通过第五章的学习我最主要的是学会了运算符的重载以及了解了什么是虚函数和学会了类型的转换。C++ 允许在同一作用域中的某个函数和运算符指定多个定义，分别称为函数重载和运算符重载。

重载声明是指一个与之前已经在该作用域内声明过的函数或方法具有相同名称的声明，但是它们的参数列表和定义（实现）不相同。

当调用一个重载函数或重载运算符时，编译器通过把所使用的参数类型与定义中的参数类型进行比较，决定选用最合适的定义。选择最合适的重载函数或重载运算符的过程，称为重载决策。

在同一个作用域内，可以声明几个功能类似的同名函数，但是这些同名函数的形式参数（指参数的个数、类型或者顺序）必须不同。您不能仅通过返回类型的不同来重载函数。

重载运算符的两种形式：即重载为类的成员函数||重载为类的非成员函数。

重载为类的非成员函数的时候:通常我们都将其声明为友元函数，因为大多数时候重载运算符要访问类的私有数据，（当然也可以设置为非友元非类的成员函数。但是非友元又不是类的成员函数是没有办法直接访问类的私有数据的），如果不声明为类的友元函数，而是通过在此函数中调用类的公有函数来访问私有数据会降低性能。所以一般都会设置为类的友元函数，这样我们就可以在此非成员函数中访问类中的数据了。

虚函数的实现是由两个部分组成的，虚函数指针与虚函数表。

虚函数指针 (virtual function pointer) 从本质上来说就只是一个指向函数的指针，与普通的指针并无区别。它指向用户所定义的虚函数，具体是在子类里的实现，当子类调用虚函数的时候，实际上是通过调用该虚函数指针从而找到接口。虚函数指针是确实存在的数据类型，在一个被实例化的对象中，它总是被存放在该对象的地址首位，这种做法的目的是为了保证运行的快速性。与对象的成员不同，虚函数指针对外部是完全不可见的，除非通过直接访问地址的做法或者在DEBUG模式中，否则它是不可见的也不能被外界调用。

只有拥有虚函数的类才会拥有虚函数指针，每一个虚函数也都会对应一个虚函数指针。所以拥有虚函数的类的所有对象都会因为虚函数产生额外的开销，并且也会在一定程度上降低程序速度。