

# Introduction to C# Programming

By: Kristofer Gafvert

## **Copyright Information**

Copyright © 2004 Kristofer Gafvert (kgafvert@ilopia.com). No part of this publication may be transmitted, reproduced, or republished in any way, without written permissions by the author. The only website that is allowed to publish this document is ilopia.com, and its sub domains. If this document was downloaded from another website, please contact the author by using the email address above.

If any of these rules are broken, legal actions will be taken for plagiarism. Plagiarism is against the law!

## **Version**

0.3.3 – April 04, 2004

## 1. Table of Contents

Copyright Information .....	2
Version .....	2
1. Table of Contents .....	3
2. Introduction .....	4
2.1. Assumptions .....	4
3. Your first application .....	4
3.1. Hello World .....	4
3.2. It is not working! .....	5
3.3. The code .....	7

## 2. Introduction

The first section of this tutorial will get you started with C# (pronounced ‘see-sharp’). I will not use an IDE like Visual Studio .NET for this. I have chosen to use a simple text editor (I will use ConTEXT, but you can use any text editor you like, but I recommend one with syntax highlighting and auto indent), because it is important that you learn how to write code, and not drag and drop components. This will give you a better understanding of programming, and the code will be written in a much more structured way than you would write it in if you started with an IDE. Later, when you know how to write code, I encourage you to use an IDE such as Visual Studio .NET. It will speed up the code writing, but I do not recommend using it until you are working on “real” projects.

In this first section, you will also only write plain DOS-like applications. The purpose of the first section is to teach you how to program, and teach you the language, not write enterprise applications. Before writing Windows-application, you must know the language, and you will learn the language much faster if you stick to simple applications.

### 2.1. Assumptions

I assume that you have basic knowledge about Windows. I also assume that you have some kind of previous programming experience, so you know the basic concepts. For example what it means to compile a program, what a compile error is, what a run-time error is and so on. You can probably follow this tutorial without this knowledge, but since I am not explaining this in detail, you might have problems to understand what is happening. I do NOT assume that you have previous C# programming experience.

## 3. Your first application

In this chapter you will write your first application. I will also give you some basic troubleshooting guidelines, if your application does not compile correctly.

### 3.1. Hello World

It would probably be a good idea to start with a simple “Hello World”-application (couldn’t resist starting with the classic one).

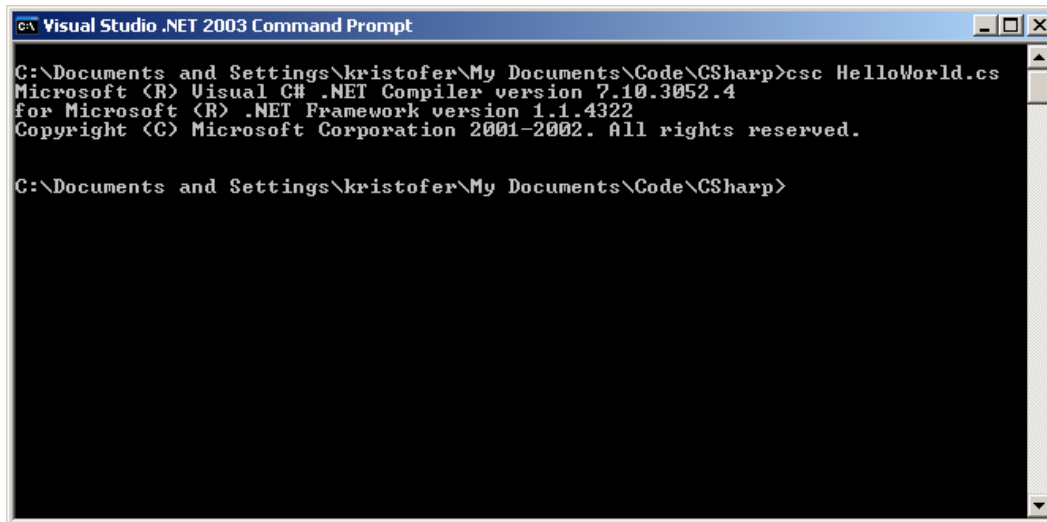
```
using System;
```

```
namespace HelloWorldApplication
{
    /* The Hello World class */
    class HelloWorld
    {
        static void Main(string[] args)
        {
            //Write "Hello World" to the screen
            Console.WriteLine("Hello World!");
        }
    }
}
```

Save this code as HelloWorld.cs (if you are using NotePad, make sure that you do not get the extension cs.txt). After you have saved the file, open a command prompt (Start->Run, type "cmd"). Go to the folder that you saved the file to, and write:

```
csc HelloWorld.cs
```

If you are lucky, you will get this screen:



```
Visual Studio .NET 2003 Command Prompt
C:\Documents and Settings\kristofer\My Documents\Code\CSharp>csc HelloWorld.cs
Microsoft (R) Visual C# .NET Compiler version 7.10.3052.4
for Microsoft (R) .NET Framework version 1.1.4322
Copyright (C) Microsoft Corporation 2001-2002. All rights reserved.

C:\Documents and Settings\kristofer\My Documents\Code\CSharp>
```

This one tells you that your code compiled, and that you are ready to run the application. To run the application, type:

```
HelloWorld.exe
```


If everything went as expected, it should now write "Hello World!" on the screen.

### 3.2. *It is not working!*

What if it did not work as I said? Don't worry, I will tell you about some common errors.

**'csc' is not recognized as an internal or external command,  
operable program or batch file.**

If you get the above message, there are two things that could be wrong. The first one is that you have not installed the .NET Framework SDK. In order to run and compile the applications we write in this tutorial, you need the .NET Framework SDK. You can download it from <http://go.microsoft.com/fwlink/?LinkId=8684> . Make sure that you download the latest edition of .NET Framework SDK.

 **If you have Visual Studio .NET installed, you already have what you need installed, but you will still get the same error message. Start the Visual Studio Command Prompt in the Visual Studio .NET Program Group, and it will work.**

If you still have the above problem after installing the .NET Framework SDK, the folder where the csc executable is located is probably not included in the PATH environment variable. To fix this, follow these steps (for Windows XP, if you are using another edition of Windows, and the steps are not the same, please use the Windows documentation to find out how to do this):

- ✓ Right click “My Computer” and click Properties
- ✓ Click on the “Advanced” tab
- ✓ Click the button “Environment Variables”
- ✓ In the “System Variables” section, find PATH. Click on it, and click on the “Edit” button
- ✓ Write: ;%systemroot%\Microsoft.NET\Framework\v1.1.4322 at the end of the list. The string starts with a semicolon (that is used to separate the strings in the PATH variable), and make sure that you type the correct version (when you read this, the version I wrote might not be the latest one).
- ✓ Click “Ok” three times, and now start a new command prompt.
- ✓ Compile again, and this time it should work

It is also possible that you get a compile error. Say for example that you forgot to end the line that writes to the console, with a semicolon. Then you would get an error similar to this one:

**HelloWorld.cs(11,43): error CS1002: ; expected**

This tells you that you have an error on line 11, column 43, and that the compiler thinks that you forgot a semicolon. Note however that you cannot always trust what is written here, the compiler could be wrong about what in your code is wrong. As a general rule, start where the compiler says that you have an error, and then walk upwards, until you find an error. If the compiler says that you have several errors, start with the first one, all other errors might be fixed automatically by this :-)

### 3.3. The code

So, what did we really do? Let's walk through the code, so that you get a brief introduction (all this will be explained more deeply later) to C# coding.

If you have previously used C++, you would probably notice that this simple C# code already involves classes, whereas in a C++ programming book, you would not see classes until chapter 5 (or something like that). C# is an object oriented language, and everything must be written in classes. It is not possible, like in C++, to have a Main method without a class. If you try to remove the class, and compile the Main method only, you will get an error similar to:

**HelloWorld.cs(8,33): error CS1518: Expected class, delegate, enum, interface, or struct**

The namespace HelloWorldApplication contains our class HelloWorld. A namespace can be compared to a package in Java. A namespace is simply a way to group elements (classes, other namespaces) that belongs together. It is also a way to distinguish two classes with the same name. For example, company A writes their own HelloWorld class, whereas company B writes another HelloWorld class. It would not be possible to use these both classes if they did not exist in different namespaces, because the compiler would not know which class we wanted to use.

As you can understand, putting everything in different namespaces, can result in quite lengthy names. And more words, means more spelling errors. So of course, there is a way to write less, we can use the "using" directive, as we do on the first line.

"Console" is a class in the "System" namespace, and instead of "using System", we could have written the fully qualified name:

```
System.Console.WriteLine("Hello World!");
```

This would give us the very same result, but why write "System" all the time when we do not have to?

You have probably also noticed Main with a capitalized M. This is not a typo! A design guideline (for C#) is to begin all method names with a capitalized letter, and this applies to the Main method as well. So if you are a C++ developer, do not forget that this differs! And as you have probably already guessed, C# is cAsE sEnSiTivE.

You can also see two different kinds of comments in the example code. Exactly as for C++, you have the single line comment that starts with //, and the multi line comment beginning with /\* and ends with \*/. C# also has one other set of comments, the single line comment /// and the multi line comment that starts with /\*\* and ends with \*/, that can

## Introduction to C# Programming by Kristofer Gafvert

<http://www.ilopia.com>

generate HTML documentation when you run csc with the doc switch. This will be explained better in a later chapter.

So, that is it! You have written your first application using C#, and you hopefully understand most of the code.