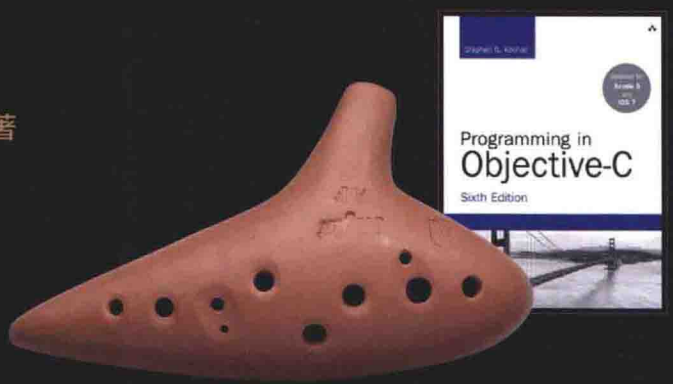
Broadview
www.broadview.com.cn
PEARSON

·原味精品书系·

Objective-C程序设计（英文版）（第6版）

Programming in Objective-C (6th Edition)

[美] Stephen G. Kochan 著



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

· 原味精品书系 ·

Objective-C 程序设计（英文版）（第6版）

Programming in Objective-C (6th Edition)

[美] Stephen G. Kochan 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

本书是为在苹果 iOS 和 OS X 平台上，使用 Objective-C 语言和面向对象程序设计模式进行专业开发而编写的简洁、细致的入门读物。本书假设读者无面向对象程序语言或 C 语言编程经验，以保障初学者与有经验的程序员一样，可用本书迅速和有效地学习 Objective-C。本书提供的学习方法独特，配有众多程序示例及章末练习，适合自学和课堂教学。第 6 版已全面更新，充分纳入 Objective-C 的新功能与技术，同时覆盖对新版 Xcode、iOS 和 Mac OS X Mavericks 的介绍。

版权贸易合同登记号　图字：01-2015-5602

图书在版编目（CIP）数据

Objective-C 程序设计：第 6 版 =Programming in Objective-C (6th Edition)：英文 /（美）寇肯（Kochan,S.G.）著. — 北京：电子工业出版社，2016.4
（原味精品书系）
ISBN 978-7-121-27275-2
I. ① O…II. ① 寇…III. ① C 语言－程序设计－英文 IV. ① TP312

中国版本图书馆 CIP 数据核字（2015）第 229093 号

策划编辑：张春雨
责任编辑：徐津平
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980 1/16　　印张：34　　字数：816 千字
版　　次：2016 年 4 月第 1 版
印　　次：2016 年 4 月第 1 次印刷
定　　价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

献给我深深思念的 Roy 和 Ve。

献给 Ken Brown，“*It's just a jump to the left*”。

作者简介

Stephen G. Kochan 是多本畅销书的作者或合著者，其中有关于C语言的，包括 *Programming in C*（Sams，2004）、*Programming in ANSI C*（Sams，1994）和 *Topics in C Programming*（Wiley，1991），也有关于UNIX的，包括 *Exploring the UNIX System*（Sams，1992）和 *UNIX Shell Programming*（Sams，2003）。从1984年Mac最初引进时，他就开始在Macintosh计算机上编程了，他编写的 *Programming C for the Mac* 是Apple Press Library的一部分。2003年，Kochan编写了 *Programming in Objective-C*（Sams，2003），之后编写了另一本与Mac有关的书籍 *Beginning AppleScript*（Wiley，2004）。

技术审校人员简介

Michael Trent 从1997年开始使用Objective-C编程，之前在Mac上编程。他定期为Steven Frank的网站（www.cocoadev.com）供稿，为大量的书籍和杂志文章做过技术审校，偶尔也涉足Mac OS X开源项目。目前，他正在使用Objective-C和苹果计算机的Cocoa框架生成Mac OS X使用的专业视频应用程序。Michael拥有比洛特学院（位于威斯康辛州比洛特）的计算机科学学士学位和音乐艺术学位。他与妻子Angela居住在加利福尼亚州的圣克拉拉。

Wendy Mui 是旧金山湾区的程序员和软件开发经理。她通过Steve G. Kochan的 *Programming in Objective-C, 2e* 学习了Objective-C，在Bump Technologies公司找到了一份工作，将她的编程技能用在客户端应用程序和Bump第三方开发者使用的API/SDK上。在从事iOS开发之前，Wendy在位于硅谷和加利福尼亚的Sun公司及其他科技公司工作。她迷上编程是在加州大学伯克利分校获得数学硕士学位的时候。Wendy不工作的时候，就在冲击她的跆拳道黑带四段。

目录

III: Cocoa, Cocoa Touch, and the iOS SDK

Appendixes

1

Introduction

Dennis Ritchie at AT&T Bell Laboratories pioneered the C programming language in the early 1970s. However, this programming language did not begin to gain widespread popularity and support until the late 1970s. This was because, until that time, C compilers were not readily available for commercial use outside of Bell Laboratories. Initially, this growth in popularity was also partly spurred by the equal, if not faster, growth in popularity of the UNIX operating system, which was written almost entirely in C.

Brad J. Cox designed the Objective-C language in the early 1980s. The language was based on a language called SmallTalk-80. Objective-C was *layered* on top of the C language, meaning that extensions were added to C to create a new programming language that enabled *objects* to be created and manipulated.

NeXT Software licensed the Objective-C language in 1988 and developed its libraries and a development environment called NEXTSTEP. In 1992, Objective-C support was added to the Free Software Foundation's GNU development environment. The copyrights for all Free Software Foundation (FSF) products are owned by the FSF. It is released under the GNU General Public License.

In 1994, NeXT Computer and Sun Microsystems released a standardized specification of the NEXTSTEP system, called OPENSTEP. The FSF's implementation of OPENSTEP is called GNUStep. A Linux version, which also includes the Linux kernel and the GNUStep development environment, is called, appropriately enough, LinuxSTEP.

On December 20, 1996, Apple Computer announced that it was acquiring NeXT Software, and the NEXTSTEP/OPENSTEP environment became the basis for the next major release of Apple's operating system, OS X. Apple's version of this development environment was called Cocoa. With built-in support for the Objective-C language, coupled with development tools such as Project Builder (or its successor Xcode) and Interface Builder, Apple created a powerful development environment for application development on Mac OS X.

In 2007, Apple released an update to the Objective-C language and labeled it Objective-C 2.0. That version of the language formed the basis for the second edition of the book.

When the iPhone was released in 2007, developers clamored for the opportunity to develop applications for this revolutionary device. At first, Apple did not welcome third-party application development. The company's way of placating wannabe iPhone developers was to allow them to develop Web-based applications. A Web-based application runs under the iPhone's built-in Safari Web browser and requires the user to connect to the website that hosts the application in order to run it. Developers were not satisfied with the many inherent limitations of Web-based applications, and Apple shortly thereafter announced that developers would be able to develop so-called *native* applications for the iPhone.

A native application is one that resides on the iPhone and runs under the iPhone's operating system, in the same way that the iPhone's built-in applications (such as Contacts, Stocks, and Weather) run on the device. The iPhone's OS is actually a version of OS X, which means that applications can be developed and debugged on a MacBook Pro, for example. In fact, Apple soon provided a powerful software development kit (SDK) that allowed for rapid iPhone application development and debugging. The availability of an iPhone simulator made it possible for developers to debug their applications directly on their development system, obviating the need to download and test the program on an actual iPhone or iPod touch device.

With the introduction of the iPad in 2010, Apple started to genericize the terminology used for the operating system and the SDK that now support different devices with different physical sizes and screen resolutions. The iOS SDK allows you to develop applications for any iOS device, and as of this writing, iOS 7 is the current release of the operating system.

What You Will Learn from This Book

When I contemplated writing a tutorial on Objective-C, I had to make a fundamental decision. As with other texts on Objective-C, I could write mine to assume that the reader already knew how to write C programs. I could also teach the language from the perspective of using the rich library of routines, such as the Foundation and UIKit frameworks. Some texts also take the approach of teaching how to use the development tools, such as the Mac's Xcode and the tool formerly known as Interface Builder to design the UI.

I had several problems adopting this approach. First, learning the entire C language before learning Objective-C is wrong. C is a *procedural* language containing many features that are not necessary for programming in Objective-C, especially at the novice level. In fact, resorting to some of these features goes against the grain of adhering to a good object-oriented programming methodology. It's also not a good idea to learn all the details of a procedural language before learning an object-oriented one. This starts the programmer in the wrong direction, and gives the wrong orientation and mindset for fostering a good object-oriented programming style. Just because Objective-C is an extension to the C language doesn't mean you have to learn C first.

So, I decided neither to teach C first nor to assume prior knowledge of the language. Instead, I decided to take the unconventional approach of teaching Objective-C and the underlying C language as a single integrated language, from an object-oriented programming perspective. The purpose of this book is, as its name implies, to teach you how to program in Objective-C.

It does not profess to teach you in detail how to use the development tools that are available for entering and debugging programs, or to provide in-depth instructions on how to develop interactive graphical applications. You can learn all that material in greater detail elsewhere, after you have learned how to write programs in Objective-C. In fact, you will find mastering that material much easier when you have a solid foundation of how to program in Objective-C. This book does not assume much, if any, previous programming experience. In fact, if you are a novice programmer, with some dedication and hard work you should be able to learn Objective-C as your first programming language. Other readers have been successful at this, based on the feedback I have received from the previous editions of this book.

This book teaches Objective-C by example. As I present each new feature of the language, I usually provide a small complete program example to illustrate the feature. Just as a picture is worth a thousand words, so is a properly chosen program example. You are strongly encouraged to run each program and compare the results obtained on your system to those shown in the text. By doing so, you will learn the language and its syntax, but you will also become familiar with the process of compiling and running Objective-C programs.

How This Book Is Organized

This book is divided into three logical parts. Part I, "The Objective-C Language," teaches the essentials of the language. Part II, "The Foundation Framework," teaches how to use the rich assortment of predefined classes that form the Foundation framework. Part III, "Cocoa, Cocoa Touch, and the iOS SDK," gives you an overview of the Cocoa and Cocoa Touch frameworks and then walks you through the process of developing a simple iOS application using the iOS SDK.

A *framework* is a set of classes and routines that have been logically grouped together to make developing programs easier. Much of the power of programming in Objective-C rests on the extensive frameworks that are available.

Chapter 2, "Programming in Objective-C," begins by teaching you how to write your first program in Objective-C.

Because this is not a book on Cocoa or iOS programming, graphical user interfaces (GUIs) are not extensively taught and are hardly even mentioned until Part III. So, an approach was needed to get input into a program and produce output. Most of the examples in this text take input from the keyboard and produce their output in a window pane: a Terminal window if you're using the command line, or a debug output pane if you're using Xcode.

Chapter 3, "Classes, Objects, and Methods," covers the fundamentals of object-oriented programming. This chapter introduces some terminology, but it is kept to a minimum. I also introduce the mechanism for defining a class and the means for sending messages to instances or objects. Instructors and seasoned Objective-C programmers will notice that I use *static* typing for declaring objects. I think this is the best way for the student to get started because the compiler can catch more errors, making the programs more self-documenting and encouraging the new programmer to explicitly declare the data types when they are known. As a result,

the notion of the `id` type and its power is not fully explored until Chapter 9, "Polymorphism, Dynamic Typing, and Dynamic Binding."

Chapter 4, "Data Types and Expressions," describes the basic Objective-C data types and how to use them in your programs.

Chapter 5, "Program Looping," introduces the three looping statements you can use in your programs: `for`, `while`, and `do`.

Making decisions is fundamental to any computer programming language. Chapter 6, "Making Decisions," covers the Objective-C language's `if` and `switch` statements in detail.

Chapter 7, "More on Classes," delves more deeply into working with classes and objects. Details about methods, multiple arguments to methods, and local variables are discussed here.

Chapter 8, "Inheritance," introduces the key concept of inheritance. This feature makes the development of programs easier because you can take advantage of what comes from above. Inheritance and the notion of subclasses make modifying and extending existing class definitions easy.

Chapter 9 discusses three fundamental characteristics of the Objective-C language. Polymorphism, dynamic typing, and dynamic binding are the key concepts covered here.

Chapters 10–13 round out the discussion of the Objective-C language, covering issues such as initialization of objects, blocks, protocols, categories, the preprocessor, and some of the underlying C features, including functions, arrays, structures, and pointers. These underlying features are often unnecessary (and often best avoided) when first developing object-oriented applications. It's recommended that you skim Chapter 13, "Underlying C Language Features," the first time through the text and return to it only as necessary to learn more about a particular feature of the language. Chapter 13 also introduces a recent addition to the C language known as *blocks*. This should be learned after you learn about how to write functions, since the syntax of the former is derived from the latter.

Part II begins with Chapter 14, "Introduction to the Foundation Framework," which gives an introduction to the Foundation framework and how to use its voluminous documentation.

Chapters 15–19 cover important features of the Foundation framework. These include number and string objects, collections, the file system, memory management, and the process of copying and archiving objects.

By the time you're done with Part II, you will be able to develop fairly sophisticated programs in Objective-C that work with the Foundation framework.

Part III starts with Chapter 20, "Introduction to Cocoa and Cocoa Touch." Here you get a quick overview of the frameworks that provide the classes you need to develop sophisticated graphical applications on the Mac and on your iOS devices.

Chapter 21, "Writing iOS Applications," introduces the iOS SDK and the UIKit framework. This chapter illustrates a step-by-step approach to writing a simple iOS application, followed