Top of Form



**Introduction to C# Programming: Lesson 2**

**Chapter 1**

**Introduction**

The previous lesson gave you some background about a personal computer and the history of programming languages. With all of this in mind, you will now look at the C# programming language to see what makes it unique. Specifically, this lesson will take you through the Microsoft .NET Framework and how it attempts to change the way we look at programming. Once this is discussed, I will show you where you can get a free copy of a C# compiler. I will also walk you through installation of this compiler and give you a sample program that you can create, compile, and run. After this lesson, you will be ready to start learning the fundamental concepts of the C# programming language, which means you will be ready to start writing fun and interesting programs!

Before we go much further, I want to provide a word of warning. This course will not work with an Integrated Development Environment (IDE). Although IDEs are very common, they tend to do a lot of extra things for you. These extras are helpful, but they can do things that are confusing. And because we're here to learn C# and not necessarily the IDE, we'll just work at the DOS prompt.   
  
  
  
**Chapter 2**

**The Microsoft .NET Framework**

The previous lesson probably started you thinking about the question of why there are so many different programming languages. You learned the majority of programmers today use high-level, object-oriented languages. However, C++, Java, and C# are all object-oriented languages. For that matter, Visual Basic, for the most part, is an object-oriented language as well. So which one is better? When I am asked that question, I usually respond, "I am more comfortable writing programs in one language, but I sometimes use another based on the program I am writing. This is because some tasks are very easy to program in one language but difficult to do in another."

Microsoft realized this and developed what they call the *.NET initiative*. One of the features this strategy has is that it allows different programming languages to work together while providing security and program portability. This means that programmers no longer have to write their programs entirely in one language. They can now develop one part of their program in C#, one in Visual Basic, and one in Java.

This ability to use different languages within one program accomplished using what is called the *Common Language Runtime (CLR)*. The CLR manages the execution of .NET code. This is done in a two-step process. The first step occurs when you compile your code. Recall, compiling is a process of taking high-level language code and converting it into instructions that the computer can understand. However, instead of doing this, .NET converts the code into an intermediate language called *Microsoft Intermediate Language (MSIL)*. When the program is run, the CLR then takes the MSIL code and translates it into machine language code that can be run on that specific computer. This two-step process is what allows a program to be written in more than one language, because each .NET language is converted into MSIL before the executable is created. While this might not be of great importance to you now (you haven't even learned C# yet), it is a concept that may come in very handy to you in the future.

The second part of the .NET Framework that is important is the *.NET class library*. Every program you write will have access to this library that contains many useful methods. These methods are sets of code that have been written for you. The library includes things such as the ability to display characters on the screen and getting input from the keyboard.  
  
  
  
  
**Chapter 3**

**Installing and Using the C# Compiler**

So far, I have spent a lot of time with history and background, but what about actually doing some programming? Believe me, it has been my experience that taking the time to learn all of the background will make you a better programmer. Now you are ready to start working with the compiler. Before you began this course, you were asked to install the C# compiler on your computer. Just in case you have not done so yet, this chapter will discuss how you can go about doing so.

By far, the easiest way to get the C# compiler onto your computer is to go out and buy a copy of Microsoft Visual C# Professional. You can purchase this online from one of the big retailers like Amazon.com. You should expect to pay somewhere around $250 for this product. Of course, there is a free version called the Express Edition. You can find a link to the download page in the Supplementary Materials section for this lesson. It will take some time to download and install, but hey it's free! Everything you do in this class can be done with either version.

This product will come with the .NET Framework, the compiler, and an *Integrated Development Environment (IDE)*. The IDE provides a nice interface for creating your applications. It has everything you are used to seeing in a Windows program. It includes things like menus and buttons that allow you to open and close source code files, compile and run your programs, and many more things. It will also make it easy for you to add common Windows controls to your programs by just dragging and dropping them. For those of you who are familiar with Visual Basic, it is very similar to the Visual Basic IDE.

While all of those features are very helpful and useful in the IDE, this course will work with the command prompt and Notepad. That way, if you do not want to spend the money on the IDE, you can still take part in this course. If you have already purchased or are planning to purchase the C# IDE, then you can skip the remainder of this chapter and go right to Chapter 4.

**Getting the Free Compiler**

You can get a free copy of the C# compiler through Microsoft. If you visit the Microsoft Developer's Network home page at http://www.msdn.com and search for "Microsoft .NET Framework Redistributable Package," you will find lots of information about this product. At this point, I have a warning for you about the .NET Framework SDK (Software Development Kit). Notice how the names of these two things are very close to one another. The SDK is another download, and this one is much bigger than the first (106 MB). This download will provide you with some useful tools about .NET. The tools consist mostly of different Help files. If you have time and space to download the SDK, by all means do so. However, for this course, you will only need to download the .NET Framework Redistributable Package.

With that being said, I will provide a direct link to the Redistributable Package in the Supplementary Material section for Lesson 2. When you get to this page, you should find a Download link in the upper right corner. Click this link and save the file to your computer. When the file is done downloading, just double-click on the file name to run the program. Follow the prompts. It should only take a few minutes to install. Then you are ready to go!

**Chapter 4**

**Writing Your First C# Program**

The first program you will write will be the classic "Hello World!" program. This program's goal will be to display a single line of text to the screen. While this may seem very elementary to some of you, believe me, it is a very important way to begin. Often when I was taking programming classes, I would write a similar version of this program before I started writing the program I was asked to do for class. The reason I did this was because I figured if I couldn't get my program to compile and print some text to the screen, then there was no way I could ever get it to do something more complex.

**Typing the Program**

Let's get started. You will need to open up a text editor. I usually recommend people use Notepad. This is a simple text editor that comes with most computers. It should be located in the **Accessories** submenu.

Just a quick note about formatting: To distinguish my text from C# code, I will format the text using a fixed font. For example, if you see System.Console.Out.WriteLine(); it is C# code.

**Reminder:** Whenever you see a ¬ character at the end of a line of code, this denotes a break in that code-line due to our HTML page size limitations. The remaining section of that code-line can be found on the next line below. After pasting your code into your text-editing program, simply select and delete that character, plus the hidden-break following it, to realign the code-line properly.

Now you are ready to type the following program in (I will discuss what each line means in a moment):

// My very first C# program

public class Hello

{

// The program begins inside Main

public static void Main( )

{

System.Console.Out.WriteLine ("Hello, world!");

} // end of Main

} // end of Hello

I want to offer a couple words of warning about the code used in this course. First, if you attempt to use Copy and Paste to try out the code used, you may find that there are issues in getting the code to compile. This will usually come from the fact that the quotation marks (" and ') are different on the web. When you attempt to use the quotes from the web, the compiler will not recognize them and will not compile. This can usually be fixed by deleting the quotes in Notepad and re-typing them. However, I think you would be better off retyping the entire program on your own. This will help eliminate this problem and will also help you to better learn C# because it will provide reinforcement as you type.

A second problem you will encounter later in this course is due to the size limitations of the web. Since this course is presented on a web page, there is a limitation to the number of characters that can be printed on one line. I have attempted to get all of the text on the same line; however, this is not always possible. In that case, the remainder of the line of code is shown on the following line, which is indented. Again, if you Copy and Paste the text from the web, you will end up with compiler errors. To solve this problem, it is best to retype the entire program. Remember when you are typing that every line of code, which a couple exceptions that you will learn about, end with a semicolon (;). If you are typing a line of code and it does not end with a semicolon and it is not one of the named exceptions, look at the next line of text and see if it is indented. If so, you should put all of that code on the same line.

Now, I will explain each line of code. The first line of code, // My very first C# program, is called a *comment line*. This is text that is meant to be read by a human being, *not* the computer. It is very important to place comment lines in your program. They will make it easier for you and anyone reading your program to identify what the computer will be doing with this code. Comment lines start with two forward slashes (//). This basically tells the compiler, "Don't translate this into machine language; it is meant to be read by a human, not the computer." This is true for all of the text that you type after the slashes until you press ENTER.

The next line of code, public class Hello, is known as the *class header*. You can think of this line as a direction to the computer to let it know that you are about to define the class. Every program will have a class header that will look exactly like the one above *except* that the name of the class will be different. In our example here, we are about to define class Hello. We must let the compiler know where the class begins and where it ends. We do this by enclosing the class body, or the statements inside of the class, with curly braces ({ }).

Note the next line of code, {, is a single open curly brace. This tells the compiler that we are about to begin the definition of our class.

The next line of code is a second comment. It is to remind the programmer that the program will begin with the first line inside the Main method.

Our next line of code, public static void Main( ), is the method header for the Main method. You will notice that there are a bunch of words that you probably are not familiar with. Do not worry about that right now. By the end of the course, you *will* know what they mean. For now, just type them in exactly as they look above. Every program you write in the course will contain a Main method, and the method header will *always* look like the one above.

Again, we need to let the computer know where the method begins and ends. Therefore, the next line of code is a second opening curly brace, {.

The next line of code is actually the only line of code that will be executed and run when you run your program. This line, System.Console.Out.WriteLine("Hello, world!");, tells the computer to display the text "Hello, world!" on the output screen. Any time you want to display anything on the screen, you will need to use a statement similar to this one. That is, you will type System.Console.Out.WriteLine(" ") and put the text you want to display inside the quotes.

In the next line, } // end Main, the closing curly brace closes out the Main method, and the // end Main is a comment to remind you that this is the end of the Main method.

The final line, } // end of Hello, has another closing curly brace, which closes the class, and the // end of program is a comment to remind you that this is the end of the class.

This brings up a few important things to remember about our programs. First, organization. Every statement in C# must be contained inside a method, and every method must be contained inside a class. For now, your programs will consist of only one class with only the Main method. As you work through this course, you will learn why other methods are important, and you will even define your own methods!

A second important aspect of the above program is the syntax. C# is a case-sensitive programming language, meaning that the letter *a* is different than *A*. So, if you accidentally typed the second line as:

Public Class Hello

it would have a totally different meaning. Actually, your program would not compile, because the compiler would have no idea that you were trying to define a class. Also, every statement in C# ends with a semicolon (;). This may seem foreign to you right now, but you will get used to it. This is similar to writing a sentence in English, where you end every sentence with a period (.). I had a student once who came to class one day shaking his head. "I know I have been writing too many programs," he said. When I asked why, he said, "I just got back an English paper, and I had ended a few of my sentences with semicolons!" I agreed that this student had been spending too much time programming.

You may now be thinking, "Wait a minute, there was only *one* line of code that ended with a semicolon, but you are saying that *every* line must end with one?" Well, that is exactly correct, but let's look at all of those lines that do not end with semicolons. First, look at the comment line. It does not end with a semicolon, but why should it? After all, didn't we say that the two slashes told the compiler to ignore the rest of the line? Then look at the class header and the method header, they do not end in semicolons. But again, they really shouldn't because they are just the beginning of the class or method definition, respectively. The body of each of these things is enclosed in curly braces. They themselves show the beginning and ending of the body, so they do not need semicolons either.

Now that you have typed in your program, you will need to save your work. You will need to give your file name the same name as the name of your class, with a .cs extension. Click **File** and **Save As**. In the File name box, type *Hello.cs*, and in the **Save as type** box, be sure to choose **All Files**. The second part of this step is very important, because if you forget to do this step, your file name will be *Hello.cs.txt*. The compiler will only compile files that have a .cs extension, so it will ignore Hello.cs.txt. Also, be sure you know the folder where you are saving your file. I recommend that you create a directory named *csharp* on your C:/ drive where you can save your work. This will make your work easier to find.

**Compiling the Program**

1. You will need to know where your compiler is located. If you do not know this, you will need to search the files on your computer for *csc.exe*. It should be something like *C:\Windows\Microsoft.NET\Framework\v3.5*. One way to search for the file is to:
   * Click the **Start** button in the lower left corner of the screen.
   * Windows Vista users should choose **Computer**. Windows XP/2000 users should choose **My Computer**.
   * Double-click **Local Disk (C:)** to list the folders on the C: drive.
   * Double click **Windows** to list the items inside this folder.
   * Scroll down until you see **Microsoft.NET**. Double-click this folder to open it.
   * Double-click the **Framework** folder.

Most likely you'll now see a number of folders that start with the letter 'v'. These are the different versions of the .NET Framework installed on your computer. Possible folder names are:  
v1.0.3705  
v1.1.4322  
v2.0.50727  
v3.0  
v3.5  
Though there may be others as well.

* + Double-click to open the folder with the highest version number (in my list above v3.5). Scroll through the list to confirm that you have 'csc.exe'. Note that your computer may be set up to hide the file extensions, in which case, you'll only see "csc" listed. That's okay.
  + Right-click the **csc.exe** file and choose **Properties**.
  + Near the top of the resulting dialog box will be an item named *Location*, which shows the full path to this file. Using your mouse, left drag over the path to highlight it.
  + Once the path is highlighted, right-click and choose **Copy**. This is the path that you'll use in step 3 below. Note that you may want to either write down this full path on a sheet of paper or open a text editor to paste this path for use later. You will need it every time you start up your command prompt.

1. Now it's time to open the command prompt. Click **Start** and select **All Programs > Accessories > Command Prompt**.
2. Next, tell the computer where the C# compiler is located by setting the path. This is done by typing the following:

path= C:\Windows\Microsoft.NET\Framework\v3.5

**Note:** You will need to type in the path to the compiler on *your* machine, which you found in step 1.

1. Using the command prompt, navigate to the folder where your source code is. Here are a couple of DOS commands to help you navigate to your folder:
   * To get to a folder named *csharp* on your c:\ drive, just type: *cd c:\csharp*
   * To change drives, just type the drive letter followed by a colon (:). For example, if your file is on your floppy disk, you can type *A:* to switch to the A:\ drive.
   * To navigate down a folder—that is, to a folder inside the folder you are currently in—type *cd* and the name of the folder you want to go to. For example, if you want to navigate to a folder named *first* that is inside the folder you are currently in, you can type *cd first*.
   * To move one folder up—that is, go to the folder outside the folder you are currently in—type *cd* (the letters *cd* and two periods).
   * To move directly to the root directory, type *cd\*.
   * To see the contents of the folder you are currently in, type *dir*.

If you took my recommendation, you can just type *cd c:\csharp* to move to the directory where your file is stored. Confirm that Hello.cs is located in that folder using the *dir* command.

1. Compile your program. Type *csc Hello.cs*. This will have the compiler attempt to translate your code into machine language. This will have one of the following results:
   * You will receive an error from the operating system that states something like "Bad command or file name" or "csc is not recognized as an internal or external command." This means that you did not correctly set the path to the compiler. Double-check the location of csc.exe on your computer, and be sure that you set the path at the command prompt correctly using the path= statement.
   * You will receive one or more errors in the program. The compiler will tell you the name of the file and the line number that it thinks the error occurred on. For example, I intentionally left the semicolon off of the statement that writes the text to the screen, and I got the following error message:

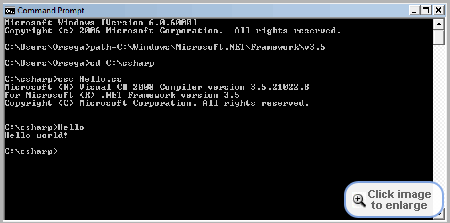
Hello.cs(5, 47): error CS1002: ; expected

This tells me that the compiler thinks that I need a semicolon on line 5. To resolve this problem, I would need to open my source code file in Notepad, make any necessary changes, save the file, and attempt to compile the program again.

* + You will receive no error messages (only a copyright statement from the compiler). While this may seem like a bad thing, it is the opposite. Just remember, no news is good news. This means that your program compiled just fine and you can now run the program.

1. To run your new program, just type the name of the class in at the command prompt. For example, since the class name for this file was Hello, just type *Hello*.

That's it! Right now, it may seem as if there are a lot of steps to go through in order to get your program compiled and running, but I want to point a couple of things out to you. First, you will only need to install the compiler once. Second, once you find the folder where the compiler is located, you will never have to worry about where it is again. Just be sure to write this location down on a sheet of paper somewhere. Third, you will only need to set the path to the compiler if you have started a new command prompt window.

[](https://api.ed2go.com/CourseBuilder/2.0/images/resources/prod/cpb-0/L02-01S.gif)  
Command prompt commands and corresponding output from  
working through the five steps of compiling and running a program

Another, easier way to set the path to the compiler is to change the settings in the Operating System to always include this path. After doing this, you will not need to set the path on your computer again because the Operating System will take care of it for you. To do this, follow the steps listed according to the Operating System you are using.

**For Windows Vista / Windows 7**

1. Click the **Start** button in the lower left corner of the screen.
2. Right-click **Computer** in the list.
3. Choose **Properties**.
4. Click **Advanced system settings** in the upper right corner of the dialog box. (NOTE: Windows 7 displays link in the upper *left* corner of the dialog)
5. Click the **Advanced** tab and then the **Environment Variables** button.
6. In the System Variables section of the Environment Variables dialog, scroll down until you find the **Path** variable.
7. Select **Path** and then click the **Edit** button.

At the end of the string in the value field, add:

*;C:\Windows\Microsoft.NET\Framework\v3.5*

(**Note:** The semicolon (;) is crucial as it separates the directories. Also, realize that you'll need to add the path that you found when you searched for the csc.exe file from before.)

1. Now click **OK** until you close all of the system dialogs and then close the Control Panel. (There's no need to reboot your system.)

**For Windows 2000/XP**

1. Right-click on the **My Computer** icon.
2. Choose **Properties**.
3. Click the **Advanced** tab and then the **Environment Variables** button.
4. In the System Variables section of the Environment Variables dialog, scroll down until you find the **Path** variable.
5. Select it and then click the **Edit** button.

At the end of the string in the value field, add:

*;C:\Windows\Microsoft.NET\Framework\v3.5*

(**Note:** The semicolon (;) is crucial as it separates the directories. Also, realize that you will need to add the path that you found when you searched for the csc.exe file from before.)

1. Now click **OK** until you close all of the system dialogs and then close the Control Panel. (There is no need to reboot your system.)

**For Windows 95/98/ME**

1. Open the autoexec.bat file with Notepad or your favorite text editor. The autoexec.bat file can be found in the root directory of your C: drive. Be sure to select **All Files** from the **Files of Type** drop down list in the **Open** dialog box.
2. Scroll down and look for the line beginning with the word *path*.
3. At the end of the line, add:  
   *;C:\Windows\Microsoft.NET\Framework\v3.5*

(**Note:** The semicolon (;) is crucial as it separates the directories. Also, realize that you will need to add the path that you found when you searched for the csc.exe file from before.)

1. Save the file and close Notepad.
2. Reboot your system for your new entries to take effect.

When I am writing programs at the command prompt, I will usually type my program in Notepad, save it somewhere that I can find it easily, and leave Notepad open. Then I will open the command prompt, set the path, and navigate to the folder where my source code was saved. Finally, I will compile and run the program. If there are any changes that I need to make, I will switch over to the Notepad window (leaving the command prompt window running), make my changes in Notepad, and save my changes. I will then go back to the command prompt and recompile my program. This switching back and forth between windows may be a little confusing at first, but you will get more comfortable as you do it more and more.

**Chapter 5**

**Summary**

This lesson began with a short description of the Microsoft .NET Framework and how it is important to you as a C# programmer. Next, you installed the C# compiler onto your computer. Finally, you spent some time learning about and typing in a C# program. You may have thought that this program was very simple, and you were correct. However, what you may not have realized is that this program demonstrates the basic shell of a C# program. This shell can be stated as:

public class YourClassName  
{  
public static void Main( )  
{  
// Place statements to be executed here  
}  
}

Every program you write in this course will use this basic shell of a program. The only difference in the programs will be the name of your class (in place of YourClassName) and the statements that you will place in Main.

In this lesson, you also learned how to type your program into a text editor like Notepad, and how to save, compile, and run your program. In the process, you learned a little bit about navigating through different folders using DOS commands, and you learned a little bit about how the compiler communicates its results to you.

With all of this knowledge about how a program is developed, compiled, and run, you are now ready to start making your programs more useful. The program you wrote in this lesson just printed some text onto the screen. In the next lesson, you will learn how to create and use variables to store data inside your program.  
  
  
  
**Supplementary Material**

|  |
| --- |
| [Microsoft .NET Framework 3.5 Redistributable Package](http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en)  http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en |
| This "Redistributable Package" will contain all of the files necessary to compile the C# programs for this course.  **Note**: Please be sure that you run Windows Update (**Start** > **All Programs** > **Windows Update**) on your machine before installing the Redistributable Package. |
| [Visual C# Express Edition Download](http://www.microsoft.com/express/vcsharp/)  http://www.microsoft.com/express/vcsharp/ |
| This is a link to the Visual C# Express Edition download. Realize that downloading this will give you the compiler (that is, you don't need to do this AND the redistributable package). The Express Edition contains the compiler and an Integrated Development Environment (IDE).  Please realize that this course does NOT use the IDE at all. In fact, if you do choose to download this, I strongly urge you not to use it for this course because it will create extra files and extra work for you. This will take away from your learning about the basics of programming with C#, which we do cover in this course.  If you have any further questions about this, or anything else, please post a message in the Discussion area. |

**FAQs**   
  
**Q:** When I try to compile my program, I keep getting an error message that says that csc is not an internal or external command. What does that mean?  
  
**A:** That means that you have not correctly set the path to the compiler. You should search your computer for the csc.exe file and write down its location on a sheet of paper. Then at the command prompt type:  
  
path =  
  
and the path that you wrote down on the piece of paper.  
  
  
**Q:** I have the path to the compiler set correctly, but when I try to compile the program, I get a long list of errors. Where should I start?  
  
**A:** In my experience, it is always best to start with the first error that the compiler shows you. Sometimes if you open a class body and forget to close it, it will generate more than one error. If you fix the first, often times this will fix some subsequent errors.

**Assignment**   
  
  
Write a class named *Favorites*.

This class should contain a Main method that has statements in it to display your name on the screen. The program should also display your favorite movie on a second line and your favorite color on a third line.

When you are finished typing the file, save it, compile it and run it.

|  |
| --- |
| https://api.ed2go.com/CourseBuilder/2.0/images/resources/prod/global/images/info.png**Unzipping a File**  In the remaining lessons of the course you'll be able to download the Solution for each Assignment. These Solutions for this course are *zipped*, meaning it's compressed to make easier to download.  To download the file, **right-click** the link and choose **Save Target As** from the menu. Save the zip file to a location on your computer that you will remember.  After downloading the file, you'll have to unzip it to get out the document. **Here's how using Windows XP**:   1. After the file has finished downloading, locate the zip file. 2. Right-click the **Favorites.zip** icon, and from the menu that pops up, click **Open With**. 3. You'll see another menu. Click **Compressed (zipped) Folders**.   how to unzip   1. This will open a new window showing the file that is in the Zip folder. In the left hand pane of this window, click **Extract all files**.   how to unzip- part 2   1. This will extract the file from the ZIP file, and you'll see the **Favorites.cs** document in your folder.   **Here's how using Windows Vista or Windows 7**:   1. After the file has finished downloading, locate the Favorites.zip file. 2. Right-click the Favorites.zip icon, and from the menu that pops up, click Extract All.   Extract All   1. A dialog box appears, letting you choose the folder where you want to extract the files. Click on a folder and then click the Extract button.   how to unzip in Windows Vista   1. This will extract the .CS file from the ZIP file, and you'll see two new document in your folder.   Unzipping files is pretty simple! If you have any trouble with it, let me know in the Discussion Area. Please be sure to save and unzip the files in each lesson, as you'll want to see the solutions to the assignments. |

[Click here to download the solution: **Favorites.zip**](https://api.ed2go.com/CourseBuilder/2.0/images/resources/prod/cpb-0/Favorites.zip)

|  |
| --- |
| Copyright © 1997 - 2012 Education To Go. All rights reserved. The material on this site cannot be reproduced or redistributed unless you have obtained prior written permission from Education To Go. Education To Go and ed2go are registered trademarks of Education To Go, a part of Cengage Learning. |

Bottom of Form