



普通高等教育“十一五”国家级规划教材

高等应用型人才培养规划教材

数据结构

(第4版)(C语言版)

邓文华 主编



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

普通高等教育“十一五”国家级规划教材
高等应用型人才培规划教材

数据结构（第4版）

（C语言版）

邓文华 主编

電子工業出版社·

Publishing House of Electronics Industry

北京·BEIJING

本书对常用的数据结构做了系统的介绍,力求概念清晰,注重实际应用。全书共9章,依次介绍了数据结构的基本概念、线性表、栈和队列、串和数组、树与二叉树、图,以及查找和排序等基本技术。第9章为实验部分,共设计了11个实验,涵盖了数据结构的主要内容,以便学生在实验时参考。全书以C语言为算法描述语言,每一章后面均列举了一些典型应用实例,并对本章知识点进行小结,列出本章的重点,以便学生掌握。

本书可作为应用型本科、高职高专计算机类及相关专业的教材使用。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

数据结构: C语言版 / 邓文华主编. —4版. —北京: 电子工业出版社, 2015.6
高等应用型人才培养规划教材
ISBN 978-7-121-26205-0

I. ①数… II. ①邓… III. ①数据结构—高等学校—教材 ②C语言—程序设计—高等学校—教材 IV. ①TP311.12 ②TP312

中国版本图书馆CIP数据核字(2015)第118526号

策划编辑: 吕 迈

责任编辑: 吕 迈

印 刷: 涿州市京南印刷厂

装 订: 涿州市京南印刷厂

出版发行: 电子工业出版社

北京市海淀区万寿路173信箱 邮编 100036

开 本: 787×1 092 1/16 印张: 16.75 字数: 428.8千字

版 次: 2004年9月第1版

2015年6月第4版

印 次: 2015年6月第1次印刷

定 价: 36.00元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

前 言

“数据结构”是计算机程序设计的重要理论基础，是计算机及其应用专业的一门重要基础课程和核心课程。它不仅是计算机软件专业课程的先导课程，而且逐渐被其他工科类专业所重视。

本教材自第1版2004年出版、第2版2007年出版、第3版2011年出版发行以来，受到了广大师生、读者的热烈欢迎，至今已出版发行近5万册，在此对广大师生、读者表示衷心的感谢。为了更好地适应新形势的发展与需要，特别是针对应用技术型院校的需要，我们在广泛收集读者意见的基础上对本书进行了修订再版。本版在保留第3版特点的基础上主要做了以下修改：

5.2节增加了线索二叉树；6.4节增加了任意一对顶点间的最短路径；7.3节增加了平衡二叉树及B₊树；8.7节增加了基数排序；第9章实验中增加了一个预备实验。

本教材有以下特点：

(1) 基础理论知识的阐述由浅入深、通俗易懂。内容组织和编排以应用为主线，略去了一些理论推导和数学证明的过程，淡化算法的设计分析和复杂的时空分析。

(2) 各章（除第1章、第9章外）都配有相应的典型例题，列举分析了许多实用的例子，大多数算法都直接给出了其相应的C语言设计程序，以便学生上机练习、实践。

(3) 为了便于学生复习及掌握每章的重点，本书在每章的结尾处对该章进行了小结，并列出了本章的重点。

(4) 配有实验一章，该章给出了11个实验项目，涵盖了数据结构的主要内容，以便学生实验参考。

(5) 全书配有电子讲稿（PPT）、习题解答及实验参考答案，从而极大地方便了教师备课。

本书目录中带*号的为高职高专院校选讲内容。

本书由邓文华任主编。本书的主要执笔者是：第1、3、5、9章由邓文华编写；第2、7、8章由张琰编写；第4、6章由李益明编写。其他执笔者还有：邹华胜、毕保祥、谢胜利、施作芳、邓泽川、梅志红、戴大蒙、孔繁胜、李元华、谢翠华、刘文斌、赵丽央。

由于编者水平有限，书中难免存在不妥之处，敬请读者赐教指正。

编 者
2015年3月

目 录

第 1 章 绪论.....	(1)	3.2 队列.....	(51)
1.1 从问题到程序.....	(1)	3.2.1 队列的定义及其基本运算 ..	(51)
1.2 有关概念和术语.....	(3)	3.2.2 队列的存储结构和基本运算 的实现	(52)
1.3 算法及算法分析.....	(6)	3.2.3 队列应用举例	(57)
1.3.1 算法特性	(6)	3.3 典型例题.....	(58)
1.3.2 算法描述	(7)	本章小结.....	(60)
1.3.3 算法分析	(8)	习题 3.....	(61)
1.4 关于数据结构的学习.....	(10)	第 4 章 串和数组.....	(65)
本章小结.....	(12)	4.1 串.....	(65)
习题 1.....	(12)	4.1.1 串的基本概念	(65)
第 2 章 线性表.....	(15)	4.1.2 串的基本运算	(66)
2.1 线性表的逻辑结构.....	(15)	4.1.3 串的存储结构及其基本运算 的实现	(67)
2.1.1 线性表的定义	(15)	4.1.4 串的其他运算举例	(69)
2.1.2 线性表的基本操作	(15)	4.2 数组.....	(70)
2.2 线性表顺序存储及其操作的 实现	(16)	4.2.1 数组的逻辑结构和基本 操作	(70)
2.2.1 顺序表	(16)	4.2.2 数组的存储结构	(71)
2.2.2 顺序表基本操作的实现	(18)	4.2.3 稀疏矩阵	(72)
2.2.3 顺序表的其他操作举例	(21)	4.2.4 矩阵的其他运算举例	(76)
2.3 线性表的链式存储及其操作的 实现	(23)	4.3 典型例题.....	(76)
2.3.1 单链表	(23)	本章小结.....	(78)
2.3.2 单链表基本操作的实现	(25)	习题 4.....	(78)
2.3.3 循环链表	(31)	第 5 章 树与二叉树.....	(81)
2.3.4 双向链表	(31)	5.1 树的概念与基本操作.....	(81)
2.3.5 单链表的其他操作举例	(33)	5.1.1 树的定义及相关术语	(81)
2.4 典型例题.....	(35)	5.1.2 树的基本操作	(83)
本章小结.....	(37)	5.2 二叉树.....	(83)
习题 2.....	(38)	5.2.1 二叉树的基本概念	(83)
第 3 章 栈和队列.....	(40)	5.2.2 二叉树的主要性质	(85)
3.1 栈.....	(40)	5.2.3 二叉树的存储结构与基本 操作	(86)
3.1.1 栈的定义及其基本运算	(40)	5.2.4 二叉树的遍历	(89)
3.1.2 栈的存储结构和基本运算 的实现	(41)	*5.2.5 线索二叉树	(94)
3.1.3 栈的应用举例	(43)	5.2.6 二叉树的其他操作举例	(97)
3.1.4 栈与递归的实现	(47)		

5.3 树与森林.....	(99)	7.4.1 哈希表与哈希方法.....	(158)
5.3.1 树的存储.....	(99)	7.4.2 常用的哈希函数构造方法.....	(159)
5.3.2 树、森林与二叉树的相互转换.....	(102)	7.4.3 处理冲突的方法.....	(161)
5.3.3 树和森林的遍历.....	(104)	7.4.4 哈希表的查找算法.....	(163)
5.4 最优二叉树——哈夫曼树.....	(105)	7.4.5 哈希表的性能分析.....	(163)
5.4.1 哈夫曼树的基本概念.....	(105)	7.5 典型例题.....	(164)
5.4.2 哈夫曼树的构造算法.....	(107)	本章小结.....	(170)
5.4.3 哈夫曼编码.....	(108)	习题 7.....	(170)
5.4.4 哈夫曼编码的算法实现.....	(110)	第 8 章 排序	(174)
5.5 典型例题.....	(110)	8.1 基本概念.....	(174)
本章小结.....	(113)	8.2 三种简单的排序方法.....	(175)
习题 5.....	(114)	8.2.1 直接插入排序.....	(175)
第 6 章 图	(117)	8.2.2 冒泡排序.....	(176)
6.1 图的基本概念.....	(117)	8.2.3 简单选择排序.....	(179)
6.1.1 图的定义和术语.....	(117)	8.3 希尔排序.....	(180)
6.1.2 图的基本操作.....	(119)	8.4 快速排序.....	(181)
6.2 图的存储结构.....	(120)	8.5 堆排序.....	(184)
6.2.1 邻接矩阵.....	(120)	8.6 归并排序.....	(186)
6.2.2 邻接表.....	(122)	*8.7 基数排序.....	(188)
6.3 图的遍历.....	(123)	8.7.1 多关键码排序.....	(188)
6.3.1 深度优先搜索.....	(124)	8.7.2 链式基数排序.....	(189)
6.3.2 广度优先搜索.....	(125)	8.8 各种排序方法的比较与讨论.....	(191)
6.4 图的应用.....	(126)	8.9 典型例题.....	(192)
6.4.1 最小生成树.....	(126)	本章小结.....	(195)
6.4.2 最短路径.....	(130)	习题 8.....	(195)
6.4.3 拓扑排序.....	(136)	第 9 章 实验	(199)
6.5 典型例题.....	(138)	实验 0 预备知识实验——复数 ADT 及其实现.....	(199)
本章小结.....	(142)	实验 1 顺序表的基本操作.....	(201)
习题 6.....	(143)	实验 2 链表的基本操作.....	(205)
第 7 章 查找	(146)	实验 3 栈的基本操作.....	(211)
7.1 基本概念与术语.....	(146)	实验 4 队列的基本操作.....	(218)
7.2 静态查找表.....	(147)	实验 5 字符串的基本操作.....	(228)
7.2.1 静态查找表结构.....	(147)	实验 6 二叉树的基本操作.....	(233)
7.2.2 顺序查找.....	(148)	实验 7 树的遍历和哈夫曼树.....	(238)
7.2.3 有序表的折半查找.....	(149)	实验 8 图的基本操作.....	(245)
7.2.4 分块查找.....	(151)	实验 9 排序.....	(251)
7.3 动态查找表.....	(152)	实验 10 查找.....	(256)
7.3.1 二叉排序树.....	(152)	参考文献	(262)
*7.3.2 平衡二叉树及 B_树.....	(156)		
7.4 哈希表.....	(158)		

第1章 绪 论

数据作为计算机加工处理的对象，如何在计算机中表示和存储是计算机科学研究的主要内容之一，更是计算机技术需要解决的关键问题之一。数据是计算机化的信息，是计算机处理的主要对象。科学计算、数据处理、过程控制、文件存储、数据库技术等，都是对数据进行加工处理的过程。因此，要设计出一个结构好、效率高的程序，必须研究数据的特性、数据间的相互关系及其对应的存储表示方法，并利用这些特性和关系设计相应的算法和程序。

1.1 从问题到程序

“数据结构”是计算机科学与技术专业的专业基础课，也是十分重要的核心课程，其主要研究内容是数据之间的逻辑关系和物理实现，即探索有利的数据组织形式及存取方式。计算机系统软件和应用软件的设计、开发要用到各种类型的数据结构。因此，要想更好地运用计算机来解决实际问题，仅仅依赖几种计算机程序设计语言是不够的，还必须学习和掌握数据结构的有关知识。

在计算机发展的初期，人们使用计算机的目的主要是处理数值计算问题。使用计算机来解决一个具体问题时，一般需要经过下列几个步骤：首先要从该具体问题中抽象出一个适当的数学模型，然后设计或选择一个解此数学模型的算法，再编写程序并进行调试、测试，最后运行程序并得到答案（如图 1.1 所示）。例如，求解梁架结构中应力数学模型的线性方程组，该方程组可以使用迭代算法来求解。

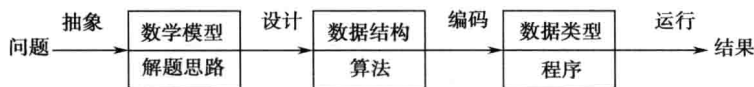


图 1.1 计算机解决问题的一般过程

由于当时所涉及的运算对象是简单的整型数据、实型数据或布尔型数据，所以程序设计者的主要精力集中于程序设计的技巧上，而无须重视数据结构。随着计算机应用领域的扩大和软/硬件的发展，非数值计算问题显得越来越重要。据统计，当今处理非数值计算性问题占用了 90% 以上的机器时间。这类问题涉及的处理对象不再是简单的数据类型，其形式更加多样，结构更为复杂，数据元素之间的相互关系一般无法直接用数学方程式加以描述。因此，解决这类问题的关键不再是数学分析和计算方法，而是设计出合适的数据结构，以便有效地解决问题。

【例 1.1】 图书信息检索系统。在现代图书馆中，人们往往借助计算机图书检索系统来查找需要的图书信息；或者直接通过图书馆信息系统进行图书借阅。为此，需要将图书信息分类编排，建立合适的数据结构进行存储和管理，按照某种算法编写相关程序，实现计算机自动检索。由此，一个简单的图书信息检索系统包括一张按图书分类号和登录号顺序排列的图书信息表，以及分别按作者、出版社等顺序排列的各类索引表，如图 1.2 所示。由这三张

表构成的文件便是图书信息检索的数学模型，计算机的主要操作便是按照用户的要求（如给定作者）通过不同的索引表对图书信息进行检索、查询。

序 号	图书分类号	登 录 号	书 名	作 者	出 版 社
1	B259.1	3240	梁启超家书	张品兴	中国文联出版社
2	C52	5231	探寻语碎	李泽厚	上海文艺出版社
3	D035.5	6712	市政学	张永桃	高等教育出版社
4	G206	1422	传播学	邵培仁	高等教育出版社
5	H319.4	1008	英语阅读策略	李宗宏	兰州大学出版社
6	K825.4.00	5819	围棋人生	聂卫平	中国文联出版社
7	P1.00	8810	通向太空之路	邹惠成	科学出版社
8	TN915	7911	通信与网络技术概论	刘云	中国铁道出版社
9	TP312	7623	计算机软件技术基础	王宇川	科学出版社
10	TP393.07	8001	网络管理与应用	张琳	人民邮电出版社
11	Q3.00	2501	普通遗传学	杨业华	高等教育出版社

(a) 图书信息表

姓 名	序 号
邵培仁	4
李泽厚	2
李宗宏	5
刘云	8
聂卫平	6
王宇川	9
杨业华	11
张琳	10
张品兴	1
张永桃	3
邹惠成	7

(b) 作者姓名索引表

出 版 社	序 号
高等教育出版社	3,4,11
科学出版社	7,9
兰州大学出版社	5
人民邮电出版社	10
上海文艺出版社	2
中国铁道出版社	8
中国文联出版社	1,6

(c) 出版社索引表

图 1.2 图书信息检索系统中的数据结构

诸如此类的还有电话自动查号系统、学生信息查询系统、仓库库存管理系统等。在这类数学模型中，计算机处理的对象之间通常存在着一种简单的线性关系，这类数学模型是线性数据结构的。

【例 1.2】 人机对弈问题。人机对弈是一个古老的人工智能问题，其解题思想是将对弈的策略事先存入计算机，策略包括对弈过程中所有可能的情况及响应的对策。在决定对策时，根据当前状态，考虑局势发展的趋势做出最有利的选择。因此，计算机操作的对象（数据元素）是对弈过程中的每一步棋盘状态（格局），数据元素之间的关系由比赛规则决定。通常，这个关系不是线性的，因为从一个格局可以派生出多个格局，所以通常用树形结构来表示，图 1.3 所示的是井字棋对弈树。

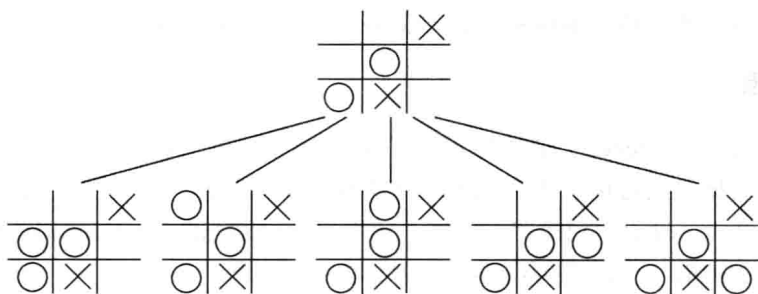
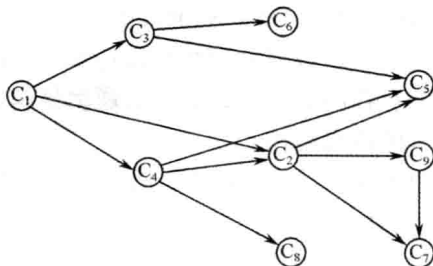


图 1.3 井字棋对弈树

【例 1.3】 教学计划编排问题。一个教学计划包含许多课程，在教学计划包含的许多课程之间，有些课程必须按规定的先后次序进行学习，有些则没有次序要求。课程之间先修和后修的次序关系可用一个称做图的数据结构来表示，如图 1.4 所示。有向图中的每个顶点表示一门课程，如果从顶点 v_i 到 v_j 之间存在有向边 $\langle v_i, v_j \rangle$ ，则表示课程 i 必须先于课程 j 进行学习。

课程编号	课程名称	先修课程
C ₁	计算机导论	无
C ₂	数据结构	C ₁ , C ₄
C ₃	汇编语言	C ₁
C ₄	C 程序设计语言	C ₁
C ₅	计算机图形学	C ₂ , C ₃ , C ₄
C ₆	接口技术	C ₃
C ₇	数据库原理	C ₂ , C ₉
C ₈	编译原理	C ₄
C ₉	操作系统	C ₂

(a) 计算机专业的课程设置



(b) 表示课程之间优先关系的有向图

图 1.4 教学计划编排问题的数据结构

由以上几个例子可见，描述非数值计算问题的数学模型不再是数学方程，而是诸如表、树、图之类的数据结构。因此，数据结构课程是研究非数值计算的程序设计问题中计算机处理对象及它们之间关系和操作的学科。

学习数据结构的目的是了解和掌握计算机处理对象的特性，将实际问题中所涉及的处理对象在计算机中表示出来并对它们进行处理。同时，通过算法训练来提高学生的思维能力，通过程序设计的技能训练来促进学生的综合应用能力和专业素质的提高。

1.2 有关概念和术语

在系统地学习数据结构知识之前，先对一些基本概念和术语赋予确切的定义。

1. 数据

数据 (Data) 是信息的载体，它能够被计算机识别、存储和处理。数据是计算机程序加工的原料，应用程序能处理各种各样的数据，包括数值数据和非数值数据。数值数据是一些

整数、实数或复数；非数值数据包括字符、文字、图形、图像、语音等。

2. 数据元素

数据元素（Data Element）是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。一个数据元素可由若干个数据项（Data Item）组成。在不同的条件下，数据元素又可称为元素、结点、顶点、记录等。例如，学生信息检索系统中学生信息表中的一个记录、教学计划编排问题中的一个顶点等，都被称为一个数据元素。

3. 数据项

数据项（Data Item）指不可分割的、具有独立意义的最小数据单位，数据项有时也称为字段（field）或域。例如，学籍管理系统中学生信息表的每一个数据元素就是一个学生记录。它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据项。这些数据项可以分为两种：一种叫做初等项，如学生的性别、籍贯等，这些数据项是在数据处理时不能再分割的最小单位；另一种叫做组合项，如学生的成绩，它可以再划分为数学、物理、化学等更小的项。通常，在解决实际问题时把每个学生记录当做一个基本单位进行访问和处理。

4. 数据结构

数据结构（Data Structure）是指互相之间存在着一种或多种关系的数据元素的集合。在任何问题中，数据元素都不会是孤立的，在它们之间存在着这样或那样的关系，这种数据元素之间存在的关系称为数据的**逻辑结构**。根据数据元素之间关系的不同特性，通常有以下 4 类基本的逻辑结构。

（1）集合结构：在集合结构中，数据元素之间的关系是“属于同一个集合”。数据元素之间除了同属一个集合外，不存在其他关系。

（2）线性结构：在该结构中，数据元素除了同属于一个集合外，数据元素之间还存在着一对一的顺序关系。

（3）树形结构：该结构的数据元素之间存在着一对多的层次关系。

（4）图状结构：该结构的数据元素之间存在着多对多的任意关系，图状结构也称为网状结构。

上述 4 类基本结构的示意图如图 1.5 所示。

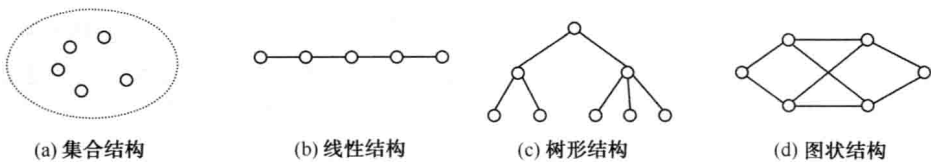


图 1.5 4 类基本结构的示意图

由于集合是数据元素之间极为松散的一种结构，本书不专门讨论。因此，本书主要讨论线性结构（表、栈、队、串等）和非线性结构（树、图或网）。

从上面所介绍的数据结构的概念中可以知道，一个数据结构有两个要素：一是数据元素，二是数据元素之间的关系。因此，数据结构通常可以采用一个二元组来表示：

$$\text{Data_Structure} = (D, R)$$

其中, D 是数据元素集合, R 是 D 中元素之间关系的集合。

【例 1.4】 假设一个数据结构定义如下:

$$\begin{aligned} DS &= (D, R) \\ D &= \{ a, b, c, d, e, f, g \} \\ R &= \{ \langle a, b \rangle, \langle a, c \rangle, \langle a, d \rangle, \langle c, e \rangle, \langle c, f \rangle, \langle d, g \rangle \} \end{aligned}$$

则该数据结构的逻辑示意图如图 1.6 所示, 显然是一个树形结构。

数据结构包括数据的逻辑结构和物理结构。数据的**逻辑结构**可以看做从具体问题抽象出来的数学模型, 它与数据的存储无关。数据的逻辑结构在计算机中的存储表示(又称映像)称为数据的**物理结构**(或称**存储结构**), 它所研究的是数据结构在计算机中的实现方法, 包括数据结构中数据元素的存储表示及数据元素之间关系的表示。

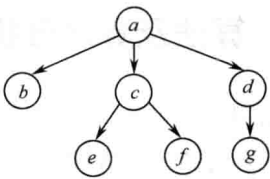


图 1.6 例 1.4 的数据结构逻辑示意图

在计算机中, 数据的存储方法包括**顺序存储**和**链式存储**。

(1) **顺序存储**方法通过数据元素在计算机中存储位置关系来表示元素间的逻辑关系, 通常把逻辑上相邻的元素存储在物理位置相邻的存储单元中。顺序存储是一种最基本的存储表示方法, 通常借助程序设计语言中的数组来实现。

(2) **链式存储**方法对逻辑上相邻的元素不要求其物理位置相邻, 元素间的逻辑关系通过指针字段来表示, 链式存储结构通常借助程序设计语言中的指针来实现。

除了顺序存储方法和链式存储方法外, 有时为了查找方便还采用索引存储方法和散列表(Hash)存储方法。

讨论数据结构的目的是在计算机中实现对数据的操作, 因此在讨论数据的组织结构时必然要考虑在该结构上进行的操作(或称运算)。事实上, 数据结构是专门研究某一类数据的表示方法及其相关操作实现算法的一门学科。

5. 数据类型

数据类型(Data Type)是和数据结构密切相关的一个概念, 在高级程序设计语言中用以限制变量取值范围和可能进行的操作的总和称为数据类型。因此, 所谓数据类型, 一是限制了数据的取值范围(实际上与存储形式有关); 二是规定了数据能够进行的一组操作(运算)。

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再分解的, 如 C 语言中的基本类型(整型、实型、字符型及指针类型和空类型); 另一类是结构类型, 它的成分可以由多个结构类型组成, 并可以分解。结构类型的成分可以是非结构的, 也可以是结构的。例如, 数组的值由若干分量组成, 每个分量可以是整数等基本类型, 也可以是数组等结构类型。

6. 抽象数据类型

抽象数据类型(Abstract Data Type, ADT)是指一个数学模型及定义在该模型上的一组操作。抽象数据类型的定义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。即无论其内部结构如何变化, 只要它的数学特性不变, 就不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如, 各种计算机都拥有的整数类型就是一个抽象数据类型, 尽管它们在不同处理器上的实现方法可以不同, 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。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组操作组成，而数据结构又包括数据元素及元素间的关系，因此抽象数据类型一般可以由元素、关系及操作三个要素来定义。

本书在讨论各种数据结构时，针对其逻辑结构和具体的存储结构给出相应的数据类型，并在确定的数据类型上通过各种算法实现各种操作。

1.3 算法及算法分析

算法与数据结构的关系非常紧密，在算法设计时总是先要确定相应的数据结构，而在讨论某一种数据结构时也必然会涉及相应的算法。下面就从算法特性、算法描述和算法分析三个方面对算法进行介绍。

1.3.1 算法特性

算法 (Algorithm) 是对特定问题求解步骤的一种描述，是指令的有限序列。其中每一条指令表示一个或多个操作。一个算法应该具有下列特性。

- (1) **有穷性**：一个算法必须在有穷步之后结束，即必须在有限时间内完成。
- (2) **确定性**：算法的每一步必须有确切的定义，无二义性，且在任何条件下算法只有唯一一条执行路径，即对于相同的输入只能得出相同的输出。
- (3) **可行性**：算法中的每一步都可以通过已经实现的基本运算的有限次执行得以实现。
- (4) **输入**：一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。
- (5) **输出**：一个算法具有一个或多个输出，这些输出同输入之间存在某种特定的关系。

算法的含义与程序十分相似，但又有区别。一个程序不一定满足有穷性。例如，对于操作系统，只要整个系统不遭破坏，它将永远不会停止，即使没有作业需要处理，它仍处于动态等待中。因此，操作系统不是一个算法。另外，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的求解方法，而程序则是算法在计算机上的特定实现。一个算法若用程序设计语言来描述，就是一个程序。

算法与数据结构是相辅相成的。解决某一类特定问题的算法可以选定不同的数据结构，而且选择恰当与否直接影响算法的效率。反之，一种数据结构的优劣由各种算法的执行效果来体现。

在算法设计时通常需要考虑以下几个方面的要求。

(1) **正确性**：算法的执行结果应当满足预先规定的功能和性能要求。正确性要求表明算法必须满足实际需求，达到解决实际问题的目标。

(2) **可读性**：一个算法应当思路清晰、层次分明、简单明了、易读易懂。可读性要求表明算法主要是人与人之间交流解题思路 and 进行软件设计的工具，因此可读性必须要强。同时一个可读性强的算法，其程序的可维护性、可扩展性都要好得多，因此，许多时候人们往往在一定程度上牺牲效率来提高可读性。

(3) **健壮性**：当输入不合法数据时，应能适当处理，不至于引起严重后果。健壮性要求表明算法要全面细致地考虑所有可能的边界情况，并对这些边界条件做出完备的处理，尽可能使算法没有意外的情况。

(4) 高效性：有效使用存储空间和有较好的时间效率。高效性主要是指时间效率，即解决相同规模的问题时间尽可能短。

一般来说，数据结构上的**基本操作**主要有以下几种。

- (1) 查找：寻找满足特定条件的数据元素所在的位置。
- (2) 读取：读出指定位置上数据元素的内容。
- (3) 插入：在指定位置上添加新的数据元素。
- (4) 删除：删去指定位置上对应的数据元素。
- (5) 更新：修改某个数据元素的值。

1.3.2 算法描述

算法的描述方法很多，根据描述方法的不同，大致可将算法描述分为以下 4 种。

(1) 自然语言算法描述：用人类自然语言（如中文、英文等）来描述算法，同时还可插入一些程序设计语言中的语句来描述，这种方法也称为非形式算法描述。其优点是不需要专门学习，任何人都可以直接阅读和理解，但直观性很差，复杂的算法难写难读。

(2) 框图算法描述：这是一种图示法，可以采用方框图、流程图、N-S 图等来描述算法，这种描述方法在算法研究的早期曾流行过。它的优点是直观、易懂，但用来描述比较复杂的算法就显得不够方便，也不够清晰简洁。

(3) 伪代码算法描述：如类 C 语言算法描述。这种算法描述很像程序，但它不能直接在计算机上编译、运行。这种方法很容易编写、阅读算法，而且格式统一，结构清晰，专业设计人员经常使用类 C 语言来描述算法。

(4) 高级程序设计语言编写的程序或函数：这是直接用高级语言来描述算法，它可在计算机上运行并获得结果，使给定问题能在有限时间内被求解，通常这种算法描述也称为程序。

【例 1.5】 求两个整数 m 、 n ($m \geq n$) 的最大公因子，该算法的不同描述方法如下。

(1) 非形式算法描述（自然语言算法描述）如下。

- ① [求余数]以 n 除 m ，并令 r 为余数 ($0 \leq r < n$)；
- ② [判断余数是否为零]若 $r = 0$ ，则结束算法， n 就是最大公因子；
- ③ [替换并返回步骤①]若 $r \neq 0$ ，则 $m \leftarrow n$ ， $n \leftarrow r$ ，返回步骤①。

(2) 算法的框图描述如图 1.7 所示。

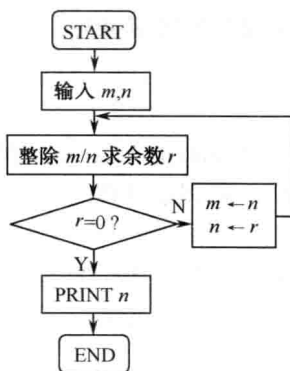


图 1.7 算法的框图描述

(3) C 语言函数描述如下。

```
int max_common_factor(int m, int n)
{
    int r;
    r=m%n ;
    while(r!=0)
        { m=n; n=r; r=m%n; }
    return n ;
}
```

本书主要介绍算法的思路和实现过程,且尽可能地给出算法对应的C语言函数或程序(或类C语言算法描述),方便读者阅读或上机运行,以便更好地理解算法。

1.3.3 算法分析

所谓好的算法,除了满足上文提到的几个基本要求外,还必须以较少的时间与空间代价来解决相同规模的问题。因此,一个算法的优劣,可以从该算法在计算机上运行的时间和所占存储空间来衡量和评判。算法分析就是预先分析算法在实际执行时的时空代价指标。

当一个算法被转换成程序并在计算机上执行时,其运行所需要的时间一般取决于下列几个因素。

(1) 硬件的速度。即主机本身运行速度,主要与CPU的主频和字长有关,也与主机系统采用的技术有关,如多机系统的运算速度一般比单机系统要快。

(2) 实现算法的程序设计语言。实现算法的语言的级别越高,其执行效率相对就越低。

(3) 编译程序所生成目标代码的质量。代码优化较好的编译程序所生成的程序质量较高。

(4) 算法所采用的策略。采用不同设计思路与解题方法,其时空代价是不同的,一般情况下时间指标与空间指标常常是矛盾的两个方面。

(5) 问题的规模。例如,求100以内的素数与求1000以内的素数的执行时间必然不同。

显然,在各种因素都不能确定的情况下,很难比较算法的执行时间。也就是说,用算法的绝对执行时间来衡量算法的效率是不合适的。为此,可以将上述各种与计算机相关的软、硬件因素都确定下来,仅对采用不同策略的算法,分析其运行代价随问题规模大小变化的对应关系,即运行代价仅依赖于问题的规模(通常用正整数 n 表示),或者说它是问题规模的函数。这种函数被称为算法的**时间复杂度**和**空间复杂度**。

1. 时间复杂度

一个程序的**时间复杂度 (Time Complexity)**是指该程序的运行时间与问题规模的对应关系。一个算法是由控制结构和原操作(所谓原操作是指从算法中选取对于所研究问题是基本运算的操作)构成的,其执行时间取决于两者的综合效果。为了便于比较同一问题的不同的算法,通常的做法是:从算法中选取一种对于所研究的问题来说是基本运算的原操作,以该原操作重复执行的次数为算法的时间度量。一般情况下,算法中原操作重复执行的次数是该算法所处理问题的规模 n 的某个函数 $T(n)$ 。

【例 1.6】 两个 $n \times n$ 阶的矩阵相乘的程序中的主要语句及其重复次数如下。

原操作语句的执行频度

```

for (i=0; i<n; i++)
    for (j=0; j<n; j++)
        {
            s[i][j]=0;
            for (k=0; k<n; k++)
                s[i][j]=s[i][j]+a[i][k]*b[k][j];
        }

```

n^2
 n^3

则该段程序的时间复杂度 $T(n) = cn^3 + n^2$ ，其中 c 为常量，表示算术运算时间是简单赋值运算时间的常数倍。

许多时候，精确地计算 $T(n)$ 是困难的，人们引入渐进时间复杂度在数量上估计一个算法的执行时间，也能够达到分析算法的目的。

定义（大 O 记号）： 如果存在两个正常数 c 和 n_0 ，使得对所有的 n ($n \geq n_0$)，有：

$$T(n) \leq c * f(n)$$

则 $T(n) = O(f(n))$ 。

例如，一个程序的实际执行时间为 $T(n) = 2.7n^3 + 3.8n^2 + 5.3$ ，则 $T(n) = O(n^3)$ 。

使用大 O 记号表示的算法的时间复杂度称为**算法的渐进时间复杂度（Asymptotic Time Complexity）**。

通常用 $O(1)$ 表示常数级时间复杂度，表明这样的算法执行时间是恒定的，不随问题规模的扩大而增长，显然这是最理想的，但往往难以实现。此外，常见的渐进时间复杂度还有：

- (1) $O(\log_2 n)$ ，对数级复杂度；
- (2) $O(n)$ ，线性复杂度；
- (3) $O(n^2)$ 和 $O(n^3)$ ，分别为平方级和立方级复杂度；
- (4) $O(2^n)$ ，指数级复杂度。

上述时间复杂度随问题规模 n 的扩大其增长速度是不同的，其增长速度的快慢次序表示如下：

$$O(1) < O(\log_2 n) < O(n) < O(n \log_2 n) < O(n^2) < O(n^3) < O(2^n)$$

2. 空间复杂度

一个程序的**空间复杂度（Space Complexity）**是指程序运行从开始到结束所需的存储量与问题规模的对应关系，记做：

$$S(n) = O(f(n))$$

其中 n 为问题的规模（或大小）。

一个上机执行的程序除了需要存储空间来寄存本身所用指令、常数、变量和输入数据外，还需要一些对数据进行操作的工作单元和存储为实现计算所需信息的辅助空间。若输入数据所占空间只取决于问题本身、和算法无关，则只需分析除输入数据和程序之外的额外空间，否则应同时考虑输入数据本身所需空间（和输入数据的表示形式有关）。若额外空间相对于输入数据量来说是常数，则称此算法为原地工作，辅助空间为 $O(1)$ 。如果所占空间量依赖于特定的输入，则除特别指明外，均按最坏情况来分析。

算法执行时间的耗费和所占存储空间的耗费是相互矛盾的，难以兼得。即算法执行时间上的节省是以增加存储空间为代价的，反之亦然。不过，一般而言，常常以算法执行时间作

为算法优劣的主要衡量指标。

1.4 关于数据结构的学习

计算机发展始终遵循摩尔（1965）法则：“芯片容量每 18 个月加倍”，新摩尔定理：“计算机性能每 18 个月提高一倍，价格每半年降低一半”，是否已经到达极限？是否会不再遵循摩尔法则？杨振宁在西安科协 2000 年会议上，明确地答复了这个问题：是什么原因促使芯片容量长期成倍地增长，新原理、新方法、新道理，是维持创新的源泉，创新是人类知识发展、生产发展的重要因素。

计算机性能价格比持续提高，硬件发展如此之快，是否没有必要去追求提高算法的时间复杂度、没有必要去追求节省算法占用存储空间数目呢？不是没有必要，而是要求越来越高。原因之一是，由于机器性能价格比的提高，人们所面临的处理问题的规模越来越大，要把过去不可能解决的问题变得可能，必须要求高性能的算法；另一个原因是，即使在同一问题规模情况下，算法性能好坏差别很大，一个是 $O(n)$ 数量级，一种是 $O(2^n)$ 数量级，当 $n=32$ 时， 2^n 的结果都已很大，即使 n 再增大一倍， 2^n 几乎都已经无法表述，这不是硬件发展速度所能满足的。由此说明，硬件速度的提高决不是人们可以不重视算法性能的理由，而是人们追求高性能算法的动力。图 1.8 所示为数据结构与其他课程关系图。

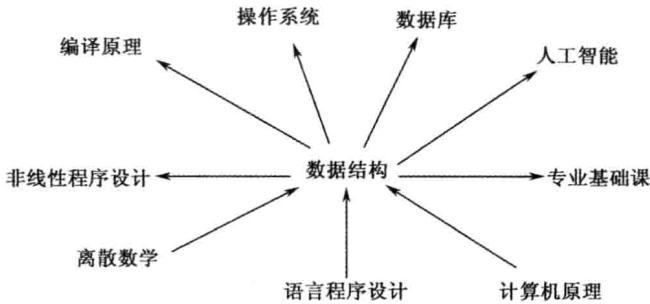


图 1.8 数据结构与其他课程关系图

1. 数据结构课程地位

明确提出数据结构概念不过 30 多年，“数据结构”作为一门独立课程在国外 1968 年开始设立，我国从 20 世纪 80 年代初才开始正式开设“数据结构”课程。“数据结构”课程较系统地介绍了软件设计中常用数据结构及相应的存储结构和算法，系统地介绍了常用的查找和排序技术，并对各种结构与技术进行分析和比较，内容非常丰富。数据结构涉及多方面的知识，如计算机硬件范围的存储装置和存取方法，软件范围中的文件系统、数据的动态管理、信息检索，以及数学范围中关于集合、逻辑的知识，还有一些综合性的知识（如数据类型、程序设计方法、数据表示、数据运算、数据存取等），是计算机专业一门重要的专业技术基础课程。数据结构的内容将为操作系统、数据库原理、编译原理等后续课程的学习打下良好的基础，数据结构课程不仅讲授数据信息在计算机中的组织和表示方法，还训练高效地解决复杂问题程序设计的能力，因此数据结构是数学、计算机硬件、计算机软件三者之间的一门核心课程，

“数据结构”课程是计算机专业提高软件设计水平的一门关键性课程。

数据结构发展趋势包括两个方面：一方面，面向专门领域中特殊问题的数据结构的研究和发展，如图形数据结构、知识数据结构、空间数据结构；另一方面，从抽象数据类型的角度，用面向对象观点来讨论数据结构，已成为新的发展趋势。

2. 数据结构课程学习特点

“数据结构”课程教学目标要求学生学会分析数据对象特征，掌握数据组织方法和计算机的表示方法，以便为应用所涉及的数据选择适当的逻辑结构、存储结构及相应算法，初步掌握算法时间、空间分析的技巧，培养良好的程序设计技能。

人类解决问题的思维方式可分为两大类：一类是推理方式，凭借公理系统思维方法，从抽象公理体系出发，演绎、归纳、推理，求证结果，解决特定问题；另一类是算法方式，凭借算法构造思维方式，从具体操作规范入手，通过操作过程的构造和实施解决特定问题。开发一个优秀的软件系统过程中所凭借的思维方法本质上不同于常规数学训练的公理系统思维方法，而是一种算法构造性思维方法。系统开发是创造性思维过程的实现，因而，对于一名开发人员，只知道开发工具的语言规则和简单使用过程是不够的。首先要有科学方法指导开发过程；然后在编程技术应用技能上积累提高。让学生理解、习惯、熟悉这一套算法构造思维方法，是计算机软件课程教学的重要内容和主要难点。

“数据结构”的学习过程是进行复杂程序设计的训练过程。技能培养的重要程度不亚于知识传授。难点在于：理解授课内容与应用知识解答复杂问题之间的素质能力差距。培养优良的算法设计思想、方法技巧与风格，进行构造性思维训练过程，强化程序抽象能力，培养数据抽象能力。从某种意义上说，数据结构是程序设计的后继课程。如同学习英语一样，学习英语不难，学好英语不易，要提高程序设计水平必须经过艰苦的磨炼。因此，学习数据结构，仅从书本上学习是不够的，必须经过大量的实践，在实践中体会构造性思维方法，掌握数据组织与程序设计的技术。

3. 关于本书内容编写说明

(1) 本书基本结构

基本结构分为如下四大部分。

第一部分：绪论。

第二部分：基本的数据结构。包括：线性结构（第2~4章）——线性表、栈和队列、串和数组；非线性结构（第5、6章）——树、图。

第三部分：基本技术。包括查找与排序（第7、8章）。

第四部分：实验实训。

(2) 本书内容编排模式

本书所列出的程序均在 Turbo C 2.0 下调试通过，所有算法均采用严谨的 C 语言进行描述，只需加以必要的类型定义与调用，即可上机运行使用。

每章附有习题，以便于读者做配套练习。