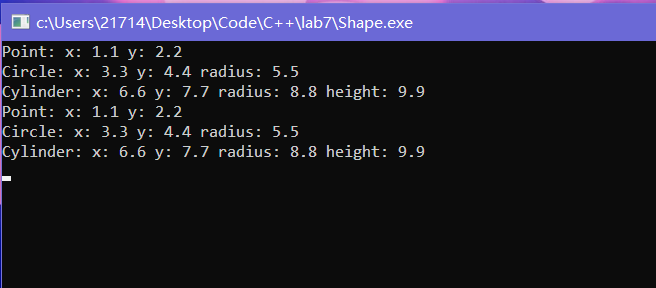
# Lab7

一、运行结果：



二、代码思路：

（1）首先声明一个Shape类，来表示图形的一些基础性质，如名称，面积，体积。由于shapeName是纯虚函数，故Shape类是一个抽象类，无法定义对象。（2）由Shape类作为抽象基类，经过公有继承后派生出Point类，表示一个点的基本性质。Point相对于Shape类多了x坐标和y坐标这两个float型数据成员，以及与其相关的各种函数。同时对虚函数shapeName进行了重新定义，使其与Point类相契合。同时还通过友元函数为Point类的对象重载了<<运算符；（3）由Point类再通过共有继承派生出Circle类，表示一个圆的基本性质。Circle类又添加了float型数据成员radius，以及与其相关的各种函数，同时对shapeName又进行了重新定义，使其与Circle类相契合。同样，也为Circle类通过友元函数重载了<<运算符。（4）由Circle类再通过公有继承派生出Cylinder类，表示一个圆柱的基本性质。Cylinder相对于Circle类又多了float型数据成员height，以及与其相关的各种函数，同时也对shapeName进行了重新定义，使其与Cylinder类相契合。同样也重载了<<运算符用于Cylinder类型成员的基本性质的输出。

在main()函数中，首先分别构造一个Point，一个Circle，一个Cylinder对象，然后再先通过静态关联的方式直接用对象调用shapeName()函数，并用重载的<<运算符输出对象的其他信息。之后再通过使用基态指针进行动态关联的方式分别让pt指向不同的对象来调用不同的shapeName函数，并利用各个对象的成员函数获取信息并输出。

三、回答问题：

（1）静态关联是指在编译时就能够确定某处调用的函数属于哪个类以及函数的具体内容是什么。而动态关联是在运行的过程中根据当时的情况来确定某处调用的函数属于哪个类以及函数的具体内容是什么。以本lab为例，前三行的shapeName()函数的调用就属于静态关联，因为是使用各个对象直接进行调用，故在编译时就能确定调用的函数具体为哪里定义的函数。而后三行的shapeName()函数的调用就属于动态关联，因为是使用基态指针pt进行调用，pt可以指向不同类的对象，故在编译时无法确定pt调用的shapeName()函数究竟属于哪个类，只有在运行过程中确定pt具体指向的对象的类型时才能够确定其所调用的shapeName()函数。

（2）纯虚函数是一种特殊的虚函数，纯虚函数在基类中没有定义，含有纯虚函数的类称为抽象类，无法建立抽象类的对象，故纯虚函数不能被调用，只有在派生类中对该函数进行定义之后，才可以在派生类中使用该函数，如果在派生类中也未对该函数进行定义，则该函数在派生类中仍为纯虚函数，不能被调用。而虚函数在基类中有定义，如果该基类不为抽象类，则可以直接通过该基类的对象调用该虚函数，同时也可以在派生类中重新对该虚函数进行定义，并在派生类中使用新定义的虚函数。以本lab为例，shapeName()属于纯虚函数，故在基类Shape中无定义，Shape类属于抽象类，无法在Shape中调用该函数，只有在Point以及之后的派生类中提供了shapeName()的定义之后才能调用该函数。

（3）对于上述设计，我认为Shape类中的area()和volume()完全可以定义为纯虚函数，因为Shape类没有数据成员，显然是无法给出area()以及volume()这两个函数的定义的。同时在Point类中可以考虑加入area()和volume()，在Circle类中可以考虑加入volume()作为Shape类area()和volume()的定义（如果在Shape中把area()和volume()定义为纯虚函数的话），并让这些加入的函数都直接返回0.0，这样也是符合逻辑的。然后除了以上我感觉不妥的地方之外，其他地方设计的都很合理，可以说是演示静态关联和动态关联的一个很好的案例了。