

# 计算机科学基础II

## 实验十二 运算符重载

曹鹏

Email: [caopeng@seu.edu.cn](mailto:caopeng@seu.edu.cn)

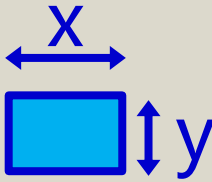

Tel: 13851945861

# 实验目的

---

1. 掌握运算符重载为成员函数的方法。
2. 理解友元函数和友元类。
3. 掌握运算符重载为友元函数的方法。
4. 使用静态数据成员。

## 在教材【例4.2】基础上增加运算符重载函数

运算符	功能	图示
加复合赋值 +=	固定长方形的左上角，对右下角的坐标进行 <b>加</b> 运算，使新矩形的长宽为原两矩形长宽之 <b>和</b>	 $+=$ 
减复合赋值 -=	固定长方形的左上角，对右下角的坐标进行 <b>减</b> 运算，使新矩形的长宽为原两矩形长宽之 <b>差</b>	 $-=$ 
加 +	矩形相 <b>加</b> ，新矩形的长宽等于rect1的长度和宽度分别 <b>加上</b> rect2的长度和宽度	 $+$
减 -	矩形相 <b>减</b> ，新矩形的长宽等于rect1的长度和宽度分别 <b>减去</b> rect2的长度和宽度	 $-$

## ◆成员函数方式声明/定义

- ◆ 加复合赋值 +=
- ◆ 减复合赋值 -=
- ◆ 加 +
- ◆ 减 -

## ◆友元函数方式声明/定义

- ◆ 加复合赋值 +=
- ◆ 减复合赋值 -=
- ◆ 加 +
- ◆ 减 -

# 主函数（输出）

```
int main(){
    Rectangle rect;
    cout<<"初始rect:"<<endl;
    rect.Show();
    rect.Assign(100,200,300,400);
    cout<<"赋值后rect:"<<endl;
    rect.Show();
    Rectangle rect1(0,0,200,200);
    cout<<"初始rect1:"<<endl;
    rect1.Show();
    rect+=rect1;
    cout<<"与rect1相加后的rect:"<<endl;
    rect.Show();
    rect-=rect1;
    cout<<"减去rect1后的rect:"<<endl;
    rect.Show();
    Rectangle rect2;
    rect2 = rect+rect1;
    cout<<"rect与rect1相加所得rect2:"<<endl;
    rect2.Show();
    rect2 = rect-rect1;
    cout<<"rect减去rect1所得rect2:"<<endl;
    rect2.Show();
    return 0;
}
```

初始rect:  
left-top point is (0,0)  
right-bottom point is (0,0)  
赋值后rect:  
left-top point is (100,200)  
right-bottom point is (300,400)  
初始rect1:  
left-top point is (0,0)  
right-bottom point is (200,200)  
与rect1相加后的rect:  
left-top point is (100,200)  
right-bottom point is (500,600)  
减去rect1后的rect:  
left-top point is (100,200)  
right-bottom point is (300,400)  
rect与rect1相加所得rect2:  
left-top point is (100,200)  
right-bottom point is (500,600)  
rect减去rect1所得rect2:  
left-top point is (100,200)  
right-bottom point is (100,200)

# 成员函数声明

```
class Rectangle {  
    int left, top ;  
    int right, bottom;  
public:  
    ...  
    Rectangle operator+=(Rectangle&);  
    Rectangle operator-=(Rectangle&);  
    Rectangle operator+(Rectangle&);  
    Rectangle operator-(Rectangle&);  
};
```

# 成员函数实现

```
Rectangle Rectangle::operator+=(Rectangle& rect){  
    int x = rect.right - rect.left;  
    int y = rect.bottom - rect.top;  
    right += x;  
    bottom += y;  
    return Rectangle(left,top,right,bottom);  
    //return *this; //也可以这样  
}
```

```
Rectangle Rectangle::operator-=(Rectangle& rect){  
    int x = rect.right - rect.left;  
    int y = rect.bottom - rect.top;  
    right -= x;  
    bottom -= y;  
    return Rectangle(left,top,right,bottom);  
    //return *this; //也可以这样  
}
```

# 成员函数实现

```
Rectangle Rectangle::operator- ( Rectangle &rect){  
    //矩形相减, 新矩形的长宽等于rect1的长度和宽度分别减去rect2的  
    长度和宽度  
    Rectangle tmp(*this);    //也可以写 rect=*this;  
    return tmp-=rect;  
}
```

```
Rectangle Rectangle::operator+ ( Rectangle &rect){  
    //矩形相加, 新矩形的长宽等于rect1的长度和宽度分别加上rect2的  
    长度和宽度  
    Rectangle tmp(*this);    //也可以写 rect=*this;  
    return tmp+=rect;  
}
```



# 友元函数声明

```
class Rectangle {  
    int left, top ;  
    int right, bottom;  
public:  
    ...  
    friend Rectangle operator+=(Rectangle &, Rectangle&);  
    friend Rectangle operator-=(Rectangle &, Rectangle&);  
    friend Rectangle operator+(Rectangle &, Rectangle&);  
    friend Rectangle operator-(Rectangle &, Rectangle&);  
};
```

# 友元函数实现

```
Rectangle operator+=(Rectangle &rect1, Rectangle& rect2){  
    int x = rect2.right - rect2.left;  
    int y = rect2.bottom - rect2.top;  
    rect1.right += x;  
    rect1.bottom += y;  
    return Rectangle(rect1.left,rect1.top,rect1.right,rect1.bottom);  
}
```

```
Rectangle operator-=(Rectangle &rect1, Rectangle& rect2){  
    int x = rect2.right - rect2.left;  
    int y = rect2.bottom - rect2.top;  
    rect1.right -= x;  
    rect1.bottom -= y;  
    return Rectangle(rect1.left,rect1.top,rect1.right,rect1.bottom);  
}
```

# 友元函数实现

```
Rectangle operator- ( Rectangle &rect1, Rectangle& rect2){  
    //矩形相减, 新矩形的长宽等于rect1的长度和宽度分别减去rect2的  
    长度和宽度  
    Rectangle rect(rect1); //也可以写 rect=rect1;  
    return rect-=rect2;  
}
```

```
Rectangle operator+ ( Rectangle &rect1, Rectangle& rect2){  
    //矩形相加, 新矩形的长宽等于rect1的长度和宽度分别加上rect2的  
    长度和宽度  
    Rectangle rect(rect1); //也可以写 rect=rect1;  
    return rect+=rect2;  
}
```

---



# End