C# programming language. The beginning

Course

Programming Languages

Semester 2, FIIT

Mayer S.F.

Mikhalkovich S.S

LECTURE # 5. Functions

Functions

Lesson 5

Functions: in the same class as Main()

```
class Program
{
    static void Main()
    {
        Console.WriteLine(Min(3,2));
    }
    static int Min(int a, int b)
    {
        if (a < b)
            return a;
        else
            return b;
    }
}
```

Short function definition:

```
static void Main()
        {
            Console.WriteLine(Min(3, 2));
        }
static int Min(int a, int b) => (a < b) ? a : b;
```

Functions (2)

- Functions can be defined as class methods only.
- If function doesn't return a value then, type "void" is used.
- We use a "return" statement to return a value from a function

```
return expression;
```

- We can define function in the same class as Main() function or in another class
- We can use short definition of a function using =>
- All functions must be static **static**. It means that we can call them without a creation of an object
- Functions defined in one class are visible in another if they are defined with **public** modifier
- If function is defined in this class
- If function is called in the same class, the "**public**" modifier is not required

Access modifier

- *private* — most restrictive and allows access to the method only from within the containing class or struct
- *public* — allowing access from any code in the application
- *protected* — allows for access from within the containing class or from within derived classes
- *internal* — accessible from files within the same assembly
- *static* — indicates the method is a static member of the class rather than a member of an instance of a specific object.

Functions in the same class as Main()

```
class Program
{
    static void Main()
    {
        Console.WriteLine(Min(3,2));
    }
    static int Min(int a, int b)
    {
        if (a < b)
            return a;
        else
            return b;
    }
}
```

Short function definition:

```
static void Main()
        {
            Console.WriteLine(Min(3, 2));
        }
static int Min(int a, int b) => (a < b) ? a : b;
```

Functions in another class

```
internal class Program
    {
        static void Main()
        { Console.WriteLine(MyFuncs.Min(3, 2)); }
    }
class MyFuncs
    {
        public static int Min(int a, int b)
        {
            if (a < b) return a;
            else return b;
        }

    }
```

Short function definition

```
internal class Program
    {
        static void Main()
        { Console.WriteLine(MyFuncs.Min(3, 2));}
    }
class MyFuncs
    {public static int Min(int a, int b) => a < b ? a : b; }
```

Ref parameter (input-output)

```
static int Square(int a)
        {
            return a * a;
        }
static void Main(string[] args)
        {
            Console.WriteLine(5); // 25
        }
```

Common function which returns a value

Function with Ref (input-output) parameter

```
static void Square(ref int a)
        {
            a = a * a;
        }
static void Main(string[] args)
        {
            int a = 5;
            Square(ref a);
            Console.WriteLine(a); // 25
        }
```

Ref keyword must be in the signature

Ref parameter must be initialized

Ref keyword must be when calling

Out parameter (output)

```
static int Square(int a)
        {
            return a * a;
        }
static void Main(string[] args)
        {
            Console.WriteLine(5); // 25
        }
```

Common function which returns a value

Function with Out (output) parameter

```
static void Square(int a, out int result)
        {
            result = a * a;
        }
        static void Main(string[] args)
        {
            int a=5;
            int result;
            Square(a, out result);
            Console.WriteLine(result); // 25
        }
```

Out keyword must be in the signature

Out parameter must be defined

Out keyword must be when calling

Function overloading

```
static void Add(int a, int b)
        {
            Console.WriteLine(a+b);
        }
static void Add(int a, int b, int c)
        {
            Console.WriteLine(a + b + c);
        }
static void Add(double a, double b)
        {
            Console.WriteLine(a + b);
        }

static void Main(string[] args)
        {
            (int a, int b, int c) = (1,3,5);
            Add(a, b); // 4
            Add(a, b, c); // 9
        }
```

Generic function

```
class Program
{
    static void Main()
    {
        (int a, int b) = (3, 5);
        Swap(ref a, ref b);
    }
    static void Swap<T>(ref T a, ref T b)
    {
        var x = a;
        a = b;
        b = x;
    }
}
```

Summarizing

- Functions can be **generic**
- In C# the Swap function is absent (!)
- Parameters, passed by reference, can be "out" and "ref".
- **Out** – output, **ref** – input-output. ref parameters must be initialized before function call
- Keywords **out** & **ref** are using in call:

```
Square(5, out res);
Swap(ref a, ref b);
```

- We can define out-parameter in a call place (!):

```
Square(5, out var res1);
```

Q & A