C++程序设计上机报告：运算符重载

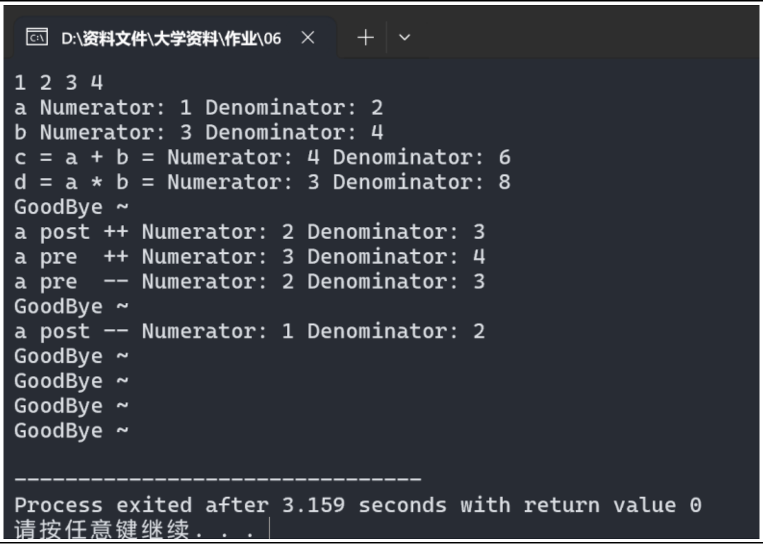
姓名	周新斌	学号	2118140201
班级	软工 212		
目的及要求	（1）掌握运算符重载的方法 （2）编写简单的一元、二元运算符重载的函数。 （3）区分运算符重载为类的成员函数和友元函数的区别。 （4）掌握特殊的运算符输出“<<”和输入“>>”的用法		
上机学时	2 学时		
设备要求	（1）主要仪器设备：微型计算机 （2）软件环境：WINDOWS 2000／XP 操作系统；Visual C++ 程序设计语言		
上机内容	Define classa named Rational that includes two private data,numerator and denominator,and the follow member functions. (1) Constructor; (2) Destructor; (3)Overloaded member functions for +, pre-increment (++), and post-increment (++); (4) Define multiplication(*), pre-decrement (--), post-decrement (--), output operator <<, and input operator >> as friend functions of the class; (5) Define a main function to call all the functions.		

源代码

```
#include<bits/stdc++.h>
using namespace std;

class Rational
{
        double numerator;
        double denominator;

    public:
        Rational(int numerator = 0, int denominator = 0)
        {
            this->numerator = numerator;
            this->denominator = denominator;
        }

        ~Rational()
        {
//          cout << "GoodBye~" << numerator << ' ' <<
denominator << endl;
            cout << "GoodBye ~" << endl;
        }


        Rational operator+(Rational & b)
        {
            Rational tmp;
            tmp.numerator = numerator + b.numerator;
            tmp.denominator = denominator +
b.denominator;
            return tmp;
        }

        double getNumerator()
        {
            return numerator;
        }

        double getDenominator()
        {
            return denominator;
        }

        Rational& operator++()
        {
            ++numerator;
            ++denominator;
            return *this;
        }

        Rational operator++(int)
        {
            Rational old(numerator, denominator);
            numerator++;
            denominator++;
            return old;
        }

        friend Rational& operator--(Rational &r);
        friend Rational operator--(Rational &r, int);

        friend ostream& operator<<(ostream&,
Rational&);
        friend istream& operator>>(istream&, Rational&);

        friend Rational operator*(Rational &a, Rational
&b);
};

Rational operator*(Rational &a, Rational &b)
{
    Rational tmp;
    tmp.numerator = a.numerator * b.numerator;
    tmp.denominator = a.denominator * b.denominator;
    return tmp;
}

Rational& operator--(Rational &r)
{
    --r.numerator;
    --r.denominator;
    return r;
}

Rational operator--(Rational &r, int)
{
    Rational old(r.numerator, r.denominator);
    r.numerator--;
    r.denominator--;
```

```
        return old;
    }

    ostream& operator<<(ostream& out, Rational& obj)
    {
        out << "Numerator: " <<obj.getNumerator() << "
    Denominator: " << obj.getDenominator();
        return out;
    }

    istream& operator>>(istream& in, Rational& obj)
    {
        in >> obj.numerator >> obj. denominator;
        return in;
    }

    int main()
    {
        Rational a, b;
        cin >> a >> b;
        cout << "a " << a << endl;
        cout << "b " << b << endl;

        Rational c = a + b;
        cout << "c = a + b = " << c << endl;

        Rational d = a * b;
        cout << "d = a * b = " << d << endl;

        a++;
        cout << "a post ++ " << a << endl;
        ++a;
        cout << "a pre    ++ " << a << endl;
        --a;
        cout << "a pre    -- " << a << endl;
        a--;
        cout << "a post -- " << a << endl;

        return 0;
}
```

程序的输入描述	1 2 3 4

程序的输出结果	 D:\资料文件\大学资料\作业\06
	`1 2 3 4`
	`a Numerator: 1 Denominator: 2`
	`b Numerator: 3 Denominator: 4`
	`c = a + b = Numerator: 4 Denominator: 6`
	`d = a * b = Numerator: 3 Denominator: 8`
	`GoodBye ~`
	`a post ++ Numerator: 2 Denominator: 3`
	`a pre  ++ Numerator: 3 Denominator: 4`
	`a pre  -- Numerator: 2 Denominator: 3`
	`GoodBye ~`
	`a post -- Numerator: 1 Denominator: 2`
	`GoodBye ~`
	`GoodBye ~`
	`GoodBye ~`
	`GoodBye ~`
	
	`--------------------------------`
	`Process exited after 3.159 seconds with return value 0`
	`请按任意键继续. . .`

程序难点分析	（1） 同是二元运算发重载为成员函数和友元函数的区别是什么？ 成员函数重载：将二元运算符重载为类的成员函数时，该函数将有一个隐含的 this 指针，指向调用该函数的对象。因此，成员函数重载是针对类的具体实例进行操作的，可以直接访问类的私有成员。 友元函数重载：将二元运算符重载为友元函数时，函数不属于类的成员函数，但可以访问类的私有成员。友元函数可以是类的友元，也可以不是，这意味着友元函数可以被多个类所访问。 （2） 为什么输出<<和输入>>运算符只能重载为友元函数？ 对称性：输出和输入运算符通常需要对两个操作数进行操作，分别是要输出的对象和输出流对象（对于输出运算符），或者要输入的对象和输入流对象（对于输入运算符）。为了保持这种对称性，重载时使用友元函数可以访问两个操作数的私有成员。 左操作数不是类的成员：对于输出和输入运算符，左操作数通常不是类的成员。因此，将它们定义为类的成员函数没有太多意义。相反，友元函数可以更自然地处理这种情况，因为它们不必成为类的成员函数，但仍然可以访问类的私有成员。