13/9/22 SY CSF ; Academic Year 2022-23 ; Sem - III

Fundamentals of Data Structures

Lab Write UP

Experiment - 01

Title = " Matrix Operations "

(A) **Problem Statement :**

Write a c program to perform the following computations on a matrix.

A. Addition of 2 Matrices
B. Subtraction of 2 Matrices
C. Multiplication of 2 Matrices
D. Transpose of a Matrix.

(A) **Objective**

1. To study memory representations and operations associated with arrays.
2. To understand the implementation of formal and actual parameters.

(A) **THEORY**

I. **One Dimensional Arrays :**

It is a simple list of elements of

Rainbow

just one row of elements arranged contiguously in memory.

eg: int arr[5] = {1, 2, 3, 4, 5};

(4) 2 Dimensional arrays

→ List of ~~integ~~ elements arranged contiguously in memory; row wise or column wise. They ~~are~~ can be represented by or as a matrix ~~(2 x 2)~~ (i x j) with i rows, j cols

eg. int arr[2][2] = { {1, 2}, {3, 4} };

(4) Matrix Operations:

(1) Addition:

→ The resulting 2D array or Matrix is the sum of each respectively element of the ~~arrs~~ given 2 input matrices

eg.

$$\underset{A}{\begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}} + \underset{B}{\begin{bmatrix} 5 & 6 \\ 7 & 3 \end{bmatrix}} = \underset{\text{Result}}{\begin{bmatrix} 6 & 6 \\ 7 & 8 \end{bmatrix}}$$

(2) Subtraction

Each element of resulting matrix is the difference of elements in given matrices.

A B Result

$$\begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} 5 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -4 & -1 \\ 1 & 2 \end{bmatrix}$$

(3) Multiplication :

→ Multiplication in matrix is done row-columnwise ; each row of Matrix A is multiplied with each column of Matrix B.

So n (rows) [A] == n(cols) [B]

eg.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a^2+bc & ab+bd \\ ac+dc & bc+d^2 \end{bmatrix}$$

(4) Transpose.

→ The rows and columns of a matrix are switched.

eg.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad T \rightarrow \quad \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

(*) Formal Parameters : The parameters of a function written in its definition or declaration.

eg. int some_function (int para_i, int i);

Actual Parameters :

→ The actual variables that are given to the function during it call ;

eg.

```
int A ( int a )          // Formal Parameter (a)
{ return a* a ; }

int main ()
{
    int val = 5 ;
    cout << A (5) ;      // actual Param = 5

    return 0 ;
}
```

Implementation :

→ Platform : 64 bit Archlinux
→ Editor : Vs code
→ Compiler : mingw gcc or g++.

Pseudo code :

(1) Add (~~int~~ ~~rows~~ M1 [][20], M2 [][20], M3 [][20]
rows, cols)
{ for i in range (rows) :
for j in range (cols) :
M3 [i][j] = M1 [i][j] + M2 [i][j] ;

rows
cols
(rows * cols)

Time complexity : O (n^2) if square with n rows.

if (Matrix.shape == ~~type~~ square)

Rainbow

(2) Subtraction:

```
void sub ( M1 [][], M2[][] ,M3[][], r, c)
r       for i in range (r):
r+1         for j in range (c):
r*c             M3 [i][j] = M1 [i][j] - M2[i][j]
```

Time complexity : $O(r^2)$
if n rows and
n cols. is matrix.

(3) Multiplication:

```
void mul ( M1[][] , M2[][] ,M3[][], r, c ):
r       for K in range (r):
r+1         for i in range (c):
(r*c)+1         for j in range (c):
r*c*c               M3[i][j] += M1[i][K] + M2[K][j]
```

Time complexity = $O(r*c*c)$
or $O(n^3)$ for
square mat.

(4) Transpose

```
void transpose ( M1 [][] , M2 [][] , r , c )
r       for i in range ( r ):
r+1         for j in range (c):
r*c             M2 [i][j] = M1 [j][i] ;
```

Time complexity = $O(r*c)$
or $O(n^2)$ if square mat

Conclusion:

Thus implemented matrix operations using different functions.

FAQs

(1) Diff b/w pass by value & pass by reference in C functions.

→ When a function call is given the actual value of the variable of data type it expects, it is passed by value to it. The actual value of the variable is ~~copied~~ Copied to the function.

eg `void add (int a, int b);`

```
void main ()
{   add (1, 2);
}
```

Pass by reference: When instead of the value, the pointer to the value is given to the function, it is passed by reference

```
int add (int* a, int* b);
void main ()
{   int a = 5, b = 4;
    int *p1 = &a;
    int *p2 = &b; add (p1, p2);
}
```

Rainbow

(12) Difference between single quoted and double quoted declaration of character array:

char a[] = "hello";

→ in this case C automatically understands that the characters entered are ordered in an array and appends '\0' to it after assigning.

C interprets this as a string.

char a[2] = {'a', 'b'};

Simply declares array with 2 bytes having 2 elements 'a' and 'b'.