University of Anbar
College of Computer Science and Information Technology
Department of Computer Networks

Two Dimensional Array

Lecture 2

Dr. Ali Al-Kubaisi

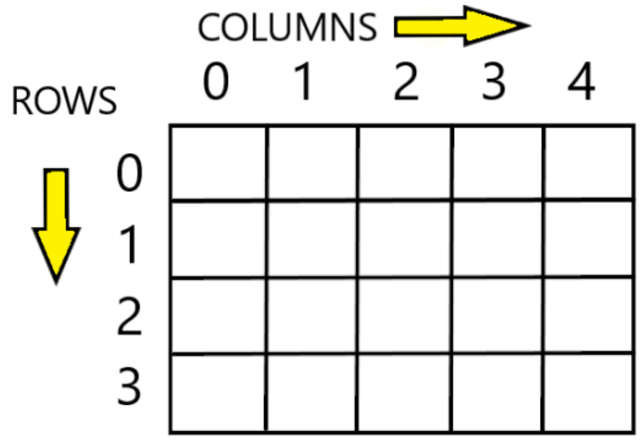
Asst.Lect. Abdulrahman Abbas Mukhlif

Outlines

- **C++ 2D Arrays**
- **Create 2D Array**
- **Initialize 2D Array**
- **Access 2D Array Elements**
- **Update 2D Array Elements**
- **Loop Through 2D Array**

2- Dimensional Arrays

- **2-D Array** can be defined as an array of arrays
- **2-D Array** is also called a **matrix**.
- **2-D array** in C++ is a collection of elements organized the form of rows and columns. It can be visualized as a table or a grid, where each element is accessed using two indices: one for the row and one for the column. Like a one-dimensional array, two-dimensional array indices also range from 0 to n-1 for both rows and columns.



Create 2D array

data_type array_name[row_size][column_size];

Example int **X**[3][3];

	Column 0	**Column 1**	**Column 2**
Row 0	X[0] [0]	X[0] [1]	X[0] [2]
Row 1	X[1] [0]	X[1] [1]	X[1] [2]
Row 2	X[2] [0]	X[2] [1]	X[2] [2]

Initialize 2D Array

```
int arr[2][4] = {{0, 1, 2, 3}, {4, 5, 6, 7}};
                        Or
int arr[2][4] = {0, 1, 2, 3, 4, 5, 6, 7};
```

0	**1**	**2**	**3**
4	**5**	**6**	**7**

The above array has 2 rows and 4 columns. The elements are filled in a way that the first 4 elements are filled in the first row and the next 4 elements are filled in the second row. The values will be initialized sequentially.

Access Elements of 2D Array

- Elements of a 2-dimensional array have to be accessed using row and column indices. It is similar to matrix element position, but the only difference is that here indexing starts from 0.

arr[i][j];

- where, **i** is the index of row and **j** is the index of the column. The range of indexes should be:

$$0 \le i \le (row\_size - 1)$$

$$0 \le j \le (col\_size - 1)$$

- Example

```
int arr[2][4] = {0, 1, 2, 3, 4, 5, 6, 7};
// Accessing 3rd element is 1st row
cout << arr[0][2] << endl;
// Accessing first element in 2nd row
cout << arr[1][0];
```

Update Elements of 2D Array

- he value at any index can be updated by using **= assignment operator.**

array_name[i][j] = new_value

```
int arr[2][4] = {0, 1, 2, 3, 4, 5, 6, 7};
// Updating 3rd element is 1st row
arr[0][2] = 22;
cout << arr[0][2] << endl;
// Updating first element in 2nd row
arr[1][0] = 99;
cout << arr[1][0];
```

Loop Through2D Array

- Two loops nested inside each other are needed to traverse a 2D array, one for moving though each dimension. First loop is used to move though the rows of 2D array, while other is used to move though columns in each row to access all the elements of the row.

```
int arr[2][4] = {0, 1, 2, 3, 4, 5, 6, 7};
// Outer loop to move through rows
for (int i = 0; i < 2; i++)
{
 // Inner loop to move though elements in each row
 for (int j = 0; j < 4; j++)
  {
   cout << arr[i][j] << " ";
  }
 cout << endl;
}
```