C is one of the most popular programming languages today. It is flexible, efficient and highly portable, and is used for writing many different kinds of programs, from compilers and assemblers to spreadsheets and games.

This book is based on ANSI C – the recently adopted standard for the C language. It assumes familiarity with basic programming concepts such as variables, constants, iteration and looping, but covers all aspects of C. In general it is as much about learning programming skills as it is about mastering the art of coding programs in C. To this end the text contains a wealth of examples and exercises that foster and test the understanding of the concepts developed in each chapter.

An outstanding feature of this book is the treatment of 'pointers'. The topic is presented in a clear logical and reasoned manner that is easy to follow. Binary files and random access files are also treated in such a manner that the reader can easily become adept at using them.

Anybody who wishes to get to grips with the art of programming in C will find this a most valuable book.

Cambridge Computer Science Texts
Edited by D. J. Cooke, Loughborough University

C by Example

Also in this series

5 An Introduction to the Uses of Computers
Murray Laver 1976
8 ALGOL 68 – A first and second course
A. D. McGettrick 1978
12 Programming via Pascal
J. S. Rohl and H. J. Barrett 1980
14 Simulation Techniques for Discrete Event Systems
I. Mitrani 1982
15 Information Representations and Manipulation using Pascal
E. S. Page and L. B. Wilson 1983
16 Writing Pascal Programs
J. S. Rohl 1983
18 Computer Mathematics
D. J. Cooke and H. E. Bez 1984
19 Recursion via Pascal
J. S. Rohl 1984
22 Program Construction
R. G. Stone and D. J. Cooke 1987
23 A Practical Introduction to Denotational Semantics
Lloyd Allison 1987
24 Modelling of Computer and Communication Systems
I. Mitrani 1987
25 The Principles of Computer Networking
D. Russel 1989
26 Concurrent Programming
C. R. Snow 1991
27 An Introduction to Functional Programming Systems Using Haskell
A. J. T. Davie 1992
28 Categories and Computer Science
R. F. C. Walters 1991

29 Cambridge Computer Science Texts

C by Example

Noel Kalicharan
University of the West Indies

CAMBRIDGE
UNIVERSITY PRESS

Published by the Press Syndicate of the University of Cambridge
The Pitt Building, Trumpington Street, Cambridge CB2 1RP
40 West 20th Street, New York, NY 10011–4211, USA
10 Stamford Road, Oakleigh, Melbourne 3166, Australia

First published 1994
Reprinted 1996

A catalogue record for this book is available from the British Library

Library of Congress cataloguing in publication data
Kalicharan, Noel.
C by example / Noel Kalicharan.
p. cm. – (Cambridge computer science texts: 29)
Includes index.
ISBN 0 521 45023 3 (hc). – ISBN 0 521 45650 9 (pb)
1. C (Computer program language) I. Title. II. Series.
QA76.73.C15835 1994
005.13'3–dc20 93-27877 CIP

ISBN 0 521 45023 3 hardback
ISBN 0 521 45650 9 paperback

Transferred to digital printing 2000

PR

To my daughters

Anushka Nikita
and
Anyara Saskia

Cambridge University Press
978-0-521-45650-0 - C by Example
Noel Kalicharan
Frontmatter
More information

Contents

Appendices

Preface

In the beginning, there was a language called BCPL. This was developed in the 1960s by Martin Richards at Cambridge University. In 1970, Ken Thompson, of Bell Laboratories, developed and implemented the language B on a DEC (Digital Equipment Corporation) PDP-7 computer running the first UNIX operating system. B was strongly influenced by BCPL. When DEC introduced their PDP-11, Dennis Ritchie (also of Bell Labs) modified B to create the language C in order to implement UNIX on the new machine.

Since those early days, C has undergone several changes. Existing features have been modified, new features have been added and some obsolete ones deleted. With the advent and proliferation of microcomputers, several implementations of C emerged. Though compatible to a great degree, there were discrepancies and anomalies in these implementations. In 1983, the American National Standards Institute (ANSI) established a committee to define a 'standard' version of the C language. This standard has been adopted by the major producers of C compilers. This book is based on ANSI C.

C has fast become one of the most popular programming languages today. Perhaps one of the reasons for its widespread popularity is its flexibility – it allows one to program in a 'structured' way yet it permits great 'freedom of expression'. It combines the control structures normally found in high-level languages such as Pascal or Ada with the ability to manipulate bits, bytes and addresses, something usually associated with assembly language. In its early days, C was thought of mainly as a language for writing systems programs – things like operating systems, editors, compilers, assemblers and input/output utility programs. But that view has changed considerably in recent times. Today, C is used for writing all kinds of applications programs as well – things like wordprocessing programs, spreadsheet programs, database management programs, accounting programs, games, educational software, etc. But flexibility is not the only reason.

C lends itself to 'modular programming'. It is easy to create 'modules' which can be treated like the proverbial 'black-box' – we need only know **what** the module does, not **how** it does it. This concept is critical to the writing of a large program, or a program which is being written by several people. A related idea is that, in C, one can create and maintain one's own 'library' of frequently used functions. In this way, duplication of effort can be kept to a minimum.

C is an 'efficient' language. The machine code produced for a C program is comparable to what would be produced if the program were written in assembly language. This is possible because many of C's features (mainly the 'operators' provided) closely resemble features of today's computers, so that translation from C to machine code is straightforward. Another reason is that C is 'small'; for example, there are only 32 keywords (reserved words) and the basic data types are simply character, integer and floating-point. In order to keep down the size of the language, C does not include features considered 'built-in' or 'standard' in other languages. For instance, there are no statements like `read` or `write` for performing input/output and no direct way of comparing two strings. These operations are provided by means of **functions** provided in a standard library.

C is highly portable. This means that a C program can run with little or no modification on different kinds of computers (computers with different processors). This is of crucial importance if, for instance, one wants to change one's computer system. If programs are not portable, then much programming effort on the old system would have been wasted, and changing to a new system would be very costly. A software developer could sell many more programs if they could run on several machines with little or no modification. With the adoption of the new ANSI C standard, C programs have become even more portable.

Finally, and perhaps, most importantly, C is popular because, quite simply, it is a joy to use. And as one's mastery of the language increases, so does the joy.

This text assumes familiarity with basic programming concepts such as variables, constants, looping and iteration, but it covers all features of the C language. It is about the learning of programming in general as much as it is about mastering the art of coding programs in C. It is a truism that learning the syntax of a language is trivial compared with learning programming ideas and being aware of situations in which the syntactic constructs can be used. To this end, there is a wealth of examples and exercises that foster and test the understanding of the concepts developed in each chapter.

One of the main features is the illustration of the use of C constructs in meaningful examples as opposed to their use in contrived examples which serve no purpose other than to illustrate C syntax. In order to develop meaningful examples, certain side topics, such as sorting, hashing and binary trees, are developed. Developing these topics in the text makes the book more self-contained. The student learns a topic that is broadly applicable and so gets to see the C construct used in a wider context. In the conventional approach to teaching a language, features of the language are presented followed by examples illustrating these features. However, in this book, many features are introduced by discussing an example, showing the need for a feature and then presenting the feature. Hopefully, this approach gives one a broader picture of the application of a particular feature.

An important highlight of this book is the treatment of pointers – perhaps the hardest facet of the language but treated cursorily in most books on C. According to one reader of an early draft, 'This book gives a clear, logical and reasoned description of the subject which is quite refreshing to read'. Another topic which is usually glossed over in most books but explored in detail here is file handling. In particular, binary files and random access files are fully treated.

The exercises at the end of each chapter range from those which directly test the understanding of concepts, statements or constructs presented in the chapter to those which require the application of the material to non-trivial problems.

- Chapter 1 presents an overview of the basic features in C – data types, operators, expressions, statements and the basic control structures `while` and `if...else`. The treatment is not meant to be complete and many of the ideas introduced are expanded in later chapters.
- Chapter 2 introduces other commonly used control structures – `for`, `do...while`, `switch` and `continue` as well as a discussion of arrays. The latter includes the use of arrays for storing strings and simple methods for searching arrays.
- Chapter 3 discusses the elementary ideas involved in writing and using functions – the building blocks of C programs.
- Chapter 4 deals with the manipulation of character and string data.

The powerful ‘search and insert’ technique of hashing and insertion sort are introduced in order to discuss more useful examples.

- Chapter 5 is a more detailed treatment of functions. The flexible and powerful concept of pointers is also introduced here. The chapter ends with a discussion of that very useful (but one that students often find difficult) programming concept – recursion.
- Chapter 6 ties up the loose ends from the previous chapters. In particular, data types, operators, expressions, storage classes and initialization are discussed more fully.
- Chapter 7 starts with an introductory discussion of structures This is followed by some detailed examples illustrating the manipulation of linked lists.
- Chapter 8 continues the discussion of structures, using the versatile binary tree as the main theme. The latter part of the chapter deals with nested structures, unions and bit-fields.
- Chapter 9 covers standard input/output in C in a fair amount of detail. This is deliberate since this is perhaps the area of C that programmers use most often. A number of subtle issues are discussed which are hardly ever treated in most books or even the compiler manual.
- Chapter 10 is devoted to functions which operate on general files. The treatment of binary and random access files rounds off the chapter.
- Chapter 11 discusses the main facilities provided by the C preprocessor and ends with a brief treatment of some of the lesser used features in C.

Welcome to C programming. Though it can be frustrating and difficult at times, it can also be interesting, exciting, fascinating and highly rewarding.

Noel Kalicharan