

The Tomes of Delphi Algorithms and Data Structures

book by Julian Bucknall

pub Wordware RRP £48.50, 524 pages including index, plus CD and supporting website

Publishers' blurb

Delphi developer Julian Bucknall provides fellow developers a comprehensive overview of using algorithms and data structures from a practical perspective. Bucknall begins with a discussion of algorithm performance, and provides comprehensive coverage of such topics as arrays, linked lists, and binary trees. The book focuses on search algorithms—such as sequential and binary search—and sort algorithms—including bubble, insertion, Shell sort, quicksort, merge sort, and heapsort—along with techniques for optimization. Additionally, the author presents hashing and hash tables, priority queues, state machines and regular expressions, and data compression techniques such as Huffman and LZ77.

Author's blurb

Although there are numerous books on algorithms in bookstores, few of them go beyond the standard Computer Science 101 course and approach algorithms from a practical perspective. Code is shown, usually merely to illustrate an algorithmic point, and generally no consideration is given to real-life, drop-in-and-use application of the technique in question. Even worse from the perspective of the commercial programmer, many are textbooks and hence some of the more interesting viewpoints are left as exercises for the reader, with little or no answers. Delphi programmers are even more isolated; more often than not they are obliged to use a book with code in C or C++ and translate. Even worse, they can look over at the C++ programmer with his Standard Template Library.

Tomes of Delphi: Algorithms and Data Structures (DADS) aims to bridge the gap for Delphi or Kylix programmers. Algorithms are introduced purely from a Delphi perspective, using the Object Pascal language. Delphi/Kylix standard classes, such as TList and the TStream family, are extensively used. Emphasis is given to drop-in-and-use units and classes.

The book is aimed at medium to advanced level programmers, those who would be looking at the standard algorithm texts and converting algorithms from them into Delphi. Little or no provision is made towards beginner programmers. The book and code is applicable to all versions of Delphi and to Kylix; it was not written just for the latest compiler. Hence, the book will not 'age' as quickly as many other Delphi books—the code in it will still work with Delphi 6 and later.

The DADS book is divided into chapters tracing the standard algorithms from the simple to the more complex. Information and code introduced in earlier chapters is used extensively in later chapters; re-use of code is emphasized. The book does not delve too much into mathematics, especially with regard to efficiency calculations; the reader is referred to other textbooks for such coverage.



Dr Bob says

This is a book that I've been waiting for for a long time. In fact, it may have been over a year since I first saw it mentioned. But like many good things, quality takes time, and it was certainly worth waiting for. This book will not be outdated anytime soon, since it does not depend on a specific version of Delphi (a good thing the publisher also noted that by not placing a Delphi version number in the title).

I received the book at the Borland Conference in Long Beach, a few days before my session on Delphi Efficiency which was very convenient, since Algorithms and Data Structures are, of course, an important part of performance optimisation. In fact, I managed to read the first few chapters before my session, and was able to use one of the little gems right away in my session. I'll share it with you here, since it's one of those things that you never realise until you bump into it (head first).

I love the built-in Pos function in Delphi, which is designed to find the location of a substring in a string, like Pos('ox', 'the quick brown fox jumps etc.'). However, personally, I also use Pos a lot for finding the location of a single character in a string, for example in Pos('@', Str) or Pos('#', Str). And that's not what Pos is meant to do, since its algorithm is designed to work on substrings and not characters. Behind the scenes, the character is converted to a LongString and then this substring (of length 1) is searched. This takes more time than necessary. In fact, a single loop to walk through a string in search of the specific character is about five times faster, as Julian pointed out in his book. I found this to be an eye-opener, as I've used Pos to search for single characters in many places in my applications. And I never even considered them an overhead (even if they are not exactly a performance bottleneck, sometimes every little bit counts).

As you continue to read this book, it gets deeper into the more complex algorithms and data structures, but this only makes it more valuable. I haven't finished it yet, so I will write a more detailed review about the entire book, but I'm very pleased with what I've read so far. Essential reading for any serious Delphi "engine" developer, I would say (and every application contains at least a little engine).

Table of contents:

Introduction
Chapter 1: What is an algorithm?
Chapter 2: Arrays
Chapter 3: Linked Lists, Stacks and Queues
Chapter 4: Searching
Chapter 5: Sorting
Chapter 6: Randomized Algorithms
Chapter 7: Hashing and Hash Tables
Chapter 8: Binary Trees
Chapter 9: Priority Queues and Heapsort
Chapter 10: State Machines and Regular Expressions
Chapter 11: Data Compression
Chapter 12: Advanced Topics
Appendices: References and Epilog