运算符重载

1 运算符

C++预定义表示对数据的运算

1） ＋，-， ， /，等

2） 只能用于基本的数据类型,包括整型,实型,字符型,逻辑型。

2 自定义数据类型与运算符重载

1）C++提供了数据抽象的手段：用户自己定义的数据类型(类)；

调用类的成员函数，操作它的对象。

2）类的成员函数在操作对象时很不方便，在数学上，两个复数可以直接进行加减等运算；

但是在C++中直接将+或-用于复数是不允许的。

3 运算符重载

1）对抽象数据类型也能够直接使用C++提供的运算符；

1.程序更简单

2.代码更容易理解

2）运算符重载:

1.对已有的运算符赋予多重的含义

2.使同一运算符作用于不同类型的数据时，有不同类型的行为

3）目的：

扩展C++中提供的运算符的适用范围，以用于类所表示的抽象数据类型

4）同一个运算符，对不同类型的操作数，所发生的行为不同

5）运算符重载的实质是函数重载

返回值类型 opreator 运算符（形参表）

6）在程序编译时：

1.把含运算符的表达式编译成对运算符函数的调用

2.把运算符的操作数成为运算符函数的实参

3.运算符被多次重载时，根据实参的类型决定调用哪个运算符函数

4.运算符可以被重载成普通函数

5.也可以被重载成类的成员函数

7）当运算符重载为：

1.全局函数：参数的个数等于运算符的目数（即操作数的个数）

2.成员函数：参数的个数等于运算符的目数减一

8）类名（构造函数实参表）

这种写法表示生成一个临时对象

4 重载赋值运算符“=”

1.赋值运算符要求左右两个操作数的类型是匹配的，或至少是兼容的

2.如果希望“=”两边的操作数即使不兼容也能够成立，这就要对“=”进行重载

3.C++规定，“=”只能重载为成员函数

4.在对运算符进行重载时，好的风格是应该尽量保留运算符原本的特性

5 浅拷贝和深拷贝

1.浅拷贝：

如果没有经过重载，“=”的作用就是把左边的对象的每个成员变量都变得和右边的对象相等，即逐步执行字节拷贝的工作

2.深拷贝：

将一个对象中指针成员变量指向的内容复制到另一个对象中指针成员变量指向的地方

6 运算符重载为友元函数

一般情况下，将运算符重载为类的成员函数是较好的选择。但有时重载为成员函数不能满足使用要求，重载为全局函数又不能访问类的私有成员，因此需要将运算符重载为友元。