

# Beginning C

Fifth Edition



Ivor Horton

Apress®

## Beginning C, Fifth Edition

Copyright © 2013 by Ivor Horton

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN 978-1-4302-4881-1

ISBN 978-1-4302-4882-8 (eBook)

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

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

President and Publisher: Paul Manning

Lead Editor: Jonathan Gennick

Technical Reviewer: Marc Gregoire

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel,

Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham,

Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft,

Gwenan Spearing, Matt Wade, Tom Welsh

Coordinating Editor: Jill Balzano

Copy Editor: Mary Bearden

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail [orders-ny@springer-sbm.com](mailto:orders-ny@springer-sbm.com), or visit [www.springeronline.com](http://www.springeronline.com). Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail [rights@apress.com](mailto:rights@apress.com), or visit [www.apress.com](http://www.apress.com).

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales-eBook Licensing web page at [www.apress.com/bulk-sales](http://www.apress.com/bulk-sales).

Any source code or other supplementary materials referenced by the author in this text is available to readers at [www.apress.com](http://www.apress.com). For detailed information about how to locate your book's source code, go to [www.apress.com/source-code/](http://www.apress.com/source-code/).

*For my daughter, Dany.*

# Contents at a Glance

<b>About the Author .....</b>	<b>xxi</b>
<b>About the Technical Reviewer .....</b>	<b>xxiii</b>
<b>Acknowledgments .....</b>	<b>xxv</b>
<b>Introduction .....</b>	<b>xxvii</b>
<b>■ Chapter 1: Programming in C .....</b>	<b>1</b>
<b>■ Chapter 2: First Steps in Programming .....</b>	<b>25</b>
<b>■ Chapter 3: Making Decisions .....</b>	<b>85</b>
<b>■ Chapter 4: Loops .....</b>	<b>135</b>
<b>■ Chapter 5: Arrays .....</b>	<b>185</b>
<b>■ Chapter 6: Applications with Strings and Text .....</b>	<b>219</b>
<b>■ Chapter 7: Pointers .....</b>	<b>263</b>
<b>■ Chapter 8: Structuring Your Programs .....</b>	<b>321</b>
<b>■ Chapter 9: More on Functions .....</b>	<b>349</b>
<b>■ Chapter 10: Essential Input and Output .....</b>	<b>395</b>
<b>■ Chapter 11: Structuring Data .....</b>	<b>429</b>
<b>■ Chapter 12: Working with Files .....</b>	<b>489</b>
<b>■ Chapter 13: The Preprocessor and Debugging .....</b>	<b>557</b>
<b>■ Chapter 14: Advanced and Specialized Topics .....</b>	<b>589</b>

■ **Appendix A: Computer Arithmetic .....621**

■ **Appendix B: ASCII Character Code Definitions .....629**

■ **Appendix C: Reserved Words in C.....635**

■ **Appendix D: Input and Output Format Specifications.....637**

■ **Appendix E: Standard Library Header Files .....643**

**Index .....645**

# Contents

<b>About the Author .....</b>	<b>xxi</b>
<b>About the Technical Reviewer .....</b>	<b>xxiii</b>
<b>Acknowledgments .....</b>	<b>xxv</b>
<b>Introduction .....</b>	<b>xxvii</b>
<b>■ Chapter 1: Programming in C .....</b>	<b>1</b>
The C Language.....	1
The Standard Library .....	2
Learning C .....	2
Creating C Programs .....	2
Editing.....	2
Compiling.....	3
Linking.....	3
Executing .....	4
Creating Your First Program .....	6
Editing Your First Program.....	7
Dealing with Errors.....	7
Dissecting a Simple Program .....	8
Comments .....	9
Preprocessing Directives.....	10
Defining the main() Function .....	10
Keywords.....	11
The Body of a Function .....	11
Outputting Information .....	13

Function Arguments .....	13
Control Characters.....	14
Trigraph Sequences.....	16
The Preprocessor .....	16
Developing Programs in C.....	17
Understanding the Problem .....	17
Detailed Design .....	17
Implementation.....	18
Testing .....	18
Functions and Modular Programming .....	18
Common Mistakes.....	22
Points to Remember.....	22
Summary.....	23
<b>■ Chapter 2: First Steps in Programming .....</b>	<b>25</b>
Memory in Your Computer.....	25
What Is a Variable?.....	27
Naming Variables.....	28
Variables That Store Integers.....	28
Using Variables .....	33
Initializing Variables.....	35
Basic Arithmetic Operations.....	35
More on Division with Integers .....	40
Unary Operators.....	41
Unary Minus Operator.....	41
Variables and Memory.....	42
Signed Integer Types .....	42
Unsigned Integer Types .....	43
Specifying Integer Constants.....	43

<b>Working with Floating-Point Numbers .....</b>	<b>45</b>
Floating-Point Number Representation .....	46
Floating-Point Variables.....	47
Division Using Floating-Point Values .....	48
Controlling the Number of Decimal Places in the Output .....	49
Controlling the Output Field Width.....	50
<b>More Complicated Expressions .....</b>	<b>50</b>
<b>Defining Named Constants.....</b>	<b>53</b>
<b>Knowing Your Limitations.....</b>	<b>56</b>
<b>Introducing the sizeof Operator.....</b>	<b>59</b>
<b>Choosing the Correct Type for the Job .....</b>	<b>60</b>
<b>Explicit Type Conversion.....</b>	<b>63</b>
Automatic Conversions.....	64
Rules for Implicit Conversions .....	64
Implicit Conversions in Assignment Statements .....	65
<b>More Numeric Data Types .....</b>	<b>66</b>
Character Type.....	66
Character Input and Character Output.....	67
Enumerations.....	71
Choosing Enumerator Values .....	72
Unnamed Enumeration Types .....	73
Variables That Store Boolean Values .....	73
<b>The op= Form of Assignment.....</b>	<b>74</b>
<b>Mathematical Functions.....</b>	<b>75</b>
<b>Designing a Program .....</b>	<b>77</b>
The Problem .....	77
The Analysis.....	77
The Solution.....	79
<b>Summary.....</b>	<b>83</b>



<b>■ Chapter 3: Making Decisions.....</b>	<b>85</b>
The Decision-Making Process.....	85
Arithmetic Comparisons .....	85
The Basic if Statement .....	86
Extending the if statement: if-else .....	90
Using Blocks of Code in if Statements.....	93
Nested if Statements .....	94
Testing Characters.....	97
Logical Operators .....	100
The Conditional Operator .....	104
Operator Precedence: Who Goes First? .....	107
Multiple-Choice Questions .....	111
Using else-if Statements for Multiple Choices .....	112
The switch Statement.....	112
The goto Statement .....	121
Bitwise Operators.....	122
The op= Use of Bitwise Operators.....	125
Using Bitwise Operators .....	125
Designing a Program.....	128
The Problem .....	129
The Analysis.....	129
The Solution.....	129
Summary.....	133
<b>■ Chapter 4: Loops.....</b>	<b>135</b>
How Loops Work.....	135
Introducing the Increment and Decrement Operators.....	136
The for Loop .....	137
General Form of the for Loop .....	141

<b>More on the Increment and Decrement Operators.....</b>	<b>142</b>
The Increment Operator.....	142
The Prefix and Postfix Forms of the Increment Operator .....	142
The Decrement Operator .....	143
<b>The for Loop Revisited.....</b>	<b>144</b>
Modifying the for Loop Control Variable .....	147
A for Loop with No Parameters.....	147
The break Statement in a Loop.....	148
Limiting Input Using a for Loop.....	150
Generating Pseudo-Random Integers.....	153
More for Loop Control Options.....	155
Floating-Point Loop Control Variables.....	156
<b>The while Loop .....</b>	<b>156</b>
<b>Nested Loops.....</b>	<b>159</b>
<b>Nested Loops and the goto Statement.....</b>	<b>165</b>
<b>The do-while Loop.....</b>	<b>166</b>
<b>The continue Statement .....</b>	<b>169</b>
<b>Designing a Program.....</b>	<b>169</b>
The Problem .....	169
The Analysis.....	169
The Solution.....	170
<b>Summary.....</b>	<b>182</b>
<b>■ Chapter 5: Arrays .....</b>	<b>185</b>
<b>An Introduction to Arrays.....</b>	<b>185</b>
Programming Without Arrays.....	186
What Is an Array? .....	187
Using an Array .....	188
<b>The Address of Operator.....</b>	<b>192</b>
<b>Arrays and Addresses.....</b>	<b>194</b>
<b>Initializing an Array.....</b>	<b>195</b>

Finding the Size of an Array .....	196
Multidimensional Arrays.....	197
Initializing Multidimensional Arrays .....	199
Variable Length Arrays .....	205
Designing a Program.....	208
The Problem .....	208
The Analysis.....	208
The Solution.....	210
Summary.....	216
<b>■ Chapter 6: Applications with Strings and Text .....</b>	<b>219</b>
What Is a String? .....	219
Variables That Store Strings.....	221
Arrays of Strings.....	224
Operations with Strings.....	227
Checking for C11 Support.....	227
Finding the Length of a String .....	228
Copying Strings .....	229
Concatenating Strings .....	229
Comparing Strings.....	233
Searching a String.....	237
Tokenizing a String .....	242
Reading Newline Characters into a String.....	247
Analyzing and Transforming Strings.....	248
Converting Character Case .....	250
Converting Strings to Numerical Values .....	253
Designing a Program.....	255
The Problem .....	255
The Analysis.....	255
The Solution.....	256
Summary.....	261

<b>■ Chapter 7: Pointers .....</b>	<b>263</b>
<b>A First Look at Pointers .....</b>	<b>263</b>
Declaring Pointers .....	264
Accessing a Value Through a Pointer .....	265
Using Pointers .....	269
Testing for a NULL Pointer .....	273
Pointers to Constants .....	273
Constant Pointers .....	274
Naming Pointers .....	275
<b>Arrays and Pointers .....</b>	<b>275</b>
<b>Multidimensional Arrays .....</b>	<b>279</b>
Multidimensional Arrays and Pointers .....	283
Accessing Array Elements .....	284
<b>Using Memory As You Go .....</b>	<b>287</b>
Dynamic Memory Allocation: The malloc() Function .....	288
Releasing Dynamically Allocated Memory .....	289
Memory Allocation with the calloc() Function .....	294
Extending Dynamically Allocated Memory .....	294
<b>Handling Strings Using Pointers .....</b>	<b>298</b>
Using Arrays of Pointers .....	299
Pointers and Array Notation .....	305
<b>Designing a Program .....</b>	<b>310</b>
The Problem .....	310
The Analysis .....	311
The Solution .....	311
The Complete Program .....	316
<b>Summary .....</b>	<b>319</b>
<b>■ Chapter 8: Structuring Your Programs .....</b>	<b>321</b>
<b>Program Structure .....</b>	<b>321</b>
Variable Scope and Lifetime .....	322
Variable Scope and Functions .....	325

<b>Functions.....</b>	<b>326</b>
Defining a Function.....	326
The return Statement .....	330
<b>The Pass-By-Value Mechanism.....</b>	<b>334</b>
<b>Function Prototypes .....</b>	<b>335</b>
<b>Pointers as Parameters and Return Types .....</b>	<b>337</b>
const Parameters .....	337
Perils of Returning Pointers.....	343
<b>Summary.....</b>	<b>346</b>
<b>■ Chapter 9: More on Functions .....</b>	<b>349</b>
<b>Pointers to Functions .....</b>	<b>349</b>
Declaring a Pointer to a Function .....	349
Calling a Function Through a Function Pointer.....	350
Arrays of Pointers to Functions .....	353
Pointers to Functions As Arguments.....	355
<b>Variables in Functions .....</b>	<b>358</b>
Static Variables: Keeping Track Within a Function.....	359
Sharing Variables Between Functions .....	361
<b>Functions That Call Themselves: Recursion .....</b>	<b>363</b>
<b>Functions with a Variable Number of Arguments .....</b>	<b>366</b>
Copying a va_list .....	369
Basic Rules for Variable-Length Argument Lists .....	369
<b>The main() Function .....</b>	<b>370</b>
<b>Terminating a Program.....</b>	<b>371</b>
The abort() Function .....	372
The exit() and atexit() Functions .....	372
The _Exit() Function.....	372
The quick_exit() and at_quick_exit() Functions.....	373

Enhancing Performance .....	373
Declaring Functions Inline .....	373
Using the restrict Keyword .....	374
The _Noreturn Function Specifier.....	374
Designing a Program.....	374
The Problem .....	375
The Analysis.....	376
The Solution.....	377
Summary.....	392
<b>■ Chapter 10: Essential Input and Output.....</b>	<b>395</b>
Input and Output Streams .....	395
Standard Streams .....	396
Input from the Keyboard.....	396
Formatted Keyboard Input.....	397
Input Format Control Strings .....	397
Characters in the Input Format String .....	403
Variations on Floating-Point Input .....	404
Reading Hexadecimal and Octal Values .....	406
Reading Characters Using scanf_s() .....	408
String Input from the Keyboard .....	410
Single Character Keyboard Input.....	411
Output to the Screen .....	416
Formatted Output Using printf_s().....	416
Escape Sequences .....	419
Integer Output.....	419
Outputting Floating-Point Values .....	422
Character Output .....	423

<b>Other Output Functions .....</b>	<b>425</b>
Unformatted Output to the Screen.....	425
Formatted Output to an Array .....	426
Formatted Input from an Array .....	426
<b>Summary.....</b>	<b>427</b>
<b>■ Chapter 11: Structuring Data.....</b>	<b>429</b>
<b>Data Structures: Using struct .....</b>	<b>429</b>
Defining Structure Types and Structure Variables .....	431
Accessing Structure Members .....	432
Unnamed Structures.....	435
Arrays of Structures.....	435
Structure Members in Expressions.....	438
Pointers to Structures.....	439
Dynamic Memory Allocation for Structures .....	439
<b>More on Structure Members .....</b>	<b>442</b>
Structures As Members of a Structure .....	442
Declaring a Structure Within a Structure.....	444
Pointers to Structures As Structure Members .....	444
Doubly Linked Lists .....	449
Bit Fields in a Structure .....	453
<b>Structures and Functions .....</b>	<b>454</b>
Structures As Arguments to Functions .....	454
Pointers to Structures As Function Arguments .....	455
Structure As a Function Return Value .....	456
Binary Trees .....	461
<b>Sharing Memory .....</b>	<b>470</b>
<b>Designing a Program.....</b>	<b>474</b>
The Problem .....	474
The Analysis.....	474
The Solution.....	475
<b>Summary.....</b>	<b>487</b>

<b>■ Chapter 12: Working with Files .....</b>	<b>489</b>
The Concept of a File.....	489
Positions in a File .....	490
File Streams.....	490
Accessing Files .....	490
Opening a File.....	491
Buffering File Operations.....	493
Renaming a File.....	494
Closing a File .....	495
Deleting a File.....	496
Writing a Text File .....	496
Reading a Text File .....	497
Reading and Writing Strings to a Text File.....	501
Formatted File Input and Output .....	505
Formatted Output to a File.....	506
Formatted Input from a File .....	506
Dealing with Errors.....	509
More Open Modes for Text Files .....	510
The <code>freopen_s()</code> Function .....	511
Binary File Input and Output.....	511
Opening a File in Binary Mode.....	512
Writing a Binary File .....	513
Reading a Binary File.....	513
Moving Around in a File.....	520
File Positioning Operations.....	520
Finding Out Where You Are .....	520
Setting a Position in a File .....	521
Using Temporary Work Files .....	528
Creating a Temporary Work File.....	528
Creating a Unique File Name .....	529



<b>Updating Binary Files .....</b>	<b>530</b>
Changing the Contents of a File.....	535
Creating a Record from Keyboard Input .....	537
Writing a Record to a File .....	538
Reading a Record from a File .....	539
Writing a File .....	539
Listing the File Contents .....	540
Updating the Existing File Contents.....	541
<b>File Open Modes Summary .....</b>	<b>548</b>
<b>Designing a Program.....</b>	<b>549</b>
The Problem .....	549
The Analysis.....	549
The Solution.....	549
<b>Summary .....</b>	<b>555</b>
<b>■ Chapter 13: The Preprocessor and Debugging .....</b>	<b>557</b>
<b>Preprocessing .....</b>	<b>557</b>
Including Header Files .....	557
Defining Your Own Header Files .....	558
Managing Multiple Source Files .....	558
External Variables .....	559
Static Functions.....	559
Substitutions in Your Program Source Code .....	560
<b>Macros .....</b>	<b>561</b>
Macros That Look Like Functions .....	561
Strings As Macro Arguments .....	563
Joining Two Arguments in a Macro Expansion .....	565
<b>Preprocessor Directives on Multiple Lines .....</b>	<b>565</b>
Logical Preprocessor Directives .....	565
Conditional Compilation.....	566
Testing for Multiple Conditions .....	567
Undefining Identifiers .....	567

Testing for Specific Values for Identifiers .....	567
Multiple-Choice Selections .....	568
Standard Preprocessing Macros .....	568
<b>Debugging Methods .....</b>	<b>569</b>
Integrated Debuggers .....	570
The Preprocessor in Debugging .....	570
Assertions .....	575
<b>Date and Time Functions .....</b>	<b>577</b>
Getting Time Values .....	577
Getting the Date .....	581
Getting the Day for a Date .....	585
<b>Summary .....</b>	<b>588</b>
<b>■ Chapter 14: Advanced and Specialized Topics .....</b>	<b>589</b>
<b>Working with International Character Sets .....</b>	<b>589</b>
Understanding Unicode .....	589
Setting the Locale .....	590
The Wide Character Type <code>wchar_t</code> .....	591
Operations on Wide Character Strings .....	594
File Stream Operations with Wide Characters .....	598
Fixed Size Types That Store Unicode Characters .....	598
<b>Specialized Integer Types for Portability .....</b>	<b>602</b>
Fixed Width Integer Types .....	602
Minimum Width Integer Types .....	603
Maximum Width Integer Types .....	603
<b>The Complex Number Types .....</b>	<b>603</b>
Complex Number Basics .....	604
Complex Types and Operations .....	605
<b>Programming with Threads .....</b>	<b>607</b>
Creating a Thread .....	608
Exiting a Thread .....	609
Joining One Thread to Another .....	609

Suspending a Thread .....	613
Managing Thread Access to Data .....	613
Summary .....	620
■ <b>Appendix A: Computer Arithmetic</b> .....	<b>621</b>
Binary Numbers .....	621
Hexadecimal Numbers .....	622
Negative Binary Numbers .....	624
Big-Endian and Little-Endian Systems .....	625
Floating-Point Numbers .....	626
■ <b>Appendix B: ASCII Character Code Definitions</b> .....	<b>629</b>
■ <b>Appendix C: Reserved Words in C</b> .....	<b>635</b>
■ <b>Appendix D: Input and Output Format Specifications</b> .....	<b>637</b>
Output Format Specifications .....	637
Input Format Specifications .....	639
■ <b>Appendix E: Standard Library Header Files</b> .....	<b>643</b>
<b>Index</b> .....	<b>645</b>

# About the Author



**Ivor Horton** graduated as a mathematician and was lured into information technology by promises of great rewards for very little work. In spite of the reality usually being a great deal of work for relatively modest rewards, he has continued to work with computers to the present day. He has been engaged at various times in programming, systems design, consultancy, and the management and implementation of projects of considerable complexity. Ivor has many years of experience in the design and implementation of computer systems applied to engineering design and manufacturing operations in a variety of industries. He has also spent a lot of time developing occasionally useful applications in a wide variety of programming languages and primarily teaching scientists and engineers to do likewise. He has been writing books on programming for many years, and his currently published works include tutorials on C, C++, and Java. At the present time, when he is not writing programming books or providing advice to others, he spends his time fishing, traveling, and enjoying life in general.

# About the Technical Reviewer



**Marc Gregoire** is a software engineer from Belgium. He graduated from the Catholic University of Leuven, Belgium, with a degree in “Burgerlijk ingenieur in de computer wetenschappen” (equivalent to master of science in engineering in computer science). The year after, he received the cum laude degree of master in artificial intelligence at the same university. After his studies, Marc started working for a software consultancy company called Ordina Belgium. As a consultant, he worked for Siemens and Nokia Siemens Networks on critical 2G and 3G software running on Solaris for telecom operators. This required working in international teams spanning from South America and the United States to Europe, the Middle East, Africa, and Asia. Now, Marc is working for Nikon Metrology on 3D laser scanning software.

His main expertise is C/C++, specifically Microsoft VC++ and the MFC framework. Next to C/C++, Marc also likes C# and uses PHP for creating web pages. In addition to his main interest of Windows development, he also has experience in developing C++ programs running 24/7 on Linux platforms (e.g., EIB home automation software).

Since April 2007, he has received the yearly Microsoft MVP (Most Valuable Professional) award for his Visual C++ expertise.

Marc is the founder of the Belgian C++ Users Group ([www.becpp.org](http://www.becpp.org)) and an active member on the CodeGuru forum (as Marc G). He also creates freeware and shareware programs that are distributed through his web site at [www.nuonsoft.com](http://www.nuonsoft.com), and maintains a blog on [www.nuonsoft.com/blog/](http://www.nuonsoft.com/blog/).

# Acknowledgments

---

The author is only one member of the large team of people necessary to get a book into print. I thank the entire Apress editorial and production teams for their help and support throughout. I would like to acknowledge the efforts of Jonathan Gennick, who initiated this new edition, and Jill Balzano, who has patiently dealt with my questions and problems in the editing process.

I would also like to thank my technical editor, Marc Gregoire, for doing such a fantastic job of reviewing the text and checking out all the code fragments and examples. He has an uncanny knack for finding my errors, and his many constructive comments and thoughtful suggestions have undoubtedly made the book a much better tutorial.

# Introduction

Welcome to *Beginning C: Fifth Edition*. With this book you can become a competent C programmer using the latest version of the C language. In many ways, C is an ideal language with which to learn programming. It's very compact, so there isn't a lot of syntax to learn before you can write real applications. In spite of its conciseness, it's extremely powerful and is used by professionals in many different areas. The power of C is such that it can be applied at all levels, from developing device drivers and operating system components to creating large-scale applications. A relatively new area for C is in application development for mobile phones.

C compilers are available for virtually every kind of computer, so when you've learned C, you'll be equipped to program in just about any context. Once you know C, you have an excellent base from which you can build an understanding of the object-oriented C++.

My objective in this book is to minimize what I think are the three main hurdles the aspiring programmer must face: coming to grips with the jargon that pervades every programming language, understanding how to use the language elements (as opposed to merely knowing what they are), and appreciating how the language is applied in a practical context.

Jargon is an invaluable and virtually indispensable means of communication for the expert professional as well as the competent amateur, so it can't be avoided. My approach is to ensure that you understand the jargon and get comfortable using it in context. In this way, you'll be able to more effectively use the documentation that comes along with the typical programming product and also feel comfortable reading and learning from the literature that surrounds most programming languages.

Comprehending the syntax and effects of the language elements is obviously an essential part of learning C, but appreciating how the language features work and how they are used is equally important. Rather than just using code fragments, I provide you with practical working examples in each chapter that show how the language features can be applied to specific problems. These examples provide a basis for you to experiment and see the effects of changing the code.

Your understanding of programming in context needs to go beyond the mechanics of applying individual language elements. To help you gain this understanding, I conclude most chapters with a more complex program that applies what you've learned in the chapter. These programs will help you gain the competence and confidence to develop your own applications and provide you with insight into how you can apply language elements in combination and on a larger scale. Most important, they'll give you an idea of what's involved in designing real programs and managing real code.

It's important to realize a few things that are true for learning any programming language. First, there is quite a lot to learn, but this means you'll gain a greater sense of satisfaction when you've mastered it. Second, it's great fun, so you really will enjoy it. Third, you can only learn programming by doing it, and this book helps you along the way. Finally, it's certain you will make a lot of mistakes and get frustrated from time to time during the learning process. When you think you are completely stuck, you just need to be persistent. You will eventually experience that eureka moment and realize it wasn't as difficult as you thought.

## How to Use This Book

Because I believe in the hands-on approach, you'll write your first programs almost immediately. Every chapter has several complete programs that put theory into practice, and these are key to the book. You should type in and run all the examples that appear in the text because the very act of typing them in is a tremendous memory aid. You should also attempt all the exercises that appear at the end of each chapter. When you get a program to work for the first time—particularly when you're trying to solve your own problems—you'll find that the great sense of accomplishment and progress makes it all worthwhile.

The pace is gentle at the start, but you'll gain momentum as you get further into the subject. Each chapter covers quite a lot of ground, so take your time and make sure you understand everything before moving on. Experimenting with the code and trying out your own ideas are important parts of the learning process. Try modifying the programs and see what else you can make them do—that's when it gets really interesting. And don't be afraid to try things out—if you don't understand how something works, just type in a few variations and see what happens. It doesn't matter if it's wrong. You'll find you often learn a lot from getting it wrong. A good approach is to read each chapter through, get an idea of its scope, and then go back and work through all the examples.

You might find some of the end-of-chapter programs quite difficult. Don't worry if it's not all completely clear on the first try. There are bound to be bits that you find hard to understand at first because they often apply what you've learned to rather complicated problems. If you really get stuck, you can skip the end-of-chapter exercises, move on to the next chapter, and come back to them later. You can even go through the entire book without worrying about them. However, if you can complete the exercises, it shows you are making real progress.

## Who This Book Is For

*Beginning C: Fifth Edition* is designed to teach you how to write useful programs in C as quickly and easily as possible. By the end of *Beginning C*, you'll have a thorough grounding in programming the C language. This is a tutorial for those who've done a little bit of programming before, understand the concepts behind it, and want to further your knowledge by learning C. However, no previous programming knowledge on your part is assumed, so if you're a newcomer to programming, the book will still work for you.

## What You Need to Use This Book

To use this book, you'll need a computer with a C compiler and library installed, so you can execute the examples, and a program text editor for preparing your source code files. The compiler you use should provide good support for the current International Standard for the C language, ISO/IEC 9899:2011, commonly referred to as C11. You'll also need an editor for creating and modifying your code. You can use any plain text editor such as Notepad or vi to create your source program files. However, you'll get along better if your editor is designed for editing C code.

I can suggest two sources for a suitable C compiler, both of which are freeware:

- The GNU C compiler, GCC, is available from <http://www.gnu.org> and supports a variety of operating system environments.
- The Pelles C compiler for Microsoft Windows is downloadable from <http://www.smorgasbordet.com/pellesc/> and includes an excellent IDE.



## Conventions Used

I use a number of different styles of text and layout in the book to help differentiate between the different kinds of information. For the most part, their meanings will be obvious. Program code will appear like this:

```
int main(void)
{   printf("Beginning C\n");
    return 0;
}
```

When a code fragment is a modified version of a previous instance, I occasionally show the lines that have changed in bold type like this:

```
int main(void)
{
    printf("Beginning C by Ivor Horton\n");
    return 0;
}
```

When code appears in the text, it has a different typestyle that looks like this: double.

I'll use different types of “brackets” in the program code. They aren't interchangeable, and their differences are very important. I'll refer to the symbols ( ) as parentheses, the symbols { } as braces, and the symbols [ ] as square brackets.

Important new words in the text are shown in italic *like this*.