数据结构

DATA STRUCTURE

廖滨华 郭思培 唐向阳 编著

湖北长江出版集团
湖北科学技术出版社

数据结构

廖滨华
郭思培 编著
唐向阳

DATA
STRUCTURE

湖北长江出版集团
湖北科学技术出版社

图书在版编目(CIP)数据
数据结构/廖滨华等编著.—武汉：湖北科学技术出版社，2006.4
ISBN 7-5352-1398-7

Ⅰ.数… Ⅱ.廖… Ⅲ.数据结构—高等学校—教材 Ⅳ.TP311.12

中国版本图书馆CIP数据核字(2006)第.018971号

数据结构

策　　划：
责任编辑：高诚毅 梁 琼
封面设计：王 梅

出版发行：湖北长江出版集团
湖北科学技术出版社
电话：87679468

地　　址：武汉市雄楚大街268号湖北出版文化城
B座12-13层
邮编：430070

印　　刷：武汉市教育学会印刷厂
邮编：430074

787毫米×1092毫米　16开　11.75印张　180千字

2006年4月第1版　2006年4月第1次印刷

ISBN 7-5352-1398-7/TP·84
定价：20.00元

前　言

随着计算机科学技术的快速发展，计算机的用途已不再局限于科学计算，而是扩展到了非数值计算领域，更多地用于控制、管理及数据处理等非数值计算的处理工作。计算机处理加工的对象由纯粹的数值发展到字符、表格、图像等各种具有一定结构的数据。非数值处理问题的特点是数据元素间的关系复杂，因此必须分析待处理的数据对象的特性以及各处理对象之间存在的关系，才能编写出一个好的程序。

《数据结构》作为一门综合性的专业基础课，是介于数学、计算机软件、计算机硬件三者之间的一门核心课程。所以，数据结构不仅是计算机专业教学计划中的核心课程，而且也是计算机应用有关的专业重要的必修课或选修课。

本书共分为 9 章，详细讨论各种基本类型的数据结构及其应用，算法基本采用 C/C++对照。本书的特点是面向应用，每一章基本上有一个完整的应用例子，可以让学生对所学的知识有一个形象的认识。所有的算法应能在 visual C++上调试通过。

本书在编写过程中得到了华中师范大学数学与统计学学院的大力支持，在此表示衷心的感谢。

本书可作为计算机类专业或信息类相关专业的本科或专科教材，也可供从事计算机工程与应用工作的科技工作者参考。

本书由廖滨华、郭思培、唐向阳共同编写，署名按章节顺序。

由于作者水平有限，缺点和错误在所难免，恳请读者批评指正。

感谢阅读本书的读者！

作者

目　录

第一章 绪 论

第一节 数据结构学科形成和发展的背景

随着计算机科学技术的快速发展，计算机的用途已不再局限于科学计算，它扩展到了非数值计算领域，更多地用于控制、管理及数据处理等非数值计算的处理工作。

计算机处理加工的对象由纯粹的数值发展到字符、表格、图像等各种具有一定结构的数据。非数值处理问题的特点是数据元素间的关系复杂，一般具有一定结构，而计算简单，因此必须分析待处理的数据对象的特性以及各处理对象之间存在的关系才能编写出一个好的程序。

用计算机解题，首先要把客观对象抽象为某种形式的数据，然后设计对这些数据进行操作的算法，计算机执行这些算法，输出问题的解答。我们可以看到解题过程要解决两个问题：数据结构的设计和算法的设计。即“算法＋数据结构＝程序”。数据结构和算法是程序设计过程相辅构成、不可分割的两个方面。

《数据结构》作为一门综合性的专业基础课，是介于数学、计算机软件、计算机硬件三者之间的一门核心课程。所以，数据结构不仅是计算机专业教学计划中的核心课程，而且也是计算机应用有关的专业重要的必修课或选修课。了解数据结构的学科背景可以帮助我们更好地学习这门课。

在 20 世纪 60 年代，数据结构并没有作为独立的一门课程来开设，它的大部分内容散布在表处理语言、图论、离散数学结构等教材中。在 60 年代末到 70 年代初，出现了大型的程序，软件也相对独立，人们也越来越重视数据结构，认为程序的实质是对确定的问题选择一种好的结构，再加上设计一种好的算法。1968 年美国 Donald E. knuth 出版了专著《计算机程序设计技巧》第 1 卷《基本算法》，首次系统地阐述数据结构的主要内容，即数据的逻辑结构、存储结构以及对数据进行各种操作的算法。到 20 世纪 70 年代中期和 20 世纪 80 年代初，各种版本的数据结构著作就相继出现，我国也于 20 世纪 70 年代末开设了数据结构课程。

现在对数据结构的研究一方面侧重从抽象数据类型的观点来讨论数据结构，另一方面注重专门领域中的特殊问题（如多维图形数据结构）。现在有关数据结构的知识已成为设计和实现编译系统、操作系统、数据库管理系统

及其他系统程序和一些应用系统的重要基础。

第二节 程序设计方法及语言

在计算机领域，计算机语言不同，对数据进行分类的规则也不同，如果我们使用实际应用领域中经常使用的程序设计语言来描述数据结构和算法，使得教材中的许多实例可以成为在计算机上运行的可被众多程序员调用的标准程序，将会更有实用价值。C语言是普遍流行的程序设计语言，系统程序员和应用程序员都经常采用C语言作为编程工具。同时，随着面向对象技术的成熟，C++已为越来越多的人使用。C++语言不仅提供丰富的描述数据结构和算法的能力，使用C++作为数据结构的描述语言，将有助于学生在学习数据结构的过程中掌握面向对象的概念、方法、实现技术，达到学以致用的目的。所以本教材采用C和C++对照算法，既可在提高学生的编程能力的同时增加程序的实用性，又可让学生认识到算法和具体语言的无关性。

数据结构是一门实践性很强的课程，掌握本课程的概念和方法必须在计算机上进行足够时间的实习。

第三节 基本概念及术语

一、数据(Data)

在计算机科学中，一切能够输入到计算机中并被计算机程序处理的符号集合，包括文字、表格、图像等，都称为数据。数据是需要经过计算机加工处理的，它是计算机处理的“原料”。信息则是经计算机加工处理以后产生的有意义的数据。

例如：一个学籍管理程序所要处理的数据可能是一张如表1-1所示的学生登记表。

表1-1 学生登记表

学号	姓名	性别	进校时间	出生日期
2003001	孙红	女	2003年	1984年
2003002	李为	男	2003年	1982年
2003003	陈强	男	2003年	1984年
2003004	张刚	男	2003年	1983年
2003005	吴卓红	女	2003年	1984年
2003006	王一生	男	2003年	1984年

二、数据元素

数据元素是组成数据的基本单位。在程序中通常把一个数据元素作为一个整体进行考虑和处理。一般情况下，一个数据元素中含有若干个字段（也叫数据项）。

例如，在表 1-1 的学生登记表中，每一行（每一个人）是一个数据的基本单位，也称为一个数据元素，因此学籍数据由 6 个数据元素构成。每个数据元素都由学号、姓名、性别、进校时间、出生日期等 5 个数据项构成。数据项是构成数据的最小单位。

三、逻辑结构

逻辑结构描述结点和结点之间的逻辑关系。

例如，在表 1-1 的学生登记表中，各数据元素之间在逻辑上有一种线性关系，它指出了 6 个数据元素在表中的排列顺序。

四、存储结构

存储结构是指数据在计算机中的存储结构。

例如，在表 1-1 的学生登记表中的数据在计算机中可以有多种存储表示：可以表示成数组，存放在内存中；也可以表示成文件，存放到磁盘上，等等。

五、数据结构

数据结构是指按某种逻辑关系组织起来的一组结点，按一定的存储方式存储于计算机中，并在其上定义了一个运算的集合。数据结构是组织和访问数据的系统方法。具体来说，数据结构包含三个方面的内容，即数据的逻辑结构，数据的存贮结构和对数据所施加的运算。这三个方面的关系为：

(1) 数据的逻辑结构独立于计算机，是数据本身所固有的。

(2) 存贮结构是逻辑结构在计算机存贮器中的映像，必须依赖于计算机。

(3) 运算是指所施加的一组操作总称。运算的定义直接依赖于逻辑结构，但运算的实现必依赖于存贮结构。

数据结构描述了计算机处理的数据元素的组织形式和相互之间的关系。

从数据元素间的不同特性分 { 线性结构 / 树形结构 / 图形结构

从不同抽象层次上看
- 逻辑数据结构
- 物理数据结构（或存储结构）
  - 顺序存储结构
  - 链式存储结构

六、数据类型

数据类型是和数据结构密切相关的一个概念，最早出现在高级程序语言中，用以描述操作对象的特性，每一个变量、常量或表达式都有一个它所属的确定的数据类型。数据类型是在程序设计语言中已经实现了的数据结构。在程序设计语言中，每一个数据都属于某种数据类型。类型明显或隐含地规定了数据的取值范围、存储方式以及允许进行的运算。在程序设计过程中，当需要引入某种新的数据结构时，必须借助编程语言所提供的数据类型来描述数据的存储结构。数据类型是一个值的集合和定义在这个值集上的一组操作的总称。如：C语言中的整型变量，其值集为某个区间上整数（区间的大小依赖于不同的机器），定义在其上的操作为：加、减、乘、除和取模等算术运算。

从用户的角度来看：数据类型是某种程序设计语言中已经实现了的数据结构，将用户不必了解的细节（如整数在计算机内部是如何表示，操作是如何实现）封装在数据结构中，从而实现了数据的抽象。

由于程序设计语言将这些基本数据结构作为其定义的一部分，因此，它们不在我们的学习范围之内。

七、算法

简单来说，算法是解决特定问题的方法（步骤）。一般来说，特定的问题可分为数值的和非数值的两类。解决数值问题的算法叫做数值算法，如工程计算和科学计算等，主要进行算术运算。数据处理方面的算法都属于非数值算法，如排序、查找插入等，主要进行比较和逻辑计算。

在计算机领域，一个算法实际上是针对所处理问题的需要，在数据的逻辑结构和存储结构上施加的一种运算。由于数据的逻辑结构和存储结构不是惟一的，在很大程度上可以由用户自行选择和设计，所以处理同一个问题的算法也不是惟一的。另外，即使对于具有相同的逻辑结构和存储结构而言，其算法的设计思想和技巧不同，编写出的算法也大不相同。我们学习数据结构这门课的目的，就是要会根据数据处理问题的需要，为待处理的数据选择合适的逻辑结构和存储结构，进而设计出比较满意的算法。

第四节 抽象数据类型

Abtract Data Type(ADT)是指一个数学模型以及在该模型上的一组操作。一方面,抽象数据类型和数据类型在实质上是一个概念,例如,int 或 char 这些基本类型,明显或隐含地规定了数据的取值范围、存储方式以及允许进行的运算,并且在不同处理器上的实现方法也可以不同,但由于它们的数学特性相同,在用户看来都是相同的。更进一步,抽象数据类型的范畴更广,不仅包括已经实现的数据类型,而且还包括用户在设计软件系统时自己定义的数据类型。

抽象的数据类型可以通过固有的数据类型来表示和实现,即利用已有的数据类型来说明新的结构,用已经实现的操作来组合新的操作。

抽象数据类型是用户在数据类型的基础上新定义的数据类型,也包括数据组成和数据处理。抽象数据类型的定义包括数据对象的定义,数据关系的定义,基本操作的定义。

抽象数据类型的定义的格式如下所示:

ADT:抽象数据类型名

数据对象:数据对象的定义(在已有的数据类型或已定义的数据对象上对新对象定义)

数据关系:数据逻辑关系的定义

基本操作:基本操作的定义(包括操作名、参数表、初始条件、操作结果的定义和描述)

其中基本操作的定义的格式如下:

操作名(参数表)

操作结果:操作结果描述

由于本书是用 C/C++对照算法来描述算法,故均采用结构化的程序设计方法(主要利用函数)来描述算法的实现,这样使得数据结构的算法的描述和讨论简明清晰,而且也很容易转化为面向对象的程序设计方法,同时让学生了解到算法和程序设计语言和程序设计方法的无关性。

第五节 数据的逻辑结构

一、数据的逻辑结构的分类

数据的逻辑结构可分为线性结构和非线性结构两大类。

1. 线性结构

线性结构的开始结点和终端结点都是惟一的，除了开始结点和终端结点以外，其余结点都有且仅有一个前驱，有且仅有一个后继。

2. 非线性结构

非线性结构又可以细分为树形结构和图状结构两类。

(1) 树形结构

树形结构的每个结点最多只有一个前驱，但可以有多个后继，可以有多个终端结点。

(2) 图状结构

图状结构的每个结点的前驱和后继的数目都可以是任意的。可能没有开始结点和终端结点，也可能有多个开始结点、多个终端结点。

二、数据的逻辑结构的表示方法

数据的逻辑结构的表示方法有图形表示法、公式表示法、自然语言描述等很多种，可以根据需要来进行选择。下面介绍几种常见的表示方法：

1. 二元组表示法

二元组 S=(D,R)；其中：D 表示结点的有限集合，R 表示结点有序对的集合，S 表示 D 上的一种关系。

例如：线性表 list 的逻辑表示为 list=(D,R)；其中 D={a,b,c,d,e}；R={⟨a,b⟩,⟨b,c⟩,⟨c,d⟩,⟨d,e⟩}

2. 图示法

图示法用小圆圈表示结点，在圈内或圈外附近标注结点名称或数值，用带箭头的弧线表示结点之间的关系。

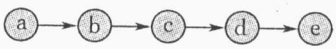

图 1-1 线性表 list 的逻辑结构

对于任意一个逻辑结构 S=(D,R)，若 a∈D，b ∈ D，⟨a,b⟩ ∈ R，则称 a 是 b 的前驱，b 是 a 的后继；若某结点没有前驱，则称该结点为开始结点；若某结点没有后继，则称该结点为终端结点。例如：在 list 结构中，a 是开始结点，e 是终端结点。

第六节 数据的存储结构

把数据存储到计算机中时，不仅要求存储各结点的数值，而且要存储结点与结点之间的逻辑关系。下面将介绍四种基本的存储结构：顺序存储、链接存储、索引存储和散列存储。

1. 顺序存储

顺序存储是把逻辑上相邻的结点存储在一组连续的存储单元中，其结点之间的逻辑关系由存储单元地址间的关系隐含表示。

例如：假定 list 结构每个结点占用一个存储单元，数据从 100 号单元开始由低地址向高地址方向存放。

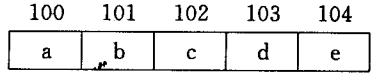

图 1-2 线性结构 list 的顺序存储结构

顺序存储有如下优点：

(1) 节省存储空间

分配给数据的存储单元全用于存放结点的数值，结点之间的逻辑关系没有占用额外的存储空间。

(2) 随机访问

存储线性结构的结点时，可实现对各结点的随机访问。——线性结构中每个结点都对应一个序号，可以根据结点的序号 i 计算出结点的存储地址：

$$Loc(i)=q+(i-1)\times p$$

其中 p 为每个结点所占单元数；q 是第一个结点所占单元的首地址。

顺序存储的缺点是不便于修改，在对结点进行插入、删除运算时要移动一系列的结点。例如，要删除 list 结构中的第 3 个结点 c，使 list 结构由(a,b,c,d,e)变成(a,b,d,e)。删除操作后要移动结点以保证剩余结点仍然占用一片连续的存储单元，具体的移动方法如图 1-3 和图 1-4 所示。

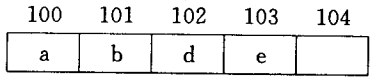

图 1-3 后面的结点前移

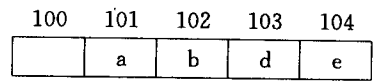

图 1-4 前面的结点后移

2. 链接存储

链接存储是给每个结点附加指针字段，用于存放相邻结点的存储地址。例如，假定给实例 list 中的每个结点附加一个“后继指针”字段，用于存放后继结点的首地址，则可得到如图所示的 list 的链接存储表示。

100	a	106
102	c	108
104	e	∧
106	b	102
108	d	104

图 1-5 list 的链接存储

链接存储的主要优点是便于修改，在进行插入、删除运算时，仅需修改结

点的指针字段值，不必移动结点。例如，在 list 的链接存储中实现删除操作和插入操作如图 1-6 和图 1-7 所示。

100	a	102
102	c	108
104	e	∧
106		
108	d	104

图 1-6 删除结点 b 后的情况

90	×	106
	⋯	
100	a	90
102	c	108
104	e	
106	b	102
108	d	104

图 1-7 在 a 后插入结点 x 的情况

链接存储的主要缺点有如下两点：

(1) 分配给数据的存储单元有一部分被用来存放结点之间的逻辑关系，存储空间的利用率较低。

(2) 逻辑上相邻的结点在存储器中不一定相邻，用这种方法存储的线性结构不能对结点进行随机访问。

3. *索引存储*

索引存储是根据结点的序号确定结点的存储地址。具体做法是首先建立索引表和结点表，然后将各结点的数值按任意顺序存放在结点表中，再将每个结点的存储地址(即在结点表中的位置)用顺序存储法存入索引表中。对于线性结构来说，各结点的存储地址在索引表中是按结点的序号依次排列的。例如 list 的索引存储：

10	100
11	104
12	102
13	101
14	103

索引表

100	a
101	d
102	c
103	e
104	b

结点表

图 1-8 list 的索引存储

索引存储有如下优点：

(1) 线性结构采用索引存储后，可以对结点进行随机访问。

(2) 在进行插入、删除运算时，由于只需移动存储在索引表中的结点的存储地址，而不必移动存储在结点表中的结点的数值，所以仍可保持较高的