普通高校计算机主干课程辅导与提高丛书

徐孝凯　编著

数据结构辅导与提高

清华大学出版社

普通高校计算机主干课程辅导与提高丛书

数据结构辅导与提高

徐孝凯　编著

清华大学出版社

北　京

内 容 简 介

本书从数据结构的学科内容出发，针对数据结构教材中的重点和难点，分 15 个专题进行深入细致的讲解和讨论。对于要解决的每个问题，从问题提出，到思路分析，再到具体数据结构的选择、算法描述和上机实现，循序渐进地给出了完整过程，非常符合学生的自学需要和获取知识的过程。书中所给的每个算法都在 C 或 C++语言环境下运行通过，并且都是经过认真比较、筛选和设计出来的，具有一定的代表性，具有较好的实用性、有效性、结构性和可读性。通过仔细分析和阅读，对于提高软件开发和程序设计水平将大有裨益。书中给出的 15 个专题基本上是相互独立的，但有些是前后关联的，它们都具有统一的风格。读者可根据学习的需要在任何时刻任选某个或某些专题参考。

本书适合高等院校计算机及相关专业学生，以及参加计算机研究生入学考试的考生使用。

图书在版编目（CIP）数据

数据结构辅导与提高/徐孝凯编著. —北京：清华大学出版社，2003

（普通高校计算机主干课程辅导与提高丛书）

ISBN 7-302-07584-0

Ⅰ. 数… Ⅱ. 徐… Ⅲ. 数据结构—高等学校—教学参考资料 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字（2003）第 103778 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社 总 机：010-62770175　　**客户服务**：010-62776969

责任编辑：刘利民

封面设计：钱　诚

版式设计：杨　洋

印 刷 者：北京世界知识印刷厂

装 订 者：北京鑫海金澳胶印有限公司

发 行 者：新华书店总店北京发行所

开　　本：185×260　**印张**：24.25　**字数**：534 千字

版　　次：2003 年 12 月第 1 版　2003 年 12 月第 1 次印刷

书　　号：ISBN 7-302-07584-0/TP・5581

印　　数：1～5000

定　　价：28.00 元

“普通高校计算机主干课程辅导与提高”丛书

组织与编著委员会

丛 书 序

随着我国国民经济持续、稳定、快速地发展，我国高等教育正在经历一个前所未有的发展阶段，招生规模不断扩大。在世界范围内，信息技术得到了空前发展和广泛应用，社会迫切需要培养大量的计算机和信息技术方面的人才。计算机和信息类专业已经成为我国当前高等教育中社会最需要、发展最迅速、招生规模最大的热门专业之一。

由于计算机和信息类专业招生规模扩大，高校师资力量显得不足，讲授课程只能由小班改为大班或合班上课，对每个学生的批改作业时间和辅导答疑时间也相应地减少，但学生所使用的教材没有适应这种变化，以前学习计算机专业教材就比较难，现在相对难度更大了。这就要求配合使用针对性强的、便于自学的辅导教材，来帮助解决在学习过程中遇到的各类问题，提高学习效率。现在社会上计算机类的教材比较多，但用于学生自主学习的辅助教材却很少，各地师生要求尽快组织出版此类教材的呼声越来越高。清华大学出版社根据学生的迫切需要，本着对学生高度负责的精神，组织一批具有相关课程教学经验和编写教材经验并深受读者好评的高校专家编写了这套辅助教材。

这套书将分批陆续推出，首批推出 6 本。它们是《计算机组成原理辅导与提高》、《数据结构辅导与提高》、《操作系统辅导与提高》、《计算机网络辅导与提高》、《数据库原理辅导与提高》、《软件工程辅导与提高》。

这套书的编委会从有利于学生自主学习的需要出发，经过多次酝酿和研讨，最后形成统一认识，采用统一的编写格式，主要特点如下：

- 从每门课程的学科内容出发，针对一般教材中的重点和难点，分专题进行循序渐进地解剖和分析，并尽量通过恰如其分的事例进行说明，使读者既能够理解和掌握重要的概念，又能够从理论和实际应用的结合上加深认识。
- 为了加强训练和应用，巩固所学的知识，在每本教材的附录中分别给出了针对课程内容的练习题和参考解答。练习题型丰富，包含选择、填空、运算、分析、设计等题型，对于较难的习题在解答中同时给出了分析过程。
- 为了自测本人的学习效果，在附录中给出了一套试卷，要求 2 小时内做完。
- 本套丛书的读者对象为学习相应课程的本、专科学生以及考研的学生，他们可以根据各自需要选取有关内容。

欢迎广大读者对本丛书提出宝贵意见，我的电子邮件地址为 xuxk@crtvu.edu.cn。

丛书主编　徐孝凯

2003 年 12 月

前　言

数据结构是计算机等相关专业开设的一门基础必修课。当今社会人们面对着各种各样的数据和信息，这些数据和信息只有用计算机处理，才能够做到快速、及时和准确。数据结构课程正是研究如何逻辑地组织数据，如何把数据有效地存储到计算机中，如何对数据进行快速和可靠的加工及处理，从而获取有用的信息。因此数据结构是数据处理的基础，是所有计算机课程中的核心课程。由于它涉及到高等数学、离散数学、概率统计、计算机原理、程序设计等各方面知识，又由于它本身的理论性较强，所以被公认为是一门比较难学的课程。但只要努力学好它，就能为学习后续的操作系统、数据库、软件工程等课程以及实际的软件开发打下坚实的基础。

现在社会上数据结构教材很多，但缺乏适合广大学生自学参考的辅导教材。本书正是根据广大学生的自学要求，为了降低学习数据结构课程的难度而精心设计和编写的一本辅导教材。

本书从数据结构的学科内容出发，针对一般数据结构教材中的重点和难点，分 15 个专题（或称知识点）进行深入细致的讲解和讨论。对于要解决的每个问题，从问题提出，到思路分析，再到具体数据结构的选择、算法描述和上机实现，循序渐进地给出了完整过程，非常符合在校学生的自学需要。书中所给的每个算法都在 C 语言或 C++语言环境下运行通过，并且都是经过认真比较、筛选和设计出来的，具有一定的代表性，具有较好的实用性、有效性、结构性和可读性，通过仔细分析和阅读，对于提高软件开发和程序设计水平将大有裨益。书中给出的 15 个专题基本上是相互独立的，但有些是前后关联的，它们都具有统一的风格。读者可根据学习的需要在任何时刻任选某个或某些专题参考。当进入该课程的复习或考研阶段，最好能够系统地浏览全部 15 个专题的内容，以便提高自己的知识水平，达到既定目标。

本书的附录 A 为综合练习题，共分为 7 个练习单元，每个单元大致包括单项选择、填空、运算、算法分析、算法设计等题型。练习题型丰富，内容充实。通过做题训练，举一反三，能够巩固所学知识，加深对内容的理解和认识。附录 B 给出了附录 A 中全部习题的参考解答，供读者独立做题后参考。附录 C 是一份自测试卷，要求在 2 个小时内做完，以便检查自己的学习成绩。

本书内容安排由浅入深，叙述条理清楚，重难点分析透彻，应用举例生动实用，算法描述规范易读，特别适合于自学。本书是广大本、专科学生学习数据结构课程所使用的辅导书，亦可作为讲授此课程教师用作参考书。

为了给广大读者学习数据结构课程提供一本好的参考书，作者尽了最大的努力，但由于水平有限，难免会出现一些缺陷，敬请给予批评指正。

徐孝凯

2003 年 12 月

目　录

专题 1　数据结构分类与抽象数据类型

1.1　数据结构分类

数据结构讨论现实世界和计算机世界中的数据及其相互之间的联系，这体现在逻辑和存储两个层面上，相应称之为逻辑结构和存储结构。也就是说，在现实世界中讨论的数据结构是指逻辑结构，在计算机世界中讨论的数据结构是指存储结构，又称为物理结构。

数据的逻辑结构总体上分为 4 种类型：集合结构、线性结构、树结构和图结构。数据的存储结构总体上也分为 4 种类型：顺序结构、链接结构、索引结构和散列结构。原则上，一种逻辑结构可以采用任一种存储结构来存储（表示）。

对于现实世界中的同一种数据，根据研究问题的角度不同，将会选用不同的逻辑结构；对于一种逻辑结构，根据处理问题的要求不同，将会选用不同的存储结构。

对于复杂的数据结构，不论从逻辑层面上还是从存储层面上看，都可能包含有多个嵌套层次。如假定一种数据结构包含有两个层次，第一层（顶层）的逻辑结构可能是树结构，存储结构可能是链接结构；第二层（底层）的逻辑结构可能是线性结构，存储结构可能是顺序结构。第一层结构就是数据的总体结构，第二层结构就是第一层中数据元素的结构。

数据的逻辑结构通常采用二元组来描述，其中一元为数据元素的集合，另一元为元素之间逻辑关系的集合，每一个逻辑关系是元素序偶的集合，如<x,y>就是一个序偶，其中 x 为前驱，y 为后继。当数据的逻辑结构存在着多个逻辑关系时，通常对每个关系分别进行讨论。

逻辑结构的另一种描述方法是图形表示，图中每个结点表示元素，每条带箭头的连线表示元素之间的前驱与后继的关系，其箭头一端为后继元素，另一端为前驱元素。

数据的存储结构通常采用一种计算机语言中的数据类型来描述，通过建立数据存储结构的算法来具体实现。

数据的逻辑结构或存储结构也时常被简称为数据结构，读者可根据上下文来理解。

下面通过例子来说明数据的逻辑结构。

假定某校教务处的职员简表如表 1.1 所示。该表中共有 10 条记录，每条记录都由 6 个数据项组成。此表整体上被看为一个数据，每个记录是这个数据中的数据元素。由于每条记录的职工号各不相同，所以可把职工号作为记录的关键字，在下面构成的各种数据结构中，将用记录的关键字代表整个记录。

表 1.1 教务处职员简表

职 工 号	姓 名	性 别	出 生 日 期	职 务	部 门
01	万明华	男	1962.03.20	处长	
02	赵 宁	男	1968.06.14	科长	教材科
03	张 利	女	1964.12.07	科长	考务科
04	赵书芳	女	1972.08.05	主任	办公室
05	刘永年	男	1959.08.15	科员	教材科
06	王明理	女	1975.04.01	科员	教材科
07	王 敏	女	1972.06.28	科员	考务科
08	张 才	男	1967.03.17	科员	考务科
09	马立仁	男	1975.10.12	科员	考务科
10	邢怀常	男	1976.07.05	科员	办公室

【例 1.1】一种数据结构的二元组表示为 set=(K,R),其中

```
K={01,02,03,04,05,06,07,08,09,10}
R={}
```

在数据结构 set 中，只存在有元素的集合，不存在有关系，或者说关系为空。这表明只考虑表中的每条记录，不考虑它们之间的任何关系。把具有此种特点的数据结构称为集合结构。

集合结构中的元素可以任意排列，无任何次序。

【例 1.2】一种数据结构的二元组表示为 linearity=(K,R),其中

```
K={01,02,03,04,05,06,07,08,09,10}
R={<05,01>,<01,03>,<03,08>,<08,02>,<02,07>,<07,04>,
   <04,06>,<06,09>,<09,10>}
```

对应的图形表示如图 1.1 所示。

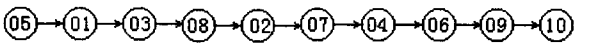

图 1.1 数据的线性结构示意图

结合表 1.1，细心的读者不难看出：R 是按职员年龄从大到小排列的关系。

在数据结构 linearity 中，数据元素之间是有序的，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 05 外），有且仅有一个直接后继元素（除结构中最后一个元素 10 外）。这种数据结构的特点是数据元素之间的 1 对 1（1∶1）联系，即线性关系。我们把具有这种特点的数据结构叫做线性结构。

【例 1.3】一种数据结构的二元组表示为 tree=(K,R),其中

```
K={01,02,03,04,05,06,07,08,09,10}
R={<01,02>,<01,03>,<01,04>,<02,05>,<02,06>,<03,07>,
   <03,08>,<03,09>,<04,10>}
```

对应的图形表示如图 1.2 所示。

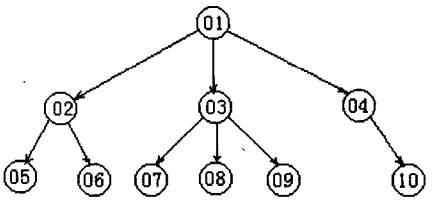

图 1.2　数据的树结构示意图

结合表 1.1，细心的读者不难看出：R 是职员之间领导与被领导的关系。

图 1.2 像倒着画的一棵树，在这棵树中，最上面的一个没有前驱只有后继的结点叫做树根结点，最下面一层的只有前驱没有后继的结点叫做树叶结点，除树根和树叶之外的结点叫做树枝结点。

在一棵树中，每个结点有且只有一个前驱结点（除树根结点外），但可以有任意多个后继结点（树叶结点可看作为含 0 个后继结点）。这种数据结构的特点是数据元素之间的 1 对 N（1∶N）联系（N≥0），即层次关系，我们把具有这种特点的数据结构叫做树结构，简称树。

【例 1.4】一种数据结构的二元组表示为 graph=(K,R)，其中

```
K={01,02,03,04,05,06,07}
R={<01,02>,<02,01>,<01,04>,<04,01>,<02,03>,<03,02>,
   <02,06>,<06,02>,<02,07>,<07,02>,<03,07>,<07,03>,
   <04,06>,<06,04>,<05,07>,<07,05>}
```

对应的图形表示如图 1.3 所示。

从图 1.3 可以看出，R 是 K 上的对称关系。为了简化起见，我们把〈x,y〉和〈y,x〉这两个对称序偶用一个无序对（x,y）或（y,x）来代替；在示意图中，我们把 x 结点和 y 结点之间两条相反的有向边用一条无向边来代替。这样 R 关系可改写为：

```
R={(01,02),(01,04),(02,03),(02,06),(02,07),
   (03,07),(04,06),(05,07)}
```

对应的图形表示如图 1.4 所示。

如果说 R 中每个序偶里的两个元素所代表的职员是好友的话，那么 R 关系就是人员之

间的好友关系。

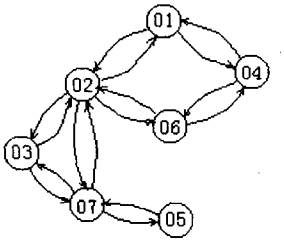

图 1.3 数据的图结构示意图

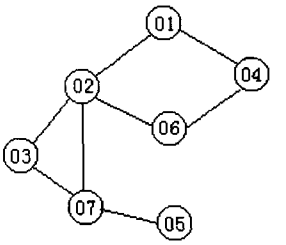

图 1.4 图 1.3 的等价表示

从图 1.3 或 1.4 可以看出，结点之间的联系是 M 对 N（M∶N）联系（M≥0，N≥0），即网状关系。也就是说，每个结点可以有任意多个前驱结点和任意多个后继结点。我们把具有这种特点的数据结构叫做图结构，简称图。

从图结构、树结构和线性结构的定义可知，树结构是图结构的特殊情况（即 M=1 的情况），线性结构是树结构的特殊情况（即 N=1 的情况）。为了区别于线性结构，我们把树结构和图结构统称为非线性结构。

集合结构是整个数据结构中的一种特殊情况，其元素之间不存在任何关系。

【例 1.5】一种数据结构的二元组表示为 B=(K,R),其中

$$K=\{k_1,k_2,k_3,k_4,k_5,k_6\}$$

$$R=\{R1,R2\}$$

$$R1=\{<k_3,k_2>,<k_3,k_5>,<k_2,k_1>,<k_5,k_4>,<k_5,k_6>\}$$

$$R2=\{<k_1,k_2>,<k_2,k_3>,<k_3,k_4>,<k_4,k_5>,<k_5,k_6>\}$$

若用实线表示关系 R1，虚线表示关系 R2，则对应的图形表示如图 1.5 所示。

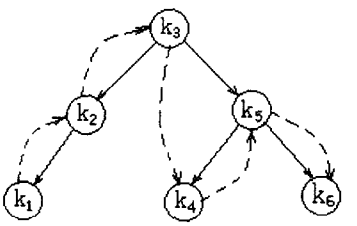

图 1.5 带有两个关系的一种数据结构示意图

从图 1.5 可以看出：数据结构 B 是图结构。但是，若只考虑关系 R1 则为树结构，若只考虑关系 R2 则为线性结构。

下面简要讨论数据的存储结构。

存储数据不仅要存储数据中的每个数据元素，而且要存储元素之间的逻辑关系。具体地说，若数据是集合结构，则只需要存储所有数据元素，不需要存储它们之间的任何关系；

若数据是线性结构、树结构或图结构，则除了要存储所有数据元素外，还要相应存储元素之间的线性关系、层次关系或网状关系。

数据的存储结构分为顺序、链接、索引和散列 4 种。

顺序存储对应一块连续的存储空间，该空间的大小要大于等于存储所有元素需占有的存储空间的大小，存储元素之间的联系（即逻辑结构）通常不需要附加空间，而是通过元素下标之间的对应关系反映出来，只要简单的计算就可以得到一个元素的前驱或后继元素的下标。顺序存储空间一般需要通过定义数组类型和数组对象来实现。

在链接存储结构中，元素之间的逻辑关系通过存储结点之间的链接关系反映出来，每个存储结点对应存储一个元素，同时存储该元素的前驱和后继元素所在结点的存储位置，或者说同时存储指向其前驱元素结点和后继元素结点的指针，通过这些指针能够直接访问到其前驱元素和后继元素。链接存储空间通过定义元素的存储结点类型和对象来实现，所有存储结点可以占用连续的存储空间（即数组空间），也可以占用不连续的存储空间，此空间是由动态分配的每个结点的空间形成的。

索引存储是首先把所有数据元素按照一定的函数关系划分成若干个子表，每个子表对应一个索引项，然后采用一种存储结构存储所有子表的索引项和采用另一种存储结构存储所有子表中的元素。如存储汉字字典时，需要采用索引存储，首先按偏旁部首划分所存汉字为若干子表，得到偏旁部首表，对于每个部首再按所属汉字的笔画多少划分子表，得到检字表，检字表中的每个汉字对应汉字解释表（即字典主体）中的一个条目；然后再分别存储部首表、检字表和汉字解释表。这里检字表是汉字解释表的索引，而偏旁部首表又是检字表的索引，它是汉字解释表的二级索引。当存储的数据量很大时，通常都需要采用索引存储，并且时常使用多级索引。

在索引存储中，各级索引表和主表（即数据元素表）通常都以文件的形式保存在外存磁盘上，访问任一数据元素时，都要根据该数据元素的特征依次访问各级索引表和最后访问主表，存取外存的次数至少等于建立索引的级数加 1。

散列存储方法是按照数据元素的关键字通过一种函数变换直接得到该元素存储地址的方法，该存储地址为相应数组空间中的下标位置。用于散列存储所有数据元素的相应数组空间称为散列表。通过定义用于计算散列存储地址的函数和定义存储数据元素的散列表能够实现散列存储结构。

以上简要叙述了数据结构的有关概念，在以后的各专题中将会做深入和具体的讨论。

1.2　抽象数据类型

抽象数据类型（Abstract Data Type，ADT）由一种数据结构和在该数据结构上的一组操作所组成。抽象数据类型包含一般数据类型的概念，但含义比一般数据类型更广、更抽象。一般数据类型由具体语言系统内部定义，直接提供给编程者定义用户数据，因此称它

们为预定义数据类型。抽象数据类型通常由编程者定义，包括定义它所使用的数据、数据结构以及所进行的操作。在定义抽象数据类型中的数据部分（含数据结构在内）和操作部分时，可以只定义数据的逻辑结构和操作说明，不考虑具体的存储结构和操作的具体实现，这样能够为用户提供一个简明的使用接口，然后再另外给出具体的存储结构和操作的具体实现，使得操作声明与实现分开，从而符合面向对象的程序设计思想。

抽象数据类型在 C++语言中是通过类类型来描述的，其数据部分通常定义为类的私有或保护的数据成员，它只允许该类或派生类直接使用，操作部分通常定义为类的公共的成员函数，它既可以提供给该类或派生类使用也可以提供给外部定义的类和函数使用。

在本书中，为了便于叙述和分析数据结构和算法，使读者容易理解和接受，所以在实现所定义的抽象数据类型时，把数据部分用一种已知的数据类型（如结构或数组等）来实现，把操作部分中的每个操作用普通函数来实现，这样能够同读者熟悉的 C 语言、C++语言，甚至其他计算机语言很好地兼容起来。

一种抽象数据类型的定义将采用如下书写格式：

```
ADT <抽象数据类型名> is
    Data:
      <数据描述>
    Operations:
      <操作声明>
end <抽象数据类型名>
```

【例 1.6】假定把矩形定义为一种抽象数据类型，其数据部分包括矩形的长度和宽度，操作部分包括初始化矩形的尺寸、求矩形的周长和求矩形的面积。

假定该抽象数据类型名用 RECtangle（矩形）表示，定义矩形长度和宽度的数据用 length 和 width 表示，并假定其类型为浮点（float）型，初始化矩形数据的函数名用 InitRectangle 表示，求矩形周长的函数名用 Circumference（周长）表示，求矩形面积的函数名用 Area（面积）表示，则矩形的 ADT（抽象数据类型）描述如下：

```
ADT RECtangle is
    Data:
       一个矩形 r，其长度和宽度分别用 length 和 width 表示
    Operations:
      //初始化矩形 r 的长度和宽度值为 len 和 wid
        void InitRectangle(Rectangle& r, float len, float wid);
      //求矩形 r 的周长并返回
        float Circumference(Rectangle& r);
      //求矩形 r 的面积并返回
        float Area(Rectangle& r);
end RECtangle
```

这里假定数据部分的矩形 r 是类型名为 Rectangle 的一个结构对象，该类型的具体定义如下：

```
struct Rectangle{
    float length, width;
};
```

下面给出每个操作的具体实现。

（1）初始化矩形尺寸

```
void InitRectangle(Rectangle& r, float len, float wid) {
     r.length=len;  //把 len 值赋给 r 的 length 域
     r.width=wid;  //把 wid 值赋给 r 的 width 域
}
```

该函数把两个值参 len 和 wid 的值分别赋给引用参数 r 的 length 域和 width 域，实现对一个矩形 r 的初始化。

（2）求矩形周长

```
float Circumference(Rectangle& r) {
    return 2*(r.length+r.width);
}
```

（3）求矩形面积

```
float Area(Rectangle& r) {
    return r.length*r.width;
}
```

求矩形周长和面积的函数分别使用一个矩形引用参数，当然也可以改为值参。

【例 1.7】把二次多项式 ax^2+bx+c 设计成一种抽象数据类型，假定起名为 QUAdratic，该类型的数据部分为三个系数项 a,b 和 c，操作部分为：

（1）初始化 a,b 和 c 的值，假定它们的默认值均为 0；

（2）做两个多项式加法，返回它们的和；

（3）根据给定 x 的值计算多项式的值并返回；

（4）计算方程 $ax^2+bx+c=0$ 的两个实数根，对于有实根、无实根和不是二次方程（即 a==0）这三种情况都要返回不同的整数值，以便返回后做不同的处理；

（5）按照 ax**2+bx+c 的格式输出二次多项式，在输出时要注意去掉系数为 0 的项，并且当 b 和 c 的值为负时，其前不能出现加号。