**《面向对象程序课程设计》上机实验报告**

班级：软件四班 学号：55210425 姓名：朱家顺

1. 实验题目

第一题：

对如下多项式编写类定义：

+++…+

a0+a1x+a2x2+…+anxn

其中，n为多项式的次数。完成如下功能：

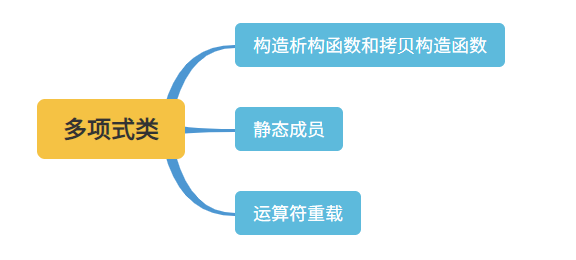
1. 可存储任意大的多项式（提示：用什么数据结构呢？）。
2. 定义构造函数、析构函数、拷贝构造函数。
3. 包含一个static成员存储定义的多项式的数量。
4. 定义一个成员函数输出多项式。（可参照-x^4-6x^3+5格式输出）
5. 定义一个成员函数计算多项式的值。
6. 写main函数测试类的功能。
7. 采用多文件实现。

考虑：哪些成员函数可以声明为const。

1. 重载“+”运算符，实现两个多项式相加。
2. 重载“-”运算符，实现两个多项式相减。
3. 重载“\*”运算符，实现两个多项式相乘。
4. 重载“=”运算符，实现两个多项式的赋值运算。

考虑：把其中某个运算符重载为友元函数。

1. 解决方案



此题着重考察了我的如上三大块知识的掌握应用。

构造函数是一种特殊的成员函数，它主要用于为对象分配空间，进行初始化。构造函数的名字必须与类名相同，而不能由用户任意命名。它可以有任意类型的参数，但不能具有返回值。它不需要用户来调用，而是在建立对象时自动执行。

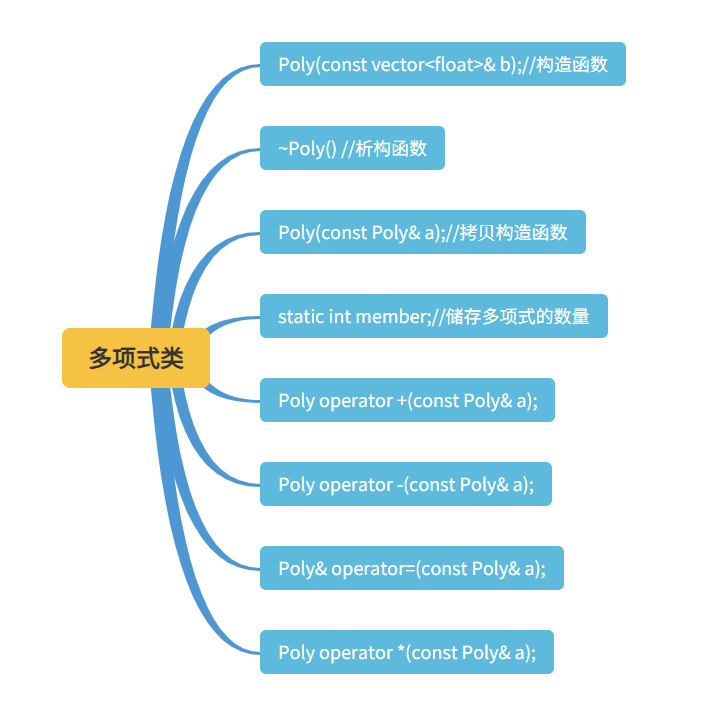
析构函数也是一种特殊的成员函数。它执行与构造函数相反的操作，通常用于撤销对象时的一些清理任务，如释放分配给对象的内存空间等。析构函数有以下一些特点：1.析构函数与构造函数名字相同，但它前面必须加一个波浪号（~）。

1. 析构函数没有参数和返回值，也不能被重载，因此只有一个。3.当撤销对象时，编译系统会自动调用析构函数。

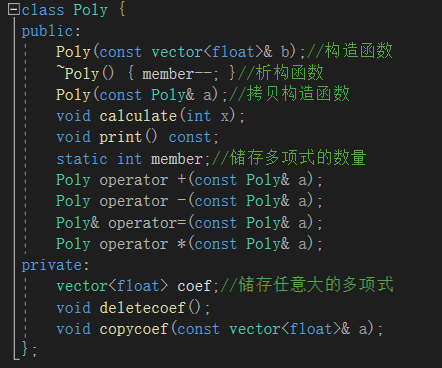
拷贝构造函数是一种特殊的构造函数，其形参是本类对象的引用。拷贝构造函数的作用是在建立一个新对象时，使用一个已存在的对象去初始化这个新对象。

静态数据成员属于类（准确地说，是属于类中对象的集合），而不像普通数据成员那样属于某一对象。

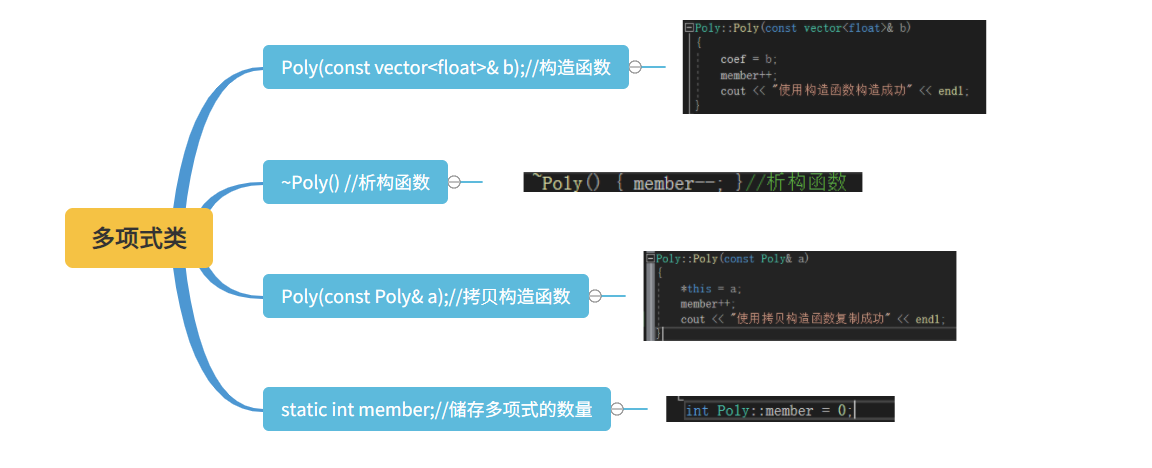
基于以上的思想，能够对多项式的类进行初步的编写类的成员函数以及数据成员。



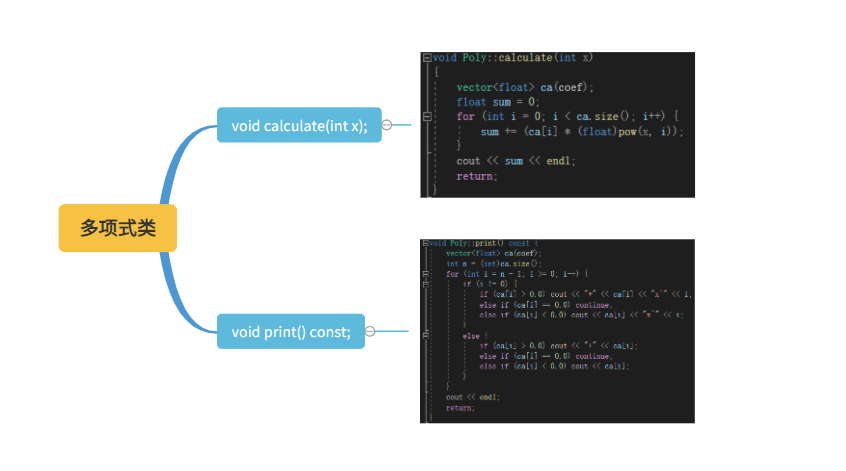
1. 程序清单

为了实现题目所需功能，最后在类的.h文件中定义的类具体成员创建如下：

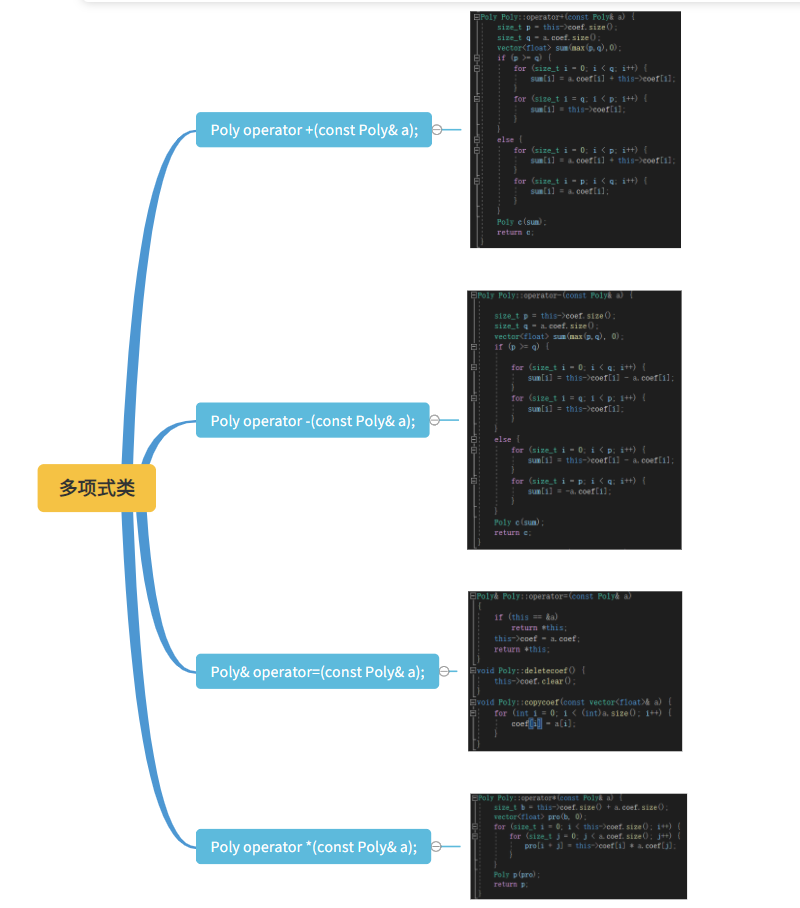
1~3项功能实现如下：



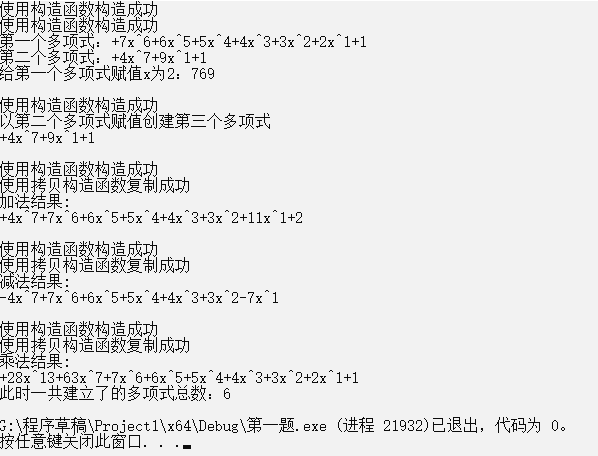
4~5项功能的实现如下：



8~11项功能实现如下：



1. 程序运行结果



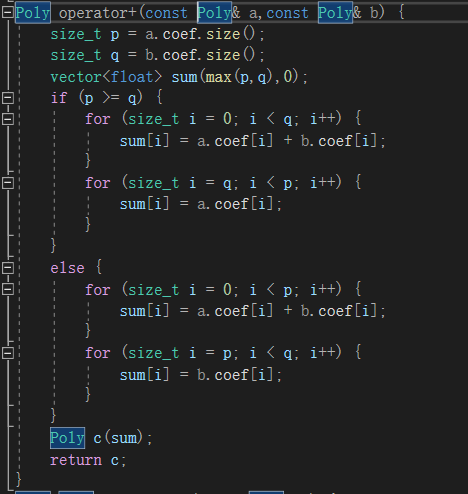
1. 体会与总结

a、对相关知识点的理解

首先，我对于const关键字有了更加深刻的理解。C++中的const关键字的用法非常灵活，而使用const将大大改善程序的健壮性。首先就在于const常量有数据类型，便于进行类型检查。因为宏常量没有数据类型，程序中我对部分int型常量和size\_t型常量相互传递值，给程序带来了隐患，程序内部默认字符替换，未进行类型安全检查更是使得我的程序产生了意想不到的运行错误，这一点，是最后我发现了这一遗漏的知识点后，对int型变量改变类型来解决的，如果一开始使用const size\_t型来定义多项式系数向量的大小，就可以让编译器查出类型不匹配的隐患。其次，const关键字更是可以保护部分成员函数，如此题中的所有运算符重载函数（我进行的是重新定义一个多项式来接受加、减、乘、赋值后的多项式内容），以及打印多项式和计算多项式的值的函数，以防止意外的修改，增强了程序的健壮性。除此之外，这依然是一个良好的习惯，因为这一方面节省了空间，避免了不必要的内存分配；另一方面提高了程序运行的效率。

其次，我对友元函数也有进一步的理解。这里我将友元函数在此补充，原程序由于疏漏，在此处给出示例函数的友元形式，首先打印和计算函数在实现友元函数形式方面没有什么实际意义，因此应该从运算符重载函数中选取合适函数采取友元函数形式调用即可。（只是稍加改动参数，并没有什么实质上的改变）





在此，我对于友元函数有了更加深刻的理解。老师上课提到了包括=、（）、[ ]、-〉在内的运算符只能重载为成员函数运算符，cout<<、cin>>（cout是其它类的对象）等只能重载为友元函数运算符，我对于其余的运算符进行了了解，将深入了解的知识列举如下：

1.在极少数的情况下才会使用普通函数，因为普通函数不能直接访问类的私有成员。解决方法是在类中定义公用的设置数据的set函数和读取数据的get函数，在重载函数中调用这些函数去访问类的私有成员。显然这样不好。

2.如果将运算符重载函数作为成员函数，它可以通过this指针自由地访问本类的数据成员，因此可以少写一个函数的参数。但必须要求运算表达式的第一个参数是一个类对象，而且与运算符函数类型相同。

3.如果运算符左侧的操本作数不是本类对象，则运算符重载函数不能作为成员函数，只能作为非成员函数；如果函数需要访问类的私有成员，则必须声明为该类的友元函数。

4.将双目运算符重载为友元函数是，在函数的形参列表必须有两个参数，不能省略，形参的顺序任意，不要求第一个参数必须为类对象。但在使用运算符的表达式中，要求运算符左侧的操作数与函数的第一个参数对应，运算符右侧的操作数与函数的第二个参数对应。数学上的交换律在此不适用。解决方法是再重载一次，交换参数顺序。

5.由于友元会破坏类的封装，因此原则上，要尽量将运算符函数作为成员函数。一般将单目运算符重载为成员函数，将双目运算符重载为友元函数。

另外，我发现此题为实现存储任意大的多项式，我采取了vector的容器，事实上采取map容器可以更加简洁的完成要求。（map<size\_t,vector<float>> m）

b、对运行情况所做的分析

程序初始由于未使用const修饰Poly类的引用，出现了编译器无法检查出来的错误，以及关于向量越界的问题，经过仔细排查和对之前知识的复习，通过逐行执行cout语句来发现问题。

整体而言，就是size\_t和int类型的不相容，编译器在运行时给出了提醒，这最终没有影响到程序的运行。以及对Poly类的const修饰和引用，需要贯彻到类中。这两大点是运行过程中所遇见的主要问题。

此外，我发现运算符-的重载完全可以借用对+运算符的重载函数，这是我之前未曾注意到的，事实上这也导致了一定程度上程序执行效率的下降。

c、对调试程序所获得的经验和体会

我深入了解了类的构造和析构、深拷贝和浅拷贝、运算符重载、静态变量在类定义中所起的独一无二的作用、以及各种系统函数返回值的类型等等。对面对对象程序设计思想有了一个全新的认识。

1. 111

一、实验题目

第二题：

C++的一般编译器都定义和封装了字符串功能，请模仿定义string类的实现，可以实现并支持如下功能：

（1）string s = “吉林大学”；

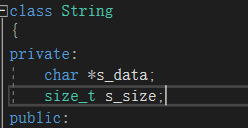
（2）string t = s;

（3）string m； m = t；

（4）m.legnth() 函数测量字符串的长度

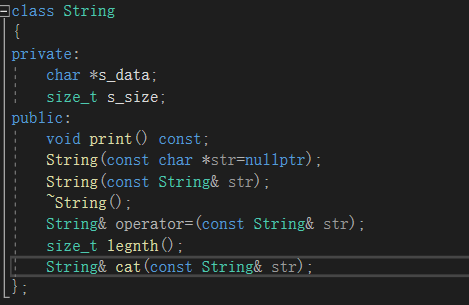
（5）m.cat(string const &)连接字符串

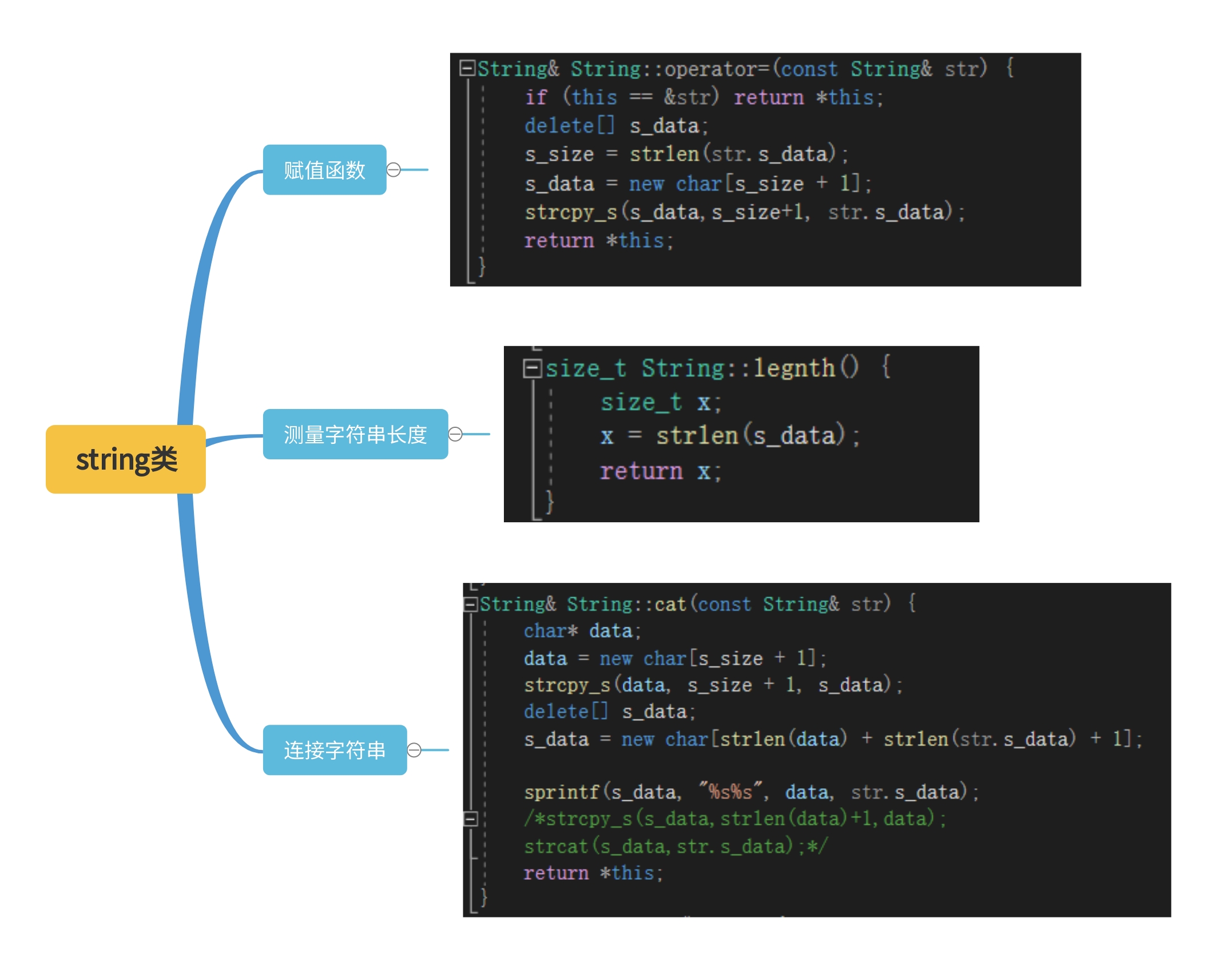
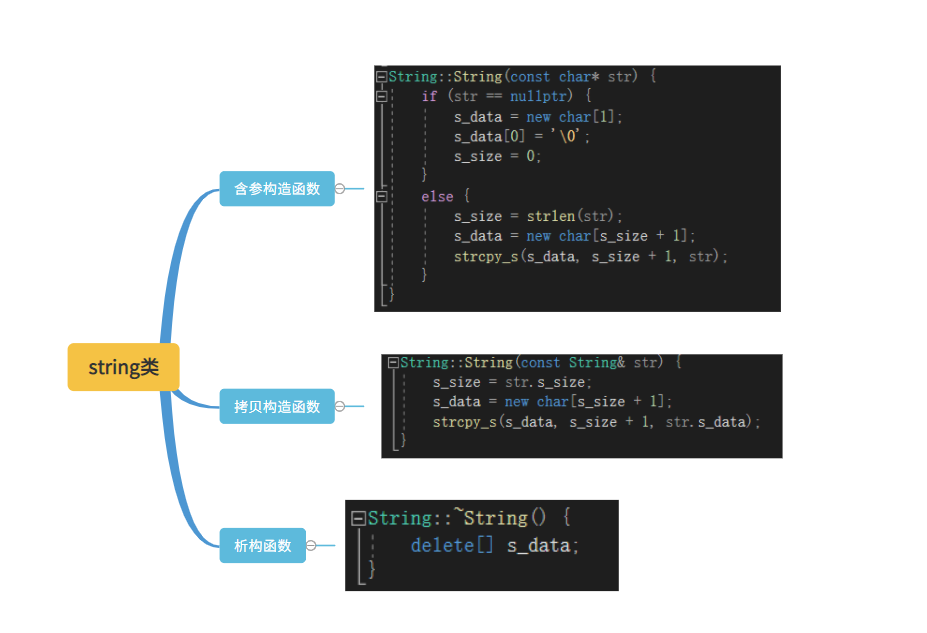
二、解决方案



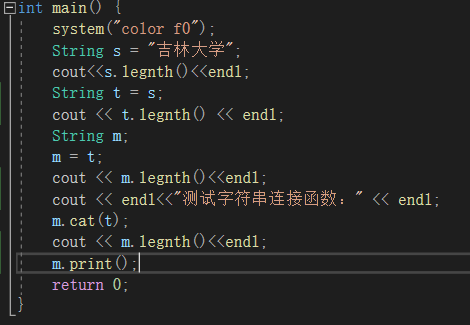
分析要求发现可以通过构造一个字符串数组和一个整型量表示字符串长度来实现题目所需的构造函数、拷贝构造函数、测量字符串长度的函数以及连接字符串长度的函数。

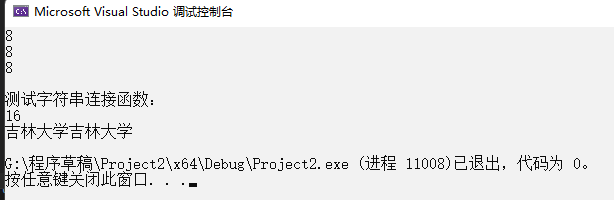
三、程序清单





四、程序运行结果





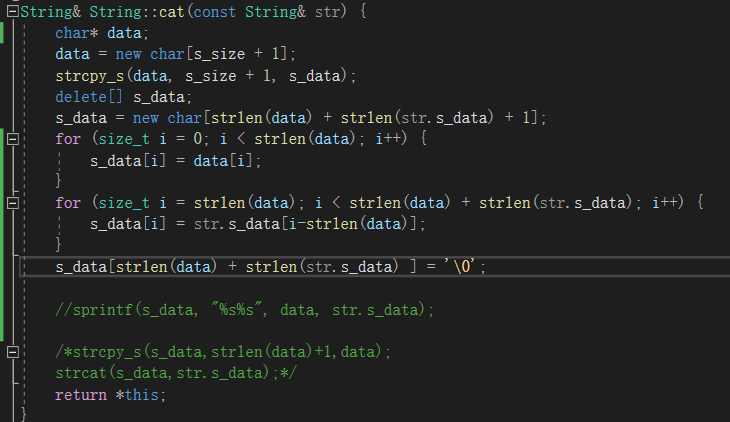
五、体会与总结

关于字符串连接函数，还有其他的实现方式，我了解到strcat函数，也就是此题希望替代的原函数，使用该函数也可以容易的实现cat函数的建立。



除此之外，使用sprintf函数也可以完成这一功能，也就是我题目中所使用的方法。

不过我还想提一提最基础的通过构造函数来实现字符串连接的方法。



除此之外，我还发现我这一种C-串流是比较落后的，也就是我通过落后的C-串流实现了更为适用的string流，这也是我想将第三种方法拿出来一提的原因，这让我更加感受到从C-串流走到string流C++标准的变化，而题目的目

一、实验题目

第三题:

我公司为仪器生产企业，目前生产摄像机和行车记录仪两种产品，分别销售给用户。

摄像机包含摄像、图像质量设定、编码算法等属性。

将摄像机增加相应芯片（具有操作菜单、自动拍摄、车速传感器、源代码等功能)后，形成一个行车记录仪。

要求：

设计摄像机类，并请根据下列不同的功能要求，采用不同的继承方式，设计行车记录仪类,并添加测试代码，体验不同继承方式下的成员访问属性。（类设计时可根据需要自行添加数据成员和其他成员函数。）

（1）行车记录仪的芯片可以使用摄像机的摄像、图像质量设定功能。

行车记录仪用户可以操作行车记录仪的操作菜单和摄像机的摄像功能。

（2）行车记录仪的芯片可以使用摄像机的拍摄、图像质量设定功能。

行车记录仪用户仅仅可以操作行车记录仪的操作菜单。

（3）行车记录仪的芯片可以使用摄像机的拍摄、图像质量设定功能。

行车记录仪用户仅仅可以操作行车记录仪的操作菜单

同时其他公司购买行车记录仪，因该公司也用于销售，不得泄露其全部内容

增强：

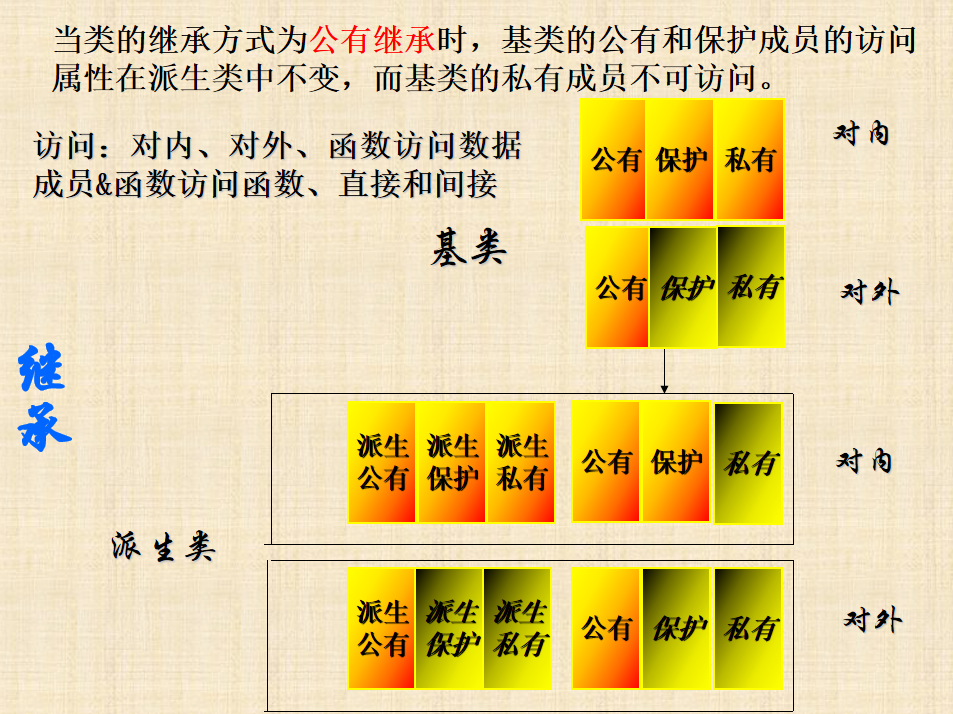
（1）采用组合方式设计行车记录仪类，增加相应测试代码，体验继承和组合的关系。

（2）分别为继承和组合方式下为各类添加构造函数、析构函数，增加相应测试代码，体验对象的初始化和构造顺序。

（3）将摄像机类和行车记录仪类功能相近的函数（如拍摄、编码等功能函数）设为同名函数，增加相应测试代码，体验同名函数覆盖。

（4）为我公司建立一个多态的产品类层次结构，使用抽象类，测试时，创建一个基类指针的容器，通过基类指针调用虚函数，体验多态。

二、解决方案

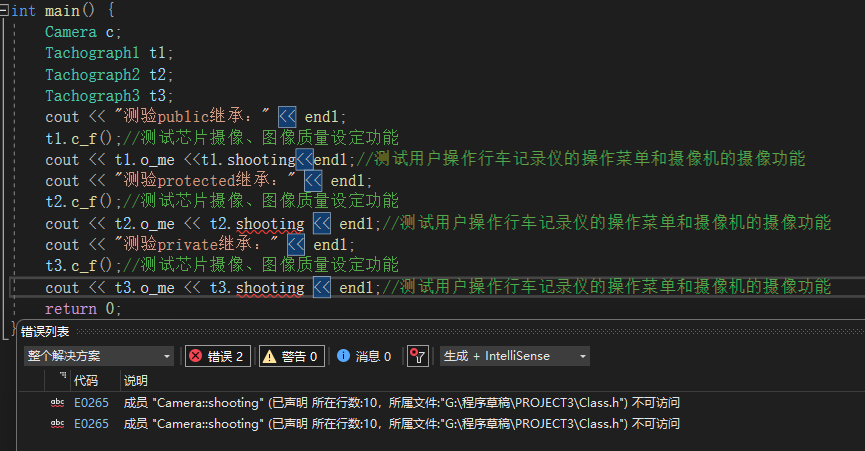


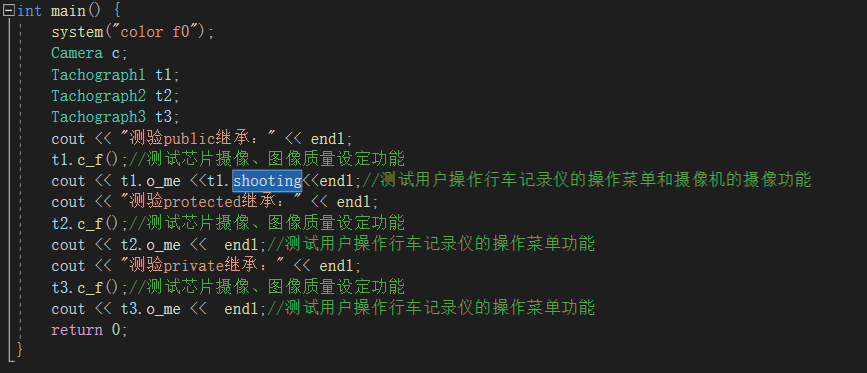
三、程序清单

这里我先行使用char类型变量来简略表示基本要求各种函数功能的实现。



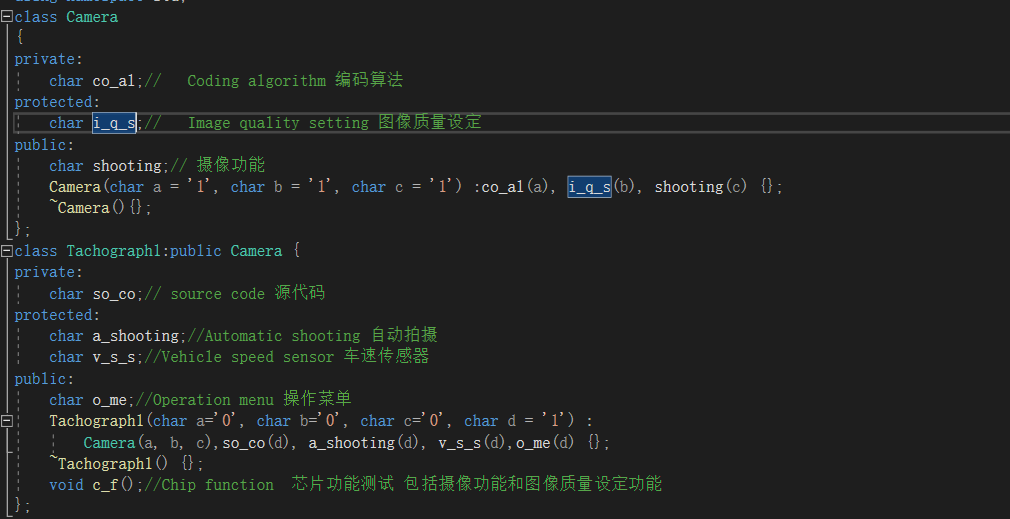
对于基本的三个要求，只需分别将 Tachograph类对应继承的Camera类的继承属性按题目顺序依次改为public、protected、private即可。





以上两张图片充分说明了三种继承关系对于基类的三种数据成员的继承后的属性变化。（第一张说明了继承下来的public成员变成了派生类的protected成员和private成员，所以不能直接在类外访问，也就是由用户来调用查看）。

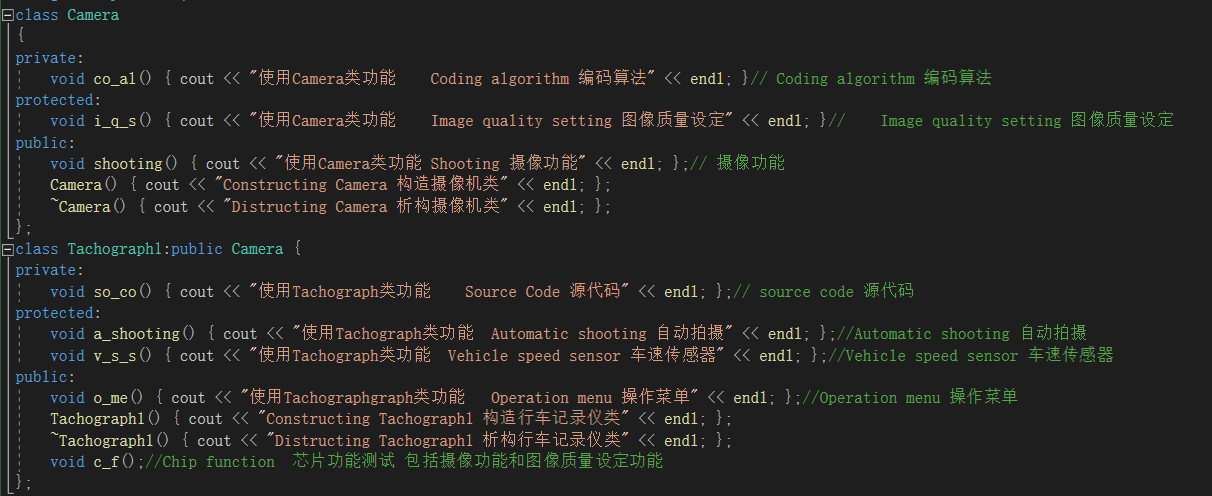
具体类的构造如下：（public继承、protected继承private继承分别由Tachograph1、Tachograph2、Tachograph3代表，即上图的类对象t1/t2/t3）



实现以上基础功能后，为实现增强的功能，我统一将上述char类型数据成员修改为void类型成员函数。（先行使用char类型表示功能也是为了体验初始化列表的知识）

对于增强的第一个要求和第二个要求，此处一并说明，且程序展示结果为程序运行结果的图二和图三。

首先是继承关系下补充构造函数和构造函数：



根据程序运行结果图二能够发现构造和初始化是自派生类向基类，而析构顺序则恰好相反。（protected继承和private继承为Tachograph2和Tachograph3，仅初始化类时初始化列表上所写继承Camera类的方式不同，故下文均仅展示public继承类、组合类行车记录仪类）



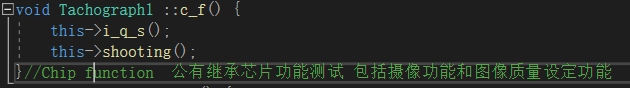
首先是组合方式下对象的初始化与构造，和继承方式下的对象初始化与构造没有特别的区别。

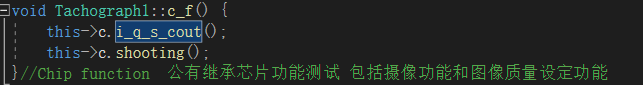
其次，是第一点所要体验的继承和组合的关系：





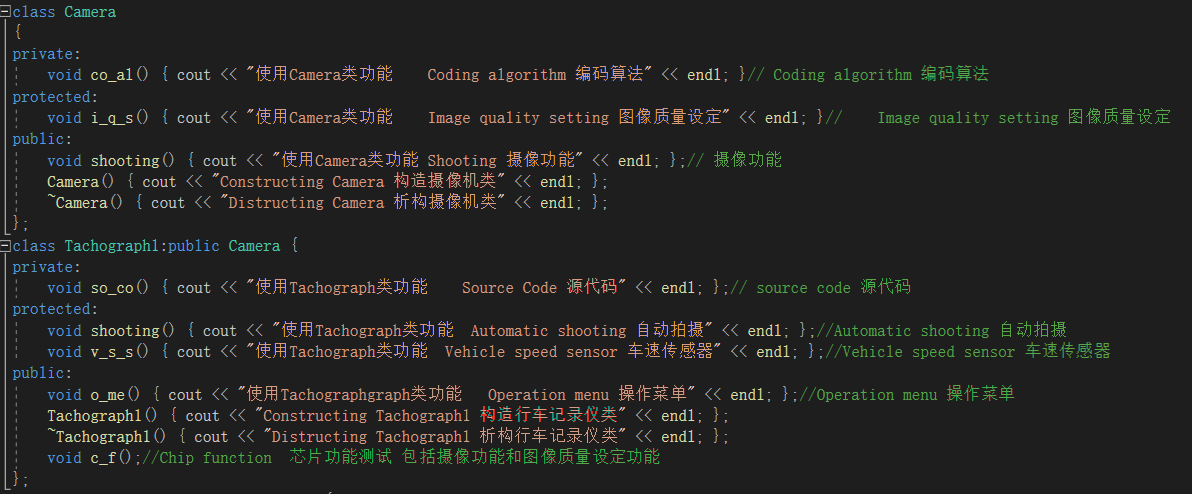
这是第一个不同，构造位置不同，这也意味着在内存空间中的储存位置不同。





这一点首先体现了派生类访问基类的方式不同，继承方式下是通过直接访问，组合方式下是通过访问public（protected和private）类对象：Camera类来实现访问基类函数的。其次是由于访问位置不同造成的，对于基类的protected和private类成员函数访问权限带来的区别，这具体体现在访问Camera类的图像质量设定功能的方式上，如图，组合方式下我通过设置一个在Camera类public域下的输出函数i\_q\_s\_cout()来实现访问基类protected成员的目的，而继承方式下仅仅需要直接访问（public继承情况下）。以上是我在采用组合方式设计行车记录仪类时，体验到的继承和组合的关系。

对于第三个要求，体验同名函数覆盖的要求，我仅仅实现了继承方式的程序，我的代码实现如下（选择了基类的摄像功能和派生类的自动拍摄功能为同名函数shooting）：

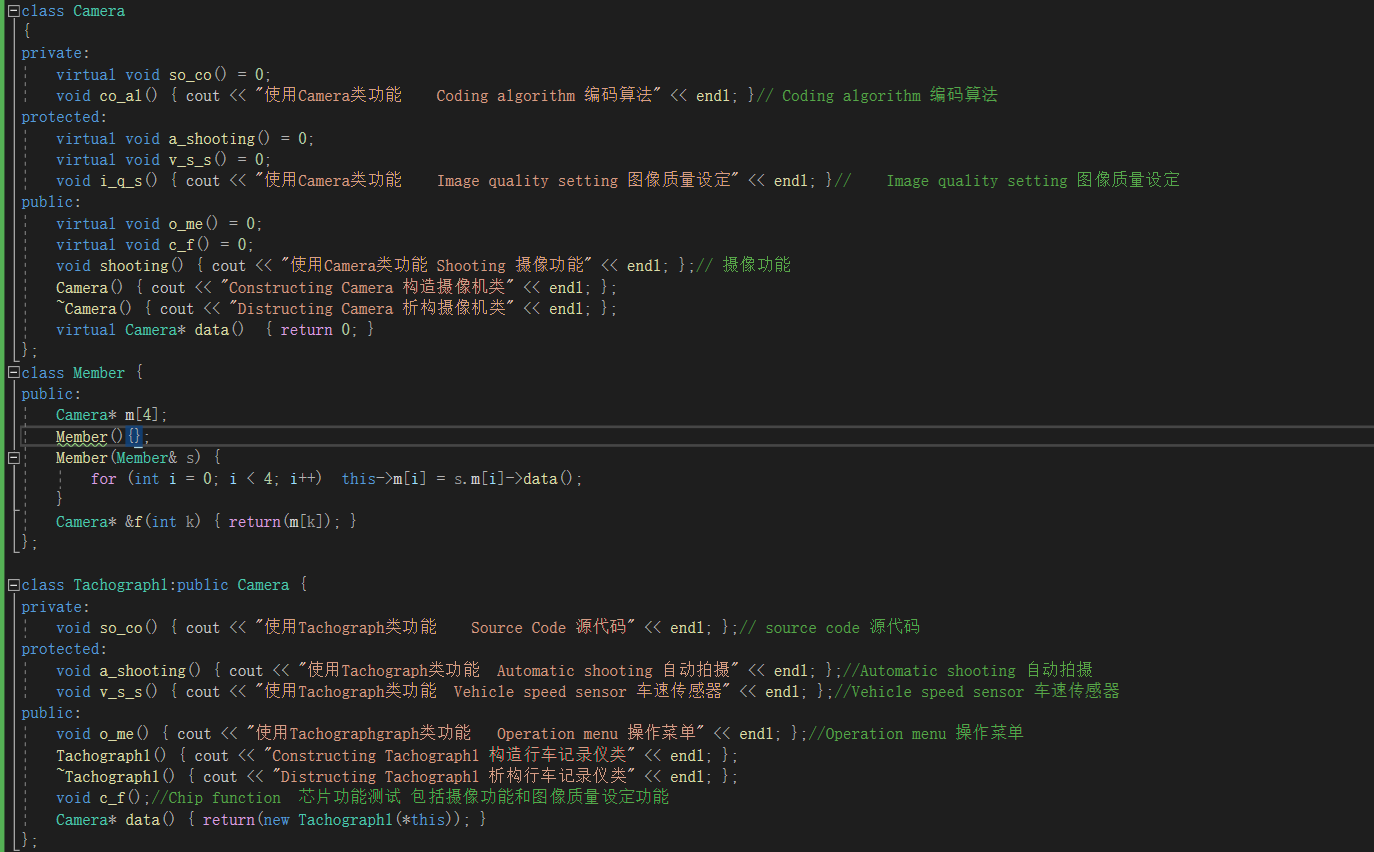


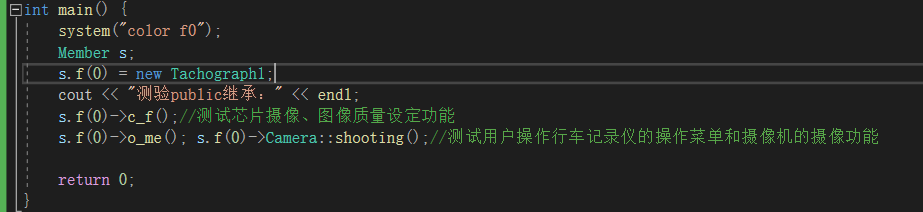
发现在继承方式下，同名函数会默认指向派生类的函数，想访问基类的同名函数，只需：



同理，在组合方式下，访问基类的同名函数，只需访问派生类下的Camera类对象下的同名函数即可。如此体验，我感受到了C++机制对于多态的包容，通过对同名函数在不同作用域下的分界，来实现多而不乱，合理而无冲突。

对于最后一个要求，代码实现如下：





由于私有继承不允许基类指针指向派生类，所以这里以公有继承方式为例，也就是基本要求一来实现，程序运行结果如图四。

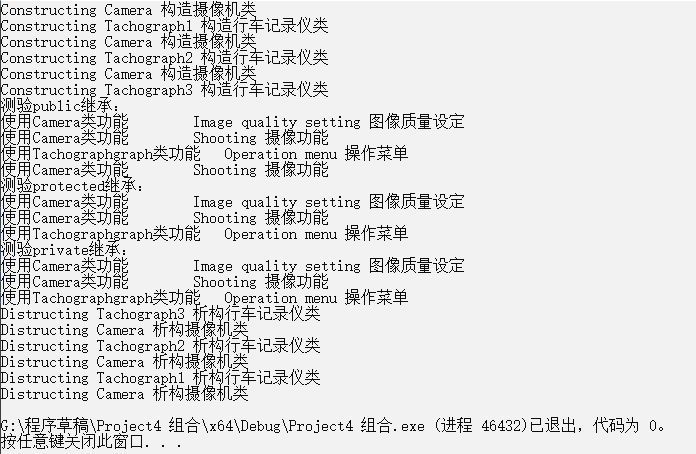
成员如果是数据成员的话访问的是基类的版本（如果派生类中含有相同名字的数据成员的话）。

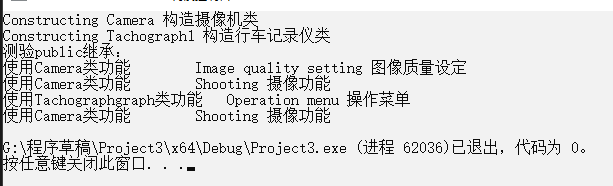
对于已经被重写的方法成员（如果没有重写则没有什么问题），如果方法成员没有用virtual修饰的话访问的仍然是基类的方法，其他方法对基类方法的调用仍然是基类的方法。如果用virtual修饰以后访问的就是派生类的方法。并且基类的方法修改的是基类的数据成员，如果派生类中含有相同名字的数据成员是不会被修改的。如果将该指针赋给一个对象，那么多态的性质将消失。这也是使用基类指针容器的原因。

四、程序运行结果









五、体会与总结

在程序运行过程中，发现这些知识都是老师上课所讲内容的直接应用，所以编写程序中所遇见的问题较少。

1. 111

一、实验题目

第四题：

为某公司设计雇员工资发放系统。

每个雇员的基本信息包括：姓名（name），工号（id）。

雇员的收入取决于雇员的类型。该公司共有四类雇员：

周薪雇员（SalariedEmployee）：收入=固定周薪。

时薪雇员（HourlyEmployee）：若工作40小时以下，收入=小时数\*每小时薪水；若工作40小时以上，收入=40\*每小时薪水+（小时数-40）\*每小时薪水\*150%。

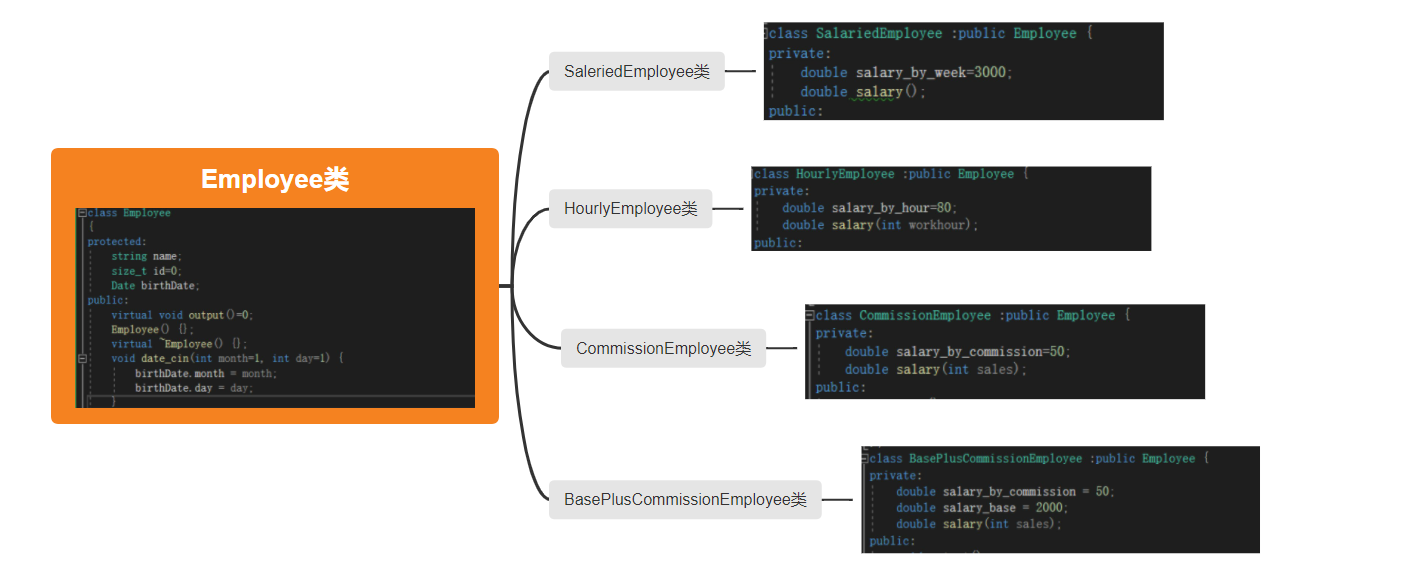
佣金雇员（CommissionEmployee）：收入=销售量\*每个商品的销售佣金

带底薪佣金雇员（BasePlusCommissionEmployee）：收入=底薪+销售量\*每个商品的销售佣金

要求：建立雇员继承层次，每个类包含计算工资和显示输出的功能，可以计算和显示输出公司雇员（Employee）的每周收入。输出时要显示该类雇员的所有信息。（包括雇员类型、姓名、工号、工资各项明细），写出主函数测试各类。在雇员基本信息中增加雇员的生日（birthDate），并设计日期类（Date）来表示生日。在主函数中创建一个容器用来管理公司各种雇员对象，多态地计算并输出每个雇员的工资。如果雇员的生日在本月，就奖给该雇员100元。同时，在本次工资发放阶段，公司决定奖励带薪佣金雇员，把他们的基本工资提高10%。

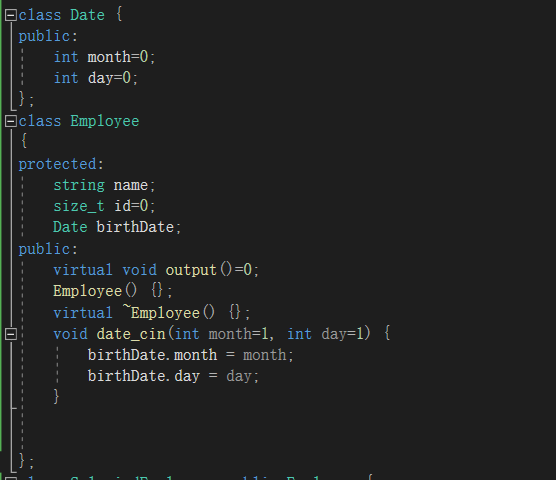
设计提示：使用抽象类。

二、解决方案



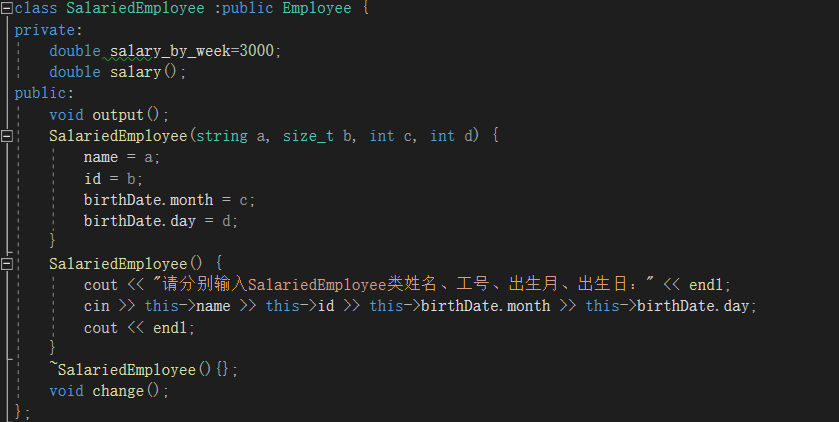
通过运用继承思想，建立一个以雇员（employee）类为基类的多态职员工资系统。

三、程序清单

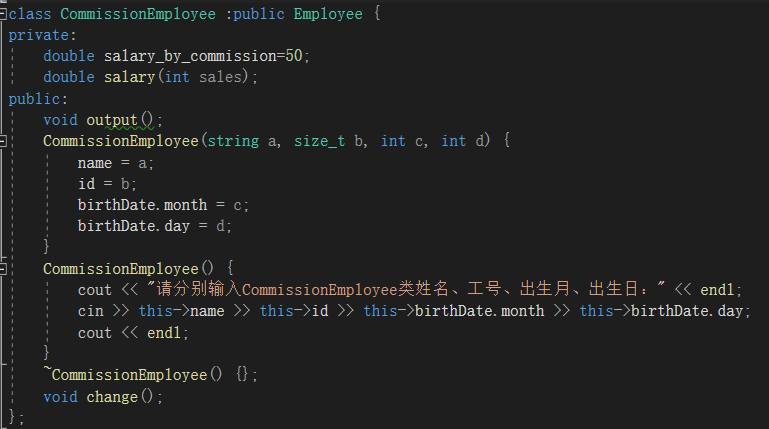


首先建立Date类和Employee基类，这里我选择将name、id、birthDate数据建立在基类中，而Employee类中output()函数和change()为纯虚函数，意味着基类将成为一个抽象类。

然后是建立四个雇员派生类：







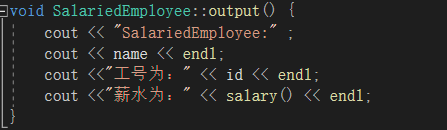


此处其实salary函数也是共同的函数，但是因为我设计成成员函数外传参的形式，所以函数原型不尽相同，故此没有设计成虚函数。

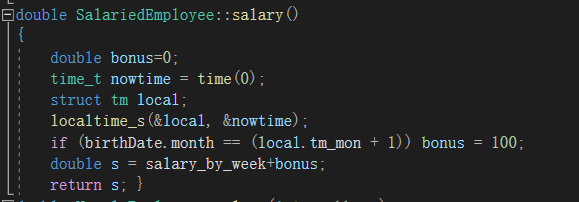
四种派生类均提供了两种构造方式，无参构造和含参构造。代码如上。

下以派生类SalariedEmployee类为例，介绍派生类的功能函数：

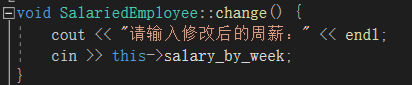
首先是输出函数output()：



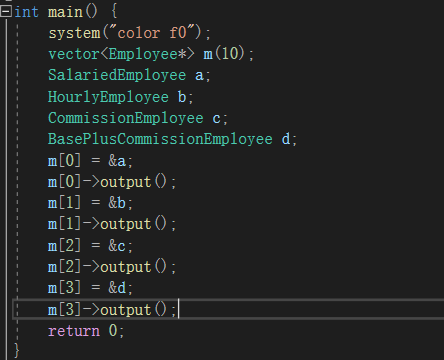
其次是计算总周薪：（在带薪雇员中加了一个乘以1.1的系数）



最后是修改函数change()：



四、程序运行结果

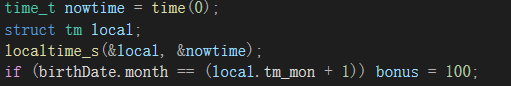




五、体会与总结

此次调试程序没有面临很多问题，但是就判断生日来说，因为要考虑到调试程序时的具体系统时间，所以我上网学习了调用系统时间的库：time.h

在计算薪水的函数中体现了我所额外学习使用的知识。



一、实验题目

第五题：

以下为定义的图形继承体系：

**class Shape{**

**public:**

**// . . .**

**virtual double area(){}**

**};**

**class Circle:public Shape{**

**public:**

**double area(){…}**

**// . . .**

**};**

**class Triangle:public Shape{**

**public:**

**double area(){…}**

**// . . .**

**};**

// . . .

以下为容器类：

**class Manage{**

**shape \*a[100];**

**public:**

**// . . .**

**}**

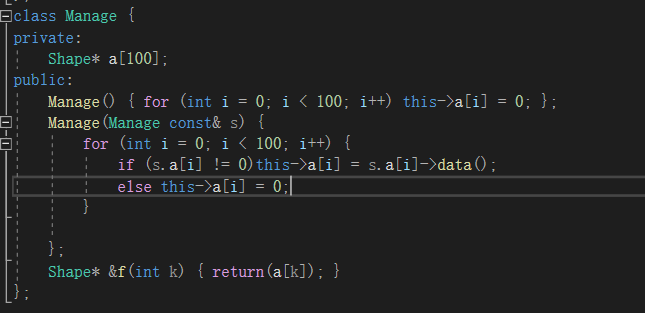
**要求完成Manage类的拷贝构造函数，实现深度拷贝；可以改动所有的类，并给出main函数测试程序。**

二、解决方案

此题思路与上一题大同小异，通过构造一个基类抽象类和基于此派生的派生类，来实现对象类的创建，然后再创建一个指针容器类管理即可。

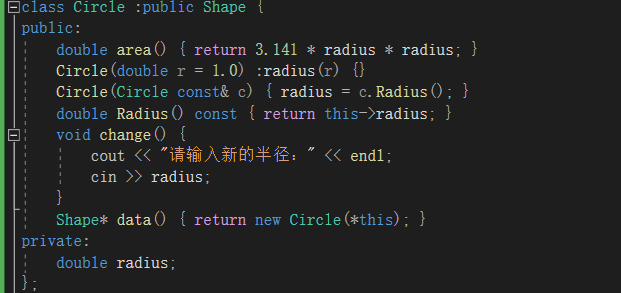
三、程序清单

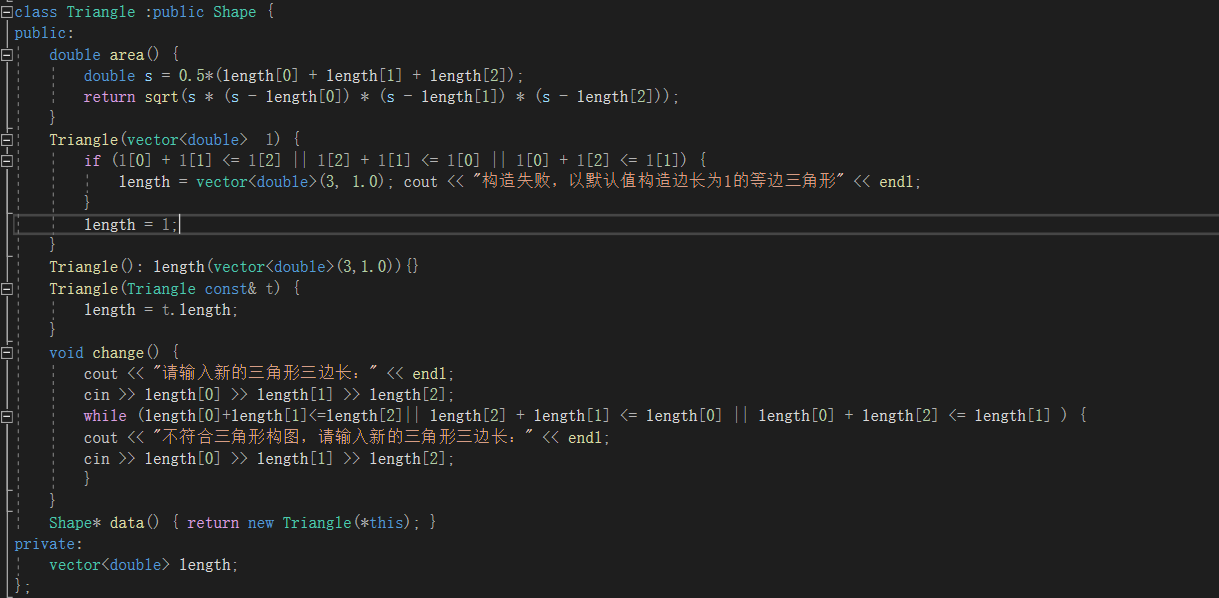
对于第一个要求，实现如下：



创建了一个无参构造函数和一个含参拷贝构造函数，实现了深度拷贝。

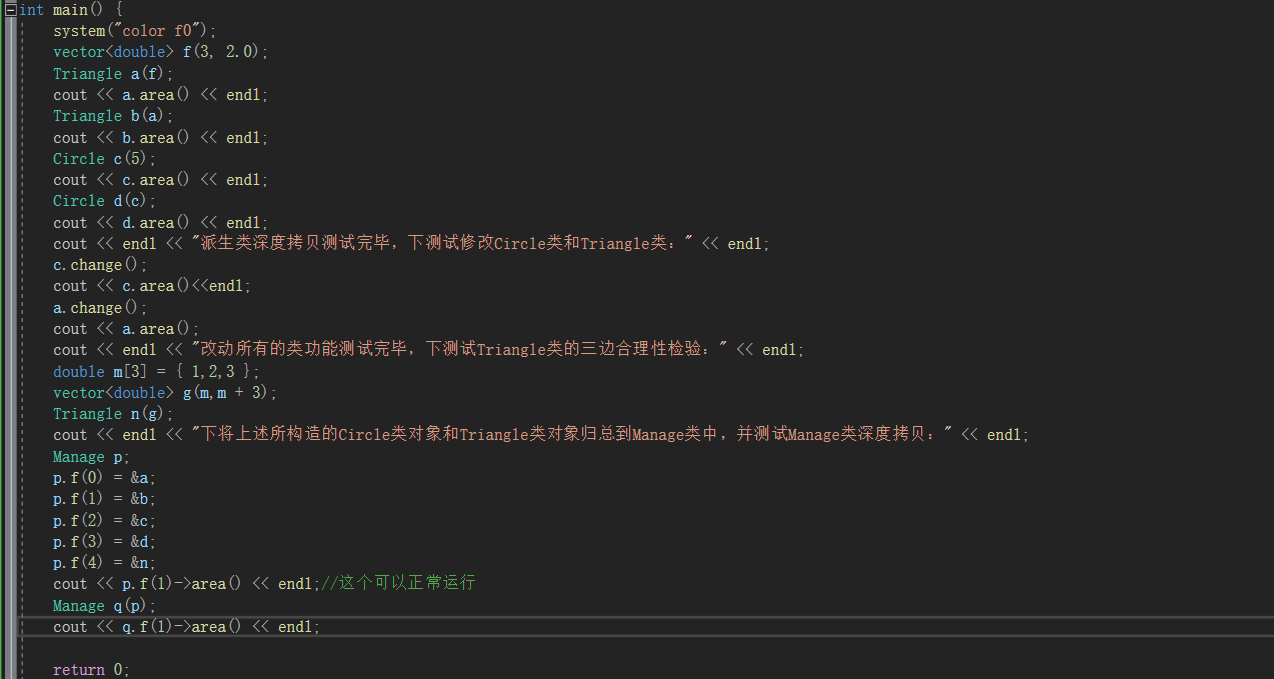
而对于可以改动所有的类这一要求，我的处理方式是，实现派生类已有参数的重新赋值，即Circle类的半径重新赋值、Triangle类的三条边长重新赋值。

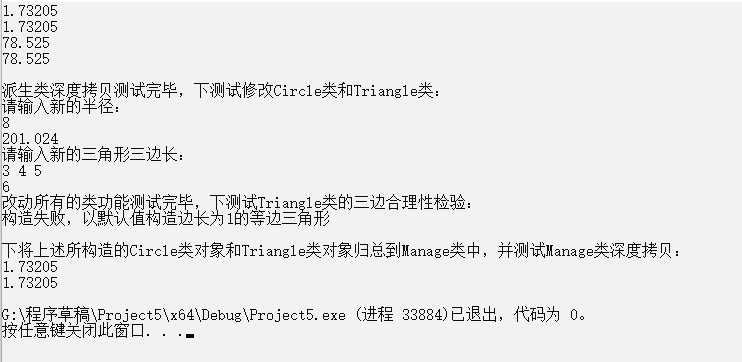




因为此题功能实现代码并不冗杂，所以对应函数实现均是在类的定义中声明的，也就是头文件中。

四、程序运行结果





五、体会与总结

对于这一道题，我对多态知识有了更加全面细致的了解。

撰写此题过程中，遇到的一些小问题整理总结如下：

首先是我发现vector容器在类中初始化的格式发生了改变，最初我希望在类中对向量初始化为三个初始值为一的量，然后使用了vector<double> a(3,1.0)的形式，发现程序报错，希望我使用类型名，因此程序应该是默认为我要将此语句重载为函数，至于为何如此，我上网发现vector库中有个initializer\_list（列表初始化容器 {}初始化），因此如果要保持原有想法，就应该使用列表进行初始化，也就是{1.0,1.0,1.0}的格式，最后我选择了在构造函数中初始化。

其次，我发现指针容器类的private成员需要初始化，不然程序会出现无法检查出来的错误，应该是因为不初始化的话指针就指向了未知的区域，这是有违C++的法则的。

顺带一提，我发现此题隐藏了一个要求，就是要判定构造的三角形是否符合三角形的要求，即两边之长小于第三边，以及边长和圆的半径要大于0，这些隐藏条件就像边界，也是面对对象程序设计中需要注意的点。