高等学校计算机专业教材

数 据 结 构

晋良颖 编

人民邮电出版社

图书在版编目(CIP)数据

数据结构/晋良颖编. -北京:人民邮电出版社,2002.2
高等学校计算机专业教材
ISBN 7-115-09371-7
Ⅰ.数… Ⅱ.晋… Ⅲ.数据结构-高等学校-教材 Ⅳ.TP311.12
中国版本图书馆 CIP 数据核字(2002)第 000932 号

高等学校计算机专业教材

数 据 结 构

◆ 编 晋良颖
责任编辑 滑 玉

◆ 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@ pptph.com.cn
网址 http://www.pptph.com.cn
读者热线:010-67180876
北京汉魂图文设计有限公司制作
北京朝阳展望印刷厂印刷
新华书店总店北京发行所经销

◆ 开本:787×1092 1/16
印张:19.75
字数:477 千字 2002 年 2 月第 1 版
印数:1-5 000 册 2002 年 2 月北京第 1 次印刷
ISBN 7-115-09371-7/TP·2262

定价:26.00 元

出版者的话

为了适应我国大学本科计算机专业教育发展对教材的需要，我社特邀请教育部所属的中国人民大学、中国地质大学、中国农业大学、北京科技大学、北京林业大学、北京语言文化大学（排名不分先后）6 所高等学校的“计算机科学与技术系”系主任及资深教授组成专家组，规划、编写、审订了本套教材。

读者对象　本套教材的主要读者对象是普通高等院校计算机科学与技术专业的学生，兼顾信息、自动化及机电类专业学生学习计算机课程的需要。由于这些专业对学生的培养目标是：掌握计算机软件与硬件的基本理论和方法，能从事计算机应用、软件研制、技术开发和管理工作的高级技术人才，因此，本套教材的内容既注意计算机高级技术人才所应具有的完整知识结构，又适当侧重主要专业知识。

教材特点　本套教材参考了美国 IEEE/CS 和 ACM 技术委员会 2001 年推荐的课程 CC2001(Computing Curricula 2001，参见 http://www.csab.org/~csab, http://cs.nju.edu.cn/~gchen/teaching/cc2001/overview-bok.html 2001)及我国几十所高等院校计算机专业的 2001 年教学计划进行规划。教材内容在选择国际上先进的计算机理论、技术的同时，务求符合我国目前教学环境的实际情况，有一定深度，又有较高的实用性。

根据大学教学的特点，本套教材主要包括必修课程、选修课程和辅助课程三类教材。除了每本教材内容自成体系外，还考虑了它在整个教学计划中的安排顺序，适当增加了承上启下的内容。

写作风格　本套教材根据内容需要，沿着“这是什么、有什么用、怎样用、怎样用得更好”的思路编写教材，通过讲解具体知识，传授学习方法，使学生达到掌握理论和技术的目的；同时力求文笔流畅，言简意赅。

教材每一章除基本知识外，还有本章要点、小结、思考与练习题。有些教材还附加教学大纲（包括教学重点、难点，讲授知识点的参考学时数），操作性较强的课程还配有实验教材（包括上机应具备的软硬件环境和实验内容及方法）。为了方便教师教学，本套教材提供了演示稿，可到人民邮电出版社网站（http://www.ptpress.com.cn）的“教材出版图书出版中心”→“教材附件”→“课件”下载。

本套教材的作者都具有多年教学经验，教材的草稿也都在各自的授课过程中多次使用。这套教材的出版，为计算机专业的师生提供了新的选择。我们希望这套以易教、易学，朴实、实用为特色的教材在培养信息化建设专业人才方面做出应有的贡献。

欢迎广大读者对本套教材的不足之处提出批评和建议。

2002 年 1 月

编 者 的 话

数据结构是高等学校计算机专业和信息管理专业必修的一门核心课程。它不仅是众多后续课程的综合性专业技术基础课，而且是程序设计特别是非数值计算的程序设计的重要基础。

学习数据结构，要学会从问题入手，分析研究计算机加工的数据特性，以便为应用所涉及的数据，选择适当的逻辑结构、存储结构及相应操作的算法，并初步掌握时间和空间的分析技术。学习本课程要进行复杂程序设计的训练，要学习大量的算法设计并进行上机实践，同时也为后续课程的学习做好必要的知识准备。

这门课程内容丰富，知识量大，所用到的技术多，又大多开设在低年级，学生会因知识准备和编程能力有限，而感到有较大学习难度。在此，编者力图奉献给广大读者一本可读性较强的教材，希望对读者掌握本课程有较大帮助。

本书的特点表现在以下几个方面。

（1）在介绍每种数据结构的主要操作时，都从描述算法思想入手，以类 C 语言的函数形式给出算法，并做了详细的注释。用类 C 语言使我们有时可以不拘泥于语言细节，突出算法的思路与框架；但另一方面，由于 C 语言已经相当精练，本书中绝大多数算法是直接用 C 语言写的，只要加上主函数，就可以上机运行。本书对所有函数的形参和局部变量都给出了类型定义，并对它们的含义和功用给出了注释，以便于读者对算法的理解。

（2）为了帮助读者掌握比较难于理解的算法，如在“图”一章的某些算法中，编者不仅给出了图示说明，还给出了举例数据在执行算法时的动态变化过程，使原来难懂的算法变得更易于接受。

（3）对一些典型问题，编者都尽量给出几种解法，例如，对比较经典的汉诺塔问题、经典的迷宫问题等。其中有些算法是编者自己长期教学经验的提炼和总结。

（4）对有的综合性较强，需要较大篇幅的算法，如迷宫和背包问题、用败者树实现多路归并的外排序、B+树的部分操作等，编者给出了能够上机运行的 C 语言完整程序示例。编者在给出程序之前，对程序中的算法思想进行了必要的说明，而且程序有较好的可读性和详细注释，相信对读者阅读程序能力和编程能力的训练都是有好处的。

（5）本书的算法实例丰富，有的选用了以往的“考研”试题，因此也可用于“考研”参考。

（6）本书在每章之后都有小结，并配有大量习题，读者可以从中选择课后算法练习和上机练习题。

本书共分 9 章。第 1 章引出数据结构与算法的一些基本概念和术语，并对算法描述和算法分析进行简要说明；第 2 章到第 4 章介绍了线性结构中的线性表、线性表的特例栈和队、线性表的推广字符串和数组；第 5 章和第 6 章介绍非线性结构树和图；对这些数据结构，给出它们的逻辑特性、存储表示及其应用和相应算法；第 7 章介绍排序，其中，除介绍各种实现方法外，还介绍了对算法进行改进的思路、对算法的时间复杂度的定性分析以及对各种算法的应用场合、适用范围的分析比较；第 8 章介绍查找，除对一般的内查找进行介绍外，还

结合文件组织中的文件索引及散列方法进行了详细介绍；第 9 章介绍文件组织。

本书是为计算机专业的本科教材，也可作电子信息类相关专业的选修教材，教授学时为 60 到 80。

由于本人水平所限，书中错误之处敬请读者赐教。

编者

2001 年 12 月

目　　录

第1章 绪 论

早期计算机主要应用于科学计算，其数据结构的特点是数据类型简单、算法复杂，所以，侧重于建立程序，称之为数值计算。随着计算机技术的发展与普及，其应用早已深入到人类社会的各个领域，早已从单纯的科学计算发展到控制管理和数据处理等非数值计算的工作中，处理对象不仅可以是数值，还可以是字符、表格、声音、图像等各种信息。非数值计算的数据十分广泛，而且通常有一定的结构。要对数据进行组织，不仅要研究处理对象的特性，还需要研究它们之间存在的关系。这就是“数据结构”形成和发展的背景。

1.1 数据结构研究什么

非数值计算的特点是数据类型复杂，而且数据量十分庞大。要对非数值计算设计出好的处理程序就必须解决好三方面的问题。

（1）要对所加工的对象进行逻辑组织。

这包括两方面的内容：①确定作为整体考虑和处理的数据元素，它包括哪些数据项；它们是什么类型；以及项与项之间存在着什么关系；②数据元素之间存在什么关系，例如，它们之间是线性关系还是非线性关系（树形关系还是网状关系）。

（2）如何把加工对象存储到计算机中。

数据结构在计算机中的表示（又称映像）称为数据的物理结构或存储结构。当我们选好了待处理问题的数据元素及它们之间的关系之后，就要把它们存储到计算机中去，以便用计算机进行处理。所以就有一个物理组织的问题。既要存储数据元素又要存储元素间的关系，其原则是要尽可能地兼顾节省存储和便于处理，即不仅便于算法的实现，还要尽可能使算法在空间和时间上都比较节省。

（3）数据运算。

数据运算是指定义在数据的逻辑结构上的一组操作。要对这些操作设计出相应的算法。“算法+数据结构=程序设计”是瑞士计算机学者 Niklaus Wirthrn 所著的一本书名，这一直是计算机工作者所公认的一句名言。当选好了数据的逻辑结构和物理结构之后，如果没有行之有效的算法，则不能对数据进行操作，实际问题得不到解决；同样，因为算法是作用于数据结构之上的，所以，没有数据结构，算法也就无用武之地了。数据运算的定义依赖于数据的逻辑结构，而运算的实现依赖于存储结构和所使用的程序设计语言。

我们举一个简单的例子作示意性说明。

［**例 1**］ 设有一本电话号码簿，有 N 个人的姓名和电话号码。要求设计一个程序，按人名查找号码，若不存在则给出不存在的信息。

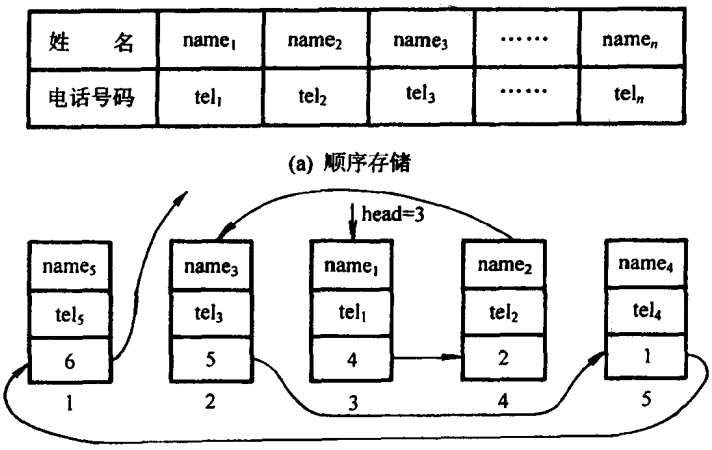
（1）逻辑结构的选取

首先，选取数据元素。很显然，人名和电话号码必须一一对应，必须作为一个整体来处理，所以应取作数据元素。

其次，结点间关系的选取。电话号码簿，实际上是一种查找表的结构。它的结点间的关系通常仅仅是同属一个集合的关系，在逻辑上，严格说并不存在某种特定的关系。所以，在选取结点间的关系时，要具体问题具体分析，有的是客观存在的关系，有时则是为了处理的方便而人为强加的关系。例如，假定号码簿中的人员是一个大学中的教职工，根据号码簿中人员的情况，我们可以选择一种顺序关系：或按工作单位排列，或按住址排列，或按姓名升序排列。总之，结点间关系不同，就意味着逻辑结构不同，因而相应的算法也会不同。

（2）存储结构的选取

存储结构的选取是指按什么方式把数据存入计算机。存入计算机的顺序称为物理顺序。例如，我们选取的逻辑顺序是按人名升序排列的，其存储结构就可以有两种不同的选择：一种是逻辑顺序和物理顺序一致；另一种是逻辑顺序和物理顺序不一致。如果一致，则存储顺序本身就代表了逻辑顺序，因此，不必另存结点间的逻辑关系了，这种存储结构称为顺序存储；如果二者不同，则还要另外存储结点间的逻辑顺序。假定我们用 $name_1$，$name_2$，$name_3$，…，代表人名的升序，并按这一顺序把结点存放在一维数组中，如；图 1-1（a）所示，由数组下标表示的物理顺序和逻辑顺序完全一致；如果结点的存放顺序是按任意的顺序存储，则其物理顺序和逻辑顺序不一致，而我们对数据的处理往往要依赖于逻辑顺序，因而，除了存储结点本身之外，还要存储结点间的关系。如图 1-1（b）所示，每个结点中有一个被称为指针的数据项，它指出结点按逻辑顺序的后继结点的地址，这种存储结构称为链式存储。



(a) 顺序存储

(b) 链式存储

图 1-1　两种存储结构

（3）算法设计

选好了逻辑结构和存储结构就可以设计算法了。存储结构不同，算法也不同。

按图 1.1（a）的方式存储，则由人名查找他的电话号码，就可以有两种方法：一种是线性查找，即从数组的第一个元素开始，逐个比较。另一种称为折半查找，这种方法速度要快

得多，特别是数据量很大时更为明显。其方法是将姓名先和查找区间的中间结点相比较，由于数据是按人名升序排列的，若待查结点较中间结点大，它必在查找区间的后半部，否则必在前半部，这样经过一次比较，就使查找区间缩小一半，再对缩小后的区间按上述方法继续查找，直到找到或得出查找失败的结论。

如果按图 1-1（b）的存储方式，不难看出，这时，只有惟一的一种方法，那就是顺序查找：从链头开始，沿着每个结点的指向其后继结点的指针，依次找下去。

存储结构不仅限于这两种，还可以用索引的方法、散列的方法等。有的数据，例如电话号码簿，还必须建成一个长期保留在外存的称为文件的存储结构，需要查找时，再按一定方式调入内存进行查找。当然，对不同的存储结构，就要有相应的各种操作的不同算法。

这个例子虽然很简单，但通过它我们可以了解到一个非数值运算的数学模型，它不是一个数学公式，而是根据实际问题的需要设计出的一个处理模型。设计过程如下：选取我们作为整体来进行考虑和处理的数据单位即选取数据元素，选取它的的逻辑结构（数据元素的构成和数据元素间的关系）和存储结构，然后设计出可能的处理方法。如果有多于一种方法，还要对它们进行比较分析，选取一个最好的方案。

数据元素之间的关系称为结构，本例仅涉及到一种简单的结构，如果我们选定的数据元素间的关系是按人名升序排列，则可认为是一种线性结构。非数值计算中还常常要涉及到更复杂的非线性结构，例如对大型文件建立树型索引、对一个复杂的交通图求最短路、对一个工程估算工期等就要用到树型结构和网状结构。

数据结构就是要研究按数据元素间的“结构”来进行分类的线性结构和非线性结构，研究它们的逻辑结构、相应的存储结构及设计各种数据结构中，有关操作的算法，并分析它们的时间和空间复杂度。

1.2 数据结构的发展概况和在计算机科学中的地位

数据结构作为一门独立的课程，是从 1968 年开始的。在此之前，它的有关内容已出现在其他课程中，如编译原理、操作系统等。1968 年，美国的一些大学的计算机系开设了本课程，但对课程的范围并未作明确规定，最初数据结构和研究表、树理论没有什么区别，后来为了研究数据的数学特性，又扩充了离散数学结构的内容。由于数据要在计算机中进行处理，就不仅要考虑数据本身的特性，而且还必须考虑存储结构，这又进一步扩大了数据结构的内容。之后，随着数据库系统的不断发展，在数据结构中，又增加了大型文件组织的内容。同年，美国的唐·欧·克努特教授所著的“计算机程序设计技巧”第一卷《基本算法》发表了，该书是第一本阐述数据的逻辑结构和存储结构及其操作的著作，从此，开创了数据结构的最初体系。

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，结构程序设计成为结构程序方法学的主要内容，人们就越来越重视数据结构，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。

我国从 1978 年开设本课程，目前，它不仅是计算机专业教学计划中的核心课程之一，而且是其他非计算机专业的主要选修课程之一。

数据结构在计算机科学中是一门综合性的专业基础课。它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，而且是实现编译程序、操作系统、数据库系统及其他系统和大型应用程序的重要基础。

数据结构发展的新趋势反映在两个方面，其一是面向各专门领域中特殊问题的数据结构的研究，如多维图形数据结构等；其二是从抽象数据类型的观点来讨论数据结构越来越被人们所重视。

1.3 基本概念和术语

数据（Data）是指对客观事物的符号表示。是所有能输入到计算机中被程序处理的符号的总称。其范围极其广泛，例如，声音、图像等，都可以通过编码成为计算机所能处理的数据。

数据元素指在计算机程序中，作为整体进行考虑和处理的数据的基本单位。一个数据元素可以由若干数据项组成，例如，我们讨论的电话号码簿的例子中，数据元素指姓名和电话号码的总体，而姓名和电话号码是组成它的两个数据项。数据项是数据的具有独立逻辑含义的不可分割的最小单位。又如一个学生的基本情况可以在学籍管理中选为数据元素，它应该包括学号、姓名、性别、出生年月等。在这些数据项中，能够惟一标识一个数据元素的项，例如学生的学号，称之为关键字或主关键字，而不能惟一标识一个数据元素的数据项，如学生的性别或成绩等，称为次关键字。

数据元素是一个逻辑上的概念，当我们把它存储到计算机中后，就把它的存储映像称为结点，也就是说结点是存储了的数据元素。有时，我们也并未对它们加以严格区分。

数据对象（Data Object）是性质相同的数据元素的集合，是数据的一个子集。例如，在电话号码簿的例子中，电话号码簿中的一行是一个数据元素，而整个电话号码簿就是数据对象。这里，整个电话号码簿究竟包括哪些人员则由应用的需要决定。

数据结构（Data Structure）是指相互之间存在一种或多种特定关系的数据元素的集合。数据结构分为逻辑结构和存储结构两种。

数据结构可以表示成一个二元组，形式地定义为：

$$\text{Data-Structure}=(D,S)$$

其中，D 是数据元素的有限集合；S 是 D 上关系的有限集合。

（1）逻辑结构。数据元素间的相互关系称为结构。由于这个关系是指数据元素间的逻辑关系，所以又称为逻辑结构，上述数据结构的二元组表示中，S 指的就是数据的逻辑结构。

按照数据元素间关系的不同，逻辑结构通常可以分为四种结构。

① 集合。集合中的元素，除了同属一个集合外，没有其他任何关系。

② 线性结构。线性结构中的数据元素存在一对一的关系，除始点无前驱，终点无后继外，其余所有数据元素都有惟一的一个前驱和一个惟一的后继。

[**例 2**]　linear=(D,R)

D={1,2,3,4,5,6,7,8,9,10}

R={<1,2>,<2,3>,<3,4>,<4,5>,<5,6>,<6,7>,<7,8>,<8,9>,<9,10>}

D 表示 10 个整数元素的集合，R 表示 D 上关系的集合，而 R 中只有一个关系即数据值（本例每元素只包含一个数据项）的大小关系。数据元素间的关系可用序偶来表示，例如<3,

4>就是一对序偶，其中，3 称为 4 的前驱，而 4 称为 3 的后继。还可以用图形的方式来直观形象地表示数据元素间的关系，如图 1-2 所示。图中我们用一个圆圈（有时也用矩形）表示数据元素（或结点），其中的数字或字母表示数据元素值。数据元素可以有很多数据项，在研究它们的抽象特性时，不妨把它们看成只有一项，也就是说，抽象成一个“点”，我们用“结点”来表示存储了的数据元素，同时也有抽象成一点的喻义，因为只考虑一个数据项（用它代表一个数据元素）并不影响我们对问题的分析和处理，反而给我们对问题的研究带来极大方便。但在实际应用时，则要根据需要认真地选取数据元素中的各数据项。在所有结构的图形表示中，结点间的关系用线段表示（包括有向线段和无向线段两种）。

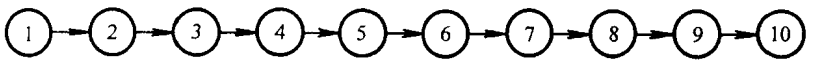


图 1-2　线性结构

③ 树型结构。树型结构中的元素存在一对多的关系，其中有惟一的一个数据元素没有前驱，称为根，其他所有数据元素都只有惟一的一个前驱（称为它的双亲），但可以有多个后继（称为它的子女）。

[例 3]　设 tree=（*D*,*R*）

D={a,b,c,d,e,f,g,h,i,j,k,l}

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

树型结构的图形表示如图 1-3 所示（由于树中结点间的关系都是自上而下的，所以图中省去了有向线段的箭头）。

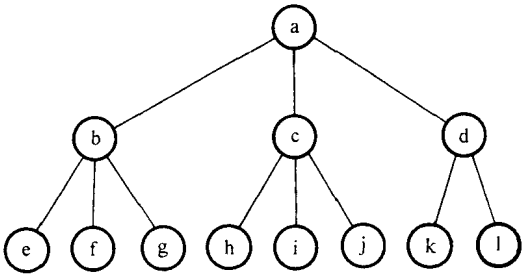


图 1-3　树型结构

④ 图型结构。图型结构中的数据元素存在多对多的关系，即任意两个数据元素都可能存在关系。

[例 4]　graph=(*D*,*R*)

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

R={<1,2>,<1,3>,<2,4>,<2,5>,<2,6>,<2,8>,<3,2>,<3,4>,<4,5>,<5,7>,<6,7>,<6,9>,<7,9>,<8,9>}

图型结构的表示如图 1-4 所示。图 1-4 表示的是一个有向图，结点间的关系是有方向性的，图中用有向线段表示结点间的关系。此外，还有另一种无向图，它的结点间的关系无方向性，图中用无向线段表示结点间的关系。

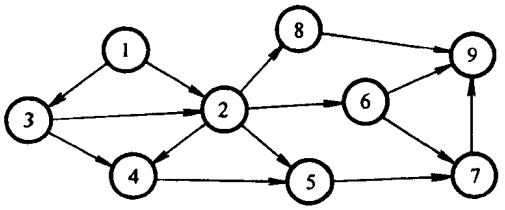


图 1-4　图型结构

（2）存储结构。数据结构在计算机中的表示（又称映像）称为数据的存储结构，又称物理结构。数据的存储结构包括数据元素的表示和元素间关系的表示。一个数据元素在计算机中的映像称为元素或结点。

① 顺序存储结构。顺序存储结构是利用数据元素在存储器中的相对位置来表示数据元素间的逻辑顺序。对于线性结构来说，如果数据元素的物理顺序和逻辑顺序完全相同，那么，这个物理顺序就代表了逻辑顺序，如图 1-1（a）所示。

② 链式存储结构。链式存储结构利用结点中的指针来表示数据元素间的关系，如图 1-1（b）所示。

③ 其他存储结构。如散列、索引等。

虚拟存储结构　由于我们讨论物理结构时是利用高级语言的数据类型来实现的。假如我们把 C 语言看成是执行 C 指令和 C 数据类型的虚拟处理器，则我们就把数据结构在 C 虚拟处理器中的表示称为虚拟存储结构。但在文中，简称为存储结构。

在 C 语言中，我们用它的结构类型来定义数据元素，用数组和指针来描述数据元素（结点）间的关系。

1.4　数据类型和抽象数据类型

在高级语言中，数据类型是一个值的集合及定义于其上的一组操作的总称。例如“整型”就是一个数据类型，我们也可以称之为一个抽象数据类型。抽象是强调数据类型的数学特性，而不管它们在不同处理器上的实现方法和细节。通常，在软件内部对它们实现了信息隐蔽和封装，所以，不论在不同处理器中其内部细节如何不同，在用户看来都是一样的，并且可按相同方法去使用它们。以这种观点看，高级语言中的基本类型都是抽象数据类型。

除此之外，抽象数据类型还可以有更广的应用范畴，即它不仅限于高级语言中已实现了的数据类型，还可以包括用户在设计软件系统时，自己定义的数据类型。为了提高软件的复用率，在近代程序设计方法学中指出，一个软件系统的框架应建立在数据之上，而不是建立在操作之上。即在构成软件系统的每个相对独立的模块中，定义一组数据和作用于这些数据上的一组操作，并在模块内部给出它们的表示和实现细节，而在模块外部，只使用抽象的数据和操作。该抽象类型定义的层次越高，其复用率也就越高。

由以上讨论，我们已经知道，数据结构是一个二元组，是结点的集合和结点间关系的集合，这是数据结构的逻辑定义，而我们研究数据结构更为重要的方面是要设计定义在其上的一组操作的算法，操作的种类和数目不同，数据结构能起的作用也不同（即使逻辑结构相同）。由此看来，如果我们能把一个数据结构和定义在其上的一组操作封装起来，使之成为一个抽

象数据类型，则可以提高它们的复用率。这是近年来人们对研究数据结构所寄予的更高的期望。

和数据结构的形式定义相对应，抽象数据类型可以用一个三元组来表示：

$ADT=(D, S, P)$

其中，D 是数据对象，用结点的有限集合表示；

S 是 D 上的关系的集合，通常用结点间序偶（或偶对）的集合来表示；

P 是对 D 的基本操作的集合。基本操作的定义格式为：

基本操作名（参数表）

初始条件：〈初始条件描述〉

操作结果：〈操作结果描述〉

在我们的课程中，最重要的是要让读者通过学习，掌握根据结点间关系（R）的不同：有哪些主要的数据结构种类；通常有哪些存储方式；在各种存储方式中，有哪些常用的操作（P），对这些操作，如何设计出相应的算法，对所设计的算法的时间复杂度和空间复杂度应如何进行分析和评价等。我们不可能对一个数据结构的所有操作都去描述和讨论，只能对那些重点的、重要的和有代表性的算法进行详细介绍，而对那些比较简单的或读者可以根据已学内容自己“举一反三”的算法，就粗略介绍或不介绍了，因此不罗列出各种数据结构的所有或绝大部分操作，而只对将要介绍的某些操作的算法用 C 语言的函数给出。通过其形参表，就知道调用时应给出何种“初始条件”，而函数的运行结果或返回值，也就是“操作结果”了，因此，不再另行列出。

采用抽象数据类型的方法要求把数据结构和作用于其上的操作封装在一起，这里有两点原因，使本书不拟采用此种描述方法。其一，定义抽象数据类型，最好用面向对象的方法，用 C++语言的类来描述比较方便、有效，但本课程大都在低年级开设，用 C 语言的描述方法更符合学生的实际情况；其二，由于实际问题千变万化，数据模型和算法也形形色色，因此，抽象数据类型的设计和实现，就不可能像基本数据类型那样规范、内置和一劳永逸，所以，结合具体问题的实际需要来定义抽象数据类型，会更具实用性。本书仍以面向过程的方法，将主要精力用在对算法的设计和描述上，让学生在掌握各种数据结构的主要功能的同时，更进一步提高软件设计的能力。

1.5 算法和算法分析

本书在高级语言的虚拟层次上来讨论数据结构的存储和算法的实现。过去，我们常用一种专门描述算法的“伪码”，例如笔者曾经选用过多种不同的描述算法的语言，如 SPARKS，类 Pascal，类 C 等，其目的是为了把主要精力集中到对算法的描述上。本教材采用类 C 语言作描述算法的语言，我们一律以函数的形式讨论算法。由于读者都熟悉 C 语言，对形参和函数内部局部变量的类型定义，只会使读者对它们的理解更清楚，而不会有“拘泥于语言细节”之嫌，所以，本书不省略对形参和局部变量的类型声明，而且，本书中大多数算法应该说就是 C 语言的子函数，加上主函数就可对它们进行调用、运行了。但我们也并不需要保证每个函数都是可以直接上机运行的，毕竟我们是讨论算法，而不是程序设计，所以，也允许在算法中，省去某些例如输入、输出之类的细节描述或用文字说明某个无须详细展开描述的功能。

1. 算法

算法是对特定问题的求解步骤的描述，它是指令的有限序列，其中，每一条指令表示一个或多个操作。算法有以下几个重要特性。

（1）一个算法有零个或多个输入，这些输入取自某个特定的对象的集合。当我们用函数来描述算法时，输入往往是通过形参表示的，在它们被调用时，从主调函数获得输入值。

（2）一个算法有一个或多个输出。它们是和输入有着某些特定关系的量，通常是对输入进行加工得到的结果。当我们用函数表示算法时，输出多用返回值或指针类型的形参表示。

（3）有穷性。一个算法必须总是在执行有穷步后结束，且每一步都必须在有穷的时间内完成。

（4）可行性。一个算法中所描述的操作，必须可以通过已经实现的基本运算的有限次执行来完成。

（5）确定性。算法中的每一条指令必须没有二义性，对相同的输入，必须有相同的执行结果。

2. 算法设计的要求

一个好的算法应满足以下要求。

（1）正确性（Correctness）。算法是针对一个具体问题的求解而设计的。算法是否正确，通常要将其转化为程序，通过上机检验。程序要对几组输入数据得出满足合乎规格说明要求的结果；更进一步的要求是对精心选择的、苛刻的以及带有刁难性的几组输入数据得出满足合乎规格说明要求的结果；对于最高层次的要求是对一切合法的输入数据都能得出满足规格说明要求的结果；由于输入数据量太大，逐一验证是不可能的。对大型软件需要进行专业测试。

（2）可读性（Readability）。算法的设计主要是为了人的阅读与交流，算法的可读性好，有利于人们对算法的理解，也就便于对算法进行改进、调试、移植以及维护等。

（3）健壮性（Robustness）。指对算法适应性的要求，一个算法只能处理合法的输入是不够的，实际应用中，各种错误的输入都有可能遇到，因此，算法应考虑到各种可能的非法输入，对它们做出相应的反应或处理。

（4）高效性。要求算法时间复杂度和空间复杂度都尽可能低。

当然，除了正确性之外，其余几条有时是不可兼得的，例如，时间复杂度和空间复杂度就常常是矛盾的，所以要根据实际情况进行权衡。

3. 算法的时间复杂度

算法执行时间需通过依据算法编制的程序在计算机上运行时所消耗的时间来度量。而度量一个程序的执行时间通常有两种方法。

（1）事后统计法。此方法利用计算机内部的计时功能，对解决同一问题的多个不同算法所编写的程序，通过一组或多组相同的统计数据来对比。这种方法的缺点是，统计量依赖于计算机硬、软件，有时甚至会掩盖算法的优劣。