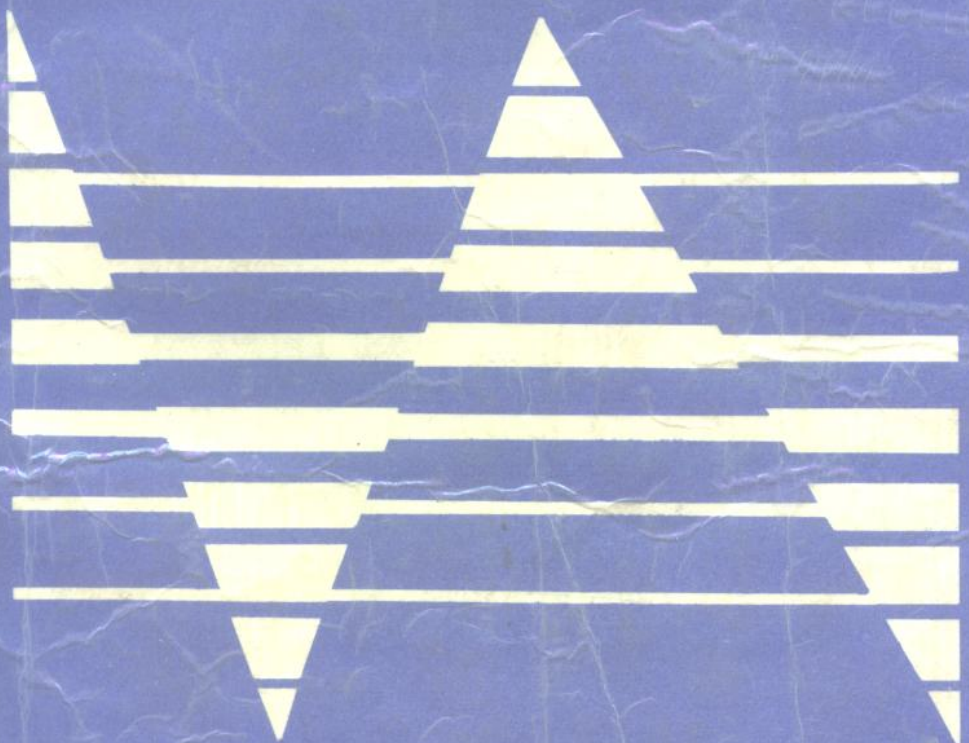
数据结构教程

唐发根　刘又诚　编著



北京航空航天大学出版社

数据结构教程

唐发根　刘又诚　编著

北京航空航天大学出版社

内 容 提 要

本书是1994年出版的《数据结构》的修订版。它在原来版本的基础上适当增删了部分内容，并增加了部分习题的解答。

本书共分十一章，分别介绍了各种数据结构的基本概念、逻辑结构与存储结构，讨论了在各种结构上所实施的一些运算。算法用SPARKS语言给出，简明易懂，具有较好的可读性与可移植性。

本书不仅可以作为高等学校计算机专业本科生与专科生的专业基础课教材，也可以用作从事计算机系统软件和应用软件设计与开发人员的参考资料。

图书在版编目(CIP)数据

数据结构教程/唐发根等编著. —北京：北京航空航天大学出版社，1996.11

ISBN 7-81012-669-5

Ⅰ.数… Ⅱ.唐… Ⅲ.数据结构-教材 Ⅳ.TP311.12

中国版本图书馆CIP数据核字(96)第16473号

●书　　名：**数据结构教程**
SHUJU JIEGOU JIAOCHENG
●编 著 者：唐发根　刘又诚
●责任编辑：冯学民
●责任校对：张韵秋
●出 版 者：北京航空航天大学出版社
●印 刷 者：朝阳区科普印刷厂
●发　　行：新华书店总店北京发行所
●经　　售：全国各地新华书店
●开　　本：787×1092　1/16
●印　　张：16.75
●字　　数：429千字
●印　　数：7 000册
●版　　次：1996年10月第1版
●印　　次：1996年10月第1次印刷
●书　　号：ISBN 7-81012-669-5/TP·221
●定　　价：19.00元

前　言

随着计算机科学的迅速发展，数据结构作为一门新兴学科已经越来越受到计算机界的重视，被认为是计算机领域的一门十分重要的基础学科。正是基于这样的原因，我们在保持1994年出版的《数据结构》基本框架和特色的基础上对原书做了必要的修改和增删，编写了这本《数据结构教程》。

本书较详细地介绍了线性结构、非线性结构和文件三大类数据结构，阐述了在这些结构上实施的有关算法的设计与实现。程序设计语言与数据结构之间存在着密切的联系：程序设计语言为数据结构的描述提供了很好的手段；数据结构为程序设计语言类型系统的发展与完善奠定了基础。为此，本书强调了数据结构本身的技术及其在程序设计中的应用。

全书共分十一章。第一章阐述了数据结构的一些基本概念；第二章至第六章主要讨论线性结构，其中包括线性表、堆栈、队列以及字符串等；第七章与第八章讨论非线性结构，重点讨论树和二叉树、图的基本概念及其应用；第九章至第十一章主要讨论文件的基本概念和有关的操作。

在本书的编写过程中，我们本着着重基础、注意应用的原则，每一章除了必要的例题之外，还配有适量的习题与上机题供读者练习。凡是阅读过本书并独立完成习题的读者，都能掌握本书所要求的基本概念、基本技术与基本方法。

数据结构是一门实践性很强的课程。通过该课程的学习，应使学生能够运用数据结构提供的方法与技巧更好地进行算法设计和程序设计。所以，本书在讨论各种数据结构基本运算的同时，都给出了相应的算法。算法全部采用SPARKS语言描述，具有较好的可读性与可移植性。SPARKS语言属于一种类PASCAL语言，它几乎包括了所有程序设计语言所共有的、基本的可执行语句，且不受具体程序设计语言在词法、语法以及语义上的严格限制。因而，用SPARKS语言描述的算法简洁、易懂。熟悉任何一种高级程序设计语言的人，只要对书中的算法稍做修改或补充，便可得到计算机所能接受的程序。

本书取材广泛、内容全面，作为高等学校计算机专业本科学生的教材，讲授学时数为50～70个，也可以根据具体情况做一些增删。另外，考虑到专科教育的针对性、实践性和应用性，本书也可以作为高等专科学校、职工大学、职业大学、夜大以及函授大学等大专类计算机硬件、软件专业的教材或参考书。同时，本书还可以用作从事计算机系统软件和应用软件设计与开发人员的参考资料。

除了作者以外，钱士湘教授曾多次讲授过“数据结构”课程。可以说，本书是我们在多年教学实践的基础上，根据原有讲义和教材，并参考了兄弟院校同类教材编著完成的。因此，在内容上努力遵循深入浅出的原则，在基本概念的阐述方面力求通俗详尽，以便于读者自学。

第一、二、三、四、五、六、七、十章由唐发根撰写，第八、九、十一章由刘又诚撰写。张福渊同

志为全书的描图工作付出了辛勤的劳动，在此表示衷心的感谢。

由于数据结构本身也是一门年轻的学科，且内容还在不断变化与更新，加上作者水平有限以及时间仓促等原因，书中的错误在所难免。作者恳切希望读者批评指正。

作　者

1996 年 7 月

于北京航空航天大学

目 录

第一章 绪 论 …… (1)

1.1 什么是数据结构 …… (1)
1.2 数据结构的发展简史及其在计算机科学中的地位 …… (3)
1.3 算 法 …… (3)
1.4 SPARKS 语言简介 …… (5)
1.4.1 算法格式 …… (5)
1.4.2 SPARKS 语句 …… (6)
1.5 算法分析 …… (9)
1.5.1 时间复杂度 …… (9)
1.5.2 空间复杂度 …… (10)
1.5.3 其他方面 …… (11)
1.6 算法设计的基本步骤 …… (11)
习题 …… (12)

第二章 线 性 表 …… (14)

2.1 线性表及其基本运算 …… (14)
2.1.1 线性表的定义 …… (14)
2.1.2 关于线性表的基本运算 …… (15)
2.2 线性表的顺序存储结构 …… (16)
2.3 线性表的链式存储结构 …… (18)
2.3.1 线性链表 …… (19)
2.3.2 线性链表的有关算法 …… (21)
2.4 循环链表及其运算 …… (27)
2.5 双向链表及其运算 …… (30)
2.5.1 双向链表的构造 …… (30)
2.5.2 双向链表的插入与删除算法 …… (31)
2.6 链表的应用举例 …… (33)
2.6.1 链式存储结构下的一元多项式加法 …… (33)
2.6.2 动态存储管理 …… (35)
习题 …… (41)

第三章 数 组 …… (43)

3.1 数组的概念 …… (43)
3.1.1 一维数组 …… (43)

3.1.2 多维数组……(43)
3.2 数组的存储结构……(44)
3.3 矩阵的压缩存储……(45)
3.3.1 对称矩阵的压缩存储……(46)
3.3.2 对角矩阵的压缩存储……(46)
3.4 稀疏矩阵的三元组表示……(47)
3.5 稀疏矩阵的十字链表表示……(52)
3.6 数组的应用举例……(56)
3.6.1 一元多项式的数组表示……(56)
3.6.2 n 阶魔方……(57)
习题……(58)

第四章 堆栈和队列……(60)

4.1 堆栈的概念及其运算……(60)
4.1.1 堆栈的定义……(60)
4.1.2 堆栈的有关运算……(61)
4.2 堆栈的顺序存储结构……(61)
4.3 堆栈的链式存储结构……(64)
4.4 堆栈的应用举例……(65)
4.4.1 堆栈在递归中的应用……(65)
4.4.2 表达式的计算……(69)
4.4.3 一个趣味游戏——迷宫问题……(73)
4.5 队列的概念及其运算……(75)
4.5.1 队列的定义……(75)
4.5.2 队列的有关运算……(76)
4.6 队列的顺序存储结构……(76)
4.7 队列的链式存储结构……(79)
习题……(81)

第五章 广 义 表……(83)

5.1 广义表的概念……(83)
5.2 广义表的存储结构……(84)
5.3 多元多项式的表示……(86)
习题……(87)

第六章 串……(88)

6.1 串的概念……(88)
6.1.1 串的定义……(88)
6.1.2 串的几个概念……(89)

6.2 串的基本运算…… (89)
6.3 串的存储结构…… (90)
6.3.1 串的顺序存储结构…… (90)
6.3.2 串的链式存储结构…… (91)
6.4 串的几个运算…… (92)
习题 …… (97)

第七章 树与二叉树 …… (98)

7.1 树的基本概念…… (98)
7.1.1 树的定义…… (98)
7.1.2 树的逻辑表示方法 …… (100)
7.1.3 基本术语 …… (100)
7.1.4 树的基本运算 …… (101)
7.2 树的存储结构 …… (102)
7.2.1 多重链表表示法 …… (102)
7.2.2 三重链表表示法 …… (102)
7.3 二 叉 树 …… (103)
7.3.1 二叉树的定义 …… (103)
7.3.2 二叉树的基本运算 …… (104)
7.3.3 满二叉树与完全二叉树 …… (105)
7.3.4 二叉树的性质 …… (105)
7.3.5 二叉树与树、树林之间的转换…… (106)
7.4 二叉树的存储结构 …… (109)
7.4.1 二叉树的顺序存储结构 …… (109)
7.4.2 二叉树的链式存储结构 …… (110)
7.5 树的遍历 …… (112)
7.5.1 二叉树的遍历 …… (112)
7.5.2 树和树林的遍历 …… (118)
7.5.3 由遍历序列恢复二叉树 …… (119)
7.6 线索二叉树 …… (120)
7.6.1 线索二叉树的构造 …… (121)
7.6.2 线索二叉树的利用 …… (122)
7.6.3 二叉树的线索化算法 …… (124)
7.6.4 线索树的更新 …… (124)
7.7 二叉排序树 …… (125)
7.7.1 二叉排序树的定义 …… (126)
7.7.2 二叉排序树的构造 …… (126)
7.7.3 在二叉排序树中删除结点 …… (128)
7.7.4 二叉排序树的查找 …… (130)

7.8 平衡二叉树 …… (132)
7.9 哈夫曼树及其应用 …… (138)
7.9.1 哈夫曼树的概念 …… (138)
7.9.2 哈夫曼编码 …… (139)
7.10 树的一个应用——判定树 …… (142)
习题 …… (144)

第八章 图 …… (147)

8.1 图的基本概念 …… (147)
8.1.1 图的定义和基本术语 …… (147)
8.1.2 图的基本运算 …… (150)
8.2 图的存储方法 …… (150)
8.2.1 邻接矩阵存储方法 …… (151)
8.2.2 邻接表存储方法 …… (151)
8.2.3 有向图的十字链表存储方法 …… (154)
8.2.4 无向图的多重邻接表存储方法 …… (154)
8.3 图的遍历 …… (155)
8.3.1 深度优先搜索(Depth First Search) …… (156)
8.3.2 广度优先搜索(Breadth First Search) …… (158)
8.4 最小生成树 …… (159)
8.5 最短路径问题 …… (162)
8.5.1 某个源点到其余各个顶点的最短路径 …… (162)
8.5.2 每一对顶点之间的最短路径 …… (165)
8.6 AOV 网与拓扑排序 …… (168)
8.6.1 AOV 网(Activity on vertex network) …… (168)
8.6.2 拓扑排序 …… (169)
8.6.3 拓扑排序算法 …… (169)
8.7 AOE 网与关键路径 …… (173)
8.7.1 AOE 网(Activity on edge network) …… (173)
8.7.2 关键路径 …… (174)
8.7.3 关键路径的确定 …… (174)
习题 …… (176)

第九章 文件及查找 …… (179)

9.1 文件概述 …… (179)
9.1.1 文件的基本术语 …… (179)
9.1.2 文件的存储介质 …… (180)
9.1.3 文件的基本操作 …… (181)
9.2 顺序文件 …… (182)

9.2.1 连续顺序文件 …… (182)
9.2.2 链接顺序文件 …… (185)
9.3 索引文件 …… (186)
9.3.1 稠密索引文件 …… (186)
9.3.2 非稠密索引文件 …… (186)
9.3.3 多级索引文件 …… (187)
9.4 索引顺序存取文件 …… (189)
9.4.1 空间的划分 …… (189)
9.4.2 ISAM 的索引结构 …… (189)
9.4.3 ISAM 文件的基本操作 …… (191)
9.4.4 主索引和柱面索引的最佳位置 …… (193)
9.5 B_树和 B$^{\pm}$树 …… (193)
9.5.1 B_树概述 …… (193)
9.5.2 B_树的基本操作 …… (194)
9.5.3 B$^{\pm}$树的概念 …… (199)
9.5.4 B$^{\pm}$树的基本操作 …… (200)
9.6 虚拟存储存取文件 …… (200)
9.6.1 VSAM 的结构 …… (200)
9.6.2 VSAM 的操作 …… (201)
9.7 静态索引与动态索引的比较 …… (202)
9.8 杂凑(Hash)文件 …… (202)
9.8.1 概　述 …… (202)
9.8.2 杂凑函数的几种构造方法 …… (203)
9.8.3 处理冲突的方法 …… (205)
9.8.4 杂凑文件的操作 …… (206)
9.8.5 散列法的平均查找长度 …… (208)
9.9 多重链表文件 …… (208)
9.10 倒排文件 …… (209)
习题 …… (210)

第十章 内 排 序 …… (212)

10.1 概　述 …… (212)
10.1.1 排序的概念 …… (212)
10.1.2 排序的分类 …… (212)
10.2 插入排序(INSERTION SORT) …… (213)
10.3 选择排序(SELECTION SORT) …… (215)
10.4 泡排序(BUBBLE SORT) …… (216)
10.5 谢尔排序(SHELL SORT) …… (218)
10.6 快速排序(QUICK SORT) …… (219)

10.7 堆积排序(HEAP SORT) …… (221)
10.7.1 堆积的定义 …… (221)
10.7.2 堆积排序算法 …… (222)
10.8 二路归并排序(2-WAY MERGE SORT) …… (225)
10.8.1 归并子算法 …… (225)
10.8.2 一趟归并扫描子算法 …… (226)
10.8.3 二路归并排序算法 …… (227)
10.9 基数排序(RADIX SORT) …… (228)
10.10 各种内排序算法的比较 …… (231)
10.10.1 稳定性比较 …… (231)
10.10.2 复杂性比较 …… (231)
习题 …… (233)

第十一章 外 排 序 …… (235)

11.1 概 述 …… (235)
11.2 磁带排序 …… (236)
11.2.1 多路平衡归并排序法 …… (236)
11.2.2 多步归并排序 …… (238)
11.3 初始归并段的合理分布与产生 …… (239)
11.3.1 初始归并段的合理分布 …… (239)
11.3.2 一种产生初始归并段的方法——置换选择排序 …… (240)
11.4 磁盘排序 …… (242)
11.4.1 最佳归并树 …… (242)
习题 …… (245)

附录 部分习题参考答案 …… (246)

参考文献 …… (258)

第一章 绪 论

被人们称之为第二次工业革命标志的电子计算机自本世纪40年代问世以来，发展神速，无论是在软件方面还是在硬件方面都已经远远超出了人们对它的估计。随着计算机技术的高速发展以及微型计算机的日益普及，计算机已经广泛深入到人类社会的各个领域。现在，计算机已经不再局限于解决那些纯数值计算的问题，更多、更广泛的是应用在非数值计算的各个领域，例如，企业管理、工业过程控制、经济系统工程、管理信息系统等等。与此相对应，计算机处理的对象也由纯粹的数值数据发展到诸如字符、图像、声音、信号等各种各样具有一定结构的数据。为了有效地组织和管理好这些数据，设计出高质量的程序，高效率地使用计算机，必须深入研究这些数据的特性以及它们之间的相互联系。这也正是数据结构这门新兴学科形成与发展的背景。

1.1 什么是数据结构

众所周知，计算机是一种信息处理装置。信息中的各个数据元素并不是孤立存在的，它们之间有着一定的结构关系。如何表示这些结构关系，如何在计算机中存储数据和信息，采用什么样的方法和技巧去加工处理这些数据，这些都是数据结构这门课程要努力解决的问题。

究竟什么是数据结构呢？在回答这个问题以前，先来复习几个并不陌生的名词术语。

数据(data)：描述客观事物的数字、字符以及一切能够输入到计算机中、并能被计算机程序处理的符号集合。简言之，数据就是计算机加工处理的“原料”。数据的含义十分广泛，在不同场合具有不同的含义。例如，在数值计算的问题中，计算机处理的对象大多数都是整数或者实数；在一些文字处理程序中，计算机处理的对象是一些符号串；而在某些控制过程问题中，可能又是某种信号。在很多场合，人们对数据和信息的引用没有加以区分。信息应是指数据这个集合中元素的含义；而数据则是信息的某种特定的符号表示形式，可以从中提取信息。

数据元素(data element)：数据的基本单位，即数据这个集合中的一个个的客体。数据元素也称为**数据结点**。有时候，一个数据元素可以由若干个数据项组成(可见，**数据项**是数据的最小单位)。例如，数据文件中每一个数据记录就是一个数据元素，而每个数据记录又是由若干个描述客体某一方面特征的数据项组成。

数据对象(data object)：具有相同特性的数据元素的集合，是数据这个集合的一个子集。例如，自然数的数据对象是集合{1,2,3,…}，而由26个英文字母字符组成的数据对象则是集合{A,B,C,…,Z}。

数据结构的概念目前还没有一个被一致公认的定义，不同的人在使用这个词时所表达的意思有所不同。尽管如此，我们还是从几个不同角度给数据结构下一个定义。

在客观世界中，任何事物及活动都不会孤立地存在，都在一定意义上相互影响、相互联系，甚至相互制约。同样，数据元素之间也必然存在着某种联系，我们称这种联系为**结构**。另外，不

仅要考虑到数据的这种结构，而且还要考虑将要施加于数据上的操作及其种类。从这个意义上说，**数据结构**就是具有结构的数据元素的集合。我们可以给数据结构一个形式化的描述。

数据结构是一个二元组

$$Data\_Structure = (D,R)$$

其中，D 是数据元素的有限集合，R 是 D 上关系的集合。

在上面的定义中，关系指的是数据元素之间的逻辑关系，又称为数据的**逻辑结构**。与此相对应的还有数据的物理结构，即数据结构在计算机中的映象，或者说，是数据在计算机存储器中的表示。因此，物理结构也称为**存储结构**，如向量、链表都属于存储结构。

一种逻辑结构通过映象便得到它的存储结构。由于映象的方法不同（具体实现时可以有顺序、链接、索引、散列等方法），同一种逻辑结构可以映象成不同的存储结构，如顺序存储结构和非顺序存储结构（或称为链式存储结构）；反之，数据的存储结构一定要正确反映出数据元素之间的逻辑关系。

为了具体说明这一点，下面举一个例子。

例如，一个具有 30 条记录的反映学生情况的数据文件，其记录在文件中的先后顺序是按学生年龄从小到大排列的，如图 1.1 所示。

	姓　名	性　别	民　族	出生年月	其　他
1	刘晓光	男	汉	1971.9	……
2	王　敏	女	汉	1971.5	……
3	马广生	男	回	1971.3	……
⋮	⋮	⋮	⋮	⋮	
30	张玉华	女	汉	1970.4	……

图 1.1　学生情况表

该数据文件中记录之间的逻辑关系是一个线性关系（因此也称该数据文件为一个线性表）。对应于这个逻辑结构，在计算机内可以有两种物理结构，即顺序存储结构和链式存储结构。前者用一片地址连续的存储空间依次存放该文件的 30 条记录，记录物理上的先后关系映射了记录逻辑上的先后关系；也就是说，逻辑上相邻的两个数据元素，其存储位置也相邻。后者则用 30 个称为链结点的存储空间分别存放这 30 条记录，每个结点物理地址上可以不连续，但通过链指针来映射记录之间的逻辑关系。见图 1.2。

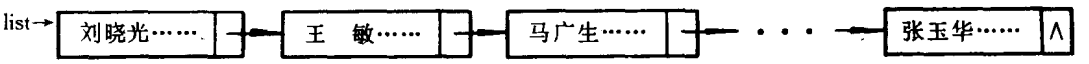


图 1.2　一个线性链表的例子

由此不难想到，在实现某种运算之前，首先需要选择合适的存储结构，而同一种运算在不同的存储结构中实现的方法不同，有的则完全依赖于所选择的存储结构。可见，数据的逻辑结构与存储结构是紧密相连的两个方面。

综上所述，数据结构所要研究的主要内容可以简要地归纳为以下三个方面：

（1）研究数据元素之间固有的客观联系（逻辑结构）。

(2) 研究数据在计算机内部的存储方法(存储结构)。

(3) 研究如何在数据的各种结构(逻辑的和物理的)上实施有效的操作或处理(算法)。

为此,应该说数据结构是一门抽象地研究数据之间结构关系的学科。

1.2 数据结构的发展简史及其在计算机科学中的地位

在国外,数据结构成为一门独立的课程始于1968年,在我国还要稍晚一些。此前,现在的"数据结构"课程的某些内容出现在其他一些诸如"编译方法"、"操作系统"的课程中。尽管后来美国一些大学的计算机系在教学计划中将"数据结构"列为一门独立的课程,但对课程的范围仍然没有做出明确的规定。当时的"数据结构"几乎同图论,特别是表处理、树的理论是一回事。随后,数据结构的概念被扩充到包括网络、集合代数论等一些方面,从而变成了称为"离散结构"的内容。60年代中期出现了类似于现在"数据结构"的课程,叫做"表处理语言",主要介绍处理表结构和树结构的语言。1968年美国出版了一本大型著作《计算机程序设计技巧》第一卷《基本算法》(作者:D·E·克努特),较为系统地阐述了数据的逻辑结构与物理结构以及运算的理论方法与技巧。60年代末到70年代初,出现了大型程序,软件也相对独立,结构程序设计逐步成为程序设计方法学的主要内容。人们越来越重视数据结构,已经认识到程序设计的实质就是对所确定的问题选择一种好的结构,从而设计一种好的算法。从那以后,各种版本的《数据结构》著作相继问世。

目前在我国,随着计算机基础教育的普及以及计算机越来越广泛应用于非数值计算问题的处理,"数据结构"不仅作为计算机专业教学计划中的重点核心课程之一,而且也开始成为其他许多非计算机专业的主要选修课。作为计算机专业在"程序设计"课程之后、各专业课程之前的一门最重要的专业基础课,"数据结构"不仅为学习"编译原理"、"操作系统"、"数据库原理"等后继课程提供必要的基础,也直接为从事各类系统软件和应用软件的设计提供了必备的知识与方法。

数据结构在计算机科学领域有着十分重要的地位。它有着自己的理论、研究对象和应用范围,而且其研究内容还在不断扩充和深化。为此,作为一门课程或者一本教材,因受到特定对象和时期的限制,难以全面反映整个数据结构的全貌。数据结构作为一门方兴未艾的新兴学科,目前仍然处在一个蓬勃发展的阶段。

1.3 算　法

数据结构与算法之间存在着密切的联系。可以说,不了解施加于数据上的算法需求就无法决定数据结构;反之,算法的结构设计和选择在很大程度上又依赖于作为其基础的数据结构,即数据结构为算法提供了工具,而算法则是运用这些工具来实施解决问题的最优方案。凡是从事过程序设计的人都会有这样一个体会:程序设计过程中相当多的时间花费在构思算法上,一旦有了合适的算法,用某种具体的程序设计语言来实现(编写程序)并不是一件很困难的事情。难怪有人说:**算法+数据结构=程序**。从这个角度上说,要设计出一个好的程序,很大程度上取决于设计出一个好的算法。

什么是算法?

简单地说，**算法**就是解决问题的办法。算法是用来完成某个特定课题的一些指令的集合；或者说，是由人们组织起来准备加以实施的有限的基本步骤。由此可以说，程序就是用计算机语言表述的算法，流程图就是图形化的算法，甚至一个公式也叫做算法。

在计算机领域，一个算法实质上是根据所处理问题的需要，在数据的逻辑结构和物理结构的基础上施加的一种运算。由于数据的逻辑结构与物理结构不是唯一的，在很大程度上可以由用户自行选择和设计，因而处理同一个问题的算法也不一定是唯一的；另外，即使具有相同的逻辑结构与物理结构，但如果算法的设计思想和技巧不同，设计出来的算法也可大不相同。显然，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与物理结构，进而设计出比较满意的算法，是学习数据结构这门课程的主要目的。

作为一个完整的算法应该满足下面五个标准，通常称为算法的基本性质。

输入 由外部提供 $n \geqslant 0$ 个有限量作为输入。

输出 至少有一个量作为输出。

有穷性 算法必须在有限的步骤内结束。这并不意味着在人们可以容忍的时间内结束，因为算法的性质不应与具体的机器相联系，只是说明算法要有结束。

确定性 组成算法的指令必须清晰、无二义性。就是说，算法的每一步骤都必须准确定义。自然语言传递的信息具有许多语义不清之处，人们在处理它们时可以根据上下文信息和推理准确地接受它，而机器却不能。

有效性 算法的指令必须具有可执行性。也就是说，算法所实现的动作都应该是基本的，可以付诸实施的；同时，实施了的算法应该能够达到预期的目的。

算法还有一个重要的侧面，就是构成这个算法所依据的办法（公式、方案、原则），即通常所说的解题思路。有许多问题，只要对数据对象进行细致的分析，就能确定处理方法；有的问题则不然。不过，作为寻找设计思路的基本思想方法对任何算法设计都是有用的。这些方法通常有枚举法、归纳法、回溯法以及模拟。限于篇幅，不能在本书中对每一种思想方法做具体的分析，请读者参考有关程序设计技巧与方法方面的书籍。

算法独立于具体的计算机与具体的程序设计语言。在设计一个算法时，如何选择一种合适的方式来表达算法思想呢？或者说，有了解决问题的算法思想，如何选择一种合适的语言来描述算法的各个步骤呢？这也是本节要讨论的重要问题之一。

在计算机发展的初期，人们往往用自然语言（如中国人多用汉语）来表达自己的算法思想。下面看一个简单例子。

例 1.1 求两个正整数 M 与 N 的最大公因子。

若采用自然语言来描述解决该问题的各个步骤，则可以表达为：

① M 除以 N，将余数送中间变量 R；

② 判断 R 是否为零？

(a) 若 R 等于零，算法到此结束，求得的最大公因子为当前 N 的值；

(b) 若 R 不等于零，将 N 值送 M，R 值送 N，重复算法的①和②。

类似的简单问题用自然语言表达还是可以的，但很快就会发现，采用自然语言描述算法很不方便，也不直观，更谈不上有良好的可读性，稍微复杂一些的算法就难以表达，甚至无法表达。另外，由于自然语言本身的一些限制，用它描述的算法可能会出现二义性现象。后来采用流程图的形式来描述算法（见图 1.3），比采用自然语言表达直观了一些，但依然没有解决复杂

算法的表达，而且移植性也不好。

有人提出，算法最终是要通过程序设计语言来实现而变成程序的，不如直接采用某种具体的程序设计语言（如 PASCAL、FORTRAN 或 C 等等）来描述。但人们很快就意识到这样做会带来诸多不便，因为这种方式过分地依赖于具体的程序设计语言，受到具体语言语法细节的限制（如繁琐的变量说明，语句的书写规则等等），使程序不能做到一目了然，也不便于在各种不同类型语言之间进行移植。于是，人们现在通常的做法是设计出一种既脱离某种具体的程序设计语言，又具有各种程序设计语言共同特点的形式化语言来描述算法。这种能正确方便地表达算法思想的语言省去了一般程序设计语言对各种类型的数据（如变量、数组或语句标号）进行的说明或定义，仅仅引用了大多数程序设计语言所具有的可执行语句的格式来确定所需要的语句。用这种语言编写的“程序”（实际上是算法）虽然不能直接在计算机上执行，但经过熟悉任何一种程序设计语言的人稍做修改或补充，就能很容易地变成计算机所能接受的程序。

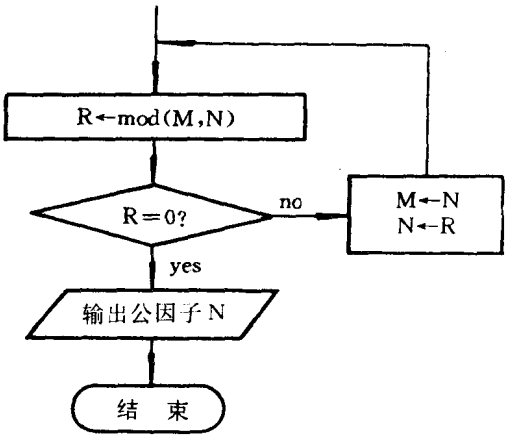


图 1.3 求 M、N 的最大公因子

本书所采用的这种描述算法的语言取名为 SPARKS 语言，实际上是一种类 PASCAL 语言。对于例 1.1 求最大公因子的问题，用 SPARKS 语言描述的算法为：

```
function COMFACTOR(M,N)
    loop
      R←mod(M,N)   //求 M 除以 N 的余数//
      if R=0  then
        return(N)
      M←N
      N←R
    forever
end
```

1.4 SPARKS 语言简介

SPARKS 语言是一种不依赖具体计算机与具体程序设计语言的形式化语言，主要是用来描述或表达算法思想。其特点是表达能力强，用它描述的算法结构简单、易于理解，具有较好的可移植性；书写自然，没有通常的程序设计语言那样繁杂的语法、语义上的严格限制，除了要求算法名必须以大写字母开头，并以大写字母、下划线或数字组成的字符串表示外，算法中其他任何地方出现的字母大小写完全等价（字符串常量除外）。

1.4.1 算法格式

一个完整的 SPARKS 语言编写的算法必须具有下面的形式之一：

```
procedure 算法名(参数表)          function 算法名(参数表)
    语句串                            语句串
end                               end
```

其中,参数表可以含有多个参数,也可以没有参数,若有参数,参数之间用逗号分开;语句串由一个或多个 SPARKS 语句组成,指明该算法所要执行的动作。

函数形式一般出现在表达式中,它返回一个结果作为表达式的一部分参加运算。一个算法至少有一个 end 与 procedure 或 function 对应,作为算法的结束标志。一个完整的 SPARKS 语言算法可由一个或多个算法组成,不妨约定其中第一个算法为主算法。

下面分别介绍 SPARKS 语言各种语句的用法。

1.4.2 SPARKS 语句

1. 赋值语句

V←E

其中,"←"为赋值号;V 为变量名或数组元素名;E 为表达式。

赋值语句的功能是先计算表达式 E 的值,然后将所得值赋给 V。这里,允许出现"V_1←V_2←…←E"的赋值形式,即将一个表达式的值分别赋给多个不同的变量,也允许如"(V_1,V_2,…,V_n)←(E_1,E_2,…,E_n)"的成组赋值形式。

2. 条件语句(两种)

(1) if 条件 then
    [语句串]

(2) if 条件 then
    [语句串 1]
  else
    [语句串 2]

其中,条件为关系表达式或者逻辑表达式;语句串、语句串 1、语句串 2 均可由一条或多条语句组成,当它们分别为一条语句时,方括号[ ]可以省去(该左右方括号相当于 PASCAL 语言中的 begin 和 end,或者 C 语言中的左右花括号{}。下同)。

该条件语句的执行过程与一般的高级程序设计语言条件语句的执行过程几乎完全相同。

3. 循环语句(四种)

(1) while 循环条件 do
      语句串
  end

(2) repeat
      语句串
  until 循环结束条件

(3) for v←e_1 $\left\{\begin{matrix}\text{to}\\ \text{downto}\end{matrix}\right\}$ e_2 by e_3 do
      语句串
  end