




Two dimensional arrays



Declaration

You can create two-dimensional arrays as well, using the [,] symbol with the type:


string[,] grid2d = new string[2,2];

 2nd index / 1st index →	0	1
0	null	null
1	null	null

Setting values


To set values, you need to provide two indices separated by comma now [x,y]:

```
grid2d[0,0] = "topLeft";
```

 2nd index / 1st index →	>0<	1
>0<	topLeft	null
1	null	null


Setting values

`grid2d[0,1] = "bottomLeft";`

 2nd index / 1st index →	>0<	1
0	topLeft	null
>1<	bottomLeft	null


Setting values

`grid2d[1,0] = "topRight";`

 2nd index / 1st index →	0	>1<
>0<	topLeft	topRight
1	bottomLeft	null

Setting values

```
grid2d[1,1] = "bottomRight";
```

 2nd index / 1st index →	0	>1<
0	topLeft	topRight
>1<	bottomLeft	bottomRight

One-Dimensional 2D-Array?

- Alternatively, you can use a one-dimensional array and a helper index to get the right cell.
- Two-dimensional arrays do the same internally.
- No advantage in doing this yourself.
- But UnityEngine for example can serialize One-Dimensional Arrays, but not Two-Dimensional ones.

```
string[ ] grid1d = new string[4];
```

```
// You have to create this yourself to handle indexing correctly
static int GetIndex(int x, int y, int width) {
    return y * width + x;
}
```

Index	0	1	2	3
Value	null	null	null	null

`grid1d[GetIndex(0, 0, 2)] = "bottomLeft"; // $0 * 2 + 0 = 0$`

Index	0	1	2	3
Value	bottomLeft	null	null	null

```
grid1d[GetIndex(0, 1, 2)] = "topLeft"; // 1 * 2 + 0 = 2
```

Index	0	1	2	3
Value	bottomLeft	null	topLeft	null

```
grid1d[GetIndex(1, 0, 2)] = "bottomRight"; //  $0 * 2 + 1 = 1$ 
```

Index	0	1	2	3
Value	bottomLeft	bottomRight	topLeft	null

```
grid1d[GetIndex(1, 1, 2)] = "topRight"; // 1 * 2 + 1 = 3
```

Index	0	1	2	3
Value	bottomLeft	bottomRight	topLeft	topRight

Jagged arrays

Another solution is to use an array of arrays.

It has worse performance though!

The advantage is: its rows can have different size.

Not recommended! (Unless you want to have rows of different sizes)

```
string[ ][ ] gridJagged = new string[2][ ];
```

Index	0	1
Value	null	null

```
for (var i = 0; i < gridJagged.Length; i++) {  
    gridJagged[i] = new string[2];  
}
```

Index	0					1				
Value		Index	0	1			Index	0	1	
		Value	null	null			Value	null	null	



`gridJagged[0][1] = "bLft"; // bottom left array`

Index	>0<					1				
Value		Index	0	>1<			Index	0	1	
		Value	null	"bLft"			Value	null	null	

Goal - 2D canvas

- Create a two-dimensional array as the canvas. We want to be able to store letters in that array. What type do you need? An array of size 5x5 should be enough for this demo.
- Repeatedly:
 - Print all the contents of the two-dimensional array to the user by printing the complete array to the user. But also print the coordinates, so users understand which is the x and y coordinate and what values they have.
 - Ask the user for a x coordinate
 - Ask the user for a y coordinate
 - Ask the user for a symbol
- Store the symbol at the correct index of the two dimensional array

Note: If you want to print a `\`, you need to escape the character: `\\`.