心得总结（第五章）

本章主要练习了编写虚函数计算程序结构，用友元函数前缀后缀方式重载运算符等。

多态性表示在不同时刻有不同的形态，c++中，多态性总是涉及使用指针或引用来调用对象的函数成员。多态也涉及c++的继承机制.  
多态性意味着，属于一组继承性相关的类的对象可以通过基类指针或引用来传送和操作。  
显然多态性的实现依赖于以下两个要素：  
1. 指针或引用  
2. 继承机制  
多态性可以简单地概括为“一个接口，多种方法”，程序在运行时才决定调用的函数，它是面向对象编程领域的核心概念。多态，字面意思多种形状。  
C++多态性是通过虚函数来实现的，虚函数允许子类重新定义成员函数，而子类重新定义父类的做法称为覆盖，或者称为重写。重写的话可以有两种，直接重写成员函数和重写虚函数，只有重写了虚函数的才能算作是体现了C++多态性，而重载则是允许有多个同名的函数，而这些函数的参数列表不同，允许参数个数不同，参数类型不同，或者两者都不同。编译器会根据这些函数的不同列表，将同名的函数的名称做修饰，从而生成一些不同名称的预处理函数，来实现同名函数调用时的重载问题。但这并没有体现多态性。  
多态与非多态的实质区别就是函数地址是早绑定还是晚绑定。如果函数的调用，在编译器编译期间就可以确定函数的调用地址，并生产代码，是静态的，就是说地址是早绑定的。而如果函数调用的地址不能在编译器期间确定，需要在运行时才确定，这就属于晚绑定。  
那么多态的作用是什么呢，封装可以使得代码模块化，继承可以扩展已存在的代码，他们的目的都是为了代码重用。而多态的目的则是为了接口重用。也就是说，不论传递过来的究竟是那个类的对象，函数都能够通过同一个接口调用到适应各自对象的实现方法。  
最常见的用法就是声明基类的指针，利用该指针指向任意一个子类对象，调用相应的虚函数，可以根据指向的子类的不同而实现不同的方法。如果没有使用虚函数的话，即没有利用C++多态性，则利用基类指针调用相应的函数的时候，将总被限制在基类函数本身，而无法调用到子类中被重写过的函数。因为没  
有多态性，函数调用的地址将是一定的，而固定的地址将始终调用到同一个函数，这就无法实现一个接口，多种方法的目的了。