CONCEPTS OF PROGRAMMING LANGUAGES
TWELFTH EDITION
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 Louisiana State University
Donald Kraft University of Maryland, University College
Duane J. Jarc University of Central Florida
Euripides Montagne Nova Southeastern University
Frank J. Mitropoulos California State University–Northridge
Gloria Melara University of Kansas
Hossein Saiedian Wentworth Institute of Technology
I-ping Chu DePaul University
Ian Barland Radford University
K. N. King Georgia State University
Karina Assiter University of Central Florida
Mark Llewellyn
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
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.
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

## Chapter 1  Preliminaries  

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   

Summary • Review Questions • Problem Set  

## Chapter 2  Evolution of the Major Programming Languages  

2.1 Zuse’s Plankalkül   

2.2 Pseudocodes   

2.3 The IBM 704 and Fortran   

2.4 Functional Programming: Lisp   

2.5 The First Step Toward Sophistication: ALGOL 60   

2.6 Computerizing Business Records: COBOL   

2.7 The Beginnings of Timesharing: Basic   

Interview: ALAN COOPER—User Design and Language Design   

2.8 Everything for Everybody: PL/I   

2.9 Two Early Dynamic Languages: APL and SNOBOL   

2.10 The Beginnings of Data Abstraction: SIMULA 67   

2.11 Orthogonal Design: ALGOL 68   

2.12 Some Early Descendants of the ALGOLs   

Summary • Review Questions • Problem Set  

xv
Chapter 3  Describing Syntax and Semantics  
3.1 Introduction ................................................................. 110
3.2 The General Problem of Describing Syntax ..................... 111
3.3 Formal Methods of Describing Syntax ............................ 113
3.4 Attribute Grammars ..................................................... 128

History Note ............................................................................ 128
3.5 Describing the Meanings of Programs: Dynamic Semantics .... 134

History Note ............................................................................ 142
Summary • Bibliographic Notes • Review Questions • Problem Set ...... 155

Chapter 4  Lexical and Syntax Analysis  
4.1 Introduction ..................................................................... 162
4.2 Lexical Analysis ............................................................. 163
4.3 The Parsing Problem ...................................................... 171
4.4 Recursive-Descent Parsing ............................................. 175
4.5 Bottom-Up Parsing ......................................................... 183

Summary • Review Questions • Problem Set • Programming Exercises .... 191

Chapter 5  Names, Bindings, and Scopes  
5.1 Introduction ..................................................................... 198
5.2 Names ............................................................................. 199
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>Variables</td>
<td>200</td>
</tr>
<tr>
<td>5.4</td>
<td>The Concept of Binding</td>
<td>203</td>
</tr>
<tr>
<td>5.5</td>
<td>Scope</td>
<td>211</td>
</tr>
<tr>
<td>5.6</td>
<td>Scope and Lifetime</td>
<td>222</td>
</tr>
<tr>
<td>5.7</td>
<td>Referencing Environments</td>
<td>223</td>
</tr>
<tr>
<td>5.8</td>
<td>Named Constants</td>
<td>224</td>
</tr>
<tr>
<td></td>
<td><strong>Summary • Review Questions • Problem Set • Programming Exercises</strong></td>
<td>227</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>236</td>
</tr>
<tr>
<td>6.2</td>
<td>Primitive Data Types</td>
<td>238</td>
</tr>
<tr>
<td>6.3</td>
<td>Character String Types</td>
<td>242</td>
</tr>
<tr>
<td></td>
<td><strong>History Note</strong></td>
<td>243</td>
</tr>
<tr>
<td>6.4</td>
<td>Enumeration Types</td>
<td>247</td>
</tr>
<tr>
<td>6.5</td>
<td>Array Types</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td><strong>History Note</strong></td>
<td>251</td>
</tr>
<tr>
<td>6.6</td>
<td>Associative Arrays</td>
<td>261</td>
</tr>
<tr>
<td>6.7</td>
<td>Record Types</td>
<td>263</td>
</tr>
<tr>
<td>6.8</td>
<td>Tuple Types</td>
<td>266</td>
</tr>
<tr>
<td>6.9</td>
<td>List Types</td>
<td>268</td>
</tr>
<tr>
<td>6.10</td>
<td>Union Types</td>
<td>270</td>
</tr>
<tr>
<td>6.11</td>
<td>Pointer and Reference Types</td>
<td>273</td>
</tr>
<tr>
<td></td>
<td><strong>History Note</strong></td>
<td>276</td>
</tr>
<tr>
<td>6.12</td>
<td>Optional Types</td>
<td>285</td>
</tr>
<tr>
<td>6.13</td>
<td>Type Checking</td>
<td>286</td>
</tr>
<tr>
<td>6.14</td>
<td>Strong Typing</td>
<td>287</td>
</tr>
<tr>
<td>6.15</td>
<td>Type Equivalence</td>
<td>288</td>
</tr>
<tr>
<td>6.16</td>
<td>Theory and Data Types</td>
<td>292</td>
</tr>
<tr>
<td></td>
<td><strong>Summary • Bibliographic Notes • Review Questions • Problem Set • Programming Exercises</strong></td>
<td>294</td>
</tr>
</tbody>
</table>
Chapter 7  Expressions and Assignment Statements  301
  7.1  Introduction .......................................................................................302
  7.2  Arithmetic Expressions ......................................................................302
  7.3  Overloaded Operators .......................................................................311
  7.4  Type Conversions ..............................................................................313
  History Note .............................................................................................315
  7.5  Relational and Boolean Expressions .................................................316
  History Note .............................................................................................316
  7.6  Short-Circuit Evaluation ...................................................................318
  7.7  Assignment Statements .....................................................................319
  History Note .............................................................................................323
  7.8  Mixed-Mode Assignment ..................................................................324

Summary • Review Questions • Problem Set • Programming Exercises ...324

Chapter 8  Statement-Level Control Structures  329
  8.1  Introduction .......................................................................................330
  8.2  Selection Statements .........................................................................332
  8.3  Iterative Statements ...........................................................................343
  8.4  Unconditional Branching ..................................................................355
  History Note .............................................................................................356
  8.5  Guarded Commands .........................................................................356
  8.6  Conclusions .......................................................................................359

Summary • Review Questions • Problem Set • Programming Exercises ...360

Chapter 9  Subprograms  365
  9.1  Introduction .......................................................................................366
  9.2  Fundamentals of Subprograms .........................................................366
  9.3  Design Issues for Subprograms .........................................................374
  9.4  Local Referencing Environments .....................................................375
  9.5  Parameter-Passing Methods ..............................................................376
  History Note .............................................................................................384
Contents

9.6 Parameters That Are Subprograms ................................................. 392

History Note ............................................................................................. 394

9.7 Calling Subprograms Indirectly ..................................................... 394

9.8 Design Issues for Functions ....................................................... 396

9.9 Overloaded Subprograms .......................................................... 397

9.10 Generic Subprograms ............................................................... 398

9.11 User-Defined Overloaded Operators ......................................... 404

9.12 Closures ................................................................................ 405

9.13 Coroutines .............................................................................. 407

Summary • Review Questions • Problem Set • Programming Exercises .... 410

Chapter 10 Implementing Subprograms ............................ 417

10.1 The General Semantics of Calls and Returns ......................... 418

10.2 Implementing “Simple” Subprograms ...................................... 419

10.3 Implementing Subprograms with Stack-Dynamic
Local Variables ...................................................................................... 421

10.4 Nested Subprograms ............................................................... 429

10.5 Blocks .................................................................................. 436

10.6 Implementing Dynamic Scoping ............................................... 437

Summary • Review Questions • Problem Set • Programming Exercises .... 441

Chapter 11 Abstract Data Types and Encapsulation Constructs ........................................ 447

11.1 The Concept of Abstraction ..................................................... 448

11.2 Introduction to Data Abstraction .............................................. 449

11.3 Design Issues for Abstract Data Types ..................................... 452

11.4 Language Examples ................................................................ 453

Interview: BJARNE STROUSTRUP—C++: Its Birth,
Its Ubiquitousness, and Common Criticisms ..................................... 454

11.5 Parameterized Abstract Data Types .......................................... 466

11.6 Encapsulation Constructs ........................................................ 471

11.7 Naming Encapsulations ............................................................. 474

Summary • Review Questions • Problem Set • Programming Exercises .... 478
Contents

14.7 Event Handling in C# .................................................................613
Summary • Bibliographic Notes • Review Questions • Problem Set • Programming Exercises .......................................................616

Chapter 15 Functional Programming Languages 623
15.1 Introduction .................................................................................624
15.2 Mathematical Functions .............................................................625
15.3 Fundamentals of Functional Programming Languages ............628
15.4 The First Functional Programming Language: Lisp .................629
15.5 An Introduction to Scheme ..........................................................633
15.6 Common Lisp ..............................................................................651
15.7 ML ..............................................................................................653
15.8 Haskell .......................................................................................658
15.9 F# .............................................................................................663
15.10 Support for Functional Programming in Primarily Imperative Languages .................................................................666
15.11 A Comparison of Functional and Imperative Languages .........669
Summary • Bibliographic Notes • Review Questions • Problem Set • Programming Exercises .......................................................671

Chapter 16 Logic Programming Languages 679
16.1 Introduction .................................................................................680
16.2 A Brief Introduction to Predicate Calculus ...............................680
16.3 Predicate Calculus and Proving Theorems .................................684
16.4 An Overview of Logic Programming ...........................................686
16.5 The Origins of Prolog .................................................................688
16.6 The Basic Elements of Prolog .....................................................688
16.7 Deficiencies of Prolog .................................................................703
16.8 Applications of Logic Programming ..........................................709
Summary • Bibliographic Notes • Review Questions • Problem Set • Programming Exercises .......................................................710

Bibliography ....................................................................................715
Index ..................................................................................................725
CONCEPTS OF
PROGRAMMING LANGUAGES
TWELFTH EDITION