# 实验十三、函数重载和运算符重载

**计算机学院 熊明 20305055**

## 一、实验目的

1. 掌握运算符重载的定义及实现。
2. 掌握友元函数的定义方法。
3. 掌握用友元函数重载运算符的方法。
4. 掌握用类成员函数重载运算符的方法

## 二、实验原理

在实际程序设计中，有时候我们需要实现几个功能类似的函数，只是有些细节不同。例如希望在显示器上输出显示一个变量的值，这个变量有多种类型，可以是 int、float、char、bool 等。在C语言中，程序员往往需要分别设计出三个不同名的函数。但在C++中，这完全没有必要。C++ 允许多个函数拥有相同的名字，只要它们的参数列表不同就可以，这就是函数的重载。借助重载，一个函数名可以有多种用途。参数列表又叫参数签名，包括参数的类型、参数的个数和参数的顺序，只要有一个不同就叫做参数列表不同。

函数重载（Function Overloading）是指在同一作用域内（同一个类、同一个命名空间等），有一组具有相同函数名，不同参数列表的函数，这组函数被称为重载函数。重载函数通常用来命名一组功能相似的函数，这样做减少了函数名的数量，避免了名字空间的污染，对于程序的可读性有很大的好处。函数重载是一种静态多态。函数重载的规则：

1. 函数名称必须相同。
2. 参数列表必须不同（个数不同、类型不同、参数排列顺序不同等）。
3. 函数的返回类型可以相同也可以不相同。
4. 仅仅返回类型不同不足以成为函数的重载。
5. 使用函数重载时，对哪一函数进行调用，千万要注意不能引起歧义。例如：
   - void rotate（int&，int&）；
   - void rotate（int，int）；

   产生值传递和引用传递表意含混不清的情况。
6. 使用函数重载时，注意C++在函数调用时有时候会进行类型自动转换

## 三、实验内容

1. > 用C++编写程序：设计Student类，构造函数初始化学生信息（学号，姓名，成绩），重载提取运算符>>和插入<<实现学生信息的输入输出功能。

```
//student.h，类定义
#include<iostream>
#include<string>
using namespace std;
class Student {
private://学号，姓名，成绩
    string num;
    string name;
```

```
    double score;
public:
    Student(string = "0", string = "zhangsan", double = 0.00);
    friend istream& operator>>(istream& is,Student &A);
    friend ostream& operator<<(ostream& os, Student& A);
    ~Student()
};
```

```
//student.cpp
#include "student.h"

Student::Student(string num, string name, double score) {
    this->num = num;
    this->name = name;
    this->score = score;
}

Student::~Student()
{
}

istream& operator>>(istream& is, Student& A) {
    cout << "enter num: ";
    is >> A.num;
    cout << "enter name: ";
    is >> A.name;
    cout << "enter score: ";
    is >> A.score;
    return is;
}

ostream& operator<<(ostream& os, Student& A) {
    cout << "student num is: " << A.num << endl;
    cout << "student name is: " << A.name << endl;
    cout << "student score is: " << A.score << endl;
    return os;
}
```

类外定义

```
//test.cpp
#include<iostream>
#include<string>
#include"student.h"
using namespace std;
int main() {
    Student s1;
    Student s2;
    cout << s1;
    cin >> s2;
    cout << s2;
}
```

测试代码，s1是默认构建的student对象，学号为0，姓名为zhangsan，分数为0；

s2是用户键入对象。结果如下：

```
student num is: 0
student name is: zhangsan
student score is: 0
enter num: 20305055
enter name: xm
enter score: 99.9
student num is: 20305055
student name is: xm
student score is: 99.9
```

2. 用C++编写程序：有两个整数矩阵a和b，均为2行3列。求两个矩阵之和。重载运算符“+”，使之能用于矩阵相加。如c=a+b。

```cpp
//matrix.h
#include<iostream>
using namespace std;

class Matrix {
private:
    double matrix[2][3];
public:
    Matrix(){
        this->matrix[0][0] = { 0 };
    }
    void set_value() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 3; j++) {
                cout << "enter " << i + 1 << " , " << j + 1 << " value:" << endl;
                cin >> this->matrix[i][j];
            }
        }
    }
    Matrix operator+(Matrix& A) {
        Matrix res;
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 3; j++) {
                res.matrix[i][j] = this->matrix[i][j] + A.matrix[i][j];
            }
        }
        return res;
    }
    void display() {
        cout << "[";
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 3; j++) {
                cout << this->matrix[i][j];
                if (j != 2) cout << " ";
            }
            if (i == 1)cout << "]";
            cout << endl;
        }
```

```
    }
};
```

测试代码：

```
#include<iostream>
#include"matrix.h"
using namespace std;
int main() {
    Matrix a,b,c;
    a.set_value();
    b.set_value();
    c = a + b;
    cout << "a: " << endl;
    a.display();
    cout << "b: " << endl;
    b.display();
    cout << "c: " << endl;
    c.display();
}
```

运行结果：

```
enter 1 , 1 value:
1
enter 1 , 2 value:
2
enter 1 , 3 value:
3
enter 2 , 1 value:
4
enter 2 , 2 value:
5
enter 2 , 3 value:
6
enter 1 , 1 value:
7
enter 1 , 2 value:
8
enter 1 , 3 value:
9
enter 2 , 1 value:
10
enter 2 , 2 value:
11
enter 2 , 3 value:
12
a:
[1 2 3
4 5 6]
b:
[7 8 9
```

> 10 11 12]
> c:
> [8 10 12
> 14 16 18]

## 四、实验心得体会

无