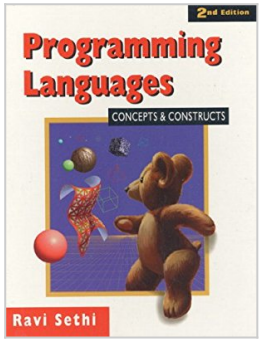

Programming Languages: Concepts and Constructs (2nd Edition)

By Ravi Sethi

Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi

Programming Languages: Concepts and Constructs, Second Edition retains the "character" of the original, emphasizing concepts and how they work together. This classic book has been thoroughly revised to provide readable coverage of the major programming paradigms. Dr. Sethi's treatment of the core concepts of imperative programming in languages like Pascal and C flows smoothly into object-oriented programming in C++ and Smalltalk. The charm of functional languages is illustrated by programs in standard ML and the Scheme dialect of Lisp. Logic programming is introduced using Prolog. Novices, who have been introduced to programming in some language, will learn from this book how related concepts work together while designers and implementers willp be exposed to the major programming paradigms. Example programs from the book are available as source code. These are available by anonymous ftp at ftp://ftp.aw.com/cseng/authors/sethi/pl2e. 0201590654B04062001

Programming Languages: Concepts and Constructs (2nd Edition)

By Ravi Sethi

Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi

Programming Languages: Concepts and Constructs, Second Edition retains the "character" of the original, emphasizing concepts and how they work together. This classic book has been thoroughly revised to provide readable coverage of the major programming paradigms. Dr. Sethi's treatment of the core concepts of imperative programming in languages like Pascal and C flows smoothly into object-oriented programming in C++ and Smalltalk. The charm of functional languages is illustrated by programs in standard ML and the Scheme dialect of Lisp. Logic programming is introduced using Prolog. Novices, who have been introduced to programming in some language, will learn from this book how related concepts work together while designers and implementers willp be exposed to the major programming paradigms. Example programs from the book are available as source code. These are available by anonymous ftp at ftp://ftp.aw.com/cseng/authors/sethi/pl2e. 0201590654B04062001

Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi Bibliography

- Sales Rank: #1138612 in Books
- Published on: 1996-01-07
- Original language: English
- Number of items: 1
- Dimensions: 9.00" h x 1.30" w x 7.30" l, 2.52 pounds
- Binding: Paperback
- 624 pages

Editorial Review

From the Back Cover

Programming Languages: Concepts and Constructs, Second Edition retains the "character" of the original, emphasizing concepts and how they work together. This classic book has been thoroughly revised to provide readable coverage of the major programming paradigms. Dr. Sethi's treatment of the core concepts of imperative programming in languages like Pascal and C flows smoothly into object-oriented programming in C++ and Smalltalk. The charm of functional languages is illustrated by programs in standard ML and the Scheme dialect of Lisp. Logic programming is introduced using Prolog.

Novices, who have been introduced to programming in some language, will learn from this book how related concepts work together while designers and implementers willp be exposed to the major programming paradigms.

Example programs from the book are available as source code. These are available by anonymous ftp at ftp://ftp.aw.com/cseng/authors/sethi/pl2e.

0201590654B04062001

About the Author

About Ravi Sethi

Ravi Sethi, director of Computing Science Research, has been at AT&T Bell Laboratories in Murray Hill, New Jersey since 1976. He has held teaching positions at Pennsylvania State university and the University of Arizona, and has taught at Princeton University and Rutgers. Dr. Sethi is co-author of the "dragon book", *Compilers: Principles, Techniques and Tools* and has written numerous articles. His books have been translated in Japanese, German, French, Italian, Spanish, and Korean.

0201590654AB04062001

This book is designed for junior/senior level courses on programming languages. A minimal pre-requisite is an introductory programming course. With supplementary readings, the book can also be used for graduate courses.

What's New in this Edition?

Changes on the language scene and feedback from the use of the book have prompted a thorough revision. Instructors liked the emphasis on concepts, but asked that the concepts be illustrated using fewer languages. Meanwhile, Modula-2 has faded, and C++ has taken off as a language for production programming. Candidates for functional languages now include Standard ML, Haskell, and Miranda.

The new outline has 15 chapters, three more than the first edition. The role of the three new chapters is as follows:

- Data types like arrays, records, and pointers have a new chapter.
- Functional programming is introduced using ML in a new chapter.
- Language summaries appear in a final chapter.

Language description and syntax are now treated early, in Chapter 2.

Organization of this Book

The emphasis is on concepts and how they work together, rather than on language features. Related concepts are therefore covered together, to allow meaningful examples and programming exercises along the way. Just enough of a language is introduced, as needed, for the examples and exercises. Language summaries appear in Chapter 15.

Part I: Introduction

Chapter 1 traces the role and development of programming languages. It introduces the programming paradigms in this book. They include imperative, object-oriented, functional, and logic programming.

Syntax description is treated in Chapter 2, so it can be applied in the rest of the book. The examples in the chapter deal with expressions, since methods for describing the syntax of expressions carry over to the rest of a language.

Part II: Imperative Programming

The imperative family is treated in Chapters 3-5. The term "imperative" comes from command or action; the computation model is that of a sequence of actions on an underlying machine.

Chapter 3 deals with control flow. Structured constructs like **while** statements organize the flow of control so that the unit of programming is a structured statement, instead of an individual assignment. Students in a course that emphasizes imperative programming are usually familiar with Pascal, so this chapter goes beyond assignments and structured statements to consider programming with invariants. The examples deal with basic values, like integers,

and arrays.

Chapter 4 deals with data in imperative languages. Data representation facilities such as arrays, records, and pointers, have been stable since Pascal and C appeared. The treatment of these facilities anticipates their use to represent objects in Chapters 6 and 7.

Chapter 5 rounds out the discussion of the core of imperative languages, embodied in a language like Pascal or C. Among the topics are the distinction between the source text of a procedure and its activations, parameter passing, scope rules, and storage allocation.

This book illustrates imperative programming using Pascal, where possible. Pascal suffices as a vehicle for Chapters 3-5. C is an alternative.

Part III: Object-Oriented Programming

As programs get larger, the natural unit of programming is a grouping of data and operations. The progression of concepts for such groupings can be described in terms of modules, user-defined types (for example, stacks), and classes (as in object-oriented programming).

Chapter 6 begins with of programming with procedures, modules, and classes. These constructs serve distinct needs and can be used in combination with each other: procedures are needed to implement operations in a module or class; modules can be used to statically partition the source text of a program with classes. Some versions of Pascal support modules; they can be used for the first half of Chapter 6 as well. C++, an extension of C, is introduced in Chapter 6.

The model of computation in Chapter 7 is that of independent objects. The objects interact by sending messages to each other. The first third of the chapter introduces object-oriented programming in general, using a running example that has similar implementations in C++ and Smalltalk. The rest of the chapter has independent coverage of C++ and Smalltalk, so either one can be used to explore object-oriented programming. Based on feedback from instructors, this edition covers C++ before Smalltalk, inverting the order in the previous edition. Object-oriented programming is illustrated using both C++ and Smalltalk, since the two represent different approaches.

All of the concepts in Chapters 3 - 7 can be illustrated using C++. Students can be introduced directly to C++, without going through C.

Part IV: Functional Programming

Functional programming is worth studying as a programming style in its own right; as a setting for studying concepts such as types; and as a technique for language description. The emphasis in Chapter 8 is on concepts, in Chapters 9 and 10 on programming style, and in Chapter 13 on language description. The computational model is based on an expression interpreter; an expression consists of a function applied to subexpressions.

The emphasis in Chapter 8 is on concepts. The simplicity of functional languages makes them convenient for introducing concepts such as values, types, names, and functions. The simplicity results from the emphasis on expressions and values, independent of the underlying machine. The chapter treads ground common to functional languages, using ML as the working language.

The fundamental difference between ML and Lisp is that ML is typed; the influence of types permeates the language. Chapter 9 uses ML to illustrate the use of functions and datatypes. As first-class citizens, functions have the same status as any other values in functional programming. This first-class status permits the creation of powerful operations on collections of data.

Functional programming originated with Lisp. Programs and data are both represented by lists in Lisp; the name is a contraction of "List Processor." The uniform use of lists makes Lisp eminently extensible. Chapter 10 explores the use of lists, using the Scheme dialect of Lisp.

See also Chapter 13, which contains an interpreter for a small subset of Scheme, and Chapter 14, which covers the lambda calculus.

Part V: Other Paradigms

Logic programming goes hand in hand with Prolog, in Chapter 11. Logic programming deals with relations rather than functions. Where it fits, programs are concise, consisting of facts and rules. The languages uses the facts and rules to deduce responses to queries.

Concurrent programming is illustrated using Ada, in Chapter 12. An alternative approach would have been to cover concurrent programming after object-oriented programming. Processes can be formed by giving each object its own thread of computation. The present organization puts functional programming before concurrent programming.

Part VI: Language Description

The methods for language description in Chapter 13 are aimed at specialists. The methods range from attributes used for language translation, to logical rules for used type inference, to interpreters used for clarifying subtle language questions.

A language can be described by writing a definitional interpreter for it, so called because its purpose is to define the interpreted language; efficiency is not a concern. McCarthy's & original definitional interpreter for Lisp in Lisp remains important for language description, so language description is illustrated using the Scheme dialect of Lisp. Chapter 13 develops an interpreter for a small subset of Scheme.

The lambda calculus is the intellectual ancestor of functional languages. The small syntax of the lambda calculus has also led to its use as a vehicle for studying languages. Variants of the lambda calculus are introduced in Chapter 14. The chapter progresses from the pure untyped lambda calculus to typed lambda calculi.

Chapter 15 contains brief summaries of the languages in this book.

Acknowledgments From the First Edition

A graduate seminar at Rutgers University gave me both the opportunity and the incentive to collect material on programming languages. I'd like to thank Alex Borgida, Martin Carroll, Fritz Henglein, Naftaly Minsky, Bob Paige, and Barbara Ryder for keeping the seminar lively.

An undergraduate course at Harvard University used an early draft of this book. Written comments by the students in the course were very helpful.

The organization of this book has benefited greatly from the comments and especially the criticism of the then anonymous reviewers contacted by Addison-Wesley. They are Tom Cheatham, Harvard University, John Crenshaw, Western Kentucky University, Paul Hilfinger, University of California, Berkeley, Barry Kurtz, New Mexico State University, Robert Noonan, College of William and Mary, Ron Olsson, University of California, Davis, William Pervin, University of Texas at Dallas, Paul Reynolds, University of Virginia, David Schmidt, Kansas State University, and Laurie Werth, University of Texas at Austin.

For all their technical help, I am grateful to Al Aho, Jon Bentley, Gerard Berry, Eric Cooper, Bruce Duba, Tom Duncan, Rich Drechsler, Peggy Ellis, Charlie Fischer, Dan Friedman, Georges Gonthier, Bob Harper, Mike Harrison, Bruce Hillyer, Brian Kernighan, Kim King, Chandra Kintala, Dave MacQueen, Dianne Maki, Doug McIlroy, John Mitchell, Mike O'Donnell, Dennis Ritchie, Bjarne Stroustrup, Chris Van Wyk, and Carl Woolf.

This book on programming languages was produced with the help of a number of little languages. The diagrams were drawn using Brian Kernighan's Pic language; the grey-tones in the diagrams rely on the work of Rich Drechsler. The tables were laid out using Mike Lesk's Tbl program. Eqn, Lorinda Cherry and Brian Kernighan's language for typesetting mathematics, handled the pseudo-code as well. The Troff program was originally written by the late Joe Ossanna and is kept vital by Brian Kernighan. Page layout would have suffered without a new Troff macro package and post-processor by Brian Kernighan and Chris Van Wyk. The indexing programs were supplied by Jon Bentley and Brian Kernighan. Cross references were managed using scripts written with the help of Al Aho for managing the text of the "dragon" book.

Finally, I'd like to thank AT&T Bell Laboratories for its support. I have learnt more from my colleagues here than they might suspect. Whenever a question occurred, someone in the building always seemed to have the answer.

Acknowledgments

I really appreciate the comments I have received on the first edition. The

experience of instructors and the frank opinions of reviewers have guided the revision.

Debbie Lafferty of Addison-Wesley has been the voice on the phone through the months, coordinating reviews and credits, and generally keeping the project on track. I now know that the reviewers include Bill Appelbe, Michael Barnett, Manuel E. Bermudez, Ray Ford, Aditya P. Mathur, L. A. Oldroyd, and Hamilton Richards -- thanks.

For technical help and discussions, I am grateful to Jon Bentley, Lorinda Cherry, Brian Kernighan, Dave MacQueen, Jon Riecke, and Rich Wolf. My colleagues at AT&T Bell Laboratories have been greatly supportive.

A lot has happened while I have been immersed in the Book, including a death, a birth, a move, a fire. Dianne Maki has helped me navigate through it all.

RS

0201590654P04062001

Users Review

From reader reviews:

David Browning:

Book will be written, printed, or created for everything. You can recognize everything you want by a book. Book has a different type. As we know that book is important matter to bring us around the world. Next to that you can your reading proficiency was fluently. A guide Programming Languages: Concepts and Constructs (2nd Edition) will make you to possibly be smarter. You can feel considerably more confidence if you can know about everything. But some of you think that will open or reading the book make you bored. It is not make you fun. Why they might be thought like that? Have you searching for best book or ideal book with you?

Nettie Powers:

In this 21st centuries, people become competitive in each and every way. By being competitive right now, people have do something to make all of them survives, being in the middle of the actual crowded place and notice by simply surrounding. One thing that often many people have underestimated this for a while is reading. That's why, by reading a publication your ability to survive boost then having chance to stand than other is high. To suit your needs who want to start reading a new book, we give you this particular Programming Languages: Concepts and Constructs (2nd Edition) book as beginning and daily reading reserve. Why, because this book is greater than just a book.

Keith Mayo:

Reading a e-book tends to be new life style on this era globalization. With reading through you can get a lot of information that will give you benefit in your life. Using book everyone in this world may share their idea. Books can also inspire a lot of people. A lot of author can inspire their particular reader with their story or even their experience. Not only the storyplot that share in the ebooks. But also they write about the knowledge about something that you need instance. How to get the good score toefl, or how to teach your sons or daughters, there are many kinds of book that exist now. The authors in this world always try to improve their skill in writing, they also doing some analysis before they write on their book. One of them is this Programming Languages: Concepts and Constructs (2nd Edition).

Wanda Holmes:

The publication with title Programming Languages: Concepts and Constructs (2nd Edition) has a lot of information that you can understand it. You can get a lot of gain after read this book. That book exist new know-how the information that exist in this book represented the condition of the world today. That is important to yo7u to find out how the improvement of the world. This particular book will bring you inside new era of the internationalization. You can read the e-book on your smart phone, so you can read that anywhere you want.

Read Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi for online ebook

Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi Free PDF d0wnl0ad, audio books, books to read, good books to read, cheap books, good books, online books, books online, book reviews epub, read books online, books to read online, online library, greatbooks to read, PDF best books to read, top books to read Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi books to read online.

Online Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi ebook PDF download

Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi Doc

Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi Mobipocket

Programming Languages: Concepts and Constructs (2nd Edition) By Ravi Sethi EPub