C# Programming Essential

Tahaluf Training Center 2021

Day 5

1. **Methods Structure**
2. Method Parameters
3. Method Overloading

Methods Structure

A **method** is a block of code which only runs when it is called.

You can pass data, known as parameters, into a method.

Methods are used to perform certain actions, and they are also known as **functions**.

Why use methods? To reuse code: define the code once, and use it many times.

Create a Method

A method is defined with the name of the method, followed by parentheses (). C# provides some pre-defined methods, which you already are familiar with, such as Main(), but you can also create your own methods to perform certain actions:

```
class Program
{
static void MyMethod()
{
// code to be executed
}
}
```

MyMethod() is the name of the method.

static means that the method belongs to the Program class and not an object of the Program class. You will learn more about objects and how to access methods through objects later in this tutorial.

void means that this method does not have a return value.

To call (execute) a method, write the method's name followed by two parentheses () and a semicolon;

In the following example, MyMethod() is used to print a text (the action), when it is called:

```
static void Main(string[] args)
{
MyMethod();
}
```

Call a Method

In the following example, create and call a method:

```
static void MyMethod()
{
Console.WriteLine("Welcome to Tahaluf");
}


static void Main(string[] args)
{
MyMethod();
}
```

Exercise 1:

Define a function that reads your personal information from the keyboard and displays it on the screen in an elegant way.

Exercise Solution:

```
public static void UserDetails()
{
Console.WriteLine("Enter your name");
string name = Console.ReadLine();
Console.WriteLine("Enter your age");
double age = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("Name: {0}, Age: {1}", name, age);
}
```

Method Parameters

Information can be passed to methods as **parameter.** Parameters act as variables inside the method.

They are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a method that takes a string called fname and lname as parameter. When the method is called, we pass along a full name, which is used inside the method to print the full name:

```
static void MyName(string fname, string lname)
{
Console.WriteLine(fname + lname);
}

 MyName("Ahmad", "\tMohammed");
```

Exercise 2:

Create a recursive function to calculate the factorial of the number specified as parameter

Exercise Solution:

```
public static int Factorial(int num)
{
if (num == 0)
return 1;
else
return num * Factorial(num - 1);
}
```

Exercise Solution:

```
Console.WriteLine("Enter a number:");
int number = Convert.ToInt32(Console.ReadLine());
Console.WriteLine(Factorial(number));
```

Exercise 3:

Create a C# program that implements a function called IsAlphabetic that receives a text parameter from the user and check if the text has alphabetic characters from 'a' to 'z' including upper and lower case letters.

Exercise Solution:

```
public static bool IsAlphabetical(string text)
{
text = text.ToUpper();
for (int i = 0; i < text.Length; i++)
{
if (text[i] < 'A' || text[i] > 'Z')
{
return false;
}
}
return true;
}
```

Exercise Solution:

```
string text = Console.ReadLine();
Console.WriteLine(IsAlphabetical(text));
```

Exercise 4:

Create a program in C# that implements a function called Spaces that receives as a parameter a text requested from the user. Then show the total spaces that the text has.

Exercise Solution:

```
public static void Spaces(string text)
{
int count = 0;
string letter;
for (int i = 0; i < text.Length; i++)
{
letter = text.Substring(i, 1);
if (letter == " ")
{
count++;
}
}
Console.WriteLine(count);
}
```

Exercise Solution:

```
string text = Console.ReadLine();
Spaces(text);
```

Default Parameter Value

You can also use a default parameter value, by using the equals sign (=). If we call the method without an argument, it uses the default value (“Irbid"):

```
static void Country(string country = "Irbid")
{
Console.WriteLine(country);
}

Country("Amman");
Country("Irbid");
```

Method Overloading

With **method overloading**, multiple methods can have the same name with different parameters:

```
int MyMethod(int x)
float MyMethod(float x)
double MyMethod(double x, double y)
```

Method Overloading

Instead of defining two methods that should do the same thing, it is better to overload one.

In the example below, we overload the PlusMethod method to work for both int and double:

```
static int PlusMethod(int x, int y)
{
return x + y;
}
static double PlusMethod(double x, double y)
{
return x + y;
}
```

Method Overloading


```
int myNum1 = PlusMethod(8, 5);
double myNum2 = PlusMethod(4.3, 6.26);
Console.WriteLine("Int: " + myNum1);
Console.WriteLine("Double: " + myNum2);
```

Exercise 5:

Create a C# program that implements a three functions called Mul that receives a different number of parameters from the user and display the result of multiplication.

Exercise Solution

```
public static int Mul(int one, int two)
{
return one * two;
}
public static int Mul(int one, int two, int three)
{
return one * two * three;
}
public static int Mul(int one, int two, int three, int four)
{
return one * two * three * four;
}
```

Exercise Solution

```
Console.WriteLine("Multiplication of two numbers: " +
Mul(10, 15));
Console.WriteLine("Multiplication of three numbers: " +
Mul(8, 13, 20));
Console.WriteLine("Multiplication of four numbers: " +
Mul(3, 7, 10, 7));
```

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

On The E-Learning Portal