Why we have both jagged array and multidimensional array?

Asked 8 years, 9 months ago Active yesterday Viewed 36k times

68

19

1. What is the difference between jagged array and Multidimensional array. Is there a benefit of one on another?
2. And why would the Visual Studio not allow me to do a

```
MyClass[][] abc = new MyClass[10][20];
```

(We used to do that in C++, but in C# it underlines [20] with red wriggly line.. Says invalid rank specifier)

but is happy with

```
MyClass[,] abc = new MyClass[10,20];
```

3. Finally how can I initialize this in a single line (like we do in simple array with `{new xxx...}{new xxx....}`)

```
MyClass[][,][,] itemscollection;
```

c# .net arrays visual-studio-2010

edited Jul 29 '17 at 8:45

Cœur
22.9k 10 130 188

asked Jan 10 '11 at 16:20

Shekhar_Pro
14.9k 6 45 77

10 The whole point of a jagged array is that the "nested" arrays needn't be of uniform size. – Ani Jan 10 '11 at 16:22

1 msdn.microsoft.com/en-us/library/2yd9wwz4(v=vs.71).aspx - Multidimensional array syntax as [X,Y] is valid according to docs – ndtreviv Jan 10 '11 at 16:24

possible duplicate of What is differences between Multidimensional array and Array of Arrays in C#? – nawfal Apr 25 '13 at 12:34

10 Answers

81

1. A jagged array is an array-of-arrays, so an `int[][]` is an array of `int[]`, each of which can be of different lengths and occupy their own block in memory. A multidimensional array (`int[,]`) is a single block of memory (essentially a matrix).
2. You can't create a `MyClass[10][20]` because each sub-array has to be initialized separately, as they are separate objects:

```
MyClass[][] abc = new MyClass[10][];

for (int i=0; i<abc.Length; i++) {
    abc[i] = new MyClass[20];
}
```

   A `MyClass[10,20]` is ok, because it is initializing a single object as a matrix with 10 rows and 20 columns.
3. A `MyClass[][,][,]` can be initialized like so (not compile tested though):

```
MyClass[][,][,] abc = new MyClass[10][,][,];

for (int i=0; i<abc.Length; i++) {
    abc[i] = new MyClass[20,30][,];

    for (int j=0; j<abc[i].GetLength(0); j++) {
        for (int k=0; k<abc[i].GetLength(1); k++) {
            abc[i][j,k] = new MyClass[40,50];
        }
    }
}
```

Bear in mind, that the CLR is heavily optimized for single-dimension array access, so using a jagged array will likely be faster than a multidimensional array of the same size.

edited yesterday

Viktor Borítás
29 5

answered Jan 10 '11 at 16:25

thecoop
36.5k 11 109 169

6 @GreyCloud - stackoverflow.com/questions/597720/… – thecoop Jan 10 '11 at 16:46

Is there a (common) use case for the multi-dimensional array? – ryanwebjackson Oct 30 '17 at 16:01

Examples: checkered board `var board = new Piece[8, 8];` , a transformation matrix `var m = new double[2, 2];` . – Olivier Jacot-Descombes Apr 28 at 15:53

33

A jagged array is an array of arrays. Each array is not guaranteed to be of the same size. You could have

```
int[][] jaggedArray = new int[5][];
jaggedArray[0] = { 1, 2, 3 }; // 3 item array
jaggedArray[1] = new int[10]; // 10 item array
// etc.
```

It's a *set* of related arrays.

A multidimensional array, on the other hand, is more of a cohesive grouping, like a box, table, cube, etc., where there are no irregular lengths. That is to say

```
int i = array[1,10];
int j = array[2,10]; // 10 will be available at 2 if available at 1
```

answered Jan 10 '11 at 16:24

Anthony Pegram
104k 23 196 236

2 Straight to the point. Great answer. +1 – wassimans Jul 2 '11 at 15:25

I tried your code. It did not compile. Try adding int[3] so Try `jaggedArray[0] = int[3]{ 1, 2, 3 } ;` – barlop Feb 22 '16 at 9:51

I know this is old, but just for informational purposed int[3] is not neccessary. a simple int[] is all that matters. int[][] myArray = new int[5][]; myArray[0] = new int[] {1, 2, 3, 4}; This is all that is neccessary. – Velocibadgery Feb 16 '18 at 23:37

A rectangular array always has the same amount of columns for every row

Every row has 30 columns, whereas in a jagged array, this is not required. Therefore, I think you'll have to initialize every 'row' in a jagged array separately:

```
MyClass[][] x = new MyClass[10][];

for(int i = 0; i < 10; i++)
{
    x[i] = new MyClass[30];
}
```

In fact, this means that not every row in the jagged array must contain the same number of elements. (In my example, it does have the same number of elements, but this is not required).

You can perfectly do this, for instance:

```
MyClass[][] x = new MyClass[10][];

for(int i = 0; i < 10; i++)
{
    x[i] = new MyClass[(30 + i)];
}
```

This might be an interesting article for you.

edited Dec 22 '15 at 13:08

PiotrWolkowski
6,362 5 32 51

answered Jan 10 '11 at 16:26

Frederik Gheysels
50.1k 8 87 143

4

Ad 3) To initialize such a monster like `[][,][,]`, you can do sth like:

```
int [,][,] multiArr1 = { { new int[,] { { 2, 2 }, { 1, 1 } },
                           new int[,] { { 2, 2 }, { 1, 1 } } },
                           { new int[,] { { 2, 2 }, { 1, 1 } },
```

```
        new int[,] { { 2, 2 }, { 1, 1 } } } };

int [][,][,] superMultiArray = { multiArr1, multiArr2 };
```

answered Jan 10 '11 at 16:42

nan
9,199 6 35 62

1

If you're looking for a multi-dimensional array that has set bounds, always use the `[,]` style syntax. This will make sure that each portion is equally sized.

When you use `[][]` what is really going is that you're creating an array of arrays. This then means that each array can be sized differently. For example:

```
int[][] jaggedArray = new int[5][]
for(int index = 0; index < jaggedArray.Length ; ++index)
{
    jaggedArray[index] = new int[index + 1];
}
```

answered Jan 10 '11 at 16:26

Joshua Rodgers
4,692 2 25 29

1

The inline declaration would look something like this:

```
int[,] numbers = { {1, 2}, {3, 4}, {5, 6} };
```

answered Jan 10 '11 at 16:27

Josiah Ruddell
26.4k 7 56 63

1

For jagged or multidimensional inline arrays, see this programming guide:

```
// Three-dimensional array.
int[, ,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },
{ { 7, 8, 9 }, { 10, 11, 12 } } };

// Same array with dimensions specified at declaration.
int[, ,] array3Da = new int[2, 2, 3] { { { 1, 2, 3 }, { 4, 5, 6 } },
{ { 7, 8, 9 }, { 10, 11, 12 } } };
```

You don't have to specify the dimensions (array3D), but if you know they're never going to change, it's helpful to know what dimensions you're using (array3Da).

edited yesterday

Viktor Borítás
29 5

answered Jan 10 '11 at 16:30

rownage
2,132 3 19 29

0

You would need to understand the internal working of the array the multi-dimensional array act as a single dimension array except that the double indexing is converted into a single one.

Your Jagged array in c# is an array of objects which are in turns arrays.

answered Jan 10 '11 at 16:32

dvhh
4,064 23 30

0

I think that 2d jagged arrays memory allocation in C# is like 2d arrays in C++ and C. Because 2d jagged arrays have pointer which points to array of pointers that each of this pointers points to an array of elements (for example integer elements); like this code in C++,

```
int** 2DArr {new int* [number1]};
for (int i = 0; i < number1; i++)
{
   2DArr[i] = new int[number2];
```

the memory allocation of code below is the same as 2d jagged arrays in C#? But I am doubtful about , could you please explain more if I think in wrong way.

edited Feb 1 '18 at 10:00

answered Feb 1 '18 at 9:28

A.salehy
13 5

0

This post is old but here are my thoughts on that.

Jagged arrays are multidimensional arrays. Multidimensional arrays come in two varieties: rectangular and jagged. Rectangular arrays represent an n-dimensional block of memory, and jagged arrays are arrays of arrays.

Rectangular arrays

Rectangular arrays are declared using commas to separate each dimension. The following statement declares a rectangular two-dimensional array, where the dimensions are 3 × 3:

```
int[,] matrix = new int [3, 3];
```

Jagged arrays

Jagged arrays are declared using successive square brackets to represent each dimension. Here is an example of declaring a jagged two-dimensional array, where the outermost dimension is 3:

```
int[][] matrix = new int[3][];
```

answered Aug 29 at 17:45

Imir Hoxha
945 5 25 55
