2D Matrix in C++

A **2D matrix** is a collection of data arranged in rows and columns. In C++, it is typically represented using a **2D array**.

1. Declaring a 2D Matrix

```
int rows = 3, cols = 3;

int matrix[rows][cols];
```

---

2. Initializing a 2D Matrix

```
int rows = 3, cols = 3;

int matrix[rows][cols] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

---

3. Taking Input

```
for (int i = 0; i < rows; ++i)

{

    for (int j = 0; j < cols; ++j)

    {

        cin >> matrix[i][j];

    }

}
```

---

4. Printing the Matrix

```
for (int i = 0; i < rows; ++i)
{
    for (int j = 0; j < cols; ++j)
    {
        cout << matrix[i][j] << " ";
    }
    cout << endl;
}
```

---

5. Accessing Rows

```
int rowIndex = 1; // 2nd row
for (int j = 0; j < cols; ++j)
    cout << matrix[rowIndex][j] << " ";
```

---

6. Accessing Columns

```
int colIndex = 2; // 3rd column
for (int i = 0; i < rows; ++i)
    cout << matrix[i][colIndex] << " ";
```

---

7. Accessing Diagonal Values

Types of Diagonals:

- **Primary Diagonal**: Top-left to bottom-right → `i == j`
- **Secondary Diagonal**: Top-right to bottom-left → `i + j == n - 1`

7.1. Primary Diagonal:

```
for (int i = 0; i < rows; i++)
{
    for (int j = 0; j < cols; j++)
    {
        if (i == j)
            cout << matrix[i][j] << " ";
    }
}
```

7.2. Secondary Diagonal:

```
for (int i = 0; i < rows; i++)
{
    for (int j = 0; j < cols; j++)
    {
        if (i + j == rows - 1)
            cout << matrix[i][j] << " ";
    }
}
```

---

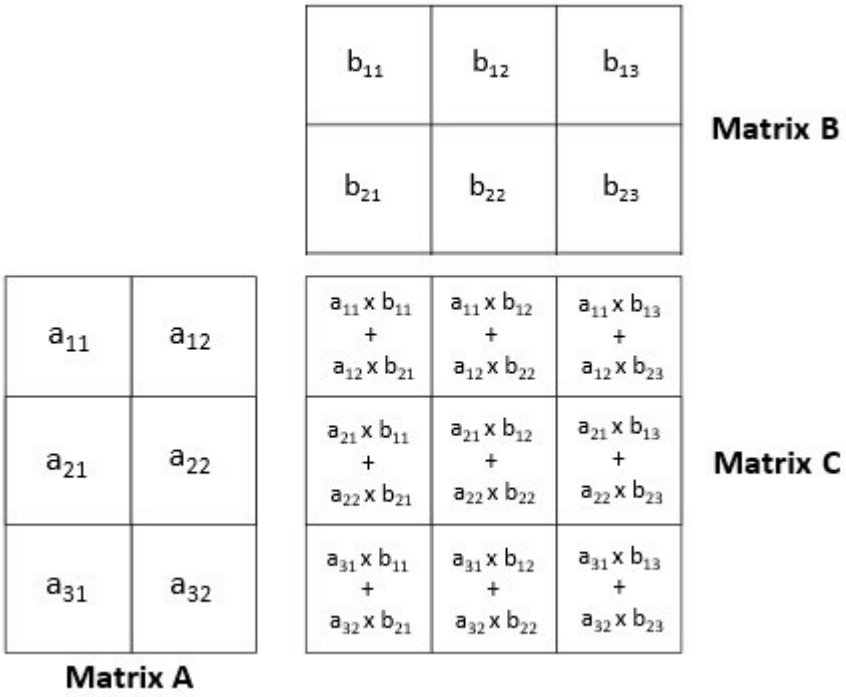
8. Matrix Summation (A + B)

```
for (int i = 0; i < n; ++i)

{

    for (int j = 0; j < n; ++j)

    {

        C[i][j] = A[i][j] + B[i][j];

    }

}
```

---

9. Matrix Multiplication (A * B)

To multiply matrices `A[m][n]` and `B[n][p]`:



```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int m, n, p;
    cin >> m >> n;
    int A[m][n];

    for (int i = 0; i < m; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            cin >> A[i][j];
        }
    }
    cin >> n >> p;
    int B[n][p];
    int C[m][p];
    memset(C, 0, sizeof(C));
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < p; ++j)
        {
            cin >> B[i][j];
        }
    }
    for (int i = 0; i < m; ++i)
    {
        for (int j = 0; j < p; ++j)
```

```
        {
            for (int k = 0; k < n; ++k)
            {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    for (int i = 0; i < m; ++i)
    {
        for (int j = 0; j < p; ++j)
        {
            cout << C[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

---

10. Solution of Search In Matrix

Problem: https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/S

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n, m;
    cin >> n >> m;
    int a[n][m];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cin >> a[i][j];
        }
    }
    int x;
    cin >> x;
    int found = false;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            if (a[i][j] == x)
            {
                found = true;
                break;
            }
        }
    }
    if (found)
        cout << "will not take number" << endl;
    else
        cout << "will take number" << endl;
    return 0;
}
```

---

11. Solution of Matrix

Problem: https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/T

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
```

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

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
                primary += a[i][j];
            if (i + j == n - 1)
                secondary += a[i][j];
        }
    }
    cout << abs(primary - secondary) << endl;
    return 0;
}
```

---

12. Solution of Mirror Array

Problem: https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/W
Practice: Try a different approach to swap columns

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n, m;
    cin >> n >> m;
    int a[n][m];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cin >> a[i][j];
        }
    }

    for (int i = 0; i < n; i++)
    {
        int x = 0, y = m - 1;
        while (x < y)
```

```
        {
            swap(a[i][x], a[i][y]);
            x++;
            y--;
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

---

13. Solution of 8 Neighbors

Problem: https://codeforces.com/group/MWSDmqGsZm/contest/219774/problem/X
Practice: Try with 8 if-else

```
#include <bits/stdc++.h>
using namespace std;
bool isValid(int i, int j, int n, int m)
{
    return i >= 0 && i < n && j >= 0 && j < m;
}
int dx[] = {0, 0, -1, 1, -1, 1, -1, 1};
int dy[] = {1, -1, 0, 0, 1, 1, -1, -1};

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

    int x, y;
    cin >> x >> y;
    x--;
    y--;
    int ans = 0;
    for (int i = 0; i < 8; i++)
```

```
    {
        int ni = x + dx[i];
        int nj = y + dy[i];
        if (!isValid(ni, nj, n, m))
        {
            ans++;
            continue;
        }
        if (a[ni][nj] == 'x')
            ans++;
    }
    if (ans == 8)
        cout << "yes" << endl;
    else
        cout << "no" << endl;
    return 0;
}
```