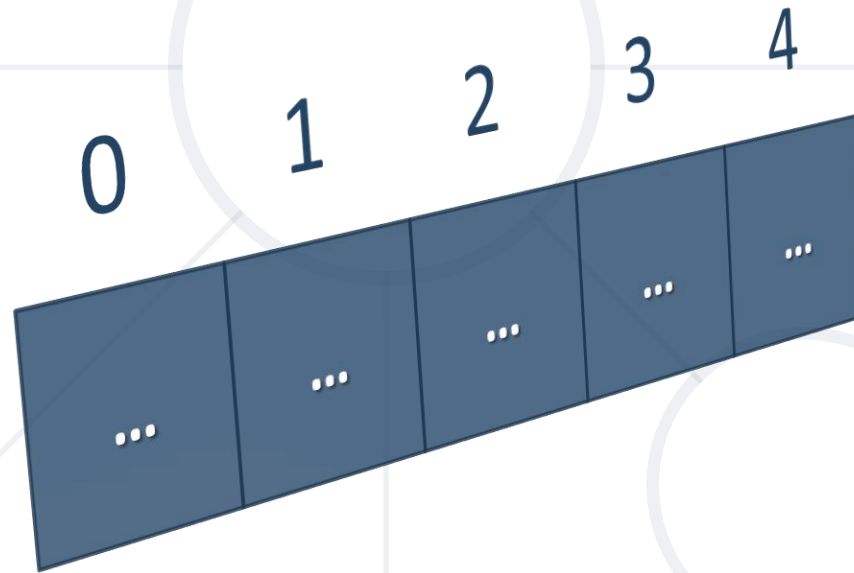# Arrays

Fixed-Size Sequences of Elements

SoftUni Team

Technical Trainers

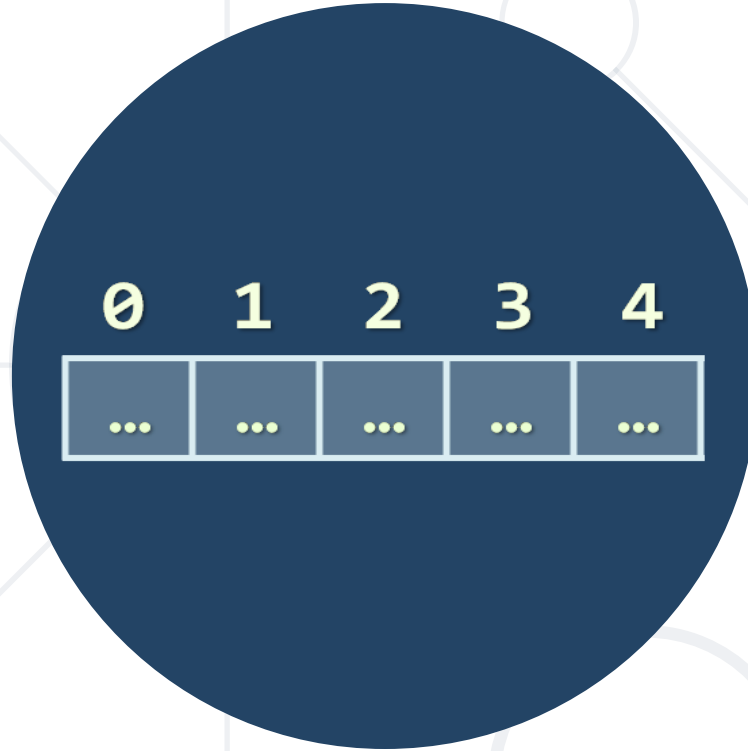SoftUni

Software University

Table of Contents

1. Arrays

2. Reading Arrays from the Console

3. Foreach Loop

sli.do

#fund-csharp

Working with Arrays of Elements

Arrays

What Are Arrays?

- In programming, an **array** is a **sequence of elements**

Array of 5 elements

Element index

0 1 2 3 4

| ... | ... | ... | ... | ... |

Element of an array

- Elements are numbered from **0** to **Length-1**

- Elements are of the **same type** (e.g., integers)

- Arrays have **fixed size** (**Array.Length**) and cannot be resized

Creating Arrays

- Use the **new** keyword

  - It is used to create the **array** and initialize the array elements to their default values

- Allocating an **array** of 10 **integers**

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

All elements are initially == 0

- An array that stores **string** elements can be declared in the same way

```
string[] names = new string[10];
```

All elements are initially == null

Working with Arrays

- **Assigning values** to the array elements
  - The **Length** holds the number of array elements

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

- **Accessing** array elements by index
  - The **[ ]** operator accesses elements by index

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

Days of Week – Example

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

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

Operator	Notation in C#
days[0]	Monday
days[1]	Tuesday
days[2]	Wednesday
days[3]	Thursday
days[4]	Friday
days[5]	Saturday
days[6]	Sunday

Problem: Day of Week

- Enter a day number [1...7] and print
  the day name (in English) or "Invalid day!"

Name	Value	Type
⌐ • days	{string[7]}	string[]
• [0]	"Monday"	⊙ string
• [1]	"Tuesday"	⊙ string
• [2]	"Wednesday"	⊙ string
• [3]	"Thursday"	⊙ string
• [4]	"Friday"	⊙ string
• [5]	"Saturday"	⊙ string
• [6]	"Sunday"	⊙ string

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#0

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

int day = int.Parse(Console.ReadLine());


if (day >= 1 && day <= 7)

    Console.WriteLine(days[day - 1]);

else

    Console.WriteLine("Invalid day!");
```

The first day in our array is on index 0, not 1.

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#0

Reading Arrays

Using a For Loop or string.Split()

Reading Arrays from the Console

- First, read from the console the array's **length**

```
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 **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]);
```

Split() splits by space into **string[]**

Shorter: Reading Array from a Single Line

- Read an array of integers

```
var inputLine = Console.ReadLine();
string[] items = inputLine.Split(", ");
int[] arr = items.Select(int.Parse).ToArray();
```

uses **System.LINQ**;

```
int[] arr = Console.ReadLine().Split(", ")
                .Select(int.Parse).ToArray();
```
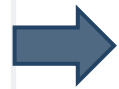
Or shorter

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

```csharp
string[] arr = {"one", "two"};
// == new string [2] {"one", "two"};
// 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: Print Numbers in Reverse Order

- Read an array of integers (**n** lines of integers), **reverse** it and print its elements on a single line, space-separated:

```
3
10      →     30 20 10
20
30
```

```
4
-1      →     5 99 20 -1
20
99
5
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#1

Solution: Print Numbers in Reverse Order

```csharp
// Read the array (n lines of integers)
var n = int.Parse(Console.ReadLine());
var arr = new int[n];
for (int i = 0; i < n; i++)
    arr[i] = int.Parse(Console.ReadLine());
// Print the elements from the last to the first
for (int i = n-1; i >= 0; i--)
    Console.Write(arr[i] + " ");
Console.WriteLine();
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#1

Problem: Rounding Numbers

- Read an array of real numbers (space separated), round them in "**away from 0**" style and print the output as in the examples:

```
0.9 1.5 2.4 2.5 3.14
```

```
0.9 => 1
1.5 => 2
2.4 => 2
2.5 => 3
3.14 => 3
```

```
-5.01 -1.599 -2.5 -1.50 0
```

```
-5.01 => -5
-1.599 => -2
-2.5 => -3
-1.50 => -2
0 => 0
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#2

Solution: Rounding Numbers

- **Rounding** turns each value to the nearest integer

```
double[] nums = Console.ReadLine().Split().Select(double.Parse).ToArray();

int[] roundedNums = new int[nums.Length];

for (int i = 0; i < nums.Length; i++) {

    roundedNums[i] = (int)Math

                    .Round(nums[i], MidpointRounding.AwayFromZero);

}

// TODO: Print each number
```

2.5 => 3

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#2

Printing Arrays with for / string.Join(…)

- Use **for** loop

```
int[] arr = { 10, 20, 30, 40, 50};
for (int i = 0; i < arr.Length; i++)
        Console.WriteLine(arr[i]);
```
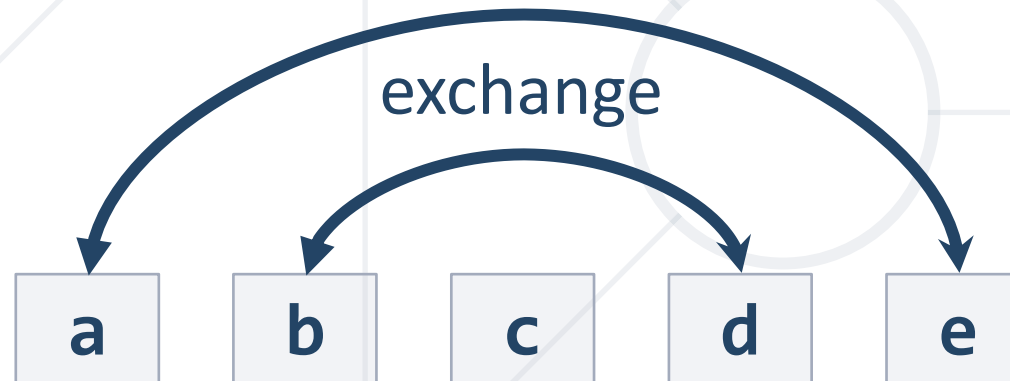
- Use **string.Join(separator, array)**

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

Problem: Reverse Array of Strings

- Read an array of strings (space separated values), reverse it and print its elements:

| a b c d e | ➡ | e d c b a | | -1 hi ho w | ➡ | w ho hi -1 |

- Reversing array elements:

exchange

| a | b | c | d | e |

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#3

Solution: Reverse Array of Strings

```
var items = Console.ReadLine().Split(' ').ToArray();
for (int i = 0; i < items.Length / 2; i++)
{
    var oldElement = items[i];
    items[i] = items[items.Length - 1 - i];
    items[items.Length - 1 - i] = oldElement;
}

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

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1202#3
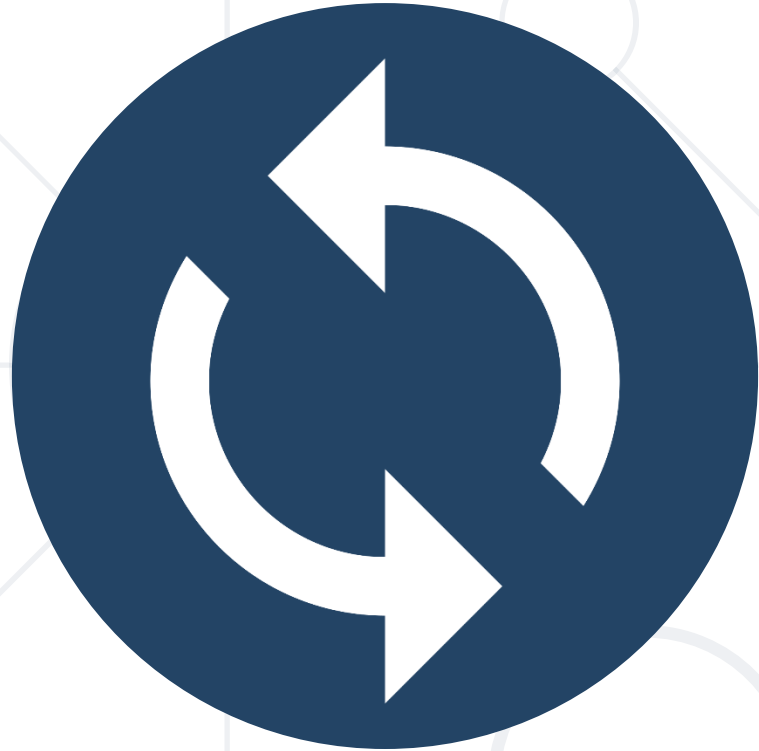
Iterate Through Collections

Foreach Loop

Foreach Loop

- **Iterates** through all elements in a collection

- Cannot access the current index
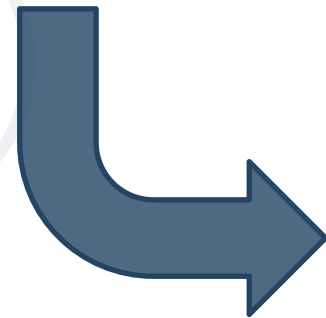
- Read-only

```
foreach (var item in collection)
{
    // Process the value here
}
```

Print an Array with Foreach

```csharp
int[] numbers = { 1, 2, 3, 4, 5 };
foreach (int number in numbers)
{
    Console.Write($"{number} ");
}
```
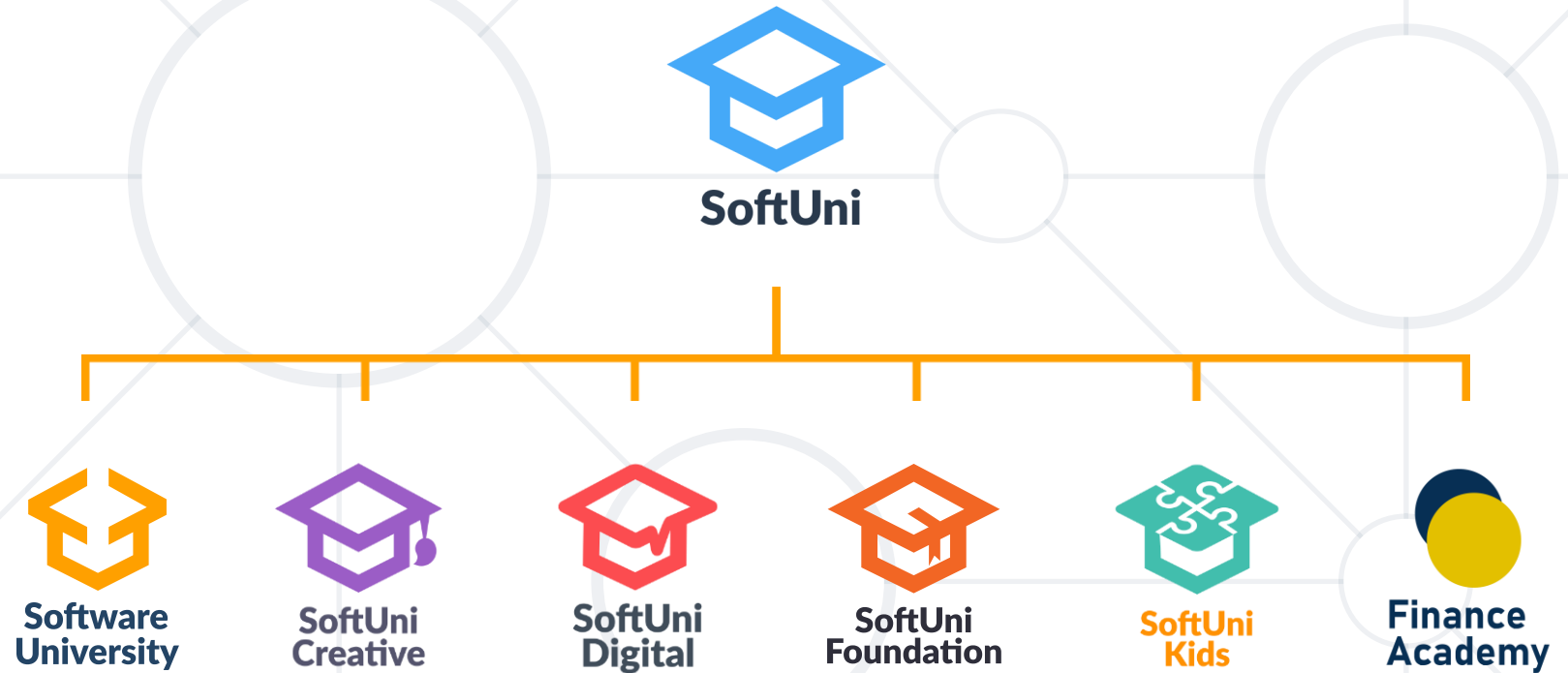
1 2 3 4 5

Summary

- Arrays hold a **sequence** of elements
    - Elements are numbered from **0** to `length-1`
- Creating (allocating) an array
- Accessing array elements by **index**
- Printing array elements

Questions?

SoftUni Diamond Partners

Trainings @ Software University (SoftUni)

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

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**

- Unauthorized copy, reproduction or use is illegal

- © SoftUni – https://about.softuni.bg/

- © Software University – https://softuni.bg