① OOP **Modern College of Engineering**

Shivajinagar, Pune 5. SE A

Assignment no. 1

Name: Mahesh Jagtap Roll NO: 21027

Title: Arithmetic operations on complex numbers using operator overloading.

Problem statement: Implement a class Complex which represents the complex Number data type. Implement the following operations.

① constructor (including default constructor which creates complex no. 0+0i)
② Overloaded operator + to add two complex numbers. ③ overload operator * to multiply two complex numbers. ④ overload << and >> to print & read complex numbers.

Prerequisites: object oriented programming, constructors, operators in C++.

Objectives:
To learn the concept of constructor, default constructor, operator overloading using member functions & friend function.

Scanned with CamScanner

Theory :-

operator overloading

It is a specific case of polymorphism where diff. operators have diff. implementations depending on their arguments. In c++ the overloading principle applies not only to functions, but to operators too. That is, of operators can be extended to work not just with built in types but also classes. A programmer can provide his or her own operator to a class by overloading ~~but~~ the built-in operator to perform some specific computation when the operator is used on objects of that class.

An example of ~~q~~operator overloading:

```
complex a(1.2, 1.3)
Complex b(2.1, 3)
Complex c = a+b
```

Arithmetic operators :-

Arithmetic operators are used to do basic arithmetic operations like addition, subtraction, multiplication, division & modulus.

operator	use
+	addition
−	subtraction
*	multiplication
/	division
%	modulus

With c++ feature to overload operators, we can design classes able to perform operations using standard operators.
Here is a list of all operators that can be overloaded:

+ , − , * / = < > += −=
*= /= << >> <<= >>= ==
!= <= >= −− % & ^ ! |
~ &= ^= |= && %= [ ]

To overload an operator in order to use it with classes we declare operator functions, which are regular functions whose names are the operator keyword followed by the operator sign that we want to overload.
The format is,
type operator op-symbol (parameters) { /* ... */ }

The operator keyword declares a function specifying what operator-symbol means when applied to instances of a class.
This gives the operator more than one meaning, or "overloads" it.
syntax:

```
return-typeclass_name :: operator op(arg list)
{
    // function body
}
}
```

Process of overloading has 3 steps:
1. create a class that define a data type that is used in the overloading operator
2. Declare the operator function op() in public part of class. It may be either a member function on a friend function.
3. Define the operator function to implement the required operation.

e.g. overloading Binary operators:

```
C = sum (A, B) ;        // function notation
```

This functional notation can be replaced by a natural looking expression.

```
C = A+B  ;
```

Algorithm :

step1 : start

step2 : create a class Complex

step3 : Define a default Constructor

step4 : Declare the operator function which are going to be overloaded & display.

step5 : Define overloaded functions such as +, -, /, *, & the display function for addition.

$(a+bi) + (x+yi) = (a+x) + (b+y)i$

for multiplication

$(a+bi) * (x+yi) = ((a*x) - (b*y)) + ((a*y) + (x*b))i$

step6 : create Objects for Complex class in main() function.

step 7 : create a menu for addition, multiplication of Complex no. & display result.

Step 8 : Depending upon the choice from the user the arithmetic operators will invoke the overloaded operator automatically & returns the result.

step 9 : Display the result using display function.

6

Modern College of Engineering
Shivajinagar, Pune 5.

Input
complex nos with real & img values for two complex nos.
example:

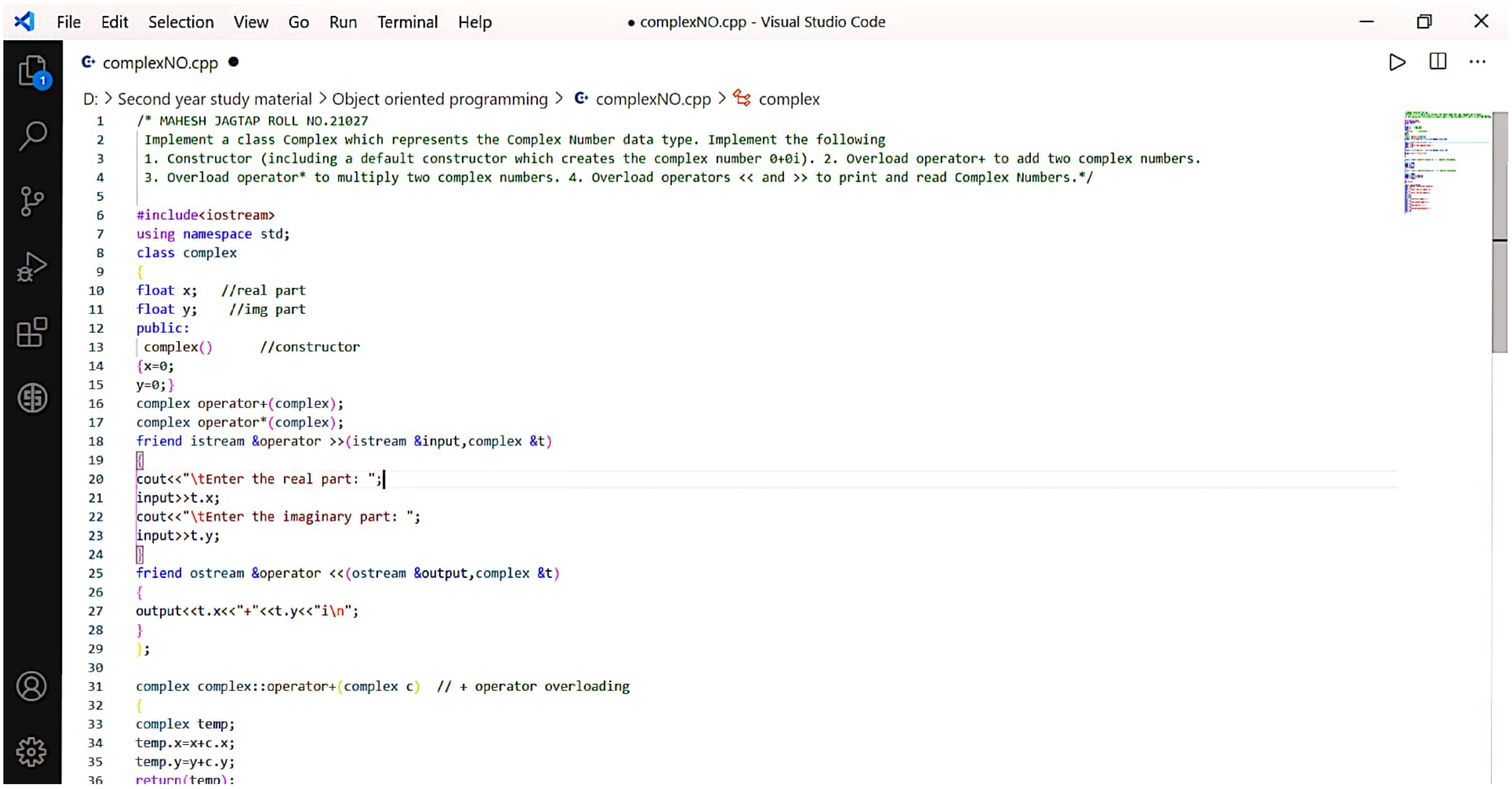
complex no. 1	real part = 5	img part = 4
complex no. 2	real part = 2	img part = 8

O/P:
Default constructor value = 0+0i

first no. is 5+4i
Second no. is 2+8i

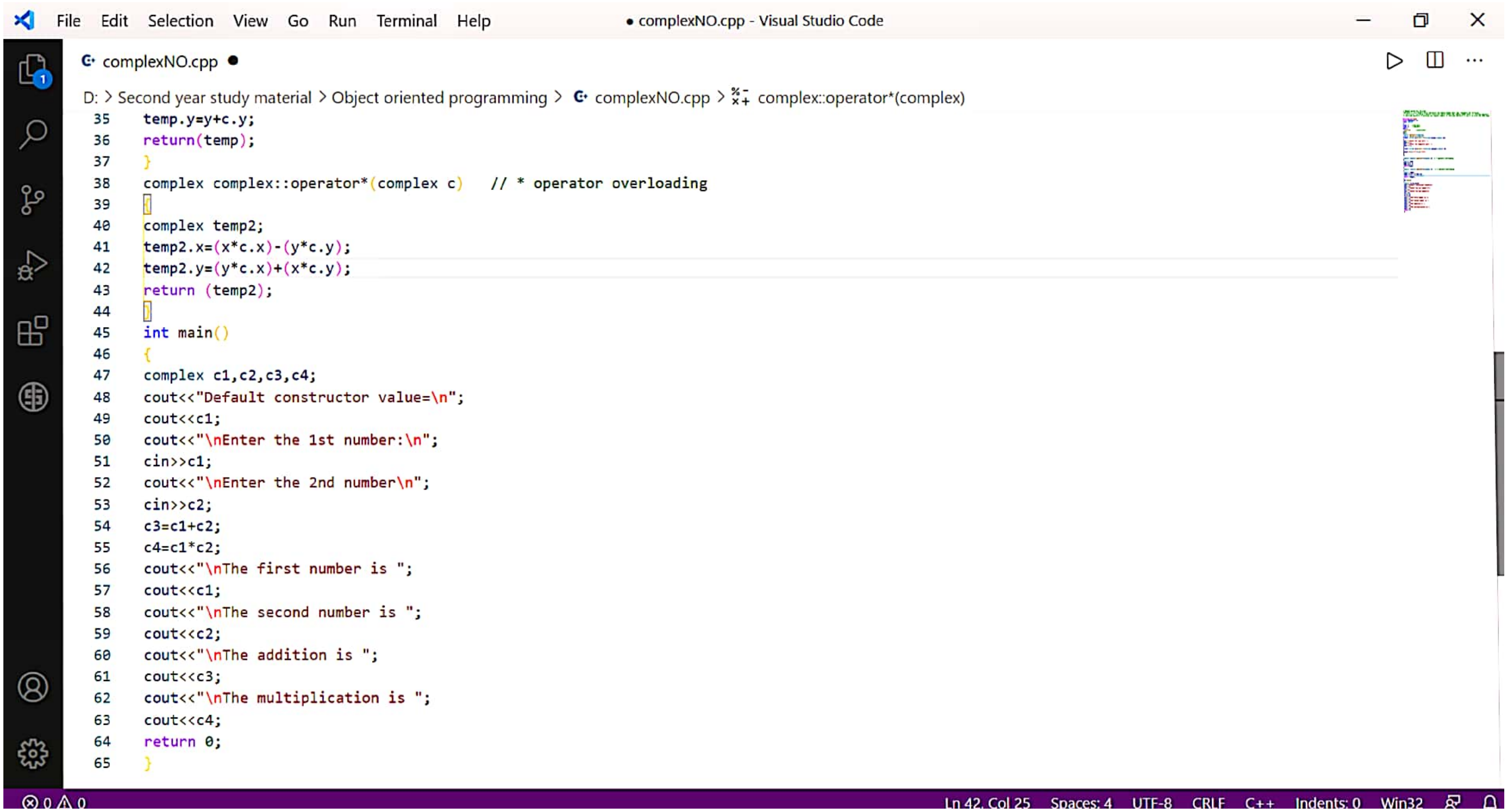
addition is 7+12i
multiplication is -22+48i

Conclusion:
In this assignment, we have studied the concept of operator overloading & using it implemented the program which adds & multiplies two complex numbers using (+) & (*) operator overloading.

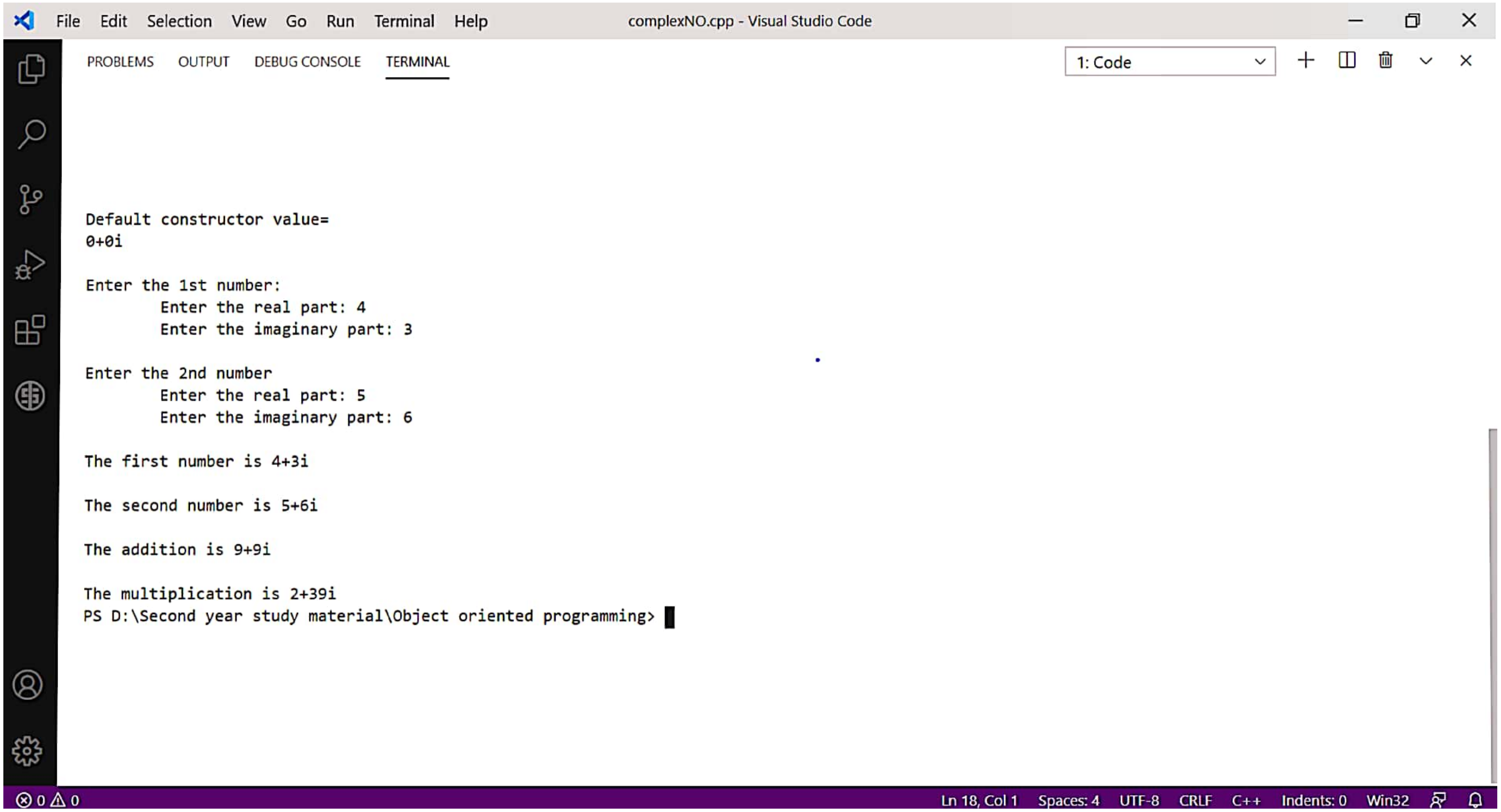

File Edit Selection View Go Run Terminal Help
• complexNO.cpp - Visual Studio Code
complexNO.cpp
D: > Second year study material > Object oriented programming > complexNO.cpp > complex

```cpp
/* MAHESH JAGTAP ROLL NO.21027
 Implement a class Complex which represents the Complex Number data type. Implement the following
 1. Constructor (including a default constructor which creates the complex number 0+0i). 2. Overload operator+ to add two complex numbers.
 3. Overload operator* to multiply two complex numbers. 4. Overload operators << and >> to print and read Complex Numbers.*/

#include<iostream>
using namespace std;
class complex
{
float x;   //real part
float y;    //img part
public:
 complex()      //constructor
{x=0;
y=0;}
complex operator+(complex);
complex operator*(complex);
friend istream &operator >>(istream &input,complex &t)
{
cout<<"\tEnter the real part: ";
input>>t.x;
cout<<"\tEnter the imaginary part: ";
input>>t.y;
}
friend ostream &operator <<(ostream &output,complex &t)
{
output<<t.x<<"+"<<t.y<<"i\n";
}
};

complex complex::operator+(complex c)  // + operator overloading
{
complex temp;
temp.x=x+c.x;
temp.y=y+c.y;
return(temp);
```

Scanned with CamScanner


File Edit Selection View Go Run Terminal Help
• complexNO.cpp - Visual Studio Code
complexNO.cpp
D: > Second year study material > Object oriented programming > complexNO.cpp > complex::operator*(complex)
Ln 42, Col 25 Spaces: 4 UTF-8 CRLF C++ Indents: 0 Win32

```
temp.y=y+c.y;
return(temp);
}
complex complex::operator*(complex c)   // * operator overloading
{
complex temp2;
temp2.x=(x*c.x)-(y*c.y);
temp2.y=(y*c.x)+(x*c.y);
return (temp2);
}
int main()
{
complex c1,c2,c3,c4;
cout<<"Default constructor value=\n";
cout<<c1;
cout<<"\nEnter the 1st number:\n";
cin>>c1;
cout<<"\nEnter the 2nd number\n";
cin>>c2;
c3=c1+c2;
c4=c1*c2;
cout<<"\nThe first number is ";
cout<<c1;
cout<<"\nThe second number is ";
cout<<c2;
cout<<"\nThe addition is ";
cout<<c3;
cout<<"\nThe multiplication is ";
cout<<c4;
return 0;
}
```



```
Default constructor value=
0+0i

Enter the 1st number:
        Enter the real part: 4
        Enter the imaginary part: 3

Enter the 2nd number
        Enter the real part: 5
        Enter the imaginary part: 6

The first number is 4+3i

The second number is 5+6i

The addition is 9+9i

The multiplication is 2+39i
PS D:\Second year study material\Object oriented programming>
```