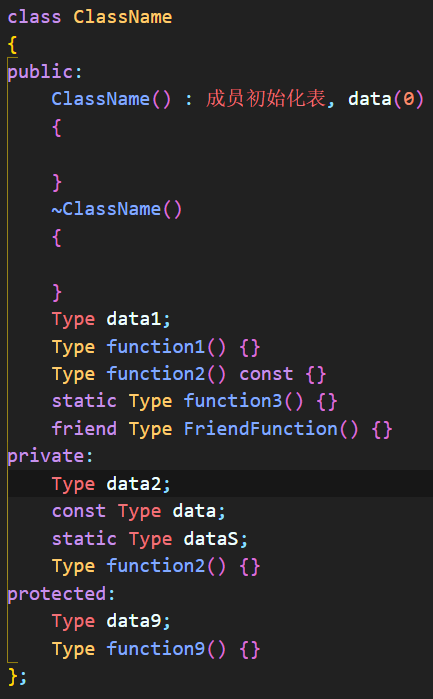
**第14&15周 作业练习题**

**一、第14周上机作业：**

**（一）简答题（请简明扼要地回答下面问题）：**

1. 尝试写一个类，包含最全的基本结构，包括：构造函数（成员初始列表）；析构函数；公开类型(public)数据成员和函数成员，函数成员有一般函数, 带const结尾的函数，静态成员函数，友元函数；私有类型(private)的数据成员，私有的常量；保护类型(protected)的成员数据和函数。
2. 
3. 数据成员初始化的方法有哪些？常量怎样初始化？

构造函数，初始化列表。

1. 类与对象是什么关系？对象有哪些创建方法？对象能使用类的哪些成员？

对象是类的具体实例化。与一般类型一样，直接方法和间接方法（通过指针分配空间）。公开成员。

1. 什么是拷贝构造函数，什么时候会被使用？什么是隐式的拷贝构造函数？

ClassName(ClassName &a){}

隐式的拷贝构造函数将成员数据逐个拷贝

1. const成员函数意味着什么？

class A

{ Type function() const

{}

}

不能修改类的数据成员

1. 什么是静态成员? 静态成员怎样访问？静态成员函数中能使用非静态数据吗？为什么？

class A

{

static Type data;

static Type function(){}

}

静态成员是声明类型前加上static关键字从而只能被实例化一次。

可以通过类名.访问，对象.访问，或对象指针访问。

通常不能，静态成员函数只能被实例化一次，属于类而不是对象，所以只能访问静态数据。

但可以在静态成员函数扇入值中传入该类的指针，使用该指针指向(->)非静态数据,或非静态函数使用。。

1. 友元能否访问对象里的私有成员？为什么能？

友元函数与友元类均可以，使用了friend关键字

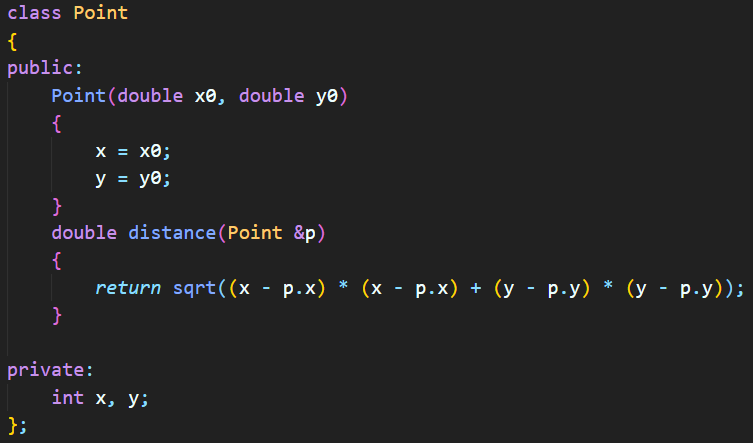
1. **编程题——实例应用（circle）：**
2. **写一个点的类和圆的类**

先写一个点的类，

点包括x,y坐标的成员数据；

一个构造函数Point(double x0, double y0)

一点与另外一点的距离double distance(Point& p)

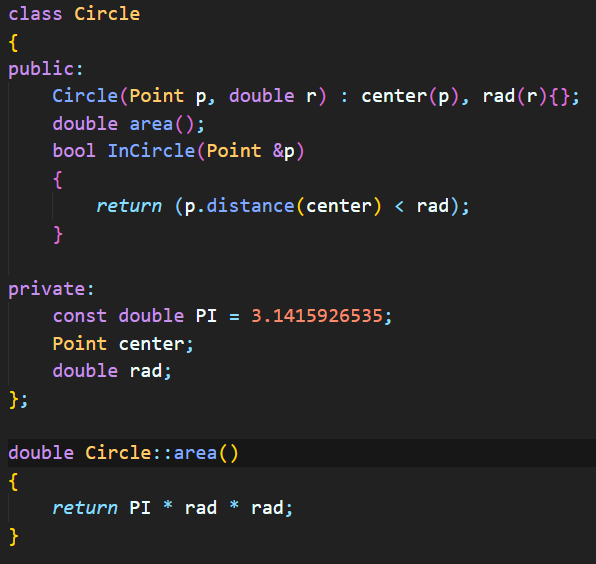


再写一个圆的类，

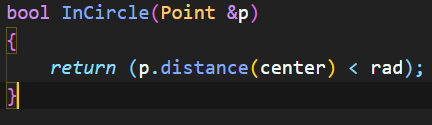
包含Point类型的圆心，半径rad，均为私有类型;

常量PI;

面积double area(), 公开类型;



判断一个点是否在圆内, bool InCircle(Point &p)，用上点类里的距离函数。





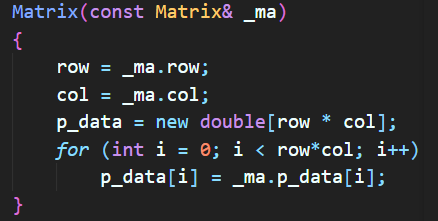
1. **利用附件提供的Matrix, Vector类，设计并编写下面的成员函数。（matrix\_vector）**

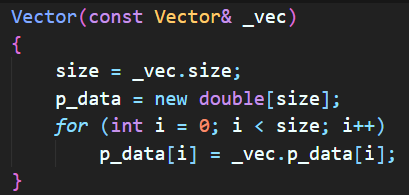
这里的设计与讲义有类似

1. 写Matrix, Vector拷贝构造函数，并测试其正确性；

Matrix(*Matrix& a*){}

Vector(Vector& a){}



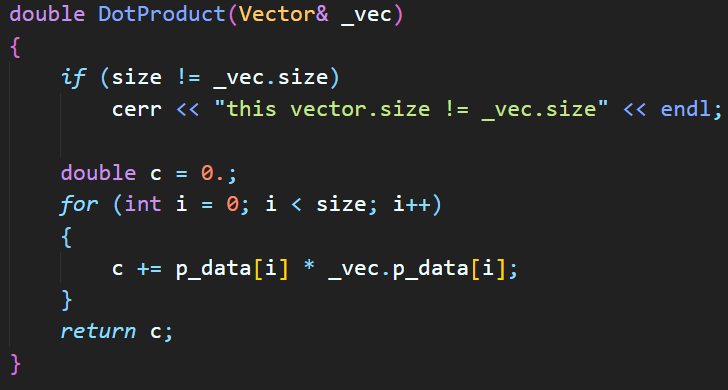


1. 写向量与向量点乘的运算。

double Vector::DotProduct(Vector& b)

{

}

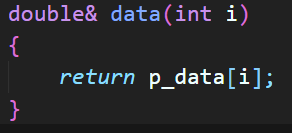


1. 再设计一个Vector的成员函数，返回第i个元素值的引用。

double& Vector::data(int i)

{

}



这个设计的目的是获得第i个元素值的引用后，可以在类外面设置其值。方便第5)问。

1. 以不同方式写矩阵乘以矩阵的函数，函数返回值为矩阵的引用，即在函数里分配好矩阵的空间。这也是为什么我们一定要做好第一问题的原因，如果1)的拷贝构造函数不提供，程序会怎样？

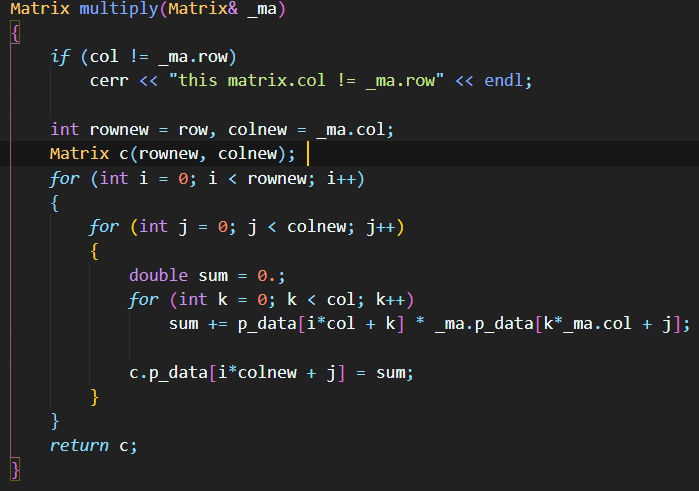
构造函数使用传值方式，则传入的是参数副本，当友元函数传入一个类时，也会产生对象副本，但该副本不会调用构造函数，但当返回时会执行析构函数，而在析构函数内会删除p\_data数组，但该内存空间对原始对象依然有效，即原始对象销毁时会产生同一地方被删除两次的错误

成员函数

Matrix Matrix::Multiply(Matrix& b)

{

}

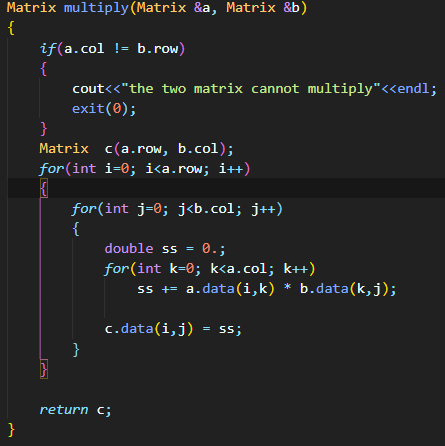


友元函数

friend Matrix Multiply(Matrix& a, Matrix & b)

{

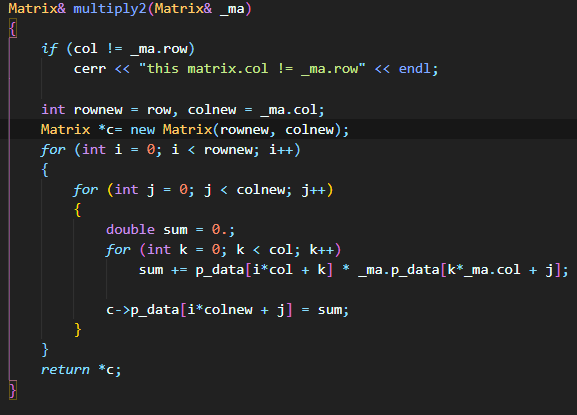
}



如果返回值为Matrix&, 函数应该如何设计? 注意返回的值为矩阵时，这个矩阵要在函数里使用动态分配的方法来构造，避免函数结束时释放空间。

Matrix& Matrix::Multiply2(Matrix& b)

{}



返回值为Matrix与Matrix&，从效率上看，哪个设计更好？为什么？

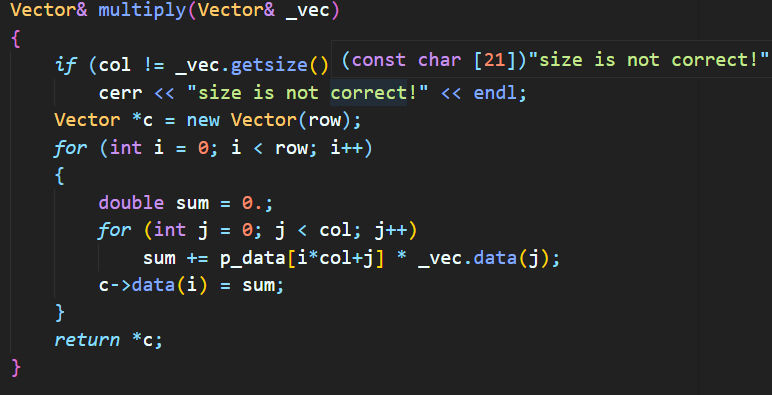
Matrix&设计更好，返回内容可以很大，而使用指针可以减小开销

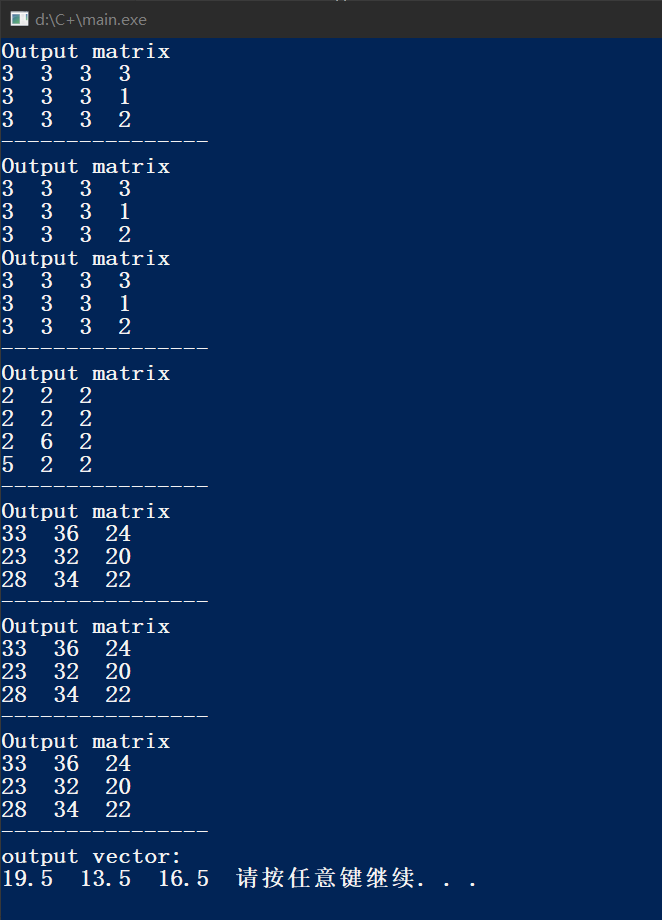
1. 重载一个矩阵和矢量相乘的函数，同样地，注意返回的值为向量时，这个向量要在函数里使用动态分配的方法来构造，避免函数结束时释放空间。

Vector& Matrix::Multiply{Vector& b}

{

}





1. **编程题（student teacher）：**

这里的三个类：Student（一般学生）, GraduatedStudent（研究生）, Teacher.

GraduatedStudent是继承Student的派生类，且增加导师的的成员对象。

Teacher可以拥有多个学生。

要求：

1. 补充GraduatedStudent里构造函数的内容和一个成员函数的内容；
2. 补充Teacher里构造函数的内容和一个成员函数的内容；
3. 上面程序能跑后，再根据下面要求排布程序的结构。注意到这个程序必须分成多文件来写，因为GraduatedStudent用到Teacher的类指针，Teacher用到GraduatedStudent的类指针，相互包含的。请将程序换成5个文件来写。分别是main.C, student.h, student.C, teacher.h, teacher.C. 把所有的类成员的函数实现均放在源文件里。 重新编译，运行。

