数据结构(C语言描述)

SHUJU JIEGOU

(C YUYAN MIAOSHU)

李素若　陈万华　游明坤　黄桂平　编著

化学工业出版社

数据结构(C语言描述)

SHUJU JIEGOU
(C YUYAN MIAOSHU)

李素若　陈万华　游明坤　黄桂平　编著

化学工业出版社
·北京·

本书介绍了数据结构的基本概念和基本算法。全书共 11 章，主要内容包括：绪论、线性表、栈和队列、串、数组和广义表、树、图、查找、内排、文件和上机实验等。全书内容深入浅出，条理清晰，概念清楚，逻辑推理严谨，内容翔实，既注重数据结构和算法原理，又十分强调程序设计训练。书中算法都配有完整的 C 程序，程序结构清晰，构思精巧，所有程序都已在 Win-TC2.0 下编译通过并能正确运行，它们既是学习数据结构和算法的很好示例，也是很好的程序设计示例。本书配有大量的实例和图示，并有丰富的习题，适于自学。

本书是供普通高等院校计算机科学与技术专业本、专科学生使用的教材，也可供从事计算机工作者和其他希望学习数据结构的人员参考。

图书在版编目(CIP)数据

数据结构（C 语言描述）/ 李素若等编著. —北京：化学工业出版社，2009.3

ISBN 978-7-122-04728-1

Ⅰ. 数… Ⅱ. 李… Ⅲ. ①数据结构②C 语言-程序设计 Ⅳ. TP311.12 TP312

中国版本图书馆 CIP 数据核字（2009）第 010922 号

责任编辑：王听讲　　文字编辑：李　曦

责任校对：陈　静　　装帧设计：刘丽华

出版发行：化学工业出版社（北京市东城区青年湖南街 13 号　邮政编码 100011）

印　　装：北京白帆印务有限公司

787mm×1092mm　1/16　印张 17¾　字数 438 千字　　2009 年 4 月北京第 1 版第 1 次印刷

购书咨询：010-64518888（传真：010-64519686）　售后服务：010-64518899

网　　址：http: // www. cip. com. cn

凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：30.00 元

前　言

数据结构是计算机软件专业和计算机应用专业的核心课程之一，在众多的计算机系统软件和应用软件中都要用到各种数据结构。因此，仅掌握几种计算机语言是难以应付众多复杂的课题，要想有效地使用计算机，还必须学习数据结构的有关知识。

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，本课程的学习将为后续课程的学习以及软件设计水平的提高打下良好的基础。因此，数据结构课程是计算机专业的一门核心的关键性课程。

在本书的编写中，编者结合自己的教学和编程实践经验，力图用生动、通俗易懂的语言并结合编程实例来讲解各个知识点，便于读者理解和掌握。本书系统地介绍了软件设计中常用的数据结构以及相应的存储结构和实现算法，常用的多种查找和排序技术还重点介绍了外存储器上的数据结构——文件，并对其进行了性能分析和比较，内容非常丰富。全书共 11 章，第 1 章主要讲述数据结构和算法的基本概念。第 2～7 章分别讲述线性表、栈和队列、多维数组和广义表、树和图这几种基本数据结构的特点、存储方法和基本运算，作为本书的重点，书中花了大量的篇幅来介绍这些基本数据结构的实际应用。第 8 章和第 9 章讲述查找和排序的基本原理与方法。第 10 章围绕数据在外存上的组织方法介绍了文件的若干基本结构。另外，对书中所涉及的有关概念及背景知识做了详细交代；对相关的定理和性质给出了简单证明；对所有算法，都详细讨论其设计思想和实现方法，最后给出了完整的 C 语言代码，并都运行通过。

本书强调数据结构和算法的应用，教学中可通过实例具体运用各种数据结构和算法设计方法，使学生不但可以印证许多基本概念，而且能加深理解，以激发学生学习数据结构和算法的兴趣；前 10 章都配有小结和习题，便于读者掌握各章的重点和难点，并进行必要的训练；实训中为了方便学生上机实践练习，本书还专门设计了 8 套上机实验题，供学生在每章学习过后上机练习。

本书第 2～5 章、第 11 章由李素若编写，第 6 章、第 8～9 章由陈万华编写，第 7 章由游明坤编写，第 1 章、第 10 章由黄桂平编写，全书由李素若负责审核和统稿。参加本书编写大纲讨论的教师还有严永松、胡玉荣、任正云、武永成、张牧等。

由于编者水平有限，加之时间仓促，书中难免有疏漏之处，敬请广大读者批评指正，以使本书质量得到进一步提高。

编　者

2008 年 11 月

目　录

第 1 章　绪论……1
1.1　什么是数据结构……1
1.2　基本概念和常用术语……2
1.3　数据抽象和抽象数据类型……6
1.3.1　数据抽象……6
1.3.2　抽象数据类型……7
1.3.3　抽象数据类型描述和实现……8
1.4　算法和算法分析……10
1.4.1　算法及其性能标准……10
1.4.2　算法时间复杂度和渐近时间复杂度……11
1.4.3　算法的空间复杂度……13
小结……13
习题……14
第 2 章　线性表……15
2.1　线性表概念……15
2.2　线性表的顺序表示和实现……17
2.2.1　线性表的顺序存储结构……17
2.2.2　线性表在顺序存储结构下的运算……17
2.3　线性表的链式表示和实现……21
2.3.1　线性链表……21
2.3.2　循环链表……28
2.3.3　双向循环链表……29
2.3.4　顺序表和链表的比较……32
2.4　一元多项式的表示及相加……33
小结……36
习题……36
第 3 章　栈和队列……39
3.1　栈……39
3.1.1　栈的定义及其运算……39
3.1.2　顺序栈……40
3.1.3　多栈共享邻接空间……42
3.1.4　链栈……44
3.1.5　栈的应用举例……46
3.1.6　栈与递归的实现……51

3.2 队列 ……54
3.2.1 队列的定义 ……54
3.2.2 顺序队列 ……56
3.2.3 链队列 ……59
3.2.4 队列应用举例 ……60
小结 ……63
习题 ……64
第 4 章 串 ……67
4.1 串的类型定义 ……67
4.2 串的定长顺序存储 ……70
4.3 串的堆存储结构 ……73
4.3.1 串名存储映像 ……73
4.3.2 堆存储结构 ……75
4.3.3 基于堆结构的基本运算 ……75
4.4 串的块链存储结构 ……78
4.5 模式匹配 ……79
4.6 串的应用举例——正文编辑 ……84
小结 ……85
习题 ……86
第 5 章 数组和广义表 ……88
5.1 数组类型的定义 ……88
5.2 数组顺序存储和实现 ……90
5.3 矩阵压缩存储 ……92
5.3.1 对称矩阵 ……92
5.3.2 三角矩阵 ……93
5.3.3 带状矩阵 ……94
5.4 稀疏矩阵 ……95
5.4.1 稀疏矩阵三元组表存储 ……95
5.4.2 稀疏矩阵十字链表存储 ……103
5.5 广义表 ……107
5.5.1 广义表的定义和基本运算 ……107
5.5.2 广义表的存储 ……108
5.5.3 广义表基本操作的实现 ……110
小结 ……113
习题 ……113
第 6 章 树 ……115
6.1 树的基本概念 ……115
6.1.1 树的定义 ……115
6.1.2 树的逻辑表示方法 ……116
6.1.3 树的基本术语 ……117

6.1.4 树的抽象数据类型定义 ……118
6.1.5 树的存储结构 ……119
6.2 二叉树的概念和性质 ……122
6.2.1 二叉树的概念 ……122
6.2.2 二叉树的性质 ……123
6.2.3 二叉树与树、森林之间的转换 ……125
6.3 二叉树的存储结构 ……127
6.3.1 二叉树的顺序存储结构 ……127
6.3.2 二叉树的链式存储结构 ……128
6.4 二叉树的遍历 ……129
6.4.1 二叉树遍历的概念 ……129
6.4.2 二叉树遍历递归算法 ……130
6.4.3 二叉树遍历非递归算法 ……131
6.5 二叉树的基本运算及其实现 ……134
6.5.1 二叉树的基本运算 ……134
6.5.2 二叉树的基本运算算法实现 ……135
6.6 二叉树的构造 ……137
6.7 线索二叉树 ……138
6.7.1 线索二叉树的概念 ……138
6.7.2 线索化二叉树 ……139
6.7.3 遍历线索化二叉树 ……140
6.8 哈夫曼树 ……141
6.8.1 哈夫曼树的概述 ……141
6.8.2 哈夫曼树的构造算法 ……142
6.8.3 哈夫曼编码 ……143
小结 ……146
习题 ……146
第 7 章 图 ……149
7.1 图的基本概念 ……149
7.1.1 图的定义 ……149
7.1.2 图的基本术语 ……151
7.2 图的存储结构 ……152
7.2.1 邻接矩阵存储方法 ……152
7.2.2 邻接表存储方法 ……155
7.2.3 十字邻接表存储方法 ……157
7.2.4 邻接多重表存储方法 ……159
7.3 图的遍历 ……160
7.3.1 图的遍历的概念 ……160
7.3.2 深度优先搜索遍历 ……161
7.3.3 广度优先搜索遍历 ……162

7.3.4 非连通图的遍历……164
7.4 生成树和最小生成树……165
7.4.1 生成树的概念……165
7.4.2 最小生成树的定义……165
7.4.3 无向图的连通分量和生成树……166
7.4.4 有向图的强连通分量……166
7.4.5 普里姆算法……167
7.4.6 克鲁斯卡尔算法……168
7.5 最短路径……171
7.5.1 路径的概念……171
7.5.2 从一个顶点到其余各顶点的最短路径……171
7.5.3 每对顶点之间的最短路径……174
7.6 拓扑排序……176
7.7 AOE 网与关键路径……179
小结……184
习题……184
第 8 章 查找……186
8.1 查找的基本概念……186
8.2 线性表的查找……188
8.2.1 顺序查找……188
8.2.2 二分查找……189
8.2.3 分块查找……192
8.3 树表的查找……194
8.3.1 二叉排序树……194
8.3.2 平衡二叉树……201
8.3.3 B–树……210
8.3.4 B+树……214
8.4 哈希表查找……215
8.4.1 哈希表的基本概念……215
8.4.2 哈希函数构造方法……216
8.4.3 哈希冲突解决方法……218
8.4.4 哈希表上的运算……221
小结……224
习题……224
第 9 章 内排序……226
9.1 排序的基本概念……226
9.2 插入排序……227
9.2.1 直接插入排序……228
9.2.2 希尔排序……229
9.3 交换排序……231

9.3.1 冒泡排序……231
9.3.2 快速排序……233
9.4 选择排序……236
9.4.1 直接选择排序……237
9.4.2 堆排序……238
9.5 归并排序……242
9.6 基数排序……245
9.7 各种内排序方法的比较和选择……248
小结……250
习题……250
第 10 章 文件……252
10.1 文件的基本概念……252
10.2 顺序文件……254
10.3 索引文件……255
10.4 索引顺序文件……257
10.4.1 ISAM 文件……257
10.4.2 VSAM 文件……259
10.5 散列文件……261
10.6 多关键字文件……262
10.6.1 多重表文件……262
10.6.2 倒排文件……263
小结……264
习题……264
第 11 章 上机实验题……266
11.1 实验一 线性表的顺序存储结构……266
11.2 实验二 单向链表……267
11.3 实验三 双向链表……267
11.4 实验四 栈、队列……268
11.5 实验五 二叉树……269
11.6 实验六 图……270
11.7 实验七 查找……271
11.8 实验八 排序……272
参考文献……274

第1章 绪 论

“数据结构”作为一门独立的课程在国外是从1968年才开始设立的。在这之前，它的某些内容曾在其他课程，如表处理语言中有所阐述。1968年在美国一些大学的计算机系的教学计划中，虽然把数据结构规定为一门课程，但对课程的范围仍没有作明确规定。当时，数据结构几乎和图论，特别是和表、树的理论为同义语。随后，数据结构这个概念扩充到包括网络、集合代数论、格、关系等方面，从而变成了现在称之为“离散结构”的内容。然而，由于数据必须在计算机中进行处理，因此，不仅考虑数据本身的数学性质，还必须考虑数据的存储结构，这就进一步扩大了数据结构的内容。近年来，随着数据库系统的不断发展，在“数据结构”课程中又增加了文件管理（特别是大型文件的组织等）的内容。

1968年美国唐·欧·克努特教授开创了数据结构的最初体系，他所著的《计算机程序设计》第一卷《基本算法》是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作，从20世纪60年代末到70年代初，出现了大型程序，软件也相对独立，结构程序设计成为程序设计方法学的主要内容，人们就越来越重视数据结构，并认为程序设计的实质是对确定的问题选择一种好的结构，加上设计一种好的算法。从20世纪60年代中期到80年代初，各种版本的数据结构著作相继出现。

目前，在我国数据结构已经不仅仅是计算机专业教学计划中的核心课程之一，也是其他非计算机专业的主要选修课程之一。

1.1 什么是数据结构

什么是数据结构？这是一个难以直接回答的问题。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**（algorithm），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。为了说明这个问题，首先举一个例子，然后再给出明确的含义。

假定有一个学生通讯录，记录了某校全体学生的姓名和相应住址，现在要写一个算法，要求当给定任何一个学生的姓名时，该算法能够查出该学生的住址。这样一个算法的设计将完全取决于通讯录中的学生姓名及相应的住址是如何组织的，以及计算机是怎样存储通讯录中的信息的。

如果通讯录中的学生姓名是随意排列的，其次序没有任何规律，那么当给定一个姓名时，则只能对通讯录从头开始逐个与给定的姓名比较，顺序查对，直至找到所给定的姓名为止。这种方法相当费时间，效率很低。然而，若我们对学生通讯录进行适当的组织，按学生所在班级来排列，并且再造一个索引表，这个表用来登记每个班级学生姓名在通讯录中的起始位置。这样一来，情况将大为改善。这时，当要查找某学生的住址时，则可先从索引表中查到该学生所在班级的学生姓名是从何处起始的，然后就从此起始处开始查找，而不必去查看其他部分的姓名。由于采用了新的结构，于是就可以写出一个完全不相同的算法。

上述的学生通讯录就是一个数据结构问题。可以看到，计算机算法与数据的结构密切相关，算法无不依附于具体的数据结构，数据结构直接关系到算法的选择和效率。

下面再对学生通讯录作进一步讨论。我们知道，当有新学生进校时，通讯录需要添加新学生的姓名和相应的住址；在老学生毕业离校时，应从通讯录中删除毕业学生的姓名和住址。这就要求在已安排好的结构上进行插入（insert）和删除（delete）。对于一种具体的结构，如何实现插入和删除？是把要添加的学生姓名和住址插入到前头，还是末尾，或是中间某个合适的位置上？插入后，对原有的数据是否有影响？有什么样的影响？删除某学生的姓名和住址后，其他数据（学生的姓名和住址）是否要移动？若需要移动，则应如何移动？这一系列的问题说明，为适应数据的增加和减少的需要，还必须对数据结构定义一些运算。上面只涉及两种运算，即插入和删除运算。当然，还会提出一些其他可能的运算，如学生搬家后，住址变了，为适应这种需要，就应该定义修改（modify）运算等。

对于这些运算，显然是由计算机来完成，这就要设计相应的插入、删除和修改的算法。也就是说，数据结构还需要给出每种结构类型所定义的各种运算的算法。

通过以上讨论，可以直观地认为：**数据结构是研究程序设计中计算机操作的对象以及它们之间的关系和运算的一门学科。**

1.2 基本概念和常用术语

在本节中，将讲解与数据结构相关的一些重要的基本概念和常用术语。这些概念和术语将在以后的章节中多次出现。

1. 数据

数据是描述客观事物的数值、字符以及能输入机器且能被处理的各种字符的集合，即数据就是计算机化的信息。换句话说，数据就是对客观事物采用计算机能够识别、存储及处理的形式所进行的描述。

在计算机科学中，数据的含义非常广泛，我们把一切能够输入到计算机中并被计算机程序处理的信息，包括文字、表格、图像等，统称为数据。例如，一个学生成绩管理程序所要处理的数据，如表 1.1 所示。

表 1.1 学生成绩表

学　号	姓　名	数据结构	大学物理	高等数学	平均成绩
0232101	王刚	95	90	85	90
0232102	李娟	90	80	85	85
0232103	赵平	99	95	91	95
0232104	王强	86	70	84	80
0232105	张雪	92	91	84	89

2. 数据元素

数据元素也叫结点，它是组成数据的基本单位，是一个数据整体中相对独立的单元。例如，在表 1.1 所示的学生成绩中，为了便于处理，把其中的每一行（代表一个学生成绩）作为一个基本单位来考虑，故该数据由 5 个结点构成。

一般情况下，一个结点还可以分割成若干具有不同属性的字段（也叫数据项）。例如，在表 1.1 所示的表格数据中，每个结点都由学号、姓名、数据结构、大学物理、高等数学和平均

成绩 6 个字段构成。字段是构成数据的最小单位。

3. 数据对象

在数据结构中，将性质相同的数据元素的集合称之为数据对象，它是数据的一个子集。在上例中，一个班级的学生成绩表可以看作一个数据对象。

4. 数据结构

数据结构由某一数据元素集合及该集合中所有数据元素之间的关系组成。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、数据的存储结构和对数据所施加的操作。比如，在表 1.1 中，除了有 5 个学生成绩的数据外，这 5 条记录还存在着一对一的关系（逻辑结构）以及这些记录在存储器中以怎样的方式进行存储（存储结构）。

根据数据结构中数据元素之间的结构关系的不同特征，通常将数据结构分为如下四种基本结构。

（1）集合结构（set） 数据元素的有限集合。数据元素之间除了“属于同一个集合”的关系之外没有其他关系。元素顺序是随意的。

（2）线性结构（linear）或称序列（sequence）结构 数据元素的有序集合。数据元素之间形成一对一的关系。

（3）树形结构（tree） 树是层次结构，树中数据元素之间存在一对多的关系。

（4）图形结构（graph） 图中数据元素之间的关系是多对多的。

图 1.1 给出了上述四种基本结构的示意图。

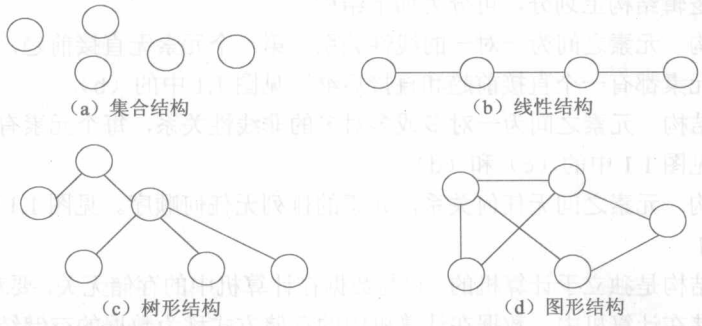

图 1.1 四种基本结构示意图

数据结构的形式定义为：数据结构是一个二元组。

Data_Structure=(D，S)

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

【例 1.1】 部门的上级领导下级的数据结构，如图 1.2 所示，a 领导 b，a 领导 c，b 领导 d，b 领导 e。则可以如下定义数据结构

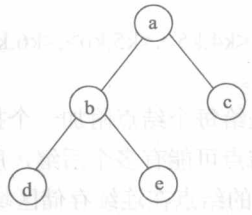

图 1.2 部门上下级领导关系图

T=(D，R)

其中，D 是数据元素的集合，D={a，b，c，d}

R 是 D 上的关系集合，R={<a，b>，<a，c>，<b，d>，<b，e>}

【例 1.2】 一小组有 a、b、c 三个学生，一个导师 A 和一个辅导员 B，此小组的数据结构如图 1.3 所示。则可以定义如下数据结构

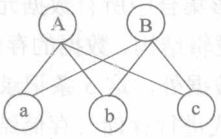

图 1.3 导师和辅导员与学生关系图

T=(D，R)

其中，D={A，B，a，b，c}

R={P1，P2}

P1={<A，a>，<A，b>，<A，c>}

P2={<B，a>，<B，b>，<B，c>}

5．逻辑结构

结点和结点之间的逻辑关系称为数据的**逻辑结构**。

数据结构以逻辑结构上划分，可分为如下结构。

（1）线性结构　元素之间为一对一的线性关系，第一个元素无直接前趋，最后一个元素无直接后继，其余元素都有一个直接前趋和直接后继。见图 1.1 中的（b）。

（2）非线性结构　元素之间为一对多或多对多的非线性关系，每个元素有多个直接前趋或多个直接后继。见图 1.1 中的（c）和（d）。

（3）集合结构　元素之间无任何关系，元素的排列无任何顺序。见图 1.1 中（a）。

6．存储结构

数据的逻辑结构是独立于计算机的，它与数据在计算机中的存储无关，要对数据进行处理，就必须将数据存储在计算机中。数据在计算机中的存储方式称为数据的**存储结构**。数据的存储结构主要有以下 4 种。

（1）顺序存储　顺序存储通常用于存储具有线性结构的数据。将逻辑上相邻的结点存储在连续存储区域 M 的相邻的存储单元中，使得逻辑相邻的结点一定是物理位置相邻。这种映象是通过物理上存储单元的相邻关系来体现结点间相邻的逻辑关系。

例如，对于一个数据结构 B=(K，R)

其中，K={k1,k2,k3,k4,k5,k6,k7,k8,k9}

R={r}

r={<k1,k2>,<k2,k3>,<k3,k4>,<k4,k5>,<k5,k6>,<k6,k7>,<k7,k8>,<k8,k9>}

它的顺序存储方式如图 1.4 所示。

（2）链式存储　链式存储方式是给每个结点附加一个指针段，一个结点的指针所指的是该结点的后继的存储地址，因为一个结点可能有多个后继，所以指针段可以是一个指针，也可以是多个指针。在链式存储中逻辑相邻的结点在连续存储区域 M 中可以不是物理相邻的。前面讲到，数据的存储结构一定要体现它的逻辑结构，这里是通过指针来体现的。

例如，数据的逻辑结构 B=(K，R)

其中，K ={k1,k2,k3,k4,k5}

R ={r}

r ={<k1,k2>,<k2,k3>,<k3,k4>,<k4,k5>}

这是一个线性结构，它的链式存储如图 1.5 所示。

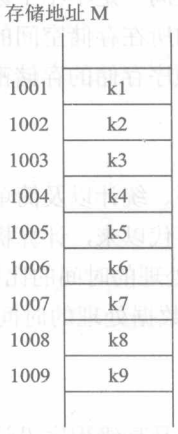

图 1.4 顺序存储的图示

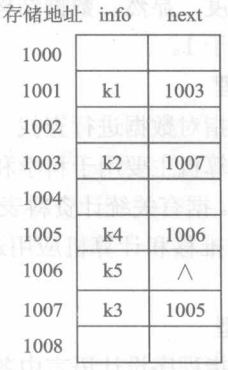

图 1.5 一个线性结构的链式存储的图示

很明显，在这种存储方式下，必须要知道开始结点的存储地址。

例如，数据的逻辑结构 B=(K，R)

其中，K ={k1,k2,k3,k4,k5}

R={r}

r={< k1,k2>,<k1,k3>,<k2,k4>,<k4,k5>}

这是一种树形结构，它的链式存储表示如图 1.6 所示。

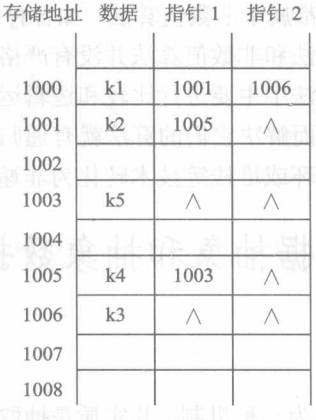

图 1.6 一个树形结构的链式存储的图示

（3）索引存储 在线性结构中，设开始结点的索引号为 1，其他结点的索引号等于其前继结点的索引号加 1，则每一个结点都有唯一的索引号，索引号就是根据结点的索引号确定该结

点的存储地址。例如，一本书的目录就是各章节的索引，目录中每个章节后面标示的页码就是该章节在书中的位置。如果某本书的每个章节所占页码相同，那么可以由一个线性函数来确定每个章节在书中的位置。

（4）散列存储　散列存储的思想是构造一个从集合 K 到存储区域 M 的一个函数 h，该函数的定义域为 K，值域为 M，K 中的每个结点 k_i 在计算机中的存储地址由 $h(k_i)$确定。

一个数据结构存储在计算机中，整个数据结构占的存储空间一定不小于数据本身所占的存储空间，通常把数据本身所占存储空间的大小与整个数据结构所在存储空间的大小之比称为数据结构的存储密度。显然，数据结构的存储密度不大于 1。顺序存储的存储密度为 1，链式存储的存储密度小于 1。

7．数据处理

数据处理是指对数据进行查找、插入、删除、合并、排序、统计以及简单计算等操作的过程。在早期，计算机主要用于科学和工程计算。20 世纪 80 年代以来，计算机主要用于各种非数值数据的处理。据有关统计资料表明，现在计算机用于数据处理的时间的比例达到 80%以上，而且随着时间的推移和计算机应用进一步普及，计算机用于数据处理的时间比例必将进一步增大。

8．数据类型

数据类型是指程序设计语言中各变量可取的数据种类，它是高级程序设计语言中的一个基本概念，和数据结构的概念密切相关。一方面，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，可以认为，数据类型是在程序设计中已经实现了的数据结构。另一方面，在程序设计过程中，当需要引入某种新的数据结构时，总是借助编程语言所提供的数据类型来描述数据的存储结构。

9．算法

简单地说，算法就是解决特定问题的方法。特定问题可以是数值的，也可以是非数值的。

解决数值问题的算法叫做数值算法。科学和工程计算方面的算法都属于数值算法，如求解数值积分、求解线性方程组、求解代数方程以及求解微分方程等。解决非数值问题的算法叫做非数值算法。数据处理方面的算法都属于非数值算法，如各种排序算法、查找算法、插入算法、删除算法以及遍历算法等。数值算法和非数值算法并没有严格的区别。一方面，在数值算法中主要进行算术运算，而在非数值算法中主要进行比较和逻辑运算。另一方面，特定的问题可能是递归的，也可能是非递归的，因而解决它们的算法就有递归算法和非递归算法之分。从理论上讲，任何递归算法都可以通过循环或堆栈等技术转化为非递归算法。

1.3　数据抽象和抽象数据类型

1.3.1　数据抽象

抽象（abstraction）可以被理解为一种机制，其实质是抽取共同的和实质的东西，忽略非本质的细节。抽象可以使求解问题过程以自顶向下的方式分步进行：首先考虑问题的最主要方面，然后再逐步细化，进一步考虑问题的某些细节，并最终实现之。

在程序设计中，数据和运算是两个不可缺少的因素。所有的程序设计活动都是围绕着数据和其上的相关运算而进行的。机器指令、汇编语言中的数据没有类型的概念，到现在的面向对

象程序设计语言中抽象数据类型概念的出现，程序设计中的数据经历了一次次抽象，数据的抽象经历了三个发展阶段。

第一个发展阶段是从无类型的二进制数到基本数据类型的产生。

第二个发展阶段是从基本数据类型到用户自定义类型的产生。

第三个发展阶段是从用户自定义类型到抽象数据类型的出现。

数据和运算（即对数据的处理）是程序设计的核心，数据表示的复杂性决定了其上运算实现的复杂性，这也是整个系统复杂性的关键所在。20 世纪 60 年代末期出现的“软件危机”就是因为在软件开发中不能有效地控制数据表示，无法控制整个软件系统的复杂性，最终导致软件系统的失败。“软件危机”使人们认识到，在基于功能抽象的模块化设计方法中，模块间的连接是通过数据进行的，数据从一个模块传送到另一个模块，每一个模块在其上施加一定的操作，并完成一定的功能。尽管 Pascal、C 等语言的用户自定义类型机制使得用户可以将这些连接数据作为某种类型的对象直接处理。然而，由于这些用户自定义类型的表示细节是对外部公开的，没有任何保护措施，程序设计人员可以随意地修改这种类型对象的某些成分，添加一些不合法的操作，而处理这个数据对象的其他模块却一无所知，从而危害整个软件系统。这些不利因素将会给由多人合作进行的大型软件系统开发产生致命的危害。为了有效地控制大型程序系统的复杂性，必须从两个方面加以了考虑。一方面是更新程序设计语言中的类型定义机制，使类型的内部表示细节对外界不可见，程序设计中不需要依赖于数据的某种具体表示；另一方面要寻求连接模块的新方法，尽可能缩小模块间的界面。面向对象程序设计语言 C++中的类就是实现抽象数据类型的机制。

1.3.2 抽象数据类型

抽象数据类型（abstract data type，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。

抽象数据类型是数据类型的进一步抽象，是大家熟知的基本数据类型的延伸和发展。众所周知，整数类型是整数值数据模型（或简单理解为整数）和加、减、乘、除四则运算等的统一体，人们在程序设计中大量使用整数类型及其相关的四则运算，而没有人去追究这些运算是如何实现的。如果人们问到此问题，可能有人会简单地回答：计算机自动进行处理。实际上，每一种基本数据类型都有一组与其相关的一组运算，而这组运算的具体实现都被封装起来，人们不知道也确实不必去关心这些细节。试想一下，如果程序设计人员在程序设计中要考虑基本数据类型的相关运算是如何具体实现的，这将是多么繁杂和令人头痛的事情，程序中的错误也会增加许多。这其中就孕育着抽象数据类型的思想。将基本数据类型的概念进行延伸，程序设计人员可以在进行问题求解时，首先确定该问题所涉及的数据模型以及定义在该数据模型上的运算集合，然后求出从数据模型的初始状态到达目标状态所需的运算序列，就成为该问题的求解算法，这样做就是一种抽象，它使得用户不必去关心数据模型中的数据具体表示及相关运算的具体实现，这将大大提高软件开发中的开发效率和程序的正确性等。

抽象数据类型是与表示无关的数据类型，是一个数据模型及定义在该模型上的一组运算。对一个抽象数据类型进行定义时，必须给出它的名字及各运算的运算符名，即函数名，并且规定这些函数的参数性质。一旦定义了一个抽象数据类型及具体实现，程序设计中就可以像使用基本数据类型那样，十分方便地使用抽象数据类型。

1.3.3 抽象数据类型描述和实现

抽象数据类型的描述包括给出抽象数据类型的名称、数据的集合、数据之间的关系和操作的集合等方面的描述。抽象数据类型的设计者根据这些描述给出操作的具体实现，抽象数据类型的使用者依据这些描述使用抽象数据类型。

抽象数据类型形式化定义可以用以下三元组表示。

ADT=(D，S，P)

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。

抽象数据类型描述的一般形式如下。

```
ADT 抽象数据类型名称 {
        数据对象：
                ……
        数据关系：
                ……
        操作集合：
                操作名 1：
                ……
                操作名 n：
}ADT 抽象数据类型名称
```

抽象数据类型的具体实现依赖于程序设计语言。面向对象的程序设计语言中的“类”支持抽象数据类型、支持信息隐藏。本书设定读者使用 C 语言进行程序设计，书中使用 C 语言进行算法的描述和实现，而 C 语言在对抽象数据类型的支持上是欠缺的，一个抽象数据类型的不同实现，如一个使用顺序存储，一个使用链式存储，尽管在不同的实现中可以使同一操作对应的函数名相同，但是抽象数据类型的操作所对应的函数原型一般是不同的。因此，要用 C 语言完整地实现抽象数据类型是不现实的。本书采用介于伪码和 C 语言之间的类 C 语言作为描述工具，有时也用伪码描述一些只含抽象操作的抽象算法。这使得数据结构与算法的描述和讨论简明清晰，不拘于 C 语言的细节，又能容易转换成 C 或者 C++程序。

本书采用了类 C 语言，精选了 C 语言的一个核心子集，同时作了若干扩充修改，增强了语言的描述功能。以下对其作简要说明。

（1）预定义常量和类型

```
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE –1
#define OVERFLOW –2
typedef in Status; //Status 是函数的类型，其值是函数结果状态代码。
```

（2）数据结构的存储结构

```
typedef ElemType first;
```