DAHL, O-J., DIJKSTRA, E. W. and HOARE, C. A. R. (1972). *Structured Programming*, Academic Press, London.
DIJKSTRA, E. W. (1968). GOTO statement considered harmful, *CACM*, Vol. 11, pp. 147-148, 538, 541.
FITTER, M. J. (1976). Computers as aids in forecasting and control, MRC SAPU Memo No. 117.
GREEN, T. R. G. (1977). Conditional program statements and their comprehensibility to professional programmers, *J. Occup. Psychology*, Vol. 50, pp. 93-109.
IBM PROGRAMMING RESEARCH GROUP. (1954). Specifications for the IBM mathematical FORmula TRANslating system, FORTRAN preliminary report, IBM Applied Science Division, IBM Corporation, New York.
LANDIN, P. J. (1966). The next 700 programming languages, *CACM*, Vol. 9, pp. 157-166.
KNUTH, D. E. (1974). Structured programming with GOTO statements, Stanford University Comp. Sci. Dept. Memo No. STAN-CS-74-416.
NAUR, P. (1963). Goto statements and good Algol style, *BIT*, Vol. 3, pp. 204-208.
NAUR, P. (1966). Program translation viewed as a general data processing problem, *CACM*, Vol. 9, pp. 176-179.
SIME, M. E., ARBLASTER, A. T. and GREEN, T. R. G. (1977a). Reducing programming errors in nested conditionals by prescribing a writing procedure, *Int. J. Man-Machine Studies*, Vol. 9, pp. 119-126.
SIME, M. E., ARBLASTER, A. T. and GREEN, T. R. G. (1977b). Structuring the programmer's task, *J. Occup. Psychol*, Vol. 50, pp. 205-216.
SIME, M. E., GREEN, T. R. G. and GUEST, D. J. (1973). Psychological evaluation of two conditional constructions used in computer languages, *Int. J. Man-Machine Studies*, Vol. 5, pp. 105-113.
SIME, M. E., GREEN, T. R. G. and GUEST, D. J. (1977). Scope marking in computer conditionals—a psychological evaluation, *Int. J. Man-Machine Studies*, Vol. 9, pp. 107-118.
WIRTH, N. (1971). Program development by stepwise refinement, *CACM*, Vol. 14, pp. 221-227.
WIRTH, N. (1977). MODULA: a language for modular multiprogramming, *Software—practice and experience*, Vol. 7, pp. 3-36.

Book reviews

Optimization Techniques, parts 1 and 2, edited by J. Stoer, 1977; 528 and 512 pages. (*Springer-Verlag* Lecture Notes in Control and Information Sciences, Vols 6 and 7, $21.50 each)

The Conference on Optimization Techniques, Würzburg, 1977, was attended by 240 participants and the proceedings contain over 240 papers, some of which have multiple authorship. It is difficult within the constraints of a short review to give the full flavour of the meeting. Largely this is a collection of papers on applied mathematics, ranging over stochastic processes, computational techniques, differential equations, control theory, a lot of mathematical programming, some very mathematical OR and economics, computer networks and differential games.

Although the proceedings of the conference are divided into two volumes, the first mainly dealing with optimal control, and the second dealing with mathematical programming, it is rather confusing to find that a number of papers are in both volumes. It is a pity that some of the very important areas, such as world modelling, are dealt with so briefly and in extremely general and chatty terms.

The conference itself was probably more lively than would appear from these proceedings, which are extremely difficult to read and have but little relevance to the application of optimising techniques in that messy, muddled, chaotic real world in which we all live. If only the world was simple enough for mathematical methods to be applied widely, it would indeed be a Platonic heaven.

PATRICK RIVETT (Brighton)

A Primer on Disciplined Programming, by R. Conway, 1978; 419 pages. (*Prentice-Hall*, £6·95)

It was fashionable, a few years ago, for each new textbook on numerical analysis to start with a few chapters on ALGOL, FORTRAN, or latterly BASIC before launching into the main subject material when, in many cases, there would be no further mention of the particular language studied in the early part of the book. Disciplined programming, programming style and structured programming are topics on which there are a number of newly published texts and this compares well with all of them but emulates the numerical analysis books in commencing with a detailed study of a particular dialect of PL/I.

The book is divided into five parts, parts I and III being essentially a beginners guide to PL/I, part II is on program development, part IV is devoted to quality of programs and part V to the limits of computing. Parts I and III take about two thirds of the book and this will inhibit any prospective purchaser who does not wish to learn PL/I or who regards himself as competent in that language. This is a pity since the presentation is well thought out and using 'Programming Proverb 24' (1975) which states 're-read the manual', there is much to be learnt from the description of the features of the language. Part II on the development of programs contains some good ideas, put across in a competent manner. It covers program structure, top-down development and the structure of data. It is rather strange that this should appear before the introduction of procedures and functions which inhibits coverage of modularity. This section is not dependent on PL/I to any great extent.

Part IV on the quality of programs contains a number of ideas, if not original, then rare in print, and ought to be on the reading list of every serious high level language programmer. It covers program correctness and program efficiency. The final, very short, Part V on the limits of computing covers popular wisdom about computers (i.e. myths) and problems impossible to compute.

The book was produced by an editing program (FORMAT) run on the IBM 370/168 of Cornell University's Office of Computer Services. At first view the small typeface is unattractive but despite the smallness of the print it is not difficult to read, the only criticism of the print being that, possibly due to use of the computer, there are no flowcharts of the examples and exercises. There are many (too many?) books on programming and few really good ones. This wil hold its own with the best, it could be strongly recommended to any student of PL/I even though its subtitle and presentation are directed to the Cornell compilers. It should also prove interesting to any lecturer or teacher of programming who could learn from the presentation of the material and the well constructed exercise at the end of the chapters.

D. W. B. BALE (Swanage)

Reference

LEDGARD, H. F. (1975). *Programming Proverbs*, Hayden