shared by other ADMIN subsystems that they became acceptable.

During the initial testing stages of the inputter over fifty different types of form were input and all the facilities mentioned were found to be useful. Typically a form with about one hundred non-null primitive sections took about one minute to decode. This time is acceptable, since for the first attempt at defining and coding the inputter, correct logic and speed of implementation were considered to be more important than efficiency. A logical system can usually be made efficient, but not necessarily vice versa.

Probably the most important spin-off from the above work is the way in which delimiters were employed. In the field of lexical analysis there would appear to be the possibility of using similar techniques. Finally it should be emphasised that the inputter was designed for large systems where quite possibly thousands of forms are involved. Its flexibility can only be justified in such an environment.

I would like to express my grateful thanks to all my former colleagues for their help, encouragement and patience. In particular, J. Thomas, who suggested the use of a tree structure technique and some of the parameters for the inputter.

## Appendix 1

The purpose of this section is to define some of the terms used in this paper. No claim is made for the universality of the definitions. **Fig. A1** shows a tree structure with the nodes numbered and these are the ones referred to in definitions.

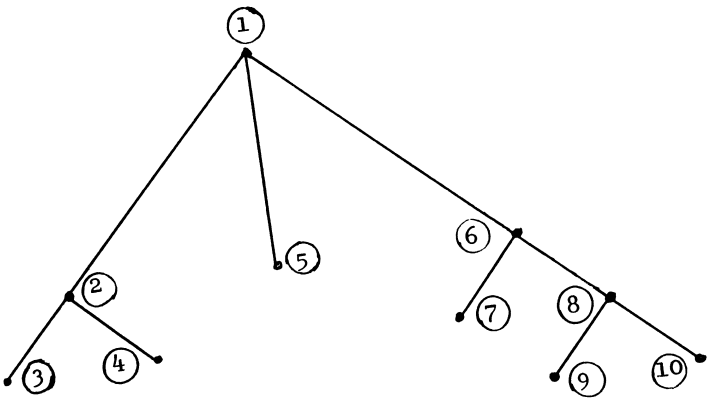


**Fig. A1**

The *head* node is number 1.
*Tip* nodes are 3, 4, 5, 7, 9 and 10.
Nodes 2, 5 and 6 are *sons* of node 1.
Node 8 is the *father* of nodes 9 and 10.
Node 5 is the *elder brother* (or major) of node 6 and the *younger brother* (or minor) of node 2.
Node 2 is the *eldest son* (first son) of node 1.
Node 6 is the *youngest son* (last son) of node 1.

Transversing the tree from the head node the *yo-yo path* through the nodes is that indicated by the ascending numbering.

Each node of a tree has a data stack associated with it. This stack is empty when the node is added to the tree. Subsequently any number of the following items in any order can be placed in this data stack.

(*a*) an integer (held directly)
(*b*) a stack (held as a pointer)
(*c*) a string (held as a pointer)
(*d*) a tree (held as a pointer)

The facility (*d*) is not used by the inputter but is given for completeness. A string in the ADMIN system is not that defined in the ALGOL 60 report. However, ALGOL 60 type strings are often converted into ADMIN type strings but *not* vice-versa.

# Book reviews

*Job Control Language and File Definition*, by Ivan Flores, 1971; 268 pages. (*Prentice-Hall Inc.*, £6·25)

This is a disappointing book. There is certainly a need for a book which explains job control language for the IBM 360 series in a way which reveals the underlying structure of its semantics, rather than concentrating on its rather arbitrary syntax; and this book sets out to do just that. Unfortunately Professor Flores' book falls a long way short of success.

The idea is sound: to introduce the operating system and its components, and file structure; interleaved with chapters which describe the associated JCL, and attempt to link the two together. The execution is disastrous. The one thing above all others which a book of this sort must be is accurate. Next most importantly it must be clear. Finally it should be complete on its chosen level. This book is none of these things.

A first reading revealed no less than one hundred and ten errors of various sorts—mostly printers errors, which in themselves could be fatal to a book on JCL, but also errors of fact. Particularly prone to error are the examples, which frequently contain faults which would cause strings of diagnostics from the assembler or the reader/interpreter, e.g.

'DISP = (OLD,MOD)' and 'OPEN (IND,OUTD)'.

Where clarity is concerned, Professor Flores and I have opposite views. He seems to believe that the clarity of a text improves in direct proportion to the number of different typefaces in use. I do not; particularly when the same word in different typefaces is used to mean different things.

The decision as to how much information should be provided on each topic is a difficult one for any author. Professor Flores says in his preface that his book 'will be a reference source for system programmers' and 'find good application in a senior or graduate course in computer science'. In my opinion he provides far too little information for the former group. As to the latter . . . well, in view of the other shortcomings of this book, it hardly matters.

MARTYN THOMAS (London)

*Computer Control in Process Industries*, by E. I. Lowe and A. E. Hidden, 1971; 279 pages. (*Peter Peregrinus Limited*, £4·00)

This is a project-oriented textbook sponsored by the Council of the Chemical Industries Association and produced by two members of its Instrumentation Advisory Committee, to meet a need which was not satisfied by previous books in this area. Encouraged by such a need, the authors have produced a well-written and easily read textbook, which, whilst primarily aimed at satisfying the needs of the chemical engineer and instrumentation manager, will be of considerable interest to computer professionals working in the field of process control. Whilst those sections dealing with computers—in particular Chapter 3—will be elementary to those involved with computers on a day-to-day basis, the remainder of this book is concerned with two main areas of considerable importance—the interface between the computer system and the process, and the organisation and management of computer projects in this particular field. Several portions of Chapter 10 concerned with the organisation of computer projects closely resemble portions of the BCS Code of Good Practice, although written independently.

Summarising, this is a book which deserves a place on the bookshelves of all concerned with the control of continuous processes, in the light of the wide range of training and reference material which it contains; ranging from central processors, through interfaces such as CAMAC and BS 4421, to management and codes of practice.

F. E. TAYLOR (Manchester)