



Topics Covered

- Multi-Dimensional Data
- C# Multi-dimensional Arrays
 - Initialization
 - > Rows / Columns
 - Properties

- C# Jagged Arrays
- Accessing Elements

Multi-Dimensional Data

Linear arrays cover a lot of territory, but there are occasions when we need to represent data in multiple dimensions. For example:

- Digital Images (2-dimensional)
- Stereo Audio (2-dimensional)
- Surround Sound (N-dimensional)
- Board Games (2-dimensional)

C# supports multi-dimensional arrays and jagged arrays.



Multi-Dimensional Arrays

Two-dimensional arrays are declared using the syntax:

datatype[,]

Additional commas between the square brackets create additional dimensions. For example:

```
int[,] chessBoard = new int[8,8];
double[,,] cube = new double[12, 12, 12];
```



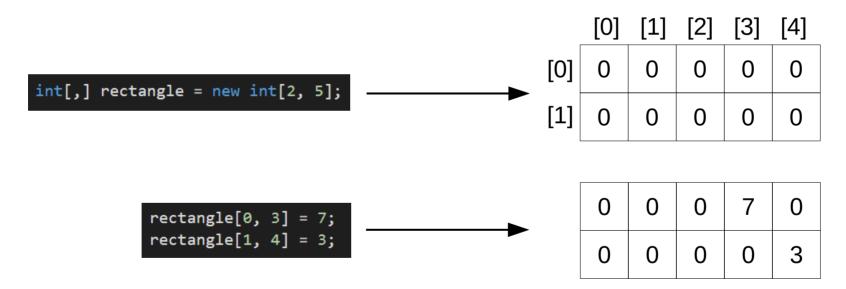
Initializing 2D Arrays

As with linear arrays, multi-dimensional arrays can be initialized by providing a braced structure corresponding to the array elements:



Rows and Columns

For a 2D array, the 1st index is the row index and the 2nd index is the column index.





Array Properties

Both Length and LongLength return the total # elements (10).

This is a 2-dimensional array, so its Rank is 2.

```
Console.WriteLine(rectangle.Length); → 10
Console.WriteLine(rectangle.LongLength); → 10
Console.WriteLine(rectangle.Rank); → 2
```



Jagged Arrays

A jagged array is an array of arrays. Each element of the array can be a unique length, thus the "jagged" moniker. Here are 3 examples of creating jagged arrays:

```
// Create an array of 8 int[].
// Each element is initially null:
int[][] iarrs = new int[8][];
```

```
Console.WriteLine(iarrs.Length);

Console.WriteLine(iarrs.LongLength);

Console.WriteLine(iarrs.Rank);

Console.WriteLine(iarrs2.Length);

Console.WriteLine(iarrs2.LongLength);

Console.WriteLine(iarrs2.Rank);
```

Each of these arrays has a Rank of 1 - they are technically 1-dimensional.

```
// Create an array of 3 int[].
// Each element is initialized.
int[][] iarrs2 = new int[][]
{
    new int[] {1, 2, 3, 4},
    new int[] {5, 6, 7, 8, 9},
    new int[] {10, 11, 12, 13, 14, 15}
};
```

```
// Short-hand for the same:
int[][] iarrs3 =
{
    new int[] {1, 2, 3, 4},
    new int[] {5, 6, 7, 8, 9},
    new int[] {10, 11, 12, 13, 14, 15}
};
```

Accessing Jagged Array Elements

Here are two loops that each calculates the sum of all the elements in the jagged array *iarrs2* from the previous example.

The first calculation uses for loops and indexing.

The second calculation uses foreach loops.

In the upper loop using indexing, we also have the option of assigning new values to the elements of *iarrs2*.

```
int grandTotal = 0;
for (int i = 0; i < iarrs2.Length; ++i)</pre>
  for (int j = 0; j < iarrs2[i].Length; ++j)</pre>
    grandTotal += iarrs2[i][j];
Console.WriteLine(grandTotal); // 120
grandTotal = 0;
foreach (int[] arr in iarrs2)
  foreach (int i in arr) grandTotal += i;
Console.WriteLine(grandTotal); // 120
```





Topics Covered

- Multi-Dimensional Data
- C# Multi-dimensional Arrays
 - Initialization
 - > Rows / Columns
 - Properties

- C# Jagged Arrays
- Accessing Elements

Exercises

