Cambridge Computer Science Texts • 15

Information Representation and Manipulation using Pascal

E. S. PAGE

L. B. WILSON

Cambridge Computer Science Texts · 15

Information Representation and Manipulation using Pascal

E.S. PAGE

Vice-Chancellor
University of Reading

L.B. WILSON

Professor of Computing Science
University of Stirling

CAMBRIDGE UNIVERSITY PRESS

Cambridge

London New York New Rochelle

Melbourne Sydney

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

First published 1983

Printed in Great Britain at the University Press , Cambridge

Library of Congress catalogue card number: 82–4505

ISBN 0 521 24954 6 hard covers
ISBN 0 521 27096 0 paperback

British Library cataloguing in publication data

Page, E. S.
Information representation and manipulation using Pascal. -
(Cambridge computer science texts; 15)
1. Electronic data processing
I. Title II. Wilson, L. B.
001.64'42 QA76

ISBN 0 521 24954 6
ISBN 0 521 27096 0 Pbk

Preface

For several years a large number of students at universities, polytechnics and schools have been taught or have taught themselves a programming language and have written and run programs on a computer that has been conveniently available. Only very little experience of this kind is required to show that there are other topics which it is important for a computer user to know and the foremost among these are the ways in which information of different kinds can be represented and manipulated in the computer. Such material is largely independent of the actual computer and there is even a degree of independence of the actual programming language. It will, of course, be easier and more convenient for students to study algorithms in a language with whose form they are familiar. Since the second edition of Information Representation and Manipulation in a Computer was published in 1978, there has been a movement away from Algol-like languages (e.g. Algol 60, Algol W, Algol 68) to Pascal in British universities. This movement is particularly noticeable in specialist computing science courses. Even in North America Pascal has made considerable inroads on the established languages (most commonly Fortran and PL/I) in the universities. While these changes in the university world have been very marked, a modest move to Pascal has occurred in industry and such a movement seems likely to continue. It was never intended that the original text would be heavily dependent upon the programming language used but in view of the changing situation we felt students would benefit from having the algorithms in a language familiar to them, especially those needing a self-teaching text. This text is intended for students on an introductory course and requires only a knowledge of Pascal, at least as far as loops, arrays, procedures and functions. We expect readers to be able to write and test programs on a computer. Mathematical demands on the reader have purposely been kept as slight as possible and only in a few places, where the full analysis of some algorithm demands it, is more than elementary algebra needed. The main thread of the book will not be lost if the reader has to skip such portions. Much can be gained from teaching the more advanced parts of Pascal, particularly records and pointers and recursive procedures, with the material of this book. The programming principles and appropriate examples of data structures have been found to go very well together. Examples, such as adding sparse matrices or manipulating polynomials, are ideal in conjunction with records and pointers, whilst trees, searching and sorting contain many natural examples of recursion which are a much better introduction to this fundamental and important concept than Ackerman's function or even the more familiar and simpler factorial.

The material in this book runs parallel to that in our earlier volume. It can be covered in about two dozen 1-hour lectures and can be given at a quite elementary stage in a student's study of computing. It can form part of the first-year work of undergraduates intending to specialise to some extent in computing science but it can just as easily fit into a course at a later stage for those using computing as a tool, spending a smaller proportion of their time on the subject. The text also contains little that is not required for the first stage of the British Computer Society's examination.

The changes to material in the earlier volume proved more extensive than expected and some algorithms (e.g. quicksort) in the original editions have had to be considerably modified. That we know will come as no surprise to some of our friends who suggested changes to these algorithms before now. We are very conscious that altering algorithms and programs is a very fruitful source of errors. It is too much to hope that we have entirely avoided all errors in spite of our striving to do so and we will, of course, be grateful for any suggested corrections or improvements.

At the end of each of the chapters examples are given to enable the student to test his understanding of the contents. Some of the examples are straightforward exercises of drill, while others are rather longer questions taken from university examination papers. Such a written examination is unlikely to form the whole of the assessment for any course and will normally be supplemented at least by some programming project but these questions represent a type of assessment which is usually present at some stage. We have not indicated the year of the course in which the question has appeared because such information would only be meaningful when coupled with the knowledge of how a particular university arranged its courses. In the same way, certain questions have been taken from papers testing the work of some MSc conversion courses. The third type of exercise which is included are some suggestions for rather longer programming projects which might require a month or more of a student's own time to be devoted to the subjects. Some hints on the solutions of the examples are given at the end of the book in sufficient detail, we hope, to indicate how someone who is stuck should proceed and yet sufficiently tersely to discourage a student from presenting our hints as his solution should it be set by his lecturer.

We shall not repeat the acknowledgments which were made in the previous book but we are still indebted to the many mentioned there. We hope it will be helpful for those who need to learn Pascal, or to brush up their knowledge before starting on this material, to have a short list of what we consider to be the best of the recent books published on Pascal programming and a list is given at the end of this Preface, [1]-[6]. It is perhaps to be expected that the designer of

Pascal should write his algorithms with clarity and style and certainly Wirth's book [7] is a joy and thoroughly recommended. Two other books which cover data structures using Pascal are those by Coleman [8] and Tanenbaum and Augenstein [9].

E. S. Page and L. B. Wilson

SELECTED BIBLIOGRAPHY ON PASCAL

[1] I. R. Wilson and A. M. Addyman: A Practical Introduction to Pascal, Macmillan Computer Science Series, 1978.

[2] Jim Welsh and John Elder: Introduction to Pascal, Prentice-Hall, 1979.

[3] W. Findlay and D. A. Watt: Pascal: An Introduction to Methodical Programming, Pitman, 1978.

[4] G. M. Schneider, S. W. Weingart and D. M. Perlman: An Introduction to Programming and Problem Solving with Pascal, John Wiley, 1978.

[5] Peter Grogono: Programming in Pascal (revised edition), Addison-Wesley, 1980.

[6] J. S. Rohl and H. J. Barrett: Programming via Pascal, Cambridge University Press, 1980.

[7] N. Wirth: Algorithms + Data Structures = Programs, Prentice-Hall, 1976.

[8] Derek Coleman: A Structured Programming Approach to Data, Macmillan Computer Science Series, 1978.

[9] A. M. Tanenbaum and M. J. Augenstein: Data Structures using Pascal, Prentice-Hall, 1981.

Contents

1 · Symbols on Paper

1.1 INTRODUCTION

The good teacher or lecturer aims to present interesting and challenging ideas, results and arguments which may be quite complex in a manner which seems straightforward and simple to the student. In contrast, the composers of different forms of puzzles and brain teasers for newspapers and magazines aim to produce formulations of problems which appear intriguing and perhaps quite difficult but which still allow a simple solution once the correct method of attack has been discovered. The ways the problems are presented have a large effect upon the ease of solving them. The puzzle setter and the teacher have each made a choice of how their material should be presented - a choice guided by the effects they hope to achieve. At a more detailed level the notation chosen for the various quantities occurring in a problem can have a dominating influence on the ease with which a solution may be found. For example, some of the early representations of numbers seem to have been devised primarily for recording amounts of goods; so many cattle, so much corn, and so on. Different units had symbols of their own and other numbers were composed by the appropriate repetitions of those symbols. Such a system of representation does not place too much hindrance in the way of the operations of addition and subtraction; the symbols themselves can be repeated or deleted easily and the occasional replacement by equivalent groups of symbols performed. It is much more difficult to perform multiplication and division - operations presumably less frequently required in the applications that were routine. The same sort of difficulties occur with the more recent Roman numerals, although some additional complexities appear (and produce the material for a sequence of elementary programming examples). The hindrances to multiplication and division become much less once the cypher (a figure 0) is introduced together with the usual positional notation. For example, in the scale of ten, 3080 represents three thousands and eight tens $(3\times10^3+0\times10^2+8\times10^1+0\times10^0)$. Even here, however, some operations are performed more easily than others. In this scale multiplication by ten is an easier operation to perform than multiplication by two even, and certainly by seven. Other scales have their own properties which ease some operations and make others more difficult.

The same kind of differences in the ease of performing given operations can be noticed in many areas far removed from arithmetic or mathematics. For example, given a tuned guitar or banjo and an air described by the musical score, those with as little musical training as the authors would be at a loss to produce anything recognizable. If, however, the grid representation of the fingering is given we might manage to produce some of the right notes in the right order if not in the right tempo. Conversely, however, the fingering symbols alone would not make it easy for even a skilled musician to play the tune on a piano or on a wind instrument. This theme will constantly recur in different guises throughout this book. Any representation of information that has been chosen will govern what operations are easy and convenient to perform and, conversely, a representation should be selected taking fully into account the operations which it is necessary to perform.

1.2 COMPUTER OPERATIONS

The early uses to which computers were put were predominantly numerical. Ballistic and navigation tables were produced; the numerical solutions were printed for mathematical problems arising in different branches of engineering and science. Thus, at first computers tended to be regarded solely as devices for performing arithmetic operations; however, it soon came to be recognized that there was a substantial amount of logical and administrative work contained even in a computer program for numerical calculations. The contents of a storage location had to be examined to see if it was positive or negative, perhaps to control a count of how many times a loop had been obeyed or whether another iteration was needed. The contents of stores had to be moved to different locations in the machine; during the moving operation the meaning of the pattern of electrical signals would be irrelevant - a copy was being made for later interpretation. Sometimes parts of the contents of a location had to be abstracted or the contents moved relative to themselves as in a shift operation. Counts of the different sorts of operations showed that even in a numerical calculation the proportion of the administrative and non-numerical operations was high. There was thus a change in emphasis to cause computers to be regarded as devices for performing operations on symbols; numerical digits just became special cases of the more general class of all the symbols represented. We therefore shall look first at single symbols and then at groups of symbols which are used in several different applications independent of computers and shall consider the different sorts of operations that are performed upon them; later we restrict attention to uses in computers. For our purposes, therefore, we regard information as being conveyed by symbols which are distinguishable one from another and any meaning that they have will be governed by the rules of the particular context in which they appear.

1.3 SINGLE SYMBOLS

The most commonly used single symbols in the western world are surely the letters of the English alphabet, a, b, c, ... Notice, however, that there are many variations possible even in this simple example. In typescript letters may occur in lower case a, b, c, ... or in upper case A, B, C, ... In printing they can occur in a variety of different type founts as well; for example, italic, bold, script and many others. The letters can be different sizes - from the small print often used for the limiting conditions of guarantees and legal agreements through the sizes used for the headlines of the popular newspapers to the display characters for advertisements on the hoardings. Notice again that the choices of representation on the paper have been made in order to try to achieve one or more principal aims while satisfying to some extent subsidiary aims. For example, the 'small print' on a purchase contract can, at least charitably, be justified by the need to compress many symbols into a small space, with a subsidiary requirement that the symbols can be read with normal aids to eyesight - the microdot recording used by spies not being permitted.

Representations of characters which we recognize as the same in some respects are produced by very many different means. For example, from an ordinary typewriter upper case letters are caused by depressing a case shift key and then striking the key for the letter so that a different part of the head of the moving arm strikes the inked ribbon onto the paper. On some machines (e. g. flexowriters, teleprinters) which produce a paper tape output as well as a printed copy, a case shift key has to be struck which causes a pattern of holes on the paper tape to be produced as well as placing the machine in the state ready to print letters in upper case when the keys are struck. In some methods of printing both upper and lower case letters are represented uniquely by individual keys.

The familiar decimal digits, 0, 1, 2 . . . 9 appear on most keyboards and are adequate for representing numbers in scales of 10 or less but need to be supplemented for scales with higher radices. For example, hexadecimal numbers, i. e. those in the scale of 16, need symbols to represent the decimal 10, 11, 12, 13, 14 and 15; by one convention these have been taken to be A, B, C, D, E, F, so that the hexadecimal A5 is 165 in the scale of ten. In this case it is quite usual for some of the keys to be used for two different purposes - BED represents both a hexadecimal number and an English word meaning something to sleep in or plant flowers in. It is worth noticing, however, that we do not need to go to exotic scales of notation to find examples of double usage of symbols. Even on some common keyboards not all the decimal digits are available; for example, on many typewriter keyboards the digits zero and one are absent - the typist is supposed to produce them by using the upper case O (i. e. the letter between N and P in the alphabet) and lower case 'el', a substitution which produces errors often noticed in the output from inexpert typists.

Restrictions on the size of keyboards lead to the omission of some needed symbols and so to tricks to avoid them. Consider the mathematical signs $+ - \times \div /$ $> \geq = \neq$. The first two signs are usually available but multiplication might have to be indicated by the letters x or X - for example, on some typewriters. Other signs like $\neq$ might need to be constructed from an over-printing of $=$ by $/$; yet others like $\geq$ may have to be replaced by some combination like . GE. Some character sets require such a construction for a large number of their symbols; many of the operators of the Iverson notation in his language APL require more than one key stroke. In all these cases we notice that some choice has been made of what set of characters should be represented by a single key, how big that set should be, how other characters may be represented (if at all); a good choice has regard to the most frequent or most important uses.

1.4 DESCRIPTIONS OF SOUNDS AND POSITIONS

Not all phenomena are described most conveniently by alphabetical or numerical symbols for the operations which have to be performed. Descriptions by these symbols may take up too much space or too much time, or they may be less readily distinguished than a suitable stylized two-dimensional picture. A secretary taking dictation from someone speaking only moderately fluently will record the words spoken in one of the shorthand systems that have been developed. Two of the most common, Pitman's and Gregg's, represent the sounds of the words rather than their spelling. A number of basic outlines written in a few positions relative to the horizontal lines on the paper are combined to form all the words of the language. The operation of encoding by a skillful shorthand-writer can be fast enough to record what even quick speakers are saying, and the systems have surely been designed with this primarily in mind. Decoding, accompanied as it usually is by some form of transcription on a typewriter, need not be quite as quick or even in most cases quite as accurate if the context or the secretary's memory can afford some clue in the case of a slightly inaccurate written outline. The ordinary representation of a music score as well as the one quoted earlier for a banjo or guitar, uses a two-dimensional display on the printed page. In these cases the operations that need to be performed quickly are the recognition of the sounds required and what has to be done to produce them from an instrument. The converse construction, printed representation produced from the sounds, need not normally be performed at the same speed and the notation is not well adapted for this purpose. In some other spheres it is necessary to describe positions in two or three dimensions, for example, in ballet or modern dance routines. In both of these cases essentially two-dimensional forms of representation on the paper have been devised but because of the nature of the activity are perhaps more suited to recording a sequence of movements rather than to assist their execution at the desired rate.

1.5 OTHER REPRESENTATIONS BY SYMBOLS

The previous sections have given examples of some representations on the written or printed page and have focussed attention on properties of the representations. Paper and printing are, of course, not the only means of recording or transmitting information. The deaf and dumb alphabet, formed from positions of the hands and fingers, gives another representation of the alphabet plus a small number of words and phrases. Another visual representation of the alphabet is given by semaphore which uses the positions of flags held in the hands to describe the letters and digits. A much more extensive system for conveying messages from a previously determined list has been constructed for ships at sea which uses a variety of devices including flags of various colours. Sight is only one of the senses available for the recognition of information. Braille notation is recognized by touch, and the Morse Code in one of its varieties typically uses long or short sounds or long or short flashes of light in patterns to represent the printed symbols. Even smell gives some information, especially to animals. All these representations and the ones mentioned earlier allow some operations to be performed more easily than others. If the more awkward operations have to be performed relatively infrequently it may be worthwhile retaining the representation because it is familiar. It must, however, be recognized that a decision about the choice of a representation needs to be made, and the consequences of the choice contemplated and weighed against alternatives. The examples quoted are naturally those which have stood the test of considerable use and have proved their suitability under the most frequent types of use. We shall see later that some sort of standardization is appearing for the representation within computers of the most common characters, but whenever more unusual applications appear a choice may have to be made ab initio.

EXAMPLES 1

[1. 1] For any keyboards producing a hard copy with which you are familiar, list:

(a) the single symbols in the printed representation which require more than one key stroke to produce them;

(b) those keys which represent different printed symbols according to context (as with the lower case 'el' quoted in section 1. 2);

(c) those keys which produce no printing.

[1. 2] Suggest principal and subsidiary aims that might be involved in a choice of:

(a) type size for newspaper headlines;

(b) dimensions for microfilm or microfiche records of research articles;

(c) dimensions for a microdot;

(d) foreign characters for certain scientific or mathematical quantities (e. g. γ-rays, $\pi = 3.14159 \ldots$).