面向21世纪课程教材
Textbook Series for 21st Century

数据结构

刘大有　唐海鹰　孙舒杨　虞强源　杨鲲　编著

高等教育出版社
HIGHER EDUCATION PRESS

面向21世纪课程教材
Textbook Series for 21st Century

数据结构

刘大有　唐海鹰　孙舒杨　虞强源　杨鲲　编著

高等教育出版社
HIGHER EDUCATION PRESS

图书在版编目(CIP)数据

数据结构/刘大有等编著 .—北京:高等教育出版社,2001
ISBN 7-04-008908-4

Ⅰ.数… Ⅱ.刘… Ⅲ.数据结构 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2001)第 18741 号

责任编辑 倪文慧 **封面设计** 张 楠 **责任绘图** 尹文军
版式设计 马静如 **责任校对** 王效珍 **责任印制** 宋克学

数据结构
刘大有等 编著

出版发行	高等教育出版社		
社　　址	北京市东城区沙滩后街 55 号	**邮政编码**	100009
电　　话	010-64054588	**传　　真**	010-64014048
网　　址	http://www.hep.edu.cn http://www.hep.com.cn		
经　　销	新华书店北京发行所		
排　　版	高等教育出版社照排中心		
印　　刷	北京印刷二厂		
开　　本	787×960 1/16	**版　　次**	2001 年 7 月第 1 版
印　　张	29.5	**印　　次**	2001 年 7 月第 1 次印刷
字　　数	550 000	**定　　价**	24.70 元

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换。

内 容 提 要

本书是教育部“高等教育面向21世纪教学内容和课程体系改革计划”的研究成果，是面向21世纪课程教材。本书介绍了数据结构的概念和内容，主要内容包括绪论、基本数据结构、典型算法、复杂数据结构和应用等五部分。其中，绪论部分为本书的预备知识，主要对ADL语言、数据结构与算法、算法分析基础、OOP和C++作了简单介绍；基本数据结构部分包括线性表、堆栈与队列、数组、字符串、整数集合类、树和图等；典型算法部分主要介绍了若干典型算法的实现，并给出必要的复杂性分析和比较过程，具体包括递归、排序、查找和内存管理等；复杂数据结构部分主要包括优先级队列、不相交集合类和文件结构等；应用部分是上述数据结构和典型算法的一些应用示例，具体包括：事件驱动模拟、在线等价类、残缺棋盘和图像压缩等问题。

本书强调了算法与数据结构的紧密结合，对书中的算法都给出了时间复杂性分析，并注重分析的严格性，对典型算法还给出了算法正确性证明；本书特别使用了ADL算法描述语言，且在ADL的基础上引入了基于C++的算法描述，从而使了解OOP或C++的读者可从ADL角度和OOP两个方面认识和把握算法。全书注重讲、练结合，在各章后均附有丰富的习题和上机实习题目。

本书配有单机版多媒体课件和基于Web的网络版多媒体课件，可作为高等院校计算机专业教材，也可供专业技术人员参考使用。

前　言

随着计算机科学技术的发展和其应用领域的不断扩大，一方面计算机面对的数据具有结构十分复杂、数据量巨大且形式多样化的特点，另一方面在大多数应用中存取大量数据的能力被认为是其最重要的特性之一。这些都充分表明，要设计出高效、准确、适应性和可重用性强的程序，就必须对数据的性质和数据元素间的关系进行深入的研究，研究内容主要包括数据内部的逻辑结构，以及在计算机内对它的表示、存储、操作和应用等问题。准确阐明解决这些问题的方法和研究思路，是本书的主要内容。

本书强调了算法与数据结构的紧密结合。算法与数据结构是一对不可分割的孪生兄弟，不了解施加于数据上的算法就不知道怎样去构造数据；反之，若不深入研究作为其基础的数据结构，就无法设计出优美的算法。为此，对本书中的算法都给出了时间复杂性分析，并注重分析的严格性，对典型算法还给出了算法正确性证明。此外，本书在面向对象的途径和基本数据结构的系统阐述之间进行了折中，把一些数据结构中的重要典型算法，如排序、查找等，以显式的方式成章讲述。

本书特别使用了一种称为 ADL 的算法描述语言，该语言避免了过多涉及数据类型定义，能在较高层面上勾画出算法的轮廓，算法书写简洁、要点明晰，它既继承了 D. E. Knuth 书写算法的风格又易于递归算法的设计 。本书对大部分算法都给出了 ADL 描述和基于 ADL 的算法复杂性分析。引入 ADL 的另一点考虑是，使不太熟悉面向对象程序设计(OOP)和 C++ 语言的读者也能方便地使用本书。

鉴于 OOP 的先进性和 C++ 语言使用的广泛性，本书在 ADL 的基础上引入了基于 C++ 的算法描述，从而使了解 OOP 或 C++ 的读者可从 ADL 角度和 OOP 两个方面认识和把握算法。在基于 C++ 的算法描述中，我们的侧重点又有所不同：对基本数据结构，如线性表、树、图等，我们首先给出其规范化的 ADT(抽象数据类型)描述，使读者能从 OOP 的观点去把握其本质，然后再给出具体实现——C++ 类定义和基本方法的实现；对典型算法，如递归、排序、查找、内存管理等，则重点从方法实现的角度给出其 C++ 描述。

将“树”放在“递归”前，是考虑这样做有利于基本数据结构编排的系统性，不影响对“树”的理解。先学“树”为深入掌握“递归”概念奠定了基础。

本书给出了数据结构和典型算法的一些应用实例，如游戏问题、文件与数据压缩算法、模拟问题(事件驱动模拟)等，以期提高学生对数据结构课掌握的程度和用所学知识解决实际问题的能力。

本书对数据结构的讲解由浅入深，循序渐进，对一些有代表性的数据结构和算法采用渐进的、逐步求精的方式阐述：先给出基本解决方法——→做第一次改进——→做第二次改进——→做第三次改进——→……逐步深入；从教材的选材方面，既突出了基本内容，又兼顾了高级和扩展内容；从章节安排上，既有必学内容，又有选学内容(以 * 表示)，可以适应不同层次读者的要求。

本书注意把讲授、习题和上机实习相结合，在每章后附有丰富的习题和上机实习题目。本书还配有单机版多媒体课件和基于 Web 的网络版多媒体课件(2001 年年底完成)。

全书共分五部分：绪论、基本数据结构、典型算法、复杂数据结构和应用。绪论部分——本书的预备知识，主要对 ADL 语言、数据结构与算法、算法分析基础、OOP 和 C++ 作了简单介绍。基本数据结构部分包括线性表、堆栈与队列、数组、字符串、整数集合类、树和图等。典型算法部分主要介绍了若干典型算法的实现，并给出必要的复杂性分析和比较过程，具体包括递归、排序、查找和内存管理等。复杂数据结构部分主要包括优先级队列、不相交集合类和文件结构等。应用部分是上述数据结构和典型算法的一些应用示例，具体包括事件驱动模拟、在线等价类、残缺棋盘和图像压缩等问题。

参加本书编写工作的人员主要包括：刘大有、唐海鹰、孙舒杨、虞强源和杨鲲等。感谢山东大学计算机系张景淮老师在百忙之中对本书进行了审阅，并提出了宝贵意见。

鉴于时间仓促和水平有限，书中错误及不足在所难免，敬请专家和读者批评指正。

作 者

2000 年 12 月于吉林大学

目　录

第一章 绪 论

本章将简述数据结构与算法的关系，介绍数据、数据结构、面向对象程序设计、抽象数据类型和算法等概念，并给出一种算法描述语言 ADL。

1.1 数据结构概念

本书是阐明和研究数据结构的，因此必须回答“什么是数据结构?”这一问题。为了弄清数据结构这一概念，首先要明确“数据”这一术语的含义。简单地说，数据就是计算机程序要处理的“原料”，它可以被计算机识别、存储和加工处理。例如，一个简单的数值计算程序所使用的数据是一些实数或整数，一个编译程序使用和加工的数据是源程序，而一个能修改自身的计算机程序使用和加工的数据(对象)就是自身。

数据元素是组成数据的基本单位，在不同情况下，数据元素可以有元素、结点或者顶点等不同的名称。一个数据元素可以由若干个域(或称字段)组成。

数据结构指的是数据之间的相互关系，它主要包含 3 部分内容：

(1) 数据的逻辑结构，也就是数据元素之间的逻辑关系。

(2) 数据的存储结构，也就是数据元素及其相互关系(逻辑结构)在存储器中的实现方式。

(3) 对数据需要施加的操作，主要包括：查找、插入、删除、修改和排序等。

数据的逻辑结构可形式地表示为一个二元组：$L=(N,R)$，其中 $N(L)$是结点(数据元素的别称)的有限集合，$R(L)$是 N 上的关系集合。

设 $L=(N,R)$是一个逻辑结构，$R=\{r\}$(本书一般只讨论包含一个关系 r 的关系集合 R)，若 $a,b\in N$，且关系$(a,b)\in r$，则称 a 是 b 的前趋结点，称 b 是 a 的后继结点，称 a 和 b 是相邻结点。如果不存在 $a\in N$，使$(a,b)\in r$，则称 b 为始结点；如果不存在 $b\in N$，使$(a,b)\in r$，则称 a 为终结点；既非始结点又非终结点的结点被称为内结点。

数据的逻辑结构又可分为两大类：

(1) 线性结构

其特点是：结构中有且仅有一个始结点和一个终结点，始结点只有一个后继

结点，终结点只有一个前趋结点，每个内结点有且仅有一个前趋结点和一个后继结点。

线性结构最一般的情形是线性表，我们将在第四章给出线性表的具体说明。

(2) 非线性结构

其特点是：结构中的结点可能有多个前趋结点和多个后继结点。

最重要的非线形结构是“树”，树中有且仅有一个没有前趋结点的结点，称之为根结点；其他结点都仅有一个前趋结点，但允许有多个后继结点。从根结点到任一非根结点，都有且仅有一条路径。

非线形结构的最一般情形是“图”，图中任意结点的前趋结点和后继结点的个数都不受限制。对树和图的具体说明将分别在第六章和第七章给出。

学习数据结构的目的，就是使我们能够利用高级程序语言，在计算机存储器中通过一个数据结构的存储结构来实现其逻辑结构。

例 1.1 是一个数据结构的例子。该例子可使读者对数据结构有一个大体的概念。

例 1.1 通讯录。

姓名	区号	电话号码
赵一	010	53644587
钱二	020	89634159
孙三	021	45976528
李四	024	63427541

如上所示的通讯录表就是一个数据结构，其中，一行表示一个结点，每个结点由姓名、区号和电话号码等 3 个域组成。

分析该通讯录表的逻辑结构。由观察可知，表的第一行(“赵一”所在结点)是始结点，因为它没有前趋结点；最后一行(“李四”所在结点)是终结点，因为它没有后继结点；中间两行都是内结点，它们各有一个前趋结点和一个后继结点。显然，通讯录表的逻辑结构是线性结构。

该表可以有两种存储结构：(1) 顺序存储结构：把这 4 个结点连续地存放在存储器中，从而逻辑相邻的两个结点必物理相邻；(2) 链接存储结构：4 个结点在存储器中随意分别存放，结点之间的物理关系与逻辑关系无关，逻辑关系用附加的指针域来表示。我们将在第四章详细介绍顺序存储结构和链接存储结构。

对该表实施的操作：若通讯录表(简称为通讯录)主人结识新友，欲将新友信息(包括姓名、区号和电话号码)添入通讯录，则要对该表实施插入操作；若通讯录中的某人更新其电话号码，则要对该表实施修改操作；若主人要查找某友人的

区号及电话号码,则要对该表实施查找操作;等等。

结合例 1.1,可以给出数据结构的如下定义:(1) 按某种逻辑关系将一批数据元素组织起来;(2) 按一定的存储方式把它们存储起来;(3) 在数据上定义一个运算集合,就得到(或者说形成)了一个数据结构。

1.2 面向对象程序设计 OOP 与抽象数据类型 ADT

在传统的大型结构化程序中,一个数据结构可能被多个过程调用,修改该数据结构,意味着必须修改相关的过程,这样做烦琐而又容易产生错误。

利用面向对象程序设计(Object-Oriented Programming,简称 OOP)思想,可以直接解决这个问题。面向对象程序设计是建立在结构化程序设计基础上的,其最重要的改变是:程序围绕被操作的数据来设计,而不是操作本身。面向对象程序设计以"类"作为构造程序的基本单位,"类"具有封装、数据抽象、继承和多态性等特点。

目前应用最广泛的 OOP 语言是 C++ 语言.C++ 语言是对 C 语言的扩展,它继承了 C 语言高效、灵活等特点,完善了 C 语言的类型检查、代码重用、数据抽象机制,并且扩充了对面向对象程序设计的支持。

为了从面向对象的观点来研究数据结构,本书采用 C++ 语言来实现数据结构的基本数据类型。

根据面向对象程序设计的思想,可以将每种数据结构都视为一个抽象类型,它定义数据的组织方式,且给出数据上的基本操作,这种结构称为**抽象数据类型**(Abstract Data Type,简称 ADT)。ADT 是一种描述用户和数据之间接口的抽象模型,也就是说,ADT 给出了一种用户自定义的数据类型,并且定义了在该类型数据上的基本操作(操作内容与数据运算符的功能相似,指明用户应该如何操作这种自定义的数据)。因为 ADT 是一种与具体应用无关的抽象模型,所以程序员可以把注意力集中在数据及其操作的理想模型上。

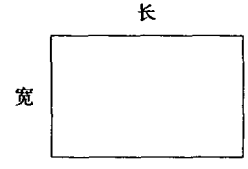
下面给出 ADT 描述的规范形式:

```
ADT Name is
  Data
    构成该抽象类型所必须的基本数据项
  Operations
    构造函数(声明一个该类型的对象时,对基本数据项进行初始化)
      Initial Values:      赋给基本数据项的值
```

```
        Process:        初始化对象
    操作 1
        Input:          操作 1 要求用户输入的值
        Preconditions:  系统执行操作 1 前数据所需的状态
        Process:        对数据执行操作 1
        Output:         操作 1 结束后返回的数据
        Postconditions: 执行操作 1 后数据的状态
    操作 2
        ...
    操作 n
        ...
end ADT Name
```

例 1.2 给出了矩形的 ADT 描述。众所周知,一个矩形必须具备长度和宽度,因此,矩形的抽象数据类型的基本数据项应该是矩形的长(Length)和宽(Width)。另外,对矩形的基本操作应该包括:修改矩形长度、修改矩形宽度、求矩形的周长、求矩形的面积。

例 1.2 矩形(Rectangle)的 ADT 描述。




```
ADT Rectangle is
  Data
    float Length
    float Width
  Operations
    Constructor
      Initial values:   构造矩形时赋给长和宽的初值
      Process:          给矩形的长和宽赋初值
    GiveLength
      Input:            赋给变量 Length 的新值
      Preconditions:    无
      Process:          将矩形的长度值修改为新值
      Output:           无
      Postconditions:   矩形长度值被修改
    GiveWidth
      Input:            赋给变量 Width 的新值
      Preconditions:    无
      Process:          将矩形的宽度值修改为新值
      Output:           无
      Postconditions:   矩形宽度值被修改
    GetArea
```

Input: 无
Preconditions: 无
Process: 计算矩形面积
Output: 返回矩形面积值
Postconditions: 无
GetPerimeter
Input: 无
Preconditions: 无
Process: 计算矩形周长
Output: 返回矩形周长值
Postconditions: 无
end ADT Rectangle

在第三章将详细介绍面向对象程序设计与抽象数据类型。本书所涉及到的基本数据类型,在相应章节中均会按照规范形式,给出其 ADT 描述。

1.3 算法概念和算法描述语言

让计算机完成解题任务,除了要选取适当的数据结构外,还需要制定出解决问题的切实可行的方法和步骤,这就是所谓的计算机算法。1967 年,D. E. 克努特(D. E. Knuth)在他的《计算机程序设计技巧》一书中指出,"计算机科学是有关算法的学问",并指出"对所有的计算机程序设计来说,算法的概念总是最基本的"。1969 年,C. A. R. 霍尔(C. A. R. Hoare)的论文"计算机程序设计公理化基础"和 1972 年 E.W. 戴克斯特拉(E. W. Dijkstra)的论文"结构程序设计札记",都集中研究了由程序正文所代表的算法结构。然而,对程序构造进行系统而科学的研究,首先是对包含复杂数据集的大型复杂程序而言的,因此,程序设计的方法学必然包含数据结构的所有方面。实际上,程序就是在数据的某些表示方式和结构的基础上对抽象算法的具体表达。霍尔通过"数据结构札记"一文明确阐明了"程序的构成与数据结构是不可分割地联系在一起的"。1976 年,N. 沃思(N. Wirth)旗帜鲜明地表达了这一观点,他用"算法 + 数据结构 = 程序"作为其专著的题目。

事实上,对于同一个问题,在选择了恰当的数据结构之后,不同的算法解决问题的好坏程度可能不一样,运算所需要的时间也可能有很大的差别。我们的目的是不断改进算法,尽快、尽好地解决问题。因此,讨论算法也是本书的重要内容。

毫无疑问,数据结构和算法两者是密切相连的,而程序则是用计算机可以接

受的语言(比如某种高级程序语言),在计算机上实现的算法。一个算法可以用自然语言、数学语言或约定的符号语言来描述,本书使用 ADL 语言和 C++语言交替进行算法描述,对于比较有代表性的算法,将用两种语言分别给出算法的描述。

下面给出用算法描述语言 ADL 书写算法时的约定和书写格式。

(1) 算法都有一个名字,有相关的输入与输出,具体规定如下:

算法〈标识符〉(变量 i_1,...,变量 i_m.变量 j_1,...,变量 j_n)

其中:〈标识符〉表示算法的名字,〈标识符〉是由字母和数字组成的有限字符串,且串中第一个符号必须是字母。在变量表中,变量 i_k 为输入变量,$m \geqslant 0$,当 $m=0$ 时,表示没有输入变量;变量 j_k 为输出变量,$n \geqslant 0$。

(2) 在算法的变量表后,用注释语句对整个算法进行概括说明。较短的注释语句(注释内容不超过一行)用符号"//"开头;较长的注释语句用符号"/*"开头,用符号"*/"结尾。

(3) 算法的每一步骤都要有步骤名(或称步骤标识符),步骤名由算法名(或算法名的缩写)后接数字组成;步骤名后紧接(不计空格)一对方括号,将该步骤所执行的操作用高度概括性文字表示在该方括号内。

(4) 对表达式而言:

① 算术表达式可使用常用的算术运算符,如 +,-,*,/,DIV,MOD,⌊⌋(取地板运算如⌊x⌋,其值是小于等于 x 的最大整数),⌈⌉(取天棚运算,如⌈x⌉,其值是大于等于 x 的最小整数),...

② 逻辑表达式可使用关系运算符(=,≠,<,>,≤,≥),逻辑运算符(AND,OR,NOT)和逻辑常量(true,false)。

③ 集合运算符有∩,∪,-(差),⊂,⊃,...

(5) 对语句而言:

① 每条语句都用"."作为结束符。

② 赋值语句的形式如下:

$$a \leftarrow b$$

其中 a 是变量,b 是表达式。$a \leftrightarrow b$ 表示将变量 a 和变量 b 的内容进行交换。$a \leftarrow b \leftarrow c$ 表示将 c 的值同时赋给变量 a 和变量 b。

③ 条件语句有如下 3 种:

- IF <逻辑表达式> THEN
  (语句 1.….语句 m)。
- IF <逻辑表达式> THEN
  (语句 1.….语句 m).
  ELSE