**BOOK REVIEWS**


This book presents programming concepts and various programming paradigms that may form part of a tool-kit for the design of algorithms and abstract data types. The concepts are illustrated by a large range of examples written in Pascal, with extensions where needed. The book is divided into two parts. The first part covers the topics of procedural abstraction; backtracking; efficiency of algorithms and sorting. The second part concentrates on the topic of data abstraction. Here, it demonstrates how to define a data structure in an ideal programming language, then discusses how one can simulate such an implementation in Pascal. It goes on to examine the implementation of various structures such as sequences, sets and graphs. It has a good range of exercises at the end of each chapter, though at least some sample solutions would have been useful. It also presents a nice variety of example algorithms. There is a very complete chapter on sorting which presents nine different algorithms, with a discussion of their relative efficiencies. This reinforces the preceding chapter on efficiency.

I feel that this is a useful book, especially with its emphasis on data structures. However, I do have a number of criticisms about both the way the subject matter is presented and aspects of the book which contribute to its value as a textbook. Though the book purports to be about design, there is not a lot of discussion, especially in the early chapters, about how one tackles this very difficult problem. Instead, algorithms are presented with their specifications, and the skill of learning their design is assumed to come from reading and understanding this range of algorithms, which exemplify different programming paradigms. Although I accept such exposure is a necessary part of the learning process, I feel that a little more discussion about the issues in their construction would have been useful.

The introduction lays the ground rule that the specification is the guideline for writing procedures and functions. Thus, the authors very laudably insist that a procedure or function must always be accompanied by a description of its effect in terms of pre- and postconditions (though this requirement does not extend to whole
programs). What the book does not do is discuss how these pre- and postconditions are constructed, or indeed, how one progresses from the pre- and postconditions to an algorithm. The use of pre- and postconditions and invariants seems almost incidental to the process and these annotations to program fragments are often left out if that fragment is repeated in the text. Such omissions belie the claim that the annotations are integral and essential to the program, undermining this principle, especially as when they are included, there is often no discussion about the specification. Similarly, in the chapter on backtracking, pre- and postconditions are omitted after the first example procedure, because the rest of the procedures are "analogous". In my opinion, the introduction of problems and concepts just by examples is sometimes too informal. For example, there is no formal definition of a sequence (or a set), and the problem solution introducing this key concept is just a program fragment, with no headings, no specification, no invariants, and so on.

Looking at the technical aspects of writing any textbook, I found this one wanting. The index is extremely skimpy. There are only about 150 entries for a book of more than 300 pages, which is inadequate. Another deficiency is the lack of references—there are only two to other works, and these are embedded in the text, so are not easy to find. A wide range of topics are touched upon, and in an introductory book it is impossible to do everything justice. However, because of the superficial nature of some of the material (for example, eight pages on complexity), it would have been very useful to have references to books which cover the material in more depth. I would also have expected references to other languages such as Modula-2 and Ada, which do contain the features which are presented in this book as the authors' own language extensions to Pascal. Both explicit and implicit (axiomatic) specifications for abstract data types are presented. However, there is no clue as to how the specifications were constructed and, again, no references to further work. Difficulties such as completeness, consistency and readability are discussed for axiomatic specifications, but there is no indication of how to go about solving these problems or references to works which would go further into these issues. Similarly, there are no references in the section on verifying the correctness of an implementation, which is a page in length.

Another problem is that terms are sometimes used before they are defined and some quite important terms, such as "statement" and "domain", are not defined at all. The assumptions about the reader's knowledge of Pascal are not always consistent. Sometimes concepts are explained, and sometimes not. This is especially confusing for a Pascal novice when the notation used is not standard *Pascal*, but the authors' own extension. Sometimes it is difficult to distinguish between what is standard and what is an extension, as in the description of conformant arrays which begins with the phrase "we shall allow ...".

In summary, this book has tackled an important topic, illustrating it with a wide selection of examples. However, it could have been much improved by explaining the examples more and being more formal and precise in the introduction of concepts. Having a clear and consistent policy about the knowledge of the targeted reader
and attention to details such as references, the index, and defining terms before use would have improved it immeasurably.

Vicky Bush

Department of Computer Science
University of Manchester
Manchester, United Kingdom


This volume is a reprint of an issue of Daedalus (volume 117, number 1) from the Proceedings of the American Academy of Arts and Sciences. Its contributors number some well-known authors, including Seymour Papert, Hubert and Stuart Dreyfus, Pamela McCorduck, David Waltz, Tomaso Poggio, Sherry Turkle, Hilary Putnam, Daniel Dennett and John McCarthy.

What we have here is a collection of fourteen twenty-odd page essays, all of a reflective or retrospective character, on various aspects of artificial intelligence. The Dreyfuses, Putnam and Dennett all provide us with professionally produced pieces of philosophy holding little in the way of novelty or excitement. John McCarthy has a short and uninspired paper on AI and logic: not, I think, vintage stuff. Pamela McCorduck, in what is perhaps the most enjoyable article in the volume, engages us with a description of a painting program called AARON. Anya Hurlbert and Tomaso Poggio write about computer vision. There is a clutch of papers on connectionism. And so it goes on. Broadly unobjectionable material, if rather uninspiring.

I cannot imagine who would want to buy this book. The positions taken in the papers are not, by-and-large, novel, and no attempt has been made by the editor to summarize the papers or to diagnose trends. The student seeking an introduction to cognitive science would do better with P.N. Johnson-Laird's The Computer and the Mind (Fontana, 1988). The layman interested in the theoretical foundations of artificial intelligence should invest in Zenon Pylyshyn's Computation and Cognition (MIT Press, 1989) or Roger Penrose's The Emperor's New Mind (Oxford, 1989). And the professional seeking to keep abreast of developments, and au fait with what his colleagues are doing can simply skim through the appropriate issue of Daedalus, which will be in his university library anyway. But if you must buy it, you will find it well-produced, with few typographical errors, and a bibliography following every paper.

Ian Pratt

Department of Computer Science
University of Manchester
Manchester, United Kingdom