

资深Mac OS X和iOS应用开发者Robert Clair倾力打造，内容全面且深入，是学习Objective-C 2.0的最佳入门指南

既展示Objective-C的强大功能，又全方位讲解利用Objective-C为Mac OS X或iOS开发应用的各种技术细节、方法和最佳实践，包含Objective-C 2.0的新增特性

Learning Objective-C 2.0

A Hands-on Guide to Objective-C for Mac and iOS Developers, Second Edition

Objective-C 2.0 Mac 和iOS开发实践指南

(原书第2版)

(美) Robert Clair 著
陈宗斌 李江晋 等译



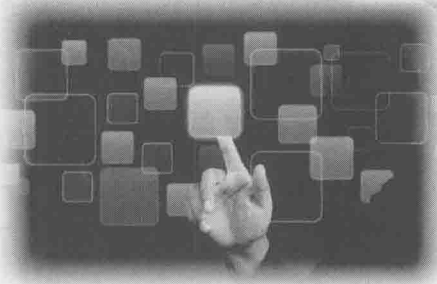
机械工业出版社
China Machine Press

Learning Objective-C 2.0
A Hands-on Guide to Objective-C for Mac and iOS Developers, Second Edition

Objective-C 2.0 Mac 和iOS开发实践指南

(原书第2版)

(美) Robert Clair 著
陈宗斌 李江晋 等译



机械工业出版社
China Machine Press

图书在版编目(CIP)数据

Objective-C 2.0 Mac 和 iOS 开发实践指南(原书第2版)/(美)克莱尔(Clair, R.)著;
陈宗斌等译. —北京:机械工业出版社, 2014.12

(华章程序员书库)

书名原文: Learning Objective-C 2.0: A Hands-on Guide to Objective-C for Mac and
iOS Developers, Second Edition

ISBN 978-7-111-48456-1

I. O… II. ①克… ②陈… III. ①移动终端—应用程序—程序设计—指南 ②C语言—
程序设计—指南 IV. ①TN929.53-62 ②TP312-62

中国版本图书馆CIP数据核字(2014)第258289号

本书版权登记号: 图字: 01-2013-1390

Authorized translation from the English language edition, entitled *Learning Objective-C 2.0: A Hands-on Guide to Objective-C for Mac and iOS Developers, 2nd Edition*, 9780321832085 by Robert Clair, published by Pearson Education, Inc., Copyright © 2013.

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.

Chinese simplified language edition published by Pearson Education Asia Ltd., and China Machine Press Copyright © 2015.

本书中文简体字版由 Pearson Education(培生教育出版集团)授权机械工业出版社在中华人民共和国境内(不包括中国台湾地区和香港、澳门特别行政区)独家出版发行。未经出版者书面许可,不得以任何方式抄袭、复制或节录本书中的任何部分。

本书封底贴有 Pearson Education(培生教育出版集团)激光防伪标签,无标签者不得销售。

Objective-C 2.0 Mac 和 iOS 开发实践指南 (原书第2版)

出版发行:机械工业出版社(北京市西城区百万庄大街22号 邮政编码:100037)

责任编辑:秦健

责任校对:殷虹

印刷:三河市宏图印务有限公司

版次:2015年1月第1版第1次印刷

开本:186mm×240mm 1/16

印张:23.75

书号:ISBN 978-7-111-48456-1

定价:79.00元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

客服热线:(010)88378991 88361066

投稿热线:(010)88379604

购书热线:(010)68326294 88379649 68995259

读者信箱:hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问:北京大成律师事务所 韩光/邹晓东

Praise for the First Edition 第1版赞誉

“在本书中，Clair 不仅开门见山地全面介绍了 Objective-C，而且给出了从现实的亲身经历中提取的省时、省力的深刻见解。本书融合了简洁的概述、示例和具体的实现细节，允许快速、完整、全面地理解这种语言及其核心特性和概念。”

——Scott D. Yelich，移动应用程序开发人员

“有许多关于 Objective-C 的图书试图把面向对象程序设计、Objective-C 计算机语言和 Apple 平台上的应用程序开发一网打尽。如此范围的主题太宽泛了，不可能在单独一篇篇幅有限的图书中透彻地加以介绍。Clair 的图书专注于掌握 Objective-C 的基础知识，使合格的程序员能够开始编写 Objective-C 代码。”

——Joseph E. Sacco 博士，Sacco & Associates 公司

“Clair 的这本书经过了精心设计，提供对 Objective-C 语言深刻和有趣的见解，使程序开发新手和经验丰富的专业人员都能够深受启发。当不熟悉这种语言的程序员询问他们应该从哪里开始起步时，我现在就建议他们阅读本书。”

——Matt Long, Cocoa Is My Girlfriend (www.cimgf.com)

“Clair 精通 Objective-C 语言，并以一种使之更容易学习的方式介绍它。无论你是初学者还是专业程序员，都可以在本书中有所收获，而这并不需要预先了解 C 语言。”

——Cory Bohon，Mac/Life 的独立开发人员和博主

“我喜欢这本书，是因为它是一本一点也不枯燥的技术书，容易阅读，但并不肤浅。”

——Andy Lee，AppKiDo 的作者

译者序 *The Translator's Words*

作为 Mac OS X 与 iOS 应用程序的开发语言，Objective-C 近年来颇受关注。尤其是智能手机与平板电脑兴起之后，越来越多的开发人员将目光转向移动开发平台，并开始学习 Objective-C 语言。

Objective-C 最初是由 Brad J. Cox 于 20 世纪 80 年代早期创建的，它是 C 语言的面向对象扩展，可以称为“带有对象的 C”。目前，虽然从技术上讲使用其他的语言编写完整的 OS X 程序是可能的，但是编写一个遵循 Apple 的 Human Interface Guidelines（人机界面指南）并且具有正确的 Mac“外观和感觉”的程序则需要使用 Objective-C Cocoa 框架。即使使用不同的语言（比如纯 C 语言或 C++）编写 Mac 应用程序的核心，也应该用 Objective-C 编写用户界面层。在为 iOS 编写程序时，将别无选择：iOS 应用程序的外层和用户界面必须用 Objective-C 编写。随着移动应用程序开发热潮到来，关注与学习 Objective-C 的人也越来越多。

本书集中介绍了 Objective-C 语言，采用了一种适合教学的方法讲解内容。本书分为 4 个部分。第一部分回顾了 C 语言，然后介绍了面向对象程序设计和 Objective-C。第二部分详细介绍了 Objective-C 语言，还介绍了 Foundation 框架。第三部分介绍了 Objective-C 中的内存管理和 Objective-C 块。第四部分是附录，介绍了一些额外的知识。

本书使用小示例来强调语言的工作方式。除了学习语言的语法之外，还鼓励读者考虑“在后台”所发生的事情。在本书大多数章的末尾都提供了一组练习，可以帮助读者巩固所学的知识。

本书适合于那些先前具有一些编程经验并且想要学习 Objective-C 以便为 OS X 或 iOS 编写程序的人阅读。对于那些想要使用来自 GNUStep 项目（OpenStep 库的开源实现）的软件为其他平台编写 Objective-C 程序的程序员来说，本书也非常具有参考价值。

参加本书翻译的人员有：陈宗斌、李江晋、戴锋、许瑛琪、易小丽、孙朝辉、宋如杰、蔡江林、孙丽、易志东、叶守运、李刚生、傅强、徐茜、张旭。

由于时间紧迫，加之译者水平有限，错误在所难免，恳请广大读者批评指正。

Preface 前言

Objective-C 是 C 语言的面向对象扩展，可以称为“带有对象的 C”。如果你正在阅读本书，就可能对学习 Objective-C 感兴趣，以便可以为 Mac OS X 或 iOS 编写应用程序。但是，学习 Objective-C 还有另一个理由：它是一门有趣的语言，并且相对容易学习。像现实世界中的其他任何事物一样，学习 Objective-C 也有一些难点。但是总体而言，它比其他面向对象语言要简单得多，特别是和 C++ 相比就更是如此。Objective-C 对 C 语言所进行的增补可以在一两页中列举出来。

Objective-C 最初是由 Brad J. Cox 于 20 世纪 80 年代早期创建的。在 1988 年，由 Steve Jobs 在离开 Apple 后创建的 NeXT Computer 公司许可使用 Objective-C，并使之成为用于创建在 NeXT 的 NeXTSTEP 操作系统下运行的应用程序的开发环境的基础。NeXT 工程师开发了一组 Objective-C 库，用于构建应用程序。在 NeXT 于 1993 年从硬件业务中全身而退之后，它与 Sun Microsystems 合作创建了 OpenStep，这是一种针对面向对象系统的开放性规范，并且基于 NeXTSTEP API。Sun 最终失去了对 OpenStep 的兴趣。NeXT 继续销售其 OpenStep 版本，直到 Apple 于 1997 年早期收购了它为止，NeXTSTEP 操作系统也随之变成了 OS X 的基础。

在 Apple 世界，Objective-C 并不是在孤军奋战，它与许多类库（称为框架（framework））协同工作。OS X 上两个最重要的框架是 Foundation 框架和 AppKit 框架，Foundation 框架包含用于基本实体（比如字符串和数组）的类以及包装了与操作系统进行交互的类。AppKit 包含用于窗口、视图、菜单、按钮以及构建图形用户界面所需的其他各种构件（widget）的类。这两个框架合称为 Cocoa。在 iOS 上，用一个名为 UIKit 的不同框架代替了 AppKit，Foundation 和 UIKit 则合称为 Cocoa Touch。

虽然从技术上讲使用其他语言编写完整的 OS X 程序是可能的，但是编写一个遵循 Apple 的 Human Interface Guidelines（人机界面指南）^①并且具有正确的 Mac “外观和感觉”的程序则需要使用 Objective-C Cocoa 框架。即使用不同的语言（比如纯 C 语言或 C++）编

① <http://developer.apple.com/mac/library/documentation/UserExperience/Conceptual/AppleHIGuidelines>。

写 Mac 应用程序的核心，也应该用 Objective-C 编写用户界面层。在为 iOS 编写程序时，将别无选择：iOS 应用程序的外层和用户界面必须用 Objective-C 编写。

关于本书

本书集中介绍了学习 Objective-C 语言。它将不会讲述如何编写 Cocoa 或 Cocoa Touch 程序，而是将介绍和利用 Foundation 框架的一小部分，并且只会顺便提及 AppKit 和 UIKit。本书假设：如果你已经很好地理解了 Cocoa 和 Cocoa Touch 所基于的语言，那么学习 Cocoa 和 Cocoa Touch 编程将容易得多。

一些计算机图书在编写时，采用了我喜欢视之为“跟我学”的风格。将邀请用户复制或下载一些代码，其后接着代码的简要讨论。在引入新特性时，将要求读者更改相关的代码行并观察结果。在进行一点讨论之后，将继续介绍下一个特性。我发现将这种图书风格用于语言类图书无法让人满意。通常，对于实际的工作方式只有非常少的解释。这种图书风格容易引起误解，当读者面临的编程任务不是图书中使用的某个示例的小变体时，他们的自信心将会消失。

本书采用了一种更适宜于教学的方法，并且使用小示例来强调语言的工作方式。除了学习语言的语法之外，还鼓励你考虑“在后台”所发生的事情。这种方法需要你付出更多的努力，但是在你第一次面临新的编程任务时，它将会给你很好的补偿。

本书读者对象

本书适合于那些先前具有一些编程经验并且想要学习 Objective-C 以便为 OS X 或 iOS 编写程序的人阅读（iOS 用于 iPhone、iPod touch 和 iPad）。

对于那些想要使用来自 GNUStep 项目（OpenStep 库的开源实现）[Ⓔ]的软件为其他平台编写 Objective-C 程序的程序员来说，本书也很有用。

你需要知道什么

本书假设读者掌握 C 语言的基础知识。Objective-C 是 C 语言的一个扩展，本书主要关注 Objective-C 向 C 语言中添加了什么功能。对于那些对 C 语言感到生疏以及那些擅长于快速学习一门新语言的人，第 1 章和第 2 章回顾了 C 语言的基本部分，这些是你在编写 Objective-C 程序时很可能要用到的。如果没有使用 C 语言或任何类似 C 的语言（C++、Java 和 C#）的经验，有必要再阅读一本关于 C 语言的图书。此前接触过一种面向对象语言

[Ⓔ] www.gnustep.org/。

是有帮助的，但这并不是绝对需要的。随着本书的推进，将会介绍必需的面向对象概念。

关于示例

为一本介绍性的图书创建代码示例提出了一个挑战：在阐释要点时，怎样才能不迷失在建立一个工作程序可能需要的样板代码的汪洋大海中。在许多情况下，本书都采用了以下方法：使用在一定程度上假想的示例，并对它们进行了“瘦身”，以便帮助你集中精力关注所讨论的要点。因此，示例中省略了不相关的代码部分，并用省略号（…）代替它们。

例如：

```
int averageScore = ...
```

上面的代码行表示应该意味着 `averageScore` 是一个整型变量，并从程序的其他某个部分获取它的值。`averageScore` 值的来源对于该示例来说是不相关的，你只需考虑它具有一个值。

关于练习

本书中大多数章的末尾都有一组练习，当然鼓励你完成它们。许多练习要求你编写小程序来验证在那一章中介绍的要点，这样的练习可能看上去很冗长，但是编写代码并查看结果将会提供比仅仅阅读代码更生动的学习体验。编写小程序来测试你的理解是应该养成的良好习惯；无论何时不清楚某个要点，都应该编写一个这样的程序，即便本书没有提供相关的练习，也应该这样做。在我编写完本书时，有一个目录中装满了小测试程序。当你学完本书时，应该也是如此。

练习所建议的程序都不需要用户界面。可以利用文本编辑器编写代码并从命令行编译和运行它（如第 2 章中的练习之前所示的）或者使用简单的 Xcode 项目（如第 4 章），来编码、编译和运行所有这些程序。

Objective-C——一个移动的目标

Objective-C 是一个移动的目标。在过去几年间，Apple 定期向 Objective-C 中添加一些新的特性和语法。尽管添加了这些特性，但是 Apple 决定不继续版本化该语言。Objective-C 2.0 是它们推出的最高版本。指定语言的特定版本或特性集的唯一方式是：在提及 Objective-C 时，带上 Xcode 或 LLVM 编译器的特定版本。本书这一版介绍的是截止到 Xcode 4.4（随同 OS X 10.8 Mountain Lion 一起发布）的 Objective-C，或者相当于在 LLVM Compiler/Clang 4.0 中实现的 Objective-C。

是否使用 ARC

Objective-C 使用称为保留计数 (retain counting) 的内存管理系统。每个对象都会保留正在使用它的其他对象的数量统计值。当这个计数归零时, 将把对象的内存返还给系统以便重用。在历史上, 这个系统是“手动”的——必须编写代码在合适的时间操纵对象的保留计数。对于许多人来说, 要 100% 正确地遵守这个系统的规则是很困难的。不遵守规则而造成的不幸后果是内存泄漏和程序崩溃。

2011 年春季, Apple 引入了 ARC (Automatic Reference Counting, 自动引用计数)。ARC 通过分析程序并且自动插入将正确地保留计数的代码, 来自动执行引用计数系统。程序员一方无需编写任何代码。

一些人坚称 ARC 避免了学习手动内存管理以及引用计数如何工作的需要。他们说: “无需知道引擎如何工作, 也能驾驶汽车, 同样, 无需知道手动引用计数, 也能利用 ARC 编写 Objective-C 程序。”事实确实如此, 但是就像知道一些有关汽车的引擎如何工作的知识也可能是有价值的一样, 在一些情况下, 理解手动内存管理可能是有价值的或者甚至是必要的。

- ❑ 有许多现有的代码还没有转换成使用 ARC。如果要求你处理非 ARC 代码, 或者如果你想要使用非 ARC 的开源项目, 将不得不理解手动引用计数。
- ❑ 在 Objective-C 框架下面有一组 C 语言库 (“核心” 级的库)。这些库是用面向对象风格编写的, 并且具有它们自己的引用计数系统。虽然要尽可能使用 Objective-C 框架, 但是在一些情况下 (例如, 在图形中) 必须使用更低级的库。要正确地使用这些库, 必须理解手动引用计数的概念。
- ❑ 一些对象是“免费桥接”的 (参见第 8 章)。可以把一个指向低级 C 语言对象之一的指针强制转换为一个指向 Objective-C 框架对象的指针, 反之亦然。在 ARC 下执行该操作需要进行几种特殊的强制转换之一, 而决定使用哪种强制转换则需要理解手动引用计数以及 ARC 正在为你自动做什么。
- ❑ 在一些情况下 (例如, 在一个紧凑循环中创建大量的临时对象), 可以通过执行一些手动调优, 保持程序的内存占用比较少。执行这种调优需要理解引用计数是如何工作的。
- ❑ ARC 仍然相对较新, 在极端情况下仍然会出现怪异的错误或者意想不到的行为。如果遇到其中一种情况, 将需要理解幕后所发生的事情, 以便推断如何消除障碍。

在为本书第 2 版做准备工作时, ARC 让我有点进退两难。我应该放弃关于手动引用计数的章节并且只使用 ARC 吗? 我深深地感觉这将是一个糟糕的选择, 但是为了帮助我做决定, 我问了许多同事这个问题。他们的答案全都一致: 理解引用计数如何工作很重要, 首先要讲述它, 然后介绍 ARC。因此, 本书讲述了手动内存管理, 直到第 16 章为止。在你理解了第 16 章中的知识之后, 就可以返回到前面章节中的练习上来, 并使用 ARC 完成它们。你将发现学习如何进行手动引用计数然后享受在大多数情况下不用这样做的自由要比在紧

急情况的基础上学习它容易得多，因为你遇到了上述列表中的情况之一。

本书组织结构

本书包括4个部分。第一部分回顾了C语言，然后介绍了面向对象程序设计和Objective-C。第二部分详细介绍了Objective-C语言，还介绍了Foundation框架。第三部分介绍了Objective-C中的内存管理和Objective-C块。第四部分是附录，介绍了一些额外的知识。

第一部分：Objective-C 简介

- ❑ 第1章概述C语言的基础知识，并且介绍了在编写Objective-C程序时最有可能需要用到C语言知识。
- ❑ 第2章继续回顾C语言，其中讨论了C和Objective-C程序的内存布局，以及不同类型的变量的内存位置和生存期。即使你知道C语言，也可能想要通读该章。许多经验丰富的C程序员也不完全知道该章包含的知识。
- ❑ 第3章开始介绍面向对象程序设计的概念，并且继续探讨这些概念是如何在Objective-C中体现的。
- ❑ 第4章引领你逐行查看一个简单的Objective-C程序，还介绍了如何使用Apple的集成开发环境Xcode创建一个项目，然后编译和运行一个Objective-C程序。往后就可以使用这些知识来完成本书其余部分的练习。

第二部分：语言基础

对象（object）是面向对象程序设计的主要实体；它们把变量（称为实例变量（instance variable））和类似于函数的代码块（称为方法（method））组织到单个实体中。类是对象的规范，它列出了组成给定类型的对象的实例变量和方法，并且提供了实现这些方法的代码。对象更加具体，它是一个内存区域，类似于C语言的结构，用于保存由对象的类所定义的变量。一个特定的对象被称为定义它的类的一个实例。

- ❑ 第5章开始对Objective-C语言进行全面介绍。在Objective-C中，可以通过给一个对象发送一条消息（message）来让它“做某件事情”。消息是一个方法的名称加上该方法所带有的任何参数。为了响应接收到的消息，对象会执行相应的方法。该章将介绍方法、消息以及Objective-C消息传递系统是如何工作的。
- ❑ 第6章介绍如何定义类，以及创建和复制对象实例。该章还介绍继承（inheritance），即通过扩展现有类（而不是从头开始）来定义一个类的过程。

在执行一个Objective-C程序的过程中，使用的每个类都由一段内存来表示，其中包含关于该类的信息。这段内存称为类的类对象（class object）。类也可以定义类方法

(class method)，它们是代表类（而不是类的实例）执行的方法。

- ❑ 第 7 章介绍类对象和类方法。与某些其他的面向对象语言中的类不同，Objective-C 的类没有类变量，即类的所有实例共享的变量。该章的最后几节说明如何通过使用静态变量来获得类变量的效果。
- ❑ 第 8 章描述 Apple 封装动态链接库的方式。该章介绍框架的定义和结构，并简要描述在编写 OS X 或 iOS 程序时将会遇到的一些常见框架。
- ❑ 第 9 章介绍最常用的 Foundation 类：用于字符串、数组、字典、集合和数字对象等的类。
- ❑ 第 10 章讨论当使用带有 C 语言控制结构的 Objective-C 构造时，需要考虑的额外一些问题。该章继续介绍 Objective-C 所添加的额外的控制结构，包括 Objective-C 2.0 的快速枚举（Fast Enumeration）构造。该章最后解释 Objective-C 的异常处理系统。
- ❑ 第 11 章说明如何向现有类中添加方法而不需要子类化它，以及如何隐藏你认为私有的方法和实例变量的声明。该章最后讨论 Objective-C 的安全性问题。
- ❑ 第 12 章介绍 Objective-C 2.0 的声明式属性（declared property）特性。属性是对象的特征，通常通过对象的实例变量之一来建模。设置或获取属性的方法称为访问器方法（accessor method）。通过使用声明式属性这种特性，可以要求编译器合成属性的访问器方法及其实例变量，从而大大节省你自己的精力。
- ❑ 第 13 章介绍一种表征对象的不同方法。协议是一组定义的方法，类可以选择实现它们。在许多情况下，重要的不是一个对象的类，而是对象的类是否通过实现协议中声明的方法来采用特定的协议（多个类可以采用一个给定的协议，一个类也可以采用多个协议）。Java 的接口概念就借用自 Objective-C 协议。

第三部分：高级概念

第三部分中的各章详细介绍内存管理和 Objective-C 块。

- ❑ 第 14 章讨论内存管理的问题，并且简要介绍 Objective-C 提供的两种内存管理系统。
- ❑ 第 15 章介绍 Objective-C 的传统手动引用计数系统。引用计数也称为保留计数（retain counting）或托管内存（managed memory）。在使用引用计数的程序中，每个对象都会保存一个计数，称为保留计数（retain count），这是使用该对象的其他对象的数量。当这个计数减少到 0 时，就会释放该对象。该章介绍使保留计数保持有序所需的规则。
- ❑ 第 16 章介绍自动引用计数（ARC）。ARC 不是一种完全不同的内存管理系统。相反，ARC 会自动执行 Objective-C 的传统引用计数系统。ARC 是一个编译时进程，它将分析 Objective-C 代码，并插入合适的内存管理消息。
- ❑ 第 17 章讨论 Objective-C 2.0 的块（block）特性。块类似于匿名函数，但是，除此之外，块还携带有其包围环境中变量的值。块是 Apple 的 GCD（Grand Central

Dispatch, 大中央调度) 并发机制的核心特性。

- ❑ 第 18 章介绍几个次要的知识点, 它们不适合放在本书中的其他位置。

第四部分: 附录

- ❑ 附录 A 提供对编译器具有特殊含义的名称表以及 Objective-C 编译器指令的列表。编译器指令是以 @ 字符开头的单词, 它们是在各种情况下发出给编译器的指令。
- ❑ 附录 B 给出 Foundation 类的列表, 它们的实例具有与低级 C 语言的 Core Foundation 框架中的对应对象相同的内存布局, 并且可能互换使用。
- ❑ 附录 C 简要讨论 32 位和 64 位的环境。
- ❑ 附录 D 描述一个会影响一些面向对象程序设计语言的问题, 以及 Objective-C 如何避免该问题。
- ❑ 附录 E 列出一些对于 Objective-C 开发人员有用的图书和 Web 站点。

读者服务

访问我们的 Web 站点并在 informit.com/register 上注册本书, 就可以方便地访问可能为本书提供的任何更新、下载或勘误表。

致 谢 *Acknowledgements*

任何写过书的人都知道，即使是一位作者写的一本书，那也是团队努力的结果。本书也不例外，Scott D. Yelich、Andy Lee、Matt Long、Cory Bohon 和 Joachim Bean 评阅了本书第 1 版的原稿。Scott 和 Duncan Champney 为本书第 2 版提供了相同的服务。他们不仅发现了错误，还促使我更认真地思考我最初忽视了的一些问题。Steve Peter 激发我开始编写本书，Daniel Steinberg 对于初稿的形成给予了帮助。我要感谢 Addison-Wesley 的 Romny French、Olivia Basegio 以及编辑：Chuck Toporek（第 1 版）和 Trina MacDonald（第 2 版）。由于我初次使用 Microsoft Word 且备受挫折，所以脾气暴躁，但是 Chuck 非常富有同情心并且包容了这一切。

当事情进展不顺利时，每个人都需要富有同情心的倾听。在我编写本书以及准备开始写这本书之前几十年的时间里，我的朋友 Pat O'Brien、Michael Sokoloff 和 Bill Schwartz 每个人都乐于扮演这种倾听的角色。

有两个人值得特别提及：

- Joseph E. Sacco 博士阅读了本书的多份草稿，并且现场测试了其中的练习。Joe 热衷于发现 Objective-C 的一些鲜为人知的方面，并鼓励我探索它们。在编写本书的两个版本期间，他还提供了众所周知的“许多有价值的技术讨论”，以及许多有价值的非技术讨论。
- Ekko Jennings 阅读了一些章节，此外，还提供了精神上的支持和帮助，即便在本该轮到我做饭时她也亲自下厨，并且总是忍受我不停地写作。当我完成了本书第 1 版时，我告诉 Ekko，如果我再一次做像这样的任何事情，她可以对我拍砖。但是她值得我致以深深的谢意，她非常通情达理地制止我这样做，因此我就编写了本书第 2 版。谢谢她，Chérie！

Contents 目 录

第1版赞誉

译者序

前言

致谢

第一部分 Objective-C 简介

第1章 C 语言——Objective-C 的

基础..... 2

1.1 C 程序的结构..... 3

1.1.1 main 例程..... 3

1.1.2 格式化..... 4

1.1.3 注释..... 4

1.1.4 变量和函数名称..... 5

1.1.5 命名约定..... 5

1.1.6 文件..... 6

1.2 变量..... 6

1.2.1 整型..... 7

1.2.2 浮点型..... 8

1.2.3 真值..... 8

1.2.4 初始化..... 8

1.2.5 指针..... 9

1.2.6 数组..... 10

1.2.7 多维数组..... 11

1.2.8 字符串..... 11

1.2.9 结构..... 12

1.2.10 typedef..... 13

1.2.11 枚举常量..... 13

1.3 运算符..... 14

1.3.1 算术运算符..... 14

1.3.2 余数运算符..... 14

1.3.3 递增和递减运算符..... 15

1.3.4 优先级..... 15

1.3.5 取负运算符..... 16

1.3.6 比较运算符..... 16

1.3.7 逻辑运算符..... 16

1.3.8 逻辑“非”运算符..... 17

1.3.9 赋值运算符..... 17

1.3.10 转换和强制转换..... 18

1.3.11 其他赋值运算符..... 19

1.4 表达式和语句..... 19

1.4.1 表达式..... 19

1.4.2 表达式求值..... 20

1.4.3 语句..... 20

1.4.4 复合语句..... 20

1.5 程序流程.....	20	2.5 作用域.....	43
1.5.1 if 语句.....	21	2.5.1 自动变量的作用域.....	43
1.5.2 条件表达式.....	22	2.5.2 复合语句和作用域.....	43
1.5.3 while 语句.....	22	2.5.3 外部变量的作用域.....	44
1.5.4 do-while 循环.....	23	2.6 动态分配.....	44
1.5.5 for 语句.....	23	2.7 小结.....	47
1.5.6 break 语句.....	24	2.8 练习.....	47
1.5.7 continue 语句.....	24		
1.5.8 逗号表达式.....	25	第3章 面向对象程序设计简介.....	49
1.5.9 switch 语句.....	25	3.1 面向对象程序设计.....	49
1.5.10 goto 语句.....	26	3.1.1 类和实例.....	50
1.5.11 函数.....	27	3.1.2 方法.....	50
1.5.12 声明函数.....	29	3.1.3 封装.....	50
1.6 预处理器.....	29	3.1.4 继承.....	51
1.6.1 包括文件.....	29	3.1.5 多态性.....	51
1.6.2 #define.....	30	3.1.6 面向对象语言的主要特点 是什么.....	51
1.6.3 条件编译.....	30	3.2 Objective-C 简介.....	52
1.6.4 printf.....	31	3.2.1 定义类.....	52
1.7 命令行编译和调试.....	33	3.2.2 使用类名作为类型.....	55
1.8 小结.....	35	3.2.3 消息传递(调用方法).....	55
1.9 练习.....	35	3.2.4 类对象和对象创建.....	57
		3.2.5 内存管理.....	58
第2章 关于C变量的更多知识.....	37	3.3 Objective-C 的增补功能.....	58
2.1 C 程序的内存布局.....	37	3.3.1 运行库.....	58
2.2 自动变量.....	38	3.3.2 名称.....	59
2.3 外部变量.....	39	3.3.3 消息表达式.....	59
2.4 声明关键字.....	40	3.3.4 编译器指令.....	59
2.4.1 auto.....	40	3.3.5 字面量字符串.....	60
2.4.2 extern.....	40	3.3.6 Objective-C 的关键字.....	60
2.4.3 static.....	41	3.3.7 框架.....	62
2.4.4 register.....	41	3.3.8 框架数值类型.....	63
2.4.5 const.....	41		
2.4.6 volatile.....	42		

3.4 小结	64	5.7 自省及其他运行库趣事	101
第4章 你的第一个 Objective-C 程序	65	5.8 小结	102
4.1 利用 Xcode 构建程序	65	5.9 练习	102
4.2 Objective-C 的程序结构	69	第6章 类和对象	104
4.3 面向对象的“Hello World”	72	6.1 定义类	104
4.3.1 Greeter.h	73	6.1.1 接口部分	104
4.3.2 Greeter.m	75	6.1.2 @class 指令	106
4.4 main.m	78	6.1.3 实现部分	106
4.5 小结	80	6.1.4 导入	107
4.6 练习	80	6.2 子类化一个类	108
第二部分 语言基础		6.2.1 定义子类	108
第5章 消息传递	84	6.2.2 子类化的示例	108
5.1 方法	84	6.2.3 类层次结构	112
5.1.1 一个简单的方法	84	6.2.4 类层次结构示例	112
5.1.2 带有参数的方法	85	6.2.5 抽象类	113
5.2 消息传递	87	6.3 创建对象	114
5.3 消息传递的细节	89	6.3.1 对象分配	114
5.3.1 嵌套	89	6.3.2 对象初始化	115
5.3.2 消息传递 nil	90	6.4 销毁对象	122
5.3.3 把消息发送给 self	91	6.5 复制对象	123
5.3.4 重写并给 super 发送消息	91	6.5.1 浅复制和深复制	124
5.3.5 选择器	93	6.5.2 可变复制和不可变复制	124
5.3.6 具有相同名称的方法	95	6.5.3 在自己的类中实现复制	125
5.3.7 动态和静态类型化	96	6.6 小结	127
5.4 幕后的工作原理	97	6.7 练习	128
5.5 消息转发	98	第7章 类对象	129
5.6 效率	99	7.1 类对象	129
		7.1.1 Class 类型	130
		7.1.2 类方法	132
		7.2 其他类方法	133

7.2.1 便利构造函数	133	9.3.3 NSMutableString	161
7.2.2 单例	134	9.3.4 字面量字符串	162
7.2.3 初始化类	135	9.4 集合类	162
7.3 模拟类变量	136	9.4.1 NSArray	162
7.4 小结	141	9.4.2 NSDictionary	165
7.5 练习	142	9.4.3 NSSet	166
第8章 框架	144	9.5 NSNumber	167
8.1 框架是什么	145	9.6 NSNull	169
8.2 Cocoa 和 Cocoa Touch	145	9.7 NSData	170
8.2.1 OS X	146	9.7.1 访问 NSData 的字节	170
8.2.2 iOS	146	9.7.2 在文件与 NSData 之间转移 内容	171
8.3 AppKit	147	9.8 NSURL	171
8.4 UIKit	147	9.9 Objective-C 字面量和对象 下标	172
8.5 Core Foundation 框架	148	9.9.1 NSArray 字面量	172
8.5.1 Core Foundation 对象的内存 管理	149	9.9.2 NSDictionary 字面量	173
8.5.2 免费桥接	150	9.9.3 NSNumber 字面量	173
8.6 Core Graphics	151	9.9.4 装箱表达式	174
8.7 Core Animation	152	9.9.5 对象和下标	176
8.8 Apple 提供的其他框架	153	9.9.6 给自己的类添加下标	177
8.9 第三方框架	153	9.10 结构	178
8.10 幕后的工作原理	154	9.11 iOS 上的几何结构	180
8.11 小结	155	9.12 小结	180
第9章 常用的 Foundation 类	156	9.13 练习	180
9.1 不可变类和可变类	156	第10章 Objective-C 中的控制 结构	183
9.2 类簇	157	10.1 if 语句	183
9.3 NSString	158	10.2 for 语句和隐式循环	187
9.3.1 NSString 示例	159	10.2.1 for 语句	187
9.3.2 在 C 字符串与 NSString 之间 来回转换	161	10.2.2 隐式循环	188