# MATRIX USING POINTER

## AIM:

To create a matrix using pointer and perform addition nd multiplication using pointer itself

## ALGORITHM:

· stdio.h and stdlib.h are included to perform input, output, and dynamic memory allocation.

· Function prototypes are declared for matrix addition and matrix multiplication using pointers.

· Variables are declared to store the row and column sizes of both matrices.

· The user enters the row and column size of the first matrix.

· Dynamic memory is allocated for the first matrix using malloc().

· The elements of the first matrix are read using pointer arithmetic.

· The user enters the row and column size of the second matrix.

· Dynamic memory is allocated for the second matrix using malloc().

· The elements of the second matrix are read using pointer arithmetic.

· Matrix dimensions are checked to verify whether addition is possible.

· Dynamic memory is allocated for the sum matrix using calloc().

· AddMatix() is called to add the two matrices using pointers.

· The sum of the two matrices is displayed.

· Matrix dimensions are checked to verify whether multiplication is possible.

· Dynamic memory is allocated for the product matrix using calloc().

· MultiplyMatrix() is called to multiply the two matrices using pointers.

· The product of the two matrices is displayed.

· All dynamically allocated memory is released using free().

· The program ends after completing matrix addition and multiplication.

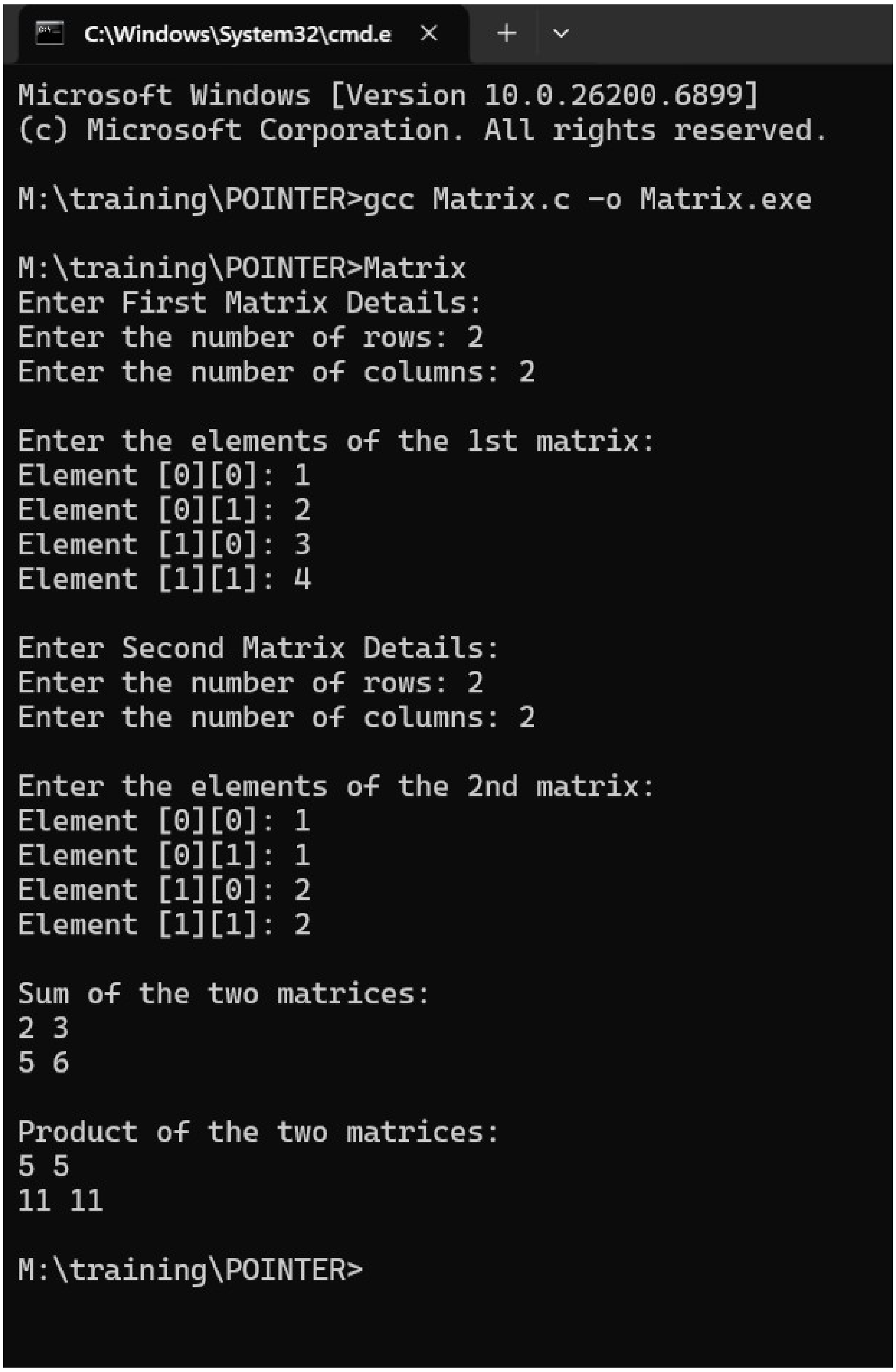
·

## PROGRAM:

```
/*
*   Program to perform Matrix calculation by Pointers
*   Author   : MUTHUGANESH S
*   Date     : 16/01/2026
```

```c
*   Filename : Matrix.c
*   retval   : void
*/

#include <stdio.h>
#include <stdlib.h>

void AddMatix(int *Mat1,int *Mat2,int *SumMat,int Rows,int Cols);
void MultiplyMatrix(int *Mat1,int *Mat2,int *MulMat,int Rows1,int Cols1,int
Rows2,int Cols2);

int main(void){
    int RowSize1, ColSize1, RowSize2, ColSize2;

    printf("Enter First Matrix Details:\n");
    printf("Enter the number of rows: ");
    scanf("%d", &RowSize1);
    printf("Enter the number of columns: ");
    scanf("%d", &ColSize1);

    int *Matrix1=(int *)malloc(RowSize1*ColSize1*sizeof(int));

    if (Matrix1 == NULL){
        printf("Memory allocation failed\n");
        return 0;
    }

    printf("\nEnter the elements of the 1st matrix:\n");
    for(int i = 0; i < RowSize1; i++){
        for(int j = 0; j < ColSize1; j++){

            printf("Element [%d][%d]: ", i, j);
            scanf("%d", &Matrix1[i*ColSize1 + j]);

        }
    }

    printf("\nEnter Second Matrix Details:\n");
    printf("Enter the number of rows: ");
    scanf("%d", &RowSize2);
    printf("Enter the number of columns: ");
    scanf("%d", &ColSize2);

    int *Matrix2=(int *)malloc(RowSize2*ColSize2*sizeof(int));

    if (Matrix2 == NULL){
        printf("Memory allocation failed\n");
        free(Matrix1);
        return 0;
    }

    printf("\nEnter the elements of the 2nd matrix:\n");
    for(int i = 0; i < RowSize2; i++){
        for(int j = 0; j < ColSize2; j++){
            printf("Element [%d][%d]: ", i, j);
            scanf("%d", &Matrix2[i*ColSize2 + j]);

        }
    }

    if(RowSize1 != RowSize2 || ColSize1 != ColSize2){
        printf("Matrix addition not possible due to dimension mismatch.\n");
        free(Matrix1);
        free(Matrix2);
        return 0;
    }

    int *SumMatrix=(int *)calloc(RowSize1*ColSize1, sizeof(int));
    if (SumMatrix == NULL){
        printf("Memory allocation failed\n");
        free(Matrix1);
        free(Matrix2);
        return 0;
    }

    AddMatix(Matrix1, Matrix2, SumMatrix, RowSize1, ColSize1);

    printf("\nSum of the two matrices:\n");

    for(int i = 0; i < RowSize1; i++){
        for(int j = 0; j < ColSize1; j++){

            printf("%d ", SumMatrix[i*ColSize1 + j]);

        }
        printf("\n");
    }

    if(ColSize1 != RowSize2){
        printf("Matrix multiplication not possible due to dimension 
mismatch.\n");
        free(Matrix1);
        free(Matrix2);
        free(SumMatrix);
        return 0;
    }

    int *MulMatrix=(int *)calloc(RowSize1*ColSize2, sizeof(int));

    if (MulMatrix == NULL){
        printf("Memory allocation failed\n");
        free(Matrix1);
        free(Matrix2);
        free(SumMatrix);
        return 0;
    }
    MultiplyMatrix(Matrix1, Matrix2, MulMatrix, RowSize1, ColSize1, RowSize2, 
ColSize2);

    printf("\nProduct of the two matrices:\n");
    for(int i = 0; i < RowSize1; i++){
        for(int j = 0; j < ColSize2; j++){

            printf("%d ", MulMatrix[i*ColSize2 + j]);
        }
        printf("\n");
    }

    free(Matrix1);
    free(Matrix2);
    free(SumMatrix);
    free(MulMatrix);

    return 0;
}

// Function to add two matrices
void AddMatix(int *Mat1, int *Mat2, int *SumMat, int Rows, int Cols){
    for(int i = 0; i < Rows; i++){
        for(int j = 0; j < Cols; j++){

            SumMat[i*Cols + j] = Mat1[i*Cols + j] + Mat2[i*Cols + j];

        }
    }
}

// Function to multiply two matrices
void MultiplyMatrix(int *Mat1, int *Mat2, int *MulMat, int Rows1, int Cols1, 
int Rows2, int Cols2){

    for(int i = 0; i < Rows1; i++){
        for(int j = 0; j < Cols2; j++){

            MulMat[i*Cols2 + j] = 0;

            for(int k = 0; k < Cols1; k++){

                MulMat[i*Cols2 + j] += Mat1[i*Cols1 + k] * Mat2[k*Cols2 + j];

            }
        }
    }
}
```

**OUTPUT:**



```
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

M:\training\POINTER>gcc Matrix.c -o Matrix.exe

M:\training\POINTER>Matrix
Enter First Matrix Details:
Enter the number of rows: 2
Enter the number of columns: 2

Enter the elements of the 1st matrix:
Element [0][0]: 1
Element [0][1]: 2
Element [1][0]: 3
Element [1][1]: 4

Enter Second Matrix Details:
Enter the number of rows: 2
Enter the number of columns: 2

Enter the elements of the 2nd matrix:
Element [0][0]: 1
Element [0][1]: 1
Element [1][0]: 2
Element [1][1]: 2

Sum of the two matrices:
2 3
5 6

Product of the two matrices:
5 5
11 11

M:\training\POINTER>
```