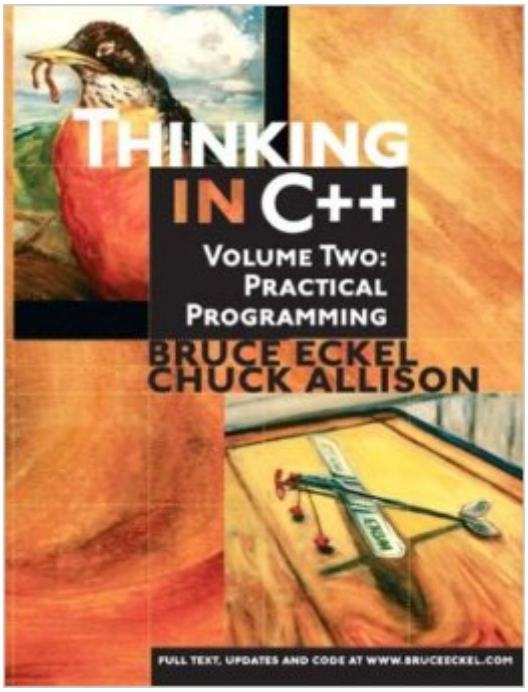
Thinking In C++, Volume 2: Practical Programming



Synopsis

The long awaited sequel to the highly successful Thinking in C++. More coverage of advanced topics professional developers must master. Emphasis on advanced testing techniques to produce optimized error free code. In depth coverage of STL with real world reusable code examples. Simple short exercises that simplify complex programming routines. Both authors are highly respected and widely known.

Book Information

Paperback: 832 pages

Publisher: Pearson; 1 edition (December 27, 2003)

Language: English

ISBN-10: 0130353132

ISBN-13: 978-0130353139

Product Dimensions: 6.9 x 1.7 x 9.2 inches

Shipping Weight: 2.6 pounds (View shipping rates and policies)

Average Customer Review: 4.4 out of 5 starsÂ Â See all reviewsÂ (57 customer reviews)

Best Sellers Rank: #385,295 in Books (See Top 100 in Books) #249 inÂ Books > Computers & Technology > Programming > Languages & Tools > C & C++ > C++ #294 inÂ Books > Computers & Technology > Programming > Microsoft Programming > C & C++ Windows Programming #1642 inÂ Books > Textbooks > Computer Science > Programming Languages

Customer Reviews

There are plenty of C++ books out there. When it comes down to it, one would really need two books on C++. Well, this is the second book you need; with the first one being any of many classics including the first volume of this book. Just when you think you have read or have a reference to all the C++ topics, Eckel and Allison come out with their continuation of a classic - volume two of Thinking in C++.Exceptions, Templates, STL, Design Patterns, Multiple Inheritance, concurrency and parallel programming are just some of the main topics covered in this book. With the book being over 800 pages, one can imagine how deep each of these topics must have been covered. Exceptions grab you right off the bat. Just when you think you know all about exceptions, the authors throw you a curve ball with "Exception Specification", and how to handle the "unexpected". You are blown away by the true control that C++ gives you. .Strings, along with examples given to depict the string class in full - as part of the standard template library (STL) of C++, begin the authors' discussion of the Standard Template Library. Vectors, sets, lists and many other features of

the STL have their own dedicated section which talks about generic containers. The authors set the stage for STL by describing the containers as:"Container classes are the solution to a specific kind of code reuse problem. ...A container class describes an object that holds other objects..."The authors then continue on to cover the very important and broad topic in C++ - containers. Examples after examples are used to convey the details and the tricky parts of the C++ STL. The key by reading this chapter is to portray and teach efficient techniques to common problems using the generic container classes. Not only the reader can learn most of what s/he needs to know about containers in this book, a small introduction is also given to show how to actually write a generic container - a linked list. The example is simple, yet powerful in conveying to the reader the ins and outs of writing generic containers. Speaking of generic, generic algorithms are covered very well in this book. All of the algorithms currently in the C++ library are covered with an accompanying example for each one. A special attention is given to the use of function objects as the means of customizing these algorithms."A function object is an instance of a class that overloads operator(), the function call operator. This operator allows an object to be used with function call syntax"Probably the simplest and the easiest definition of algorithm complexity theory is given in this book. The authors make the concept so easy to understand that even a non-programmer or mathematician can understand the reason and the complexity of complexity theory!Templates are probably my favorite topics in this book. Next to the C++ template "bible" by Vaqndervoorde and Josuttis, this book is the best source for generic programming and templates. The authors cover the three main topics of templates well:1) Templates with types as parameters2) Templates with compile-time constant as values3) Templates with other templates as values.The authors start with small and simple examples and build on top of them. Templates are generally a difficult topics to cover and to convey, but the authors do a great job in depicting the examples and teaching the reader how templates work and to create powerful programs with them.The advanced topics in this book (multiple inheritance, runtime type identification, design patterns, and concurrent programming) are my favorite topics. Multiple Inheritance is covered in depth, better than any other book that I have seen out there. The authors do caution the readers regarding the difficulties and the pitfalls of multiple inheritance, but they also explain with examples, pictures, etc... how a developer can overcome these difficulties (using upcast to resolve naming conflicts for example or avoiding the diamond-shaped inheritance tree).Concurrency and concurrent programming is one of the more difficult things to do in C++. Threads are usually used to achieve such task, but it takes practice and lots of sleepless nights to debug threaded programs and resolve deadlock issues with threads. The advantages of threads (ability to run multiple tasks concurrently, very low context

switching relative to processes, allow better code organization and the added convenience to the user specially for graphical user interface tasks) are depicted and shown thru a number of examples. The most important part of this section is how the authors use examples to show what the right way of doing something is as oppose to simply putting bunch of source code together to fill in the blank pages. The examples are priceless to me and very beneficial to any programmer.All and all, Bruce Eckel and Chuck Allison did a great job putting this book together. The topics covered are very beneficial to any serious C++ programmer or anyone wishing to become one. I particularly like the examples and "how-to's" given throughout the book as they are a valuable source when I am stuck with a programming challenge.

Volume Two picks up where the first left off, without skipping a beat. It starts by covering exceptions and unit tests, both of which should see more use in the real world. It then goes onto cover the standard C++ library in more depth that the first book. He then covers templates in a chapter that he calls 'Templates in Depth'. Yes, the coverage is long, about one hundred pages, but I would rename the chapter 'Templates at a practical level', which is exactly the where the coverage should be left. Templates, like macros, can be overused and have had whole books that cover the topic. Eckel chooses, and I think wisely so, to cover the topical to the extent that it would help you write practical templates yourself and to be able to use template libraries such as the Standard Template Library (STL).The STL is covered, well, in the two chapters that follow the template chapter. Once again the coverage is not absolutely complete because of the grand scope of the field. There are long books on the STL, but these chapters provide a pragmatic and thorough introduction which should serve for most practical purposes.The final chapters cover advanced topics. Notable are the chapters on Design Patterns which are designs for templates and classes what are considered industry 'best practice'. So instead of redesigning the wheel you use a design pattern, where appropriate. If you get into design patterns you should also read the extremely popular Design Patterns book, now in it's 25th printing.In the final chapters is also a discussion on multiple inheritance and threading. Both of which are covered at a pragmatic level and have whole books dedicated to the subject.This is an excellent, and needed addition to the original Thinking in C++ book. Both of the books are written in an accessible style and cover the topics at a practical level without rat-holing. For aspiring C++ programmers there is probably no better set of books to read as an introduction to C++.

I've been writing in C++ for about five years now (and in C for about ten years before that). Reading this book changed me from a C programmer writing code that the C++ compiler would (eventually)

accept, to a programmer who "thinks in C++".If you seriously want to learn C++, and you know "C", read (and re-read) this book, and you'll know more than 90% of the people out there who call themselves C++ programmers. I can say that, because I've "tech screened" many, many dozens of alleged C++ programmers; about 5% were competent. Work your way through this book, and you'll never be embarrassed during a code review!

The First Law of Technical Documentation states that:"The more complicated the subject, the less will be written about it, and the more likely there will be errors in what is written".This explains why typical programming books will have fifteen pages on If statements, but only a paragraph that says that "Interrupts can be serviced".Bruce does a masterful job of building the readers up so they are able to gradually yet thoroughly assimilate the subject matter. Thanks to years of putting on seminars and taking comments from readers, he teaches in useful and productive increments without overwhelming the readers. His examples are well thought out and useful. He actually responds to questions and comments. Please don't spam him, he seems like a genuinely nice guy! I look forward to taking one of his seminars in person.

Course to Learning the Basics of Python (Python Programming, Python Programming Course, Python Beginners Course) Swift Programming Artificial Intelligence: Made Easy, w/ Essential Programming Learn to Create your * Problem Solving * Algorithms! TODAY! w/ Machine ... engineering, r programming, iOS development) Delphi Programming with COM and ActiveX (Programming Series) (Charles River Media Programming) Programming #8:C Programming Success in a Day & Android Programming in a Day! PowerShell: For Beginners! Master The PowerShell Command Line In 24 Hours (Python Programming, Javascript, Computer Programming, C++, SQL, Computer Hacking, Programming) Python: Python Programming For Beginners - The Comprehensive Guide To Python Programming: Computer Programming, Computer Language, Computer Science (Machine Language)