Arrays in C#

Școala Informală de IT

Cluj-Napoca

Școala
informală
de IT

Arrays

Processing Sequences of Elements

Agenda

- Declaring and Creating Arrays
- Accessing Array Elements
- Console Input and Output of Arrays
- Iterating Over Arrays Using `for` and `foreach`
- Copying Arrays

Agenda

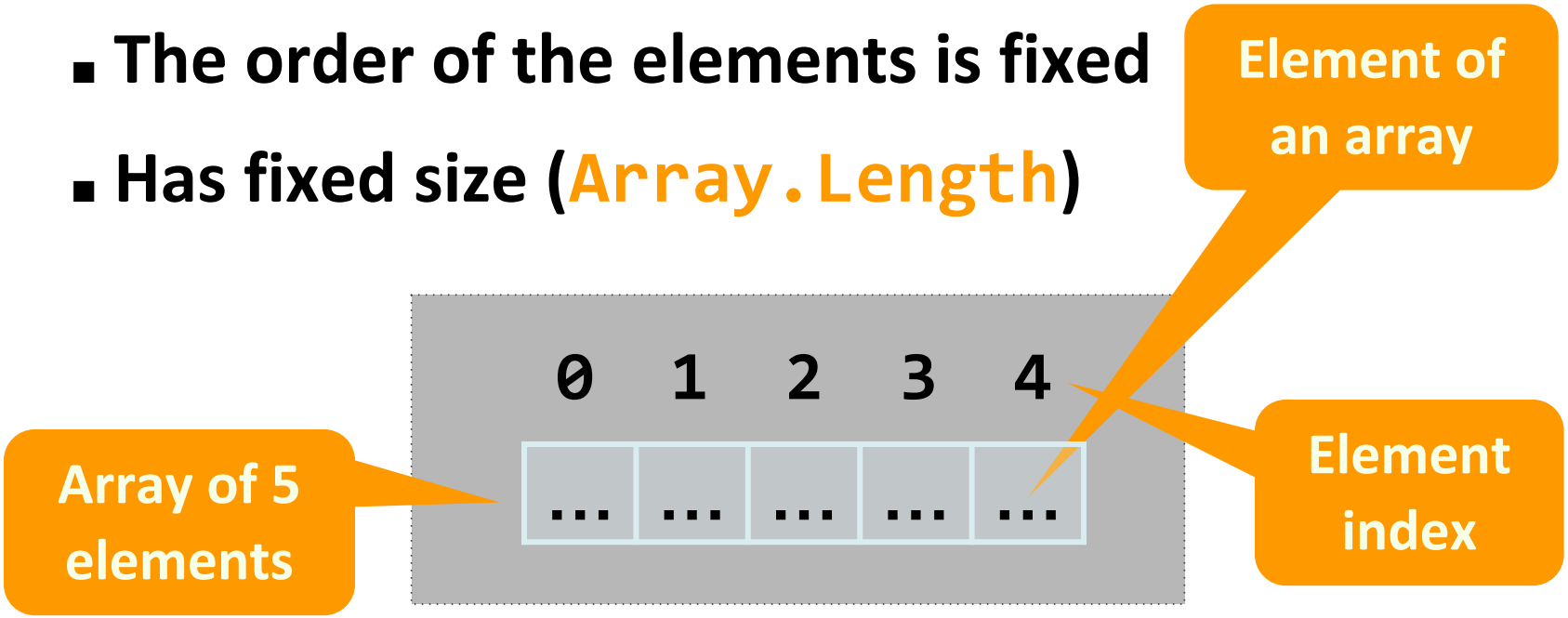
- Matrices and Multidimensional Arrays
  - Declaring and Usage
- Jagged Arrays
  - Declaring and Usage
- The `Array` Class
  - Sorting
  - Binary Search

Școala
informală
de IT

Declaring and Creating Arrays

What are Arrays?

- An array is a sequence of elements
  - All elements are of the same type
  - The order of the elements is fixed
  - Has fixed size (`Array.Length`)



Declaring Arrays

- Declaration defines the type of the elements
- Square brackets [ ] mean "array"
- Examples:
  - Declaring array of integers:

```
int[] myIntArray;
```

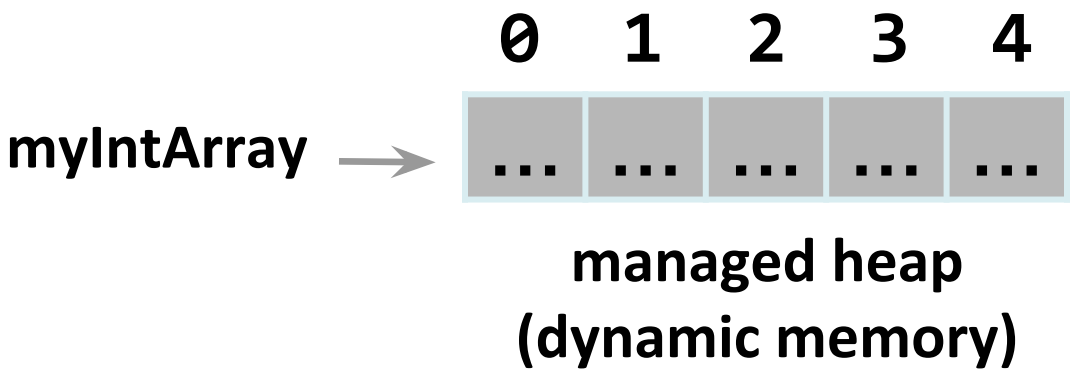
  - Declaring array of strings:

```
string[] myStringArray;
```

Creating Arrays

- Use the operator `new`
  - Specify array length
- Example create (allocating) array of 5 integers:

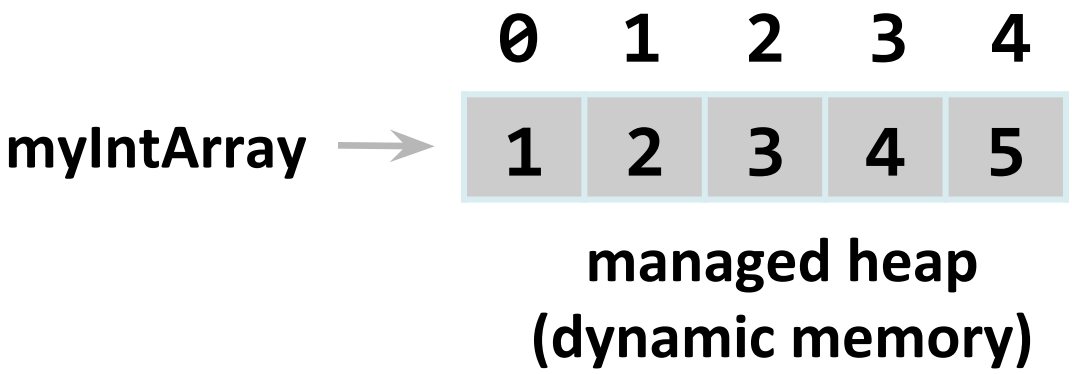
```
myIntArray = new int[5];
```



Creating and Initializing Arrays

- **Creating and initializing can be done together:**

```
myIntArray = {1, 2, 3, 4, 5};
```



- **The new operator is not required when using curly brackets initialization**

Creating Array – Example

- **Creating an array that contains the names of the days of the week**

```
string[] daysOfWeek =
{
    "Monday",
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday",
    "Saturday",
    "Sunday"
};
```

Accessing Array Elements

Read and Modify Elements by Index

Școala informală de IT

How to Access Array Element?

- Array elements are accessed using the square brackets operator `[]` (indexer)
  - Array indexer takes element's index as parameter
  - The first element has index `0`
  - The last element has index `Length-1`
- Array elements can be retrieved and changed by the `[]` operator

Reversing an Array – Example

- **Reversing the contents of an array**

```
int[] array = new int[] {1, 2, 3, 4, 5};

// Get array size
int length = array.Length;

// Declare and create the reversed array
int[] reversed = new int[length];

// Initialize the reversed array
for (int index = 0; index < length; index++)
{
    reversed[length-index-1] = array[index];
}
```

Școala
informală
de IT

Reversing an Array

Live Demo

Școala
informală
de IT

Arrays: Input and Output

Reading and Printing Arrays on the Console

Școala informală de IT

Reading Arrays From the Console

- First, read from the console the length of the array

```
int n = int.Parse(Console.ReadLine());
```

- Next, create the array of given size and read its elements in a `for` loop

```
int[] arr = new int[n];
for (int i=0; i<n; i++)
{
   arr[i] = int.Parse(Console.ReadLine());
}
```

Școala informală de IT

Printing Arrays on the Console

- Process all elements of the array
- Print each element to the console
- Separate elements with white space or a new line

```
string[] array = {"one", "two", "three"};

// Process all elements of the array
for (int index = 0; index < array.Length; index++)
{
    // Print each element on a separate line
    Console.WriteLine("element[{0}] = {1}",
        index, array[index]);
}
```

Școala
informală
de IT

Printing Arrays

Live Demo

Școala
informală
de IT

Processing Array Elements Using `for` and `foreach`

Processing Arrays: for Statement

- Use `for` loop to process an array when
  - Need to keep track of the index
  - Processing is not strictly sequential from the first to the last element
- In the loop body use the element at the loop index (`array[index]`):

```
for (int index = 0; index < array.Length; index++)
{
   squares[index] = array[index] * array[index];
}
```

Școala informală de IT

Processing Arrays Using for Loop – Examples

- **Printing array of integers in reversed order:**

```
Console.WriteLine("Reversed: ");
for (int i = array.Length-1; i >= 0; i--)
{
    Console.Write(array[i] + " ");
}
// Result: 5 4 3 2 1
```

- **Initialize all array elements with their corresponding index number:**

```
for (int index = 0; index < array.Length; index++)
{
    array[index] = index;
}
```

Processing Arrays: foreach

- How `foreach` loop works?

```
foreach (type value in array)
```

  - `type` – the type of the element
  - `value` – local name of variable
  - `array` – processing array
- Used when no indexing is needed
  - All elements are accessed one by one
  - Elements can not be modified (read only)

Processing Arrays Using foreach – Example

- Print all elements of a `string[]` array:

```
string[] capitals =
{
    "Sofia",
    "Washington",
    "London",
    "Paris"
};
foreach (string capital in capitals)
{
    Console.WriteLine(capital);
}
```

Școala
informală
de IT

Processing Arrays

Live Demo

Arrays

- Lets have an array with capacity of 5 elements

```
int[] intArray = new int[5];
```

- If we want to add a sixth element (we have already added 5) we have to manually resize

```
int[] copyArray = intArray; // create backup
intArray = new int[6];      // resize
for (int i = 0; i < 5; i++)
{
   intArray[i] = copyArray[i];
}
intArray[5] = newValue; // add sixth element
```

Școala
informală
de IT

Copying Arrays

The Array Class

Copying Arrays

- Sometimes we must copy the values from one array to another one
  - Doing the intuitive way we would copy not only the values but the reference to the array so changing some of the values in one array will affect the other
  - The way to avoid this is using `Clone()`

```
int[] copyArray = array;
```

  - This way only the values will be copied but not the reference

```
int[] copyArray = (int[])array.Clone();
```

Școala
informală
de IT

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

Școala informală de IT

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];
```

Școala informală de IT

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)**

Școala informală de IT

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

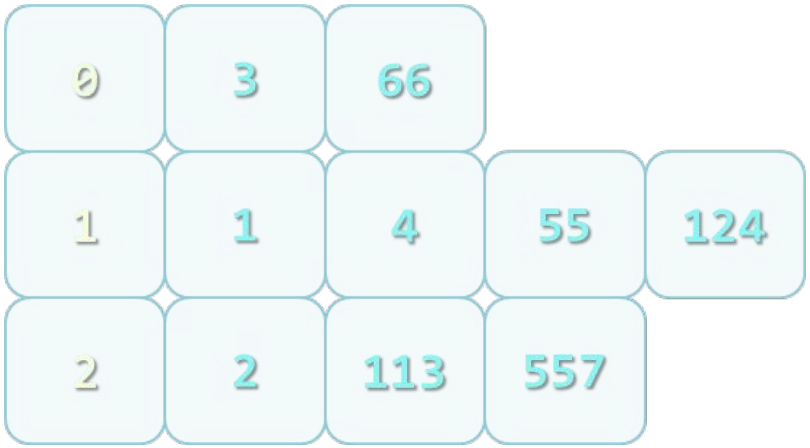
- Reading a matrix from the console

```
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();
}
```

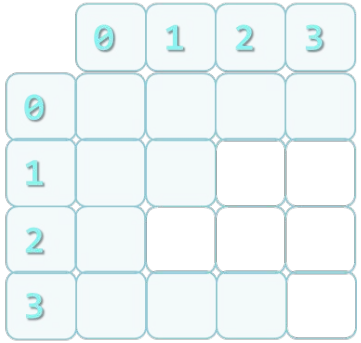


Jagged Arrays

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];
}
```

Școala informală de IT

Advices for Working with Arrays

- When given method returns an array and should return an empty array, return an 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

Summary

- Arrays are a fixed-length sequences of elements of the same type
- Array elements are accessible by index
  - Can be read and modified
- Iteration over array elements can be done with `for` and `foreach` loops

Resources

- http://csharp.net-informations.com/
- Telerik