Wednesday, December 10, 2008

# Books Programmers Don't Really Read

Mark Twain once said that a classic novel is one that many people want to have read, but few want to take the time to actually read. The same could be said of "classic" programming books.

Periodically over on Stack Overflow (and in many other programming forums) the question comes up about what books are good for programmers to read. The question has been asked and answered several times, in several different ways. The same group of books always seems to rise to the top, so it's worth it to take a look at these books to see what everyone is talking about.

## Books Most Programmers Have Actually Read

1. Code Complete
2. The Pragmatic Programmer
3. C Programming Language (2nd Edition)
4. Refactoring: Improving the Design of Existing Code
5. The Mythical Man-Month
6. Code: The Hidden Language of Computer Hardware and Software
7. Head First Design Patterns
8. Programming Pearls
9. Effective Java (2nd Edition)
   or Effective C++
10. Test Driven Development: By Example

I've read all of these books myself, so I have no difficulty believing that many moderately competent programmers have read them as well. If you're interested enough in programming that you're reading this blog, you've probably read most, if not all of the books in this list, so I won't spend time reviewing each one individually. I'll just say that each of the books on the list in an exceptional book on its respective topic. There's a good reason that many software developers who are interested in improving their skills read these books.

Among the most commonly recommended programming books there is another group that deserves special consideration. I call the next list "Books Programmers Claim to Have Read". This isn't to say that *no one* who recommends these books has actually read them. I just have reason to suspect that a lot more people *claim* to have read the following books than have *actually* read them. Here's the list.

## Books Programmers Claim to Have Read

1. Introduction to Algorithms (CLRS)
   This book may have the most misleading title of any programming book ever published. It's widely used at many universities, usually in graduate level algorithms courses. As a result, any programmer who has taken an algorithms course at university probably owns a copy of CLRS.

However, unless you have at least a Masters degree in Computer Science (and in Algorithms specifically), I doubt you've read more than a few selected chapters from *Introduction to Algorithms*.

The title is misleading because the word "Introduction" leads one to believe that the book is a good choice for beginning programmers. It isn't. The book is as comprehensive a guide to algorithms as you are likely to find anywhere. Please stop recommending it to beginners.

2. Compilers: Principles, Techniques, and Tools (the Dragon Book).
   The Dragon Book covers everything you need to know to write a compiler. It covers lexical analysis, syntax analysis, type checking, code optimization, and many other advanced topics. Please stop recommending it to beginning programers who need to parse a simple string that contains a mathematical formula, or HTML. Unless you actually need to implement a working compiler (or interpreter), you probably don't need to bring the entire force of the Dragon to bear. Recommending it to someone who has a simple text parsing problem proves you haven't read it.

3. The Art of Computer Programming (TAOCP)
   I often hear TAOCP described as the series of programming books "that every programmer should read." I think this is simply untrue. Before I'm burned at the stake for blasphemy, allow me to explain. TAOCP was not written to be read from cover to cover. It's a reference set. It looks impressive (it *is* impressive) sitting on your shelf, but it would take several years to read it through with any kind of retention rate at all.

   That's not to say that it's not worthwhile to have a copy of TAOCP handy *as a reference*. I've used my set several times when I was stuck and couldn't find help anywhere else. But TAOCP is always my reference of last resort. It's very dense and academic, and the examples are all in assembly language. On the positive side, if you're looking for the solution to a problem in TAOCP (and the appropriate volume has been published) and you can't find it, the solution probably doesn't exist. It's *extremely* comprehensive over the topic areas that it covers.

4. Design Patterns: Elements of Reusable Object-Oriented Software (Gang of Four)
   Design Patterns is the only book on this list I've personally read from cover to cover, and as a result I had a hard time deciding which list it belongs on. It's on this list not because I think that few people have read this book. Many have read it, it's just that a lot more people *claim* to have read it than have *actually* read it.

   The problem with *Design Patterns* is that much of the information in the book (but not enough of it) is accessible elsewhere. That makes it easy for beginners to read about a few patterns on Wikipedia, then claim in a job interview that they've read the book. This is why Singleton is the new global variable. If more people took the time to read the original Gang of Four, you'd see fewer people trying to cram 17 patterns into a logging framework. The very best part of the GoF book is the section in each chapter that explains when it is appropriate to use a pattern. This wisdom is sadly missing from many of the other sources of design pattern lore.

5. The C++ Programming Language

> This book is more of a language reference than a programming guide. There's certainly plenty of evidence that someone has read this book, since otherwise we wouldn't have so many C++ compilers to choose from.
>
> Beginning programmers (or even experts in other languages) who want to learn C++, though, should not be directed to *The C++ Programming Language*. Tell them to read C++ Primer instead.

As I said before, I know there are a few of you who have actually read these books. This post isn't intended for you, it's intended for the multitudes who are trying to appear smarter by pretending to have read them. **Please stop recommending books to others that you haven't read yourself.** It's counter productive, as often there is a better book (more focused on a specific problem domain, easier to understand, geared more toward a specific programming language or programming skill level) that someone more knowledgeable could recommend. Besides that, you may end up embarrassing yourself when someone who has actually read TAOCP decides to give you a MMIX pop quiz (if you don't know what I'm talking about, then **this means you**).

Posted by Bill the Lizard on 12/10/2008
Labels: books, programming