C# Programming Tutorial
Lesson 1: Introduction to Programming

About this tutorial

This tutorial will teach you the basics of programming and the basics of the C# programming language.

If you are an absolute beginner this tutorial is suited for you.

If you already know one or more programming languages, you might find it a bit boring and skip to the next lesson.

To follow this tutorial you need to have Visual C# Express Edition 2008 or 2010 installed on your computer. These applications are free to download and install.

The best way to learn this is by practicing. Make sure you write all the examples yourself and test them, and that you do the tasks that I have put at the end. The tasks at the end will probably help you the most to get used to C#.

This tutorial has been entirely created by Davide Vitelaru (http://davidevitelaru.com/).

Note: You can use the table of contents at page 20 to get around the document quickly

Software required:

- Visual C# Express Edition 2008/2010

You must know:

- What programming is
- What a programming language is

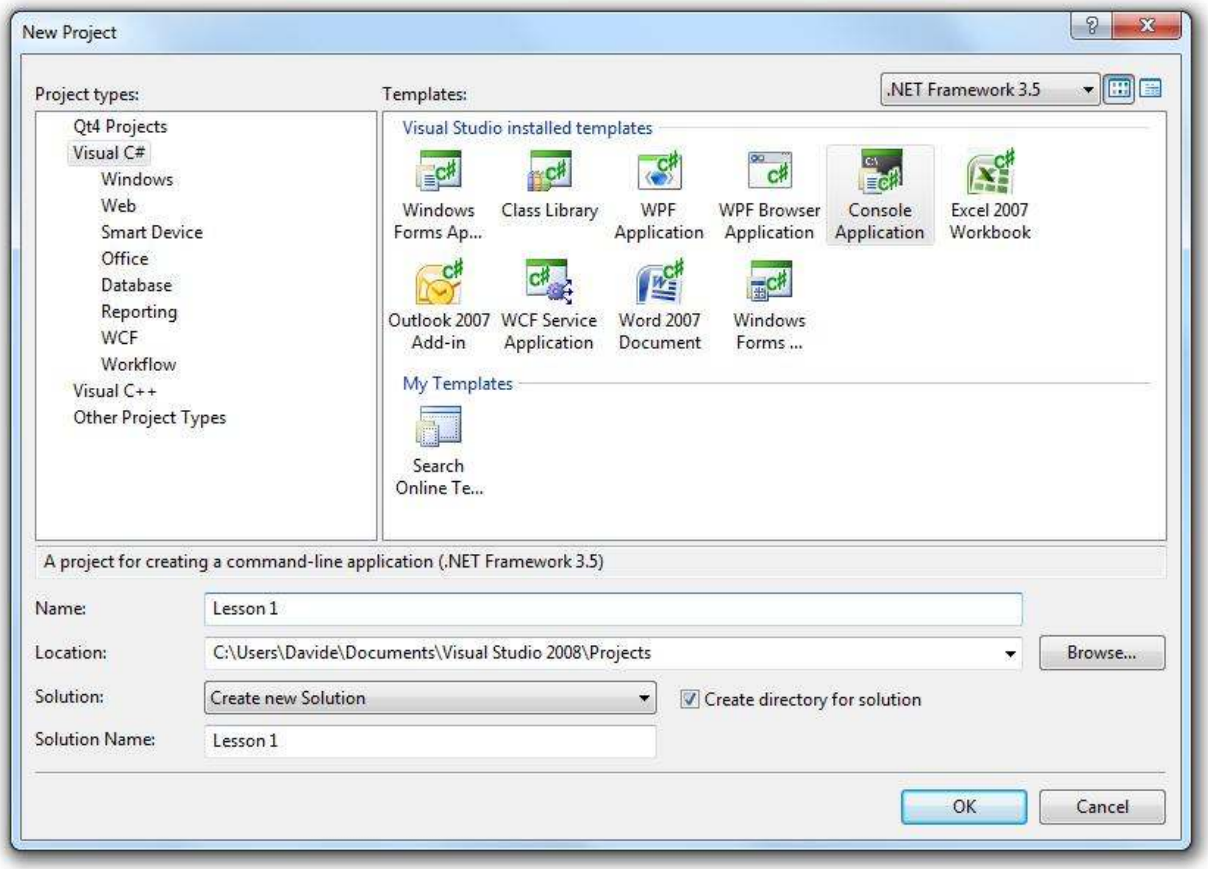
You will learn:

- Some Basics
- Variables
- Variable Operations
- Decisions
- Loops

Some Basics

Throughout this tutorial I will refer to Visual C# Express 2008/2010 as the IDE (Integrated Development Editor).

To start with, open your IDE and create a new project (File >> New >> Project or Ctrl + Shift + N). Select the **Visual C# Console Application** template from the window that appears and click OK:



.NET Framework 3.5

A project for creating a command-line application (.NET Framework 3.5)

Name: Lesson 1

Location: C:\Users\Davide\Documents\Visual Studio 2008\Projects

Solution: Create new Solution

☑ Create directory for solution

Solution Name: Lesson 1

Once you created your project, you will see this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Lesson_1
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

I know it looks scary, but it's not that complicated. You only have to worry about this section:

```
        static void Main(string[] args)
        {

        }
```

This is the exact place where you will write your source code, to be exact, between the braces following `static void Main(string[] args)`.

At this point, your application won't do anything. To start you application, press F5. You will see a black windows appearing and closing immediately.

It closes immediately because it does exactly what you told it to do: nothing. Let's "tell" it to open and wait for a keystroke to close.

Write the following line between the braces of `static void Main(string[] args)`:

```
            Console.ReadKey();
```

Now, press F5 to run your application. You will end up with a black window awaiting you to press any key so it closes.

Let's make it even more fun, make your code look like this:

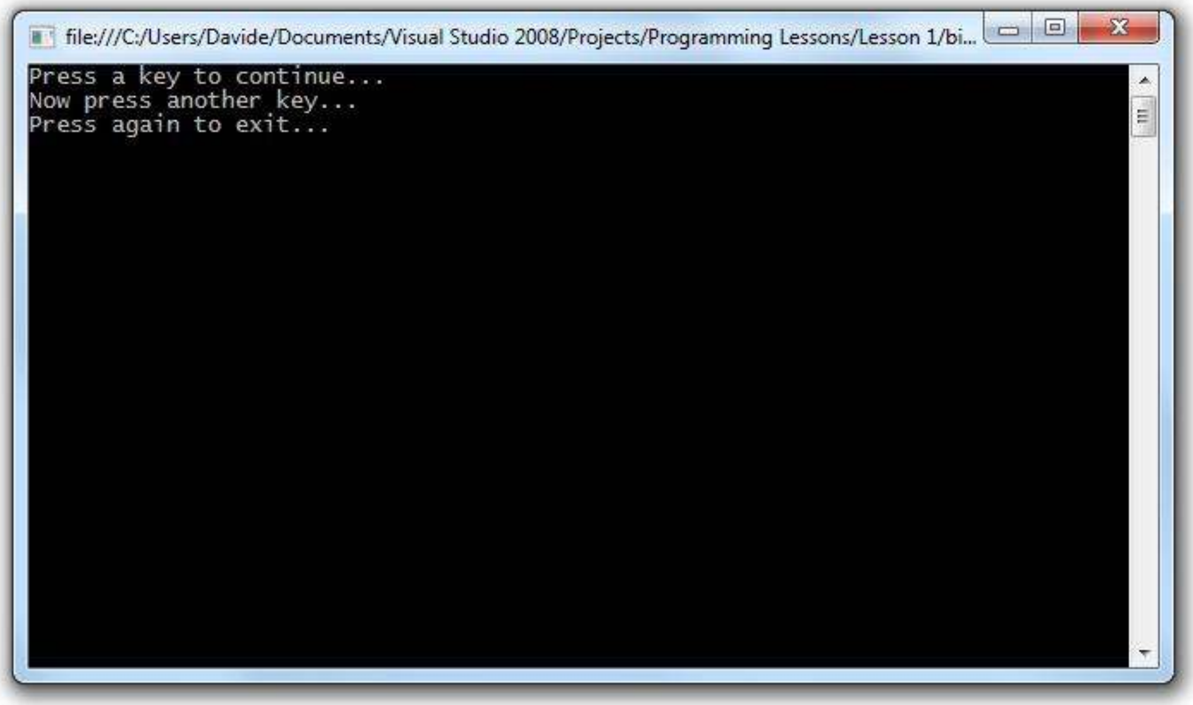
```
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Console.ReadKey();
        }
```

Again, press F5 to run your application. This time the application will display "Hello World" and then it will wait for you to press a key. If you get an error, make sure you typed everything correctly. Also, don't forget the semicolons at the end; they are very important (and annoying for beginners that keep forgetting them).

A statement can be used multiple times. Do the following:

```
        static void Main(string[] args)
        {
            Console.WriteLine("Press a key to continue...");
            Console.ReadKey();
            Console.WriteLine("Now press another key...");
            Console.ReadKey();
            Console.WriteLine("Press again to exit...");
            Console.ReadKey();
        }
```

Just change the text between the quotation marks in the `Console.WriteLine("")` statement to change the displayed message.


file:///C:/Users/Davide/Documents/Visual Studio 2008/Projects/Programming Lessons/Lesson 1/bi...

```
Press a key to continue...
Now press another key...
Press again to exit...
```

What's the catch with the black window?

The black window that you are currently working at is called a console window. Back in the 1980's computers didn't have taskbars and windows like they do now, the only had this text-based interface. Your application has a text-based interface at the moment.

Creating an application with a user interface (windows, buttons, text boxes, etc…) is usually harder, but thanks to Microsoft's .NET framework we can create one in a few easy steps; yet, that is not the point of this lesson.

This lesson is supposed to show you the basics, and once you finish it you will be able to move on to further lessons and create useful and good-looking applications.

Data manipulation

A program that displays messages and waits for keystrokes won't be of use to anyone, so let's make it do something useful. Let's make it add two numbers.

Variables

Variables are like boxes, you can put things in them. In our case, we will use them to store values.

Variables are of different types, depending on the type it can store different values, for example and **integer** variable can hold a number, while a **string** can hold characters (ex. "hello my name is john" – 21 characters, spaces included).

To start with, let's use variables display information:

```
        static void Main(string[] args)
        {
            string name;
            name = "John";
            Console.WriteLine(name);
        }
```

Press F5, run your application and see the result. If you receive an error, make sure you typed everything correctly.

How does it work?

To use a variable, we must first create it. To create it (a better term would be to "declare" it), you must type the variable type, followed by the name you want the variable to have:

```
            string variable;
            int another_variable;
```

At this point, both of these variables are empty. To assign a value to a variable, type the name of the variable, equal and the value you want it to hold. If it is a string, never forget to type the value between quotation marks:

```
           variable = "hello there";
           another_variable = 22;
```

Make sure you assign the correct type of value to the variable, or you will receive an error; In this case `variable` is a `string` so it can hold a string value, and is `another_variable` an `int`eger so it can hold a number. You can name the variables however you like as long as you don't use reserved words (like `int`, you can't do `int int` because it would return an error), and the name doesn't contain some particular symbols, and the name doesn't start with a number.

Let's make the computer ask for our name, and then greet us: