

FURTHER COMPUTER PROGRAMMING IN BASIC

73.87221

13622

FURTHER COMPUTER PROGRAMMING IN BASIC

Peter Bishop

550 6444

Nelson

5506444

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

© Peter Bishop 1982

First published by Thomas Nelson and Sons Ltd 1982

ISBN 0-17-431266-7

NCN 200-3427-0

All rights reserved. No part of this
publication may be reproduced, stored
in a retrieval system, or transmitted,
in any form or by any means, electronic,
mechanical, photocopying, recording or
otherwise, without the prior permission
of the publishers.

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

244002

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

- 1 Introduction
- 2 Basic Language

Features of Basic Language

- 3 Input, Processing, Output
- 4 Branches
- 5 Loops
- 6 Character Handling
- 7 Arrays
- 8 Functions
- 9 Subprograms
- 10 File Handling

An Approach to Writing Programs

- 11 Program Design
- 12 Program Checking
- 13 Program Documentation
- 14 Program Maintenance

Fundamental Programming Operations

- 15 Sorting
- 16 Searching
- 17 Merging

Programming Fundamental Data Structures

- 18 Stacks
- 19 Queues
- 20 Lists
- 21 Trees

Programming Applications

- 22 Commercial Data Processing
- 23 Interactive Programs
- 24 Graphics
- 25 Critical Path Analysis
- 26 Numerical Analysis
- 27 Simulation
- 28 Syntax Analysis
- 29 Simulating a Model Computer

Projects

- 30 Project Suggestions

Revision Exercises
Basic Summary
Glossary of Terms
Teachers' Notes
Index

Contents

I Introduction	1
1.1 The Nature of Computer Programs	1
1.2 Views of Computer Programs	2
1.3 The Elementary View	2
1.4 The Toolsetting View	2
1.5 The Layer View	2
1.6 Conclusion	4
Exercise 1	4
2 Basic Language	5
2.1 The Development of Basic Language	5
2.2 The Objectives of Basic Language	5
2.3 The Problem: Many Versions of Basic	5
2.4 The Solution: One Reference Language and Many Implementation Languages	6
2.5 The Reference Language	6
2.6 Implementation Languages	6
2.7 Essential Features of Basic Language	6
2.8 Program Structure	7
2.9 Line Structure	7
2.10 Data	7
2.11 Constants	7
2.12 Variables	8
2.13 Remarks	8
2.14 Implementation-dependent Features	8
2.15 Conclusion	9
Exercise 2	9
Features of Basic Language	
3 Input, Processing, Output	10
3.1 Input	10
3.2 Keyboard Input	10
3.3 Input from Data Loaded with the Program	10
3.4 The RESTORE Statement	11
3.5 Output	11
3.6 Output to a Printer or Display Screen	12
3.7 Example Program 3.1	13
3.8 Tabulating Output	13
3.9 Example Program 3.2	14
3.10 Processing	16
3.11 Arithmetic Expressions	16
3.12 Example Program 3.3	17
□ 3.13 Sequences of Statements on a Single Line	18
3.14 PEEK and POKE	18
3.15 Conclusion	19
Exercise 3	19

4	Branches	22
4.1	Simple Conditional Operations	22
4.2	Example Program 4.1	24
4.3	Compound Conditions	24
4.4	Example Program 4.2	26
4.5	Multi-way Branching	29
4.6	Example Program 4.3	30
□ 4.7	Conditional Statements in Advanced Basic	32
□ 4.8	Example Program 4.4	32
4.9	Conclusion	33
	Exercise 4	33
5	Loops	36
5.1	The Concept of Repetition	36
5.2	Repetition for a Specified Number of Times	36
5.3	Example Program 5.1	37
5.4	The General Form of the FOR . . . TO Statement	39
5.5	Nested Loops	39
5.6	Example Program 5.2	40
5.7	Branching Into and Out Of Loops	43
5.8	Example Program 5.3	43
5.9	Repetition While a Certain Condition is True	46
5.10	Example Program 5.4	47
5.11	Repetition Until a Certain Condition Becomes True	48
5.12	Example Program 5.5	49
5.13	Conclusion	51
	Exercise 5	52
6	Character Handling	54
6.1	String Handling Facilities in Basic	54
6.2	Functions and Arguments	54
6.3	The Joining Operator +	55
6.4	ASC(X\$)	55
6.5	CHR\$(X)	55
6.6	LEN(X\$)	56
6.7	MID\$(X\$, A, B)	56
6.8	STR\$(X)	56
6.9	VAL(X\$)	56
6.10	Example Programs	57
6.11	Example Program 6.1	57
6.12	Example Program 6.2	59
6.13	Conclusion	62
	Exercise 6	63
7	Arrays	64
7.1	The Nature of an Array	64
7.2	Some Uses for Arrays	64
7.3	Example Program 7.1	65
7.4	Two-dimensional Arrays	69
7.5	Example Program 7.2	70
7.6	Conclusion	78
	Exercise 7	78

8	Functions	80
8.1	The Nature of a Function	80
8.2	Standard Basic Functions	81
8.3	ABS(X)	81
8.4	EXP(X)	81
8.5	LOG(X)	82
8.6	INT(X)	82
8.7	Example Program 8.1	83
8.8	RND(X)	84
8.9	Example Program 8.2	85
8.10	SGN(X)	86
8.11	SQR(X)	86
8.12	SIN(X)	86
8.13	COS(X)	87
8.14	TAN(X)	87
8.15	ATN(X)	87
8.16	TAB(X)	87
8.17	Example Program 8.3	87
8.18	User-defined Functions	88
8.19	Defining a Function	90
8.20	Using a Defined Function	90
8.21	Example Program 8.4	90
8.22	Additional Features of User-defined Functions	94
8.23	Conclusion	94
	Exercise 8	94
9	Subprograms	96
9.1	The Nature of a Subprogram	96
9.2	Subprograms in Basic	97
9.3	Example Program 9.1	97
9.4	Passing Data to and from Subprograms	104
9.5	Example Program 9.2	105
9.6	Conclusion	108
	Exercise 9	110
10	File Handling	112
10.1	The Nature and Uses of Files	112
10.2	File Handling in Basic	112
10.3	Creating a File	113
10.4	Writing to a File	113
10.5	Closing a File	113
10.6	Example Program 10.1	114
10.7	Opening a File for Reading	116
10.8	Reading from a File	116
10.9	Renaming a File	116
10.10	Erasing a File	117
10.11	Example Program 10.2	117
10.12	File Handling Statements in Other Versions of Basic	119
10.13	Conclusion	120
	Exercise 10	120

An Approach to Writing Programs

11	Program Design	122
11.1	What is Program Design?	122
11.2	The Objectives of a Well-designed Program	122
11.3	Less Desirable Objectives	123
11.4	Some Constraints	124
11.5	Design Techniques	124
11.6	Stepwise Refinement	124
11.7	Flow Diagrams	124
11.8	Worked Examples	124
11.9	Example Program 11.1	125
11.10	Example Program 11.2	128
11.11	Conclusion	142
	Exercise 11	143
12	Program Checking	145
12.1	The Objectives of Program Checking	145
12.2	Some Checking Techniques	145
12.3	Dry Runs	146
12.4	Example 12.1	146
12.5	Test Data	146
12.6	Example 12.2	147
12.7	Determining Under What Conditions a Program Will Work	148
12.8	Example 12.3	148
12.9	Conclusion	149
	Exercise 12	150
13	Program Documentation	151
13.1	Types of Documentation	151
13.2	Programmer's Documentation	151
13.3	Objectives of Programmer's Documentation	151
13.4	Techniques for Writing Programmer's Documentation	151
13.5	Variables and Flow Diagrams	152
13.6	Example 13.1	152
13.7	User Documentation	153
13.8	Objectives of User Documentation	153
13.9	Techniques for Writing User Documentation	154
13.10	Example 13.2	154
13.11	Conclusion	155
	Exercise 13	156
14	Program Maintenance	157
14.1	Types of Program Maintenance	157
14.2	Some General Guidelines	157
14.3	Changes Within Program Modules	158
14.4	Example 14.1	158
14.5	Replacement of Modules	161
14.6	Example 14.2	161
14.7	Changes to the Structure of a Program	162
14.8	Example 14.3	162

x	14.9 Program Maintenance and Program Design	166
Contents	14.10 Conclusion	167
	Exercise 14	167

Fundamental Programming Operations

15	Sorting	169
15.1	The Concept of Sorting	169
15.2	Sorting Techniques	169
15.3	The Bubble Sort	169
15.4	Example Program 15.1	170
15.5	Conclusion	175
	Exercise 15	175
16	Searching	178
16.1	The Concept of Searching	178
16.2	Searching Techniques	178
16.3	Sequential Search	178
16.4	Binary Search	179
16.5	Example Program 16.1	179
16.6	Conclusion	183
	Exercise 16	183
17	Merging	184
17.1	The Concept of Merging	184
17.2	A Merging Technique	184
17.3	Example Program 17.1	185
17.4	Merging and Sorting	188
17.5	Conclusion	188
	Exercise 17	188

Programming Fundamental Data Structures

18	Stacks	190
18.1	Properties of a Stack	190
18.2	Representing a Stack in Basic	190
18.3	Example Program 18.1	190
18.4	Some Applications of Stacks	195
18.5	Conclusion	195
	Exercise 18	196
19	Queues	198
19.1	Properties of a Queue	198
19.2	Representing a Queue in Basic	198
19.3	Example Program 19.1	198
19.4	Some Applications of Queues	204
19.5	Conclusion	205
	Exercise 19	205
20	Lists	207
20.1	Properties of a List	207
20.2	Representing a List in Basic	207

20.3	Example Program 20.1	208
20.4	Some Applications of Lists	220
20.5	Conclusion	221
	Exercise 20	221

21	Trees	224
21.1	Properties of a Tree	224
21.2	Representing a Tree in Basic	224
21.3	Example Program 21.1	224
21.4	Some Applications of Trees	232
21.5	Conclusion	232
	Exercise 21	233

Programming Applications

22	Commercial Data Processing	235
22.1	The Nature and Scope of Commercial Data Processing	235
22.2	Accounting	236
22.3	Example Program 22.1	237
22.4	Conclusion	251
	Exercise 22	251

23	Interactive Programs	254
23.1	Characteristics of Interactive Programs	254
23.2	Basic - an Interactive Language	254
23.3	Example Program 23.1	254
23.4	Conclusion	264
	Exercise 23	264

24	Graphics	266
24.1	Machine Dependence of Graphics Facilities	266
24.2	The Cathode Ray Canvas	266
24.3	The Basic Brushstrokes	268
24.4	Clear Screen	268
24.5	Write Character to Screen	269
24.6	Read Character from Screen	270
24.7	Example Program 24.1	270
24.8	Example Program 24.2	276
24.9	Interactive Graphics	281
24.10	Conclusion	282
	Exercise 24	282

25	Critical Path Analysis	284
25.1	The Context of Critical Path Analysis	284
25.2	Activities and Events	284
25.3	Networks	285
25.4	Early and Late Event Times	285
25.5	The Critical Path	285
25.6	Calculating Early and Late Event Times	286
25.7	Example Program 25.1	286
25.8	Conclusion	292
	Exercise 25	293

26	Numerical Analysis	295
26.1	Techniques of Numerical Analysis	295
26.2	Approximate Solutions to Equations: The Transposition Method	295
26.3	Example Program 26.1	296
26.4	Gauss's Method for Solving Simultaneous Equations	302
26.5	Example Program 26.2	303
26.6	Numerical Integration	307
26.7	Example Program 26.3	309
26.8	Using Numerical Analysis	313
26.9	Conclusion	314
	Exercise 26	314
27	Simulation	317
27.1	Simulation – Making a Model	317
27.2	The Modelling Process	317
27.3	Desirable Features of a Model	318
27.4	A Population Model	318
27.5	Example Program 27.1	320
27.6	Conclusion	322
	Exercise 27	323
28	Syntax Analysis	326
28.1	Approaches to Syntax Analysis	326
28.2	Areas of Syntax Analysis	326
28.3	Number Recognition – a State Table	327
28.4	Example Program 28.1	328
28.5	Syntax of Arithmetic Expressions - a Precedence Table	335
28.6	Example Program 28.2	337
28.7	Methods of Top-down Syntax Analysis	347
28.8	Conclusion	348
	Exercise 28	349
29	Simulating a Model Computer	351
29.1	A Simple Model Computer – the SMC	351
29.2	SMC Register Layout	351
29.3	SMC Machine Language	353
29.4	Example SMC Machine Language Program	354
29.5	SMC Instruction Cycle	354
29.6	Example Program 29.1	354
29.7	Assembly Language	369
29.8	Example SMC Assembly Language Program	370
29.9	An Assembler	370
29.10	Conclusion	371
	Exercise 29	371
30	Project Suggestions	373
30.1	Some General Guidelines	373
30.2	Project 1: A Statistics Package	374
30.3	Project 2: File Processing	375
30.4	Project 3: A Data Enquiry System	376
30.5	Project 4: Heat Flow Systems	376

xiii			
Contents	30.6	Project 5: Projectiles	378
	30.7	Project 6: A Railway Junction Signalling System	379
	30.8	Project 7: Can't Stop the Music	380
	30.9	Project 8: Economic Models	382
	30.10	Project 9: Genetics	383
	30.11	Project 10: An Ecosystem	384
	30.12	Sources of Further Information	385
		Revision Exercise	387
		Basic Summary	390
		Glossary of Terms	392
		Teachers' notes	397
		Index	424

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 introduce 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 newcomers.

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.

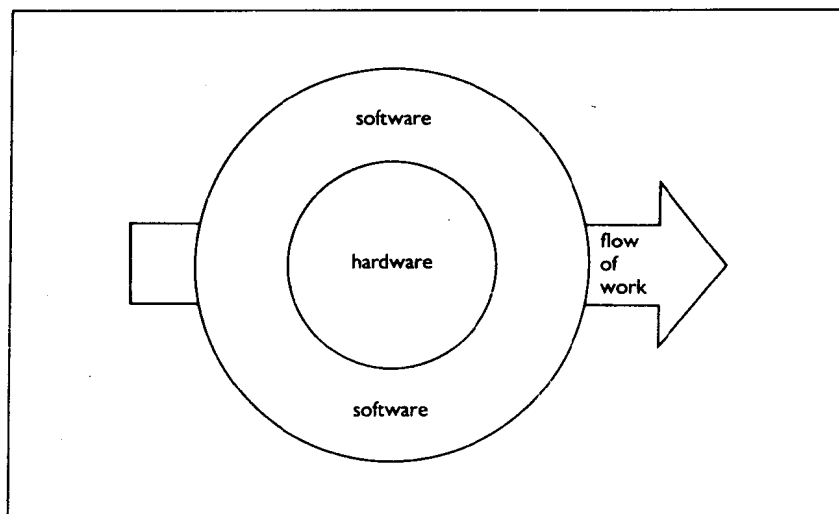


Figure 1.1
Hardware and software

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

