

Three Books on the C++ Standard Template Library

Reviewed for SIGOOT by Conrad Weisert, March, 1996

Mark Nelson: *C++ Programmer's Guide to the Standard Template Library*, IDG Books, 1995, ISBN 1-56884-314-3, \$50.00.

I enthusiastically recommend this comprehensive guide. I particularly liked the author's easy-going style. He addresses us as respected colleagues with whom he wants to share some exciting information. Unlike some writers, he doesn't oversell the STL or try to defend its most ill-advised features and peculiar terminology.

The 21 chapters are divided into three main parts:

- I. Introducing the STL
- II. The Essentials: Containers, Iterators, Algorithms, Functions
- III. The Public Interface: Reference Information

If you're a C++ programmer, the first two parts are *must* reading whether or not you plan to use the STL soon. The third part is too detailed and repetitive for the casual reader, but undoubtedly a helpful reference while you're developing a program.

The index is comprehensive, and definitions are provided for important terms, with the exception of algorithm *complexity* and $O(n)$ notation, concepts well known to Computer Science majors but not to every practicing programmer.

The publisher has applied to the back cover the amazing advice that the "Reader Level" is "beginning to advanced". Beginning what?

This book, like the others, will be utterly incomprehensible to the experienced application programmer who lacks a grasp of C++ classes, and will be heavy going even for C++ programmers who haven't worked much with containers beyond simple arrays.

Highly recommended.

Graham Glass & Brett Schuchert: *The STL Primer*, Prentice Hall, 1996, ISBN 0-13-454976-7, \$36.00.

This 310-page book contains about 75 pages of information. Apparently the authors' main goal was to get something onto bookstore shelves ahead of the competition.

The book makes a strong beginning. The first third gives an overview of the main features of the STL -- container classes, iterators, and function objects -- with some discussion of background and motivation and an attempt to classify the components.

The rest consists of alphabetically ordered *specifications* of each class and independent (algorithm) function in the STL. This material is painfully repetitive. The authors give rarely-used facilities equal prominence with those that should be part of every programmer's repertoire, and offer little guidance in choosing between alternative ways of accomplishing a result.

Both parts contain usage examples, but contrived ones with no connection to any real-world programming problem. Code samples are unnecessarily repetitive, space consuming, and hard to read. They take the form of complete `main` functions, with results displayed on the standard output stream.

Typos, while fairly obvious, reflect careless editing and undermine the reader's confidence in the correctness of other details. For example:

p. 34: "Some algorithms require more powerful iterators than others."

p. 69: "In the worst cast, . . ."

The index is incomplete, and crucial definitions are omitted or circular. For example, *heap* is used both in the sense of the data structure and in the specialized C sense of the pool of free storage. You won't find either under H, but have to look under M for `make_heap`, which offers no explanation to the reader who doesn't already know what a heap is.

Not recommended

David R. Musser & Atul Sahni: *STL Tutorial and Reference Guide*, Addison-Wesley, 1996, ISBN 0-201-63398-1, \$38.50.

Having reviewed three new books on the STL, I'm frustrated by the lack of a practical application point of view. Part II of this book has a promising title: "Putting it All Together -- Example programs." However, the best the authors come up with is a set of programs to generate anagrams, hardly of interest to mainstream programmers. Application developers who are dubious about the usefulness of C++¹, not to mention the STL, will find little here to reassure them.

With that exception, this is a good book. The explanations are lucid and the tone is friendly. Part I ("A Tutorial Introduction to the STL") takes a spiral approach through 11 chapters. If you're comfortable with C++ templates and with measures of algorithm efficiency, you'll find it very readable. Part III is a reference guide. The few typos are minor and obvious.

I'm still looking for a treatment of the STL that addresses two crucial topics:

How can application developers exploit the STL in real-life applications, especially business applications? Why should they care about the STL? How about some realistic examples of the use of at least a few of the container classes?

What sort of standards are appropriate for the way we use the STL? How can an organization or a project team exploit the STL's strengths while minimizing the impact of its shortcomings?

There are now a couple dozen books on the STL that I haven't seen, so there may well be some that meet this need. Let me know if you find one.

Recommended.

¹ See my article "Making C++ Practical for Business Applications", ACM *SIGPLAN Notices*, December, 1995, volume 30, no. 12, pp. 4-8.