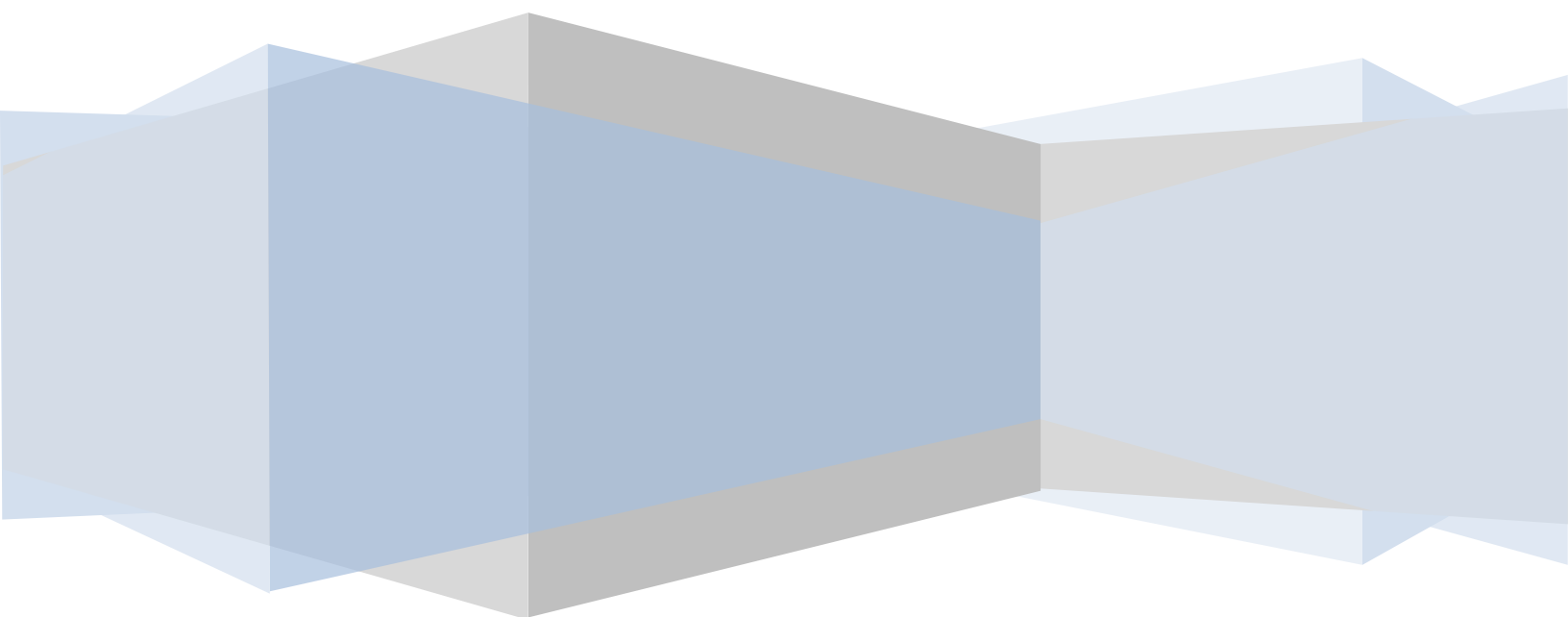


C# Essentials



C# Essentials – First Edition

© 2009 Techotopia.com. This eBook is provided for personal use only. Unauthorized use, reproduction and/or distribution strictly prohibited. All rights reserved.

Table of Contents

Chapter 1.	About C# Essentials.....	11
Chapter 2.	The C# Language and Environment	12
2.1	A Brief History of Computer Programming Languages	12
2.2	What exactly is C#?	13
2.3	The Common Language Infrastructure (CLI)	13
2.4	Common Intermediate Language (CIL)	14
2.5	Virtual Execution System (VES)	14
2.6	Common Type System (CTS) & Common Language Specification (CLS)	15
2.7	The Framework (Base Class and Framework Class Libraries)	15
2.8	Non Microsoft Implementations of the CLI	15
Chapter 3.	A Simple C# Console Application	16
3.1	Options for Installing a C# Environment	16
3.2	C# Source File Naming Conventions	16
3.3	Creating a Sample C# Program.....	17
3.4	Compiling C# Code from the Command Line.....	17
3.5	Executing a Compiled C# Program	18
Chapter 4.	Creating a Simple C# GUI Application with Visual Studio.....	19
4.1	Installing Visual Studio with C# Support	19
4.2	Creating a new Visual Studio C# Project	19
4.3	Adding Components to the Windows Form.....	19
4.4	Changing Component Names.....	20
4.5	Changing Component Properties.....	21
4.6	Adding Behavior to a Visual Studio C# Application.....	21
Chapter 5.	C# Variables and Constants	24
5.1	What is a C# Variable?.....	24
5.2	What is a C# Constant?	25
5.3	C# Integer Variable Types	25

5.4	C# Floating Point Variables	26
5.5	The C# Decimal Variable Type.....	26
5.6	C# Boolean Variable Type	27
5.7	C# Character Variable Type.....	27
5.8	C# String Variables	28
5.9	Casting Variable Types in C#	28
Chapter 6. C# Operators and Expressions		30
6.1	What is an Expression?	30
6.2	The Basic Assignment Operator	30
6.3	C# Arithmetic Operators	31
6.4	C# Operator Precedence	32
6.5	Compound Assignment Operators.....	33
6.6	Increment and Decrement Operators.....	34
6.7	Comparison Operators	35
6.8	Boolean Logical Operators	36
6.9	The Ternary Operator.....	37
Chapter 7. C# Flow Control with <i>if</i> and <i>else</i>		38
7.1	Using the if Statement	38
7.2	Using <i>if ... else ..</i> Statements	39
7.3	Using <i>if ... else if ..</i> Statements	39
7.4	Summary	40
Chapter 8. The C# switch Statement		41
8.1	Why Use a switch Statement?	41
8.2	Using the switch Statement Syntax	42
8.3	A switch Statement Example	43
8.4	Explaining the Example	44
8.5	Using goto in a C# switch Statement	44
8.6	Using continue in a C# switch Statement	46

Chapter 9. C# Looping - The <i>for</i> Statement	47
9.1 Why Use Loops?	47
9.2 C# Loop Variable Scope	49
9.3 Creating an Infinite for Loop	49
9.4 Breaking Out of a for Loop	49
9.5 Nested for Loops	50
9.6 Breaking From Nested Loops	50
9.7 Continuing for Loops	51
Chapter 10. C# Looping with <i>do</i> and <i>while</i> Statements	52
10.1 The C# while Loop	52
10.2 C# <i>do ... while</i> loops	53
10.3 Breaking from Loops	53
10.4 The continue Statement.....	54
Chapter 11. C# Object Oriented Programming	56
11.1 What is an Object?	56
11.2 What is a Class?.....	56
11.3 Declaring a C# Class.....	56
11.4 Creating C# Class Members.....	57
11.5 Static, Read-only and Const Data Members	57
11.6 Instantiating an Object from a C# Class	58
11.7 Accessing C# Object Members.....	59
11.8 Adding Methods to a C# Class.....	60
11.9 C# Constructors and Finalizers.....	61
Chapter 12. C# Inheritance	64
12.1 What is Inheritance?	64
12.2 An Example of Inheritance	64
12.3 Creating a Subclass in C#.....	65
12.4 Passing Arguments to the Base Class Constructor.....	67

Chapter 13. Understanding C# Abstract Classes	68
13.1 What is a C# Abstract Class?	68
13.2 Abstract Members.....	68
13.3 Declaring a C# Abstract Class	68
13.4 Deriving from an Abstract Class	69
13.5 The Difference Between abstract and virtual Members	70
Chapter 14. Introducing C# Arrays	73
14.1 Creating Arrays in C#.....	73
14.2 Declaring a Multidimensional Array.....	74
14.3 Accessing Array Values.....	74
14.4 Getting the Length of an Array.....	75
14.5 Sorting and Manipulating C# Arrays	76
Chapter 15. C# List and ArrayList Collections	78
15.1 What are C# Collection Classes	78
15.2 Creating C# List Collections - List<T> and ArrayList	78
15.3 Adding Items to Lists	79
15.4 Accessing List Items.....	79
15.5 Removing Items From Lists	80
15.6 Inserting Items into a C# List.....	80
15.7 Sorting Lists in C#	80
15.8 Finding Items in a C# List or ArrayList	80
15.9 Obtaining Information About a List	81
15.10 Clearing and Trimming C# Lists	81
Chapter 16. Working with Strings in C#.....	83
16.1 Creating Strings in C#	83
16.2 Obtaining the Length of a C# String.....	84
16.3 Treating Strings as Arrays.....	84
16.4 Concatenating Strings in C#	85

16.5	Comparing Strings in C#	85
16.6	Changing String Case	87
16.7	Splitting a C# String into Multiple Parts	87
16.8	Trimming and Padding C# Strings	88
16.9	C# String Replacement	89
16.10	Summary.....	89
Chapter 17.	Formatting Strings in C#.....	90
17.1	The Syntax of the String.Format() Method	90
17.2	A Simple C# String Format Example	90
17.3	Using Format Controls.....	91
17.4	As Simple Format Control Example.....	91
17.5	The C# String.Format() Format Controls	91
Chapter 18.	Working with Dates and Times in C#	93
18.1	Creating a C# Date Time Object	93
18.2	Getting the Current System Time and Date	94
18.3	Adding or Subtracting from Dates and Times	94
18.4	Retrieving Parts of a Date and Time.....	96
18.5	Formatting Dates and Times in C#	97
Chapter 19.	C# and Windows Form	100
19.1	Creating a New Form.....	100
19.2	Changing the Form Name	100
19.3	Changing the Form Title	100
19.4	Changing the Form Background Color	101
19.5	Changing The Form Background Image	102
19.6	Configuring the Minimize, Maximize and Close Buttons	103
19.7	Setting Minimum and Maximum Form Sizes	104
19.8	Specifying the Position of a Form on the Display	104
19.9	Changing the Form Border	104

19.10	Stopping a Form from Appearing the Windows Taskbar	104
19.11	Creating a Transparent Form.....	105
Chapter 20.	Designing Forms in C# and Visual Studio.....	106
20.1	Visual Basic Forms and Controls	106
20.2	Double Clicking the Control in the Toolbox	107
20.3	Dragging a Dropping Controls onto the Form.....	108
20.4	Drawing a Control on the Form.....	108
20.5	Positioning and Sizing Controls Using the Grid	108
20.6	Positioning Controls Using Snap Lines	110
20.7	Selecting Multiple Controls	110
20.8	Aligning and Sizing Groups of Controls	111
20.9	Setting Properties on a Group of Controls.....	111
20.10	Anchoring and Autosizing Form Controls.....	112
20.11	Setting Tab Order in a Form	113
Chapter 21.	Understanding C# GUI Events.....	114
21.1	Event Triggers.....	114
21.2	Wiring Up Events in Visual Studio	115
21.3	Compile and Running the Application	117
21.4	Setting Up a C# Timer Event.....	117
Chapter 22.	C# Events and Event Parameters	120
22.1	The Anatomy of an Event Handler	120
22.2	A C# EventArgs Example.....	120
22.3	C# EventArgs Object Properties.....	122
22.4	Identifying which Mouse Button was Clicked	123
Chapter 23.	Hiding and Showing Forms in C#	125
23.1	Creating a C# Application Containing Multiple Forms.....	125
23.2	Understanding Modal and Non-modal Forms	126
23.3	Writing C# Code to Display a Non-Modal Form.....	127

23.4	Writing C# Code to Display a Modal Form	128
23.5	Hiding Forms in C#	128
23.6	Closing Forms in C#	129
Chapter 24.	Creating Top-Level Menus in C#	130
24.1	Creating a Top-Level Menu	130
24.2	Deleting and Moving Menu Items.....	134
24.3	Assigning Keyboard Shortcuts to Menu Items.....	135
24.4	Programming Menu Items in C#	135
Chapter 25.	Creating Context Menus in C#	137
25.1	Adding Context Menus to a C# Form	137
25.2	Associating a Component with a Context Menu.....	139
25.3	Programming C# Context Menu Options.....	139
25.4	Compiling and Running the Application.....	140
Chapter 26.	Building a Toolbar with C# and Visual Studio	141
26.1	Creating a Toolbar	141
26.2	Adding Tooltip Text to Toolbar Controls.....	142
26.3	Programming Toolbar Controls.....	143
26.4	Changing the Toolbar Position	144
Chapter 27.	Drawing Graphics in C#	145
27.1	Persistent Graphics	145
27.2	Creating a Graphics Object.....	145
27.3	Creating a Pen In C#	146
27.4	Drawing Lines in C#	146
27.5	Drawing Squares and Rectangles in C#	147
27.6	Drawing Ellipses and Circles in C#	149
27.7	Drawing Text with C#	151
Chapter 28.	Using Bitmaps for Persistent Graphics in C#	153
28.1	Why Use Bitmaps for Graphics Persistence in C#?	153

28.2	Creating a Bitmap.....	153
28.3	Instantiating a Bitmap and Creating a Graphics Object.....	154
28.4	Drawing onto the Bitmap.....	155
28.5	Rendering a Bitmap Image on a Control.....	156
28.6	Changing the Background Color of a Bitmap.....	157

Chapter 1. About C# Essentials

The C# Essentials eBook contains 28 chapters of detailed information intended to provide everything necessary to gain proficiency as a C# programmer.

The book begins with a detailed overview of the C# development and runtime environment with overviews of the Common Language Infrastructure (CLI), Common Intermediate Language (CIL) and Virtual Execution System (VES) followed by simple steps to creating both Windows and console based C# applications.

A number of chapters are dedicated to the fundamentals of the C# programming language, including topics such as variables, constants, operators, flow control, looping and object oriented programming (including topics such as inheritance and abstract classes). Subsequent chapters focus on issues such as manipulating and formatting strings and working with arrays and collection classes.

Once the basics of the C# language have been covered the book focuses on the design of GUI based applications using C# in conjunction with Visual Studio. Topics include the design of key GUI elements (such as forms, toolbars and menus) and a detailed overview of event handling in GUI based C# applications.

The two final chapters of the book are dedicated to the topic of drawing graphics in C#, including the use of bitmap images to create persistent graphics.

On completing C# Essentials it is intended that the reader will have a firm knowledge foundation on which to begin developing complex C# based applications.

Chapter 2. The C# Language and Environment

C# is the latest progression in a never ending quest to make it as easy and efficient as possible for humans to program computers. Whilst it would be easy to simply describe C# as just another object oriented programming language developed by Microsoft (and ratified by ECMA and ISO), the fact is that C# is actually an integral part of an entire development and execution infrastructure. The primary object of this chapter of C# Essentials is to provide an overview of both the C# language and the infrastructure on which it relies. By the end of this chapter it also is intended that the reader will have a clear understanding of what acronyms such as CLI, CLR, VES, JIT and .NET mean.

2.1 A Brief History of Computer Programming Languages

The problem with programming is that computers think exclusively in numbers (the numbers 0 and 1 to be precise) known as *machine code* while humans communicate using words. In the very early days programmers actually entered machine code directly into computers to program them. This, as you can imagine, was a laborious and error prone process. The next evolution was to associate brief human readable commands with the corresponding machine code. For example, a programmer could enter the command *MOV* to transfer a value from one microprocessor register to another. These commands would then be translated into machine code by a piece of software called an *assembler*, thereby giving this command syntax the name *Assembly Language*.

Next came a series of *high level* languages designed to make it easier for humans to write programs. These programs are written using a human readable syntax and then either compiled to machine code by a *compiler* or interpreted on behalf of the processor by an *interpreter*. Such languages include BASIC, COBOL, Pascal and Fortran. One other such language is called *C* which was created at AT&T Bell Labs in the late 1960s and early 1970s. In the late 1970s and early 1980s work started on an object oriented approach to C programming culminating in a new, object oriented variant of C known as *C++*.

The story, however, does not end there. The problem with C++ was that it was an incredibly easy language in which to make programming mistakes. C++ would quite happily allow a programmer to make coding mistakes that would cause buffers to overflow, memory locations to be arbitrarily overwritten and introduce memory leaks that would cause applications to bloat to the point of using up the entire physical memory and swap space on a system. Another problem encountered with C, C++ and all other compiled languages is the fact that the source code has to re-compiled for each different processor type making it difficult to port an application from one hardware platform to another.

In order to address the short-comings of C and C++, Sun Microsystems started work on a new programming language and execution environment in the 1990s. The end result was called Java. Java consists of a programming language with many of the pitfalls of C++ removed, a portable intermediate byte code format, a runtime environment (called the *virtual machine*) that executes the byte code and handle issues such as memory management, and a vast suite of libraries providing all the functionality required to develop enterprise class applications (such as networking, file handling, database access, graphics etc).

Java gained rapid acceptance and for a time Microsoft began their Java embrace and extend campaign. Sun were happy for Microsoft to embrace Java but reached for their lawyers when they realized that the *extend* part was a plan for Microsoft to introduce their own proprietary version of the language. Politics ensued and Microsoft eventually walked away from Java. Not long after, Microsoft started talking about something called .Net, following by something else called C#.

2.2 What exactly is C#?

"What does all this history have to do with C#?" I hear you ask. Well, the origins of the C# programming syntax can be traced right back to C and C++. If you are already familiar with C or C++ then you have a big head start in terms of learning C#. In fact the same can be said of syntax similarities between Java, C, C++ and C# syntax. In addition, C# also inherits many of the benefits of Java in terms of memory handling (better known as *garbage collection*) and an intermediate byte code that negates the need to recompile an application for each target hardware platform. C# is also accompanied by a vast framework of libraries designed to provide the programmer with readymade solutions to just about every imaginable scenario.

Despite these similarities there are differences between the Java and C# infrastructures. The remainder of this chapter will be dedicated to providing an overview of the C# infrastructure.

2.3 The Common Language Infrastructure (CLI)

C# is an object oriented programming language. It is essentially a standard defining what constitutes valid syntax. On its own C# is actually of little use because it is dependent upon something called the *Common Language Infrastructure (CLI)* both for compilation and execution of applications. The CLI in turn, is actually a standard which defines specifications for the following components:

- Virtual Execution System (VES)
- Common Intermediate Language (CIL)

- Common Type System (CTS)
- Common Language Specification (CLS)
- Framework

In the remainder of this chapter we will look at each of these CLI components in order to build up a picture of how the CLI environment fits together.

2.4 Common Intermediate Language (CIL)

Unlike the C and C++ compilers which compile source code down to the machine code understood by the target microprocessor, the C# compiler compiles to an intermediate byte code format known as the Common Intermediate Language (CIL). This code can, in theory, be taken to any system where there is a CLI compliant Virtual Execution System (VES) and executed. There is, therefore, no need to compile an application for each and every target platform.

The word *Common* in Common Intermediate Language is used because this format is common to more than just the C# programming language. In fact any programming language may target the CIL allowing libraries and code modules from different languages to execute together in the same application. Typical languages for which CIL compilation is available include Visual Basic, COBOL, SmallTalk and C++.

2.5 Virtual Execution System (VES)

The VES (usually referred to as the *runtime*) is the environment in which the CIL byte code is executed. The VES reads the byte code generated by the C# compiler and uses something called a *Just in Time (JIT)* compiler to compile the byte code down to the native machine code of the processor on which it is running. While this code is executing it does so in conjunction with a runtime agent which essentially manages the execution process. As a result, this executing code is known as *managed code* and the process handles issues such as garbage collection (to handle memory allocation and de-allocation), memory access and type safety to ensure that the code does not do anything it is not supposed to do.

A term that is often used in connection with the VES is the *Common Language Runtime (CLR)*. The CLR is officially the name given to Microsoft's implementation of the VES component of the CLI specification.

It is worth noting that the JIT process can introduce a startup delay on execution of an application. One option available with .Net to avoid this problem is to pre-compile CLI byte code down to native machine code using the *NGEN* compiler. Because the NGEN compilation

must take place on the target processor architecture this step is often performed at the point that the application in question is installed by the user.

2.6 Common Type System (CTS) & Common Language Specification (CLS)

As mentioned previously a number of different programming languages target the CLI allowing, for example, code from C# sources to interact with code from Visual Basic. In order to achieve this feat, each language must have the same concept of how data types are stored in memory. The CTS, therefore, defines how a CLI compatible language must view the bit patterns of values and layout and behavior of objects to ensure interoperability.

The CLS is essentially a subset of the CTS aimed at creating interoperable libraries.

2.7 The Framework (Base Class and Framework Class Libraries)

The CLI specifies a set of base classes that must be available to executing CLI code, otherwise known as the *Base Class Library (BCL)*. The BCL contains APIs that enable executing CIL code to interact with the runtime environment and the underlying operating system.

Beyond the basics there is also the *Framework Class Library*. This is a Microsoft library which contains APIs for the creation of graphical user interfaces, database applications, web access and much, much more.

2.8 Non Microsoft Implementations of the CLI

Microsoft's implementation of the CLI stack is called .NET. .NET is not, however, the only implementation available. Another implementation provided by Microsoft is called *Rotor*. Rotor is available for Windows, Mac OS X and FreeBSD and is available in source form. Rotor, however, is primarily a learning tool and as such is licensed under terms which prohibit use as the basis of commercial applications.

Other significant open source implementations are the Mono and DotGNU projects targeted at Windows, Linux and UNIX platforms.