

3/9/22

SY CSF ; Academic Year 2022-23 ; Sem - III

## Fundamentals of Data Structures

### Lab Write Up

#### Experiment - 01

Title = "Matrix Operations"



#### 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



#### Objectives:

1. To study memory representations and operations associated with arrays.

2. To understand the implementation of formal and actual Parameters



#### THEORY

##### I. One Dimensional Arrays :

It is a simple list of elements of

just one row of elements arranged contiguously in memory.

e.g.:  $\text{int arr[5]} = \{1, 2, 3, 4, 5\};$

### (\*) 2 Dimensional arrays

→ List of integer elements arranged contiguously in memory; row wise or column wise.  
They can be represented by or as a matrix  $(2 \times 2) (i \times j)$  with  $i$  rows,  $j$  cols

e.g.  $\text{int arr[2][2]} = \{ \{1, 2\}, \{3, 4\} \};$

### (\*) Matrix Operations:

#### (1) Addition:

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

e.g.

$$\begin{matrix} A \\ \left[ \begin{matrix} 1 & 0 \\ 0 & 5 \end{matrix} \right] \end{matrix} + \begin{matrix} B \\ \left[ \begin{matrix} 5 & 6 \\ 7 & 3 \end{matrix} \right] \end{matrix} = \begin{matrix} \text{Result} \\ \left[ \begin{matrix} 6 & 6 \\ 7 & 8 \end{matrix} \right] \end{matrix}$$

#### (2) Subtraction

Each element of resulting matrix is the difference of elements is given

$$\begin{array}{c}
 A \\
 \left[ \begin{array}{cc} 1 & 0 \\ 2 & 3 \end{array} \right] * \left[ \begin{array}{cc} 5 & 1 \\ 1 & 1 \end{array} \right] = \left[ \begin{array}{cc} -4 & -1 \\ 1 & 2 \end{array} \right]
 \end{array}$$

(3) Multiplication:

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

$$\text{so } n(\text{rows}) [A] = n(\text{cols}) [B]$$

e.g.

$$\left[ \begin{array}{cc} a & b \\ c & d \end{array} \right] \left[ \begin{array}{cc} a & b \\ c & d \end{array} \right] = \left[ \begin{array}{cc} a^2 + bc & ab + bd \\ ac + dc & bc + d^2 \end{array} \right]$$

(4)

Transpose:

→ The rows and columns of a matrix are switched.

e.g.

$$\left[ \begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right] \xrightarrow{T} \left[ \begin{array}{cc} 1 & 3 \\ 2 & 4 \end{array} \right]$$

(\*)

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

e.g.

int some\_function (int para<sub>1</sub>, int i)



## Actual Parameters :

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

e.g.

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

```
int main ()
```

```
{
```

```
int val = 5;
```

```
cout << A(5); // actual param = 5
```

```
: return 0;
```

```
}
```



## Implementation :

→ Platform : 64 bit Archlinux

→ Editor : VS code

→ Compiles : mingw gcc or g++.



## Pseudo Code :



Add ( ~~int~~ M1 [ ] [ 20 ], ~~M2 [ ] [ 20 ]~~, ~~M3 [ ] [ 20 ]~~  
rows, cols )

rows

```
{ for i in range (rows) :
```

cols

```
    for j in range (cols) :
```

(rows \* cols)

```
        M3 [ i ] [ j ] = M1 [ i ] [ j ] + M2 [ i ] [ j ] ;
```

Rainbow

Time complexity :  $O(n^2)$  if square with  $n$  rows.  
if (Matrix.shape == ~~not~~ square)

(2)

### Subtraction:

```
void sub ( M1 [][], M2 [][], M3 [][], r, c )
```

r  
r+1  
r\*c

for i in range (r) :

for j in range (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) :

$$M3[i][j] += M1[i][K] * M2[K][j]$$

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) :

$$M2[i][j] = M1[j][i];$$

Time complexity =  $O(r * c)$

or  $O(n^2)$  if square mat

(\*)

## Conclusion:

Thus implemented matrix operations using different functions.

(\*)

## FADs

(1)

Difference between 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 to the function.

e.g. void add (int a, int b);

void ~~is~~ 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;

    int a = 5, b = 4; \*;

    int \*p1 = &a;

    int \*p2 = &b; add (p1, p2); }

(2)

Difference between single quoted and double quoted declaration of character array:

```
char a[5] = " 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 declared array with 2 bytes having 2 elements 'a' and 'b'.

✓

Q.

03/10