PROGRAMMING IN C

PROGRAMMING IN C

(For M.C.A., M.Tech., M.Sc., Engineering, B.C.A., B.I.T., B.Sc., C-DAC, DOEACC O-Level and A-Level, P.G.D.C.A. and other Computer Courses)

By

J.B. DIXIT

M.Sc. (Applied Mathematics)
Post M.Sc. Diploma in Computer Science,
Kurukshetra University, Kurukshetra
Haryana

FIREWAL MEDIA

(An Imprint of Laxmi Publications Pvt. Ltd.)

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

NEW DELHI

Published by :

FIREWAL MEDIA

(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 : **Rs. 395.00** *Only.*

Edited by : Sangeeta Dixit

First Edition : 2005
Second Edition : 2009
Third Edition : 2010

OFFICES

✆ **Bangalore**	080-26 61 15 61	✆ **Chennai**	044-24 34 47 26
✆ **Cochin**	0484-237 70 04, 405 13 03	✆ **Guwahati**	0361-254 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 95 78
✆ **Mumbai**	022-24 91 54 15, 24 92 78 69	✆ **Ranchi**	0651-221 47 64

FPR-2802-395-PROGRAMMING IN C-DIX **C—**

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

Dedicated to

All Beloved Readers, Friends
and
Family Members

CONTENTS

PREFACE TO THE THIRD EDITION

After the tremendous success of the second edition, it is a matter of great pleasure for me to write the preface for the third edition of this book. The most comprehensive, authoritative, and up-to-date book on C programming language. It provides a unique combination of programming and problem solving through 'C' language.

This book is written for M.C.A., M.Tech., M.Sc., Engineering, B.C.A., B.I.T., B.Sc., C-DAC, DOEACC O-Level and A-Level, P.G.D.C.A. and other computer programme's students. In addition it can be of great help to those who feel that programming is not their cup of tea. I am sure that the novice will feel quite comfortable after going through the subject matter due to its *practicality, readability* and *correctness.* All the programs in this book have been tested on Turbo C compiler. This book is organized into fourteen chapters as per the contents. A brief overview of all the chapters is presented below :

Chapter 1: Introduction to Problem Solving

It introduces the meaning of algorithms, flowcharts, decision tables and their need. The various problem solving methodologies and programming languages are also given.

Chapter 2: Overview of C

It includes character set, C tokens, keywords and identifiers. Structure of a C program and its execution are also given. Constants, variables and data types are also explained here.

Chapter 3: Operators and Expressions

It discusses the different types of operators available in C, their precedence and associativity. Mathematical functions, header files are also included.

Chapter 4: Input and Output Functions

It includes the input/output functions. Formatted functions—scanf( ), printf( ) and unformatted functions—getch( ), getche( ), getchar( ), gets( ), putch( ), putchar( ), puts( ) are explained in detail. Programming examples are also given.

Chapter 5: Control Statements

The *if* statement, *if-else* statement, *switch* statement, loops : *while* loop, *do while, for* loop, nested loops, infinite loops, jumps in loops (*break* and *continue* statement), *goto* statement and *exit( )* function are given. Programming examples are also given.

Chapter 6: Functions

It includes need for user defined functions. Defining and using functions, types of functions, passing arguments to a function : call by value, call by reference. Recursion and data storage classes are also discussed. Programming examples are also given.

Chapter 7: Preprocessors
It includes various preprocessor directives. Programming examples are also included.

Chapter 8: Arrays
It includes the meaning of an array, one dimensional and multidimensional (two or more dimensional) arrays. Different operations on arrays are given with the help of programming examples.

Chapter 9: Strings
It includes null terminated strings as array of characters and various operations on strings. User defined string functions are also given. Programming examples are also provided in the end of the chapter.

Chapter 10: Pointers
It discusses address operators, pointer data type declaration, pointer assignment, pointer initialization, pointer arithmetic, functions and pointers, arrays and pointers, pointer arrays. Command line arguments and dynamic manipulation of memory is provided here. Programming examples are also given.

Chapter 11: Structures and Unions
It includes structure variables, initialization, structure assignment, nested structure, structures and functions, structures and arrays : arrays of structures, structures containing arrays, bit fields in structures and self referential structures. An introduction to unions is also provided. Programming examples are also given.

Chapter 12: File Handling
It discusses concept of files, file opening in various modes and closing a file, reading from a file, writing onto a file. Random access is explained. Programming examples are also given.

Chapter 13: Linked Lists
It includes creation of a singly linked list, traversing a linked list, insertion into a linked list, deletion from a linked list and applications of linked lists. Programming examples are also given.

Chapter 14: Graphics in C
It discusses the concept of pixel, resolution. Various graphics functions are explained with appropriate examples. Animation is described to make the reader comfortable in graphics. Programming examples are also given.

I have put my sincere efforts and knowledge to make you understand the subject matter in the simplest and easiest form. A great care has been taken to avoid mistakes.

The author invites feedback from readers as in the past, which has encouraged me a lot in improving this book to a great extent. Please do send your comment and suggestions to the publisher or the author.

WISH YOU A GRAND SUCCESS in your examination, and a very bright future in the field of Computer Science.

—AUTHOR

ACKNOWLEDGEMENT

It is with a great sense of satisfaction that I acknowledge the help and support rendered to me by many people in bringing this book in its current form.

My heartiest thanks to Mr. Raman Sharma (Manager—WIPRO) for his creative and thoughtful association in the preparation of this book.

My lovely children Apoorva, Aanchal and Vansh always remind me about the work to be completed with their ever smiling faces. So, a special thank to them also.

I would like to thank all my teachers, members of my family, students and well wishers whose blessing, knowledge, advice and interaction have made this project a possible venture in my life.

—AUTHOR

Salient Features of the Present Edition

- ➢ **Motivates the unmotivated students** and provides the teachers unequaled approach that allows them to teach students with a disparity of computer experience backgrounds.
- ➢ Covers **detailed theory** supplemented with **appropriate figures, tables** and **examples.**
- ➢ Covers **algorithms, flowcharts, decision tables,** *i.e., problem-solving methodology* and *programming languages.*
- ➢ Covers abundance of **C programming examples** with proper testing.
- ➢ Makes you able to **develop and debug programs independently.**
- ➢ Covers **summary, adequate number of exercises** and **review questions** of all types.
- ➢ Provides **user-friendly approach.**

Other Books by the Same Author:

1. Mastering C++ Programs (*Published by Firewal Media*)
2. Fundamentals of Computer Programming and Information Technology
3. Programming in C and Numerical Analysis (For B.A. / B.Sc. III)
4. Solutions to Numerical Analysis
5. New Approach to CBSE Computer Science (with C++) XII
6. New Approach to CBSE Computer Science (with C++) XI
7. New Approach to Informatics Practices XI
8. New Approach to Informatics Practices XII
9. Excel with Objective Book for NIMCET (MCA Entrance Examination) (*Published by Golden Bells*)
10. Computer Concepts and Programming in C (*University Science Press*)
11. Fundamentals of Computers and Programming in C
12. Mastering C Programs (*Published by Firewal Media*)
13. Computer Programming
14. Programming in C++ (*Published by Firewal Media*)
15. Programming in C and Numerical Analysis (*Published by Firewal Media*)
16. Computer Fundamentals and Programming in C (*For B.Sc. I*)
17. Digital Design and Computer Organisation (*Published by University Science Press*)
18. Structured System Analysis and Design (*Published by University Science Press*)
19. Comprehensive Mathematics, Mathematics Activities and Projects (Class IX)
20. Comprehensive Mathematics, Mathematics Activities and Projects (Class X)
21. Information Technology in Business Management
22. Excel with Information and Communications Technology
23. Computer Programming and Utilization
24. Mastering Data Structures Through 'C' Language
25. Intelligent Instrumentation for Engineers
26. Fundamentals of Computing and Programming
27. Mastering Java Programs
28. Fundamentals of Computers
29. Presentation Software and Computer Communication
30. Electrical Power Quality

CHAPTER 1

Introduction to Problem Solving

Introduction

A **program** is a sequence of instructions written in a programming language. There are various programming languages, each having its own advantages for program development. Generally every program takes an input, manipulates it and provides an output as shown below :

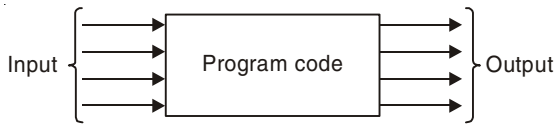


Fig. 1. A conceptual view of a program.

Some programs may have millions of lines of code. Building complex programs requires a disciplined approach. Most modern programs are too big for a person to write. When a group of people work together on a program and break it into separate parts that can be worked on independently, it is essential that everyone follow the same rules or programming methodologies. Over the years, there have been numerous attempts to develop methodologies that make programming more a science than art.

Planning the Computer Program

For better designing of a program, a systematic planning must be done. Planning makes a program more **efficient** and more **effective.** A programmer should use planning tools before coding a program. By doing so, all the instructions are properly interrelated in the program code and the logical errors are minimized.

Problem solving is one of the most significant advantages of a computer. Before writing programs, it is a good practice to understand the complete problem, analyse the various solutions and arrive at the best solution. Figure 2 illustrates the problem-solving logic.

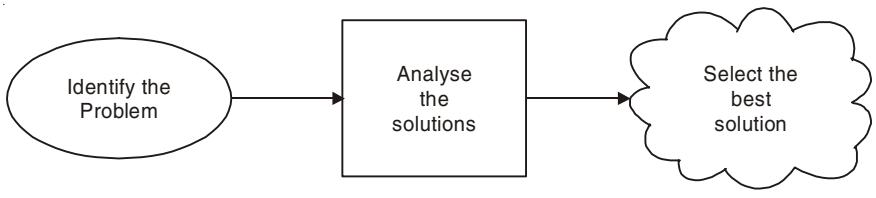


Fig. 2. Problem-solving logic.

There are various planning tools for mapping the program logic, such as **algorithms, flowcharts, pseudocode, decision tables** and **hierarchy charts** etc. A program that does the desired work and achieves the goal is called an effective program whereas the program that does the work at a faster rate is called an efficient program.

The software designing includes mainly two things—*program structure* and *program representation*. The program structure means how a program should be. The program structure is finalised using top-down approach or any other popular approach. The program structure is obtained by joining the subprograms. Each subprogram represents a logical subtask.

The program representation means its presentation style so that it is easily readable and presentable. A user friendly program (which is easy to understand) can be easily debugged and modified, if need arises. So, the programming style should be easily understood by everyone to minimize the wastage of time, efforts and cost.

Change is a way of life, so is the case with software. The modification should be easily possible with minimum efforts to suit the current needs of the organization. This modification process is known as **program maintenance.**

Flowcharting technique is quite helpful in describing program structure and explaining it. Some of the other useful techniques for actually designing the programs are :

(*i*) **Modular programming**

(*ii*) **Top-down design (Stepwise refinement)**

(*iii*) **Structured programming.**

Now let us discuss the above mentioned problem solving techniques.

Modular Approach

Breaking down of a problem into smaller independent pieces (modules) helps us to focus on a particular module of the problem more easily without worrying about the entire problem. No processing outside the module should affect the processing inside the module. It should have only one entry point and one exit point. We can easily modify a module without affecting the other modules. Using this approach the writing, debugging and testing of programs becomes easier than a monolithic program. A *modular* program is readable and easily modifiable. Once we have checked that all the modules are working properly, these are linked together by writing the main module. The main module activates the various modules in a predetermined order. For example, Figure 3 illustrates this concept :

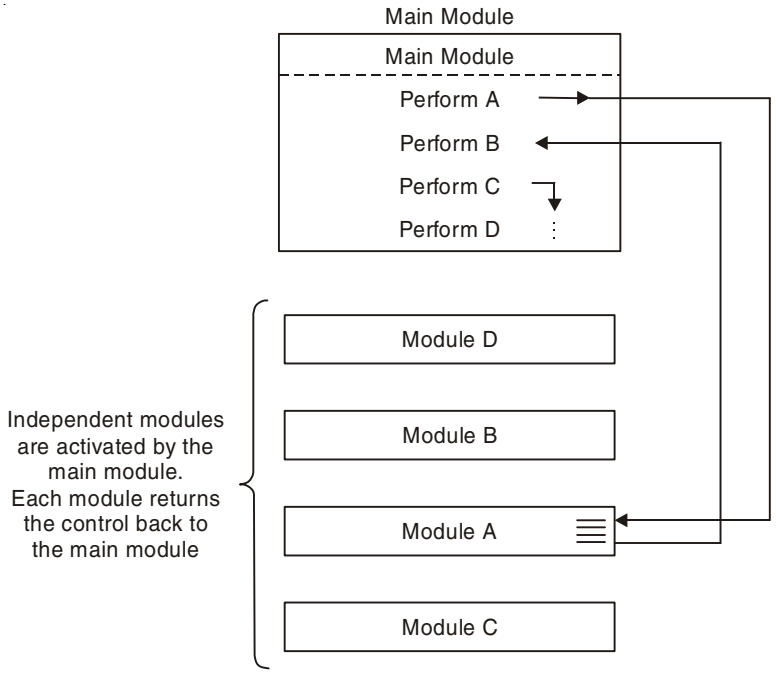


Fig. 3. Illustration of modular approach.

It must be noted that each module can be further broken into other submodules.

Characteristics of Modular Approach

The characteristics of modular approach are given below :

(*i*) The problem to be solved is broken down into major components, each of which is again broken down if required. So the process involves working from the most general, down to the most specific.

(*ii*) There is one entry and one exit point for each module.

(*iii*) In general each module should not be more than half a page long. If not so, it should be split into two or more submodules.

(*iv*) Two-way decision statement are based on IF..THEN, IF..THEN..ELSE, and nested IF structures.

(*v*) The loops are based on the consistent use of WHILE..DO and REPEAT..UNTIL loop structures.

Advantages of Modular Approach

The advantages of modular approach are given below :

(*i*) Some modules can be used in many different problems.

(*ii*) Modules being small units can be easily tested and debugged.

(*iii*) Program maintenance is easy as the malfunctioning module can be quickly identified and corrected.

(*iv*) The large project can be easily finished by dividing the modules to different programmers.

(*v*) The complex modules can be handled by experienced programmers and the simple modules by junior ones.

(*vi*) Each module can be tested independently.

(*vii*) The unfinished work of a programmer (due to some unavoidable circumstances) can be easily taken over by someone else.

(*viii*) A large problem can be easily monitored and controlled.

(*ix*) This approach is more reliable.

(*x*) Modules are quite helpful in clarification of the interfaces between major parts of the problem.

Top-down Design (Stepwise Refinement)

Program development includes designing, coding, testing and verification of a program in any computer language. For writing a good program, the top down design approach can be used. It is also called **systematic programming** or **hierarchical program design** or **stepwise refinement**. A complex problem is broken into smaller subproblems, further each subproblem is broken into a number of smaller subproblems and so on till the subproblems at the lowest level are easy to solve. Similarly a large program is broken into a number of subprograms and in turn each subprogram is further decomposed into subprograms ans so on. Suppose we want to solve a problem S, which can be decomposed into subproblems S1, S2 and S3 and so on. Let the program for S, S1, S2, S3 be denoted by P, P1, P2, P3 respectively. Further suppose that S2 is solved by decomposing it into subproblems S21 and S22 and program P21 and P22 are written for these. This operation of coding a subprogram in terms of lower level subprograms is known as the **process of stepwise refinement.** Figure 4 shows the hierarchical decomposition of P into its subprograms and sub-subprograms.

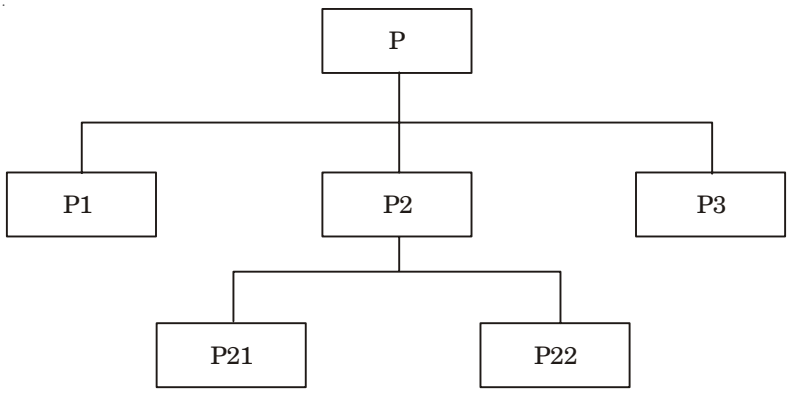


Fig. 4. Illustration of stepwise refinement.

The advantages of the top-down design approach are :

1. A large problem is divided into a number of smaller problems using this approach. The decomposition is continued till the subproblems at the lowest level become easy to solve. So the overall problem solving becomes easy.

2. If we use the top-down approach for a problem then top-down programming method can be used for coding modules at various stages. So, the top level modules can be coded without coding the lower level modules earlier. This approach, is better than the bottom-up approach where programming starts first at the lowest level modules.
3. It helps in top-down testing and debugging of programs.
4. The programs become user friendly (that is easy to read and understand) and easy to maintain and modify.
5. Different programmers can write the modules for different levels.

Structured Programming

Structured programming takes a top-down approach that breaks programs into modular forms. The main objectives of structured programming are :

- Efficiency
- Readability
- Clarity of programs
- Easy modification
- Reduced testing problems.

The **goto** statement should be avoided so far as possible. The three basic building blocks for writing structured programs are discussed below :

1. **Sequence Control Structure.** In the ***sequence control structure,*** **one program statement follows another in logical order.** There are no decisions to make, no choices between "yes" or "no". The boxes logically follow one another in sequential order. For example, Figure 5 illustrates a sequence of N statements :

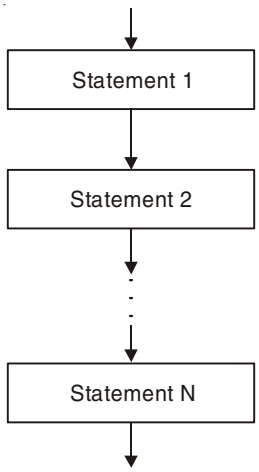
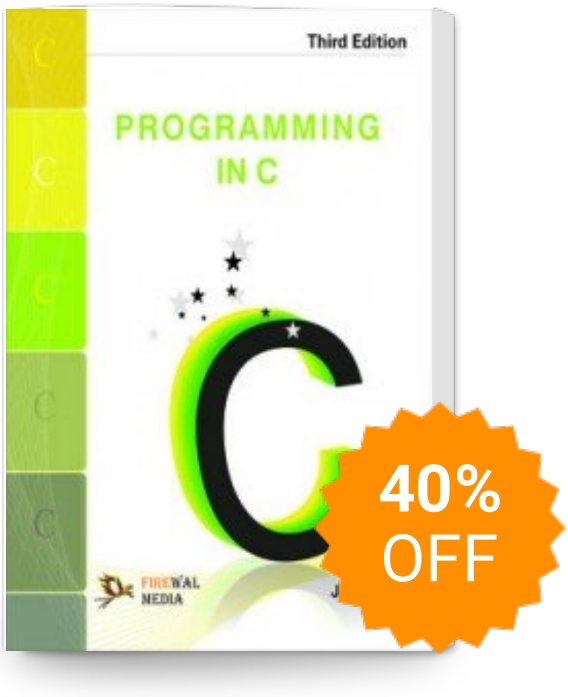


Fig. 5. Sequence control structure.

2. **Selection Control Structure.** The **selection control structure—also known as an IF-THEN-ELSE structure—represents a choice. It offers two paths to**

Programming In C By J.B. Dixit



Publisher : Laxmi Publications ISBN : 9789380298399 Author : J.B. Dixit