

<http://csharpfundamentals.telerik.com>

Telerik Academy

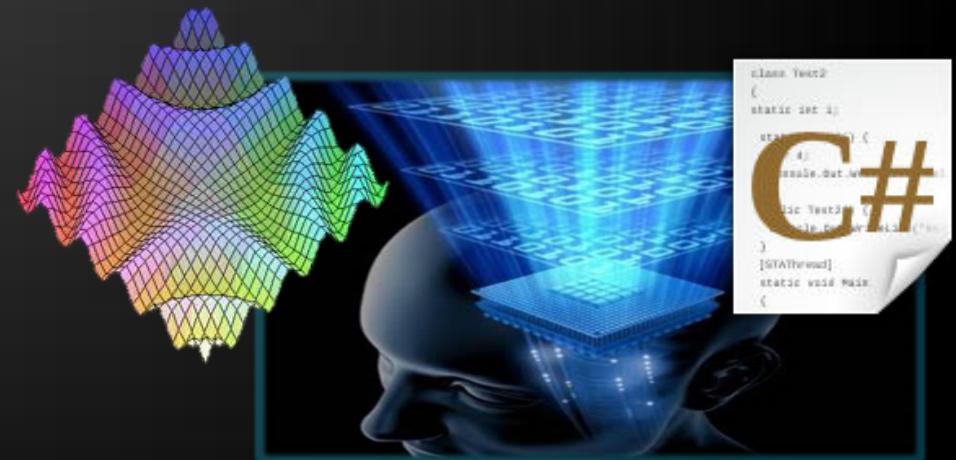


C# FUNDAMENTALS

Multidimensional Arrays

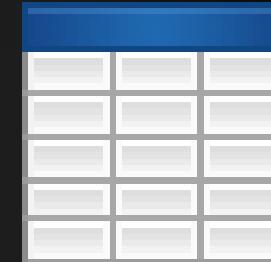
Processing Matrices and Multidimensional Tables

Telerik Software Academy
Learning & Development Team
<http://academy.telerik.com>



1. Matrices and Multidimensional Arrays

- ◆ Declaring
- ◆ Usage



2. Jagged Arrays

- ◆ Declaring
- ◆ Usage

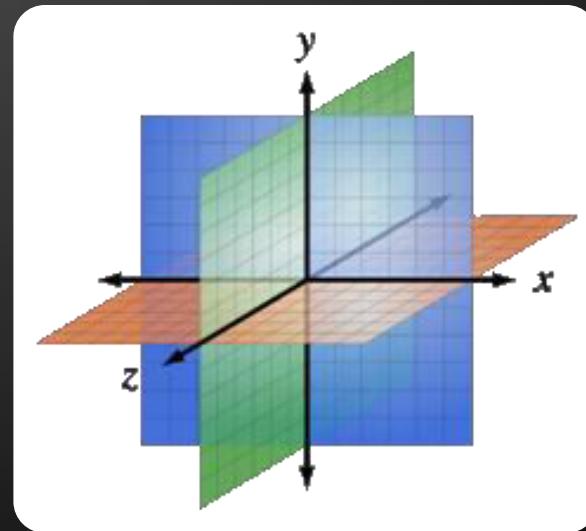


3. The Array Class

- ◆ Sorting
- ◆ Binary Search

Multidimensional Arrays

Using Array of Arrays, Matrices and Cubes



What is Multidimensional Array?

- ◆ Multidimensional arrays have more than one dimension (2, 3, ...)
 - ◆ The most important multidimensional arrays are the 2-dimensional
 - ◆ Known as matrices or tables
- ◆ Example of matrix of integers with 2 rows and 4 columns:

	0	1	2	3
0	5	0	-2	4
1	5	6	7	8

Declaring and Creating Multidimensional Arrays

- ◆ Declaring multidimensional arrays:

```
int[,] intMatrix;  
float[,] floatMatrix;  
string[,,] strCube;
```

- ◆ Creating a multidimensional array

- ◆ Use new keyword
- ◆ Must specify the size of each dimension

```
int[,] intMatrix = new int[3, 4];  
float[,] floatMatrix = new float[8, 2];  
string[,,] stringCube = new string[5, 5, 5];
```

Initializing Multidimensional Arrays with Values

- ◆ Creating and initializing with values multidimensional array:

```
int[,] matrix =  
{  
    {1, 2, 3, 4}, // row 0 values  
    {5, 6, 7, 8}, // row 1 values  
}; // The matrix size is 2 x 4 (2 rows, 4 cols)
```

- Matrices are represented by a list of rows
 - Rows consist of list of values
- The first dimension comes first, the second comes next (inside the first)

Accessing The Elements of Multidimensional Arrays

- ◆ Accessing N-dimensional array element:

```
nDimensionalArray[index1, ... , indexn]
```

- ◆ Getting element value example:

```
int[,] array = {{1, 2}, {3, 4}}
int element11 = array[1, 1]; // element11 = 4
```

- ◆ Setting element value example:

```
int[,] array = new int[3, 4];
for (int row=0; row<array.GetLength(0); row++)
    for (int col=0; col<array.GetLength(1); col++)
        array[row, col] = row + col;
```

Number
of rows

Number of
columns

Reading a Matrix – Example

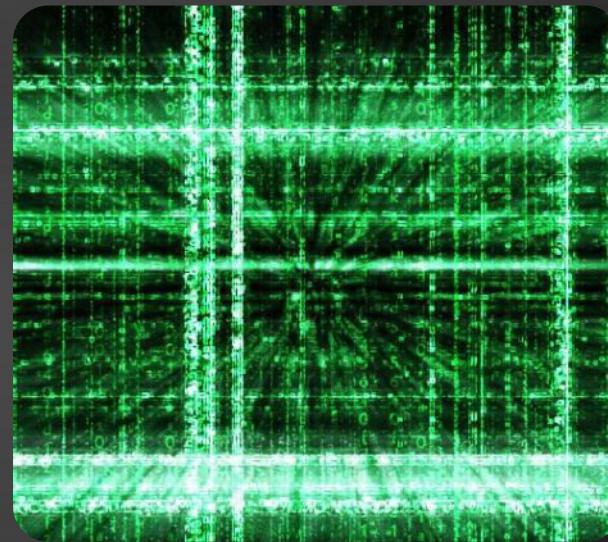
- ◆ Reading a matrix from the console

```
int rows = int.Parse(Console.ReadLine());
int columns = int.Parse(Console.ReadLine());
int[,] matrix = new int[rows, columns];
String inputNumber;
for (int row=0; row<rows; row++)
{
    for (int column=0; column<cols; column++)
    {
        Console.Write("matrix[{0},{1}] = ", row, column);
        inputNumber = Console.ReadLine();
        matrix[row, column] = int.Parse(inputNumber);
    }
}
```

Printing Matrix – Example

- ◆ Printing a matrix on the console:

```
for (int row=0; row<matrix.GetLength(0); row++)  
{  
    for (int col=0; col<matrix.GetLength(1); col++)  
    {  
        Console.Write("{0} ", matrix[row, col]);  
    }  
    Console.WriteLine();  
}
```



Reading and Printing Matrices

Live Demo

Maximal Platform – Example

- ◆ Finding a 2×2 platform in a matrix with a maximal sum of its elements

```
int[,] matrix = {
    {7, 1, 3, 3, 2, 1},
    {1, 3, 9, 8, 5, 6},
    {4, 6, 7, 9, 1, 0}
};
int bestSum = int.MinValue;
for (int row=0; row<matrix.GetLength(0)-1; row++)
    for (int col=0; col<matrix.GetLength(1)-1; col++)
    {
        int sum = matrix[row, col] + matrix[row, col+1]
            + matrix[row+1, col] + matrix[row+1, col+1];
        if (sum > bestSum)
            bestSum = sum;
    }
}
```



Maximal Platform

Live Demo

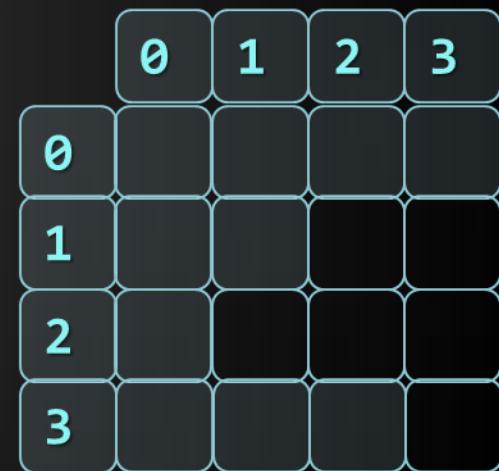


Jagged Arrays

What are Jagged Arrays and How to Use Them?

Jagged Arrays

- ◆ Jagged arrays are like multidimensional arrays
 - ◆ But each dimension has different size
 - ◆ A jagged array is array of arrays
 - ◆ Each of the arrays has different length
- ◆ How to create jagged array?



```
int[][] jagged = new int[3][];  
jagged[0] = new int[3];  
jagged[1] = new int[2];  
jagged[2] = new int[5];
```

Initialization of Jagged Arrays

- ◆ When creating jagged arrays
 - ◆ Initially the array is created of null arrays
 - ◆ Need to initialize each of them

```
int[][] jagged=new int[n][];
for (int i=0; i<n; i++)
{
    jagged[i] = new int[i];
}
```



Jagged Arrays

Live Demo

Example of Jagged Arrays

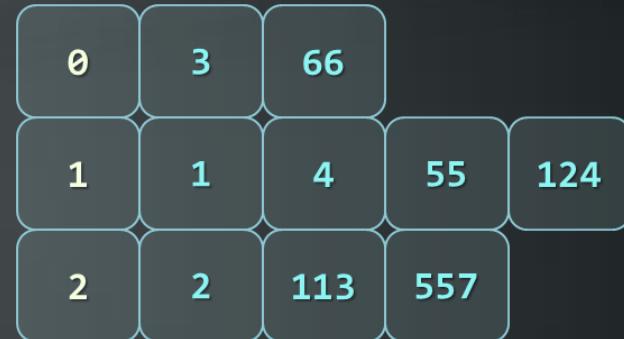
- ◆ Check a set of numbers and group them by their remainder when dividing to 3 (0, 1 and 2)
- ◆ Example: 0, 1, 4, 113, 55, 3, 1, 2, 66, 557, 124, 2
- ◆ First we need to count the numbers
 - ◆ Done with a iteration
- ◆ Make jagged array with appropriate sizes
- ◆ Each number is added into its jagged array

0	3	66
1	1	4
2	2	113

55	124
557	

Example of Jagged Arrays

```
int[] numbers = {0,1,4,113,55,3,1,2,66,557,124,2};  
int[] sizes = new int[3];  
int[] offsets = new int[3];  
foreach (var number in numbers)  
{  
    int remainder = number % 3;  
    sizes[remainder]++;  
}  
int[][] numbersByRemainder = new int[3][] {  
    new int[sizes[0]], new int[sizes[1]],  
    new int[sizes[2]] };  
foreach (var number in numbers)  
{  
    int remainder = number % 3;  
    int index = offsets[remainder];  
    numbersByRemainder[remainder][index] = number;  
    offsets[remainder]++;  
}
```



Remainders of 3

Live Demo





Array Class

What Can We Use?

- ◆ The **System.Array** class

- ◆ Parent of all arrays
- ◆ All arrays inherit from it
- ◆ All arrays have the same:
 - ◆ Basic functionality
 - ◆ Basic properties
 - ◆ E.g. Length property



- ◆ Important methods and properties of **System.Array**
 - ◆ **Rank** – number of dimensions
 - ◆ **Length** – number of all elements through all dimensions
 - ◆ **GetLength(index)** – returns the number of elements in the specified dimension
 - ◆ Dimensions are numbered from 0

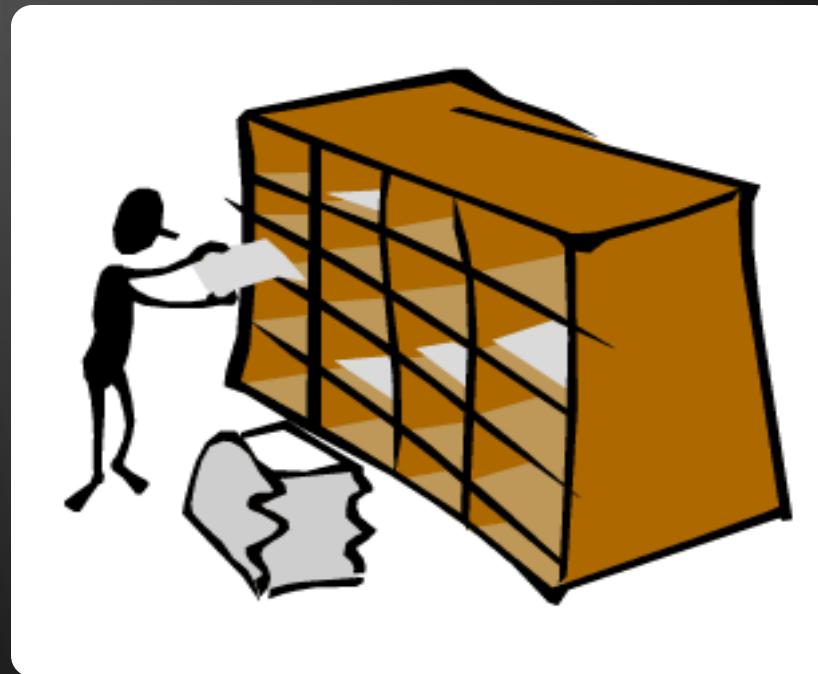
Methods of Array (2)

- ◆ `GetEnumerator()` – returns `IEnumerator` for the array elements
- ◆ `BinarySearch(...)` – searches for a given element into a sorted array (uses binary search)
- ◆ `IndexOf(...)` – searches for a given element and returns the index of the first occurrence (if any)
- ◆ `LastIndexOf(...)` – searches for a given element and returns the last occurrence index
- ◆ `Copy(src, dest, len)` – copies array elements; has many overloads

Methods of Array (3)

- ◆ `Reverse(...)` – inverts the arrays elements upside down
- ◆ `Clear(...)` – assigns value `0` (null) for each elements
- ◆ `CreateInstance(...)` – creates an array
 - ◆ Accepts as parameters the number of dimensions, start index and number of elements
- ◆ Implements `ICloneable`, `IList`, `ICollection` and `IEnumerable` interfaces

Sorting Arrays



- ◆ Sorting in .NET is usually done with **System.Array.Sort()**
 - ◆ **Sort(Array)** – sorts array elements
 - ◆ Elements should implement **IComparable**
 - ◆ **Sort(Array, IComparer)** – sorts array elements by given external **IComparer**
 - ◆ **Sort(Array, Comparison<T>)** – sorts array elements by given comparison operation
 - ◆ Can be used with lambda expression

Sorting Arrays – Example

```
static void Main()
{
    String[] beers = {"Zagorka", "Ariana",
                      "Shumensko", "Astika", "Kamenitza", "Bolqrka",
                      "Amstel"};
    Console.WriteLine("Unsorted: {0}",
                      String.Join(", ", beers));
    // Elements of beers array are of String type,
    // which implement IComparable
    Array.Sort(beers);
    Console.WriteLine("Sorted: {0}",
                      String.Join(", ", beers));
    // Result: Sorted: Amstel, Ariana, Astika,
    // Bolyarka, Kamenitza, Shumensko, Zagorka
}
```



Sorting with `IComparer<T>` and Lambda Expressions – Example

```
class Student
{
    ...
}

public class StudentAgeComparer : IComparer<Student>
{
    public int Compare(Student firstStudent, Student
        secondStudent)
    {
        return firstStudent.Age.CompareTo(secondStudent.Age);
    }
}

...
Array.Sort(students, new StudentAgeComparer());
...

Array.Sort(students, (x, y) => x.Name.CompareTo(y.Name));
```

Sorting with `IComparer<T>` and Lambda Expressions

Live Demo



Binary Search



- ◆ Binary search is a fast method for searching for an element in a sorted array
 - ◆ Has guaranteed running time of $O(\log(n))$ for searching among arrays of with n elements
- ◆ Implemented in the `Array.BinarySearch(Array, object)` method
 - ◆ Returns the index of the found object or a negative number when not found

Binary Search (2)

- ◆ All requirements of the Sort() method are applicable for BinarySearch()
 - ◆ Either all elements should implement `IComparable<T>` or instance of `IComparer<T>` should be passed

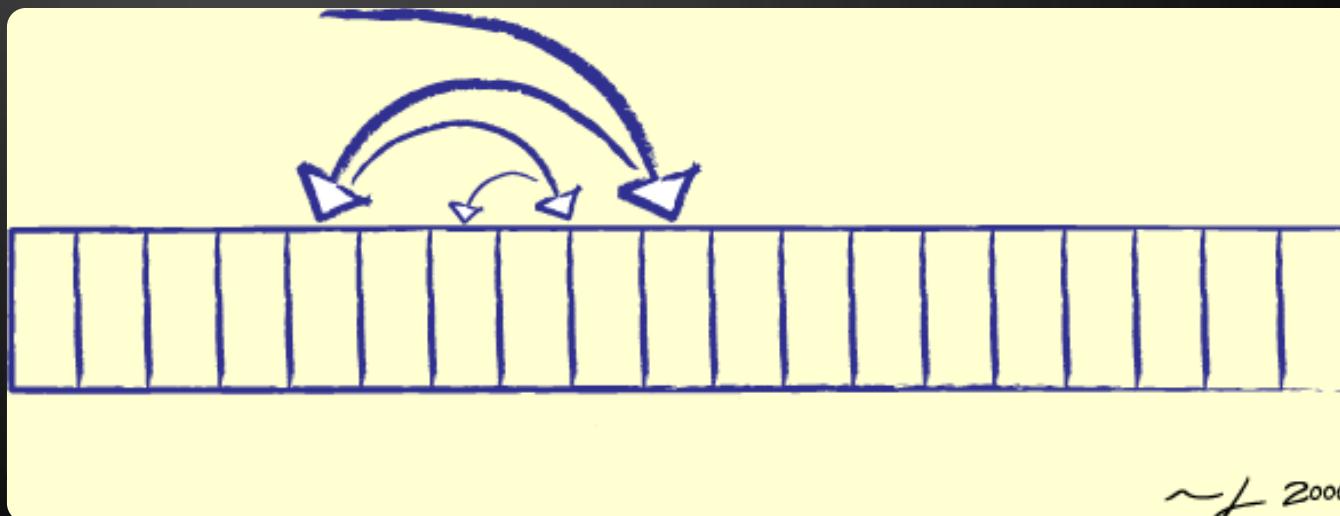


Binary Search – Example

```
static void Main()
{
    String[] beers = {"Zagorka", "Ariana",
                      "Shumensko", "Astika", "Kamenitza", "Bolqrka",
                      "Amstel"};
    Array.Sort(beers);
    string target = "Astika";
    int index = Array.BinarySearch(beers, target);
    Console.WriteLine("{0} is found at index {1}.",
                      target, index);
    // Result: Astika is found at index 2.
    target = "Heineken";
    index = Array.BinarySearch(beers, target);
    Console.WriteLine("{0} is not found (index={1}).",
                      target, index);
    // Result: Heineken is not found (index=-5).
}
```

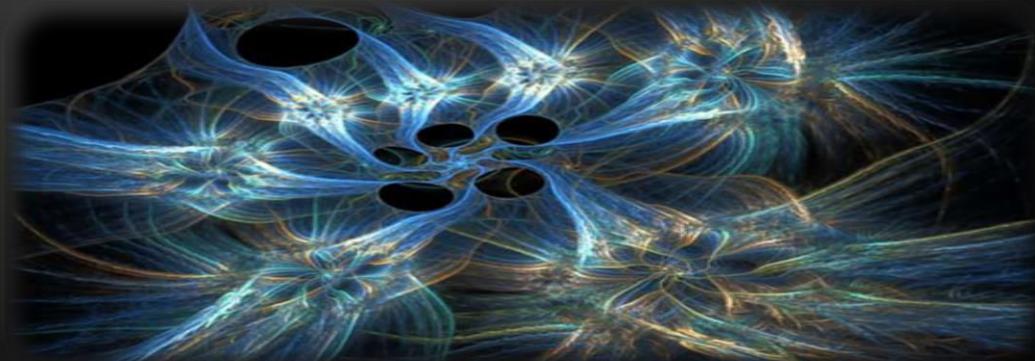
Binary Search

Live Demo



Working with Arrays

Best Practices



Advices for Working with Arrays

- ◆ When given method returns an array and should return an empty array (array with 0 elements) instead of null
- ◆ Arrays are passed by reference
 - To be sure that given method will not change the passed array, pass a copy of it
- ◆ Clone() returns shallow copy of the array
 - You should implement your own deep clone when working with reference types

Questions?

1. Write a program that fills and prints a matrix of size (n, n) as shown below: (examples for $n = 4$)

a)

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

b)

1	8	9	16
2	7	10	15
3	6	11	14
4	5	12	13

c)

7	11	14	16
4	8	12	15
2	5	9	13
1	3	6	10

d) *

1	12	11	10
2	13	16	9
3	14	15	8
4	5	6	7

Exercises (2)

2. Write a program that reads a rectangular matrix of size $N \times M$ and finds in it the square 3×3 that has maximal sum of its elements.
3. We are given a matrix of strings of size $N \times M$. Sequences in the matrix we define as sets of several neighbor elements located on the same line, column or diagonal. Write a program that finds the longest sequence of equal strings in the matrix. Example:

ha	fifi	ho	hi
fo	ha	hi	xx
xxx	ho	ha	xx

→ ha, ha, ha

s	qq	s
pp	pp	s
pp	qq	s

→ s, s, s

4. Write a program, that reads from the console an array of N integers and an integer K, sorts the array and using the method `Array.BinSearch()` finds the largest number in the array which is $\leq K$.
5. You are given an array of strings. Write a method that sorts the array by the length of its elements (the number of characters composing them).
6. * Write a class `Matrix`, to holds a matrix of integers. Overload the operators for adding, subtracting and multiplying of matrices, indexer for accessing the matrix content and `ToString()`.

7. * Write a program that finds the largest area of equal neighbor elements in a rectangular matrix and prints its size. Example:

1	3	2	2	2	4
3	3	3	2	4	4
4	3	1	2	3	3
4	3	1	3	3	1
4	3	3	3	1	1

→ 13

Hint: you can use the algorithm "Depth-first search" or "Breadth-first search" (find them in Wikipedia).

Free Trainings @ Telerik Academy

- ◆ “C# Programming @ Telerik Academy

- ◆ csharpfundamentals.telerik.com



- ◆ Telerik Software Academy

- ◆ academy.telerik.com



- ◆ Telerik Academy @ Facebook

- ◆ facebook.com/TelerikAcademy



- ◆ Telerik Software Academy Forums

- ◆ forums.academy.telerik.com

