PROGRAMMING LANGUAGES—"DESIGN AND CONSTRUCTS"

PROGRAMMING LANGUAGES—"DESIGN AND CONSTRUCTS"

[For B.E./B.Tech. (Computer Science/Information Technology), B.C.A., M.C.A., M.Sc. (Computer Science)]

By

SHARAD CHAUHAN

Assistant Professor and H.O.D.
E-Max Institute of Engg. and Technology,
Vill-Badholi, Ambala (Haryana)

UNIVERSITY SCIENCE PRESS

(An Imprint of Laxmi Publications Pvt. Ltd.)

BANGALORE • CHENNAI • COCHIN • GUWAHATI • HYDERABAD
JALANDHAR • KOLKATA • LUCKNOW • MUMBAI • RANCHI

NEW DELHI • BOSTON, USA

Published by :

UNIVERSITY SCIENCE PRESS

(An Imprint of Laxmi Publications Pvt. Ltd.)

113, Golden House, Daryaganj,
New Delhi-110002

Phone : 011-43 53 25 00
Fax : 011-43 53 25 28

www.laxmipublications.com
info@laxmipublications.com

Price : ₹ **225.00** *Only.* *First Edition :* 2013

OFFICES

✆ **Bangalore**	080-26 75 69 30	✆ **Chennai**	044-24 34 47 26
✆ **Cochin**	0484-237 70 04, 405 13 03	✆ **Guwahati**	0361-251 36 69, 251 38 81
✆ **Hyderabad**	040-24 65 23 33	✆ **Jalandhar**	0181-222 12 72
✆ **Kolkata**	033-22 27 43 84	✆ **Lucknow**	0522-220 99 16
✆ **Mumbai**	022-24 91 54 15, 24 92 78 69	✆ **Ranchi**	0651-220 44 64

UPL–9667–225–PROG LANG DESIGN CONST–CHA **C—**

Typeset at : ABRO Enterprises, Delhi. *Printed at :*

CONTENTS

PREFACE

It gives me immense pleasure in presenting the first edition of the book **Programming Languages—"Design and Constructs"** to the esteemed readers. The necessity for programming languages in Computer Science curriculum arises because of its key paradigms used in developing modern programming languages. The book covers all the topics related to subject 'Programming Languages' taught in all Engineering Institutions. It also covers all the problems appeared in university examinations in past few years.

This book contains ten chapters. These chapters contain the reasonable amount of exercises, examples with solutions. Since examples are part of the text; they are clearly explained without leaving any ambiguity in the reader's mind. The contents of the book include introduction to programming languages, elementary and structured data objects, syntax and semantics, sequence control, concurrency control, data control, subprogram sequence control, storage management related issues and overview of different programming languages like imperative, procedural, non-procedural, functional and logical programming languages.

The objective of this book is to present the subject and concepts in a clear and more interesting manner. My intention is to cover all topics in the book related to subject. The book is based on the experience gained in teaching a course on the subject. I am sure that this book will help a student to understand each topic without any difficulty. I would welcome the indulgence of my readers in intimating me about any error or shortcoming noticed by them in the book. The valuable suggestions from the readers are invited for further improvement of the book.

—Author

ACKNOWLEDGEMENT

It is difficult to acknowledge adequately the help and encouragement I have received during completion and publication of this work. First, I must express my gratitute to Sh. Om Prakash Aggarwal, Chairman, E-Max group of Institutes, Sh. P.R. Bansal, Vice Chairman, E-Max group of Institutes and all the Directors of E-Max group of Institutes, Badholi, Ambala, for their confidence building encouragement and kind suggestions in writing this book. I should also express my grateful thanks to my colleagues and friends with whom I had occasions to discuss the details of this book.

I owe my idebtedness to my family for their perpetual love and concern, great support, encouragement and prayers while I was writing this book. The main source of inspiration behind this book are my parents, my wife and my little daughter 'PARI'. I express my appreciation to them for their assistance and constant encouragement without which the venture would have failed.

In the end I would like to thank Laxmi Publications who encouraged me for development of such a project in a most desired way.

—Author

Chapter 1

INTRODUCTION

In the early 1950s, symbolic notations started to appear. Grace Hopper led a group at Univac to develop the A-0 language, and John Backus developed speedcoding for the IBM 701. Both were designed to compile simple arithmetic expressions into executable machine language. The real break through occurred in 1957 due to arrival of FORTRAN in 1957. FORTRAN data were oriented around numerical calculations.

The formula $b^2 - 4ac$ was written in FORTRAN as the expression

B ** 2 – 4.0 * A * C.

FORTRAN takes its name from Formula Translation; readable formulas were translated into machine instructions for the IBM 704. FORTRAN was extremely successful and was the dominant language through the 1970s. Because of the suceess of FORTRAN, hundreds of programming languages have since been designed and implemented.

Under the leadership of Peter Naur, the committee developed the International Algorithmic Language (IAL). Although ALGorithmic Language (ALGOL) was proposed. A revision occurred in 1960, and ALGOL 60 comes. In 1963, IBM developed NPL (New Programming Language) at its Hursley Laboratory in England. After some complaints by the English National Physical Laboratory, the name was changed to MPPL (Multi-Purpose Programming Language) which was then shortened to just PL/I. In 1960, COBOL (Common Business Oriented Language) will be designed.

1.1. CONCEPT OF PROGRAMMING LANGUAGES

Prorgramming Languages are notations or symbols. They are used for specifying, organizing and reasoning about computations. The designers of Programming Languages have two goals :

- Making computing convenient for people.
- Making efficient use of computing machines.

The first goal takes the priority, without convenience, efficiency is irrelevant.

A Programming Language have following definitions:

1. It is a tool for instructing machines.
2. It is a tool for controlling computerized devices.
3. A notation for algorithms.
4. A way of expressing relationships between concepts.
5. A tool for high level designs.

The study of programming languages is valuable for a number of important reasons. It increase our capacity to use different constructs in writing programs, enables us to choose languages for projects more intelligently and makes learning new languages easier.

1.2. WHY WE STUDY PROGRAMMING LANGUAGES

It is natural for students to wonder how they will benefit from the study of Programming Language concepts. The following is what we believe to be a compelling list of potential benefits of studying language concepts.

1. Improved Background for Choosing Appropriate Languages

A knowledge of a variety of languages may allow the choice of just the right language for a particular project. Because many programmers have had little formal education in computer science, they have learned programming on their own or through in house training programs. Such programs often teach one or two languages. When given a choice of language for a new project, continue to use the language with which they are most familiar, even if it is poorly suited to the new project. If they know the basic seat uses of each language then the programmer have broader choice of alternatives.

2. Increased Ability to Learn New Language

Through a deep understanding of the underlying structure of natural languages, often can learn a new foreign language more quickly and easily than struggling novices. The process of learning a new Programming Lanugage can be lengthy and difficult, especially for someone who is comfortable with only one or two languages. Once a thorough understanding of the fundamental concepts of languages is acquired, it becomes for easier to see how these concepts are incorporated into design of the language being learned.

3. Easy to Design a New Language

It is difficult for a programmer who have little knowledge of concepts of language to design a new programming language. However, most professional programmers occasionally do design languages. For example, most software systems require the user to interact in some way, even if only to enter data and commands. Many new languages are based on C or Pascal as implementation models. This aspect of program design is often simplified if the programmes is familiar with a variety of constructs and implementation method from ordinary programming languages.

4. Better Understanding of the Significance of Implementation

By learning the concepts of programming languages, it is both interesting and necessary to touch on the implementation issues that affect those concepts. By understanding of implementation issues leads to an understanding of why languages are designed the way they are. Many kinds of programs bugs can only be found and fixed by a programmer who knows some related implementation details.

5. To Improve your Ability to Develop Effective Algorithms

By studying programming languages, it will improve ability to develop effective algorithms. Many languages provide features, that when used properly, are of benefit to the programmer but, when used improperly, may waste large amount of computer time. For example, recussion—a handy programming feature that, when properly used, allow the direct implementation of elegant and efficient algorithms. When used improperly, it may cause an astronomical increase in execution time. New technology, such as the Internet and worldwide web, change the nature of programming. How best to develop techniques applicable in these new environments depends on an understanding of languages.

6. Overall Advancement of Computing

Finally, there is a global view of computing that can justify the study of Programming Language Concepts. Although, it is usually possible to determine why a particular programming language become popular, it is not always clear, at least in retrospect that the most popular langauges are the best available.

For example, many people believe it would have been better if AlGOL 60 had displaced FORTRAN in the early 1960, because it was more elegant and had much better control statements that FORTRAN, among other reasons.

In general, if those who choose languages are better informed, perhaps better languages would more quickly squeeze out poorer ones.

1.3. A BRIEF HISTORY OF PROGRAMMING LANGUAGES

Programming languages designs and implementation methods have evolved continuously since the earliest high level languages appeared in the 1950s. Of the 12 languages described in some detail, the first version of FORTRAN and LISP were designed during the 1950s; Ada, C, Pascal, Prolog, and smalltalk data from the 1970s; C++, ML, Perl and Postscript data from the 1980s; and Java dates from the 1990s. In the 1960s and 1970s, new languages were often developed as part of major software development projects.

We briefly summarise language development during the early days of computing, generally from the mid-1950s to the early 1970s.

FORTRAN OVERVIEW

In 1957, Backus managed a team to develop FORTRAN or FORMULA TRANSLATOR. FORTRAN data were oriented around numerical calculations, but the goal was a full-fledged programming language including control structures, conditionals and input and output statements.

FORTRAN was extremely successful and was the dominant language through the 1970s. FORTRAN was revised as FORTRAN II in 1958 and FORTRAN IV a few years later. Finally, in 1966, FORTRAN IV becomes a standard under the name FORTRAN 66 and has been upgraded twice since, to FORTRAN 77 and FORTRAN 90.

The environment in which FORTRAN was developed was as follows:

1. Computers were still small, slow and relatively unreliable.
2. The primary use of computers was for scientific computations.
3. Because of the high cost of computers compared to the cost of programmers, speed of the general object code was the primary goal of the first FORTRAN compilers.
4. There are no existing efficient ways to program computers.

ALGOL 60

ALGOL 60 has a great influence on subsequent programming language. ALGOL had very different goals:

- ALGOL notation should be close to standard mathematics.
- ALGOL should be useful for the description of algorithms.
- Programs in ALGOL should be compilable into machine language.
- ALGOL should not be bound to a single computer architecture.

The most important new developments are:

1. The concept of block structure was introduced. This allowed the programmer to localize parts of programs by introducing new data environments, or scopes.
2. Procedures were allowed to be recursive. Note that although this recursion is new for the imperative languages, LISP had already provided recursive functions in 1959.
3. Two different means of passing parameters to subprograms were allowed : Pass by value and pass by name.
4. Stack-dynamic arrays were allowed. A stack-dynamic array is one for which the subscript range or ranges are specified by variables, so that the size of the array is set at the time storage in allocated to the array, which happens when the declaration is reached during execution.

   One of the most important contribution to computer science that is associated with ALGOL 60, BNF, was also a factor in its lack of acceptace.

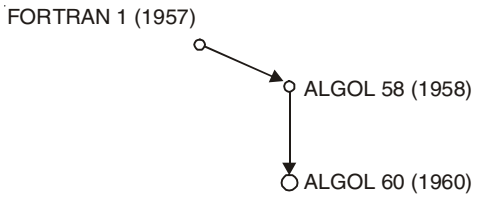


Fig. 1.1 Geneology of ALGOL 60.

PL/I

PL/I represents the first large scale attempt to design a language that could be used for a broad spectrum of application areas. PL/I merged the numerical attributes of FORTRAN with the

business programming feature of COBOL. PL/I was the first programming language to have the following facilities:

1. Programs were allowed to create concurrently executing tasks. Although this was a good idea, it was poorly developed in PL/I.
2. Pointers were included as a data type.
3. It was possible to detect and handle 23 different types of exceptions, or runtime errors.
4. Procedures were allowed to be used recursively, but the capability could be disabled, allowing more efficient code for non-recursive procedures.
5. Cross-sections of array could be referenced. For example, the third row of a matrix could be referenced as if it was a vector.

COBOL

In 1959, the U.S. department of Defense sponsored a meeting to develop common business language, which would be business-oriented language that used English as much as possible for its notation. The specifications, published in 1960, were the design for COBOL.

The COBOL language originated a number of novel concepts, some of which eventually appeared in other languages. For example, the DEFINE verb of COBOL 60 was the first high level language construct for macros. Overall, the data division is the strong part of COBOL is design, whereas the procedure division is relatively weak. Every variable is defined in detail in the data division, including the no. of decimal digits and location of the implied decimal point. The poor performance of the early compilers simply made it for too expensive to use. Eventually, of course, people learned more about designing compilers, and computers became much faster. Together, these factors have made COBOL a great success, inside and outside DOD (Department of Defence).

Functional Programming (LISP)

The major break through occurred when John McCarthy of MIT designed List Processing (LISP) for the IBM 704. LISP was designed as a list processing functional language.

Pure LISP has only two kinds of data structures atoms and list. Atoms are either symbols, which have the form of identifiers, or numeric literals. The concepts of storing symbolic information in linked list is natural.

Lists are specified by delimiting their elements with parentheses. Simple lists, in which elements are restricted to atoms, have the form of the example,

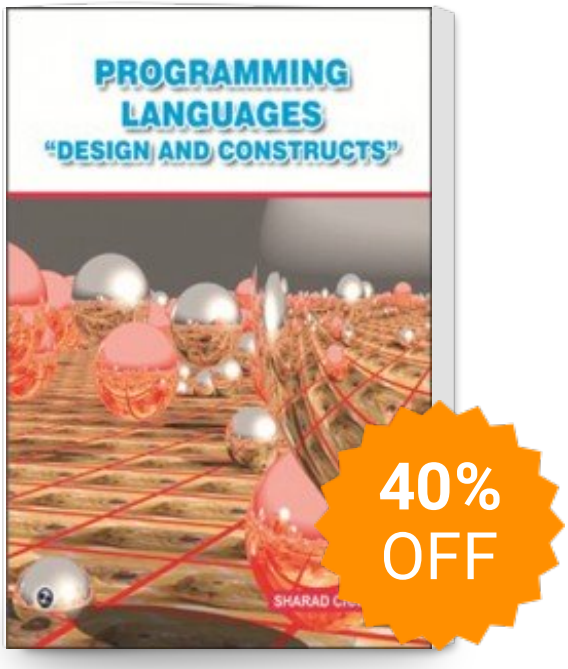
(ABCD)

LISP totally dominated AI applications for a quarter of a centuary. It is still the most widely used language for AI. In addition to its success in AI, LISP pioneered functional programming, which has proven to be a lively area of research in programming languages.

Prolog

Prolog is logical programming language. It uses the logical notations to communicate computational processes to a computer. In 1970s, Alain Colmeraner and Phillie Roussel in the AI group at the university of Aix-Marseille, developed the fundamental design of Prolog. Prolog programs consists of collection of statements. Prolog has only a few kind of statement but they can be complex. One common use of Prolog is as a kind of intelligent database.

Programming Languages - “Design And Constructs” By Sharad Chauhan



Publisher : Laxmi Publications ISBN : 9789381159415 Author : Sharad Chauhan