

Concepts of Programming Languages, Tenth Edition

PEARSON

编程语言原理

(第10版)

PEARSON

[美] Robert W. Sebesta 著
马跃 王敏 王国栋 译

清华大学出版社

013027846

TP312
934

编程语言原理

(第 10 版)

[美] Robert W. Sebesta

著

马 跃 王 敏 王国栋

译



TP312
934

清华大学出版社

北 京



北航

C1637048

018, 2010

Authorized translation from the English language edition, entitled Concepts of Programming Languages, Tenth Edition, 978-0-13-139531-2 by Robert W. Sebesta, published by Pearson Education, Inc, publishing as Addison-Wesley, Copyright © 2012.

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 TSINGHUA UNIVERSITY PRESS Copyright © 2012.

北京市版权局著作权合同登记号 图字: 01-2012-8974

本书封面贴有 Pearson Education(培生教育出版集团)公司防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

编程语言原理(第10版)/(美)塞巴斯塔(Sebesta, R. W.) 著; 马跃, 王敏, 王国栋 译. —北京: 清华大学出版社, 2013.3

书名原文: Concepts of Programming Languages, Tenth Edition

ISBN 978-7-302-31112-6

I. ①编… II. ①塞… ②马… ③王… ④王… III. ①程序语言 IV. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 308695 号

责任编辑: 王 军 刘伟琴

装帧设计: 牛静敏

责任校对: 成凤进

责任印制: 沈 露

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者: 清华大学印刷厂

装 订 者: 三河市新茂装订有限公司

经 销: 全国新华书店

开 本: 185mm×260mm 印 张: 41 字 数: 998 千字

版 次: 2013 年 3 月第 1 版 印 次: 2013 年 3 月第 1 次印刷

印 数: 1~3000

定 价: 98.00 元

产品编号: 048510-01

前言

第 10 版的变化

本书沿用了前 9 版的写作目标、总体结构和论述手法。本书的首要目标是介绍当代编程语言的主要构成，为读者提供必要的工具来评价已有的和未来的编程语言。另一个目标是，深入讨论编程语言的结构，展示语法描述的形式化方法，介绍词法和语法分析的手段，让读者做好学习编译器设计的准备。

第 10 版与第 9 版相比，有几种不同的变化。为了保持内容的新颖性，删除了对一些过时的编程语言的讨论。例如，第 6 章删除了 COBOL 的记录操作，第 8 章删除了 Fortran 的 Do 语句，第 9 章删除了 Ada 的泛型子程序，第 13 章删除了 Ada 的异步消息传送。

另一方面，第 9 章增加了闭包、间接调用子程序和 F# 中的泛型函数等内容，第 11 章和第 12 章都增加了 Objective-C 一节，第 13 章增加了函数式编程语言中的并发，第 14 章添加了 C# 事件处理一节，第 15 章增加了 F#、基本命令式语言对函数式编程的支持这两节。

有些内容还进行了移动。例如，函数式编程语言中的几个不同结构从第 15 章移到了前面的章节中。其中，函数式编程语言中的控制语句移到了第 8 章，Scheme 和 ML 的列表及列表操作移到了第 6 章。这些移动表示本书理念的重大变化——在某种程度上，是函数式编程语言的某些结构回归主流。在以前的版本中，函数式编程语言的全部结构都放在第 15 章中。

第 11、12 和 15 章也做了大幅修订，第 12 章还增加了 5 个新图。

最后，本书的许多章节都有很多小改动，主要是为了使论述更加清晰。

为了清晰起见，下面列出第 10 版的新增内容：

- 第 5 章：增加了函数式编程语言中的 let 结构。
- 第 6 章：删除了 COBOL 的记录操作，增加了 F# 中的列表、元组和联合等章节。
- 第 8 章：删除了 Fortran 的 Do 语句和 Ada 的 case 语句，函数式编程语言中的控制语句从第 15 章移至该章。
- 第 9 章：增加了闭包、间接调用子程序和 F# 中的泛型函数等节，删除了 Ada 的泛型子程序。
- 第 11 章：增加了 Objective-C 一节，该章的内容做了大幅修订。
- 第 12 章：增加了 Objective-C 一节和 5 个新图。
- 第 13 章：增加了函数式编程语言中的并发，删除了 Ada 的异步消息传送。
- 第 14 章：增加了 C# 事件处理一节。

- 第 15 章：增加了 F#、基本命令式语言对函数式编程的支持这两节，函数式编程语言中的几个不同结构从该章移至前面的章节中。

写作宗旨

本书描述了编程语言的基本原理，讨论了各种语言构成的设计问题，分析了一些常见语言对这些结构所做的设计选择，并细致地比较了各种备选设计方案。

为了严肃认真地研究学习编程语言，需要探讨一些相关的主题，如编程语言语法和语义的形式化描述方法(参见第 3 章)。同时，不同语言构成的实现技术也是需要考虑的：词法和语法分析参见第 4 章，子程序链接的实现参见第 10 章。其他一些语言构成的实现参见本书的其他章节。

下面将概述第 10 版的内容。

各章概要

第 1 章首先介绍了研究程序设计语言的根本原因，接着讨论了用于评价编程语言和语言构成的准则，也探讨了语言设计的主要影响、常见的设计折中以及基本的实现手段。

第 2 章概述了本书讨论的大多数重要语言的发展过程。尽管每种语言都没有叙述得很完整，但讨论了每种语言的起源、目标和贡献。这种历史回顾是有价值的，因为它为理解当代语言设计的实践基础和理论基础提供了必要的背景知识。它也激发了学生深入学习语言设计与评估的兴趣。另外，由于本书的其他章节都不依赖第 2 章，所以它自成体系，可以独立于其他章节进行阅读。

第 3 章介绍了描述编程语言语法的主要形式化方法 BNF。接着讨论了属性文法(用来描述语言的语法和静态语义)。随后探讨了语义描述的难点任务，包括对 3 种最常见方法的简要介绍：操作语义、指称语义以及公理语义。

第 4 章介绍了词法和语法分析。这一章面向课程表中不需要编译器设计课程的大学生。和第 2 章一样，这一章是独立的，可与本书其他部分分开阅读。

第 5 章~第 14 章详细叙述了命令式编程语言的基本构成的设计问题。对每个问题，都以几种语言为例介绍了设计选择，并进行了评估。具体而言，第 5 章介绍了变量的许多特征，第 6 章阐述了数据类型，第 7 章解释了表达式和赋值语句，第 8 章叙述了控制语句，第 9 章和第 10 章讨论了子程序及其实现方式，第 11 章研讨了数据抽象机制，第 12 章深入讨论了支持面向对象编程的语言特征，包括继承和动态方法绑定等，第 13 章讨论了并发程序单元，第 14 章介绍了异常处理，还简要讨论了事件处理。

最后两章(第 15 章和第 16 章)描述了两种最重要的非主流程序设计范型：函数式编程与逻辑编程。但函数式编程语言的一些数据结构和控制语句放在第 6 章和第 8 章讨论。第

15 章介绍了 Scheme, 包括一些基本函数、特殊结构、函数形式以及用 Scheme 写的简单函数示例。对 ML、Haskell 和 F# 的简要介绍展示了一些不同类型的函数式语言设计。第 16 章介绍了逻辑编程和逻辑编程语言 Prolog。

寄语教师

在科罗拉多大学科罗拉多斯普林分校低年级的编程语言课程中, 本书是这样使用的: 我们会详细讲解第 1 章和第 3 章, 而对第 2 章, 尽管学生发现它很有趣, 并从阅读中受益, 但我们在第 2 章上花费的教学时间较少, 因为它没有什么难懂的技术内容。就像前面讲过的, 因为后续各章的内容都不依赖第 2 章, 所以这一章可以完全跳过去。另外, 因为我们有一门编译器设计课程, 所以第 4 章也不在教学安排中。

对于在 C++、Java 或 C# 方面有大量编程经验的学生, 第 5 章~第 9 章的内容是比较容易的。第 10 章~第 14 章则具有一些挑战性, 需要更细致地加以讲解。

第 15 章和第 16 章对大多数低年级学生来说完全是新的。理想情况下, 对于要求学习这些章节内容的学生, 应该给学生提供 Scheme 和 Prolog 的语言处理器。这两章包含了充足的资料, 可以让学生拿一些简单的程序来练习。

本科生课程或许并不能涵盖最后两章的所有内容。不过, 研究生课程应该跳过最初几章中对命令式语言的讨论, 以便完整地讨论这两章的内容。

补充材料

在网站 www.pearsonhighered.com/cssupport 上, 为本书所有读者提供了以下补充材料:

- 一系列教学幻灯片, 包含本书每一章的 PowerPoint 幻灯片。
- 包含本书所有图片的 PowerPoint 幻灯片。

本书的配套网站是 www.pearsonhighered.com/sebesta。这个站点包含关于几种语言的微型手册(约 100 页的教程)。这些手册假定学生掌握了如何使用其他一些语言编程。这些手册为学生提供了足够的信息, 各章内容都有每种语言的完整版本。当前该站点包含 C++、C、Java 和 Smalltalk 的手册。

有资格的教师可以在教师资源中心 www.pearsonhighered.com/irc 上找到许多习题的答案。请联系您所在学校的 Pearson Education 代表, 或访问 www.pearsonhighered.com/irc 进行注册。

获取语言处理器的途径

对于本书讨论的一些编程语言, 可以从下面的站点找到它们的处理器和相关信息:

C、C++、Fortran 和 Ada	gcc.gnu.org
C#和 F#	microsoft.com
Java	java.sun.com
Haskell	haskell.org
Lua	www.lua.org
Scheme	www.plt-scheme.org/software/drscheme
Perl	www.perl.com
Python	www.python.org
Ruby	www.ruby-lang.org

几乎所有浏览器都包含了 JavaScript; 几乎所有 Web 服务器都包含了 PHP。
上述所有信息也都可以在本书的相关网站上找到。

致 谢

对本书目前的形式，出色的审阅人员提供了许多很好的建议。他们是(按字母排序)：

Matthew Michael Burke	
I-ping Chu	德宝大学
Teresa Cole	博伊西州立大学
Pamela Cutter	卡拉马祖学院
Amer Diwan	科罗拉多大学
Stephen Edwards	弗吉尼亚科技大学
David E. Goldschmidt	
Nigel Gwee	南部大学巴顿鲁治分校
Timothy Henry	罗德岛大学
Paul M. Jackowitz	斯克兰顿大学
Duane J. Jarc	马里兰大学
K. N. King	佐治亚州立大学
Donald Kraft	路易斯安那州立大学
Simon H. Lin	加州州立大学北岭分校
Mark Llewellyn	中佛罗里达大学
Bruce R. Maxim	密歇根大学迪尔本分校
Robert McCloskey	斯克兰顿大学
Curtis Meadow	缅因州立大学
Gloria Melara	加州州立大学北岭分校
Frank J. Mitropoulos	诺瓦东南大学
Euripides Montagne	中佛罗里达大学
Serita Nelesen	加尔文学院
Bob Neufeld	卫奇塔州立大学
Charles Nicholas	马里兰大学-巴尔的摩分校
Tim R. Norton	科罗拉多大学斯普林斯分校
Richard M. Osborne	科罗拉多大学丹佛分校
Saverio Perugini	代顿大学
Walter Pharr	查尔斯顿学院
Michael Prentice	纽约州立大学布法罗分校
Amar Raheja	加州理工大学波莫纳分校

Hossein Saiedian	堪萨斯大学
Stuart C. Shapiro	纽约州立大学布法罗分校
Neelam Soundarajan	俄亥俄州立大学
Ryan Stansifer	佛罗里达理工学院
Nancy Tinkham	罗恩大学
Paul Tymann	罗彻斯特理工学院
Cristian Videira Lopes	加州大学欧文分校
Sumanth Yenduri	南密西西比大学
Salih Yurttas	德克萨斯州 A&M 大学

在本书以前版本的不同发展阶段，很多人都提供了写作素材。他们所有的评论对我都很有用，我非常感谢。他们是(按字母排序): Vicki Allan、Henry Bauer、Carter Bays、Manuel E. Bermudez、Peter Brouwer、Margaret Burnett、Paosheng Chang、Liang Cheng、John Crenshaw、Charles Dana、Barbara Ann Griem、Mary Lou Haag、John V. Harrison、Eileen Head、Ralph C. Hilzer、Eric Joanis、Leon Jololian、Hikyoo Koh、Jiang B. Liu、Meiliu Lu、Jon Mauney、Robert McCoard、Dennis L. Mumaugh、Michael G. Murphy、Andrew Oldroyd、Young Park、Rebecca Parsons、Steve J. Phelps、Jeffery Popyack、Raghvinder Sangwan、Steven Rapkin、Hamilton Richard、Tom Sager、Joseph Schell、Sibylle Schupp、Mary Louise Soffa、Neelam Soundarajan、Ryan Stansifer、Steve Stevenson、Virginia Teller、Yang Wang、John M. Weiss、Franck Xia 和 Salih Yurnas.

还要感谢编辑 Matt Goldstein、编辑助理 Chelsea Kharakozova、Addison-Wesley 的高级产品主管 Marilyn Lloyd 和 Aardvark 出版服务组的 Gillian Hall，他们为快速而又严谨地出版第 10 版付出了极大的努力。

目 录

第 1 章 预备知识	1	2.2.2 Speedcoding 系统	31
1.1 学习程序设计语言原理的原因	2	2.2.3 UNIVAC “编译”系统	32
1.2 程序设计领域	4	2.2.4 相关工作	32
1.2.1 科学应用	4	2.3 IBM 704 计算机与	
1.2.2 商务应用	4	Fortran 语言	32
1.2.3 人工智能	5	2.3.1 历史背景	32
1.2.4 系统程序设计	5	2.3.2 设计过程	33
1.2.5 网络软件	5	2.3.3 Fortran I 概述	33
1.3 语言评价标准	6	2.3.4 Fortran II	34
1.3.1 可读性	6	2.3.5 Fortran IV、77、90、95、	
1.3.2 可写性	10	2003 和 2008	34
1.3.3 可靠性	11	2.3.6 评价	35
1.3.4 成本	13	2.4 函数式程序设计: LISP 语言	36
1.4 影响语言设计的因素	14	2.4.1 人工智能的起源和表处理	36
1.4.1 计算机体系结构	14	2.4.2 LISP 语言的设计过程	37
1.4.2 程序设计方法学	15	2.4.3 语言概述	37
1.5 程序设计语言的分类	16	2.4.4 评价	39
1.6 语言设计中的权衡	17	2.4.5 LISP 的两种后代语言	39
1.7 实现方法	18	2.4.6 相关语言	40
1.7.1 编译	19	2.5 迈向成熟的第一步:	
1.7.2 完全解释	21	ALGOL 60	41
1.7.3 混合实现系统	22	2.5.1 历史背景	41
1.7.4 预处理器	23	2.5.2 早期设计过程	41
1.8 编程环境	23	2.5.3 ALGOL 58 概述	42
第 2 章 主要程序设计语言的发展	27	2.5.4 对 ALGOL 58 报告的响应	42
2.1 Zuse 的 Plankalkül 语言	29	2.5.5 ALGOL 60 的设计过程	43
2.1.1 历史背景	29	2.5.6 ALGOL 60 概述	43
2.1.2 语言概述	29	2.5.7 评价	44
2.2 伪代码	30	2.6 商务记录的计算机化:	
2.2.1 Short Code 语言	31	COBOL 语言	45
		2.6.1 历史背景	46

2.6.2	FLOW-MATIC 语言	46	2.14	历史上规模最大的设计工作:	
2.6.3	COBOL 语言的设计过程	46		Ada 语言	62
2.6.4	评价	47	2.14.1	历史背景	62
2.7	分时处理的开始:		2.14.2	设计过程	62
	BASIC 语言	49	2.14.3	语言概述	63
2.7.1	设计过程	49	2.14.4	评价	64
2.7.2	语言概述	50	2.14.5	Ada 95 和 Ada 2005	65
2.7.3	评价	50	2.15	面向对象的程序设计:	
2.8	满足所有人的需要: PL/I	51		Smalltalk	65
2.8.1	历史背景	51	2.15.1	设计过程	65
2.8.2	设计过程	52	2.15.2	语言概述	66
2.8.3	语言概述	53	2.15.3	评价	67
2.8.4	评价	53	2.16	结合命令式和面向对象的特性:	
2.9	两种早期的动态语言:			C++	67
	APL 和 SNOBOL	54	2.16.1	设计过程	68
2.9.1	APL 语言的起源与特点	54	2.16.2	语言概述	68
2.9.2	SNOBOL 语言的起源与特点	55	2.16.3	评价	69
2.10	数据抽象的开始:		2.16.4	一种相关语言:	
	SIMULA 67	55		Objective-C	69
2.10.1	设计过程	55	2.16.5	另一种相关语言: Delphi	69
2.10.2	语言概述	55	2.16.6	一种关系不大的语言: Go	70
2.11	正交设计: ALGOL 68	56	2.17	基于命令式的面向对象语言:	
2.11.1	设计过程	56		Java	70
2.11.2	语言概述	56	2.17.1	设计过程	70
2.11.3	评价	56	2.17.2	语言概述	71
2.12	ALGOL 系列语言的		2.17.3	评价	72
	早期后代语言	57	2.18	脚本语言	73
2.12.1	为简单性而设计:		2.18.1	Perl 的起源与特点	73
	Pascal 语言	57	2.18.2	JavaScript 的起源与特点	75
2.12.2	可移植的系统语言:		2.18.3	PHP 的起源与特点	76
	C 语言	59	2.18.4	Python 的起源与特点	77
2.13	基于逻辑的程序设计:		2.18.5	Ruby 的起源与特点	77
	Prolog 语言	60	2.18.6	Lua 的起源与特点	78
2.13.1	设计过程	61	2.19	一流的.NET 语言: C#	79
2.13.2	语言概述	61	2.19.1	设计过程	79
2.13.3	评价	61	2.19.2	语言概述	79

2.19.3 评价.....	80	4.4.1 递归下降的语法分析过程.....	144
2.20 标记与程序设计混合的语言.....	81	4.4.2 LL 文法类.....	149
2.20.1 XSLT.....	81	4.5 自下而上的语法分析.....	152
2.20.2 JSP.....	82	4.5.1 自下而上的语法分析器的 分析问题.....	152
第 3 章 描述语法和语义	89	4.5.2 移进-归约算法.....	154
3.1 概述.....	90	4.5.3 LR 语法分析器.....	155
3.2 描述语法的普遍问题.....	91	第 5 章 名字、绑定和作用域	163
3.2.1 语言识别器.....	91	5.1 概述.....	164
3.2.2 语言生成器.....	92	5.2 名字.....	164
3.3 描述语法的形式化方法.....	92	5.2.1 设计问题.....	165
3.3.1 巴科斯-诺尔范式和 上下文无关文法.....	92	5.2.2 名字形式.....	165
3.3.2 扩展的 BNF.....	102	5.2.3 特殊字.....	166
3.3.3 文法与识别器.....	104	5.3 变量.....	166
3.4 属性文法.....	104	5.3.1 名字.....	167
3.4.1 静态语义.....	105	5.3.2 地址.....	167
3.4.2 基本概念.....	105	5.3.3 类型.....	168
3.4.3 属性文法定义.....	105	5.3.4 数值.....	168
3.4.4 本质属性.....	106	5.4 绑定的概念.....	168
3.4.5 属性文法的例子.....	106	5.4.1 属性与变量绑定.....	169
3.4.6 计算属性值.....	108	5.4.2 绑定类型.....	169
3.4.7 评价.....	109	5.4.3 存储绑定和生存期.....	172
3.5 描述程序的意义: 动态语义.....	109	5.5 作用域.....	175
3.5.1 操作语义.....	110	5.5.1 静态作用域.....	175
3.5.2 指称语义.....	112	5.5.2 块.....	177
3.5.3 公理语义.....	116	5.5.3 声明的次序.....	179
第 4 章 词法分析和语法分析	133	5.5.4 全局作用域.....	180
4.1 概述.....	134	5.5.5 静态作用域的评价.....	182
4.2 词法分析.....	135	5.5.6 动态作用域.....	183
4.3 语法分析问题.....	142	5.5.7 动态作用域的评价.....	183
4.3.1 语法分析概述.....	142	5.6 作用域和生存期.....	184
4.3.2 自顶向下的语法分析器.....	142	5.7 引用环境.....	185
4.3.3 自底向上的语法分析器.....	143	5.8 命名常量.....	186
4.3.4 语法分析的复杂度.....	144	第 6 章 数据类型	195
4.4 递归下降的语法分析.....	144	6.1 概述.....	196

6.2	基本数据类型	197	6.10.2	判别式联合与自由联合	227
6.2.1	数值类型	197	6.10.3	Ada 的联合类型	228
6.2.2	布尔类型	200	6.10.4	F#的联合类型	229
6.2.3	字符类型	200	6.10.5	评价	230
6.3	字符串类型	200	6.10.6	联合类型的实现	230
6.3.1	设计问题	201	6.11	指针和引用类型	231
6.3.2	字符串及其操作	201	6.11.1	设计问题	232
6.3.3	字符串长度的设计选项	203	6.11.2	指针操作	232
6.3.4	评价	203	6.11.3	指针的相关问题	233
6.3.5	字符串类型的实现	203	6.11.4	Ada 中的指针	234
6.4	用户定义的序数类型	204	6.11.5	C 和 C++中的指针	235
6.4.1	枚举类型	205	6.11.6	引用类型	236
6.4.2	子界类型	207	6.11.7	评价	237
6.4.3	实现用户定义的有序类型	208	6.11.8	指针和引用类型的实现	237
6.5	数组类型	208	6.12	类型检查	242
6.5.1	设计问题	208	6.13	强类型化	242
6.5.2	数组和索引	209	6.14	类型等价	243
6.5.3	下标的绑定和数组的种类	210	6.15	理论和数据类型	246
6.5.4	数组的初始化	212	第 7 章	表达式与赋值语句	257
6.5.5	数组操作	213	7.1	概述	258
6.5.6	矩形数组和不规则数组	215	7.2	算术表达式	258
6.5.7	切片	215	7.2.1	运算符的运算顺序	259
6.5.8	评价	216	7.2.2	操作数的运算顺序	264
6.5.9	数组类型的实现	216	7.3	运算符重载	266
6.6	关联数组	218	7.4	类型转换	267
6.6.1	结构和操作	219	7.4.1	表达式中的强制类型转换	267
6.6.2	关联数组的实现	220	7.4.2	显式类型转换	269
6.7	记录类型	220	7.4.3	表达式中的错误	269
6.7.1	记录的定义	221	7.5	关系表达式和布尔表达式	270
6.7.2	记录域的引用	222	7.5.1	关系表达式	270
6.7.3	评价	222	7.5.2	布尔表达式	270
6.7.4	记录类型的实现	223	7.6	短路求值	271
6.8	元组类型	223	7.7	赋值语句	273
6.9	列表类型	224	7.7.1	简单赋值	273
6.10	联合类型	227	7.7.2	条件赋值	273
6.10.1	设计问题	227	7.7.3	混合赋值运算符	273

7.7.4 一元赋值运算符	274	9.5.5 参数的类型检查	332
7.7.5 赋值表达式	275	9.5.6 多维数组作为参数	333
7.7.6 多重赋值	276	9.5.7 设计考虑	336
7.7.7 函数式编程语言中的赋值	276	9.5.8 参数传递的例子	336
7.8 混合模式赋值	277	9.6 子程序作为参数	339
第 8 章 语句级控制结构	283	9.7 间接调用子程序	340
8.1 概述	284	9.8 重载子程序	342
8.2 选择语句	285	9.9 泛型子程序	343
8.2.1 双路选择语句	285	9.9.1 C++中的泛型函数	343
8.2.2 多重选择结构	289	9.9.2 Java 5.0 中的泛型方法	345
8.3 迭代语句	295	9.9.3 C# 2005 中的泛型方法	346
8.3.1 计数控制循环	296	9.9.4 F#中的泛型函数	347
8.3.2 逻辑控制循环	300	9.10 函数的设计问题	348
8.3.3 用户自定义的循环 控制机制	302	9.10.1 函数的副作用	348
8.3.4 基于数据结构的迭代	303	9.10.2 返回值的类型	348
8.4 无条件分支	306	9.10.3 返回值的个数	348
8.5 防护命令	307	9.11 用户定义重载运算符	349
8.6 结论	309	9.12 闭包	349
第 9 章 子程序	315	9.13 协同程序	351
9.1 概述	316	第 10 章 实现子程序	359
9.2 子程序的基本原理	316	10.1 调用和返回的一般语义	360
9.2.1 子程序的一般性质	316	10.2 实现“简单”的子程序	360
9.2.2 子程序的基本定义	316	10.3 通过栈动态局部变量实现 子程序	362
9.2.3 参数	318	10.3.1 更复杂的活动记录	363
9.2.4 过程与函数	321	10.3.2 一个不含递归 调用的例子	365
9.3 子程序的设计问题	322	10.3.3 递归调用	367
9.4 局部引用环境	323	10.4 嵌套子程序	368
9.4.1 局部变量	323	10.4.1 基础知识	368
9.4.2 嵌套子程序	324	10.4.2 静态链	369
9.5 参数传递方式	325	10.5 块	373
9.5.1 参数传递的语义模型	325	10.6 动态作用域的实现	375
9.5.2 参数传递的实现模型	326	10.6.1 深层访问	375
9.5.3 参数传递方法的实现	329	10.6.2 浅层访问	376
9.5.4 常见语言的参数传递方法	330		

第 11 章 抽象数据类型与封装结构	383
11.1 抽象的概念	384
11.2 数据抽象简介	385
11.2.1 抽象数据类型之浮点型	385
11.2.2 用户自定义的抽象数据类型	386
11.2.3 示例	387
11.3 抽象数据类型的设计问题	387
11.4 语言示例	388
11.4.1 Ada 中的抽象数据类型	388
11.4.2 C++中的抽象数据类型	391
11.4.3 Objective-C 中的抽象数据类型	395
11.4.4 Java 中的抽象数据类型	400
11.4.5 C#中的抽象数据类型	401
11.4.6 Ruby 中的抽象数据类型	403
11.5 参数化的抽象数据类型	406
11.5.1 Ada	406
11.5.2 C++	407
11.5.3 Java 5.0	409
11.5.4 C# 2005	411
11.6 封装结构	411
11.6.1 引言	411
11.6.2 C 中的封装	412
11.6.3 C++中的封装	412
11.6.4 Ada 包	413
11.6.5 C#程序集	413
11.7 命名封装	414
11.7.1 C++命名空间	414
11.7.2 Java 包	415
11.7.3 Ada 包	416
11.7.4 Ruby 模块	417

第 12 章 面向对象程序设计的支持	425
12.1 概述	426
12.2 面向对象编程	426
12.2.1 引言	426
12.2.2 继承	426
12.2.3 动态绑定	428
12.3 面向对象语言的设计问题	430
12.3.1 对象的排他性	430
12.3.2 子类是子类型吗	430
12.3.3 单继承与多继承	431
12.3.4 对象的分配和释放	432
12.3.5 动态绑定与静态绑定	433
12.3.6 嵌套类	433
12.3.7 对象的初始化	433
12.4 Smalltalk 对面向对象编程的支持	434
12.4.1 一般特征	434
12.4.2 继承	434
12.4.3 动态绑定	435
12.4.4 Smalltalk 的评价	435
12.5 C++对面向对象编程的支持	436
12.5.1 一般特征	436
12.5.2 继承	436
12.5.3 动态绑定	441
12.5.4 评价	443
12.6 Objective-C 对面向对象编程的支持	444
12.6.1 一般特征	444
12.6.2 继承	445
12.6.3 动态绑定	446
12.6.4 评价	447
12.7 Java 对面向对象编程的支持	447
12.7.1 一般特征	447
12.7.2 继承	447
12.7.3 动态绑定	449

12.7.4 被嵌套的类	449	13.3.2 合作同步	476
12.7.5 评价	450	13.3.3 竞争同步	478
12.8 C#对面向对象编程的支持	450	13.3.4 评价	479
12.8.1 一般特征	450	13.4 管程	479
12.8.2 继承	450	13.4.1 概述	479
12.8.3 动态绑定	450	13.4.2 竞争同步	480
12.8.4 被嵌套的类	451	13.4.3 合作同步	480
12.8.5 评价	451	13.4.4 评价	480
12.9 Ada 95 对面向对象编程的 支持	452	13.5 消息传递	481
12.9.1 一般特征	452	13.5.1 概述	481
12.9.2 继承	452	13.5.2 同步消息传递的原理	481
12.9.3 动态绑定	453	13.6 Ada 对并发的支持	482
12.9.4 子包	454	13.6.1 基本原理	482
12.9.5 评价	455	13.6.2 合作同步	485
12.10 Ruby 对面向对象编程的 支持	455	13.6.3 竞争同步	486
12.10.1 一般特征	455	13.6.4 任务终止	487
12.10.2 继承	457	13.6.5 优先级	487
12.10.3 动态绑定	457	13.6.6 受保护对象	488
12.10.4 评价	457	13.6.7 评价	489
12.11 面向对象构造的实现	457	13.7 Java 线程	489
12.11.1 存储实例数据	457	13.7.1 Thread 类	490
12.11.2 方法调用到方法的 动态绑定	458	13.7.2 优先级	491
第 13 章 并发	467	13.7.3 信号量	492
13.1 概述	468	13.7.4 竞争同步	492
13.1.1 多处理器体系结构	469	13.7.5 合作同步	493
13.1.2 并发的种类	470	13.7.6 非阻塞同步	496
13.1.3 使用并发的目的	471	13.7.7 显式锁定	496
13.2 子程序级并发概述	471	13.7.8 评价	497
13.2.1 基本概念	471	13.8 C#线程	497
13.2.2 为并发而设计的语言	475	13.8.1 基本线程操作	497
13.2.3 设计问题	475	13.8.2 同步线程	500
13.3 信号量	475	13.8.3 评价	500
13.3.1 概述	475	13.9 函数式语言中的并发	501
		13.9.1 Multilisp	501
		13.9.2 并发 ML	501
		13.9.3 F#	502
		13.10 语句级并发	503

第 14 章 异常处理和事件处理	511
14.1 异常处理概述	512
14.1.1 基本概念	513
14.1.2 设计问题	514
14.2 Ada 中的异常处理	517
14.2.1 异常处理程序	517
14.2.2 将异常绑定到 处理程序	518
14.2.3 继续	518
14.2.4 其他设计选择	519
14.2.5 例子	520
14.2.6 评价	522
14.3 C++中的异常处理	522
14.3.1 异常处理程序	522
14.3.2 异常与处理 程序的绑定	523
14.3.3 继续	524
14.3.4 其他设计选择	524
14.3.5 例子	524
14.3.6 评价	526
14.4 Java 中的异常处理	526
14.4.1 异常类	526
14.4.2 异常处理程序	527
14.4.3 异常与处理 程序的绑定	527
14.4.4 其他设计选择	528
14.4.5 例子	529
14.4.6 finally 子句	530
14.4.7 断言	531
14.4.8 评价	531
14.5 事件处理概述	532
14.6 Java 的事件处理	533
14.6.1 Java Swing 的 GUI 组件	533
14.6.2 Java 事件模型	534
14.7 C#中的事件处理	537

第 15 章 函数式程序设计语言	545
15.1 概述	546
15.2 数学函数	547
15.2.1 简单函数	547
15.2.2 函数形式	548
15.3 函数式程序设计语言基础	549
15.4 第一种函数式程序设计 语言 LISP	550
15.4.1 数据类型和结构	550
15.4.2 第一个 LISP 解释器	551
15.5 Scheme 概述	553
15.5.1 Scheme 的起源	553
15.5.2 Scheme 解释器	553
15.5.3 基本数值函数	553
15.5.4 定义函数	554
15.5.5 输出函数	556
15.5.6 数值谓词函数	556
15.5.7 控制流	557
15.5.8 表函数	557
15.5.9 用于符号原子和表的 谓词函数	560
15.5.10 Scheme 函数示例	561
15.5.11 LET	564
15.5.12 Scheme 中的尾递归	565
15.5.13 函数形式	566
15.5.14 构建代码的函数	567
15.6 Common LISP	568
15.7 ML	570
15.8 Haskell	574
15.9 F#	578
15.10 基本命令式语言对函数式 编程的支持	581
15.11 函数式语言和命令式语言的 比较	582
第 16 章 逻辑程序设计语言	589
16.1 概述	590