
Project - 1 MA517M-Basic Programming Laboratory	Last Date: 14 November 2026
Name	Roll No.: MA26M109

Polynomials over the Complex Field using C++ Classes and Operator Overloading

Objective: To design a C++ program that implements polynomials with complex coefficients using classes, supports polynomial arithmetic and evaluation, and demonstrates nested classes, operator overloading, and complex arithmetic.

A polynomial over the complex field $\mathbb{C}$ is an expression of the form

$$P(z) = a_0 + a_1z + a_2z^2 + \cdots + a_nz^n, \quad a_k \in \mathbb{C}.$$

Here both the variable z and the coefficients a_k may be complex numbers.

For example,

$$P(z) = (2 + i) + (3 - 2i)z + (1 + i)z^2.$$

The arithmetic of such polynomials is performed using the usual rules of polynomial arithmetic together with complex-number arithmetic.

Problem Description

Design two C++ classes:

1. **Complex** — represents a complex number $z = a + bi$.
2. **ComplexPolynomial** — represents a polynomial whose coefficients are complex numbers.

The **ComplexPolynomial** class should support polynomial addition, subtraction, multiplication, evaluation at a complex number, equality comparison, and display using operator overloading.

For example,

$$P(z) = (1 + i) + (2 - i)z + (3 + 2i)z^2$$

should be represented internally using an array of complex coefficients.

Class Specification: Complex

- **Class Name:** Complex
- **Private Data Members:**
 - double real;
 - double imag;
- **Public Member Functions:**
 - Complex(double r=0, double i=0);
 - double realPart() const;
 - double imagPart() const;
 - double modulus() const;
 - Complex conjugate() const;

Complex Operator Overloading

The following operators should be overloaded:

- operator +() — complex addition
- operator -() — complex subtraction
- operator *() — complex multiplication
- operator /() — complex division
- operator ==(), !=() — comparison
- operator <<() — display

For

$$z_1 = a + bi, \quad z_2 = c + di,$$

multiplication is

$$z_1 z_2 = (ac - bd) + (ad + bc)i.$$

Class Specification: ComplexPolynomial

- **Class Name:** ComplexPolynomial
- **Private Data Members:**
 - int degree;
 - Complex coeff[20];
- **Public Member Functions:**

- ComplexPolynomial(int degree=0);
- void setCoefficient(int power, Complex value);
- Complex getCoefficient(int power) const;
- int getDegree() const;
- Complex evaluate(Complex z) const;
- ComplexPolynomial derivative() const;
- void display() const;

Polynomial Addition

Let

$$P(z) = a_0 + a_1z + \cdots + a_nz^n$$

and

$$Q(z) = b_0 + b_1z + \cdots + b_mz^m,$$

where

$$a_k, b_k \in \mathbb{C}.$$

Then

$$(P + Q)(z) = (a_0 + b_0) + (a_1 + b_1)z + \cdots .$$

For example,

$$P(z) = (1 + i) + (2 - i)z + (3 + 2i)z^2$$

and

$$Q(z) = (2 - i) + (1 + 3i)z + (4 - i)z^2.$$

Then

$$P(z) + Q(z) = (3 + 0i) + (3 + 2i)z + (7 + i)z^2.$$

Polynomial Subtraction

Similarly,

$$(P - Q)(z) = (a_0 - b_0) + (a_1 - b_1)z + \cdots .$$

For the above polynomials,

$$P(z) - Q(z) = (-1 + 2i) + (1 - 4i)z + (-1 + 3i)z^2.$$

Polynomial Multiplication

If

$$P(z) = \sum_{i=0}^n a_i z^i$$

and

$$Q(z) = \sum_{j=0}^m b_j z^j,$$

then

$$P(z)Q(z) = \sum_{k=0}^{n+m} c_k z^k,$$

where

$$c_k = \sum_{i+j=k} a_i b_j.$$

The important difference from the real-coefficient polynomial case is that every coefficient multiplication

$$a_i b_j$$

is now a **complex multiplication**.

Polynomial Evaluation

The overloaded function-call operator

`operator()()`

should be used to evaluate the polynomial at a complex value.

For example, if

$$P(z) = (1 + i) + (2 - i)z + (3 + 2i)z^2,$$

then the following should be possible:

```
Complex z(1, 2);
```

```
cout << P(z);
```

Students should implement evaluation using Horner's method:

$$P(z) = ((a_n z + a_{n-1})z + a_{n-2})z + \cdots + a_0.$$

This reduces the number of polynomial multiplications and additions.

Operator Overloading for ComplexPolynomial

The following operators should be overloaded:

- `operator +()` Polynomial addition
- `operator -()` Polynomial subtraction
- `operator *()` Polynomial multiplication
- `operator ==()` Polynomial equality
- `operator !=()` Polynomial inequality
- `operator ()()` Polynomial evaluation at a complex number
- `operator <<()` Polynomial display

Tasks

1. Create the polynomial

$$P(z) = (1 + i) + (2 - i)z + (3 + 2i)z^2.$$

2. Create the polynomial

$$Q(z) = (2 - i) + (1 + 3i)z + (4 - i)z^2.$$

3. Compute

$$P(z) + Q(z).$$

4. Compute

$$P(z) - Q(z).$$

5. Compute

$$P(z)Q(z).$$

6. Evaluate $P(z)$ at

$$z = 1 + i.$$

7. Evaluate $Q(z)$ at

$$z = 2 - i.$$

8. Compute

$$P'(z)$$

using the derivative member function.

9. Verify polynomial equality using `operator ==()`.
10. Display all results using the overloaded `<<` operator.

Expected Output Example

$$P(z) = (1 + i) + (2 - i)z + (3 + 2i)z^2$$

$$Q(z) = (2 - i) + (1 + 3i)z + (4 - i)z^2$$

$$\begin{aligned} P(z) + Q(z) \\ = (3) + (3 + 2i)z + (7 + i)z^2 \end{aligned}$$

$$\begin{aligned} P(z) - Q(z) \\ = (-1 + 2i) + (1 - 4i)z + (-1 + 3i)z^2 \end{aligned}$$

$$\begin{aligned} P(z) * Q(z) \\ = [\text{complex polynomial obtained by multiplication}] \end{aligned}$$

$$\begin{aligned} P(1 + i) \\ = [\text{complex value}] \end{aligned}$$

$$\begin{aligned} Q(2 - i) \\ = [\text{complex value}] \end{aligned}$$

$$\begin{aligned} P(z) == Q(z) &: \text{False} \\ P(z) != Q(z) &: \text{True} \end{aligned}$$

Additional Challenge 1: Derivative

Implement

```
ComplexPolynomial derivative() const;
```

For

$$P(z) = a_0 + a_1z + a_2z^2 + \cdots + a_nz^n,$$

the derivative is

$$P'(z) = a_1 + 2a_2z + \cdots + na_nz^{n-1}.$$

Note that the coefficients a_k are complex numbers.

Additional Challenge 2: Polynomial Conjugate

Define the conjugate polynomial

$$\overline{P}(z) = \overline{a_0} + \overline{a_1}z + \cdots + \overline{a_n}z^n.$$

Implement

```
ComplexPolynomial conjugate() const;
```

and verify that

$$\overline{P + Q} = \overline{P} + \overline{Q}$$

and

$$\overline{PQ} = \overline{P} \overline{Q}.$$

Optional Extension: Polynomial Division

Extend the class to implement polynomial division over $\mathbb{C}$.

For

$$P(z) = Q(z)S(z) + R(z),$$

where

$$\deg(R) < \deg(Q),$$

implement

`divide()`

to return the quotient and remainder.

Verify the polynomial division identity numerically.

Project - 2: Number Slide Game Using C++ Classes

Problem Statement

Design and implement a **Number Slide Game** using **C++ classes**. The program should simulate a 4×4 sliding puzzle where the goal is to arrange numbers from 1 to 15 in order, leaving one empty space. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods for handling game logic.

Project Requirements

1. Create a **Matrix** class to represent the 4×4 game board.
 - (a) Include a method to generate a random matrix of size 4×4 using `srand()` in the range $[1, 16]$.
 - (b) Include a method to check that all entries of the matrix are distinct.
 - (c) Remove the entry with the value 16 to represent the empty space.
2. Create a **Game** class to manage gameplay.
 - (a) Display options for the user: **Play** or **Solution**.
 - (b) If the user chooses **Play**, show the navigation commands:
 - A/a for Left

- S/s for Down
- D/d for Right
- W/w for Up
- Q/q for Quit

(c) When the user presses Q/q, confirm again before quitting.

(d) After each move, display the updated matrix in a nicely formatted way.

3. If the user chooses **Solution**, display the sequence of movements leading to the solved puzzle, and show the final output.
4. Ensure proper encapsulation of game logic and board operations within the respective classes.

Suggested Class Structure

1. **Matrix** Class:

- Data member: 4×4 integer array representing the board
- Methods: generateMatrix(), checkDistinct(), removeEmpty(), displayMatrix()

2. **Game** Class:

- Data member: **Matrix** object, user choice
- Methods: playGame(), showSolution(), processMove(char move), confirmQuit()

Reference

For more details about the game, visit: <http://www.artbylogic.com/puzzles/numSlider/numberShuffle.htm>