JENSEN, K. and WIRTH, N. (1975). *Pascal User Manual and Report*, Springer-Verlag.
SCHMIDT, J. W. (1977). Some high-level language constructs for data of type relation, *ACM Transactions on Database Systems*, Vol. 2 No. 3.
SMITH, J. M. and CHANG, P. Y. T. (1975). Optimizing the performance of a relational algebra database interface, *CACM*, Vol. 18 No. 10.
STONEBRAKER, M., WONG, E., KREPS, P. and HELD, G. (1976). The design and implementation of INGRES, *ACM Transactions of Database Systems*, Vol. 1 No. 3.
WEDEKIND, H. (1974). On the selection of access paths in database systems, in *Data Base Management* North-Holland.
WIRTH N. (1975). *Pascal-S: A subset and its implementation*, ETH, Zurich.

Book reviews

Software Psychology—Human Factors in Computer and Information Systems, by Ben Schneiderman, 1980; 320 pages. (*Prentice-Hall*, £16·20)

As the range of subject matter covered may not be entirely clear from the title, I will start with a list of chapter headings. These are: 1. Motivation for a psychological approach; 2. Research methods; 3. Programming as human performance; 4. Programming style; 5. Software quality evaluation; 6. Team organisations and group processes; 7. Database systems and data models; 8. Database query and manipulation languages; 9. Natural language; 10. Interactive interface issues; 11. Designing interactive systems; 12. Computer power to, of and by the people.

I found the book both readable and interesting. It is mostly devoted to discussions and comparisons of different approaches and methods in the areas covered by the chapter headings, supported by extensive references to published work. The book has a 20 page bibliography. Two examples of topics covered in some detail are the value of flowcharting in program development and comprehension, and different approaches to software quality evaluation. A satisfactory feature is that, where appropriate, the statistical significance of results is quoted and the author usually attempts to reach a view on what the (sometimes conflicting) evidence means. Although—perhaps inevitably—firm conclusions are rare, the summaries of research evidence on a range of software topics could be of interest to quite a wide spectrum of readers. At the end of each chapter is a useful 'practitoner's summary' followed by a 'researcher's agenda' indicating where further work is needed.

There are, however, some gaps, and it is not entirely clear how the main areas covered in the book have been selected. For example, the direct interaction between human beings and computers via various types of language is given extensive treatment. On the other hand the problem of establishing, defining and specifying requirements for software, which is at least as difficult and which has considerable psychological content, is hardly covered at all. Perhaps this is because there has been little psychological research in this area due to a number of factors but one would have expected it to be discussed and listed as an important field for future work.

Software Psychology is a valuable book for anyone seriously concerned with the practice of programming and particularly those involved in research into human factors aspects. Those with a more general interest in computing and software will find much that is useful, although for such readers the book, at £16·20, will probably be one to borrow rather than buy. It is well produced with clear printing and I did not spot one typographical error.

J. N. G. BRITTAN (Chertsey)

Programming Standard Pascal, by R. C. Holt and J. N. P. Hume, 1980; 381 pages. (*Prentice-Hall* £7·75)

This book, like most elementary programming texts, is good in parts. Its aim is to teach basic programming in Pascal by using series of subsets of the language that include one another like a set of Russian dolls. The technique was developed for PL/I by R. Holt and D. Wortman. (The Venn diagram used to illustrate the subset idea is labelled wrongly, since the largest, all enclosing, set is labelled PS/1, the name of the smallest subset that the authors define.)

The first two chapters are devoted to an introduction to computers and programming and a justification of the subset approach. This seems overlong; especially when trying to explain structured programming to readers who cannot program. Each of the following three chapters introduces a new subset of the language. The first contains only arithmetic expressions and printing of values; it seems rather small, and could profitably have been combined with the second subset which introduces variables and declaratives as well. The third of these subsets is far too large—entitled 'Control flow'. The chapter includes **while, repeat** and **for** loops, **if** and **case** statements, **Boolean** variables and expressions—all within 20 pages. I would introduce this wide selection of topics much more slowly—probably leaving the **case** statement until enumerated types had been introduced and develop the **for** statement with **arrays.**

After the introduction of **arrays** and the various sorts of type (subrange, named, enumerated), the presentation improves a great deal. A chapter is devoted to top-down development of solutions and choice of data structures. Procedures and functions are introduced competently.

An excellent section now follows with chapters devoted to Modular programming, Searching + sorting, Making sure the program works and Data structures. Records and pointers are painlessly introduced here. The chapter on data structures introduces stacks, queues and trees. Unfortunately, recursive procedures are introduced, and explained in terms of using a stack to print a list in reverse order. This is a most un-natural use for them—even worse than the usual function to calculate $n!$. The example implementations given for stacks and queues using arrays contain no check on overflow or underflow. This is most serious in the case of the queue implementation since modular arithmetic is used to keep the indices within the array bounds.

The remaining chapters of the book are again of limited usefulness. Scientific calculations gives a procedure for plotting approximate curves on the line printer, but is otherwise little more than a superficial overview of the problems attacked by numerical analysis. Similarly the chapter Numerical methods only presents an arm's length picture of the problems involved.

A short chapter gives examples of programs in other languages, and the two final chapters are devoted to a discussion of machine language and assemblers and the development of a compiler for a (very) restricted subset of Pascal.

Each chapter concludes with a short summary and a set of exercises. There were few typographical errors and the only factual error I noted was the statement 'In division the relative error of the quotient is the difference between the relative errors of the divisor and the dividend'. (It is, in fact, the sum).

P. KING (Newcastle)

Architecture and the Microprocessor, by John Paterson, 1980; 229 pages. (*John Wiley*, £13·80)

Despite its title this book is not concerned too much with architecture or the microprocessor—and most certainly not about microprocessor architecture. It is in fact a middlebrow Sunday supplement social history which looks at how cheap computing power might affect the practice of architecture. To do this the author traces the development of the art of architecture showing how changes in society have affected the architect and goes on to discuss how current changes in our society, and the microprocessor revolution in particular, might shape our future environment. The book may be an interesting diversion for the computer professional in two ways. Firstly, it is a good example of the impact of computers on society generally. Secondly, because the design of buildings is, in many ways, analogous to the design of computer software, this book may give an insight into the type of problem which may soon be confronting computer scientists.

ALAN H. BRIDGES (Glasgow)