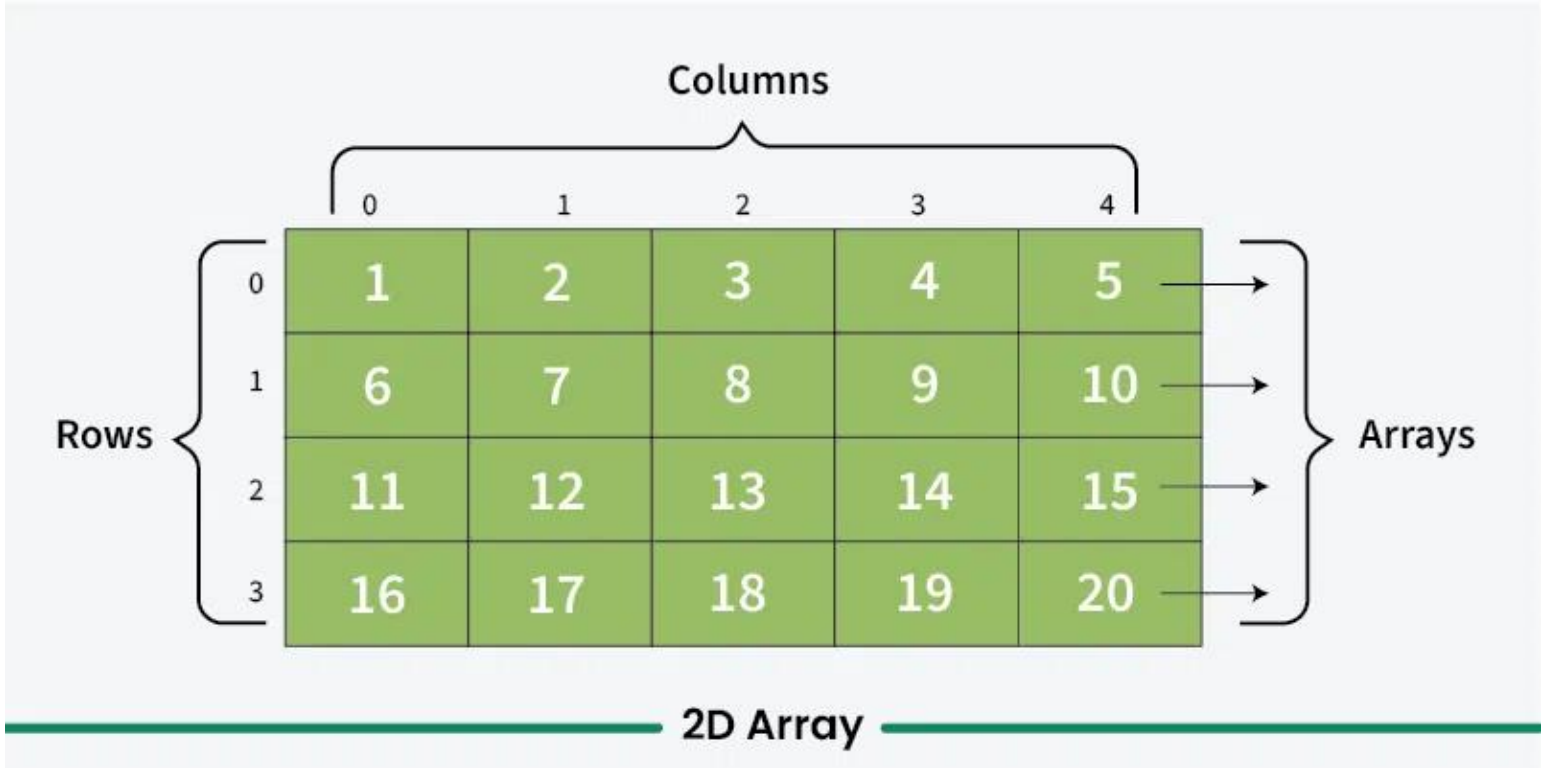
Fundamentals of Computer Programming

CS-110

Course Instructor: Dr. Momina Moetesum

Two-Dimensional Arrays

Week 10



Learning Objectives

01

To understand how to declare and initialize a two-dimensional array

02

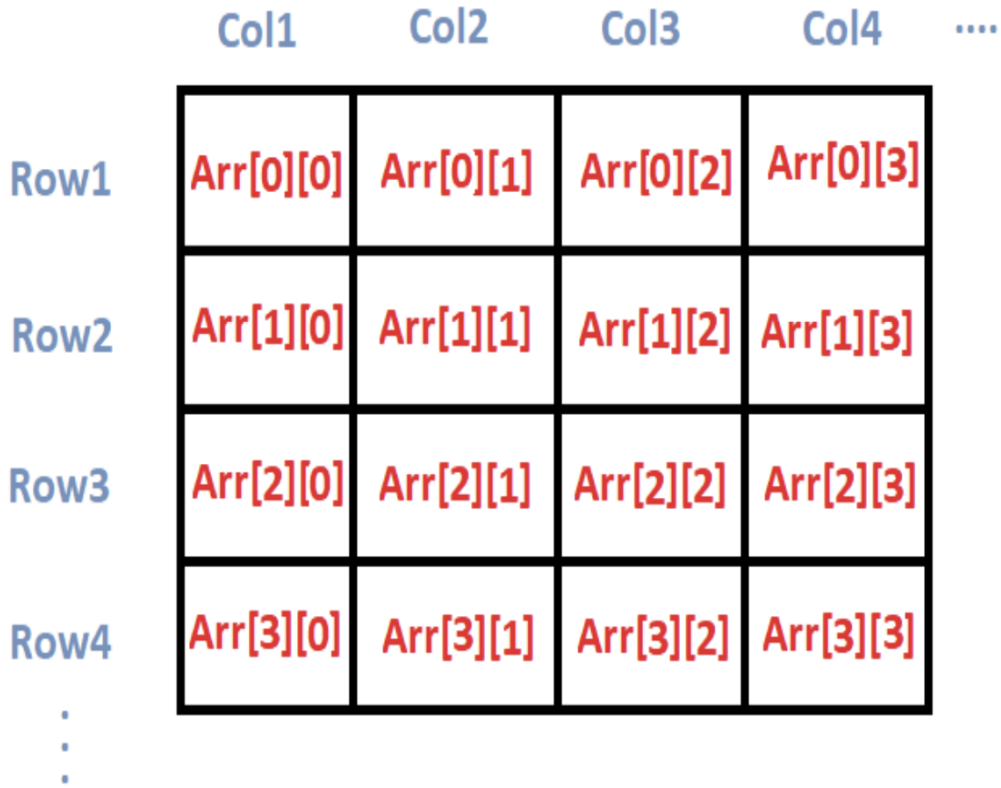
To understand how to access an array item from two-dimensional array

03

To practice how to pass a 2D array in a function

2-Dimensional Arrays

- A two-dimensional array in C++ is the simplest form of a multi-dimensional array.
- It can be visualized as an array of arrays.
- A two-dimensional array is also called a matrix. It can be of any type like integer, character, float, etc. depending on the initialization.



Example

```
Printing a 2D Array:
        10      11
        20      21
        30      31
        40      41
```

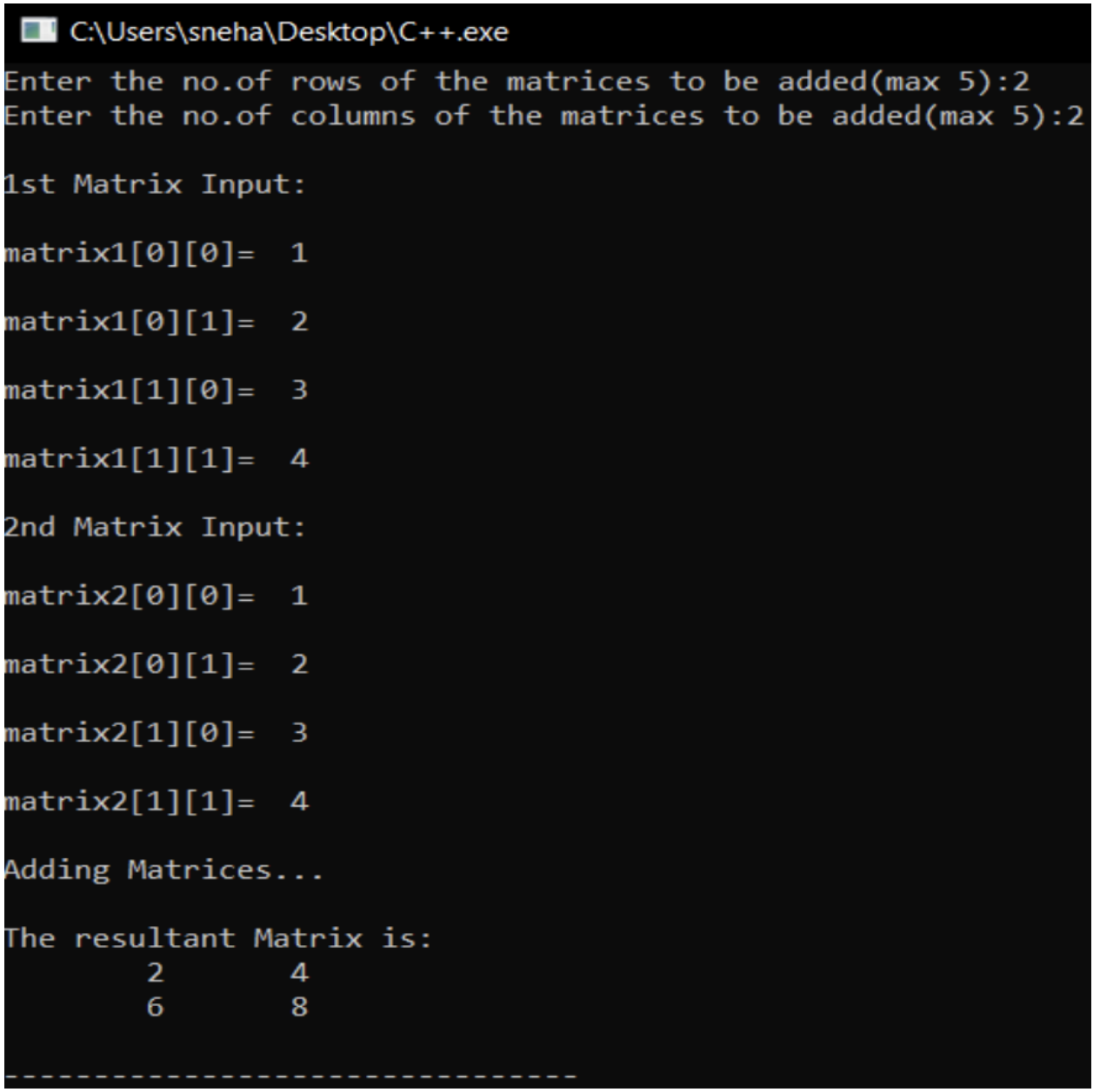
```
#include<iostream>
using namespace std;
main( )
{
        int arr[4][2] = {
                { 10, 11 },
                { 20, 21 },
                { 30, 31 },
                { 40, 41 }
                } ;

        int i,j;

        cout<<"Printing a 2D Array:\n";
        for(i=0;i<4;i++)
        {
                for(j=0;j<2;j++)
                {
                        cout<<"\t"<<arr[i][j];
                }
                cout<<endl;
        }
}
```

Matrix Addition using 2-D Arrays

- We take two matrices m1 and m2 with a maximum of 5 rows and 5 columns. And another matrix m3 in which we are going to store the result,
- As user inputs, we took the number of rows and columns for both the matrices. Since we are performing matrix addition, the number of rows and columns should be the same for both the matrices,
- After that, we take both the matrices as user inputs, again using nested for loops, • At this point, we have both the matrices m1 and m2,
- then we traverse through the m3 matrix, using two for loops and update the respective elements m3[i][j] by the value of m1[i][j]+m2[i][j]. In this way, by the end of the outer for loop, we get our desired matrix,
- At last, we print out the resultant matrix m3.

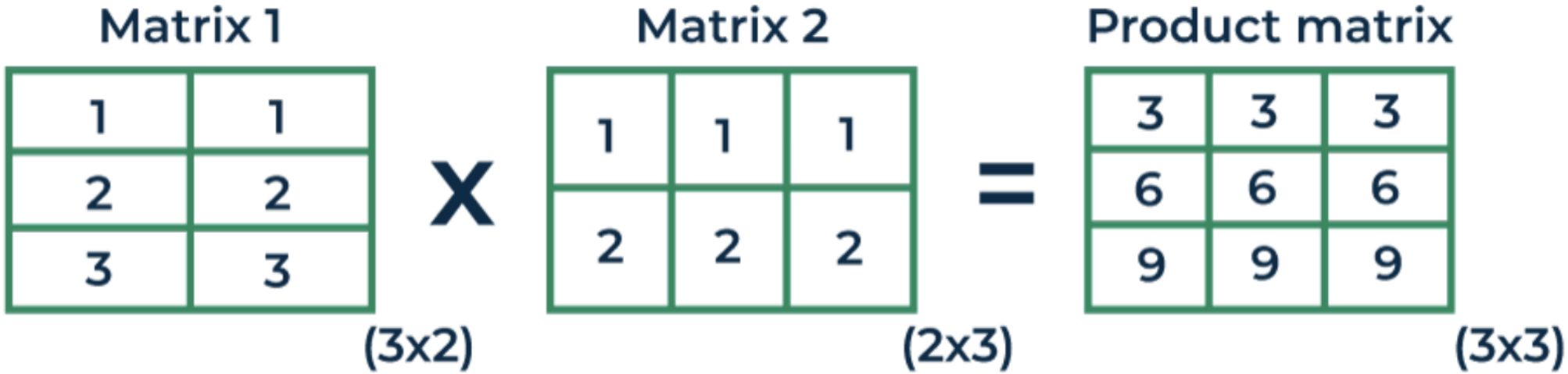
```
#include<iostream>
using namespace std;
main()
{
    int  m1[5][5], m2[5][5], m3[5][5];
    int  i, j, r, c;

    cout<<"Enter the no.of rows of the matrices to be added(max 5):";
    cin>>r;
    cout<<"Enter the no.of columns of the matrices to be added(max 5):";
    cin>>c;

    cout<<"\n1st Matrix Input:\n";
            for(i=0;i<r;i++)
            {
                    for(j=0;j<c;j++)
                    {
                            cout<<"\nmatrix1["<<i<<"]["<<j<<"]= ";
                            cin>>m1[i][j];
                    }
            }

            cout<<"\n2nd Matrix Input:\n";
            for(i=0;i<r;i++)
            {
                    for(j=0;j<c;j++)
                    {
                            cout<<"\nmatrix2["<<i<<"]["<<j<<"]= ";
                            cin>>m2[i][j];
                    }
            }

            cout<<"\nAdding Matrices...\n";

            for(i=0;i<r;i++)
            {
                    for(j=0;j<c;j++)
                    {
                            m3[i][j]=m1[i][j]+m2[i][j];
                    }
            }

            cout<<"\nThe resultant Matrix is:\n";

            for(i=0;i<r;i++)
            {
                    for(j=0;j<c;j++)
                    {
                            cout<<"\t"<<m3[i][j];
                    }
                    cout<<endl;
            }

}
```

Output


C:\Users\sneha\Desktop\C++.exe
Enter the no.of rows of the matrices to be added(max 5):2
Enter the no.of columns of the matrices to be added(max 5):2
1st Matrix Input:
matrix1[0][0]= 1
matrix1[0][1]= 2
matrix1[1][0]= 3
matrix1[1][1]= 4
2nd Matrix Input:
matrix2[0][0]= 1
matrix2[0][1]= 2
matrix2[1][0]= 3
matrix2[1][1]= 4
Adding Matrices...
The resultant Matrix is:
2 4
6 8

Matrix Multiplication

- We can multiply two matrices if the number of columns in the first matrix should be equal to the number of rows in the second matrix. The product matrix has the number of rows the same as the first matrix and the number of columns the same as the second matrix.



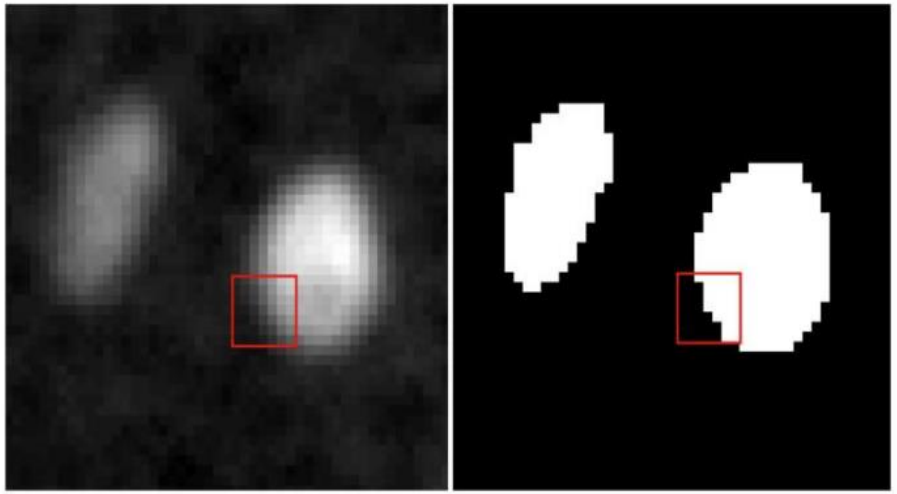
Example Code

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

// Edit MACROs here, according to your Matrix Dimensions for
// mat1[R1][C1] and mat2[R2][C2]
#define R1 2 // number of rows in Matrix-1
#define C1 2 // number of columns in Matrix-1
#define R2 2 // number of rows in Matrix-2
#define C2 3 // number of columns in Matrix-2

void mulMat(int mat1[][C1], int mat2[][C2])
{
    int rslt[R1][C2];

    cout << "Multiplication of given two matrices is:\n";

    for (int i = 0; i < R1; i++) {
        for (int j = 0; j < C2; j++) {
            rslt[i][j] = 0;

            for (int k = 0; k < R2; k++) {
                rslt[i][j] += mat1[i][k] * mat2[k][j];
            }

            cout << rslt[i][j] << "\t";
        }

        cout << endl;
    }
}
```

Example Code

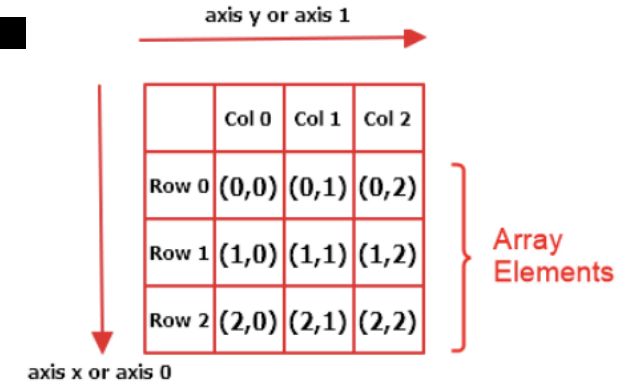
```
// Driver code
int main()
{
    // R1 = 4, C1 = 4 and R2 = 4, C2 = 4 (Update these
    // values in MACROs)
    int mat1[R1][C1] = { { 1, 1 }, { 2, 2 } };

    int mat2[R2][C2] = { { 1, 1, 1 }, { 2, 2, 2 } };

    if (C1 != R2) {
        cout << "The number of columns in Matrix-1 must "
                "be equal to the number of rows in "
                "Matrix-2"
             << endl;
        cout << "Please update MACROs according to your "
                "array dimension in #define section"
             << endl;

        exit(EXIT_FAILURE);
    }

    // Function call
    mulMat(mat1, mat2);

    return 0;
}
```

Image Manipulation using 2-D Arrays



61	66	75	86	96
59	64	73	83	92
59	64	73	83	92
56	60	68	79	88
54	58	64	72	81

0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1



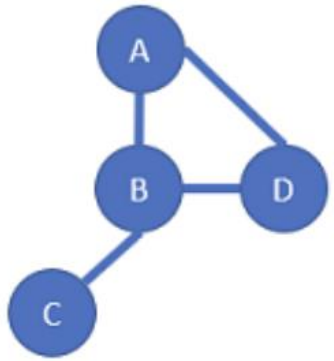
The most common way to convert a gray-level image into a binary image is to select a single threshold value(T).Then all the gray level values below T will be classified as black(0) i.e. background and those above T will be white(1) i.e. objects.

The Thresholding operation is a grey value remapping operation g defined by:

$$g(x,y)=\begin{cases}0 & \text{if } f(x,y) < T \\ 1 & \text{if } f(x,y) \geq T,\end{cases}$$

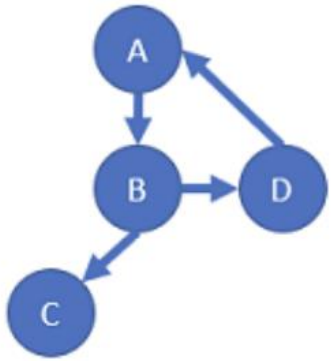
Where **(x,y)** represents a gray value/ are the coordinates of the threshold value p
T represent threshold value
g(x,y) represents threshold image
f(x,y) represents gray level image pixels/ input image

Adjacency Matrix Representation of Graphs



	A	B	C	D
A	0	1	0	1
B	1	0	1	1
C	0	1	0	0
D	1	1	0	0

Undirected



Ending Points

Starting Points

	A	B	C	D
A	0	1	0	0
B	0	0	1	1
C	0	0	0	0
D	1	0	0	0

Directed

Let A be the adjacency matrix of a graph G with vertices a, b, c, d . Find the degrees of the vertices of G.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Answer: The degrees of the vertices a, b, c and d of G are 3, 2, 2 and 1, respectively.

Acknowledgment

- Content of these slides are taken from:
  - https://www.geeksforgeeks.org/
  - https://www.tutorialspoint.com/
  - https://www.programiz.com/
  - https://www.w3schools.com/