

<<数据结构与问题求解>>

图书基本信息

书名：<<数据结构与问题求解>>

13位ISBN编号：9787302237617

10位ISBN编号：7302237611

出版时间：2010-10

出版时间：韦斯(Mark Allen Weiss) 清华大学出版社 (2010-10出版)

作者：韦斯

页数：961

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<数据结构与问题求解>>

前言

This book is designed for a two-semester sequence in computer science, beginning with what is typically known as Data Structures and continuing with advanced data structures and algorithm analysis. It is appropriate for the courses from both the two-course and three-course sequences in "B.1 Introductory Tracks," as outlined in the final report of the Computing Curricula 2001 project (CC2001) — a joint undertaking of the ACM and the IEEE. The content of the Data Structures course has been evolving for sometime. Although there is some general consensus concerning topic coverage, considerable disagreement still exists over the details. One uniformly accepted topic is principles of software development, most notably the concepts of encapsulation and information hiding. Algorithmically, all Data Structures courses tend to include an introduction to running-time analysis, recursion, basic sorting algorithms, and elementary data structures. Many universities offer an advanced course that covers topics in data structures, algorithms, and running-time analysis at a higher level. The material in this text has been designed for use in both levels of courses, thus eliminating the need to purchase a second textbook.

<<数据结构与问题求解>>

内容概要

《数据结构与问题求解（Java语言版）（第4版）（影印版）》是专为计算机科学专业的两个学期课程而设计的，从介绍什么是数据结构开始，继而对高级数据结构与算法进行分析。

《数据结构与问题求解（Java语言版）（第4版）（影印版）》以独特的方式，清晰地将每种数据结构的接口与其实现分离开来，即将如何使用数据结构与如何对数据结构编程相分离。

<<数据结构与问题求解>>

作者简介

作者：（美国）韦斯（Mark Allen Weiss）

<<数据结构与问题求解>>

书籍目录

part one Tour of Java1 primitive java1.1 the general environment 41.2 the first program 51.2.1 comments 51.2.2 main 61.2.3 terminal output 61.3 primitive types 61.3.1 the primitive types 61.3.2 constants 71.3.3 declaration and initialization of primitive types1.3.4 terminal input and output 81.4 basic operators 81.4.1 assignment operators 91.4.2 binary arithmetic operators 101.4.3 unary operators 101.4.4 type conversions 101.5 conditional statements 111.5.1 relational and equality operators1.5.2 logical operators 121.5.3 the if statement 131.5.4 the while statement 141.5.5 the for statement 141.5.6 the do statement 15 1.5.7 break and continue 161.5.8 the switch statement 171.5.9 the conditional operator 171.6 methods 181.6.1 overloading of method names 191.6.2 storage classes 20summary 20key concepts 20common errors 22on the internet 23exercises 23references 252 reference types2.1 what is a reference? 272.2 basics of objects and references 302.2.1 the dot operator (.) 302.2.2 declaration of objects 302.2.3 garbage collection 312.2.4 the meaning of = 322.2.5 parameter passing 332.2.6 the meaning of == 332.2.7 no operator overloading for objects 342.3 strings 352.3.1 basics of string manipulation 352.3.2 string concatenation 352.3.3 comparing strings 362.3.4 other String methods 362.3.5 converting other types to strings 372.4 arrays 372.4.1 declaration, assignment and methods 382.4.2 dynamic array expansion 402.4.3 ArrayList 422.4.4 multidimensional arrays 452.4.5 command-line arguments 452.4.6 enhanced for loop 462.5 exception handling 472.5.1 processing exceptions 482.5.2 the finally clause 482.5.3 common exceptions 492.5.4 the throw and throws clauses 512.6 input and output 512.6.1 basic stream operations 522.6.2 the Scanner type 532.6.3 sequential files 56summary 59key concepts 60common errors 61on the internet 62exercises 62references 683 objects and classes3.1 what is object-oriented programming? 693.2 a simple example 713.3 Javadoc 733.4 basic methods 763.4.1 constructors 763.4.2 mutators and accessors 763.4.3 output and toString 783.4.4 equals 783.4.5 main 783.5 example: using java.math.BigInteger 783.6 additional constructs 793.6.1 the this reference 813.6.2 the this shorthand for constructors 823.6.3 the instanceof operator 823.6.4 instance members versus static members 833.6.5 static fields and methods 833.6.6 static initializers 863.7 example: Implementing a BigRational class 863.8 packages 903.8.1 the import directive 913.8.2 the package statement 933.8.3 the CLASSPATH environment variable 943.8.4 package visibility rules 953.9 a design pattern: composite (pair) 95summary 96key concepts 97common errors 100on the internet 100exercises 101references 1074 inheritance4.1 what is inheritance? 1104.1.1 creating new classes 1104.1.2 type compatibility 1154.1.3 dynamic dispatch and polymorphism 1164.1.4 inheritance hierarchies 1174.1.5 visibility rules 1174.1.6 the constructor and super 1184.1.7 final methods and classes 1194.1.8 overriding a method 1214.1.9 type compatibility revisited 1214.1.10 compatibility of array types 1244.1.11 covariant return types 1244.2 designing hierarchies 1254.2.1 abstract methods and classes 1264.2.2 designing for the future 1304.3 multiple inheritance 1314.4 the interface 1344.4.1 specifying an interface 1344.4.2 implementing an interface 1354.4.3 multiple interfaces 1354.4.4 interfaces are abstract classes 136.....part two Algorithms and Building Blocks6 the collections api7 recursion8 sorting algorithms9 randomizationpart three Applications10 fun and games11 stacks and compilers12 utilities.....

<<数据结构与问题求解>>

章节摘录

插图：As discussed in Section 6.9, the priority queue supports the access and deletion of the minimum item with `findMin` and `deleteMin`, respectively. We could use a simple linked list, performing insertions at the front in constant time, but then finding and/or deleting the minimum would require a linear scan of the list. Alternatively, we could insist that the list always be kept sorted. This condition makes the access and deletion of the minimum cheap, but then insertions would be linear. Another way of implementing priority queues is to use a binary search tree, which gives an $O(\log N)$ average running time for both operations. However, a binary search tree is a poor choice because the input is typically not sufficiently random. We could use a balanced search tree, but the structures shown in Chapter 19 are cumbersome to implement and lead to sluggish performance in practice. (In Chapter 22, however, we cover a data structure, the *play tree*, that has been shown empirically to be a good alternative in some situations.)

<<数据结构与问题求解>>

编辑推荐

《数据结构与问题求解(Java语言版)(第4版)(影印版)》：大学计算机教育国外著名教材系列

<<数据结构与问题求解>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>