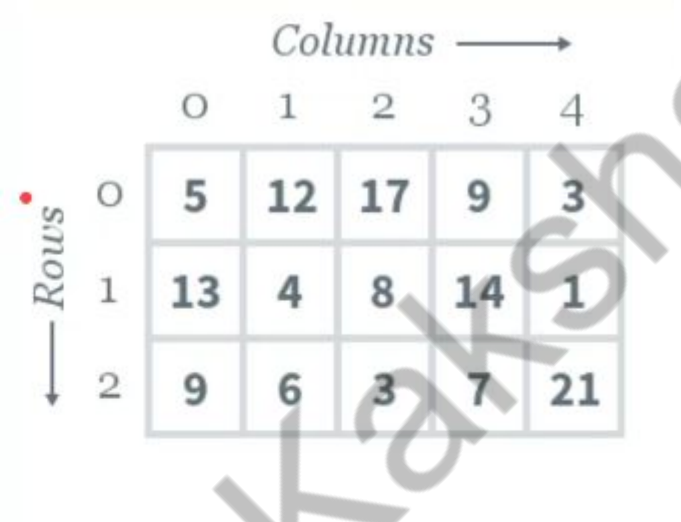
2 Dimensional Arrays

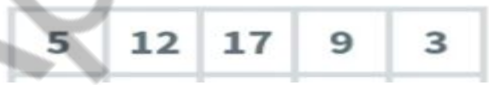
Notes

1. It is similar to 2D matrices that we studied in 11th and 12th class.

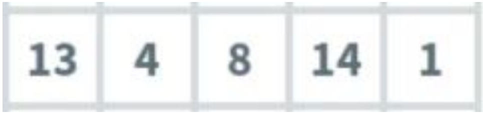

Columns →
Rows ↓

	0	1	2	3	4
0	5	12	17	9	3
1	13	4	8	14	1
2	9	6	3	7	21

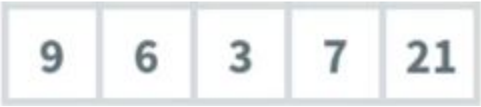
2D Array of size 3 x 5

2. It has 2 parts
    a. **Rows** - Horizontal Arrays in the 2D matrix. For instance, in the above example, we have 3 rows:
        i. 

5	12	17	9	3

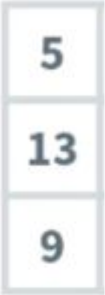
        ii. 

13	4	8	14	1

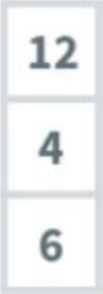
        iii. 

9	6	3	7	21

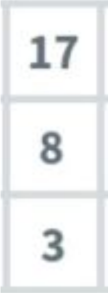
    b. **Columns** - Vertical Arrays in the 2D matrix. For instance, in the above example, we have 5 columns:

i. 

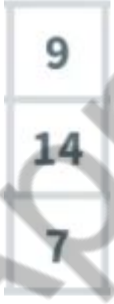
5
13
9

ii. 

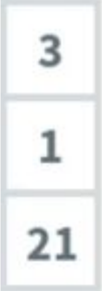
12
4
6

iii. 

17
8
3

iv. 

9
14
7

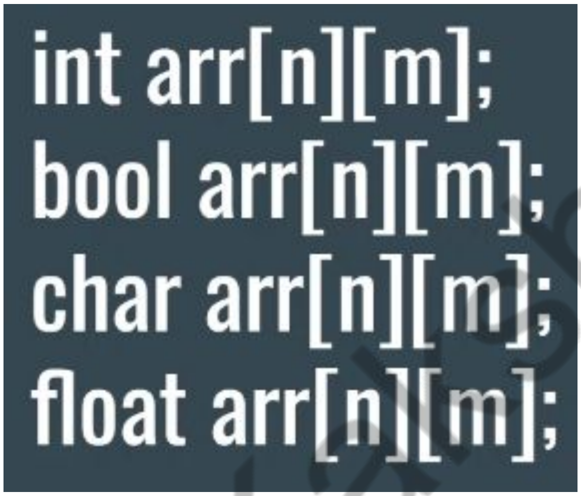
v. 

3
1
21

3. Note: Indexing of both rows and columns starts with 0.

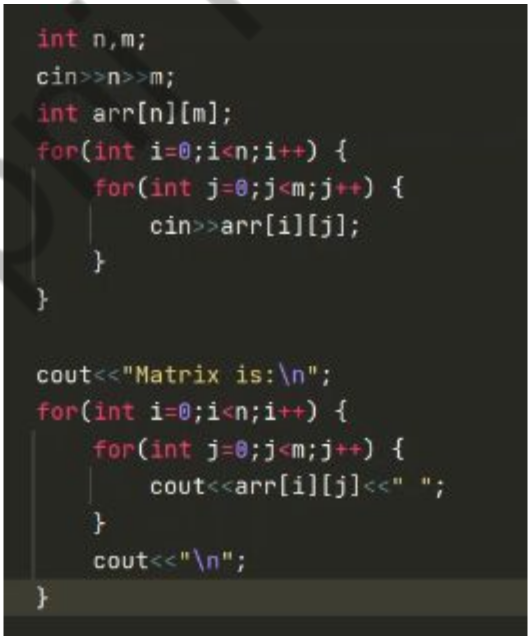
Declaration of 2D matrices

1. 2D Arrays are declared similar to 1D arrays but with an additional dimension.
   Syntax: int arr[rows][columns]
   For example,



```
int arr[n][m];
bool arr[n][m];
char arr[n][m];
float arr[n][m];
```

Code:

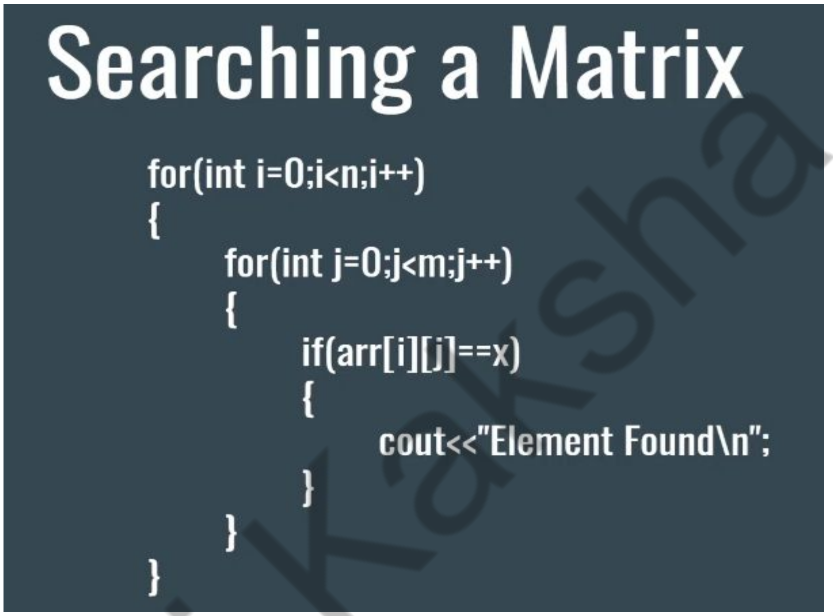


```
int n,m;
cin>>n>>m;
int arr[n][m];
for(int i=0;i<n;i++) {
    for(int j=0;j<m;j++) {
        cin>>arr[i][j];
    }
}

cout<<"Matrix is:\n";
for(int i=0;i<n;i++) {
    for(int j=0;j<m;j++) {
        cout<<arr[i][j]<<" ";
    }
    cout<<"\n";
}
```

Searching in a matrix

Problem: We have to find if value x is present in the 2D array.

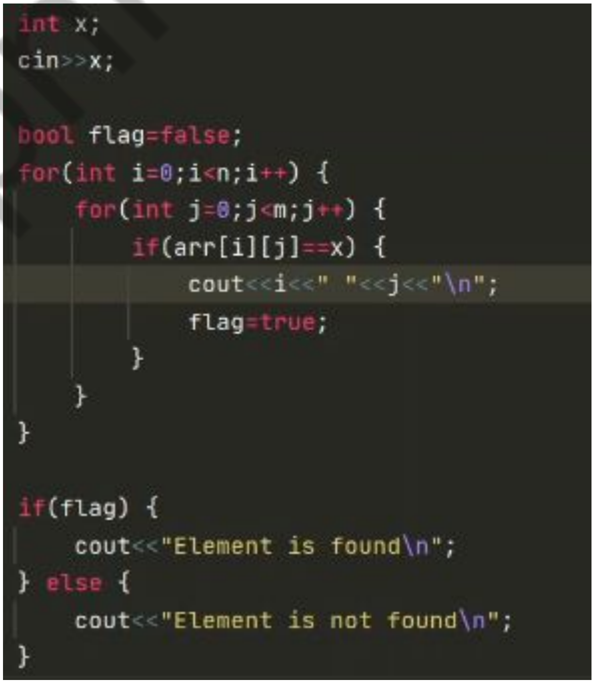
1. While traversing in the 2D matrix, just we have to put one if statement which checks if(a[i][j] == x) , then x is present otherwise not.



Searching a Matrix

```
for(int i=0;i<n;i++)
{
        for(int j=0;j<m;j++)
        {
                if(arr[i][j]==x)
                {
                        cout<<"Element Found\n";
                }
        }
}
```

Code:

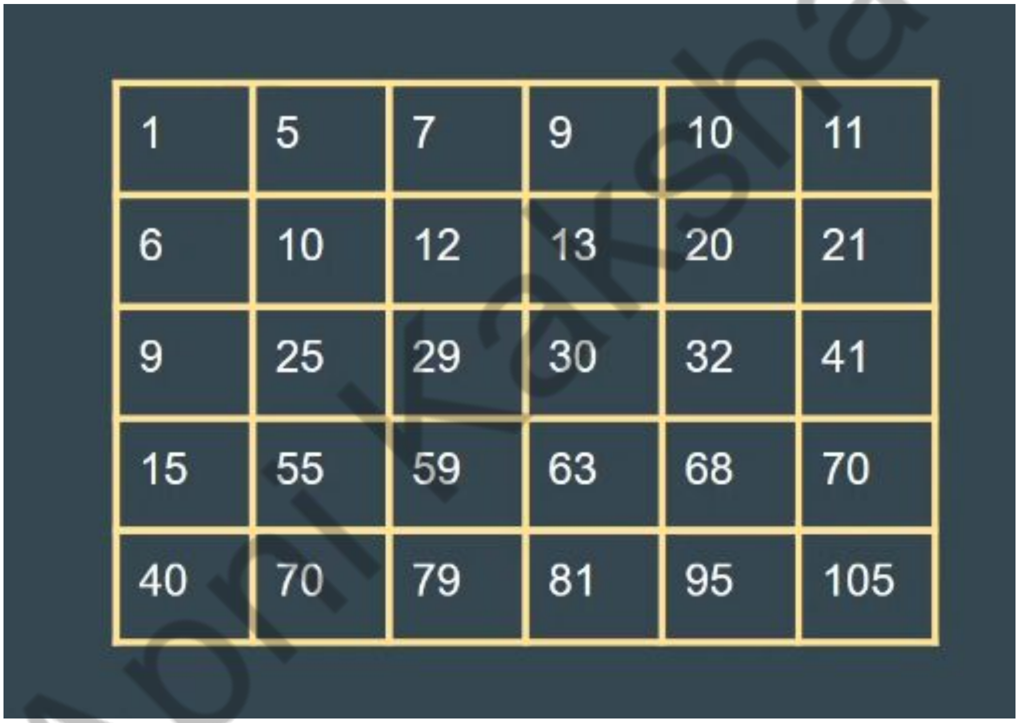


```
int x;
cin>>x;

bool flag=false;
for(int i=0;i<n;i++) {
    for(int j=0;j<m;j++) {
        if(arr[i][j]==x) {
            cout<<i<<" "<<j<<"\n";
            flag=true;
        }
    }
}

if(flag) {
    cout<<"Element is found\n";
} else {
    cout<<"Element is not found\n";
}
```

Spiral Order Matrix Traversal

Problem: We have to print the given 2D matrix in the spiral order. Spiral Order means that firstly, first row is printed, then last column is printed, then last row is printed and then first column is printed, then we will come inwards in the similar way.

For example: for the given matrix,



1	5	7	9	10	11
6	10	12	13	20	21
9	25	29	30	32	41
15	55	59	63	68	70
40	70	79	81	95	105

Spiral order is given by:
1 5 7 9 10 11 21 41 70 105 95 81 79 70 40 15 9 6 10 12 13 20 32 68 63 59 55 25 29 30 29.

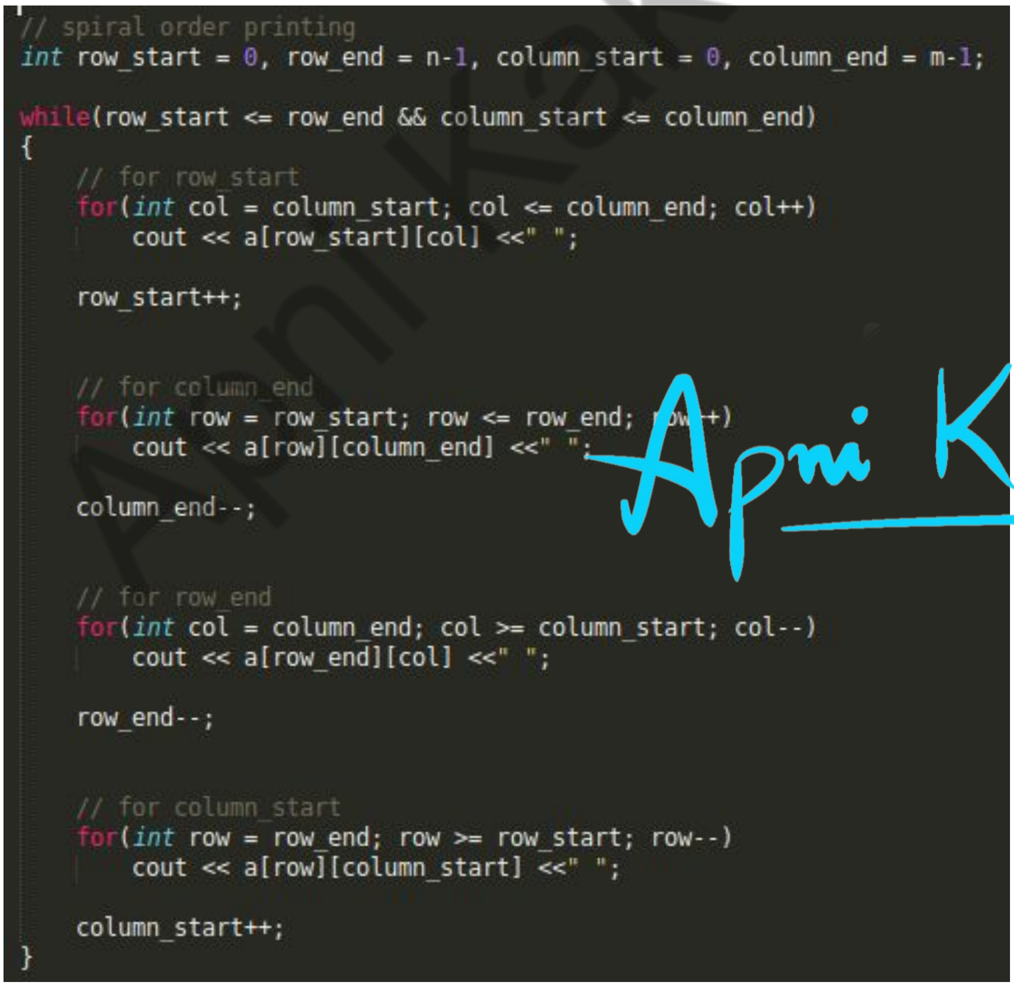
Algorithm: (We are given 2D matrix of n X m).

1. We will need 4 variables:
    a. *row_start* - initialized with 0.
    b. *row_end* - initialized with n-1.
    c. *column_start* - initialized with 0.

   d. *column_end* - initialized with m-1.

2. First of all, we will traverse in the row *row_start* from *column_start* to *column_end* and we will increase the *row_start* with 1 as we have traversed the starting row.
3. Then we will traverse in the column *column_end* from *row_start* to *row_end* and decrease the *column_end* by 1.
4. Then we will traverse in the row *row_end* from *column_end* to *column_start* and decrease the *row_end* by 1.
5. Then we will traverse in the column *column_start* from *row_end* to *row_start* and increase the *column_start* by 1.
6. We will do the above steps from 2 to 5 until *row_start* <= *row_end* **and** *column_start* <= *column_end*.

Code: Input the array first then perform the following code



```
// spiral order printing
int row_start = 0, row_end = n-1, column_start = 0, column_end = m-1;

while(row_start <= row_end && column_start <= column_end)
{
    // for row_start
    for(int col = column_start; col <= column_end; col++)
        cout << a[row_start][col] <<" ";

    row_start++;


    // for column_end
    for(int row = row_start; row <= row_end; row++)
        cout << a[row][column_end] <<" ";

    column_end--;


    // for row_end
    for(int col = column_end; col >= column_start; col--)
        cout << a[row_end][col] <<" ";

    row_end--;


    // for column_start
    for(int row = row_end; row >= row_start; row--)
        cout << a[row][column_start] <<" ";

    column_start++;
}
```