Exploring C#'s Building Blocks

Paolo Perrotta

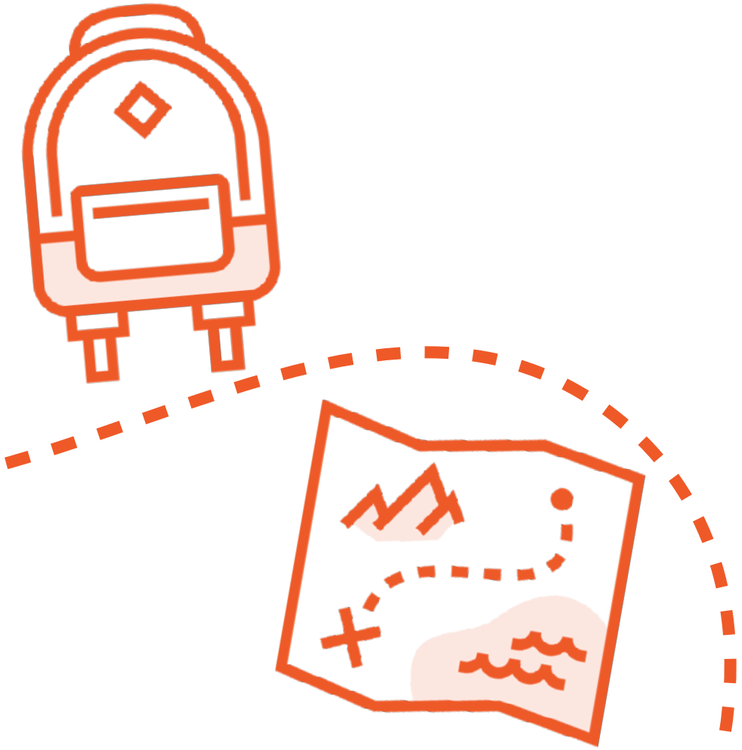
Freelance Developer

@nusco

A Short Hike

```
using System;

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



The General Concept of a Function

```
function_name(argument1, argument2, …) {
    result = …
    return result
}
```

A Function that Converts Fahrenheit to Celsius

```
fahrenheit_to_celsius(temperature_fahrenheit) {
    temperature_celsius = (temperature_fahrenheit - 32) / 1.8
    return temperature_celsius
}
```

Instead of traditional “functions”,
C# has static methods.

Static Methods and Classes

```
class Temperature
{
    static FahrenheitToCelsius(temperatureFahrenheit)
    {
        temperatureCelsius = (temperatureFahrenheit - 32) / 1.8;
        return temperatureCelsius;
    }
}
```

```
Temperature.FahrenheitToCelsius(32);
```

Classes are much more than
just containers of methods.

Floating Point Types

float	double
4 bytes	8 bytes
~6-9 digits precision	~15-17 digits precision
Based on .NET's *System.Single* type	Based on .NET's *System.Double* type

C# is fussy about types.

The Finished Method

```
static float FahrenheitToCelsius(float temperatureFahrenheit)
{
    float temperatureCelsius = (temperatureFahrenheit - 32) / 1.8f;
    return temperatureCelsius;
}
```

C# vs. JavaScript

C# (statically typed)

```
static float FahrenheitToCelsius(float temperatureFahrenheit)
{
    float temperatureCelsius = (temperatureFahrenheit - 32) / 1.8f
    return temperatureCelsius;
}
```

JavaScript (dynamically typed)

```
function FahrenheitToCelsius(temperatureFahrenheit)
{
    temperatureCelsius = (temperatureFahrenheit - 32) / 1.8;
    return temperatureCelsius;
}
```

“Static” Languages

On the minus side...

...they’re more verbose.

On the plus side...

...they’re safer.

C# is a “static” language.

Integer Types (Signed)

sbyte	**8 bits (from -128 to 127)**
short	**16 bits (from -32,768 to 32,767)**
int	**32 bits (from -2,147,483,648 to 2,147,483,647)**
long	**64 bits (from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)**

Integer Types (Unsigned)

byte	**8 bits (from 0 to 255)**
ushort	**16 bits (from 0 to 65,535)**
uint	**32 bits (from 0 to 4,294,967,295)**
ulong	**64 bits (from 0 to 18,446,744,073,709,551,615)**

Integer Types (Native)

nint	**It depends on the platform.**
nuint	**It depends on the platform.**

The two integer types you should remember about are *int* and *long*.

Decimal Types

float	**32 bits, floating point (~6-9 digits)**
double	**64 bits, floating point (~15-17 digits)**
decimal	**128 bits, fixed point (from ±1.0 x 10^{-28} to ±7.9228 x 10^{28})**

Other Built-in Types

string	**Sequences of characters**
char	**Unicode UTF-16 characters**
bool	**Booleans (either *true* or *false*)**

To be continued...

An Array of Strings

```
string[] interestingAnimals = { "Aardvark", "Binturong", "Pangolin" };
string anInterestingAnimal = interestingAnimals[1];
```

Back to “Hello, World!”

```
using System;

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

Summary

- Static methods
- Classes
- Types (and type safety)
- Built-in types
- Type inference
- Arrays
- Main()

Up Next:
Assembling a C# Program