## Day_6 C-Programming

(RECAP OF PREVIOUS DAY)

Array(Two dimensional), its uses in matrix computation(Addition, multiplication,sum of diagonal elements, transpose). Passing array to function.

### Two-Dimensional Arrays

### Definition

A two-dimensional array is an array of arrays. It is used to store data in a tabular format, where data is organized in rows and columns.

### Declaration and Initialization

**Syntax:**

<datatype> arrayName[rows][columns];

**Example:**

```
#include <stdio.h>

int main() {

    int matrix[2][3] = {

        {1, 2, 3},

        {4, 5, 6}

    };
```

```
    printf("Two-dimensional array elements:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

**Output**:

Two-dimensional array elements:

1 2 3

4 5 6

---

**Uses in Matrix Computation**

**Matrix Addition**

Addition of two matrices involves adding corresponding elements of the matrices.

**Example:**

```
#include <stdio.h>

int main() {
    int A[2][2] = {{1, 2}, {3, 4}};
    int B[2][2] = {{5, 6}, {7, 8}};
    int C[2][2];
    printf("Matrix Addition:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            C[i][j] = A[i][j] + B[i][j];
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

**Output**:

Matrix Addition:

6 8

10 12

**Matrix Multiplication**

Matrix multiplication involves taking the dot product of rows and columns.

**Example:**

```
#include <stdio.h>

int main() {
    int A[2][2] = {{1, 2}, {3, 4}};
    int B[2][2] = {{5, 6}, {7, 8}};
    int C[2][2] = {0};
    printf("Matrix Multiplication:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            for (int k = 0; k < 2; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
            printf("%d ", C[i][j]);
        }
```

```
        printf("\n");
    }
    return 0;
}
```

**Output**:

Matrix Multiplication:

19 22

43 50

**Sum of Diagonal Elements**

Diagonal elements are those where the row index equals the column index.

**Example:**

```
#include <stdio.h>
int main() {
    int matrix[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
```

```
    };
    int sum = 0;
    for (int i = 0; i < 3; i++) {
        sum += matrix[i][i]; // Primary diagonal
    }

    printf("Sum of diagonal elements: %d\n", sum);
    return 0;
}
```

**Output**:

Sum of diagonal elements: 15

**Matrix Transpose**

Transpose of a matrix is obtained by interchanging its rows and columns.

**Example:**

```
#include <stdio.h>
int main() {
    int matrix[2][3] = {
```

```
    {1, 2, 3},
    {4, 5, 6}
};
int transpose[3][2];
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 3; j++) {
        transpose[j][i] = matrix[i][j];
    }
}

printf("Transpose of the matrix:\n");
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 2; j++) {
        printf("%d ", transpose[i][j]);
    }
    printf("\n");
}
```

```
    return 0;
}
```

**Output**:

Transpose of the matrix:

1 4

2 5

3 6

---

**Passing Arrays to Functions**

Two-dimensional arrays can be passed to functions as arguments by specifying the size of the second dimension.

**Example:**

```
#include <stdio.h>

void printMatrix(int matrix[2][3], int rows, int cols) {
    printf("Matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
```

```
        printf("\n");
    }
}

int main() {
    int matrix[2][3] = {
        {1, 2, 3},
        {4, 5, 6}
    };
    printMatrix(matrix, 2, 3);
    return 0;
}
```

**Output**:

Matrix:

1 2 3

4 5 6

---

## Practice Problems

1. Write a program to find the trace (sum of diagonal elements) of a 4x4 matrix.
2. Implement a function to compute the determinant of a 3x3 matrix.
3. Write a program to check if a given matrix is symmetric.
4. Implement addition and subtraction of two matrices using functions.
5. Create a program to find the row-wise and column-wise sum of a matrix.
6. Write a program to rotate a square matrix 90 degrees clockwise.
7. Implement a function to multiply two matrices of size MxN and NxP.
8. Write a program to find the largest element in each row of a matrix..
9. Write a program to check if two matrices are equal.
10. Write a program to find addition of Lower Triangular Elements in a Matrix.
11. Write a program to calculate sum of Upper Triangular Elements.