

# Table of Contents

## Preface v

### Chapter 1

#### Introduction

- 1.1 The Origins of Programming Languages . . . . . 3
- 1.2 Abstractions in Programming Languages . . . . . 8
- 1.3 Computational Paradigms . . . . . 15
- 1.4 Language Definition . . . . . 16
- 1.5 Language Translation . . . . . 18
- 1.6 The Future of Programming Languages . . . . . 19

### Chapter 2

#### Language Design Criteria

- 2.1 Historical Overview . . . . . 27
- 2.2 Efficiency . . . . . 28
- 2.3 Regularity . . . . . 30
- 2.4 Security . . . . . 33
- 2.5 Extensibility . . . . . 34
- 2.6 C++: An Object-Oriented Extension of C . . . . . 35
- 2.7 Python: A General-Purpose Scripting Language . . . . . 38

### Chapter 3

#### Functional Programming

- 3.1 Programs as Functions . . . . . 47
- 3.2 Scheme: A Dialect of Lisp . . . . . 50
- 3.3 ML: Functional Programming with Static Typing . . . . . 65
- 3.4 Delayed Evaluation . . . . . 77
- 3.5 Haskell—A Fully Curried Lazy Language with Overloading . . . . . 81
- 3.6 The Mathematics of Functional Programming: Lambda Calculus . . . . . 90

### Chapter 4

#### Logic Programming

- 4.1 Logic and Logic Programs . . . . . 105
- 4.2 Horn Clauses . . . . . 109
- 4.3 Resolution and Unification . . . . . 111
- 4.4 The Language Prolog . . . . . 115
- 4.5 Problems with Logic Programming . . . . . 126
- 4.6 Curry: A Functional Logic Language . . . . . 131

### Chapter 5

#### Object-Oriented Programming

- 5.1 Software Reuse and Independence . . . . . 143
- 5.2 Smalltalk . . . . . 144
- 5.3 Java . . . . . 162
- 5.4 C++ . . . . . 181
- 5.5 Design Issues in Object-Oriented Languages . . . . . 191
- 5.6 Implementation Issues in Object-Oriented Languages . . . . . 195

### Chapter 6

#### Syntax

- 6.1 Lexical Structure of Programming Languages . . . . . 204
- 6.2 Context-Free Grammars and BNFs . . . . . 208
- 6.3 Parse Trees and Abstract Syntax Trees . . . . . 213
- 6.4 Ambiguity, Associativity, and Precedence . . . . . 216
- 6.5 EBNFs and Syntax Diagrams . . . . . 220
- 6.6 Parsing Techniques and Tools . . . . . 224
- 6.7 Lexics vs. Syntax vs. Semantics . . . . . 235
- 6.8 Case Study: Building a Syntax Analyzer for TinyAda . . . . . 237

### Chapter 7

#### Basic Semantics

- 7.1 Attributes, Binding, and Semantic Functions . . . . . 257
- 7.2 Declarations, Blocks, and Scope . . . . . 260
- 7.3 The Symbol Table . . . . . 269
- 7.4 Name Resolution and Overloading . . . . . 282
- 7.5 Allocation, Lifetimes, and the Environment . . . . . 289
- 7.6 Variables and Constants . . . . . 297
- 7.7 Aliases, Dangling References, and Garbage . . . . . 303
- 7.8 Case Study: Initial Static Semantic Analysis of TinyAda . . . . . 309

## Chapter 8

**Data Types**

- 8.1 Data Types and Type Information . . . . . 328
- 8.2 Simple Types . . . . . 332
- 8.3 Type Constructors . . . . . 335
- 8.4 Type Nomenclature in Sample Languages . . . . . 349
- 8.5 Type Equivalence. . . . . 352
- 8.6 Type Checking . . . . . 359
- 8.7 Type Conversion . . . . . 364
- 8.8 Polymorphic Type Checking . . . . . 367
- 8.9 Explicit Polymorphism . . . . . 376
- 8.10 Case Study: Type Checking in TinyAda . . . 382

## Chapter 9

**Control I—Expressions and Statements**

- 9.1 Expressions . . . . . 403
- 9.2 Conditional Statements and Guards . . . 410
- 9.3 Loops and Variations on WHILE . . . . . 417
- 9.4 The GOTO Controversy and Loop Exits . . 420
- 9.5 Exception Handling. . . . . 423
- 9.6 Case Study: Computing the Values of Static Expressions in TinyAda. . . . . 432

## Chapter 10

**Control II—Procedures and Environments**

- 10.1 Procedure Definition and Activation. . . . 445
- 10.2 Procedure Semantics. . . . . 447
- 10.3 Parameter-Passing Mechanisms. . . . . 451
- 10.4 Procedure Environments, Activations, and Allocation . . . . . 459
- 10.5 Dynamic Memory Management. . . . . 473
- 10.6 Exception Handling and Environments. . . 477
- 10.7 Case Study: Processing Parameter Modes in TinyAda . . . . . 479

## Chapter 11

**Abstract Data Types and Modules**

- 11.1 The Algebraic Specification of Abstract Data Types. . . . . 494
- 11.2 Abstract Data Type Mechanisms and Modules . . . . . 498
- 11.3 Separate Compilation in C, C++ Namespaces, and Java Packages. . . . . 502
- 11.4 Ada Packages . . . . . 509
- 11.5 Modules in ML . . . . . 515
- 11.6 Modules in Earlier Languages . . . . . 519
- 11.7 Problems with Abstract Data Type Mechanisms . . . . . 524
- 11.8 The Mathematics of Abstract Data Types 532

## Chapter 12

**Formal Semantics**

- 12.1 A Sample Small Language. . . . . 543
- 12.2 Operational Semantics . . . . . 547
- 12.3 Denotational Semantics . . . . . 556
- 12.4 Axiomatic Semantics . . . . . 565
- 12.5 Proofs of Program Correctness . . . . . 571

## Chapter 13

**Parallel Programming**

- 13.1 Introduction to Parallel Processing. . . . 583
- 13.2 Parallel Processing and Programming Languages . . . . . 587
- 13.3 Threads . . . . . 595
- 13.4 Semaphores . . . . . 604
- 13.5 Monitors . . . . . 608
- 13.6 Message Passing . . . . . 615
- 13.7 Parallelism in Non-Imperative Languages . . . . . 622