第七次上机实验

**1、程序代码**

**实验一**

#include<iostream>

using namespace std;

class Complex {

double real;

double imag;

public:

Complex(double r = 0.0, double i = 0.0);

void print();

Complex operator\*(Complex c);

};

Complex::Complex(double r, double i)

{

real = r; imag = i;

}

**①、成员运算符重载函数（需要在类内先定义）**

Complex Complex::operator\*(Complex c)

{

Complex temp;

temp.real = real \* c.real - imag \* c.imag;

temp.imag = real \* c.imag + imag \* c.real;

return temp;

}

void Complex::print()

{

cout << "total real=" << real << " " << " total imag=" << imag << endl;

}

int main()

{

Complex com1(2.3, 4.6), com2(3.6, 2.8), total1;

total1 = com1 \* com2;

total1.print();

return 0;

}

**②、友元运算符重载函数（需要在类内先定义）**

Complex operator\*(Complex co1, Complex co2)

{

Complex temp;

temp.real = co1.real \* co2.real - co1.imag \* co2.imag;

temp.imag = co1.real \* co2.imag + co1.imag \* co2.real;

return temp;

}

**③、类外定义运算符重载函数（无需在类内先定义，但成员变量需要是公有变量）**

Complex operator\*(Complex co1, Complex co2)

{

Complex temp;

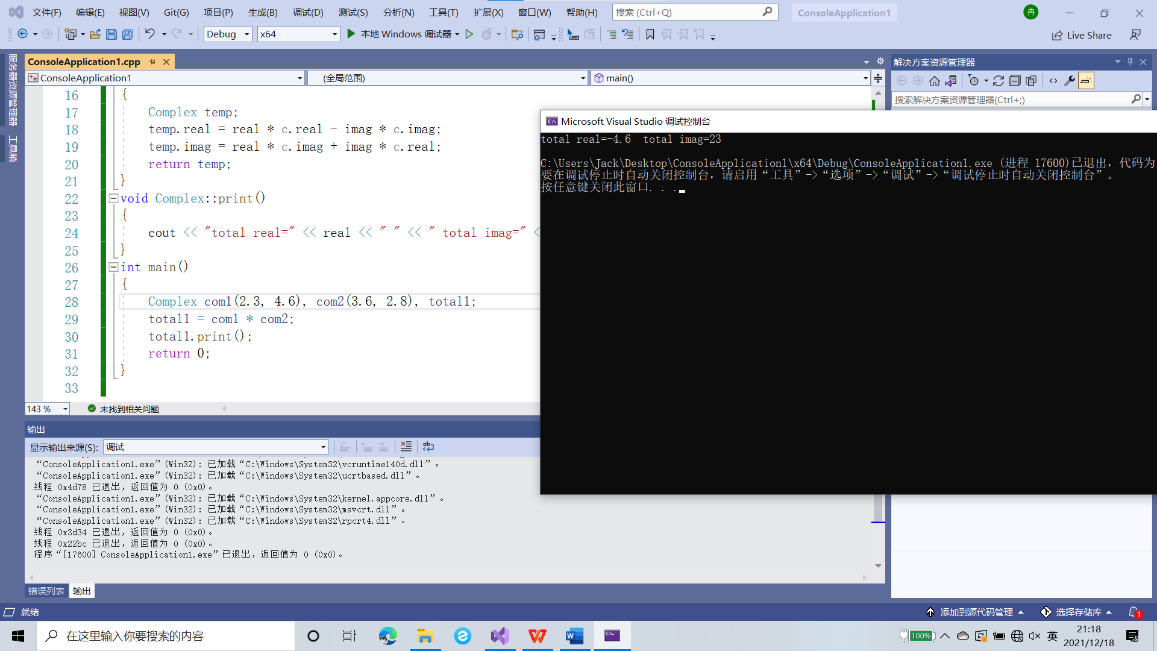
temp.real = co1.real \* co2.real - co1.imag \* co2.imag;

temp.imag = co1.imag \* co2.real + co1.real \* co2.imag;

return temp;

}

**2、程序结果**

**实验一**

**①②③三种方式的结果都一致**

**3、感想心得**

本次上机实验学习了C++最后一个基本特征——多态，多态其实就是用一个名字定义不同的函数，这些函数执行不同但又类似的操作，这样就可以用同一个函数名调用不同内容的函数。

本次上机实验中所要求的内容不难，只要学过一点复数知识，写这个程序不是很难，所以我讲三种形式的运算符重载都写了一遍，从而让我很清楚这三种运算符重载的相同点和不同点在哪里。

相同点在于三种运算符重载所要写的函数体都基本差别不大，所要求的含义是一样的，但在一些要求上有所不同，在类外定义运算符需要保证所调用的成员都是公有的，否则无法调用；友元和成员运算符重载则不需要，但是它们两个的不同点在于友元运算符重载还是需要有两个参数来定义，而成员运算符重载只需要一个。