digital resources for students

Your new textbook provides 12-month access to digital resources that may include VideoNotes (step-by-step video tutorials on programming concepts), source code, web chapters, quizzes, and more. Refer to the preface in the textbook for a detailed list of resources.


1. Go to www.pearsonglobaleditions.com/Sebesta
2. Click Companion Website
3. Click Register and follow the on-screen instructions to create a login name and password

Use a coin to scratch off the coating and reveal your access code.
Do not use a sharp knife or other sharp object as it may damage the code.

Use the login name and password you created during registration to start using the digital resources that accompany your textbook.

**IMPORTANT:**

This access code can only be used once. This subscription is valid for 12 months upon activation and is not transferable. If the access code has already been revealed it may no longer be valid.

For technical support go to http://247pearsoned.custhelp.com
CONCEPTS OF PROGRAMMING LANGUAGES

ELEVENTH EDITION
GLOBAL EDITION

ROBERT W. SEBEESTA
University of Colorado at Colorado Springs

Global Edition contributions by

Soumen Mukherjee
RCC Institute of Information Technology

Arup Kumar Bhattacharjee
RCC Institute of Information Technology
Changes for the Eleventh Edition of Concepts of Programming Languages

- **Chapter 6**: Deleted the discussions of Ada’s subrange types, array initialization, records, union types, pointers, and strong typing
- **Chapter 7**: Deleted the discussions of Ada operator associativity and mixed-mode expressions
- **Chapter 8**: Expanded the paragraph on F# selection statements in Section 8.2.1.5
  Deleted the discussion of the Ada `for` statement
- **Chapter 9**: Added three design issues for subprograms in Section 9.3
  Deleted the discussions of Ada and Fortran multidimensional parameters
- **Chapter 10**: Replaced example program `Main_2`, written in Ada, with an equivalent program written in JavaScript in Section 10.4.2
  Changed Figure 10.9 to reflect this new JavaScript example
- **Chapter 11**: Deleted the discussions of Ada abstract data types, generic procedures, and packages
  Added a new paragraph to Section 11.4.3 (Abstract Data Types in Java)
- **Chapter 12**: Added a paragraph to Section 12.2.2 (Inheritance) that discusses access control
  Expanded the discussion of class variables in Section 12.2.2
  Added a paragraph to Section 12.4.4 that discusses final classes in Objective-C
  Reorganized Sections 12.5 to 12.9 into a single section
  Added Table 12.1 on language design choices to Section 12.4.6.4
  Added a new section, Section 6 (Reflection), including example programs in Java and C#
- **Chapter 13**: Deleted the discussions of Ada task termination and task priorities
- **Chapter 14**: Deleted exception handling in Ada
  Added a new section, 14.4 (Exception Handling in Python and Ruby)
Preface

Changes for the Eleventh Edition

The goals, overall structure, and approach of this eleventh edition of Concepts of Programming Languages remain the same as those of the ten earlier 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 syntactic analysis.

The eleventh edition evolved from the tenth through several different kinds of changes. To maintain the currency of the material, much of the discussion of older programming languages, particularly Ada and Fortran, has been removed. For example, the descriptions of Ada’s records, union types, and pointers were removed from Chapter 6. Likewise, the description of Ada’s for statement was removed from Chapter 8 and the discussion of Ada’s abstract data types was removed from Chapter 11.

On the other hand, a section on reflection that includes two complete program examples was added to Chapter 12, a section on exception handling in Python and Ruby was added to Chapter 14, and a table of the design choices of a few common languages for support for object-oriented programming was added to Chapter 12.

In some cases, material has been moved. For example, Section 9.10 was moved backward to become the new Section 9.8.

In one case, example program MAIN_2 in Chapter 10 was rewritten in JavaScript, previously having been written in Ada.

Chapter 12 was substantially revised, with several new paragraphs, two new sections, and numerous other changes to improve clarity.

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 eleventh 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.
Preface

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

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 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.pearsonglobaleditions.com/Sebesta.

- 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.pearsonglobaleditions.com/Sebesta. This site contains mini-manuals (approximately 100-page tutorials) on a handful of languages. These assume that the student knows how to program
in some other language, giving the student enough information to complete the chapter materials in each language. Currently the site includes manuals for C++, C, Java, and Smalltalk.

Solutions to many of the problem sets are available to qualified instructors in our Instructor Resource Center at www.pearsonglobaleditions.com/Sebesta.

**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:

<table>
<thead>
<tr>
<th>Language</th>
<th>Web Site</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, C++, Fortran, and Ada</td>
<td>gcc.gnu.org</td>
</tr>
<tr>
<td>C# and F#</td>
<td>microsoft.com</td>
</tr>
<tr>
<td>Java</td>
<td>java.sun.com</td>
</tr>
<tr>
<td>Haskell</td>
<td>haskell.org</td>
</tr>
<tr>
<td>Lua</td>
<td><a href="http://www.lua.org">www.lua.org</a></td>
</tr>
<tr>
<td>Scheme</td>
<td><a href="http://www.plt-scheme.org/software/drscheme">www.plt-scheme.org/software/drscheme</a></td>
</tr>
<tr>
<td>Perl</td>
<td><a href="http://www.perl.com">www.perl.com</a></td>
</tr>
<tr>
<td>Python</td>
<td><a href="http://www.python.org">www.python.org</a></td>
</tr>
<tr>
<td>Ruby</td>
<td><a href="http://www.ruby-lang.org">www.ruby-lang.org</a></td>
</tr>
</tbody>
</table>

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
Amar Raheja
Amer Diwan
Bob Neufeld
Bruce R. Maxim
Charles Nicholas
Cristian Videira Lopes
Curtis Meadow
David E. Goldschmidt
Donald Kraft
Duane J. Jarc
Euripides Montagne
Frank J. Mitropoulos
Gloria Melara
Hossein Saiedian
I-ping Chu
Ian Barland
K. N. King
Karina Assiter
Mark Llewellyn
Matthew Michael Burke
Michael Prentice
Nancy Tinkham
Neelam Soundarajan
Nigel Gwee
Pamela Cutter
Paul M. Jackowitz
Paul Tymann
Richard M. Osborne
Richard Min
Robert McCloskey
Ryan Stansifer
Salih Yurttas
Saverio Perugini
Serita Nelesen
Simon H. Lin

University of Maryland at Eastern Shore
California State Polytechnic University–Pomona
University of Colorado
Wichita State University
University of Michigan–Dearborn
University of Maryland–Baltimore County
University of California–Irvine
University of Maine
Louisiana State University
University of Maryland, University College
University of Central Florida
Nova Southeastern University
California State University–Nortbridge
University of Kansas
DePaul University
Radford University
Georgia State University
Wentworth Institute of Technology
University of Central Florida
SUNY Buffalo
Rowan University
Ohio State University
Southern University–Baton Rouge
Kalamazoo College
University of Scranton
Rochester Institute of Technology
University of Colorado–Denver
University of Texas at Dallas
University of Scranton
Florida Institute of Technology
Texas A&M University
University of Dayton
Calvin College
California State University–Nortbridge
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, editor; Kelsey Loanes, editorial assistant; Team Lead Product Management, Scott Disanno; Pavithra Jayapaul, and Mahalatchoumy Saravanan, all deserve my gratitude for their efforts to produce the eleventh edition both quickly and carefully.

The publishers would like to thank the following for reviewing the Global Edition:

Sandeep B. L., *M. S. Ramaiah Institute of Technology*
Koushik S., *M. S. Ramaiah Institute of Technology*
Sheena V. M., *Don Bosco College*
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

Chapter 1  Preliminaries  25
1.1  Reasons for Studying Concepts of Programming Languages..............26
1.2  Programming Domains.................................................................29
1.3  Language Evaluation Criteria......................................................30
1.4  Influences on Language Design..................................................41
1.5  Language Categories....................................................................44
1.6  Language Design Trade-Offs........................................................45
1.7  Implementation Methods...............................................................46
1.8  Programming Environments .........................................................53
Summary • Review Questions • Problem Set.........................................54

Chapter 2  Evolution of the Major Programming Languages  57
2.1  Zuse’s Plankalkül ............................................................................60
2.2  Pseudocodes..................................................................................61
2.3  The IBM 704 and Fortran ..............................................................64
2.4  Functional Programming: Lisp.......................................................69
2.5  The First Step Toward Sophistication: ALGOL 60 .........................74
2.6  Computerizing Business Records: COBOL.................................80
2.7  The Beginnings of Timesharing: Basic............................................85
Interview: ALAN COOPER—User Design and Language Design........ 88
2.8  Everything for Everybody: PL/I.....................................................90
2.9  Two Early Dynamic Languages: APL and SNOBOL.......................93
2.10 The Beginnings of Data Abstraction: SIMULA 67 .........................94
2.11 Orthogonal Design: ALGOL 68.....................................................95
2.12 Some Early Descendants of the ALGOLs......................................97
# Contents

2.13 Programming Based on Logic: Prolog .............................................101
2.14 History’s Largest Design Effort: Ada.............................................103
2.15 Object-Oriented Programming: Smalltalk.......................................107
2.16 Combining Imperative and Object-Oriented Features: C++ ..........109
2.17 An Imperative-Based Object-Oriented Language: Java...............112
2.18 Scripting Languages..........................................................................115
2.19 The Flagship .NET Language: C#...................................................122
2.20 Markup-Programming Hybrid Languages.......................................124

Summary • Bibliographic Notes • Review Questions • Problem Set • Programming Exercises .................................................................126

## Chapter 3 Describing Syntax and Semantics

3.1 Introduction..........................................................................................134
3.2 The General Problem of Describing Syntax........................................135
3.3 Formal Methods of Describing Syntax................................................137
3.4 Attribute Grammars..............................................................................152

*History Note* ..............................................................................................152

3.5 Describing the Meanings of Programs: Dynamic Semantics..............158

*History Note* ..............................................................................................166

Summary • Bibliographic Notes • Review Questions • Problem Set........179

## Chapter 4 Lexical and Syntax Analysis

4.1 Introduction..........................................................................................186
4.2 Lexical Analysis......................................................................................187
4.3 The Parsing Problem ...........................................................................195
4.4 Recursive-Descent Parsing..................................................................199
4.5 Bottom-Up Parsing..............................................................................207

Summary • Review Questions • Problem Set • Programming Exercises ..................................................................................215

## Chapter 5 Names, Bindings, and Scopes

5.1 Introduction..........................................................................................222
5.2 Names.....................................................................................................223
Contents

History Note ............................................................................................................. 223

5.3 Variables............................................................................................................. 224
5.4 The Concept of Binding ....................................................................................... 227
5.5 Scope..................................................................................................................... 235
5.6 Scope and Lifetime.............................................................................................. 246
5.7 Referencing Environments .................................................................................. 247
5.8 Named Constants ............................................................................................... 248

Summary • Review Questions • Problem Set • Programming Exercises .................................................. 251

Chapter 6 Data Types .................................................................................................. 259

6.1 Introduction........................................................................................................... 260
6.2 Primitive Data Types........................................................................................... 262
6.3 Character String Types......................................................................................... 266

History Note ................................................................................................................. 267

6.4 Enumeration Types.............................................................................................. 271
6.5 Array Types........................................................................................................... 274

History Note ................................................................................................................ 275

History Note ................................................................................................................ 275

6.6 Associative Arrays............................................................................................... 285

Interview: ROBERTO IERUSALIMSCHY—Lua ............................................................. 286

6.7 Record Types........................................................................................................ 289
6.8 Tuple Types........................................................................................................... 292
6.9 List Types............................................................................................................... 294
6.10 Union Types......................................................................................................... 296
6.11 Pointer and Reference Types ............................................................................. 299

History Note ................................................................................................................ 302

6.12 Type Checking.................................................................................................... 311
6.13 Strong Typing....................................................................................................... 312
6.14 Type Equivalence................................................................................................ 313
6.15 Theory and Data Types...................................................................................... 317

Summary • Bibliographic Notes • Review Questions • Problem Set • Programming Exercises .......................................................... 319
## Chapter 7  Expressions and Assignment Statements  325

### 7.1  Introduction  

### 7.2  Arithmetic Expressions  

### 7.3  Overloaded Operators  

### 7.4  Type Conversions  

### 7.5  Relational and Boolean Expressions  

### 7.6  Short-Circuit Evaluation  

### 7.7  Assignment Statements  

### 7.8  Mixed-Mode Assignment  

Summary • Review Questions • Problem Set • Programming Exercises ... 348

## Chapter 8  Statement-Level Control Structures  353

### 8.1  Introduction  

### 8.2  Selection Statements  

### 8.3  Iterative Statements  

### 8.4  Unconditional Branching  

### 8.5  Guarded Commands  

### 8.6  Conclusions  

Summary • Review Questions • Problem Set • Programming Exercises ... 383

## Chapter 9  Subprograms  389

### 9.1  Introduction  

### 9.2  Fundamentals of Subprograms  

### 9.3  Design Issues for Subprograms  

### 9.4  Local Referencing Environments  

### 9.5  Parameter-Passing Methods  

Summary • Review Questions • Problem Set • Programming Exercises ... 401
## Contents

- History Note ........................................................................................................... 409
- **9.6** Parameters That Are Subprograms .......................................................... 417
  - History Note ......................................................................................................... 419
- **9.7** Calling Subprograms Indirectly .................................................................. 419
- **9.8** Design Issues for Functions .......................................................................... 421
- **9.9** Overloaded Subprograms ............................................................................ 423
- **9.10** Generic Subprograms ................................................................................ 424
- **9.11** User-Defined Overloaded Operators ....................................................... 430
- **9.12** Closures ...................................................................................................... 430
- **9.13** Coroutines .................................................................................................. 432

Summary • Review Questions • Problem Set • Programming Exercises .... 435

### Chapter 10  Implementing Subprograms  441

- **10.1** The General Semantics of Calls and Returns ........................................... 442
- **10.2** Implementing “Simple” Subprograms ....................................................... 443
- **10.3** Implementing Subprograms with Stack-Dynamic Local Variables ........... 445
- **10.4** Nested Subprograms ................................................................................ 453
- **10.5** Blocks ......................................................................................................... 460
- **10.6** Implementing Dynamic Scoping ............................................................... 461

Summary • Review Questions • Problem Set • Programming Exercises .... 465

### Chapter 11  Abstract Data Types and Encapsulation Constructs  471

- **11.1** The Concept of Abstraction ...................................................................... 472
- **11.2** Introduction to Data Abstraction ............................................................... 473
- **11.3** Design Issues for Abstract Data Types ..................................................... 476
- **11.4** Language Examples .................................................................................. 477

Interview: **BJARNE STROUSTRUP—C++: Its Birth, Its Ubiquitousness, and Common Criticisms**  478

- **11.5** Parameterized Abstract Data Types ......................................................... 496
- **11.6** Encapsulation Constructs ........................................................................ 500
- **11.7** Naming Encapsulations ............................................................................ 504

Summary • Review Questions • Problem Set • Programming Exercises .... 507
## Chapter 12  Support for Object-Oriented Programming  513

12.1 Introduction ....................................................................................... 514
12.2 Object-Oriented Programming ........................................................ 515
12.3 Design Issues for Object-Oriented Languages ................................. 519
12.4 Support for Object-Oriented Programming in Specific Languages ................................................................. 524

Interview: BJARNE STROUSTRUP—On Paradigms and Better Programming ................................................................. 528

12.5 Implementation of Object-Oriented Constructs ................................. 552
12.6 Reflection ........................................................................................... 555

Summary • Review Questions • Problem Set • Programming Exercises.... 561

## Chapter 13  Concurrency  567

13.1 Introduction ....................................................................................... 568
13.2 Introduction to Subprogram-Level Concurrency .................................. 573
13.3 Semaphores ........................................................................................ 578
13.4 Monitors ............................................................................................ 583
13.5 Message Passing ................................................................................. 585
13.6 Ada Support for Concurrency ............................................................ 586
13.7 Java Threads ....................................................................................... 594
13.8 C# Threads ........................................................................................ 604
13.9 Concurrency in Functional Languages .............................................. 609
13.10 Statement-Level Concurrency ........................................................... 612

Summary • Bibliographic Notes • Review Questions • Problem Set • Programming Exercises.................................................. 614

## Chapter 14  Exception Handling and Event Handling  621

14.1 Introduction to Exception Handling .................................................... 622

History Note ............................................................................................. 626

14.2 Exception Handling in C++ ............................................................... 628
14.3 Exception Handling in Java ............................................................... 632
14.4 Exception Handling in Python and Ruby ............................................ 639
14.5 Introduction to Event Handling ......................................................... 642
14.6 Event Handling with Java ................................................................. 643
<table>
<thead>
<tr>
<th>Chapter 15</th>
<th>Functional Programming Languages</th>
<th>657</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1</td>
<td>Introduction</td>
<td>658</td>
</tr>
<tr>
<td>15.2</td>
<td>Mathematical Functions</td>
<td>659</td>
</tr>
<tr>
<td>15.3</td>
<td>Fundamentals of Functional Program</td>
<td>662</td>
</tr>
<tr>
<td>15.4</td>
<td>The First Functional Program Lang</td>
<td>663</td>
</tr>
<tr>
<td>15.5</td>
<td>An Introduction to Scheme</td>
<td>667</td>
</tr>
<tr>
<td>15.6</td>
<td>Common Lisp</td>
<td>685</td>
</tr>
<tr>
<td>15.7</td>
<td>ML</td>
<td>687</td>
</tr>
<tr>
<td>15.8</td>
<td>Haskell</td>
<td>692</td>
</tr>
<tr>
<td>15.9</td>
<td>F#</td>
<td>697</td>
</tr>
<tr>
<td>15.10</td>
<td>Support for Functional Program in</td>
<td>700</td>
</tr>
<tr>
<td></td>
<td>Imperative Languages</td>
<td></td>
</tr>
<tr>
<td>15.11</td>
<td>A Comparison of Functional and I</td>
<td>703</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16</th>
<th>Logic Programming Languages</th>
<th>713</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.1</td>
<td>Introduction</td>
<td>714</td>
</tr>
<tr>
<td>16.2</td>
<td>A Brief Introduction to Predicate Calculus</td>
<td>714</td>
</tr>
<tr>
<td>16.3</td>
<td>Predicate Calculus and Proving Theorems</td>
<td>718</td>
</tr>
<tr>
<td>16.4</td>
<td>An Overview of Logic Programming</td>
<td>720</td>
</tr>
<tr>
<td>16.5</td>
<td>The Origins of Prolog</td>
<td>722</td>
</tr>
<tr>
<td>16.6</td>
<td>The Basic Elements of Prolog</td>
<td>722</td>
</tr>
<tr>
<td>16.7</td>
<td>Deficiencies of Prolog</td>
<td>737</td>
</tr>
<tr>
<td>16.8</td>
<td>Applications of Logic Programming</td>
<td>743</td>
</tr>
</tbody>
</table>

| Bibliography | 749 |
| Index        | 761 |