

PEARSON

ML

UNIX

PERL

C++

FORTRAN

PASCAL

Programming Languages

Design and Implementation

Fourth Edition

Ferrance W. Pratt

Marvin V. Zelkowitz • T. V. Gopal

This edition is manufactured in India and is authorized for sale only in India, Bangladesh, Bhutan, Pakistan, Nepal, Sri Lanka and the Maldives. Circulation of this edition outside of these territories is UNAUTHORIZED.

Contents

Preface

1 Language Design Issues

1.1	Why Study Programming Languages?	1
1.2	A Short History of Programming Languages	3
1.2.1	Development of Early Languages	6
1.2.2	Evolution of Software Architectures	8
1.2.3	Application Domains	14
1.3	The Impact of Programming Paradigms	16
1.3.1	Problem Solving	17
1.3.2	What is a Programming Language?	19
1.3.3	Software Design	21
1.4	Role of Programming Languages	22
1.4.1	What Makes a Good Language?	23
1.4.2	Language Paradigms	28
1.4.3	Language Standardization	32
1.4.4	Internationalization	35
1.5	Programming Environments	36
1.5.1	Effects on Language Design	36
1.5.2	Environment Frameworks	38
1.5.3	Job Control and Process Languages	38
1.6	C Overview	39
1.7	Suggestions for Further Reading	41
1.8	Problems	42

2 Impact of Machine Architectures

2.1	The Operation of a Computer	45
2.1.1	Computer Hardware	46
2.1.2	Firmware Computers	50
2.1.3	Translators and Virtual Architectures	52

Syntax semantics & V.C. computers

2.2	Virtual Computers and Binding Times	
2.2.1	Virtual Computers and Language Implementations	
2.2.2	✓ Hierarchies of Virtual Machines	
2.2.3	✓ Binding and Binding Time	
2.2.4	Java Overview	
2.3	Suggestions for Further Reading	
2.4	Problems	
3	Language Translation Issues	
3.1	Programming Language Syntax	
3.1.1	General Syntactic Criteria	
3.1.2	Syntactic Elements of a Language	
3.1.3	Overall Program-Subprogram Structure	
3.2	Stages in Translation	
3.2.1	Analysis of the Source Program	
3.2.2	Synthesis of the Object Program	
3.3	Formal Translation Models	
3.3.1	BNF Grammars	
3.3.2	Finite-State Automata	
3.3.3	Perl Overview	
3.3.4	Pushdown Automata	
3.3.5	General Parsing Algorithms	
3.4	Language Translation in Perspective	
3.5	Recursive Descent Parsing	
3.6	Pascal Overview	
3.7	Suggestions for Further Reading	
3.8	Problems	
4	Modeling Language Properties	
4.1	Formal Properties of Languages	
4.1.1	Chomsky Hierarchy	
4.1.2	Undecidability	
4.1.3	Algorithm Complexity	
4.2	Language Semantics	
4.2.1	Attribute Grammars	
4.2.2	Denotational Semantics	
4.2.3	ML Overview	
4.2.4	Program Verification	
4.2.5	Algebraic Data Types	
4.3	Program Validation	
4.4	Suggestions for Further Reading	
4.5	Problems	

5	Elementary Data Types	145
5.1	Properties of Types and Objects	146
5.1.1	Data Objects, Variables, and Constants	146
5.1.2	Data Types	150
5.1.3	Declarations	155
5.1.4	Type Checking and Type Conversion	156
5.1.5	Assignment and Initialization	161
5.2	Scalar Data Types	163
5.2.1	Numeric Data Types	164
5.2.2	Enumerations	170
5.2.3	Booleans	172
5.2.4	Characters	173
5.3	Composite Data Types	173
5.3.1	Character Strings	173
5.3.2	Pointers and Programmer-Constructed Data Objects	176
5.3.3	Files and Input-Output	178
5.4	FORTRAN Overview	183
5.5	Suggestions for Further Reading	184
5.6	Problems	184
6	Encapsulation	189
6.1	Structured Data Types	190
6.1.1	Structured Data Objects and Data Types	190
6.1.2	Specification of Data Structure Types	191
6.1.3	Implementation of Data Structure Types	192
6.1.4	Declarations and Type Checking for Data Structures	196
6.1.5	Vectors and Arrays	197
6.1.6	Records	206
6.1.7	Lists	212
6.1.8	Sets	215
6.1.9	Executable Data Objects	218
6.2	Abstract Data Types	218
6.2.1	Evolution of the Data Type Concept	218
6.2.2	Information Hiding	219
6.3	Encapsulation by Subprograms	221
6.3.1	Subprograms as Abstract Operations	221
6.3.2	Subprogram Definition and Invocation	223
6.3.3	Subprogram Definitions as Data Objects	227
6.4	Encapsulation and Good Program Design	228
6.5	Type Definitions (Earlier Section Number 6.4.1)	230
6.5.1	Type Equivalence	232
6.5.2	Type Definitions with Parameters	235
6.6	C++ Overview	236

6.7 Suggestions for Further Reading

6.8 Problems

7 Inheritance

7.1 Abstract Data Types Revisited

7.2 Inheritance

7.2.1 Derived Classes

7.2.2 Methods

7.2.3 Abstract Classes

7.2.4 Smalltalk Overview

7.2.5 Objects and Messages

7.2.6 Abstraction Concepts

7.2.7 Inheritance and Software Reuse

7.3 Polymorphism

7.4 Suggestions for Further Reading

7.5 Problems

8 Sequence Control

8.1 Implicit and Explicit Sequence Control

8.2 Sequencing with Arithmetic Expressions

8.2.1 Tree-Structure Representation

8.2.2 Execution-Time Representation

8.3 Sequence Control Between Statements

8.3.1 Basic Statements

8.3.2 Structured Sequence Control

8.3.3 Prime Programs

8.4 Sequencing with Nonarithmetic Expressions

8.4.1 Prolog Overview

8.4.2 Pattern Matching

8.4.3 Unification

8.4.4 Backtracking

8.4.5 Resolution

8.5 Suggestions for Further Reading

8.6 Problems

9 Subprogram Control

9.1 Subprogram Sequence Control

9.1.1 Simple Call-Return Subprograms

9.1.2 Recursive Subprograms

9.1.3 The Pascal Forward Declaration

9.2 Attributes of Data Control

9.2.1 Names and Referencing Environments

9.2.2 Static and Dynamic Scope

9.2.3	Block Structure	336
9.2.4	Local Data and Local Referencing Environments	338
9.3	Parameter Transmission	343
9.3.1	Actual and Formal Parameters	343
9.3.2	Methods for Transmitting Parameters	345
9.3.3	Transmission Semantics	349
9.3.4	Implementation of Parameter Transmission	350
9.4	Explicit Common Environment	359
9.4.1	Dynamic Scope	361
9.4.2	Static Scope and Block Structure	364
9.5	Suggestions for Further Reading	371
9.6	Problems	371
10	Storage Management	377
10.1	Elements Requiring Storage	378
10.2	Programmer- and System-Controlled Storage	379
10.3	Static Storage Management	380
10.4	Heap Storage Management	380
10.4.1	LISP Overview	381
10.4.2	Fixed-Size Elements	383
10.4.3	Variable-Size Elements	390
10.5	Suggestions for Further Reading	392
10.6	Problems	392
11	Distributed Processing	395
11.1	Variations on Subprogram Control	395
11.1.1	Exceptions and Exception Handlers	395
11.1.2	Coroutines	399
11.1.3	Scheduled Subprograms	401
11.2	Parallel Programming	402
11.2.1	Concurrent Execution	404
11.2.2	Guarded Commands	405
11.2.3	Ada Overview	407
11.2.4	Tasks	409
11.2.5	Synchronization of Tasks	411
11.3	Hardware Developments	420
11.3.1	Processor Design	421
11.3.2	System Design	423
11.4	Software Architecture	425
11.4.1	Persistent Data and Transaction Systems	425
11.4.2	Networks and Client-Server Computing	426
11.5	Suggestions for Further Reading	432
11.6	Problems	432

12 Network Programming

- 12.1 Desktop Publishing
 - 12.1.1 L^AT_EX Document Preparation
 - 12.1.2 WYSIWYG Editors
 - 12.1.3 Postscript
 - 12.1.4 Postscript Virtual Machine
- 12.2 The World Wide Web
 - 12.2.1 The Internet
- 12.3 Evolution of Scripting Languages
 - 12.3.1 CGI Scripts
- 12.4 Applets
 - 12.4.1 Java Applets
- 12.5 XML
- 12.6 Suggestions for Further Reading
- 12.7 Problems

A Language Summaries

- A.1 Ada
 - A.1.1 Data Objects
 - A.1.2 Sequence Control
- A.2 C
 - A.2.1 Data Objects
 - A.2.2 Sequence Control
- A.3 C++
 - A.3.1 Data Objects
 - A.3.2 Sequence Control
- A.4 FORTRAN
 - A.4.1 Data Objects
 - A.4.2 Sequence Control
- A.5 Java
 - A.5.1 Data Objects
 - A.5.2 Sequence Control
- A.6 LISP
 - A.6.1 Data Objects
 - A.6.2 Sequence Control
- A.7 ML
 - A.7.1 Data Objects
 - A.7.2 Sequence Control
- A.8 Pascal
 - A.8.1 Data Objects
 - A.8.2 Sequence Control

A.9	Perl	539
A.9.1	Data Objects	539
A.9.2	Sequence Control	540
A.10	Postscript	542
A.10.1	Data Objects	542
A.10.2	Painting Commands	545
A.11	Prolog	546
A.11.1	Data Objects	547
A.11.2	Sequence Control	548
A.12	Smalltalk	551
A.12.1	Data Objects	553
A.12.2	Sequence Control	555
A.13	Suggestions for Further Reading	559
Bibliography		561
References		565
Index		573
Addendum		590

1.1 WHY STUDY PROGRAMMING LANGUAGES?

Hundreds of different programming languages have been designed and implemented over the years. Some have become widely used and many others have been developed but are not used. Most programmers, however, never venture to use more than a few languages and most of these programming experts are not even familiar with the rest. It is not surprising, therefore, that a study of a variety of different languages that are widely used in computer environments would be of great value to the programmer who is interested in those that are implemented on a particular machine and how structures may be formed a program. This book is a study of the design and implementation of the various components of a language in detail. The goal is to look at language features independent of any particular implementation from a wide class of commonly used languages. Throughout the book we illustrate the application of these concepts in the design of 12 languages: Ada, C, C++, FORTRAN, Java, LISP, ML, Pascal, Perl, Prolog, Postscript, and Smalltalk. In addition, we also give brief summaries about other languages such as APL, BASIC, COBOL, Fortran, PL/I, and SNOBOL. Although approaching the general study of programming languages, however, it is worth understanding why one should study in such a study to a computer programmer.

There are several reasons for such a study. First, it provides you with a broad view of the various features of the "grammar" of languages and why they are designed. Second, it provides you with a broad view of the various features of the "grammar" of languages and why they are designed. Third, it provides you with a broad view of the various features of the "grammar" of languages and why they are designed. Many languages provide features that when used properly are of benefit to the programmer but when used improperly may cause large amounts of computer time to be lost. The programmer who has a good understanding of the various features of a language can avoid such a situation. A typical example is recursion—a useful programming feature that, when properly used, allows the direct implementation of elegant and efficient algorithms. When used improperly it may cause an exponential increase in execution time. The programmer who