Conditionals

A4 – Constructs and Techniques and Their Implementation in Programming

Ikromkhujaev Ilyoskhuja

Syntax

- In the previous chapter, we created a C# file called Program.cs, and we used the following code to print "Hello World" to the screen:

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

- Line 1: using System means that we can use classes from the System namespace.
- Line 2: A blank line. C# ignores white space. However, multiple lines makes the code more readable.
- Line 3: namespace is used to organize your code, and it is a container for classes and other namespaces.
- Line 4: The curly braces {} marks the beginning and the end of a block of code.
- Line 5: class is a container for data and methods, which brings functionality to your program. Every line of code that runs in C# must be inside a class. In our example, we named the class Program.

Syntax – Example Explained

Line 1: `using System` means that we can use classes from the `System` namespace.

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

Line 2: A blank line. C# ignores white space. However, multiple lines makes the code more readable.

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

Line 3: `namespace` is used to organize your code, and it is a container for classes and other namespaces.

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

Line 4: The curly braces {} marks the beginning and the end of a block of code.

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

Line 5: `class` is a container for data and methods, which brings functionality to your program. Every line of code that runs in C# must be inside a class. In our example, we named the class Program.

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

Line 7: Another thing that always appear in a C# program, is the `Main` method. Any code inside its curly brackets {} will be executed.

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

Line 9: `Console` is a class of the `System` namespace, which has a `WriteLine()` method that is used to output/print text. In our example it will output "Hello World!"

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

If you omit `the using System` line, you would have to write `System.Console.WriteLine()` to print/output text.

Program.cs

```
using System;

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

Result:

```
Hello World!
```

Syntax – Example Explained

- Every C# statement ends with a semicolon ;
- C# is case-sensitive: "MyClass" and "myclass" has different meaning.

Program.cs

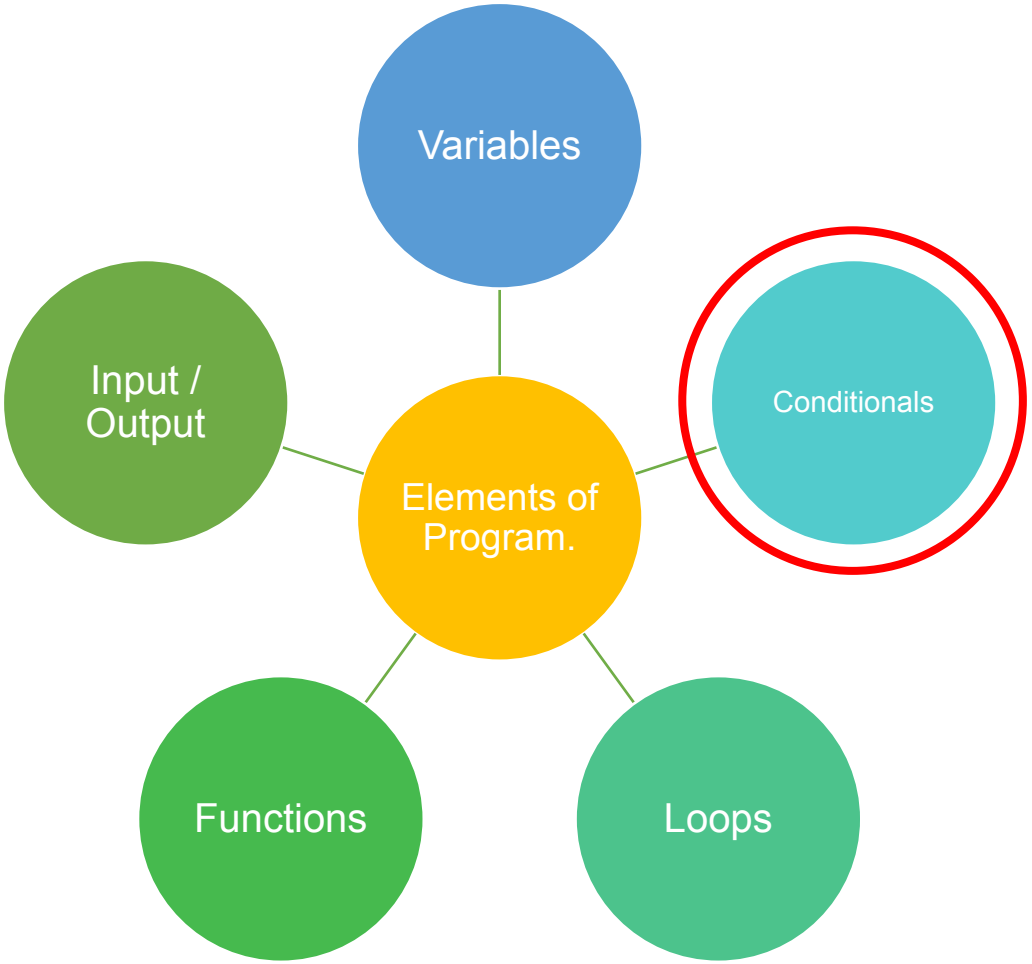
```
using System;

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

Result:

```
Hello World!
```

Elements

- To develop any instruction there are some elements needed or we can essentially present in all language.
- So any programming language is made up of 5 basic elements of the instructions.



Conditionals – If Statements

C# supports the usual logical conditions from mathematics:

- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`
- Equal to `a == b`
- Not Equal to: `a != b`

Conditionals – If Statements

You can use these conditions to perform different actions for different decisions. C# has the following conditional statements:

- `if` to specify a block of code to be executed, if a specified condition is true
- `else` to specify a block of code to be executed, if the same condition is false
- `else if` to specify a new condition to test, if the first condition is false
- `switch` to specify many alternative blocks of code to be executed

Conditionals – If Statements

Use the `if` statement to specify a block of C# code to be executed if a condition is `True`:

Syntax

```
if (condition)
{
  // block of code to be executed if the condition is True
}
```

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.

Conditionals – If Statements

Example

```
if (20 > 18)
{
  Console.WriteLine("20 is greater than 18");
}
```

Example

```
int x = 20;
int y = 18;
if (x > y)
{
  Console.WriteLine("x is greater than y");
}
```

Conditionals – Else Statements

Use the `else` statement to specify a block of code to be executed if the condition is `False`.

Syntax

```
if (condition)
{
  // block of code to be executed if the condition is True
}
else
{
  // block of code to be executed if the condition is False
}
```

Conditionals – Else Statements

Example

```
int time = 20;
if (time < 18)
{
  Console.WriteLine("Good day.");
}
else
{
  Console.WriteLine("Good evening.");
}
// Outputs "Good evening."
```

Conditionals – Else Statements

Use the `else if` statement to specify a new condition if the first condition is `False`.

Syntax

```
if (condition1)
{
  // block of code to be executed if condition1 is True
}
else if (condition2)
{
  // block of code to be executed if the condition1 is false and condition2 is True
}
else
{
  // block of code to be executed if the condition1 is false and condition2 is False
}
```

Conditionals – Else Statements

Example

```
int time = 22;
if (time < 10)
{
  Console.WriteLine("Good morning.");
}
else if (time < 20)
{
  Console.WriteLine("Good day.");
}
else
{
  Console.WriteLine("Good evening.");
}
// Outputs "Good evening."
```

Conditionals – Short Hand If…Else

There is also a short-hand if else, which is known as the **ternary operator** because it consists of three operands.

- It can be used to replace multiple lines of code with a single line.
- It is often used to replace simple if else statements:

Syntax

```
variable = (condition) ? expressionTrue :  expressionFalse;
```

Conditionals – Short Hand If…Else

Example

```
int time = 20;
if (time < 18)
{
  Console.WriteLine("Good day.");
}
else
{
  Console.WriteLine("Good evening.");
}
```

```
int time = 20;
string result = (time < 18) ? "Good day." : "Good evening.";
Console.WriteLine(result);
```

W3 schools

Conditionals – Switch Statements

Use the `switch` statement to select one of many code blocks to be executed.

Syntax

```
switch(expression)
{
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
    break;
}
```

Conditionals – Switch Statements

This is how it works:

- The `switch` expression is evaluated once
- The value of the expression is compared with the values of each `case`
- If there is a match, the associated block of code is executed
- The `break` and `default` keywords will be described later in this lesson

Syntax

```
switch(expression)
{
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
    break;
}
```

Conditionals – Switch Statements

The example below uses the weekday number to calculate the weekday name:

Example

```
int day = 4;
switch (day)
{
  case 1:
    Console.WriteLine("Monday");
    break;
  case 2:
    Console.WriteLine("Tuesday");
    break;
  case 3:
    Console.WriteLine("Wednesday");
    break;
  case 4:
    Console.WriteLine("Thursday");
    break;
  case 5:
    Console.WriteLine("Friday");
    break;
  case 6:
    Console.WriteLine("Saturday");
    break;
  case 7:
    Console.WriteLine("Sunday");
    break;
}
// Outputs "Thursday" (day 4)
```

Conditionals – Switch Statements

The Break Keyword

- When C# reaches a `break` keyword, it breaks out of the switch block.
- This will stop the execution of more code and case testing inside the block.
- When a match is found, and the job is done, it's time for a break. There is no need for more testing.
- A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block.

Example

```
int day = 4;
switch (day)
{
  case 1:
    Console.WriteLine("Monday");
    break;
  case 2:
    Console.WriteLine("Tuesday");
    break;
  case 3:
    Console.WriteLine("Wednesday");
    break;
  case 4:
    Console.WriteLine("Thursday");
    break;
  case 5:
    Console.WriteLine("Friday");
    break;
  case 6:
    Console.WriteLine("Saturday");
    break;
  case 7:
    Console.WriteLine("Sunday");
    break;
}
// Outputs "Thursday" (day 4)
```

Conditionals – Switch Statements

The default Keyword

- The default keyword is optional and specifies some code to run if there is no case match.

Example

```
int day = 4;
switch (day)
{
  case 6:
    Console.WriteLine("Today is Saturday.");
    break;
  case 7:
    Console.WriteLine("Today is Sunday.");
    break;
  default:
    Console.WriteLine("Looking forward to the Weekend.");
    break;
}
// Outputs "Looking forward to the Weekend."
```

Next lecture

- In the next lecture we continue focusing on «Constructs and techniques and their implementation»
- “Microsoft Visual C# Step by Step” Microsoft Press