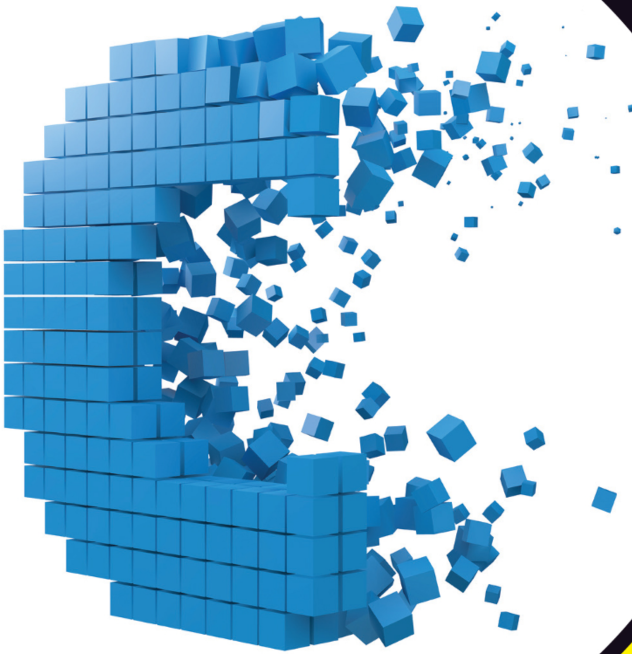
LEARNING MADE EASY

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

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

Table of Contents

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!

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.

   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.**

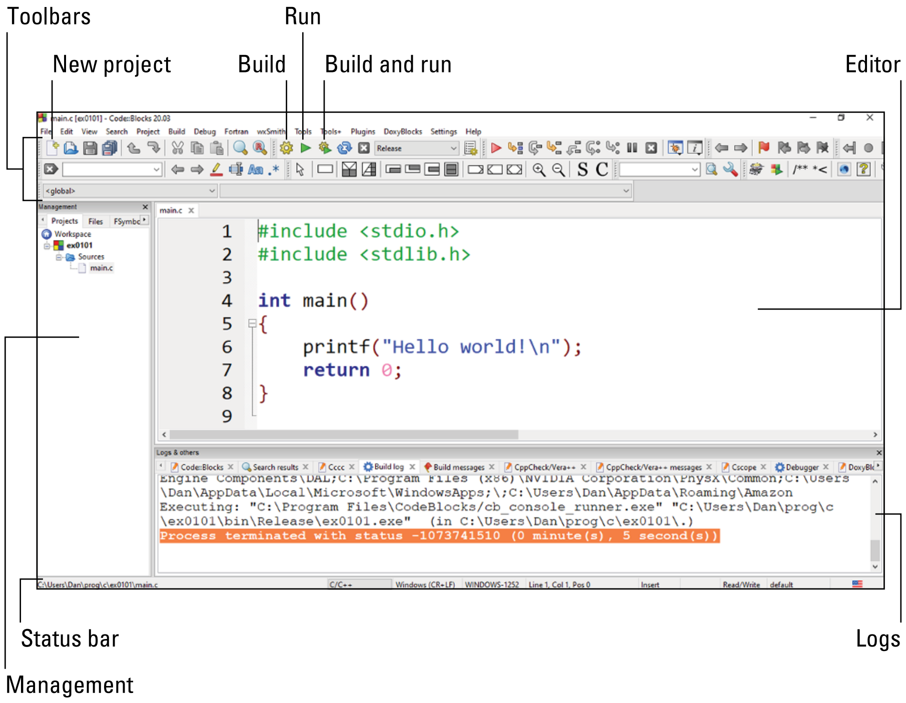
    If not, click the Browse button (three dots) to locate the `gcc.exe` program, installed in the `MinGW\bin` directory by default.

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.


Toolbars
Run
New project
Build
Build and run
Editor
Status bar
Logs
Management
main.c [ex0101] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Release
<global>
Management
Projects Files FSymbols
Workspace
ex0101
Sources
main.c
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks
C:\Users\Dan\prog\c\ex0101\main.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

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

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

```

```
Engine Components\DAL;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Users
\Dan\AppData\Local\Microsoft\WindowsApps;\;C:\Users\Dan\AppData\Roaming\Amazon
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\Dan\prog\c
\ex0101\bin\Release\ex0101.exe"  (in C:\Users\Dan\prog\c\ex0101\.)
Process terminated with status -1073741510 (0 minute(s), 5 second(s))
```

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.