NEUSER, W. (1977). ExperimentelleUntersuchung ausgeglichener Binärbäumen, Master's Thesis, Karlsruhe, 1977.
OTTMANN, T. and SIX, H.-W. (1976). Eine neue Klasse von ausgeglichenen Binärbäumen, *Angewandte Informatik*, Vol. 18 No. 9, pp. 395-400.
OTTMANN, T., SIX, H.-W. and WOOD, D. (1978a). Right Brother Trees, *ACM*, Vol. 21, pp. 769-776.
OTTMANN, T., SIX, H.-W. and WOOD, D. (1978b). A Survey of New Results in Balanced Search Trees, *Datenstrukturen, Graphen, Algorithmen* (ed., J. Muhlbacher), Carl Hanser Verlag, Munchen, pp. 107-124.
OTTMANN, T., SIX, H.-W. and WOOD, D. (1979). On the Correspondence between AVL Trees and Brother Trees, *Computing*, to appear.
OTTMANN, T. and WOOD, D. (1978). Deletion in One-sided Height-Balanced Search Trees, *International Journal of Computer Mathematics*, Vol. 6, pp. 265-271.
RÄIHÄ, K. J. and ZWEBEN, S. H. (1979). An Optimal Insertion Algorithm for One-sided Height-balanced Binary Search Trees, *CACM*, Vol. 22, pp. 508-512.
YAO, A. C-C. (1978). On Random 2-3 Trees, *Acta Informatica*, Vol. 9, pp. 159-170.
ZWEBEN, S. H. and MCDONALD, M. A. (1978). An Optimal Method for Deletion in One-sided Height-balanced trees, *CACM*, Vol. 21, pp. 441-445.

Book reviews

Structured Programming, by R. C. Linger, H. D. Mills and B. I. Witt, 1979; 402 pages. (*Addison-Wesley*, £14·25)

The term 'structured programming' has long ago lost most of its meaning; for many potential readers it will mean nothing more than acceptance of a simple constraint on patterns of control flow (DO-WHILE and IF-THEN-ELSE), an anathema on GO TO and an adherence to the broad church of top down design. For Linger, Mills and Witt, 'structured programming' means much more than this: it means above all a detailed and careful consideration of control flow, and an insistence that program development should be 'rigorous' (i.e. provably correct) rather than 'heuristic' (i.e. proceeding by trial and error).

After a chapter briefly introducing some basic concepts such as sets, relations, digraphs, state machines, regular expressions and others, a program design language (PDL) is defined. PDL has an outer syntax providing various control flow constructs, jobs, procedures and modules, and some structuring of data. Attention is then concentrated, in the central portion of the book, on methodical analysis of control flow, including the rewriting of 'unstructured' into 'structured' programs by creating, where necessary, new variables to represent the value of the text pointer. Chapters on program reading and correctness proofs then follow, leading into the last part of the book. This last part is concerned with designing and writing structured programs and is based on a number of small examples, each illustrated by comparing a faulty, intuitive, heuristic design with the recommended form of rigorous stepwise refinement. The exposition in this part is not, perhaps, entirely successful. The main examples (long division, making change, tic tac toe) invite discussion of algorithmic aspects which lie far from most programmers' daily concerns. The classic IBM pollution reporting problem is a more typical task, but is handled rather less satisfactorily and with less confidence.

In some ways this is an old-fashioned book. The underlying idea of a program is that of a hierarchy of procedures; there is no mention of processes or of parallelism. The notion of a 'job' in the PDL is a very primitive, implementation based notion derived essentially from OS/360. But these defects are less than the book's virtues. Above all, it conveys clearly the idea that program development should be a rigorous activity relying on sound theoretical foundations: that is the book's chief message, and it deserves to be heard.

M. A. JACKSON (London)

Structured Systems Analysis: Tools and Techniques, by Chris Gane and Trish Sarson, 1979; 241 pages. (*Prentice-Hall*, £12·05)

The tools and techniques recommended and briefly described are these: data flow diagrams (DFDs), hierarchically decomposed; data dictionary; decision tables and decision trees; narrative procedural description in 'structured English' or 'tight English'; data base normalisation into third normal form; and program design according to Constantine's methods, as expounded by Myers, Yourdon and others. As a compendium of some well loved ideas the book passes muster. Most of the ideas are described very superficially, but there are references to more substantial works: Date's book on data base systems, Pollack or London on decision tables, Yourdon and Constantine on structured design. A reader interested in understanding and using these tools would need to supplement this book by studying many of the references given. This is no criticism: much of the value of the book lies in the conjunction of the various tools, which is claimed to provide at least the basis of a method.

The chief novelties are the 'structured' or 'tight' English, and the data flow diagrams. The former is no more than a slight relaxation of a typical pseudocode to make it more acceptable to a lay customer; 'structured' English adds a small dose of syntactic sugar to the pseudocode; 'tight' English adds a rather larger dose of syntactic salt which seems, to this reviewer at least, to spoil the dish. There is more substance in the data flow diagramming method. The general idea is that the function of the system is decomposed into lower level functions which communicate by sequential data flows and by updating globally accessible data stores. A function is an 'order to a dumb clerk', capable of 'being carried out in a simple clerical circumstance in 5-30 minutes'. Decomposition proceeds hierarchically, with some obvious rules (not always observed) relating the successive levels. Much attention is paid to the mechanics of the representation (such as the avoidance of crossing data flows). The claim is made that the DFD technique is 'logical' rather than 'physical', avoiding premature implementation decisions, but this claim is not well founded: the partitioning of the system established in the data flow diagrams is retained in the final implementation, and there is no discussion of how the system might be repartitioned during the implementation stage.

Too much of the book is taken up by vague generalisations, and there is little to exert the reader's intellect. But the book is not without content, and will certainly appeal to some. Enthusiasts could usefully compare it with de Marco's book on the same subject and from the same stable.* M. A. JACKSON (London)

Reference

TOM DE MARCO, (1979). *Structured Analysis and System Specification* Prentice-Hall, £16·25

Speech Communication with Computers, edited by Leonard Bolc, 1979; 206 pages. (*Carl Hanser Verlag* and *Macmillan*, £10·00)

The mechanical analysis of speech was recognised as a challenging problem long before the advent of computers. When computers became available the recognition of speech by computer quickly became an established problem, and its imminent solution is announced with great regularity. For instance it is reported that in 1965 it was announced that Dartmouth College would have a recogniser 'for any language within two years'.

The present volume is a collection of six articles describing the work in progress at laboratories in the United States and Europe in the mid-1970's. An article from Carnegie-Mellon University describes a comprehensive project, whereas others deal with more specialised topics, one specialised topic being the identification of a speaker, discussed by P. Jesorsky. This volume is intended to be the first in a survey series on natural communication with computers; it is a useful reference and worthy of a place in those libraries where student projects start.

J. J. FLORENTIN (London)