Microsoft

Start Here!™

FUNDAMENTALS OF

Microsoft®

.NET Programming

Rod Stephens

Ready to learn programming?

Start Here!™

Get ready to learn Microsoft .NET programming by exploring the basic concepts that drive all .NET-based languages. If you have absolutely no previous experience, no problem—simply start here! This book introduces must-know concepts and techniques through easy-to-follow explanations, examples, and exercises.

Here's where you start learning how software works

- Understand how the .NET development environment helps you design and run programs
- Delve into basic object-oriented concepts such as properties, methods, and events
- Learn how programs store data in files, object stores, and databases
- Dig into controls—labels, text boxes, menus, scroll bars
- See how programs take advantage of multicore processors
- Get an extensive glossary of key programming terms

NOTE: This title is included as a free companion eBook for the Start Here! books for Microsoft Visual Basic® and Visual C#®. This print edition is available for sale for anyone interested in .NET fundamentals.

ISBN: 978-0-7356-4104-1

U.S.A. $19.99
Canada $20.99
[Recommended]

Programming/Microsoft Visual Studio

Start Here! Fundamentals of Microsoft® .NET Programming

For Skill Level: Beginner
Prerequisites: None

RESOURCE ROADMAP

Start Here
- Beginner-level instruction
- Easy to follow explanations and examples
- Exercises to build your first projects

Step by Step
- For experienced developers learning a new topic
- Focus on fundamental techniques and tools
- Hands-on tutorials with practice files plus eBook

Developer Reference
- For professional developers, intermediate to advanced
- Expert coverage of essential topics and techniques
- Features extensive, adaptable code examples

Focused Topics
- For programmers who develop complex or advanced solutions
- Specialized topics in narrow focus; deep coverage
- Features extensive, adaptable code examples

microsoft.com/mspress

About the Author

Rod Stephens is president of a computer consulting company, has 20 years of experience as a professional developer, and is the author of more than 20 books and 250 articles about programming.

Microsoft

Microsoft

Start Here!™

Fundamentals of Microsoft® .NET Programming

Rod Stephens

Published with the authorization of Microsoft Corporation by:
O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, California 95472

ISBN: 978-0-7356-6168-4

1 2 3 4 5 6 7 8 9 LSI 6 5 4 3 2 1

Printed and bound in the United States of America.

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at *mspinput@microsoft.com*. Please tell us what you think of this book at *http://www.microsoft.com/learning/booksurvey*.

Acquisitions and Developmental Editor: Russell Jones

Production Editor: Jasmine Perez

Editorial Production: S4Carlisle Publishing Services

Technical Reviewer: Debbie Timmins

Indexer: WordCo Indexing Services, Inc.

Cover: Valerie DeGiulio

Contents at a Glance

Contents

Introduction

Programming languages do one very simple thing: they allow you to write programs that tell the computer what to do. You can tell a computer to read a value from the keyboard, add two numbers, save a result in a file on the hard disk, or draw a smiley face on the screen.

No matter what programming language you use, the underlying commands that the computer can execute are exactly the same. Whether you use Java, C#, Microsoft Visual Basic, COBOL, LISP, or any other language, you can make the computer perform roughly the same tasks. Two languages may have very different syntaxes, and some languages make some tasks easier than others, but the fundamental operations they can perform are the same. All these languages can carry out numeric calculations and manipulate files; unfortunately, none of them can reliably pick lottery winners. (If you write a program that can, let me know!)

At a more conceptual level, programming concepts have been refined over the years until most modern languages share a common set of fundamental concepts, such as variables, classes, objects, forms, menus, files, and multiprocessing. Don't worry if you don't know what these are—the purpose of this book is to provide more information about such terms and concepts.

Because programming languages share so many operations and concepts, programming books tend to cover the same topics as well. Books about databases or graphics cover these specialized topics in great detail. Different authors may place emphasis on different subjects, but there's a lot of overlap, particularly in beginning and general "how to program" books. Every one of these books explains what a variable is, how to create objects, and what a text file contains.

All this means that if you want to learn more than one programming language (a practice that I highly recommend), you're going to encounter much of the same material repeatedly. Even if you skim the familiar sections, you still have to pay for the content. You may start with a 600-page book about Visual Basic programming. Later, when you buy a 500-page C# book, you'll discover that 200 of those pages cover things you already know. Next, when your boss decides you need to learn LISP, you'll find that your new 550-page book contains 100 pages that you already know. (There are a lot of differences between LISP and the other two languages, so there will be less overlap if you shift to that language.)

This edition of the *Start Here!* series changes all that. Rather than making each *Start Here!* book cover the exact same topics, those common topics have been moved into this volume for easy reference. Now if you read *Start Here! Learn Microsoft Visual C# 2010 Programming* or *Start Here! Learn Microsoft Visual Basic Programming,* you won't need to rehash the exact same topics. Instead, those books refer you to this one for background information, such as how disk buffering works, freeing the other books to focus on language-specific issues.

There still will be some overlap between any books about different languages. For example, Visual Basic and C# both let you read and write disk files. Although *Start Here! Fundamentals of .NET Programming* explains in general what disk files are and how programs interact with them, the other books still need to explain the syntax for the code that their respective languages use to read and write files.

Note that this book doesn't necessarily cover every last detail of each background topic. It just gives you the information you need to understand how programs fit into a larger context so that you can get the most out of them. For example, this book explains some important programming issues relating to disk drives, but it doesn't explain in detail how disk drives work.

Moving underlying common topics into this separate book provides several benefits, including the following:

- Other *Start Here!* books can spend less time on the background material covered in this book and more time on language-specific issues. Those books can rely on and refer to this book to provide extra detail as needed.
- This book provides more room for, and spends more time on, basic concepts that beginning programming books often must gloss over to make room for language-specific concepts.
- This book provides a single location for learning about general computer topics, without focusing on a particular language. This is important because it can give you a broader understanding of what you can make computers do easily and what might be difficult to make a computer do, regardless of which programming language you choose.
- This book can act as an enhanced glossary, giving you a place to look for explanations of common computer terms. A normal glossary briefly defines key terms, but in addition, the rest of the book provides much more detail about important concepts.

Who Should Read This Book

This book is for anyone who wants a basic understanding of computers and the environments in which programs operate. It provides background information that is useful when you are trying to learn to use any programming language. It also provides information that can help you understand how programs work in general. For example, it explains what multithreading is and why multi-core computers may not always perform much better than single-core systems.

Assumptions

This book does *not* assume that you have any previous programming experience. In fact, it doesn't even assume that you have a computer! Instead, this book is about understanding computers and programs in general, and Microsoft Windows and .NET concepts in particular, not about writing programs in a specific language.

This book is intended for two main audiences: those who want to learn a new programming language, particularly those who are reading one of the other books in the *Start Here!* series, and those who want a better overall understanding of computers.

Although the content of this book is as general as possible, it is not primarily intended as a stand-alone work; instead, it's intended as an accompanying volume for use with other *Start Here!* books, which cover a range of languages and technologies. Most of the information you'll find here applies to computers and programs running Windows, but many of the concepts also apply to other operating systems, such as Unix, Linux, or OS X. Sometimes, however, specificity aids clarity, so in some places this book is targeted toward Windows.

Who Should Not Read This Book

If you're interested in general programming—particularly non-Windows and non–Microsoft .NET Framework programming—this book is not for you. Much of the information in this book applies to programming in general, but a substantial portion of the information applies to .NET Framework topics, and this book does not make any particular effort to distinguish between general and Windows- or .NET Framework–specific information.

What You Need to Use This Book

If you want to use this book, all you'll need is this book. No computer, no software, no programming language, and no programming experience is required!

Organization of This Book

If you just want a better understanding of computers and programming concepts, you can simply read the book.

If you're reading this book along with one of the other *Start Here!* books, you can take a couple of different approaches. First, you can use this book as a reference for the other. When you reach the part of *Start Here! Learn Microsoft Visual C# 2010 Programming* that discusses operators, you may want to read this book's chapter on operators for additional background. *Start Here! Learn Microsoft Visual C# 2010 Programming* may also explicitly refer to places in this book where you can get additional information on a topic.

Another approach to using this book with another *Start Here!* book would be to read this one at odd moments when it's hard to read the other one. For example, I like to use my computer to work through examples and experiment with the code as I'm learning a new language. That makes it hard for me to work through a book like *Start Here! Learn Microsoft Visual C# 2010 Programming* on the bus, waiting at the dentist's office, or while sunning myself on the beach. In contrast, *Start Here! Fundamentals of Microsoft .NET Programming* doesn't require a computer, so it's easy to read just about anywhere (although at the beach, I'd rather play volleyball anyway).

This book is divided into 14 chapters plus a glossary. The chapters are independent, so you can read them in any order. In fact, many of the sections in the chapters are independent, so you can jump around within a chapter to suit your interests and needs.

- **Chapter 1, "Computer Hardware,"** briefly describes the hardware of a computer system. It explains terms such as *computer processing unit (CPU), graphics processing unit (GPU), random access memory (RAM),* and *multi-core*, and explains why those terms are important to programmers. It explains how memory, disk accesses, and other hardware issues can affect a program's performance.
- **Chapter 2, "Multiprocessing,"** summarizes some of the challenges that face programmers writing multiprocessing programs. It explains how the future of programming is likely to be highly parallel and summarizes the Task Parallel Library (TPL) that makes programming for multi-core systems easier.

- **Chapter 3, "Programming Environments,"** explains what a programming environment is and describes some of the features that make Microsoft Visual Studio one of the best programming environments available. It explains how a program's code must be compiled and how a programming environment can make that code transparent to the programmer.
- **Chapter 4, "Windows Program Components,"** describes the pieces of a Windows program from the user's point of view. It describes menus, content menus, accelerators, shortcuts, and dialog boxes. It also mentions several design considerations that beginning programmers should understand if they want to make programs easier to use.
- **Chapter 5, "Controls,"** describes in general terms what controls and components are and how they are used. It also mentions some common properties, such as *Dock* and *Anchor,* which make using many controls easier for the programmer.
- **Chapter 6, "Variables,"** explains the concept of a variable. It explains variable concepts such as data types, conversions, strong and weak type checking, value versus reference types, scope, and accessibility.
- **Chapter 7, "Control Statements,"** describes control statements, such as *If Then* and *For Each,* which a program uses to manage a program's flow. It describes these statements in general terms, provides some examples in pseudocode, and shows a few simple examples in Visual Basic and C# for comparison.
- **Chapter 8, "Operators,"** explains operators. It discusses precedence rules and operator overloading.
- **Chapter 9, "Routines,"** explains what routines are and how they are useful in programming. It describes different kinds of routines, such as methods, subroutines, and functions. It also defines parameters and explains the confusing topic of parameters passed by value or passed by reference.
- **Chapter 10, "Object-Oriented Programming,"** provides an introduction to object-oriented programming. It explains classes, constructors, and destructors. It describes non-deterministic finalization.
- **Chapter 11, "Development Techniques,"** describes basic programming techniques such as using comments, naming conventions, interfaces, and generic classes.

- **Chapter 12, "Globalization,"** explains how to localize a program in Visual Studio so that it works in multiple places. It explains several localization issues, such as different date, number, and currency formats.
- **Chapter 13, "Data Storage,"** describes different methods for storing data such as using the registry, configuration files, and files on disk. It explains different kinds of files, such as Extensible Markup Language (XML) files and databases, and mentions some of the classes that a program would use to work with different kinds of files.
- **Chapter 14, ".NET Libraries,"** summarizes some of the libraries that are most useful in writing NET programs. These libraries let a program encrypt and decrypt information, work with data structures such as stacks and queues, interrogate objects and types to learn about them, and work with multiple threads of execution.
- **The glossary** provides a brief summary of key terms to remind you of their meaning. (In a sense, the whole book acts as a glossary for use by the other *Start Here!* books.) It summarizes key concepts in one or two sentences.

Conventions and Features in This Book

To help you get the most from the text and keep track of what's happening, I've used several conventions throughout the book.

Splendid Sidebars

Sidebars such as this one contain additional information and side topics.

Warning Boxes with a Warning icon like this one hold important, not-to-be-forgotten information that is directly relevant to the surrounding text.

Note The Note icon indicates notes and asides to the current discussion. They are offset and placed in a box like this.

Tip The Tip icon indicates tips, bits and pieces of advice on effective programming. They are offset and placed in a box like this.

More Info The More Info icon indicates somewhere you can go to learn for more information on a particular topic, such as a webpage. They are offset and placed in a box like this.

As for styles in the text:

- New terms and important words are *italicized* when they are introduced. You can also find many of them in the glossary at the end of the book.
- Keyboard keystrokes look like this: Ctrl+A. The plus sign means that you should hold down the Ctrl key and then press the A key.
- Uniform Resource Locators (URLs), code, and email addresses within the text are shown in italics, as in *http://www.vb-helper.com*, *x = 10*, and *RodStephens@vb-helper.com*.

```
Separate code examples use a monofont type with no highlighting.
Bold  text emphasizes code that's particularly important in the current
context.
```

Note The code editor in Visual Studio provides a rich color scheme to indicate various parts of code syntax such as variables, comments, and Visual Basic keywords. The code editor and the Intellisense feature of Visual Studio are excellent tools to help you learn language features in the editor and help you prevent mistakes as you code. However, the colors that you can see in Visual Studio don't show up in the code in this book.

Source Code

Because this book covers concepts that are independent of any particular programming language, it also includes little source code from any particular language. You'll find occasional bits of source code used to contrast the syntaxes of different languages; but more often, this book uses *pseudocode* to demonstrate programming constructs. Pseudocode is an informal high-level "language" that looks sort of like a programming

language, but isn't really. It's intended to describe a situation sufficiently so that you *could* implement the actual code in whatever language you are using.

For example, the following code shows a *for* loop in pseudocode, which repeats a particular operation a specific number of times:

```
For <variable> From 1 To 100
    Do something
```

This pseudocode says the program should make a variable (however you create a variable in the language you're using) and then loop starting at value 1 and finishing at value 100. For each trip through the loop, the program should "Do something."

Contrast this with the following C# code:

```
for (int i = 1; i <= 100; i++)
{
    DoSomething();
}
```

This code does the same thing as the previous pseudocode, but its syntax makes understanding the code harder—unless, of course, you know C# (or some related language, such as C++ or Java). If you don't know C#, you may have trouble understanding the point that this code is illustrating.

Acknowledgments

Thanks to Russell Jones, Diane Kohnen, Dan Fauxsmith, Jasmine Perez, and all the others at O'Reilly Media and Microsoft Press who worked so hard to make this book possible. Also thanks to John Mueller, Evangelos Petroutsos, and authors of the language-centric books in this Start Here! series. Between us I think we've put together a great set of resources!

Errata & Book Support

We've made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site at *oreilly.com:*

http://go.microsoft.com/FWLink/?Linkid=230431

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software is *not* offered through the addresses above.

We Want to Hear from You

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://www.microsoft.com/learning/booksurvey

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in Touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

Other Resources

You can find more information about this book at *http://www.vb-helper.com/start_here_fundamentals.html* or at *http://www.CSharpHelper.com/start_here_fundamentals.html*. Both of these pages provide links to updates, addenda, and other information related to this book.

If you're interested, subscribe to one of my Visual Basic newsletters at *www.vb-helper.com/newsletter.html* or visit my C# blog at *blog.CSharpHelper.com*.

If you have questions, comments, or suggestions, please feel free to email me at *RodStephens@vb-helper.com* or *RodStephens@CSharpHelper.com*. I can't promise to solve all your problems, but I do promise to try to help.