

Contents

Foreword	xxiii
Preface	xxv
FOUNDATIONS	3
I Introduction	5
1.1 The Art of Language Design	7
1.2 The Programming Language Spectrum	11
1.3 Why Study Programming Languages?	14
1.4 Compilation and Interpretation	17
1.5 Programming Environments	24
1.6 An Overview of Compilation	26
1.6.1 Lexical and Syntax Analysis	28
1.6.2 Semantic Analysis and Intermediate Code Generation	32
1.6.3 Target Code Generation	34
1.6.4 Code Improvement	36
1.7 Summary and Concluding Remarks	37
1.8 Exercises	38
1.9 Explorations	39
1.10 Bibliographic Notes	40
2 Programming Language Syntax	43
2.1 Specifying Syntax: Regular Expressions and Context-Free Grammars	44
2.1.1 Tokens and Regular Expressions	45
2.1.2 Context-Free Grammars	48
2.1.3 Derivations and Parse Trees	50

2.2 Scanning	54
2.2.1 Generating a Finite Automaton	56
2.2.2 Scanner Code	61
2.2.3 Table-Driven Scanning	65
2.2.4 Lexical Errors	65
2.2.5 Pragmas	67
2.3 Parsing	69
2.3.1 Recursive Descent	73
2.3.2 Writing an LL(1) Grammar	79
2.3.3 Table-Driven Top-Down Parsing	82
2.3.4 Bottom-Up Parsing	89
2.3.5 Syntax Errors	C-1 • 102
2.4 Theoretical Foundations	C-13 • 103
2.4.1 Finite Automata	C-13
2.4.2 Push-Down Automata	C-18
2.4.3 Grammar and Language Classes	C-19
2.5 Summary and Concluding Remarks	104
2.6 Exercises	105
2.7 Explorations	112
2.8 Bibliographic Notes	112
3 Names, Scopes, and Bindings	115
3.1 The Notion of Binding Time	116
3.2 Object Lifetime and Storage Management	118
3.2.1 Static Allocation	119
3.2.2 Stack-Based Allocation	120
3.2.3 Heap-Based Allocation	122
3.2.4 Garbage Collection	124
3.3 Scope Rules	125
3.3.1 Static Scoping	126
3.3.2 Nested Subroutines	127
3.3.3 Declaration Order	130
3.3.4 Modules	135
3.3.5 Module Types and Classes	139
3.3.6 Dynamic Scoping	142
3.4 Implementing Scope	C-26 • 144
3.4.1 Symbol Tables	C-26
3.4.2 Association Lists and Central Reference Tables	C-31

3.5 The Meaning of Names within a Scope		145
3.5.1 Aliases		145
3.5.2 Overloading		147
3.6 The Binding of Referencing Environments		152
3.6.1 Subroutine Closures		153
3.6.2 First-Class Values and Unlimited Extent		155
3.6.3 Object Closures		157
3.6.4 Lambda Expressions		159
3.7 Macro Expansion		162
3.8 Separate Compilation	C-36	165
3.8.1 Separate Compilation in C	C-37	
3.8.2 Packages and Automatic Header Inference	C-40	
3.8.3 Module Hierarchies	C-41	
3.9 Summary and Concluding Remarks		165
3.10 Exercises		167
3.11 Explorations		175
3.12 Bibliographic Notes		177
4 Semantic Analysis		179
4.1 The Role of the Semantic Analyzer		180
4.2 Attribute Grammars		184
4.3 Evaluating Attributes		187
4.4 Action Routines		195
4.5 Space Management for Attributes	C-45	200
4.5.1 Bottom-Up Evaluation	C-45	
4.5.2 Top-Down Evaluation	C-50	
4.6 Tree Grammars and Syntax Tree Decoration		201
4.7 Summary and Concluding Remarks		208
4.8 Exercises		209
4.9 Explorations		214
4.10 Bibliographic Notes		215
5 Target Machine Architecture	C-60	217
5.1 The Memory Hierarchy	C-61	
5.2 Data Representation	C-63	

5.2.1 Integer Arithmetic	C-65
5.2.2 Floating-Point Arithmetic	C-67
5.3 Instruction Set Architecture (ISA)	C-70
5.3.1 Addressing Modes	C-71
5.3.2 Conditions and Branches	C-72
5.4 Architecture and Implementation	C-75
5.4.1 Microprogramming	C-76
5.4.2 Microprocessors	C-77
5.4.3 RISC	C-77
5.4.4 Multithreading and Multicore	C-78
5.4.5 Two Example Architectures: The x86 and ARM	C-80
5.5 Compiling for Modern Processors	C-88
5.5.1 Keeping the Pipeline Full	C-89
5.5.2 Register Allocation	C-93
5.6 Summary and Concluding Remarks	C-98
5.7 Exercises	C-100
5.8 Explorations	C-104
5.9 Bibliographic Notes	C-105



CORE ISSUES IN LANGUAGE DESIGN

221

6 Control Flow	223
6.1 Expression Evaluation	224
6.1.1 Precedence and Associativity	226
6.1.2 Assignments	229
6.1.3 Initialization	238
6.1.4 Ordering within Expressions	240
6.1.5 Short-Circuit Evaluation	243
6.2 Structured and Unstructured Flow	246
6.2.1 Structured Alternatives to goto	247
6.2.2 Continuations	250
6.3 Sequencing	252
6.4 Selection	253
6.4.1 Short-Circuited Conditions	254
6.4.2 Case/Switch Statements	256
6.5 Iteration	261

6.5.1 Enumeration-Controlled Loops	262
6.5.2 Combination Loops	266
6.5.3 Iterators	268
6.5.4 Generators in Icon	C-107 • 274
6.5.5 Logically Controlled Loops	275
6.6 Recursion	277
6.6.1 Iteration and Recursion	277
6.6.2 Applicative- and Normal-Order Evaluation	282
6.7 Nondeterminacy	C-110 • 283
6.8 Summary and Concluding Remarks	284
6.9 Exercises	286
6.10 Explorations	292
6.11 Bibliographic Notes	294
7 Type Systems	297
7.1 Overview	298
7.1.1 The Meaning of “Type”	300
7.1.2 Polymorphism	302
7.1.3 Orthogonality	302
7.1.4 Classification of Types	305
7.2 Type Checking	312
7.2.1 Type Equivalence	313
7.2.2 Type Compatibility	320
7.2.3 Type Inference	324
7.2.4 Type Checking in ML	326
7.3 Parametric Polymorphism	331
7.3.1 Generic Subroutines and Classes	333
7.3.2 Generics in C++, Java, and C#	C-119 • 339
7.4 Equality Testing and Assignment	340
7.5 Summary and Concluding Remarks	342
7.6 Exercises	344
7.7 Explorations	347
7.8 Bibliographic Notes	348
8 Composite Types	351
8.1 Records (Structures)	351

8.1.1 Syntax and Operations	352
8.1.2 Memory Layout and Its Impact	353
8.1.3 Variant Records (Unions)	C-136 · 357
8.2 Arrays	359
8.2.1 Syntax and Operations	359
8.2.2 Dimensions, Bounds, and Allocation	363
8.2.3 Memory Layout	368
8.3 Strings	375
8.4 Sets	376
8.5 Pointers and Recursive Types	377
8.5.1 Syntax and Operations	378
8.5.2 Dangling References	C-144 · 388
8.5.3 Garbage Collection	389
8.6 Lists	398
8.7 Files and Input/Output	C-148 · 401
8.7.1 Interactive I/O	C-148
8.7.2 File-Based I/O	C-149
8.7.3 Text I/O	C-151
8.8 Summary and Concluding Remarks	402
8.9 Exercises	404
8.10 Explorations	409
8.11 Bibliographic Notes	410
9 Subroutines and Control Abstraction	411
9.1 Review of Stack Layout	412
9.2 Calling Sequences	414
9.2.1 Displays	C-163 · 417
9.2.2 Stack Case Studies: LLVM on ARM; gcc on x86	C-167 · 417
9.2.3 Register Windows	C-177 · 419
9.2.4 In-Line Expansion	419
9.3 Parameter Passing	422
9.3.1 Parameter Modes	423
9.3.2 Call by Name	C-180 · 433
9.3.3 Special-Purpose Parameters	433
9.3.4 Function Returns	438
9.4 Exception Handling	440

9.4.1	Defining Exceptions	444
9.4.2	Exception Propagation	445
9.4.3	Implementation of Exceptions	447
9.5	Coroutines	450
9.5.1	Stack Allocation	453
9.5.2	Transfer	454
9.5.3	Implementation of Iterators	C-183 • 456
9.5.4	Discrete Event Simulation	C-187 • 456
9.6	Events	456
9.6.1	Sequential Handlers	457
9.6.2	Thread-Based Handlers	459
9.7	Summary and Concluding Remarks	461
9.8	Exercises	462
9.9	Explorations	467
9.10	Bibliographic Notes	468
10	Data Abstraction and Object Orientation	471
10.1	Object-Oriented Programming	473
10.1.1	Classes and Generics	481
10.2	Encapsulation and Inheritance	485
10.2.1	Modules	486
10.2.2	Classes	488
10.2.3	Nesting (Inner Classes)	490
10.2.4	Type Extensions	491
10.2.5	Extending without Inheritance	494
10.3	Initialization and Finalization	495
10.3.1	Choosing a Constructor	496
10.3.2	References and Values	498
10.3.3	Execution Order	502
10.3.4	Garbage Collection	504
10.4	Dynamic Method Binding	505
10.4.1	Virtual and Nonvirtual Methods	508
10.4.2	Abstract Classes	508
10.4.3	Member Lookup	509
10.4.4	Object Closures	513
10.5	Mix-In Inheritance	516
10.5.1	Implementation	517
10.5.2	Extensions	519

10.6 True Multiple Inheritance	C-194 • 521
10.6.1 Semantic Ambiguities	C-196
10.6.2 Replicated Inheritance	C-200
10.6.3 Shared Inheritance	C-201
10.7 Object-Oriented Programming Revisited	522
10.7.1 The Object Model of Smalltalk	C-204 • 523
10.8 Summary and Concluding Remarks	524
10.9 Exercises	525
10.10 Explorations	528
10.11 Bibliographic Notes	529



ALTERNATIVE PROGRAMMING MODELS 533

11 Functional Languages	535
11.1 Historical Origins	536
11.2 Functional Programming Concepts	537
11.3 A Bit of Scheme	539
11.3.1 Bindings	542
11.3.2 Lists and Numbers	543
11.3.3 Equality Testing and Searching	544
11.3.4 Control Flow and Assignment	545
11.3.5 Programs as Lists	547
11.3.6 Extended Example: DFA Simulation in Scheme	548
11.4 A Bit of OCaml	550
11.4.1 Equality and Ordering	553
11.4.2 Bindings and Lambda Expressions	554
11.4.3 Type Constructors	555
11.4.4 Pattern Matching	559
11.4.5 Control Flow and Side Effects	563
11.4.6 Extended Example: DFA Simulation in OCaml	565
11.5 Evaluation Order Revisited	567
11.5.1 Strictness and Lazy Evaluation	569
11.5.2 I/O: Streams and Monads	571
11.6 Higher-Order Functions	576
11.7 Theoretical Foundations	C-212 • 580
11.7.1 Lambda Calculus	C-214

11.7.2 Control Flow	c-217
11.7.3 Structures	c-219
11.8 Functional Programming in Perspective	581
11.9 Summary and Concluding Remarks	583
11.10 Exercises	584
11.11 Explorations	589
11.12 Bibliographic Notes	590
12 Logic Languages	591
12.1 Logic Programming Concepts	592
12.2 Prolog	593
12.2.1 Resolution and Unification	595
12.2.2 Lists	596
12.2.3 Arithmetic	597
12.2.4 Search/Execution Order	598
12.2.5 Extended Example: Tic-Tac-Toe	600
12.2.6 Imperative Control Flow	604
12.2.7 Database Manipulation	607
12.3 Theoretical Foundations	c-226 · 612
12.3.1 Clausal Form	c-227
12.3.2 Limitations	c-228
12.3.3 Skolemization	c-230
12.4 Logic Programming in Perspective	613
12.4.1 Parts of Logic Not Covered	613
12.4.2 Execution Order	613
12.4.3 Negation and the "Closed World" Assumption	615
12.5 Summary and Concluding Remarks	616
12.6 Exercises	618
12.7 Explorations	620
12.8 Bibliographic Notes	620
13 Concurrency	623
13.1 Background and Motivation	624
13.1.1 The Case for Multithreaded Programs	627
13.1.2 Multiprocessor Architecture	631
13.2 Concurrent Programming Fundamentals	635

13.2.1	Communication and Synchronization	635
13.2.2	Languages and Libraries	637
13.2.3	Thread Creation Syntax	638
13.2.4	Implementation of Threads	647
13.3	Implementing Synchronization	652
13.3.1	Busy-Wait Synchronization	653
13.3.2	Nonblocking Algorithms	657
13.3.3	Memory Consistency	659
13.3.4	Scheduler Implementation	663
13.3.5	Semaphores	667
13.4	Language-Level Constructs	669
13.4.1	Monitors	669
13.4.2	Conditional Critical Regions	674
13.4.3	Synchronization in Java	676
13.4.4	Transactional Memory	679
13.4.5	Implicit Synchronization	683
13.5	Message Passing	C-235 • 687
13.5.1	Naming Communication Partners	C-235
13.5.2	Sending	C-239
13.5.3	Receiving	C-244
13.5.4	Remote Procedure Call	C-249
13.6	Summary and Concluding Remarks	688
13.7	Exercises	690
13.8	Explorations	695
13.9	Bibliographic Notes	697
14	Scripting Languages	699
14.1	What Is a Scripting Language?	700
14.1.1	Common Characteristics	701
14.2	Problem Domains	704
14.2.1	Shell (Command) Languages	705
14.2.2	Text Processing and Report Generation	712
14.2.3	Mathematics and Statistics	717
14.2.4	“Glue” Languages and General-Purpose Scripting	718
14.2.5	Extension Languages	724
14.3	Scripting the World Wide Web	727
14.3.1	CGI Scripts	728
14.3.2	Embedded Server-Side Scripts	729

14.3.3 Client-Side Scripts	734
14.3.4 Java Applets and Other Embedded Elements	734
14.3.5 XSLT	C-258 • 736
14.4 Innovative Features	738
14.4.1 Names and Scopes	739
14.4.2 String and Pattern Manipulation	743
14.4.3 Data Types	751
14.4.4 Object Orientation	757
14.5 Summary and Concluding Remarks	764
14.6 Exercises	765
14.7 Explorations	769
14.8 Bibliographic Notes	771

IV

A CLOSER LOOK AT IMPLEMENTATION

773

15 Building a Runnable Program	775
15.1 Back-End Compiler Structure	775
15.1.1 A Plausible Set of Phases	776
15.1.2 Phases and Passes	780
15.2 Intermediate Forms	780
15.2.1 GIMPLE and RTL	C-273 • 782
15.2.2 Stack-Based Intermediate Forms	782
15.3 Code Generation	784
15.3.1 An Attribute Grammar Example	785
15.3.2 Register Allocation	787
15.4 Address Space Organization	790
15.5 Assembly	792
15.5.1 Emitting Instructions	794
15.5.2 Assigning Addresses to Names	796
15.6 Linking	797
15.6.1 Relocation and Name Resolution	798
15.6.2 Type Checking	799
15.7 Dynamic Linking	C-279 • 800
15.7.1 Position-Independent Code	C-280
15.7.2 Fully Dynamic (Lazy) Linking	C-282

15.8 Summary and Concluding Remarks	802
15.9 Exercises	803
15.10 Explorations	805
15.11 Bibliographic Notes	806
16 Run-Time Program Management	807
16.1 Virtual Machines	810
16.1.1 The Java Virtual Machine	812
16.1.2 The Common Language Infrastructure	C-286 · 820
16.2 Late Binding of Machine Code	822
16.2.1 Just-in-Time and Dynamic Compilation	822
16.2.2 Binary Translation	828
16.2.3 Binary Rewriting	833
16.2.4 Mobile Code and Sandboxing	835
16.3 Inspection/Introspection	837
16.3.1 Reflection	837
16.3.2 Symbolic Debugging	845
16.3.3 Performance Analysis	848
16.4 Summary and Concluding Remarks	850
16.5 Exercises	851
16.6 Explorations	853
16.7 Bibliographic Notes	854
17 Code Improvement	C-297 · 857
17.1 Phases of Code Improvement	C-299
17.2 Peephole Optimization	C-301
17.3 Redundancy Elimination in Basic Blocks	C-304
17.3.1 A Running Example	C-305
17.3.2 Value Numbering	C-307
17.4 Global Redundancy and Data Flow Analysis	C-312
17.4.1 SSA Form and Global Value Numbering	C-312
17.4.2 Global Common Subexpression Elimination	C-315
17.5 Loop Improvement I	C-323
17.5.1 Loop Invariants	C-323
17.5.2 Induction Variables	C-325
17.6 Instruction Scheduling	C-328

17.7 Loop Improvement II	C-332
17.7.1 Loop Unrolling and Software Pipelining	C-332
17.7.2 Loop Reordering	C-337
17.8 Register Allocation	C-344
17.9 Summary and Concluding Remarks	C-348
17.10 Exercises	C-349
17.11 Explorations	C-353
17.12 Bibliographic Notes	C-354
A Programming Languages Mentioned	859
B Language Design and Language Implementation	871
C Numbered Examples	877
Bibliography	891
Index	911