

数据结构教程 (C语言版)

北京科海 总策划

李春葆 曾平 喻丹丹 编著

- 介绍各种最常用的数据结构，从编程角度出发，分析各种数据结构运算算法的实现过程
- 教育专家和一线教师精心编写，融入作者多年的教学经验
- 结构清晰，实例丰富，具有很强的操作性和实用性

国家“十一五”高等院校计算机应用型规划教材

数据结构教程 (C语言版)

李春葆 曾 平 喻丹丹 编著

中国人民大学出版社

• 北京 •

北京科海电子出版社

www.khpc.com.cn

图书在版编目(CIP)数据

数据结构教程 (C 语言版) / 李春葆等编著.

北京: 中国人民大学出版社, 2009

国家“十一五”高等院校计算机应用型规划教材

ISBN 978-7-300-10155-2

I. 数…

II. 李…

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. TP311.12 TP312

中国版本图书馆 CIP 数据核字 (2008) 第 204430 号

国家“十一五”高等院校计算机应用型规划教材

数据结构教程 (C 语言版)

李春葆 曾平 喻丹丹 编著

出版发行 中国人民大学出版社 北京科海电子出版社

社 址 北京中关村大街 31 号

邮政编码 100080

北京市海淀区上地七街国际创业园 2 号楼 14 层

邮政编码 100085

电 话 (010) 82896442 62630320

网 址 <http://www.crup.com.cn>

<http://www.khp.com.cn> (科海图书服务网站)

经 销 新华书店

印 刷 北京市艺辉印刷有限公司

规 格 185 mm×260 mm 16 开本

版 次 2009 年 2 月第 1 版

印 张 14.75

印 次 2009 年 2 月第 1 次印刷

字 数 359 000

定 价 25.00 元

版权所有 侵权必究 印装差错 负责调换

丛书序

计算机教材建设是计算机专业教学工作的重要组成部分,高质量的教材是培养高素质人才的基本保证,是体现教育特色的知识载体和教学的基本工具,直接关系到计算机专业教育能否为一线岗位培养符合要求的高技术应用型人才。教育部也把教材建设作为衡量高等院校深化教育教学改革的重要指标,作为检验各院校人才培养工作的标准。近年来,许多院校都十分重视计算机专业教材建设工作,编写和出版了一批质量较高的精品教材,但仍然远远满足不了应用型教育发展的需要,所以我们组织了由全国高校计算机专业的专家教授组成的国家“十一五”高等院校计算机应用型规划教材课题研究组,通过对应用型本科院校和高职高专院校计算机应用技术专业全面、细致的调研和讨论,并结合我国当前的教学现状,编写了本丛书。丛书突出系统性、科学性和实践性,以培养社会需求的计算机应用型专门人才为宗旨。

丛书特色

☑ 课程体系的系统性: 注重教学内容和体系的创新

本丛书根据教育部颁布的应用型专门人才培养目标来编写,适合应用型本科院校和高职高专院校的教学需求和教学特色。基础理论型课程体现以应用为目的,以必需、够用为度,以讲清概念、强化应用为教学重点;专业技术型课程强调实用性,以满足社会需求为目标,以强化实践操作为教学重点。

☑ 教学方法的先进性: 加强全方位的教学配套资源建设

本丛书针对计算机专业教学工作量大、课时多、讲授课程种类全的特点,注重资源和手段的改革,并逐步建立专门的论坛网站,为计算机专业教学提供一个现代化的平台,包括教材推荐和评论、学生提问和教师答疑、教师课程教学博客、教学论文发表、教学实验基地信息发布等功能。

☑ 教学内容的多样性: 力求介绍最新的技术和方法

先进性和时代性是教材的生命,计算机应用技术专业的教学具有更新快、内容多的特点,本丛书在体例安排和实际讲述过程中都力求介绍最新的技术和方法,并注重拓宽学生的知识面,激发他们的学习热情和创新欲望。

☑ 理论与实践并重: 阐明基础理论, 强调实践应用

理论是实践的基础,实践是理论的升华;不能有效指导实践的理论是空头理论,没有理论指导的实践是盲目的实践。对于时代呼唤的信息化人才而言,二者缺一不可。本丛书以知识点为主线,穿插演示性案例于理论讲解之中,使枯燥的理论变得更易于理解、易于接受;此外,还在每一章的最后提供大量的练习题和综合示例,以提高学生综合利用所学知识解决实际问题的能力。

☑ 易教易学: 创新体例, 合理布局, 通俗易懂

本丛书结构清晰,内容系统详实,布局合理,语言精炼实用(不讲深奥的原理),实例难度适中;力求把握各门课程的核心,通俗易懂,便于教学的展开,也便于学生融会贯通,熟练掌握所学知识。

☑ 版式设计：简洁大方

精心设计的版式简洁、大方和实用。对于标题、正文、注释、技巧等都设计了醒目的字体，读者阅读起来会感到轻松愉快。

涵盖领域

本丛书涵盖了计算机各个应用领域，包括：

- 计算机操作基础
- 计算机硬件基础
- 程序设计技术
- 数据库应用技术
- 计算机网页设计与制作
- 计算机网络技术
- 图形图像
- 软件工程

丛书编委会

主编：李春葆

副主编：曾平 金晶 赵丙秀

编委：余云霞 董尚燕 张牧 黎永壹 孙扬波 冯春辉 刘宇君 冯晋军 喻丹丹

教学服务与支持

本丛书的出版者和作者竭诚为读者提供服务。网络支持与服务网址为 <http://www.khp.com.cn>。包括：

- 提供实用的相关资源与最新信息，读者可以方便地下载本丛书的实例源代码及相关教学素材。
- 作者和专家邮件答疑（E-mail: khservice@khp.com.cn），将努力高效快捷地解决读者在图书使用和学习中遇到的疑难问题。
- 免费为教师提供的 PowerPoint 演示文档，该文档可将书中的内容及图片以幻灯片的形式呈现在学生面前，在很大程度上减轻了教师的备课负担，所以深受广大教师的欢迎。请用书教师致电：010-82896438 或发 E-mail: feedback@khp.com.cn 获取电子教案。

编者寄语

教学改革是教育工作不变的主题。要紧跟教学改革，不断创新，编写出真正满足新形势下教学需求的教材，还需要我们不断地努力实践、探索和完善。本丛书的作者和出版者虽然竭尽全力进行细致的编写与校订，仍难免有疏漏和不足，我们真诚希望使用本丛书的教师和学生提出宝贵的意见和建议，以便能不断改进和日臻完善。

本丛书作者的电子邮件：licb1964@126.com

本丛书出版者的电子邮件：feedback@khp.com.cn

丛书编委会

2009年1月

内容提要

本书根据高等院校计算机专业数据结构课程的教学大纲要求，由教育专家和一线教师精心编写，融入作者多年的教学经验。全书按照课程的授课顺序，阐述了线性表、栈和队列、串和数组、树和二叉树、图、查找、排序等内容。

本书注重实用性和可读性，对概念原理的阐述准确、精炼、通俗易懂；在介绍数据结构的基本算法时，不仅介绍了算法思路，更注意程序的实现过程；源程序都经过上机验证，准确无误；每章最后都配备了大量的练习题，并在附录中给出了详细的参考答案，使学生能够深化对基本概念的理解，提高分析问题、解决问题的能力。

书中提到的参考文件均可从 <http://www.khp.com.cn> 网站免费下载，以便于读者参阅，快速提高编程能力。

本书可作为应用型本科、示范性高职高专以及计算机培训学校计算机相关课程的教材，也可以作为计算机专业成人教育、自考考试的教材。

前言

“数据结构”是计算机及相关专业的一门专业基础课，它主要讨论数据组织方法和典型问题求解策略，学习本课程需要具备一定的 C 语言程序设计知识。本书由教育专家和一线教师精心编写，融入作者多年的教学经验，并结合计算机专业的相关教学大纲，介绍了各种最常用的数据结构，讨论它们在计算机中的存储结构，以及这些数据结构的操作和实际的算法。

本书力求在阐明各种数据结构的内在逻辑关系、存储结构和相关运算的同时，从编程角度出发，分析各种数据结构运算算法的实现过程，不仅能使读者掌握数据结构涵盖的理论基础知识，更重要的是能使读者得到程序设计的训练和编程能力的提高。

本书共分为 8 章：第 1 章是概论部分，讨论数据结构的基本概念和算法描述；第 2 章是线性表，讨论线性表的逻辑结构、线性表的顺序存储和链式存储；第 3 章是栈和队列，讨论栈和队列的特点及其各种存储结构与基本运算的实现，并给出了相应的应用实例；第 4 章是串和数组，讨论串的各种存储结构及其基本运算的实现、数组和稀疏矩阵的各种存储结构及其基本运算的实现；第 5 章是树和二叉树，讨论树的定义与表示、二叉树的基本操作和哈夫曼树；第 6 章是图，讨论图的各种存储结构和遍历的实现；第 7 章是查找，讨论了各种常用的查找方法及其实现；第 8 章是排序，讨论了各种常用排序的方法及其实现。每章都提供了一定数量的练习题，供读者练习。本书最后是两个附录，附录 A 给出了各章练习题的参考答案，附录 B 给出了本书中 C/C++ 程序的功能索引。

本书具有很强的实用性和可读性，概念原理的阐述力求准确、精炼，写作风格上尽量通俗易懂。书中提到的相关文件均可从 <http://www.khp.com.cn> 网站免费下载，以便于读者参阅，快速提高编程能力。

本书可作为应用型本科、示范性高职高专以及计算机培训学校计算机相关课程的教材，也可以作为计算机专业成人教育、自学考试教材。

尽管编者不遗余力，但由于水平所限，仍可能存在不足之处，敬请读者批评指正。

编者

2009 年 1 月

目 录

第 1 章 概论	1
1.1 数据结构	1
1.1.1 数据的逻辑结构	1
1.1.2 数据的存储结构	3
1.1.3 数据的运算	6
1.1.4 数据结构和数据类型	6
1.2 算法和算法分析	8
1.2.1 算法及其特征	8
1.2.2 算法描述	9
1.2.3 算法分析	11
1.3 本章小结	12
练习题 1	12
第 2 章 线性表	14
2.1 线性表的基本概念	14
2.1.1 线性表的定义	14
2.1.2 线性表的基本运算	14
2.2 线性表的顺序存储结构	15
2.2.1 顺序表的定义	15
2.2.2 顺序表的基本运算	16
2.2.3 顺序表相关算法的分析	21
2.2.4 顺序表的应用举例	21
2.3 线性表的链式存储结构	22
2.3.1 单链表	23
2.3.2 循环单链表	30
2.3.3 双链表	35
2.3.4 循环双链表	40
2.4 链表的应用——两个多项式相加	44
2.4.1 设计多项式存储结构	44
2.4.2 设计多项式基本运算	45
2.4.3 设计多项式相加运算	47
2.4.4 设计主函数	49

2.5 本章小结	50
练习题 2	50
第 3 章 栈和队列	51
3.1 栈	51
3.1.1 栈的基本概念	51
3.1.2 栈的顺序存储及其基本运算	52
3.1.3 栈的链式存储及其基本运算	56
3.1.4 栈的应用	59
3.2 队列	61
3.2.1 队列的基本概念	61
3.2.2 队列的顺序存储及其基本运算	62
3.2.3 队列的链式存储及其基本运算	67
3.2.4 队列的应用	71
3.3 本章小结	73
练习题 3	73
第 4 章 串和数组	74
4.1 串	74
4.1.1 串的定义	74
4.1.2 串的顺序存储及其基本运算	75
4.1.3 串的链式存储及其基本运算	81
4.2 数组	87
4.2.1 数组的定义	87
4.2.2 数组存储的排列顺序	88
4.2.3 数组的基本运算	88
4.2.4 特殊矩阵的压缩存储	89
4.3 稀疏矩阵	91
4.3.1 稀疏矩阵的三元组表示	91
4.3.2 稀疏矩阵的十字链表表示	95
4.4 本章小结	97
练习题 4	97
第 5 章 树和二叉树	99
5.1 树	99
5.1.1 树的定义	99
5.1.2 树的表示	100
5.1.3 树的基本术语	101
5.1.4 树的存储结构	101

5.2 二叉树	103
5.2.1 二叉树的定义	103
5.2.2 二叉树的性质	104
5.2.3 二叉树的存储结构	106
5.2.4 递归算法设计方法	108
5.2.5 二叉树的基本运算	109
5.2.6 二叉树的遍历及其应用	115
5.2.7 二叉树的构造	119
5.2.8 二叉树、树及森林之间的转换	121
5.3 哈夫曼树	124
5.3.1 哈夫曼树的定义	124
5.3.2 哈夫曼树的构造	125
5.3.3 哈夫曼编码	127
5.4 本章小结	129
练习题 5	129
第 6 章 图	131
6.1 图的基本概念	131
6.1.1 图的定义	131
6.1.2 图的基本术语	132
6.2 图的存储结构	134
6.2.1 邻接矩阵	134
6.2.2 邻接表	137
6.3 图的遍历	141
6.3.1 广度优先搜索	141
6.3.2 深度优先搜索	143
6.3.3 图遍历算法的应用	145
6.4 最小生成树	146
6.4.1 普里姆算法	146
6.4.2 克鲁斯卡尔算法	149
6.5 最短路径	152
6.5.1 单源最短路径	152
6.5.2 每对顶点之间的最短路径	154
6.6 拓扑排序	158
6.7 AOE 网与关键路径	160
6.8 本章小结	163
练习题 6	163

第7章 查找	164
7.1 顺序查找	164
7.2 二分查找	166
7.3 分块查找	168
7.4 二叉排序树查找	170
7.4.1 二叉排序树的基本概念	170
7.4.2 二叉排序树的基本运算	171
7.5 哈希表查找	176
7.5.1 哈希表查找的基本概念	176
7.5.2 哈希函数的构造方法	176
7.5.3 哈希冲突的解决方法	178
7.6 本章小结	185
练习题7	186
第8章 排序	187
8.1 排序的基本概念	187
8.2 插入排序	187
8.2.1 直接插入排序	188
8.2.2 希尔排序	190
8.3 选择排序	191
8.3.1 直接选择排序	191
8.3.2 堆排序	193
8.4 交换排序	196
8.4.1 冒泡排序	197
8.4.2 快速排序	198
8.5 归并排序	200
8.6 基数排序	203
8.7 本章小结	205
练习题8	206
附录A 练习题参考答案	207
附录B C/C++程序的功能索引	224
参考文献	226

第 1 章

概 论

数据的组织形式和表示方式直接关系到计算机对数据的处理效率，因此为了更好地进行程序设计，有效地利用计算机资源，就需要对计算机程序加工处理的对象进行系统和深入的研究。研究各种数据的特性及数据之间存在的关系，进而根据实际应用的要求，合理地组织和存储数据，设计出相应的算法，这就是“数据结构”要讨论的问题。本章介绍数据结构的基本概念和算法分析方法。

1.1 数 据 结 构

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，它是计算机程序加工的“原料”。例如，一个班的全部学生记录、 $a \sim z$ 的字母集合、 $1 \sim 1000$ 之间的所有素数等都称为数据。

数据元素（也称为结点）是数据的基本单位，在程序中通常作为一个整体进行考虑和处理。有时一个数据元素可以由若干个数据项组成。数据项是具有独立含义的最小标识单位。如在整数这个集合中，10 这个数就可以称为一个数据元素。又比如在一个数据库（关系数据库）中，一个记录可称为一个数据元素，而这个元素中的某一字段就是一个数据项。

数据结构是相互之间存在一种或多种特定关系的数据元素的集合。这些数据元素不是孤立存在的，而是有着某种关系，这种关系称为结构。

1.1.1 数据的逻辑结构

数据元素和数据元素之间的逻辑关系称为数据的逻辑结构。根据数据元素之间逻辑关系的不同特性，分为下列 4 类基本结构。

- 集合：结构中的数据元素同属于一个集合（集合类型由于元素之间的关系过于松散，数据结构课程中较少讨论）。
- 线性结构：结构中的数据元素存在一个对一个的关系。
- 树形结构：结构中的数据元素存在一个对多个的关系。
- 图状结构：结构中的数据元素存在多个对多个的关系。

大多数情况下,数据的逻辑结构可以用二元组来表示:

$$S=(D,R)$$

其中, D 是数据结点的有限集合; R 是 D 上的关系的有限集合, 其中每个关系都是从 D 到 D 的关系。在表示每个关系时, 用尖括号表示有向关系, 如 $\langle a,b \rangle$ 表示存在结点 a 到结点 b 之间的关系; 用圆括号表示无向关系, 如 (a,b) 表示既存在结点 a 到结点 b 之间的关系, 又存在结点 b 到结点 a 之间的关系。设 r 是一个 D 到 D 的关系, $r \in R$, 若 $d, d' \in K$, 且 $\langle d, d' \rangle \in r$, 则称 d' 是 d 的后继结点, d 是 d' 的前驱结点, 这时 d 和 d' 是相邻的结点 (都是相对 r 而言的); 如果不存在一个 d' , 使 $\langle d, d' \rangle \in r$, 则称 d 为 r 的终端结点; 如果不存在一个 d' , 使 $\langle d', d \rangle \in r$, 则称 d 为 r 的开始结点; 如果 d 既不是终端结点, 也不是开始结点, 则称 d 是内部结点。

例如, 一个城市表, 如表 1.1 所示, 就是一个数据结构, 它由很多记录 (这里的数据元素就是记录) 组成, 每个元素又包括多个字段 (数据项)。那么这个表的逻辑结构是怎样的呢? 分析数据结构都是从数据元素之间的关系开始分析, 对于这个表中的任意一个记录, 它只有一个前驱结点和一个后继结点, 整个表只有一个开始结点和一个终端结点。当知道这些关系之后, 就能明白这个表的逻辑结构, 即为线性结构。其逻辑结构表示如下:

City=(D,R)

D={北京, 上海, 武汉, 西安, 南京}

R={r}

r={⟨北京, 上海⟩, ⟨上海, 武汉⟩, ⟨武汉, 西安⟩, ⟨西安, 南京⟩}

表 1.1 城市表

城市	区号	说明
北京	010	首都
上海	021	直辖市
武汉	027	湖北省省会
西安	029	陕西省省会
南京	025	江苏省省会

数据的逻辑结构可以用相应的关系图来表示, 称为逻辑结构图。

【例 1.1】 设数据逻辑结构如下:

B1=(D,R)

D={1, 2, 3, 4, 5, 6, 7, 8, 9}

R={r}

r={⟨1, 2⟩, ⟨1, 3⟩, ⟨3, 4⟩, ⟨3, 5⟩, ⟨4, 6⟩, ⟨4, 7⟩, ⟨5, 8⟩, ⟨7, 9⟩}

试画出对应的逻辑结构图, 并给出哪些是开始结点? 哪些是终端结点? 说明是何种数据结构?

解: B1 对应的逻辑结构如图 1.1 所示。其中, 1 是开始结点; 2、6、8 和 9 是终端结

点。它是一种树形结构。

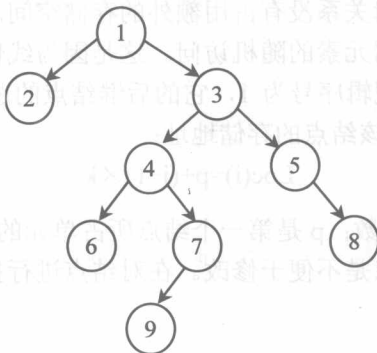


图 1.1 B1 的逻辑结构图

1.1.2 数据的存储结构

数据在计算机中的存储表示称为数据的存储结构，也称为物理结构。

把数据存储到计算机中时，通常要求既存储各结点的数值，又存储结点与结点之间的逻辑关系。在实际应用中，数据的存储方法是灵活多样的，可以根据问题规模（通常是指结点数目的多少）和运算种类等因素适当选择。本书将介绍 4 种基本的存储结构，它们是：顺序存储结构、链式存储结构、索引存储结构和散列存储结构。下面简要介绍这些存储结构的特点。

1. 顺序存储结构

顺序存储结构是把逻辑上相邻的元素存储在一组连续的存储单元中，其元素之间的逻辑关系由存储单元地址间的关系隐含表示。

对于前面的逻辑结构 City，假定每个元素占用 30 个存储单元，数据从 100 号存储单元开始由低地址向高地址方向存放，对应的顺序存储结构如图 1.2 所示。

地址	城市名	区号	说明
100	北京	010	首都
130	上海	021	直辖市
160	武汉	027	湖北省省会
190	西安	029	陕西省省会
220	南京	025	江苏省省会

图 1.2 City 的顺序存储结构

在顺序存储结构中，通常按照顺序从头到尾查找元素，例如，为了查找“武汉”的区号，先和第 1 个元素比较，再和第 2 个元素比较，直到和第 3 个元素比较时才成功，返回其区号 027。若给出要查找的序号 3，则可以直接找到该元素，不需要从头开始查找，这称为顺序存储结构的随机查找特性。

顺序存储结构的主要优点是节省存储空间。因为分配给数据的存储单元全用于存放结点的数据值, 结点之间的逻辑关系没有占用额外的存储空间。用这种方法存储线性结构的数据元素时, 可实现对各数据元素的随机访问。这是因为线性结构中每个数据元素都对应一个逻辑序号 (开始结点的逻辑序号为 1, 它的后继结点的逻辑序号为 2, ……), 可以根据结点的逻辑序号 i , 计算出该结点的存储地址:

$$\text{Loc}(i) = p + (i - 1) \times k$$

其中, k 是每个元素所占单元数; p 是第一个结点所占单元的首地址。

顺序存储结构的主要缺点是不便于修改。在对结点进行插入、删除运算时, 可能要移动一系列的结点。

2. 链式存储结构

链式存储结构是给每个结点附加指针字段, 用于存放相邻结点的存储地址。假定给实例 City 中的每个结点附加一个“下一个结点地址”, 即后继指针字段, 用于存放后继结点的首地址, 则可得到如图 1.3 所示的 City 的链式存储结构。图中, 每个结点占用两个连续的存储单元, 一个存放结点的数值, 另一个存放后继结点的首地址。

地址	城市名	区号	说明	下一个结点地址
100	北京	010	首都	220
130	南京	025	江苏省省会	^
160	西安	029	陕西省省会	130
190	武汉	027	湖北省省会	160
220	上海	021	直辖市	190

图 1.3 City 的链式存储结构

在链式存储结构中查找过程是沿着后继指针进行的。例如, 为了查找“武汉”的区号, 先查找地址为 100 的结点, 再沿后继指针查找地址为 220 的结点, 再沿后继指针查找地址为 190 的结点, 这时查找成功, 返回其区号 027。

链式存储结构的主要优点是便于修改, 在进行插入、删除运算时, 仅需修改结点的指针字段值, 不必移动结点。

与顺序存储结构相比, 链式存储结构的主要缺点是存储空间利用率较低, 因为分配给数据的存储单元有一部分用来存放结点之间的逻辑关系。另外, 由于逻辑上相邻的结点在存储器中不一定相邻, 因此, 在用这种方法存储的线性结构上不能对结点进行随机访问。

3. 索引存储结构

索引存储结构是在存储结点信息的同时, 还建立附加的索引表。索引表中的每一项称为索引项, 索引项的一般形式是: (关键字, 地址)。关键字唯一标识一个结点; 地址作为指向结点的指针。对于线性结构来说, 各结点的地址在索引表中是按结点的序号依次排列的。图 1.4 所示是 City 的一种索引存储结构。

地址	关键字	指针	地址	城市名	区号	说明
300	010	100	100	北京	010	首都
310	021	130	130	上海	021	直辖市
320	025	220	160	武汉	027	湖北省省会
330	027	160	190	西安	029	陕西省省会
340	029	190	220	南京	025	江苏省省会

(a) 索引表

(b) 城市表

图 1.4 City 的索引存储结构

在进行关键字查找时，可以先在索引表中快速查找（因为索引表中按关键字有序排列，可以采用二分查找法）到相应的关键字，然后通过地址找到结点表中对应的记录。例如，为了查找区号为 027 的城市，先在索引表中查找关键字为 027 的记录（由于索引表中按关键字递增排列，可以采用二分查找法快速查找），再通过地址 230 在城市表中直接找到对应的元素，返回其区号 027。

线性结构采用索引存储结构后，可以对结点进行随机访问。在进行插入、删除运算时，由于只需移动存储在索引表中结点的存储地址，而不必移动存储在结点表中结点的数值，所以仍可保持较高的运算效率（这是因为在一般情况下结点中总包含有多个字段，移动一个结点的数值要比移动一个结点的地址花费更多的时间）。

索引存储结构的缺点是，为建立索引表而增加了时间和空间的开销。

4. 散列存储结构

散列存储结构是根据结点的值确定结点的存储地址。具体做法是：以结点中某个字段的值为自变量，通过某个函数（称为散列函数）计算出对应的函数值 i ，再把 i 当做结点的存储地址。

对于 City 结构，假设以城市名的值作为自变量 key ，选用函数（称为散列函数）：

$$H(key) = \text{ASC}(\text{LEFT}(\text{Hz}(key), 1)) \bmod 8$$

来计算结点的存储地址。其中， $\text{Hz}(x)$ 表示提取汉字 x 的拼音串； $\text{ASC}(\text{LEFT}(\text{Hz}(key), 1))$ 表示取汉字串 key 中第一个拼音字符的 ASCII 码， $\bmod$ 是取模运算，计算结果如表 1.2 所示。

表 1.2 散列函数计算结果

key	北京	上海	武汉	西安	南京
$\text{ASC}(\text{LEFT}(\text{Hz}(key), 1))$	66	83	87	88	78
$H(key)$	2	3	7	0	6

于是，可以得到如图 1.5 所示的 City 的散列存储结构。

散列存储结构的优点是查找速度快，只要给出待查找结点的数值，就可以立即算出结点的存储地址。例如，为了查找“武汉”的区号，先计算 $H(\text{"武汉"})$ 的值为 7，然后在散列表中找到对应的元素，通过比较得知查找成功，返回其区号 027。

地址	城市名	区号	说明
0	西安	029	陕西省省会
1		000	
2	北京	010	首都
3	上海	021	直辖市
4		100	
5		520	
6	南京	025	江苏省省会
7	武汉	027	湖北省省会

图 1.5 City 的散列存储结构

与前 3 种存储方法不同的是,散列存储结构只存储结点的数据值,不需要存储结点与结点之间的逻辑关系。散列存储结构一般只用于要求对数据能够进行快速查找、插入的场合。采用散列存储结构的关键是要选择一个好的散列函数和处理“冲突”的办法。

在用高级语言编程时,可以用编程语言提供的数据类型来描述数据的存储结构。例如,用“一维数组”表示一组连续的存储单元来实现顺序存储结构、索引存储结构和散列存储结构;用 C/C++ 语言中的“指针”来实现链式存储结构。

1.1.3 数据的运算

数据运算就是施加于数据的操作。例如,有一张表格,需要对其进行查找、添加、修改和删除记录等操作。在数据结构中,运算不仅仅是加、减、乘、除等算术运算,还常常涉及算法问题。算法的实现与数据的存储结构是密切相关的。

1.1.4 数据结构和数据类型

将按某种逻辑关系组织起来的一组数据元素,按一定的存储方法存储在计算机中,并在其上定义了一个运算的集合,称为一个数据结构。

而数据类型是高级语言中的一个基本概念,它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。数据类型明确或隐含地规定了数据的取值范围、存储方法及允许进行的运算。因此可以认为,数据类型是程序设计语言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,必须借助程序设计语言所提供的数据类型来描述数据的存储结构。

下面总结 C/C++ 语言中常用的数据类型。

(1) C/C++ 语言的基本数据类型

C/C++ 语言中的基本数据类型有 int 型、float 型和 char 型。int 型可以有 3 个修饰符: short (短整数)、long (长整数) 和 unsigned (无符号整数)。