

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.

<p>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</p>
--

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)

C Programming For Dummies®

To view this book's Cheat Sheet, simply go to www.dummies.com and search for “C Programming For Dummies Cheat Sheet” in the Search box.

Table of Contents

[Cover](#)

[Title Page](#)

[Copyright Page](#)

[Introduction](#)

[Why the C Language?](#)

[The C Programming For Dummies Approach](#)

[How This Book Works](#)

[Icons Used in This Book](#)

[Parting Thoughts](#)

[**Part 1: The ABs of C**](#)

[**Chapter 1: A Quick Start for the Impatient**](#)

[What You Need to Program](#)

[Command Prompt Programming](#)

[IDE Programming](#)

[Your First Program](#)

Chapter 2: The Programming Thing

[The History of Programming](#)

[The Programming Process](#)

Chapter 3: Anatomy of C

[Parts of the C Language](#)

[Behold the Typical C Program](#)

Part 2: C Programming 101

Chapter 4: Trials and Errors

[Display Stuff on the Screen](#)

[More Text Output Nonsense](#)

Chapter 5: Values and Simple Math

[A Venue for Various Values](#)

[The Computer Does the Math](#)

Chapter 6: A Place to Put Stuff

[Values That Vary](#)

[Variable Madness!](#)

[Constants Always the Same](#)

Chapter 7: Input and Output

[Character I/O](#)

[Text I/O, but Mostly I](#)

Chapter 8: Decision Making

[What If?](#)

[Multiple Decisions](#)

[Multiple Comparisons with Logic](#)

[The Old *Switch Case* Trick](#)

[The Weird ?: Decision Thing](#)

Chapter 9: Loops, Loops, Loops

[A Little Déjà Vu](#)

[The Thrill of *for* Loops](#)

[The Joy of the *while* Loop](#)

[Loopy Stuff](#)

Chapter 10: Fun with Functions

[Anatomy of a Function](#)

[Functions and Variables](#)

[Constants of the Global Kind](#)

Part 3: Build Upon What You Know

Chapter 11: The Unavoidable Math Chapter

[Math Operators from Beyond Infinity](#)

[Math Function Mania](#)

[It's Totally Random](#)

[The Holy Order of Precedence](#)

Chapter 12: Give Me Arrays

[Behold the Array](#)

[Multidimensional Arrays](#)

[Arrays and Functions](#)

Chapter 13: Fun with Text

[Character Manipulation Functions](#)

[String Functions Galore](#)

[Fun with *printf\(\)* Formatting](#)

[Gently Down the Stream](#)

Chapter 14: Structures, the Multivariable

[Hello, Structure](#)

[Weird Structure Concepts](#)

Chapter 15: Life at the Command Prompt

[Conjure a Terminal Window](#)

[Arguments for the *main\(\)* Function](#)

[Time to Bail](#)

Chapter 16: Variable Nonsense

[Variable Control](#)

[Variables, Variables Everywhere](#)

Chapter 17: Binary Mania

[Binary Basics](#)

[Bit Manipulation](#)

[The Joy of Hex](#)

Part 4: The Advanced Part

Chapter 18: Introduction to Pointers

[The Biggest Problem with Pointers](#)

[Sizing Up Variable Storage](#)

[The Hideously Complex Topic of Pointers](#)

Chapter 19: Deep into Pointer Land

[Pointers and Arrays](#)

[Strings Are Pointer-Things](#)

[Pointers in Functions](#)

Chapter 20: Memory Chunks and Linked Lists

[Give Me Memory!](#)

[Lists That Link](#)

Chapter 21: It's About Time

[What Time Is It?](#)

[Time to Program](#)

Part 5: And the Rest of It

Chapter 22: Permanent Storage Functions

[Sequential File Access](#)

[Random File Access](#)

Chapter 23: File Management

[Directory Madness](#)

[Fun with Files](#)

Chapter 24: Beyond Mere Mortal Projects

[The Multi-Module Monster](#)

[Other Libraries to Link](#)

Chapter 25: Out, Bugs!

[Simple Tricks to Resolve Problems](#)

[The Debugger](#)

[Improved Error Messages](#)

Part 6: The Part of Tens

Chapter 26: Ten Common Boo-Boos

[Conditional Foul-Ups](#)

[== v. =](#)

[Dangerous Loop Semicolons](#)

[Commas in *for* Loops](#)

[Missing break in a switch Structure](#)

[Missing Parentheses and Curly Brackets](#)

[Don't Ignore a Warning](#)

[Endless Loops](#)

[*scanf\(\)* Blunders](#)

[Streaming Input Restrictions](#)

Chapter 27: Ten Reminders and Suggestions

[Maintain Good Posture](#)

[Use Creative Names](#)

[Write a Function](#)

[Work on Your Code a Little Bit at a Time](#)

[Break Apart Larger Projects into Several Modules](#)

[Know What a Pointer Is](#)

[Add Whitespace before Condensing](#)

[Know When *if-else* Becomes *switch-case*](#)

[Remember Assignment Operators](#)

[When You Get Stuck, Read Your Code Out Loud](#)

Part 7: Appendices

Appendix A: ASCII Codes

Appendix B: Keywords

Appendix C: Operators

Appendix D: Data Types

Appendix E: Escape Sequences

[Appendix F: Conversion Characters](#)

[Appendix G: Order of Precedence](#)

[Index](#)

[About the Author](#)

[Connect with Dummies](#)

[End User License Agreement](#)

List of Tables

Chapter 3

[TABLE 3-1 C Language Keywords](#)

Chapter 4

[TABLE 4-1: Escape Sequences](#)

Chapter 5

[TABLE 5-1: Basic Math Operators](#)

Chapter 6

[TABLE 6-1 Basic C Language Variable Types](#)

[TABLE 6-2 More C Language Data Types](#)

Chapter 8

[TABLE 8-1 C Language Comparison Operators](#)

[TABLE 8-2 Logical Comparison Operators](#)

Chapter 11

[TABLE 11-1 C Math Operators](#)

[TABLE 11-2 C Math Assignment Operators](#)

[TABLE 11-3 Common, Sane Math Functions](#)

Chapter 13

[TABLE 13-1 CTYPE Testing Functions](#)

[TABLE 13-2 CTYPE Manipulation Functions](#)

[TABLE 13-3 String Functions](#)

Chapter 17

[TABLE 17-1 Binary Groupings](#)

[TABLE 17-2 Powers of 2](#)

[TABLE 17-3 Binary Operators](#)

[TABLE 17-4 Hexadecimal Values](#)

Chapter 19

[TABLE 19-1 Pointers and Peekers In and Out of Parentheses](#)

[TABLE 19-2 Array Notation Replaced by Pointers](#)

[TABLE 19-3 Pointer Notation and Array Notation](#)

Chapter 22

[TABLE 22-1: Access Modes for the *fopen\(\)* Function](#)

Appendix G

[TABLE G-1 Standard Operator Precedence](#)

[TABLE G-2 Pointers and Precedence](#)

List of Illustrations

Chapter 1

[FIGURE 1-1: The Code::`Blocks` workspace.](#)

[FIGURE 1-2: Program output.](#)

Chapter 5

[FIGURE 5-1: Matching *printf\(\)* conversion characters and arguments.](#)

Chapter 11

[FIGURE 11-1: Degrees and radians.](#)

Chapter 12

[FIGURE 12-1: Storing strings in a two-dimensional array.](#)

Chapter 15

[FIGURE 15-1: Text mode in a terminal window.](#)

[FIGURE 15-2: Setting command-line arguments in Code::`Blocks`.](#)

Chapter 17

[FIGURE 17-1: Base 2 values in a byte.](#)

Chapter 18

[FIGURE 18-1: How a structure fits in memory.](#)

[FIGURE 18-2: Variable locations in memory.](#)

[FIGURE 18-3: Using a pointer to read values.](#)

Chapter 19

[FIGURE 19-1: Filling an array by using a pointer.](#)

[FIGURE 19-2: How the ** thing works.](#)

Chapter 20

[FIGURE 20-1: A linked list in memory.](#)

[FIGURE 20-2: Removing an item from a linked list.](#)

[FIGURE 20-3: Adding an item to a linked list.](#)

Chapter 24

[FIGURE 24-1: Two source code files in the Code::Blocks IDE.](#)

Chapter 25

[FIGURE 25-1: The Compiler toolbar.](#)

[FIGURE 25-2: A breakpoint in the code.](#)

[FIGURE 25-3: Monitoring variable values.](#)

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 monofont 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!

Part 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

Chapter 1

A Quick Start for the Impatient

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
-

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.

» The VIM editor can be obtained from vim.org.



» **REMEMBER** 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.



TIP

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.

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.

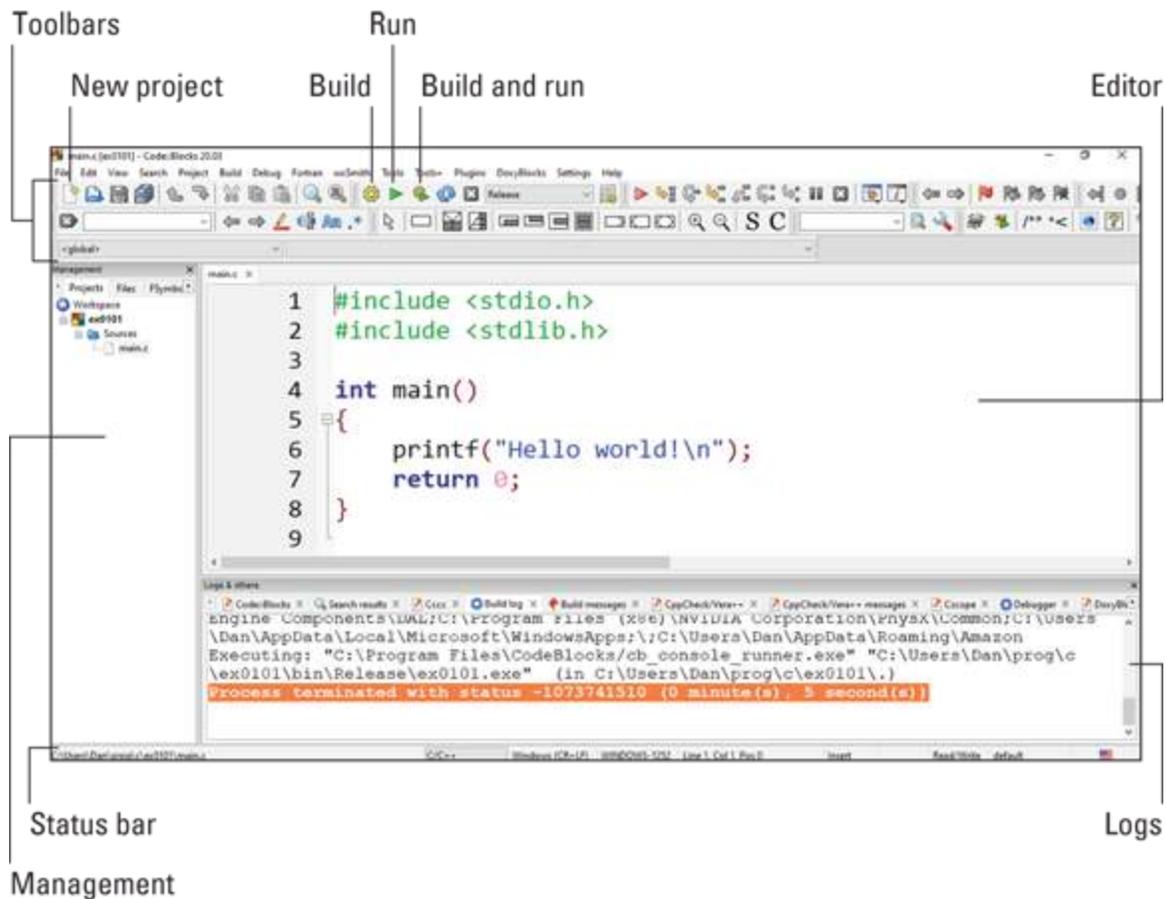


FIGURE 1-1: The Code::Blocks workspace.

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.