What is C#?

So, you want a C# reference? OK, well the best place to begin is by looking at what C# is and where it came from.

The Name

First off, the name. According to the ECMA-334 C# Language Specification (`http://www.ecma-international.org/publications/standards/Ecma-334.htm`), the name is combined of a Latin capital letter C (U+0043) followed by the number symbol # (U+0023). C# is pronounced "C sharp" or "see sharp."

The origin of the name is somewhat shrouded in mystery. Some believe that it may have been chosen by Microsoft to imply a progression from C++, with the # symbol composed of four + symbols arranged to form a square. Another origin for the name could be more musical, implying that it's not as far from C as C++ is, because ++ is the symbol for the increment operator. In music, a # indicates a note that is one half step above the other, so C# might show that it is only a half step above C.

The musical readers among you might have recognized that the # symbol on the keyboard is not the proper symbol for sharp. It is instead the number sign. This is used because the symbol for a musical sharp (U+266F) is not present on a standard keyboard, so expecting people to type it would be a bit of an inconvenience. Despite this symbol being used, the language is not called "see pound" or "see hash" or even "see gate"!

C# Overview

C# is an object-oriented programming language developed by Microsoft to become a key part of their .NET software development platform. Being object-oriented, C# is composed of a collection of individual programming units called classes that can interact with each other.

C# is based on the C++ language, but there is no doubt that it was influenced by Microsoft's other popular language, Visual Basic. One of the biggest advantages of C# is that its syntax (in other words, the structure of the code) is similar to that of a number of other popular programming languages, notably C++, Visual Basic, Java, and Delphi, which means that programmers from a variety of backgrounds can start programming with minimal learning. It is, however, simpler than C++ and Java.

History

C#'s principal designer at Microsoft was Anders Hajlsberg. Hajlsberg brought to Microsoft considerable experience from Borland, where he wrote a Pascal compiler way back in the 1980s. In 1996 Hajlsberg left Borland to go to Microsoft, where he developed J++ and the Windows Foundation Classes before going to work on C# and the Common Language Runtime (CLR), the virtual machine and runtime library that is the cornerstone of .NET. (The .NET Framework allows code to be run on the host system). Hajlsberg had been very critical of the flaws present in languages such as C++, Delphi, Java, and Smalltalk, and these were in part what drove him to develop a better language — C#. This also explains why C# shares a number of similarities with C++, Delphi, and Java, to name but a few.

C# and CLR

C# was designed to take advantage of the Common Language Runtime that .NET programs all rely upon. All applications written in C# require the CLR (in other words, the Microsoft .NET framework) to run, just as Visual Basic applications needed the appropriate runtime library to run.

Information on the .NET Framework, along with download information, can be found at the Microsoft website: `http://msdn.microsoft.com/netframework/`.

The main features of the CLR include:

- **Managed code.** Managed code outputted by Visual Studio applications and is run by the .NET Framework.
- **Easy/automatic application installation.** This can be carried out using Global Assembly Cache.
- **Memory management.** The CLR offers programmers an easy yet effective way to manage memory. This means better performance with less code.
- **Automatic garbage collection.** The .NET Framework automatically frees up memory when objects are no longer required.
- **Excellent levels of security during execution.** The .NET Framework includes an integrated security model that grants permission to resources based on evidence found in assemblies.

Diversions Into .NET

Just a quick diversion into .NET.

The Microsoft .NET platform has four cornerstone components:

- .NET Building Block Services such as Passport
- .NET Compact Framework which runs on devices such as mobile phones and PDAs
- .NET user experience through XML integration (forms and so on)
- .NET Infrastructure such as the .NET Framework Common Language Runtime and .NET Framework Class Libraries and development applications such as Microsoft Visual Studio.NET

All the .NET programming languages have the .NET Framework class libraries integrated into them. The .NET class libraries also support functions such as file I/O, database operations, XML (Extensible Markup Language) and SOAP (Simple Object Access Protocol).

The important thing to remember about .NET programming or .NET development is that this means leveraging the .NET Framework, which includes the runtime environment and the class libraries.

Standards

One of the great things about C# is that Microsoft submitted the language to ECMA (European Computer Manufacturers Association) for format standardization.

In December 2001, ECMA released the ECMA-334 C# Language Specification, and in 2003, C# became an ISO standard (ISO/IEC 23270).

The ECMA-334 language specification can be downloaded free of charge from the ECMA website: `http://www.ecma-international.org/publications/standards/Ecma-334.htm`.

The ISO/IEC 23270 standard is available for purchase from the ISO website (`http://www.iso.org`) or an electronic version can be downloaded free of charge.

In Visual Studio 2005, Microsoft added support to C# for generics, partial types, and other features. While standardization has been proposed for these features, they are not currently part of the specification.

Other Implementations

C# has evolved from just being a Microsoft language to the point where there are independent implementations of C# in development. Two of the biggest are:

- DotGNU — `http://www.dotgnu.org/`
- Mono — `http://www.gotmono.com/`

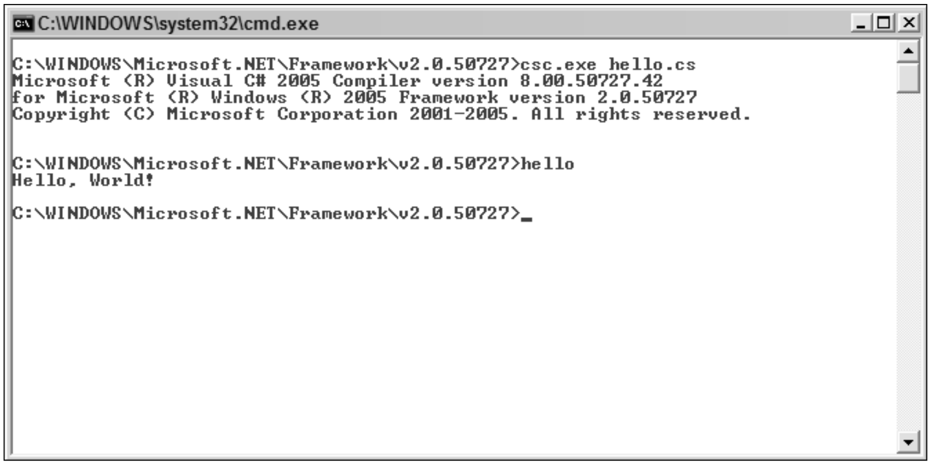
It's great to see a flourishing community build up around C#. This will give programmers wanting to make use of C# greater choice and flexibility. As with all independent implementations, however, you have to expect a certain amount of drift from the standards.

Sample C# Code

So, what does C# code look like? Well, we'll be looking at C# code a lot later in this book, but to begin with, here's a simple "Hello, World!" sample:

```
public class MyClass
{
    public static void Main()
    {
        System.Console.WriteLine("Hello, World!");
    }
}
```

What will this code do when it has been compiled? Nothing exciting, just output the text "Hello, World!" to the output console (as shown in Figure 1-1).



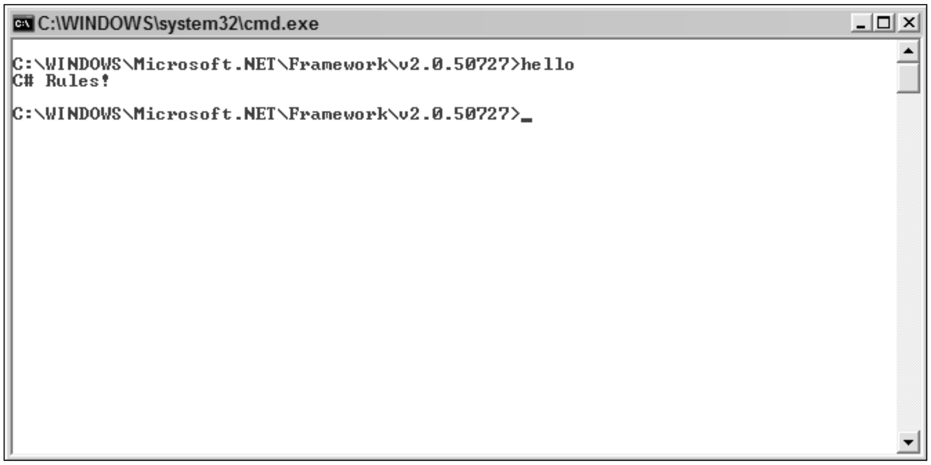
```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>csc.exe hello.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.


C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>hello
Hello, World!

C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>_
```

Figure 1-1

The great thing about C# is that even if you knew nothing about the language, you could probably figure out how to change the message displayed on the screen to say something else with little or no difficulty. For example:

```
public class MyClass
{
    public static void Main()
    {
        System.Console.WriteLine("C# Rules!");
    }
}
```

This simple change changes the message displayed onscreen (see Figure 1-2).

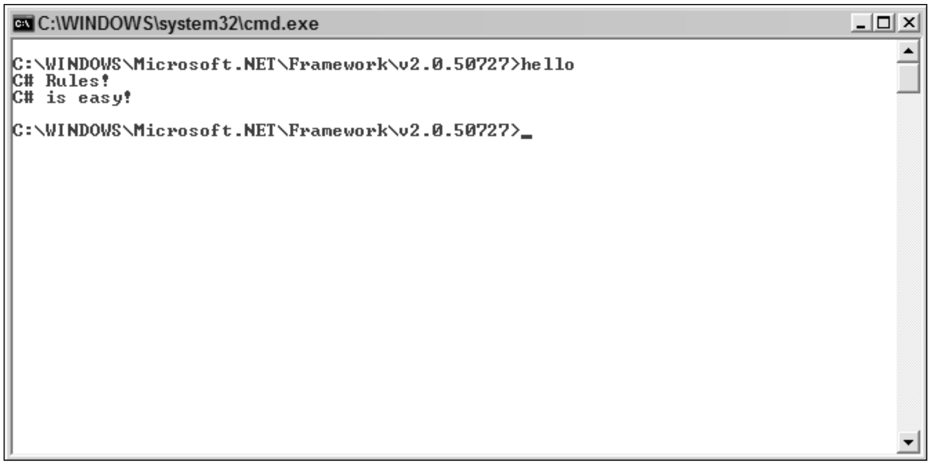


```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>hello
C# Rules!

C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>_
```

Figure 1-2

The simplicity of C# would also allow someone with very little experience to change the code to allow for multiple lines of text to be displayed (see Figure 1-3).

```
public class MyClass
{
    public static void Main()
    {
        System.Console.WriteLine("C# Rules!");
        System.Console.WriteLine("C# is easy!");
    }
}
```



```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>hello
C# Rules!
C# is easy!

C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>_
```

Figure 1-3

That's pretty simple stuff, even for a total beginner to grasp! Such ease of understanding is one of the elements that have made C# such a popular programming language.

Don't think that it's just simple stuff that C# is up to; this is merely the tip of the iceberg. C# is a full-featured and robust programming language that's up to any task to which you set it.

The Benefits of Learning C#

So, what are the advantages of taking the C# route?

Well, as you have just seen, the main advantage that C# offers is a far shallower learning curve than that presented by other languages. Anyone with even a casual background in C, C++, or Java will have minimal problems with C#. C# also makes it easy for those with background in JavaScript, Visual Basic, or even VBScript to make the transition.

Venturing into the realm of opinion (and your mileage may vary on this), we find that C# even beats Visual Basic .NET because C#'s language is a lot less verbose, which makes even complicated programs seem more readable and concise.

Summary

This chapter provided a very quick look at what C# is. You examined the origin of its name and had a very quick tour of the language, starting with its history and moving on to look at how C# fits in with Microsoft .NET.

You then took a look at the standards behind C# and discovered that there are implementations of C# by groups and companies other than Microsoft.

Finally, you saw some very simple C# code (just to get some code into this chapter!) before looking at the benefits of learning C#.

With all that out of the way, Chapter 2 looks at how you can get started using C#! We think you'll be surprised just how little you need!