# FURTHER COMPUTER PROGRAMMING IN BASIC

73·87221
B62:

# FURTHER COMPUTER PROGRAMMING IN BASIC

**Peter Bishop**

550 6444

Nelson

DR87/17

**Thomas Nelson and Sons Ltd**
Nelson House Mayfield Road
Walton-on-Thames Surrey KT12 5PL

P.O. Box 18123 Nairobi Kenya

116-DJTC Factory Building
Lorong 3 Geylang Square Singapore 1438

**Thomas Nelson (Hong Kong) Ltd**
Watson Estate Block A 13 Floor
Watson Road Causeway Bay Hong Kong

**Thomas Nelson (Nigeria) Ltd**
8 Ilupeju Bypass PMB 21303 Ikeja Lagos

First published by Thomas Nelson and Sons Ltd 1982

ISBN 0-17-431266-7

NCN 200-3427-0

Designed by The New Book Factory, London
Phototypesetting by Parkway Group, London and Abingdon
Printed and bound in Great Britain by
Butler & Tanner Ltd
Frome and London

[illegible]

# Preface

The aim of this book is to teach computer programming in a high level language, using Basic as an example. The emphasis is on the concepts, skills and techniques of programming, and on a disciplined approach to the task of designing and writing programs. The book is tailored to meet the requirements of all the examination boards in the United Kingdom which are offering Computer Studies or Computing Science at Advanced level. Questions from past examination papers in these courses are included, as is a list of suggestions for projects which are generally written as part of these courses.

This book has also been written with the requirements of the computing industry in mind, particularly with regard to programming standards and approaches to writing programs.

It is realised that a book of this nature is useful for a number of other purposes. These include a reference book for teachers preparing to teach programming at O or A-level, a textbook for students of equivalent courses at colleges of further education, and a foundation book for students embarking on computing courses at universities or polytechnics. It is also ideal for the owner of a personal microcomputer who wants something more than a superficial treatment of the subject.

The book is structured into five parts, each part comprising several chapters. The first part introduces the features of Basic language, emphasising at the same time the essential concepts of programming. 'Frills' of the language, which are limited to certain types of computer, are deliberately avoided. Flow diagrams are introduced to illustrate program structure, but are not regarded as an essential part of designing and writing programs. Program examples and exercises in this part are deliberately kept short and simple. The second part of the book is independent of any programming language, and concerns an approach to writing programs. It deals with the vital questions of program design and program structure, as well as checking, documenting and maintaining programs. Basic language programs are used to illustrate the techniques discussed.

The third and fourth parts relate programming to the wider theory of computing. Part three describes the fundamental programming operations of sorting, searching and merging, while part four deals with programming using such data structures as stacks, queues and lists.

The final part of the book deals with a wide variety of programming applications. By and large the applications are self-contained, and include graphics, simulation and syntax analysis. Although the majority of this book has a low mathematical content, two chapters in this part are an exception to this rule. They enable those with some knowledge of numerical analysis to combine it with their skill in programming, but may be omitted without loss of continuity.

The book concludes with a list of project suggestions, a revision exercise, a Basic summary and a glossary of programming terms.

## Acknowledgements

A number of people have helped me in the planning and writing of this book. At Imperial College, Valerie Downes reviewed the plans for the book and a number of specimen chapters. John Darlington gave advice on the topic of program design, and Jill Rout typed the specimen chapters.

I am also most grateful to Patrick Sutton who reviewed the book, and to Ruth Bush who typed the entire manuscript.

Thanks are also due to the Associated Examining Board for the General Certificate of Education, the Joint Matriculation Board, the Oxford Delegacy of Local Examinations, and the University of London School Examinations Department for their kind permission to reproduce questions from past examination papers.

# Contents overview

# Contents

## An Approach to Writing Programs

# 1 Introduction

The aim of this book is to provide a broad, thorough grounding in high level language computer programming, using Basic language as a medium. The emphasis is on concepts, skills and techniques of programming, rather than on details of the language.

Although this book is intended for use in advanced computing courses, it assumes no previous knowledge of programming or Basic language, and, apart from two specialised chapters towards the end of the book, requires no more than a common sense level of mathematics. For those with some experience in programming, it presents a fresh look at the subject, in considerably more depth than before.

The book aims to assist in the development of a number of skills, notably:

- a systematic approach to a programming task;
- an ability to analyse a task in terms of programming concepts, techniques and available computing facilities;
- an ability to design and write programs of a high standard;
- a knowledge of the concepts of program structure, and how to apply these concepts;
- a familiarity, through programming, with a number of fundamental concepts of computing, such as data structures and operations like sorting and merging;
- an ability to understand programs other people have written;
- an ability to carry out systematic tests on a program, and to identify and correct errors;
- an ability to write adequate descriptions of the workings of programs.

Several more general skills are also involved, particularly the ability to think clearly, and to write clear, concise English.

This may seem an ambitious and somewhat daunting set of objectives, but the advantages of possessing such skills are considerable, and entirely worth the effort involved in developing and maintaining them.

Programming reaches across the boundary between art and science. It is creative, and yet restricted by a number of rules, standards and conventions. Programming can perhaps best be summed up in the word **discipline.** Learning to program is acquiring a mental discipline.

## 1.1 The Nature of Computer Programs

Before starting to indroduce some techniques of computer programming, in a particular programming language, it is necessary to take a step back from the subject, and examine computer programs in perspective. This exercise is as important to those with some experience of programming as it is to new-comers.

The remainder of this chapter aims to clarify ideas about the nature of computer programs. It establishes the environment in which a computer program operates. This material relates to the rest of the book in a number of ways. Several of the general ideas introduced here are expressed again, in a more specific form, later in the book. This chapter should give a sense of purpose and direction to much of the rest of the book. A chapter on program design, in the middle of the book, follows fairly closely from the ideas introduced here.

## 1.2 Views of Computer Programs

The next few sections examine three different views of computer programs, namely the elementary view, the 'toolsetting' view and the 'layer' view. These different views should give some insight into the rather elusive question – 'What is a computer program?'.

## 1.3 The Elementary View

The elementary view of a computer program is that it is a set of instructions to a computer. This is rather like calling a house a collection of bricks. It is quite correct, but inadequate for many purposes. Just as a house is far more than a collection of bricks, so is a computer program far more than a set of instructions to a computer.

A rather misleading extension of the view is to regard a computer program as a solution to a particular problem. This view originates in the use of computers to solve mathematical problems. Such programs are 'one-off jobs'. Once they have solved the problem, they are discarded. It must be emphasised that in today's world of computing, such programs are extremely rare.

## 1.4 The Toolsetting View

The 'toolsetting' view of programs derives from a particular view of computers. A computer is a general-purpose information processing machine. It may be regarded as a tool, capable of performing a wide variety of operations. A computer program 'sets up' the computer to perform certain precisely specified operations. In other words, a program creates a particular kind of tool, which is then used in certain ways. There are several important consequences of this view of computer programs. These are discussed below.

A tool is not much use if it breaks down frequently. Consequently, a computer program must be designed for repeated or continuous running. It must be robust, able to withstand accidental or deliberate misuse.

The tool created by a computer program must be compatible with its working environment. In other words, programs must be user-oriented. They must be as easy as possible to use, and co-ordinate with other work the user may be doing. In practice, most tasks performed by computers form part of larger operations. Computer programs must be written with the requirements of the larger operation in mind.

These requirements add up to the fact that computer programs must be well-designed, having a sound structure. The question of program design is discussed in some detail in Chapter 11, but must be kept in mind from the start.

## 1.5 The Layer View

Another view of computer programs is to regard them as a 'layer' between the hardware of a computer and the outside world. In this context, computer programs are often called **software**. A useful computer consists of hardware 'surrounded' by a layer of software. See Figure 1.1.

An extension of this view is that some programs set up a computer as a tool to run other programs. In consequence, the computer requires several layers of software. The innermost layer interacts with the hardware of the computer,

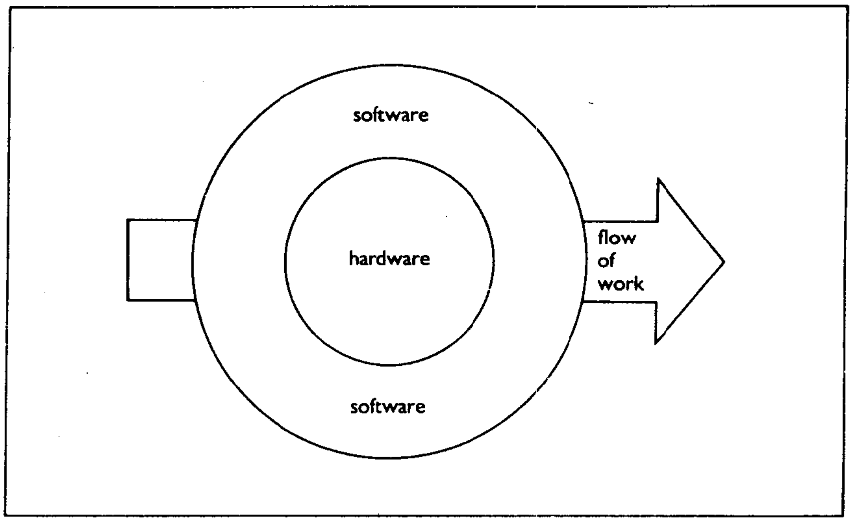


Figure 1.1
Hardware and software

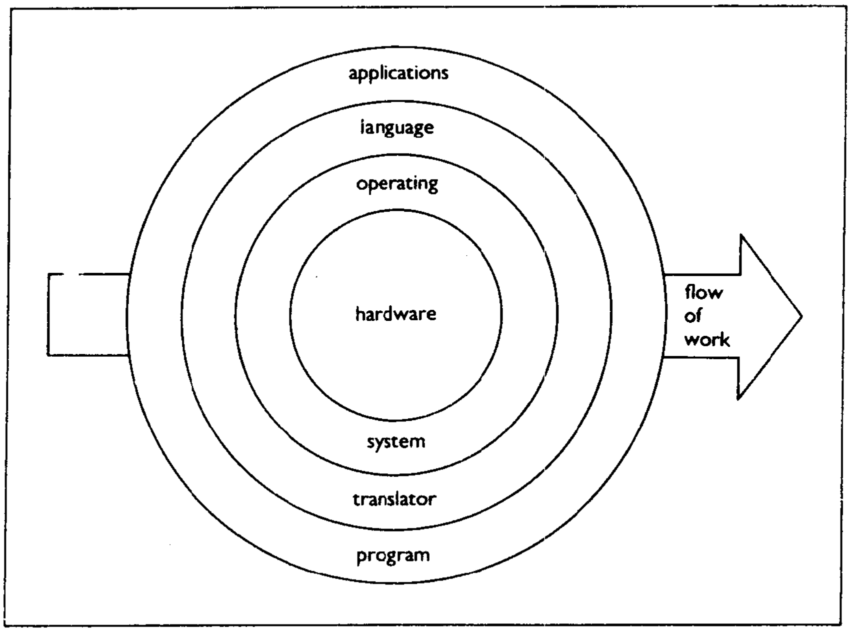
while the outer layer interacts with the user. Each layer of software is said to **support** the layers outside it.

A very common pattern of layers of software is the following:

1 the innermost layer is the **operating system**, which interacts directly with the hardware, and manages the resources of the computer;
2 next there is a layer of **language translation programs**, which translate from a language such as Basic to the machine language of the computer hardware;
3 the outermost layer is made up of **applications programs**, which set up the computer to perform certain tasks.

This pattern is illustrated in Figure 1.2. Programs are discussed in this book from all three of the above layers.

Figure 1.2
Layers of software


applications
language
operating
hardware
system
translator
program
flow
of
work