第七次上机实验报告

电自2003 宿颢宝

实验一：

代码：

#include<iostream>

//#include<cstring>

#include<cmath>

using namespace std;

class complex {

private:

double real;

double image;

public:

complex(double real1=0 ,double image1=0):real(real1),image(image1){}

friend complex operator\*(complex& x, complex& y);

void show();

};

void complex::show()

{

cout << "The multiplication of the complexes is : " << endl;

cout << real;

if (image > 0)

cout << "+";

if(image!=0)

cout << image << "i" << endl;

}

complex operator\*(complex& x, complex& y)

{

complex temp;

temp.real = x.real \* y.real;

temp.image = x.image \* y.image;

return temp;

}

int main()

{

double x1, x2, y1, y2;

cout << "Please input the first complex: " << endl;

cin >> x1 >> y1;

cout << "Pease input the second complex: " << endl;

cin >> x2 >> y2;

complex a(x1 ,y1);

complex b(x2, y2);

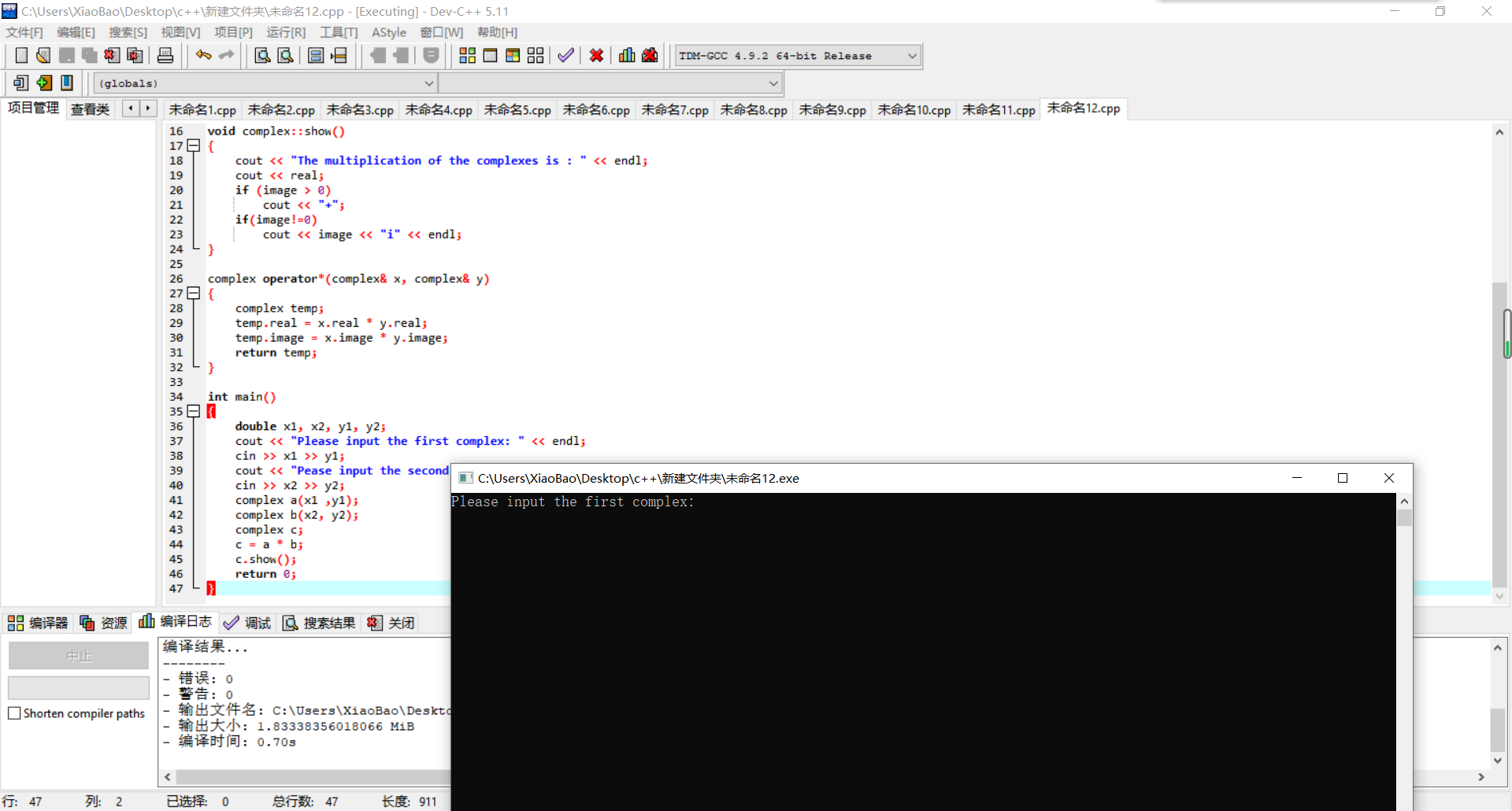
complex c;

c = a \* b;

c.show();

return 0;

}

图片：  


实验二：

代码：

#include<iostream>

//#include<cstring>

#include<cmath>

using namespace std;

class complex {

private:

double real;

double image;

public:

complex(double real1=0 ,double image1=0):real(real1),image(image1){}

friend complex operator\*(complex& x, complex& y);

void show();

};

void complex::show()

{

cout << "The multiplication of the complexes is : " << endl;

cout << real;

if (image > 0)

cout << "+";

if(image!=0)

cout << image << "i" << endl;

}

complex operator\*(complex& x, complex& y)

{

complex temp;

temp.real = x.real \* y.real-x.image\*y.image;

temp.image = x.real \* y.image+x.image\*y.real;

return temp;

}

int main()

{

double x1, x2, y1, y2;

cout << "Please input the first complex: " << endl;

cin >> x1 >> y1;

cout << "Pease input the second complex: " << endl;

cin >> x2 >> y2;

complex a(x1 ,y1);

complex b(x2, y2);

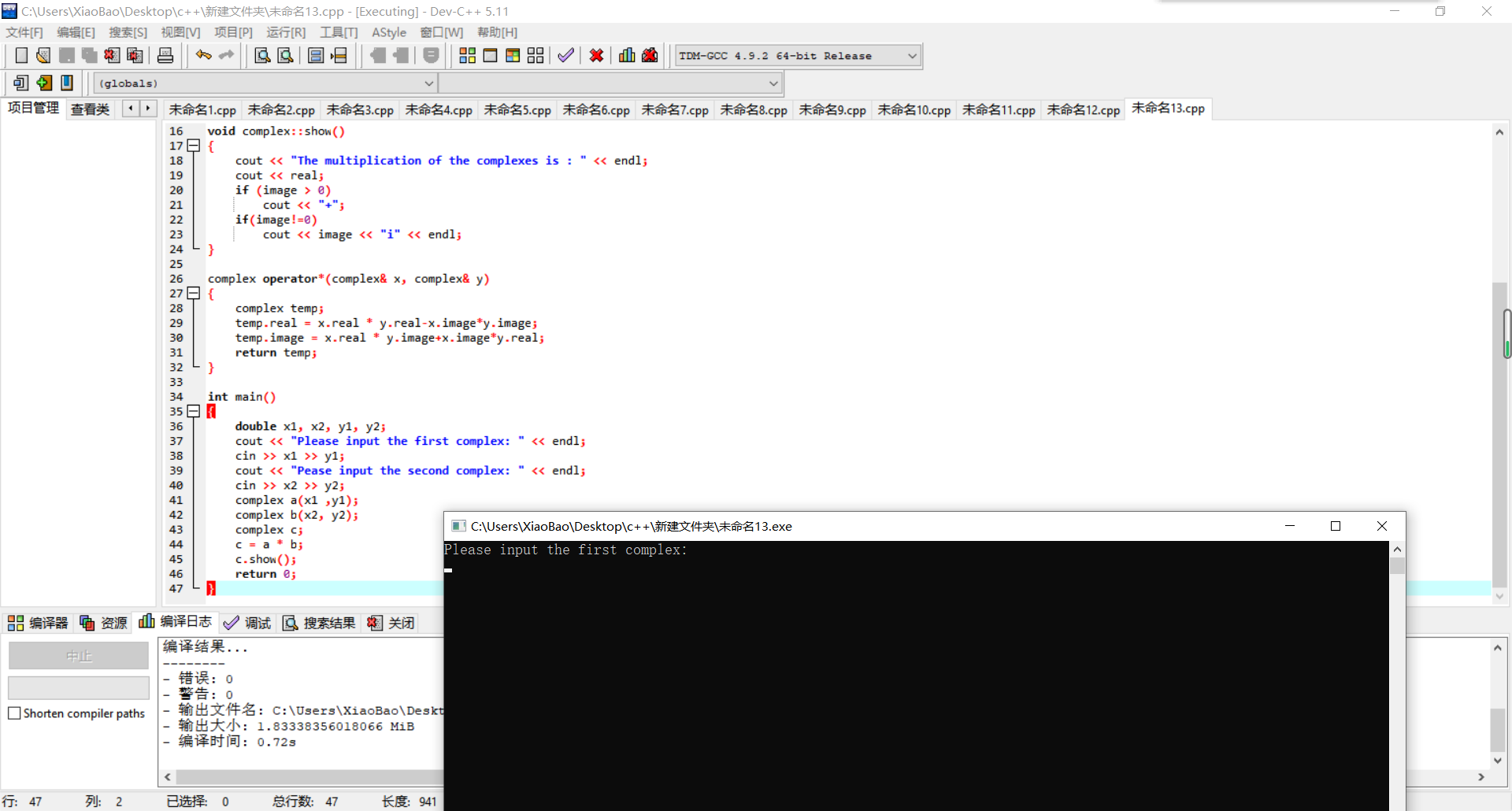
complex c;

c = a \* b;

c.show();

return 0;

}



总结：

本次上机实验主要考察运算符重载函数的运用，运算符重载函数本质上是用户为了实现特殊的函数运算而自定义的一种函数，在使用时可将其看作一个运算符号。其函数名是由关键字 operator 和其后要重载的运算符符号构成的。与其他函数一样，重载运算符有一个返回类型和一个参数列表，若定义了一个类 A 那么一般定义格式为：

A operator@（基础运算符号）(A &x, A &b)

{

（函数体）

}

若要实现对类中私有成员的运算操作，则将运算符重载函数定义为友元函数即可。该函数实际上也体现了面向对象程序设计的多态性，即不同对象收到相同的消息时，能够产生不同的行为。