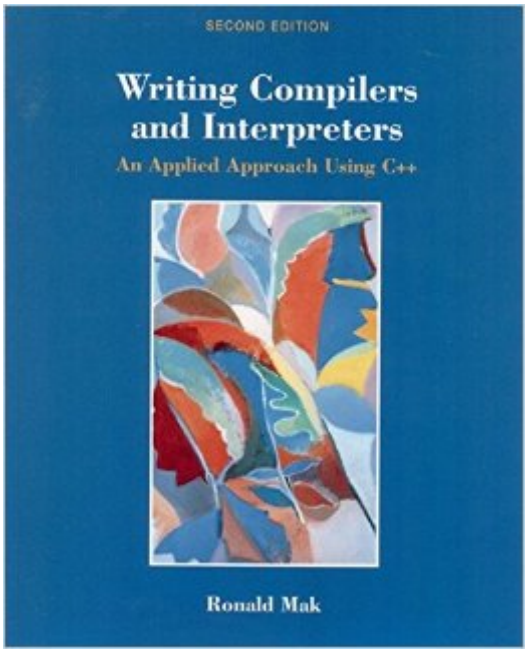
Writing Compilers And Interpreters



Synopsis

Quickly master all the skills you need to build your own compilers and interpreters in C++ Whether you are a professional programmer who needs to write a compiler at work or a personal programmer who wants to write an interpreter for a language of your own invention, this book quickly gets you up and running with all the knowledge and skills you need to do it right. It cuts right to the chase with a series of skill-building exercises ranging in complexity from the basics of reading a program to advanced object-oriented techniques for building a compiler in C++. Here's how it works: Every chapter contains anywhere from one to three working utility programs that provide a firsthand demonstration of concepts discussed, and each chapter builds upon the preceding ones. You begin by learning how to read a program and produce a listing, deconstruct a program into tokens (scanning), and how to analyze it based on its syntax (parsing). From there, Ron Mak shows you step by step how to build an actual working interpreter and an interactive debugger. Once you've mastered those skills, you're ready to apply them to building a compiler that runs on virtually any desktop computer. Visit the Wiley Computer Books Web page at: http://www.wiley.com/compbooks/

Book Information

Paperback: 864 pages
Publisher: Wiley; 2 edition (August 10, 1996)
Language: English
ISBN-10: 0471113530
ISBN-13: 978-0471113539
Product Dimensions: 7.4 x 1.8 x 9.2 inches
Shipping Weight: 2.8 pounds
Average Customer Review: 3.9 out of 5 starsÂ Â See all reviewsÂ (29 customer reviews)
Best Sellers Rank: #749,732 in Books (See Top 100 in Books) #48 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compiler Design #137 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compilers #371 inÂ Books > Computers & Technology > Hardware & DIY > Microprocessors & System Design

Customer Reviews

There are several things you should know about this book:1) The book implements a top-down or recursive-descent parser, as opposed to a standard shift-reduce parser. This is *very* important, as lex/yacc, Visual Parse++, and other parsing tools are efficient shift-reduce macines. Thus, the

parser isn’t really portable. Even so, I did find the the symbol table design that’s used by the parser to be critical for what I needed.2) The printed material is mostly (say 70%) code listings, thus even though the book is a whopping 838 pages, it would be much slimmer with fewer listings. The code is downloadable from the pusblisher’s (Wiley) site.3) The 30% of text and figures that are in the book could be much more insightful. For example, Chapter 11 - the interactive debugger should at least have some description (screenshots perhaps) of how to use the debugger. (Hint, the commands end with a semi-colon.)4) Even though this book is C++ oriented, it doesn’t use standard containers like linked lists, or trees (maps/sets). The classes have pointers in them that makes the class also act as a its own node in a list or whatever. This makes the design much more confusing than it needs to be.5) The symbol table implementation has heavy circular dependencies. Quite honestly I don’t know of a better implementation (yet). This does, however pose a problem if you’ll need to extend the design (to use STL containers, to self-serialize, etc.)The book has been a godsend, but I couldn’t honestly let the 4 and 5 star reviews sit unchallenged. If I had known the above sooner, I could have saved quite a few weekends.I think an Ideal Writing Compilers book would come bundled with a thirty day version of Visual Parse++ or Dr. Parse, and work from there.

The book describes step-by-step how the author would write a compiler for PASCAL. It could do with some more explanations of the logic behind some of the decisions,as it tends to quickly explain what the following C++ code does,before launching into pages of (well written) programming. If you have been tasked to write a specific compiler, then this book is probably what you want to get. If you are wanting to further your knowledge of the art, then you would be better looking at some of the more weighty volumes.

I bought this book in 1996 when I was a CS graduate student. The course text was the traditional "dragon book" which is a complete nigthmare to understand. I read this book in hopes of better understanding how compilers and interpreters are implemented and to this day I feel like I hit the jackpot.The book focuses primarily on the practical implementation of language interpreters and compilers and includes the code (C++) for a full featured Pascal interpreter (not just a minimal implementation that interprets a few statements). The author walks the reader through each class virtually line by line and presents the material in a way that any intermediate level C++ developer can easily understand.Notwithstanding the pragmatic focus of this book, it also provides excellent treatment of the theory of compiler design. While it is at least 5 years old, I still keep this book in my library.

This text fully accomplishes its goal of providing a simpleand practical introduction to this subject. Students andself-taught programmers having difficulty understandingcompiler theory from texts like the "dragon book" will findthis book very useful in getting started.Working thru all the well written C++ code also providesexcerise in polishing your C++ programming skills, beyond thefirst class introducing C++. Though there is a lot of code,I feel there is significant "added value" in the presentationof code segments and textual descriptions which helps novicesgrasp implementation of the concepts being discussed.As every author knows, books like all other projects can berefined further. One enhancement to this book is thatevery chapter should include a (small) section dedicateddiscussing the theoretical concepts without any referenceto the code. Alternative approaches and advanced conceptscould be mentioned here with a word about using simpletechniques to stay in line with the goal of the book.

This book delivers exactly what it promises--a complete step-by-step example of writing 'a compiler'. The book is simply a description of one way to build one compiler (and interpreter, and debugger, and various useful utilities).The basics are well presented. First a topic is described, then source code is presented and explained. The results of test runs are shown, and then off to the next topic. Advanced topics, such as optimization, are intentially left out.When a person is ready to read a first book about compilers, this is a good one. All source code developed/described in the book is available on-line.

I used to think you had to be some kind of super human being to write a compiler.Guess what? I was wrong. If you buy this book and you have good c++ programming skills as well as knowledge of data structures(lists,trees etc) you are well on your way.Ronald is the man!He breaks the code down into small objects and shows all the code with great insight into what the code is doing. Man, this is how to write a book on such a complex topic. Ronald really shows the benefits of OOP.Now I feel very confident to take on any programming project. I have over come my fears. I can now get more advanced books on the subject.

Instruction, Writing a Book) The Good Guide: A Sourcebook for Interpreters, Docents, and Tour Guides Reading Between the Signs: Intercultural Communication for Sign Language Interpreters 3rd Edition Reading Between the Signs: Intercultural Communication for Sign Language Interpreters 2nd Edition The Bilingual Courtroom: Court Interpreters in the Judicial Process (With a New Chapter) Compilers: Principles, Techniques, and Tools Compilers: Principles, Techniques, and Tools (2nd Edition) Algorithms, Languages, Automata, And Compilers: A Practical Approach Compilers: Principles and Practice High-Performance Compilers for Parallel Computing Optimizing Compilers for Modern Architectures: A Dependence-based Approach Non Fiction Writing Templates: 44 Tips to Create Your Own Non Fiction Book (Writing Templates, Writing Non Fiction, Kindle Publishing) Fiction Writing Templates: 30 Tips to Create Your Own Fiction Book (Writing Templates, Fiction Writing, Kindle Publishing) Love Writing - How to Make Money Writing Romantic or Erotic Fiction (Secrets to Success Writing Series Book 5) Coloring Journal (black): Therapeutic journal for writing, journaling, and note-taking with coloring designs for inner peace, calm, and focus (100 ... and stress-relief while writing.) (Volume 11) Digital Paper: A Manual for Research and Writing with Library and Internet Materials (Chicago Guides to Writing, Editing, and Publishing) The Erotica Handbook: (How to Write Erotica) A guide to making $100 an hour writing erotica short stories and selling them online (Emily Baker Writing Skills and Reference Guides)