电子科技大学信息与软件工程学院

**实验报告**

点名册序号 304-3 学 号 2015220103022

姓 名 张健顺

（实验） 课程名称 面向对象程序设计(C++)

理论教师 陈安龙

实验教师 陈安龙

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：张健顺 学号：2015220103022 指导教师：陈安龙**

**实验地点：信软楼西304 实验时间：16.11.01**

1. **实验名称：C++的函数及运算符重载程序设计**
2. **实验学时：4**
3. **实验目的：**

通过实验练习，让学生理解类的成员函数重载的含义，理解构造函数的重载的意义，并掌握如何对构造函数重载；掌握函数重载的两种方式，分别通过成员函数和友元函数实现重载，理解两种实现方式的区别，最后，理解和掌握几种常用运算符重载的实现方法。

1. **实验内容：**

设计复数类MyComplex，复数的实部和虚部均为double类型，并且放在private段；重载复数类的构造函数，及复制构造函数；重载复数运算+，-，\*，/；重载复数流运算符<<,>>。同时实现返回复数实部、虚部、模的成员函数。编程设计实现下列函数：

myComplex();

myComplex(int a);

myComplex(int a,int b)

myComplex( myComplex& v)

double getReal(); //返回复数的实部

double getImaginary (); //返回复数的虚部

double getModulus (); //返回复数的模

myComplex& operator=( myComplex& rhs); //类对象的赋值

myComplex& operator+=( myComplex& rhs);

myComplex& operator-=( myComplex& rhs);

myComplex& operator\*=( myComplex& rhs);

myComplex& operator/=( myComplex& rhs);

friend myComplex operator+(myComplex m,myComplex n)

friend myComplex operator-(myComplex m,myComplex n)

friend myComplex operator\*(myComplex m,myComplex n)

friend myComplex operator/(myComplex m,myComplex n)

friend ostream& operator<<(ostream& os,myComplex c);

friend istream& operator>>(istream& is,myComplex& c);

要求：

（1）理解类的成员函数重载的含义；

（2）掌握类构造函数的重载定义与使用;

（3）掌握运算符的重载定义与使用；

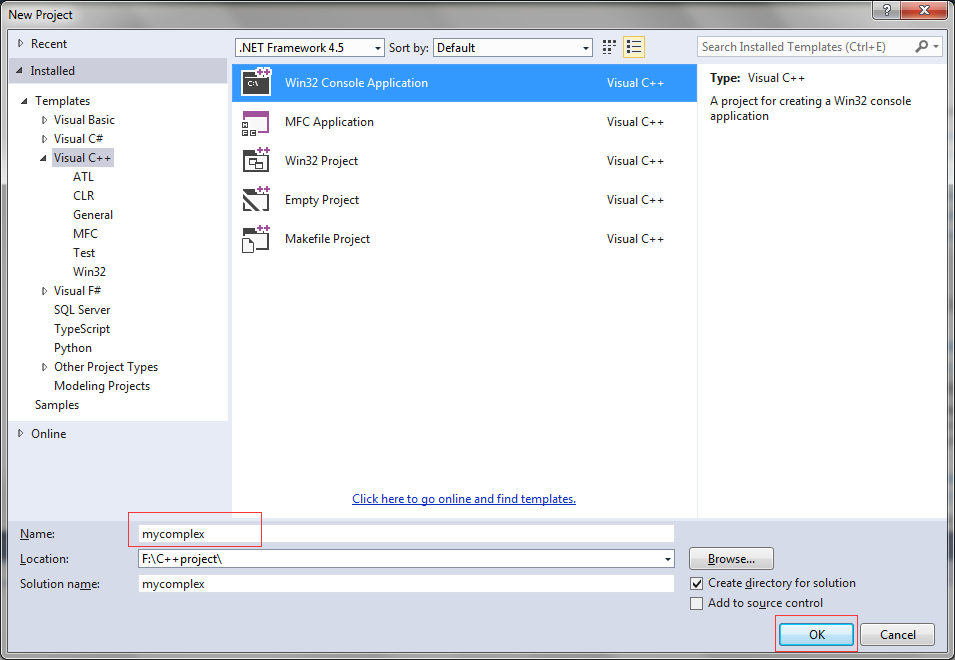
（4）掌握使用友元方法实现重载。

1. **实验器材（设备、元器件）：**

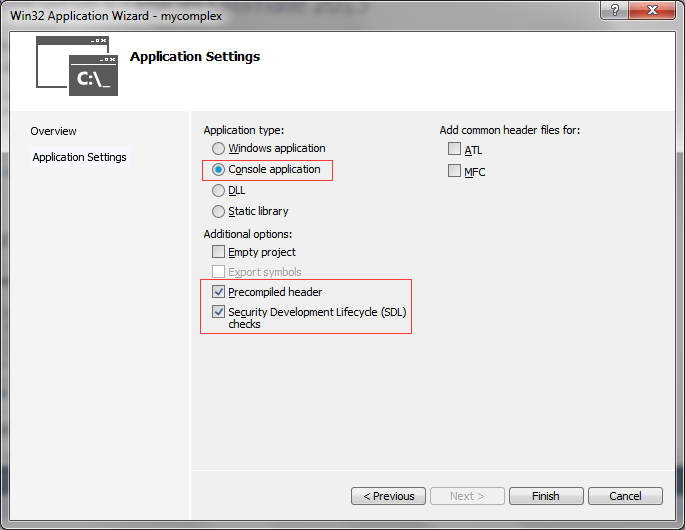
PC机一台，Windows 8.1，Visual Studio 2013

1. **实验步骤：**

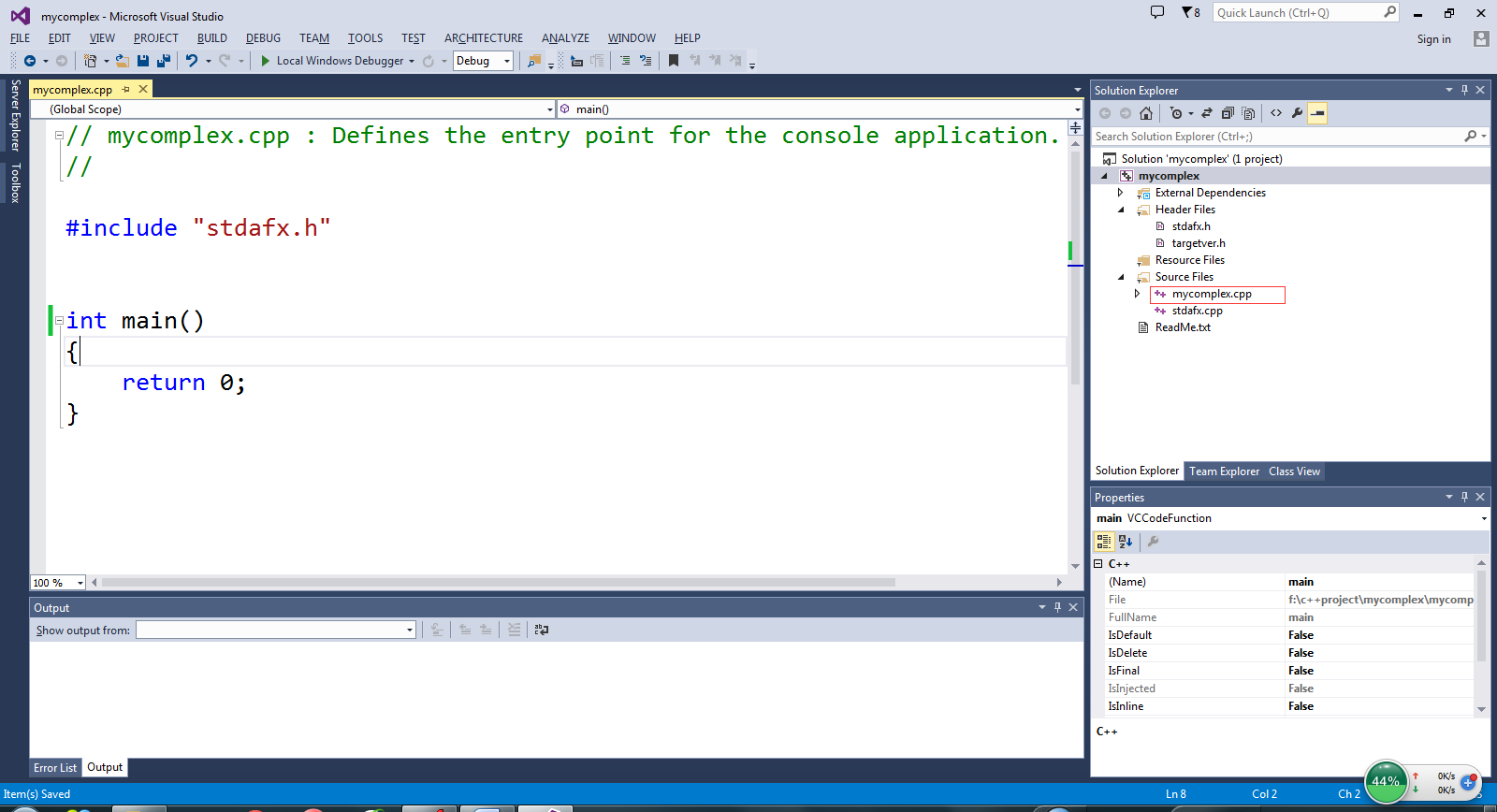
(1) 建立工程mycomplex



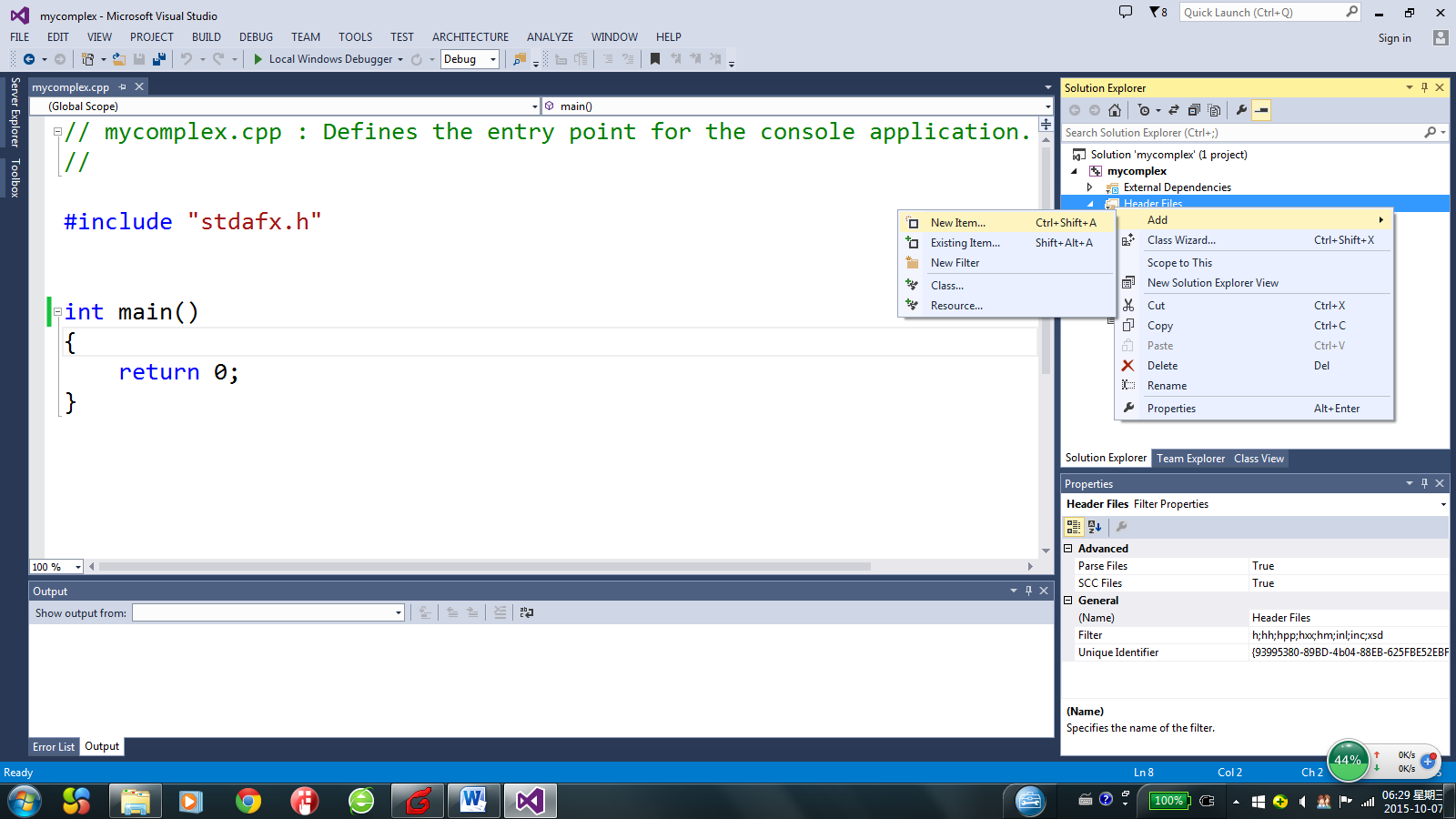
(2) 选择Console application，Precompiled header，SDL

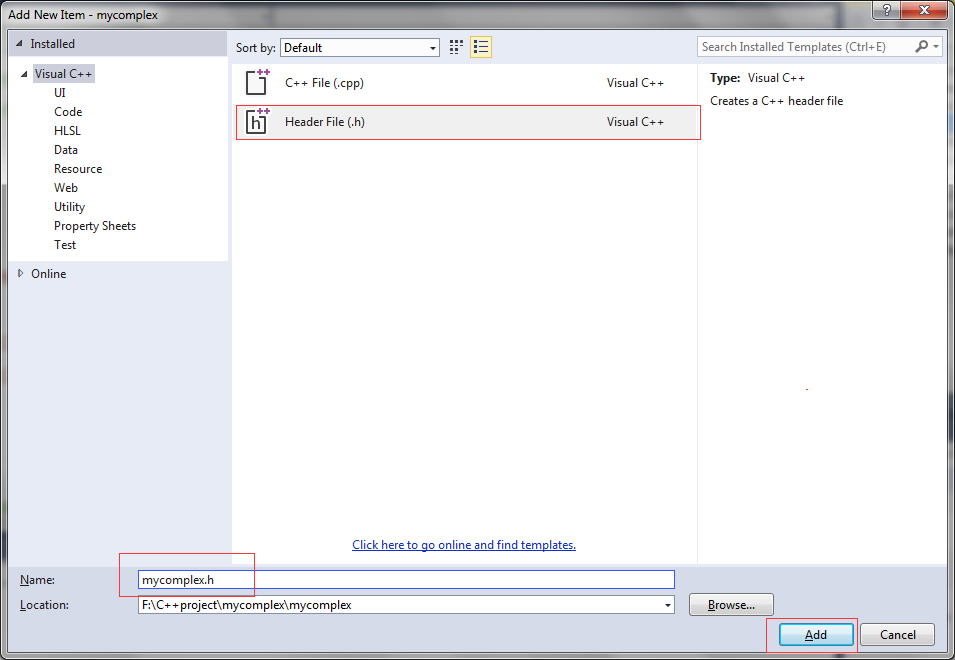


（3）生成如下所示mycomplex.cpp, 并将主程序入口改为 int main（）

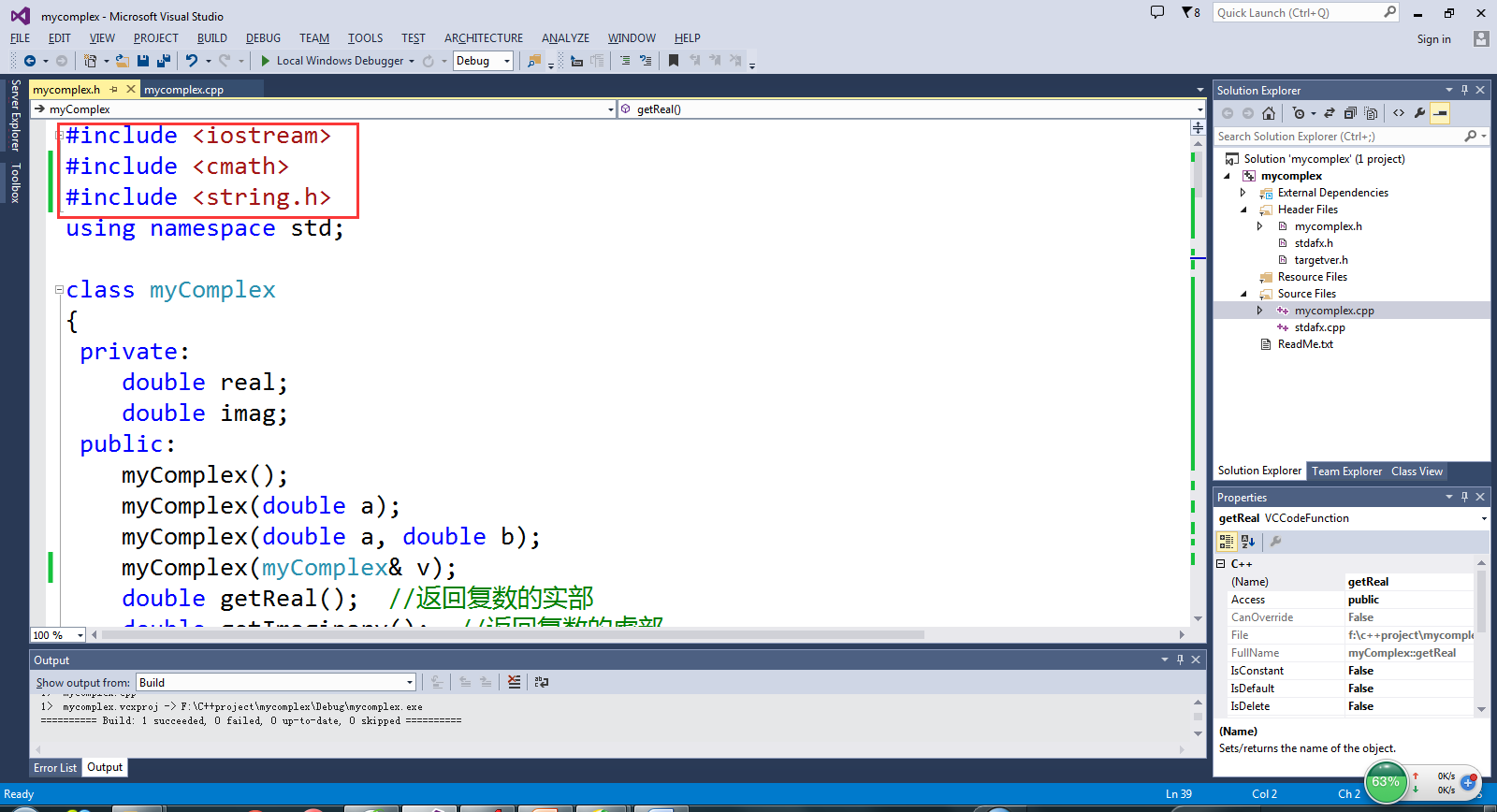


（4）参照下列图例，创建header文件mycomplex.h

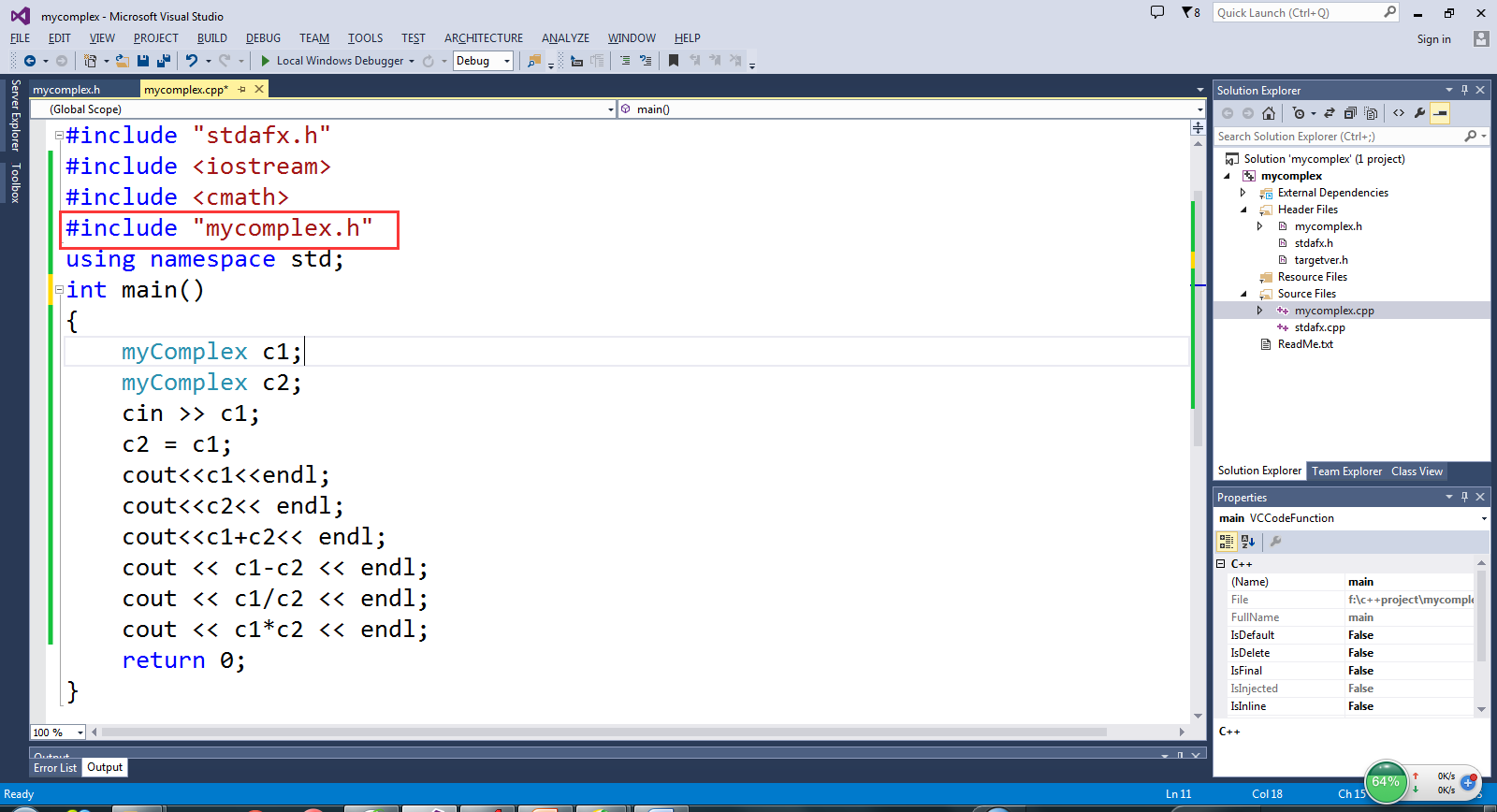




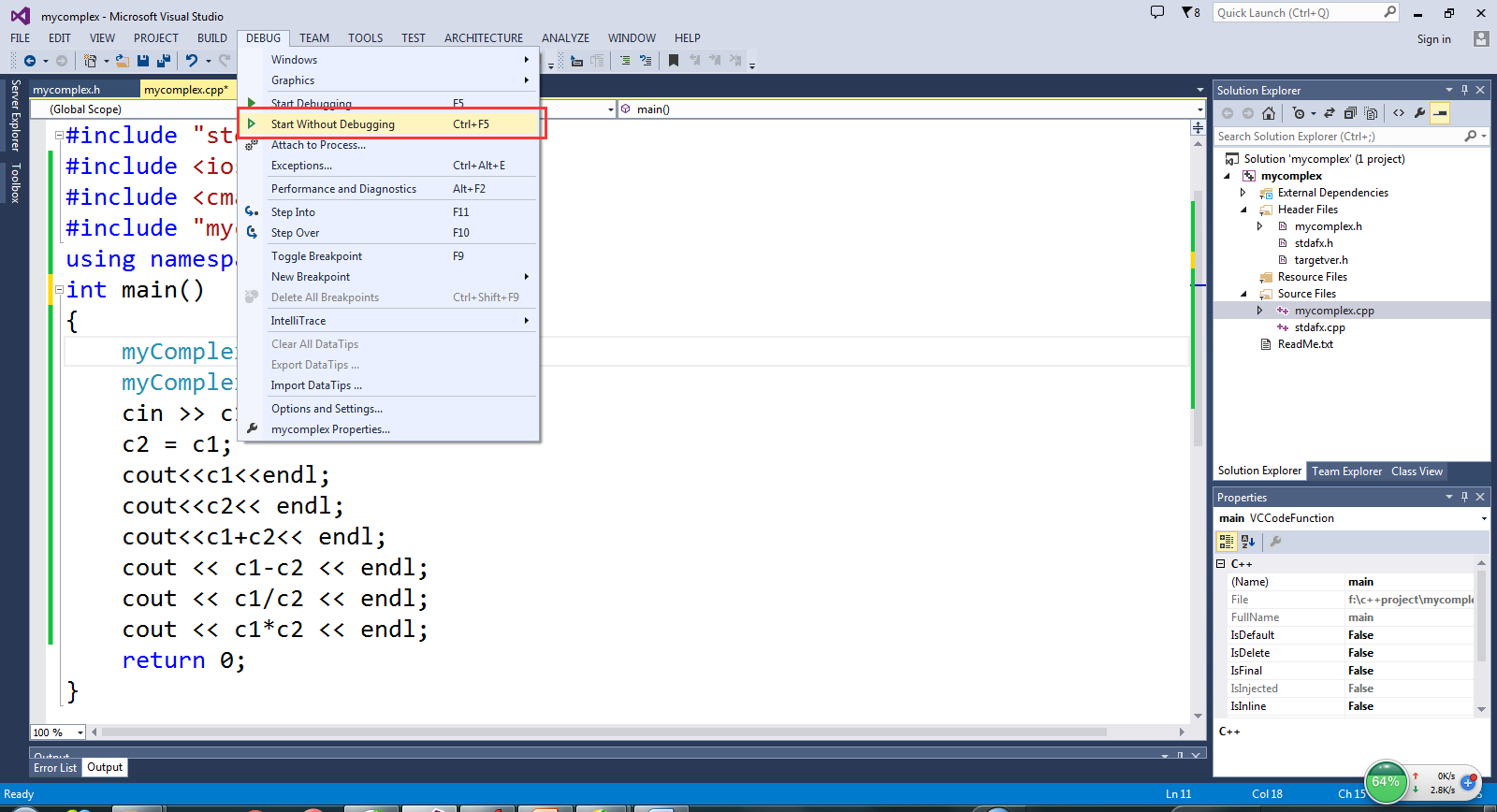
（5）在头文件中创建myComplex类，并实现类的构造函数、复制构造函数、成员函数、运算符重载的成员函数、友元重载的成员函数。



（6）在myComplex.cpp编辑主程序，验证重载后的运算符号。



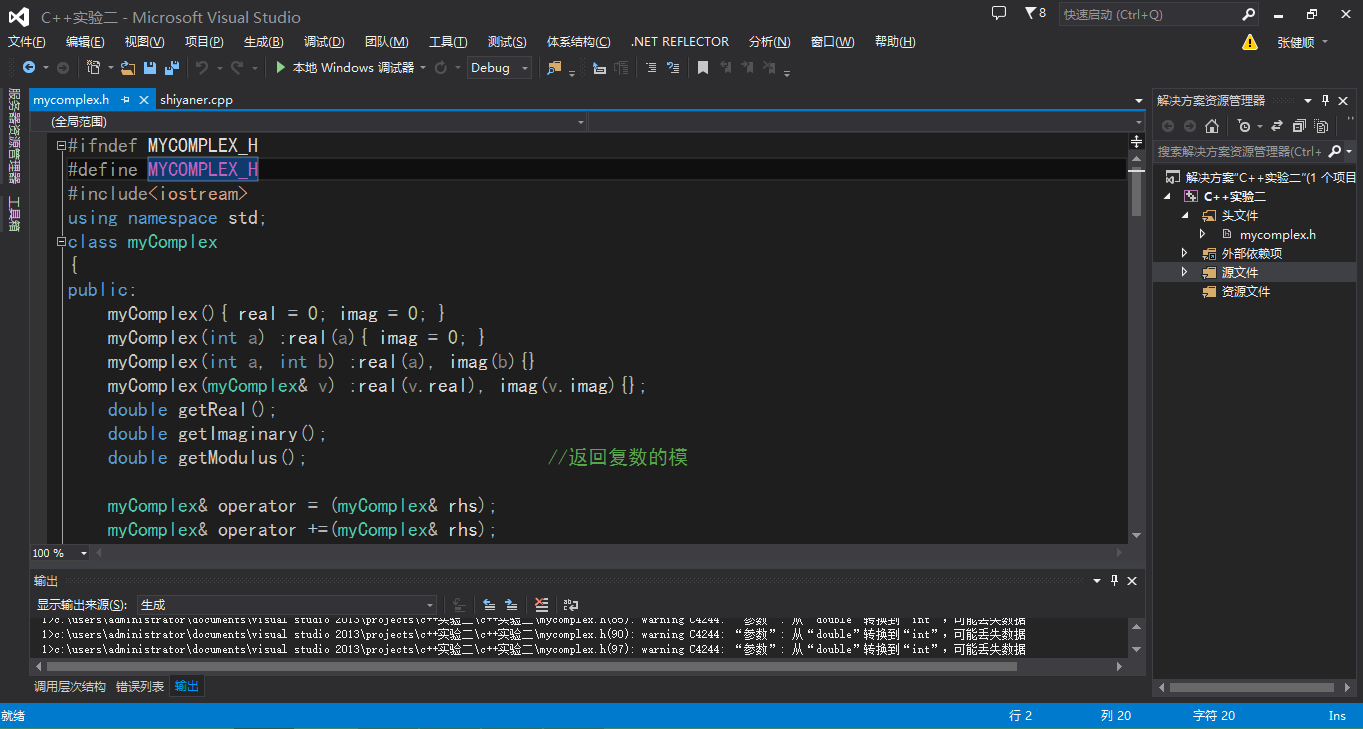
（7）调试运行程序



（8）运行程序，输入复数的实部和虚部后，显示结果

1. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**
2. 建立工程后编写头文件，在头文件中创建myComplex类，并实现类的构造函数、复制构造函数、成员函数、运算符重载的成员函数、友元重载的成员函数；

其中类的构造函数可以考虑多种情形，参数的有无以及参数的个数，同时考虑运算符重载的成员函数以及友元重载的成员函数的区别。



class myComplex{

public:

myComplex(){ real = 0; imag = 0; }

myComplex(int a) :real(a){ imag = 0; }

myComplex(int a, int b) :real(a), imag(b){}

myComplex(myComplex& v) :real(v.real), imag(v.imag){};

double getReal(); //返回实部

double getImaginary(); //返回虚部

double getModulus(); //返回复数的模

myComplex& operator = (myComplex& rhs);

myComplex& operator +=(myComplex& rhs);

myComplex& operator -=(myComplex& rhs);

myComplex& operator \*=(myComplex& rhs);

myComplex& operator /=(myComplex& rhs);

friend myComplex operator + (myComplex& rhs1, myComplex& rhs2);

friend myComplex operator - (myComplex& rhs1, myComplex& rhs2);

friend myComplex operator \* (myComplex& rhs1, myComplex& rhs2);

friend myComplex operator / (myComplex& rhs1, myComplex& rhs2);

friend ostream& operator <<(ostream& os, myComplex &rhs);

friend istream& operator >>(istream& is, myComplex &rhs);

private:

double real,imag;

};

在类外定义类中声明的运算符重载的成员函数（以“=”,“+=”和“\*=”例，其他同理，其中“=”只能用运算符重载）：

myComplex& myComplex::operator = (myComplex& rhs){

real = rhs.real; imag = rhs.imag; return \*this;

}

myComplex& myComplex::operator +=(myComplex& rhs)

{ /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*this为对象指针，\*this则是指对象本身\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*所以可以调用重载后的复数加法运算\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\* -= 、\*= 、/= 、同理 但必须要先声明对应的加减乘除\*\*\*\*/

\*this = \*this + rhs; return \*this;

}

myComplex& myComplex::operator \*=(myComplex& rhs){

\*this=\*this \* rhs; return \*this;

}

......

在类外定义类中声明的友元重载的成员函数（以“\*”，“/”和“>>”为例）：

myComplex operator \* (myComplex& rhs1, myComplex& rhs2){

return myComplex(rhs1.real \* rhs2.real - rhs1.imag \* rhs2.imag,

rhs1.real \* rhs2.imag + rhs1.imag \* rhs2.real);

}

myComplex operator / (myComplex& c1, myComplex& c2){

return myComplex((c1.real \* c2.real + c1.imag \* c2.imag)

/ (c2.real \* c2.real + c2.imag \* c2.imag),

(c1.imag \* c2.real - c1.real \* c2.imag)

/ (c2.real \* c2.real + c2.imag \* c2.imag));

}

istream& operator >>(istream& is, myComplex &rhs){

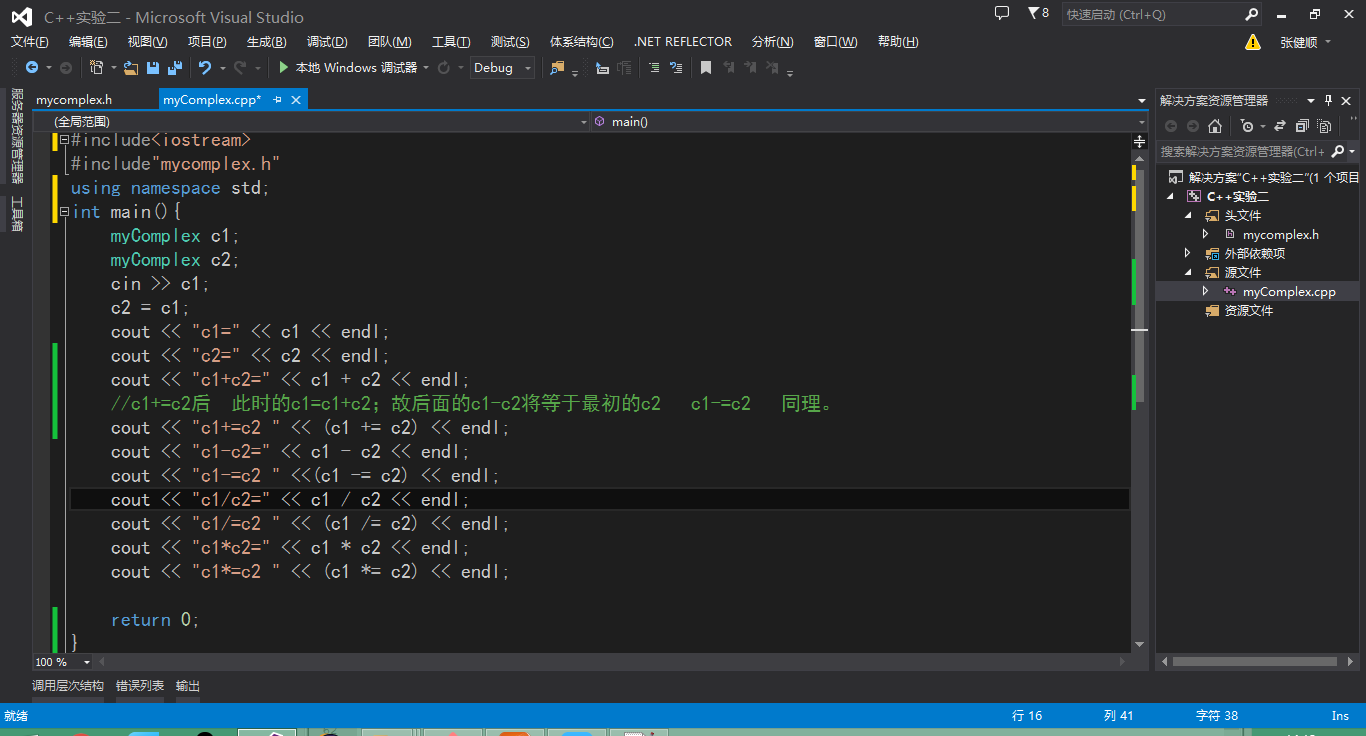
is >> rhs.real >> rhs.imag;

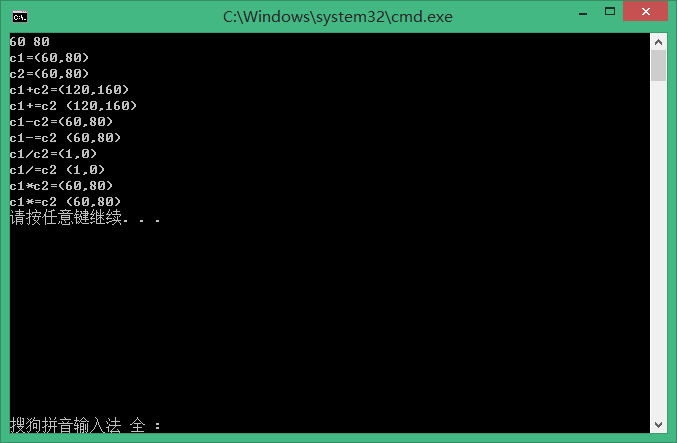
return is;

}

注意：“\*”和“/”的友元函数的重载返回值都是相应的构造函数初始化的临时对象，而输入输出的操作符重载后的返回值为相应的输入输出流。

（2）在myComplex.cpp编辑主程序，验证重载后的运算符号。调试运行程序,输入复数的实部和虚部后，显示结果.





c1的初始化是由用户自定义输入附属的实部和虚部部分，由于重载输入流的输入操作符，从而使得能够按照先实部后虚部的顺序来初始化c1，其中c2的初始化调用了复制构造函数，将c1复制给了c2。验证各类重载后的运算符当中，注意c1+=c2之后的运算里c1的值。

#include<iostream>

#include"mycomplex.h"

using namespace std;

int main(){

myComplex c1，c2;

cin >> c1;

c2 = c1;

cout << "c1=" << c1 << endl;

cout << "c2=" << c2 << endl;

cout << "c1+c2=" << c1 + c2 << endl;

//c1+=c2后此时的c1=c1+c2；故后面的c1-c2将等于最初的c2，c1-=c2同理。

cout << "c1+=c2 " << (c1 += c2) << endl;

cout << "c1-c2=" << c1 - c2 << endl;

cout << "c1-=c2 " <<(c1 -= c2) << endl;

cout << "c1/c2=" << c1 / c2 << endl;

cout << "c1/=c2 " << (c1 /= c2) << endl;

cout << "c1\*c2=" << c1 \* c2 << endl;

cout << "c1\*=c2 " << (c1 \*= c2) << endl;

return 0;

}

1. **总结及心得体会：**

（1）运算符重载的成员函数以及友元重载的成员函数的区别有：

A.运算符重载的成员函数参数个数为原操作数个数减一，而友元重载的成员函数的参数个数为原操作数个数；

B.成员函数有this指针，友元函数无this指针；

C.成员函数运算的结果可以作为左值使用，不是一个新产生的结果，而友元重载的运算结果是一个新产生的值，并且这个值不能作为左值使用。

1. 对于运算符“=，（），[]，->”只能作为成员函数；
2. 重载函数：通过函数的参数表唯一标识并且来区分函数的一种特殊的函数。
3. **对本实验过程及方法、手段的改进建议：**

“+=、-=、\*=、/=”运算符可以对之前“+、-、\*、/”的代码进行重用，从而达到简化代码的目的。重用的时候利用this指针，而\*this则是指对象本身。

**报告评分：**

**指导教师签字：**