
Project - 1	MA517M-Basic Programming Laboratory	Last Date: 09 November 2025
Name		Roll No.: MA25M018

Polynomial Arithmetic using C++ Classes and Operator Overloading

Objective: To design a C++ program that implements polynomials using classes, supports arithmetic operations, and demonstrates operator overloading.

A polynomial of degree n is of the form

$$P(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

where $a_i \in \mathbb{R}$. Arithmetic operations on polynomials include addition, subtraction, multiplication, evaluation, differentiation, and comparison.

Problem Description

Design a class `Polynomial` that represents polynomials. The class should support arithmetic operations, evaluation at a point, derivation, and comparison using operator overloading.

Class Specification

- **Class Name:** `Polynomial`
- **Private Data Members:**
 - `int degree;` // degree of polynomial
 - `double *coeffs;` // array of coefficients
- **Public Member Functions:**
 - `Polynomial(int deg, double c[]);` The constructor initializes the polynomial
 - `Polynomial derivative() const;` Returns a derivative polynomial
 - `Polynomial integral() const;` Returns an integral polynomial (with an arbitrary integration constant)
 - `double evaluate(double x) const;` Evaluates a polynomial at a given x
 - `void display() const;` Displays a polynomial in standard form

Operator Overloading

- `operator +()`, `operator -()` Polynomial addition and subtraction
- `operator *()` Polynomial multiplication
- `operator ==()` Compares if two polynomials are identical
- `operator <<()` Displays polynomial

Tasks

1. Create polynomials $P_1(x) = 3 + 2x + x^2$ and $P_2(x) = 1 - x + 4x^3$
2. Compute $P_1 + P_2$, $P_1 - P_2$, $P_1 * P_2$
3. Evaluate P_1 and P_2 at $x = 2$
4. Compute derivatives P_1' and P_2'
5. Compare polynomials using overloaded equality operator

Expected Output Example

$P_1(x) = 3 + 2x + x^2$

$P_2(x) = 1 - x + 0x^2 + 4x^3$

$P_1 + P_2 = 4 + x + x^2 + 4x^3$

$P_1 - P_2 = 2 + 3x + x^2 - 4x^3$

$P_1 * P_2 = 3 + 3x + 2x^2 + 11x^3 + 2x^4 + 4x^5$

$P_1(2) = 11$

$P_2(2) = 31$

Derivative $P_1' = 2 + 2x$

Derivative $P_2' = -1 + 12x^2$

$P_1 == P_2 : \text{False}$

Project - 2: Match-3 Game (Candy Crush Variant) Using C++ Classes

Problem Statement

Design and implement a **Match-3 Game** using **C++ classes**. The game consists of a grid of colored tiles (or symbols), where the player swaps adjacent tiles to form a sequence of three or more identical tiles in a row or column. When a match is formed, the tiles disappear, points are awarded, and new tiles fall from the top to fill empty spaces. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods to handle grid updates, scoring, and user interaction.

Project Requirements

1. Create a **Tile** class to represent an individual tile in the grid.
 - (a) Data member: type or color of the tile.
 - (b) Methods: `getType()`, `setType()`.
2. Create a **Board** class to represent the game grid.

- (a) Initialize the grid with random tiles.
 - (b) Display the current state of the grid in a readable format.
 - (c) Detect and remove matches of three or more tiles in a row or column.
 - (d) Drop new tiles from the top to fill empty spaces.
 - (e) Include a method to check if possible moves exist.
3. Create a **Game** class to manage gameplay.
 - (a) Display options for the user: **Play** or **Solution**.
 - (b) Allow the user to swap adjacent tiles using input commands.
 - (c) Keep track of the player's score.
 - (d) Allow the user to quit and confirm before exiting.
 - (e) Optionally, allow the user to reset the board or play multiple rounds.
 4. Ensure proper encapsulation of tile and board operations within the respective classes.

Suggested Class Structure

1. **Tile** Class:

- Data member: `int type` or `char color`
- Methods: `getType()`, `setType()`

2. **Board** Class:

- Data member: 2D array of **Tile** objects
- Methods: `initializeBoard()`, `displayBoard()`, `detectMatches()`, `removeMatches()`, `dropTiles()`, `hasPossibleMoves()`

3. **Game** Class:

- Data member: **Board** object, user choice, score
- Methods: `playGame()`, `showSolution()`, `processSwap(int row1, int col1, int row2, int col2)`, `confirmQuit()`, `resetBoard()`

Reference

For more details about Match-3 games and mechanics, visit: https://en.wikipedia.org/wiki/Tile-matching_video_game