



河南工业大学

面向对象程序设计

实验报告

项目名称： 运算符重载

专业班级： 联软件 2001

学 号： 202041030102

学生姓名： 张圃源

实验成绩：

批阅老师：

2021 年 11 月 20 日

目录

1	目的要求	1
2	实验内容	1
3	实验说明	1
4	实验过程及结果	2
5	实验小结	3

项目 2 运算符重载 (2 学时)

1 目的要求

(1) 理解运算符重载的方法和时机；(2) 掌握流插入/流提取运算符、算术运算符、关系运算符、赋值运算符、++ 和--运算符等的重载方法。

2 实验内容

(1) 程序编写：根据问题描述和程序的输出结果，对给出的程序代码进行修改，最终给出自己的解决方案，本次实验内容包括：

- 重载 string 类的“+”运算符以实现字符串的连接；
- 重载 HugeInt 类的算术运算符和比较运算符；
- 重载 Rational Number 类的流插入运算符，算术运算符和关系运算符。

(2) 程序调试：根据给出的存在问题的类 Decimal 的程序代码，修改程序中的编译错误使之能够正确地编译执行；对照程序的正确输出结果，修改程序中的逻辑错误使其输出结果和给定的正确输出结果一致。

3 实验说明

这是一个比较哲学的问题：我们为什么要重载运算符？

理由就是，我们 C++ 语言中已经给出的运算符（包括算数运算符和逻辑运算符）只是针对 C++ 语言中已经给定的数据类型进行运算，假如我们想要对我们的自定义数据类型进行运算的话，则需要重载运算符，我们可以把重载运算符理解成对已有的运算符的一种重新定义。

比如：

```
double a, b, c;  
a=1/3;  
b=1/2;
```



```
c=a+b;  
printf("%lf",c);
```

这段程序输出的肯定不是两个分数相加的结果。这时候我们就可以重载运算符 +。

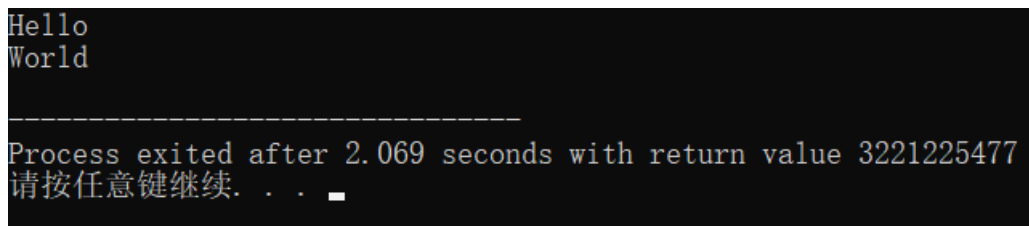
4 实验过程及结果

```
#include<iostream>  
#include<string.h>  
using namespace std;  
class String  
{  
    public:  
        String()  
        {  
            p=NULL;  
        }  
        String(char *str);  
        friend String operator + (String &string1, String &  
            string2);  
        void display();  
        ~String()  
        {  
            delete [] p;  
        }  
    private:  
        char *p;  
        int len;  
};  
String::String(char *str)  
{  
    p=str;  
}  
  
void String::display()
```

```
{
    cout<<p<<endl;
}

String operator + (String &string1 ,String &string2)
{
    String string3;
    string3.len=string1.len+string2.len;
    string3.p=new char [string3.len+1];
    strcpy(string3.p,string1.p);
    strcat(string3.p,string2.p);
    return string3;
}

int main()
{
    String s1("Hello"),s2("World"),s3;
    s1.display();
    s2.display();
    s3=s1+s2;
    s3.display();
    return 0;
}
```



```
Hello
World
-----
Process exited after 2.069 seconds with return value 3221225477
请按任意键继续. . .
```

图 1: 运行截图

5 实验小结

什么是运算符的重载?

运算符与类结合, 产生新的含义。

为什么要引入运算符重载?

作用: 为了实现类的多态性 (多态是指一个函数名有多种含义)

怎么实现运算符的重载?

方式：类的成员函数或友元函数（类外的普通函数）

规则：不能重载的运算符有. 和.* 和?: 和:: 和 sizeof

友元函数和成员函数的使用场合：一般情况下，建议一元运算符使用成员函数，二元运算符使用友元函数

1、运算符的操作需要修改类对象的状态，则使用成员函数。如需要做左值操作数的运算符（如 =, +=, ++）

2、运算时，有数和对象的混合运算时，必须使用友元

3、二元运算符中，第一个操作数为非对象时，必须使用友元函数。如输入输出运算符 « 和 »