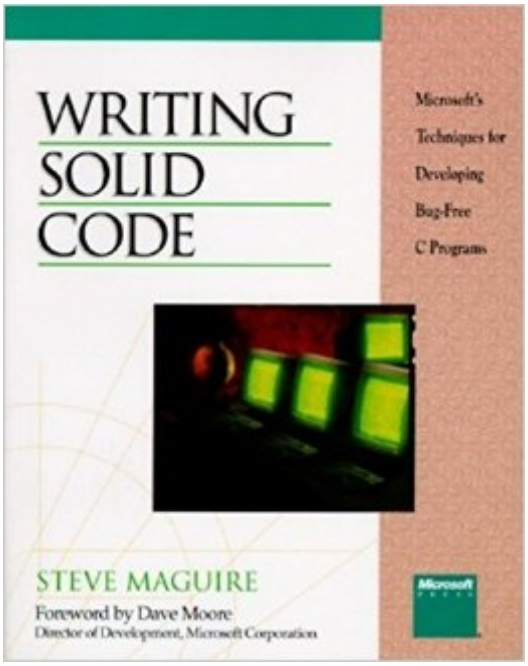
Writing Solid Code (Microsoft Programming Series)



Synopsis

For professional intermediates to advanced C programmers who develop software, here is a focused and practical book based on writing bug-free programs in C. Includes practical solutions to detect mistakes before they become a costly problem.

Book Information

Paperback: 256 pages

Publisher: Microsoft Press; 1 edition (January 1, 1993)

Language: English

ISBN-10: 1556155514

ISBN-13: 978-1556155512

Product Dimensions: 7.4 x 0.8 x 9.1 inches

Shipping Weight: 1.4 pounds

Average Customer Review: 3.8 out of 5 starsÂ Â See all reviewsÂ (49 customer reviews)

Best Sellers Rank: #124,628 in Books (See Top 100 in Books) #4 inÂ Books > Computers & Technology > Programming > Languages & Tools > Debugging #78 inÂ Books > Computers & Technology > Programming > Microsoft Programming > C & C++ Windows Programming #352 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Software Development

Customer Reviews

The negative reviews I've read tend to fall into two categories: 1) Anti-Microsoft Bashing and 2) Nitpicking.This book isn't a recipe book, and it's a bit dated, having been written during the days of DOS and the first Macintosh, but the underlying themes and general advice are still valid:- Enable compiler warnings and pay attention to them.- Use assertions to validate your assumptions.- Don't quietly ignore error conditions or invalid input.- For a complicated, critical algorithm, consider using a second algorithm to validate the first. (e.g. validate binary search with a linear search).- Don't write multi-purpose functions such as realloc (it can grow memory, shrink memory, free memory, or allocate new memory -- it does it all).- Check boundary conditions carefully.- Avoid risky language idioms.- Write code for the "average" programmer. Don't make the "average" programmer reach for a reference manual to understand your code.- Fix bugs now, not later.- There are no free features; don't allow needless flexibility (like realloc).- Ultimately the developer is responsible for finding bugs; he shouldn't write sloppy code and hope that QA will find all his bugs.

I was shocked to see this book get some negative reviews. Those that blasted Microsoft missed the point. This book provides invaluable advice in a quick read. For example, "If you have to look it up, the code is not obvious," or, "If you find yourself designing a function so that it returns an error, stop and ask yourself whether there's any way you can redfined the function to eliminate the error condition." This is the book that convinced me to single-step all my code. The heuristics on proactive bug prevention, which are summarized in the appendix by the way, will save your team time and let you move on to adding features rather than fighting fires.

This book in many ways changed the way I approached programming. In particular, the perspective that it took towards designing and implementing programs in a way such that bugs are automatically detected (preferably by the compiler, or by other tools) has proved invaluable to me in my later experience as a software engineer. Other advice (use some form of 'assert' heavily, deliberately write 'brittle' code that will break loudly as soon as the slightest thing goes wrong, so errors aren't hidden) was similarly insightful, if occasionally a bit counterintuitive.On the other hand, the book's source language (the C programming language, as used by Microsoft tools) is increasingly outdated. This might prevent some people from appreciating the finer insights of this book. Also, the author recommends use of Hungarian notation, which I despise in a strongly typed language. (My recommendation: use PC-Lint, with strong types. Both more powerful, and more readable.) Still, the overall philosophy of the book is very powerful, and the concepts and techniques that it discusses can easily be applied to other programming languages and other software development efforts. I was writing in C++ when I read the book, and I currently program in Java, but I still apply the lessons I learned from this book.This book is certainly not the ONLY way to write solid code (and I have found improvements on a number of techniques discussed in the book), but it does sketch out a fairly workable process which if followed would produce a quality product. Most importantly, the PHILOSOPHY that the author uses to approach writing bug-resistant programs is quite powerful.Note that this is a book on software implementation, not software design (except in the small scale). Large-scale design is a whole separate issue which this book really does not address much.

This is one of the best books on programming ever written. Maguire concentrates on how to eradicate bugs early in the cycle. At first what he says seems so simple. But these ideas are so powerful, and so well presented, that I'd give a copy to every programmer I know if I could afford to.Don't worry that his examples are in C. The ideas transcend the source language. If half of the

programmers followed half of his suggestions half of the time, the software industry would undergo a revolution in quality. There is no silver bullet, but these suggestions are so practical. It’s just a matter of adopting a few good habbits.This book will be a classic. Scratch that. It *is* a classic.If you’re a programmer, it belongs on your shelf beside _Programming Pearls_, _Code Complete_, and Knuth.

I’m posting this mostly to counteract the buffoons who obviously gave this book a poor rating only because it was published by Microsoft Press. That’s like criticizing Core Java 2 because you don’t like Sun.MS Press actually has an enviable stable of writers, and publishes many fine books. Ever heard of Steve McConnell? Jeff Prosise? Charles Petzold?Anyway, this book is a very good source of advice on preventing C bugs on any platform. At one company where I worked, the VP of Engineering used to loan it to many programmers fresh out of college. Like Practice of Programming, it helps you get from what you learn in school to what will help you write production code in the real world.I don’t give it 5 stars because it’s just not really a classic. You should give it a read if you program C or C++, though.

Master’s Handbook: A TRUE Beginner’s Guide! Problem Solving, Code, Data Science, Data Structures & Algorithms (Code like a PRO in ... web design, tech, perl, ajax, swift, python,) Java Programming: Master’s Handbook: A TRUE Beginner’s Guide! Problem Solving, Code, Data Science, Data Structures & Algorithms (Code like a PRO in ... web design, tech, perl, ajax, swift, python) Sentences and Paragraphs: Mastering the Two Most Important Units of Writing (The Writing Code Series Book 8) Inside the Registry for Microsoft Windows 95: Developer’s Guide to Tapping the Power of the Registry (Microsoft Programming Series) By Charles Petzold - Programming Windows 5th Edition Book/CD Package: The definitive guide to the Win32 API (Microsoft Programming Series) (5th Edition) (10.2.1998) Microsoft Win32 Developer’s Reference Library - GDI (Microsoft Developers Library Win 32 GDI) (Microsoft Windows GDI) Microsoft Win32 Developer’s Reference Library - (Microsoft Developers Library Win 32 BASE SERVICES (Microsoft Win 32 - Base Services) GO! with Microsoft PowerPoint 2013 Brief, GO! with Microsoft Excel 2013 Brief, GO! with Microsoft Access 2013 Brief Delphi Programming with COM and ActiveX (Programming Series) (Charles River Media Programming) Microsoft Official Course 2778A Writing Queries Using Microsoft SQL Server 2008 Transact-SQL