**TABLE OF CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO.** | **DATE** | **TITLE** | **PAGE NO.** | **SIGNATURE** |
| 1 |  | COMPLEX NUMBER   1. SET AND SHOW THE VALUE OF THE COMPLEX NUMBER |  |  |
| 1. ADD, SUBTRACT AND MULTIPLY TWO COMPLEX NUMBERS |  |  |
| 1. MULTIPLYING THE COMPLEX NUMBER WITH A SCALAR VALUE |  |  |
| 2 |  | A 2-D POINT IN A PLANE   1. SET AND SHOW THE VALUE OF A POINT |  |  |
| 1. FIND THE DISTANCE BETWEEN TWO POINTS |  |  |
| 1. CHECK WHETHER TWO POINTS ARE EQUAL OR NOT |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |

**Ex.1 (a) SET AND SHOW THE VALUE OF THE COMPLEX NUMBER**

**Aim**

To write a C++ program to set and show the value of the complex number

**Algorithm**

**Step 1:** Start the program.

**Step 2:** Declare and Define the Class Complex.

**Step 3:** Declare private data members as double real, double imag to store the real and imaginary part.

**Step 4:** Declare public member functions set(double r, double i) to initialize the complex number and show() to display the complex number.

Step 5: Declare an object c1 of class Complex.

Step 6: Display the Complex Number

Step 8: Stop the program.

**Program**

#include <iostream.h>

class Complex

{

private:

double real, imag;

public:

void set(double r, double i)

{

real = r;

imag = i;

}

void show()

{

cout<<"Complex Number:"<<real<<"+"<<imag<<"i"<<endl;

}

};

int main()

{

Complex c1;

c1.set(3.4,5.6);

c1.show();

return 0;

}

**OUTPUT**

Complex Number:3.4+5.6i

**Result**

Thus, the C++ program to set and show the value of the complex number has been created and executed successfully.

**Ex.1 (b) ADD, SUBTRACT AND MULTIPLY TWO COMPLEX NUMBERS**

**Aim**

To write a C++ program to add, subtract and multiply two complex number.

**Algorithm**

**Step 1:** Start the program.

**Step 2:** Declare and Define the Class Complex.

**Step 3:** Create Objects of Complex Class as c1, c2 and c3.

**Step 4:** Set Values for Complex Numbers c1 and c2.

**Step 5:** Call the add() method perform the Addition operation with c2 on c1 and Store the result in c3.

**Step 6:** Call the subtract()method perform the Addition operation with c2 on c1 and Store the result in c3.

**Step 7:** Call the multiply() method perform the Addition operation with c2 on c1 and Store the result in c3.

**Step 8:** Display the Complex Number results

**Step 9:** Stop the program.

**Program**

#include <iostream.h>

class Complex {

private:

double real, imag;

public:

// Set the value of the complex number

void set(double r, double i) {

real = r;

imag = i;

}

// Show the value of the complex number

void show() {

cout << "Complex Number: " << real << " + " << imag << "i" << endl;

}

// Add two complex numbers

Complex add(Complex c) {

Complex temp;

temp.real = real + c.real;

temp.imag = imag + c.imag;

return temp;

}

// Subtract two complex numbers

Complex subtract(Complex c) {

Complex temp;

temp.real = real - c.real;

temp.imag = imag - c.imag;

return temp;

}

// Multiply two complex numbers

Complex multiply(Complex c) {

Complex temp;

temp.real = real \* c.real - imag \* c.imag;

temp.imag = real \* c.imag + imag \* c.real;

return temp;

}

};

int main() {

Complex c1, c2, c3;

c1.set(3.4, 5.6);

c2.set(1.2, 3.4);

cout << "First Complex Number: ";

c1.show();

cout << "Second Complex Number: ";

c2.show();

// Add

c3 = c1.add(c2);

cout << "Addition: ";

c3.show();

// Subtract

c3 = c1.subtract(c2);

cout << "Subtraction: ";

c3.show();

// Multiply

c3 = c1.multiply(c2);

cout << "Multiplication: ";

c3.show();

return 0;

}

**OUTPUT**

First Complex Number: Complex Number: 3.4 + 5.6i

Second Complex Number: Complex Number: 1.2 + 3.4i

Addition: Complex Number: 4.6 + 9i

Subtraction: Complex Number: 2.2 + 2.2i

Multiplication: Complex Number: -14.96 + 18.28i

**Result**

Thus, the C++ program to add, subtract and multiply two complex number has been created and executed successfully.

**Ex.1 (c) MULTIPLYING THE COMPLEX NUMBER WITH A SCALAR VALUE**

**Aim**

To write a C++ program to multiply a complex number with scalar value.

**Algorithm**

**Step 1:** Start the program.

**Step 2:** Create Objects of Complex Class as c1.

**Step 3**: Initialize the complex number Set the real part of c1 to 3.4 and imaginary part of c1 to 5.6.

**Step 4:** Call the show() method of c1 to display the output of original complex number.

**Step 5:**  Define the scalar value for multiplication. Set a scalar variable to 2.0.

**Step 6:** Multiply the complex number with the scalar Call the multiply() method of c1 with scalar as argument.

**Step 7:** Call the multiply() method perform the Addition operation with c2 on c1 and Store the result in c3.

**Step 8:** Display the result.

**Step 9:** Stop the program.

**Program**

#include <iostream.h>

class Complex {

private:

double real, imag;

public:

// Set the value of the complex number

void set(double r, double i) {

real = r;

imag = i;

}

// Show the value of the complex number

void show() {

cout << "Complex Number: " << real << " + " << imag << "i" << endl;

}

// Multiply the complex number with a scalar value

void multiply(double scalar) {

real \*= scalar;

imag \*= scalar;

}

};

int main() {

Complex c1;

c1.set(3.4, 5.6);

cout << "Original Complex Number: ";

c1.show();

double scalar = 2.0;

c1.multiply(scalar);

cout << "After Multiplying with Scalar " << scalar << ": ";

c1.show();

return 0;

}

**OUTPUT**

Original Complex Number: Complex Number: 3.4 + 5.6i

After Multiplying with Scalar 2: Complex Number: 6.8 + 11.2i

**Result**

Thus, the C++ program to multiply a complex number with scalar value has been created and executed successfully.

**Ex.2 (a) A 2-D POINT PLANE - SET AND SHOW THE VALUE POINT**

**Aim**

To write a C++ program to set and show the value point in the 2-D plane.

**Algorithm**

**Step 1:** Start the program.

**Step 2:** Define and declare the class Point. Declare private data members x and y of type double.

**Step 3**: Declare public member functions: set(double x\_val, double y\_val) to assign values to x and y, show() to display the point's coordinates.

**Step 4:** Implement set() and show() functions.

**Step 5:**  Create an object p of class Point.

**Step 6:** Call the set() method on p with arguments 3.0 and 4.0 to initialize the point.

**Step 7:** Call the show() method on p to display the coordinates.

**Step 8:** Stop the program.

**Program**

#include <iostream.h>

class Point

{

private:

double x, y;

public:

void set(double x\_val, double y\_val)

{

x = x\_val;

y = y\_val;

}

void show()

{

cout << "Point: (" << x << ", " << y << ")" << endl;

}

};

int main()

{

Point p;

p.set(3.0, 4.0);

p.show();

return 0;

}

**OUTPUT**

Point: (3, 4)

**Result**

Thus, the C++ program to set and show the value point in the 2-D plane has been created and executed successfully.

**Ex.2 (b) FIND THE DISTANCE BETWEEN TWO POINTS**

**Aim**

To write a C++ program to find the distance between two points in the 2-D plane.

**Algorithm**

**Step 1:** Start the program.

**Step 2:** Define and declare the class Point. Declare private data members x and y of type double.

**Step 3**: Declare public member functions: set(double x\_val, double y\_val) to assign values to x and y.

**Step 4:** Declare show() to display the point's coordinates.

**Step 5:** Declare distance(Point p): Calculates and returns the Euclidean distance between the current point and another point p.

**Step 6:** Implement set(), show() and distance functions.

**Step 7:**  Calculate the difference in x coordinates: dx = x - p.x. Calculate the difference in y coordinates: dy = y - p.y.

**Step 8 :** Compute the distance using the formula: sqrt(dx\*dx + dy\*dy). Return the computed distance.

**Step 9 :** Create an object p1 and p2 of class Point.

**Step10:** Calculate and display the distance between the two points.

**Step 11:** Stop the program.

**Program**

#include <iostream.h>

#include <cmath.h>

class Point {

private:

double x, y;

public:

void set(double x\_val, double y\_val)

{

x = x\_val;

y = y\_val;

}

void show()

{

cout << "Point: (" << x << ", " << y << ")" << endl;

}

double distance(Point p)

{

double dx = x - p.x;

double dy = y - p.y;

return sqrt(dx \* dx + dy \* dy);

}

};

int main()

{

Point p1, p2;

p1.set(3.0, 4.0);

p2.set(6.0, 8.0);

cout << "First Point: ";

p1.show();

cout << "Second Point: ";

p2.show();

cout << "Distance between points: " << p1.distance(p2) << endl;

return 0;

}

**OUTPUT**

First Point: Point: (3, 4)

Second Point: Point: (6, 8)

Distance between points: 5

**RESULT**

Thus, the C++ program to find the distance between two points in the 2-D plane has been created and executed successfully.

**Ex.2 (c) CHECK WHETHER TWO POINTS ARE EQUAL OR NOT**

**Aim**

To write a C++ program to check whether two points are equal or not in the 2-D plane.

**Algorithm**

**Step 1:** Start the program.

**Step 2:** Define and declare the class Point. Declare private data members x and y of type double.

**Step 3**: Declare public member functions: set(double x\_val, double y\_val) to assign values to x and y.

**Step 4:** Declare show() to display the point's coordinates.

**Step 5:** Declare isEqual(Point other) Checks if the current point is equal to another point other.

**Step 6:** Implement set(), show() and isEqual() functions.

**Step 7:**  Compare the current point's x with other.x. Compare the current point's y with other.y.

**Step 8:** Return true if both are equal; otherwise, return false.

**Step 9:** Create an object p1, p2 and p3 of class Point.

**Step10:** Check and display equality.

**Step 11:** Stop the program.

**Program**

#include <iostream.h>

class Point {

private:

double x, y;

public:

// Methods to set and show the point

void set(double x\_val, double y\_val) {

x = x\_val;

y = y\_val;

}

void show() {

cout << "(" << x << ", " << y << ")";

}

// Method to check equality between two points

bool isEqual(Point other) {

return (x == other.x && y == other.y);

}

};

int main() {

// Create three points and set their coordinates

Point p1, p2, p3;

p1.set(3.0, 4.0);

p2.set(6.0, 8.0);

p3.set(3.0, 4.0); // p3 has the same coordinates as p1

// Display points

cout << "First Point: ";

p1.show();

cout << endl;

cout << "Second Point: ";

p2.show();

cout << endl;

cout << "Third Point: ";

p3.show();

cout << endl;

// Check equality between points and print results

cout << "First Point and Second Point are " << (p1.isEqual(p2) ? "equal." : "not equal.") << endl;

cout << "First Point and Third Point are " << (p1.isEqual(p3) ? "equal." : "not equal.") << endl;

return 0;

}

OUTPUT

First Point: (3, 4)

Second Point: (6, 8)

Third Point: (3, 4)

First Point and Second Point are not equal.

First Point and Third Point are equal.

**RESULT**

Thus, the C++ program to check whether two points are equal or not in the 2-D plane has been created and executed successfully