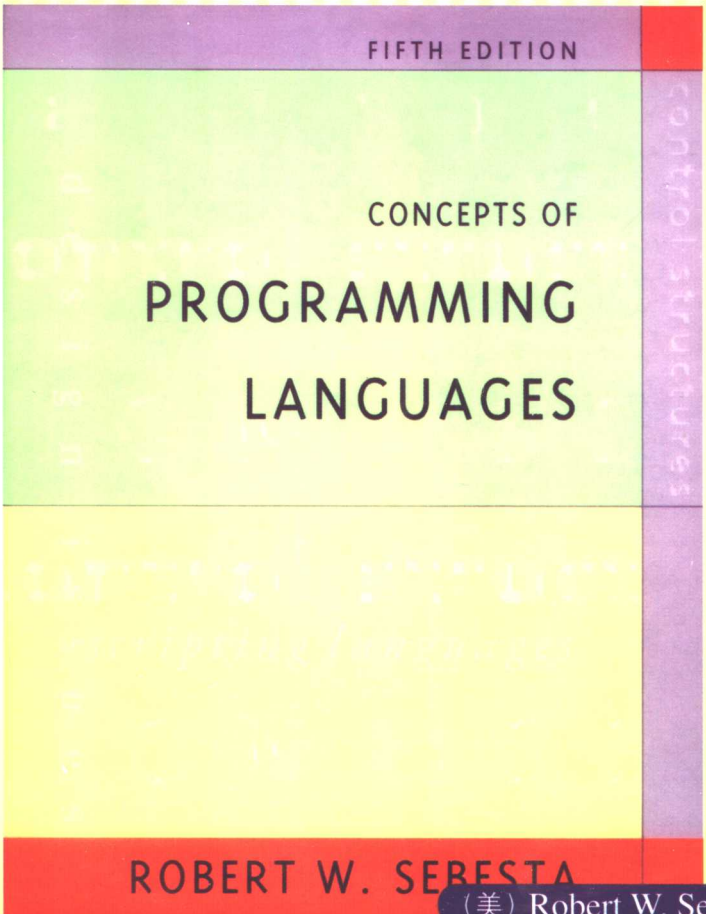
经 典 原 版 书 库

程序设计语言原理

（英文版 · 第5版）



（美）Robert W. Sebesta 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

程序设计语言原理

(英文版 · 第5版)

Concepts of Programming Languages

Fifth Edition

江苏工业学院图书馆
藏书章

(美) Robert W. Sebesta 著

燕山大学图书馆藏书
(561)
TP311.1/X4

机械工业出版社

0398280
K05

English reprint Copyright © 2002 by PEARSON EDUCATION NORTH ASIA LTD and China Machine Press.

Original English language title: Concepts of Programming Languages, Fifth Edition by Robert W. Sebesta, Copyright © 2002. ISBN 0-201-75295-6.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison Wesley.

This edition is authorized for sale only in People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macau).

本书封面贴有Pearson Education培生教育出版集团激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书版权登记号：图字：01-2002-1838

图书在版编目（CIP）数据

程序设计语言原理（英文版·第5版）/（美）谢拜什陶（Sebesta, R. W.）著. -北京：机械工业出版社，2003.1

（经典原版书库）

书名原文：Concepts of Programming Languages, Fifth Edition

ISBN 7-111-10161-8

Ⅰ.程… Ⅱ.谢… Ⅲ.程序语言-英文 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第092132号

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：华 章

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2003年1月第1版第1次印刷

850mm×1168mm 1/32·22.625印张

印数：0 001-3 000册

定价：45.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：针对本科生的核心课程，剔抉外版菁华而成“国外经典教材”系列；对影印版的教材，则单独开辟出“经典原版书库”；定位在高级教程和专业参考的“计算机科学丛书”还将保持原来的风格，继续出版新的品种。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

“经典原版书库”是响应教育部提出的使用原版国外教材的号召，为国内高校的计算机教学度身订造的。在广泛地征求并听取丛书的“专家指导委员会”的意见后，我们最终选定了这30多种篇幅内容适度、讲解鞭辟入里的教材，其中的大部分已经被M.I.T.、Stanford、U.C. Berkley、C.M.U.等世界名牌大学采用。丛书不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzedu@hzbook.com
联系电话：（010）68995265
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

此为试读，需要完整PDF请访问：www.ertongbook.com

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

Preface

The goals, overall structure, and approach of this fifth edition of *Concepts of Programming Languages* remain the same as those of the four earlier editions. The principal goal is to provide the reader with the tools necessary for the critical evaluation of existing and future programming languages. An additional goal is to prepare the reader for the study of compiler design.

The fifth edition is an updated version of the fourth, with the addition of a chapter on lexical and syntax analysis, as well as descriptions of some interesting features of JavaScript. The discussion of Java threads has been revised to reflect the changes made in later versions of that language. In many places, material on older languages such as ALGOL 68 and Modula-2 has either been removed or replaced by similar material in contemporary languages such as C++, Java, and Perl.

This book describes the fundamental concepts of programming languages by discussing the design issues of the various language constructs, examining the design choices for these constructs in some of the most common languages, and critically comparing design alternatives.

Any serious study of programming languages requires an examination of some related topics, among which are formal methods of describing the syntax and semantics of programming languages. Also, implementation techniques for various language constructs must be considered: syntax and semantics are the topics of Chapter 3, lexical and syntax analysis are discussed in Chapter 4, and implementation of subprogram linkage is the topic of Chapter 10. Implementation of some other language constructs is discussed in various other parts of the book.

The following paragraphs outline the contents of the fifth edition:

Chapter 1 begins with a rationale for studying programming languages. It then discusses the criteria used for evaluating programming languages and language constructs. The primary influences on language design, common design trade-offs, and the basic approaches to implementation are also examined.

Chapter 2 outlines the evolution of most of the important languages discussed in this book. Although no language is described completely, the origins, purposes, and contributions of each are discussed. This historical overview is valuable because it provides the background necessary to understanding the practical and theoretical basis for contemporary language

design. It also motivates further study of language design and evaluation. In addition, because none of the remainder of the book depends on Chapter 2, it can be read on its own, independent of the other chapters.

Chapter 3 describes the primary formal method for describing the syntax of programming language, BNF. This is followed by a description of attribute grammars, which play a prominent role in compiler design. The difficult task of semantic description is then explored, including brief introductions to the three most common methods: operational, axiomatic, and denotational semantics.

Chapter 4, which is new, introduces lexical and syntax analysis. This chapter is targeted to those colleges that no longer require a compiler design course in their curricula. None of the rest of the book depends on the material in Chapter 4.

Chapters 5 through 14 describe in detail the design issues for the primary constructs of the imperative languages. In each case, the design choices for several example languages are presented and evaluated. Specifically, Chapter 5 covers the many characteristics of variables, Chapter 6 covers data types, and Chapter 7 explains expressions and assignment statements. Chapter 8 describes control statements, Chapters 9 and 10 look at subprograms and their implementation, and Chapter 11 examines data abstraction facilities. Chapter 12 is about language features that support object-oriented programming (inheritance and dynamic method binding), Chapter 13, about concurrent program units, and Chapter 14 is about exception handling.

The last two chapters (15 and 16) describe two of the most important alternative programming paradigms: functional programming and logic programming. Chapter 15 presents an introduction to Scheme, including descriptions of some of its primitive functions, special forms, and functional forms, as well as some examples of simple functions written in Scheme. Brief introductions to COMMON LISP, ML, and Haskell are given to illustrate some different kinds of functional language. Chapter 16 introduces logic programming and the logic programming language, Prolog.

To the Instructor

In the junior-level programming language course at the University of Colorado at Colorado Springs, the book is used as follows: We typically cover Chapters 1 and 3 in detail, and though students find it interesting and beneficial reading, Chapter 2 receives little lecture time due to its lack of hard technical content. Because no material in subsequent chapters depends on Chapter 2, as noted earlier, it can be skipped entirely, and because we require a course in compiler design, Chapter 4 is not covered.

Chapters 5 through 9 and 11 should be relatively easy for students with extensive programming experience in C++, Ada, or Java. Chapters 10, 12, 13, and 14 are more challenging and require more detailed lectures.

Chapters 15 and 16 are entirely new to most students at the junior level. Ideally, language processors for Scheme and Prolog should be available for students required to learn the material in these chapters. Sufficient material is included to allow students to dabble with some simple programs.

Undergraduate courses will probably not be able to cover all of the last two chapters in detail. Graduate courses, however, by skipping over parts of the early chapters on imperative languages, will be able to completely discuss the nonimperative languages.

Supplements

1. Solutions to many of the problem sets are exclusively available to professors teaching a course. To obtain these please call your local AW sales representative.
2. A set of lecture notes slides is also available. These slides are in the form of Microsoft PowerPoint source files, one for each of the first 15 chapters of the book. They were developed over the past few years in teaching a course based on the book.
3. PowerPoint slides of all the figures in the book are included in the supplement package.
4. Please check online information for this book at www.aw.com/cssupport for more information on obtaining these supplements.

Language Processor Availability

Processors for and information about some of the programming languages discussed in this book can be found at the following Web sites:

Java	http://java.sun.com
Haskell	http://haskell.org
Scheme	http://www.cs.rice.edu/CS/PLT/packages/drscheme/
Perl	http://www.perl.com

Acknowledgements

The suggestions from outstanding reviewers contributed greatly to this book's present form. In alphabetical order, ther are:

- Carter Bays, *University of South Carolina*
- Manuel E. Bermudez, *University of Florida*
- Margaret Burnett, *Oregon State University*
- Eileen Head, *Binghampton University*
- Ralph C. Hilzer, *California State University, Chico*
- Jiang B. Liu, *Bradley University*
- Meiliu Lu, *California State University, Sacramento*
- Bruce R. Maxim, *University of Michigan, Dearborn*
- John M. Weiss, *South Dakota School of Mines and Technology*

Numerous other people provided input for the previous editions of *Concepts of Programming Languages* at various stages of its development. All of their comments were useful and greatly appreciated. In alphabetical order, they are: Vicki Allan, Henry Bauer, Peter Brouwer, Paosheng Chang, John Crenshaw, Barbara Ann Griem, Mary Lou Haag, Hikyoo Koh, Jon Mauney, Robert McCoard, Michael G. Murphy, Andrew Oldroyd, Rebecca Parsons, Jeffery Popyack, Steven Rapkin, Hamilton Richard, Tom Sager, Joseph Schell, and Mary Louise Soffa.

Maite Suarez-Rivas, Editor, Katherine Harutunian, Project Editor, and Juliet Silveri, Production Supervisor at Addison-Wesley and Daniel Rausch at Argosy, all deserve my gratitude for their efforts to produce the fifth edition quickly, as well as help make it significantly more complete than the fourth.

Finally, I thank my children, Jake and Darcie, for their patience in enduring my absence from them throughout the endless hours of effort I invested in writing the five editions of this book.

About the Author

Robert Sebesta is an Associate Professor and Chairman of the Computer Science Department at the University of Colorado, Colorado Springs. Professor Sebesta received a B.S. in applied mathematics from the University of Colorado in Boulder and his M.S. and Ph.D. degrees in Compuer Science from the Pennsylvania State University. He has taught computer science for over 30 years. His professional interests are the design and evaluation of programming languages, compiler design, and software testing methods and tools. He is a member of the ACM and the IEEE Computer Society.

FIFTH EDITION

Concepts of Programming Languages

此为试读,需要完整PDF请访问: www.ertongbook.com

Contents

Chapter 1 Preliminaries 1

1.1 Reasons for Studying Concepts of Programming Languages 2
1.2 Programming Domains 5
1.3 Language Evaluation Criteria 8
1.4 Influences on Language Design 20
1.5 Language Categories 23
1.6 Language Design Trade-Offs 24
1.7 Implementation Methods 25
1.8 Programming Environments 31

Chapter 2 Evolution of the Major Programming Languages 37

2.1 Zuse's Plankalkül 38
2.2 Minimal Hardware Programming: Pseudocodes 41
2.3 The IBM 704 and FORTRAN 44
2.4 Functional Programming: LISP 49
2.5 The First Step Toward Sophistication: ALGOL 60 55
2.6 Computerizing Business Records: COBOL 61
2.7 The Beginnings of Timesharing: BASIC 66
2.8 Everything for Everybody: PL/I 68
2.9 Two Early Dynamic Languages: APL and SNOBOL 73
2.10 The Beginnings of Data Abstraction: SIMULA 67 74
2.11 Orthogonal Design: ALGOL 68 75
2.12 Some Important Descendants of the ALGOLs 77
2.13 Programming Based on Logic: Prolog 84
2.14 History's Largest Design Effort: Ada 85
2.15 Object-Oriented Programming: Smalltalk 91
2.16 Combining Imperative and Object-Oriented Features: C++ 94
2.17 Programming the World Wide Web: Java 97

Chapter 3 Describing Syntax and Semantics 105

3.1 Introduction 106
3.2 The General Problem of Describing Syntax 107
3.3 Formal Methods of Describing Syntax 109
3.4 Attribute Grammars 123
3.5 Describing the Meanings of Programs: Dynamic Semantics 129

Chapter 4 Lexical and Syntax Analysis 153

4.1 Introduction 154
4.2 Lexical Analysis 155
4.3 The Parsing Problem 159
4.4 Recursive-Descent Parsing 162
4.5 Bottom-Up Parsing 167

Chapter 5 Names, Bindings, Type Checking, and Scopes 179

5.1 Introduction 180
5.2 Names 181
5.3 Variables 183
5.4 The Concept of Binding 186
5.5 Type Checking 193
5.6 Strong Typing 194
5.7 Type Compatibility 196
5.8 Scope 199
5.9 Scope and Lifetime 207
5.10 Referencing Environments 207
5.11 Named Constants 209
5.12 Variable Initialization 211

Chapter 6 Data Types 219

6.1 Introduction 220
6.2 Primitive Data Types 221
6.3 Character String Types 225
6.4 User-Defined Ordinal Types 230
6.5 Array Types 234
5.6 Associative Arrays 246
6.7 Record Types 248

6.8 Union Types 252
6.9 Set Types 258
6.10 Pointer Types 260

Chapter 7 Expressions and Assignment Statements 281

7.1 Introduction 282
7.2 Arithmetic Expressions 283
7.3 Overloaded Operators 291
7.4 Type Conversions 293
7.5 Relational and Boolean Expressions 296
7.6 Short-Circuit Evaluation 298
7.7 Assignment Statements 300
7.8 Mixed-Mode Assignment 304

Chapter 8 Statement-Level Control Structures 309

8.1 Introduction 310
8.2 Compound Statements 311
8.3 Selection Statements 312
8.4 Iterative Statements 324
8.5 Unconditional Branching 337
8.6 Guarded Commands 339
8.7 Conclusions 341

Chapter 9 Subprograms 349

9.1 Introduction 350
9.2 Fundamentals of Subprograms 350
9.3 Design Issues for Subprograms 356
9.4 Local Referencing Environments 357
9.5 Parameter-Passing Methods 358
9.6 Parameters That Are Subprogram Names 377
9.7 Overloaded Subprograms 380
9.8 Generic Subprograms 381
9.9 Separate and Independent Compilation 385
9.10 Design Issues for Functions 387
9.11 Accessing Nonlocal Environments 388
9.12 User-Defined Overloaded Operators 390
9.13 Coroutines 391

Chapter 10 Implementing Subprograms 399

10.1 The General Semantics of Calls and Returns 400
10.2 Implementing FORTRAN 77 Subprograms 401
10.3 Implementing Subprograms in ALGOL-like Languages 403
10.4 Blocks 421
10.5 Implementing Dynamic Scoping 422
10.6 Implementing Parameters That Are Subprogram Names 426

Chapter 11 Abstract Data Types 433

11.1 The Concept of Abstraction 434
11.2 Encapsulation 435
11.3 Introduction to Data Abstraction 436
11.4 Design Issues 439
11.5 Language Examples 440
11.6 Parameterized Abstract Data Types 450

Chapter 12 Support for Object-Oriented Programming 457

12.1 Introduction 458
12.2 Object-Oriented Programming 458
12.3 Design Issues for Object-Oriented Languages 463
12.4 Overview of Smalltalk 468
12.5 Introduction to the Smalltalk Language 469
12.6 Smalltalk Example Programs 480
12.7 Large-Scale Features of Smalltalk 485
12.8 Evaluation of Smalltalk 487
12.9 Support for Object-Oriented Programming in C++ 488
12.10 Support for Object-Oriented Programming in Java 496
12.11 Support for Object-Oriented Programming in Ada 95 498
12.12 Support for Object-Oriented Programming in Eiffel 502
12.13 The Object Model of JavaScript 505
12.14 Implementation of Object-Oriented Constructs 508

Chapter 13 Concurrency 515

13.1 Introduction 516
13.2 Introduction to Subprogram-Level Concurrency 519
13.3 Semaphores 523
13.4 Monitors 528

13.5 Message Passing 533
13.6 Concurrency in Ada 95 543
13.7 Java Threads 546
13.8 Statement-Level Concurrency 551

Chapter 14 Exception Handling 557

14.1 Introduction to Exception Handling 558
14.2 Exception Handling in PL/I 564
14.3 Exception Handling in Ada 569
14.4 Exception Handling in C++ 575
14.5 Exception Handling in Java 579

Chapter 15 Functional Programming Languages 591

15.1 Introduction 592
15.2 Mathematical Functions 593
15.3 Fundamentals of Functional Programming Languages 595
15.4 The First Functional Programming Language: LISP 597
15.5 An Introduction to Scheme 600
15.6 COMMON LISP 616
15.7 ML 618
15.8 Haskell 619
15.9 Applications of Functional Languages 623
15.10 A Comparison of Functional and Imperative Languages 624

Chapter 16 Logic Programming Languages 629

16.1 Introduction 630
16.2 A Brief Introduction to Predicate Calculus 630
16.3 Predicate Calculus and Proving Theorems 634
16.4 An Overview of Logic Programming 636
16.5 The Origins of Prolog 638
16.6 The Basic Elements of Prolog 638
16.7 Deficiencies of Prolog 652
16.8 Applications of Logic Programming 658
16.9 Conclusions 660

Bibliography 665
Index 677

1 Preliminaries

Konrad Zuse

Konrad Zuse designed a series of electromechanical computers between 1936 and 1944 in Germany. In 1945, he designed a complete algorithmic programming language, Plankalkül, which was never implemented, and its complete description was not even published until 1972.

CHAPTER OUTLINE

1.1 Reasons for Studying Concepts of Programming Languages
1.2 Programming Domains
1.3 Language Evaluation Criteria
1.4 Influences on Language Design
1.5 Language Categories
1.6 Language Design Trade-offs
1.7 Implementation Methods
1.8 Programming Environments

此为试读，需要完整PDF请访问：www.ertongbook.com