First Steps in Programming

Data Types, Variables, Conditional Statements

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg/

Table of Contents

1. What is Programming?
2. IDE and Visual Studio
3. Working with the Console
4. Writing Commands and Simple Operations
5. Variables, Expressions, Simple Calculations
6. Conditional Statements
7. Basic Debugging Techniques

Have a Question?

Software University

sli.do

#prgm-for-qa

What is Programming?

Introduction

What is Programming?

- Programming is the process of **giving instructions** to a computer to perform specific tasks or solve problems
- It involves **writing code** using a programming language, which serves as a **set of commands** and instructions that the computer can understand and execute
- Programming involves creating **step-by-step algorithms** that outline the **logical sequence of operations** to achieve a desired outcome

What is a Programming Language?

- A programming language is a **set of instructions** that allows humans to communicate with computers
- It serves as an **intermediary between human logic and machine operations**, enabling programmers to write code to perform specific tasks
- Each programming language has its **unique syntax**, which defines the rules and structure for writing code

What is a Computer Program?

- A computer program is a **set of instructions** written in a programming language that directs a computer to perform specific tasks or operations
- These instructions are **executed sequentially or conditionally**, and the program's logic determines how the computer responds to different inputs
- Computer Programs are written in text format:
  - The text of the program is called **source code**
- The source code is compiled into an executable file:
  - For example: **Program.cs** is compiled to **Program.exe**

Integrated Development Environment

Visual Studio

IDE – Integrated Development Environment

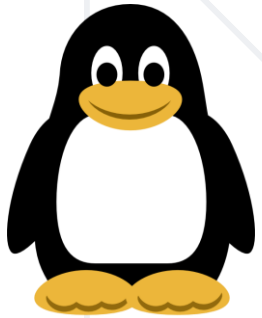
- An **Integrated Development Environment** (**IDE**) is used to streamline and enhance the software development process by providing programmers with tools for coding, debugging, testing, and project management, all within a single unified interface
- An Integrated Development Environment (IDE) is a software application that provides programmers with a comprehensive and **user-friendly platform for developing software**

Visual Studio

- **Visual Studio** is a powerful and widely-used IDE that will be your primary tool throughout this course
- Visual Studio is a development environment for the C# programming language
- Install Visual Studio on your computer:
  - Installation guide

Visual Studio - Platforms

- Visual Studio is available on **multiple platforms**, making it accessible to a wide range of developers
  - Windows
  - MacOS
  - Linux

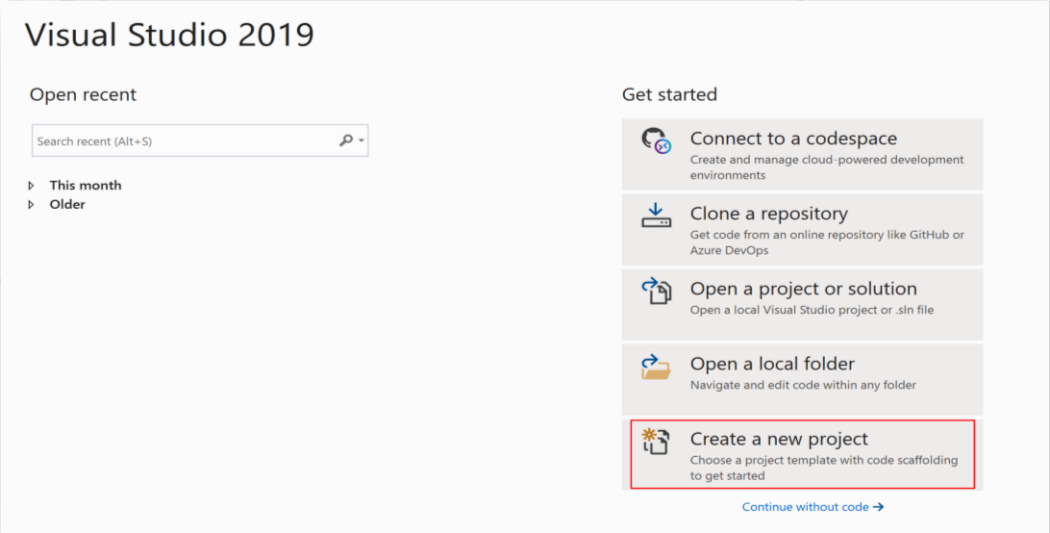


First Console Program

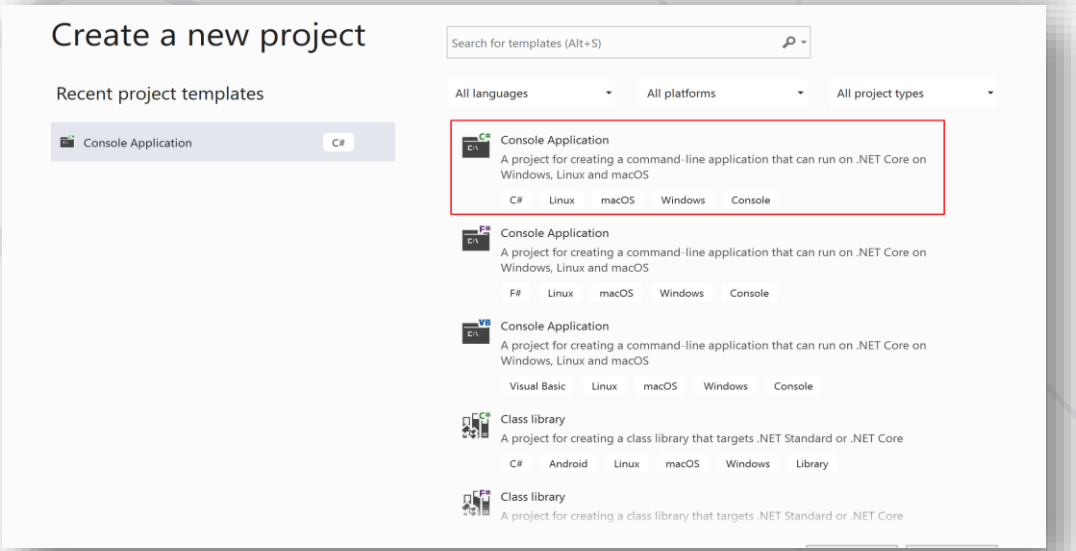
Hello, World!

Creating a Console Program (1)

- Start Visual Studio
- Select Create a new project

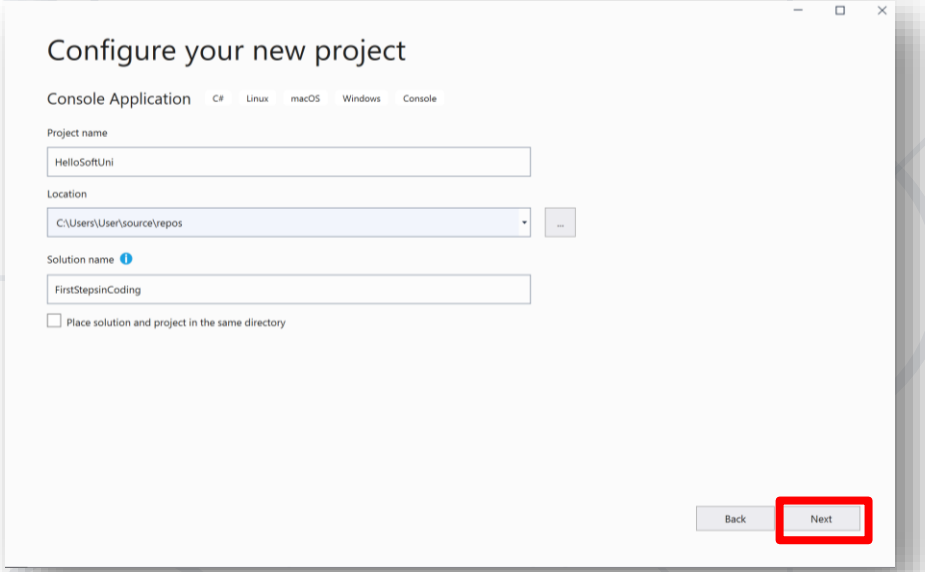


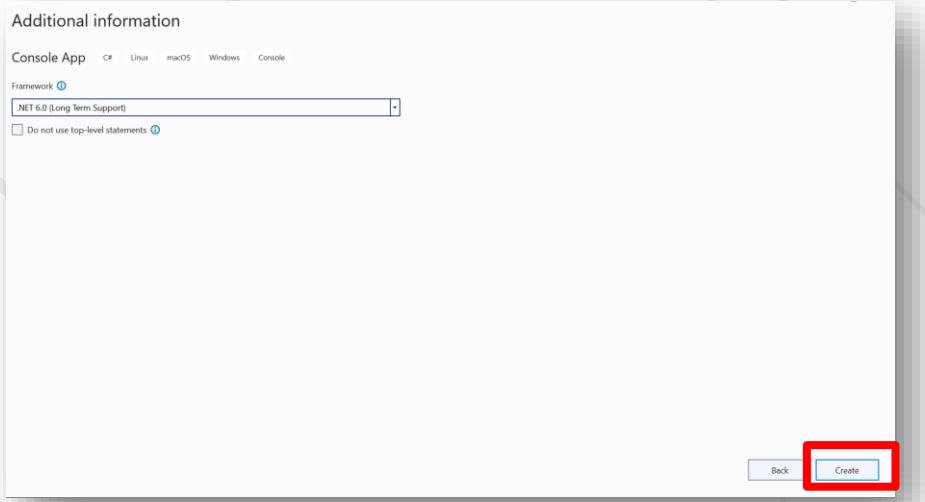
- Select Console Application
- Click Next



Creating a Console Program (2)

- Enter a **proper project name** and **choose** the desired **directory for the new project**
- Choose **Next**
- **Select the .NET 6.0 framework**
- Choose **Create**





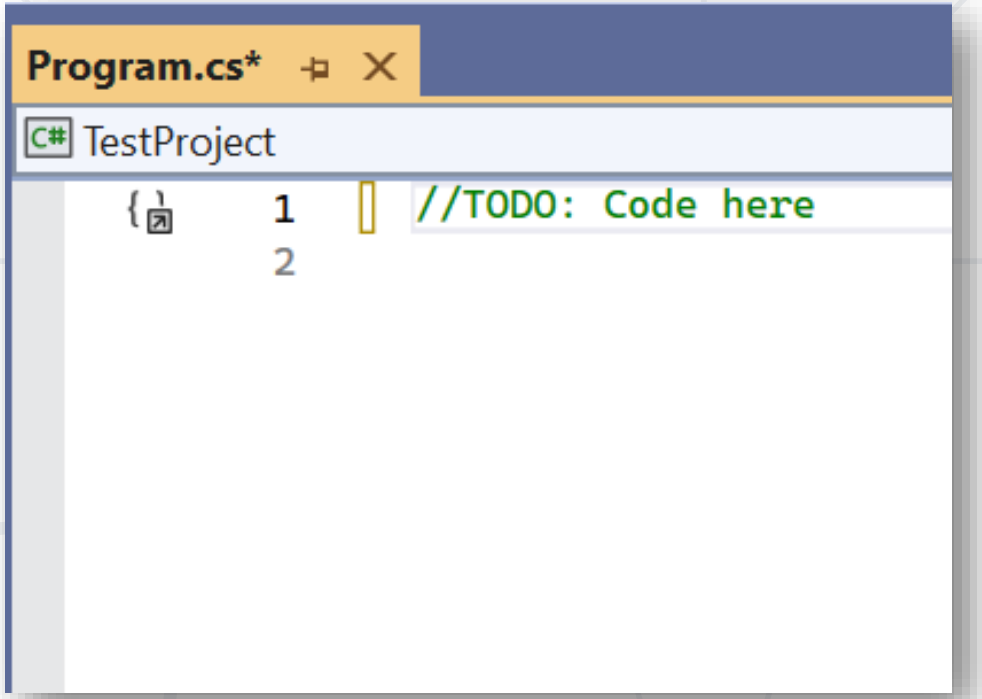
Writing Source Code (1)

- We **used to** write the source code in the following section, so far: `Main(string[] args)`
  - Between the curly braces `{ }`
- Hit [Enter] after the opening bracket `{`
- The source code should be written **indented**

```
namespace HelloSoftUni
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            // TODO: Code here
        }
    }
}
```

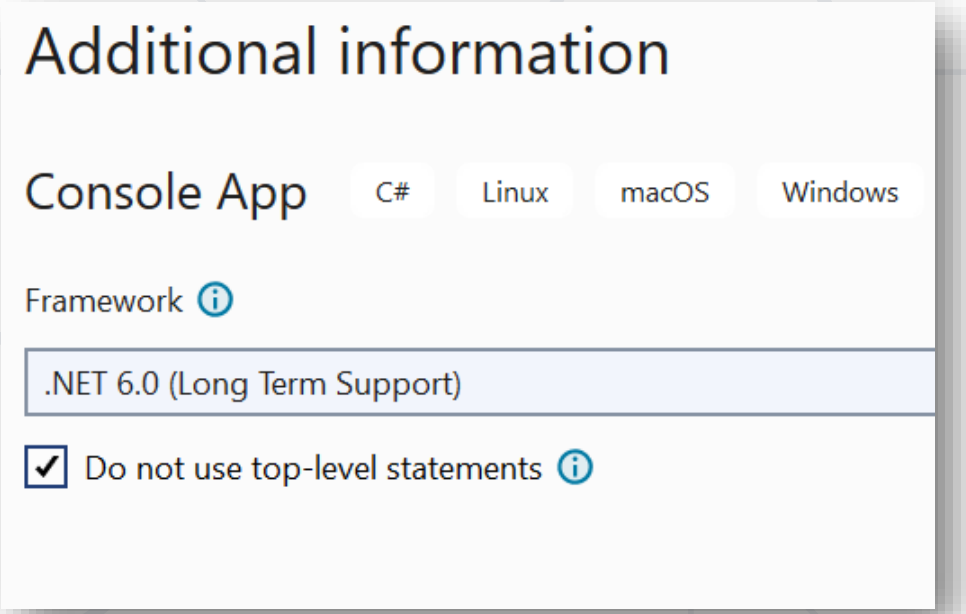
Writing Source Code (2)

- In **.NET 6.0**, a new approach simplifies your first console program, **using top-level statements**
- Instead of the traditional **Main** method, you start writing code directly, making it more intuitive
- Just start typing your code, and .NET 6.0 framework will handle the setup and the execution for you



Writing Source Code (3)

- While .NET 6.0 introduces the convenience of top-level statements, we'll begin by understanding the foundational concepts of **traditional C# programming**
- We'll start with the familiar structure of the **Main** method, encapsulating our code within it
- Starting **without top-level statements**, gives you a deeper appreciation for the structure and logic behind every line of code



Problem – First Console Program

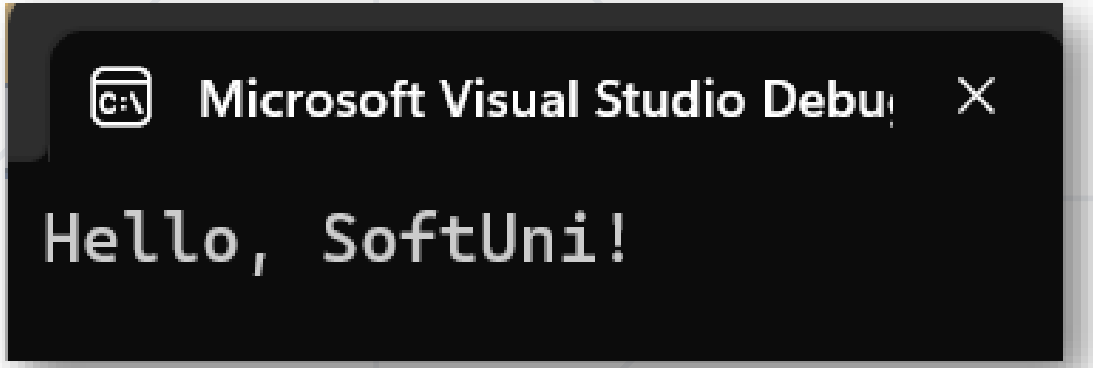
- Congratulations! It's time to write your very first C# program
- Locate the Main method, enclosed by curly brackets
- Inside the Main method, add the following code:

```
Console.WriteLine("Hello, SoftUni!");
```

```
namespace HelloSoftUni
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, SoftUni!");
        }
    }
}
```

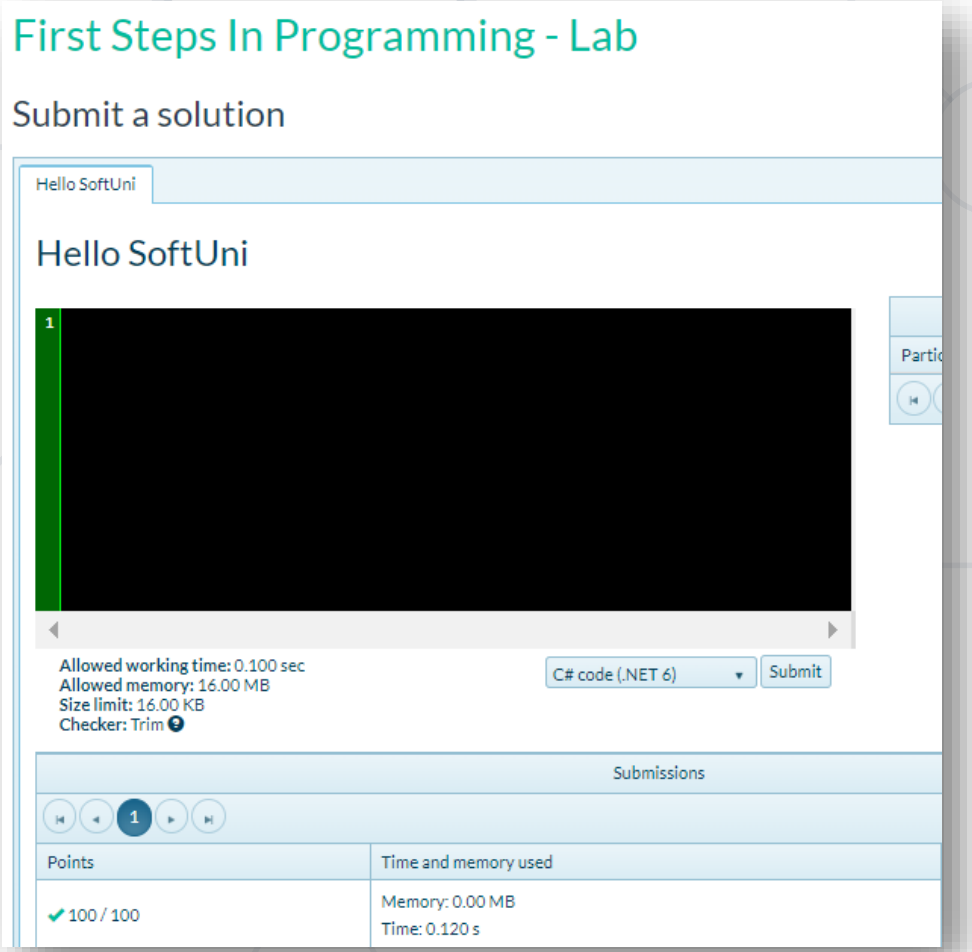
Build And Run The Program

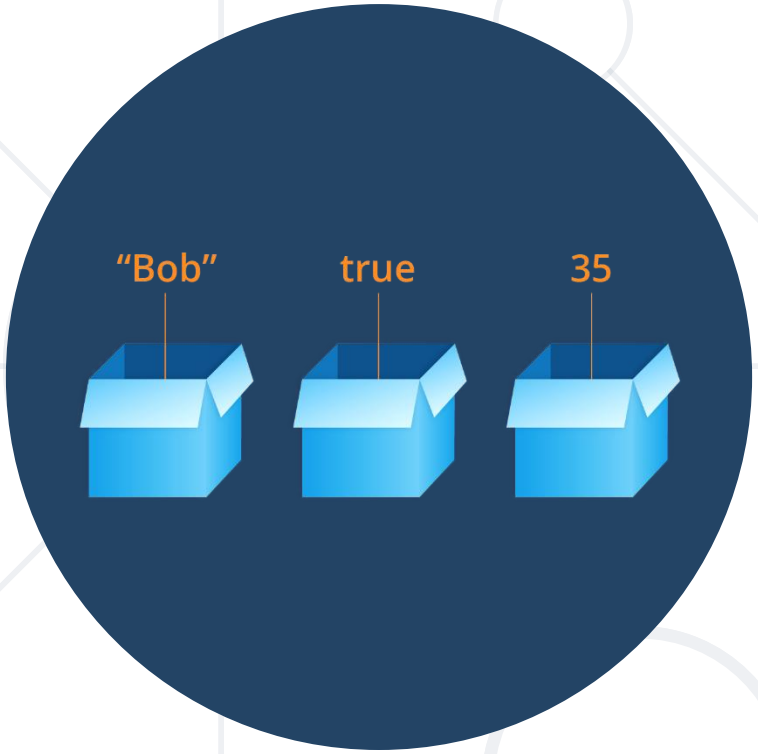
- Press **Ctrl + F5** to build and run the program
- If no exceptions are thrown, your program will execute
- The desired result will be displayed in the console:



Testing the Project

- Test your code online in the Judge system:





Variables & Data Types

Variables

- Computers are machines that process **data**
  - Data is stored in the computer memory using **variables**
  - Variables have **name**, **type** and **value**
- **Defining** a variable and **assigning** a value:

Type

Name of the variable

```
int count = 5;
```

Value

Data Types

- Variables store **values of a given type**
  - number, letter, text (string), date, color, image, list, …
- Data Types - examples:
  - `int` - integer: **1**, **2**, **3**, **4**, **5**, …
  - `double` – floating-point number: **0.5**, **3.14**, **-1.5**…
  - `string` – text: **"Hello"**, **"Hi"**, **"Car"**, …
  - `char` - character: **'A'**, **'#'**, **'@'**, **'+'**, …

Working with the Console

Reading Text

- Everything **received** from the Console, comes in **text format**
- Everything **printed** on the Console is **transformed into text**
- Command for reading from the console:

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

  - Returns the text entered by the user

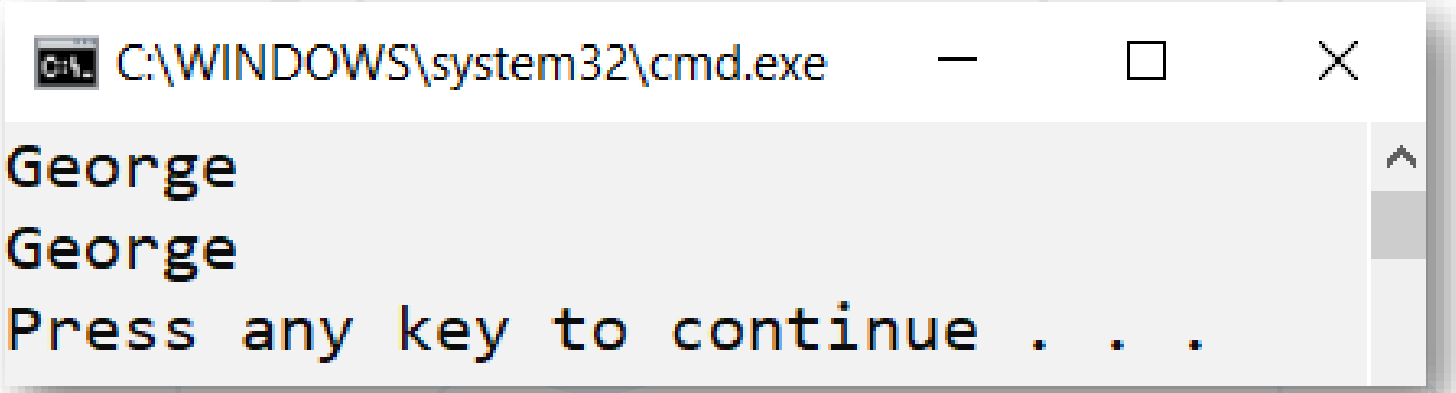
Problem - Text Reading

A program, that **reads** a name from the Console and **prints** it:

```
string name = Console.ReadLine();
Console.WriteLine(name);
```

Example input

Output



Reading Numbers

- Reading an integer value:

```
string input = Console.ReadLine();
int num = int.Parse(input);
```

- Example: calculating the area of a square with side length **a**:

```
int a = int.Parse(Console.ReadLine());
int area = a * a;
Console.WriteLine(area);
```

Reading a integer value on a single line

Reading floating point number

- Reading a floating-point number from the Console:

```
string input = Console.ReadLine();
double num = double.Parse(input);
```

- Example: convert inches to centimeters

```
double inches = double.Parse(Console.ReadLine());
double centimeters = inches * 2.54;
Console.WriteLine(centimeters);
```

Reading a floating point number on a single line

Expressions

Arithmetical operations: + and -

- Adding numbers (**operator +**):

```
int a = 5;
int b = 7;
int sum = a + b;  // 12
```

- Subtracting numbers (**operator -**):

```
int a = int.Parse(Console.ReadLine());
int b = int.Parse(Console.ReadLine());
int result = a - b;
Console.WriteLine(result);
```

Arithmetical operations: * and /

- Multiplication of numbers (**operator ***):

```
int a = 5;
int b = 7;
int product = a * b;  // 35
```

- Division of numbers (**operator /**):

```
int a = 25;
int b = a / 4;   // 6 – the fractional part is cut off
double c = a / 4.0;  // 6.25 – fractional division
int error = a / 0;   // Error: division by 0
```

Division of numbers - particularities

- When dividing integers, the result is **integer**:

```
int a = 25;
Console.WriteLine(a / 4);    // Integer result: 6
Console.WriteLine(a / 0);    // Error: division by 0
```

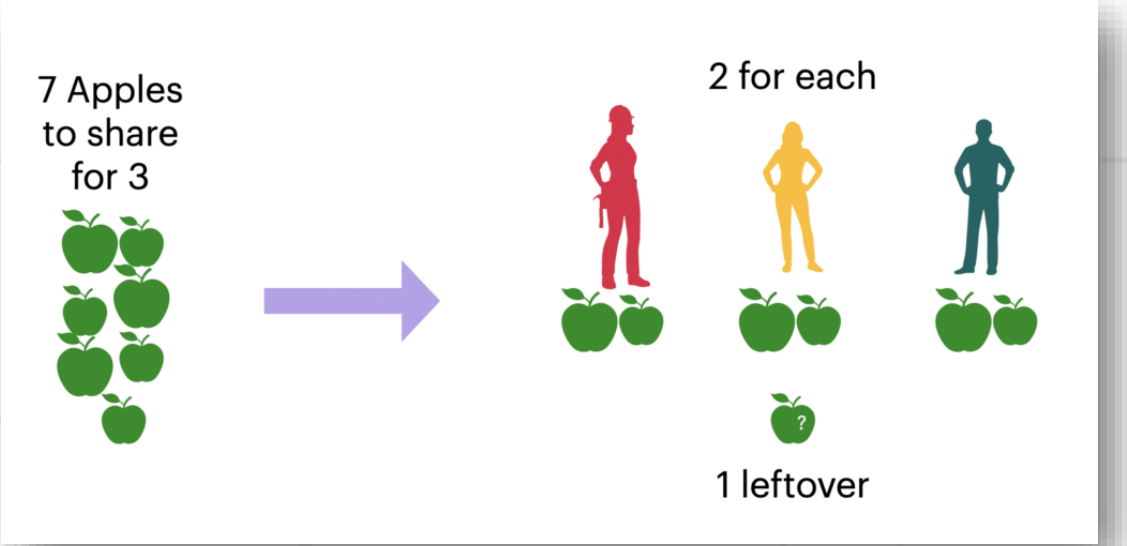
- When dividing floating point numbers, the result is **floating point number (double, decimal, float)**:

```
double a = 15;
Console.WriteLine(a / 2.0);    // Floating point: 7.5
Console.WriteLine(a / 0.0);    // Result: Infinity
Console.WriteLine(0.0 / 0.0); // Result: NaN
```

Arithmetical operations: %

- Modulo / remainder of integer division of numbers (**operator %**):

```
int a = 7;
int b = 3;
int product = a % b; // 1
```




```
int odd = 3 % 2;      // 1 - number 3 is odd
int even = 4 % 2;     // 0 - number 4 is even
int error = 3 % 0;    // Error: division by 0
```

Conditional Statements

if-else, switch case

Comparing Numbers

- In computer programming, we can compare values:

```
var a = 5;
var b = 10;
Console.WriteLine(a < b);                    // True
Console.WriteLine(a > 0);                    // True
Console.WriteLine(a > 100);                  // False
Console.WriteLine(a < a);                    // False
Console.WriteLine(a <= 5);                   // True
Console.WriteLine(b == 2 * a);               // True
```

Operator < (less than)

Operator > (greater than)

Operator <= (less than or equal)

Operator == (equal)

Comparing Operators

Operator	Designation	Can be used with
Equality check	==	numbers, text, date
Difference check	!=	
Greater than	>	numbers, date, other comparable types
Greater than or equal	>=	
Less than	<	
Less than or equal	<=	

- Example:

```
var result = (5 <= 6);
Console.WriteLine(result); // True
```

Simple Verifications

- In computer programming, we often **check conditions** and perform various actions based on the result of the check
  - **Example**: We enter a grade and check if it is excellent

```
double grade = double.Parse(Console.ReadLine());
if (grade >= 5.50)
{
   Console.WriteLine("Excellent!");
}
```

If-else conditional statements

- Enter a grade and check if it is excellent or not

```
double grade = double.Parse(Console.ReadLine());
if (grade >= 5.50)
{
   Console.WriteLine("Excellent!");
}
else
{
   Console.WriteLine("Not excellent.");
}
```

Curly braces { } after If / Else

- The curly braces { } initiate a block (group of commands)
  - Without the braces after **if** the following lines will be executed:

```
string color = "red";
if (color == "red")
  Console.WriteLine("tomato");
else
  Console.WriteLine("banana");
  Console.WriteLine("bye");
```

tomato
bye

```
string color = "red";
if (color == "red")
{
  Console.WriteLine("tomato");
}
else
{
  Console.WriteLine("banana");
  Console.WriteLine("bye");
}
```

tomato

Even or Odd – Problem example

- Verification if an integer value is even or odd:

```
int num = int.Parse(Console.ReadLine());
if (num % 2 == 0)
{
   Console.WriteLine("even");
}
else
{
   Console.WriteLine("odd");
}
```

The Greater Number – Problem example

- Write a program that reads **two integers** from the Console and outputs the greater of the two numbers
  - **Constraints:** Numbers should not be equal

```
int num1 = int.Parse(Console.ReadLine());
int num2 = int.Parse(Console.ReadLine());
if (num1 > num2)
  { Console.WriteLine("Greater number: " + num1); }
else
  { Console.WriteLine("Greater number: " + num2); }
```

Chain of Probabilities

- The construction **if-else-if-else…** might be in a sequence:
  - Example: write the names of the numbers in order (1 to 10)

```
int num = int.Parse(Console.ReadLine());
if (num == 1)
   { Console.WriteLine("one"); }
else if (num == 2)
   { Console.WriteLine("two"); }
else if (num == 3)
   { Console.WriteLine("three"); } // TODO: add more checks
else
   { Console.WriteLine("number out of range"); }
```

Conditional statement Switch-case

- **Switch-case** works just as the sequence **if-else-if-else** does
- **Example**: Print on the console the day of the week, depending on the input number (1…7)

```
int day = int.Parse(Console.ReadLine());
switch (day)
{
  case 1: Console.WriteLine("Monday"); break;
  case 2: Console.WriteLine("Tuesday"); break;
  …
  case 7: Console.WriteLine("Sunday"); break;
  default: Console.WriteLine("Error"); break;
}
```

Multiple conditions in Switch-case

- Write a computer program, that prints on the Console the type of the animal, depending on its name: dog → mammal; crocodile, tortoise, snake → reptile; others → unknown

```
switch (animal)
{
  case "dog": Console.WriteLine("mammal"); break;
  case "crocodile":
  case "tortoise":
  case "snake": Console.WriteLine("reptile"); break;
  default: Console.WriteLine("unknown"); break;
}
```

Debugging

Basic Debugging Techniques

Debugging

- The process of "attaching" to the program's execution allows us to trace the execution process, enabling us to detect errors in the program (bugs)

```
1   using System;
2
3   class Program
4   {
5       static void Main()
6       {
7           var number = int.Parse(Console.ReadLine());
8
9           number += number;   ≤ 4,319ms elapsed
10          Console.WriteLine(number);
11      }
12  }
13
```

84 %

Locals

Name	Value
System.Console.ReadLine returned	"10"
int.Parse returned	10
number	10

Debugging in Visual Studio

- Pressing [F10] will start the program in debug mode
- We can proceed to the next step using [F10]
- We can create breakpoints with [F9]
  - We can reach them directly using [F5]

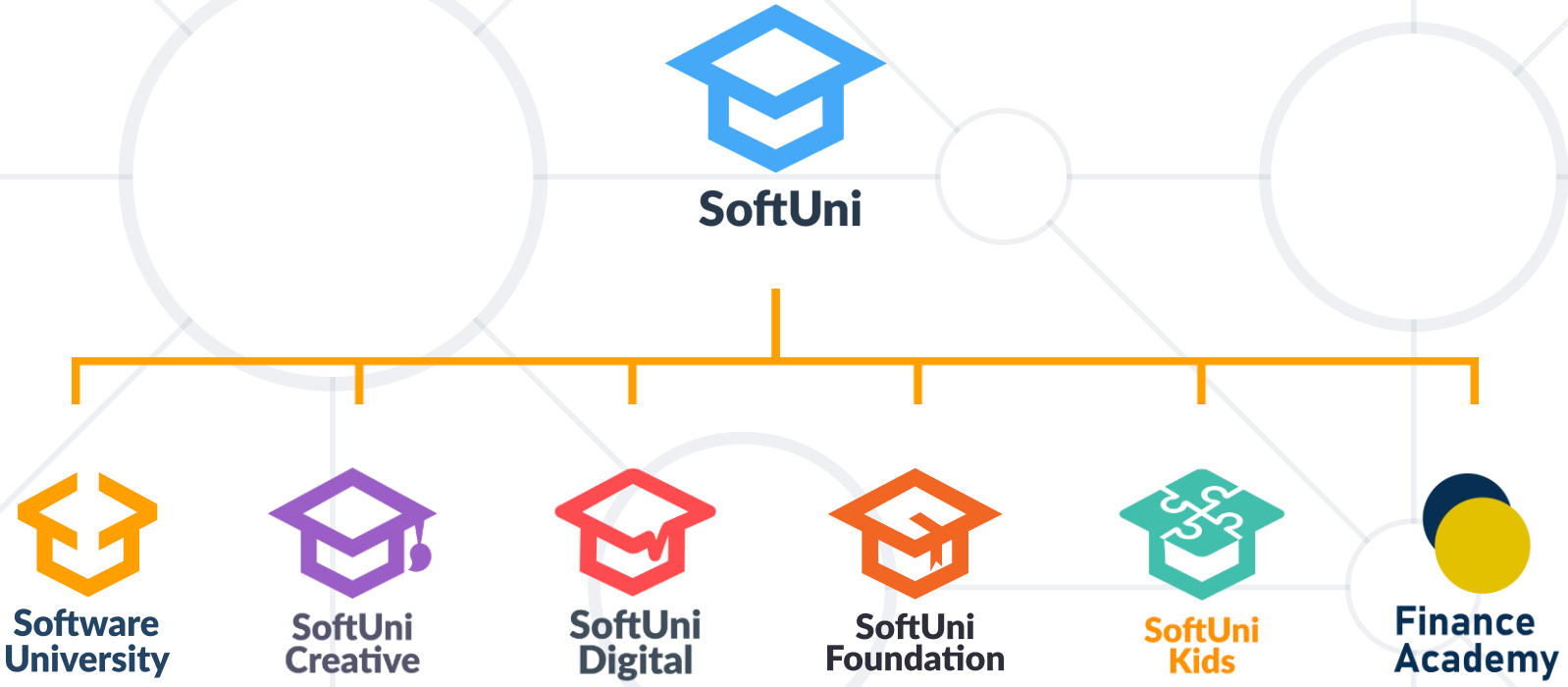
Summary (1)

- Explored the **concept of Programming** and its significance
- Familiarized ourselves with **IDE**s and the role of **Visual Studio**
- Practiced working with the **Console** for **input** and **output**
- Learned how to write and **execute Commands** and **Simple Operations**

Summary (2)

- Explored **Variables**, **Expressions** and performed **Simple Calculations**
- Gained the understanding of **Conditional Statements** and decision making
- Acquired basic **Debugging Techniques** for identifying and fixing errors

Questions?



SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

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