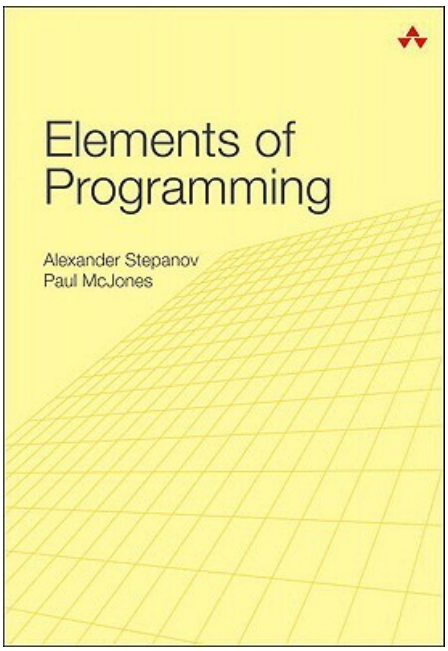

Elements of Programming

Alexander Stepanov , Paul McJones

Elements of Programming

Alexander Stepanov , Paul McJones

Elements of Programming Alexander Stepanov , Paul McJones
"Ask a mechanical, structural, or electrical engineer how far they would get without a heavy reliance on a firm mathematical foundation, and they will tell you, 'not far.' Yet so-called software engineers often practice their art with little or no idea of the mathematical underpinnings of what they are doing. And then we wonder why software is notorious for being delivered late and full of bugs, while other engineers routinely deliver finished bridges, automobiles, electrical appliances, etc., on time and with only minor defects. This book sets out to redress this imbalance. Members of my advanced development team at Adobe who took the course based on the same material all benefited greatly from the time invested. It may appear as a highly technical text intended only for computer scientists, but it should be required reading for all practicing software engineers."
--Martin Newell, Adobe Fellow

"The book contains some of the most beautiful code I have ever seen."
--Bjarne Stroustrup, Designer of C++

"I am happy to see the content of Alex's course, the development and teaching of which I strongly supported as the CTO of Silicon Graphics, now available to all programmers in this elegant little book."
--Forest Baskett, General Partner, New Enterprise Associates

"Paul's patience and architectural experience helped to organize Alex's mathematical approach into a tightly-structured edifice--an impressive feat!"
--Robert W. Taylor, Founder of Xerox PARC CSL and DEC Systems Research Center

Elements of Programming provides a different understanding of programming than is presented elsewhere. Its major premise is that practical programming, like other areas of science and engineering, must be based on a solid mathematical foundation. The book shows that algorithms implemented in a real programming language, such as C++, can operate in the most general mathematical setting. For example, the fast exponentiation algorithm is defined to work with any associative operation. Using abstract algorithms leads to efficient, reliable, secure, and economical software.

This is not an easy book. Nor is it a compilation of tips and tricks for incremental improvements in your programming skills. The book's value is more fundamental and, ultimately, more critical for insight into programming. To benefit fully, you will need to work through it from beginning to end, reading the code, proving the lemmas, and doing the exercises. When finished, you will see how the application of the deductive method to your programs assures that your system's software components will work together and behave as they must.

The book presents a number of algorithms and requirements for types on which they are defined. The code for these descriptions--also available on the Web--is written in a small subset of C++ meant to be accessible to any experienced programmer. This subset is defined in a special language appendix coauthored by Sean Parent and Bjarne Stroustrup.

Whether you are a software developer, or any other professional for whom programming is an important activity, or a committed student, you will come to understand what the book's experienced authors have been

teaching and demonstrating for years--that mathematics is good for programming, and that theory is good for practice.

Elements of Programming Details

Date	:	Published June 1st 2009 by Addison-Wesley Professional (first published 2009)
ISBN	:	9780321635372
Author	:	Alexander Stepanov , Paul McJones
Format	:	Hardcover 262 pages
Genre	:	Computer Science, Programming, Science, Computers, Coding, Technology, Mathematics

From Reader Review Elements of Programming for online ebook

Nick Black says

did i never review this??! i thought for sure i had! oh man this is the book god read before he coded the universe. sloooow going, but don't be daunted.

Amazon 2009-07-07. I'm looking forward to this being the most exciting thing I've read in months, maybe years.

Maxim Chetru?ca says

A very complex book. I think its description promises more than it delivers. The language used by the author is pretty difficult to understand. The path from chapter to chapter and through a chapter is not always clear, you don't understand where the author is going with it. I definitively did not understand a lot.

Artem Komisarenko says

????? ?????, ??? ???? ?????? ???????, ??? ?????????. ????????? ?????. ???? ?????? ?????? ?? ?????????? ???????, ???? ?? ? ??? ? ?????????.

Koppektop says

Very hard-to-read book for me.

Christian Kotz says

excellent, must read for computer scientist. Very systematic and mathematical. It is from the designer of the C++ Standard Library, which shows its relevance for real world programming, despite its mathematical character.

Maxim Razin says

The name is misleading. It's more like a theoretical background beyond STL

Chad Brewbaker says

Must read on generic programming.

Gary Lang says

You might enjoy the combination of math theory and applying it to practical coding. If you do, then you should love this book. Usually stuff like this doesn't have as much application to real life (see Z Notation).

I had forgotten how elegant C++ templates could be; reading this brought it all back.

Mark says

Brilliant! A synthesis of practical programming and rigorous mathematics. No head-in-the-clouds formalisms like the lambda calculus or Turing machines here, this is the thinking that directly inspired C++ templates and the STL. A word of advice - take a course in abstract algebra before reading this, and it may make much more sense.

Chris Sharpe says

This one was a bit of a slog. Because it tries to be both a Mathematics and a Computer Science book, it skims a little over both, and I can't honestly recommend it unless you have a little background in both areas (which seems likely, if you are considering this). For instance, it uses the taxonomy of algebraic structures (monoids, groups, rings), with requirements on operations (associativity, commutativity, existence of identity, existence of inverses) to illustrate constructing a taxonomy of concepts that apply to data structures, but it doesn't dedicate much explanation to the mathematical structures, so you can only really follow if you already know them.

In the other direction, it is also not a book from which to learn algorithms or C++ template metaprogramming techniques, at least from scratch, although you will learn something about those. If you already have some idea about how gcd, rotate, and partition work, then you should be reasonably well set to learn a lot more about possible implementation variations, with their respective asymptotic complexities and requirements on the types on which they operate.

Some of the above might seem negative, but I simply want to point out what the book is *not*. What it *is*, is an excellent academic guide to carefully designing simple components, and building complex behaviour out of those, along with a catalogue of some useful such components. I learnt a lot about fundamentals such as partitioning, sorting, vectors, and deques. One key component that Stepanov has talked about a lot, but that didn't seem to get quite as much stress here, is his associative binary counter, absolutely essential for efficient iterative reduction. I found the description in this book finally made this idea click, even though I had seen it several times before.