**Name:** Saad Ahmad

**Enrolment:** 01-134222-130

**Lab 5**

**Object As Arguments**

**Objectives:**

To explain passing of class objects and accessing information.

**Objects as Function argument:**

The objects of a class can be passed as arguments to member functions as well as nonmember functions either by value or by reference. When an object is passed by value, a copy of the actual object is created inside the function. This copy is destroyed when the function terminates. Moreover, any changes made to the copy of the object inside the function are not reflected in the actual object. On the other hand, in pass by reference, only a reference to that object (not the entire object) is passed to the function. Thus, the changes made to the object within the function are also reflected in the actual object.

Whenever an object of a class is passed to a member function of the same class, its data members can be accessed inside the function using the object name and the dot operator. However, the data members of the calling object can be directly accessed inside the function without using the object name and the dot operator.

To understand how objects are passed and accessed within a member function, consider this example.

## **Example 5.3**

|  |
| --- |
| #include<iostream>  using namespace std;  **class** weight {  **int** kilogram;  **int** gram;  **public**:  **void** getdata ();  **void** putdata ();  **void** sum\_weight (weight,weight) ;  } ;  **void** weight :: getdata() {      cout<<"/nKilograms:";      cin>>kilogram;      cout<<"Grams:";      cin>>gram;  }  **void** weight :: putdata () {    cout<<kilogram<<" Kgs. and"<<gram<<" gros.\n";  }  **void** weight :: sum\_weight(weight w1,weight w2) {    gram = w1.gram + w2.gram;    kilogram=gram/1000;    gram=gram%1000;    kilogram+=w1.kilogram+w2.kilogram;  }  **int** main () {   weight w1,w2 ,w3;     cout<<"Enter weight in kilograms and grams\n";     cout<<"\n Enter weight #1" ;     w1.getdata();     cout<<" \n Enter weight #2" ;     w2.getdata();     w3.sum\_weight(wl,w2);     cout<<"/n Weight #1 = ";     w1.putdata();     cout<<"Weight #2 = ";     w2.putdata();     cout<<"Total Weight = ";     w3.putdata();  **return** 0;  } |

**Lab Journal**

### Exercise 5.1

Create a class called **Complex** for performing arithmetic with complex numbers. Complex numbers have the form **real + *i* imag**, where *i* = √-1.

Use floating point variables to represent the private data of the class. Provide a constructor to initialize a complex number when it is declared. The constructor should contain the default values in case no initializing values are provided. Provide public member functions to perform the following operations:

* **Addition** of Complex numbers by passing other Complex number as an argument
* **Subtraction** of Complex numbers by passing other Complex number as an argument
* **Multiplication** of Complex numbers by passing other Complex number as an argument
* **Printing** of a Complex number in a+b*i* format. Write a driver program to test your class.

Follow the following formulae:

r1 and r2 denotes real number 1 and 2.

i1 and i2 denotes imaginary number 1 and 2.

Addition: (r1 + i1) + (r2 + i2) = (r1 + r2) + (i1 + i2)

Subtraction: (r1 + i1) – (r2 + i2) = (r1 – r2) + (i1 – i2)

Multiplication: (r1 + i1). (r2 + i2) = (r1r2 – i1i2) + (r1i2 + r2i1)

**Your output screen should be like this:**

Enter real and imaginary part of first complex number

5

6

Enter real and imaginary part of second complex number

1

3

Sum of two complex numbers is

6+9i

**Code:**

#include <iostream>

using namespace std;

class Complex {

float real;

float img;

public:

Complex(float a = 0, float b = 0) {

real = a;

img = b;

}

Complex addition(Complex a1, Complex a2) {

Complex a3;

a3.real = a1.real + a2.real;

a3.img = a1.img + a2.img;

return a3;

}

Complex subtraction(Complex a1, Complex a2) {

Complex a3;

a3.real = a1.real - a2.real;

a3.img = a1.img - a2.img;

return a3;

}

Complex multiplication(Complex a1, Complex a2) {

int a, b, c;

Complex a3;

a = a1.real \* a2.real;

b = a1.img \* a2.img;

c = a - b;

a3.real = c;

a = a1.real \* a2.img;

b = a1.img \* a2.real;

c = a + b;

a3.img = c;

return a3;

}

void getInput() {

cout << "Enter the real part : " << endl;

cin >> real;

cout << "Enter the imaginary part : " << endl;

cin >> img;

}

void displyAdd() {

cout << "Sum of two complex numbers is " << endl;

cout << real << "+" << img << "i" << endl;

}

void displySub() {

cout << "Difference of two complex numbers is " << endl;

cout << real << "+" << img << "i" << endl;

}

void displyMul() {

cout << "Product of two complex numbers is " << endl;

cout << real << "+" << img << "i" << endl;

}

};

int main() {

Complex a1, a2, a3;

char choice;

a1.getInput();

a2.getInput();

do {

cout << "Enter + for addition, - for subtraction, \* for multiplication" << endl;

cin >> choice;

switch (choice) {

case '+': a3 = a3.addition(a1, a2);

a3.displyAdd();

break;

case '-': a3 = a3.subtraction(a1, a2);

a3.displySub();

break;

case '\*': a3 = a3.multiplication(a1, a2);

a3.displyMul();

break;

default:

cout << "Invalid option" << endl;

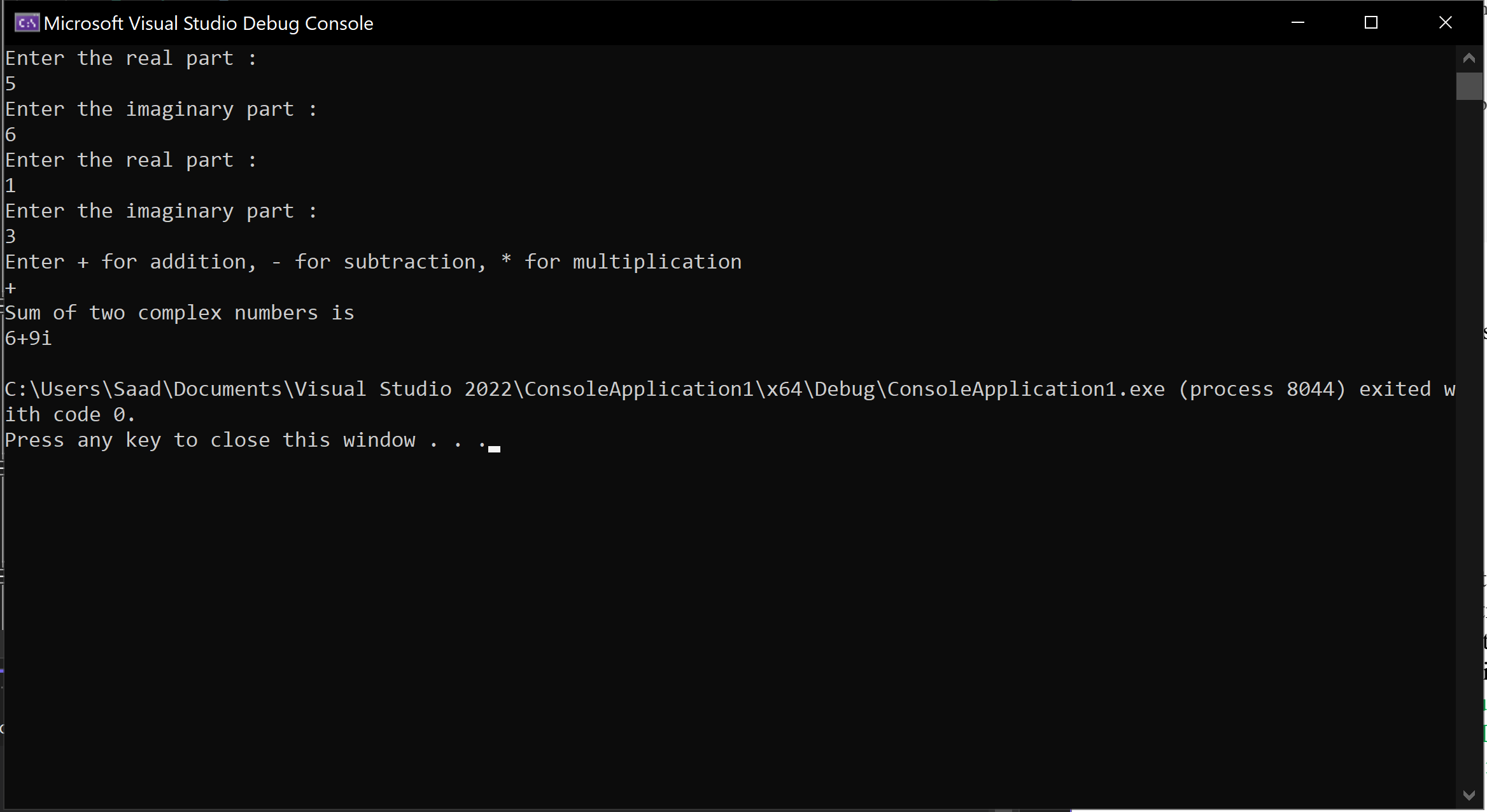
}

} while (choice != '+' && choice != '-' && choice != '\*');

return 0;

}

**Output:**



### Exercise 5.2

Consider the following class Fraction ADT and implement the following functions.

Class Fraction

{

private:

int numerator;

int denominator;

public:

Fraction addFraction(Fraction);

Fraction subFraction(Fraction);

Fraction mulFraction(Fraction);

Fraction divFraction(Fraction);

int getNum(); // returns Numerator

int getDenom(); // returns Denominator

void Set(int a, int b); // set numenator to a and denominator to b

};

write a driver (main function) that uses class Fraction to do the following calculations:

½ + ¼

½ - ¼

½ \* ½

½ / ½

write the output clearly labeled.

**Code:**

#include <iostream>

using namespace std;

class Fraction

{

private:

int numerator;

int denominator;

public:

Fraction addFraction(Fraction);

Fraction subFraction(Fraction);

Fraction mulFraction(Fraction);

Fraction divFraction(Fraction);

int getNum(); // returns Numerator

int getDenom(); // returns Denominator

void Set(int a, int b); // set numenator to a and denominator to b

};

void Fraction::Set(int a, int b) {

numerator = a;

denominator = b;

}

Fraction Fraction::addFraction(Fraction a) {

int b = a.denominator;

int num = numerator;

int denum = denominator;

a.numerator \*= denum;

a.denominator \*= denum;

num \*= b;

denum \*= b;

Fraction f1;

f1.numerator = a.numerator + num;

f1.denominator = denum;

return f1;

}

Fraction Fraction::subFraction(Fraction a) {

int b = a.denominator;

int num = numerator;

int denum = denominator;

a.numerator \*= denum;

a.denominator \*= denum;

num \*= b;

denum \*= b;

Fraction f1;

f1.numerator = a.numerator - num;

f1.denominator = denum;

return f1;

}

Fraction Fraction::mulFraction(Fraction a) {

Fraction f1;

f1.numerator = a.numerator \* numerator;

f1.denominator = denominator \* a.denominator;

return f1;

}

Fraction Fraction::divFraction(Fraction a) {

Fraction f1;

f1.numerator = a.numerator \* denominator;

f1.denominator = a.denominator\*numerator;

return f1;

}

int Fraction::getNum() {

return numerator;

}

int Fraction::getDenom() {

return denominator;

}

int main() {

const int size = 3;

Fraction f[size];

int a, b ,c ,d;

for (int i = 0; i < 2; i++) {

cout << "Enter numerator :" << endl;

cin >> a;

cout << "Enter denominator : " << endl;

cin >> b;

f[i].Set(a, b);

}

int e = f[0].getNum();

int x = f[0].getDenom();

int g = f[1].getNum();

int h = f[1].getDenom();

f[2] = f[1].addFraction(f[0]);

c = f[2].getNum();

d = f[2].getDenom();

cout << e << "/" << x << " + " << g << "/" << h << " = " << c << "/" << d << " = " << float(c) / float(d) << endl;

f[2] = f[1].subFraction(f[0]);

c = f[2].getNum();

d = f[2].getDenom();

cout << e << "/" << x << " - " << g << "/" << h << " = " << c << "/" << d << " = " << float(c) / float(d) << endl;

f[2] = f[1].mulFraction(f[0]);

c = f[2].getNum();

d = f[2].getDenom();

cout << e << "/" << x << " \* " << g << "/" << h << " = " << c << "/" << d << " = " << float(c) / float(d) << endl;

f[2] = f[1].divFraction(f[0]);

c = f[2].getNum();

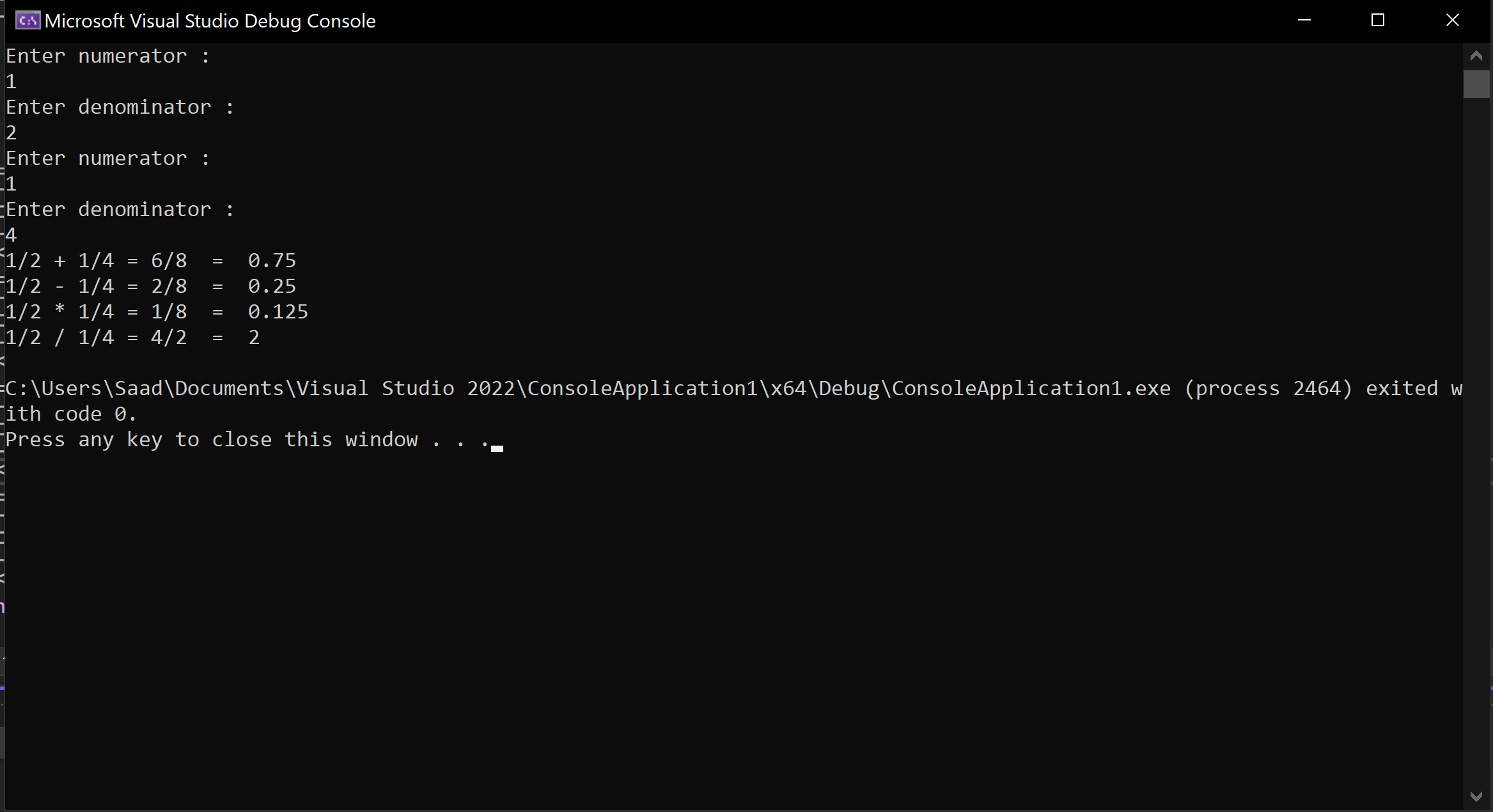
d = f[2].getDenom();

cout << e << "/" << x << " / " << g << "/" << h << " = " << c << "/" << d << " = " << float(c) / float(d) << endl;

return 0;

}

**Output:**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*