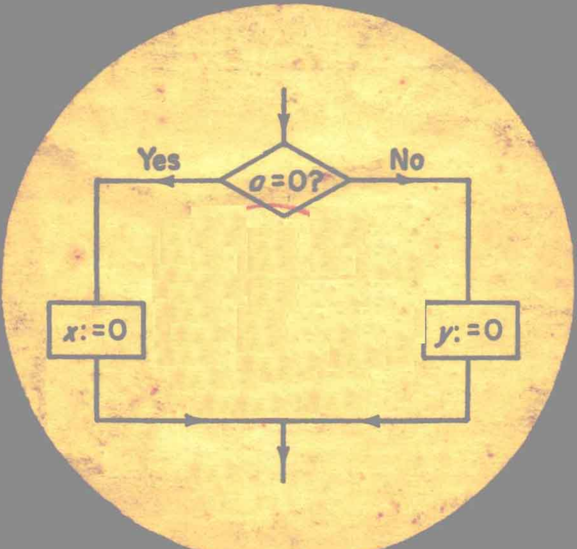

J. M. Rushforth

J. Ll. Morris

Computers and Computing

J. M. RUSHFORTH and J. LL. MORRIS

The University of Dundee Scotland

JOHN WILEY & SONS LONDON NEW YORK SYDNEY TORONTO

Library of Congress Catalog Card No. 72-8616

ISBN 0 471 74540 5

Printed in Great Britain by J. W. Arrowsmith Ltd, Bristol

Introductory Mathematics for Scientists and Engineers

Foreword to the Series

The past few years have seen a steady increase in the courses of mathematics and computing provided for students undertaking higher education. This is partly due to high speed digital computation being more readily available and partly because so many disciplines now find mathematics an essential element of their curriculum. Many of the students, e.g. those of physics, chemistry, engineering, biology and economics, will be concerned with mathematics and computing mainly as tools, but tools with which they must acquire proficiency. On the other hand, courses for mathematicians must take cognizance of the existence of electronic computers. All these students may therefore study similar material though possibly at different stages of their careers—some, perhaps, will encounter it shortly after commencing their training while others may not come to grips with it until after graduating. This series is designed to cater for these differing requirements; some of the books are appropriate to the basic mathematical training of students in many disciplines while others, dealing with more specialized topics, are intended both for those for whom such topics are an essential ingredient of their course and for those who, although not specialists, find the need for a working knowledge of these areas. However, the presentation of all the books has been planned so as to demand the minimal mathematical equipment for the topics discussed. Instructors will therefore often be able to extract a shorter introductory course when a fuller treatment is not desired.

The authors have, in general, avoided the strict axiomatic approach which is favoured by some writers, but there has been no dilution of the standard of mathematical argument. Learning to follow and construct a logical sequence of ideas is one of the important attributes of courses in mathematics and computing.

While the authors' purpose has been to stress mathematical ideas which are central to applications and necessary for subsequent investigations, they have attempted, when appropriate, to convey some notion of the connection between a mathematical model and the real world. They have also taken account of the fact that most students now have access to electronic digital computers.

The careful explanation of difficult points and the provision of large numbers of worked examples and exercises should ensure the popularity of the books in this series with students and teachers alike.

D. S. JONES
Department of Mathematics
University of Dundee

Preface

This book is written for first-year university students who require an introduction to both computer programming and to the modern techniques of numerical analysis. The two topics are developed side by side so that the student can apply programming techniques to the solution of problems occurring in the numerical analysis.

The chapters on numerical analysis are not intended primarily as a rigorous treatment of the subject, but more as a working text in which the student can find an outline of the theory behind the various applications discussed and also the practical problems associated with their evaluation.

The programming language described is ALGOL 60, and the reader is introduced to all the main features of it. One or two features which are only infrequently used (e.g. switches and own variables) have been omitted, but a student who has a grasp of the language obtained from the chapters on programming would be well equipped to tackle any problem he is likely to encounter.

In order to get a grasp of both the numerical analysis and programming it is necessary to do a good number of the exercises. These exercises are an integral part of the text. Those in the programming chapters are specifically aimed at giving practice in the various techniques described in the preceding sections. They do not require a great deal of mathematical knowledge and it is not necessary to have made a detailed study of the numerical analysis sections. The exercises in the numerical analysis chapters can be successfully attempted by a reader who has understood the programming techniques so far described.

A student wishing to know more of the mathematical background to numerical analysis is referred to the following books which are published in the same series as this:

Introductory Analysis: Vols. I and II by D. S. Jones and D. W. Jordan
Computational Methods in Ordinary Differential Equations by J. D. Lambert

Computational Methods in Partial Differential Equations by A. R. Mitchell

and to the forthcoming book:

Computational Methods in Elementary Numerical Analysis by J. Ll. Morris.

The authors would like to express their thanks to Professor D. S. Jones, Ivory Professor of Mathematics, and Mr. E. J. Gillespie of the University Computing Laboratory of the University of Dundee for reading various parts of the manuscript and offering many helpful suggestions.

Contents

1

An introduction to computers

1.1 Introduction

Before discussing computers let us look at a common domestic occurrence—that of ordering groceries from a shop.

The first thing we do is to make up a list of what we need; we then take this to the shop, or else send the order over the telephone. In either case, the grocer receives our instructions and uses them for making up the order. He reads the list and as he comes to each item he takes the goods off the shelf and adds them to our order. When it is complete he hands it over, or else puts it aside for delivery later.

The instructions we give the grocer may be a straightforward list of requirements such as:

3 lb flour
2 lb butter
2 lb sugar
1 lb icing sugar

and so on, but we may also say such things as:

1 lb sultanas, but if you haven't any sultanas make it a pound of currants instead,

or even:

if you haven't any cherries leave out the rest of the order.

What we have done is to give the grocer a list of instructions which he is to carry out and provided he can supply the goods mentioned, we will get our order properly made up. If he is out of stock of some item, he will write a note which says 'Sorry, no tinned pears'—so that we are advised of any shortages.

1.2 The program

When we use a computer to perform calculations for us, we have to do very much the same sort of thing as making up our grocery list.

We have to prepare a list of instructions for the computer which will carry out our calculations. These instructions are known as the *program* and are written in a language which can be interpreted by the computer. This language will be much more concise than English, and will be tailored to the needs of our problem.

1.3 Input

When the program is written we have to get the instructions into the computer and this is similar to the situation of giving the order to the grocer. This process in computing is known as *input* and we have to have special devices to do this. In our parallel case the input device was either a telephone or the action of passing a list over the counter.

1.4 The store

The computer has a store which can retain information for as long as is required, and the input device will read the program into this store. When the whole program has been stored, the program is 'run' by starting at the first instruction and letting the computer carry out each one in turn. The 'control' part of the computer which interprets the instructions and carries them out is equivalent to the grocer himself who reads the grocery list and takes each item from his shelves.

Our analogy can be pursued a little further because in computing we often require to store numbers in our computer, and obviously we must know where we are putting them, sc that they can be retrieved later. In just the same way, the grocer knows on which shelf his particular goods are kept, so that when he gets an order for a tin of pears he knows where to find it.

We can regard the store of the computer as a set of pigeon holes, each of which can contain one piece of information. We could number each pigeon hole and call this number the *address* of the *location*. In some basic programming languages this is done, but it is more usual to attach a name to each piece of information and allow the computer to choose the address in which to keep it. It is not usually of interest to us where the computer keeps the information, just as it does not matter to us where the grocer keeps his tins of pears—all we want to be able to do is to refer to our particular item by name and let the grocer (or the computer) look after the internal organization of the store.

1.5 Alternative paths in a program

In our example we give an instruction which contained alternative paths of action. 'If sultanas are available, then supply 1 lb sultanas otherwise supply 1 lb currants.' This type of instruction is available in many programming languages. Some condition is examined and if this condition is found to be true then one course of action follows, if the condition is not true (i.e. is false), then another course is taken. In our example, when either sultanas or currants have been supplied the two alternative paths of action merge together when the next instruction is obeyed.

Our last example shows that we can leave the sequence of instructions and rejoin it elsewhere. The statement 'if you haven't any cherries leave out the rest of the order' means 'if no cherries then jump to end of order'. Once again, in a computer program we can jump from one part of the program to another, and these jumps can be either forward to instructions not yet obeyed, or backwards to repeat some instructions which may already have been given.

1.6 Output

When the grocer has completed his instructions, he either hands over the goods he has collected, or else he arranges to deliver them later. In either case we could regard this final action as the 'result' of the order.

In order to get results from a computer program we often produce answers in the form of a printed text. The devices required to produce answers in this and other forms are known as *output* devices.

1.7 Secondary storage

Besides making up orders for customers, the grocer has to concern himself with keeping his shelves full, and he will replenish these by bringing in more goods from his store-room, or by getting further deliveries from his wholesalers. In both these cases he is going to larger secondary stores which contain more goods than he can have on his shelves.

It is uneconomical for the grocer to keep his stock on a large area of shelving immediately accessible to him. What he requires is sufficient of this shelving to allow him to run his day-to-day business efficiently, backed up by bulk stock in store-rooms which are cheaper to maintain.

He will try to keep his shelves properly stocked so that he doesn't need to go to the store-room whilst making up an order, so that time is not lost during this process. The shelves can be filled in a more leisurely manner at his convenience. In fact, he can tell his assistant what he requires from the store-room and let the assistant do the replenishing whilst he is serving more customers.

This idea of secondary storage is used in computers, because the main or *immediate-access store* is very expensive so that only a limited amount is available. If a great deal of information is to be stored, for instance the accounts of all the customers of a bank, or the information required to do the payroll calculations, then this information must be kept in secondary storage. In a computer this secondary storage consists of different devices which store information magnetically. We will describe these in more detail later.

Another reason for using secondary storage devices is that we do not normally leave programs in the main store of the computer once they are finished. The computer has to be used for other purposes. Its use is much more diverse than the examples we gave when we used our analogy of the grocer's shop. In that instance we did all our operations on one set of data—the contents of the shelves—but suppose the grocer sells his business and an ironmonger takes it over, then we use the same shop for different kinds of jobs. In turn the ironmonger may sell to a butcher, the butcher to an insurance company, the insurance company to a bank, the bank to a baker and so on. If we can imagine changes like these taking place many times a day it gives us a better idea of how many different jobs a computer may be called upon to do.

1.8 The computer configuration

The computer consists of a *central processor*, *immediate-access store* and *peripheral devices*. The whole is referred to as the computer configuration and this configuration can be varied by choice of processor, size of store and by the different peripheral devices which can be attached to the processor.

1.9 The central processor

The central processor is that part of the computer which can examine the instructions of a program and cause them to be carried out. It communicates with the immediate-access store and all the peripheral devices. It also does all the arithmetical operations required.

The central processor consists of many complex electronic circuits which direct information to and collect information from other parts of the computer system. It also contains *registers* or *accumulators* in which arithmetic operations are carried out and in which information regarding the state of the program is retained. The electronics of a computer are frequently referred to as its *hardware*, as opposed to *software* which consists of programs written to help the user to make full use of the computer's capabilities. Most users of a computer need to know little about computer hardware and can regard the various parts of the computer as 'black boxes' which do what is required of them.

1.10 The immediate-access store

The immediate-access store is the part of the computer which stores all the information currently being used. It also stores the program instructions which in turn go to the processor. The store keeps its information in units called *words* and these words are recorded magnetically. A computer's main store may have anything from 8,000 to over a quarter of a million words in it, and any word can be retrieved by the central processor within millionths of a second. A millionth of a second is called a *microsecond*, and an *access time* of one or two microseconds is quite common.

The access time of some stores is now measured in nanoseconds (1 nanosecond = 1 thousandth of a microsecond) and 750 nanoseconds is frequently quoted as access time.

With magnetic recording, both in the main store and in secondary storage devices, we can draw a comparison with the ordinary tape recorder. When we record, or *write*, on to our tape recorder we store our voice by magnetizing an oxide coating on the tape. Once the recording has been made it can be played back, or *read*, as many times as we like. If we *overwrite* the message we have recorded, by recording something else on the same piece of tape, then the original is lost for ever.

In the same way we can write information into locations in our computer stores and recall and use it as many times as we like subsequently, provided we do not overwrite it with some other pieces of information.

1.11 Data preparation equipment

Before discussing the peripheral devices which can be attached to a computer it is advisable to discuss how we convert the program we have written on a piece of paper into a form which the computer can read.

One way is to translate our text into a pattern of holes on a continuous piece of paper tape. We can use a *teleprinter* consisting of a typewriter keyboard and a paper-tape punch. Each time a key is depressed a pattern of holes, called a character, is punched across the paper tape, and the tape is moved on so that when another key is pressed another character is punched behind the first.