**一、程序代码**

#include<iostream>

using namespace std;

class Complex {

double real;

double imag;

public:

Complex(double r = 0, double i = 0)

{

real = r; imag = i;

}

void print();

friend Complex operator\*(Complex co1, Complex co2);

};

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;

}

void Complex::print()

{

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

}

int main()

{

float a, b, c, d;

cout << "please input real and imag:" << endl;

cin >> a >> b;

cout << "please input real and imag:" << endl;

cin >> c >> d;

Complex com1(a, b), com2(c, d), total1;

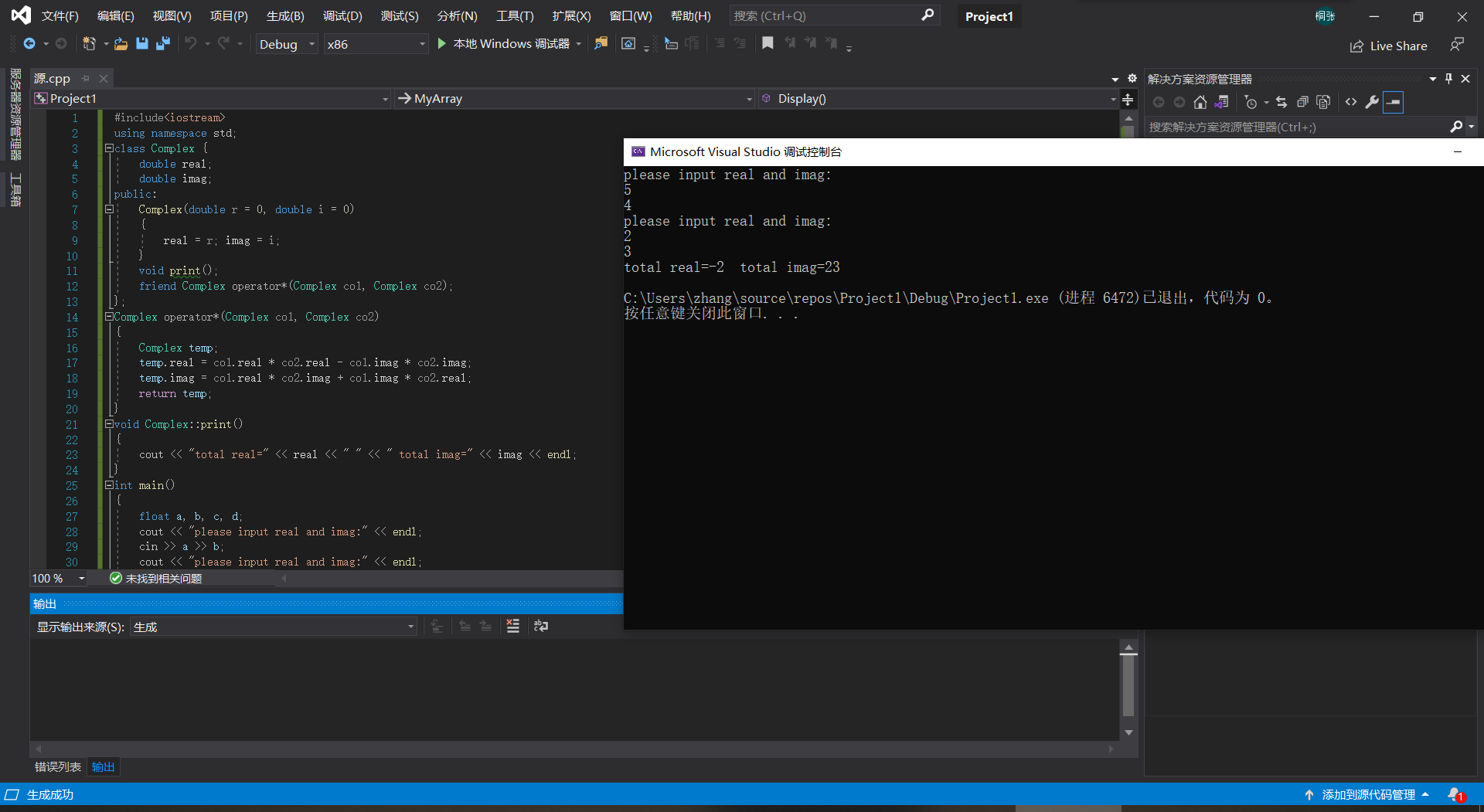
total1 = com1 \* com2;

total1.print();

return 0;

}

**二、实验结果**



**三、实验心得**

1. 友元函数具有自己的的访问规则：友元函数属于当前类的成员函数，是独立于当前类的另一个外部函数，但它可以访问该类所有的成员，包括私有成员、保护成员和工有成员。在使用友元函数时（类外定义），要注意不必使用作用域运算符：：。
2. 运算符重载的应用过程中，要注意调用方式有两种，一种是operate\*（co1，co2）；而第二种是直接使用被重载的运算符进行运算。
3. 通过利用友元函数，我们可以在类外进行了运算符的重载，直接使用了重载后的\*；同时将数据的输入与输出放在主函数中，虽然降低了类的重用性，没能很好的体现封装性，使程序变得更加的简洁易读。

Copyright ©2021-2099 TongmingZhang. All rights reserved