

# 计算机学院\_高级语言程序设计\_课程实验报告

实验题目： 多态（一） 运算符重载（静态_编译时多态）		学号： 202200130048
日期： 2023. 4. 18	班级： 6	姓名： 陈静雯
Email： 1205037094@qq. com		

## 实验步骤与内容：

1. 练习第 8 章 PPT 例 8-1，复数类加减法用成员函数实现运算符重载，形参是引用。截图并回答一下问题
2. 练习第 8 章 PPT 例 8-2，单目运算符重载，返回类型是引用。在 main() 中试用++++myclock 和后++++，看能否实现连续加 1？
3. 实践第 8 章 PPT 例 8-3，以非成员函数形式重载 Complex 的加减法运算和“<<”运算符，两个形参都为引用。不用友元函数能不能实现对各操作符的非成员函数的重载？如果可以的话可以通过什么样的方法？
4. 将课上常用的 Point 类（有数据成员 x 和 y）用友元函数或成员函数的方式实现双目操作符+，-，=，==的重载。
5. 通过重载赋值操作符实现深拷贝。

## 结论分析与体会：

1.

```
or- 在编辑窗口中打开文件 (Ctrl + 单击) t-MIEngine-Pid-3ykn4l3o.fsj'
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
c1 = (5, 4)
c2 = (2, 10)
c3 = c1 - c2 = (3, -6)
c3 = c1 + c2 = (7, 14)
PS D:\code repository\code> █
```

(1)

```
✓ G++ 000.cpp cplusplus
✗ assignment of member 'Complex::real' in read-only object gcc [行 25, 列 14]
✗ assignment of member 'Complex::imag' in read-only object gcc [行 26, 列 16]
```

分析：不能，无法修改等号前面的对象值，“read-only”就是只读不能改

(2)

```
or-xazmzokd.qss --pid=Microsoft-MIEngine-Pid-00ze10vs.qmr
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
c1 = (5, 4)
c2 = (2, 10)
c3 = c1 - c2 = (3, -6)
c3 = c1 + c2 = (7, 14)
PS D:\code repository\code> █
```

分析：能，const 只是保证参数不被改变，对加法运算无影响。

(3)

```
Complex Complex::operator + (const double &db) const{  
    return Complex(real + db,imag);  
}
```

```
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'  
c1 = (5, 4)  
c2 = (2, 10)  
c3 = c1 - c2 = (3, -6)  
c3 = c1 + c2 = (7, 14)  
(5.5, 4)  
PS D:\code_repository\code> █
```

2.

```
or-b2stc1yr.1ij' '--pid=Microsoft-MIEngine-Pid-e435w3rm.zbs' '--dl  
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'  
First time output: 23:59:59  
Show myClock++: 23:59:59  
0:0:0  
Show ++myClock: 0:0:2  
PS D:\code_repository\code> █
```

分析：后加加不能连续加，只能加 1；前加加可以连续加 1。具体原因应该是和后置++运算时内部原理不同，后++是返回一个临时对象，所以不能连续加；前置++是在原来参数的基础上直接加，所以可以连续。

3.

```
or-vzicbm5b.ny3' '--pid=Microsoft-MIEngine-Pid-tpqs2cc.rh2' '--dbgE  
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'  
c1 = (5, 4)  
c2 = (2, 10)  
c3 = c1 - c2 = (3, -6)  
c3 = c1 + c2 = (7, 14)  
PS D:\code_repository\code> █
```

```
3 class Complex { //复数类定义  
4 public: //外部接口  
5     Complex(double r = 0.0, double i = 0.0) : real(r), imag(i)  
6  
7     double real; //复数实部  
8     double imag; //复数虚部  
9 private: //私有数据成员  
10
```

分析：可以是可以，直接把私有成员变成公有成员，就可以直接全局函数重载然

后使用了。

4.

```
#include <iostream>
using namespace std;
class point { //复数类定义
public:    //外部接口
    point(double r = 0.0, double i = 0.0) : x(r), y(i) { }
    point operator + (const point &c2) const;
    point operator - (const point &c2) const;
    void operator = (const point &c2);
    int operator == (const point &c2);
    void display() const;
private:
    double x;
    double y;
};

point point::operator + (const point &c2) const {
    return point(x + c2.x, y + c2.y);
}

point point::operator - (const point &c2) const {
    return point(x - c2.x, y - c2.y);
}

void point::display() const {
    cout << "(" << x << ", " << y << ")" << endl;
}

int point::operator == (const point &c2) {
    if(x==c2.x&&y==c2.y) {
        return 1;
    }
    else{
        return 0;
    }
}

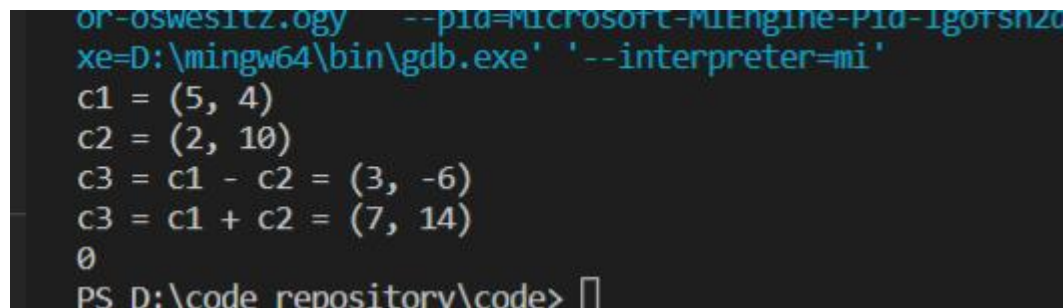
void point::operator = (const point &c2) {
    x = c2.x;
    y = c2.y;
}

int main() {
    point c1(5, 4), c2(2, 10), c3;
    cout << "c1 = "; c1.display();
    cout << "c2 = "; c2.display();
    c3 = c1 - c2;
```

```

    cout << "c3 = c1 - c2 = "; c3.display();
    c3 = c1 + c2;
    cout << "c3 = c1 + c2 = "; c3.display();
    if(c1==c2) {
        cout<<"1";
    }
    else{
        cout<<"0";
    }
    return 0;
}

```



```

or-oswesit2.org --pid=Microsoft-MIEngine-Pid-1gotsh20
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
c1 = (5, 4)
c2 = (2, 10)
c3 = c1 - c2 = (3, -6)
c3 = c1 + c2 = (7, 14)
0
PS D:\code repository\code>

```

5.

(1)

```

void MyArray::operator = (const MyArray &arr2) {
    if(array_size!=arr2.array_size) {
        cout<<"size error"<<endl;
    }
    if(array_size>=arr2.array_size) {
        for(int i=0;i<arr2.array_size;i++) {
            arr[i]=arr2.arr[i];
        }
    }
}

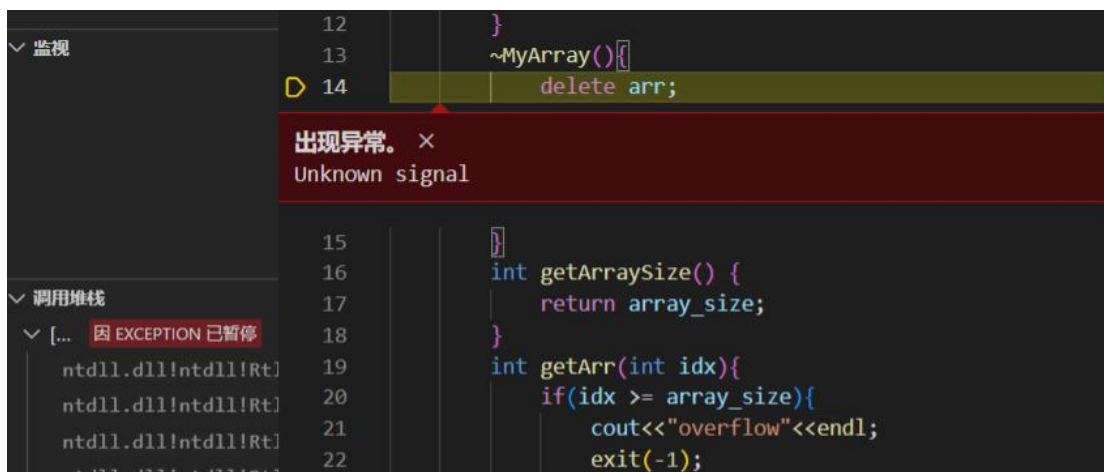
int main() {
    MyArray arr1(11), arr2(10);
    for(int i=0;i<10;i++) {
        arr1.setArr(i, i);
        arr2.setArr(i, 2*i);
    }
    arr1.setArr(10, 10);
    arr1.print();
    arr2.print();
    arr1 = arr2;
    arr1.print();
}

```

```
}
```

```
or-ctqwq11k.4se' --pid=Microsoft-MIEngine-Pid-c2b1cgz1.3oy
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
addr: 0xdb1a50 contents:  0 1 2 3 4 5 6 7 8 9 10
addr: 0xdb3f00 contents:  0 2 4 6 8 10 12 14 16 18
size error
addr: 0xdb1a50 contents:  0 2 4 6 8 10 12 14 16 18 10
PS D:\code repository\code> █
```

(2)



分析：结果没有影响，但是调用堆栈对报错。

(3)

```
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
addr: 0xe81a50 contents:  0 1 2 3 4 5 6 7 8 9
addr: 0xe83f00 contents:  0 2 4 6 8 10 12 14 16 18
addr: 0xe83f00 contents:  0 2 4 6 8 10 12 14 16 18
PS D:\code repository\code> █
```

```
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
addr: 0xd81a50 contents:  0 1 2 3 4 5 6 7 8 9
addr: 0xd83f00 contents:  0 2 4 6 8 10 12 14 16 18
addr: 0xd81a50 contents:  0 2 4 6 8 10 12 14 16 18
PS D:\code repository\code> █
```

分析：重载之前的拷贝是直接吧 arr1 的地址改成 arr2 的，也就是 arr1 和 arr2 指的是同一块内存区域；重载后的拷贝是把 arr2 里的值拷贝给 arr1，arr1 的内存区域不变。