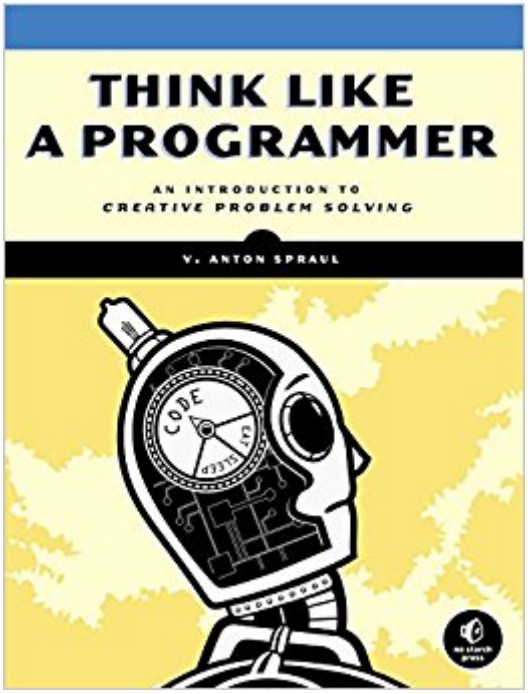
Think Like A Programmer: An Introduction To Creative Problem Solving



Synopsis

The real challenge of programming isn’t learning a language’s syntaxâ ”it’s learning to creatively solve problems so you can build something great. In this one-of-a-kind text, author V. Anton Spraul breaks down the ways that programmers solve problems and teaches you what other introductory books often ignore: how to Think Like a Programmer. Each chapter tackles a single programming concept, like classes, pointers, and recursion, and open-ended exercises throughout challenge you to apply your knowledge.You’ll also learn how to:â “Split problems into discrete components to make them easier to solveâ “Make the most of code reuse with functions, classes, and librariesâ “Pick the perfect data structure for a particular jobâ “Master more advanced programming tools like recursion and dynamic memoryâ “Organize your thoughts and develop strategies to tackle particular types of problemsAlthough the book’s examples are written in C++, the creative problem-solving concepts they illustrate go beyond any particular language; in fact, they often reach outside the realm of computer science. As the most skillful programmers know, writing great code is a creative artâ ”and the first step in creating your masterpiece is learning to Think Like a Programmer.

Book Information

Paperback: 256 pages
Publisher: No Starch Press; 1 edition (August 12, 2012)
Language: English
ISBN-10: 1593274246
ISBN-13: 978-1593274245
Product Dimensions: 7.1 x 0.7 x 9.2 inches
Shipping Weight: 1.4 pounds (View shipping rates and policies)
Average Customer Review: 4.3 out of 5 stars 32 customer reviews
Best Sellers Rank: #81,771 in Books (See Top 100 in Books) #21 inÂ Books > Computers & Technology > Programming > Languages & Tools > C & C++ > Tutorials #25 inÂ Books > Textbooks > Computer Science > Algorithms #59 inÂ Books > Computers & Technology > Programming > Algorithms

Customer Reviews

A Message From Author V. Anton Spraul â œOver the past 15 years, I’ve taught programming to countless students from every sort of background. A few of them were naturals, but most struggled, even the ones who would eventually turn into excellent programmers. However, they weren’t

struggling with the syntax of the programming language, but rather with applying it to solve the assigned problems. Knowing how to read a program is very different from knowing how to write one. While typical books or training courses are often effective at explaining the individual elements of programming, they tend to offer little advice on combining these elements to solve particular problems. That's where Think Like a Programmer comes in.â •

V. Anton Spraul has taught introductory programming and computer science for more than 15 years. He is the author of Computer Science Made Simple (Broadway) and How Software Works (No Starch Press). He offers advice for beginning programmers in his series "Learning to Program: A Guide" on his website (http://www.vantonspraul.com).

While I was working on my PhD in computer science, part of my job as a TA was to run the computer lab. I never cared for Java, but I learned enough of it to be able to help the undergraduates when they got stuck working on their homework assignments.What struck me is that often the undergrads know more Java than I did; their problem was that they didn't understand how to solve problems. Once I walked them through the process of designing a solution, then they could write the program. When I interviewed at Microsoft, the interviewer said the same thing: that many of the people he had talked to were not able to even answer the first interview question (which required figuring out a solution to a problem and then coding it).As such, it's no surprise that I was happy to see this book, with its promise of helping people understand how to solve problems rather than simply how to write code. The first chapter immediately dives in to solving some logic puzzles; while these aren't computer related (and some are classic problems that everyone knows) they get the point across that programming is about solving problems. The actual language is secondary; what's important is being able to break the problem down to the relevant information and figure out a way to solve it. Once you have an approach that allows you to tackle the problem, then you can figure out how to do each individual step.Chapter two switches to solving problems using C++, rather than generic logic puzzles, and then we're off and running. We follow that with one chapter each on solving problems using arrays, pointers and dynamic memory, classes, recursion, and code reuse. Finally we have a chapter about working to your strengths as a programmer to find solutions efficiently.For the most part, I enjoyed the book. There are a few places where it seems that the author made a change to a problem or assumption and then didn't fix later text that referred to the original version; for example, in chapter 5 the default constructor for a student object initializes grade to 0 and studentID to -1, but the following text refers to a possible error due to grade being

initialized to zero. Except for a problem in chapter two where relevant information is introduced in the solution rather than the problem description, though, these don’t detract too much from the reading.At the end of each chapter is a list of simply-described programming problems that require you to thoroughly explore the concept covered in that chapter. Working through them all should take some time, but will probably be worth it; even as a working programmer, I’m tempted to go through all of them just to get a better handle on C++, which I rarely use.The book assumes that the reader understands how to write a program with C++, but everything except the absolute basics has a review at the start of the relevant chapter. I think this would be a very good text to use for a freshman course on programming, assuming you can find one these days that still uses C++ rather than Java! (Although for the most part, the concepts apply to any language, so you could use it with Java anyway.) Having read this, a novice programmer should be much better equipped to break down a problem and get to work, rather than staring at the description wondering where to start.I give this 4.5 stars due to the errors mentioned above, which rounds up to five for a highly recommended book.Disclosure: I received a free review copy of this book for Vulcan Ears Book Reviews (vulcanears.com).

This well written book aims to fill the gap between understanding computer code and actually writing computer code. It does just that :)

Like robot.

Great book to help you think like a programmer..

I recently read Andy Hunt and Dave Thomas’ "The Pragmatic Programmer". I thought to myself that it was interesting to see this book from the perspective of 13 years later and what was still being practiced actively and where we may have moved on, as well as the suggestions they made to be effective and, yes, pragmatic programmers.To book end this experience, I wanted to look at another title that was in a similar vein, but much more recent, as in this title was released just a few months ago. This book, V. Anton Spraul’s "Think Like a Programmer", covers much of the same ground as "The Pragmatic Programmer", but does so with a much narrower scope. While "The Pragmatic Programmer" looked to focus on many different aspects of being effective, "Think Like a Programmer" puts the bulk of its energy on one issue; problem solving and the tools necessary to approach problems and develop solutions. The goal of this book is to help answer that age old

challenge... I understand the syntax, I can read code, I can modify code, I can work with other people’s code and understand what it’s doing, but when I sit down in front of a blank editor, I’m lost!"Think Like a Programmer" takes the reader down a number of paths to help explain some of programmings more challenging aspects and do so with generally basic coding structures. The entire book’s examples are in C++, so there is a unity to the problems being presented. While the examples are in C++, all but a few of the problems could be ported to other languages like Java, Ruby, Python, etc.. The Chapter on pointers might be the sole exception, and the chapter on classes comes from the perspective of a language that can be used with both procedural and object oriented approaches.Don’t let the code intimidate you, especially if you are not a programmer by trade. I did enough C++ programming in college to recognize the structures and the methods used for the solutions, so there was nothing in the code itself that was terribly frightening or all that advanced. Beginners, or those who have never seen any C++ code, may feel a little lost in spots, but Anton takes the time to explain every line. The goal of the book is not to focus specifically on coding syntax, but on the problem solving domain. Getting good and usable code, that’s a definite bonus, and he makes the case for doing all of the things that Andy and Dave talk about in "Practical Programmer".The problems all use standard elements of the C++ programming language. Arrays, pointer, classes and recursion all get a chapter dedicated to their own issues. Problems are presented and a detailed breakdown as to how to go about solving them is shown. Each chapter has a variety of exercises to work through.The last chapter focuses on allowing the programmer to devise their own game plan to solve problems, and each game plan will be unique to that particular programmer. Strengths and weaknesses are rarely the same, so making a "best practices" list for everyone would be pointless. Instead, Anton works the reader through methods and aspects that can help them create their own unique problem solving methodology, using all the tips and techniques practiced in the book, and leveraging their own individual strength, weaknesses, and areas of focus and interest. One thing to be aware of... there are no answers to the exercises provided. That’s by design. the point to the exercises is to see what you would come up with, and help break the cycle of "blank page, I’m lost!"Bottom Line:Seasoned developers may find this a bit basic for their tastes. Rank beginners might find the examples intimidating. Those in between, even those who are not C++ programmers, will likely learn a good deal and help de-mystify a few areas by working through the examples and problems. As a tester, rather than a programmer, I think that these titles are helpful to look at the issues that programmers face, as well as the issues and methods used to solve problems. Since we have to test those solutions, understanding how programmers get there and the tools they use to get there is helpful. "Think Like a Programmer" is a

solid step in helping both programmers and testers look at these situations in interesting ways.

My college did a decent job of training computer majors to be problem solvers. This book would have made an excellent companion text to instruction on data structures and algorithms.The author recommends knowledge of or simultaneous study of C++. I feel it would be accessible to any programmer in the C language family. Programmers in other languages could also obtain some benefit, but understanding would be a harder climb.Sure wish my workplace had allowed the luxury of studying a language before building production applications with it. Some of the author's recommendations in the last chapter are not always within the working programmer's control. The difference between academia and the rest of the world.All in all, well written, good use of examples, and sensible exercises to put the concepts into practice.

Very thought provoking and interesting way of organizing your programming

Great book. Uses C++ and elaborates on most programming concepts and issues and puts them in words that allows you to conceptually understand. This book had arrived badly damaged, as if it was kicked across a concrete floor and then put into the box for shipping.. contacted for a replacement so all is well.

InDesign Creative Cloud (Stay Current with Adobe Creative Cloud) Introduction to Orthotics: A Clinical Reasoning and Problem-Solving Approach, 4e (Introduction to Splinting) Do You Think What You Think You Think?: The Ultimate Philosophical Handbook Creative Problem Solving: Multiple Strategies for the Same Answer, Grade 7 Strategies for Creative Problem Solving (3rd Edition) Creative Problem Solving, Grade 6: Multiple Strategies for the Same Answer Creative Problem Solving: Multiple Strategies for the Same Answer, Grade 5 Creative Problem Solving, Grade 8: Multiple Solutions for the Same Answer Creative Problem Solving, Grade 3: Multiple Strategies for Finding the Same Answer Creative Problem Solving, Grade 4: Multiple Solutions for the Same Answer