

LEARNING MADE EASY



2nd Edition

C Programming

for
dummies[®]
A Wiley Brand



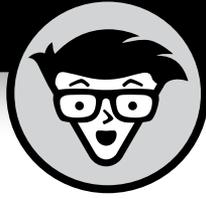
Discover C language and
start coding in no time

Build your knowledge to create
advanced projects and programs

Kickstart your new career
as a C programmer

Dan Gookin

C programming author
and online trainer



C Programming

2nd Edition

by Dan Gookin

for
dummies[®]
A Wiley Brand

C Programming For Dummies®, 2nd Edition

Published by: **John Wiley & Sons, Inc.**, 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2021 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit <https://hub.wiley.com/community/support/dummies>.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2020945155

ISBN: 978-1-119-74024-7; 978-1-119-74025-4 (ebk); 978-1-119-74026-1 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Introduction	1
Part 1: The ABs of C	7
CHAPTER 1: A Quick Start for the Impatient.	9
CHAPTER 2: The Programming Thing	21
CHAPTER 3: Anatomy of C	29
Part 2: C Programming 101	43
CHAPTER 4: Trials and Errors	45
CHAPTER 5: Values and Simple Math	57
CHAPTER 6: A Place to Put Stuff	67
CHAPTER 7: Input and Output	83
CHAPTER 8: Decision Making	97
CHAPTER 9: Loops, Loops, Loops	115
CHAPTER 10: Fun with Functions	133
Part 3: Build Upon What You Know	151
CHAPTER 11: The Unavoidable Math Chapter	153
CHAPTER 12: Give Me Arrays	173
CHAPTER 13: Fun with Text	193
CHAPTER 14: Structures, the Multivariable.	211
CHAPTER 15: Life at the Command Prompt	221
CHAPTER 16: Variable Nonsense	233
CHAPTER 17: Binary Mania	249
Part 4: The Advanced Part	267
CHAPTER 18: Introduction to Pointers.	269
CHAPTER 19: Deep into Pointer Land	285
CHAPTER 20: Memory Chunks and Linked Lists	305
CHAPTER 21: It's About Time	329
Part 5: And the Rest of It	337
CHAPTER 22: Permanent Storage Functions	339
CHAPTER 23: File Management	359
CHAPTER 24: Beyond Mere Mortal Projects	369
CHAPTER 25: Out, Bugs!	381

Part 6: The Part of Tens	393
CHAPTER 26: Ten Common Boo-Boos	395
CHAPTER 27: Ten Reminders and Suggestions	403
Part 7: Appendices	411
APPENDIX A: ASCII Codes	413
APPENDIX B: Keywords	419
APPENDIX C: Operators	421
APPENDIX D: Data Types	423
APPENDIX E: Escape Sequences	425
APPENDIX F: Conversion Characters	427
APPENDIX G: Order of Precedence	429
Index	431

Table of Contents

INTRODUCTION	1
Why the C Language?	1
The C Programming For Dummies Approach	2
How This Book Works	3
Icons Used in This Book	4
Parting Thoughts	5
PART 1: THE ABs OF C	7
CHAPTER 1: A Quick Start for the Impatient	9
What You Need to Program	9
Command Prompt Programming	10
IDE Programming	11
Installing Code::Blocks	12
Touring the Code::Blocks workspace	13
Your First Program	14
Coding at the command prompt	15
Building a new Code::Blocks project	16
Building and running	18
CHAPTER 2: The Programming Thing	21
The History of Programming	21
Reviewing early programming history	21
Introducing the C language	22
The Programming Process	23
Understanding programming	23
Writing source code	24
Compiling and linking	26
Running and testing	27
CHAPTER 3: Anatomy of C	29
Parts of the C Language	29
Keywords	30
Functions	31
Operators	33
Variables and values	33
Statements and structure	33
Comments	35

Behold the Typical C Program	37
Understanding C program structure	37
Setting the main() function.	38
Returning something to the operating system.	39
Adding a function	40
PART 2: C PROGRAMMING 101	43
CHAPTER 4: Trials and Errors	45
Display Stuff on the Screen	45
Displaying a humorous message	45
Introducing the <i>puts()</i> function	46
Adding more text.	47
Commenting out a statement	49
Goofing up on purpose.	49
More Text Output Nonsense	51
Displaying text with <i>printf()</i>	52
Introducing the <i>printf()</i> function	52
Understanding the newline	53
Employing escape sequences	54
Goofing up on purpose again	55
CHAPTER 5: Values and Simple Math	57
A Venue for Various Values	57
Understanding values.	58
Displaying values with <i>printf()</i>	59
Minding the extra zeros	61
The Computer Does the Math.	61
Doing simple arithmetic	61
Reviewing the float-integer thing	63
Pretending integers are floats.	64
CHAPTER 6: A Place to Put Stuff	67
Values That Vary	67
Setting up a quick example	68
Introducing data types	69
Using variables.	70
Variable Madness!	73
Using more-specific data types	73
Working with several variables	75
Assigning a value upon creation	77
Reusing variables.	77
Constants Always the Same.	79
Using the same value over and over	79
Constants in your code.	80
Putting constants to use.	81

CHAPTER 7: Input and Output	83
Character I/O	83
Understanding input and output devices	83
Fetching characters with <i>getchar()</i>	84
Using the <i>putchar()</i> function	86
Working with character variables	87
Text I/O, but Mostly I.	88
Storing strings	89
Introducing the <i>scanf()</i> function	90
Reading a string with <i>scanf()</i>	91
Reading values with <i>scanf()</i>	93
Using <i>fgets()</i> for text input	94
CHAPTER 8: Decision Making	97
What If?	97
Making a simple comparison	97
Introducing the <i>if</i> keyword	99
Comparing values in various ways	100
Knowing the difference between = and ==	102
Forgetting where to put the semicolon	103
Multiple Decisions	104
Making more-complex decisions	104
Adding a third option	105
Multiple Comparisons with Logic	106
Building a logical comparison	106
Adding some logical operators	107
The Old <i>Switch Case</i> Trick	108
Making a multiple-choice selection	108
Understanding the <i>switch-case</i> structure	110
Taking no breaks	111
The Weird ?: Decision Thing	112
CHAPTER 9: Loops, Loops, Loops	115
A Little Déjà Vu	115
The Thrill of <i>for</i> Loops	116
Doing something x number of times	116
Introducing the <i>for</i> loop	117
Counting with the <i>for</i> statement	119
Looping letters	120
Nesting <i>for</i> loops	121
The Joy of the <i>while</i> Loop	123
Structuring a <i>while</i> loop	123
Using the <i>do while</i> loop	125

Loopy Stuff	126
Looping endlessly	126
Looping endlessly but on purpose	127
Breaking out of a loop	128
Adding multiple <i>for</i> loop conditions	129
Screwing up a loop	130
CHAPTER 10: Fun with Functions	133
Anatomy of a Function	133
Constructing a function	133
Prototyping (or not)	136
Functions and Variables	139
Using variables in functions	139
Sending a value to a function	140
Sending multiple values to a function	142
Creating functions that return values	143
Returning early	145
Constants of the Global Kind	147
Introducing defined constants	147
Putting defined constants to use	148
PART 3: BUILD UPON WHAT YOU KNOW	151
CHAPTER 11: The Unavoidable Math Chapter	153
Math Operators from Beyond Infinity	153
Incrementing and decrementing	154
Prefixing the ++ and -- operators	156
Discovering the remainder (modulus)	158
Saving time with assignment operators	158
Math Function Mania	160
Exploring some common math functions	161
Suffering through trigonometry	163
It's Totally Random	166
Spewing random numbers	166
Making the numbers more random	167
The Holy Order of Precedence	170
Getting the order correct	170
Forcing order with parentheses	171
CHAPTER 12: Give Me Arrays	173
Behold the Array	173
Avoiding arrays	173
Understanding arrays	174
Initializing an array	177

Playing with character arrays (strings)	177
Working with empty <i>char</i> arrays	179
Sorting arrays	181
Multidimensional Arrays	183
Making a two-dimensional array	183
Going crazy with three-dimensional arrays	186
Declaring an initialized multidimensional array	188
Arrays and Functions	189
Passing an array to a function	189
Returning an array from a function	191
CHAPTER 13: Fun with Text	193
Character Manipulation Functions	193
Introducing the CTYPEs	193
Testing characters	195
Changing characters	197
String Functions Galore	198
Reviewing string functions	198
Comparing text	199
Building strings	200
Fun with <i>printf()</i> Formatting	202
Formatting floating point	202
Setting the output width	204
Aligning output	206
Gently Down the Stream	207
Demonstrating stream input	207
Dealing with stream input	208
CHAPTER 14: Structures, the Multivariable	211
Hello, Structure	211
Introducing the multivariable	211
Understanding <i>struct</i>	213
Filling a structure	215
Making an array of structures	216
Weird Structure Concepts	218
Putting structures within structures	218
Passing a structure to a function	219
CHAPTER 15: Life at the Command Prompt	221
Conjure a Terminal Window	221
Starting a terminal window	222
Running code in text mode	223
Arguments for the <i>main()</i> Function	225
Reading the command line	225
Understanding <i>main()</i> 's arguments	227

Time to Bail	229
Quitting the program	229
Running another program	230
CHAPTER 16: Variable Nonsense	233
Variable Control	233
Typecasting into disbelief	233
Creating new things with <i>typedef</i>	235
Making <i>static</i> variables	238
Variables, Variables Everywhere	241
Using external variables	241
Creating an external structure variable	243
Enumerating	245
CHAPTER 17: Binary Mania	249
Binary Basics	249
Understanding binary	249
Outputting binary values	251
Bit Manipulation	253
Using the bitwise <code> </code> operator	253
Using bitwise <code>&</code>	256
Operating exclusively with XOR	257
Understanding the <code>~</code> and <code>!</code> operators	259
Shifting binary values	259
Explaining the <code>binbin()</code> function	263
The Joy of Hex	264
PART 4: THE ADVANCED PART	267
CHAPTER 18: Introduction to Pointers	269
The Biggest Problem with Pointers	269
Sizing Up Variable Storage	270
Understanding variable storage	270
Reading a variable's size	271
Checking a variable's location	275
Reviewing variable storage info	278
The Hideously Complex Topic of Pointers	279
Introducing the pointer	279
Working with pointers	282
CHAPTER 19: Deep into Pointer Land	285
Pointers and Arrays	285
Getting the address of an array	285
Working pointer math in an array	287
Substituting pointers for array notation	293

Strings Are Pointer-Things	294
Using pointers to display a string	294
Using a pointer to declare a string	295
Building an array of pointers	296
Sorting strings	300
Pointers in Functions	302
Passing a pointer to a function	302
Returning a pointer from a function	303
CHAPTER 20: Memory Chunks and Linked Lists	305
Give Me Memory!	306
Introducing the malloc() function	306
Creating string storage	308
Using the calloc() function	309
Getting more memory	311
Freeing memory	313
Lists That Link	314
Allocating space for a structure	314
Creating a linked list	316
Editing a linked list	323
Saving a linked list	328
CHAPTER 21: It's About Time	329
What Time Is It?	329
Understanding the calendar	330
Working with time in C	330
Time to Program	331
Checking the clock	331
Viewing a timestamp	333
Slicing through the time string	334
Snoozing	336
PART 5: AND THE REST OF IT	337
CHAPTER 22: Permanent Storage Functions	339
Sequential File Access	339
Understanding C file access	340
Writing text to a file	341
Reading text from a file	342
Appending text to a file	345
Writing binary data	346
Reading binary data	348

Random File Access	350
Writing a structure to a file	351
Reading and rewinding	353
Finding a specific record	355
Saving a linked list to a file	357
CHAPTER 23: File Management	359
Directory Madness	359
Calling up a directory	359
Gathering more file info	361
Separating files from directories	363
Exploring the directory tree	364
Fun with Files	365
Renaming a file	365
Copying a file	367
Deleting a file	368
CHAPTER 24: Beyond Mere Mortal Projects	369
The Multi-Module Monster	369
Linking two source code files	370
Sharing variables between modules	372
Creating a custom header file	374
Other Libraries to Link	378
CHAPTER 25: Out, Bugs!	381
Simple Tricks to Resolve Problems	381
Documenting the flow	382
Talking through your code	382
Writing comments for future-you	382
The Debugger	383
Debugging setup	383
Working the debugger	385
Setting a breakpoint	387
Watching variables	388
Improved Error Messages	390
PART 6: THE PART OF TENS	393
CHAPTER 26: Ten Common Boo-Boos	395
Conditional Foul-Ups	395
== v. =	396
Dangerous Loop Semicolons	397
Commas in <i>for</i> Loops	398
Missing break in a switch Structure	398

Missing Parentheses and Curly Brackets	399
Don't Ignore a Warning	399
Endless Loops	400
<i>scanf()</i> Blunders	401
Streaming Input Restrictions	402
CHAPTER 27: Ten Reminders and Suggestions	403
Maintain Good Posture	404
Use Creative Names	404
Write a Function	405
Work on Your Code a Little Bit at a Time	405
Break Apart Larger Projects into Several Modules	406
Know What a Pointer Is	406
Add Whitespace before Condensing	407
Know When <i>if-else</i> Becomes <i>switch-case</i>	407
Remember Assignment Operators	408
When You Get Stuck, Read Your Code Out Loud	409
PART 7: APPENDICES	411
APPENDIX A: ASCII Codes	413
APPENDIX B: Keywords	419
APPENDIX C: Operators	421
APPENDIX D: Data Types	423
APPENDIX E: Escape Sequences	425
APPENDIX F: Conversion Characters	427
APPENDIX G: Order of Precedence	429
INDEX	431

Introduction

When I was in school, I'd open a new math textbook and look in the back, marveling at the problems. Someday, I thought, I would understand all this nonsense.

You should do that with this book right now: Open it up to one of the final chapters. Look over the C programming code and think to yourself, "Someday soon, this will all make perfect sense to me."

Say "Hello, world" to *C Programming For Dummies*, 2nd Edition, the book that sets you on the path to become a computer programmer. Once despised vermin, banished to basement server rooms and suffering from a lack of personal hygiene, programmers are now valued and contributing members of society. Some are billionaires. And they all started their careers by learning to program.

The C language lets you master a number of electronic gizmos. You can craft your own programs, dictating your every whim and desire to computers, tablets, and cell phones. The electronics dutifully obey. Given the information offered in this book, you can pass that programming class, impress your friends, be admired by Hollywood, or even start your own software company. Truly, learning to program is a worthy investment of your time.

This book helps make learning how to program understandable and enjoyable. You don't need any programming experience — you don't even need to buy new software. You just need the desire to program in C and the ability to have fun while doing so.

Why the C Language?

An argument surfaces every few years that learning C is a road to nowhere. Newer, better programming languages exist, and it's far better to learn them than to waste time learning C.

Poppycock.

C continues to dominate the charts for best and most useful programming languages, often beating out the newer languages the cool programmers use. Further, C is like the Latin of computer languages: Just about every Johnny-come-lately programming language uses C syntax. C keywords and even certain functions find their way into other popular languages, from C++ to Java to Python to whatever the latest, trendy language might be.

My point is that once you learn C programming, all those other programming languages come easy. In fact, many of the books that teach those other languages often assume that you know a little C before you start out. This assumption is frustrating for a beginner — but not when you already know C.

So ignore the lofty pundits and know-it-all poohbahs. C is still relevant. Programming for microcontrollers, operating systems, and major software packages is still done using good ol' C. You are not wasting your time.

The C Programming For Dummies Approach

As a programmer, I've toiled through many programming books. I know what I really don't like to see, and, lamentably, I see it often — that is, where the author writes pages-long code or boasts about what he knows, impressing his fellow nerds and not really teaching anything. Too much of this type of training exists, which is probably why you've picked up this book.

My approach here is simple: Short programs. To-the-point demonstrations. Lots of examples. Plenty of exercises.

The best way to learn something is by doing it. Each concept presented in this book is coupled with sample code. The listings are short enough that you can quickly type them in — and I recommend that you do so. You can then build and run the code to see how things work. This immediate feedback is not only gratifying, it's also a marvelous learning tool.

Sample programs are followed by exercises you can try on your own, testing your skills and expanding your knowledge. Suggested answers to the exercises and all the source code can be found on this book's companion website:

```
https://c-for-dummies.com/cprog
```

How This Book Works

This book teaches the C programming language. It starts out by assuming that you know little to nothing about programming, and it finishes by covering some of the more advanced C operations.

To program in C, you need a computer. This book makes no assumptions about the computer you select: It can be a Windows PC, a Macintosh, or a Linux system. You can choose to use an integrated development environment (IDE) such as Code::Blocks, or you can compile and run the sample programs at a command prompt.

This book also wastes no time, getting you started immediately in Chapter 1. Nothing is introduced without a full explanation first. Due to the nature of programming, I've made a few exceptions; they're carefully noted in the text. Otherwise, the book flows from front to back, which is how best to read this book.

C language keywords and functions are shown in italic text, as in *printf()* and *break*. Some keywords, such as *for* and *if*, may make the sentence read in a goofy way, which is why those words are shown in italic.

Filenames and variable names are shown in monospace type, such as `program.exe` and `loop`.

If you need to type something, that text is shown in bold. For example, "Type the **blorfus** command" means that you should type *blorfus* at the keyboard. You're directed when to press the Enter key, if at all.

When working numbered steps, text to type appears in regular (roman) type:

3. Type *exit* and press the Enter key.

You type the word *exit* and then press the Enter key.

Program samples are shown as snippets on the page, similar to this one:

```
if( i == 1)
    printf("eye won");
```

You don't need to type an example unless you're directed to do so.

Full program listings are shown and numbered in each chapter; for example:

LISTING 1-1: The Code::Blocks Skeleton

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return(0);
}
```

Because of this book's margins, text in a listing may occasionally wrap, extending from one line to the next. You do not need to split up your code in a similar manner, and I remind you whenever such a thing occurs.

The listings in this book don't contain line numbers, but your text editor might. This book references the sample code listings by using the line numbers, which you can also use in your editor to examine the code.

Exercises are numbered by chapter and then sequentially. So the third exercise in Chapter 13 is Exercise 13-3. You're directed in the text to work an exercise. Here's an example:

Exercise 1-1: Type the source code from Listing 1-1 into your editor. Save it under the filename `ex0101.c`. Build and run.

Answers for all exercises can be found on the web:

<https://c-for-dummies.com/cprog>

Go to this web page if you want to copy-and-paste the source code as well.

Icons Used in This Book



REMEMBER

This icon flags information worthy enough to remember. Though I recommend remembering as much as you can, these icons flag the stuff you just can't forget.



TIP

A tip is a suggestion, a special trick, or something super fancy to help you out.



WARNING

This icon marks something you need to avoid. It's advice that could also be flagged with a Tip or Remember icon but has dire consequences if ignored.



TECHNICAL
STUFF

Face it: All of programming is technical. I reserve the use of this icon for extra-technical tidbits, asides, and anecdotes. Call it “nerd stuff.”

Parting Thoughts

I enjoy programming. It's a hobby, and I find it incredibly relaxing, frustrating, and rewarding. I assume that you share these feelings, though you may also be a struggling student or someone who wants a career. Regardless, *enjoy* programming. If you can imagine the program you want to write on a screen, you can make it happen. It may not happen as fast as you like, but it can happen.

Please work the exercises in this book. Try some on your own, variations on a theme. Continue working at problems until you solve them. The amazing thing about programming is that no single, absolutely correct way to do something exists. Anytime you try, you're learning.

If possible, find a programming friend who can help you. Don't make them do the work or explain how things run, but rely on them as a resource. Programming can be a solo thing, but it's good to occasionally commiserate with others who also program in C — or in any language.

This book has a few companion websites. The primary one is found here:

```
https://c-for-dummies.com/cprog
```

You can also check out my C programming blog, which is updated every Saturday with new lessons and offers a monthly Exercise challenge:

```
https://c-for-dummies.com/blog
```

The publisher also features a companion website, which I'm obliged to mention here, though it's not updated as frequently as my own site. Visit www.dummies.com and type **C programming** into the search box to find this book's support page and other goodies.

For more help, or just to say hi, you can send me email at

`dan@c-for-dummies.com`

I'm happy to hear from you, though I won't write code for you. I also cannot explain university assignments. (I don't do B-trees. No one does.) And if you have any questions specific to this book — especially any errors or typos — feel free to pass them along.

Enjoy your C programming!

1

The ABs of C

IN THIS PART . . .

Get started with C coding

Work through your very first program

Learn how programming works

Discover the parts of the C language

Craft a basic C skeleton

IN THIS CHAPTER

- » Reviewing software requirements
- » Programming at the command prompt
- » Using an IDE
- » Creating a command prompt program
- » Working in Code::Blocks
- » Compiling a program

Chapter 1

A Quick Start for the Impatient

You're most likely eager to get started programming in C. I shan't waste your time.



TIP

If you already have a compiler or an IDE set up and are ready to go, skip to Chapter 2.

What You Need to Program

The two most important things you need to begin your programming adventure are

- » A computer
- » Access to the Internet

The computer is your primary tool for writing and compiling code. Yes, even if you're writing a game for the Xbox, you need a computer to be able to code. The computer can be a PC or a Macintosh. The PC can run Windows or Linux.

Internet access is necessary to obtain the programming software. You need a text editor to write the code and a compiler to translate the code into a program. The compiler generally comes with other tools you need, such as a linker and a debugger. All these tools are found at no cost on the Internet.

The software tools offer two approaches to programming: command line and IDE.

If you want to learn C programming as I did back in the dark ages, you use a terminal window and traditional command-line tools: editor, compiler, and linker. The process is fast, but complicated because you're using text mode commands. Still, it offers a spiritual connection with those who built the foundations upon which the computer industry roosts.

The most common way to craft code, however, is to obtain an integrated development environment — called an *IDE* by the cool kids. It combines all the tools you need for programming into one compact, terrifying, and intimidating unit.

Don't freak! The terms *compiler*, *linker*, *debugger*, and *terrifying* are all defined in Chapter 2.

Command Prompt Programming

To re-create the environment where the C language was born, use a Unix or Linux terminal window running a shell program such as *bash*. This environment is available to all major computing platforms, and the programming tools used are reliable and well-documented. Programming at the command prompt earns you a nerd merit badge and the admiration of your peers.

For Windows 10, open the Microsoft Store app and install Ubuntu, a free Linux shell. Ensure that you follow the directions to install the Windows Subsystem for Linux, which is an extra step you'll probably miss.

For Linux, you're ready to go: Open a terminal window to access the shell.

For Mac OS X, use the Terminal app. I also recommend obtaining the Homebrew package manager. Visit <https://brew.sh> for directions. Homebrew allows you to install programming tools not available to OS X.

For an editor, you can use any text mode editor available at the command prompt, such as *vi* or *emacs*. You can also “mix it up” and use a window-based editor. I’m fond of using the Windows version of the VIM editor while I simultaneously work at the command prompt in an Ubuntu terminal window.

A C compiler comes native to a Unix/Linux command prompt. The standard version is *cc* or *gcc*, but I recommend that you use the shell’s package manager to acquire the LLVM *clang* compiler. In Ubuntu Linux for Windows 10, type this command to install *clang*:

```
sudo apt-get install clang
```

Type your account password to initiate the process. To verify the installation, type

```
clang --version
```

Various Linux distros offer similar package managers, which you can use to obtain an editor and the *clang* compiler.



REMEMBER

- » The VIM editor can be obtained from vim.org.
- » Your choice to use the command prompt means you’re taking on an extra layer of complexity when it comes to programming. I find it fast and enjoyable, but if you believe it to be too much, especially when first learning the C language, rely instead upon an IDE, as covered in the next section.

IDE Programming

Plenty of programming IDEs are available for your C coding pleasure. On the Mac, use Xcode, which you can install from the App Store. For Windows and Linux, I recommend obtaining the Code::Blocks IDE, which is found at codeblocks.org. You can choose any other IDE you prefer, but Code::Blocks for Windows is fairly stable and comes with everything you need — providing that you install the correct version.

Installing Code::Blocks

The Code::Blocks website will doubtless be altered over time, so follow these general steps to install the IDE and confirm that the C compiler is accessible:

1. **On the main Code::Blocks website page, click the Downloads link.**
2. **Click the binary release link.**

The “binary release” means you’re installing a runnable program, not source code or something equally strange.

3. **Choose the proper installation program for your computer’s operating system.**

For Windows 10, I recommend that you choose the installation with the text *mingw-setup* appended. For example:

```
codeblocks-20.03mingw-setup.exe
```

The 20.03 part of the filename is the release number, which will change in the future. The *mingw-setup* choice means you’re downloading both the IDE and the MinGW compiler.

For Linux, click the link to install the proper version for your distro, but keep in mind that Code::Blocks might be more easily acquired by using the Linux GUI package/software manager.



TIP

4. **Open the downloaded archive to extract the Code::Blocks IDE installer.**

In Windows, you see a User Account Control warning when you open the archive. Click Yes to proceed with installation.

5. **Run the installation program.**

Perform a default installation; you need not customize anything.

6. **Choose to run Code:Blocks: Click the Yes button.**

Code::Blocks appears, showing its splash screen. Don’t start coding now. Instead, confirm the compiler’s installation:

7. **Choose Settings, Compiler.**

The Compiler Settings dialog box appears.

8. **With Global Compiler Settings chosen on the left, click the Toolchain Executables tab on the right side of the dialog box.**

9. **Ensure that the Compiler’s Installation Directory text box is filled.**

In a default confirmation, the following pathname is listed:

```
C:\Program Files (x86)\CodeBlocks\MinGW
```

If the text box is blank, use the Browse button (three dots to the right of the text box) to locate the MinGW installation directory.

10. Confirm that `gcc.exe` is set in the Compiler text box.

If not, click the Browse button (three dots) to locate the `gcc.exe` program, installed in the `MinGW\bin` directory by default.

11. Close the Compiler Settings dialog box; click OK.

Installation is complete. I recommend you close the Code::Blocks window. Finish the installation program as well.

Touring the Code::Blocks workspace

Figure 1-1 illustrates the Code::Blocks *workspace*, which is the official name of the massive mosaic of windows and toolbars you see arranged on the screen.

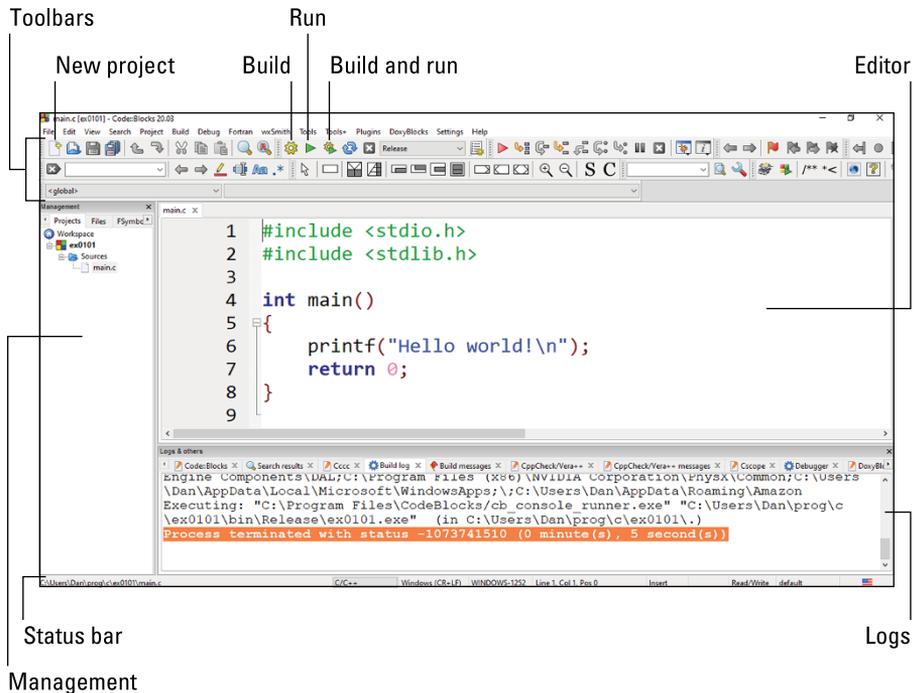


FIGURE 1-1:
The Code::Blocks workspace.

On your computer, as well as in Figure 1-1, locate the following parts of the workspace:

Toolbars: These messy strips, adorned with various command buttons, cling to the top of the Code::Blocks window. The toolbars can be rearranged or hidden, so don't mess with them until you get comfy with the interface.

Management: The pane on the left side of the workspace features four tabs, though you may not see all four at one time. The window provides a handy oversight of your programming endeavors.

Status bar: At the bottom of the screen, you see information about the project, editor, and other activities that take place in Code::Blocks.

Editor: The big window in the center-right area of the screen is where you type code.

Logs: The bottom of the screen features a window with many, many tabs displaying various tidbits about the programming process. The tab used most often is named Build Log.

The View menu controls the visibility of every item displayed in the window. Choose the pane name, such as Manager, from the View menu to show or hide that item. Control toolbars by using the View, Toolbars submenu.



TIP

- » Maximize the Code::Blocks program window so that it fills the screen. You need all that real estate.
- » In addition to color-coding your text, the Code::Blocks editor offers an autocomplete feature. Items that must be typed in pairs, such as quotes, parentheses, and so on, are generated automatically for you. Certain C language keywords and functions are presented automatically, along with hints for their arguments and options.
- » The editor features a tabbed interface, which lets you work on multiple source code files at one time.

Your First Program

The traditional first program written for any programming language is called Hello World. It's boring, like all traditions.