Day 11: Matrix Addition and Subtraction

Gati Goyal

"Mathematics is the music of reason."
— James Joseph Sylvester

---

1 Introduction

Matrix addition and subtraction are fundamental operations in linear algebra. These operations involve adding or subtracting corresponding elements of two matrices of the same dimensions. This document provides the implementation of these operations using C programming.

2 Problem Statement

Problem: Perform addition and subtraction of two matrices of size $m \times n$. **Hint:** Use nested loops to access and manipulate individual elements of the matrices. **Condition:** Matrix addition and subtraction are valid only if the matrices have the same dimensions.

3 Matrix Notation

Let $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ be two matrices of size $m \times n$.

- **Addition:** $\mathbf{C} = \mathbf{A} + \mathbf{B}$, where $c_{ij} = a_{ij} + b_{ij}$ for $1 \leq i \leq m$, $1 \leq j \leq n$.
- **Subtraction:** $\mathbf{C} = \mathbf{A} - \mathbf{B}$, where $c_{ij} = a_{ij} - b_{ij}$ for $1 \leq i \leq m$, $1 \leq j \leq n$.

4 Algorithm

4.1 Steps to Solve the Problem

1. Input the dimensions of the matrices (m and n).
2. Input the elements of the two matrices.
3. For addition:
    - Iterate through each element of the matrices using nested loops.

  - Add corresponding elements and store the result in a new matrix.

4. For subtraction:
   - Iterate through each element of the matrices using nested loops.
   - Subtract corresponding elements and store the result in a new matrix.

5 Code

```
#include <stdio.h>

// Function to perform matrix addition
void addMatrices(int rows, int cols, int mat1[][10], int mat2[][10], int r
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}

// Function to perform matrix subtraction
void subtractMatrices(int rows, int cols, int mat1[][10], int mat2[][10], i
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = mat1[i][j] − mat2[i][j];
        }
    }
}

// Function to display a matrix
void displayMatrix(int rows, int cols, int mat[][10]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf(”%d␣”, mat[i][j]);
        }
        printf(”\n”);
    }
}

int main() {
    int rows, cols;
    int mat1[10][10], mat2[10][10], result[10][10];

    // Input dimensions of the matrices
    printf(”Enter␣the␣number␣of␣rows␣and␣columns␣of␣the␣matrices:␣”);
    scanf(”%d␣%d”, &rows, &cols);

    // Input first matrix
    printf("Enter the elements of the first matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }

    // Input second matrix
    printf("Enter the elements of the second matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }

    // Perform matrix addition
    addMatrices(rows, cols, mat1, mat2, result);
    printf("\nResult of matrix addition:\n");
    displayMatrix(rows, cols, result);

    // Perform matrix subtraction
    subtractMatrices(rows, cols, mat1, mat2, result);
    printf("\nResult of matrix subtraction:\n");
    displayMatrix(rows, cols, result);

    return 0;
}
```

6 Complexity Analysis

- **Time Complexity:** $O(m \cdot n)$ Each element of the matrices is accessed once for addition and once for subtraction.
- **Space Complexity:** $O(m \cdot n)$ An additional matrix is used to store the result.

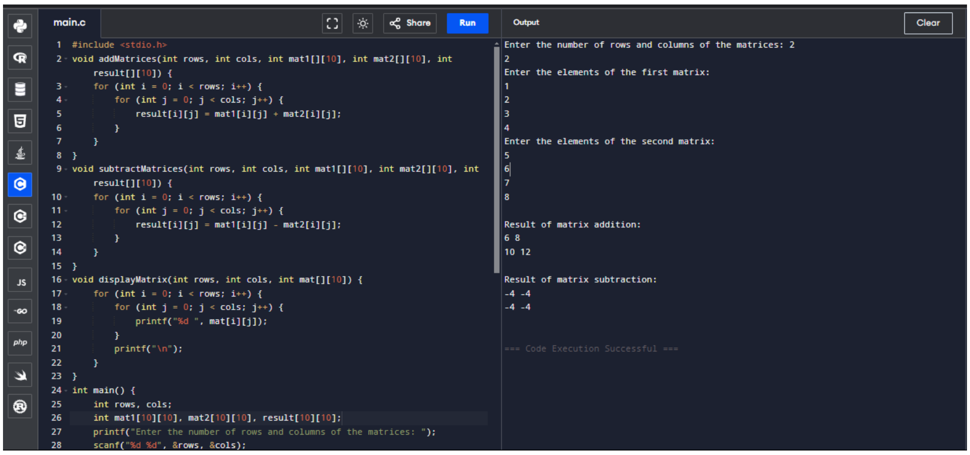
7 Examples and Edge Cases

Input Matrices	Addition Result	Subtraction Result
$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$	$\begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$	$\begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$

8 Conclusion

The program successfully performs addition and subtraction of two matrices of size $m \times n$. By leveraging nested loops for element-wise operations, the solution is both efficient and easy to understand. The conditions for valid matrix operations, such as ensuring both matrices have the same dimensions, are handled effectively. This implementation provides a clear foundation for understanding basic matrix operations in programming.

9 Output



```c
#include <stdio.h>
void addMatrices(int rows, int cols, int mat1[][10], int mat2[][10], int
    result[][10]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}
void subtractMatrices(int rows, int cols, int mat1[][10], int mat2[][10], int
    result[][10]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = mat1[i][j] - mat2[i][j];
        }
    }
}
void displayMatrix(int rows, int cols, int mat[][10]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}
int main() {
    int rows, cols;
    int mat1[10][10], mat2[10][10], result[10][10];
    printf("Enter the number of rows and columns of the matrices: ");
    scanf("%d %d", &rows, &cols);
```

```
Enter the number of rows and columns of the matrices: 2
2
Enter the elements of the first matrix:
1
2
3
4
Enter the elements of the second matrix:
5
6
7
8

Result of matrix addition:
6 8
10 12

Result of matrix subtraction:
-4 -4
-4 -4


=== Code Execution Successful ===
```

Figure 1: Output of the Matrix Addition and Subtraction Program