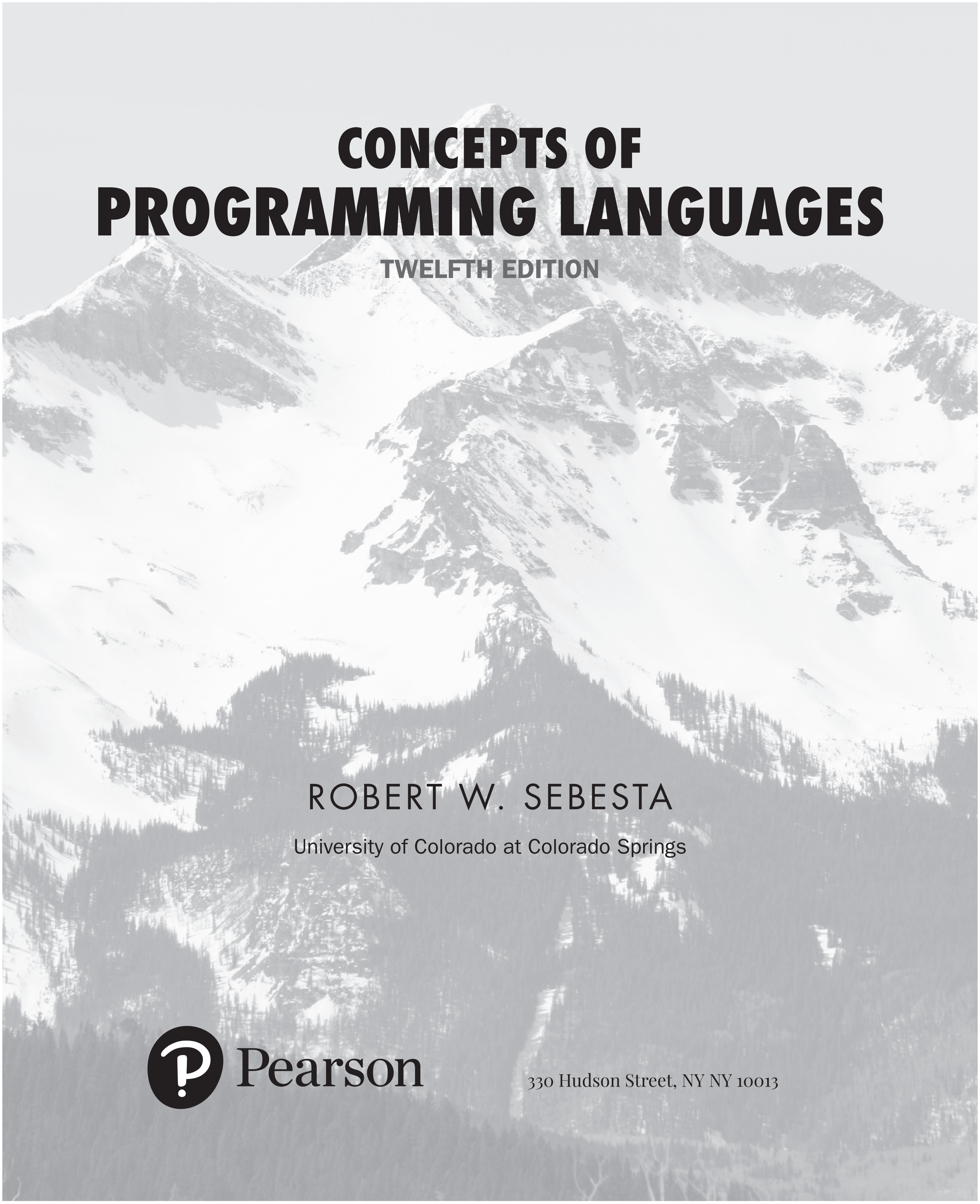
CONCEPTS OF PROGRAMMING LANGUAGES

TWELFTH EDITION



CONCEPTS OF PROGRAMMING LANGUAGES

TWELFTH EDITION

ROBERT W. SEBESTA

University of Colorado at Colorado Springs

Pearson

330 Hudson Street, NY NY 10013

Senior Vice President, Courseware Portfolio Management: *Marcia Horton*
Director, Portfolio Management: Engineering, Computer Science & Global Editions: *Julian Partridge*
Specialist, Higher Ed Portfolio Management: *Matt Goldstein*
Portfolio Management Assistant: *Meghan Jacoby*
Managing Content Producer: *Scott Disanno*
Content Producer: *Carole Snyder*
Web Developer: *Steve Wright*
Rights and Permissions Manager: *Ben Ferrini*
Manufacturing Buyer, Higher Ed, Lake Side Communications Inc (LSC): *Maura Zaldivar-Garcia*
Inventory Manager: *Ann Lam*
Product Marketing Manager: *Yvonne Vannatta*
Field Marketing Manager: *Demetrius Hall*
Marketing Assistant: *Jon Bryant*
Cover Designer: *Joyce Wells, jWellsDesign*
Full-Service Project Manager: *Prathiba Rajagopal, SPi Global*
Composition: *SPi Global*

Library of Congress Cataloging-in-Publication Data

Names: Sebesta, Robert W., author.
Title: Concepts of programming languages / Robert W. Sebesta, University of
Colorado at Colorado Springs.
Description: Twelfth edition. | Pearson, [2019]
Identifiers: LCCN 2017059077| ISBN 9780134997186 (alk. paper) | ISBN
0134997182 (alk. paper)
Subjects: LCSH: Programming languages (Electronic computers)
Classification: LCC QA76.7 .S43 2019 | DDC 005.13--dc23 LC record available at https://lccn.loc.gov/2017059077

1 18

Pearson

ISBN 10: 0-13-499718-2
ISBN 13: 978-0-13-499718-6

Changes for the Twelfth Edition of Concepts of Programming Languages

- **Chapter 2**: Added Section 2.16.4 A Replacement for Objective-C: Swift
  Added Section 2.16.5 Another Related Language: Delphi
  Deleted Section 2.18.6 Origins and Characteristics of Lua
- **Chapter 5**: Rewrote several paragraphs in Section 5.5.3 to correct and clarify
- **Chapter 6**: Added a paragraph to Section 6.3.2 to describe support for strings in Swift
  Added a paragraph to Section 6.4.2 to describe support the enumeration types in Swift
  Added a paragraph to Section 6.5.3 to describe support for arrays in Swift
  Added a paragraph to Section 6.6.1 to describe support for associative arrays in Swift
  Deleted the interview in Section 6.6.1
  Added Section 6.12 Optional Types
- **Chapter 8**: Added a design issue and a brief discussion of it to Section 8.3.1.1
  Added several paragraphs to Section 8.3.4 that describe iterators in Python
- **Chapter 9**: Added a paragraph to Section 9.5.4 on Swift parameters
- **Chapter 11**: Deleted Section 11.4.2 (Abstract Data Types in Objective-C)
- **Chapter 12**: Deleted Section 12.4.5 (Objective-C)
  Deleted Objective-C column from Table 12.1
  Added a paragraph in the Summary on reflection

Preface

Changes for the Twelfth Edition

The goals, overall structure, and approach of this twelfth edition of *Concepts of Programming Languages* remains the same as those of the eleven previous editions. The principal goals are to introduce the fundamental constructs of contemporary programming languages and to provide the reader with the tools necessary for the critical evaluation of existing and future programming languages. A secondary goal is to prepare the reader for the study of compiler design, by providing an in-depth discussion of programming language structures, presenting a formal method of describing syntax, and introducing approaches to lexical and syntax analysis.

The twelfth edition evolved from the eleventh through several different kinds of changes. To maintain the currency of the material, nearly all discussion of some programming languages, specifically Lua and Objective-C, has been removed. Material on the newer language, Swift, was added to several chapters.

In addition, a new section on optional types was added to Chapter 6. Material was added to Section 8.3.4 to describe iterators in Python. In numerous places in the manuscript small changes were made to correct and/or clarify the discussion.

The Vision

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, which are covered in Chapter 3. Also, implementation techniques for various language constructs must be considered: Lexical and syntax analysis are discussed in Chapter 4, and implementation of subprogram linkage is covered in 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 twelfth edition.

Chapter Outlines

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 the languages that are discussed in this book. Although no attempt is made to describe any language 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. 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 describe both the syntax and static semantics of languages. The difficult task of semantic description is then explored, including brief introductions to the three most common methods: operational, denotational, and axiomatic semantics.

Chapter 4 introduces lexical and syntax analysis. This chapter is targeted to those Computer Science departments that no longer require a compiler design course in their curricula. Similar to Chapter 2, this chapter stands alone and can be studied independently of the rest of the book, except for Chapter 3, on which it depends.

Chapters 5 through 14 describe in detail the design issues for the primary constructs of programming 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, and Chapters 9 and 10 discuss subprograms and their implementation. Chapter 11 examines data abstraction facilities. Chapter 12 provides an in-depth discussion of language features that support object-oriented programming (inheritance and dynamic method binding), Chapter 13 discusses concurrent program units, and Chapter 14 is about exception handling, along with a brief discussion of event handling.

Chapters 15 and 16 describe two of the most important alternative programming paradigms: functional programming and logic programming. However, some of the data structures and control constructs of functional programming languages are discussed in Chapters 6 and 8. 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 ML, Haskell, and F# are given to illustrate some different directions in functional language design. Chapter 16 introduces logic programming and the logic programming language, Prolog.

To the Instructor

Chapters 1 and 3 are typically covered 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. If a course in compiler design is required, Chapter 4 is not covered.

Chapters 5 through 9 should be relatively easy for students with extensive programming experience in C++, Java, or C#. Chapters 10 through 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 material in the last two chapters. Graduate courses, however, should be able to completely discuss the material in those chapters by skipping over some parts of the early chapters on imperative languages.

Supplemental Materials

The following supplements are available to all readers of this book at *www.pearson.com/cs-resources*.

- A set of lecture note slides. PowerPoint slides are available for each chapter in the book.
- All of the figures from the book.

A companion Web site to the book is available at *www.pearson.com/cs-resources*. This site contains mini-manuals (approximately 100-page tutorials) on a handful of languages.

Solutions to many of the problem sets are available to qualified instructors in our Instructor Resource Center at *www.pearson.com*. Please contact your school's Pearson representative or visit *www.pearson.com* to register.

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:

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

JavaScript is included in virtually all browsers; PHP is included in virtually all Web servers.

All this information is also included on the companion Web site.

Acknowledgments

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

Aaron Rababaah	*University of Maryland at Eastern Shore*
Amar Raheja	*California State Polytechnic University–Pomona*
Amer Diwan	*University of Colorado*
Bob Neufeld	*Wichita State University*
Bruce R. Maxim	*University of Michigan–Dearborn*
Charles Nicholas	*University of Maryland–Baltimore County*
Cristian Videira Lopes	*University of California–Irvine*
Curtis Meadow	*University of Maine*
David E. Goldschmidt	
Donald Kraft	*Louisiana State University*
Duane J. Jarc	*University of Maryland, University College*
Euripides Montagne	*University of Central Florida*
Frank J. Mitropoulos	*Nova Southeastern University*
Gloria Melara	*California State University–Northridge*
Hossein Saiedian	*University of Kansas*
I-ping Chu	*DePaul University*
Ian Barland	*Radford University*
K. N. King	*Georgia State University*
Karina Assiter	*Wentworth Institute of Technology*
Mark Llewellyn	*University of Central Florida*
Matthew Michael Burke	
Michael Prentice	*SUNY Buffalo*
Nancy Tinkham	*Rowan University*
Neelam Soundarajan	*Ohio State University*
Nigel Gwee	*Southern University–Baton Rouge*
Pamela Cutter	*Kalamazoo College*
Paul M. Jackowitz	*University of Scranton*
Paul Tymann	*Rochester Institute of Technology*
Richard M. Osborne	*University of Colorado–Denver*
Richard Min	*University of Texas at Dallas*
Robert McCloskey	*University of Scranton*
Ryan Stansifer	*Florida Institute of Technology*
Salih Yurttas	*Texas A&M University*
Saverio Perugini	*University of Dayton*
Serita Nelesen	*Calvin College*
Simon H. Lin	*California State University–Northridge*

Stephen Edwards	*Virginia Tech*
Stuart C. Shapiro	*SUNY Buffalo*
Sumanth Yenduri	*University of Southern Mississippi*
Teresa Cole	*Boise State University*
Thomas Turner	*University of Central Oklahoma*
Tim R. Norton	*University of Colorado–Colorado Springs*
Timothy Henry	*University of Rhode Island*
Walter Pharr	*College of Charleston*
Xiangyan Zeng	*Fort Valley State University*

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, 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, Steven Rapkin, Hamilton Richard, Tom Sager, Raghvinder Sangwan, Joseph Schell, Sibylle Schupp, Mary Louise Soffa, Neelam Soundarajan, Ryan Stansifer, Steve Stevenson, Virginia Teller, Yang Wang, John M. Weiss, Franck Xia, and Salih Yurnas.

Matt Goldstein, Portfolio Management Specialist; Meghan Jacoby, Portfolio Management Assistant; Managing Content Producer, Scott Disanno; and Prathiba Rajagopal, all deserve my gratitude for their efforts to produce the twelfth edition both quickly and carefully.

About the Author

Robert Sebesta is an Associate Professor Emeritus in the Computer Science Department at the University of Colorado–Colorado Springs. Professor Sebesta received a BS in applied mathematics from the University of Colorado in Boulder and MS and PhD degrees in computer science from Pennsylvania State University. He has taught computer science for more than 40 years. His professional interests are the design and evaluation of programming languages and Web programming.

Contents

CONCEPTS OF PROGRAMMING LANGUAGES

TWELFTH EDITION