**LAB REPORT NO 2**



**Spring 2020**

**CSE102L Computer Programming Lab**

Submitted by:  **Muhammad Ali**

Registration No: **19PWCSE1801**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

**MAM. Sumayyea salahuddin**

(December 4, 2020)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Activity Number 2.41**

**C++ C0DE:-**

#include<iostream>

using namespace std;

class complex {

double real ,imag;

public:

complex();

complex(double real ,double imag);

void input();

void addcomplex(complex s1 ,complex s2);

void subcomplex(complex s1 ,complex s2);

void mulcomplex(complex s1 ,complex s2);

void show();

};

complex::complex():real(0),imag(0){}

complex::complex(double r,double i){

real=r;

imag=i;

}

void complex::input(){

cout<<"enter real part complex number :\n";

cin>>real;

cout<<"\nenter imaginary part complex number :\n";

cin>>imag;

}

void complex::addcomplex( complex c1,complex c2){

real=c1.real+c2.real;

imag=c1.imag+c2.imag;

}

void complex::subcomplex( complex c1,complex c2){

real=c1.real-c2.real;

imag=c1.imag-c2.imag;

}

void complex::mulcomplex( complex c1,complex c2){

real=c1.real\*c2.real-c1.imag\*c2.imag;

imag=c1.real\*c2.imag+c2.real\*c1.imag;

}

void complex::show(){

if(imag>0)

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

else

cout<<real<<imag<<"i"<<endl;

}

main(){

complex c1(0,0),c2(0,0),c;

cout;c1.input();

cout;c2.input();

cout<<"\nfirst complex number is ";c1.show();

cout<<"second complex number is ";c2.show();

c.addcomplex(c1,c2);

cout<<"sum of complex number is ";c.show();

c.subcomplex(c1,c2);

cout<<"difference of complex number is ";c.show();

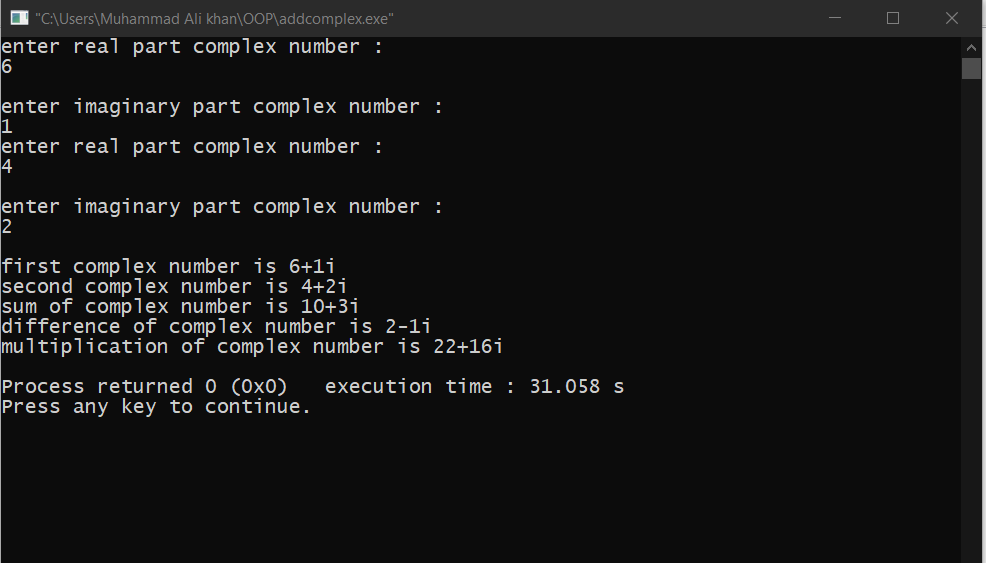
c.mulcomplex(c1,c2);

cout<<"multiplication of complex number is ";c.show();

return 0;

}

**Output display:-**



**Flowchart:-**

Class complex

Double real, imag

Complex (double r, double i)

void addcomplex(complex s1 ,complex s2), subcomplex(complex s1 ,complex s2), show(),

mulcomplex(complex s1 ,complex s2),input();

Initialize constructor

real=r, imag=i

Input();

Input real ,imag

addcomplex()

real= c1.real+c2.real;

imag=c1.imag+c2.imag;

subcomplex()

real= c1.real-c2.real;

imag=c1.imag-c2.imag;

mulcomplex()

real= c1.real\*c2.real-c1.imag\*c2,imag;

imag=c1.imag\*c2.real+c1.real\*c2.imag;

Show()

real”+”imag

main()

complex c1,c2,c3

Display

c1.input()

c2.input()

“first cmplex”c.show()

“second complex”c.show()

c.addcomplex(c1,c2);

“addition of complex number is”c.sh0w()

c.subcomplex(c1,c2);

“subtraction of complex number is”c.sh0w()

c.mulcomplex(c1,c2);

“multiplication of complex number is”c.sh0w()

**Activity Number 2.42**

**C++ code:-**

#include<iostream>

using namespace std;

class complex {

double real ,imag;

public:

complex();

complex(double real ,double imag);

void input();

complex addcomplex(complex s1);

complex subcomplex(complex s1);

complex mulcomplex(complex s1);

void show();

};

complex::complex():real(0),imag(0){}

complex::complex(double r,double i){

real=r;

imag=i;

}

void complex::input(){

cout<<"enter real part complex number :\n";

cin>>real;

cout<<"\nenter imaginary part complex number :\n";

cin>>imag;

}

complex complex::addcomplex( complex c1){

complex one(c1.real+real,c1.imag+imag);

return one;

}

complex complex::subcomplex( complex c1)

{

complex one(c1.real-real,c1.imag-imag);

return one;

}

complex complex::mulcomplex( complex c1){

complex one(c1.real\*real-c1.imag\*imag,c1.real\*imag+real\*c1.imag);

return one;

}

void complex::show(){

if(imag>0)

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

else

cout<<real<<imag<<"i"<<endl;

}

main(){

complex c,c1,c2;

cout;c1.input();

cout;c2.input();

cout<<"\nfirst complex number is ";c1.show();

cout<<"second complex number is ";c2.show();

c=c1.addcomplex(c2);

cout<<"sum of complex number is ";c.show();

c=c2.subcomplex(c1);

cout<<"difference of complex number is ";c.show();

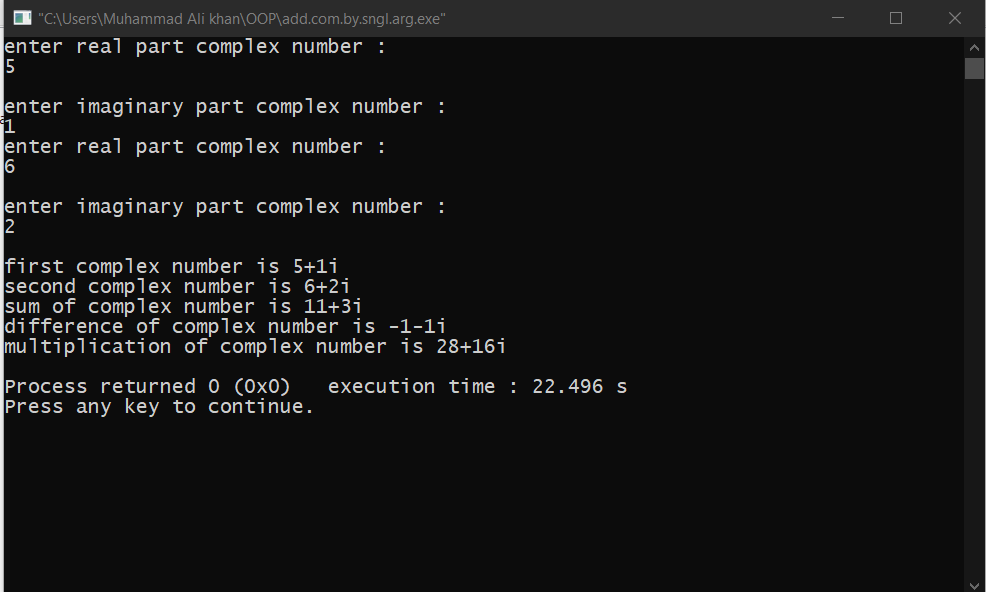
c=c2.mulcomplex(c1);

cout<<"multiplication of complex number is ";c.show();

return 0;

}

**Output display:-**



**Flowchart:-**

Class complex

Double real, imag

Complex (double r, double i)

complex addcomplex(complex s1 ), subcomplex(complex s1), show(),

mulcomplex(complex s1 ),input();

Initialize constructor

real=r, imag=i

Input();

Input real ,imag

addcomplex(c1.real+real,c1.imag+imag);

subcomplex( c1.real-real,c1.imag-imag);

mulcomplex( c1.real\*real-imag\*c1.imag, c1.imag\*real+c1.real\*imag);

Show()

real”+”imag

main()

complex c1,c2,c3

Display

c1.input()

c2.input()

“first cmplex”c.show()

“second complex”c.show()

c=c1.addcomplex(c2);

“addition of complex number is”c.sh0w()

c=c1.subcomplex(c2);

“subtraction of complex number is”c.sh0w()

c=c1.mulcomplex(c2);

“multiplication of complex number is”c.sh0w()

**Activity Number 2.4.3:-**

**C++ CODE :-**

#include<iostream>

using namespace std;

class IntegerSet {

private:

int array[50];

public:

IntegerSet() {

for(int i = 0; i < 50; i++) {

array[i] = 0;}

}

void newIntegerSet(int \* pointer) {

for(int i = 0; i < 50; i++) {

array[i] = \*(pointer + i);}

}

void insertElement(int k) {

array[k] = 1;

}

IntegerSet unionOfIntegerSets(IntegerSet par) {

IntegerSet temp;

for(int i = 0; i < 50; i++) {

if(array[i] == 1 || par.array[i]) {

temp.insertElement(i);

}}

return temp;

}

IntegerSet intersectionOfIntegerSets(IntegerSet par) {

IntegerSet temp;

for(int i = 0; i < 50; i++) {

if(array[i] == 1 && par.array[i] == 1)

temp.insertElement(i);

}

return temp;

}

void deleteElement(int indx) {

array[indx] = 0;

}

void setPrint() {

for(int i = 0; i < 50; i++)

cout<<array[i]<<" ";

cout<<endl;

}

bool isEqualTo(IntegerSet par) {

for(int i = 0; i < 50; i++) {

if(array[i] != par.array[i])

return false;}

return true; }

};

int main() {

int x[50];

for(int i = 0; i< 50; i++) x[i] = 0;

IntegerSet i1;

i1.newIntegerSet(x);

i1.insertElement(8); //Eight Position value will be display 1 //

i1.setPrint();

return 0;

}

**Output display:-**

