21世纪高等教育计算机规划教材

COMPUTER

数据结构

Data Structure

周颜军 王玉茹 关伟洲 编著

- 多年教学与实践经验的融合
- 立足考研大纲及教学需求
- 内容严谨、实例充分、教学资源丰富

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等教育计算机规划教材

COMPUTER

数据结构

Data Structure

周颜军 王玉茹 关伟洲 编著

人民邮电出版社

北京

图书在版编目（CIP）数据

数据结构 / 周颜军，王玉茹，关伟洲编著. -- 北京
：人民邮电出版社，2013.9
21世纪高等教育计算机规划教材
ISBN 978-7-115-32764-2

Ⅰ. ①数… Ⅱ. ①周… ②王… ③关… Ⅲ. ①数据结构－高等学校－教材 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2013)第189069号

内 容 提 要

本书系统地介绍了各种常用的数据结构的逻辑特征、存储方式和基本运算。主要内容包括：顺序表、栈、队列、链表、串、树形结构、图、多维数组、广义表、排序、查找和文件等。本书结构清晰，内容充实，实例丰富，符号、图表规范。既适合于教师课堂讲授，也便于自学者学习参考。

本书可作为高等院校计算机专业或信息技术等相关专业的本科教材，也可作为参加研究生入学考试、自学考试的考生以及从事计算机工程和应用的科技人员的参考用书。

◆ 编　　著　周颜军 王玉茹 关伟洲
责任编辑　许金霞
责任印制　彭志环　焦志炜

◆ 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　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：18.75　　2013年9月第1版
字数：492千字　　2013年9月北京第1次印刷

定价：39.80元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前 言

数据结构是计算机学科的一门重要的专业基础课，也是一门核心课程。数据结构在计算机专业的诸门课程中起到相互衔接、承前启后的作用。它的先修课程有离散数学、程序设计语言（如 C/C++ 语言）等，同时它又是其后续课程，如操作系统、编译原理、数据库系统、人工智能等的重要基础。数据结构也是培养非计算机专业学生计算机素质和软件设计能力而重点选择的一门基础课。

数据结构课程的目的是介绍各种常用的数据结构，阐明各种数据结构之间的内在逻辑关系，讨论它们在计算机中的存储表示，以及对这些数据进行的操作和实际算法。不仅为读者学习后续软件课程提供必要的基础知识，而且更重要的是在软件设计和编程水平上得以进一步的提高。通过对不同存储结构和相应算法的分析对比及上机实践，增强读者根据实际问题特征选择合适的数据结构并掌握求解算法的时间、空间复杂性的能力。

本书是根据数据结构课程的教学大纲，在分析和借鉴国内外同类教材的基础上，结合作者多年讲授“数据结构”课程的教学经验和体会编写而成的。全书分为 10 章和 3 个附录：第 1 章是概论部分，主要介绍数据结构的基本概念和算法描述、算法分析等内容。第 2 章讨论线性表及线性表的顺序存储结构——顺序表，主要介绍向量、栈和队列三种数据结构及递归的实现机制。第 3 章讨论线性表的链接存储结构——链表，主要介绍单链表、双链表、循环链表、栈和队列等内容。第 4 章介绍串，主要介绍串的基本概念、串的存储结构、基本操作和模式匹配。第 5 章是树形结构，主要介绍树与二叉树两种数据结构。包括树（森林）和二叉树的概念、存储结构、树(森林)和二叉树的遍历、线索二叉树、堆、哈夫曼树和树形结构的应用问题。第 6 章介绍图结构，主要介绍图的基本概念、图的存储结构及图的遍历、最小（代价）生成树、最短路径、拓扑排序、关键路径等内容。第 7 章介绍多维数组和广义表，主要讨论多维数组、特殊矩阵的存储表示与寻址公式，稀疏矩阵的存储方式，以及广义表的概念、存储结构与运算。第 8 章介绍各种排序方法，包括排序的基本概念、直接插入排序、Shell 排序、折半插入排序、表插入排序、冒泡排序、快速排序、直接选择排序、堆排序、归并排序、基数排序和外排序等内容。第 9 章介绍各种查找方法，主要讨论线性表的顺序查找、折半查找和分块查找，树形表有二叉排序树、最佳二叉排序树、AVL 树、B-树与 B+ 树，以及散列表的查找。第 10 章文件结构，主要介绍有关文件的基本概念、顺序文件、索引文件、索引顺序文件、散列文件、多重表文件和倒排文件。为方便读者的学习，本书还安排了 2 个附录。附录 A 是“Visual C++ 6.0 集成开发环境介绍”，书中有的算法给出了对应的 C/C++程序和机器执行结果，习题中也有不少算法题。安排这部分内容是方便读者将算法转换为程序并上机运行，以加深对问题的理解和实际能力的提高。附录 B 是“常用字符与 ASCII 码对照表”，这在讨论字符类型数据（如：第 4 章串）时会用到。

本书力求叙述通俗易懂，严谨流畅，内容充实，示例丰富，符号、图表规范，既适合于教学又便于自学。本书注意数据结构和算法的有机结合，注重算法的完整性，在给出算法之前，对其实现的基本思想和要点都做了详细的讨论，算法除了有对应 C/C++程序外，有的还给出了机器运行的结果，便于读者深入的理解和掌握。

本书的另一个特点是：对算法的描述采用多种方式，书中采用接近于自然语言的伪语言和 C 语言两种方式来同时描述算法。另外主要章节还给出了数据结构的 ADT 和 C++ 的类描述（可从人民邮电出版社教学服务与资源网 www.ptpedu.com.cn 免费下载）。这不仅可以使读者从中学到多种描述算法的方法，也便于各类读者的使用。

本书是可作为普通高等学校计算机专业数据结构课程的教材，也可作为信息技术与管理等专业的教材和教学参考书；同时也可供从事计算机软件开发和计算机应用相关的工程技术人员参考。书中全部内容可在 60～80 学时内完成。“*”的章节可由教师根据授课时数进行取舍。

本书在编写过程中，得到了人民邮电出版社的大力支持，也得到了东北师范大学计算机科学与信息技术学院领导和同事的鼓励和帮助，在此一并表示诚挚的谢意。

由于时间仓促和作者水平有限，书中难免存在疏漏和错误，敬请读者批评指正。

作　者

2013 年 6 月

目 录

第1章 概论

用计算机解决各种实际问题的实质就是数据表示和数据处理，这也可以归结为对数据结构和算法的探讨。数据结构和算法是计算机科学中的两个基本问题，本章主要围绕这两个问题做概括性的介绍。

1.1 数据结构的概念

数据（data）是信息的载体，是描述客观事物的数字、字符以及所有能够输入到计算机中并被计算机程序处理的符号的集合。计算机能够处理多种形式的数据。主要分为两大类：一类是数值型的数据，主要用于工程和科学计算等领域；另一类是非数值数据，如字符型数据，以及图形、图像、声音等多媒体数据。

数据元素（data element）是表示数据的基本单位，是数据这个集合中的一个个体。在数据结构中数据元素经常称之为结点（node）。一个数据元素又可以由若干个数据项（data item）组成。数据项有两种：一种是初等项，是具有独立含义的最小标识单位；另一种是组合项，是具有独立含义的标识单位，它通常由一个或多个初等项和组合项组成。

数据对象（data object）是具有相同性质的数据元素的集合，是数据这个集合的一个子集。例如，整数数据对象可以是集合 I = { 0, ±1, ±2, … }。英文字母的数据对象可以是集合 letter = { A, a, B, b, …, Z, z }。有些数据对象表现为复杂的形式，例如表 1.1 所示为一个单位职工工资情况的一个数据对象。

表 1.1　　职工工资表

编号	姓名	发给项			扣除项			实发工资
		基本工资	工龄工资	交通补助	房租费	水电费	托儿费	
001	丁一	2200.00	10.00	150.00	100.00	80.00	60.00	2120.00
002	王二	3000.00	25.00	150.00	120.00	150.00	0.00	2905.00
003	张三	2400.00	12.00	130.00	100.00	100.00	60.00	2282.00
004	李四	2800.00	20.00	20.00	120.00	130.00	0.00	2590.00
…	…	…	…	…	…	…	…	…

每个职工的工资情况占一行，每一行则是一个数据元素。每一个数据元素由编号、姓名、发给项（基本工资、工龄工资、交通补助）。扣除项（房租费、水电费、托儿费）、实发工资等数据项组成。其中发给项和扣除项为组合项，其他的数据项均为初等项。而这个工资表就是一个数据对象。

通常，数据对象中的数据元素不是孤立的，而是彼此相关的，它们彼此之间存在的相互关系称为结构。简单地说，数据结构就是要描述数据元素之间的相互关系，而一般并不着重于数据元素的具体内容。虽然数据结构至今还没有一个公认的标准定义，但一般数据结构都联系着：数据之间的逻辑关系、数据在计算机中的存储方式以及数据的运算（操作）三个方面。分别称为数据的*逻辑结构*（logical structure）、*存储结构*（storage structure）和运算（即对数据所施加的操作）集合，存储结构也称为*物理结构*（physical structure）。因此，*数据结构*可以定义为：按某种逻辑关系组织起来一批数据，以一定的存储方式把它存储于计算机的存储器之中，并在这些数据上定义了一个运算的集合，就叫作一个数据结构。

1.2 数据结构的组成与分类

1.2.1 数据的逻辑结构

数据的逻辑结构可以形式地描述为一个二元组

$$Data\_Structure = (D,R)$$

其中：D 是数据元素（结点）的有穷集合；R 是 D 上关系的有穷集合，每个关系都是 D 到 D 上的关系。在不发生混淆的情况下，通常把数据的逻辑结构直接称为数据结构。

例如，线性表的逻辑结构可以表示为

$Linear\_List = (D,R)$

$D = \{ a_0, a_1, \cdots, a_{n-1} \}$

$R = \{ r \}$

$r = \{ < a_{i-1}, a_i > \mid a_i \in D, 1 \leqslant i \leqslant n-1 \}$

即根据 r，D 中的结点可以排成一个线性序列：

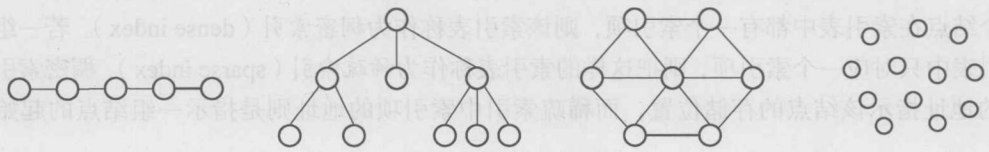
$$a_0, a_1, \cdots, a_{n-1}$$

其中有序对 $<a_{i-1}, a_i>$ 表示 a_{i-1} 与 a_i 这两个结点之间存在（邻接）关系，并称 a_{i-1} 是 a_i 的前驱，a_i 是 a_{i-1} 的后继。a_0 为开始结点，它对于关系 r 来说没有前驱，a_{n-1} 为终端结点，它对于关系 r 来说没有后继，D 中的每个结点至多只有一个前驱和一个后继。

数据的逻辑结构可以分为两大类：一类是线性结构，另一类是非线性结构。

线性结构有且仅有一个开始结点和一个终端结点，并且每个结点至多只有一个前驱和一个后继。线性表是一种典型的线性结构。前面介绍的职工工资表就是一个线性表。

非线性结构中的一个结点可能有多个前驱和后继。如果一个结点至多只有一个前驱而可以有多个后继，这种结构就是树形结构。树形结构是一种非常重要的非线性结构。如果对结点的前驱和后继的个数都不作限制，即任何两个结点之间都可能有邻接关系，我们把这种结构就叫作图。图是更一般、更为复杂的一种数据结构。数据这几种逻辑结构如图 1-1 所示，图中的小圆圈表示结点，结点之间的连线代表逻辑关系，即相应数据元素之间有邻接关系。



（a）线性结构　　（b）树形结构　　（c）图形结构（网状结构）　　（d）集合结构

图 1-1　基本的逻辑结构

1.2.2　数据的存储结构

数据的逻辑结构是从逻辑关系上来描述数据，它与数据的存储无关，是独立于计算机的。因此，数据的逻辑结构属于用户的视图，是面向实际问题的。可以看作是从具体问题中抽象出来的数学模型，它反映了数据组织的“本质”，是数据组织的主要方面。抓住了这种本质，就可以根据解题需要重新组织数据，即把数据中的所有数据元素按所需的逻辑结构重新进行安排。例如，对于表 1.1 表示职工工资情况的表格，既可以按照原样组织成线性表，也可以重新组织成树形结构，具体的选择应根据解题的需要来决定。数据的存储结构是数据的逻辑结构在计算机存储器中的实现，它是依赖于计算机的。也可以说，数据的存储结构是数据的逻辑结构用计算机语言的实现（或称为存储映象），它是依赖于计算机语言的。对机器语言而言，存储结构是具体的，但在数据结构中只在高级语言的层次上来讨论存储结构。

计算机的存储器（内存）是由有限多个存储单元组成的，每个存储单元都有一个唯一的地址，各存储单元的地址是连续编码的，每个存储单元 w 都有唯一的后继单元 w'= suc(w)。w 和 w' 称为*相邻单元*。一片连续的存储单元的整体称为*存储区域*（M），假设一个存储单元的长度为 c，则有 w' = suc(w) = w + c。

设有逻辑结构 B = (D, R)，要把 B 存储在计算机中，就应该建立一个从 D 的结点到 M 的单元的映象 f：D → M，即对于每一个 k ∈ D，都有唯一一个 w ∈ M，使得 f(k) = w，w 为结点 k 所占存储空间的首单元地址。这里使用 LOC(k) 表示结点 k 对应的存储单元的首地址。

数据的存储结构有以下四种基本的存储方式。

1. 顺序存储方式

顺序存储方式是把逻辑上相邻的结点存储在物理位置也相邻的存储单元里，结点之间的逻辑关系用存储单元的邻接关系来体现，即在所存储的区域中原来逻辑上相邻的结点其物理位置也相邻。由此得到的存储表示称为*顺序存储结构*（sequential storage structure）。顺序存储主要用于线性结构，非线性结构也可以通过某种线性化的方法来实现顺序存储。通常顺序存储是用高级语言的一维数组来描述的。

2. 链接存储方式

链接存储方式对逻辑上相邻的结点不要求在存储的物理位置上亦相邻，结点间的逻辑关系是由附加的指针字段来表示的。由此得到的存储表示称为*链接存储结构*（linked storage structure），通常要借助于程序设计语言的指针类型来描述它。非线性结构常用链接存储方式，线性结构也可以使用链接存储方式。

3. 索引存储方式

索引存储方式是在存储结点数据的同时，还需建立附加的索引表。索引表中的每一项称为

索引项，一般由关键字和地址组成。关键字（key）是能够唯一标识一个结点的一个或多个字段。若每个结点在索引表中都有一个索引项，则该索引表称作为稠密索引（dense index）。若一组结点在索引表中只对应一个索引项，则把这样的索引表称作为稀疏索引（sparse index）。稠密索引中索引项的地址指示该结点的存储位置，而稀疏索引中索引项的地址则是指示一组结点的起始存储位置。

4. 散列存储方式

散列存储方法就是根据结点的关键字通过反映结点与存储地址之间对应关系的函数直接计算出该结点的存储地址（或位置）。

这 4 种基本的存储方法形成了四种不同的存储结构，如图 1-2 所示。

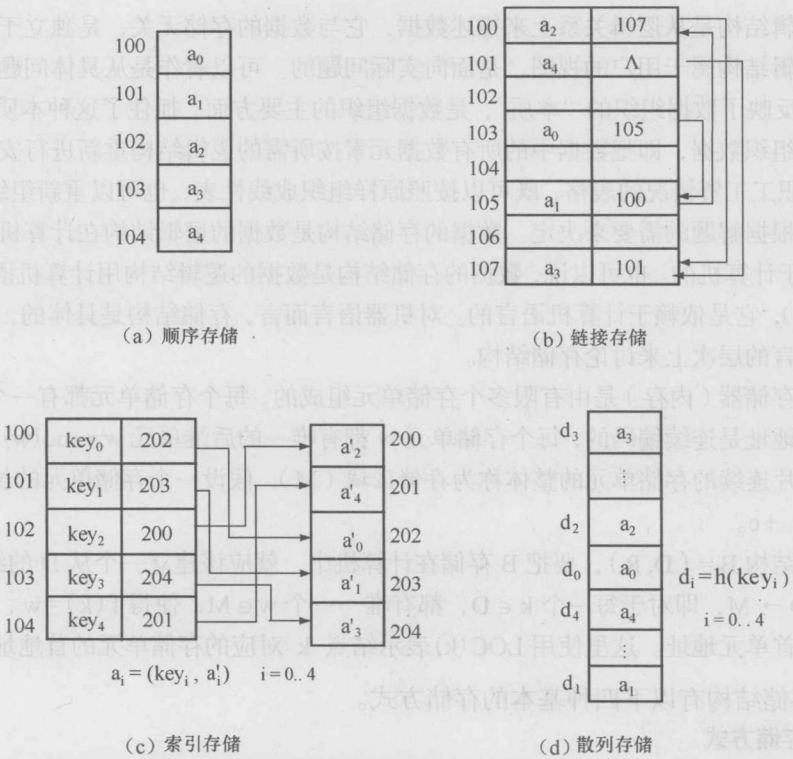


图 1-2　4 种基本的存储方式

如果结构所占用的存储区域都分配给了数据，则这样的存储结构称为紧凑结构，否则称为非紧凑结构。顺序存储方式是紧凑结构，而链接存储方式是非紧凑结构。

结构的存储密度（storage density）定义为数据本身所占的存储量与整个结构所占的存储量之比，即

$$d = \frac{\text{数据本身所占的存储量}}{\text{整个结构所占的存储量}}$$

显然，紧凑结构的存储密度为 1，非紧凑结构的存储密度小于 1。存储密度越大，则存储空间的利用率越高。但是，存储附加的信息也会带来运算上的方便。例如，在链表中，由于存储了指针，所以链表与顺序表相比，它进行插入和删除运算就会方便得多。这也体现了“牺牲空间来换取时间”的思想。

1.2.3　数据的运算（集合）

数据的运算是定义在数据的逻辑结构上的，每种逻辑结构都有一个运算的集合。这些运算实际上是定义在抽象的数据上所施加的一系列的操作，所谓抽象的操作，是指我们只知道这些操作是“做什么”，而无需考虑“如何做”的问题。但运算的实现要在存储结构上进行，只有确定了的存储结构之后，我们才考虑如何具体实现这些运算（即算法）。如何描述算法的问题在本章后面讲述。

常见的一些运算有如下几种。

（1）查找（检索）；

（2）插入；

（3）删除；

（4）修改（更新）；

（5）排序；

（6）合并；

（7）分拆等。

数据的逻辑结构、存储结构和运算（集合）是数据结构的三要素，三者之间既有区别、又有联系。严格地说，这三者共同构成了一个完整的数据结构。如果有两个数据结构被视为是相同的，则它们必须在逻辑结构、存储结构和运算集合三个方面均相同。只要有一个方面两者不相同，就把它们视为是不同的数据结构。例如，对于相同的逻辑结构 —— 线性表，由于采用了不同的存储结构：一个采用了顺序存储结构，而另一个采用了链接存储结构，因而把它们看作是两种不同的数据结构，一个叫作顺序表，而另一个叫作链表。又例如，在顺序表中，两个数据结构的逻辑结构和存储结构均相同，但是定义的运算集合及其运算的性质不同，也导致出完全不同的数据结构，这就是我们在第 2 章里要讲述的向量、顺序栈和顺序队列。

1.3　数据类型与抽象数据类型

1.3.1　数据类型

数据类型（data type）是一组性质相同的值的集合以及在这些值上定义一组操作（运算）的总称。每一种计算机高级语言都有自己的数据类型定义。在通用的计算机高级语言中，一般都具有整数、实数（浮点数）、枚举、字符、字符串、指针、数组、记录（结构）、联合和文件等数据类型。如整数类型在计算机系统中通常用两个字节和四个字节表示，如果采用两个字节，则整数的表示范围在 -32768 ~ 32767 之间；如果采用四个字节，则整数的表示范围在 -2 147 483 648 ~ 2 147 483 647 之间。这依赖于具体的机器实现系统。除了定义整数的取值范围之外，还规定了对整数可施加的加、减、乘、除和取模等运算。

按值是否可“分解”，数据类型可分为简单类型（也称为初等类型或基本类型）和结构类型（或组合类型）两种。

简单类型中的每个数据都是无法再分割的整体，例如：

FORTRAN 语言提供了整型、实型、复型和布尔型等简单数据类型；

PASCAL 语言提供了整型（integer）、实型（real）、字符型（char）、布尔型（boolean）和指针型（pointer）等简单数据类型；

C 语言提供了整型、实型（浮点型）、字符型、枚举类型、指针型和空类型等简单数据类型。

结构类型其值可分解为若干个成份（或称为分量），如 C 语言的数组、结构等数据类型。初等类型通常是由语言直接提供的，而组合类型则是由用户借助于语言提供的描述机制自己来定义的，它通常是由标准类型派生的，因此它也是一种导出类型。通常数据类型可以看作是程序设计语言中已实现的数据结构。

程序设计语言允许的数据类型越多，它的处理能力就越强、应用范围也越广。因此，程序设计语言所提供的数据类型的多寡，也是衡量程序设计语言功能强弱的一个重要指标。

1.3.2 抽象数据类型

抽象是对事物的简化描述，就是抽取反映问题本质的东西，而忽略非本质的一些细节。抽象可以分为不同的层次，低层次抽象可以作为高层次抽象的一种实现。抽象是人们理解复杂现象和求解复杂问题时经常使用的一种方法。

数据类型已经反映出对数据的抽象。例如，在计算机中使用二进制定点数和浮点数实现数据的存储和运算，而在汇编语言中使用者可以直接使用它们的自然表示，如 123，1.4E10, 25.2 等，不必考虑它们实现的细节，它们是对二进制数据的抽象。在高级语言中，出现了整型、实型、字符型、指针等数据类型，给出了更高一级的数据抽象。随着抽象数据类型（abstract data type）和面向对象程序设计语言的出现，可以进一步定义出更高层次的数据抽象，各种表、树和图，甚至窗口和管理器等。这种数据抽象的层次为软件设计者提供了有力的手段，使设计者可以从抽象的概念出发，从整体上进行考虑，然后自顶向下，逐步展开，最后得到所需要的结果。

抽象数据类型是指抽象数据的组织和与之相关的操作。抽象数据类型通常是由用户自己定义，用以表示应用问题的数据模型，它可以看作是数据的逻辑结构及逻辑结构上定义的操作。

一个 ADT 可形式描述为：

```
ADT 抽象数据类型名 {
    Data
      数据元素集合及数据元素之间的逻辑关系的描述
    Operations
      构造函数
        Initial value:      用来初始化对象的数据
        Process:        初始化对象
      操作 1
        Input:              用户输入的数据
        Preconditions:      系统执行本操作前数据所需的状态
        Process:            对数据进行的处理
        Output:             操作后返回的数据
        Postconditions:     系统操作后数据的状态
      操作 2
        … …
      操作 n
        … …
} //ADT  抽象数据类型名
```

例 1.1　圆的 ADT 描述。圆是平面上与圆心等距离的所有点的集合。从图形显示的角度看，圆的抽象数据类型应包括圆心和半径；但从计量的角度看，其抽象数据类型只需要半径。这里仅从计量的角度给出圆的 ADT，它包括求圆的周长和面积的操作。

```
ADT  Circle {
  Data
    非负实数，表示圆的半径
  Operations
    构造函数
      Initial value:  圆的半径
      Process:        给圆的半径赋初值
    Circumference
      Input:          无
      Preconditions:  无
      Process:        计算圆的周长
      Output:         返回圆的周长
      Postconditions: 无:
    Area
      Input:          无
      Preconditions:  无
      Process:        计算圆的面积
      Output:         返回圆的面积
      Postconditions: 无:
} //ADT  Circle
```

抽象数据类型可以看作是描述问题的模型，它独立于具体实现。它的优点是将数据和操作封装在一起，使得用户程序只能通过在 ADT 里定义的某些操作来访问其中的数据，从而实现了信息屏蔽与隐藏。在 C++ 中，可以用类的说明来表示 ADT，用类的实现来实现 ADT，因此 C++ 中实现的类相当于数据的存储结构及其在存储结构上实现对数据的操作。

ADT 和类的概念反映了软件设计的两层抽象：ADT 相当于在概念层（抽象层）上描述问题，而类相当于在是实现层上描述问题。

本书在正文中还仍然用自然语言的方式来描述数据的逻辑结构，它对应用于在概念层（抽象层）上描述的 ADT。用 C 语言的类型定义来描述数据的存储结构，并用接近于自然语言的一种伪语言与 C/C++ 语言两种方式来描述对数据的操作（即算法）。这样来处理，可以使读者从中了解和学习到描述算法的多种方式。

1.4　算法的概念与描述

1.4.1　算法的概念

要让计算机求解问题，除了要选择恰当的数据结构之外，还需要制定出解决问题的切实可行的方法和步骤，这就是所谓的计算机算法。由于数据结构包含着运算集合，这些运算的实现是通过算法来完成的，本书在讨论各种数据结构的同时，还介绍了很多相关的算法。早在 20 世纪 70

年代，D.E.Knuth 就指出，计算机科学就是研究算法的学问。因此，算法也是本课程的重点内容之一。下面先介绍算法的概念。

算法（algorithm）是规则的有穷集合，这些规则规定了解决某一特定类型问题的一个运算（操作）序列。此外一个算法应当具有以下特性。

（1）输入：一个算法必须有若干个输入（包括 0 个）。

（2）输出：一个算法应该有一个或多个输出。

（3）有穷性：一个算法必须总是在执行有穷步之后结束。

（4）确定性：算法的每一步都应确切地、无歧义地定义。即对于每一种情况，需要执行的动作都应当严格的、清晰的规定。

（5）可行性：算法中每一个运算都应是基本的、可行的，也就是说，它们原则上都是能够由人们仅用笔和纸做有穷次运算即可完成的。

需要指出的是，算法的含义与程序十分类似的，但也有明显的差别。一个程序并不一定需要满足上述的第 3 个特性（有穷性），例如操作系统，只要整个系统不受破坏，操作系统就无休止地为用户提供服务，永不结束。另外，程序应是用机器可执行的某种程序设计语言来书写的，而算法通常并没有这样的限制。

1.4.2 算法的描述

算法设计人员在构思和设计了一个算法之后，必须准确清楚地把这些所涉及的解题步骤记录下来，或提供交流，或编写成程序以供计算机来执行。

常用的描述算法的方式有自然语言、数学语言、流程图、伪语言和程序设计语言等。无论采用哪一种方式，都必须能够精确地描述计算过程。一般而言，描述算法最合适的语言是介于自然语言和程序设计语言之间的伪语言，它的控制结构往往非常类似于 Pascal、C 等程序语言，但其中可使用任何表达能力强的方法，使算法表达更加清晰和简捷，而不至于陷入具体的程序语言的某些细节。我们将采用这种接近于自然语言，并与 C 语言的控制结构又非常类似的伪语言（以下简称伪语言）来描述算法。同时考虑到读者易于上机验证算法和提高读者的编程能力，本书在给出算法的同时基本上也对应地给出了 C/C++语言的描述。由于上机环境设定在 Visual C++ 6.0 开发平台上，因此我们也使用了 C++ 的一些功能，如单行注释（//...）、引用（&）、存储空间的申请（new）与释放（delete），以及输入（cin）和输出（cout）等功能，这样可以避免 C 语言的繁琐和不便之处。

下面给出伪语言和要使用到的 C++ 对 C 语言的部分扩充功能的概要说明。

一、用伪语言描述算法

1. 算法的总体轮廓

一个算法应由算法标题、算法内容体、结束标记三部分组成。格式如下：

算法 <算法编号> <中文书写的算法名>

<英文书写的算法名>（<参数表>）

1. ----------
2. ----------
3. ----------

（1）----------

（2）----------
　　i ）----------
　　　ⓐ ----------
　　　ⓑ ----------
　　　ⓒ ----------
　　　⋮
　　ii ）----------
　　iii ）----------
（3）----------
　　⋮
4. ----------
5. [算法结束] ▉

① 上面给出的算法总体轮廓描述中，头两行为算法标题，算法标题应写在一行的中间，算法名应反映出该算法所能完成的主要功能。如果它不被其他算法所调用，算法标题的第二行可以省略。

② 后面的各行共同组成了算法体，算法体的书写采用层次结构，并按上面的规定，分层的标明，书写时同一层按列对齐。

③ 算法右下角的黑方块记号“▉”为结束标志，表示整个算法的结束。

2. 算法使用的主要语句

（1）注释。

算法中的任何位置必要时可加入汉字书写的注释，说明其功能，注释的形式是用方括号括起来的作为解释说明用的汉字串，即

[汉字串]

（2）赋值语句。

格式：变量名 ← 表达式

例如：X ← 500

　　A[i] ← i+3

（3）条件语句。

条件语句可以有如下两种格式：

格式 1：若　　条件
　　则　　语句 1
　　否则　语句 2

格式 2：若　　条件
　　则　　语句

（4）循环语句。

ⓐ for 语句

格式：循环 <循环变量>步长为<表达式 1>，从<表示式 2>到<表达式 3>，反复执行
　　循环体