新世纪高等学校计算机系列教材

数据结构

（C语言版）

◎ 李云清 杨庆红 揭安全 编著

人民邮电出版社
POSTS & TELECOM PRESS

新世纪高等学校计算机系列教材

数据结构（C语言版）

李云清　杨庆红　揭安全　编著

人民邮电出版社

图书在版编目（CIP）数据

数据结构：C 语言版 / 李云清，杨庆红，揭安全编著. —北京：人民邮电出版社，2004.6
（新世纪高等学校计算机系列教材）
ISBN 7-115-12221-0

Ⅰ. 数… Ⅱ. ①李…②杨…③揭… 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. ①TP311.12②TP312

中国版本图书馆 CIP 数据核字（2004）第 038871 号

新世纪高等学校计算机系列教材
数据结构（C 语言版）

◆ 编　　著　李云清　杨庆红　揭安全
责任编辑　邹文波
执行编辑　王亚娜

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67129259
北京汉魂图文设计有限公司制作
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：19.5
字数：467 千字　　2004 年 6 月第 1 版
印数：1-5 000 册　　2004 年 6 月北京第 1 次印刷

ISBN 7-115-12221-0/TP・3938

定价：26.00 元

本书如有印装质量问题，请与本社联系　电话：（010）67129223

内容提要

本书介绍了数据结构的基本概念和基本算法。全书共分为12章，包括概论、线性表及其顺序存储、线性表的链式存储、字符串、数组、特殊矩阵、递归、树型结构、二叉树、图、检索、内排序、外排序和动态存储管理等内容。

本书内容丰富，逻辑性强，文字清晰流畅，既注重理论知识，又强调工程实用。书中既体现了抽象数据类型的观点，又对每个算法的具体实现给出了完整的C语言源代码描述。

与本书配套的电子教案和书中所有算法的源代码均可以从人民邮电出版社网站（www.ptpress.com.cn）上免费下载。

本书可作为高等院校计算机专业及相关专业本科生“数据结构”课程的教材，也可以作为从事计算机工程与应用的广大读者的参考书。

编者的话

数据结构是计算机学科的一门重要专业基础课，是进行程序设计的理论和技术基础，同时，对于计算机专业其他课程的学习也是十分重要的。本书可以作为高等院校计算机专业及相关专业本科生“数据结构”课程的教材。

全书共分为 12 章。第 1 章概述数据结构、抽象数据类型、算法及其分析等内容；第 2 章至第 4 章讨论线性表、字符串、数组和特殊矩阵等数据结构及其应用；第 5 章介绍递归技术；第 6 章至第 8 章讨论树型结构、二叉树、图等数据结构及其应用；第 9 章至第 11 章讨论检索、内排序和外排序等；第 12 章介绍动态存储管理技术。

本书内容丰富，逻辑性强；文字清晰流畅，实用性强。书中注重贯穿抽象数据类型的思想，在讨论一种数据结构时，都采用抽象数据类型的观点进行描述，以便学生更好地掌握理论知识，提高抽象思维的能力；另一方面，对于数据结构和算法的描述，本书并没有使用伪代码，而是采用了大多数读者熟悉的 C 语言进行描述，每个算法都用 C 语言的函数形式给出，使本书又强调了工程实用。

为了方便教师使用，编者制作了与本书配套的电子教案，教师在使用时可以根据需要对电子教案进行修改。该电子教案和书中所有算法的源代码均可以从人民邮电出版社网站上免费下载。

本书共 12 章，第 1 章、第 2 章、第 3 章、第 10 章和第 11 章由李云清撰写，第 4 章、第 5 章、第 6 章和第 7 章由杨庆红撰写，第 8 章、第 9 章和第 12 章由揭安全撰写。

在此对黄育潜教授、周定康教授、薛锦云教授、聂承启教授、黄明和教授以及滕少华副教授、敖小玲副教授等老师对本书给予的支持表示衷心的感谢。

由于编者水平有限，加上时间仓促，书中难免有错误之处，恳请同行专家及广大读者批评指正。

编　者

试读结束：需要全本请在线购买：www.ertongbook.com

CONTENTS 目录

第1章 概论 …… 1

1.1 数据结构 …… 1
1.1.1 数据结构 …… 1
1.1.2 数据的逻辑结构 …… 3
1.1.3 数据的存储结构 …… 3
1.1.4 数据的运算集合 …… 5
1.2 数据类型和抽象数据类型 …… 6
1.2.1 数据类型 …… 7
1.2.2 数据结构 …… 7
1.2.3 抽象数据类型 …… 7
1.2.4 抽象数据类型的描述和实现 …… 8
1.3 算法和算法分析 …… 9
1.3.1 算法 …… 9
1.3.2 算法的时间和空间复杂度 …… 9
习题 …… 10

第2章 线性表及其顺序存储 …… 12

2.1 线性表 …… 12
2.2 顺序表 …… 12
2.2.1 顺序表 …… 12
2.2.2 顺序表的实现 …… 13
2.3 栈 …… 18
2.3.1 栈 …… 18
2.3.2 顺序栈及其实现 …… 19
2.3.3 栈的应用之一——括号匹配 …… 21
2.3.4 栈的应用之二——算术表达式求值 …… 23
2.4 队列 …… 28

CONTENTS

2.4.1 队列 ……28
2.4.2 顺序队列及其实现 ……29
2.4.3 顺序循环队列及其实现 ……32
习题 ……34

第3章 线性表的链式存储 ……35

3.1 链式存储 ……35
3.2 单链表 ……36
3.2.1 单链表 ……36
3.2.2 单链表的实现 ……37
3.3 带头结点的单链表 ……43
3.3.1 带头结点的单链表 ……43
3.3.2 带头结点的单链表的实现 ……44
3.4 循环单链表 ……48
3.4.1 循环单链表 ……48
3.4.2 循环单链表的实现 ……49
3.5 双链表 ……56
3.5.1 双链表 ……56
3.5.2 双链表的实现 ……57
3.6 链式栈 ……64
3.6.1 链式栈 ……64
3.6.2 链式栈的实现 ……65
3.7 链式队列 ……67
3.7.1 链式队列 ……67
3.7.2 链式队列的实现 ……68
习题 ……71

第4章 字符串、数组和特殊矩阵 ……72

4.1 字符串 ……72
4.1.1 字符串的基本概念 ……72
4.1.2 字符串类的定义 ……73
4.1.3 字符串的存储及其实现 ……74
4.2 字符串的模式匹配 ……81
4.2.1 朴素的模式匹配算法 ……81
4.2.2 快速模式匹配算法 ……82
4.3 数组 ……85
4.3.1 数组和数组元素 ……85
4.3.2 数组类的定义 ……86
4.3.3 数组的顺序存储及实现 ……86

CONTENTS

4.4 特殊矩阵 …… 90
4.4.1 对称矩阵的压缩存储 …… 90
4.4.2 三角矩阵的压缩存储 …… 92
4.4.3 带状矩阵的压缩存储 …… 93
4.5 稀疏矩阵 …… 95
4.5.1 稀疏矩阵类的定义 …… 95
4.5.2 稀疏矩阵的顺序存储及其实现 …… 95
4.5.3 稀疏矩阵的链式存储及实现 …… 98
习题 …… 102

第 5 章 递归 …… 103

5.1 递归与递归程序设计 …… 103
5.2 递归程序执行过程的分析 …… 105
5.3 递归程序到非递归程序的转换 …… 108
5.3.1 简单递归程序到非递归程序的转换 …… 109
5.3.2 复杂递归程序到非递归程序的转换 …… 112
5.4 递归程序设计的应用实例 …… 116
习题 …… 118

第 6 章 树型结构 …… 120

6.1 树的基本概念 …… 120
6.2 树类的定义 …… 122
6.3 树的存储结构 …… 123
6.3.1 双亲表示法 …… 123
6.3.2 孩子表示法 …… 124
6.3.3 孩子兄弟表示法 …… 127
6.4 树的遍历 …… 127
6.5 树的线性表示 …… 131
6.5.1 树的括号表示 …… 131
6.5.2 树的层号表示 …… 133
习题 …… 135

第 7 章 二叉树 …… 137

7.1 二叉树的基本概念 …… 137
7.2 二叉树的基本运算 …… 139
7.3 二叉树的存储结构 …… 140
7.3.1 顺序存储结构 …… 140
7.3.2 链式存储结构 …… 142
7.4 二叉树的遍历 …… 144

CONTENTS

7.4.1 二叉树遍历的定义 …… 144
7.4.2 二叉树遍历的递归实现 …… 144
7.4.3 二叉树遍历的非递归实现 …… 146
7.5 二叉树其他运算的实现 …… 150
7.6 穿线二叉树 …… 152
7.6.1 穿线二叉树的定义 …… 152
7.6.2 中序穿线二叉树的基本运算 …… 153
7.6.3 中序穿线二叉树的存储结构及其实现 …… 154
7.7 树、森林和二叉树的转换 …… 156
7.7.1 树、森林到二叉树的转换 …… 156
7.7.2 二叉树到树、森林的转换 …… 157
习题 …… 158

第8章 图 …… 159

8.1 图的基本概念 …… 159
8.2 图的基本运算 …… 162
8.3 图的基本存储结构 …… 163
8.3.1 邻接矩阵及其实现 …… 163
8.3.2 邻接表及其实现 …… 166
8.3.3 邻接多重表 …… 168
8.4 图的遍历 …… 169
8.4.1 深度优先遍历 …… 169
8.4.2 广度优先遍历 …… 171
8.5 生成树与最小生成树 …… 173
8.5.1 最小生成树的定义 …… 174
8.5.2 最小生成树的普里姆（Prim）算法 …… 175
8.5.3 最小生成树的克鲁斯卡尔（Kruskal）算法 … 178
8.6 最短路径 …… 180
8.6.1 单源最短路径 …… 180
8.6.2 所有顶点对的最短路径 …… 183
8.7 拓扑排序 …… 186
8.8 关键路径 …… 189
习题 …… 194

第9章 检索 …… 196

9.1 检索的基本概念 …… 196
9.2 线性表的检索 …… 197
9.2.1 顺序检索 …… 197
9.2.2 二分法检索 …… 199

CONTENTS

9.2.3 分块检索 …… 201
9.3 二叉排序树 …… 203
9.4 丰满树和平衡树 …… 210
9.4.1 丰满树 …… 211
9.4.2 平衡二叉排序树 …… 212
9.5 最佳二叉排序树和 Huffman 树 …… 218
9.5.1 扩充二叉树 …… 218
9.5.2 最佳二叉排序树 …… 220
9.5.3 Huffman 树 …… 225
9.6 B-树 …… 228
9.6.1 B-树的定义 …… 228
9.6.2 B-树的基本操作 …… 229
9.7 散列表检索 …… 234
9.7.1 散列存储 …… 234
9.7.2 散列函数的构造 …… 235
9.7.3 冲突处理 …… 236
习题 …… 240

第 10 章 内排序 …… 242

10.1 排序的基本概念 …… 242
10.2 插入排序 …… 243
10.2.1 直接插入排序 …… 243
10.2.2 二分法插入排序 …… 246
10.2.3 表插入排序 …… 248
10.2.4 Shell 插入排序 …… 249
10.3 选择排序 …… 251
10.3.1 直接选择排序 …… 251
10.3.2 树型选择排序 …… 253
10.3.3 堆排序 …… 256
10.4 交换排序 …… 259
10.4.1 冒泡排序 …… 259
10.4.2 快速排序 …… 261
10.5 归并排序 …… 263
10.6 基数排序 …… 267
10.6.1 多排序码的排序 …… 267
10.6.2 静态链式基数排序 …… 267
习题 …… 271

CONTENTS

第 11 章　外排序 …… 273

11.1　外存储器简介 …… 273
11.1.1　磁盘存储器 …… 273
11.1.2　磁带存储器 …… 273
11.2　文件简介 …… 274
11.2.1　文件的逻辑结构 …… 274
11.2.2　文件的存储结构 …… 274
11.3　外排序——磁盘排序 …… 274
11.3.1　磁盘排序 …… 274
11.3.2　多路归并 …… 276
11.3.3　初始有序串的生成 …… 278
11.4　外排序——磁带排序 …… 279
11.4.1　磁带排序 …… 279
11.4.2　非平衡归并 …… 281
习题 …… 282

第 12 章　动态存储管理 …… 283

12.1　概述 …… 283
12.2　可利用空间表及分配方法 …… 285
12.3　边界标识法 …… 288
12.3.1　可利用空间表的结构 …… 288
12.3.2　分配算法 …… 289
12.3.3　回收算法 …… 291
12.4　无用单元的收集 …… 293
12.5　存储压缩 …… 296
习题 …… 298

参考文献 …… 299

试读结束：需要全本请在线购买：　www.ertongbook.com

第 1 章

概　　论

数据结构讨论的是数据的逻辑结构、存储方式以及相关操作。本章讲述数据结构的基本概念及相关术语，介绍数据结构、数据类型和抽象数据类型之间的联系，介绍算法的特点及算法的时间与空间复杂度。

1.1　数据结构

1.1.1　数据结构

人们常把计算机称为数据处理机，在计算机问世的初期，计算机所处理的数据基本上都是数值型数据，也就是说，计算机发展的初期主要是用于数值计算，那时的软件设计者将主要精力用于程序设计的技巧上，而对如何在计算机中组织数据并不需要花费太多的时间和精力。然而，随着计算机软、硬件的发展，计算机的应用范围在不断扩大，计算机处理数据的数量也在不断扩大，计算机处理的数据已不再是单纯的数值数据，而更多的是非数值数据。此时，如果仅在程序设计技巧上花功夫，而不去考虑数据的组织，那么，对大量数据的处理将会是十分低效的，有时甚至是无法进行的。

需要处理的数据并不是杂乱无章的，它们一定有内在的联系，只有弄清楚它们之间本质的联系，才能使用计算机对大量的数据进行有效的处理。

例如，某电信公司的市话用户信息表如图 1.1 所示。

序号	用户名	电话号码	用户住址	
			街道名	门牌号
00001	万方林	33800***	北京西路	1659*
00002	吴金平	33800***	北京西路	2099*
00003	王　冬	55700***	瑶湖大道	1987*
00004	王　三	55700***	瑶湖大道	2008*
00005	江　凡	68800***	学府大道	5035*

图 1.1　用户信息表

对于上面的数据，每一行是一个用户的有关信息，它由序号、用户名、电话号码和用户住址等项组成。序号、用户名和电话号码等项称为基本项，是有独立意义的最小标识单位，而用户住址称为组合项，组合项由一个或多个基本项或组合项组成，是有独立意义的标识单位。这里的每一行称为一个结点，每一个组合项称为一个字段。结点是由若干个字段构成的。对于能惟一地标识一个结点的字段或几个字段的组合，如这里的序号字段，称为关键码。当要使用计算机处理用户信息表中的数据时，必须弄清楚下面 3 个问题。

1. 数据的逻辑结构

这些数据之间存在什么样的内在联系？在这些数据中，有且只有一个结点是表首结点，它前面没有其他结点，后面有一个和它相邻的结点；有且只有一个结点是表尾结点，它后面没有其他结点，前面有一个和它相邻的结点；除这两个结点之外，表中所有其他的结点都有且仅有一个和它相邻的位于它之前的结点，也有且仅有一个和它相邻的位于它之后的结点。上述这些就是用户信息表的逻辑结构。

2. 数据的存储结构

数据在计算机中的存储方式称为存储结构。将用户信息表中的所有结点存入计算机时，就必须考虑存储结构。使用 C 语言进行设计时，常见的方式是用一个结构数组来存储整个用户信息表，每一个结构数组元素是一个结构，它对应于用户信息表中的一个结点。用户信息表中相邻的结点，对应的结构数组元素也是相邻的，或者说在这种存储方式下，逻辑相邻的结点就必须物理相邻。这是一种被称为顺序存储的方式，当然，还有其他的存储方式。

3. 数据的运算集合

对数据的处理必定涉及到相关的运算。在上述用户信息表中，可以进行删除一个用户、增加一个用户和查找某个用户等操作。应该明确指出这些操作的含义。比如删除操作，是删除序号为 5 的用户还是删除用户名为王三的用户是应该明确定义的，如果需要可以定义两个不同的删除操作。为一批数据定义的所有运算（或称操作）构成一个运算（操作）集合。

对于一批待处理的数据，只有分析清楚上面 3 个方面的问题，才能进行有效的处理。

数据结构就是指按一定的逻辑结构组成的一批数据，使用某种存储结构将这批数据存储于计算机中，并在这些数据上定义了一个运算集合。

在讨论一个数据结构时，数据结构所含的 3 个方面缺一不可，也就是说，只有给定一批数据的逻辑结构和它们在计算机中的存储结构，并且定义了数据运算集合，才能确定一个数据结构。例如，在后面的章节中将要介绍的栈和队列，它们的逻辑结构是一样的，它们都可以用同样的存储结构，但是由于所定义的运算性质不同，它们成为两种不同的数据结构。

1.1.2　数据的逻辑结构

数据的逻辑结构是数据和数据之间所存在的逻辑关系，它可以用一个二元组：

$$B=(K, R)$$

来表示，其中 K 是数据，即结点的有限集合；R 是集合 K 上关系的有限集合，这里的关系是从集合 K 到集合 K 的关系。在本书的讨论中，一般只涉及到一个关系的逻辑结构。

例如，有 5 个人，分别记为 a，b，c，d，e，其中 a 是 b 的父亲，b 是 c 的父亲，c 是 d 的父亲，d 是 e 的父亲，如果只讨论他们之间存在的父子关系，则可以用下面的二元组形式化地予以表达：

$$B=(K, R)$$

其中 $K=\{a, b, c, d, e\}$

$R=\{r\}$

$r=\{<a, b>, <b, c>, <c, d>, <d, e>\}$

也可以用图形的方式表示数据的逻辑结构，K 中的每个结点 k_i 用一个方框表示，而结点之间的关系用带箭头的线段表示。这 5 人之间的逻辑结构用图形的方式表达，如图 1.2 所示。

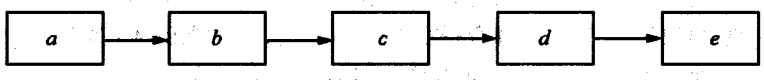

图 1.2　数据的逻辑结构图

若 $k_i \in K$，$k_j \in R$，$<k_i, k_j> \in r$，则称 k_i 是 k_j 的相对于关系 r 的前驱结点，k_j 是 k_i 的相对于关系 r 的后继结点。因为一般只讨论具有一种关系的逻辑结构，即 $R=\{r\}$，所以简称 k_i 是 k_j 前驱，k_j 是 k_i 的后继。如果某个结点没有前驱结点，称之为开始结点；如果某个结点没有后继结点，称之为终端结点；既不是开始结点也不是终端结点的结点称为内部结点。

对于一个逻辑结构 $B=(K, R)$，如果它只有一个开始结点和一个终端结点，而其他的每一个结点有且仅有一个前驱和一个后继，称为线性结构。如果它有一个开始结点，有多个终端结点，除终端结点外，每一个结点有且仅有一个前驱，称为树形结构。如果每个结点都可以有多个前驱和后继，称为图形结构。树形结构和图形结构都是非线性结构。

图 1.3 所示的是一个有 7 个结点的数据结构的逻辑关系的图形表示。

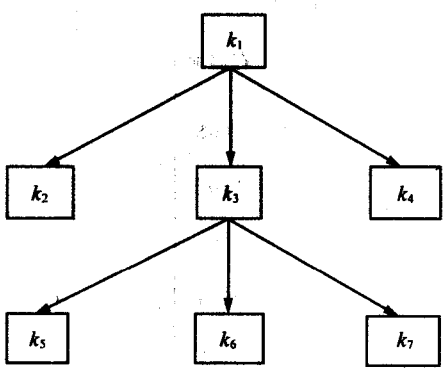

图 1.3　一个逻辑关系的图形表示

这里 k_1 是开始结点，k_2，k_4，k_5，k_6，k_7 是终端结点，k_3 是内部结点。

以后，在不引起误解的情况下，我们把数据的逻辑结构简称为数据结构。

1.1.3　数据的存储结构

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，如果将数据在计算机中无规律地存储，那么在处理时会

非常糟的，是没有用的。试想一下，如果一本英汉字典中的单词是随意编排的，那这本字典谁也不会用。对于一个数据结构 $B=(K, R)$，必须建立从结点集合到计算机某个存储区域 M 的一个映象，这个映象要直接或间接地表达结点之间的关系 R。如前所述，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 $K=\{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9\}$

$R=\{r\}$

$r=\{<k_1, k_2>, <k_2, k_3>, <k_3, k_4>, <k_4, k_5>, <k_5, k_6>, <k_6, k_7>, <k_7, k_8>, <k_8, k_9>\}$

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 $K=\{k_1, k_2, k_3, k_4, k_5\}$

$R=\{r\}$

$R=\{<k_1, k_2>, <k_2, k_3>, <k_3, k_4>, <k_4, k_5>\}$

这是一个线性结构。它的链式存储如图 1.5 所示。

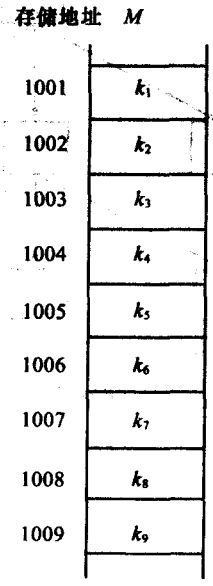

图 1.4 顺序存储的图示

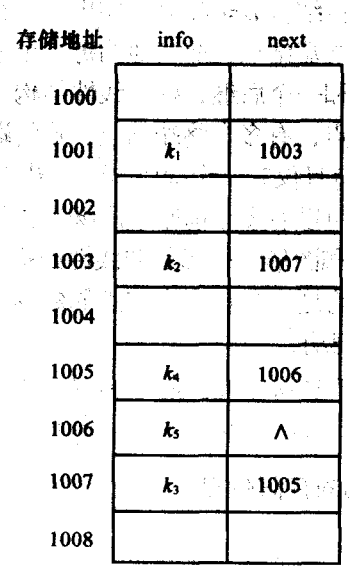

图 1.5 一个线性结构的链式存储图示

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 $K=\{k_1, k_2, k_3, k_4, k_5\}$

$R=\{r\}$

$R=\{<k_1, k_2>, <k_1, k_3>, <k_2, k_4>, <k_4, k_5>\}$

这是一树型结构，它的链式存储表示如图 1.6 所示

存储地址	数据	指针 1	指针 2
1000	k_1	1001	1006
1001	k_2	1005	∧
1002			
1003	k_5	∧	∧
1004			
1005	k_4	1003	∧
1006	k_3	∧	∧
1007			
1008			

图 1.6　一个树型结构的链式存储表示

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，则每一个结点都有惟一的索引号。索引存储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总数相同，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，整个数据结构所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占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。

1.1.4　数据的运算集合

对于一批数据，数据的运算是定义在数据的逻辑结构之上的，而运算的具体实现依赖于数据的存储结构。

例如，在一个线性结构中查找一个值为 x 的结点，可以这样定义查找运算："从该结构的第 1 个结点开始，将结点值与 x 比较，如果相等则查找成功结束；如果不相等，则沿着该结点的后继继续比较，直到找到一个满足条件的结点成功结束，或找遍所有结点都没有找到而以查找失败结束。"

如果一个线性结构采用顺序存储，比如用一个一维数组存放，则查找运算中一个结点的后继是通过数组下标的递增实现的，如 $i=i+1$。如果采用链序存储，一个结点后继是通过形如 p=p->link 的方式来实现的。

数据的运算集合要视情况而定，一般而言，数据的运算包括插入、删除、检索、输出和排序等。

插入是指在一个结构中增加一个新的结点。

删除是指在一个结构中删除一个结点。
检索是指在一个结构中查找满足条件的结点。
输出是指将一个结构中所有结点的值打印、输出。
排序是指将一个结构中所有结点按某种顺序重新排列。

1.2 数据类型和抽象数据类型

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相关运算而进行的。从机器指令、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。在机器语言中，程序设计中所涉及的一切数据，包括字符、数、指针、结构数据和程序等都是由程序设计人员用二进制数字表达的，没有类型的概念。人们难以理解、辨别这些由0和1所组成的二进制数表达的意思是什么。这种程序的易读性、可维护性和可靠性极差。随着计算机技术的发展，出现了Fortran、Algol高级程序设计语言，在这些高级程序设计语言中引入了整型、实型和布尔类型等基本数据类型，程序员可以将其他的数据对象建立其上，避免了复杂的机器表示。这样，程序员不必和繁杂的二进制数字直接打交道，就能完成相应的程序设计任务。数据类型就像一层外衣，它反映了一个抽象层次，使得程序设计人员只需知道如何使用整数、实数和布尔数，而不需要了解机器的内部细节。此外，高级程序设计语言的编译程序可以利用类型信息进行类型一致性检查，及早发现程序中的语法错误。

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。Fortran语言等只是引入了有限的整型、实型和布尔型等基本的类型，而在程序的开发过程中，许多复杂的数据对象难以用这些基本的类型表示，这就给程序的设计带来了很大的困难。例如，程序要实现一个数组栈（即栈中的元素为数组）上的操作，不能将数组作为基本对象来处理，必须通过其下标变量对数组的分量逐个处理。如果数组的分量又是一个数组，则还必须对这个用作分量的数组的分量逐一处理，直到最底层的整数、实数或布尔数这些基本类型的数据为止。也就是说，虽然经过第一个阶段的发展，用户可以不必涉及机器内部细节，但是，仅仅引入几种基本类型，用户在处理复杂的数据时，仍要涉及其数据结构的细节。如果可以把所要处理的数据对象作为某种类型的对象来直接处理，而不必涉及其数据表示细节，将会给控制程序的复杂性带来很大的益处，并且编译程序的类型检查机制可以及早发现错误。为此，PL/I曾试图引入更多的基本类型（如数组、树和栈等），以便将数组、树和栈作为直接处理的数据对象。但这不是解决问题的好方法。因为，一个大型系统所涉及的数据对象极其复杂，任何一个程序语言都没有办法使所有的类型作为其基本的类型。解决问题的根本方法是，程序设计语言必须提供这样一种机制，程序员可以依据具体问题，灵活方便地定义新的数据类型，即用户定义类型的机制。在大家熟知的Pascal和C语言中就引入了用户定义类型的机制，程序中允许有用户自己定义的新类型。

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。抽象数据类型是用户自己定义类型的一个机制。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

试读结束：需要全本请在线购买： www.ertongbook.com