Basic Syntax

Basic Syntax , I/O, Conditions, Loops and Debugging

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

Table of Contents

- Introduction and Basic Syntax
- Input / Output
- Comparison operators
- The if-else / Switch-Case Statement
- Logical Operators
- Loops
- Debugging and Troubleshooting

INTRODUCTION AND BASIC SYNTAX

C# – Introduction

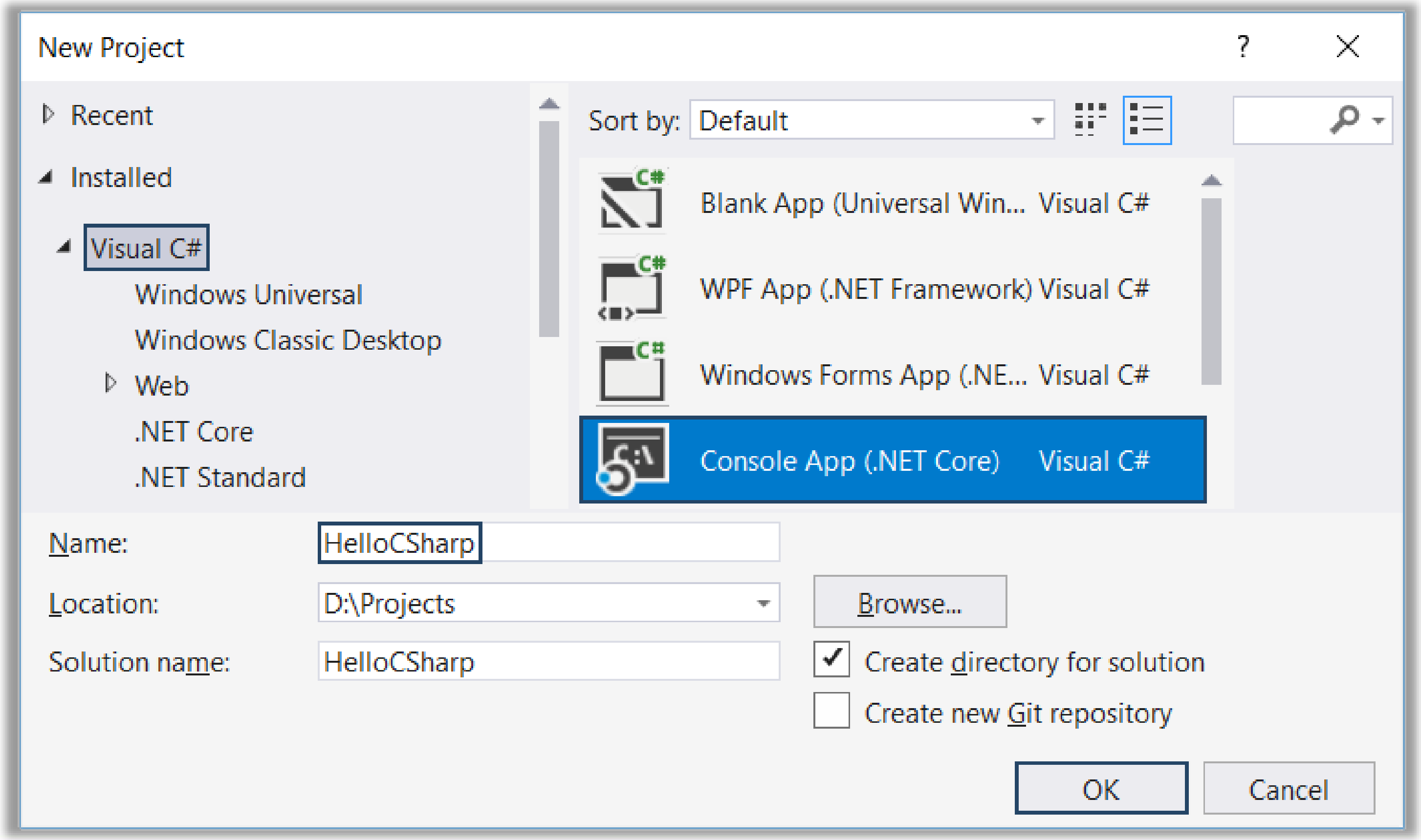
- C# is modern, flexible, general-purpose programming language
- Object-oriented by nature, statically-typed, compiled
- Runs on .NET Framework / .NET Core

```
static void Main()
{
    //Source code
}
```

Program starting point

Using Visual Studio

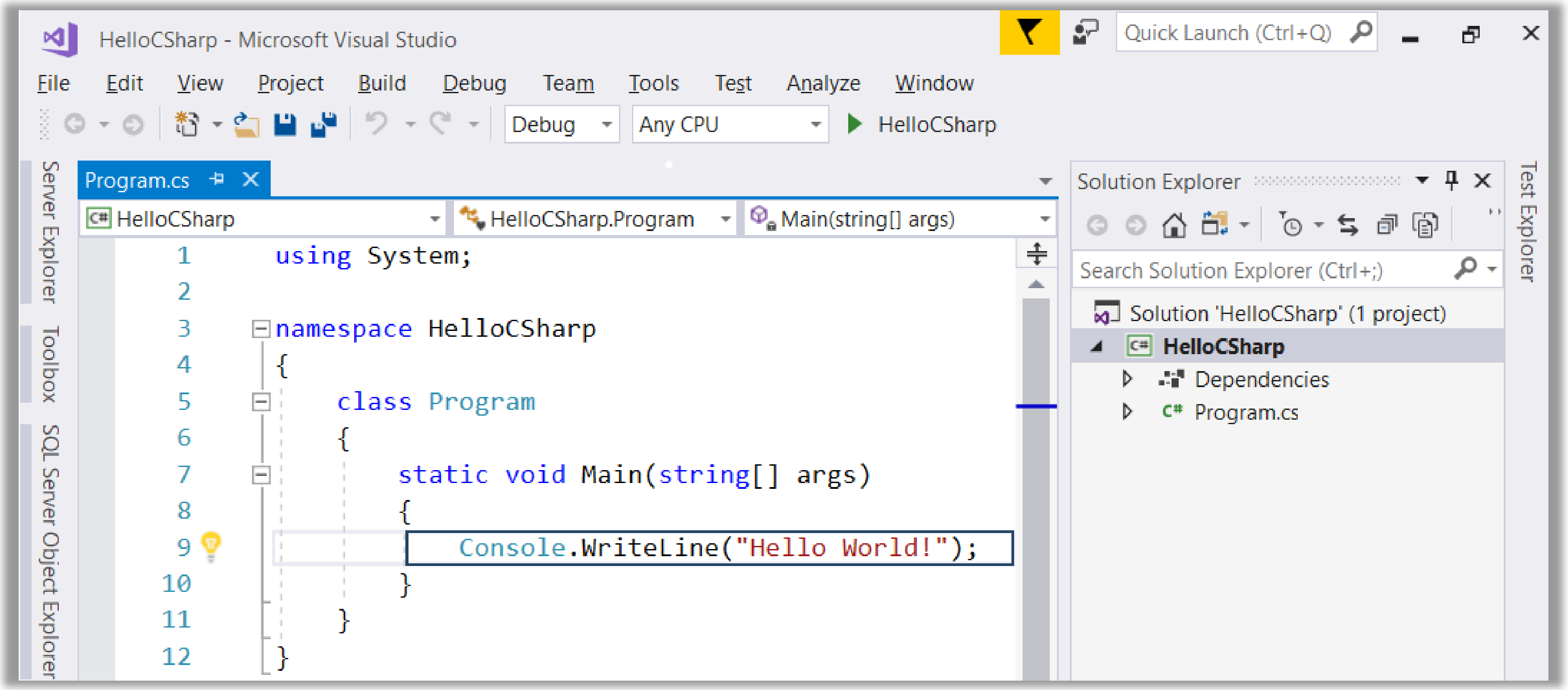
- Visual Studio (VS) is powerful IDE for C#
- Create a console application


New Project
?
Recent
Installed
Visual C#
Windows Universal
Windows Classic Desktop
Web
.NET Core
.NET Standard
Sort by: Default
Blank App (Universal Win... Visual C#
WPF App (.NET Framework) Visual C#
Windows Forms App (.NE... Visual C#
Console App (.NET Core) Visual C#
Browse...
OK
Cancel

Name: HelloCSharp
Location: D:\Projects
Solution name: HelloCSharp
[✓] Create directory for solution
[ ] Create new Git repository

Running the Program

- Start the program from VS using [Ctrl + F5]


HelloCSharp - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window
Debug Any CPU HelloCSharp
Program.cs
HelloCSharp HelloCSharp.Program Main(string[] args)
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'HelloCSharp' (1 project)
HelloCSharp
Dependencies
Program.cs
Server Explorer Toolbox SQL Server Object Explorer Test Explorer

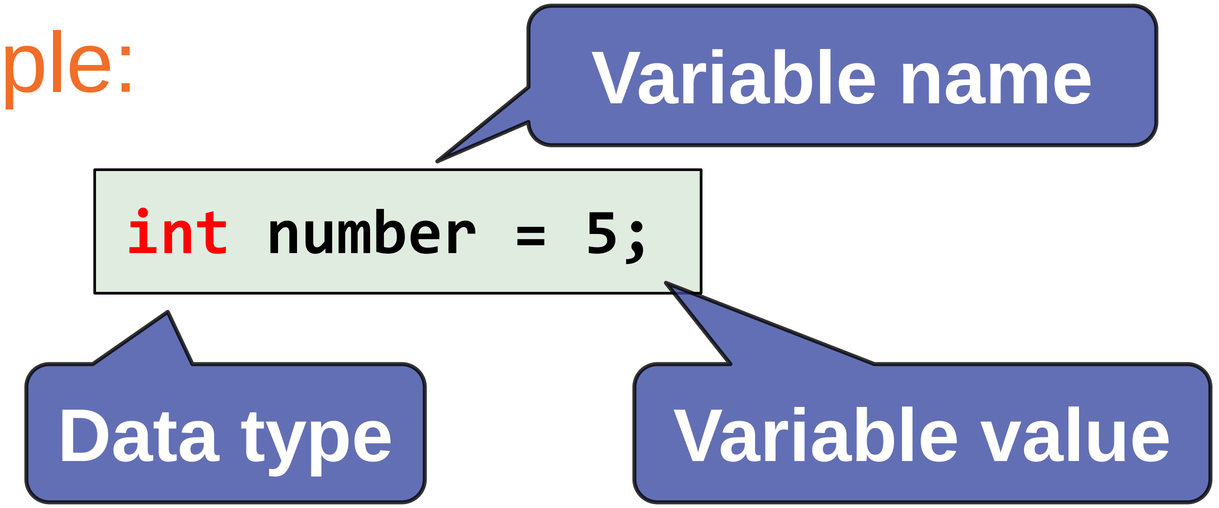
```
using System;

namespace HelloCSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Declaring Variables

- Defining and Initializing variables

```
{data type / var} {variable name} = {value};
```

- Example:



```
int number = 5;
```

CONSOLE I/O

Reading from and Writing to the Console

Reading from the Console

- We can read/write to the console, using the Console class
- Use the System namespace to access System.Console class

```
using System;
```

- Reading input from the console using Console.ReadLine():

Returns string

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

Converting Input from the Console

- Console.ReadLine() returns a string
- Convert the string to number by parsing:

```
string name = Console.ReadLine();
int age = int.Parse(Console.ReadLine());
double salary = double.Parse(Console.ReadLine());
bool isHungry = bool.Parse(Console.ReadLine());
```

Printing to the Console

- We can print to the console, using the Console class
- Use the System namespace to access System.Console class
- Writing output to the console:
  - Console.Write()
  - Console.WriteLine()

```
Console.Write("Hi, ");
Console.WriteLine("John!");
// Hi, John!
```

Using Placeholders

- Using **placeholders** to print on the console
- Examples:

```
string name = "George";
int age = 5;
Console.WriteLine("Name: {0}, Age: {1}", name, age);
// Name: George, Age: 5
```

Placeholder {0} corresponds to name

Placeholder {1} corresponds to age

Formatting Numbers in Placeholders

- D – format number to certain digits with leading zeros
- F – format floating point number with certain digits after the decimal point
- Examples:

```
double grade = 5.5334;
int percentage = 55;
Console.WriteLine("{0:F2}", grade);      // 5.53
Console.WriteLine("{0:D3}", percentage); // 055
```

Using String Interpolation

- Using string interpolation to print on the console
- Examples:

```
string name = "George";
int age = 5;

Console.WriteLine($"Name: {name}, Age: {age}");
//Name: George, Age 5
```

Put $ in front of the string to use string interpolation

Problem: Student Information

- You will be given 3 input lines:
  - Student Name, Age and Average Grade
- Print the input in the following format:
  - "Name: {name}, Age: {age}, Grade {grade}"
  - Format the grade to 2 decimal places

```
John
15
5.40
```

```
Name: John, Age: 15, Grade: 5.40
```

COMPARISON OPERATORS

Comparison Operators

Operator	Notation in C#
Equals	==
Not Equals	!=
Greater Than	>
Greater Than or Equals	>=
Less Than	<
Less Than or Equals	<=

Comparing Numbers

- Values can be compared:

```
int a = 5;
int 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
```

THE IF-ELSE STATEMENT

Implementing Control-Flow Logic

The If Statement

- The most simple conditional statement
  - Test for a condition
- Example: Take as an input a grade and check if the student has passed the exam (grade >= 3.00)

```
double grade = double.Parse(Console.ReadLine());
if (grade >= 3.00)
{
  Console.WriteLine("Passed!");
}
```

In C# the opening bracket stays on a new line

The if-else Statement

- Executes one branch if the condition is true and another, if it is false
- Example: Upgrade the last example, so it prints "Failed!", if the mark is lower than 3.00:

```
if (grade >= 3.00)
{
  Console.WriteLine("Passed!");
}
else
{
  // TODO: Print the message
}
```

The else keyword stays on a new line

Problem: Back in 30 Minutes

- Write a program that reads hours and minutes from the console and calculates the time after 30 minutes
  - – The hours and the minutes come on separate lines
- Example:

```
1
46
```
➡
```
2:16
```

```
0
01
```
➡
```
0:31
```

```
23
59
```
➡
```
0:29
```

```
11
08
```
➡
```
11:38
```

```
12
49
```
➡
```
13:19
```

```
11
32
```
➡
```
12:02
```

THE SWITCH-CASE STATEMENT

Simplified if-else-if-else

The switch-case Statement

- Works as sequence of if-else statements
- Example: read input a number and print its corresponding month:

```
int month = int.Parse(Console.ReadLine());
switch (month)
{
  case 1: Console.WriteLine("January"); break;
  case 2: Console.WriteLine("February"); break;
  // TODO: Add the other cases
  default: Console.WriteLine("Error!"); break;
}
```

Problem: Foreign Languages

- By given country print its typical language:
  - English -> England, USA
  - Spanish -> Spain, Argentina, Mexico
  - other -> unknown

England	English
Spain	Spanish

LOGICAL OPERATORS

Writing More Complex Conditions

Logical Operators

- Logical operators give us the ability to write multiple conditions in one if statement
- They return a boolean value and compare boolean values

Operator	Notation in C#	Example
Logical NOT	!	!false -> true
Logical AND	&&	true && false -> false
Logical OR	||	true || false -> true

Problem: Theatre Promotions

- A theatre has the following ticket prices according to the age of the visitor and the type of day.
- If the age is < 0 or > 122, print "Error!":

Day / Age	0 <= age <= 18	18 < age <= 64	64 < age <= 122
Weekday	12$	18$	12$
Weekend	15$	20$	15$
Holiday	5$	12$	10$

```
Weekday
42
```
→
```
18$
```

```
Holiday
-12
```
→
```
Error!
```

LOOPS

Code Block Repetition

Loop: Definition

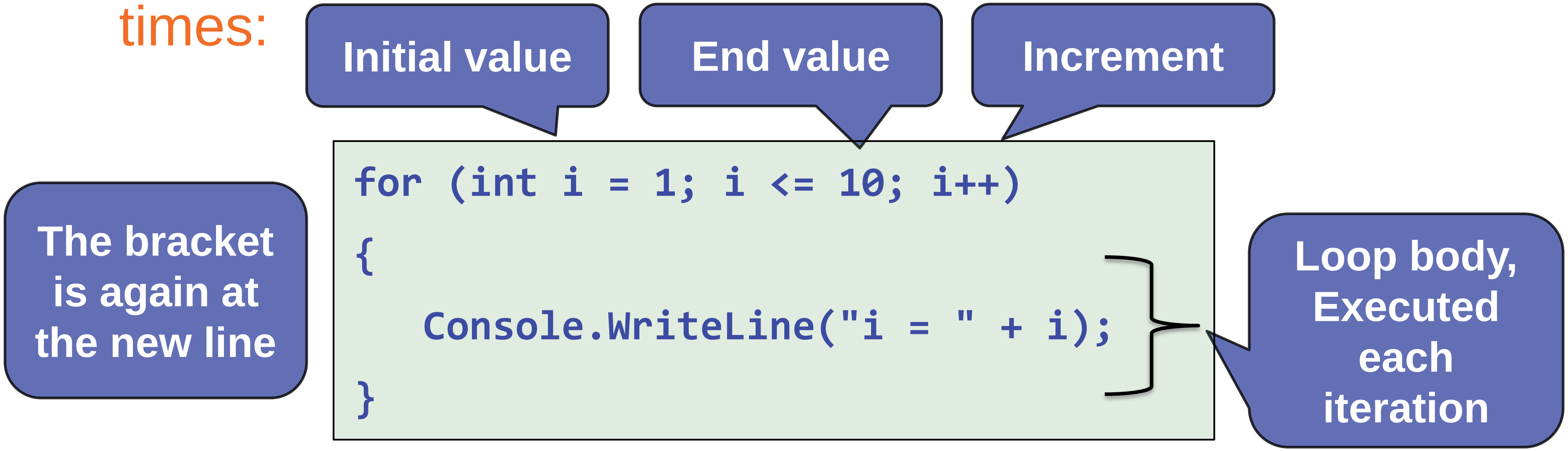
- A loop is a control statement that repeats the execution of a block of statements. The loop can:
  - Execute a code block a fixed number of times
    - for loop
  - Execute a code block while a given condition returns true
    - while
    - do…while

FOR-LOOPS

Managing the Count of the Iteration

For-Loops

- The for loop executes statements a fixed number of times:



```
for (int i = 1; i <= 10; i++)
{
    Console.WriteLine("i = " + i);
}
```

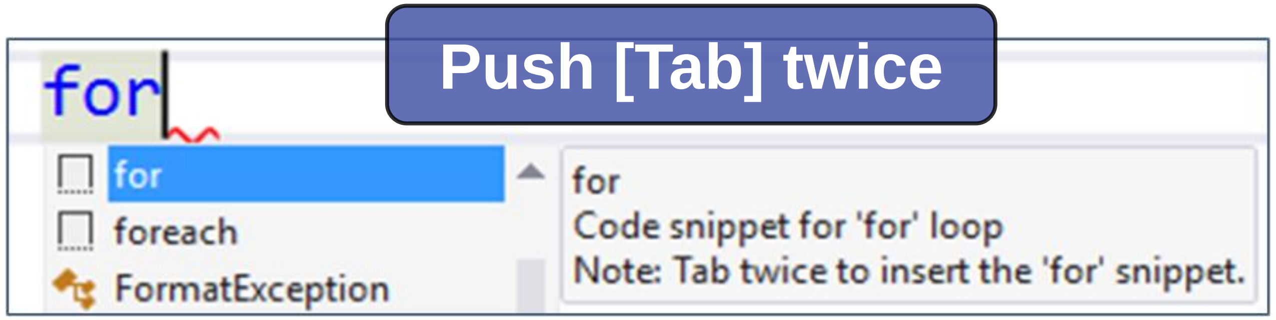
Initial value
End value
Increment
The bracket is again at the new line
Loop body, Executed each iteration

Example: Divisible by 3

- Print the numbers from 1 to 100, that are divisible by 3

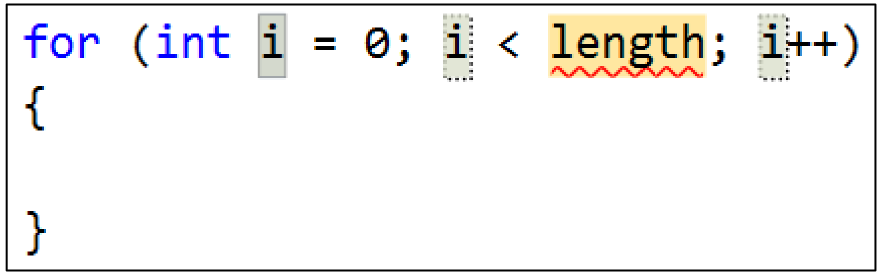
```
for (var i = 3; i <= 100; i += 3)
{
  Console.WriteLine(i);
}
```

- You can use "for" code snippet in Visual Studio



```
for
```

for
Code snippet for 'for' loop
Note: Tab twice to insert the 'for' snippet.



```
for (int i = 0; i < length; i++)
{

}
```

Problem: Sum of Odd Numbers

- Write a program to print the first n odd numbers and their sum

```
5
```

```
1
3
5
7
9
Sum: 25
```

```
3
```

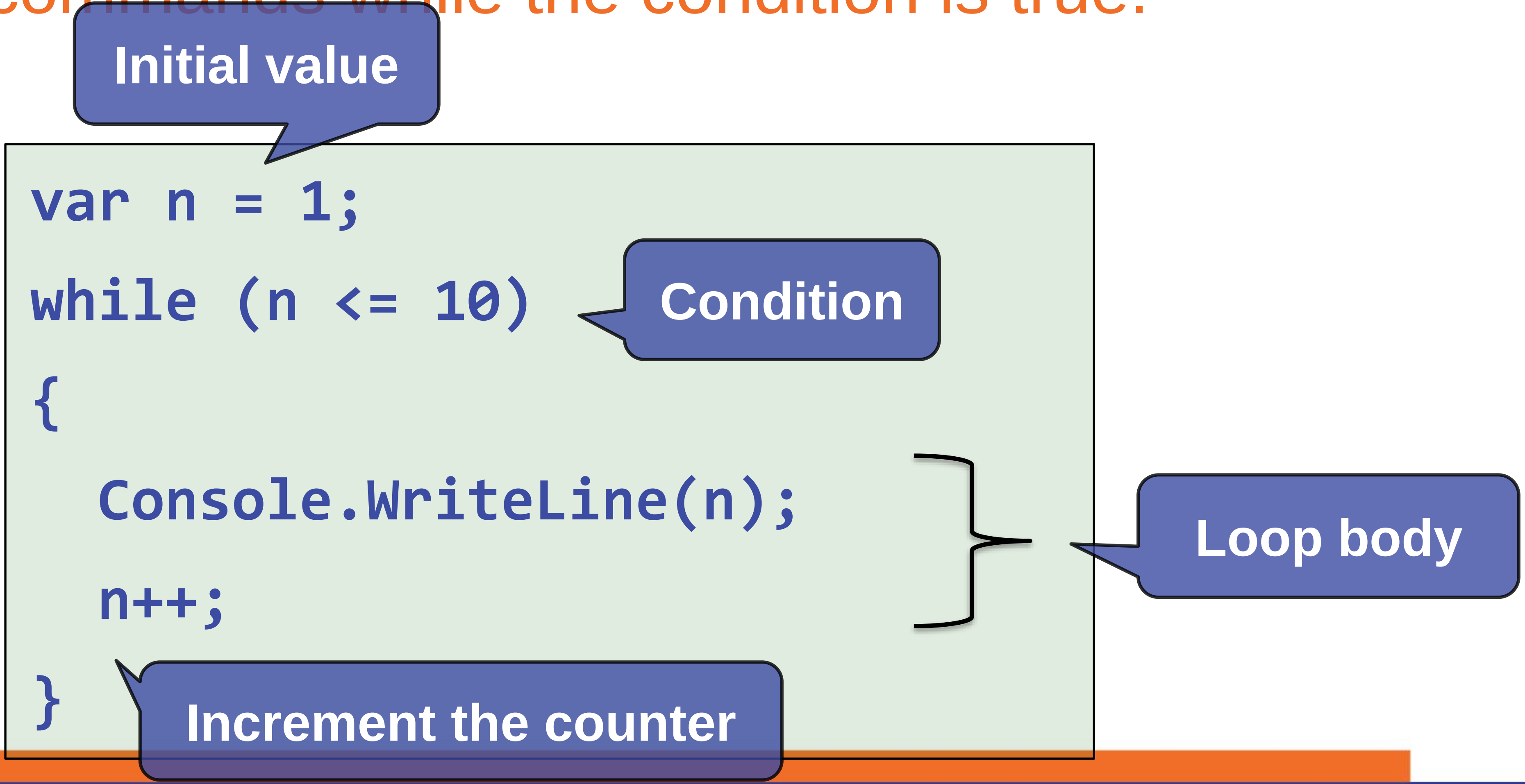
```
1
3
5
Sum: 9
```

WHILE LOOPS

Iterations While a Condition is True

While Loops

- Executes commands while the condition is true:



```
var n = 1;
while (n <= 10)
{
   Console.WriteLine(n);
   n++;
}
```

Problem: Multiplication Table

- Print a table holding number*1, number*2, …, number*10

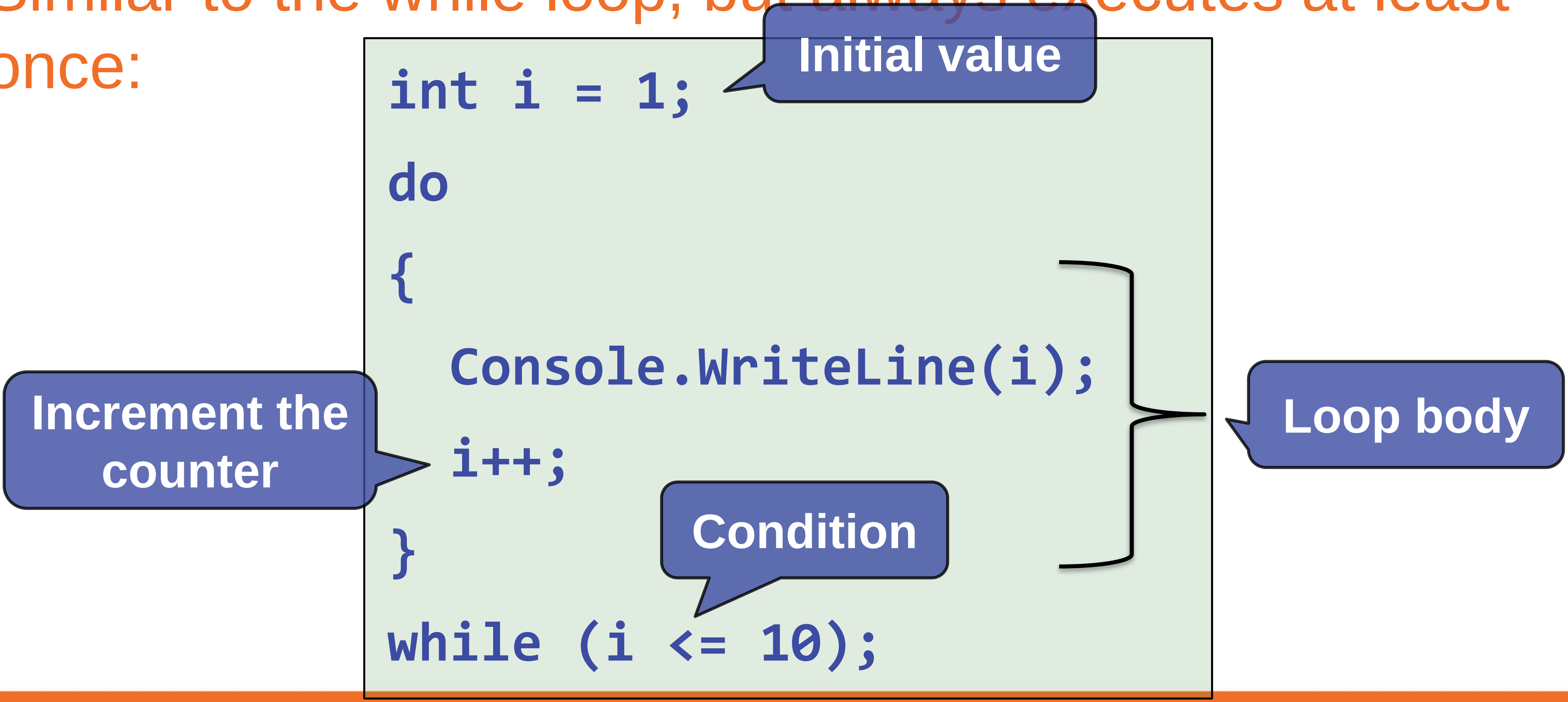
```
var number = int.Parse(Console.ReadLine());
var times = 1;
while (times <= 10)
{
  Console.WriteLine(
    $"{number} X {times} = {number * times}");
  times++;
}
```

EXECUTES CODE BLOCK ONE OR MORE TIMES

Do…While Loop

Do … While Loop

- Similar to the while loop, but always executes at least once:



```
int i = 1;
do
{
   Console.WriteLine(i);
   i++;
}
while (i <= 10);
```

Problem: Multiplication Table 2.0

- Upgrade your program and take the initial times from the console

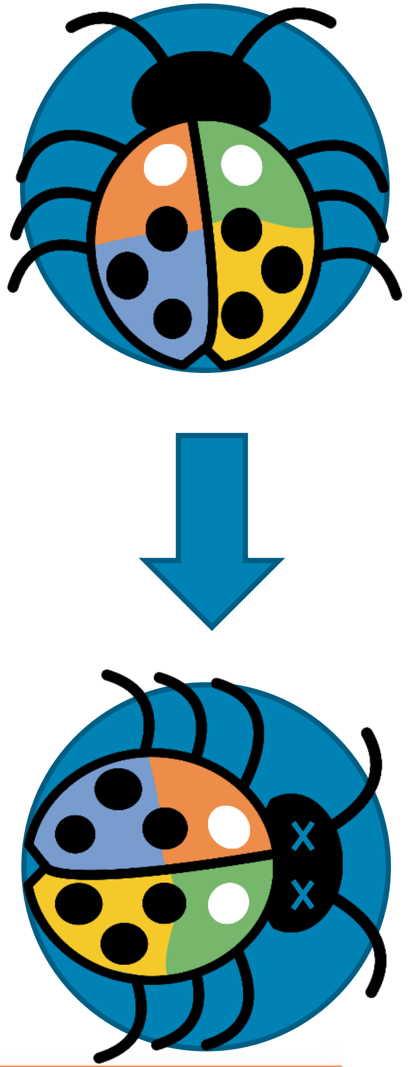
```
int number = int.Parse(Console.ReadLine());
int times = int.Parse(Console.ReadLine());
do
{
  Console.WriteLine(
    $"{number} X {times} = {number * times}"
  );
  times++;
} while (times <= 10);
```

DEBUGGING THE CODE

Using the Visual Studio Debugger

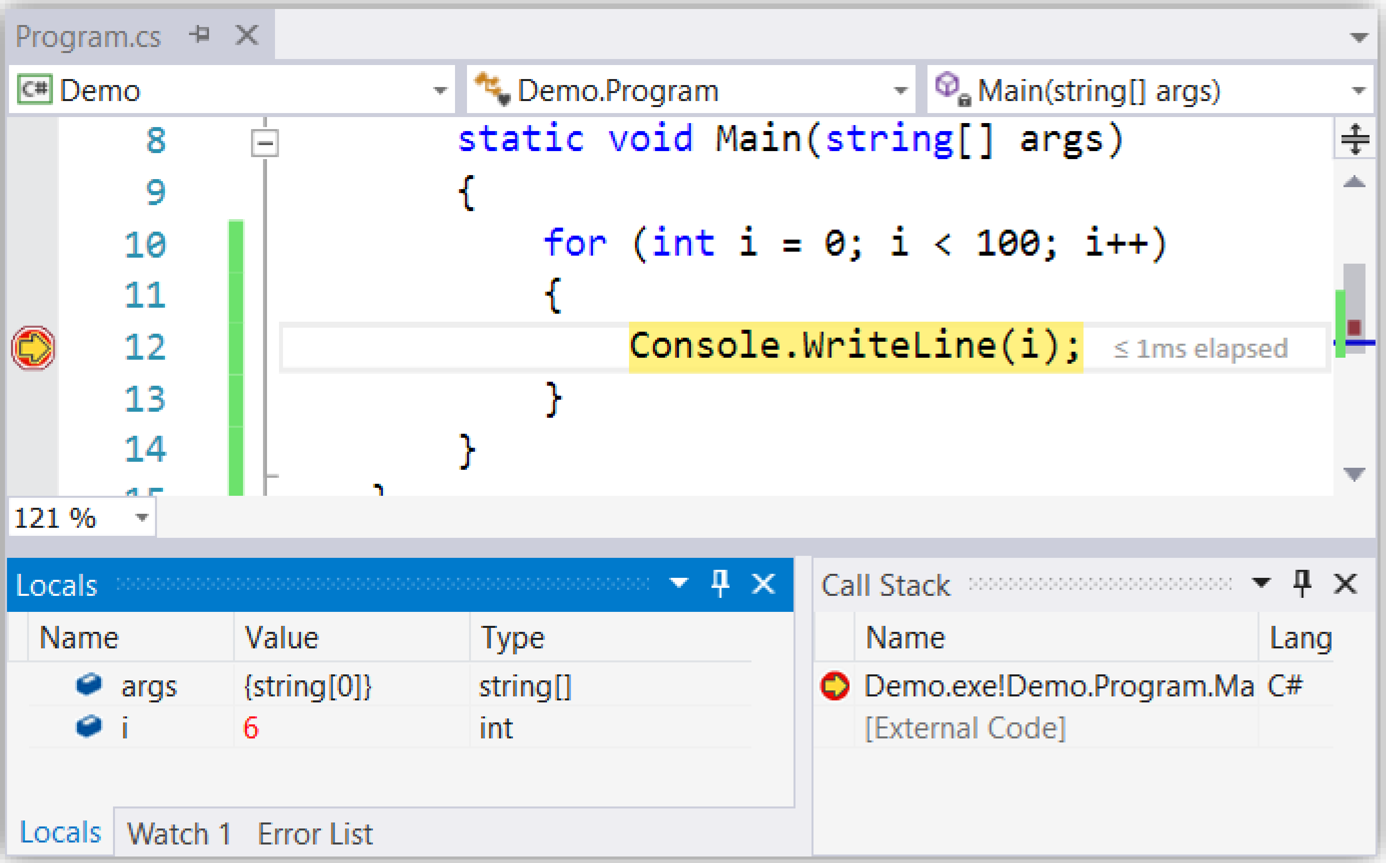
Debugging the Code

- The process of debugging application includes:
  - – Spotting an error
  - – Finding the lines of code that cause the error
  - – Fixing the error in the code
  - – Testing to check if the error is gone and no new errors are introduced
- Iterative and continuous process



Debugging in Visual Studio

- Visual Studio has a built-in debugger
- It provides:
  - – Breakpoints
  - – Ability to trace the code execution
  - – Ability to inspect variables at runtime



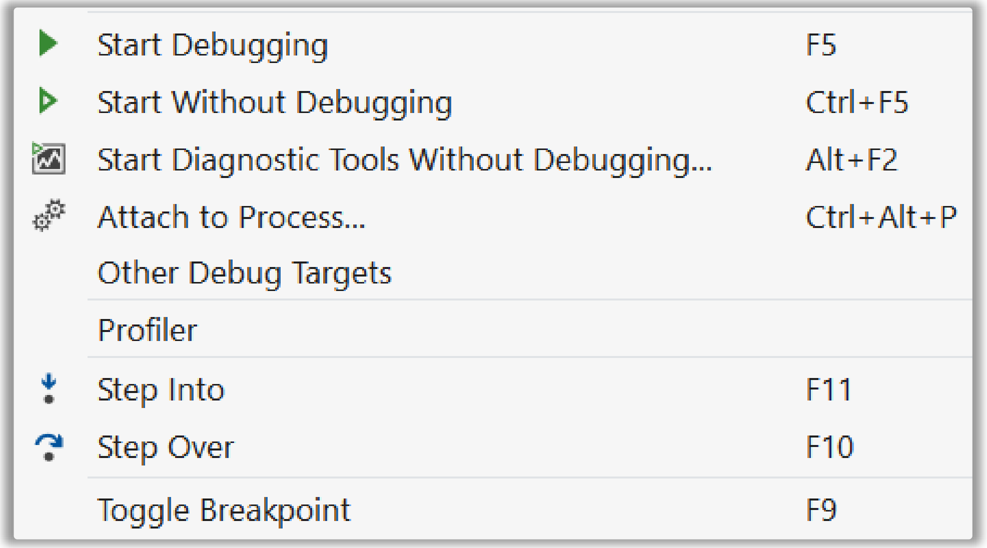
```
static void Main(string[] args)
{
    for (int i = 0; i < 100; i++)
    {
        Console.WriteLine(i);  ≤ 1ms elapsed
    }
}
```

Name	Value	Type
args	{string[0]}	string[]
i	6	int

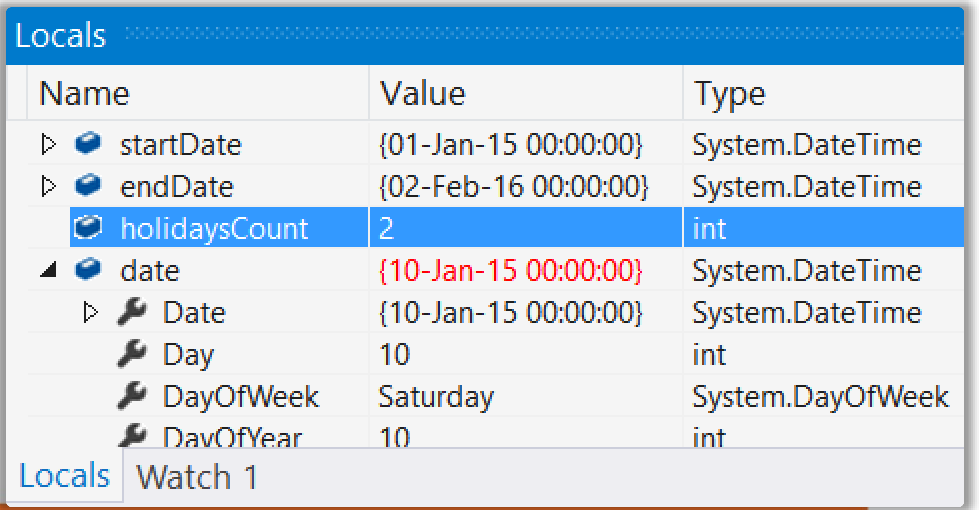
Name	Lang
Demo.exe!Demo.Program.Ma	C#
[External Code]	

Using the Debugger in Visual Studio

- Start without Debugger: [Ctrl+F5]
- Toggle a breakpoint: [F9]
- Start with the Debugger: [F5]
- Trace the program: [F10] / [F11]
- Using the Locals / Watches
- Conditional breakpoints
- Enter debug mode after exception



Start Debugging	F5
Start Without Debugging	Ctrl+F5
Start Diagnostic Tools Without Debugging...	Alt+F2
Attach to Process...	Ctrl+Alt+P
Other Debug Targets	
Profiler	
Step Into	F11
Step Over	F10
Toggle Breakpoint	F9



Locals

Name	Value	Type
startDate	{01-Jan-15 00:00:00}	System.DateTime
endDate	{02-Feb-16 00:00:00}	System.DateTime
holidaysCount	2	int
date	{10-Jan-15 00:00:00}	System.DateTime
Date	{10-Jan-15 00:00:00}	System.DateTime
Day	10	int
DayOfWeek	Saturday	System.DayOfWeek
DayOfYear	10	int

Locals Watch 1

Problem: Find and Fix the Bugs in the Code

- A program aims to count the non-working days between two dates (e.g. 1.05.2016 … 15.05.2016 → 5 non-working days). Debug it!

```
var startDate = DateTime.ParseExact(Console.ReadLine(),
  "dd.m.yyyy", CultureInfo.InvariantCulture);
var endDate = DateTime.ParseExact(Console.ReadLine(),
  "dd.m.yyyy", CultureInfo.InvariantCulture);
var holidaysCount = 0;
for (var date = startDate; date <= endDate; date.AddDays(1))
  if (date.DayOfWeek == DayOfWeek.Saturday &&
      date.DayOfWeek == DayOfWeek.Sunday) holidaysCount++;
Console.WriteLine(holidaysCount);
```

Summary

- Declaring Variables
- Using Console – Reading and Writing
- Conditional Statements allow implementing programming logic
- Loops repeat code block multiple times
- Using the debugger