*Concepts in programming languages*
John C Mitchell
Cambridge University Press, 2003
ISBN 0-521-78098-5

When I learnt to program, there were only two languages that were worth learning, Algol 60 and assembler. 'Programming languages' was once an easy course to teach, but as more languages were introduced it became harder and harder, and most books for undergraduates turned from textbooks into encyclopedias, trying to cover all known languages. Somewhere along the line the idea that it was worth thinking about different styles of programming languages got lost. Today most students are now exposed to only a few languages and they gain little appreciation of, on the one hand, the breadth of programming language paradigms and design trade offs, and on the other hand the central concepts that underpin the apparent diversity.

That can change with Mitchell's *Concepts in Programming Languages*, for we now have a thoughtful book that covers the essential concepts such as types, memory management, objects and functions, without getting overwhelmed by the diversity of available languages. It is an excellent book on programming languages, and one that lecturers would enjoy using and students would gain much from having and reading. The book reviews the main types of programming language, imperative, functional, object oriented, concurrent and logic in its 500 pages, at just the right level of detail so that undergraduates can appreciate the differences without being bogged down in trivial detail of the example languages. The book concentrates for concrete examples on just a few worthwhile languages, such as LISP, ML, Java and Prolog. The book is very pleasant to read, too, with regular vignettes about historical players such as John McCarthy, Tony Hoare and the usual suspects.

The exercises are excellent, and range from basic exercises to ones that, if they don't inspire new careers, ought at least generate fine projects.

I am sure this book will be a success and we will soon see further editions. I hope that the printer's gremlins are removed from the first edition. A problem for an author of a book like this is to know when to stop; everybody John Mitchell asked must have had a different language or feature that needed adding. He did well to cut down the encyclopedic vision into a coherent course. But there are omissions that lecturers will have to supply themselves: there is no outline history, there is not even a brief discussion of bad languages (like basic) and why they are bad; there is no discussion of scripting languages (like Perl, Python, Javascript or tcl/tk). The merits of tiny languages (e.g., Forth, or its incarnation as PostScript) are not mentioned, and there is little insightful on programming language design as such. The reader will not learn about trade-offs in compiling and interpreting, or design principles such as the principle of correspondence (even though denotational semantics is covered briefly). Like most programming language books for undergraduates there is no discussion of the psychology of programming: which is a surprising omission given that a lot of things are known about it, and that the whole point of programming languages is to link human thought to computers. But far better to leave your reader wanting more than to overwhelm them!

Indeed it is an inspiring book. Having been inspired, then, some readers would appreciate a further reading section: there are many excellent, though more specialised books, that motivated students should go on to read.

These are minor criticisms; the book is just the right length to be used in a self-contained course; with good students it could be used early on in their degrees so that they learn as soon as possible that there is more to computer science and programming languages than they would ever have realised.

Harold Thimbleby is Gresham Professor of Geometry, and Director of UCLIC, the University College London Interaction Centre.