A complete course on C programming for the beginner

Learn C on the Mac

For OS X and iOS

SECOND EDITION

David Mark | James Bucanek

Apress®

Learn C on the Mac

For OS X and iOS

David Mark

James Bucanek

Apress®

Learn C on the Mac: For OS X and iOS

Copyright © 2012 by David Mark and James Bucanek

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 978-1-4302-4533-9

ISBN 978-1-4302-4534-6 (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: Steve Anglin
Developmental Editor: James Markham
Technical Reviewer: Michael Thomas
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 Behr
Compositor: Bytheway Publishing Services
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, or visit 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, or visit 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.

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

To Deneen, Daniel, Kelley, and Ryan, ILYANMWITWWA.

To Deborah, for making time.

Contents at a Glance

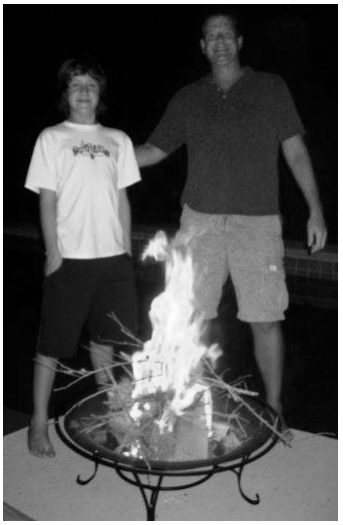
Contents

About the Authors

Dave Mark is a longtime Mac developer and author who has written a number of books on Mac and iOS development, including *Beginning iPhone 4 Development* (Apress, 2011), *More iPhone 3 Development* (Apress, 2010), *Learn C on the Mac* (Apress, 2008), *Ultimate Mac Programming* (Wiley, 1995), and the *Macintosh Programming Primer* series (Addison-Wesley, 1992). Dave was one of the founders of MartianCraft, an iOS and Android development house. Dave loves the water and spends as much time as possible on it, in it, or near it. He lives with his wife and three children in Virginia.

James Bucanek has spent the past 30 years programming and developing microprocessor systems. He has experience with a broad range of computer hardware and software, from embedded consumer products to industrial robotics. His development projects include the first local area network for the Apple II, distributed air conditioning control systems, a piano teaching system, digital oscilloscopes, silicon wafer deposition furnaces, and collaborative writing tools for K-12 education. James holds a Java Developer Certification from Sun Microsystems and was awarded a patent for optimizing local area networks. James is currently focused on OS X and iOS software development, where he can combine his deep knowledge of UNIX and object-oriented languages with his passion for elegant design. James holds an Associate's degree in classical ballet from the Royal Academy of Dance.

About the Technical Reviewer



Michael Thomas has worked in software development for over 20 years as an individual contributor, Team Lead, Program Manager, and Vice President of Engineering. Michael has over 10 years experience working with mobile devices. His current focus is in the medical sector using mobile devices to accelerate information transfer between patients and health care providers.

Acknowledgments

This book could not have been written without the support of our wonderful families. Deneen, Daniel, Kelley, Ryan, Deborah, Doug, and Amber, thank you all for everything you've done for us. We truly are lucky men.

Many, many thanks to the fine folks at Apress. Clay Andres started this ball rolling by bringing both Dave and James over to Apress. Steve Anglin is largely responsible for deciding what Apress prints, and we are flattered by his continued conviction in this book. James Markham kept a watchful eye on every paragraph, keeping our message clear and comprehensible. Michael Thomas checked every line of code and symbol to ensure complete accuracy. Any technical errors are ultimately our responsibility, but there are significantly fewer thanks to Michael. Mary Behr dotted our i's, crossed our t's, corrected our spelling, and made sure we used "whom" correctly. If you find this book easy to read, you have Mary's blue pencil to thank. Anna Ishchenko designed our beautiful cover. Last, but certainly not least, we are indebted to Coordinating Editor Jill Balzano who managed to juggle schedules, coordinate editors, track production, and herd two headstrong authors towards a common goal. To all the folks at Apress, thank you, thank you, thank you!

Dave says: A very special shout out goes to James, my incredibly talented co-author. James made many important technical contributions to this book, helping me scrub the prose and the sample code to ensure that it followed the C standard to the letter. He also added many concepts to the book that are vital to any aspiring programmer.

And from James: I am most grateful to David Mark for allowing me the opportunity to contribute to this venerable title. Dave has made learning C engaging and enjoyable for an entire generation of programmers. It's been an honor contributing to that institution. I would also like to extend thanks to Apple's Xcode development team for continually improving one of the finest software development tools in the world.

Introduction

Welcome Aboard

Welcome! Chances are that you are reading this because you love the Mac. And not only do you love the Mac, but you also love the idea of learning how to design and develop your very own Mac programs.

You've definitely come to the right place.

This book assumes that you know how to use your Mac. That's it. You don't need to know anything about programming—not one little bit. We'll start off with the basics, and each step we take will be a small one to make sure that you have no problem following along.

This book will focus on the basics of programming. At the same time, you'll learn the essentials of the C programming language.

In Douglas Adam's book The Hitchhiker's Guide to the Galaxy, the answer to "the Ultimate Question of Life, the Universe, and Everything" is determined to be "42." That answer is, of course, wrong; the correct answer is "C."

The C language is the wellspring of software. The nothing-short-of-miraculous revolution in computing and consumer electronics over the past half century has largely been accomplished using C, languages that are direct descendants of C (Objective-C, C++), or languages designed to work like C (Java, C#). Learn C and the programming world is your oyster.

■ **Note** Douglas Adams was a big Macintosh fan.

Once you get through this book, you'll be ready to move on to object-oriented programming and Objective-C—the official programming language of OS X and iOS.

Does this all sound a little overwhelming? Not to worry; in this book, we'll take small steps, so nobody gets lost. You can definitely do this!

Who Is This Book For?

When Dave wrote the very first edition of *Learn C on the Mac* back in 1991, he was writing with college students in mind. After all, college was where he really learned to program. It seems he was way off.

"My first clue that I had underestimated my audience was when I started getting e-mails from fifth graders who were making their way through the book. Fifth graders! And not just one but lots of nine-, ten-, and eleven-year-old kids were digging in and learning to program. Cool! And the best part of all was when these kids started sending me actual shipping products that they created. You can't imagine how proud I was and still am."

Dave was really on to something. Over the years, we've heard from soccer moms, hobbyists, even folks who were using the Mac for the very first time, all of whom made their way through *Learn C on the Mac* and came out the other end, proud, strong, and full of knowledge.

So what do you need to know to get started? Although learning C by just reading a book is possible, you'll get the most out of this book if you run each example program as you encounter it. To do this, you'll need a Mac running OS X (preferably version 10.6.8 or later) and an Internet connection. You'll need the Internet connection to download the free tools Apple has graciously provided for anyone interested in programming the Mac and to download the projects that go along with this book.

Again, if you know nothing about programming, don't worry. The first few chapters of this book will bring you up to speed. If you have some programming experience (or even a lot), you might want to skim the first few chapters, and then dig right into the C fundamentals that start in Chapter 3.

The Lay of the Land

Here's a quick tour of what's to come in this book.

- Chapter 1 shows you how to get the free software tools you'll use throughout this book.
- Chapter 2 explains some of the basics of how computer programs are built.
- Chapter 3 shows you how to embed a series of programming statements into a reusable function, something you can call again and again.
- Chapter 4 adds variables and operators into the mix, bringing the power of mathematical expressions into your programs.
- Chapter 5 teaches you how to watch your program execute, line-by-line, to see that it's doing the right thing, or fix it if it's not.
- Chapter 6 introduces the concept of flow control, using constructs like if, else, do, and while to control the direction your program takes.
- Chapter 7 covers pointers and parameters, two concepts that will add a dramatic new level of power to your programs.
- Chapter 8 moves beyond the simple data types used in the first half of the book, adding the ability to work with more complex numbers along with data types like arrays and text strings.

- Chapter 9 takes a break to show you how to deploy your finished program and use it from the command line.
- Chapter 10 dives even deeper into data and teaches you how to design your own custom data structures.
- Chapter 11 shows you how to save your program's data and read it back in again by introducing the concept of the data file.
- Chapter 12 gives you some techniques for dealing with errors, for when things go wrong.
- Chapter 13 covers a variety of advanced topics—typecasting, unions, recursion, sorting, collections, and much more.
- Finally, Chapter 14 wraps things up and points you to the next step on your journey.

Ready to get started? Let's go!

Chapter 1

Go Get the Tools!

If you want to build a house, you need a solid set of well-crafted tools. Building computer programs is no different. Programming requires a specialized set of *development tools*—basically, programs that make programs.

In the early days of C, you only needed a few, relatively simple tools. As computers have become more sophisticated, so has the universe of development tools. Today, it's not uncommon to employ dozens of programs to create even a "simple" application: editors, compilers, linkers, debuggers, emulators, profilers, analyzers, and more. Add to that list programs that help you find documentation, cross reference your code, record your development history, and, well, it's starting to look like a whole hardware store full of tools!

The good news is that Apple has come to your rescue. Just as Apple has used an elegant user interface to demystify their most sophisticated applications, they've done the same for software developers. (That's you!)

Installing Xcode

Apple's Xcode is a complete hardware store of software development tools, packaged and delivered as a single application. All you have to do is write your program and Xcode will—behind the scenes—direct the scores of individual development tools needed to turn your idea into reality. It would make the Wizard of Oz proud.

> **NOTE:** An application that organizes multiple development tools into a single workspace is called an *integrated development environment* (IDE). Xcode is an IDE.

And getting Xcode into your computer couldn't be easier. The entire Xcode development suite is available from the App Store.

Launch the App Store, go to the Developer Tools category (or just search for "Xcode"), and click to install Xcode, as shown in Figure 1-1. Don't worry if your screen looks a bit different than the figure. Apple is constantly updating Xcode, so there will probably be a new version of Xcode in the App Store by the time this book hits the shelves (or your screen).

Figure 1-1. *Installing Xcode from the App Store*

That's it! Sit back and wait for Xcode to download and install. And you're going to have to wait awhile, as it's a really big application. So amuse yourself with the rest of this chapter while it downloads. Switch to the Purchases view, at the top of the App Store window, if you want see how the download is progressing.

How much is that IDE in the Window?

Xcode has gone through various prices in the past. Apple really wants you to create great applications and has strived, for the most part, to make its developments tools freely available.

It used to be that Xcode was only available to *registered developers*. Becoming a registered developer usually costs money, so Xcode was "free" only in the sense that the prize inside a cereal box is "free."

For a while, Xcode was priced at $5. As of this writing, Xcode is free in the App Store. Hopefully, it will stay that way.

NOTE: If you're running an older version of OS X and don't have access to the App Store, you can still download an earlier version of Xcode—but we don't recommend it.

The first problem you're going to encounter is how to get your copy of Xcode. As of this writing, you must be a registered developer to obtain an older version of Xcode. Unfortunately, Apple no longer offers free developer registration—largely because Xcode is now available for free in the App Store—so you'll have to pay to register, and that can be expensive. If you *are* a registered developer or have access to Apple's University Program for higher education, you can log into `http://developer.apple.com/` and download the tools.

But your biggest problem is going to be the differences between the current Xcode and older versions. The code examples in this book will still work and make sense, but the commands, windows, features, and controls are all going to be substantially different. You're going to have to figure out a lot on your own.

We certainly don't want to discourage anyone from learning C on the Mac, but we strongly recommend you upgrade to the latest version of OS X so you have access to the latest version of Xcode.

What's a Registered Developer?

So what's a registered developer and do you need to be one? The short answer is "not yet."

Becoming a registered developer grants you access to even more tools and resources than just Xcode. But you don't need any of that to write great applications for OS X or iOS! You don't need it to use Xcode. You certainly don't need to be a registered developer to work through this book (or most other books, for that matter).

You *will* need to become a registered developer if you want to sell, or even give away, your masterpieces on any of Apple's app stores. How cool would that be? You can register at any time, so there's no hurry. When you are ready, visit `http://developer.apple.com/`.

Getting the Projects

While you're still waiting for Xcode to download and install, why not get the project files for this book? Everything you need to create the projects in this book is described in the text, but downloading the finished projects from the Apress web site will save you a lot of typing.

Go to `http://www.apress.com/book/view/9781430245339`. Below the book's description, you'll see some folder tabs, one of which is labeled `Source Code/Downloads`. Click that tab. Now find the link that downloads the projects for this book. Click that link and a file named `Learn C Projects.zip` will download to your hard drive.

Locate the file `Learn C Projects.zip` in your `Downloads` folder (or wherever the browser saved it). Double-click the file to extract its contents, leaving you with a folder named `Learn C Projects`. Move the folder wherever you like.

Using Xcode

Once Xcode has finished installing, launch it as you would any application, from the dock or LaunchPad. When first launched, Xcode will present its startup window (Figure 1-2).

Figure 1-2. *Xcode startup window*

The startup window has convenient buttons that create a new project, reopen a recently visited project, link to the Xcode documentation, and some other stuff we're not going to cover in this book.

Xcode organizes your work around a *project*. A project is a collection of files that ultimately produce a program. It always consists of a *project document* (the icon with the little blueprint) stored inside a folder, as shown in Figure 1-3. That folder is called the *project folder*. You open a project by opening the project document.

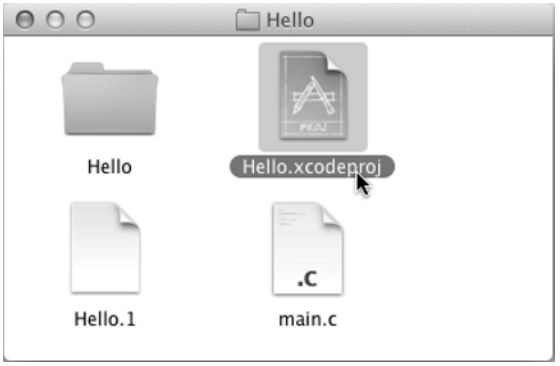


Figure 1-3. *The contents of a simple Xcode project folder*

When opened in Xcode, your project appears in a *workspace window*, as shown in Figure 1-4. The window is full of cryptic settings and seemingly complex controls, but don't worry. Until you get to some really advanced programming, you won't need to fiddle with any of these settings.

Figure 1-4. *A workspace window in Xcode*

Creating a New Xcode Project

While Xcode still has that "new car smell," let's take it for a quick spin around the block and create a new Xcode project.

To do this, either click on the link labeled *Create a new Xcode project* link in the startup window, or choose File ➤ New ➤ Project from the menubar. You'll be

presented with the *new project assistant*, shown in Figure 1-5, which will help you specify the type of new project you want to create.

Figure 1-5. *New project assistant*

The left side of the new project assistant lets you choose whether to create a project for iOS (one that will run on your iPhone, iPad, or iPod touch) or for Mac OS X (one that will run on your computer). Select *Application* in the Mac OS X section.

Next, you need to decide the type of Mac OS X application you want to build. In this book, you're going to learn how to build simple, text-only applications that display text in a window, one line at a time. Once you finish this book, you can move on to books that will teach you how to use the skills you've just mastered to build applications that will run on your iOS device or on your Mac with the graphical elements that define those devices.

Select *Command Line Tool* from the templates pane. This is the only project template you'll be using in this book. To complete your selection, click the *Next* button.

The next screen (Figure 1-6) lets you name your new project and specify a few other options. For a command-line tool the options are pretty simple. Enter *Hello* in the *Name* field.

The field *Company Identifier* allows Xcode to specify who made this application. Typically, this is a reverse of a domain name you've set up for your product. Unless you've got a specific identifier you want to use, use one we've set up for this book. Enter *com.apress.learnc* in the *Company Identifier field.*

Figure 1-6. *Project template options*

Set the *Type* pop-up menu to *C*, since you'll be writing all your programs in the C programming language.

Automatic Reference Counting doesn't apply to C. Leave the *Use Automatic Reference Counting* checkbox unchecked.

Now that your options are all set, click the *Next* button.

Finally, Xcode will prompt you for a location in which to save your project folder. Though you can save your projects anywhere you like, you might want to first

create a master folder, perhaps named *My Learn C Projects*, in which you can store all the projects you create for this book

The Workspace Window

Xcode opens your new project in a workspace window, as shown in Figure 1-7. The workspace window is divided up into panes or views. On the left are the navigators (how you get around your project). In the middle are your editors (where you write and design your application). To edit a file, double-click the file in the navigator and it will appear in the editor. On the right are utilities (inspectors, libraries, help, and such). Any of these views can be hidden as you work. In Figure 1-7, the utilities are hidden for the sake of simplicity.

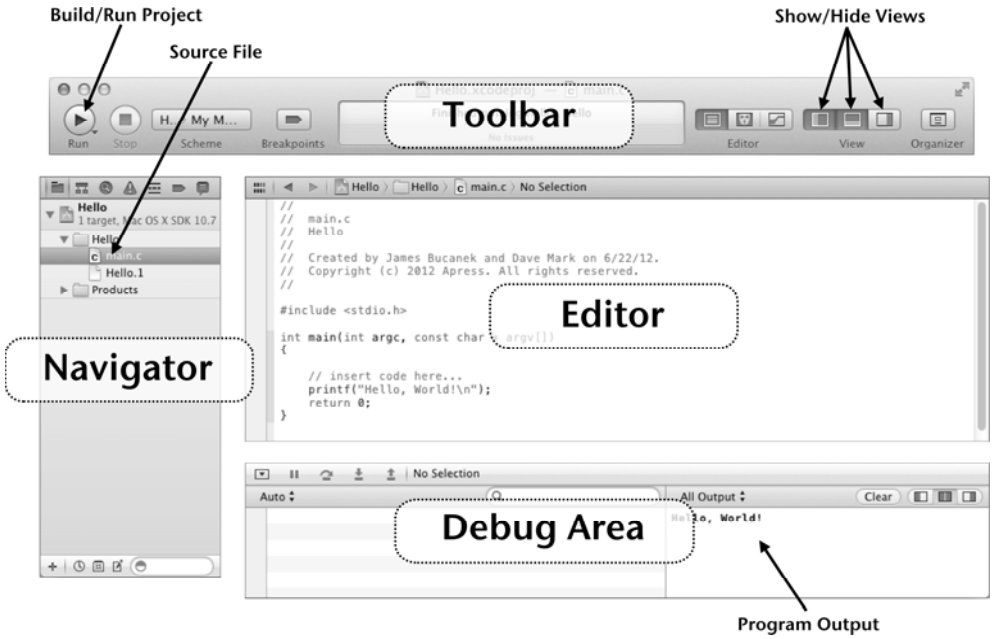


Figure 1-7. *Hello project workspace window*

At the bottom you'll find the debug area, which normally appears only while you're running or testing a program. This is where you inspect your program while it's running and view its output. At the very top is the toolbar. It has buttons and controls for things you commonly do. The big *Run* button at the left will build and run your program, which is what it's all about. Everything in the toolbar is just a shortcut for a command in the Xcode menubar; it doesn't matter which you use.

Running a Project

One really nice thing about Xcode project templates is that they always create a finished project. That is, everything it needs to build and run is ready right from the start. Of course, it won't do anything useful. In fact, it really won't do much of anything at all beyond starting and then stopping again. Changing your project to do something useful is your job.

But don't let that stop you; let's make your new project do nothing! Click the *Run* button (the big *Play* button in the upper left corner of the workspace window). Xcode will assemble all of the parts of your project (a process know as building) and will then execute it.

Don't expect fireworks. The Xcode command-line template makes a project that causes the words "Hello, World!" to appear in the lower right pane (called the *console*), as shown in Figure 1-7.

HELLO, WORLD!

Dennis Ritchie developed the original C language over a period of time between hippies and disco. Years later, he worked with Brian Kernighan to pen a complete description of the language. This version of C became known as *K&R C*.

In their seminal book, the very first example of C (it's on page 6; you can look it up) was a tiny program that caused the words "Hello, World" to appear on a console. And in those days it was probably a Teletype console—a washing-machine–sized mechanical typewriter with roll paper.

Ever since that day, practically every book that explains, teaches, or describes a programming language starts with an example that makes the words "Hello, World" appear somewhere. In the spirit of that grand tradition, we are honor bound to teach you how to make "Hello, World!" appear on your Mac!

Moving On

Believe it or not, you are now ready to learn C on the Mac!

You've installed all of the tools you need to create OS X applications, and you've created, built, and run a brand new application. That's pretty good for one chapter!

The next chapter will take a break from all of this excitement to talk about the software development process in general.

Chapter 2

Programming Basics

Before we dig into C programming specifics, let's spend a few minutes discussing the basics of programming. Why write a computer program? How do computer programs work? We'll answer these questions and look at all of the elements that come together to create a computer program, such as source code, a compiler, and the computer itself.

If you are already familiar with the basics of programming, please feel free to skim through this chapter and, if you feel comfortable with the material, skip on ahead to Chapter 3. The goal here is to get you familiar with the steps involved in creating a running a simple program.

Programming

Why write a computer program? There are many reasons. Some programs are written in direct response to a problem too complex to solve by hand. For example, you might write a program to calculate a value to 5,000 decimal places or to determine the precise moment to fire the boosters that will safely land the Mars Rover.

Other programs are written as performance aids, allowing you to perform a regular task more efficiently. You might write a program to help you balance your checkbook, keep track of your baseball card collection, or lay out this month's issue of *Dinosaur Today*.

Whatever their purpose, each of these examples shares a common theme. They are all examples of the art of programming. Your goal in reading this book is to learn how to use the C programming language to create programs of your own. Before we get into C, however, let's take a minute to look at some other ways to solve your programming problems.