Build Your Own .NET Language and Compiler

EDWARD G. NILGES

APress Media, LLC

Build Your Own .NET Language and Compiler

© Edward G. Nilges 2004

Originally published by Apress, in 2004

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 978-1-59059-134-5

ISBN 978-1-4302-0698-9 (eBook)

DOI 10.1007/978-1-4302-0698-9

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.

Lead Editor: Dan Appleman

Technical Reviewer: William Steele

Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, John Franklin, Jason Gilmore, Chris Mills, Steven Rycroft, Dominic Shakeshaft, Jim Sumser, Karen Watterson, Gavin Wray, John Zukowski

Assistant Publisher: Grace Wong

Project Manager: Beth Christmas

Copy Manager: Nicole LeClerc

Copy Editor: Marilyn Smith

Production Manager: Kari Brooks

Production Editor: Kelly Winquist

Compositor: Linda Weidemann, Wolf Creek Press

Proofreader: Elizabeth Berry

Indexer: Bill Johncocks

Artist: Kinetic Publishing

Cover Designer: Kurt Krames

Manufacturing Manager: Tom DeBolski

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, email orders@springer-ny.com, or visit <http://www.springer-ny.com>. Outside the United States: fax +49 6221 345229, email orders@springer.de, or visit <http://www.springer.de>.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, email info@apress.com, or visit <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(s) 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.

The source code for this book is available to readers at <http://www.apress.com> in the Downloads section. You will need to answer questions pertaining to this book in order to successfully download the code.

Contents at a Glance

About the Author	<i>ix</i>
Acknowledgments	<i>xi</i>
Introduction	<i>xiii</i>
Chapter 1 A Brief History of Compiler Technology	<i>1</i>
Chapter 2 A Brief Introduction to the .NET Framework	<i>15</i>
Chapter 3 A Compiler Flyover	<i>27</i>
Chapter 4 The Syntax for the QuickBasic Compiler	<i>51</i>
Chapter 5 The Lexical Analyzer for the QuickBasic Compiler	<i>91</i>
Chapter 6 QuickBasic Object Modeling	<i>133</i>
Chapter 7 The Parser and Code Generator for the QuickBasic Compiler	<i>171</i>
Chapter 8 Developing Assemblers and Interpreters	<i>205</i>
Chapter 9 Code Generation to the Common Language Runtime	<i>229</i>
Chapter 10 Implementing Business Rules	<i>243</i>
Chapter 11 Language Design: Some Notes	<i>267</i>
Appendix A quickBasicEngine Language Manual	<i>287</i>
Appendix B quickBasicEngine Reference Manual	<i>297</i>
Index	<i>375</i>

Contents

About the Authors	<i>ix</i>
Acknowledgments	<i>xi</i>
Introduction	<i>xiii</i>

Chapter 1 A Brief History of Compiler Technology.....1

The Mainstream: From Fortran to C	1
Basic Compilers	8
Summary	11
Challenge Exercise	12
Resources	12

Chapter 2 A Brief Introduction to the .NET Framework.....15

Code Reuse and DLLs	15
DLL Heaven and DLL Hell	17
.NET—Beyond DLLs	19
Inside a .NET Binary	24
Summary	25
Challenge Exercise	25
Resources	26

Chapter 3 A Compiler Flyover.....27

The Phases of a Compiler	27
Three Theories, and Keeping It Real	28
Lexical Analysis with Regular Expressions	29
Parsing and BNF	35
Interpreters and RPN	44
Summary	48
Challenge Exercise	48
Resources	49

Chapter 4 The Syntax for the QuickBasic Compiler.....	51
A Tool for Analyzing BNF	52
Analyzing and Coding BNF	54
Building the BNF for Our QuickBasic	68
Analyzing the BNF of Our QuickBasic	77
Eight Guidelines for Effective BNF	81
bnfAnalyzer Technical Notes	82
Summary	90
Challenge Exercise	90
Resources	90
 Chapter 5 The Lexical Analyzer for the QuickBasic Compiler.....	 91
The Compiler Big Picture	91
Lexical Analysis Theory	92
A Regular Expression Laboratory.....	101
The qbScanner Object	109
Scanner Object Design Considerations	120
Summary	130
Challenge Exercise	131
Resources	131
 Chapter 6 QuickBasic Object Modeling.....	 133
The Abstract QuickBasic Variable Type Model	134
QuickBasic Variables Mapped to .NET Objects	148
The qbVariableType Object	152
The qbVariable Object	162
Under the CLR Lies the Beach!	167
Summary	168
Challenge Exercise	169
Resources	170
 Chapter 7 The Parser and Code Generator for the QuickBasic Compiler.....	 171
The Recursive-Descent Algorithms	172
The qbPolish Object	181
Code Optimization	183

The Architecture of the Compiler	187
The Dynamic Big Picture	189
Error Taxonomy	202
Summary	202
Challenge Exercise	203
Resources	203
 Chapter 8 Developing Assemblers and Interpreters.....	 205
Assemblers	205
Interpreters	218
Summary	228
 Chapter 9 Code Generation to the Common Language Runtime.....	 229
CLR Generation in the quickBasicEngine.....	230
Towards a Complete Object Code Generator	242
Challenge Exercise	242
Summary	242
 Chapter 10 Implementing Business Rules.....	 243
Business Rule Solutions	243
Logic As Data	246
Case Study: Credit Granting	247
Summary	264
Challenge Exercise	265
Resources	265
 Chapter 11 Language Design: Some Notes.....	 267
Determining Your Goals	267
Deciding on the Semantics of Your Language	270
Deciding on the Syntax of Your Language	279
Documenting Your Language	283
Summary	283
Challenge Exercise	284
Conclusion	284

Appendix A quickBasicEngine Language Manual	287
Lexical Syntax	288
Keywords and System Functions	289
Parser Syntax (Backus-Naur Form)	290
Built-In Functions	294
 Appendix B quickBasicEngine Reference Manual.....	 297
Class Standards	298
qbOp	300
qbPolish	302
qbScanner	305
qbToken	316
qbTokenType	320
qbVariable	321
qbVariableType	339
quickBasicEngine	359
 Index.....	 375

About the Author

Edward G. Nilges has programmed since 1970, when he learned machine language for an 8KB IBM 1401 as part of an elaborate draft-dodging scheme that appears to have gotten out of hand.

Early on, Edward discovered the power of languages and their translation in “ordinary” management information systems (MIS) applications. He consolidated several applications into one by creating a specifications language within the 1401’s constraints, and he also provided his university with a working Fortran compiler. Some of his early adventures are relevant to today’s challenges, and this book contains some of Edward’s unexpurgated war stories.

Edward has developed millions of lines of code for MIS, telecommunications, naval architecture, and education applications. He has developed several compilers, including internal compilers for telecommunications applications at Nortel, the QuickBasic compiler of this book, and a compiler for the Mouse language that fits in 1KB of storage. He has taught at Roosevelt University in Chicago and DeVry University, and delivered training classes at Princeton University.

At Princeton, Edward was honored to assist the real-life protagonist of the recent film *A Beautiful Mind*, John Nash, with a bug in the old Microsoft C compiler. Edward was also privileged to meet Cornel West, the noted American philosopher, and Ralph Nader. He took classes in philosophy and computer science, and gained access to Firestone Library (and has since paid fines accrued).

Currently, Edward is working in China on methodologies for transferring client/server applications to the Web, while also studying written and spoken Chinese.

Edward has two grown children and a former wife who he honors as the mother of those children. Indeed, he calls himself Edward G. Nilges to disambiguate himself from his eldest son, Edward A. Nilges, who is studying philosophy at the University of Illinois and has contributed errata to Bjarne Stroustrup’s book, *The C++ Programming Language*. His other son, Peter “Chauncey” “Zeit-Bug” Nilges, recently graduated *cum laude* from DePaul.

Edward has published material on computer and general topics since 1976, when he suggested in *Computerworld* that it was possible to write structured code in assembly language, and got yelled at by Ed Yourdon. Recent articles include utilities for string conversion and display in Visual Studio, and a critical assessment of the language we use in speaking about database theory, published in the Austrian journal *Labyrinths*.

Current interests include .NET, art, running, reading, China, and world philosophy.

Acknowledgments

THIS BOOK IS DUE to a suggestion of David Treadwell of Princeton and Microsoft, because he suggested a list of potential languages for .NET implementation, including QuickBasic. Initial impetus was provided by Josef Finsel, author of *The Handbook for Reluctant Database Administrators*, and I am in Mr. Finsel's debt for this reason.

Dan Appleman's support and patience during the excitements of its development is most appreciated, as is that of Marilyn Smith, Beth Christmas, Grace Wong, Nicole LeClerc, Kari Brooks, Bill Johncocks, Kurt Krames, and Kelly Winkquist, as well as the accounting team at Apress.

Dan Appleman in fact volunteered his valuable time for a developmental edit and put up with some of my deeper nonsense with a great deal of patience.

I need to thank the gang at the Evanston YMCA, as well as the operators of various executive stay places around the world for providing working space at various times, for my day jobs have taken me to the far corners of the world.

The Silicon Valley "out-to-lunch bunch," including Rick, Ragu, William, Bill, and Jason, are also owed a debt of gratitude for their assistance, including Rick's wireless card, Ragu's thoughtfulness, William's unfailing kindness and Jason's lap-top, which is toast, I'm afraid.

Helmut Epp of DePaul University, Max Plager of Roosevelt University, the late E. D. Klemke of Roosevelt University and Iowa State University, and Gilbert Harman of Princeton University are all academics from whom I have learned a prior dedication to the truth of the matter.

Long-suffering managers to whom this book is dedicated include Rita Saltz, Robert Geiger (a Visual Basic authority in his own right from whom I learned much), Jeff Burtenshaw, and Monsieur Hugh Levaux.

Tim Tyler was and remains a source of spiritual guidance before, during, and after the writing of this book.

Lee Thé at Fawcette assisted with an earlier release of part of the software in an article on Visual Studio and was a most patient and learned editor.

My strange friend Alex, "Sasha Alexandrovich" Gaydasch, is also owed a debt of gratitude for his support and advice over a period of many years.

Of course, we all owe Edsger Wybe Dijkstra a debt for showing how integrity goes a long way.

But the main dedication of this effort is to Darlene Nilges, Eddie Nilges, and Peter Nilges (jungle Peter), for in dreams begin responsibilities.

Introduction

I mean, if 10 years from now, when you are doing something quick and dirty, you suddenly visualize that I am looking over your shoulder and say to yourself, "Dijkstra would not have liked this," well, that would be enough immortality for me.

—Edsger Wybe Dijkstra

Let us not speak falsely now, the hour is much too late.

—Bob Dylan, “All Along the Watchtower”

DIJKSTRA DIDN'T PLAY for the Chicago Cubs baseball club, to my knowledge, nor did he play for the Arsenal, Chelsea, Antwerp or Eindhoven football organizations (nor did Dylan, but you knew that). Instead, Edsger Wybe Dijkstra was a founding computer scientist who was involved in the early Algol language and either invented or reported the invention of structured programming.

And since Dijkstra passed on in August 2002, he is rolling in his grave. Here is a book on how to write a halfway decent compiler, using object-oriented techniques (about which Dijkstra was skeptical) to compile Basic (which he felt was a mental mutilation) that has the gall, the side, the cheek ... to quote the guy!

That is because Dijkstra was also one of the few computing scientists to keep steadfast in his mind the true proposition that computing science is applied science, and that is because Dijkstra refused to divorce theory and practice.

Furthermore, I cannot believe that Dijkstra would dislike the desire to know how compilers work. I have set myself the task of communicating this, at a basic level, to a wide audience of “ordinary” (ordinary?) programmers.

These are the numerous hard-working programmers who have written code, probably, for Visual Basic and C++ COM and now are working, probably, in C# and in Visual Basic .NET. I would like to show that a responsible compiler can be written in Visual Basic. I would like to provide the complete, runnable, and modifiable, source code at the Apress Web site. I would also like world peace and harmony, but I digress.

Why not C#? I didn't choose C# because of a simple theory of mine. All, or nearly all, C# programmers know Visual Basic, but not all Visual Basic programmers have made the transition to C#. And despite the flash and glamour of C#, there is nothing doable in C# that cannot be done in Visual Basic.

My goal is to demystify and to deconstruct a skill set that can be of actual use in .NET and Java. Write-once, run anywhere is a goal that entails a need for more compilers, and more generally, greater portability and ease of modification, not only of code, but also of business rules, stored as data.

To set the scene, Chapter 1 provides an overview of the history of compiler technology, and Chapter 2 describes the .NET background.

Chapter 3 is a fun and exciting chapter, if I do say so myself. That chapter builds a simple “flyover” compiler to demonstrate the key concepts in a lightweight form. This will prepare you for subsequent chapters, which build a compiler of more than 10,000 lines of code for a significant part of the old QuickBASIC language, a Microsoft forerunner of Visual Basic.

In Chapter 4, you will learn about the indispensable formal notation Backus-Naur Form for designing the syntax of a language in the context of the `bnfAnalyzer` tool, for which, as is the case with all software in this book, source code is provided.

Then, in Chapter 5, you’ll learn the basic “lexical” level of parsing, in which we recognize the elements of the language.

Chapter 6 shows a complete, object-oriented approach to storing variables and their types in the Basic language. This is extensible to other languages. In fact, it uses parsing to design an internal language for “serializing” data types and values, thereby showing one way in which this information can be stored.

Chapter 7 is the high point, for in it, I will show you how to build the actual parsing front end of the compiler, where the source code is translated to an interpreted language.

Chapter 8 shows how any computer (given enough time and memory, of course) can simulate any other computer, based on an important early result of hero computer scientist Alan Turing. You’ll see how we can use this result to test compiled Basic code while visually displaying, step by step, its execution.

Chapter 9 introduces techniques for taking this process one step further, and translating compiled code to Microsoft Intermediate Language (MSIL) for execution in the Common Language Runtime (CLR). This translation makes the formerly slow, interpreted code run much faster.

One concern of mine is how compilation techniques can make life easier for the end users who are charged with keeping business rules up-to-date in credit, banking, law, and other applications. Compiler technology can be used to represent these rules as ASCII data suitable for storage in a database. Chapter 9 shows how this could be accomplished for a small loan company, Loans for the Honest Poor, which uses flexible rules to find good risks among “ordinary” (ordinary?) working people.

Chapter 10 winds up by discussing some of the issues that can arise in designing a language, including political and business issues.

Appendix A provides a comprehensive and geeky reference manual for the language of the representative QuickBasic compiler, and Appendix B provides another reference manual for the objects used to build the compiler.

The comprehensive source for all software and tools is available to purchasers of this book at the Downloads area of the Apress Web site (<http://www.apress.com>), along with Apress forums in which you can meet authors, ask questions, and, within reason, schmooze and socialize. You will need Visual Basic 2003 Professional or Enterprise to run all of the code.

I am also available at spinoza1111@yahoo.com should you have any questions.

Thanks for buying my book, or, at least, taking it down from the shelf at Borders, Barnes and Noble, Page One (at Festival Walk in Kowloon Tong), or wherever. If your bookstore has a café, grab a latte, sit down, take a load off fanny, take a load for free, and peruse. I hope you will get excited at the prospect of actually being able to walk through the basics of compiler design theory while running practical examples on your .NET system.

That's because, if you are like me, and indeed like the mighty Dijkstra, you don't divorce theory and practice. Dijkstra's pronouncements have a common theme, and they are that he never thought himself above getting down into the actual code—whether program code or the equally demanding codes of formal, if applied, mathematics.

Programmers get irritated, on the one hand, by academics who rattle on about arcane proofs and theorems with no apparent connection to the real world of schedules and deadlines (as well as sick children and cars that don't start), and on the other hand, by managers who sketch out vast ambitions that must then be wearily constructed in reality, by programmers with little say in the end product (for let us not speak falsely now, the hour is much too late).

But in honor of the late, hero computer scientist Dijkstra, and indeed for the dear old schoolhouses (in my case, St. Viator High School, Roosevelt University, DePaul University, and Princeton as a sort of midlife crisis, for both me and dear old Nassau), I will now attempt to steer a course between the rocks of academia and management-o-rama, and sail true toward your mastery of something new that will make you feel like a devil of a fellow or a hell of a gal, at a minimum. For, the boss, in today's Big Chill, groans at us on the job to only learn the essentials, and ultimately, this is stunting. Let's learn something almost (but not quite) for its own sake.

Let's dweeb out.