# Introduction to the C# Language

## Topics Covered

- C# Features
- Integrated Development Environments
- Installing Visual Studio
- Creating a Console Application
- Parts of a C# Program
- Compiling a Program
- C# Keywords and Special Symbols
- Indenting Code
- Programming Errors

## Why C#?

C# was originally invented to compete with Java.

Modern C# is similar to Java, but offers simplified syntax, allowing programmers to do more with less code.

C# programs can be developed using libraries written in any .NET language, including Visual Basic, C++, and many others.

C# programs can run on all common platforms, including Apple and Android devices.

The Framework Class Libraries are open-source.

C# programs are compiled (not interpreted), allowing them to run with maximum speed.

C# offers numerous code-security features.

For developers who need the “pedal to the medal” capabilities of C and C++, C# can provide direct access to memory and other operating system resources.

## C# Application Types

Using C#, you can build:

- Stand-alone Console and Windowed applications;
- Applications that run in a browser;
- Applications for Android iOS devices;
- Web applications.

# C# History

| Version | .NET Framework | Visual Studio | Important Features |
|---|---|---|---|
| C# 1.0 | .NET Framework 1.0/1.1 | Visual Studio .NET 2002 | Basic features |
| C# 2.0 | .NET Framework 2.0 | Visual Studio 2005 | Generics<br>Partial types<br>Anonymous methods<br>Iterators<br>Nullable types<br>Private setters (properties)<br>Method group conversions (delegates)<br>Covariance and Contra-variance<br>Static classes |
| C# 3.0 | .NET Framework 3.0\3.5 | Visual Studio 2008 | Implicitly typed local variables<br>Object and collection initializers<br>Auto-Implemented properties<br>Anonymous types<br>Extension methods<br>Query expressions<br>Lambda expressions<br>Expression trees<br>Partial Methods |

| Version | .NET Framework | Visual Studio | Important Features |
|---|---|---|---|
| C# 4.0 | .NET Framework 4.0 | Visual Studio 2010 | Dynamic binding (late binding)<br>Named and optional arguments<br>Generic co- and contravariance<br>Embedded interop types |
| C# 5.0 | .NET Framework 4.5 | Visual Studio 2012/2013 | Async features<br>Caller information |
| C# 6.0 | .NET Framework 4.6 | Visual Studio 2013/2015 | Expression Bodied Methods<br>Auto-property initializer<br>nameof Expression<br>Primary constructor<br>Await in catch block<br>Exception Filter<br>String Interpolation |
| C# 7.0 | .NET Core | Visual Studio 2017 | out variables<br>Tuples<br>Discards<br>Pattern Matching<br>Local functions<br>Generalized async return types<br>throw Expressions |

platform By Per Scholas

## The C# Language Is ...

- A C-Family Language
- Object Oriented
- Distributed
- Compiled
- High-Performance
- Robust
- Secure
- Architecture Neutral
- Portable
- Multithreaded

# A C-Family Language

- **A C-Family Language**
- Object Oriented
- Distributed
- Compiled
- High-Performance
- Robust
- Secure
- Architecture Neutral
- Portable
- Multithreaded

If you are familiar with C, C++, Java, or even Javascript, much of the C# sytnax will be familiar to you.

C# adds automatic memory-management while also adding syntax that avoids the most common pitfalls of C and C++.

# An Object-Oriented Language

- A C-Family Language
- **Object Oriented**
- Distributed
- Compiled
- High-Performance
- Robust
- Secure
- Architecture Neutral
- Portable
- Multithreaded

C# was designed as an Object-Oriented Language from the get-go. All C# code is written in the context of a class.

C# classes support all the OOP features: inheritance, encapsulation, polymorphism, and composition.

C# provides rich capabilities for defining abstractions – abstract classes and interfaces.

Together with broad support for generic classes and methods, C# maximizes code-reusability.

# Distributed Computing

- A C-Family Language
- Object Oriented
- **Distributed**
- Compiled
- High-Performance
- Robust
- Secure
- Architecture Neutral
- Portable
- Multithreaded

Distributed computing involves multiple processes on one or more machines working together in coordinated fashion.

Networking classes in the Framework Class Library support inter-process communication through a variety of protocols, including HTTP, FTP, TCP, named pipes, and custom protocols using sockets.

Sending data “over the wire” is accomplished as easily as writing data to a file.

## A Compiled Language

- A C-Family Language
- Object Oriented
- Distributed
- **Compiled**
- High-Performance
- Robust
- Secure
- Architecture Neutral
- Portable
- Multithreaded

The C# compiler becomes active as soon as you begin writing code. It analyzes the code syntax while you type, helping you to discover errors early.

When your code is complete, compilation happens in two steps. In the first step, you code is converted to Intermediate Language (IL), a language similar to assembly language. IL is the same for all .NET languages.

The second step occurs just before the code executes, where a “just-in-time” or JIT-compiler converts the IL to machine code appropriate to the current platform. This leads to the fastest possible execution.

The C# compiler is even written using C#!

# High-Performance Computing

- A C-Family Language
- Object Oriented
- Distributed
- Compiled
- **High-Performance**
- Robust
- Secure
- Architecture Neutral
- Portable
- Multithreaded

C# is compiled to machine code, allowing the fastest possible execution times.

The Common Language Runtime implements automatic garbage collection, ensuring that memory and other system resources are reclaimed.

For computationally intensive programs, C# provides numerous constructs to easily take advantage of multiple processors and cores.

# Robustness

- A C-Family Language
- Object Oriented
- Distributed
- Compiled
- High-Performance
- **Robust**
- Secure
- Architecture Neutral
- Portable
- Multithreaded

The C# compiler is thorough and provides the first layer of defense against programming errors.

Visual Studio provides tools for static code analysis which identify code constructs and patterns known to be sources of error.

C# provides rich capabilities for exception handling, allowing for robust handling of unforeseen run-time errors.

## Code Security

- A C-Family Language
- Object Oriented
- Distributed
- Compiled
- High-Performance
- Robust
- **Secure**
- Architecture Neutral
- Portable
- Multithreaded

C# code runs in an environment called the Common Language Runtime (CLR).

The CLR is a type of “sandbox” which allows access to only those operating system features to which the person who started the program is entitled.

If a program requires access to administrative functionality, C# provides several mechanisms for requesting this access.

# Architecture Neutrality

- A C-Family Language
- Object Oriented
- Distributed
- Compiled
- High-Performance
- Robust
- Secure
- **Architecture Neutral**
- Portable
- Multithreaded

The earliest versions of C# shipped with the .NET Framework which targeted Windows operating systems.

The popularity of the C# language has led to two open-source implementations of the compiler and the Framework Class Library.

Mono allows creation of windowed applications on Windows, IOS and Android platforms.

.NET Core enables us to write ASP.NET web-server applications for Windows, Linux and Mac hosts.

# C# Portability

- A C-Family Language
- Object Oriented
- Distributed
- Compiled
- High-Performance
- Robust
- Secure
- Architecture Neutral
- **Portable**
- Multithreaded

Some C# programs written using the .NET Framework make use of Windows-specific features, such as interop calls to the Windows API. These powerful features are not portable.

If instead you write C# code using the .NET Core or Mono libraries, you code will run on Windows, Linux, Mac, Android and even microprocessors such as ARM.

# C# Multi-threading

- A C-Family Language
- Object Oriented
- Distributed
- Compiled
- High-Performance
- Robust
- Secure
- Architecture Neutral
- Portable
- **Multithreaded**

Multi-threading is when a program can literally do two (or more) things at once by utilizing multiple processors and/or cores.

.NET Framework Version 1 included classes to support multiple threads of execution within a single process.

Release 5 introduced the async and await keywords to simplify multi-threading.

Release 5 also introduced the Parallel class, which lets us transform a large loop into multiple smaller loops which run in parallel on different threads.

# Integrated Development

It is possible to write a C# program using any text editor.  You would save your finished program to a file, then invoke the C# compilere (csc.exe) to perform the first phase compilation.

Much more often, programmers choose to use an Integrated Development Environment, and IDE.

IDEs combine the many tools on which developers depend into a single application.

Modern IDE’s include:

- Code editors with syntax-highlighting, intellisense and error-indicators
- Tools for importing libraries that bring advanced functionality to an application
- Editors for non-code files (text, SQL, images, XML …)
- Editors for creating a graphic user interface (GUI)
- Tools for Debugging a running application

# Visual Studio

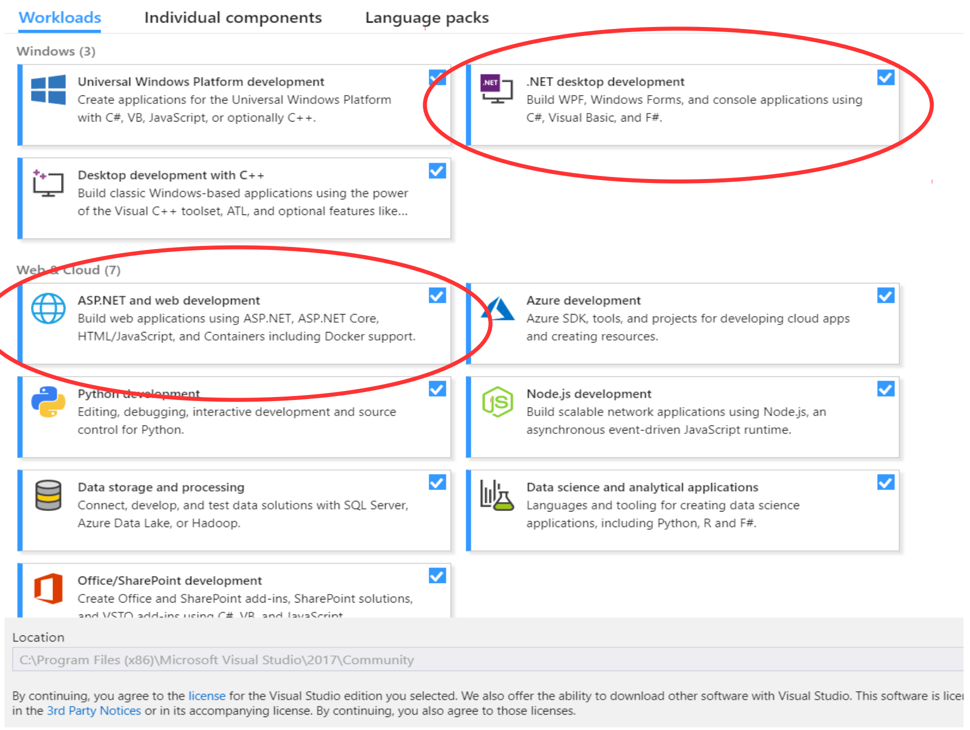
Visual Studio is without argument the best IDE for C# (and other .NET language) programs.

Visual Studio Community Edition is free, and runs on Windows and Mac computers.

Visual Studio Code offers fewer features, but runs on Windows, Mac, and numerous flavors of Linux.

# Installing Visual Studio

Visual Studio comes in three versions: Community, Professional, and Enterprise. The Community version is free and can be downloaded from VisualStudio.com.



Workloads Individual components Language packs

Windows (3)

**Universal Windows Platform development** ✓
Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

**.NET desktop development** ✓
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

**Desktop development with C++** ✓
Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like...

Web & Cloud (7)

**ASP.NET and web development** ✓
Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

**Azure development** ✓
Azure SDK, tools, and projects for developing cloud apps and creating resources.

**Python development** ✓
Editing, debugging, interactive development and source control for Python.

**Node.js development** ✓
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

**Data storage and processing** ✓
Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

**Data science and analytical applications** ✓
Languages and tooling for creating data science applications, including Python, R and F#.

**Office/SharePoint development** ✓
Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript

Location
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licer
in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Visual Studio facilitates development of many types of applications.

For this class, you will need capabilities for **.NET desktop development** and **ASP.NET and web development**.

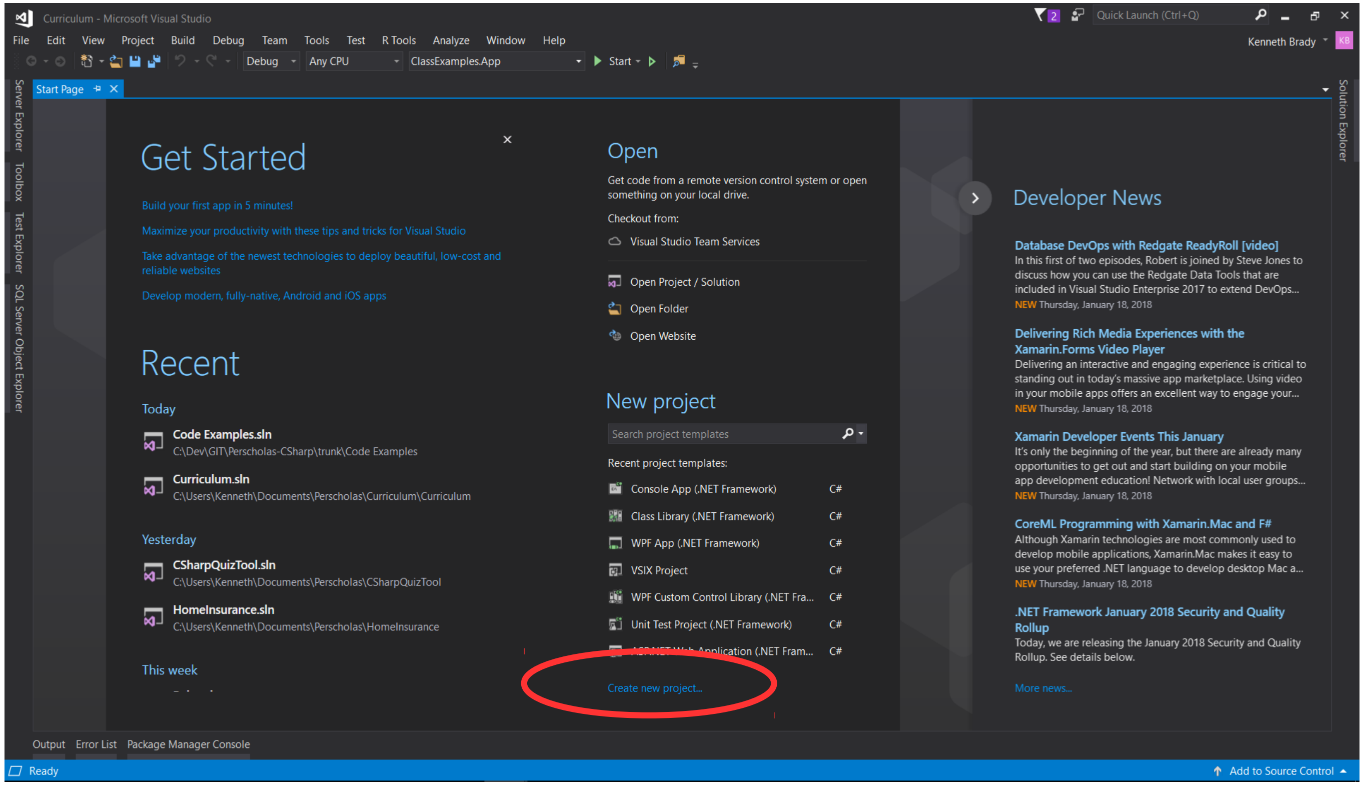
There is no harm in selecting other capabilities if you wish to explore!

# Running Visual Studio

When you first start Visual Studio, it will show you the Start Page.

The page includes links to educational resources. After you've been using it for a while, it will also provide links to recent projects as shown below.

The first thing we do with Visual Studio will be to create a new project, so click on the **create new project** link.



Get Started

Build your first app in 5 minutes!

Maximize your productivity with these tips and tricks for Visual Studio

Take advantage of the newest technologies to deploy beautiful, low-cost and reliable websites

Develop modern, fully-native, Android and iOS apps

Recent

Today

**Code Examples.sln**
C:\Dev\GIT\Perscholas-CSharp\trunk\Code Examples

**Curriculum.sln**
C:\Users\Kenneth\Documents\Perscholas\Curriculum\Curriculum

Yesterday

**CSharpQuizTool.sln**
C:\Users\Kenneth\Documents\Perscholas\CSharpQuizTool

**HomeInsurance.sln**
C:\Users\Kenneth\Documents\Perscholas\HomeInsurance

This week

Open

Get code from a remote version control system or open something on your local drive.

Checkout from:

Visual Studio Team Services

Open Project / Solution

Open Folder

Open Website

New project

Search project templates

Recent project templates:

| | |
|---|---|
| Console App (.NET Framework) | C# |
| Class Library (.NET Framework) | C# |
| WPF App (.NET Framework) | C# |
| VSIX Project | C# |
| WPF Custom Control Library (.NET Fra... | C# |
| Unit Test Project (.NET Framework) | C# |
| ASP.NET Web Application (.NET Fram... | C# |

Create new project...

Developer News

**Database DevOps with Redgate ReadyRoll [video]**
In this first of two episodes, Robert is joined by Steve Jones to discuss how you can use the Redgate Data Tools that are included in Visual Studio Enterprise 2017 to extend DevOps...
NEW Thursday, January 18, 2018

**Delivering Rich Media Experiences with the Xamarin.Forms Video Player**
Delivering an interactive and engaging experience is critical to standing out in today's massive app marketplace. Using video in your mobile apps offers an excellent way to engage your...
NEW Thursday, January 18, 2018

**Xamarin Developer Events This January**
It's only the beginning of the year, but there are already many opportunities to get out and start building on your mobile app development education! Network with local user groups...
NEW Thursday, January 18, 2018

**CoreML Programming with Xamarin.Mac and F#**
Although Xamarin technologies are most commonly used to develop mobile applications, Xamarin.Mac makes it easy to use your preferred .NET language to develop desktop Mac a...
NEW Thursday, January 18, 2018

**.NET Framework January 2018 Security and Quality Rollup**
Today, we are releasing the January 2018 Security and Quality Rollup. See details below.

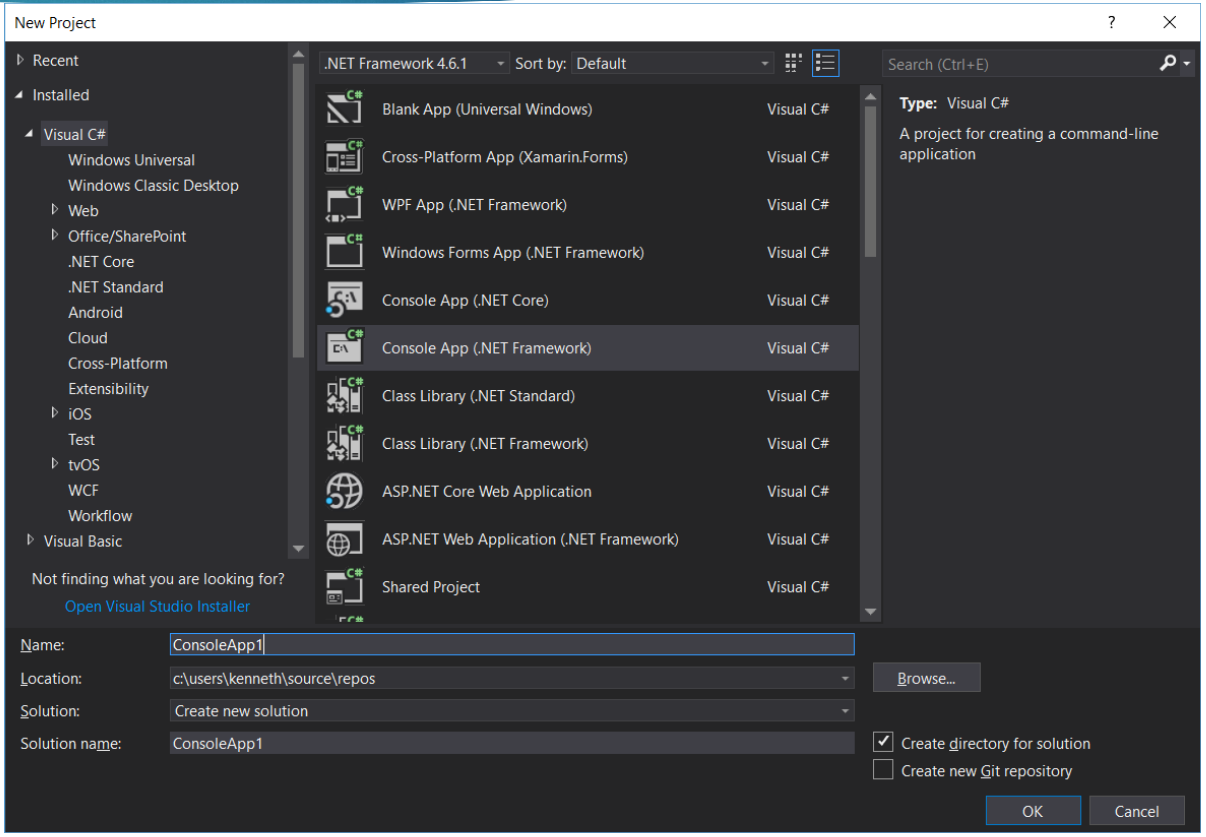
More news...

# New Project Dialog

The New Project dialog lists all the types of projects you can create.

You will see more or fewer options depending on which capabilities you selected when you installed Visual Studio.

Our first project will be a Console App utilizing the .NET Framework using the C# language.


New Project

- Recent
- Installed
  - Visual C#
    - Windows Universal
    - Windows Classic Desktop
    - Web
    - Office/SharePoint
    - .NET Core
    - .NET Standard
    - Android
    - Cloud
    - Cross-Platform
    - Extensibility
    - iOS
    - Test
    - tvOS
    - WCF
    - Workflow
  - Visual Basic

Not finding what you are looking for?
Open Visual Studio Installer

.NET Framework 4.6.1 | Sort by: Default | Search (Ctrl+E)

| | |
|---|---|
| Blank App (Universal Windows) | Visual C# |
| Cross-Platform App (Xamarin.Forms) | Visual C# |
| WPF App (.NET Framework) | Visual C# |
| Windows Forms App (.NET Framework) | Visual C# |
| Console App (.NET Core) | Visual C# |
| Console App (.NET Framework) | Visual C# |
| Class Library (.NET Standard) | Visual C# |
| Class Library (.NET Framework) | Visual C# |
| ASP.NET Core Web Application | Visual C# |
| ASP.NET Web Application (.NET Framework) | Visual C# |
| Shared Project | Visual C# |

**Type:** Visual C#
A project for creating a command-line application

| | |
|---|---|
| Name: | ConsoleApp1 |
| Location: | c:\users\kenneth\source\repos |
| Solution: | Create new solution |
| Solution name: | ConsoleApp1 |

Browse...
[✓] Create directory for solution
[ ] Create new Git repository
OK Cancel

platform
By Per Scholas

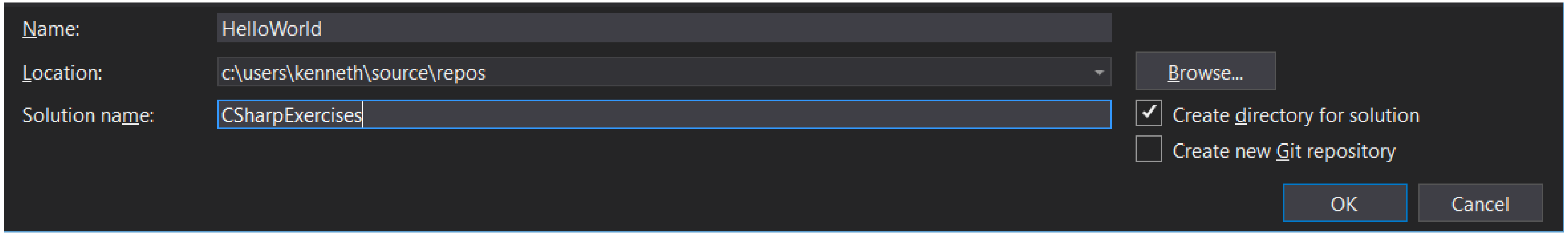
# Name the Project

Name the project **HelloWorld**.

Visual Studio will also create a Solution.  Developers often create multiple related projects that work together, and a Solution is a container for one or more projects.

Name the Solution **CSharpExercises**.  Make sure that <u>Create directory for solution</u> is checked.

Take note of the location of your solution.  This is where the solution and project files can be found.



| | |
|---|---|
| Name: | HelloWorld |
| Location: | c:\users\kenneth\source\repos |
| Solution name: | CSharpExercises |

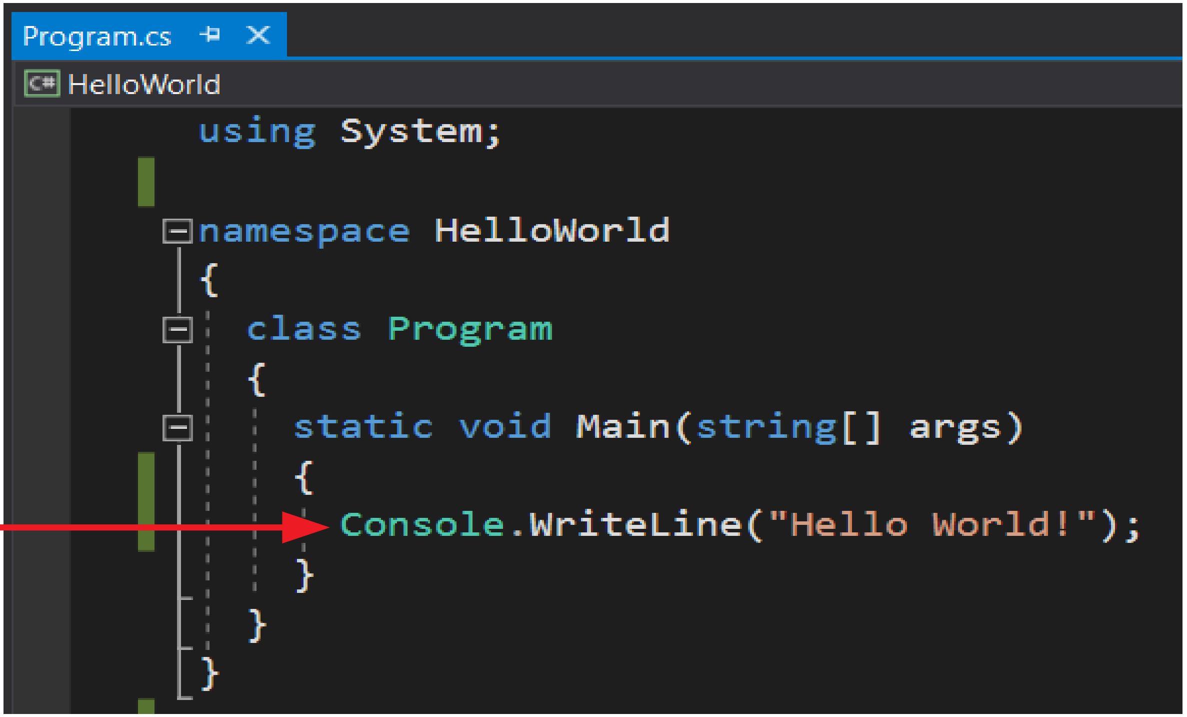
✓ Create directory for solution

☐ Create new Git repository

# A C# Console Application

Visual Studio creates an empty Console application based on the name you provided.

Type in the code shown here.



```
using System;

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

The next set of slides will walk through each part of this C# program.

platform
By Per Scholas

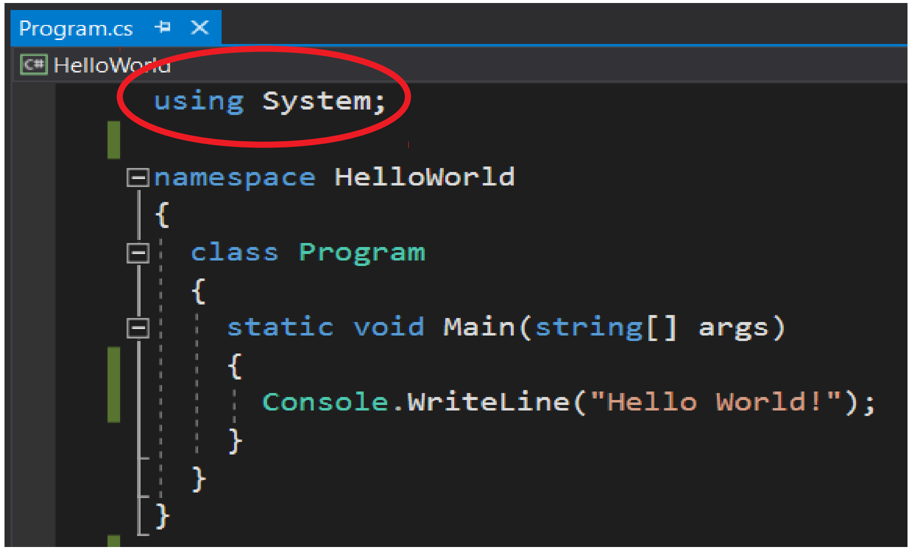
# using Directives

using directives allow us to use classes from other namespaces.

In this program, the Console class is defined in the System namespace.

If you remove (or comment out) the 'using System;' directive, the Console class is no longer recognized and the program will not compile.

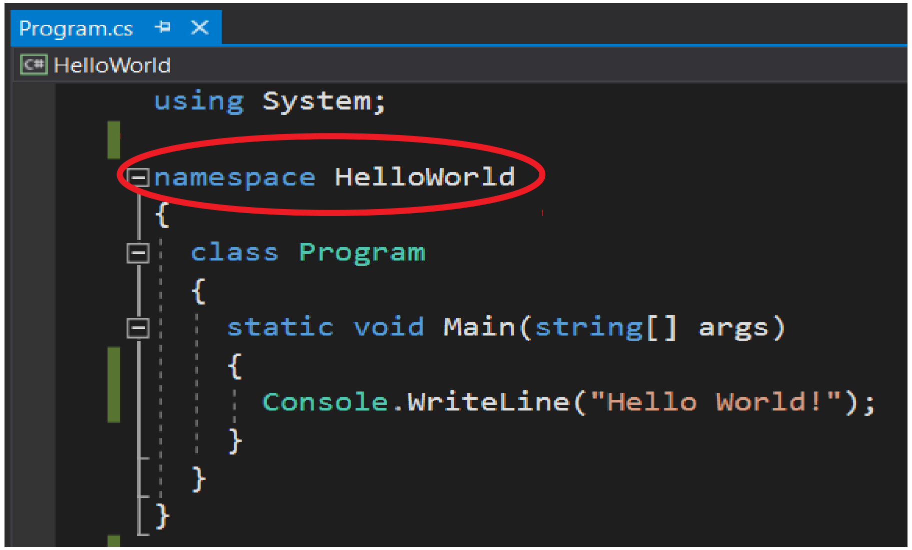
Notice that the directive is terminated with a semicolon.



```
using System;

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

# namespace



Program.cs

HelloWorld

```
using System;

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

The namespace keyword declares a scope or “space” in which related types can be defined.

Namespaces are used to organize code.

Namespace names should be chosen to give a hint about the types defined in the namespace.

By convention, namespace names use Pascal casing.

# class

The class keyword is used to define a class.

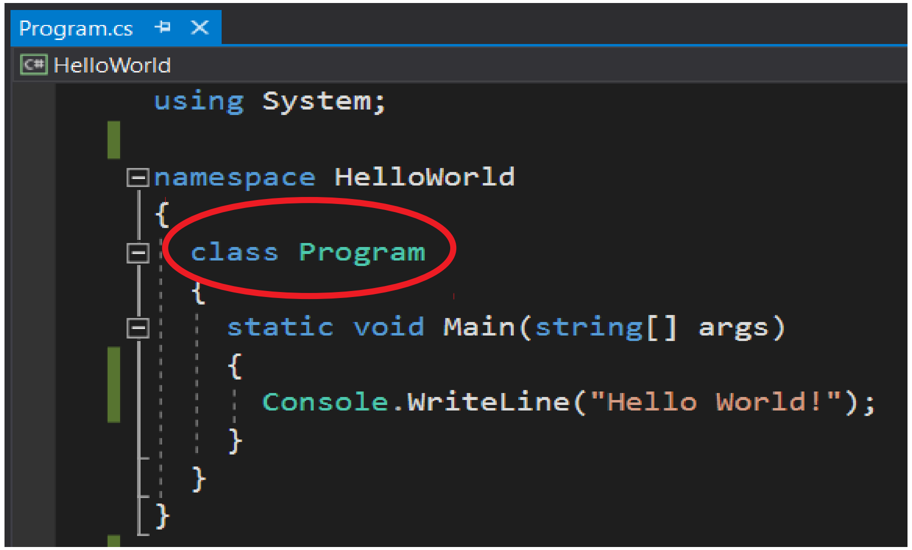
Classes are one of the five *types* that you can define using the C# language.

All C# executable code is defined within a class.

By convention, class names use Pascal casing.

Class names can use any of the alphanumeric characters, but they cannot begin with a digit.

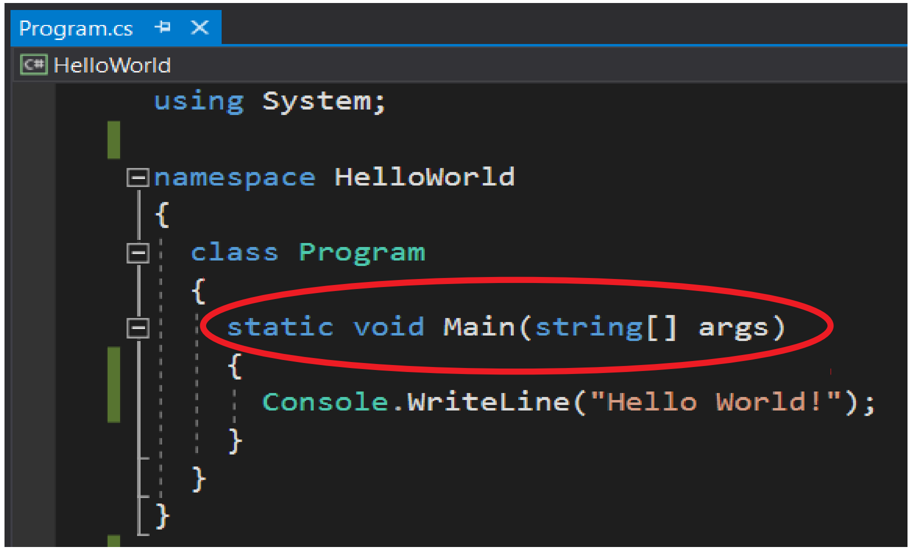
The only non-alphanumeric character allowed is the underscore: ‘_’.



```
using System;

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

# Methods



```
using System;

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

This class defines a method named **Main**.

Methods contain executable lines of code.

Methods can be defined to accept arguments. This method accepts an argument of type **string**[] - an array of strings.

Methods can optionally return a value. This does not return a value, so it has a *return type* of **void**.

# Executable Code

This program has a single line of executable code.

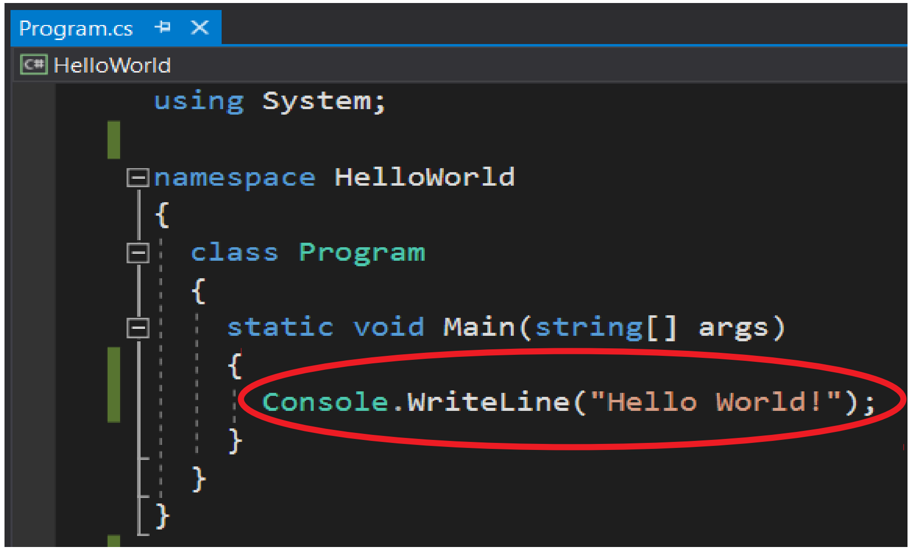
Console is a class defined in the System namespace.

WriteLine is a method defined by the Console class.

WriteLine takes a single argument, a string.

“Hello World!” is a literal string.

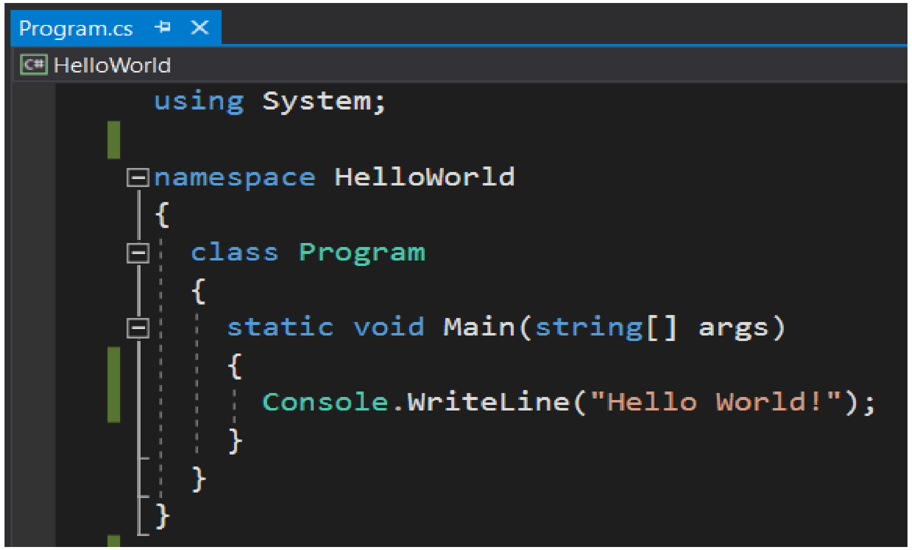
**All** executable statements end with a semicolon.



```
using System;

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

## Curly Braces: { ... }


Program.cs

HelloWorld

```
using System;

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

Like all C-family languages, C# uses pairs of curly braces to define blocks of code.

This code contains 3 pairs of curly braces.

The outermost pair is the namespace definition.

The first inner pair is the class definition.

The innermost pair is the method definition.

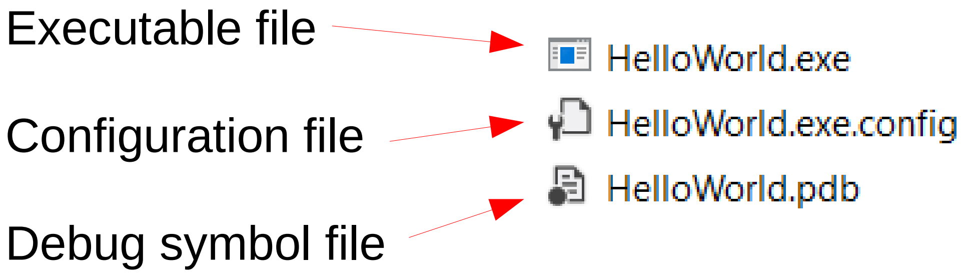
Visual Studio provides visual cues to show the pairings.

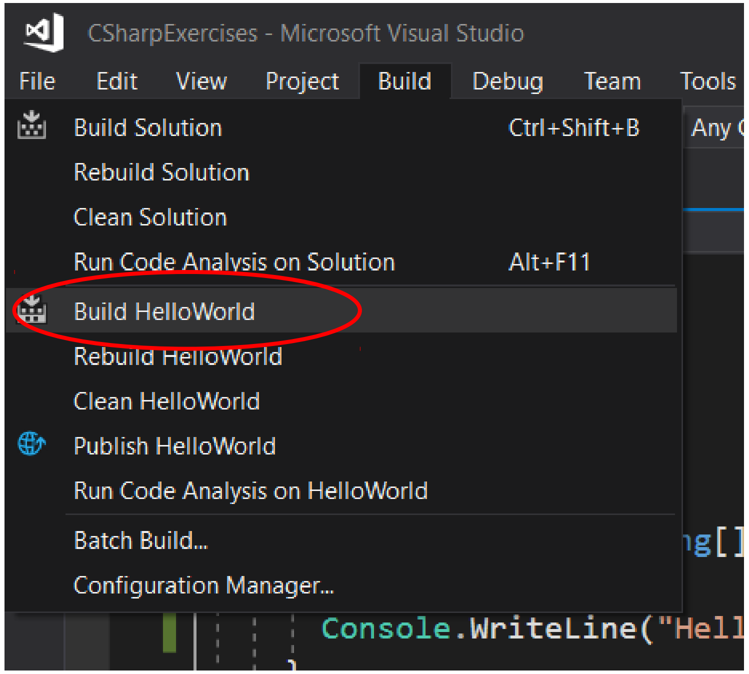
Curly braces must always be paired. Otherwise, you code will not compile!

# Compiling a C# Program

We can convert our C# source code to an executable using the menu: Build => Build HelloWorld. Alternatively you can right-click on the project in Solution Explorer and select Build.

Navigate to the project folder in Windows Explorer, and drill into the bin/Debug subfolder to view the executables:





# Intermediate Language

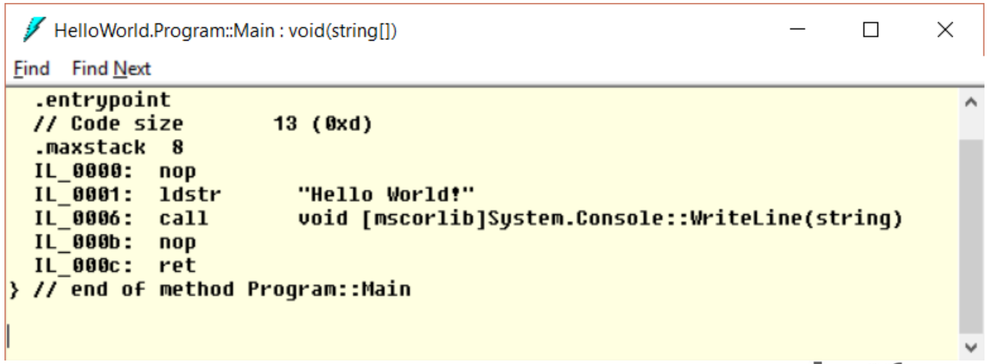
We've compiled 'HelloWorld' and have a .exe which, if you click on it, runs your program. But the actual compilation is only half done.

Visual Studio builds an application written in any of the .NET languages into Intermediate Language. We can inspect this intermediate code using the utility ILDASM. This is part of your .NET installation, and for Windows 10/.NET 4.6 is at the location:

C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6 Tools\x64

ILDASM gives us a hierarchical view of our application, and we can drill-down to the Main method to see the IL code.

It resembles Assembly code.



```
  .entrypoint
  // Code size       13 (0xd)
  .maxstack  8
  IL_0000:  nop
  IL_0001:  ldstr      "Hello World!"
  IL_0006:  call       void [mscorlib]System.Console::WriteLine(string)
  IL_000b:  nop
  IL_000c:  ret
} // end of method Program::Main
```

## JIT Compilation

The final compilation of a .NET application occurs when you actually run the program, when the Intermediate Language code is compiled “just in time” to machine code.

This is where platform-independence is achieved.  The Android JIT-compiler will compile to machine code appropriate to Android; the Linux JIT-compiler will compile to machine code appropriate to Linux; etc. etc.

This two-step process allows numerous optimizations.  An informative overview of JIT Compilation is available from Telerik.

## C# Keywords

This program uses 6 keywords: using, namespace, class, static, void and string.

Keywords are predefined, reserved identifiers that have special meanings to the compiler.

The C# language has approximately 78 keywords.

## Special Symbols

| Character | Name | Usage |
|---|---|---|
| { } | Curly Braces | Denotes a block of code |
| ( ) | Parentheses | Used to define methods |
| [ ] | Square Brackets | Used to define arrays |
| “ “ | Double Quotes | Used to declare literal strings |
| ; | Semicolon | Terminates a statement |
| // | Double Slashes | Begins a code comment |

# Indentation

Code should be indented one tab level for each pair of curly braces.

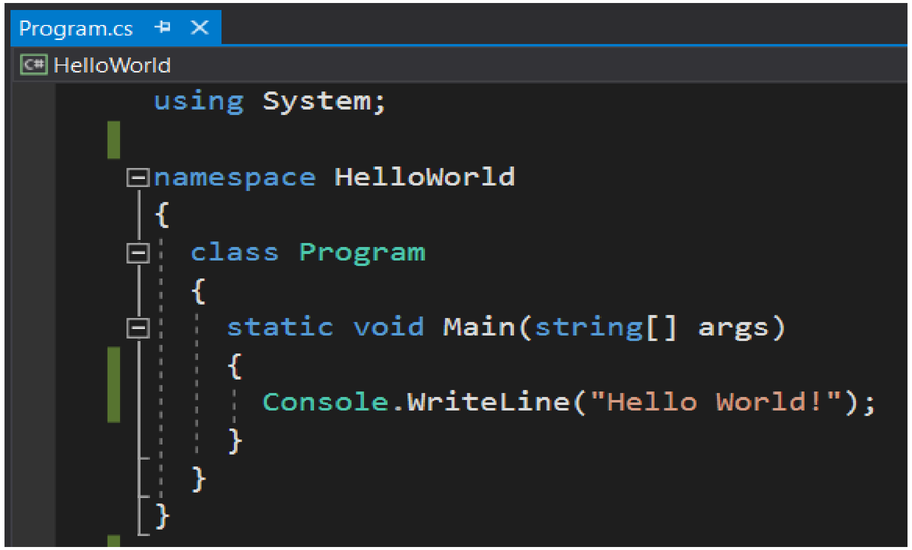
Indenting makes it easy to see the structure of your code.

Visual Studio can be configured to indent using either spaces or tab characters.

My preference is to use tabs, though some workplaces use spaces by convention.

Visual Studio can automatically indent a document using the edit menu:

Edit → Advanced → Format Document



```
using System;

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

# Programming Errors

As you begin writing programs, you will become familiar with 3 categories of programming errors:

**Compiler Errors**: these errors are detected by the compiler.  Visual Studio highlights them with red squiggles. They also show up in the error window. You cannot create an executable file (.exe) from code that contains compiler errors.

**Runtime Errors**: Runtime errors cause exceptions to be thrown while a program is running.  Unhandled exceptions cause a program to crash.  Examples of runtime errors are attempting to write past the end of an array, or to read from a file that does not exist.

**Logic Errors**: Logic errors (also called *bugs*) cause a program to produce incorrect results or behaviors.  These occur when a developer gives a program incorrect instructions.

# Topics Covered

- C# Features
- Integrated Development Environments
- Installing Visual Studio
- Creating a Console Application
- Parts of a C# Program
- Compiling a Program
- C# Keywords and Special Symbols
- Indenting Code
- Programming Errors