Multidimensional Arrays

- Arrays can be of two dimensions (i.e., subscripts) to represent tables of values consisting of information arranged in rows and columns.

- The intersection between a row and a column represents the location of arrays' elements.

- To identify a particular table element, we must specify two subscripts – by convention, the first identifies the element's row and the second identifies the element's column.

- Arrays that require two subscripts to identify a particular element are called two-dimensional arrays or 2-D arrays.

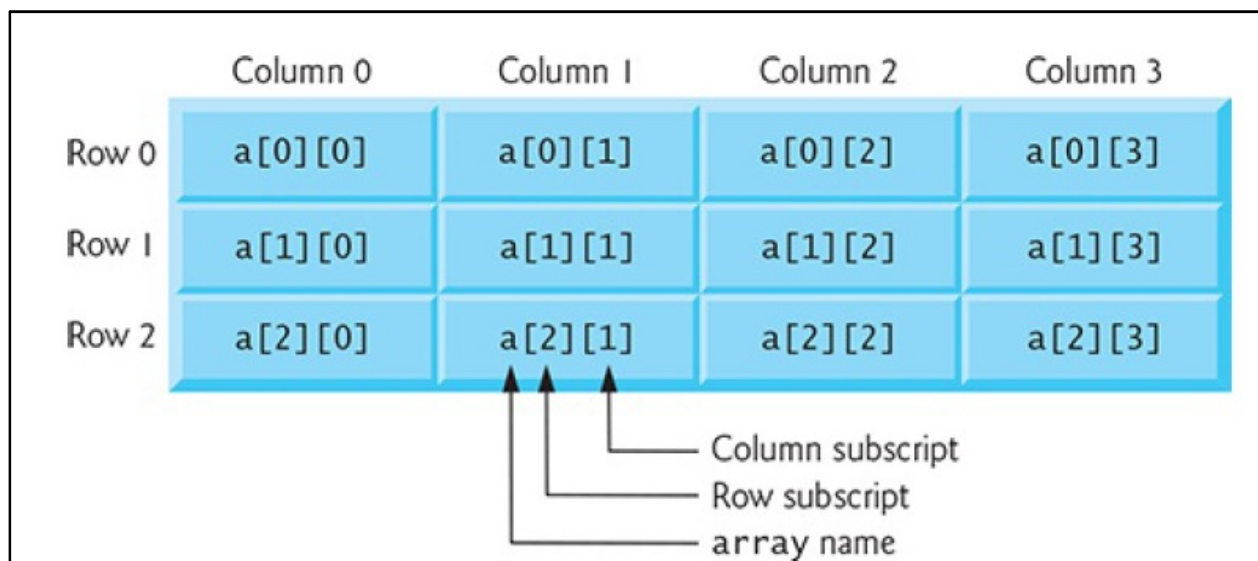- Figure 2 below illustrates a two-dimensional array.

Figure 2 Two-dimensional array with three rows and four columns.

- The array contains three rows and four columns, so it's said to be a 3-by-4 array.

- In general, an array with **m** rows and **n** columns is called an *m-by-n* array.

- To access the element at the second row and third column (if the array name is a), then we can access it as a[1, 2].

- That is due to the first row's index is 0 and the first column's index is 0 also.

- Note: Sometimes, an array may contain equal number of rows and columns, it is then called a matrix and expressed as ***n-by-n*** array.

- For example, a matrix **CH** that contains 8 rows and 8 columns can be imagined as the chess board.

- Matrices usually have two diagonals (main and secondary).

- The main diagonal represents the elements from [0, 0] to [n-1, n-1], while the secondary diagonal represents the elements from [0, n-1] to [n-1, 0].

M				S
	M		S	
		M,S		
	S		M	
S				M

- The table above shows the main diagonal elements marked with M, and the secondary diagonal elements marked with S.

Declaring Two Dimensional Arrays

```
Type[ , ] array_name = new Type[rows, cols];
```

Example:

```
int[ , ] marks = new int[25,7]; //marks is a
2Darray of 25 rows and 7 columns of int values
```

Declaring a 2D Array and Using a Loop to Read and Print the Array's Elements

- As with one dimensional arrays, two dimensional arrays' elements can be entered from keyboard using **Console.ReadLine()** statement like any other primary data type variables but using two loops.

Example:

```
static void Main(string[] args)
 {
int[,] marks = new int [25,7];  //marks is a 2D
array of 25 rows and 7 columns of int values
//reading the array from the KB
Console.WriteLine("Reading the array");
for (int i = 0; i < 3; i++)
{
 for (int j = 0; j <3; j++)
```

```csharp
    {
marks[i,j]= Convert.ToInt32(Console.ReadLine());
  }//for j
}//for i


//Printing the array as a table
  Console.WriteLine("Printing the array");
for (int i = 0; i < 3; i++)
 {
  for (int j = 0; j < 3; j++)
   {
     Console.Write(marks[i,j]+"\t");
   }//for
 Console.WriteLine();
 }//for i
}//Main
```

- Note that we used two loops each time we deal with the array, one (the outer) for the rows and the other (the inner) for columns. The output of the above program is shown above.

H.W: Write C# programs for each of the following:

1- Find and print the summation of the **main** diagonal of a matrix of size 10x10 of type integer.

2- Find and print the summation of the **secondary** diagonal of a matrix of size 10x10 of type float.

3- Find and print the summation of each **column** of 7x5 array of type integer.

4- Find and print the summation of the elements that are **above** the **main** diagonal in a matrix of size 10x10 of type integer.

5- Find and print the summation of the elements that are **under** the **main** diagonal in a matrix of size 10x10 of type integer.

6- Find and print the summation of the elements that are **above** the **secondary** diagonal in a matrix of size 10x10 of type float.

7- Find and print the summation of the elements that are **under** the **secondary** diagonal in a matrix of size 10x10 of type float.

8- Find and print the summation of each odd column in a 10x10 array of type integer.

9- Find and print the summation of each even row in a 10x10 array of type float.

10- Read an integer array of size MxN and then switch columns 3 and 5 with each other.