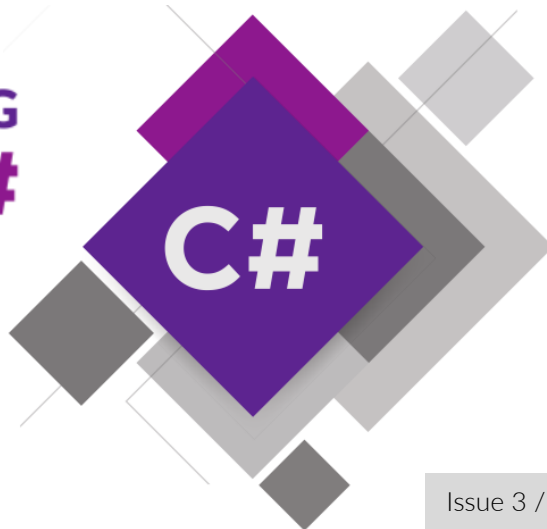


PROGRAMMING BASICS WITH C#



Issue 3 / 8

SIMPLE CONDITIONS

Learn about the if-else conditional statements, how to check conditions and how to implement conditional logic in your programs. Solve practical exercises.

Get an idea how to build simple Desktop GUI app “Multi-Currency Converter” in C# using Visual Studio.

Dr. Svetlin Nakov

Software University – <https://softuni.org>

Simple Conditions

In this mini book, we will discuss the **conditional statements in the C# language**, through which our program may have different effects, depending on a condition. We'll explain the syntax of conditional operators for checks (**if** and **if-else**) with appropriate examples and we will see in what range a variable (its **scope**) lives. Finally, we will go through **debugging** techniques to track the path that runs through our program during implementation.

Video: Overview

Watch the video about what we will learn in this mini book: <https://youtu.be/sstA00rIWkQ>.

Introduction to Simple Conditions by Examples

In programming we can **check conditions** and execute different blocks of code depending on the check. This is typically performed using the **if-else** constructs:

```
var size = decimal.Parse(Console.ReadLine());
if (size < 0)
    Console.WriteLine($"Negative size: {size}");
else if (size > 1000)
    Console.WriteLine($"Size too big: {size}");
else
{
    Console.WriteLine($"Size accepted: {size}");
    Console.WriteLine($"Area: {size * size}");
}
```

Run the above code example: <https://repl.it/@nakov/size-checker-if-else-csharp>.

When executed, the above code will **enter a decimal number** and will **check its value** several times. Depending on the above **conditions**, it will display different messages. Examples are shown below.

If we **enter -20** as input, the output will be as follows:

```
Negative size: -20
```

If we **enter 150** as input, the output will be as follows:

```
Size accepted: 150
Area: 22500
```

If we **enter 3200** as input, the output will be as follows:

```
Size too big: 3200
```

Let's explain in greater detail how to use **simple if-else conditions** in C#.

Comparing Numbers

In programming, we can compare values using the following **operators**:

- Operator `<` (less than)
- Operator `>` (greater than)
- Operator `<=` (less than or equals)
- Operator `>=` (greater than or equals)
- Operator `==` (equals)
- Operator `!=` (different from)

When compared, the result is a Boolean value **true** or **false**, depending on whether the result of the comparison is true or false.

Video: Comparing Numbers

Watch the video lesson about comparing numbers: <https://youtu.be/KTdqDWg7Wf8>.

Examples for Comparing Numbers

```
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
```

Note that when printing the **true** and **false** values in C # language, they are printed with a capital letter, respectively **True** and **False**.

Comparison Operators

In C #, we can use the following comparison operators:

Operator	Notation	Applicable for
Equals	<code>==</code>	numbers, strings, dates
Not equal	<code>!=</code>	
Greater than	<code>></code>	numbers, dates, other comparable objects
Greater than or equals	<code>>=</code>	
Less than	<code><</code>	
Less than or equals	<code><=</code>	

Example:

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

Simple If Conditions

In programming we often **check particular conditions** and perform various actions depending on the result of the check. This is done by **if** condition, which has the following structure:

```
if (condition)  
{  
    // condition body;  
}
```

Video: Simple If / If-Else Conditions

Watch the video lesson about the simple if-conditions: <https://youtu.be/M-G4nOaVDVA>.

Example: Excellent Grade

We read the grade from the console and check if it's excellent (≥ 5.50).

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

Test the code from the example locally. Try entering different grades, for example **4.75**, **5.49**, **5.50** and **6.00**. For grades **less than 5.50**, the program will not give any output, however if the grade is **5.50 or greater**, the output would be "Excellent!".

Testing in the Judge System

Test your solution from the example here:
<https://judge.softuni.org/Contests/Practice/Index/506#0>.

If-Else Conditions

The **if** construction may also contain an **else** clause to give a specific action in case the Boolean expression (which is set at the beginning **if (bool expression)**) returns a negative result (**false**). Built this way, the conditional statement is called **if-else** and its behavior is as follows: if the result of the condition is **positive (true)** – we perform some actions, when it is **negative (false)** – others. The format of the construction is:

```
if (condition)  
{  
    // condition body;  
}
```

```
else
{
    // else construction body;
}
```

Example: Excellent Grade or Not

Like the example above, we read the grade from the console and check if it's excellent, but this time we should **output the result in both cases**.

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

Testing in Judge System

Test your solution from the example here:
<https://judge.softuni.org/Contests/Practice/Index/506#1>.

About the Curly Braces {} After If / Else

When we have **only one command** in the body of the **if construction**, we can **skip the curly brackets**, indicating the conditional operator body. When we want to execute **block of code** (group of commands), curly brackets are **required**. In case we drop them, **only the first line** after the **if clause** will be executed.

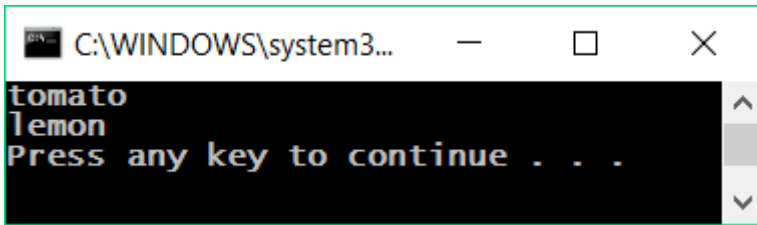


It's a good practice to **always put curly braces**, because it makes our code more readable and cleaner.

Here is an example where dropping curly braces leads to confusion:

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

Executing the above code will output the following console result:

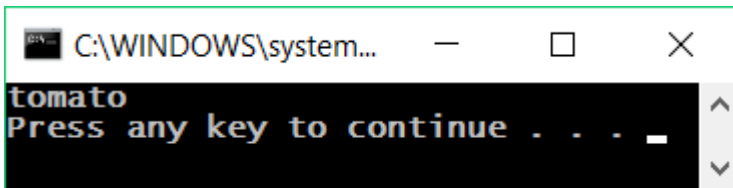


```
tomato
lemon
Press any key to continue . . .
```

With curly braces:

```
var color = "red";
if (color == "red")
{
    Console.WriteLine("tomato");
}
else if (color == "yellow")
{
    Console.WriteLine("banana");
    Console.WriteLine("lemon");
}
```

The following output will be printed on the console:



```
tomato
Press any key to continue . . .
```

If-Else Conditions – Examples

Now let's take a few examples (exercises) to learn how to use **if-else** conditional statements in practice.

Video: Examples of If-Else

Watch the video lesson about the "Even or Odd Number" and "The Larger Number" problems and their solutions: <https://youtu.be/qIUoKiObr-A>.

Example: Even or Odd Number

Write a program that checks whether an integer is **even** or **odd**.

Hint and Guidelines

We can solve the problem with one **if-else** statement and the operator **%**, which returns a **remainder by dividing** two numbers.

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

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#2>.

Example: The Larger Number

Write a program that reads from the console **two integers** and prints **the larger** of them. Print "Greater number: " + the bigger number.

The **input** comes as two numbers, each on a separate line. Sample input:

```
3
5
```

The **output** is a message like the shown below. Sample output:

```
Greater number: 5
```

Hint and Guidelines

Our first task is to **read** the two numbers from the console. Then, with a simple **if-else** statement, combined with the **operator for greater than (>)**, we do check. Part of the code is deliberately blurred, so you can test what you learned so far.

```
Console.WriteLine("Enter two integers:");
var num1 = int.Parse(Console.ReadLine());
var num2 = int.Parse(Console.ReadLine());
//if (num1 > num2)
//    Console.WriteLine("Greater number: " + num1);
//else
//    Console.WriteLine("Greater number: " + num2);
```

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#3>.

Variable Scope

Each variable has a range in which it exists, called **variable scope**. This range specifies where a variable can be used and how long is its **lifetime**. In the C# language, the scope in which a variable exists, starts from the line in which we **defined it** and ends with the first closing curly bracket **}** (of the method, the **if statement**, etc.).

Thus, it is important to know that **any variable defined inside the body of certain if statement will not be available outside of it**, unless we have defined it higher in the code.

Video: Variable Scope

Watch the video lesson about the variable scope: <https://youtu.be/J54TJBnY5vQ>.

Variable Scope – Example

In the example below, on the last line we are trying to print the variable **salary** that is defined in the **if statement**, we will get an **error** because we don't have access to it.

```
var myMoney = 500;
var payDayDate = 07;
var todayDate = 10;
if (todayDate >= payDayDate)
{
    var salary = 5000;
    myMoney = myMoney + salary;
}

Console.WriteLine(myMoney);
Console.WriteLine(salary); //Error!
```

The above code **will not compile**, because we are trying to access a variable **out of its scope**.

Sequence of If-Else Conditions

Sometimes we need to do a sequence of conditions before we decide what actions our program will execute. In such cases, we can apply the construction **if-else if ... -else** in series. For this purpose, we use the following format:

```
if (condition)
{
    // condition body;
}
else if (second condition)
{
    // condition body;
}
else if (third condition)
{
    // condition body;
}
```



```
    // condition body;
}
...
else
{
    // else construction body;
}
```

Video: Series of If-Else Checks

Watch the video lesson about the if-else checks: <https://youtu.be/PUvf7gtKSz4>.

Example: Digits in English

Print the digits in the range of 1 to 9 (digits are read from the console) in English. We can read the digit and then, through a **sequence of conditions** we print the relevant English word:

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

if (num == 1)
{
    Console.WriteLine("one");
}
else if (num == 2)
{
    Console.WriteLine("two");
}
else if (...)
{
    ...
}
else if (num == 9)
{
    Console.WriteLine("nine");
}
else
{
    Console.WriteLine("number too big");
}
```

The program logic from the above example **sequentially compares** the input number from the console with the digits from 1 to 9, when **each following comparison is being performed only in case the previous comparison is not true**. Eventually, if none of the **if** statements are true, the last **else** clause is performed.

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#4>.

Debugging – Simple Operations with Debugger

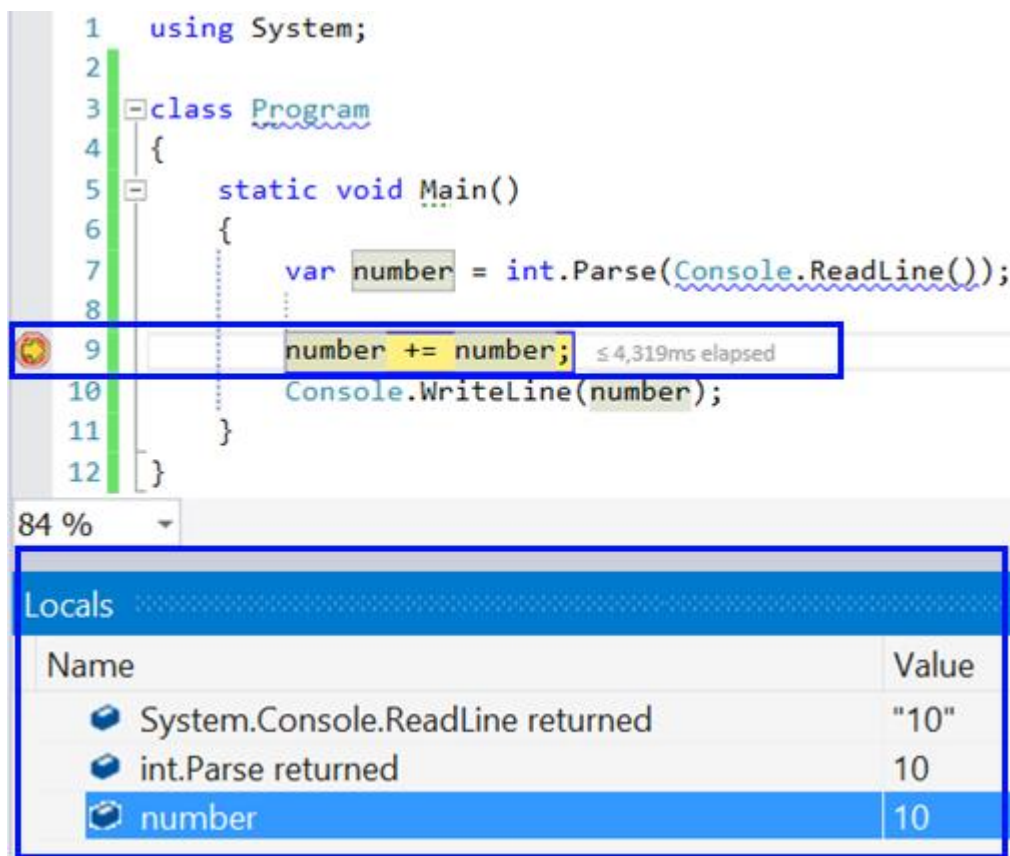
So far, we wrote a lot of code, and there were some mistakes in it, right? Now we will show a tool that can help us find mistakes more easily: the **debugger**.

Video: Debugging Code in Visual Studio

Watch the video lesson about debugging code in Visual Studio: <https://youtu.be/2asLYeJ6Tel>.

What is "Debugging"?

Debugging is the process of "attaching" to the program execution, which allows us to **track step by step the process**. We can track **line by line** what happens in our program, what path it follows, what are the values of defined variables at each step of debugging, and many other things that allow us to detect errors (**bugs**).



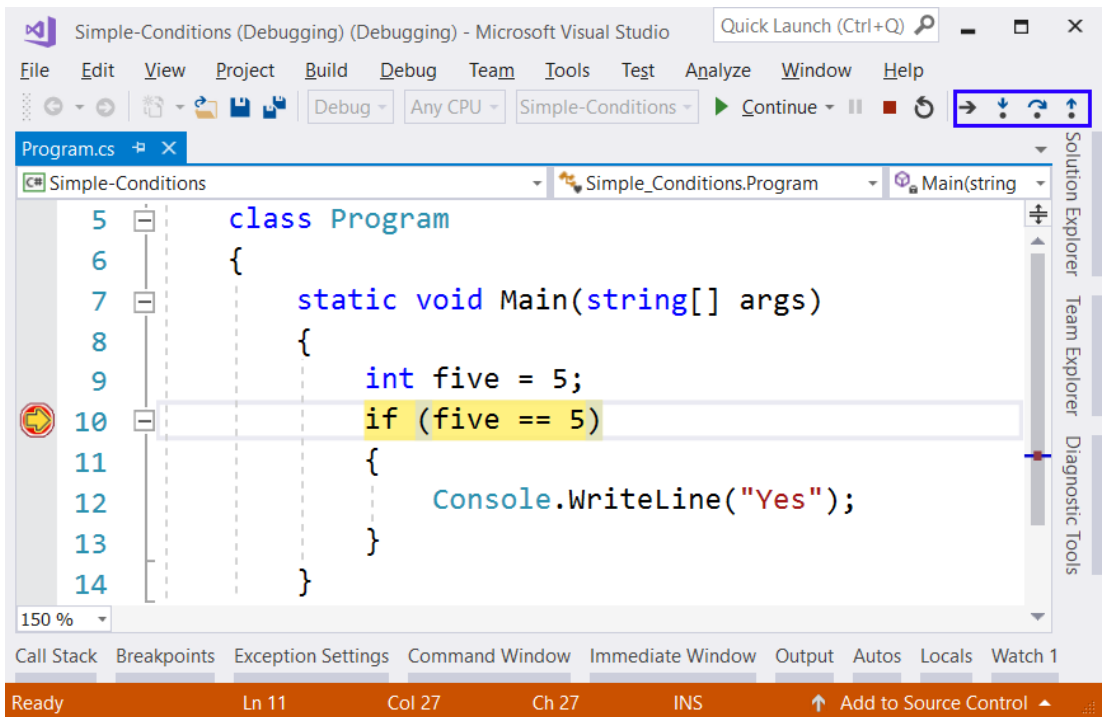
```
1 using System;
2
3 class Program
4 {
5     static void Main()
6     {
7         var number = int.Parse(Console.ReadLine());
8
9         number += number;
10        Console.WriteLine(number);
11    }
12 }
```

84 %

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

Debugging in Visual Studio

By pressing the [F10] button, we run the program in **debug mode**. We move to the next line again with [F10].



With [F9] we create the so-called **breakpoints**, that we can reach directly using [F5] when we start the program.

Exercises: Simple Conditions

Now let's **practice** the lessons learned about conditional statements **if** and **if-else**. We will solve a few **practical exercises**.

Video: Summary

Watch the following **video** to summarize what we learned: <https://youtu.be/mdv28HD5qes>.

What We Learned?

Let's summarize what we learned:

- Numbers can be **compared** by the **==**, **<**, **>**, **<=**, **>=** and **!=** operators:

```
Console.WriteLine(5 <= 10); // True
```

- Simple **if-conditions** check a condition and execute a code block if it is true:

```
if (a > 5)
{
    Console.WriteLine("The number `a` is bigger than 5");
}
```

- The **if-else construction** executes one of two blocks depending on whether a condition is **true** or **false**:

```
if (a > 5)
{
    Console.WriteLine("The number `a` is bigger than 5");
}
else
{
    Console.WriteLine("The number `a` is smaller or equal than 5");
}
```

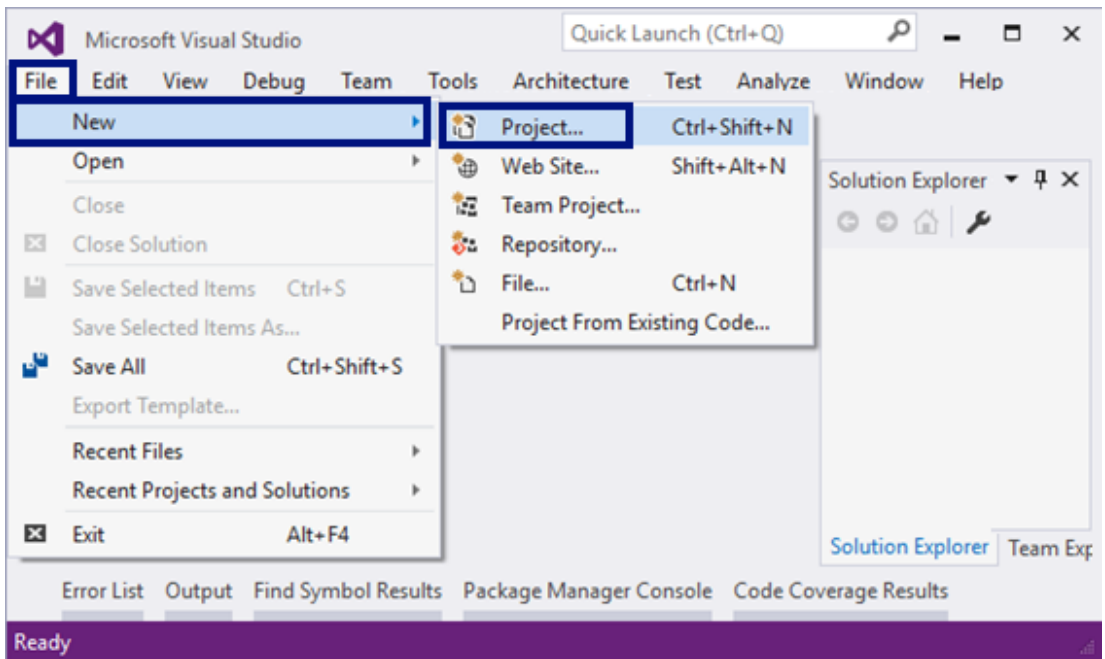
- If-else constructions can be chained as **if-else-if-else** sequences:

```
if (a > 100)
{
    Console.WriteLine("The number `a` is bigger than 100");
}
else if (a > 20)
{
    Console.WriteLine("The number `a` is bigger than 20");
}
else
{
    Console.WriteLine("The number `a` is smaller or equal than 20");
}
```

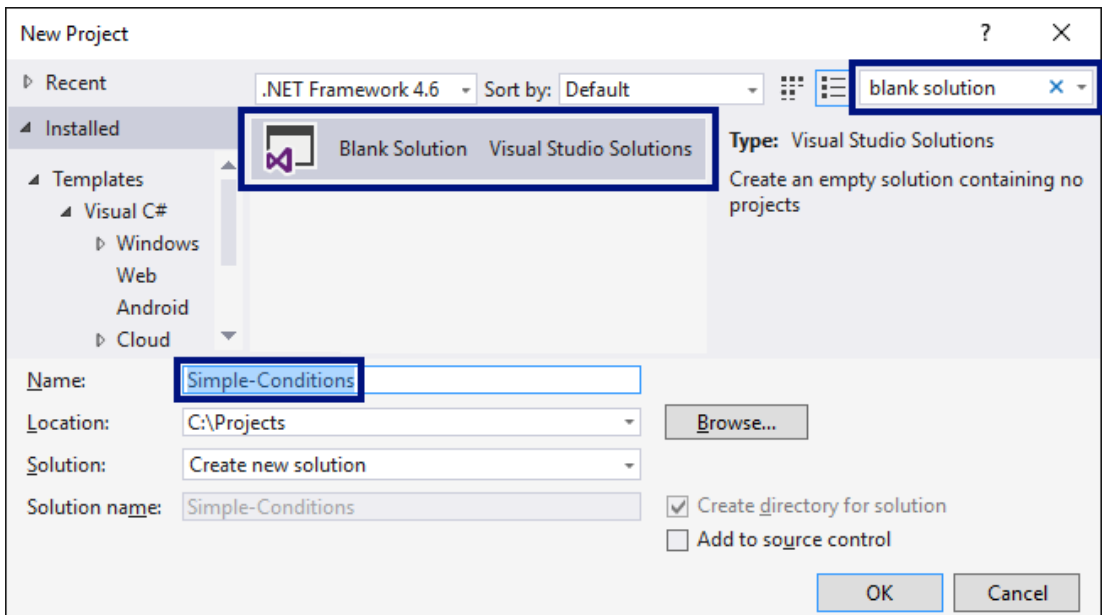
Empty Visual Studio Solution (Blank Solution)

At the start we create a **Blank Solution** in Visual Studio to organize better the task solutions from the exercise – each task will be in a separate project and all projects will be in a common solution.

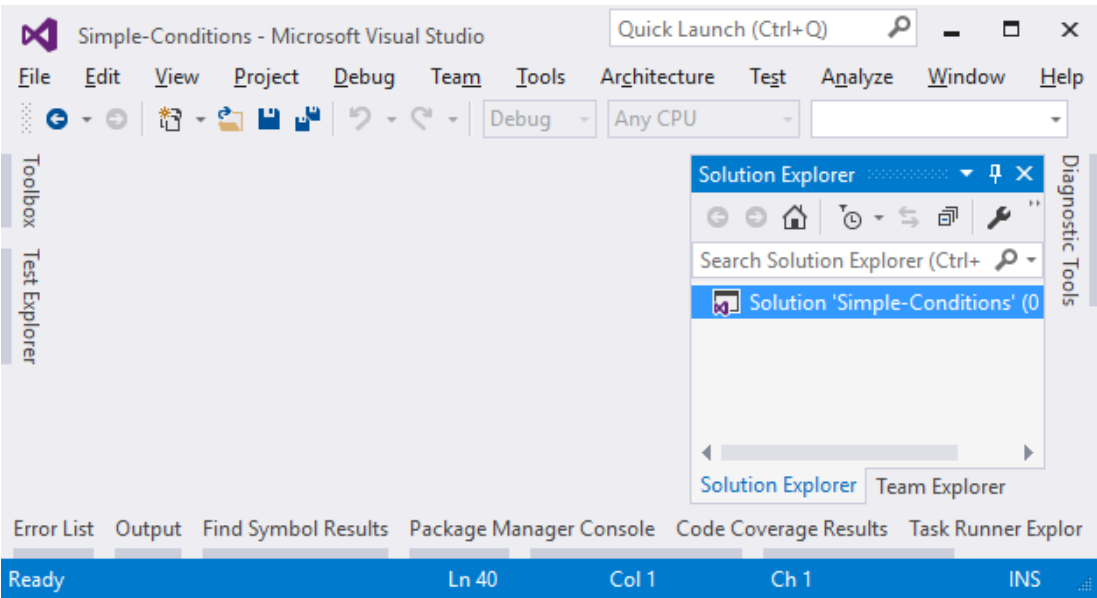
We run Visual Studio and create a new **Blank Solution**: [File] -> [New] -> [Project].



Choose from the dialog box [Templates] -> [Other Project Types] -> [Visual Studio Solutions] -> [Blank Solution] and give an appropriate project name, for example: "Simple-Conditions":



Now we have an empty Visual Studio Solution (no projects in it):



We will use this solution to create a **separate project** for each of the problems, which we will solve as exercises.

Problem: Excellent Grade

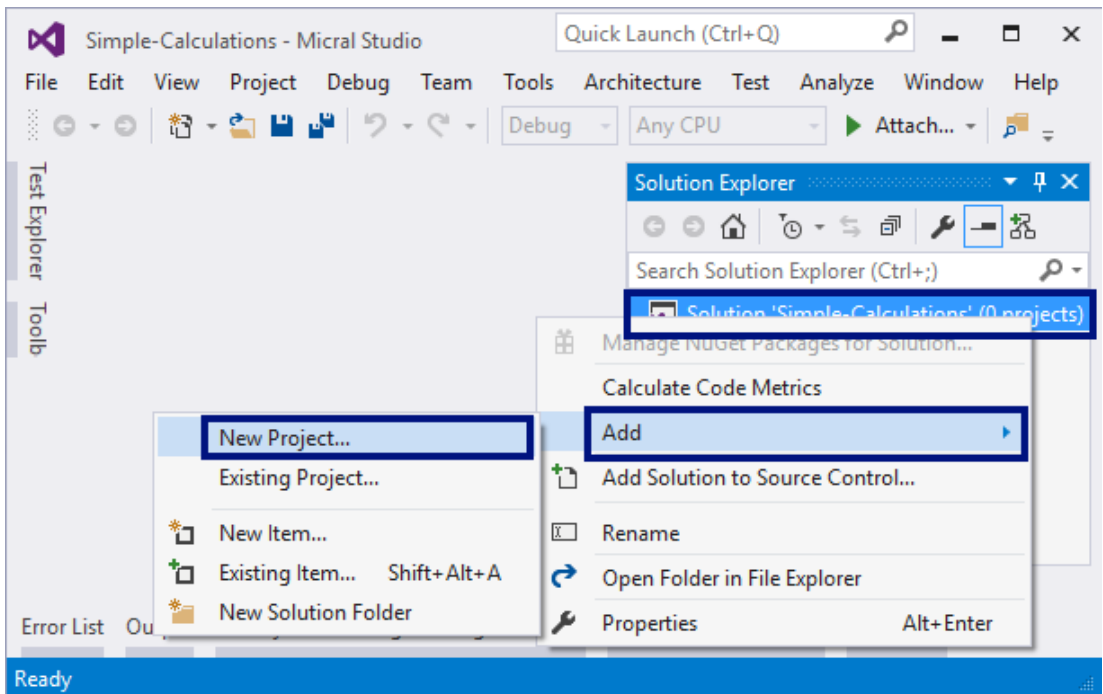
The first exercise for this topic is to write a **console program** that **inputs the grade** (decimal number) and prints **Excellent!** if the grade is **5.50** or higher.

Sample Input and Output

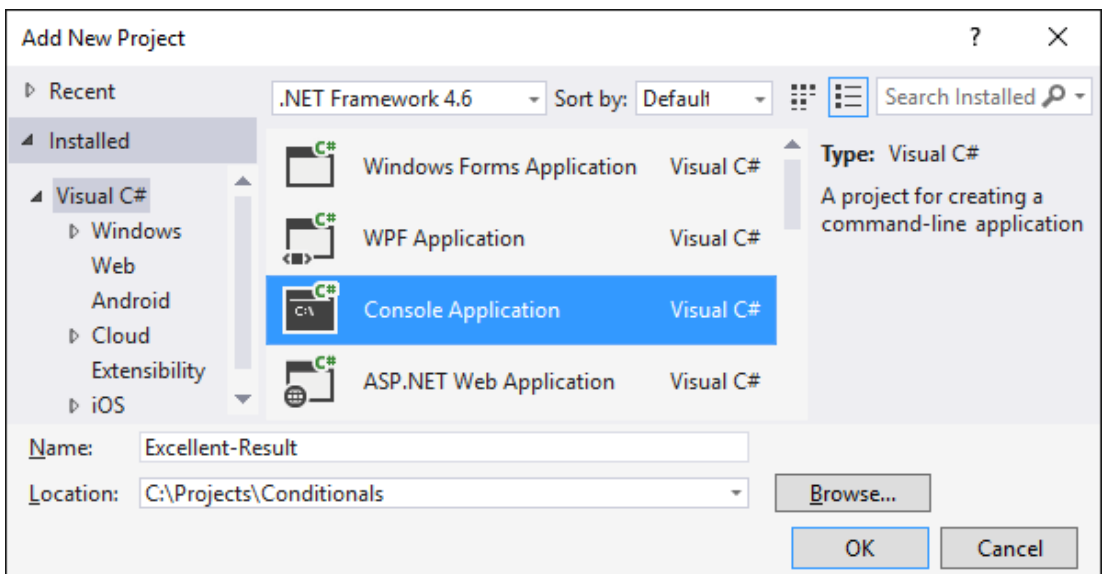
Input	Output
6	Excellent!
5	(no output)
5.5	Excellent!
5.49	(no output)

Creating a New C# Project

We create a **new project** in the existing Visual Studio solution. In **Solution Explorer**, right-click on **Solution 'Simple-Conditions'**. Then choose **[Add] -> [New Project]**:



A dialog box will open for selecting a project. Choose **C# console application** and specify a name, for example "Excellent-Result":



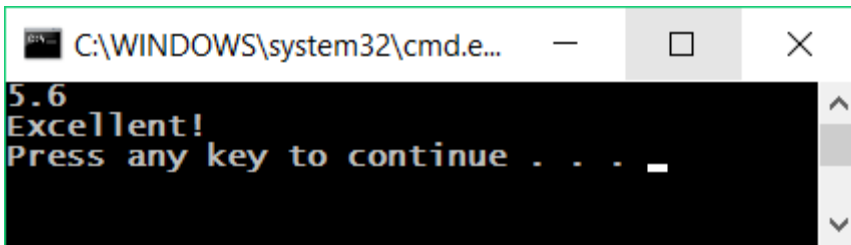
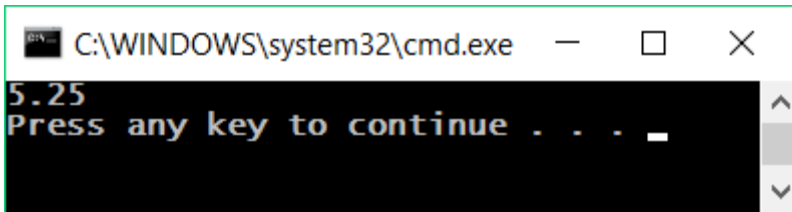
Now we have a solution with one console application in it. What remains is to write the code to solve the problem.

Writing the Program Code

For this purpose, we go into the body of the **Main (string [] args)** method and write the following code:

```
namespace Excellent_Result
{
    class Program
    {
        static void Main(string[] args)
        {
            var grade = double.Parse(Console.ReadLine());
            if (grade >= 5.50)
            {
                Console.WriteLine("Excellent!");
            }
        }
    }
}
```

Run the program with [Ctrl+F5], to test it with different input values:



Testing in the Judge System

Test your solution from the example here:
<https://judge.softuni.org/Contests/Practice/Index/506#0>.

Conditional Statements x

https://judge.softuni.bg/Contests/Practice/Index/152#0

Excellent Result

```

4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Excellent_Result
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            var grade = double.Parse(Console.ReadLine());
14            if (grade >= 5.50)
15            {
16                Console.WriteLine("Excellent!");
17            }
18        }
19    }
20 }
21

```

Allowed working time: 0.100 sec.
Allowed memory: 16.00 MB

C# code Submit

Submissions		
<div> <div>⏮</div> <div>⏪</div> <div>1</div> <div>⏩</div> <div>⏭</div> </div> <div>🔄</div>		
Points	Time and memory used	Submission date
<div> <div>✓✓✓✓✓✓</div> <div>100 / 100</div> </div>	Memory: 7.81 MB Time: 0.029 s	17:51:38 26.01.2016 <div>Details</div>

Problem: Excellent Grade or Not

The next exercise from this topic is to write a console program that inputs the grade (decimal number) and prints **Excellent!** if the grade is 5.50 or higher, otherwise "Not excellent."

Sample Input and Output

Input	Output
6	Excellent!
5	Not excellent.

Input	Output
5.5	Excellent!
5.49	Not excellent.

Creating a New C# Project and Writing the Program Code

First, we create a new C# console project in the Simple-Conditions solution.

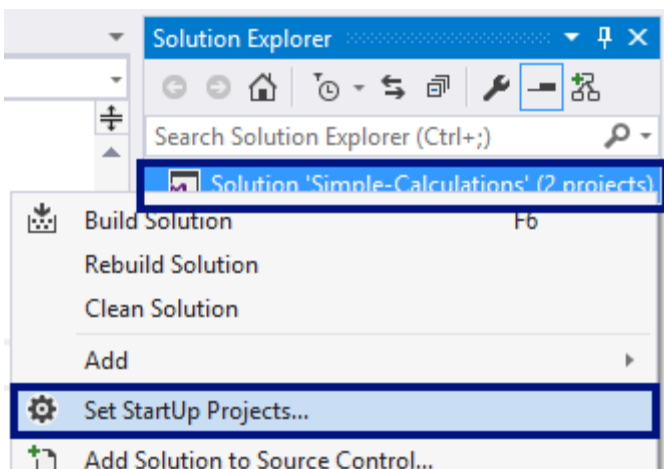
- We click on the solution in Solution Explorer and choose [Add] -> [New Project].
- We choose [Visual C#] -> [Windows] -> [Console Application] and specify a name, for example: "Excellent-or-Not".

Now we have to **write the code** of the program. You can get help by using the sample code from the picture:

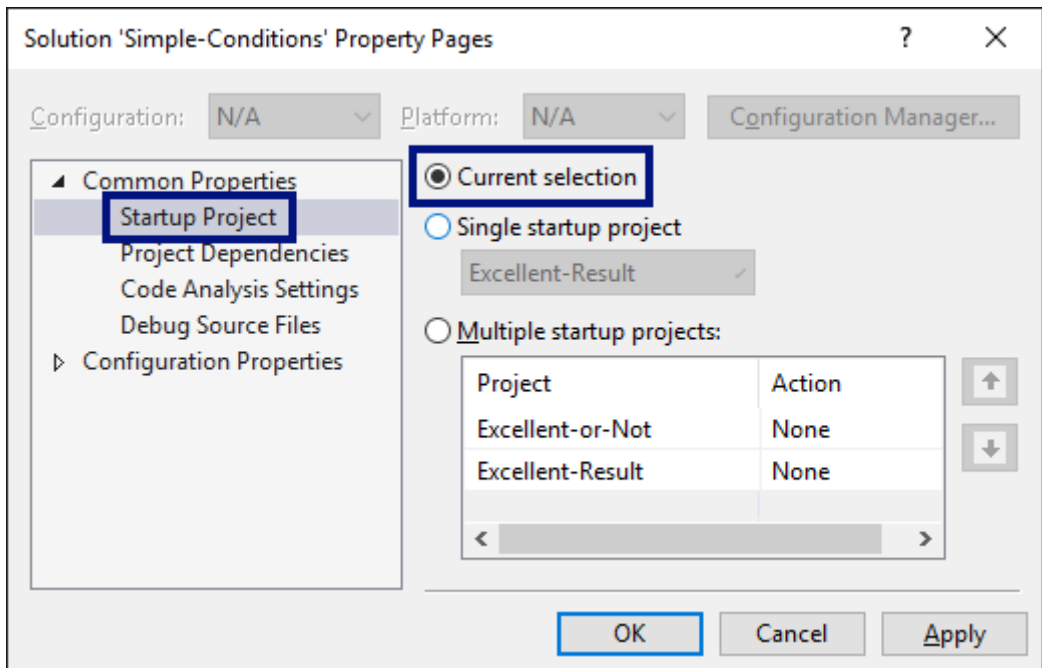
```
var grade = double.Parse(Console.ReadLine());
if (grade >= 5.50)
{
    Console.WriteLine("Excellent!");
}
else
{
    Console.WriteLine("Not excellent.");
}
```

Automatic Switching to the Current Project

We turn on the mode for automatic switching to the current project by right-clicking on the main Solution and choosing [Set StartUp Projects...]:

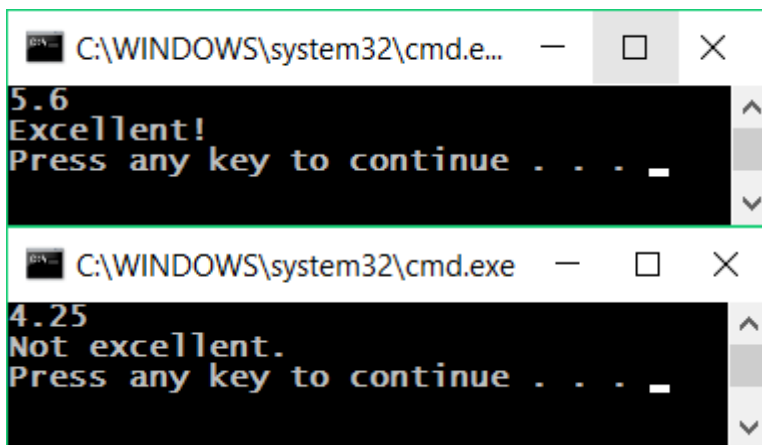


A dialog box will appear, and you have to choose [Startup Project] -> [Current selection]:



Local Testing

Now we run the program as usual with [Ctrl + F5] and test if it works correctly:



Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#1>.

Submissions		
<div><div><div><div><div></div></div></div><div><div><div>1</div></div></div><div><div><div></div></div></div><div><div><div></div></div></div></div><div></div></div>		
Points	Time and memory used	Submission date
✓✓✓✓ 100 / 100	Memory: 7.83 MB Time: 0.014 s	22:37:59 21.01.2016 <div>Details</div>
<div><div><div><div><div></div></div></div><div><div><div>1</div></div></div><div><div><div></div></div></div><div><div><div></div></div></div></div><div></div></div>		

Problem: Even or Odd

Write a program that checks whether an **integer** is **even** or **odd**.

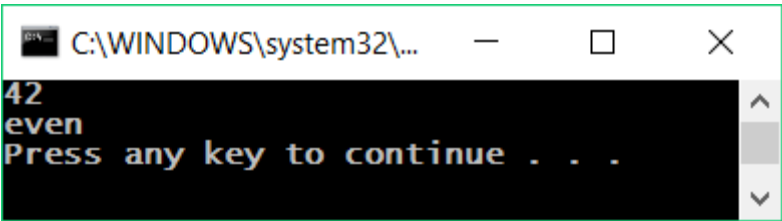
Sample Input and Output

Input	Output
2	even
3	odd
25	odd
1024	even

Hints and Guidelines

Again, first we add a new C # console project in the existing solution. In the **static void Main ()** method, we have to write the program code. Checking if a given number is even can be done with the operator **%**, which will return the **remainder from an integer divided by 2** as follows: **var isEven = (num % 2 == 0)**.

Now we **run** the program with [Ctrl + F5] and test it:



Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#2>.

Problem: Finding the Greater Number

Write a program that inputs **two integers** and prints the larger one.

Sample Input and Output

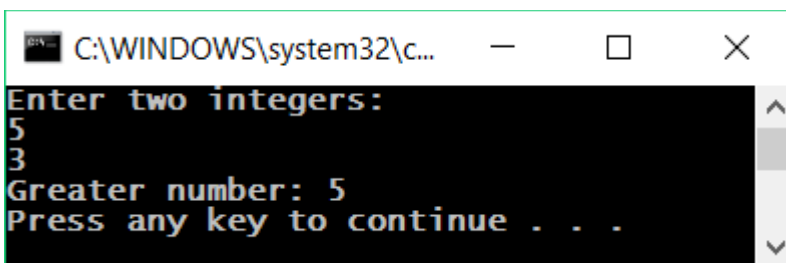
Input	Output
5 3	5
3 5	5
10 10	10
-5 5	5

Hints and Guidelines

As usual, first we need to add a **new C# console project** to the existing solution. For the code of the program we need a single **if-else** statement. You can partially get assistance for the code from the picture that is deliberately blurred to make you think about how to write it yourself:

```
Console.WriteLine("Enter two integers:");  
var num1 = int.Parse(Console.ReadLine());  
var num2 = int.Parse(Console.ReadLine());  
if (num1 > num2)  
    Console.WriteLine("Greater number: " + num1);  
else  
    Console.WriteLine("Greater number: " + num2);
```

When we are done with the implementation of the solution, we **run** the program with [Ctrl + F5] and test it:



```
Enter two integers:  
5  
3  
Greater number: 5  
Press any key to continue . . .
```

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#3>.

Problem: Typing a Digit in Words

Write a program that inputs **an integer in the range [0 ... 9]** and outputs it **in words** in English. If the number is out of range, print "number too big".

Sample Input and Output

Input	Output
5	five
1	one
9	nine
10	number too big

Hints and Guidelines

We can use a series of **if-else** statements to examine the possible **11 cases**.

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#4>.

Problem: Bonus Score

We have an **integer** – the number of points. **Bonus score** are charged on it, according to the rules described below. Write a program that calculates **bonus score** for this figure and **total points** with bonuses.

- If the number is **up to 100** including, bonus score is **5**.
- If the number is **larger than 100**, bonus score is **20%** of the number.
- If the number is **larger than 1000**, bonus score is **10%** of the number.
- Additional bonus score (accrued separately from the previous ones):
 - for **even** number -> + 1 p.
 - for number, that **ends with 5** -> + 2 p.

Sample Input and Output

Input	Output
20	6 26
175	37 212
2703	270.3 2973.3

Input	Output
15875	1589.5 17464.5

Video: Bonus Score

Watch the video lesson about the solution of the "Bonus Score" problem:
<https://youtu.be/hstZ5rNJ7vs>.

Hints and Guidelines

We can calculate the main and additional bonus score with a series of **if-else-if-else** statements. For the **main bonus score** we have **3 cases** (when the entered number is up to 100, between 100 and 1000 and larger than 1000), and for **extra bonus score** – **2 more cases** (when the number is even and odd).

```
var num = int.Parse(Console.ReadLine());
var bonusScore = 0.0;

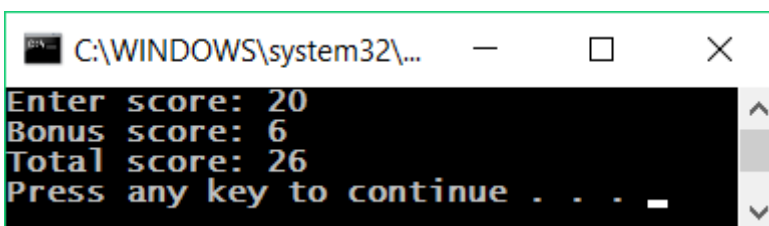
if (num > 1000)
{
    bonusScore = num * 0.10;
}
else // TODO: Write more logic here...

if (num % 10 == 5)
{
    bonusScore += 2;
}
else // TODO: Write more logic here...

// bonus score:
Console.WriteLine(bonusScore);

// total score:
Console.WriteLine(num + bonusScore);
```

Here's an example solution:



```
Enter score: 20
Bonus score: 6
Total score: 26
Press any key to continue . . . _
```

Note that for this exercise, the judge is set to ignore anything that is not a number, so we can print not only the numbers, but also specifying text.

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#5>.

Problem: Guess the Password

Write a program that **inputs a password** (one line with random text) and checks if the input **matches** the phrase "s3cr3t!P@ssw0rd". If it matches, print "Welcome", otherwise "Wrong password!".

Sample Input and Output

Input	Output
qwerty	Wrong password!
s3cr3t!P@ssw0rd	Welcome
s3cr3t!p@ss	Wrong password!

Hints and Guidelines

Use an **if-else** statement.

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#8>.

Problem: Summing Up Seconds

Three athletes finish in a particular number of **seconds** (between **1** and **50**). Write a program that introduces the times of the contestants and calculates their **total time** in "minutes:seconds" format. Seconds need to be **zeroed at the front** (2 -> "02", 7 -> "07", 35 -> "35").

Sample Input and Output

Input	Output
35 45 44	2:04
22 7 34	1:03
50 50 49	2:29

Input	Output
14 12 10	0:36

Video: Summing Up Seconds

Watch the video about the "Summing Up Seconds" problem and its step-by-step solution:
<https://youtu.be/bnNxe79X-wo>.

Hints and Guidelines

First, we sum up the three numbers to get the total result in seconds. Since **1 minute = 60** seconds, we will have to calculate the number of minutes and seconds in the range 0 to 59:

- If the result is between 0 and 59, we print 0 minutes + calculated seconds.
- If the result is between 60 and 119, we print 1 minute + calculated seconds minus 60.
- If the result is between 120 and 179, we print 2 minutes + calculated seconds minus 120.
- If the seconds are less than 10, we print the number with zero in front.

```
int firstCompetitor = int.Parse(Console.ReadLine());
// TODO: Read also second and third competitors' time

int seconds = firstCompetitor + secondCompetitor + thirdCompetitor;
int minutes = 0;

if (seconds > 59)
{
    minutes++;
    seconds = seconds - 60;
}

if (seconds > 59)
{
    minutes++;
    seconds = seconds - 60;
}

if (seconds < 10)
{
    Console.WriteLine(minutes + ":" + "0" + seconds);
}
else
{
    Console.WriteLine(minutes + ":" + seconds);
}
```

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#6>.

Problem: Metric Converter

Write a program that convert a distance between the following 8 units of measure: **m**, **mm**, **cm**, **mi**, **in**, **km**, **ft**, **yd**. Use the below table:

Input measure	Output measure
1 meter (m)	1000 millimeters (mm)
1 meter (m)	100 centimeters (cm)
1 meter (m)	0.000621371192 miles (mi)
1 meter (m)	39.3700787 inches (in)
1 meter (m)	0.001 kilometers (km)

Input measure	Output measure
1 meter (m)	3.2808399 feet (ft)
1 meter (m)	1.0936133 yards (yd)

You have three input lines:

- First line: number for converting.
- Second line: input unit.
- Third line: output unit (for result).

Sample Input and Output

Input	Output
12 km ft	39370.0788
150 mi in	9503999.99393599
450 yd km	0.41147999937455

Video: Metric Converter

Watch the following video lesson to learn how to solve the "Metric Converter" problem:
<https://youtu.be/Bd6NgaHhrko>.

Hints and Guidelines

We read the input data, and we can add **ToLower()** function when we read the measuring units. The function will make all letters small. As we can see from the table in the condition, we can only do converting **between meters and some other measuring unit**. Then, first we have to calculate the number for converting in meters. That's why, we need to make a set of checks to define the input unit and then the output unit.

```
var size = double.Parse(Console.ReadLine());
var sourceMetric = Console.ReadLine().ToLower();
var destMetric = Console.ReadLine().ToLower();
if (sourceMetric == "km")
{
    size = size / 0.001;
}
// Check the other metrics: mm, cm, ft, yd, ...
if (destMetric == "ft")
{
    size = size * 3.2808399;
}
// Check the other metrics: mm, cm, ft, yd, ...
Console.WriteLine(size + " " + destMetric);
```

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#7>.

Problem: Numbers from 100 to 200

Write a program that **inputs an integer** and checks if it is **below 100**, **between 100 and 200** or **over 200**. Print the appropriate message as in the examples below.

Sample Input and Output

Input	Output
95	Less than 100
120	Between 100 and 200
210	Greater than 200

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#9>.

Problem: Identical Words

Write a program that **inputs two words** and checks if they are the same. Do not make difference between uppercase and lowercase letters. You have to print **"yes"** or **"no"**.

Sample Input and Output

Input	Output
Hello Hello	yes
SoftUni softuni	yes
Soft Uni	no
banana lemon	no
HeLLo hELLo	yes

Hints and Guidelines

Before comparing the words, turn them into a lowercase to avoid the letter size influence (uppercase / lowercase): **word = word.ToLower()**.

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#10>.

Problem: Speed Assessment

Write a program that **inputs the speed** (decimal number) and prints **speed information**. For speed **up to 10** (inclusive), print "**slow**". For speed **over 10** and **up to 50**, print "**average**". For speed **over 50** and **up to 150**, print "**fast**". For speed **over 150** and **up to 1000**, print "**ultra fast**". For higher speed, print "**extremely fast**".

Sample Input and Output

Input	Output
8	slow
49.5	average
126	fast
160	ultra fast
3500	extremely fast

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#11>.

Problem: Areas of Figures

Write a program that **inputs the sizes of a geometric figure and calculates its area**. The figures are four types: **square**, **rectangle**, **circle** and **triangle**.

The first line of the input provides the type of the figure (**square**, **rectangle**, **circle**, **triangle**).

- If the figure is a **square**, the next line provides one number – the length of its **side**.
- If the figure is a **rectangle**, the next two lines we provide two numbers – the lengths of its **sides**.
- If the figure is a **circle**, the next line provides one number – the **radius** of the circle.
- If the figure is a **triangle**, the next two lines provide two numbers – the length of the **side** and the length of its **height**.

Round the result up to the **third digit after the decimal point**.

Sample Input and Output

Input	Output
square 5	25
rectangle 7 2.5	17.5
circle 6	113.097
triangle 4.5 20	45

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#12>.

Problem: Time + 15 Minutes

Write a program that **inputs hours and minutes** of a 24-hour day and calculates what will be the time **after 15 minutes**. Print the result in **hh:mm** format. Hours are always between 0 and 23, and minutes are always between 0 and 59. Hours are written with one or two digits. Minutes are always written with two digits and zero at the front when needed.

Sample Input and Output

Input	Output
1 46	2:01
0 01	0:16
23 59	0:14
11 08	11:23
12 49	13:04

Hints and Guidelines

Add 15 minutes and check using a few conditions. If minutes are over 59, **increase hours** with 1 and **reduce minutes** with 60. Identically, check the case when hours are over 23. When you print the minutes, you should **check for zero at the front**.

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#13>.

Problem: Equal 3 Numbers

Write a program that inputs **3 numbers** and prints whether they are the same (**yes** / **no**).

Sample Input and Output

Input	Output
5 5 5	yes
5 4 5	no
1 2 3	no

Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#14>.

Problem: * Writing the Numbers from 0 to 100 in Words

Write a program that converts a number in the range of [0 ... 100] into text.

Sample Input and Output

Input	Output
25	twenty five
42	forty two
6	six

Hints and Guidelines

First check for **one-digit numbers** and if the number is one-digit, print the appropriate word for it. Then check for **two-digit numbers**. Print them in two parts: left part (**tens** = number / 10) and right part (**units** = number % 10). If the number has 3 digits, it must be 100 and can be considered a special case.

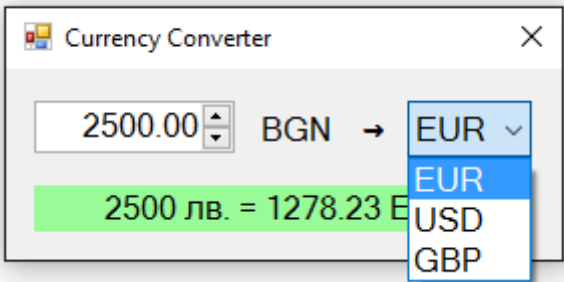
Testing in the Judge System

Test your solution here: <https://judge.softuni.org/Contests/Practice/Index/506#15>.

Lab: GUI (Desktop) Application – Currency Converter

After we've done some exercises on **conditional statements (checks)**, now let's do something more interesting: an application with a graphical user interface (GUI) for converting currencies. We will use the knowledge from this mini book to choose from several available currencies and make calculations at different rate to the selected currency.

Now let's see how to create a graphical (GUI) app for **currency conversion**. The app will look like the picture below:

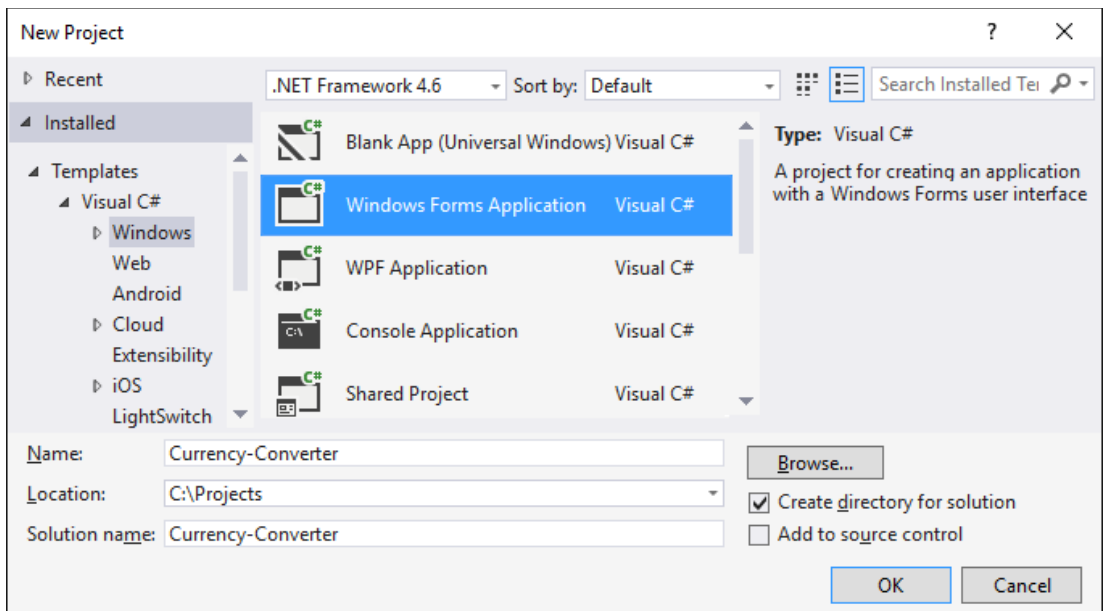


Video: Building a GUI App "Currency Converter"

Watch the video lesson about building a Windows Forms based GUI app "Currency Converter": <https://youtu.be/IlkPmoXmjdg>.

Creating a New C# Project and Adding Controls

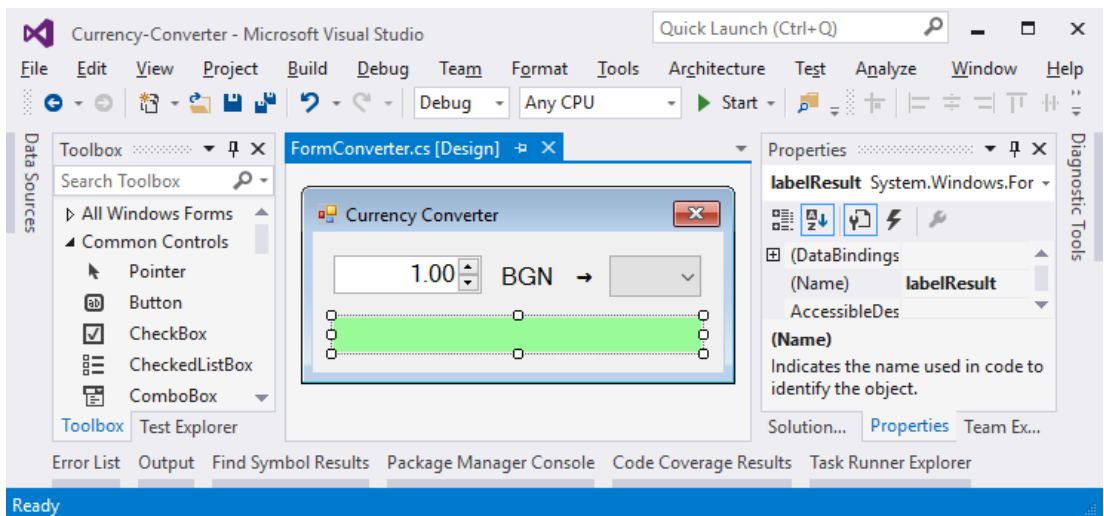
This time we create a new **Windows Forms Application** with name “Currency-Converter”:



We order the following controls in the form:

- One box for entering a number (**NumericUpDown**)
- One drop-down list with currencies (**ComboBox**)
- Text block for the result (**Label1**)
- Several inscriptions (**Label1**)

We set the **sizes** and their properties to look like the picture below:



Configuring the UI Controls

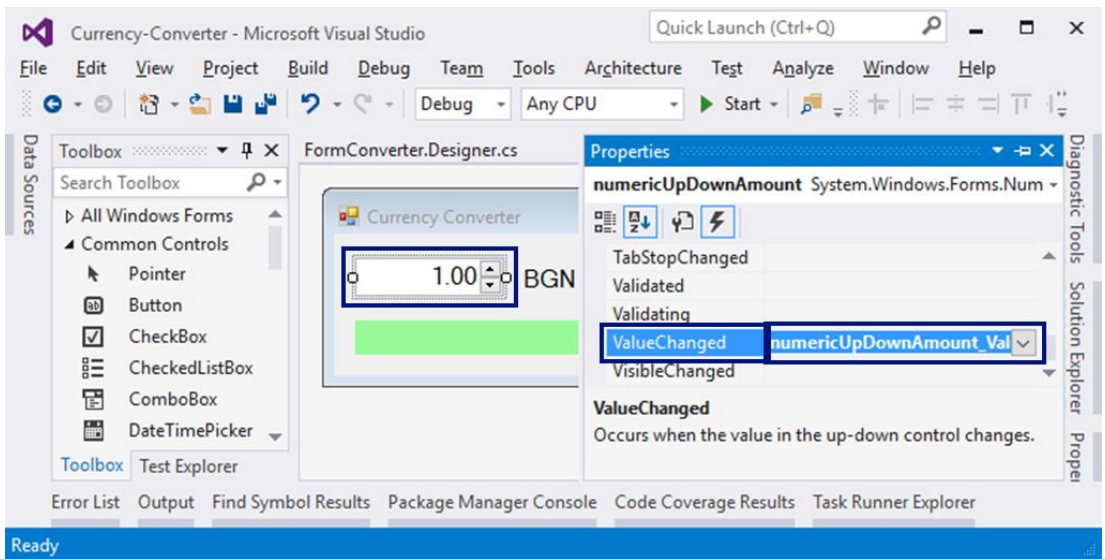
We apply the following **settings** for the UI controls:

- For the main form (Form) that contains all the controls:
 - (name) = FormConverter
 - Text = "Currency Converter"
 - Font.Size = 12
 - MaximizeBox = False
 - MinimizeBox = False
 - FormBorderStyle = FixedSingle
- For the field for entering a number (NumericUpDown):
 - (name) = numericUpDownAmount
 - Value = 1
 - Minimum = 0
 - Maximum = 1000000
 - TextAlign = Right
 - DecimalPlaces = 2
- For the drop-down list of currencies (ComboBox):
 - (name) = comboBoxCurrency
 - DropDownStyle = DropDownList
 - Items =
 - EUR
 - USD
 - GBP
- For the result text block (Label):
 - (name) = labelResult
 - AutoSize = False
 - BackColor = PaleGreen
 - TextAlign = MiddleCenter
 - Font.Size = 14
 - Font.Bold = True

Events and Event Handlers

We need to take the following **events** to write the C# code that will be executed upon their occurrence:

- The event **ValueChanged** of numeric entry control **numericUpDownAmount**:



- The event **Load** of the form **FormConverter**
- The event **SelectedIndexChanged** of the drop-down list for choosing the currency **comboBoxCurrency**

We will use the following **C#** code for event handling:

```
private void FormConverter_Load(object sender, EventArgs e)
{
    this.comboBoxCurrency.SelectedItem = "EUR";
}

private void numericUpDownAmount_ValueChanged(object sender, EventArgs e)
{
    ConvertCurrency();
}

private void comboBoxCurrency_SelectedIndexChanged(object sender, EventArgs e)
{
    ConvertCurrency();
}
```

Our task is to select the currency "EUR" when we start the program and change the values in the sum or currency field then calculating the result by calling the **ConvertCurrency()** method.

Writing the Program Code

We have to write the event **ConvertCurrency()** to convert the leva amount into the selected currency:

```
private void ConvertCurrency()
{
```

```
var originalAmount = this.numericUpDownAmount.Value;
var convertedAmount = originalAmount;
if (this.comboBoxCurrency.SelectedItem.ToString() == "EUR")
{
    convertedAmount = originalAmount / 1.95583m;
}
else if (this.comboBoxCurrency.SelectedItem.ToString() == "USD")
{
    convertedAmount = originalAmount / 1.80810m;
}
else if (this.comboBoxCurrency.SelectedItem.ToString() == "GBP")
{
    convertedAmount = originalAmount / 2.54990m;
}
this.labelResult.Text = originalAmount + " lv. = " +
Math.Round(convertedAmount, 2) + " " +
    this.comboBoxCurrency.SelectedItem;
}
```

The above code takes the amount for converting the field `numericUpDownAmount` and the selected currency for the result from the field `comboBoxCurrency`. Then with a conditional statement, according to the selected currency, the amount is divided by the exchange rate (which is fixed in the source code). Finally, a text message with the result (rounded to the second digit after the decimal point) is generated and recorded in the green box `labelResult`. Try it!

What's Next?

The “Programming Basics” series consists of publications for beginners in programming that covers the following topics:

1. First Steps in Programming – commands, programs, C#, Visual Studio,
2. Simple Calculations – variables, calculations, console input / output
3. Simple Conditions – conditional statements, the “if-else” construction
4. More Complex Conditions – nested if-else, logical “and”, “or”, “not”
5. Repetitions (Loops) – simple for-loops (repeat from 1 to n)
6. Nested Loops – nested loops and problem solving, drawing 2D figures
7. More Complex Loops – loops with a step, infinite loops with breaks
8. How to Become a Software Engineer?

The next topics are coming. Be patient!

Sign-Up to Study in SoftUni

The easiest way to become a software engineer is to go through the “Software University” training program at SoftUni. Signup now for the free Programming Basics training course:

<https://softuni.org/apply>

Join the SoftUni community and study programming for free in our interactive training platform. Get live support and mentoring from our trainers. Become a software developer now!