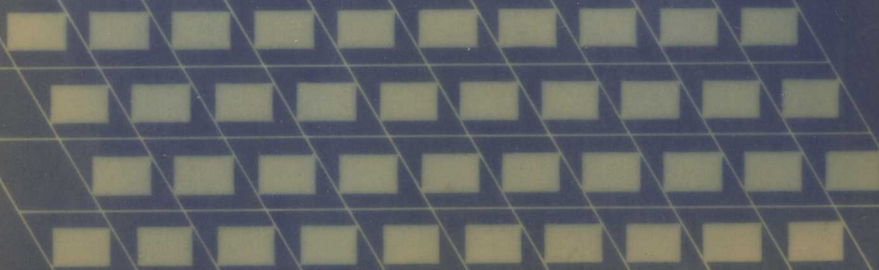


PASCAL *programming: a beginner's guide to computers and programming*

CHRIS HAWKSLEY



TP31
H15

8462891

Pascal programming

*A BEGINNER'S GUIDE TO COMPUTERS AND
PROGRAMMING*



E8462891

CHRIS HAWKSLEY

Department of Computer Science, University of Keele



CAMBRIDGE UNIVERSITY PRESS

Cambridge

London New York New Rochelle

Melbourne Sydney

1085784

Published by the Press Syndicate of the University of Cambridge
The Pitt Building, Trumpington Street, Cambridge CB2 1RP
32 East 57th Street, New York, NY 10022, USA
296 Beaconsfield Parade, Middle Park, Melbourne 3206, Australia

© Cambridge University Press 1983

First published 1983
Reprinted 1983

Printed in Great Britain by the University Press, Cambridge

Library of Congress catalogue card number: 82-19760

British Library Cataloguing in Publication Data

Hawksley, C.

Pascal programming.

1. PASCAL (Computer program language)

I. Title

001.64'24 QA76.73.P2

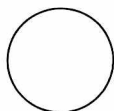
ISBN 0 521 25302 0 hard covers

ISBN 0 521 27292 0 paperback

TP31

**Pascal programming: a beginner's guide
to computers and programming**

To my parents



Preface

I was surprised to find that although there are shelves of programming texts on the market, many of which use the excellent language Pascal, none of these books was proving to be a great success with the hundreds of students taking subsidiary level computer programming courses I have taught over the last eight years. I began to ponder why this should be. Was it the standard of the text books themselves? Certainly not; several are of first-class quality with authors of recognised programming and teaching ability. Perhaps the students were not up to scratch? An easy get-out, this, but not good enough since their eventual results were on the whole normal and satisfactory. Could it be that my teaching style or ability was not good enough to support and encourage reading of a back-up text? Well, possibly, though my students tend to be a vocal lot yet they hurl no more than a fair share of verbal abuse in my direction. On the other hand, I have never found it easy to follow any of the text books closely in these courses and this gives a clue to part of the problem.

I believe that the level of background knowledge assumed in most texts is unrealistically high for a lot of students. There are two related problems. Firstly, the bias tends to be towards the numerate scientist both in the general approach and, too often, in the choice of examples. Also, the starting point of many texts is too advanced for many newcomers to computing, relying on additional course or book material to introduce some of the fundamentals of computing.

I have tried to write this book for students who are learning computer programming, probably for the first time, and probably as a subsidiary subject. It is possible that their main subject or subjects may lie in disciplines apparently far removed from computing though this is not necessarily the case. I have not assumed previous exposure to computers

nor any parallel courses in computer science. The examples require only 'common-sense' mathematical ability and have been chosen from a wide range of disciplines. In addition, I hope the book will be of value to the individual reader learning to program for the first time and as a 'starter kit' for new computer science students, who should progress fairly rapidly onto the more advanced texts covering algorithms, data and programs.

I have not tried to cover absolutely all of Pascal's features in detail in order to concentrate on using the more common ones and to keep the book short. New programming facilities are introduced usually by typical examples, with passing reference to the theoretical alternatives, though I have given diagrams covering all of Pascal in an appendix. Most of the language is covered. Left out are bits of Pascal that will appeal to the programmer growing in experience who will eventually wish to peruse more advanced texts on programming and problem solving in any case. These are dealt with in chapter 14.

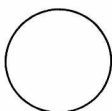
The text starts with a fairly gentle introduction to computers and programming leading into the basic foundations of programming in the Pascal language. The emphasis in part 2 is on practical applications of computer programming and I have tried to select examples from disciplines that may be familiar to the non-specialist student of computing: some text processing, social science applications, and analysis of collected data, for example.

It should be a source of encouragement to many readers to know that in my experience there is little, if any, correlation between the ability to become a competent programmer and the academic background of a student. A number of first class mathematics students have passed through the Computer Science Subsidiary Course at Keele. A proportional number of students from the departments of Physics, Chemistry, Biology, Geology, Psychology, Economics, Education, Geography, Sociology, Social Policy, Music, English, French, German, Russian, Latin, History, American Studies, Philosophy, Law, and Politics (my apologies for any omissions) have succeeded equally well. Yet, every year I hear 'Do you think I can make it? I am not very good at maths., you know.' Quite honestly, a positive attitude is a far more important prerequisite than an 'A' level in mathematics. Enjoy your programming.

I am indebted to many people for help and ideas in writing this book, amongst whom I must include the generations of students who have helped me to appreciate some of the common difficulties experienced by beginners. In particular, I would like to thank Professor Colin Reeves for his encouragement to write the book in the first place, Dr Neil White whose

meticulous knowledge of Pascal was invaluable and Lorraine Jarvis for grappling with some of my handwriting and drawings. Any mistakes or shortcomings that remain are entirely my own responsibility.

C. Hawksley
June 1982



Contents

<i>Preface</i>	xi
1 Introduction	1
Part 1 Foundations of programming	3
2 Data and information	5
2.1 The computer as a tool	5
2.2 Symbols and symbolism	6
2.3 Information representation	7
2.4 The number crunching myth	9
2.5 Data types	10
3 Algorithms	16
3.1 Problem solving	16
3.2 Algorithms and language	19
3.3 Another myth	22
4 Computers and programs	25
4.1 A computer model	25
4.2 Programming languages	29
4.3 Operating systems	31
5 First steps in Pascal programming	33
5.1 Pascal program construction	33
5.2 Identifiers and declarations	36
5.3 Statements	38
5.4 Expressions	39
5.5 Standard functions	43
5.6 Boolean expressions	44
5.7 Reading and writing	46
5.8 Semicolons and compound statements	47
5.9 A short program	48
6 Control structures	51
6.1 Conditional statements	51

6.2 Repetition	56
6.3 Choosing loop structures	59
6.4 The case statement	62
7 Procedures and functions	65
7.1 Declaring and calling procedures	65
7.2 Local variables	67
7.3 Global variables and parameters	69
7.4 Functions	74
8 Input and output	78
8.1 Modes of programming	79
8.2 Read and readln	80
8.3 Write and writeln	83
Part 2 Problem Solving by Computer	87
9 Program design	89
9.1 'Good' programs	90
9.2 Structured programs	90
9.3 Comments	93
9.4 Data structure	93
9.5 Top-down program design	94
10 Problems involving small quantities of data	99
10.1 A simple program	99
10.2 Conditionals in use	101
10.3 Looping the loop	105
10.4 Alternative strategies	106
10.5 Using procedures	109
10.6 Sets and in	111
11 Using more data	116
11.1 Arrays	116
11.2 Using arrays	117
11.3 Sorting	123
11.4 Tables of data	126
11.5 array and type	133
12 Text processing and files	137
12.1 Files	137
12.2 String manipulation	141
12.3 Word processing	143
12.4 Grammatical analysis	152
13 Analysing data and presenting results	160
13.1 Packages	161
13.2 Files, records and fields	162
13.3 Pascal records	164

13.4	Entry and verification of data	166
13.5	Presenting results	168
14	Further Pascal	173
14.1	Additional data types	173
14.2	Records and pointers	174
14.3	Recursion	174
14.4	Non-text files	174
14.5	File pointers	175
	Appendices	176
1	Pascal syntax diagrams	176
2	Reserved words	184
3	Standard functions	185
	Index	186

1

Introduction

The last thing one knows in constructing a work is what to put first.
Pensées, Blaise Pascal (1623–62)

There are certain similarities between learning to program a computer and learning to play a musical instrument. In case the music profession or the reader are alarmed by this let it be added quickly that the similarities lie in aspects of the learning process and not in the activities themselves. In common with many other learning processes, such as learning to cook or to drive a car, both require an assimilation of three basic components: background knowledge, technical skill and creative art.

In learning to play a piano, for example, it is not essential to know precisely how the piano is constructed; how the hammer mechanism is made or how to tune the instrument, but a basic level of appreciation of the mechanics is most necessary. The fact that a note is struck and then decays, that the loud and soft pedals affect the note quality in particular ways are examples of this simple, but important, background knowledge. In computer programming it is not essential to know how a computer works from an electronic viewpoint, for instance, nor even in the case of larger remote-access computers is it necessary to know where the computer is located physically. On the other hand, it is important to appreciate some of the general principles on which a digital computer operates in order to gain a 'feel' for the tasks to be performed. For this reason, the first few chapters of this book aim to introduce the kind of background information that is relevant to the programming of a computer. Terms such as data, data processing and algorithm are explained and a model is used to describe the fundamental workings of the computer itself.

In the case of the student musician, technical ability must be acquired

through the learning of scales, finger positioning and chords, for example. The repertoire of basic skills is gained partly from instruction by tutor or text and, perhaps largely, by a 'practice makes perfect' process. The parallel skills in programming entail the learning of a programming language and the way in which its constructions may be put to practical use. Again, the educational process should include a substantial element of practical involvement: the writing of small programs to reinforce the theory. With this in mind, the later chapters of part 1 introduce fundamental constructions of the Pascal programming language together with short examples to illustrate their use. It is important for the reader to supplement this by attempting short exercises of the kind found at the ends of these chapters.

The introductory material is covered fairly quickly in part 1 and more advanced details are omitted at this stage. The objective is to move on to the creative side of programming as soon as possible, since the writing of complete programs to solve actual problems is our ultimate aim. Armed with the basic techniques, part 2 begins to explore the art of problem solving: taking a loosely defined problem, creating a precise program design and writing a complete Pascal program. In musical terms we begin to play pieces of music using our own style and interpretation.

The examples in part 2 are chosen to demonstrate reasons for selecting particular kinds of programming constructions and ways of putting them together. Thus, much of the material introduced in part 1 is revisited in the case studies of part 2 with an emphasis on practical applications. If you experience difficulty in understanding a new Pascal feature introduced in the earlier chapters, bear in mind that there is likely to be a further reference or references in the index to later case studies which may be of assistance. Also in later chapters, several new Pascal facilities are introduced which can be added to the repertoire of programming skills once the fundamentals have been firmly established.

There is no substitute for practical programming experience as a way of boosting the confidence and of improving one's ability to cope with new problems. Yet problem solving is a fascinating and rewarding art which will more than repay the initial effort to master the use of the building blocks of programming. Note, finally, that one must always beware of taking analogies too far. The fact that some musical instruments and some computers possess keyboards is perhaps the only real similarity between the two after all!

Part 1

Foundations of programming

Data and information

2.1 The computer as a tool

It is easy to forget that the computer is a tool constructed by man. Perhaps due to ignorance or fear of a rapidly expanding technology many people have overlooked this fact. It is currently fashionable to attribute man-like features to the computer; even to call it a superhuman brain capable of enormous mental feats performed with tireless efficiency. Indeed, this machine has become so humanised that we read regularly of computers making mistakes. Bills issued for £0.00 form a source of amusing material for the newspaper columnist. Complaints ranging from the delivery of wrong goods to the erroneous disconnection of electricity meters are put down to the apparently unavoidable occurrence of a computer error. The consequences are not always amusing. An inanimate collection of circuits has taken over from the anonymous clerk as the perfect scapegoat for administrative irresponsibility.

Yet, in the same way that we would not entertain a claim that a carpenter's chisel made a mistake or that a writer's pen spelled a word wrongly we should recognise this twentieth century example for what it is: a bad workman blaming his tools. For a computer is as much a tool as a chisel or a pen. Furthermore, it is a deterministic device. It can and will do precisely what it is told to do and only that, in common with chisels and pens. Like all artifacts the computer is prone to malfunction, but this is not at all the same as making a mistake. How often do we encounter pens which misspell as a result of the nib breaking?

Thinking of the computer as a tool in this sense provides a convenient starting point for this text. We are faced with a device which is not human, not intelligent and which we must learn to use. The appropriate term is to *program* a computer. More than with many longer established trades the technical terms associated with this tool are numerous and apt to

bewilder the new computer user. Where it is necessary to use these terms we shall define them at the time they are first encountered.

Having introduced the computer in this way we should not be discouraged by the fact that it is a mere tool. It is an immensely powerful and general one as we shall see. More immediately, let us examine the raw materials on which our tool is to work.



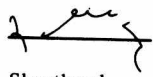
2.2 Symbols and symbolism

In computing we are concerned not with the fashioning of some physical medium as in the case of a chisel but with the manipulation of intangible symbols. Symbols have been in use considerably longer than computers. From hieroglyphics to Morse code, the Greek alphabet to shorthand the use of symbols was one of man's first steps on the road to civilisation. A few examples of symbols are shown in figure 2.1.

Interestingly, we have become so accustomed to using symbols that we take them for granted. More precisely, we do not distinguish clearly between a symbol and our interpretation of that symbol. Symbols are literally signs; marks on paper, shapes carved on stone, holes in punched cards or a character typed on a keyboard are all examples of signs. Thus, one may describe the symbol 'O' as a circle drawn on paper, no more and no less.

However, to make use of a symbol we must impose some kind of interpretation on it. Hence we may interpret the symbol 'O' as the 15th letter of the English alphabet. In this sense the symbol 'O' is an example of an item of *data* and the interpretation placed on the symbol is an example of *information* conveyed by the data. This is more than a trivial distinction.

Figure 2.1. Symbols.

CLAVDIVS IV	$\alpha \beta \gamma$	1234567890
Roman characters	Greek letters	Arabic numerals
		
Hieroglyphics	Chinese	
001101110	— • • — • •	
Binary	Morse code	Shorthand