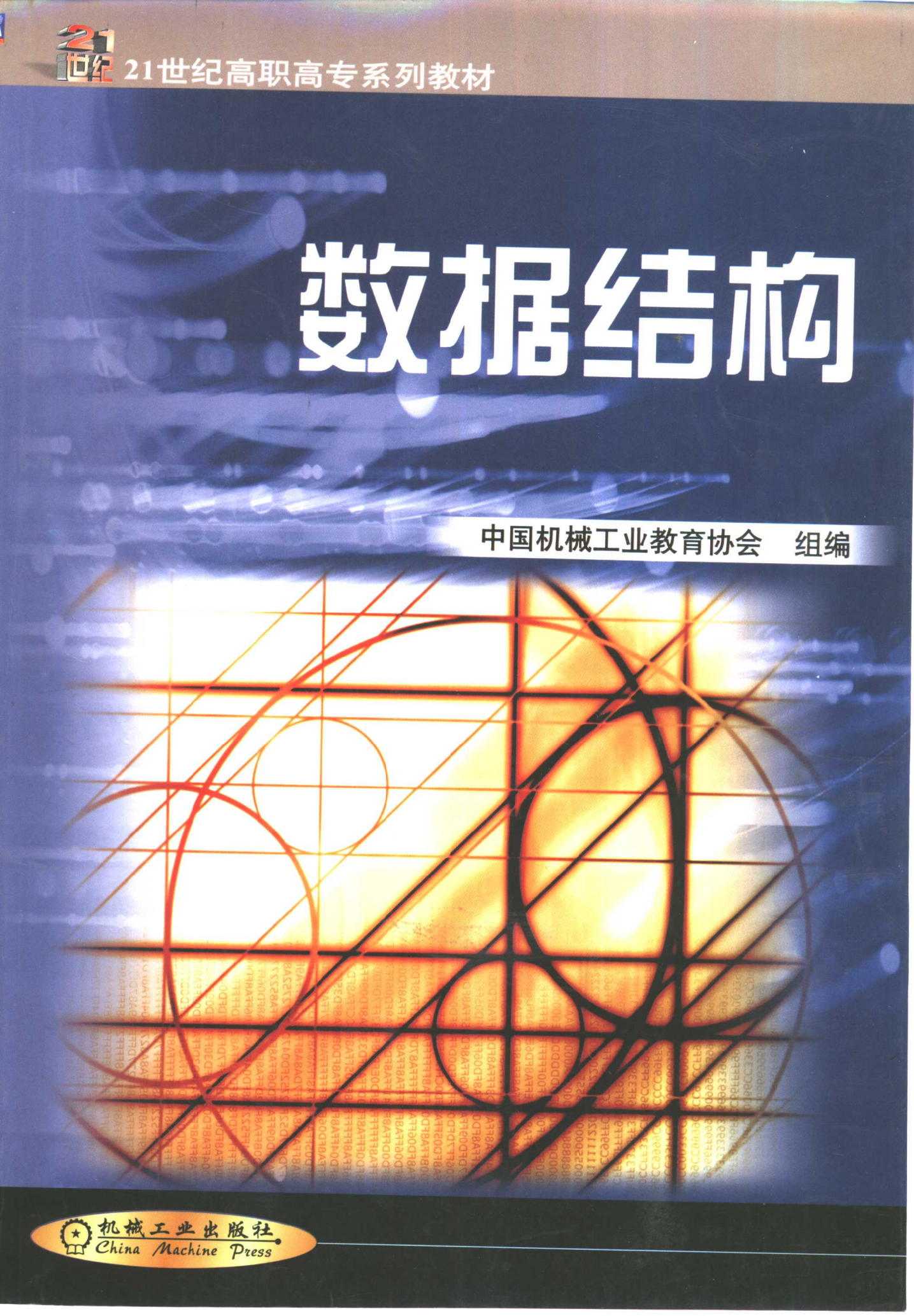
21世纪高职高专系列教材

数据结构

中国机械工业教育协会　组编



机械工业出版社
China Machine Press

本书系统地介绍了各种数据结构的特点、存储结构和有关算法。书中采用 C 语言描述算法。主要内容包括：数据结构的基本概念、算法描述和算法分析初步；线性表、栈、队列、数组、串、树、图等数据结构；查找、排序方法等。每章后面配有练习题及上机实习题。

本着注重应用的原则，本书选材精炼，叙述深入浅出，实例丰富。本书是专为计算机类高职、高专学生而编写的教材，也可作为大中专院校计算机类各专业的教材，还可作为从事计算机应用的工程技术人员的自学参考书。

图书在版编目（CIP）数据

数据结构 / 中国机械工业教育协会组编. —北京：机械工业出版社，2001.7
21 世纪高职高专系列教材
ISBN 7-111-08406-3

I. 数… II. 中… III. 数据结构—高等学校：技术学校—教材 IV.TP311.12

中国版本图书馆 CIP 数据核字（2001）第 041697 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
责任编辑：余茂祚
封面设计：姚 毅 责任印制：路 琳
中国建筑工业出版社密云印刷厂印刷·新华书店北京发行所发行
2001 年 7 月第 1 版第 1 次印刷
787mm×1092mm1/16·10 印张·246 千字
0 001—4 000 册
定价：16.00 元

21世纪高职高专系列教材编委会名单

编委会主任 中国机械工业教育协会 郝广发

编委会副主任（单位按笔画排）

山东工程学院 仪垂杰
大连理工大学 唐志宏
天津大学 周志刚
甘肃工业大学 路文江
江苏理工大学 杨继昌
成都航空职业技术学院 陈玉华
机械工业出版社 陈瑞藻（常务）
沈阳工业大学 李荣德
河北工业大学 檀润华
武汉船舶职业技术学院 郭江平
金华职业技术学院 余党军

编委会委员（单位按笔画排）

广东白云职业技术学院 谢瀚华
山东省职业技术教育师资培训中心 邹培明
上海电机技术高等专科学校 徐余法
天津中德职业技术学院 李大卫
天津理工学院职业技术学院 沙洪均
日照职业技术学院 李连业
北方交通大学职业技术学院 佟立本
辽宁工学院职业技术学院 李居参
包头职业技术学院 郑 刚
北京科技大学职业技术学院 马德青
北京建设职工大学 常 莲
北京海淀走读大学 成运花
江苏理工大学 吴向阳
合肥联合大学 杨久志
同济大学 孙 章
机械工业出版社 李超群 余茂祚（常务）
沈阳建筑工程学院 王宝金
佳木斯大学职业技术学院 王跃国
河北工业大学 范顺成
哈尔滨理工大学工业技术学院 线恒录
洛阳大学 吴 锐
洛阳工学院职业技术学院 李德顺
南昌大学 肖玉梅
厦门大学 朱立秒
湖北工学院高等职业技术学院 吴振彪
彭城职业大学 陈嘉莉
燕山大学 刘德有

序

1999 年 6 月中共中央国务院召开第三次全国教育工作会议，作出了“关于深化教育改革，全面推进素质教育的决定”的重大决策，强调教育在综合国力的形成中处于基础地位，坚持实施科教兴国的战略。决定中明确提出要大力发展高等职业教育，培养一大批具有必备的理论知识和较强的实践能力，适应生产、建设、管理、服务第一线急需的高等技术应用性专门人才。为此，教育部召开了关于加强高职高专教学工作会议，进一步明确了高职高专是以培养技术应用性专门人才为根本任务；以适应社会需要为目标；以培养技术应用能力为主线设计学生的知识、能力、素质结构和培养方案；以“应用”为主旨和特征来构建课程和教学内容体系；高职高专的专业设置要体现地区、行业经济和社会发展的需要，即用人的需求；教材可以“一纲多本”，形成有特色的高职高专教材系列。

“教书育人，教材先行”，教育离不开教材。为了贯彻中共中央国务院以及教育部关于高职高专人才培养目标及教材建设的总体要求，中国机械工业教育协会、机械工业出版社组织全国部分有高职高专教学经验的职业技术学院、普通高等学校编写了这套《21 世纪高职高专系列教材》。教材首批 80 余本（书目附书后）已陆续出版发行。

本套教材是根据高中毕业 3 年制（总学时 1 600～1 800）、兼顾 2 年制（总学时 1 100～1 200）的高职高专教学计划需要编写的。在内容上突出了基础理论知识的应用和实践能力的培养。基础理论课以应用为目的，以必需、够用为度，以讲清概念、强化应用为重点；专业课加强了针对性和实用性，强化了实践教学。为了扩大使用面，在内容的取舍上也考虑到电大、职大、业大、函大等教育的教学、自学需要。

每类专业的教材在内容安排和体系上是有机联系、相互衔接的，但每本教材又有各自的独立性。因此各地区院校可根据自己的教学特点进行选择使用。

为了提高质量，真正编写出有显著特色的 21 世纪高职高专系列教材，组织编写队伍时，采取专门办高职的院校与办高职的普通高等院校相互协作编写并交叉审稿，以便实践教学和理论教学能相互渗透。

机械工业出版社是我国成立最早、规模最大的科技出版社之一，在教材编辑出版方面有雄厚的实力和丰富的经验，出版了一大批适用于全国研究生、大学本科、专科、中专、职工培训等各种层次的成套系列教材，在国内享有很高的声誉。我们相信这套教材也一定能成为具有我国特色的、适合 21 世纪高职高专教育特点的系列教材。

中国机械工业教育协会

前　言

《数据结构》是计算机专业的一门重要的专业基础课。这门课程主要研究如何合理地组织数据，以及怎样在计算机中有效地表示数据和处理数据。通过这门课程的学习，使学生得到必要的程序设计技能训练，同时又为学习《操作系统》、《软件工程》等后续课程提供必要的知识准备。

本书主要是为在校的高等职业技术院校的学生编写的。针对高职生面向应用、注重实践的特点，本书力求做到理论知识介绍适度，注重应用。对各种常用数据结构的介绍尽量从实例出发，通过对实例的分析，使学生了解数据结构的基本概念。尽量避免抽象理论及复杂的公式推导。书中采用C语言来描述数据结构及相应的算法，为学生上机提供了方便。本书各章后面附有复习思考题及上机实习题供读者选做，以加强读者对数据结构和算法的理解。

本书由纪颖主编，张辉、叶飞副主编，陶树平主审。其中第1章、第2章、第4章、第7章由纪颖编写，第6章由张辉编写，第8章、第9章由郭芳编写，第5章由曹桂琴编写，第3章由王晓琳编写。

由于作者水平有限，书中难免存在错误或不当之处，敬请读者批评指正。

编　者

目　录

序
前言
第 1 章　概论 …… 1
1.1　数据结构的基本概念和术语 …… 1
1.2　算法描述与分析 …… 2
1.2.1　算法 …… 2
1.2.2　算法分析 …… 3
复习思考题 …… 4
第 2 章　线性表 …… 5
2.1　线性表的定义和运算 …… 5
2.1.1　线性表的定义 …… 5
2.1.2　线性表的运算 …… 5
2.2　线性表的顺序存储结构 …… 6
2.2.1　线性表的顺序存储结构 …… 6
2.2.2　顺序表的运算 …… 7
2.3　线性表的链式存储结构 …… 8
2.3.1　线性链表 …… 8
2.3.2　单链表的基本运算 …… 10
2.4　循环链表 …… 13
2.5　双向链表 …… 14
2.6　线性表的应用——多项式相加 16
实习题 …… 18
复习思考题 …… 18
第 3 章　栈和队列 …… 19
3.1　栈 …… 19
3.1.1　栈的定义及其运算 …… 19
3.1.2　顺序栈——栈的顺序存储结构 … 20
3.1.3　链栈——栈的链式存储结构 …… 22
3.1.4　栈的应用举例 …… 23
3.2　队列 …… 26
3.2.1　队列的定义及运算 …… 26
3.2.2　队列的存储结构及基本运算的实现 …… 27
3.2.3　队列的应用简介 …… 32
实习题 …… 33
复习思考题 …… 34
第 4 章　数组和广义表 …… 35
4.1　数组 …… 35
4.1.1　数组的定义 …… 35
4.1.2　数组的顺序存储结构 …… 35
4.1.3　特殊矩阵的压缩存储 …… 36
4.2　稀疏矩阵 …… 37
4.2.1　三元组表 …… 37
4.2.2　稀疏矩阵的链接存储 …… 40
4.3　数组的应用 …… 41
4.4　广义表 …… 43
4.4.1　广义表的定义 …… 43
4.4.2　广义表的存储结构 …… 44
复习思考题 …… 45
第 5 章　串 …… 46
5.1　串的基本概念和运算 …… 46
5.1.1　串的基本概念 …… 46
5.1.2　串的基本运算 …… 46
5.2　串的存储结构 …… 47
5.2.1　串的顺序存储结构 …… 47
5.2.2　串的链式存储结构 …… 47
5.3　串运算的算法 …… 48
5.4　文本编辑 …… 51
实习题 …… 52
复习思考题 …… 52

第 6 章　树和二叉树 …… 53
6.1　树的概念与存储表示 …… 53
6.1.1　树的基本概念 …… 53
6.1.2　树的基本操作 …… 55
6.1.3　树的存储表示 …… 56
6.2　二叉树 …… 59
6.2.1　二叉树的基本概念 …… 59
6.2.2　二叉树的性质 …… 60
6.2.3　二叉树的存储结构 …… 61
6.2.4　二叉树的基本操作及其实现 …… 63
6.3　二叉树的遍历 …… 66
6.3.1　二叉树遍历的基本概念 …… 66
6.3.2　前序遍历 …… 66
6.3.3　中序遍历 …… 69
6.3.4　后序遍历 …… 70
6.3.5　二叉树遍历的应用 …… 72
6.4　线索二叉树 …… 74
6.4.1　线索二叉树的基本概念 …… 74
6.4.2　线索二叉树的基本操作与实现 … 76
6.5　树、森林与二叉树的转换及遍历 …… 79
6.5.1　树转换为二叉树 …… 79
6.5.2　森林转换为二叉树 …… 80
6.5.3　二叉树还原为树或森林 …… 81
6.5.4　树和森林的遍历 …… 81
6.6　哈夫曼树及其应用 …… 82
6.6.1　路径及路径长度 …… 82
6.6.2　哈夫曼树 …… 83
6.6.3　哈夫曼树的应用 …… 84
实习题 …… 88
复习思考题 …… 89
第 7 章　图 …… 91
7.1　图的概念及术语 …… 91
7.2　图的存储结构 …… 93
7.2.1　邻接矩阵 …… 93
7.2.2　邻接链表 …… 94
7.3　图的遍历 …… 96
7.3.1　深度优先搜索遍历 …… 96
7.3.2　连通图的广度优先搜索遍历 …… 97
7.4　最小生成树 …… 98
7.4.1　生成树 …… 98
7.4.2　普里姆算法 …… 99
7.4.3　克鲁斯卡尔算法 …… 101
7.5　最短路径 …… 102
7.5.1　最短路径问题 …… 102
7.5.2　从某个源点到其他各顶点的最短路径的计算 …… 102
7.6　拓扑排序 …… 104
7.7　关键路径 …… 105
复习思考题 …… 108
第 8 章　查找 …… 110
8.1　查找的基本概念 …… 110
8.2　线性表的查找 …… 110
8.2.1　顺序查找 …… 110
8.2.2　二分查找 …… 111
8.2.3　插值查找 …… 113
8.2.4　分块查找 …… 113
8.3　树表的查找 …… 115
8.3.1　二叉排序树 …… 115
8.3.2　平衡二叉排序树 …… 120
8.3.3　B-树 …… 122
8.3.4　B^+ 树 …… 124
8.4　哈希表和查找 …… 125
8.4.1　哈希表 …… 125
8.4.2　哈希函数 …… 126
8.4.3　冲突的处理 …… 127
实习题 …… 131
复习思考题 …… 131
第 9 章　排序 …… 133
9.1　排序的基本概念 …… 133
9.2　插入排序 …… 133
9.2.1　直接插入排序 …… 133
9.2.2　二分插入排序 …… 135
9.2.3　希尔排序 …… 136
9.3　交换排序 …… 137

9.3.1 冒泡排序 …… 137
9.3.2 快速排序 …… 138
9.4 选择排序 …… 140
9.4.1 直接选择排序 …… 140
9.4.2 堆排序 …… 141
9.5 归并排序 …… 145
9.6 基数排序 …… 147
9.7 各种内排序方法的讨论 …… 150
实习题 …… 151
复习思考题 …… 151
参考文献 …… 151

第 1 章　概论

自 1946 年美国第一台电子计算机问世以来，计算机科学和软硬件技术得到了飞速的发展，与此同时，计算机的应用领域也从最初的科学计算逐步发展到人类活动的各个领域。现在，计算机处理的对象不仅是简单的数值和字符，而且还包括图形、图像、声音等多媒体数据，其数据的结构也愈加复杂。因此，要开发出一种性能良好的软件，不仅要根据实际需要掌握至少一种适合的计算机高级语言或软件开发工具，更重要的是研究数据的不同结构和组织方法以及进行数据处理的不同算法，通过分析和比较选择出较好的设计方案才行。后者正是数据结构这门学科所要解决的问题。

1.1　数据结构的基本概念和术语

这一节，我们首先介绍一些基本概念和术语。

1. 数据　是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。对计算机而言，数据的含义极为广泛，如数字、文字、图形、图像、声音等都是数据。

2. 数据元素　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。数据元素有时也称为结点、记录等。一个数据元素一般由一个或多个数据项组成。

3. 数据项　具有独立意义的最小数据单位，是对数据元素属性的描述。数据项也称域或字段。例如：在学生档案管理系统中，可以把一个学生的有关信息作为一个数据元素，它由学号、姓名、年龄、出生年月等数据项组成。

4. 数据类型　是指某种程序设计语言所允许使用的变量种类，各种高级语言提供的基本数据类型有所不同，如 C 语言提供了整型、实型和字符型等基本数据类型。除基本类型外，C 语言还允许数组型、结构型、联合型等构造类型。一个数据类型不仅定义了相应变量可以设定的值的集合和存储方法，而且还规定了对变量允许进行的一组运算及其规则。所以，可以把数据类型看作是程序设计语言中已经实现了的数据结构。

5. 数据对象　是性质相同的数据元素的集合，是数据的一个子集。例如：所有的字母构成的集合 C=('A','B', …,'Z')是一个字母数据对象；在学生档案管理系统中所有的学生构成的集合是一个学生数据对象。

6. 数据结构　是带有结构的元素的集合，它是指数据之间的相互关系，即数据的组织形式。例如：全校教师档案管理的组织方式如图 1-1 所示，这种形式实际上是一棵自顶向下生长的树，像这样按分支和层次组织的数据称为“树”结构。

我们把数据间的逻辑上的联系，称之为数据的逻辑结构，如线性表、树、图等，它是数据元素间的抽象化的相互联系。逻辑结构不考虑数据在计算机中具体的存储方式，是独立于计算机的。然而，讨论数据结构的目的是为了在计算机中实现对它的操作，因此还需要研究如何在计算机中表示它。数据的逻辑结构在计算机存储设备中的映象被称为数据的存储结构，也可以说数据的存储结构是逻辑结构在计算机存储器里的实现，又称物理结构。数据的存储

结构是依赖于计算机的，无论在形式上还是在顺序上都可能和数据的逻辑结构不同，例如顺序结构、链表结构等。

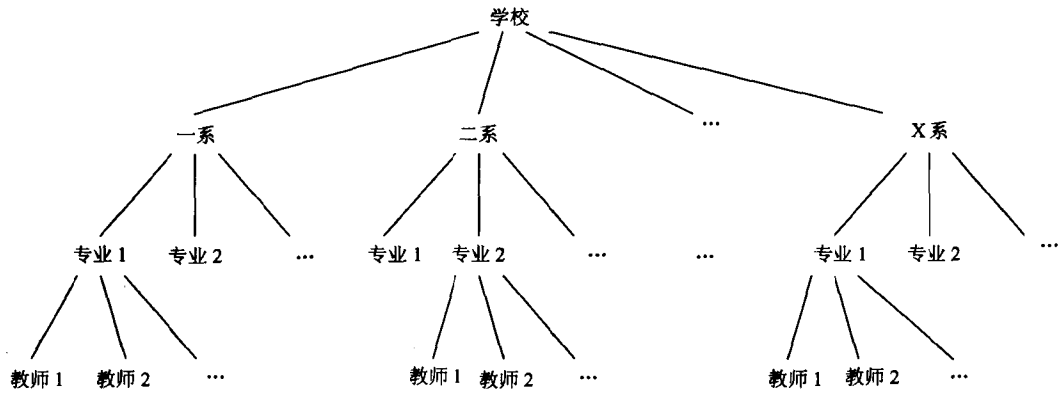


图 1-1　全校教师档案管理的组织方式

1.2　算法描述与分析

1.2.1　算法

在确定了数据的存储结构之后，需进一步研究与之相关的一组操作(也称运算)，主要操作有：插入、删除、排序、查找等。为了实现某种操作，例如查找，常常需要设计一种算法。算法是对特定问题求解步骤的一种描述，它是指令或语句的有限序列。故算法需要用一种语言来描述。一个算法一般具有下列五个重要特性：

(1) 有穷性。一个算法必须总是在执行有限步之后结束。

(2) 确定性。算法中的每一条指令必须有确切的含义，不能产生多义性。

(3) 可行性。算法中的每一条指令必须是切实可行的，即原则上是可以通过已经实现的基本运算执行有限次来实现的。

(4) 输入。一个算法有零个或多个输入，这些输入取自于特定对象的集合。

(5) 输出。一个算法有一个或多个输出，这些输出是同输入有某个特定关系的量。

算法可以用自然语言、计算机语言、流程图或专门为描述算法而设计的语言描述。在计算机上运行的算法，当然要用计算机语言描述。算法的设计可以避开具体的计算机程序设计语言，但算法的实现必须借助程序设计语言中提供的数据类型及其运算。数据结构与算法是相辅相成的，是利用计算机解决实际问题时不可缺少的两个方面。

本书采用 C 语言描述算法，在描述算法时做如下约定：

(1) 数据元素也称为结点、记录、顶点等，以后一般称为结点。

(2) 每一个算法以函数形式给出，一般形式是：

```
类型标识符　函数名(形式参数表)
    {说明部分
      语句组}
```

其中，形式参数表的描述采用 ANSI 新标准，即在列出“形参表列”时，同时说明形参

类型。例如：

```
int  max(int  x，int  y)
{... }
```

(3) 有关结点的类型定义、全局变量的说明等均在算法之前进行说明。

(4) 问题的规模尺寸用 M 表示，约定在宏定义中已经预先定义为：

```
#define M 1000
```

本书约定算法中用 NULL 表示“空指针”。

1.2.2 算法分析

一种数据结构的优劣是由实现其各种运算的算法体现的。评价一个算法主要看这个算法所要占用的机器资源的多少。而在这些资源中，时间和空间是两个最主要的方面，因此算法分析中最关心的也就是算法所需要的时间消耗和空间占用量，即算法的时间复杂度和空间复杂度。

所谓算法的空间复杂度(或称空间代价)指的是：当问题的规模以某种单位由 1 增至 n 时，解决该问题的算法实现所占用的空间也以某种单位由 1 增至 f(n)，则称该算法的空间复杂度是 f(n)。

语句频度指的是该语句重复执行的次数。

算法的时间复杂度指的是：算法中基本操作重复执行的次数依据算法中最大语句频度来估算，它是问题规模 n 的某个函数 f(n)，算法的时间量度记作 T(n)=O(f(n))，表示随问题规模 n 的增大，算法执行时间的增长率和 f(n)的增长率相同。时间复杂度往往不是精确的执行次数，而是估算的数量级，它着重体现的是随着问题规模 n 的增大，算法执行时间的变化趋势。

通常，表征问题规模的数量参数称为问题的尺寸，如矩阵的阶、图的顶点数等，用 n 表示。以算法中频度最大的语句频度 f(n)作为算法执行时间的一个量度。

例如：在下列三个程序段中

(1) x+=1;

此语句的频度为 1，则该程序段的时间复杂度为 T(n)=O(1)。

(2) for (i=1;i<=n;i++) x+=1;

这个程序段中，x+=1 语句要执行 n 次，语句的频度为 n，则该程序段的时间复杂度为 T(n)=O(n)。

(3) for (j=1;j<=n;j++)
  for (k=1;k<=n;k++) x+=1;

该程序段中 x+=1 语句要执行 n^2 次，语句频度为 n^2，则该程序段的时间复杂度为 $T(n)=O(n^2)$。

较常见的时间复杂度有常量阶 O(1)，线性阶 O(n)和平方阶 $O(n^2)$。其他可能出现的复杂度有对数阶 $O(\log_2 n)$,指数阶 $O(2^n)$等。

在有些算法中，基本语句的重复执行次数随问题的输入数据集而异，因而相应的时间复杂度也不同。要全面分析一个算法应该考虑它在最坏情况下的时间复杂度，最好情况下的时间复杂度和平均情况下的时间复杂度。本书除特别指明外，均指最坏情况下的时间复杂度。

对空间占用的讨论类似于时间复杂度。通常估算空间复杂度时，只考虑算法中所需用的辅助存储空间，不计算输入数据占用的空间。

复习思考题

1. 简述下列术语：数据，数据元素，数据对象，数据类型，数据结构，存储结构和算法。
2. 试写一算法，自大至小依次输出顺序读入的三个整数 x, y 和 z 的值。
3. 举出一个数据结构的例子，叙述其逻辑结构、存储结构和运算。

第 2 章　线性表

线性表是最简单、最常用的一种数据结构。

线性表的主要存储结构有两种：顺序存储结构和链式存储结构。我们把用顺序存储结构存放的线性表称作顺序表，把用链式存储结构存放的线性表称作线性链表。本章的基本内容是讨论线性表的定义、存储结构、基本运算以及实现算法。

2.1　线性表的定义和运算

2.1.1　线性表的定义

线性表是由 n(n＞0)个性质相同的数据元素组成的有限序列，记为(a_1，a_2，a_3，…，a_n)。数据元素的类型可以是高级语言提供的简单类型或由用户定义的任何类型，如整型、实型、字符型、记录类型等。表中数据元素的个数 n 定义为线性表的长度。n=0 的表称为空表，即该线性表不包含任何数据元素。

线性表可以用一个标识符来命名，如果用 A 来命名线性表，则 A＝(a_1，a_2，a_3，…，a_n)。a_1 是第一个数据元素，a_n 是最后一个数据元素。除 a_1 外，表中每一个数据元素 a_i(2≤i≤n)有且仅有一个直接前驱 a_{i-1}。除 a_n 外，表中每一个数据元素 a_i(1≤i≤n-1)有且仅有一个直接后继 a_{i+1}。数据元素在表中的位置只取决于它自身的序号，数据元素间的相对位置是线性的，因此线性表是一种线性结构。

例 1　(A，B，C，D，…，X，Y，Z)是一个线性表，其中的数据元素是英文大写字母，按字母表的顺序排列，共有 26 个数据元素。其中“A”是第一个数据元素，“Z”是最后一个数据元素，“A”是“B”的直接前驱，“B”是“A”的直接后继。“A”没有直接前驱，“Z”没有直接后继。

例 2　表 2-1 所示的学生情况表也是一个线性表，其中的数据元素是每个学生在表中所占的一行，包括学号、姓名、年龄等共 5 个数据项。通常把这种数据元素称为记录。表中所列出的学生人数 50 就是该表中数据元素的个数。

表 2-1　学生情况表

学号	姓名	性别	年龄	电话
1	张华	男	20	4277124
2	李红	女	20	6415021
⋮	⋮	⋮	⋮	⋮
50	赵思	女	19	3218662

2.1.2　线性表的运算

在线性表上常用的运算有：

(1) 存取。访问线性表的第 i 个元素，使用或改变该数据元素的值。

(2) 查找。在线性表中查找满足某种条件的数据元素。

(3) 插入。在线性表的第 i 个元素之前或在第 i 个元素之后，插入一个同类型的元素。

(4) 删除。删除线性表中第 i 个元素。

(5) 求表长。求出线性表中元素的个数。

(6) 置空表。建立一个空表。

(7) 清除表。将已有线性表变为空表，即删除表中所有元素。

(8) 分拆。把一个线性表分成多个线性表。

(9) 合并。把两个线性表合成一个线性表。

(10) 排序。对线性表中的元素按给定的关键字大小进行排序。

上述这些运算中的基本运算是查找、插入和删除。

2.2 线性表的顺序存储结构

2.2.1 线性表的顺序存储结构

线性表的顺序存储结构简称顺序表。在计算机内，可以用不同的方式来表示线性表，其中最简单、最常用的方式是顺序存储结构，即用一组连续的存储单元依次存放线性表的数据元素。这组连续的存储单元称为矢量。

假设线性表的每个元素需占用 1 个存储单元，则线性表$(a_1,a_2,a_3,\cdots,a_n)$的各个元素在内存中所占的位置如图 2-1 所示。

从图中可见，只要知道线性表的起始存储位置 b(也就是线性表第一个元素 a_1 的开始地址)，则可知表中任一元素 a_i 的存储地址为：

$$LOC(a_i)=LOC(a_1)+(i-1)$$

若每个数据元素占用 m 个存储单元，并以所占的第一个存储单元地址作为这个数据元素的存储位置，则表中任一元素 a_i 的存储地址为：

$$LOC(a_i)=LOC(a_1)+(i-1)*m$$

顺序存储结构的特点是：在线性表中逻辑关系相邻的数据元素，在计算机内存中的物理位置也是相邻的。换句话说，以数据元素在计算机内“物理位置相邻”来表示表中数据元素间的逻辑关系。对于这种存储方式，只要确定了存储线性表的起始位置，线性表中任一数据元素都可随机存取，所以线性表的顺序存储结构是一种随机存取的存储结构。

存储地址	内容	元素符号
L	a_1	1
L+1	a_2	2
…	…	…
L+(i-1)	a_i	i
…	…	…
L+(n-1)	a_n	n

图 2-1 线性表示意图

顺序表可用 C 语言的一维数组实现。数组的类型随数据元素的性质而定。

为简单起见，以后若不作特别说明，数据元素的取值范围均为整数，即可用 C 语言的整型数组描述。例如：

```
#define M 1000 int  v [M];
```

其中 v 是一个数组，它有 M 个数组元素(或称数组分量)，分量的下标从 0 开始。M 要大于实际线性表的长度 n，以便对线性表进行各种运算，特别是插入运算。线性表在数组 v 中的存储示意图如图 2-2 所示。其中数组元素从 v[n]开始到 v [M-1]是备用空间。数据元素的存储位置可用数组元素的下标值来表示。

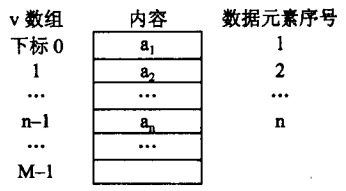


图 2-2　线性表的数组存储示意图

2.2.2　顺序表的运算

1. 插入运算　插入运算是指在线性表的第 i 个数据元素之前插入一个新的数据元素 x，使长度为 n 的线性表变成长度为 n+1 的线性表。

由于顺序表中的数据元素在计算机内是连续存放的，执行插入操作时，数据元素 a_{i-1} 和 a_i 之间的逻辑关系发生变化，而逻辑上相邻的数据元素必须存储在相邻的物理位置上，因此必须把第 n 个到第 i(共 n-i+1)个之间的所有数据元素依次向后移动一个位置，空出第 i 个位置；将 x 插入第 i 个位置，线性表长度加 1。若插入成功，函数返回值为 1；否则，函数返回值为 0。

插入算法描述如下：

```
int insertqlist(int i,int x,int v[],int *p)
{/* 在顺序表第 i 个数据元素前插入数据元素 x, 顺序表用一维数组 v 实现，*p 是指存放表长变量的指针变量 */
    int j,n;
    n=*p;
    if((i<1)||(i>n+1))
    return(0);
    else {for(j=n; j<=i; j--) v[j]=v[j-1];
          v[j]=x;
          *p=++n;
          return(1);
        }
    }
```

2. 删除运算　删除运算是指从具有 n 个数据元素的线性表中，删除其中的第 i(1<i<n)个数据元素，使线性表的长度 n 减 1。数据元素 a_{i-1} 和 a_{i+1} 之间的逻辑关系发生变化，为了在存储结构上反映这个变化，就须把表中的第 i+1 个到第 n 个之间的所有数据元素依次向前移动一个位置，以覆盖其前一个位置上的内容，线性表长度变为 n-1，若删除成功，函数返回值为 1，否则函数返回值为 0。

删除算法描述如下：

```
int delsqlist (int i, int v[],int *p)
{/*在顺序表中删除第 i 个数据元素， 顺序表用一维数组 v 实现，*p 是指存放表长变量的指针变量*/
    int j,n;
```

```
n=*p;
if(( i <1)||( i >n))
return(0);
else {for(j=i; j<=n; j++)  v[j-1]=v[j];
      *p=--n;
      return(1);
      }
}
```

从插入与删除算法中可见，插入与删除运算的时间主要耗费在元素的移动上，而移动元素的个数取决于插入或删除元素的位置。

假设 p_i 是在第 i 个元素之前插入一个元素的概率，则在长度为 n 的线性表中插入一个元素时所需移动元素的平均次数为：

$$Ein = \sum_{i=1}^{n+1} p_i (n - i + 1)$$

假设 q_i 是删除第 i 个元素的概率，则在长度为 n 的线性表中删除一个元素时所需移动元素的平均次数为：

$$Ede = \sum_{i=1}^{n} q_i (n - i)$$

如果在表的任何位置上插入或删除元素的概率相等，即

$$P_i = \frac{1}{(n+1)} \qquad q_i = \frac{1}{n}$$

则

$$Ein = \frac{1}{n+1}\sum_{i=1}^{n}(n - i + 1) = \frac{n}{2} \qquad Ede = \frac{1}{n}\sum_{i=1}^{n}(n - i) = \frac{(n-i)}{2}$$

由此可见，在顺序表中插入或删除一个数据元素时，平均约移动表中的一半数据元素。所以当 n 很大时，算法的效率是很低的。若表长为 n，则上述算法的时间复杂度为 O(n)。

2.3 线性表的链式存储结构

在线性表的顺序存储中，我们看出由于它的存储顺序与数据元素的逻辑顺序一致，所以确定任一数据元素的位置非常容易，访问任一数据元素也就非常方便。这是顺序存储方式的特点，也是它的主要优点。但是，这种存储方式也同时带来以下缺点。其一，在对线性表进行插入 、删除操作时，需移动大量数据元素，这是很费时间的，特别是当每个数据元素包含庞大的数据时，这个问题更为突出；其二，我们对表的最大长度 M 这个量很难事先准确估计出来。定得太大会浪费空间，定得太小则当表满溢出时难以补救。鉴于上述问题，本节我们考虑线性表的另一种存储结构——链式存储结构。

2.3.1 线性链表

线性表的链式存储结构是用一组任意的存储单元存放数据元素，这组存储单元可以是连续的，也可以是不连续的。为了表示数据元素间的逻辑关系，对于数据元素来说，除了要存储它本身的信息之外，还要存储指示它的直接后继的存储位置的信息。这两部分信息合起来

对应着线性表中的一个数据元素，称之为结点。结点中存储数据元素本身的域称为数据域，存储直接后继元素存储位置的域称为指针域。用指针相连接的结点序列称为链表。若链表中每个结点只包含一个指针域，则称此链表为线性链表或单链表。图 2-3 所示为线性表$(a_1,a_2,a_3,\cdots,a_n)$的线性链表存储结构，整个链表的存取必须从头指针 h 开始进行，头指针 h 指示链表中第一个结点的存储位置，同时，由于最后一个结点没有后继结点，它的指针域为空(NULL)，用∧表示。若线性表为空，则头指针 h＝NULL。

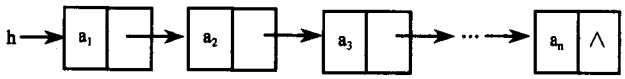


图 2-3　单链表示意图

有时为了操作方便，在单链表的第一个结点之前添加一个表头结点。通常，表头结点的结构与表中结点的结构相同，但表头结点的数据域不存放线性表元素，或者闲置不用，或者存放其他特殊信息。表头结点的链域存放单链表中含第一个数据元素的结点的指针。当表为空时，该链域为空。头指针指向表头结点，称其为带表头结点的单链表。把图 2-3 所示的单链表添上表头结点后的示意图如图 2-4 所示。

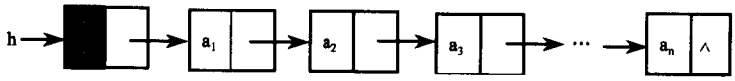


图 2-4　带表头结点的单链表

可用 C 语言中“指针”数据类型来描述线性表。定义单链表中结点类型如下：

```
typedef  struct  node {
                    datatype data;
                    struct  node *link; }NODE;
```

这里定义了一个新的 NODE 类型。

datatype 为数据域的类型，它可以是任何类型。

data 为数据域，存放该结点的数据，类型为 datatype，在以后的章节中，若不特别指明则默认为整型。

link 为指针域，指向该结点的直接后继结点。

假设 h,p,q 为指针变量，可用下列语句来说明：

```
NODE *h,*p,*q;
```

变量 p 被定义为指针型，但未指向任何实际结点，即变量中没有某实际结点所占空间的地址值。可对 p 进行“指针”赋值实现指向某一结点，使用语句：“p=(NODE*)malloc(sizeof(NODE))”可令 p 指向一个新的结点。其中 malloc(m)是 C 语言的一个库函数，其功能是在内存中分配 m 个连续可用字节的空间。函数的返回值是一个指针，该指针指向所分配空间的首地址。如果此函数未能成功地执行，则返回值为 0。如果要把结点归还系统，则用 C 语言的库函数 free(p)实现，回收后的空间可以备作再次生成结点时用。

链表和顺序存储结构不同，由于整个可用存储空间可为多个链表共同享用，每个链表占用的空间不需预先分配划定，而是可以由系统按需求即时生成，因此链表是一种动态数据结构。