# Arrays

## Simple Array Processing


**SoftUni Team**

**Technical Trainers**

**Software University**

http://softuni.bg

# Table of Contents



# Questions?


## sli.do

# #extended-softuni

SOFTWARE UNIVERSITY FOUNDATION

# Value vs Reference Types

## Memory Stack and Heap

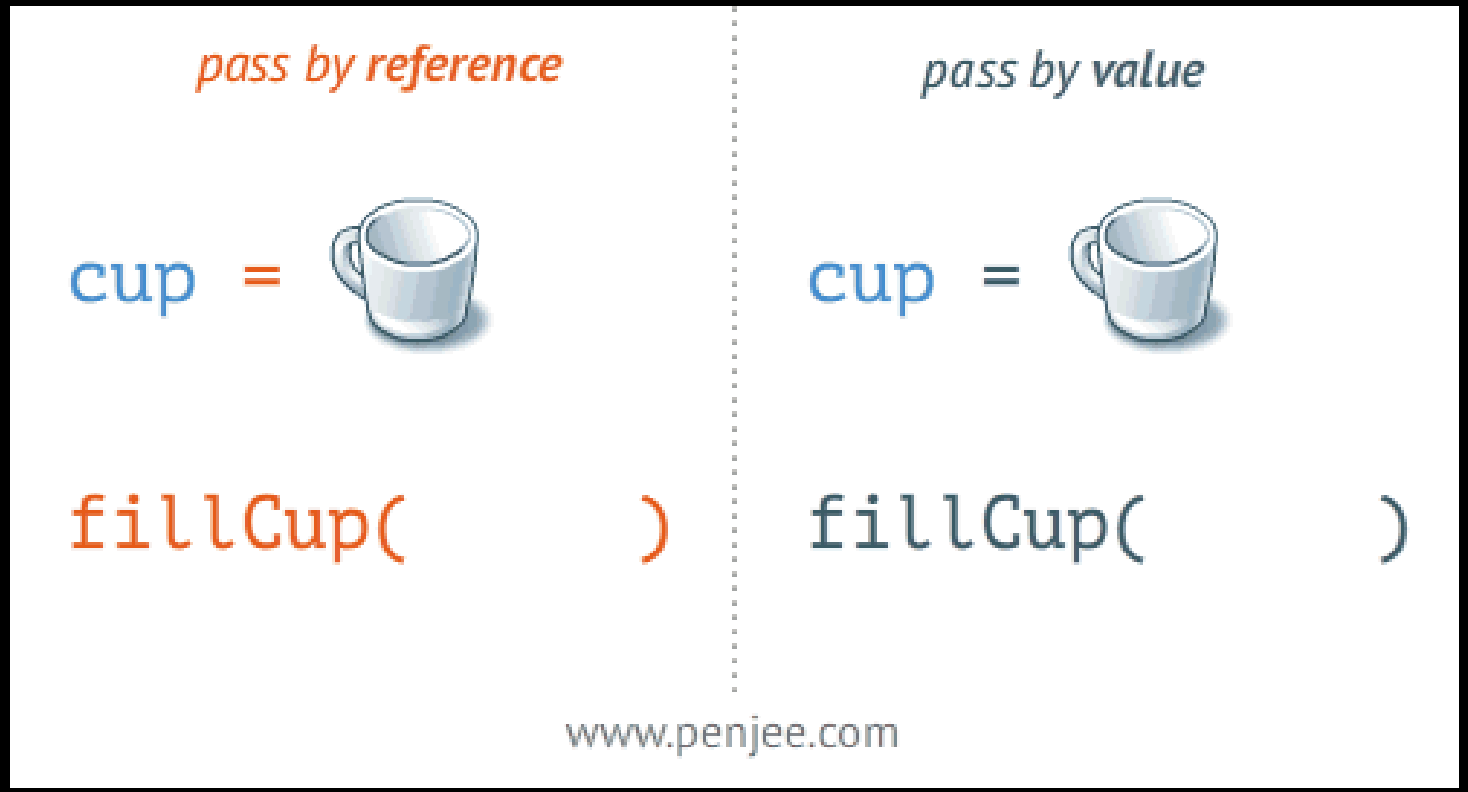
# Value types

- List of Value Types - https://msdn.microsoft.com/en-us/library/bfft1t3c.aspx
- Variables of value types directly contain their data.
- With value types, each variable has its own copy of the data, and it is not possible for operations on one variable to affect the other

# Reference types

- Some of the reference types – string, DateTime, TimeSpan, Random, any other classes, interfaces, delegates and more.
- Variables of reference types store references to their data.
- With reference types, two variables can reference the same object; therefore, operations on one variable can affect the object referenced by the other variable.

# Value vs Reference Types



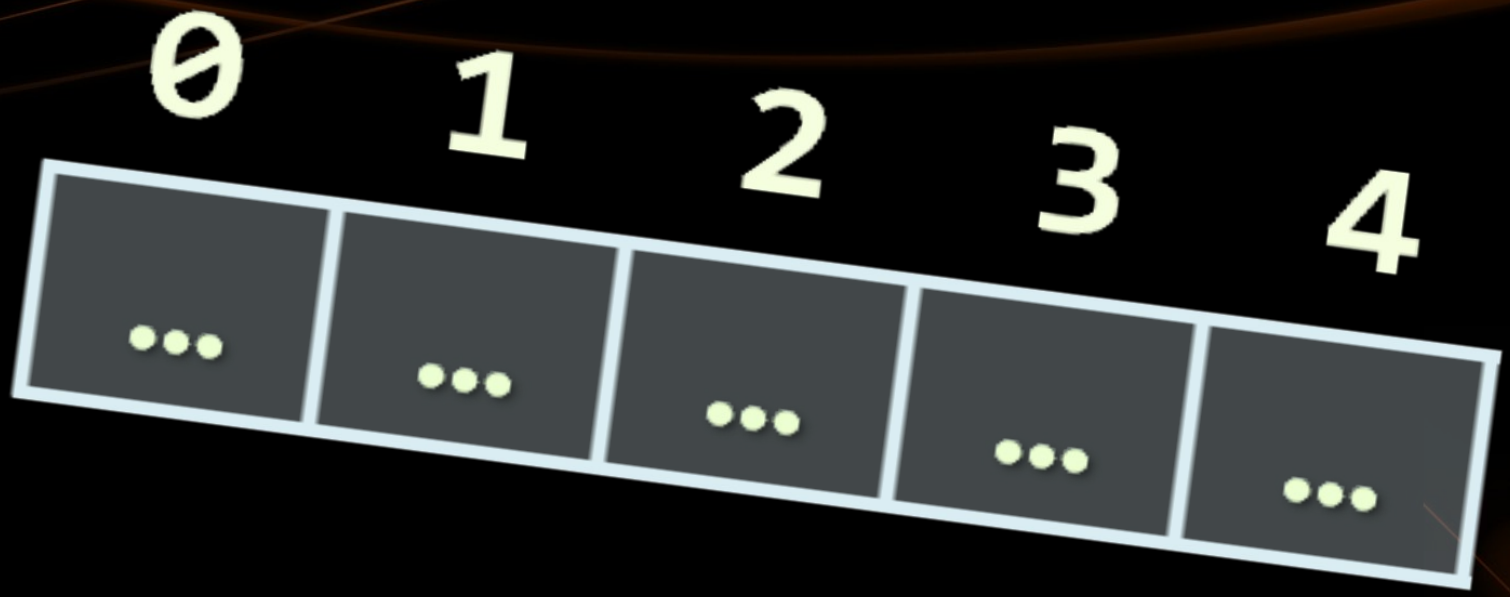
# Example: Value Types

```
public static void Main(string[] args)
{
    int num = 5;
    Increment(num, 15);
    Console.WriteLine(num);
}

private static void Increment(int num, int value)
{
     num += value;
}
```

# Example: Reference Types

```
public static void Main(string[] args)
{
    int[] nums = { 5 };
    Increment(nums, 15);
    Console.WriteLine(nums[0]);
}

private static void Increment(int[] nums, int value)
{
     nums[0] += value;
}
```

SOFTWARE UNIVERSITY FOUNDATION

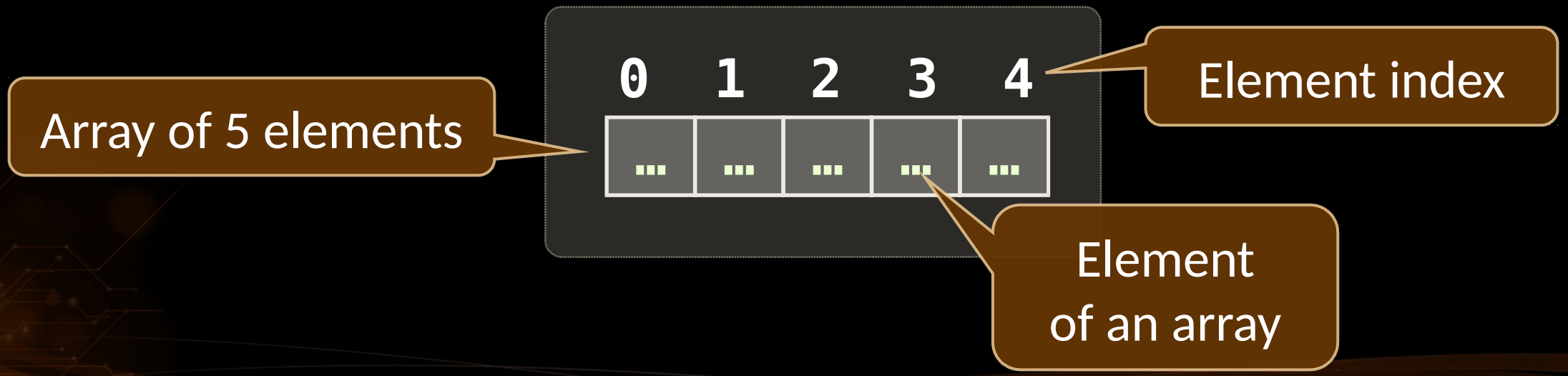



# Arrays

## Working with Arrays of Elements

# What are Arrays?

- In programming, **array** is a sequence of elements
  - Elements are numbered from `0` to `Length-1`
  - Elements are of the same type (e.g. integers)
  - Arrays have fixed size (`Array.Length`) – cannot be resized



# Working with Arrays

- Allocating an array of 10 integers:

```
int[] numbers = new int[10];
```

All elements are initially == 0

- Assigning values to the array elements:

```
for (int i = 0; i < numbers.Length; i++)
    numbers[i] = 1;
```

The Length holds the number of array elements

- Accessing array elements by index:

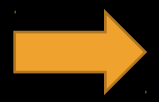
```
numbers[5] = numbers[2] + numbers[7];
numbers[10] = 1; // IndexOutOfRangeException
```

The [] operator accesses elements by index

# Example: Days of Week

- The days of week can be stored in array of strings:

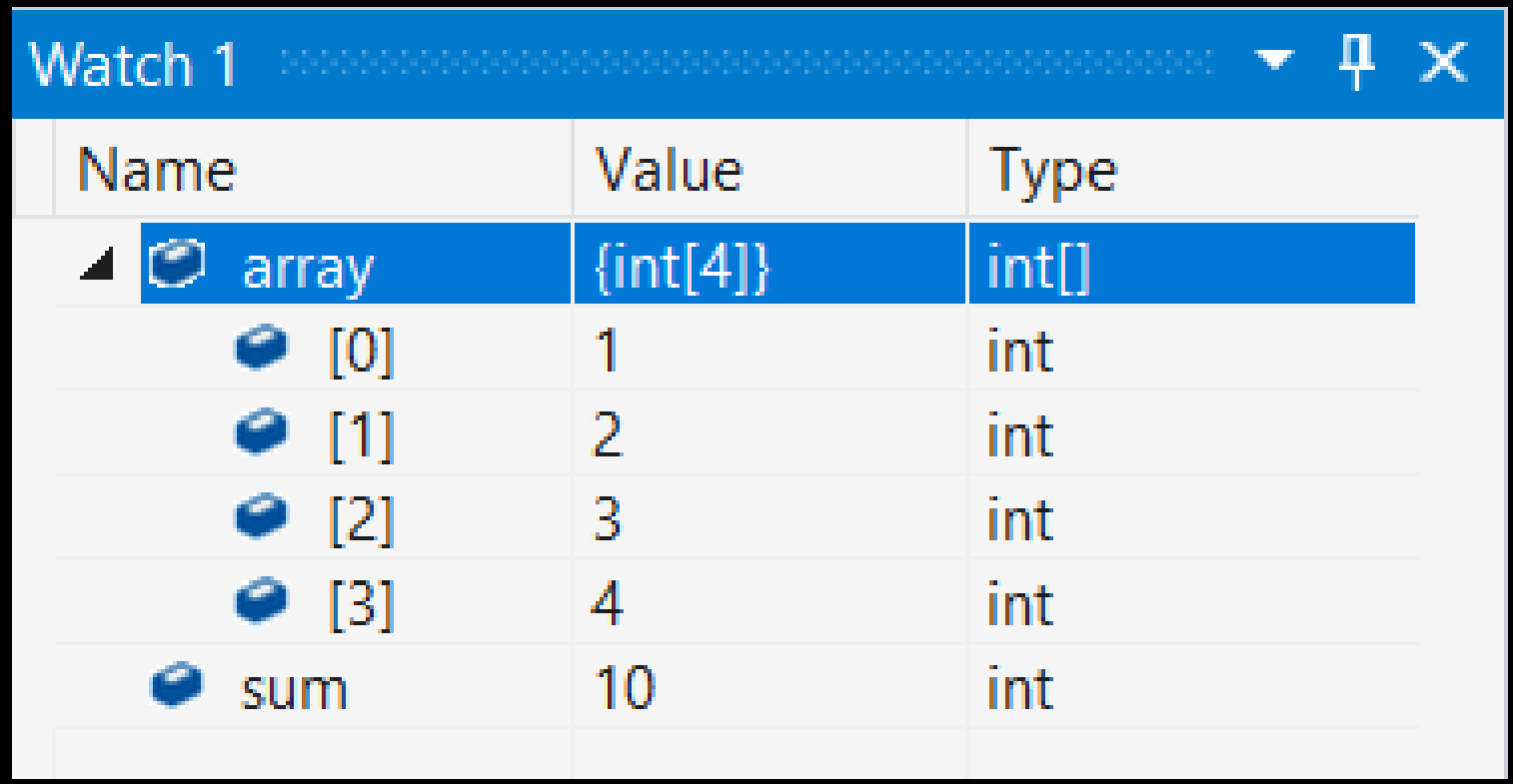
```
string[] days = {
   "Monday",
   "Tuesday",
   "Wednesday",
   "Thursday",
   "Friday",
   "Saturday",
   "Sunday"
};
```



| Expression | Value |
|---|---|
| days[0] | Monday |
| days[1] | Tuesday |
| days[2] | Wednesday |
| days[3] | Thursday |
| days[4] | Friday |
| days[5] | Saturday |
| days[6] | Sunday |

# Problem: Sum Array Elements

- Read an array of integers and calculate their sum



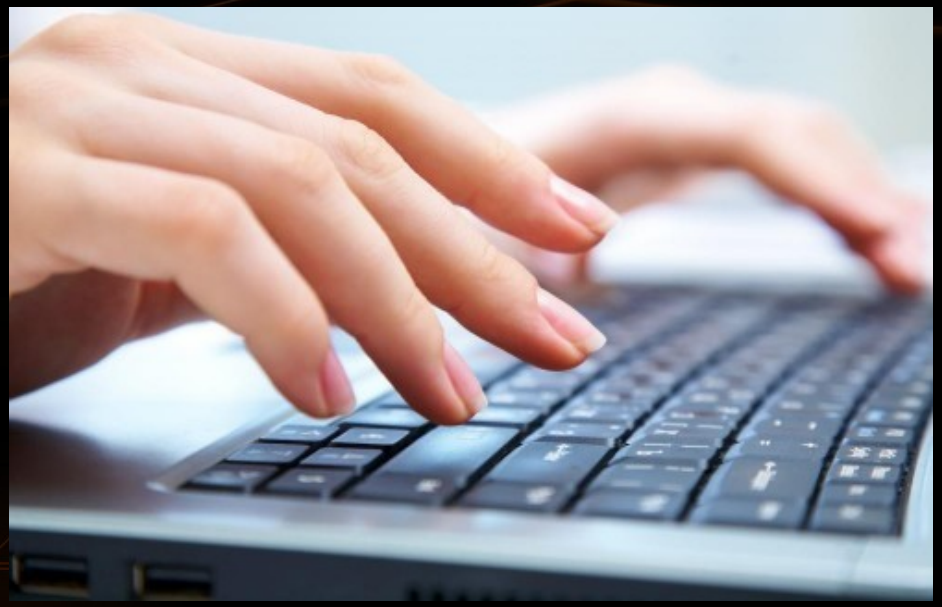

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#0

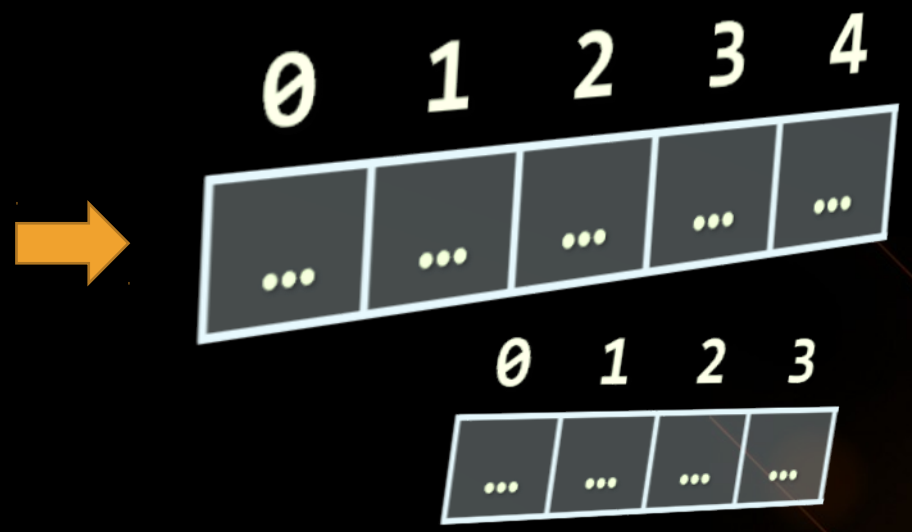
# Solution: Sum Array Elements

```
var numberOfElements =
int.Parse(Console.ReadLine());
var array = new int[numberOfElements];

for (int i = 0; i < array.Length; i++)
  array[i] = int.Parse(Console.ReadLine());

var sum = 0;
for (int i = 0; i < array.Length; i++)
  sum += array[i];

Console.WriteLine(sum);
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#0

SOFTWARE UNIVERSITY FOUNDATION






# Reading and Printing Arrays

## Using `String.Split()` and `for` loops

# Reading Arrays From the Console



- First, read the array length from the console:

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

- Next, create an array of given size **n** and read its elements:

```
int[] arr = new int[n];

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

# Reading Array Values From a Single Line

- Arrays can be read from a single line of space separated values:

```
2 8 30 25 40 72 -2 44 56
```

```
string values = Console.ReadLine();

string[] items = values.Split(' ');

int[] arr = new int[items.Length];

for (int i = 0; i < items.Length; i++)
    arr[i] = int.Parse(items[i]);
```

`string.Split(' ')` splits `string` by space and produces `string[]`

# Printing Arrays on the Console

- To print all array elements, a **`for`**-loop can be used
  - Separate elements with white space or a new line
- Example:

```
string[] arr = {"one", "two", "three", "four",
"five"};

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

# Problem: Multiply an Array of Doubles

- Read an array of real numbers (space separated values) and a real number p
- Multiply all array elements by p
- Print the multiplied elements (on a single line, space separated):

```
1.2 3.0 12.3
4.56
4
```

```
4.8 12 49.2
18.24
```

```
6.0 8.8 1.2
-9.6
0.5
```

```
3 4.4 0.6 -4.8
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#1

# Solution: Multiply an Array of Doubles

```
var stringArr = Console.ReadLine().Split();
var arr = new double[stringArr.Length];

for (int i = 0; i < arr.Length; i++)
  arr[i] = double.Parse(stringArr[i]);

var p = double.Parse(Console.ReadLine());
for (int i = 0; i < arr.Length; i++)
  arr[i] *= p;

for (int i = 0; i < arr.Length; i++)
  Console.Write(arr[i] + " ");
```

split the input string by space

convert strings to **double**

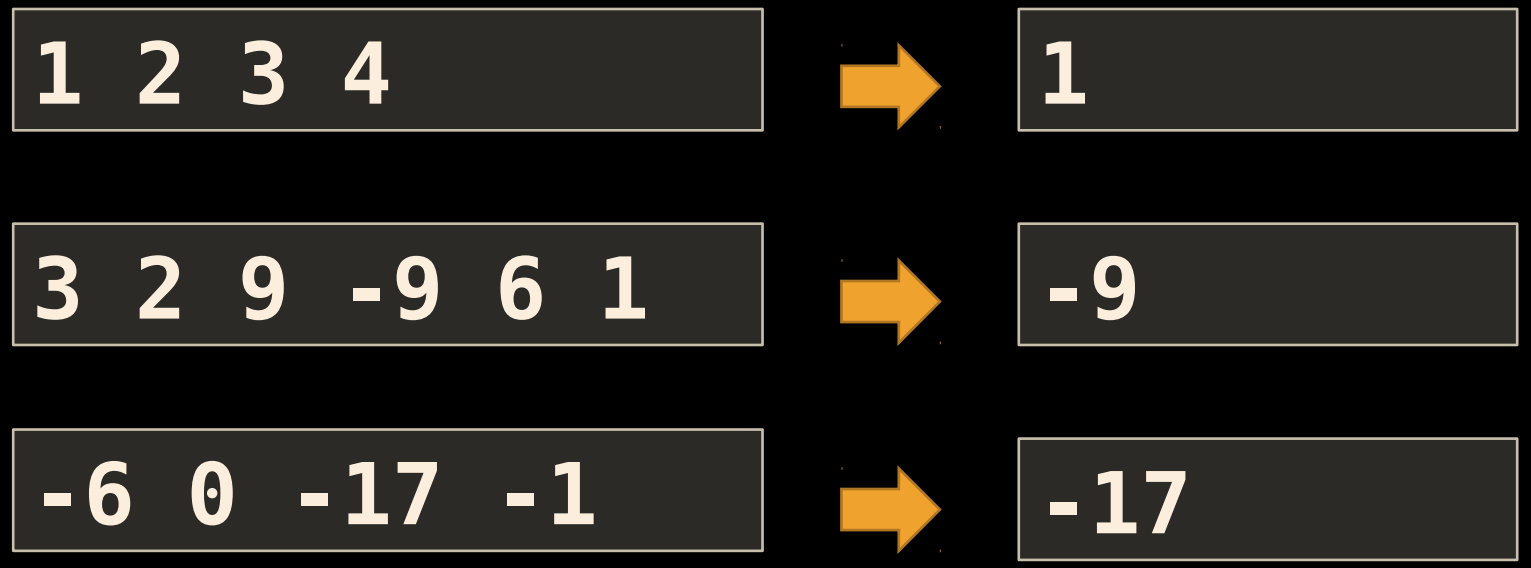
multiply array elements by p

print multiplied array

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#1

# Problem: Smallest Element in Array

- Read an array of integers (space separated values), find the smallest element and print it:




Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#4

# Solution: Smallest Element in Array

```
var stringArray = Console.ReadLine().Split();
var array = new int[stringArray.Length];

for (int i = 0; i < array.Length; i++)
  array[i] = int.Parse(stringArray[i]);

var smallest = int.MaxValue;
for (int i = 0; i < array.Length; i++)
  if (array[i] < smallest)
    smallest = array[i];
Console.WriteLine(smallest);
```

if we start at 0 instead of **MaxValue**, we won't catch positive numbers

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#4

# Printing Arrays with foreach / String.Join(…)


- Use `foreach`-loop:

```
int[] arr = { 10, 20, 30, 40, 50};
foreach (var element in arr)
    Console.WriteLine(element)
```

- Use `string.Join(separator, array)`:

```
int[] arr = { 1, 2, 3 };
Console.WriteLine(string.Join(", ", arr)); // 1, 2, 3

string[] strings = { "one", "two", "three", "four" };
Console.WriteLine(string.Join(" - ", strings));
```

# Problem: Rotate Array of Strings

- Read an array of strings (space separated values), rotate it to the right and print its rotated elements:

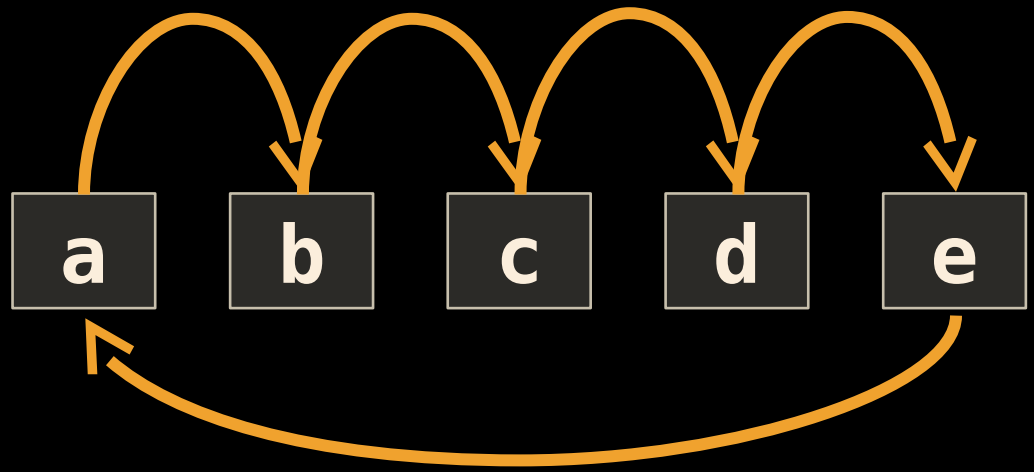
```
a b c d
e
```

```
e a b c
d
```

```
soft uni
hi
```

```
hi soft
uni
```

- Rotating array elements:




Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#5

# Solution: Rotate Array of Strings

```
var array =
Console.ReadLine().Split().ToArray();
var rotatedArray = new string[array.Length];

for (int i = 0; i < array.Length - 1; i++)
  rotatedArray[i + 1] = array[i];

var lastElement = array[rotatedArray.Length -
1];
rotatedArray[0] = lastElement;

Console.WriteLine(string.Join(" ",
rotatedArray));
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#5

# Problem: Odd Numbers at Odd Positions

- Read a list of integers and print the odd numbers at the odd positions (indexes).

```
2 3 5 2 7 9 -1
-7
```

```
Index 1 -> 3
Index 5 -> 9
Index 7 -> -7
```

```
5 0 1 2
```

```
(no output)
```

```
2 3 55 2 4 1
```

```
Index 1 -> 3
Index 5 -> 1
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#5

# Solution: Odd Numbers at Odd Positions

```
var stringArray = Console.ReadLine().Split().ToArray();
var array = new string[stringArray.Length];

// Convert string array to int array
for (int i = 0; i < array.Length; i++)
  array[i] = int.Parse(stringArray[i]);

for (int i = 0; i < array.Length; i++)
{
  if (i % 2 == 1 && Math.Abs(array[i] % 2) == 1)
    Console.WriteLine($"Index {i} -> {array[i]}");
}
```

check for odd index

check for odd element

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/384#5

SOFTWARE UNIVERSITY FOUNDATION

# Arrays

## Live Exercises in Class (Lab)

# Summary

- Arrays hold a sequence of elements
  - Elements are numbered from **0** to **length-1**
- Creating (allocating) an array:

```
int[] numbers = new int[10];
int[] nums = new int[] { 1, 2,
3 };
```


- Accessing array elements by index:

```
numbers[5] = 10;
```

- Printing array elements:

```
Console.Write(string.Join(" ", arr));
```

# Arrays

Questions?

# License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license


- Attribution: this work may contain portions from
  - "Fundamentals of Computer Programming with C#" book by Svetlin Nakov & Co. under CC-BY-SA license

# Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity


- Software University Forums – forum.softuni.bg