Programming Language Pragmatics
THIRD EDITION

Michael L. Scott
Department of Computer Science
University of Rochester
Contents

Foreword xxi
Preface xxiii

FOUNDATIONS 3

1 Introduction 5
  1.1 The Art of Language Design 7
  1.2 The Programming Language Spectrum 10
  1.3 Why Study Programming Languages? 14
  1.4 Compilation and Interpretation 16
  1.5 Programming Environments 24
  1.6 An Overview of Compilation 25
    1.6.1 Lexical and Syntax Analysis 27
    1.6.2 Semantic Analysis and Intermediate Code Generation 29
    1.6.3 Target Code Generation 33
    1.6.4 Code Improvement 33
  1.7 Summary and Concluding Remarks 35
  1.8 Exercises 36
  1.9 Explorations 37
  1.10 Bibliographic Notes 39

2 Programming Language Syntax 41
  2.1 Specifying Syntax: Regular Expressions and Context-Free Grammars 42
    2.1.1 Tokens and Regular Expressions 43
    2.1.2 Context-Free Grammars 46
    2.1.3 Derivations and Parse Trees 48
2.2 Scanning
  2.2.1 Generating a Finite Automaton 51
  2.2.2 Scanner Code 55
  2.2.3 Table-Driven Scanning 60
  2.2.4 Lexical Errors 63
  2.2.5 Pragmas 65

2.3 Parsing
  2.3.1 Recursive Descent 67
  2.3.2 Table-Driven Top-Down Parsing 70
  2.3.3 Bottom-Up Parsing 76
  2.3.4 Syntax Errors 87

2.4 Theoretical Foundations
  2.4.1 Finite Automata 13
  2.4.2 Push-Down Automata 18
  2.4.3 Grammar and Language Classes 19

2.5 Summary and Concluding Remarks 101

2.6 Exercises 102

2.7 Explorations 108

2.8 Bibliographic Notes 109

3 Names, Scopes, and Bindings 111

3.1 The Notion of Binding Time 112

3.2 Object Lifetime and Storage Management 114
  3.2.1 Static Allocation 115
  3.2.2 Stack-Based Allocation 117
  3.2.3 Heap-Based Allocation 118
  3.2.4 Garbage Collection 120

3.3 Scope Rules 121
  3.3.1 Static Scoping 123
  3.3.2 Nested Subroutines 124
  3.3.3 Declaration Order 127
  3.3.4 Modules 132
  3.3.5 Module Types and Classes 136
  3.3.6 Dynamic Scoping 139

3.4 Implementing Scope 29
  3.4.1 Symbol Tables 29
  3.4.2 Association Lists and Central Reference Tables 33

3.5 The Meaning of Names within a Scope 144
  3.5.1 Aliases 144
3.5.2 Overloading
3.5.3 Polymorphism and Related Concepts

3.6 The Binding of Referencing Environments
3.6.1 Subroutine Closures
3.6.2 First-Class Values and Unlimited Extent
3.6.3 Object Closures

3.7 Macro Expansion

3.8 Separate Compilation
3.8.1 Separate Compilation in C
3.8.2 Packages and Automatic Header Inference
3.8.3 Module Hierarchies

3.9 Summary and Concluding Remarks

3.10 Exercises

3.11 Explorations

3.12 Bibliographic Notes

4 Semantic Analysis
4.1 The Role of the Semantic Analyzer
4.2 Attribute Grammars
4.3 Evaluating Attributes
4.4 Action Routines
4.5 Space Management for Attributes
4.6 Decorating a Syntax Tree
4.7 Summary and Concluding Remarks
4.8 Exercises
4.9 Explorations
4.10 Bibliographic Notes

5 Target Machine Architecture
5.1 The Memory Hierarchy
5.2 Data Representation
5.2.1 Integer Arithmetic
5.2.2 Floating-Point Arithmetic
# Contents

5.3 Instruction Set Architecture 75
  5.3.1 Addressing Modes 75
  5.3.2 Conditions and Branches 76

5.4 Architecture and Implementation 78
  5.4.1 Microprogramming 79
  5.4.2 Microprocessors 80
  5.4.3 RISC 81
  5.4.4 Multithreading and Multicore 82
  5.4.5 Two Example Architectures: The x86 and MIPS 84

5.5 Compiling for Modern Processors 91
  5.5.1 Keeping the Pipeline Full 91
  5.5.2 Register Allocation 96

5.6 Summary and Concluding Remarks 101

5.7 Exercises 103

5.8 Explorations 107

5.9 Bibliographic Notes 109

6 Control Flow 217

6.1 Expression Evaluation 219
  6.1.1 Precedence and Associativity 220
  6.1.2 Assignments 222
  6.1.3 Initialization 224
  6.1.4 Ordering within Expressions 233
  6.1.5 Short-Circuit Evaluation 235

6.2 Structured and Unstructured Flow 241
  6.2.1 Structured Alternatives to goto 242
  6.2.2 Continuations 245

6.3 Sequencing 246

6.4 Selection 247
  6.4.1 Short-Circuited Conditions 248
  6.4.2 Case/Switch Statements 251

6.5 Iteration 256
  6.5.1 Enumeration-Controlled Loops 256
  6.5.2 Combination Loops 261
7.7.2 Dangling References 356
7.7.3 Garbage Collection 357

7.8 Lists 364

7.9 Files and Input/Output
7.9.1 Interactive I/O 367
7.9.2 File-Based I/O 357
7.9.3 Text I/O 356

7.10 Equality Testing and Assignment 368
7.11 Summary and Concluding Remarks 371
7.12 Exercises 373
7.13 Explorations 379
7.14 Bibliographic Notes 380

8 Subroutines and Control Abstraction 383

8.1 Review of Stack Layout 384

8.2 Calling Sequences 386
8.2.1 Displays 389
8.2.2 Case Studies: C on the MIPS; Pascal on the x86 389
8.2.3 Register Windows 390
8.2.4 In-Line Expansion 391

8.3 Parameter Passing 393
8.3.1 Parameter Modes 394
8.3.2 Call-by-Name 402
8.3.3 Special-Purpose Parameters 403
8.3.4 Function Returns 408

8.4 Generic Subroutines and Modules 410
8.4.1 Implementation Options 412
8.4.2 Generic Parameter Constraints 414
8.4.3 Implicit Instantiation 416
8.4.4 Generics in C++, Java, and C# 417

8.5 Exception Handling 418
8.5.1 Defining Exceptions 421
8.5.2 Exception Propagation 423
8.5.3 Implementation of Exceptions 425

8.6 Coroutines 428
8.6.1 Stack Allocation 430
8.6.2 Transfer 432
ALTERNATIVE PROGRAMMING MODELS 503

10 Functional Languages 505
10.1 Historical Origins 506
10.2 Functional Programming Concepts 507
10.3 A Review/Overview of Scheme 509
  10.3.1 Bindings 512
  10.3.2 Lists and Numbers 513
  10.3.3 Equality Testing and Searching 514
  10.3.4 Control Flow and Assignment 515
  10.3.5 Programs as Lists 517
  10.3.6 Extended Example: DFA Simulation 519
10.4 Evaluation Order Revisited 521
  10.4.1 Strictness and Lazy Evaluation 523
  10.4.2 I/O: Streams and Monads 525
10.5 Higher-Order Functions 530
10.6 Theoretical Foundations ©237 ©239 ©242 ©244 534
  10.6.1 Lambda Calculus 534
  10.6.2 Control Flow 537
  10.6.3 Structures 538
10.7 Functional Programming in Perspective 534
10.8 Summary and Concluding Remarks 537
10.9 Exercises 538
10.10 Explorations 542
10.11 Bibliographic Notes 543

11 Logic Languages 545
11.1 Logic Programming Concepts 546
11.2 Prolog
  11.2.1 Resolution and Unification 549
  11.2.2 Lists 550
### Contents

1.2.3 Arithmetic 551
1.2.4 Search/Execution Order 552
1.2.5 Extended Example: Tic-Tac-Toe 554
1.2.6 Imperative Control Flow 557
1.2.7 Database Manipulation 561

1.3 Theoretical Foundations 566
   1.3.1 Clausal Form 254
   1.3.2 Limitations 255
   1.3.3 Skolemization 257

1.4 Logic Programming in Perspective 566
   1.4.1 Parts of Logic Not Covered 566
   1.4.2 Execution Order 567
   1.4.3 Negation and the “Closed World” Assumption 568

1.5 Summary and Concluding Remarks 570
1.6 Exercises 571
1.7 Explorations 573
1.8 Bibliographic Notes 573

12 Concurrency 575

12.1 Background and Motivation 576
   12.1.1 The Case for Multithreaded Programs 579
   12.1.2 Multiprocessor Architecture 581

12.2 Concurrent Programming Fundamentals 586
   12.2.1 Communication and Synchronization 587
   12.2.2 Languages and Libraries 588
   12.2.3 Thread Creation Syntax 589
   12.2.4 Implementation of Threads 598

12.3 Implementing Synchronization 603
   12.3.1 Busy-Wait Synchronization 604
   12.3.2 Nonblocking Algorithms 607
   12.3.3 Memory Consistency Models 610
   12.3.4 Scheduler Implementation 613
   12.3.5 Semaphores 617

12.4 Language-Level Mechanisms 619
   12.4.1 Monitors 619
   12.4.2 Conditional Critical Regions 624
   12.4.3 Synchronization in Java 626
12.4.4 Transactional Memory 629
12.4.5 Implicit Synchronization 633

12.5 Message Passing 637
  12.5.1 Naming Communication Partners 263
  12.5.2 Sending 267
  12.5.3 Receiving 272
  12.5.4 Remote Procedure Call 278

12.6 Summary and Concluding Remarks 638

12.7 Exercises 640

12.8 Explorations 645

12.9 Bibliographic Notes 647

13 Scripting Languages 649

13.1 What Is a Scripting Language? 650
  13.1.1 Common Characteristics 652

13.2 Problem Domains 655
  13.2.1 Shell (Command) Languages 655
  13.2.2 Text Processing and Report Generation 663
  13.2.3 Mathematics and Statistics 667
  13.2.4 “Glue” Languages and General-Purpose Scripting 668
  13.2.5 Extension Languages 676

13.3 Scripting the World Wide Web 680
  13.3.1 CGI Scripts 680
  13.3.2 Embedded Server-Side Scripts 681
  13.3.3 Client-Side Scripts 686
  13.3.4 Java Applets 686
  13.3.5 XSLT 287 689

13.4 Innovative Features 691
  13.4.1 Names and Scopes 691
  13.4.2 String and Pattern Manipulation 696
  13.4.3 Data Types 704
  13.4.4 Object Orientation 710

13.5 Summary and Concluding Remarks 717

13.6 Exercises 718

13.7 Explorations 723

13.8 Bibliographic Notes 724
IV  A CLOSER LOOK AT IMPLEMENTATION

14 Building a Runnable Program

14.1 Back-End Compiler Structure
   14.1.1 A Plausible Set of Phases
   14.1.2 Phases and Passes

14.2 Intermediate Forms
   14.2.1 Diana
   14.2.2 The gcc IFs
   14.2.3 Stack-Based Intermediate Forms

14.3 Code Generation
   14.3.1 An Attribute Grammar Example
   14.3.2 Register Allocation

14.4 Address Space Organization

14.5 Assembly
   14.5.1 Emitting Instructions
   14.5.2 Assigning Addresses to Names

14.6 Linking
   14.6.1 Relocation and Name Resolution
   14.6.2 Type Checking

14.7 Dynamic Linking
   14.7.1 Position-Independent Code
   14.7.2 Fully Dynamic (Lazy) Linking

14.8 Summary and Concluding Remarks

14.9 Exercises

14.10 Explorations

14.11 Bibliographic Notes

15 Run-time Program Management

15.1 Virtual Machines
   15.1.1 The Java Virtual Machine
   15.1.2 The Common Language Infrastructure

15.2 Late Binding of Machine Code
   15.2.1 Just-in-Time and Dynamic Compilation
   15.2.2 Binary Translation