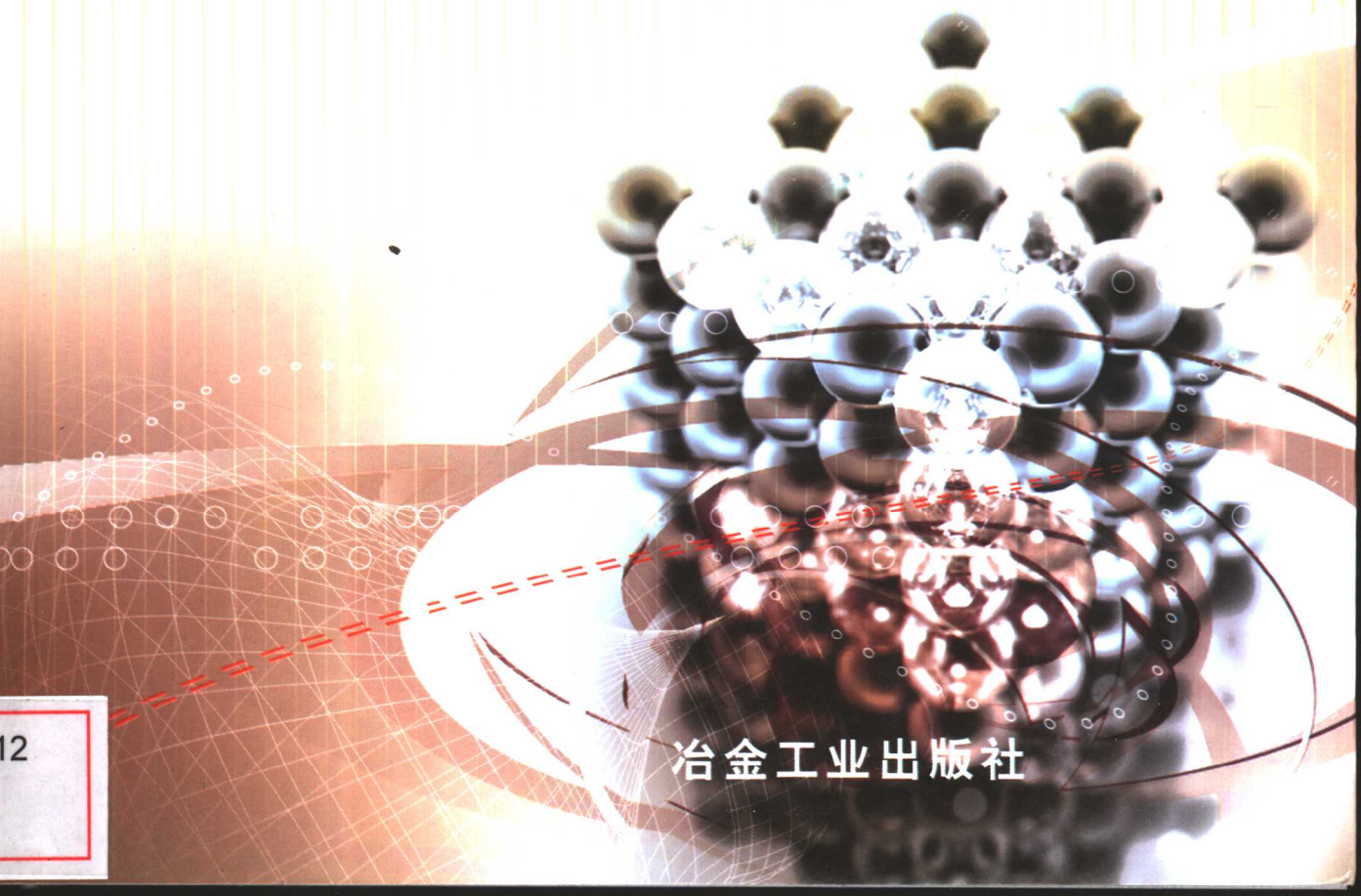
高等学校21世纪计算机教材

数据结构与算法

李乔祥　编著



冶金工业出版社

高等学校21世纪计算机教材

数据结构与算法

李乔祥　编著

北　京

冶金工业出版社

2004

内 容 简 介

“数据结构与算法”是一门与计算机联系较密切的学科。随着近年来计算机技术突飞猛进地发展，“数据结构”在计算机科学中的地位也越来越重要。“数据结构与算法”课程是计算机专业的一门核心课程，也是很多高校招收计算机专业研究生考试的科目之一。

本书内容广泛，主要介绍了数据结构的相关知识、类C语言基础知识、算法基础、顺序表、链表、串、多维数组与广义表、树和森林、查找、图、排序、文件以及算法设计基本方法等内容。本书在编写过程中尽量把数据结构的各个部分有机地结合起来，力求做到内容深入浅出、通俗易懂、条理清晰。

本书既可作为计算机专业的本科或专科教材，也可作为信息类相关专业的选修教材，还可作为从事计算机应用相关工作科技人员的参考书。

图书在版编目（CIP）数据

数据结构与算法 / 李乔祥编著. —北京：冶金工业出版社，2004.4
ISBN 7-5024-3507-7

I. 数… II. 李… III. ①数据结构②算法分析 IV. TP311.12

中国版本图书馆 CIP 数据核字（2004）第 026345 号

出版人　曹胜利（北京沙滩嵩祝院北巷 39 号，邮编 100009）
责任编辑　戈兰
佛山市新粤中印刷有限公司印刷；冶金工业出版社发行；各地新华书店经销
2004 年 5 月第 1 版，2004 年 5 月第 1 次印刷
787mm × 1092mm　1/16；　18.25 印张；　421 千字；　284 页；　1-5000 册
30.00 元
冶金工业出版社发行部　**电话**：（010）64044283　**传真**：（010）64027893
冶金书店　**地址**：北京东四西大街 46 号（100711）　**电话**：（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于本书

随着计算机软件和硬件的发展，计算机的应用已经深入到社会的各个领域，各行各业都需要对大量的非数值数据进行存储、加工和管理。如何根据实际应用的要求，对这些大量的表面上杂乱无章的数据进行有效地组织、存储和处理，编制出相应的高效率算法，这就是“数据结构与算法”这门课程所要研究并加以解决的问题。因此“数据结构与算法”是计算机程序设计的技术基础，它是计算机科学和工程系各专业的核心课程，为计算机专业技术人员提供必要的专业基础知识和技能训练，同时也是计算机应用相关学科所必须掌握的课程。

“数据结构与算法”是一门专业技术基础课，通过该课程的学习，学生能熟练掌握计算机程序设计中常见的各种数据的逻辑结构、存储结构及相应的运算，初步掌握算法的时间分析和空间分析的技术，并能根据计算机加工的数据特性运用数据结构的知识和技巧设计出更好的算法和程序。本课程的学习过程也是复杂程序设计的训练过程，要求学生编写的程序结构清晰明确、通俗易懂，符合软件工程的规范，从而提高学生的软件设计水平。

二、本书结构

本书共分为十三章，每章内容如下：

第 1 章：概论。主要介绍了什么是数据结构、数据的逻辑结构、数据的类型、数据的存储结构、数据的运算及数据的分析和性能评价。

第 2 章：类 C 语言基础知识。主要介绍了 C 语言相关知识、C 语言数据类型、C 语言运算符与表达式、C 语言函数、C 语言常用语句及综合应用实例。

第 3 章：算法基础。主要介绍了递归法、穷举法、迭代法、递推法、分治法及逐步求精法。

第 4 章：顺序表。主要介绍了向量、栈、栈的应用、队列、利用队列打印二次展开式的算法及限制存取点的表。

第 5 章：链表。主要介绍了单链表、栈的链表表示、队列的链表表示、特殊的链表及一元多项式的链表表示及其运算。

第 6 章：串。主要介绍了串的概念、串的运算实现、字符串的模式匹配及 KMP 算法。

第 7 章：多维数组与广义表。主要介绍了多维数组的定义、多维数组的逻辑结构、多维数组的顺序存储、矩阵的压缩存储及广义表。

第 8 章：树和森林。主要介绍了树的定义和基本术语、二叉树、遍历二叉树、线索二叉树、树的存储结构、森林和二叉树的转换、树和森林的遍历及赫夫曼树及其应用。

第 9 章：查找。主要介绍了基本概念、线性表的查找、树表的查找及散列表的查找。

第 10 章：图。主要介绍了图的定义和术语、图的存储结构、图的遍历、图的连通性、图的最小生成树、拓扑排序、关键路径（图）及最短路径。

第 11 章：排序。主要介绍了排序的相关知识、插入排序、交换排序、选择排序、归并排序、基数排序及外排序。

第 12 章：文件。主要介绍了文件基本概念、顺序文件、索引文件、Hash 文件及多关键字文件。

第 13 章：算法设计基本方法。主要介绍了回溯与界限剪枝法、贪心法及动态规划法。

三、本书特点

全书采用类 C 语言作为数据结构和算法的描述语言。本书文字简练、内容循序渐进、例题丰富、便于自学。本书各章末均配有较多的习题，书末还附有参考答案，但所提供的算法不一定是最佳的，仅供读者参考。

四、适用对象

本书既可作为计算机专业的本科或专科教材，也可作为信息类相关专业的选修教材，还可作为从事计算机应用相关工作科技人员的参考书。

参加本书编写工作的还有甘永辉、汤庆恩、苏秋斌、潘嘉林、叶剑文、许斯良、黄德志、邱凌苍、冯家浩，在此对他们的工作表示感谢。

由于编写时间仓促、编者水平有限，书中不足之处在所难免，恳请广大读者批评指正。

虽然经过严格的审核、精细的编辑，本书在质量上有了一定的保障，但我们的目标是力求尽善尽美，欢迎广大读者和专家对我们的工作提出宝贵建议，联系方法如下：

电子邮件：**service@cnbook.net**

网址：**www.cnbook.net**

此外，该网站还有一些其他相关书籍的介绍，可以方便读者选购参考，本书所附电子教案也可从该网站免费下载。

编　者

2004 年 2 月

目　录

第1章 概 论

自 1946 年第一台计算机问世以来，计算机产业的飞速发展已远远超出人们对它的预料，在某些生产线上，甚至几秒钟就能生产出一台微型计算机，从而产量猛增、价格低廉，这就使得它的应用范围迅速扩展。如今，计算机已深入到人类社会的各个领域。计算机的应用已不再局限于科学计算，而更多地用于控制、管理及数据处理等非数值计算领域。与此相应，计算机加工处理的对象由纯粹的数值发展到字符、表格和图像等各种具有一定结构的数据，这就给程序设计带来一些新的问题。为了编写出一个“好”的程序，必须分析待处理的对象的特性以及各处理对象之间的关系。这就是“数据结构”这门学科形成和发展的背景。

本章中将介绍有关数据和数据结构的概念，简述数据结构在计算机科学中的重要性，并扼要说明本课程的内容。第一节介绍数据结构，第二、三、四节是本章的重点内容，第五节学习对数据结构的分析与性能评价。本章可以作为全书的导引，但有些概念可能在读完全书后才会有较深刻地理解。本章主要内容如下：

（1）什么是数据结构。

（2）数据的逻辑结构。

（3）数据的类型。

（4）数据的存储结构。

（5）数据的运算。

（6）数据的分析和性能评价。

1.1 什么是数据结构

一般来说，用计算机解决一个具体问题时，大致需要经过下列几个步骤：首先要从具体问题中抽象出一个适当的数学模型，然后设计一个解此数学模型的算法，最后编出程序、进行测试、调整直至得到最终解答。寻求数学模型的实质是分析问题，从中提取操作的对象，并找出这些操作对象之间含有的关系，然后用数学的语言加以描述。

数据结构就是指数据（对象）之间的关系。但关于数据结构的概念，至今并没有一个公认的标准定义，不过在谈到任何一种结构时，都自然地联系到对这种类型数据所需要的运算以及为了在计算机上执行这些运算时需要把这些数据如何存储在计算机中。所以数据结构概念一般包括：数据之间的逻辑关系，数据在计算机中的存储方式和数据运算三个方面。

为了叙述上的方便，把上述数据结构的三个方面分别称为：数据的逻辑结构、数据的存储结构和数据的运算。下面请看两个例子。

【例 1-1】图书馆的书目检索系统自动化问题。当你想借阅一本参考书但不知道书库中是否有的时候；或者，当你想找某一方面的参考书而不知图书馆内有哪些这方面的书时，你都需要到图书馆去查阅图书目录卡片。在图书馆内有各种名目的卡片：有按书名编排的、

有按作者编排的、还有按分类编排的等等。若利用计算机实现自动检索，则计算机处理的对象便是这些目录卡片上的书目信息。列在一张卡片上的一本书的书目信息可由登录号、书名、作者名、分类号、出版单位和出版时间等若干项组成，每一本书都有惟一的一个登录号，但不同的书目之间可能有相同的书名、或者有相同的作者名和分类号。由此，在书目自动检索系统中可以建立一张按登录号顺序排列的书目文件和三张分别按书名、作者名和分类号顺序排列的索引表，如图 1-1 所示。由这四张表构成的文件便是书目自动检索的数学模型，计算机的主要操作便是按照某个特定要求（如给定书名）对书目文件进行查询。诸如此类的还有查号系统自动化、仓库账目管理等。在这类文档管理的数学模型中，计算机处理的对象之间通常存在的是一种最简单的线性关系，这类数学模型可称为线性的数据结构。

001	高等数学	樊映川	S01	…
002	理论力学	罗远祥	L01	…
003	高等数学	华罗庚	S01	…
004	线性代数	栾汝书	S02	…
⋮	⋮	⋮	⋮	⋮

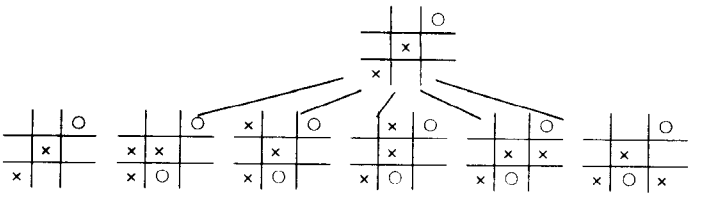
高等数学	001，003，…
理论力学	002，…
线性代数	004，…
⋮	

樊映川	001，…
华罗庚	003，…
栾汝书	004，…
⋮	

L	002，…
S	001，003，…
⋮	

图 1-1 图书目录文件示例

【例 1-2】计算机和人对弈问题。计算机之所以能和人对弈是因为人将对弈的策略事先已存入计算机。由于对弈的过程是在一定规则下随机进行的，所以，为使计算机能灵活对弈就必须对对弈过程中所有可能发生的情况以及相应的对策都考虑周全，并且一个“好”的棋手在对弈时不仅要看棋盘当时的状态，还应能预测棋局发展的趋势，甚至最后结局。因此，在对弈问题中，计算机操作的对象是对弈过程中可能出现的棋盘状态——称为格局。例如图 1-2（a）所示为井字棋的一个格局，而格局之间的关系是由比赛规则决定的。通常，这个关系不是线性的，因为从一个棋盘格局可以派生出几个格局，例如从图 1-2（a）所示的格局可以派生出五个格局，如图 1-2（b）所示，而从每一个新的格局又可派生出四个可能出现的格局。因此，若将从对弈开始到结束的过程中所有可能出现的格局都画在一张图上，则可得到一棵倒立生长的“树”。“树根”是对弈开始之前的棋盘格局，而所谓的“叶子”就是可能出现的结局，对弈的过程就是从树根沿树叉到某个叶子的过程。“树”可以是某些非数值计算问题的数学模型，它也是一种数据结构。



（a）棋盘格局示例　　（b）对弈树的局部

图 1-2 井字棋对弈“树”

以上两个例子，描述这类非数值计算问题的数学模型不再是数学方程式，而是诸如表、

树和图之类的数据结构。

例如，一个线性表，它的逻辑结构是指对下列一些问题的回答：哪一个元素是表中的第一个元素？哪一个元素是表中的最后一个元素？哪些元素在一个给定元素之前或之后？它的存储结构是指它的元素在存储器中是顺序地连接存放还是用指针连在一起的？数据运算可以包括：在表中找一个元素，从表中删去一个元素，向表中插入一个元素等等。希望读者学习时，不要孤立地去理解一个方面，而要注意它们之间的联系。

综上所述，按某种逻辑关系组织起来的一批数据，按一定的存储表示方式把它存储在计算机的存储器中，并在这些数据上定义了一个运算的集合，就叫做一个数据结构。

“数据结构”在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及到计算机硬件（特别是编码理论、存储装置和存取方法等）的研究范围，而且和计算机软件的研究有着密切的关系，无论是编译程序还是操作系统，都涉及到数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据，以便查找和存取数据元素更为方便。因此，可以认为数据结构是介于数学、计算机硬件和计算机软件三者之间的一门核心课程。在计算机科学中，数据结构不仅是一般程序设计（特别是非数值计算的程序设计）的基础，而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的重要基础。

值得注意的是，数据结构发展并未终结，一方面，面向各专业领域中特殊问题的数据结构得到研究和发展，如多维图形数据结构等；另一方面，从抽象数据类型的观点来讨论数据结构，已成为一种新的趋势，越来越被人们所重视。

1.2 数据的逻辑结构

对数据间关系的描述是数据的逻辑结构，形式上可以用一个二元组表示：

$$\text{Data\_Structure} = (D, S) \tag{1-1}$$

其中，D 是数据元素的有限集，S 是 D 上的关系有限集。下面举两个简单例子说明。

【例 1-3】在计算机科学中，复数可取如下定义：

$$\text{Complex} = (C, R) \tag{1-2}$$

其中，C 是含两个实数的集合{C1，C2}、R = {P}，而 P 是定义在集合 C 上的一种关系{〈C1，C2〉}，其中有序偶〈C1，C2〉表示 C1 是复数的实部，C2 是复数的虚部。

【例 1-4】假设需要编制一个事务管理的程序，管理学校科学研究课题小组的各项事物，则首先要为程序的操作对象——课题小组设计一个数据结构。假设每个小组有一位教师、一至三名研究生及一至六名本科生组成，小组成员之间的关系是：教师指导研究生，而由每位研究生指导一至两名本科生。则可以如下定义数据结构：

$$\text{Group} = (P, R) \tag{1-3}$$

其中：

$P = \{T, G_1, \cdots, Gn, S_{11}\cdots Sn\ m \mid 1\leqslant n\leqslant 3,\ 1\leqslant m\leqslant 2\}$，

$R = \{R1, R2\}$

$R1 = \{\langle T, Gi\rangle \mid 1\leqslant i\leqslant n,\ 1\leqslant n\leqslant 3\}$

$R2 = \{\langle Gi, Sij\rangle \mid 1\leqslant i\leqslant n,\ 1\leqslant j\leqslant m,\ 1\leqslant n\leqslant 3,\ 1\leqslant m\leqslant 2\}$

上述数据结构的定义仅是对操作对象的一种数学描述，换句话说，是从操作对象抽象出来的数学模型。在下面的叙述中，凡不易产生混淆之处，有时就把数据的逻辑结构简称为数据结构。

1.3 数据的类型

数据类型（Data Type）是和数据结构密切相关的一个概念，它最早出现在高级程序语言中，用以刻画（程序）操作对象的特性。在用高级程序语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。类型明显或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些值上允许进行的操作。因此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。例如，C 语言中的整型变量，其值集为某个区间上的整数（区间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。

按“值”的不同特性，高级程序语言中的数据类型可分为两类：一类是非结构的原子类型。原子类型的值是不可分解的。如：C 语言中的基本类型（整型、实型、字符型和枚举类型）、指针类型和空类型。另一类是结构类型。结构类型的值是由若干成分按某种结构组成的，因此是可以分解的，并且它的成分可以是非结构的。例如数组的值由若干个分量组成，每个分量可以是整数，也可以是数组等。在某种意义上，数据结构可以看成是“一组具有相同结构的值”，而数据类型可以看成由一种数据结构和定义在其上的一组操作组成。

实际上，在计算机中，数据类型的概念并非局限于高级语言中，每个处理器（包括计算机硬件系统、操作系统、高级语言、数据库等）都提供了一组原子类型或结构类型。例如，一个计算机硬件系统通常含有“位”、“字节”、“字”等原子类型，它们的操作通过计算机设计的一套指令系统直接由电路系统完成；而高级程序语言提供的数据类型，其操作需通过编译器或解释器转化成低层即汇编语言或机器语言的数据类型来实现。引入“数据类型”的目的，从硬件的角度看，是作为解释计算机内存中信息含义的一种手段，而对使用数据类型的用户来说，实现了信息的隐蔽，即将一切用户不必了解的细节都封装在类型中。例如，用户在使用“整数”类型时，既不需要了解“整数”在计算机内部是如何表示的，也不需要知道其操作是如何实现的。如“两整数求和”，程序设计者注重的仅仅是其“数学上求和”的抽象特性，而不是其硬件的“位”操作如何进行。

抽象数据类型（Abstract Data Type 简称 ADT）是指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

一方面，抽象数据类型和数据类型实质上是一样的。例如，各个计算机都拥有的“整数”类型是一个抽象数据类型，尽管它们在不同处理器上实现的方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。

另一方面，抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的

数据类型（也可称这类数据类型为固有数据类型），还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的复用率，在近代程序设计方法学中指出，一个软件系统的框架应建立在数据之上，而不是建立在操作之上（后者是传统的软件设计方法所为）。即在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作，并在模块内部给出这些数据的表示及其操作的细节，而在模块外部使用的只是抽象的数据和抽象的操作。显然，所定义的数据类型的抽象层次越高，含有该抽象数据类型的软件模块的复用程度也就越高。

一个含抽象数据类型的软件模块通常应包含定义、表示和实现三个部分。

如前所述，抽象数据类型的定义由一个值域和定义在该值域上的一组操作组成。若按其值的不同特性，可细分为下列三种类型：

（1）原子类型（Atomic Data Type）。属原子类型的变量的值是不可分解的。这类抽象数据类型较少，因为一般情况下，已有的固有数据类型足以满足需求。但有时也有必要定义新的原子数据类型，例如数位为100的整数。

（2）固有聚合类型（Fixed - Aggregate Data Type）。属该类型的变量，其值由确定数目的成分按某种结构组成。例如，复数是由两个实数依确定的次序关系构成。

（3）可变聚合类型（Variable - Aggregate Data Type）。和固定聚合类型相比较，构成可变聚合类型“值”的成分的数目不确定。例如，可定义一个“有序整数序列”的抽象数据类型，其中序列的长度是可变的。显然后两种类型可统称为结构类型。

和数据结构的形式定义相对应，抽象数据类型可用以下三元组表示：

$$(D,\ S,\ P) \tag{1-4}$$

其中，D是数据对象，S是D上的关系集，P是对D的基本操作集。抽象数据类型定义的格式为：

```
ADT 抽象数据类型名{
    数据对象:〈数据对象的定义〉
    数据关系:〈数据关系的定义〉
    基本操作:〈基本操作的定义〉
} ADT 抽象数据类型名
```

其中，数据对象和数据关系的定义用伪代码描述，基本操作的定义格式为：

```
基本操作名（参数表）
    {初始条件:〈初始条件描述〉
     操作结果:〈操作结果描述〉
    }
```

基本操作有两种参数，赋值参数只为操作提供输入值；引用参数以&打头，除可提供输入值外，还将返回操作结果。“初始条件”描述了操作执行之前数据结构和参数应满足的条件，若不满足，则操作失败，并返回相应出错信息。若初始条件为空，则省略之。“操作结果”说明了操作正常完成之后，数据结构的变化状况和应返回的结果。

多形数据类型（Polymorphic Data Type）是指其值的成分不确定的数据类型。然而，不论其元素具有何种特性，元素之间的关系相同，基本操作亦相同。从抽象数据类型的角度看，具有相同的数学抽象特性，故称之为多形数据类型。显然，需借助面向对象的程序设计语言如C++等实现之。本书中讨论的各种数据类型大多是多形数据类型，限于本书采

用类 C 语言作为描述工具，故只讨论含有确定成分的数据元素的情况。

1.4 数据的存储结构

数据的逻辑结构是从逻辑关系来观察数据，它与数据的存储无关，是独立于计算机的。数据的存储结构是逻辑结构在计算机存储器里的实现，它是依赖于计算机的。

计算机的存储器（主存）是由有限多个存储单元组成的，每个存储单元有惟一的地址，各存储单元的地址是连续编码的，每个存储单元 Z 都有惟一的后续单元 Z' = suc（Z）。Z 和 Z'称为相邻单元。一片相邻的存储单元的整体叫做存储区域，记作 M。在不产生混淆的情况下，Z 可以用来表示一个存储单元，也可以用来表示该存储单元的地址。地址 Z' = suc(Z) = Z + 1。

设有逻辑结构 G =（P，R），要把 G 存储在计算机中，首先必须建立一个从 P 的结点到 M 的单元的映像 S：P→M，即对于每一个 $p \in P$ 都有惟一的 $Z \in M$ 使得 S(p) = Z，Z 为 p 结点所占存储空间中的始单元。同时这个映象应具有明显地或隐含地体现关系 R 的能力。

由于计算机的存储区域总是有限的，所以如何合理地使用它，使得有限的存储区域发挥最大的作用，这是存储管理问题。

用 LOC(K)表示结点 k 对应的存储单元的地址。

有四种基本的存储映象方法，下面分别加以介绍。

1. 顺序的方法

这种方法主要用于线性的数据结构，它把逻辑上相邻的结点存储在物理上相邻的存储单元里，结点之间的关系由存储单元的邻接关系体现。如果结点 k 所占存储空间的第一个单元为 Z，即 LOC(K) = Z，而最后一个单元为 Z_1，那么 Z_1 的后继单元 Z' = suc (Z_1)就是 k 的后继 k'的第一个存储单元。

【例 1-5】 G =（P，R），P = $\{p_1, p_2, p_3, p_4, p_5\}$，R = {r}，r = $\{<p_1, p_2>, <p_2, p_3>, <p_3, p_4>, <p_4, p_5>\}$

假定每个结点占一个存储单元，结点 p_1 存放在 200 号单元中，则顺序存储实现如图 1-3 所示。

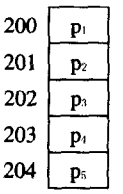


图 1-3 顺序存储的线性数据结构

这里存储区域被结点的值稠密地填满了。需要注意的是，在一般情况下，每个结点所占的存储单元并不止一个，而且所占单元个数也可以不一定相等，这时顺序存储存放时就不那么整齐，但仍然是一个接着一个地填满了整个区域。

另外对非线性的数据结构也可以采用局部线性化的方法实现顺序存储。例如，在树形结构中可以把结点按某种规则排成序列，用顺序存储方法将内部的信息稠密地存放在一起，而对结点之间的关系采用其他的存储方法。具体做法以后介绍。

2. 链接的方法

这种方法是给结点附加上指针字段。即是将结点所占的存储单元分为两部分，一部分存放结点本身的信息，称数据项；另一部分存放此结点的后继结点所对应的存储单元的地址，称指针项。指针项可以包括一个或多个指针，以指向结点的一个或多个后继，或记录其他信息（当然，一般来说，指针是用来表示某结点的地址的，并不是只能表示后继结点的地址）。

用 info 表示结点的数据项，用 link 表示结点的指针项。那么，如果 p'是 p 的后继结点，就有：

$$LOC(p') = p.link$$

前面例 1-5 的逻辑结构也可用链接的方法存储，如图 1-4 所示。

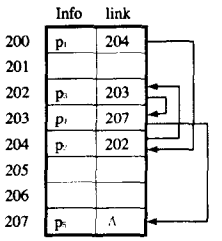


图 1-4 链接存储的线性数据结构

记号 Λ 表示空指针，即该指针不具有有意义的值（不表示任何具体结点的单元地址）。

【例 1-6】 逻辑结构为 G = （P，R），其中 $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$，$R = \{r\}$，$r = \{<p_1, p_2>, <p_1, p_3>, <p_2, p_4>, <p_2, p_5>, <p_3, p_6>\}$。

用链接的方式来实现这个逻辑结构，因为有些结点有两个后继，所以让指针项包括两个指针，分别指向两个后继，如图 1-5 所示。

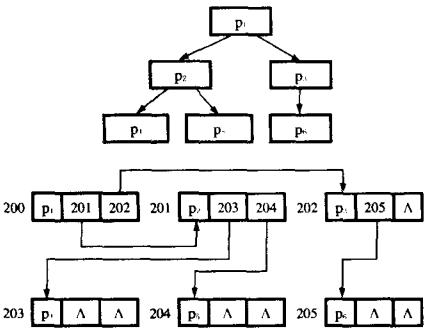


图 1-5 一个非线性数据结构的图示和链接存储

可以看到，在顺序的存储表中所有的存储空间都被结点数据占用了，而在链接的存储表示中就不是这样，有一部分存储空间里存放的是表示关系的附加信息——指针。

如果所有的存储空间都分配给了数据，则这个存储结构叫紧凑结构，否则叫非紧凑结构。显然顺序的存储表示是紧凑结构，链接的存储表示是非紧凑结构。

结构的存储密度定义为数据本身所占的存储量和整个结构所占的存储量之比，即：

$$d = \frac{\text{数据本身所占的存储量}}{\text{整个结构所占的存储量}}$$

紧凑结构的存储密度为 1，非紧凑结构的存储密度小于 1。存储密度越大，则存储空间的利用效率越高。但是存储附加的信息会带来运算上的方便。例如，在链表里，因为存储了指针，所以链表比起顺序表来作插入、删除运算要方便得多。牺牲了存储空间，换取了机器时间，这在以后还要详细讲到。

3. 索引的方法

在线性的结构里，结点可以排成一个序列：p_1，p_2，……，p_n，每个结点 p_i 在序列里都有对应的位置数 i，这个位置数就可以作为结点的索引。索引的存储结构就是用结点的索引号 i 来确定结点的存储地址。有两种实现方法：

（1）建立附加的索引表，索引表里第 i 项的值就是第 i 个结点的存储地址。

（2）当每个结点所占单元个数都相等时，可用位置数的线性函数的值来指出结点对应的存储地址，即第 i 个结点 p_i 的地址是 $LOC(p_i) = (i - 1)*p + q$。

其中，p 为结点所占单元个数，q 为开始结点 P_1 对应的存储地址。

例如，一维数组 A[N]，设 A[1]的存储地址是 d，每个数组元素占两个存储单元，那么 A[i]的存储地址为：$LOC(A[i]) = 2*(i - 1) + d$。

4. 散列的方法

这种方法的主要思想是根据结点的值来确定它的存储地址。

在结点 p 的字段里取一个（或几个）字段的值 w_{ik} 作为关键码，结点 p 对应的存储地址由函数 f（称为散列函数）确定：$LOC(k) = f(w_{ik})$。

所以这个方法也叫关键码——地址转移法。用散列法进行存储表示时，P 的结点 p 分散地存储在 M 的存储单元里。

在散列法存储表示中，关键的问题是选择散列函数和研究解决碰撞的方法。

以上介绍的是四种基本的存储映像方法，这些基本的方法还可以组合起来对数据结构进行存储映像。例如表的存储，由表的项不超过一个给定的极限 1 时，表可以顺序地存储在连续的存储区域里，当超过此极限时，由一个指针指向溢出区域，超过的项就放在这个附加的溢出区域里，这就是顺序和链接相结合的存储方法。

一般数据结构的存储映像都是这四种基本映像之一，或是它们的组合。同一个逻辑结构可以有几种不同的存储映像方法，选择哪一种要根据运算的方便来具体确定。

存储结构是数据结构的三个方面之一，若逻辑结构相同但存储结构不同，则为不同的数据结构，有时这种不同是相当大的，以至于人们用不同的数据结构名称来标识它们。例如线性表是一逻辑结构，若采用顺序方法的存储表示，在人们称该数据结构为顺序表；若采用链接方法的存储表示，则人们称该数据结构为链表。

1.5 数据的运算

为了更有效地处理数据，提高数据运算效率，就要按一定的逻辑结构把数据组织起来，并选择适当的存储表示方法把按逻辑结构组织好的数据存储到计算机的存储器里。数据的运算是定义在数据逻辑结构上的，但运算的具体实现要在存储结构上进行。数据的各种逻辑结构有相应的各种运算，每种逻辑结构都有一个运算的集合，这里只列举几种常用的运算，作简要介绍。