```
0 → TOTAL;
FOR X IN HYBRID H:
X + TOTAL → TOTAL
ENDFOR
```
(The code generated is not given—it is left to the reader to realise that it is not particularly attractive.)

Summary

What does, and what does not, constitute 'convenience' in programming languages is somewhat a question of individual choice. The author feels that coding for iteration is more easily understood if, whenever a data path is involved, the nature of the data path is immediately apparent.

Like many language features, the above facility is decidedly no 'cure-all', but its use appears advantageous for a certain range of commonly occurring situations.

The techniques presented here are only suitable for iterations involving a single data path. This is quite a serious limitation (e.g. how does one handle element-by-element assignment?). More general techniques are being developed to handle such cases (this appears to involve the manipulation of text which represents program actions having multiple entries and/or multiple exits—and the use of 'connector functions' to combine these components in a more general manner than is allowed in 'conventional' structured programming—however that is a matter for further research).

One would expect the above techniques to produce (source) program code which is relatively insensitive to choices of data structure. This may allow convenient implementation of decisions concerning the mapping of abstract data structures to physical data structures when implementing very high level languages. However, this is again a topic of further research.

The author would like to thank the referee for useful suggestions on improving the presentation of this paper.

References

Barron, D. W. (1974). APL and POP-2: What can we learn from Interactive Languages?, *High Level Languages—The Way Ahead*, BCS Conference Proceedings.

Burstall, R. M., Collins, J. S., and Popplestone, R. J. (1971). *Programming in POP-2*, Edinburgh University Press.

Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R. (1972). *Structured Programming*, Academic Press.

Denning, P. J. (1973). Letter to the Editor, *SIGPLAN Notices*, October 1973.

Dijkstra, E. W. (1968). Go To Statement Considered Harmful, *CACM*, Vol. 11, No. 3, pp. 147-148.

Gries, D. (1974). Letter to the Editor, *CACM*, Vol. 17, No. 11, pp. 655-657.

Jensen, K., and Wirth, N. (1974). PASCAL—User Manual and Report, *Lecture Notes in Computer Science*, Springer-Verlag.

Leavenworth, B. M. (1966). Syntax Macros and Extended Translation, *CACM*, Vol. 9, No. 11, pp. 790-793.

Wirth, N. (1971). Program Development by Stepwise Refinement, *CACM*, Vol. 14, No. 4, pp. 221-227.

Woodward, P. M., and Bond, S. G. (1974). *ALGOL 68-R Users Guide*, HMSO.

Wulf, W. A. (1972). A Case Against the GOTO, Proceedings of ACM National Conference, Boston, pp. 63-69.

Book reviews

Computer Science and Technology and their Application, General Editors: N. Metropolis, E. Piore and S. Ulam, 1975; 310 pages. Administrative Editors: Mark I. Halpern, William C. McGee; Contributing Editors: Louis Bolliet, Andrei P. Ershov, J. P. Laski. (*Pergamon Press*, £15·00)

Contents

It appears immediately that this volume is not annual, nor is it a review; it is hardly automatic programming, and the contributing editors did not contribute. Nevertheless, it is a selection of articles on topics closely related to high-level programming languages. They might have been contributed to a learned journal; but instead they have been collected in a book. On the whole they deserve to be: the general standard of the papers is distinctly higher than the average, and they are likely to appeal more consistently to a reader interested in high-level programming languages.

But it would be a rash reviewer who would venture to pass comment on all the papers individually; and to do so within the space allotted would be even more foolhardy.

C. A. R. Hoare (Belfast)

Computer Science: Programming in FORTRAN IV with WATFOR WATFIV, 210 pages. (*John Wiley and Sons*, £2·65)

The very successful *Computer Science: A First Course* has now been re-issued in a second, considerably expanded, edition which necessitates a similar revision of all the language supplements. This, the FORTRAN one, is the first that I have seen of the second editions. There are some changes in the set of authors and it is now in phototypescript rather than print, but the overall impression of workmanlike competence remains.

C. M. Reeves (Keele)

Environmental Data Handling, by G. B. Heaslip, 1975; 203 pages. (*John Wiley*, £12·25)

Environmental Data Handling is not a book about computing but about the way transducers and sensors work, how their outputs are generated, coded, transmitted, recorded, analysed and presented. It is simple in its approach with many a homely (in the British sense) line illustration to carry home a point. Remote sensing as practised in Earth resources programmes provides much of the inspiration to the author's approach, befitting his background in the NASA Lunar Program and the Grumman Environmental Data Services. Practical experience is evident in every page and the approach is not analytic. Transducers are only very simply described; the frequency response of an accelerometer, for example, is not discussed. Again the filtering of data at the demodulation stage, an all-important and technically interesting operation, is treated at an elementary level. The book will be a useful reference to those unfamiliar with the subject.

E. B. Dorling (Dorking)