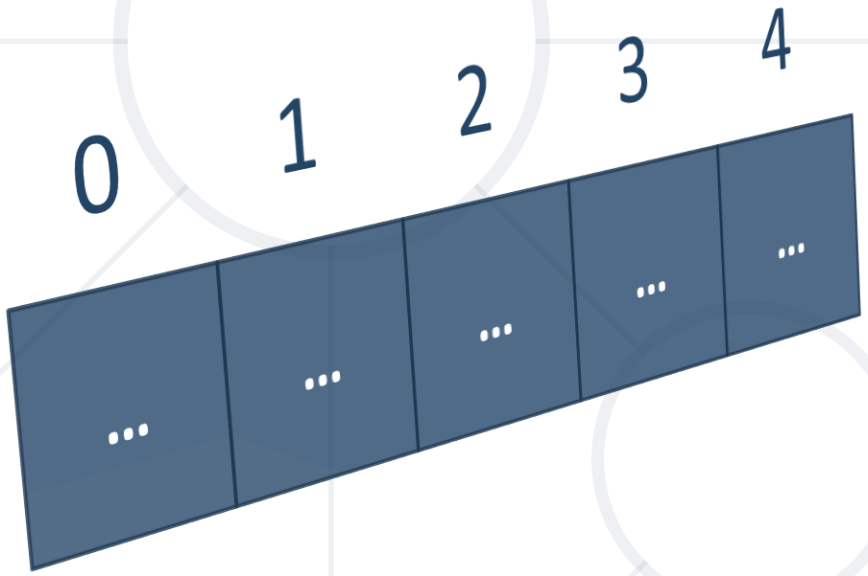
Lists

Processing Variable-Length Sequences of Elements



SoftUni Team

Technical Trainers

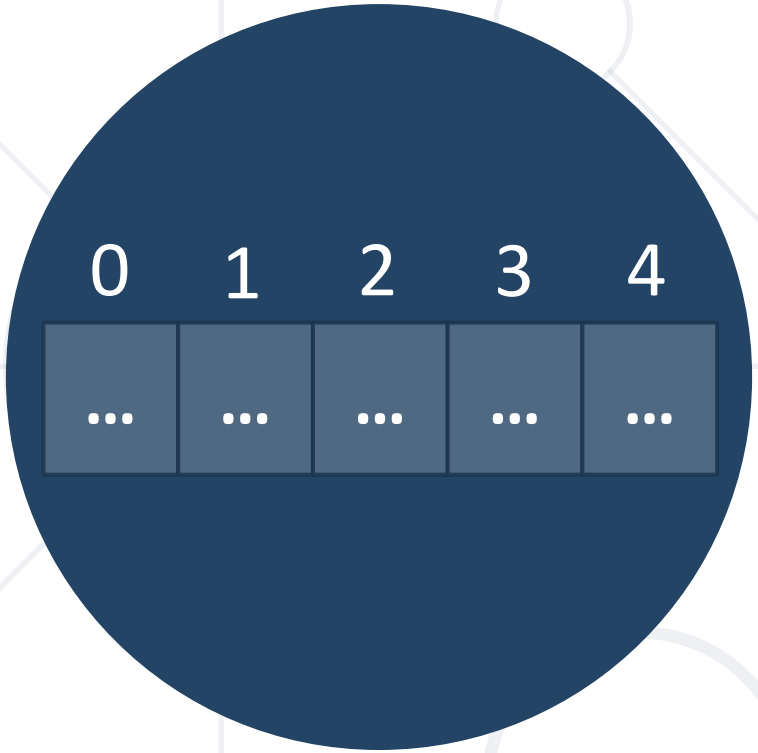
Software University

SoftUni

Software University

https://softuni.bg

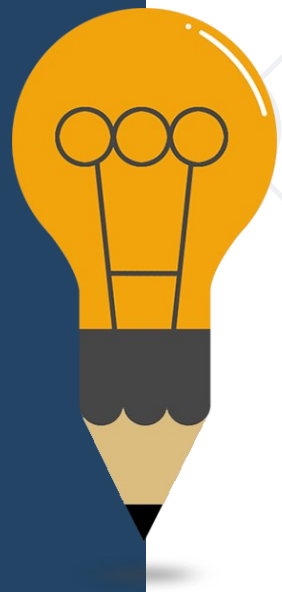
Table of Contents



Lists

List<T> – Overview

- `List<T>` holds a list of elements of the same type



```
List<string> names = new List<string>();
// Create a list of strings
names.Add("Peter");
names.Add("Maria");
names.Add("George");
// Add elements
foreach (var name in names)
  Console.WriteLine(name);
Console.WriteLine(string.Join(", ", names));
// Print elements
```

List<T> – Basic Methods

- Provides operations to **add** / **insert** / **remove** / **find** elements:
  - `Add(element)` – adds an element to the List<T>
  - `Count` – number of elements in the List<T>
  - `Remove(element)` – removes an element (returns true / false)
  - `RemoveAt(index)` – removes an element at a certain index
  - `Insert(index, element)` – inserts an element to a given index
  - `Contains(element)` – determines whether an element is in the list
  - `Sort()` – sorts the array/list in ascending order

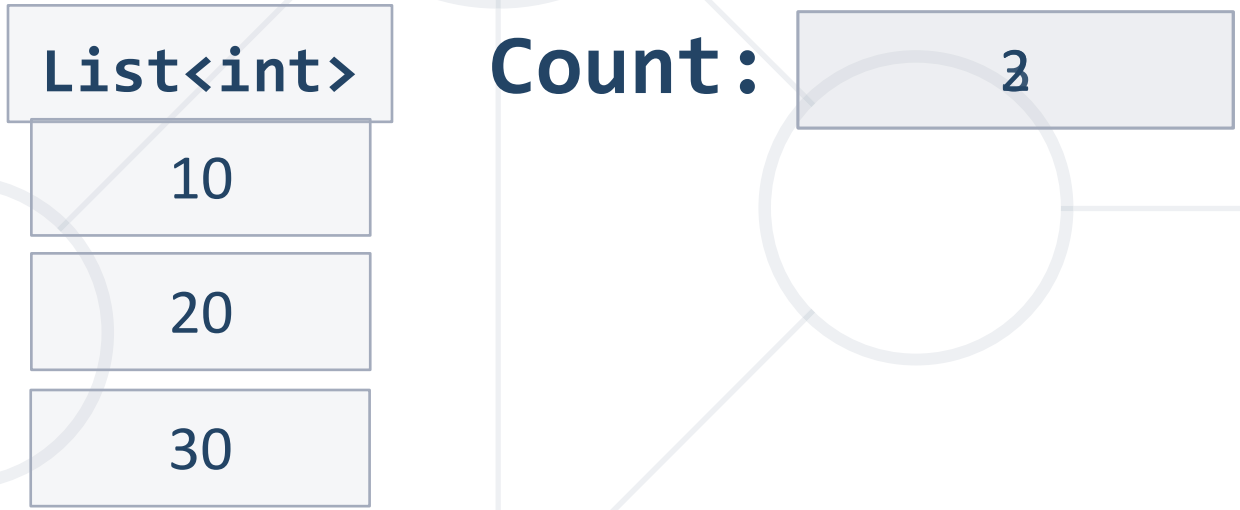
Add() – Appends an Element

- We create an empty **List** and we **add** several elements
- The **Count** increases each time we add an element

`List<int>` **Count:** 3

Remove() – Deletes an Element

- We **remove** an element from the **List**
- The **Count** decreases each time we remove an element



Insert() – Inserts an Element at Position

-10

- We **insert** an element at index 1
- Other elements **indices** are changed upon insertion

List<T> – Basic Methods Example

```
List<int> nums = new List<int> { 10, 20, 30, 40, 50, 60 };
nums.Remove(30);
nums.Add(100);
nums.Insert(0, -100);
Console.WriteLine(string.Join(", ", nums));
Console.WriteLine($"Count: {nums.Count}");
```

```
-100, 10, 20, 40, 50, 60, 100
Count: 7
```

Using for Loop or String.Split()

Reading Lists from the Console

- First, read from the console the list **length**:

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

- Next, create a list of a given size **n** and read its **elements**:

```
List<int> list = new List<int>();
for (int i = 0; i < n; i++)
{
    int number = int.Parse(Console.ReadLine());
    list.Add(number));
}
// The list now holds: {10, 20, 30, 40, 50}
```

```
5
10
20
30
40
50
```

Reading List Values from a Single Line

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

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

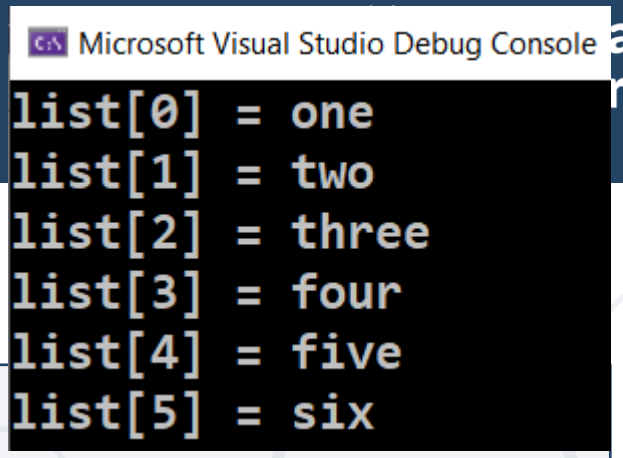
```
string values = Console.ReadLine();
List<string> items = values.Split(' ').ToList();
List<int> nums = new List<int>();
for (int i = 0; i < items.Count; i++)
  nums.Add(int.Parse(items[i]));
```

Convert a collection into List

```
List<int> items = Console.ReadLine()
  .Split(' ').Select(int.Parse).ToList();
```

Read a List of integers

Printing Lists On the Console



- Printing a list using a **for**-loop:

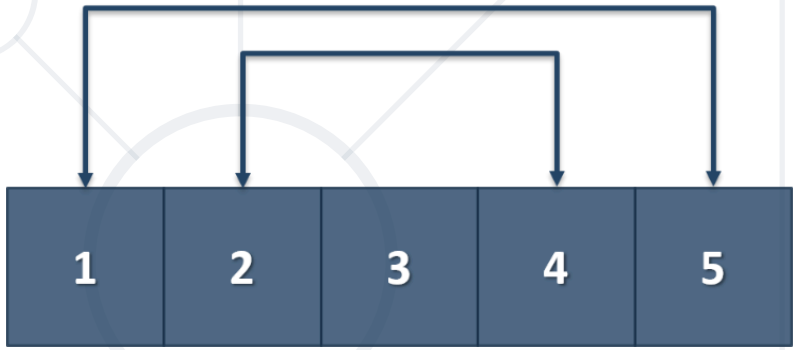
```
List<string> list = new List<string>() {
  "one", "two", "three", "four", "five", "six"};
for (int i = 0; i < list.Count; i++)
  Console.WriteLine("list[{0}] = {1}", i, list[i]);
```

- Printing a list using a **string.Join(…)**:

```
List<string> list = new List<string>() {
  "one", "two", "three", "four", "five", "six"};
Console.WriteLine(string.Join("; ", list));
// Output: one; two; three; four; five; six
```

Problem: Gauss' Trick

- Write a program that **sums all numbers** in a **list** in the following order:
  - first + last, first + 1 + last - 1, first + 2 + last - 2, … first + n, last - n
- Examples:



1 2 3 4 5	6 6 3
1 2 3 4	5 5

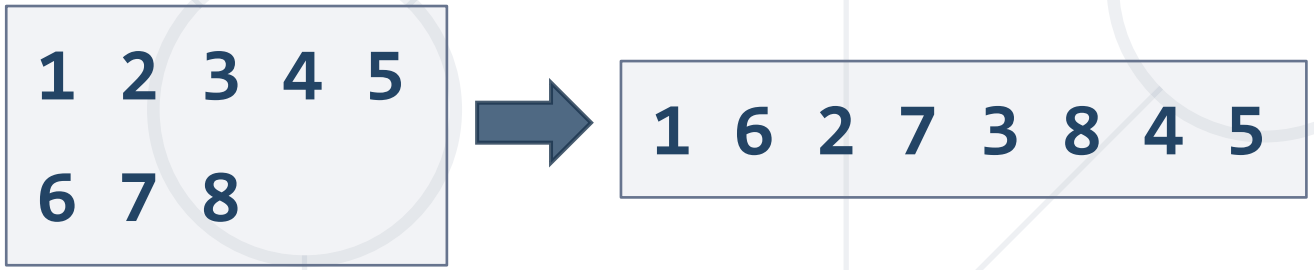
Solution: Gauss' Trick

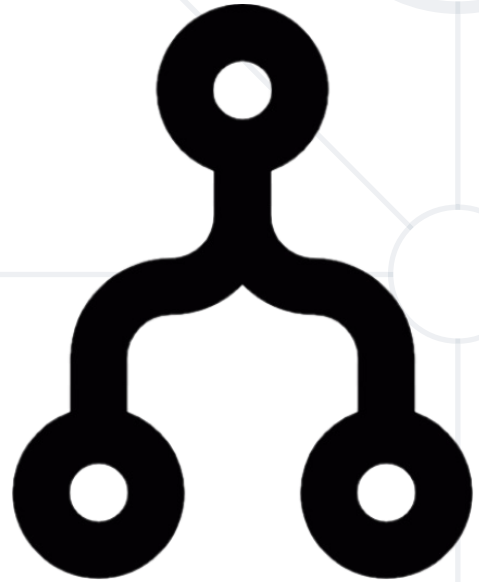
```
List<int> numbers = Console.ReadLine()
   .Split()
   .Select(int.Parse)
   .ToList();
int originalLength = numbers.Count;
for (int i = 0; i < originalLength / 2; i++)
{
   numbers[i] += numbers[numbers.Count - 1];
   numbers.RemoveAt(numbers.Count - 1);
}
Console.WriteLine(string.Join(" ", numbers));
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3171#11

Problem: Merging Lists

- You receive **two lists with numbers**. Print a **result list**, which contains the numbers from both of the lists.
  - If the **lengths of the two lists** are **not equal**, just add the remaining elements at the end of the list:
  - list1[0], list2[0], list1[1], list2[1], …





Solution: Merging Lists (1)

```
// TODO: Read the input
List<int> resultNums = new List<int>();
for (int i = 0; i < Math.Min(nums1.Count, nums2.Count); i++)
  // TODO: Add numbers in resultNums

if (nums1.Count > nums2.Count)
  resultNums.AddRange(GetRemainingElements(nums1, nums2));
else if (nums2.Count > nums1.Count)
  resultNums.AddRange(GetRemainingElements(nums2, nums1));

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

Solution: Merging Lists (2)

```
static List<int> GetRemainingElements(
  List<int> longerList, List<int> shorterList)
{
  List<int> nums = new List<int>();

  for (int i = shorterList.Count; i < longerList.Count; i++)
    nums.Add(longerList[i]);

  return nums;
}
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3171#12

Sorting Lists and Arrays

Sorting Lists

- **Sorting a list** == reorder its elements incrementally: **Sort()**
  - List items should be **comparable**, e. g. numbers, strings, dates, …

```
List<string> names = new List<string>()
 {"Peter", "Michael", "George", "Victor", "John" };
names.Sort();

Console.WriteLine(string.Join(", ", names));
// George, John, Michael, Peter, Victor
names.Sort();
names.Reverse();
Console.WriteLine(string.Join(", ", names));
// Victor, Peter, Michael, John, George
```

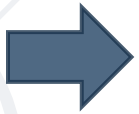
Sort in natural (ascending) order

Reverse the sorted result

Problem: List of Products

- Read a number **n** and n lines of **products**
  - Print a **numbered list** of all the products **ordered by name**
- Examples:

```
4
Potatoes
Tomatoes
Onions
Apples
```



```
1.Apples
2.Onions
3.Potatoes
4.Tomatoes
```

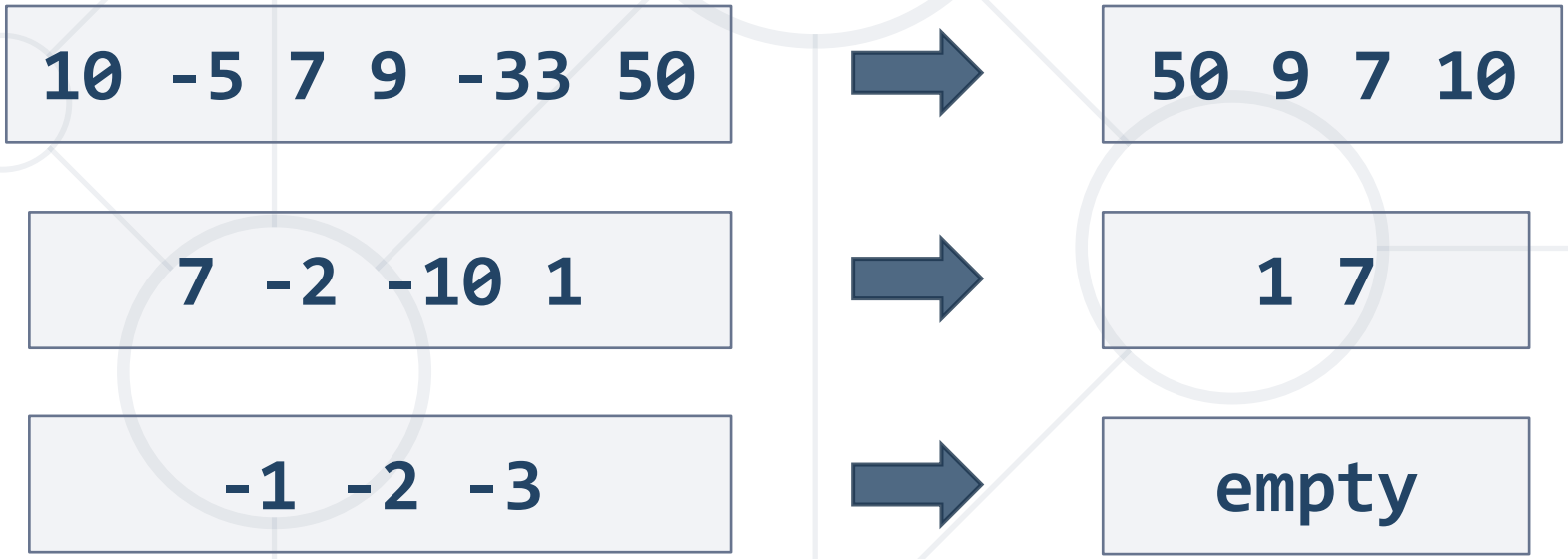
Solution: List of Products

```
int n = int.Parse(Console.ReadLine());
List<string> products = new List<string>();
for (int i = 0; i < n; i++)
{
  string currentProduct = Console.ReadLine();
  products.Add(currentProduct);
}
products.Sort();
for (int i = 0; i < products.Count; i++)
  Console.WriteLine($"{i + 1}.{products[i]}");
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3171#13

Problem: Remove Negatives and Reverse

- Read a **list of integers**, **remove all negative numbers** from it
  - Print the **remaining elements** in **reversed order**
  - In case of no elements left in the list, print "**empty**"



Solution: Remove Negatives and Reverse

```
List<int> nums = // TODO: Read the List from the console
for (int i = 0; i < nums.Count; i++)
  if (nums[i] < 0) { nums.RemoveAt(i--); }

nums.Reverse();
if (nums.Count == 0)
  Console.WriteLine("empty");
else
  Console.WriteLine(string.Join(" ", nums));
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3171#14

Summary

- **Lists** hold an editable sequence of elements (variable-length)
- Can **add** / **remove** / **insert** / **modify** elements at any time
- Creating (allocating) a list: `new List<T>()`
- Accessing list elements by index: `list[i]`
- Printing list elements: `string.Join(…)`

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

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