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。

Original edition, entitled Implementing Programming in Objective-C, 6e, 0321967607, by Stephen G. Kochan, published by Pearson Education, Inc., publishing as Addison-Wesley Professional, Copyright©2014 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education,Inc.

China edition published by Pearson Education Asia Ltd. and Publishing House of Electronics Industry Copyright © 2016.The edition is manufactured in the People's Republic of China, and is authorized for sale and distribution only in the mainland of China exclusively(except Hong Kong SAR, Macau SAR, and Taiwan).

本书英文影印版专有出版权由 Pearson Education 培生教育出版亚洲有限公司授予电子工业出版社。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书仅限中国大陆境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

本书英文影印版贴有 Pearson Education 培生教育出版集团激光防伪标签，无标签者不得销售。

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。

目录

1 Introduction 1

What You Will Learn from This Book 2

How This Book Is Organized 3

Support 5

Acknowledgments 5

Preface to the Sixth Edition 6

I: The Objective-C Language

2 Programming in Objective-C 7

Compiling and Running Programs 7

Using Xcode 8

Using Terminal 16

Explanation of Your First Program 18

Displaying the Values of Variables 22

Summary 25

Exercises 25

3 Classes, Objects, and Methods 27

What Is an Object, Anyway? 27

Instances and Methods 28

An Objective-C Class for Working with Fractions 30

The `@interface` Section 33

Choosing Names 34

Class and Instance Methods 35

The `@implementation` Section 37

The `program` Section 39

Accessing Instance Variables and Data Encapsulation 45

Summary 49

Exercises 49

4 Data Types and Expressions 51

Data Types and Constants 51

Type `int` 51

Type `float` 52

Type char 52
Qualifiers: long, long long, short, unsigned, and signed 53
Type id 54
Arithmetic Expressions 55
Operator Precedence 55
Integer Arithmetic and the Unary Minus Operator 58
The Modulus Operator 60
Integer and Floating-Point Conversions 61
The Type Cast Operator 63
Assignment Operators 64
A Calculator Class 65
Exercises 67

5 Program Looping 71
The for Statement 72
Keyboard Input 79
Nested for Loops 81
for Loop Variants 83
The while Statement 84
The do Statement 89
The break Statement 91
The continue Statement 91
Summary 91
Exercises 92

6 Making Decisions 93
The if Statement 93
The if-else Construct 98
Compound Relational Tests 101
Nested if Statements 104
The else if Construct 105
The switch Statement 115
Boolean Variables 118
The Conditional Operator 123
Exercises 125

此为试读,需要完整PDF请访问: www.ertongbook.com

7 More on Classes 127
Separate Interface and Implementation Files 127
Synthesized Accessor Methods 133
Accessing Properties Using the Dot Operator 135
Multiple Arguments to Methods 137
Methods without Argument Names 139
Operations on Fractions 139
Local Variables 143
Method Arguments 144
The `static` Keyword 144
The `self` Keyword 148
Allocating and Returning Objects from Methods 149
Extending Class Definitions and the Interface File 151
Exercises 151

8 Inheritance 153
It All Begins at the Root 153
Finding the Right Method 157
Extension through Inheritance: Adding New Methods 158
A Point Class and Object Allocation 162
The `@class` Directive 163
Classes Owning Their Objects 167
Overriding Methods 171
Which Method Is Selected? 173
Abstract Classes 176
Exercises 176

9 Polymorphism, Dynamic Typing, and Dynamic Binding 179
Polymorphism: Same Name, Different Class 179
Dynamic Binding and the `id` Type 182
Compile Time Versus Runtime Checking 184
The `id` Data Type and Static Typing 185
Argument and Return Types with Dynamic Typing 186
Asking Questions about Classes 187
Exception Handling Using `@try` 192
Exercises 195

10 More on Variables and Data Types 197

Initializing Objects 197

Scope Revisited 200

More on Properties, Synthesized Accessors, and Instance Variables 201

Global Variables 202

Static Variables 204

Enumerated Data Types 207

The `typedef` Statement 210

Data Type Conversions 211

Conversion Rules 212

Bit Operators 213

The Bitwise AND Operator 215

The Bitwise Inclusive-OR Operator 216

The Bitwise Exclusive-OR Operator 216

The Ones Complement Operator 217

The Left-Shift Operator 218

The Right-Shift Operator 219

Exercises 220

11 Categories and Protocols 223

Categories 223

Class Extensions 228

Some Notes about Categories 229

Protocols and Delegation 230

Delegation 233

Informal Protocols 233

Composite Objects 234

Exercises 235

12 The Preprocessor 237

The `#define` Statement 237

More Advanced Types of Definitions 239

The `#import` Statement 244

Conditional Compilation 245
The #ifdef, #endif, #else, and #ifndef Statements 245
The #if and #elif Preprocessor Statements 247
The #undef Statement 248
Exercises 249

13 Underlying C Language Features 251
Arrays 252
Initializing Array Elements 254
Character Arrays 255
Multidimensional Arrays 256
Functions 258
Arguments and Local Variables 259
Returning Function Results 261
Functions, Methods, and Arrays 265
Blocks 266
Structures 270
Initializing Structures 273
Structures within Structures 274
Additional Details about Structures 276
Don't Forget about Object-Oriented Programming! 277
Pointers 277
Pointers and Structures 281
Pointers, Methods, and Functions 283
Pointers and Arrays 284
Operations on Pointers 294
Pointers and Memory Addresses 296
They're Not Objects! 297
Miscellaneous Language Features 297
Compound Literals 297
The goto Statement 298
The Null Statement 298
The Comma Operator 299
The sizeof Operator 299
Command-Line Arguments 300

How Things Work 302
Fact 1: Instance Variables Are Stored in Structures 303
Fact 2: An Object Variable Is Really a Pointer 303
Fact 3: Methods Are Functions, and Message Expressions Are Function Calls 304
Fact 4: The `id` Type Is a Generic Pointer Type 304
Exercises 304

II: The Foundation Framework

14 Introduction to the Foundation Framework 307
Foundation Documentation 307

15 Numbers, Strings, and Collections 311
Number Objects 311
String Objects 317
More on the `NSLog` Function 317
The `description` Method 318
Mutable Versus Immutable Objects 319
Mutable Strings 326
Array Objects 333
Making an Address Book 338
Sorting Arrays 355
Dictionary Objects 362
Enumerating a Dictionary 364
Set Objects 367
`NSIndexSet` 371
Exercises 373

16 Working with Files 377
Managing Files and Directories: `NSFileManager` 378
Working with the `NSData` Class 383
Working with Directories 384
Enumerating the Contents of a Directory 387
Working with Paths: `NSPathUtilities.h` 389
Common Methods for Working with Paths 392
Copying Files and Using the `NSProcessInfo` Class 394

Basic File Operations: NSFileHandle 398
The NSURL Class 403
The NSBundle Class 404
Exercises 405

17 Memory Management and Automatic Reference Counting 407
Automatic Garbage Collection 409
Manual Reference Counting 409
Object References and the Autorelease Pool 410
The Event Loop and Memory Allocation 412
Summary of Manual Memory Management Rules 414
Automatic Reference Counting 415
Strong Variables 415
Weak Variables 416
@autoreleasepool Blocks 417
Method Names and Non-ARC Compiled Code 418

18 Copying Objects 419
The copy and mutableCopy Methods 419
Shallow Versus Deep Copying 422
Implementing the <NSCopying> Protocol 424
Copying Objects in Setter and Getter Methods 427
Exercises 429

19 Archiving 431
Archiving with XML Property Lists 431
Archiving with NSKeyedArchiver 434
Writing Encoding and Decoding Methods 435
Using NSData to Create Custom Archives 442
Using the Archiver to Copy Objects 446
Exercises 447

III: Cocoa, Cocoa Touch, and the iOS SDK

20 Introduction to Cocoa and Cocoa Touch 449
Framework Layers 449
Cocoa Touch 450

21 Writing iOS Applications 453
The iOS SDK 453
Your First iPhone Application 453
Creating a New iPhone Application Project 456
Entering Your Code 460
Designing the Interface 462
An iPhone Fraction Calculator 469
Starting the New Fraction_Calculator Project 471
Defining the View Controller 471
The `Fraction` Class 477
A `Calculator` Class That Deals with Fractions 480
Designing the User Interface 482
Summary 483
Exercises 484

Appendixes

A Glossary 485

B Address Book Example Source Code 493

Index 499

此为试读，需要完整PDF请访问：www.ertongbook.com

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

by a more sophisticated calculator application that enables you to use your iPhone to perform simple arithmetic calculations with fractions.

Because object-oriented parlance involves a fair amount of terminology, Appendix A, "Glossary," provides definitions of some common terms.

Appendix B, "Address Book Example Source Code," gives the source code listing for two classes that are developed and used extensively in Part II of this text. These classes define address card and address book classes. Methods enable you to perform simple operations such as adding and removing address cards from the address book, looking up someone, listing the contents of the address book, and so on.

After you've learned how to write Objective-C programs, you can go in several directions. You might want to learn more about the underlying C programming language, or you might want to start writing Cocoa programs to run on OS X, or you might want to develop more-sophisticated iOS applications.

Support

If you go to classroomM.com/objective-c, you'll find a forum rich with content. There you can get some source code (note that you won't find the "official" source code for all the examples there; I firmly believe that a big part of the learning process occurs when you type in the program examples yourself and learn how to identify and correct any errors), answers to exercises, errata, and quizzes; you can also pose questions to me and fellow forum members. The forum has turned into a rich community of active members who are happy to help other members solve their problems and answer their questions. Please go, join, and participate!

Acknowledgments

I would like to acknowledge several people for their help in the preparation of the first edition of this text. First, I want to thank Tony Iannino and Steven Levy for reviewing the manuscript. I am also grateful to Mike Gaines for providing his input.

I'd also like to thank my technical editors, Jack Purdum (first edition), Wendy Mui (third edition), and Mike Trent (first, second, fifth, and sixth editions). I was particularly lucky to have Mike review the first two editions of this text. He provided the most thorough review of any book I've ever written. Not only did he point out weaknesses, but he was also generous enough to offer his suggestions. Because of Mike's comments in the first edition, I changed my approach to teaching memory management and tried to make sure that every program example in this book was "leak free." This was prior to the fourth edition, where the strong emphasis on memory management became obsolete with the introduction of ARC. Mike also provided invaluable input for my chapter on iOS programming.

From the first edition, Catherine Babin supplied the cover photograph and provided me with many wonderful pictures to choose from. Having the cover art from a friend made the book even more special.

I am so grateful to Mark Taber (for all editions) from Pearson for putting up with all delays and for being kind enough to work around my schedule and to tolerate my consistent missing of deadlines. The same kudos to Mandie Frank from Pearson. Mandie has worked tirelessly with my late deliveries to help get various editions of this book out on time. I am extremely grateful to Michael de Haan and Wendy Mui for doing an incredible, unsolicited job proofreading the first printing of the second edition.

As noted at the start of this Introduction, Dennis Ritchie invented the C language. He was also a co-inventor of the Unix operating system, which is the foundation for OS X and iOS. Sadly, the world lost both Dennis Ritchie and Steve Jobs within the span of a week in 2011. These two people had a profound effect on my career; this book would not exist if not for them.

Finally, I'd like to thank the members of the forum at classroomM.com/objective-c for all their feedback, support, and kind words.

Preface to the Sixth Edition

Not much has changed since the previous edition. Xcode 5 was introduced, and so all the screenshots have been updated. I've needed to do this so that novices can follow along with current screenshots and not get lost before even getting started! There are also some minor additions to the language, which are reflected in this edition.

Stephen G. Kochan

October 2013

此为试读,需要完整PDF请访问: www.ertongbook.com