C# allows multidimensional arrays. Multi-dimensional arrays are also called rectangular array. You can declare a 2-dimensional array of strings as:

```csharp
string[,] names;
```
or, a 3-dimensional array of int variables as:

```csharp
int[,,] m;
```

### Two-Dimensional Arrays

The simplest form of the multidimensional array is the 2-dimensional array. A 2-dimensional array is a list of one-dimensional arrays.

A 2-dimensional array can be thought of as a table, which has $x$ number of rows and $y$ number of columns. Following is a 2-dimensional array, which contains 3 rows and 4 columns:

<table>
<thead>
<tr>
<th>Column 0</th>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a[0][0]$</td>
<td>$a[0][1]$</td>
<td>$a[0][2]$</td>
<td>$a[0][3]$</td>
</tr>
<tr>
<td>$a[1][0]$</td>
<td>$a[1][1]$</td>
<td>$a[1][2]$</td>
<td>$a[1][3]$</td>
</tr>
</tbody>
</table>

Thus, every element in the array $a$ is identified by an element name of the form $a[i,j]$, where $a$ is the name of the array, and $i$ and $j$ are the subscripts that uniquely identify each element in array $a$.

### Initializing Two-Dimensional Arrays

Multidimensional arrays may be initialized by specifying bracketed values for each row. The following array is with 3 rows and each row has 4 columns.

```csharp
int[,] a = new int[3, 4] {
    {0, 1, 2, 3},  /* initializers for row indexed by 0 */
    {4, 5, 6, 7},  /* initializers for row indexed by 1 */
    {8, 9, 10, 11} /* initializers for row indexed by 2 */
};
```

### Accessing Two-Dimensional Array Elements

An element in 2-dimensional array is accessed by using the subscripts. That is, row index and column index of the array. For example,

```csharp
int val = a[2, 3];
```

The above statement takes 4th element from the 3rd row of the array. You can verify it in the above diagram. Let us check the program to handle a two-dimensional array:

```csharp
using System;
namespace ArrayApplication
{
    class MyArray
    {
        static void Main(string[] args)
        {
            /* an array with 5 rows and 2 columns*/
```
int[,] a = new int[5, 2] {{0,0}, {1,2}, {2,4}, {3,6}, {4,8}};
int i, j;

/* output each array element's value */
for (i = 0; i < 5; i++)
{
    for (j = 0; j < 2; j++)
    {
        Console.WriteLine("a[\{\{0\},\{1\}\}] = \{2\}", i, j, a[i,j]);
    }
}

Console.ReadKey();

When the above code is compiled and executed, it produces the following result:

a[0,0]: 0
a[0,1]: 0
a[1,0]: 1
a[1,1]: 2
a[2,0]: 2
a[2,1]: 4
a[3,0]: 3
a[3,1]: 6
a[4,0]: 4
a[4,1]: 8