Practical Number -2

Aim: To write and execute a program to represent polynomials using arrays and to perform addition of two polynomials by adding coefficients of like powers, thereby understanding the use of arrays in polynomial operations.

Objective

The objective of this experiment is to understand how a polynomial can be represented using arrays and to implement the addition of two polynomials using array-based representation. Through this experiment, students learn how mathematical concepts map naturally to data structures and how array indexing can be used to model polynomial degrees.

Theory

A polynomial is an algebraic expression consisting of variables and coefficients combined using addition, subtraction, and multiplication, where the exponent of the variable is a non-negative integer. A general polynomial in one variable can be written as:

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

Here, $a_0, a_1, a_2, \ldots, a_n$ are coefficients and the highest power of x determines the degree of the polynomial.

In programming, a polynomial can be efficiently represented using a one-dimensional array. In this representation, the index of the array corresponds to the power of x, and the value stored at that index represents the coefficient of that term. For example, the polynomial:

$P(x) = 4x^3 + 3x + 7$

can be represented as an array:

P[0] = 7
P[1] = 3
P[2] = 0
P[3] = 4

This representation is especially effective for dense polynomials where most degrees are present.

Polynomial addition involves combining like terms, meaning that coefficients of the same power of x are added together. When two polynomials are represented as arrays, polynomial addition becomes equivalent to adding corresponding elements of the two arrays. If the polynomials have different degrees, the resulting polynomial will have a degree equal to the maximum degree among the two.

Algorithm

The algorithm for polynomial addition using arrays works as follows. First, the degrees of both polynomials are identified. Arrays are then created to store the coefficients of each polynomial. A third array is created to store the result, with its size determined by the maximum degree of the two input polynomials plus one.

The program iterates through each index of the result array. If both polynomials contain a coefficient for the current degree, the coefficients are added. If only one polynomial contains a coefficient for that degree, it is copied directly into the result array. After completing the addition, the result array represents the sum of the two polynomials.

C Program: Polynomial Addition Using Arrays

```
#include <stdio.h>

int main() {
   int d1, d2, max;
   int i;

   printf("Enter degree of first polynomial: ");
   scanf("%d", &d1);

   int P1[d1 + 1];
   printf("Enter coefficients of first polynomial:\n");
   for (i = 0; i <= d1; i++) {
      printf("Coefficient of x^%d: ", i);
      scanf("%d", &P1[i]);
   }

   printf("Enter degree of second polynomial: ");
   scanf("%d", &d2);

   int P2[d2 + 1];
```

```
    printf("Enter coefficients of second polynomial:\n");
    for (i = 0; i <= d2; i++) {
        printf("Coefficient of x^%d: ", i);
        scanf("%d", &P2[i]);
    }

    max = (d1 > d2) ? d1 : d2;
    int Sum[max + 1];

    for (i = 0; i <= max; i++) {
        if (i <= d1 && i <= d2)
            Sum[i] = P1[i] + P2[i];
        else if (i <= d1)
            Sum[i] = P1[i];
        else
            Sum[i] = P2[i];
    }

    printf("Resultant polynomial:\n");
    for (i = max; i >= 0; i--) {
        if (Sum[i] != 0)
            printf("%dx^%d ", Sum[i], i);
        if (i > 0 && Sum[i] != 0)
            printf("+ ");
    }

    return 0;
}
```

Java Program: Polynomial Addition Using Arrays

```
import java.util.Scanner;

public class PolynomialAddition {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter degree of first polynomial: ");
        int d1 = sc.nextInt();
        int[] P1 = new int[d1 + 1];
```

```java
    System.out.println("Enter coefficients of first polynomial:");
    for (int i = 0; i <= d1; i++) {
        System.out.print("Coefficient of x^" + i + ": ");
        P1[i] = sc.nextInt();
    }

    System.out.print("Enter degree of second polynomial: ");
    int d2 = sc.nextInt();
    int[] P2 = new int[d2 + 1];

    System.out.println("Enter coefficients of second polynomial:");
    for (int i = 0; i <= d2; i++) {
        System.out.print("Coefficient of x^" + i + ": ");
        P2[i] = sc.nextInt();
    }

    int max = Math.max(d1, d2);
    int[] Sum = new int[max + 1];

    for (int i = 0; i <= max; i++) {
        if (i <= d1 && i <= d2)
            Sum[i] = P1[i] + P2[i];
        else if (i <= d1)
            Sum[i] = P1[i];
        else
            Sum[i] = P2[i];
    }

    System.out.println("Resultant polynomial:");
    for (int i = max; i >= 0; i--) {
        if (Sum[i] != 0) {
            System.out.print(Sum[i] + "x^" + i);
            if (i > 0)
                System.out.print(" + ");
        }
    }

    sc.close();
}
```

}

Result

The program successfully adds two polynomials by representing them as arrays and storing the result in a third array. The output polynomial correctly displays the sum of corresponding terms.

Conclusion

This experiment demonstrates how arrays provide a simple and efficient way to represent and manipulate polynomials in programming. Polynomial addition using arrays closely mirrors the mathematical process of combining like terms and serves as a foundational example of applying data structures to real-world mathematical problems.