Date 30/10/2023 Page

Day - 43

2D - Arrays

=> As we have seen excel sheets.

=> So, if we want to store the data of excel sheet then with the help of 2D Arrays, we can easily do this.

=> Let assume, we want a 4 row & 3 column table, then -

int arr[4][3]; → 2D Array

	0	1	2
0	00	01	02
1	10	11	12
2	20	21	22
3	30	31	32

=> So, now, how 2D Array store in memory.

=> It store in two ways in memory - as row major & column major.

=> Row major

	00	01	02	10	11	12
20	21	22	30	31	32	

→ memory

=> Store row wise in contagious manner.

0	1	2	3	4	5	6	7	8	9	10	11
00	01	02	10	11	12	20	21	22	30	31	32

Memory

Date ______ Page ______

⇒ Now, if we want to calculate the position of any element then —

Index = row_index × col + col_index
of memory — ①

⇒ Now, if we have index, then we can calculate the indexing of the element.

row_index → Index / col;
col_index → Index % col;

⇒ As we know, $0 <= col\_index < col$
So, divide the eqn ① by col,

$$\frac{Index}{col} = \frac{row\_index \times \cancel{col}}{\cancel{col}} + \overset{0}{\cancel{\frac{col\_index}{col}}}$$

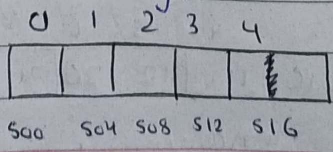
⇒ So, row_index = Index / col.

⇒ For col_index,

$$(Index) \% col = (\overset{0}{\cancel{row\_index \times col}} + col\_index) \% col$$

⇒ So, col_index = Index % col.

⇒ int arr[5];
Here, arr store the starting address of the array.



=) If you want to calculate the address of particular index —

address

arr[3] = base ~~index~~ + index X size of element

= 500 + 3X4

= 512.

=) Same thing happens in 2D-Array —

=) arr[2][1] = base address + index X size of element

= 500 + 7X4

= 528

=) arr[i][j] = base address + (i X col + j) X size of element.

Initialize

=) int arr[4][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

Update or user i/p

=) arr[3][0] = 15

=) cin >> arr[3][1];

=) Printing the elements

```
for( i = 0; i< row; i++)
    for( j = 0; j< col; j++)
        cout << arr[i][j];
```

* Search element in Array

```
    int arr [4][3] ;          int x = 7;
int main(){
    int arr [4][3] = {1, 2, 3, ..., 12}
    int x = 7;
   for (int i = 0; i<4; i++)
        for (int j = 0; j<3; j++){
            if( arr[i][j] == x )
                cout << "Yes";
        }
   cout << "No";
}
```

⇒ when we pass 2D Array in any function call then we have to must pass the col value.

```
void print (int arr[][4]){
    ;
}
```

* Add 2 Matrix

2	3	4	5
6	7	8	9
10	11	12	13

arr1[3][4]

2	3	4	5
6	7	8	9
10	11	12	13

arr2[3][4]

⇒ arr[i][j] += arr1[i][j] + arr2[i][j];

T.C. → O(row * col).

S.C. → O(row * col)

* **Print row index with Max sum**

$$\begin{array}{c|ccc|l} & 0 & 1 & 2 & \\ \hline 0 & 1 & 2 & 3 & \to 6 \\ 1 & 4 & 5 & 6 & \to 15 \\ 2 & 7 & 8 & 9 & \to 24 \checkmark \end{array}$$

```
int sum =  INT_MIN, index = 0;
for( int i = 0; i < row ; i++){
        int total = 0;
        for (int j = 0 ; j < col; j++)
              total += arr[i][j];
        if ( sum < total){
              sum = total;
              index = i ;
        }
}
cout << index;
```

T.C → O(row * col) → ~~B~~W.C.

~~O(1) → B.C.~~

* **Print sum of diagonal Element**

Date

Page

$$\begin{array}{c|c|c|c|c|} & 0 & 1 & 2 & 3 \\ \hline 0 & 5_{00} & 8 & 3 & 9_{03} \\ \hline 1 & 6 & 2_{11} & 8_{12} & 4 \\ \hline 2 & 5 & 3_{21} & 2_{22} & 2 \\ \hline 3 & 2_{30} & 8 & 1 & 9_{33} \\ \hline \end{array}$$

⇒ Condition for this question is row==col.

⇒ Here, we have two diagonal —
So, for calculating sum of first diagonal.

⇒
```
int first =0;
for(inti=0; i<row; i++){
    first += arr[i][i];

int second=0;
int i= 0; j= col-1;
while (j>= 0){
    second += arr[i][j];
    i++; j--;
}
```

T.C. → O(row).

* Reverse each row of matrix

$$\begin{array}{c|c|c|c|} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ \hline 1 & 4 & 5 & 6 \\ \hline 2 & 7 & 8 & 9 \\ \hline \end{array} \longrightarrow \begin{array}{c|c|c|c|} & 0 & 1 & 2 \\ \hline 0 & 3 & 2 & 1 \\ \hline 1 & 6 & 5 & 4 \\ \hline 2 & 9 & 8 & 7 \\ \hline \end{array}$$

Code

```
for( int i= 0; i< row; i++){
        int start= 0 , end = col-1 ;
        while( start < end){
                swap (arr[i][start], arr[i][end]);
                start++ ; end--;
```