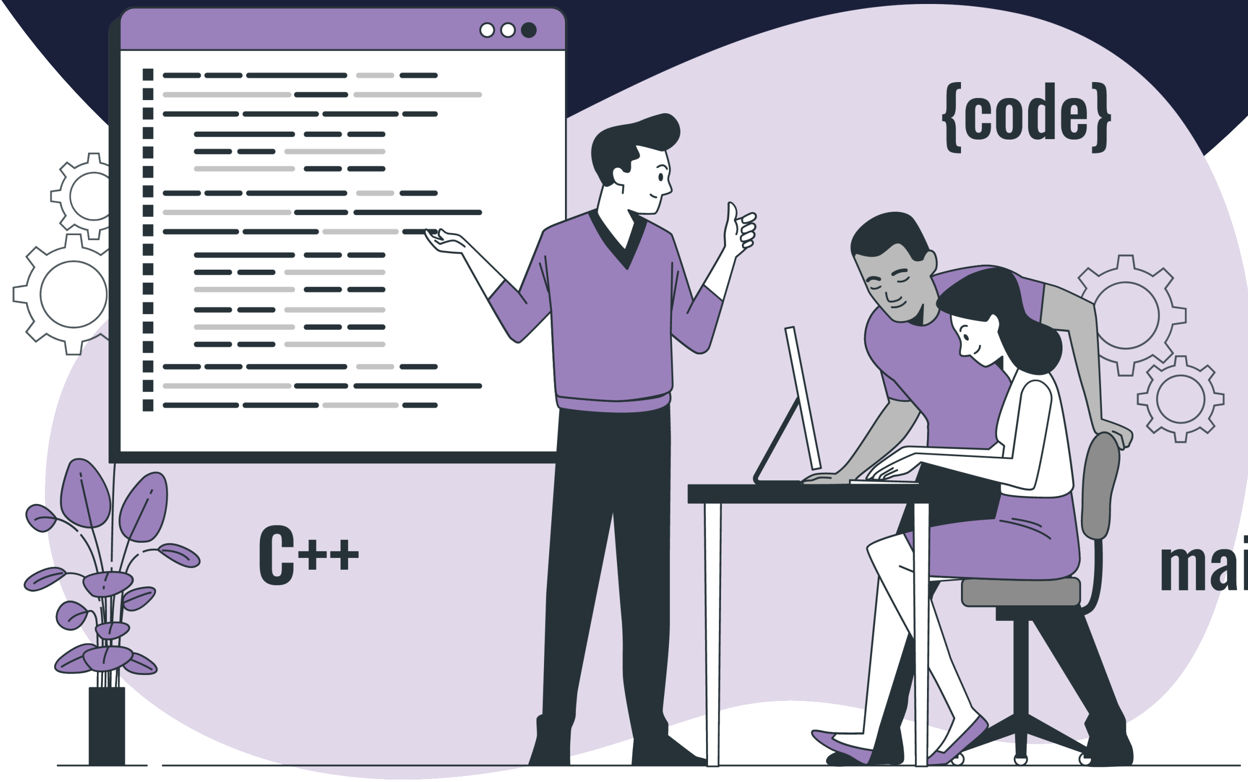
Lesson:

2D Array Problems Set 4



Pre-Requisites

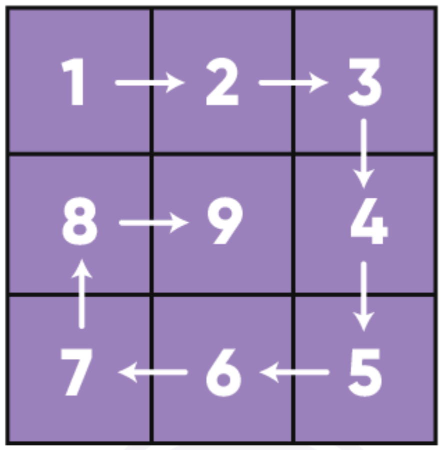
- 2D Arrays

List of Concepts Involved

- Problems based on spiral matrix.

Pattern: Spiral Matrix

Problems:
Q1. Given an nxm matrix ‘a’, return all elements of the matrix in spiral order.



Example 1:

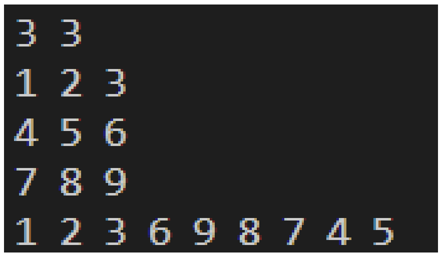
Input: a =[[1,2,3],
[4,5,6],
[7,8,9]]

Output: [1,2,3,6,9,8,7,4,5]

Code :

```cpp
#include<bits/stdc++.h>
using namespace std;


vector<int> spiralOrder(vector<vector<int>>& a) {
    int n = a.size(), m = a[0].size();
    int left = 0, right = m - 1, up = 0, down = n - 1;

    vector<int> ans;

    while (ans.size() < n * m) {
        for (int j = left; j <= right && ans.size() < n * m; j++) {
            ans.push_back(a[up][j]);
        }

        for (int i = up + 1; i <= down - 1 && ans.size() < n * m; i++) {
            ans.push_back(a[i][right]);
        }

        for (int j = right; j >= left && ans.size() < n * m; j--) {
            ans.push_back(a[down][j]);
        }

        for (int i = down - 1; i >= up + 1 && ans.size() < n * m; i--) {
            ans.push_back(a[i][left]);
        }

        left++; right--; up++; down--;
    }

    return ans;
}




int main() {
    int n, m;
    cin >> n >> m;
    std::vector<std::vector<int>> a(n, std::vector<int> (m, 0));

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

    std::vector<int> ans = spiralOrder(a);

    for (int i = 0; i < (int)ans.size(); i++) {
        cout << ans[i] << " ";
    }
    cout << '\n';

}
```

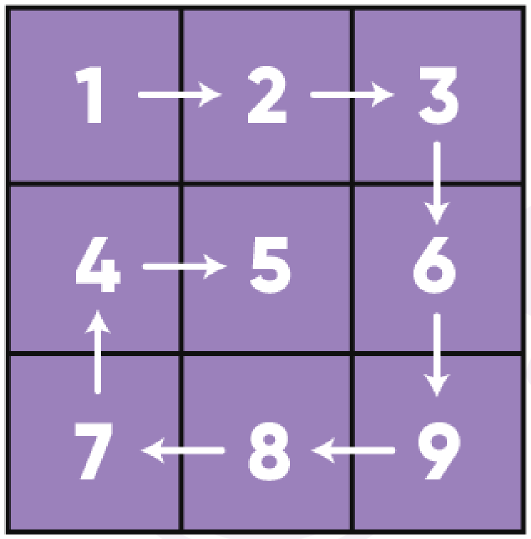
OUTPUT :



```
3 3
1 2 3
4 5 6
7 8 9
1 2 3 6 9 8 7 4 5
```

Explanation: Create four pointers for traversing in all the four directions. Then just traverse from left to right first and keep adding the elements in our answer. Do the same while traversing from top to bottom, then from right to left and from down to up. Finally increment the left and up pointer and decrement the right and down pointer as the size of the grid has been reduced. Make sure to keep a check on the answer that the maximum size of the answer can be when all the elements are present in it ie n x m.

Q2. Given a positive integer n, generate an n x n matrix filled with elements from 1 to n^2 in spiral order.



Example 1:

Input: n = 3

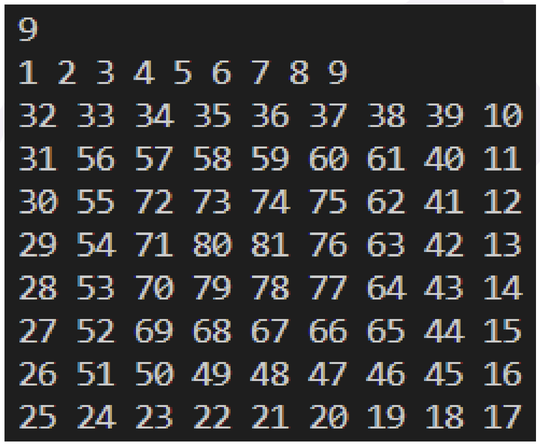
Output: [[1,2,3],
[8,9,4],
[7,6,5]]

Solution:

```
#include<bits/stdc++.h>
using namespace std;


vector<vector<int>> spiralMatrix(int n) {
    vector<vector<int>> a(n, vector<int> (n, -1));

    int val = 1;


    int left = 0, right = n - 1;
    int up = 0, down = n - 1;

    n = n * n;
    n++;

    while (val < n) {
        for (int j = left; j <= right && val < n; j++) {
            a[up][j] = val++;
        }

        for (int i = up + 1; i <= down - 1 && val < n; i++) {
            a[i][right] = val++;
        }


        for (int j = right; j >= left && val < n; j--) {
            a[down][j] = val++;
        }


        for (int i = down - 1; i >= up + 1 && val < n; i--) {
            a[i][left] = val++;
        }

        left++, right--, up++, down--;
    }


    return a;

}
```

```
int main() {
    int n;
    cin >> n;

    std::vector<std::vector<int>> a(n, std::vector<int> (n, 0));

    a = spiralMatrix(n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << a[i][j] << ' ';
        }
        cout << '\n';
    }

    cout << '\n';


}
```

OUTPUT :



```
9
1 2 3 4 5 6 7 8 9
32 33 34 35 36 37 38 39 10
31 56 57 58 59 60 61 40 11
30 55 72 73 74 75 62 41 12
29 54 71 80 81 76 63 42 13
28 53 70 79 78 77 64 43 14
27 52 69 68 67 66 65 44 15
26 51 50 49 48 47 46 45 16
25 24 23 22 21 20 19 18 17
```

Explanation: Create four pointers for traversing in all the four directions. For any input n,we need to add elements in matrix from 1 to n^2 so keep a value pointer 'val' that we will keep incrementing as we assign the values. Generate an empty matrix and traverse from left to right, then from up to down, then from right to left and finally from down to up. Finally increment the left and up pointer and decrement the right and down pointer as the size of the grid has been reduced. Make sure to keep a check on the 'val' that it should exceed n^2.

Upcoming Class Teasers

- 2D Array Problems