Functions

A function allows you to encapsulate a piece of code and call it from other parts of your code. You may very soon run into a situation where you need to repeat a piece of code, from multiple places, and this is where functions come in. In C#, they are basically declared like this:

<visibility> <return type> <name>(<parameters>)

{

<function code>

}

To call a function, you simply write its name, an open parenthesis, then parameters, if any, and then a closing parenthesis, like this:

PrintToConsole ();

Here is an example of our PrintToConsole () function:

//this is a void function

public void PrintToConsole()

{

    Console.WriteLine("I'm doing something...");

}

The first part, public, is the visibility, and is optional. If you don't define any, then the function will be private by default ([see access modifier figure](CSharpAccessModifier.PNG)). Next is the type to return. It could be any valid type in C#, or as we have done it here, void. A void means that this function returns absolutely nothing. Also, this function takes no parameters, as you can see from the empty set of parentheses.

**2 types of functions: Return and Void**

//this is a return function. Returns type int.

public int AddNumbers(int number1, int number2)

{

    int result = number1 + number2;

    return result;

}

We've changed almost everything. The function now returns an integer, it takes two parameters (both integers), and instead of outputting something, it makes a calculation and then returns the result. This means that we can add two numbers from various places in our code, simply by calling this function, instead of having to write the calculation code each time. While we don't save that much time and effort in this small example, you better believe that you will learn to love functions, the more you use C#. This function is called like this:

int result = AddNumbers(10, 5);

Console.WriteLine(result);

**Polymorphism in Functions**

**Polymorphism -> Same name, but many different things.**

**Function overloading -> Same function name, but many different functions.**

public void PrintToConsole()

{

    Console.WriteLine("I'm doing something...");

}

public void PrintToConsole(string message)

{

    Console.WriteLine(message);

}

public void PrintToConsole(int num)

{

    Console.WriteLine(num);

}

**Functions Pass by ref ->** [**See lecture on Value vs Reference types**](Value%20Types%20vs%20Reference%20Types%20and%20Functions%20Pass%20by%20Ref.docx)**.**