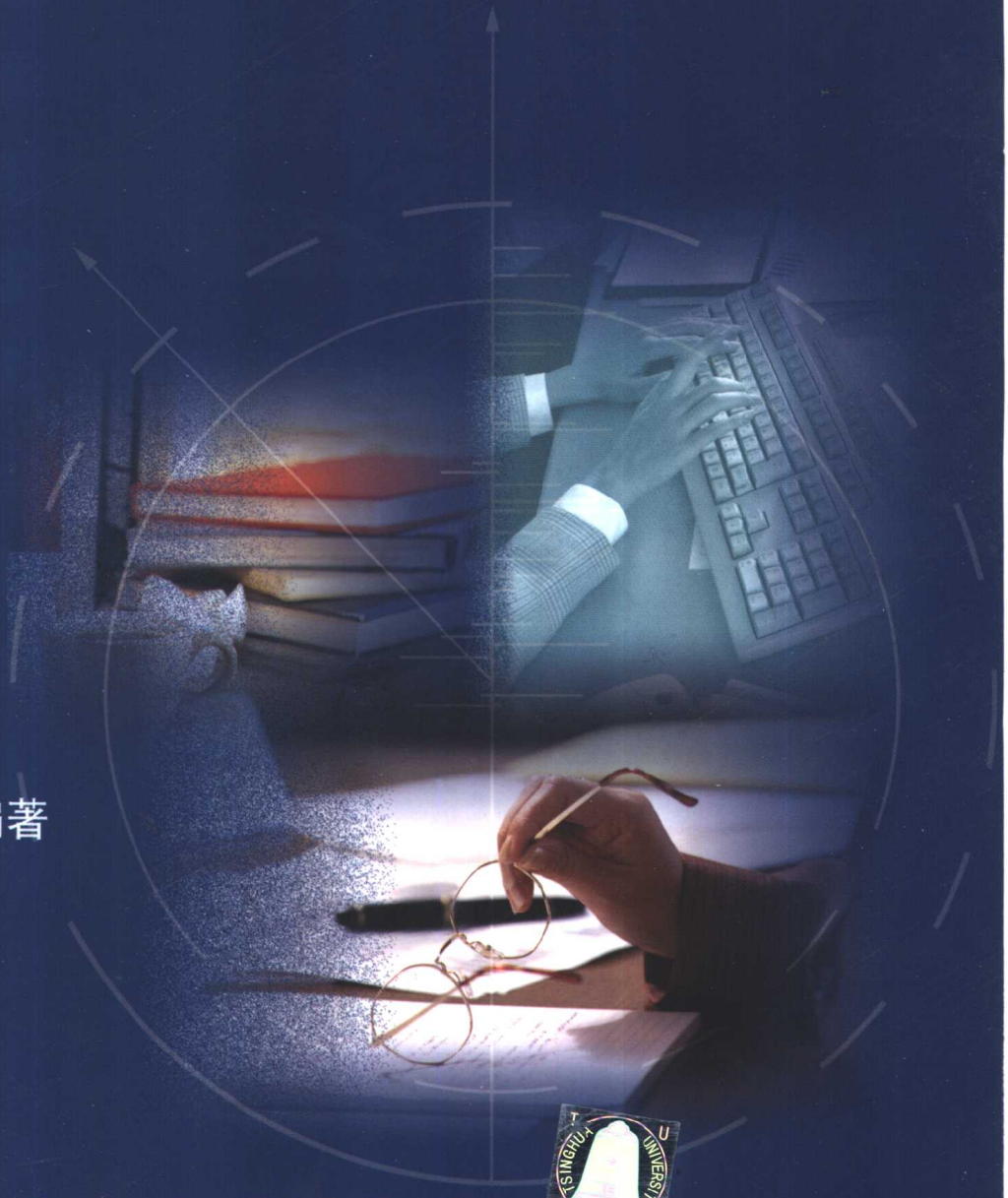
编程之路系列教材

数据结构导学



苏光奎　李春葆　编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

编程之路系列教材

数据结构导学

苏光奎　李春葆　编著

清华大学出版社

（京）新登录字158号

内 容 提 要

本书汇集作者多年讲授“数据结构”的教学经验，结合计算机专业的相关教学大纲编写而成。全书共分8章，详细介绍数据结构的基本概念、基本算法以及线性表、栈和队列、串和数组、树和二叉树、图、排序、查找等内容。

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运算时，不仅介绍算法思想，更注重程序的实现过程，并提供了各种数据结构运算算法的源程序，有助于读者深刻领会数据结构的内涵。本书还在最后提供了6个实习题。

本书面向欲通过“数据结构”来提升自己程序设计能力的读者，也可作为计算机专业的成人教育、自学考试和各类培训班的教材。

书　名：数据结构导学
作　者：苏光奎　李春葆
出版者：清华大学出版社（北京清华大学校内，邮编100084）
印刷者：北京朝阳科普印刷厂
发行者：新华书店总店北京科技发行所
开　本：787×1092 1/16　印张：16.375　字数：398千字
版　次：2002年2月第1版　　2002年2月第1次印刷
印　数：0001～8000
书　号：ISBN 7-302-05246-8/TP・3084
定　价：22.00元

《编程之路系列教材》序

随着计算机技术在我国各个领域的广泛应用，以及计算机软件平台的不断提升，计算机编程不再仅限于计算机专业人员，越来越多的计算机爱好者通过专项培训或自学，已成为计算机编程的“行家里手”；特别是我国已经入世，国内IT企业及非IT企业对人才的需求将超过40万，其中一半是软件技术人才，这是传统学历教育远远满足不了的，它需要通过各种途径为这一行业的发展提供大批的IT技术人材。本丛书就是为此目的而编写的，它以计算机编程为核心，涵盖了从基础到专业应用的一些重要课目。本套丛书包括：

1. 《C++语言程序设计导学》
2. 《数据结构导学》
3. 《Visual Basic 6程序设计导学》
4. 《PowerBuilder 7.0程序设计导学》
5. 《Visual C++ 程序设计导学》
6. 《Delphi程序设计导学》
7. 《C++ Builder程序设计导学》

本丛书具有以下特点：

○ 力求通俗易懂

本丛书不仅面向计算机专业人员，更立足于计算机编程爱好者，因此，在文字叙述和内容的安排上尽量通俗易懂，力求讲出问题的来龙去脉，把编程的“过程”讲透。

○ 强化编程的概念

作为一个编程人员，必须深入领会编程的实质，这样才能做到举一反三，融会贯通，达到编制自己的应用程序的目的。所以本丛书不同于一般的软件系统使用手册，而是针对读者学习中可能遇到的问题诠释了编程思路和编程技巧，便于读者提升编程能力。

○ 编程思想与开发工具运用相结合

学习编程，不仅要在学习编程思想上有所突破，还应学会如何更好地运用编程的开发工具，只有两者的结合才是真正的理论联系实际，事半功倍的学习方法。本丛书精选了目前流行的软件开发工具（如Visual Basic，PowerBuilder，Visual C++，Delphi，C++ Builder），这些工具中提供了许多编程技巧和功能，对编程者具有实际的应用价值。

○ 内容表述与习题、实习训练并重

本丛书提供了大量的习题和实习题，而且给出了这些习题和实习题的参考答案，便于读者练习、仿效，达到快速掌握编程方法和技巧。

由于时间仓促，本书疏漏之处在所难免。但我们相信本丛书一定会成为计算机编程爱好者的良师益友。

前　言

“数据结构”是计算机专业的一门专业基础课程，也是一门核心课程，起到承前启后的作用。本书是结合计算机专业的相关教学大纲编写的，介绍了各种最常用的数据结构，阐明各种数据结构的内在逻辑关系，讨论它们在计算机中的存储结构，以及这些数据结构上的操作和实际算法。

本书力求在阐明各种数据结构的内在逻辑关系、存储结构和相应运算的同时，从编程角度出发，强化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，更重要的是得到了程序设计的训练和编程能力的提高。

全书共分为 8 章：第 1 章是绪论部分，讨论数据结构的基本概念和算法描述；第 2 章介绍线性表，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 章为栈和队列，讨论栈和队列的特点及其各种存储结构与基本操作的实现，并给出了相应的应用实例；第 4 章介绍串和数组，讨论串的各种存储结构及其基本操作实现，数组和稀疏矩阵各种存储结构及其基本操作实现；第 5 章阐述树和二叉树，讨论树的定义与表示、二叉树的基本操作和哈夫曼树；第 6 章介绍的是图，讨论图的各种存储结构和遍历的实现；第 7 章介绍排序，讨论了各种常用的排序方法及其实现；第 8 章是查找，讨论了各种常用的查找方法及其实现。在基本内容之后是实习题部分，给出了 6 个实习题，用于进一步加深对数据结构内容的理解。最后列出了三个附录，附录 A 给出了各章习题的参考答案，附录 B 介绍了数据结构中经常用到的递归设计方法，附录 C 给出了本书程序索引。

本书有以下两个特点：

- **实用性**　掌握和提高计算机编程能力是学习计算机的重要目的之一，所以本书在介绍数据结构的基本运算时，不是仅仅局限于算法思想，而是着眼于程序的实现过程，书中给出了各种数据结构运算算法的源程序（这些源程序均已上机调试通过），便于读者更深刻地领会数据结构的内涵。
- **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。特别是对书中的所有习题和实习题都给出了参考答案，便于学生自学。

本书附录 C 列出的源程序可从 http://www.khp.com.cn 网址下载。

尽管编者长期从事计算机教学工作，但由于水平有限，书中难免存在不足和错误之处，敬请读者批评指正。

读者联系方式：licb@public.wh.hb.cn

编　者

2001 年 11 月

目 录

第1章 绪 论

“数据结构”是计算机专业的专业基础课之一，是一门十分重要的核心课程。计算机的所有系统软件和应用软件都要用到各种类型的数据结构。要想编写出“好”的程序，仅仅学习计算机语言是不够的，必须扎实地掌握数据结构的基本知识和基本技能。要想学好计算机专业的其他课程，如操作系统、数据库原理与应用、软件工程等，也应具备数据结构的基础知识。

随着计算机应用领域的不断扩大，非数值计算问题已跃居主导地位，简单的数据类型已远远不能满足需要，各数据元素之间的复杂联系不再是普通数学方程式所能表达的了。因此，掌握好数据结构方面的知识，对于增强我们解决实际问题的能力将会有很大帮助。事实上，一个“好”的程序无非是选择了一个合理的数据结构和一个好的算法，而好的算法的选择在很大程度上取决于描述实际问题的数据结构。所以，学好数据结构课程，是进一步提高程序设计水平的关键之一。

1.1 什么是数据结构

在了解数据结构的重要性之后，我们开始讨论数据结构的定义。本节先从一个简单的学生例子入手，继而给出数据结构的严格定义，然后分析数据结构的几种类型，最后给出数据结构和数据类型之间的区别与联系。

1.1.1 数据结构的定义

数据是人们利用文字符号、数字符号以及其他规定的符号对现实世界的事物及其活动所做的抽象描述。例如，文字、数字和符号都是数据。

数据元素是数据的基本单位（例如学生是一个数据元素）。有时候，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数据最小单位，称为字段或域（例如作为数据元素的学生由学号、姓名、性别和班号等数据项组成）。

数据结构是指数据以及相互之间的联系，可以看作是相互间存在特定关系的数据元素的集合。这些特定关系包括：

（1）数据元素之间的逻辑关系，即数据的逻辑结构。

（2）数据元素及其关系在计算机存储器中的存储方式，即数据的存储结构，也就是数据元素的物理结构。

（3）施加在数据上的操作，即数据的运算。

【例 1.1】 有一个学生表，如图 1.1 所示。这个表中的数据元素是学生记录，每个数据元素由四个数据项（即学号、姓名、性别和班号）组成。

学号	姓名	性别	班号
1	张斌	男	9901
8	刘丽	女	9902
34	李英	女	9901
20	陈华	男	9902
12	王奇	男	9901
26	董强	男	9902
5	王萍	女	9901

图 1.1 学生表

该表中的记录顺序反映了数据元素之间的逻辑关系。这些数据在计算机存储器中的存储方式就是存储结构。假设我们采用 C 语言中的如下结构体类型进行存储：

```
struct student
{
  int no;                    /*对应学号*/
  char name[8];              /*对应姓名*/
  char sex[2];               /*对应性别*/
  char class[4];             /*对应班号*/
  struct student *next;      /*指向下一个学生的指针*/
}
```

则其存储结构如图 1.2 所示，首指针为 head。对于该数据结构中的数据施加的操作有很多，例如增加一个学生数据、删除一个学生数据、查找性别为“女”的学生数据、查找班号为“9902”的学生数据，等等。

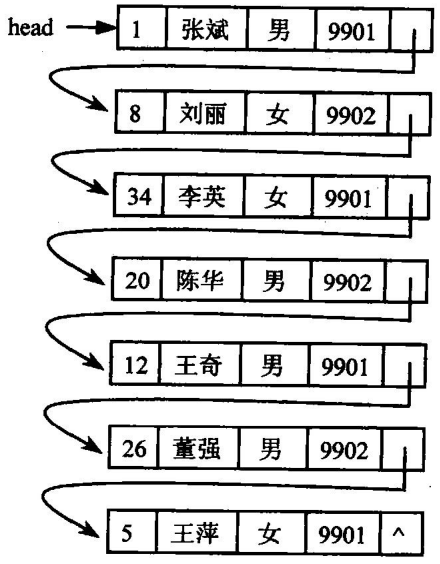


图 1.2 学生表的存储结构

从上例可以看出，数据的逻辑结构和存储结构都反映了数据的结构关系。但通常所说的数据结构是指数据的逻辑结构，不涉及存储结构的含义。

为了更确切地描述一种数据结构，通常采用二元组表示：

B=（K，R）

B 是一种数据结构，它由数据元素的集合 K 和 K 上二元关系的集合 R 所组成。

$K=\{k_i \mid 1\leqslant i\leqslant n，n\geqslant 0\}$

$R=\{r_j \mid 1\leqslant j\leqslant m，m\geqslant 0\}$

其中：k_i 表示集合 K 中的第 i 个数据元素，n 为 K 中数据元素的个数，特别是，若 n=0，则 K 是一个空集，因而 B 也就无结构可言，有时也可以认为它具有任一结构；r_j 表示集合 R 中的第 j 个二元关系（后面均简称关系），m 为 R 中关系的个数，特别是，若 m=0，则 R 是一个空集，表明集合 K 中的元素之间不存在任何关系，彼此是独立的。

K 上的一个关系 r 是序偶的集合，对于 r 中的任一序偶<x，y>（x，y∈K），我们把 x 叫做序偶的第一元素，把 y 叫做序偶的第二元素，又称序偶的第一元素为第二元素的直接前驱（通常简称前驱），称第二元素为第一元素的直接后继（通常简称后继）。如在<x，y>的序偶中，x 为 y 的前驱，而 y 为 x 的后继。

若某个结点没有前驱，则称该结点为开始结点；若某个结点没有后继，则称该结点为终端结点。

对于对称序偶，即<x，y>∈R，<y，x>∈R （x，y∈K），则用圆括号代替尖括号，即（x，y）∈R。

【例 1.2】 采用二元组表示例 1.1 的学生表。

学生表中共有 7 个数据元素，依次用 k_1~k_7 表示，则对应的二元组表示为 B=（K，R），其中

$K=\{k_1，k_2，k_3，k_4，k_5，k_6，k_7\}$

R={r}

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

数据结构还能够利用图形形象地表示出来，图形中的每个结点对应着一个数据元素，两结点之间的连线对应着关系中的一个序偶。

1.1.2 数据结构类型

数据结构分为线性结构和非线性结构两大类。

在线性结构中，开始结点和终端结点都是惟一的，除了开始结点和终端结点外，其余结点都有且仅有一个前驱，有且仅有一个后继，顺序表就是典型的线性结构。

非线性结构又可以细分为树形结构和图形结构两类。

在树形结构中，每个结点最多只有一个前驱，但可以有多个后继，可以有多个终端结点。非线性结构的树形结构简称为树。

在图形结构中，每个结点的前驱和后继的个数都可以是任意的。因此，可能没有开始结点和终端结点，也可能有多个开始结点、多个终端结点。图形结构简称为图。

由图形结构、树形结构和线性结构的定义可知，线性结构是树形结构的特殊情况，而树形结构又是图形结构的特殊情况。

【例 1.3】 有一种数据结构 B1=（K，R），其中

K={1，5，8，12，20，26，34}

R={r}

r={<1，8>，<8，34>，<34，20>，<20，12>，<12，26>，<26，5>}

则对应的图形如图 1.3 所示。

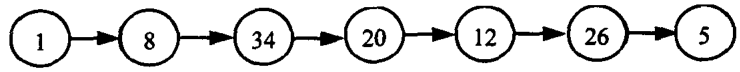


图 1.3 线性结构示意图

从该例可以看出，每个数据元素有且仅有一个直接前驱元素（除结构中第一个元素 1 外），有且仅有一个直接后继元素（除结构中最后一个元素 5 外）。这种数据结构的特点是：数据元素之间为一对一的联系，即线性关系。这种数据结构就是线性结构。

【例 1.4】 有一种数据结构 B2=（K，R），其中

K={a，b，c，d，e，f，g，h，i，j}

R={r}

r={<a，b>，<a，c>，<a，d>，<b，e>，<c，f>，<c，g>，<d，h>，<d，i>，<d，j>}

则对应的图形如图 1.4 所示。

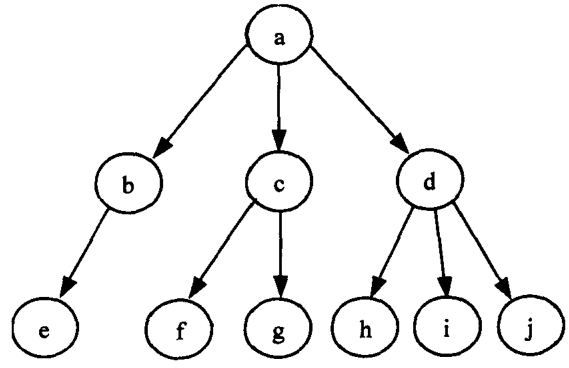


图 1.4 树形结构示意图

从该例可以看出，每个结点有且只有一个前驱结点（除树根结点外），但有任意多个后继结点（树叶结点可看做具有 0 个后继结点）。这种数据结构的特点是：数据元素之间为一

对多关系，即层次关系。这种数据结构就是树形结构。

【例 1.5】 有一种数据结构 B3=（K，R），其中

K={a，b，c，d，e}
R={r}
r={（a，b），（a，c），（b，c），（c，d），（c，e），（d，e）}

则对应的图形如图 1.5 所示。

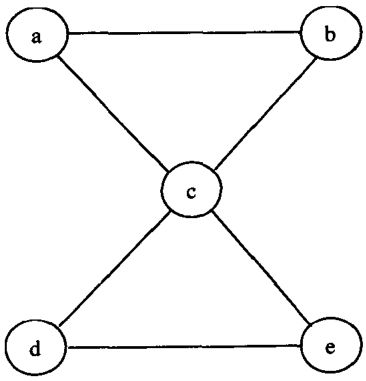


图 1.5　图形结构示意图

从该例看出，每个结点可以有多个前驱结点和多个后继结点。这种数据结构的特点是：数据元素之间为多对多关系，即图形关系。这种数据结构就是图形结构，简称图。

1.1.3　数据结构和数据类型

数据类型是与数据结构密切相关的一个概念，容易引起混淆。

数据类型最早出现在高级语言中，是对数据的取值范围、每一数据的结构以及允许施加的操作的一种描述，它刻画了程序中操作对象的特性。每一种程序语言都定义有自己的数据类型，在用程序语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。

数据类型可分为简单类型和结构类型两种。简单类型中的每个数据（即简单数据）都是无法再分割的整体，如一个整数、实数、字符、指针、枚举量等都是无法再分割的整体，所以它们所属的类型均为简单类型。结构类型由简单类型按照一定的规则构造而成，并且结构类型中可以包含结构类型，所以一种结构类型中的数据（即结构数据）可以分解为若干个简单数据或结构数据，每个结构数据仍可再分。例如 C 语言中的数组是一种结构类型，它是由固定个数的同一类型的数据顺序排列而成；结构体也是一种结构类型，它是由固定个数的不同类型的数据顺序排列而成。

数据类型也可以被定义为一种数据结构和能够对该数据结构进行的操作的集合。对于简单类型，其数据结构就是相应取值范围内的所有数据，每一个数据值是不可分的独立整体，因而数据值内部就无结构可言。对于结构类型，其数据结构就是相应元素的集合（它

实际上是该结构类型中的一个值）和元素之间所含关系的集合。

总之，数据结构是指计算机处理的数据元素的组织形式和相互关系，而数据类型是某种程序设计语言中已实现的数据结构。在程序设计语言提供的数据类型支持下，我们就可以根据从问题中抽象出来的各种数据模型，逐步构造出描述这些数据模型的各种新的数据结构。

下面总结 C 语言中常用的数据类型。

（1）C 语言的基本数据类型

C 语言中的基本数据类型有 int 型、float 型和 char 型。int 型可以有三个限定词：short，long 和 unsigned。float 型有三种形式：float，double 和 long double。

（2）C 语言的指针类型

C 语言允许直接对存放变量的地址进行操作。如定义 int i，则&i 表示变量 i 的地址，也称作指向变量 i 的指针。存放地址的变量称作指针变量。

有关指针的两个操作是：对于定义 int i，则&i 操作是取变量 i 的地址；对于定义 int *p，这里的 p 是指向一个整数的指针，则*p 操作是取 p 指针所指的值，即 p 所指地址的内容。

（3）C 语言的数组类型

数组是同一类型的一组有序数据的集合。数组有一维数组和多维数组。数组名标识一个数组，下标指示一个数组元素在该数组中的顺序位置。

数组下标的最小值称为下界，在 C 语言中总是为 0。数组下标的最大值称为上界，在 C 语言中数组上界为数组定义值减 1。例如，int a[10]，定义了包含 10 个整数的数组 a。

（4）C 语言中的结构体类型

结构体由一组称为结构体成员的项组成，每个结构体成员都有自己的标识符。例如：

```
struct teacher
   {
    int no;
    char name[8];
    int age;
   }
```

定义了一个结构体类型 teacher。下面的语句定义了该类型的两个变量 t1 和 t2：

```
struct teacher t1, t2;
```

（5）C 语言中的共用体类型

共用体是把不同的成员组织为一个整体，它们在存储器中共享一段存储单元，但以不同的方式解释不同的成员。例如：