

<<数据结构>>

图书基本信息

书名：<<数据结构>>

13位ISBN编号：9787040092035

10位ISBN编号：7040092034

出版时间：2001-1

出版范围：高等教育

作者：张乃孝//裘宗燕

页数：392

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>>

前言

本书是一本系统介绍数据结构的教材。

在本书的论述过程中会反复联系到计算机领域的许多重要问题，包括问题求解，算法的分析与设计、抽象数据类型、程序设计方法、程序设计语言等内容，根据目前计算机科学发展的情况，本书采用面向对象的思想组织与课程有关的内容，运用C++语言作为数据结构和算法的描述语言，其目的是为了保证本书的先进性和适用性。

数据结构的一般讨论并不依赖于具体的语言，然而，在具体问题求解时，使用什么语言描述数据结构会在很大程度上影响程序设计者的思维方式和方法，选择一种实用的程序语言作为工作语言，有利于使讨论更面向实际应用，使参加课程学习的学生不仅可以掌握数据结构的抽象概念和性质，也有助于了解实现的思想和方法。

由于程序设计语言的发展，适合用来讨论数据结构的语言越来越多，今天国际上新的数据结构教材已经不再用伪语言作为工具了，随着面向对象方法的广泛应用，许多学校已经把C++语言定为学习计算机的第一种程序设计语言，越来越多的教材采用C++作为数据结构的编程语言。

在本书中，不仅讨论了抽象的数据结构，而且讨论了各种数据结构的实现方式并对它们进行了比较，从中可以看到不同数据结构的相互关系，包括结构上的（逻辑）关系和实现上的（表示）关系。

采用C++语言和面向对象的方法讲解数据结构，对这些关系的讨论提供了有力的工具。

由于C++语言的一些优点，特别是它的模板机制，使我们可以比较清晰地讨论具有类型参数的数据结构。

例如，我们实现的不是整数的堆栈，也不是字符串的堆栈，而是一般的具有类型参数的堆栈。

一个实现可以满足各种程序对于堆栈的需要。

这样学习的数据结构不仅更接近那种抽象概念上的数据结构，同时也可以直接用到不同类型对象的相同计算过程中去。

本书的讨论力图反映计算机科学的最新发展，用面向对象的方法组织数据结构的主要内容，以抽象数据类型的方式定义各种结构的用户界面，以时间和空间开销作为评价各种结构使用好坏的主要标准，从简到繁，系统地介绍各种常用的数据结构，强调不同结构之间的联系。

书中采用C++语言描述各种类型的界面和实现。

把面向对象的程序设计方法与数据结构的主要原理紧密结合起来，讲解过程中还插入大量实例。

读者通过本书的学习不仅能掌握数据结构的基本原理和基本方法，同时可以把所学的知识用于实际问题求解的过程之中。

<<数据结构>>

内容概要

《数据结构：C++与面向对象的途径（修订版）》是1998年6月出版的《数据结构——C++与面向对象的途径》一书的修订版。它采用面向对象的思想组织数据结构的内容，运用C++语言作为讨论数据结构的工作语言。

在第一版的基础上，除对各章的顺序及内容安排进行了进一步的调整之外，还补充了各章的例子、习题，并增加了若干上机实习题，使读者可以更好地对数据结构进行学习、实践。在《数据结构：C++与面向对象的途径（修订版）》的最后还附加了一个上机实习报告的例子，使其具有较强的实用性。

《数据结构：C++与面向对象的途径（修订版）》除延续了第一版的风格外，内容更加充实、完整，讲解更加清楚、透彻。

可作为本科计算机专业或相关专业数据结构课程教材，也可作为面向对象程序设计课程或C++程序设计实践课程的教材和参考书。

<<数据结构>>

书籍目录

第一章 绪论1.1 问题求解1.1.1 问题1.1.2 问题的分析1.1.3 算法的选择1.1.4 解的优化1.2 数据结构1.3 算法1.3.1 算法的设计1.3.2 算法的分析1.4 抽象数据类型1.5 程序设计方法和语言小结习题第二章 C++与面向对象初步2.1 C++语言对C的基本扩充2.1.1 注释2.1.2 函数原型说明2.1.3 引用和引用参数2.1.4 重载2.1.5 缺省参数2.1.6 变量说明2.1.7 输入和输出2.1.8 动态存储分配2.1.9 类型定义2.1.10 强制类型转换2.2 对象和类2.3 类的界面描述和实现2.3.1 类的数据域2.3.2 对象的行为——成员函数2.3.3 运算符作为成员函数2.3.4 用构造函数进行实例的初始化2.4 普通运算符和普通函数2.4.1 普通运算符2.4.2 普通函数2.4.3 输入和输出2.5 类的合成、继承和多态性2.5.1 合成2.5.2 继承2.5.3 多态性小结习题第三章 字符串——数据封装技术3.1 C语言的字符和字符串3.2 字符串数据抽象的描述和实现3.2.1 字符串类的定义3.2.2 构造函数的定义3.2.3 析构函数3.2.4 基本成员函数的实现3.2.5 比较运算符3.2.6 串连接3.2.7 输入和输出3.3 子串3.4 模式匹配3.4.1 简单字符串匹配3.4.2 Knuth-Morris-Pratt模式匹配算法3.4.3 Boyer - Moore字符串匹配算法小结习题第四章 向量——类的重用技术4.1 模板类4.2 向量的实现4.3 定界向量和枚举向量、——继承方式的重用4.3.1 定界向量4.3.2 枚举向量4.4 排序向量和矩阵——合成方式的重用4.4.1 排序向量和二分法检索4.4.2 矩阵4.5 向量遍历器4.5.1 遍历器的抽象4.5.2 向量遍历器4.6 向量的排序——模板函数4.6.1 插入排序4.6.2 起泡排序4.6.3 选择排序4.6.4 快速排序算法4.7 继承和多态的若干讨论4.7.1 父类与子类4.7.2 静态类型和动态类型4.7.3 框架和框架类4.7.4 蔽和虚函数4.7.5 虚遮蔽和非虚遮蔽4.7.6 两类继承4.7.7 多态的主要形式4.7.8 参数多态性——归约4.7.9 切割问题小结习题第五章 动态数据结构——链表5.1 单链表的定义5.1.1 表类5.1.2 链类5.2 单链表的实现5.2.1 链类的实现5.2.2 表类的实现5.3 表遍历器5.3.1 表遍历器类5.3.2 表遍历器类的实现表的应用：多项式处理5.4.1 项类5.4.2 多项式类5.5 排序表5.5.1 排序表类5.5.2 排序表类的实现5.5.3 排序表的应用——表插入排序5.6 其他链表5.6.1 自组织表5.6.2 双端表5.6.3 循环表5.6.4 双链表5.7 可利用空间表小结习题第六章 栈和队列6.1 抽象类栈和队列6.2 栈的实现6.2.1 栈的向量实现6.2.2 栈的链表实现6.3 栈的应用——表达式计算6.3.1 后缀表达式的求值6.3.2 中缀表达式到后缀表达式的转换6.4 队列的实现6.4.1 队列的向量实现6.4.2 队列的链表实现6.5 队列的应用——农夫过河问题小结习题第七章 树和二叉树7.1 基本概念7.1.1 树7.1.2 二叉树7.1.3 树与二叉树的关系7.2 二叉树的实现7.2.1 二叉树结点类7.2.2 基本二叉树类7.2.3 可构造二叉树类7.3 二叉树的周游7.3.1 周游的递归实现7.3.2 通过遍历器实现周游7.3.3 前序周游器类7.3.4 中序周游器类7.3.5 后序周游器类7.3.6 层次周游算法(按宽度方向周游)7.4 二叉树的向量表示7.4.1 二叉树向量表示的一种基本方法7.4.2 记录结构信息的二叉树向量表示7.5 二叉排序树7.6 平衡的二叉排序树7.6.1 AVL树上的操作7.6.2 AVL树的设计与实现7.7 二叉树的应用——哈夫曼树小结习题第八章 优先队列8.1 优先队列的抽象8.2 堆8.3 堆排序8.4 斜堆8.5 离散事件模拟8.5.1 模拟类的结构8.5.2 冰淇淋店的模拟8.5.3 随机数小结习题第九章 集合与字典9.1 集合及其运算9.1.1 集合运算9.1.2 集合类9.2 位向量集合9.2.1 位向量9.2.2 位向量集合9.2.3 字符集合9.2.4 字符集类的应用——将字符串分解为单词9.3 集合的表实现9.4 关联与字典9.5 字典的关联表实现9.6 字典的应用9.6.1 稀疏矩阵9.6.2 排序字典9.6.3 索引的实现小结习题第十章 散列结构10.1 散列结构10.2 散列函数10.3 开地址散列向量10.4 桶散列——用桶解决碰撞10.4.1 桶散列的抽象模板类10.4.2 用树作为桶的实现10.4.3 桶散列结构操作时间的分析10.5 桶散列结构的遍历器10.6 用散列表实现集合10.6.1 应用——拼写检查器10.7 用桶散列表实现字典小结习题第十一章 图11.1 基本概念11.2 图的邻接矩阵表示和Warshall算法11.2.1 图的邻接矩阵表示11.2.2 图结点的可达性问题11.3 邻接表方式的图表示和深度优先搜索11.3.1 邻接表表示中的结点类11.3.2 用深度优先方式求解可达性问题11.4 带权图的矩阵表示和Floyd算法11.4.1 带权图的邻接矩阵11.4.2 带权图最短路径问题Floyd算法11.5 带权图的邻接表表示与Dijkstra算法11.5.1 带权图的邻接表表示11.5.2 从一个结点出发的最短路径和Dijkstra算法11.6 连通性、带权连通无向图与最小生成树11.7 有限自动机11.8 拓扑排序小结习题第十二章 文件12.1 外存、文件及其问题12.1.1 外存储器的特点与信息组织12.1.2 文件基本结构和操作12.1.3 文件与字典12.1.4 文件组织12.2 C++的字符流文件及其操作12.3 归并排序12.4 文件的随机访问12.5 文件索引结构12.5.1 索引向量12.5.2 树形索引结构12.5.3 B树12.5.4 B+树12.6 树索引文件的实现小结习题附录附录A 主要抽象数据类型及其相互关系附录B BorlandC++集成开发环境使用入门附录C “多叉路口的交通管理系统”上机报告

<<数据结构>>

章节摘录

另一种常用的算法设计方法称为“分治法”，其基本思想是：把一个规模较大的问题分成两个或多个较小的与原问题相似的子问题。

首先对子问题进行求解，然后设法把子问题的解合并起来，得出整个问题的解，即对问题分而治之。如果一个子问题的规模仍然比较大，不能很容易地求得解，就可以对这个子问题重复地应用分治策略。

“二分法检索”就是用分治策略的典型例子。

如果要在一个整数的有序数组（可以假设元素按增序排列）中找一个数，要求查出这个数在这个数组里的位置，二分法检索采用的方法是先找到数组中居于中间位置的元素，将该元素与所找数字进行比较，如果所查数字比较小，那么它一定不会出现在中间元素的右边，下面只需在左半个数组中检索。

反过来，若所查数字较大，则不必在左半个数组中继续寻找，只需在右半个数组中检索。

这样，通过一次比较就能够排除掉一半元素。

再做下去，只要对剩下的半个数组重复使用同样方式，又可以再排除掉四分之一的元素，如此重复若干次就能很快确定整数的存在与否，并确定如果存在时它在什么位置。

还有一类常用算法被称为“回溯法”。

有一些问题，只能通过彻底搜索所有可能情况寻找一个满足某些预定条件的最优解。

由于彻底搜索的运算量通常非常大，往往大到使用计算机也不能在合理时间内得到解。

回溯算法是处理这类问题的一个方法，基本思想是一步一步向前试探，当有多种选择时可以任意选择一种，只要可行（或暂时没有失败）就继续向前，一旦发现问题或失败就后退，回到上一步重新选择，借助于回溯技巧和中间判断，常常可以大大地减少搜索时间。

常见的迷宫问题以及八皇后问题都可以用回溯方法来解决。

除上面提到的几种典型方法外，常用的算法设计方法还有“动态规划”、“局部搜索”和“分支限界”等。

本书在讨论数据结构的同时，自然涉及与数据结构相关的许多算法，掌握这些算法的设计方法和共性，对于提高程序设计的水平是十分重要的。

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>