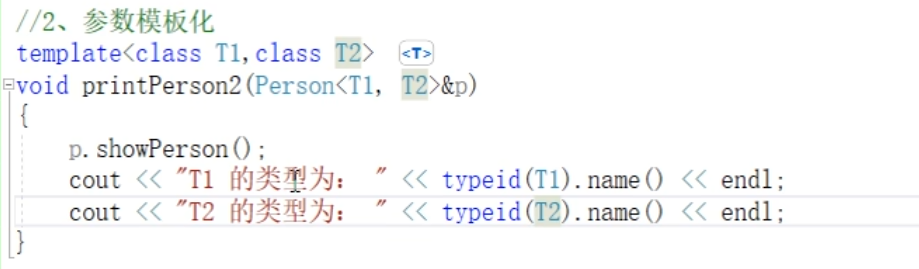
****

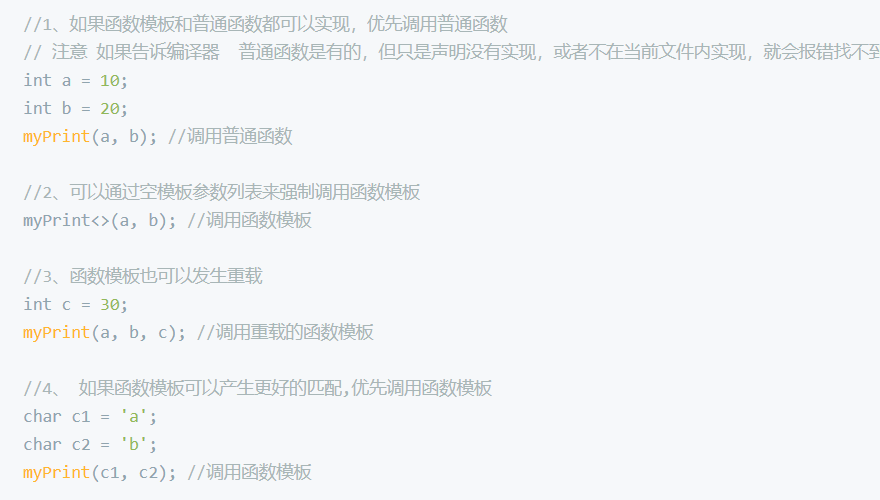
****

****

****

****

****

****

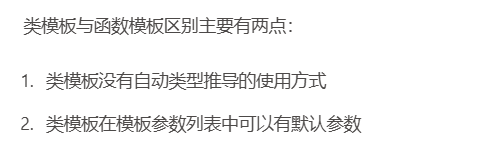
****

****

****

**类模板**

****

****

