Conditional Statements

The "**if-else**" Statement

if-else

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#prgm-for-qa

Table of Contents

Review

Variables, Data Types, Expressions and Statements

Variables

- **Variables** hold data of certain type and allow:
  - **Storing** data in named memory location
  - **Retrieving** the stored data
  - **Modifying** the stored data
- Declaring, initializing, reading and changing a variable:

```
int age = 5;
```

```
age = age + 1;
```

```
Console.WriteLine(age);
```

Data Types

- **Data types** define **ranges / domains of values**
  - Integer number – `int` – **5**, **0**, **120**, **-250**)
  - Floating-point number – `double` – **3.14159**, **0.5**
  - Boolean (true / false) – `boolean` – **true** or **false**)
  - Unicode characters (letters) – `char` – **'X'**, **'#'**, **'\n'**
  - Unicode strings (text) – `string` – **"Hello Java"**
  - Date / date and time – **24-May-2019 11:38**
  - Complex data types – **arrays, lists, maps, classes**

Expressions

- **Expressions** == **variables** and **values**, combined with **operators**

(a+b)
*c - 1

2 is a literal value expression

b * 2 + 1 is an arithmetic expression

```
a = b * 2 + 1;
```

b is a variable expression

a = b * 2 + 1 is an assignment expression

Statements

- **Statements** == commands / actions to be executed

Get the current value stored in **b**

Multiply that value by **2**

```
a = b * 2;
```

Store the result back into another variable called **a**

Method call statement

```
Console.WriteLine(a * 2);
```

Conditional Statements in the Real Life

Real Life Example: Watering Plants

- **If** it is raining:
  - I shall skip watering the plants in the garden
- **Else**:
  - I will have to water them

```
if (humidity > 90%)
  Console.WriteLine("Rain -> skip watering");
else
  Console.WriteLine("No rain -> water the plants");
```

Logical Expressions

Comparison Operators

Comparison Operators

Operators	Designation
Equal to	==
Not Equal to	!=
Greater than	>
Greater than or equal to	>=
Less than	<
Less than or equal to	<=

Value Comparison

- In programming we can **compare** values
  - The result of the logical expressions is either **true** or **false**

```
int a = 5;
int b = 10;
string str = "hi";
Console.WriteLine(a < b);        // true
Console.WriteLine(a > 100);      // false
Console.WriteLine(a <= 5);       // true
Console.WriteLine(b == 2 * a);   // true
Console.WriteLine(b != 2 * a);   // false
Console.WriteLine(str == "hi"); // true
```

Conditional Statements

Simple Conditions

Simple Conditions

- Check a **condition** and act according to the result

```
if (condition)
{
  // Code to execute when
  // the condition is true
}
```

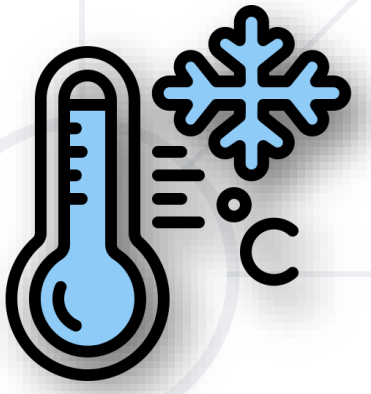
Boolean expression

```
int size = …;
if (size > 100)
{
  size = 100;
}
```

- The result is either `true` or `false`
- The code block { } may hold one or multiple commands

Problem: Freezing Weather

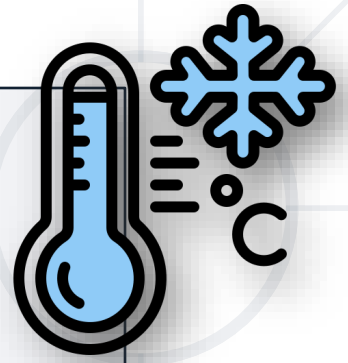
- Write a program to **check for freezing water**, which:
  - Reads a temperature in Celsius
  - **Checks** whether the temperature is **below** zero
  - Prints "**`Freezing weather!`**", if the temperature is equal or smaller than 0, otherwise print nothing



`4`	→	*`(no output)`*
`-2`	→	`Freezing weather!`

Solution: Freezing Weather

```
double temperature =
  double.Parse(Console.ReadLine());
if (temperature < 0)
{
   Console.WriteLine("Freezing weather!");
}
```



Simple Conditions: if-else

- If the condition is **false**, we may execute another code, using the statement **else**

```
if (condition)
{
    // Condition is true
}
else
{
    // Condition is false
}
```

Blocks of Code: { … }

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

Block of 2 commands

Block of 3 commands

Single Line Statements

- The curly brackets **{ }** introduce a **block** (a group of commands)
- In case the **if** statement does **not** have curly brackets, only the code on the **next line** will be executed

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

Single line statement

Single line statement

Always executed

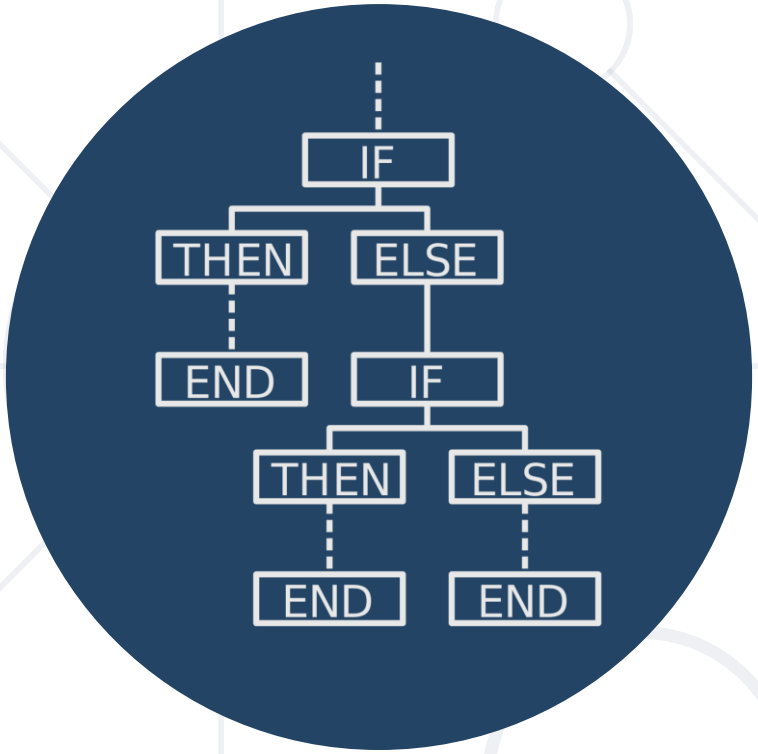
Problem: Even or Odd

- Write a program to **check for odd / even number**, which:
  - Reads an **integer**
  - If it's even, prints "**even**"
  - If it's odd, prints "**odd**"

4	→	even
7	→	odd

Solution: Even or Odd

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



Chain of Checks

Chain of Checks

- The `if-else` statement can be in a series

```
if (…)
  // Some code
else if (…)
  // Other code
else if (…)
  // Another code
else
  // Last code
```

If one condition is true, the program will NOT check the rest of the conditions

Chain of Conditions – Example

- The program checks the first condition, finds that it is true and ends

```
int a = 7;
if (a > 4)
  Console.WriteLine("Bigger than 4");
else if (a > 5)
  Console.WriteLine("Bigger than 5");
else
  Console.WriteLine("Equal to 7");
```

The output is just "Bigger than 4"

Problem: Number 1…9 as Words

- Write a program to **print a number as words**, which:
  - Reads an **integer** and checks its value [1 … 9]
  - Prints the value in the form of **English words**
  - If the number is out of range, prints "**Out of range**"

7	→	seven
10	→	Out of range
2	→	two

Solution: Number 1…9 as Words

```
int num = int.Parse(Console.ReadLine());
if (num == 1)
  Console.WriteLine("one");
else if (num == 2)
  Console.WriteLine("two");
else if (…) …
// TODO: Add the rest of the conditions
else
  Console.WriteLine("Out of range");
```

Variable Scope

Range of Use for the Variables

Variable Scope

- Variable **scope** == the **range of lines**, in which it can be used:

```
string currentDay = "Monday";
if (currentDay == "Monday")
{
  double salary = double.Parse(Console.ReadLine());
}
Console.WriteLine(salary); // Compile-time error!
```

The variable `salary` exists only in the block of code of the `if` statement

Debugging

Operations with the Debugger

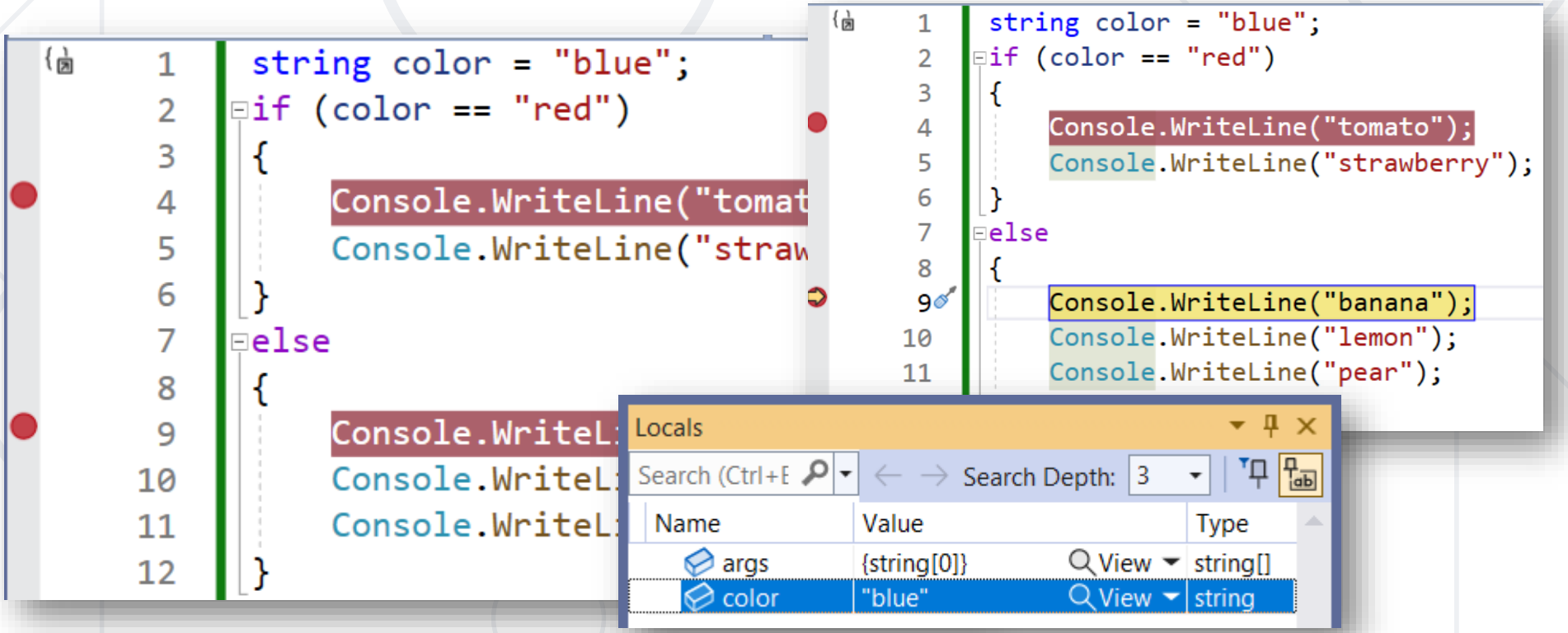
Debugging

- The process of **tracing** the code execution
  - Debugging allows finding defects (**bugs**)

```
1   int num = int.Parse(Console.ReadLine());
2
3   if (num == 1)
4       Console.WriteLine("one");
5   else if (num == 2)
6       Console.WriteLine("two");
7   else if (num == 3)   1ms elapsed
8       Console.W  num  6  "three");
9   else if (num == 4)
10      Console.WriteLine("four");
11  else if (num == 5)
12      Console.WriteLine("five");
```

Debugging in Visual Studio

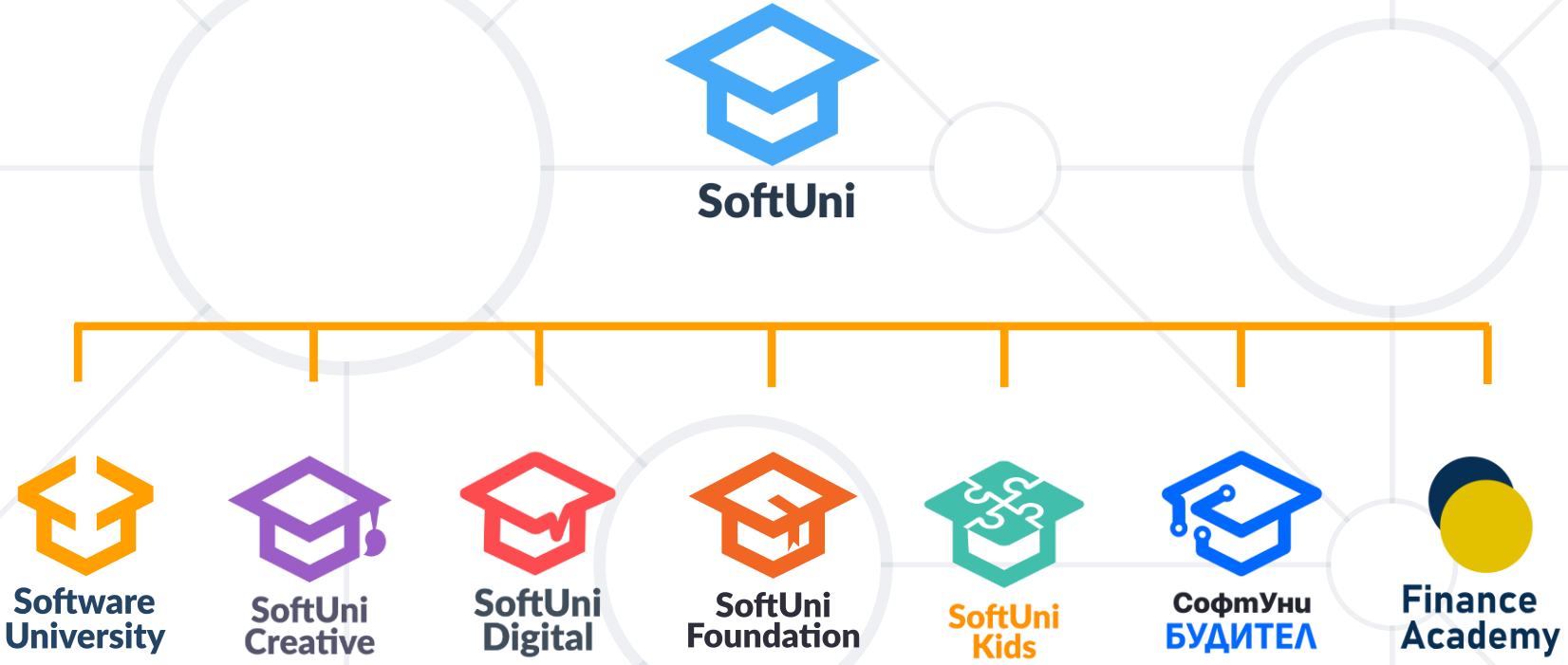
- Start the program in **debug mode**: press **[F5]**
- Go to the next step: press **[F10]**
- Add / remove **breakpoint**: press **[F9]**



Summary: Conditional Statements

- Logical Expressions
  - Comparison Operators: `<`, `>`, `==`, `!=`, …
- Conditional Statements (`if` and `if-else`)
- Chain of `if-else-if-else` Checks
- Blocks and Variable Scope
- Debugging and Breakpoints

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank
Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR
Serving the high achievers

SOFTWARE GROUP

PHAR VISION

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

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