

BOOKS FOR PROFESSIONALS BY PROFESSIONALS™



*Foreword by Anders Hejlsberg,
architect of C#, Delphi, and Turbo Pascal*

Eric Gunnerson

A Programmer's Introduction to C#

Learn C# from a member of Microsoft's C# design team.



Teaches you to program with C# and to author .NET components



Get up-to-speed quickly on the inner workings of C#

Apress™

**Release Team[oR] 2001
[x] Programming**



A Programmer's Introduction to C#

by Eric Gunnerson

ISBN: 1893115860

Apress © 2000, 358 pages

This book takes the C programmer through the all the details—from basic to advanced-- of the new Microsoft C# language.

[Companion Web Site](#)

[Table of Contents](#)

[Colleague Comments](#)

[Back Cover](#)

Synopsis

Written as an introduction to the new C#, this guide takes the experienced C programmer a few steps beyond the basics. It covers objects, data types, and flow control, and even delves into some background on the new Microsoft NET Frameworks environment. Keeping in mind that this is for those familiar with C (and even Java), the book goes into some of the advanced features and improvements found in this new language. It also offers a comparison between C#, C++, Visual Basic, and Java.

| | |
|--|--------|
| A Programmer's Introduction to C# | - 9 - |
| Foreword | - 10 - |
| About This Book | - 10 - |
| Introduction | - 11 - |
| Why Another Language? | - 11 - |
| C# Design Goals | - 11 - |
| The C# Compiler and Other Resources | - 12 - |
| Chapter 1: Object-Oriented Basics | - 13 - |
| Overview | - 13 - |
| What Is an Object? | - 13 - |
| Inheritance | - 13 - |
| Polymorphism and Virtual Functions | - 14 - |
| Encapsulation and Visibility | - 16 - |
| Chapter 2: The .Net Runtime Environment | - 16 - |
| Overview | - 16 - |
| The Execution Environment | - 17 - |
| Metadata | - 18 - |
| Assemblies | - 19 - |
| Language Interop | - 19 - |
| Attributes | - 19 - |
| Chapter 3: C# Quickstart | - 20 - |
| Overview | - 20 - |
| Hello, Universe | - 20 - |
| Namespaces and Using | - 20 - |
| Namespaces and Assemblies | - 21 - |
| Basic Data Types | - 22 - |
| Classes, Structs, and Interfaces | - 23 - |

| | |
|---|--------|
| Statements | - 23 - |
| Enums | - 23 - |
| Delegates and Events | - 24 - |
| Properties and Indexers | - 24 - |
| Attributes | - 24 - |
| Chapter 4: Exception Handling | - 25 - |
| Overview | - 25 - |
| What's Wrong with Return Codes? | - 25 - |
| Trying and Catching | - 25 - |
| The Exception Hierarchy | - 26 - |
| Passing Exceptions on to the Caller | - 28 - |
| User-Defined Exception Classes | - 30 - |
| Finally | - 31 - |
| Efficiency and Overhead | - 33 - |
| Design Guidelines | - 33 - |
| Chapter 5: Classes 101 | - 33 - |
| Overview | - 33 - |
| A Simple Class | - 33 - |
| Member Functions | - 35 - |
| ref and out Parameters | - 36 - |
| Overloading | - 38 - |
| Chapter 6: Base Classes And Inheritance | - 39 - |
| Overview | - 39 - |
| The Engineer Class | - 39 - |
| Simple Inheritance | - 40 - |
| Arrays of Engineers | - 42 - |
| Virtual Functions | - 45 - |
| Abstract Classes | - 47 - |
| Sealed Classes | - 50 - |
| Chapter 7: Class Member Accessibility | - 51 - |
| Overview | - 51 - |
| Class Accessibility | - 51 - |
| Using internal on Members | - 51 - |
| The Interaction of Class and Member Accessibility | - 52 - |
| Chapter 8: Other Class Stuff | - 52 - |
| Overview | - 53 - |
| Nested Classes | - 53 - |
| Other Nesting | - 53 - |
| Creation, Initialization, Destruction | - 54 - |
| Overloading and Name Hiding | - 56 - |
| Static Fields | - 57 - |
| Static Member Functions | - 58 - |
| Static Constructors | - 59 - |
| Constants | - 59 - |
| readonly Fields | - 60 - |
| Private Constructors | - 63 - |
| Variable-Length Parameter Lists | - 63 - |
| Chapter 9: Structs (Value Types) | - 65 - |
| Overview | - 65 - |
| A Point Struct | - 65 - |
| Boxing and Unboxing | - 66 - |
| Structs and Constructors | - 66 - |

| | |
|--|---------|
| Design Guidelines | - 67 - |
| Chapter 10: Interfaces | - 67 - |
| Overview | - 67 - |
| A Simple Example | - 67 - |
| Working with Interfaces | - 68 - |
| The <code>as</code> Operator | - 70 - |
| Interfaces and Inheritance | - 71 - |
| Design Guidelines | - 72 - |
| Multiple Implementation | - 72 - |
| Interfaces Based on Interfaces | - 77 - |
| Chapter 11: Versioning Using <code>new</code> and <code>override</code> | - 77 - |
| Overview | - 77 - |
| A Versioning Example | - 77 - |
| Chapter 12: Statements and Flow of Execution | - 79 - |
| Overview | - 79 - |
| Selection Statements | - 79 - |
| Iteration Statements | - 81 - |
| Jump Statements | - 85 - |
| Definite Assignment | - 85 - |
| Chapter 13: Local Variable Scoping | - 88 - |
| Overview | - 88 - |
| Chapter 14: Operators | - 89 - |
| Overview | - 89 - |
| Operator Precedence | - 89 - |
| Built-In Operators | - 90 - |
| User-Defined Operators | - 90 - |
| Numeric Promotions | - 90 - |
| Arithmetic Operators | - 90 - |
| Relational and Logical Operators | - 92 - |
| Assignment Operators | - 94 - |
| Type Operators | - 94 - |
| Chapter 15: Conversions | - 96 - |
| Overview | - 96 - |
| Numeric Types | - 96 - |
| Conversions of Classes (Reference Types) | - 100 - |
| Conversions of Structs (Value Types) | - 103 - |
| Chapter 16: Arrays | - 103 - |
| Overview | - 103 - |
| Array Initialization | - 103 - |
| Multidimensional and Jagged Arrays | - 104 - |
| Arrays of Reference Types | - 105 - |
| Array Conversions | - 106 - |
| System.Array Type | - 106 - |
| Chapter 17: Strings | - 107 - |
| Overview | - 107 - |
| Operations | - 107 - |
| Converting Objects to Strings | - 109 - |
| Regular Expressions | - 111 - |
| Chapter 18: Properties | - 115 - |
| Overview | - 115 - |
| Accessors | - 115 - |
| Properties and Inheritance | - 116 - |

| | |
|---|---------|
| Use of Properties | - 116 - |
| Side Effects When Setting Values | - 117 - |
| Static Properties | - 119 - |
| Property Efficiency | - 120 - |
| Chapter 19: Indexers | - 120 - |
| Overview | - 121 - |
| Indexing with an Integer Index | - 121 - |
| Indexers and <code>foreach</code> | - 125 - |
| Design Guidelines | - 128 - |
| Chapter 20: Enumerators | - 128 - |
| Overview | - 128 - |
| A Line Style Enumeration | - 128 - |
| Enumerator Base Types | - 130 - |
| Initialization | - 130 - |
| Bit Flag Enums | - 131 - |
| Conversions | - 131 - |
| Chapter 21: Attributes | - 132 - |
| Overview | - 132 - |
| Using Attributes | - 133 - |
| An Attribute of Your Own | - 136 - |
| Reflecting on Attributes | - 138 - |
| Chapter 22: Delegates | - 139 - |
| Overview | - 140 - |
| Using Delegates | - 140 - |
| Delegates as Static Members | - 141 - |
| Delegates as Static Properties | - 143 - |
| Chapter 23: Events | - 145 - |
| Overview | - 145 - |
| A New Email Event | - 145 - |
| The Event Field | - 147 - |
| Multicast Events | - 147 - |
| Sparse Events | - 147 - |
| Chapter 24: User-Defined Conversions | - 149 - |
| Overview | - 149 - |
| A Simple Example | - 149 - |
| Pre- and Post- Conversions | - 151 - |
| Conversions Between Structs | - 152 - |
| Classes and Pre- and Post- Conversions | - 157 - |
| Design Guidelines | - 163 - |
| How It Works | - 165 - |
| Chapter 25: Operator Overloading | - 167 - |
| Overview | - 167 - |
| Unary Operators | - 167 - |
| Binary Operators | - 167 - |
| An Example | - 168 - |
| Restrictions | - 169 - |
| Design Guidelines | - 169 - |
| Chapter 26: Other Language Details | - 169 - |
| Overview | - 170 - |
| The Main Function | - 170 - |
| Preprocessing | - 171 - |
| Preprocessing Directives | - 171 - |

| | |
|--|---------|
| Lexical Details | - 174 - |
| Chapter 27: Making Friends with the .NET Frameworks | - 177 - |
| Overview | - 177 - |
| Things All Objects Will Do | - 177 - |
| Hashes and GetHashCode () | - 179 - |
| Chapter 28: System.Array and the Collection Classes | - 182 - |
| Overview | - 182 - |
| Sorting and Searching | - 182 - |
| Design Guidelines | - 194 - |
| Chapter 29: Interop | - 195 - |
| Overview | - 196 - |
| Using COM Objects | - 196 - |
| Being Used by COM Objects | - 196 - |
| Calling Native DLL Functions | - 196 - |
| Chapter 30: .NET Frameworks Overview | - 196 - |
| Overview | - 196 - |
| Numeric Formatting | - 196 - |
| Date and Time Formatting | - 204 - |
| Custom Object Formatting | - 205 - |
| Numeric Parsing | - 207 - |
| Using XML in C# | - 208 - |
| Input/Output | - 208 - |
| Serialization | - 211 - |
| Threading | - 214 - |
| Reading Web Pages | - 215 - |
| Chapter 31: Deeper into C# | - 217 - |
| Overview | - 217 - |
| C# Style | - 217 - |
| Guidelines for the Library Author | - 217 - |
| Unsafe Code | - 218 - |
| XML Documentation | - 222 - |
| Garbage Collection in the .NET Runtime | - 225 - |
| Deeper Reflection | - 228 - |
| Optimizations | - 234 - |
| Chapter 32: Defensive Programming | - 234 - |
| Overview | - 234 - |
| Conditional Methods | - 234 - |
| Debug and Trace Classes | - 235 - |
| Asserts | - 235 - |
| Debug and Trace Output | - 236 - |
| Using Switches to Control Debug and Trace | - 238 - |
| Chapter 33: The Command Line | - 243 - |
| Overview | - 243 - |
| Simple Usage | - 243 - |
| Response Files | - 243 - |
| Command-Line Options | - 243 - |
| Chapter 34: C# Compared to Other Languages | - 246 - |
| Overview | - 246 - |
| Differences Between C# and C/C++ | - 246 - |
| Differences Between C# and Java | - 248 - |
| Differences Between C# and Visual Basic 6 | - 253 - |
| Other .NET Languages | - 257 - |

| | |
|---|---------|
| Chapter 35: C# Futures | - 258 - |
| List of Figures | - 258 - |
| Chapter 2: The .Net Runtime Environment | - 258 - |
| Chapter 3: C# Quickstart | - 258 - |
| Chapter 9: Structs (Value Types) | - 258 - |
| Chapter 15: Conversions | - 258 - |
| Chapter 16: Arrays | - 258 - |
| Chapter 31: Deeper into C# | - 258 - |
| List of Tables | - 258 - |
| Chapter 30: .NET Frameworks Overview | - 258 - |
| Chapter 33: The Command Line | - 258 - |
| List of Sidebars | - 258 - |
| Chapter 21: Attributes | - 258 - |

Table of Contents

[A Programmer's Introduction to C#](#)

[Foreword](#)

[About This Book](#)

[Introduction](#)

[Chapter 1](#) - Object-Oriented Basics

[Chapter 2](#) - The .Net Runtime Environment

[Chapter 3](#) - C# Quickstart

[Chapter 4](#) - Exception Handling

[Chapter 5](#) - Classes 101

[Chapter 6](#) - Base Classes And Inheritance

[Chapter 7](#) - Class Member Accessibility

[Chapter 8](#) - Other Class Stuff

[Chapter 9](#) - Structs (Value Types)

[Chapter 10](#) - Interfaces

[Chapter 11](#) - Versioning Using new and override

[Chapter 12](#) - Statements and Flow of Execution

[Chapter 13](#) - Local Variable Scoping

[Chapter 14](#) - Operators

[Chapter 15](#) - Conversions

[Chapter 16](#) - Arrays

[Chapter 17](#) - Strings

[Chapter 18](#) - Properties

[Chapter 19](#) - Indexers

[Chapter 20](#) - Enumerators

[Chapter 21](#) - Attributes

[Chapter 22](#) - Delegates

[Chapter 23](#) - Events

[Chapter 24](#) - User-Defined Conversions

[Chapter 25](#) - Operator Overloading

[Chapter 26](#) - Other Language Details

[Chapter 27](#) - Making Friends with the .NET Frameworks

[Chapter 28](#) - System.Array and the Collection Classes

[Chapter 29](#) - Interop

[Chapter 30](#) - .NET Frameworks Overview

[Chapter 31](#) - Deeper into C#

[Chapter 32](#) - Defensive Programming

[Chapter 33](#) - The Command Line

[Chapter 34](#) - C# Compared to Other Languages

[Chapter 35](#) - C# Futures

[Index](#)

[List of Figures](#)

[List of Tables](#)

[List of Sidebars](#)

Back Cover

- Provides in-depth information about the functionality of the language and C# “Quick Start”
- Shows you how to write components that fit seamlessly into the .NET Frameworks
- Includes C# reference information tailored for C++, Java and Visual Basic Programmers
- Suitable for intermediate to advanced developers and includes coverage of advanced topics in C#

Eric Gunnerson, A member of the Microsoft C# design team, has written a comprehensive C# tutorial addressed to the experienced programmer. *A Programmer’s Introduction to C#* explains how C# works, why it was designed the way it was, and how C# fits into Microsoft’s new .NET Frameworks. This book teaches programmers how to write C# components and how to truly leverage the power of the new .NET Runtime.

Gunnerson’s first chapters are for the impatient programmer. In them, he provides an introduction to object-oriented programming with C# along with a C# “Quick Start” for those who want a fast track to programming in C#. This is followed by a more comprehensive section in which he uses his unique insider’s view to explain each of the new C# language features in detail. He covers fundamentals such as classes, structs, attributes, statements and flow of execution, arrays, delegates and events, exception handling, and the unique interoperability provided by the .NET Frameworks.

In the final portion of the book, Gunnerson provides a useful overview of the .NET Frameworks. A section on the .NET Common Language Runtime and Frameworks shows how to write components that function well in the runtime and how to use the basic runtime features (such as I/O). Gunnerson also devoted time to more advanced topics such as regular expressions and collections. Final chapters include Making Friends with the .NET Frameworks, System.Array and the Collection Classes, .NET Frameworks Overview, Deeper into C# and Defensive Programming. Also included is a detailed C# language comparison that will be indispensable for programmers currently working in C++, Java, or Visual Basic.

All of the source code for this book is online at <http://www.apress.com>.

About the Author

Eric Gunnerson is a software design engineer in Microsoft's Visual C++ QA group and a member of the C# design team. In the course of his professional career, he has worked primarily on database products and tools – and is proud of the fact that nearly half of the companies he has worked for remain in business.

A Programmer's Introduction to C#

ERIC GUNNERSON

Copyright ©2000 by Eric Gunnerson

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher. ISBN (pbk): 1-893115-86-0

Printed and bound in the United States of America 2345678910

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Editorial Directors: Dan Appleman, Gary Cornell, Karen Watterson

Technical Reviewers: David Staheli, Shawn Vita, Gus Perez, Jerry Higgins, Brenton Webster

Editor: Andy Carroll

Projects Manager: Grace Wong

Production Editor: Janet Vail

Page Compositor and Soap Bubble Artist: Susan Glinert

Artist: Karl Miyajima

Indexer: Nancy Guenther

Cover and Interior Design: Derek Yee Design

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany

In the United States, phone 1-800-SPRINGER orders@springer-ny.com ;

<http://www.springer-ny.com> Outside the United States, contact orders@springer.de;

<http://www.springer.de> ; fax +49 6221 345229

For information on translations, please contact Apress directly at 901 Grayson Street, Suite 204, Berkeley, CA, 94710 Phone: 510-549-5931; Fax: 510-549-5939; info@apress.com ;

<http://www.apress.com>

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

Dedication

To Tony Jongejan, for introducing me to programming and being ahead of his time.

Acknowledgments

THOUGH WRITING A BOOK is often a lonely undertaking, no author can do it without help.

I'd like to thank all those who helped me with the book, including all those team members who answered my incessant questions and read my unfinished drafts. I would also like to thank my managers and Microsoft, both for allowing me to work on such a unique project and for allowing me to write a book about it.

Thanks to the Apress team for making a bet on an unproven author and for not pestering me when I waited to turn in content.

Thanks to all the artists who provided music to write to—all of which was commercially purchased—with special thanks to Rush for all their work.

Finally, I'd like to thank all those who supported me at home; my wife Kim and daughter Samantha who didn't complain when I was working, even when it was during our vacation, and for my cat for holding my arms down while I was writing.

Foreword

WHEN YOU CREATE a new programming language, the first question you're asked invariably is, why? In creating C# we had several goals in mind:

- *To produce the first component-oriented language in the C/C++ family.* Software engineering is less and less about building monolithic applications and more and more about building components that slot into various execution environments; for example, a control in a browser or a business object that executes in ASP+. Key to such components is that they have properties, methods, and events, and that they have attributes that provide declarative information about the component. All of these concepts are first-class language constructs in C#, making it a very natural language in which to construct and use components.
- *To create a language in which everything really is an object.* Through innovative use of concepts such as boxing and unboxing, C# bridges the gap between primitive types and classes, allowing any piece of data to be treated as an object. Furthermore, C# introduces the concept of value types, which allows users to implement lightweight objects that do not require heap allocation.
- *To enable construction of robust and durable software.* C# was built from the ground up to include garbage collection, structured exception handling, and type safety. These concepts completely eliminate entire categories of bugs that often plague C++ programs.
- *To simplify C++, yet preserve the skills and investment programmers already have.* C# maintains a high degree of similarity with C++, and programmers will immediately feel comfortable with the language. And C# provides great interoperability with COM and DLLs, allowing existing code to be fully leveraged.

We have worked very hard to attain these goals. A lot of the hard work took place in the C# design group, which met regularly over a period of two years. As head of the C# Quality Assurance team, Eric was a key member of the group, and through his participation he is eminently qualified to explain not only how C# works, but also why it works that way. That will become evident as you read this book.

I hope you have as much fun using C# as those of us on the C# design team had creating it.
Anders Hejlsberg
Distinguished Engineer
Microsoft Corporation

About This Book

C# IS ONE OF THE MOST EXCITING projects I've ever had the privilege to work on. There are many languages with different strengths and weaknesses, but once in a while a new language comes along that meshes well with the hardware, software, and programming approaches of a specific time. I believe C# is such a language. Of course, language choice is often a "religious issue." ^[1]

I've structured this book as a tour through the language, since I think that's the best and most interesting way to learn a language. Unfortunately, tours can often be long and boring, especially if the material is familiar, and they sometimes concentrate on things you don't care about, while overlooking things you're interested in. It's nice to be able to short-circuit the boring stuff and get into the interesting stuff. To do that, there are two approaches you might consider:

To start things off quickly, there's [Chapter 3](#), "C# QuickStart," which is a quick overview of the language, and gives enough information to start coding.

[Chapter 34](#), "C# Compared to Other Languages," offers language-specific comparisons for C++, VB, and Java for programmers attuned to a specific language, or for those who like to read comparisons.