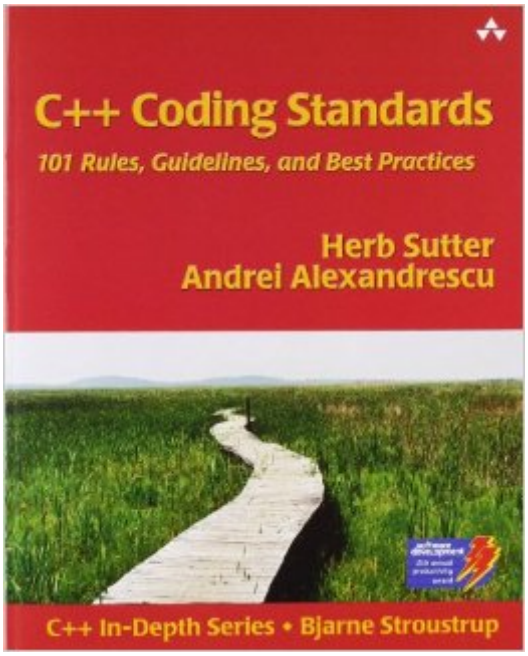
C++ Coding Standards: 101 Rules, Guidelines, And Best Practices



Synopsis

Consistent, high-quality coding standards improve software quality, reduce time-to-market, promote teamwork, eliminate time wasted on inconsequential matters, and simplify maintenance. Now, two of the world's most respected C++ experts distill the rich collective experience of the global C++ community into a set of coding standards that every developer and development team can understand and use as a basis for their own coding standards. The authors cover virtually every facet of C++ programming: design and coding style, functions, operators, class design, inheritance, construction/destruction, copying, assignment, namespaces, modules, templates, genericity, exceptions, STL containers and algorithms, and more. Each standard is described concisely, with practical examples. From type definition to error handling, this book presents C++ best practices, including some that have only recently been identified and standardized-techniques you may not know even if you've used C++ for years. Along the way, you'll find answers to questions like What's worth standardizing--and what isn't? What are the best ways to code for scalability? What are the elements of a rational error handling policy? How (and why) do you avoid unnecessary initialization, cyclic, and definitional dependencies? When (and how) should you use static and dynamic polymorphism together? How do you practice "safe" overriding? When should you provide a no-fail swap? Why and how should you prevent exceptions from propagating across module boundaries? Why shouldn't you write namespace declarations or directives in a header file? Why should you use STL vector and string instead of arrays? How do you choose the right STL search or sort algorithm? What rules should you follow to ensure type-safe code? Whether you're working alone or with others, C++ Coding Standards will help you write cleaner code--and write it faster, with fewer hassles and less frustration.

Book Information

Paperback: 240 pages
Publisher: Addison-Wesley Professional; 1 edition (November 4, 2004)
Language: English
ISBN-10: 9780321113580
ISBN-13: 978-0321113580
ASIN: 0321113586
Product Dimensions: 7.3 x 0.6 x 9.1 inches
Shipping Weight: 1.1 pounds (View shipping rates and policies)
Average Customer Review: 4.5 out of 5 starsÂ Â See all reviewsÂ (40 customer reviews)

Best Sellers Rank: #193,762 in Books (See Top 100 in Books) #123 inÂ Books > Computers & Technology > Programming > Microsoft Programming > C & C++ Windows Programming #135 inÂ Books > Computers & Technology > Programming > Languages & Tools > C & C++ > C++ #743 inÂ Books > Textbooks > Computer Science > Programming Languages

Customer Reviews

Sutter and Alexandrescu are certified C++ gurus, and have each written classic works on C++ (Exceptional C++ series, and Modern C++ Design, respectively). So why does this book fall short? Because it doesn't go into the level of detail necessary to make every recommendation meaningful, and instead relies on citations of previous works. And those citations very often fall into a handful of books that every serious C++ programmer should own and understand anyway: Effective C++ series by Scott Meyers, The C++ Programming Language by Bjarne Strousup, and Exceptional C++ by Sutter.One might argue that 5 books or more is too many, and that this book adds value by providing a one stop ultimate resource for best practices. The problem is that if proper justification isn't provided for each best practice, it's difficult for readers to internalize them. Even if these guys are experts, and a, "trust me" will suffice to believe what they say, it doesn't mean that everyone will understand what they say without diving into the other books that they so often reference. And that brings us back to my main point: you may as well just buy and read the original books in the first place.Many of the items are complete repeats of items from Scott Meyers books with much less explanation. For example, number 81 of best practices, 'Prefer range operations to single-element operations', is the same as item 5 in 'Effective STL'. However, in Coding Standards, a page is devoted to the explanation; not sufficient if you don't already fully understand why this is a good practice. Meyers, on the other hand, spends 8 pages fully convincing you it is a good idea with several examples. After reading Meyers, I'm going to understand and remember the practice of preferring range member functions.

I have great respect for both authors from reading their other books/articles, and there are many good ideas in this book, but I was expecting to agree with the authors here much more than I do.Item 0: Don't sweat the small stuff. The authors say not to overlegislate naming and bracing standards, but they also say "do be consistent" and don't mix styles. From personal experience, I can say the only way to get a group of programmers to be consistent is by "sweating the small stuff" and having well-defined policies that are strictly enforced.Item 1: Zero tolerance of warnings. Eliminating Level 4 warnings (in Visual C++) from a complex application (as opposed to a library

intended for third-party use) is more trouble than it's worth. The authors' suggestion to decrease code readability (Examples 2 and 3) to get around these warnings is quite a bad idea, in my opinion.Item 59: I wish somehow there could be a better answer to the C++ namespace issue. Giving many symbols (but not all, like preprocessor macros, classes not in a namespace, etc.) two names (the qualified and the unqualified) based on where that symbol appears seems so wrong and at the very least makes searching and cut-and-pasting more difficult.The authors clearly prefer use of stl over custom containers (although they have not always followed their own advice), but they don't address many issues related to this, like are teams using stl supposed to use the peculiar stl naming practices across the board in all code, so stl dictates naming and all projects would use naming like some_stl_vector.push_back()? Or would code like m_object.DoSomething() be mixed together with the above statement so there really is no standard?

Sea (Practical Test Standards series) Private Pilot Airman Certification Standards - Airplane: FAA-S-ACS-6, for Airplane Single- and Multi-Engine Land and Sea (Practical Test Standards series) Nursing: Scope and Standards of Practice (Ana, Nursing Administration: Scope and Standards of Practice) 2nd (second) Edition published by Amer Nurses Assn (2010) Nursing Professional Development: Scope and Standards of Practice (Ana, Nursing Professional Development: Scope and Standards o) Commercial Pilot and Flight Instructor for Helicopter Practical Test Standards: FAA-S-8081-16A/FAA-S-8081-7B (Practical Test Standards series) Airline Transport Pilot and Type Rating Practical Test Standards for Airplane: FAA-S-8081-5F (July 2008; including Changes 1 through 7) (Practical Test Standards series) Private Pilot Practical Test Standards for Airplane Single-Engine Land and Sea: FAA-S-8081-14B (Practical Test Standards series)