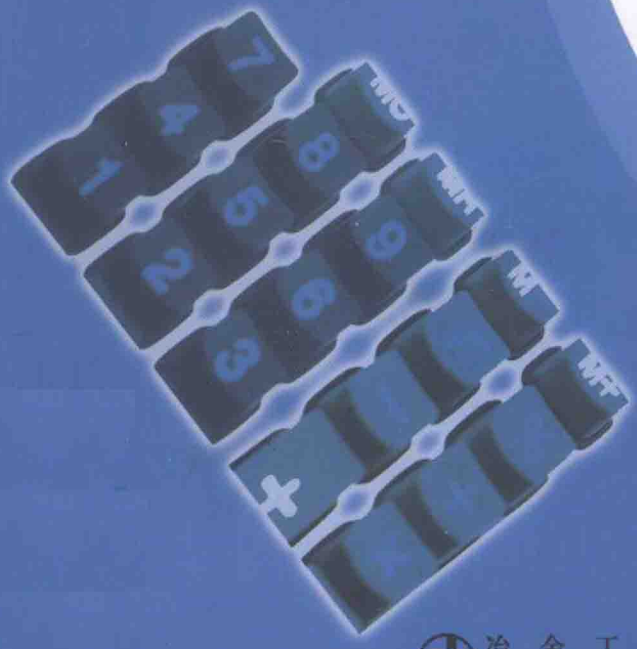
普通高等教育“十二五”规划教材

PUTONG GAODENG JIAOYU "12·5" GUIHUA JIAOCAI

数据结构

主编◎李唯



冶金工业出版社
Metallurgical Industry Press

普通高等教育"十二五"规划教材

数据结构

主　编　李　唯
副主编　贾海龙　余恒芳　余　超　王艳波
编　委　虞　沧　赵丙秀　肖　英　马艳春

北　京
冶金工业出版社
2013

内容简介

全书分为9章，第1章为绪论，介绍数据结构的基本概念；第2章为线性表，介绍线性表的两种存储结构和基本运算算法的实现；第3章为栈和队列，介绍栈和队列的概念与操作；第4章为串，介绍串的概念与应用；第5章为数组，介绍数组的概念与相关算法的实现；第6章为树和二叉树，介绍树和二叉树的概念与各种算法的实现；第7章为图，介绍图的概念和图的各种算法的实现；第8章为查找，介绍各种查找算法的实现；第9章为排序，介绍各种排序算法的实现。

本书可作为高等教育，大专院校计算机及其相关专业数据结构课程的教材和参考书，以及其他程序类课程的辅导教材，也可作为工程技术人员和自学计算机知识人员的参考资料。

图书在版编目(CIP)数据

数据结构/李唯主编. —北京：冶金工业出版社，2013.7

普通高等教育“十二五”规划教材

ISBN 978-7-5024-6356-4

Ⅰ.①数… Ⅱ.①李… 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139210号

出 版 人　谭学余

地　　址　北京北河沿大街嵩祝院北巷39号，邮编100009

电　　话　(010)64027926　电子信箱　yjcbs@cnmip.com.cn

ISBN 978-7-5024-6356-4

冶金工业出版社出版发行；各地新华书店经销；北京明兴印务有限公司印刷

2013年7月第1版，2013年7月第1次印刷

787mm×1092mm　1/16；16印张；340千字；224页

32.00元

冶金工业出版社投稿电话：(010)64027932　投稿信箱：tougao@cnmip.com.cn

冶金工业出版社发行部　电话：(010)64044283　传真：(010)64027893

冶金书店　地址：北京东四西大街46号(100010)　电话：(010)65289081(兼传真)

前　言

本书主要讲授数据的各种组织方式以及建立在这些结构之上的各种运算，是计算机专业和信息类专业的基础课和核心课程。数据结构总结了程序设计的常用方法和常用技巧，为计算机语言程序设计提供了指导。数据结构是一门集理论和实践于一体的课程，通过该课程的学习可以提高学生的程序设计水平和能力。

本书系统地介绍了各种常用的数据结构的基本概念、基本原理和基本方法以及相关的算法设计，结合数据结构课程中的重要知识点提供丰富的典型应用实例。全书采用面向对象的JAVA 语言对算法进行描述，将传统的数据结构的内容与面向对象的思想和技术完全融合，结构清晰，条理清楚、重点突出，表达通俗易懂并注重理论与实践相结合。

全书分为 9 章，第 1 章为绪论，介绍数据结构的基本概念；第 2 章为线性表，介绍线性表的两种存储结构和基本运算算法的实现；第 3 章为栈和队列，介绍栈和队列的概念与操作；第 4 章为串，介绍串的概念与应用；第 5 章为数组，介绍数组的概念与相关算法的实现；第 6 章为树和二叉树，介绍树和二叉树的概念与各种算法的实现；第 7 章为图，介绍图的概念和图的各种算法的实现；第 8 章为查找，介绍各种查找算法的实现；第 9 章为排序，介绍各种排序算法的实现。

数据结构课程具有概念多、算法灵活和抽象性强等特点，书中附有大量的图表、程序，使读者能直观地理解问题，同时每章有学习要点、习题，便于教学和自学。全书给出的所有算法和程序采用JAVA 语言描述。全书强化将理论知识应用于解决实际问题能力的培养，结合数据结构课程中的重要知识点提供丰富的应用实例。

本书由李唯担任主编，贾海龙、余恒芳、余超和王艳波担任副主编，参与本书编写的还有虞沧、赵丙秀、肖英、马艳春，全书由李唯统稿。对于在编写本书中提供了帮助和支持的人，在此表示衷心的感谢。

由于时间和编者水平有限，书中若有不妥之处，恳请广大读者批评指正。

编者

2013 年 4 月

目 录

第1章 绪 论

教学要求

(1)了解:什么是数据结构,什么是算法和算法分析。

(2)理解:基本概念和术语,抽象数据类型的表示与实现。

(3)掌握:算法的时间分析和空间分析的技术。

1.1 什么是数据结构

1.1.1 数据结构的作用

自1946年第一台计算机问世以来,计算机产业飞速发展,其应用范围迅速扩展。软件设计是计算机学科各个领域的核心,软件设计时要考虑的首要问题是数据的表示、组织和处理方法。数据结构设计和算法设计是软件设计的核心。随着应用问题的不断复杂,导致信息量剧增与信息范围拓宽,许多系统程序和应用程序的规模很大,结构又相当复杂。因此必须分析待处理问题中的对象的特征及各对象之间存在的关系,这就是数据结构这门课所要研究的问题。

为了编写出一个好的程序,必须学会分析待处理对象的特性以及各处理对象之间的关系。设计系统时先分析问题,建立数学模型,然后从方法论角度对给定的问题提出解决办法,最后选择某种程序设计语言编写程序上机求解。

编写解决实际问题的程序的一般过程:

(1)如何用数据形式描述问题,即由问题抽象出一个适当的数学模型。

(2)问题所涉及的数据量大小及数据之间的关系。

(3)如何在计算机中存储数据及体现数据之间的关系。

(4)处理问题时需要对数据作何种运算。

(5)所编写的程序的性能是否良好。

上面所列举的问题基本上由数据结构回答。

通过数据结构的学习,要学会分析研究计算机加工的数据对象的特性,以便在解决实际问题的过程中能灵活选择适当的逻辑结构、存储结构及其相应的算法。包括如下几个方面:

(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,也称为数据的物理结构。

(3)施加在该数据上的操作,即数据的运算。

1.1.2 数据结构的例子

计算机的应用已不再局限于科学计算,而更多地用于控制、管理及数据处理等非数值计算的处理工作。与此相应,计算机加工处理的对象由纯粹的数值发展到字符、表格和图象等各种具有一定结构的数据,下面是应用到数据结构的几个典型例子。

1.1.2.1 电话号码查询系统

设有一个电话号码薄,它记录了 N 个人的名字和其相应的电话号码,假定按如下形式安排:$(a_1, b_1), (a_2, b_2), \cdots (a_n, b_n)$,其中 $a_i, b_i (i=1,2\cdots n)$ 分别表示某人的名字和电话号码。这是一种典型的表格问题,见表 1-1 所示,数据与数据成简单的一对一的线性关系。

表 1-1 线性表结构

姓名	电话号码
张三	13511111111
李四	13688888888
…	…

1.1.2.2 磁盘目录文件系统

磁盘根目录下有很多子目录及文件,每个子目录里又可以包含多个子目录及文件,但每个子目录只有一个父目录,依此类推,如图 1-1 所示。数据与数据成一对多的关系,是一种典型的非线性关系结构——树形结构。

图 1-1 目录文件

1.1.2.3 交通网络图

从一个地方到另外一个地方可以有多条路径，这是一种典型的网状结构，数据与数据成多对多的关系，是一种非线性关系结构，见图 1-2。

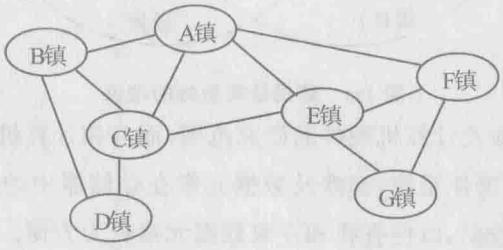


图 1-2 网状结构

解决数值问题的理论方法一般不适于解决非数值问题，需要探索新的方法和技术来解决非数值问题。数据结构就是为研究和解决这些非数值问题而提出的理论和方法，是解决这些问题的重要基础知识。数据结构是一门研究非数值计算的程序设计问题中计算机的操作对象以及它们之间的关系和操作等的学科。

1.1.3 数据结构的发展

数据结构作为一门独立的课程在国外是从 1968 年才开始设立的。当时数据结构几乎和图论，特别是和表、树的理论为同义语。随后数据结构这个概念被扩充到包括网络、集合代数论、格、关系等方面。然而，由于数据必须在计算机中进行处理，因此，不仅考虑数据本身的数学性质，而且还必须考虑数据的存储结构，这进一步扩大了数据结构的内容。

1968 年美国唐·欧·克努特教授开创了数据结构的最初体系，他所著的《计算机程序设计技巧》第一卷是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作。从 60 年代末到 70 年代初，出现了大型程序，软件也相对独立，结构化程序设计成为程序设计方法学的主要内容，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。从 70 年代中期到 80 年代初，各种版本的数据结构著作相继出现。数据结构和其他学科的关系如图 1-3 所示。

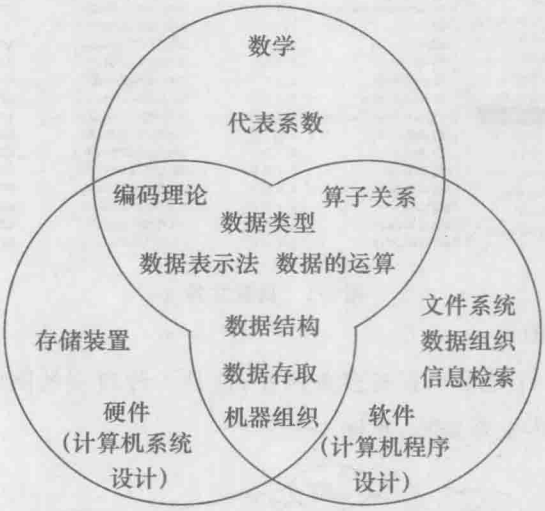


图 1-3　数据结构所处的地位

数据结构的研究不仅涉及计算机硬件的研究范围，而且和计算机软件的研究有着更密切的关系，无论是编译程序还是操作系统，都涉及数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据，以便查找和存取数据元素更为方便。

数据结构的发展并未终结，一方面，面向各专门领域中特殊问题的数据结构得到研究和发展，如多维图形数据结构等。另一方面从抽象数据类型的观点来讨论数据结构，已成为一种新的趋势，越来越被人们所重视。

1.2　数据的逻辑结构和物理结构

1.2.1　数据的逻辑结构

1.2.1.1　概念和术语

在本节中先对一些概念和术语赋以确定的含义。

(1)数据(Data)：是对客观事物的符号表示，是载荷信息的物理符号，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，一个利用数值分析方法解代数方程的程序，其处理对象是整数和实数；一个编译程序或文字处理程序的处理对象是字符串。因此，对计算机科学而言，数据的含义极为广泛，如图象、声音等都可以通过编码而归之于数据的范畴。

(2)数据元素(Data Element)：是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。例如，前面第二个例子中的“树”中的一个目录，第三个例子中的“图”中的一个圆圈都被称为一个数据元素。有时，一个数据元素可由若干个数据项(Data Item)组成，例如，第一个例子中每个人的通信信息为一个数据元素，而通信信息中的每一项(如姓名、电话号码等)为一个数据项，数据项是数据的不可分割的最小单位。

(3)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，整数数据对象是集合 N={0，±1，±2，…}，字母字符数据对象是集合 C={‘A’，‘B’，…，‘Z’}。

(4)数据结构(Data Structure) 是相互之间存在一种或多种特定关系的数据元素的集合。从前面的三个例子可以看到，在任何问题中，数据元素都不是孤立存在的，而是在它们之间存在着某种关系，这种数据元素相互之间的关系称为结构(Structure)。根据数据元素之间关系的不同特性，通常有下列四类基本结构：

1)集合：

结构中的数据元素之间除了同属于一个集合的关系外，别无其他关系，如图 1-4 所示。

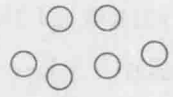


图 1-4 集合

2)线性结构：

结构中的数据元素之间存在一个对一个的关系，特点是开始结点和终端结点都是唯一的，除了开始结点和终端结点以外，其余结点都有且仅有一个前驱结点，有且仅有一个后继结点，如图 1-5 所示。顺序表就是典型的线性结构。

图 1-5 线性结构

3)树形结构：

结构中的数据元素之间存在一个对多个的关系。结点之间关系是一对多。特点是开始结点唯一，终端结点不唯一。除终端结点以外，每个结点有一个或多个后续结点；除开始结点外，每个结点有且仅有一个前驱结点，如图 1-6 所示。

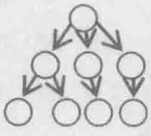


图 1-6 树形结构

4)图状结构：

结构中的数据元素之间存在多个对多个的关系。结点之间关系：多对多。特点是没有开始结点和终端结点，所有结点都可能有多个前驱结点和多个后继结点。

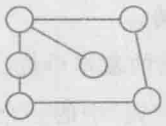


图 1-7　图状结构

1.2.1.2　数据的逻辑结构

上述数据结构的定义仅是对操作对象的一种数学描述，是从操作对象抽象出来的数学模型。结构定义中的关系描述的是数据元素之间的逻辑关系，因此又称为数据的逻辑结构。常就将数据的逻辑结构简称为数据结构。

为了更确切地描述一种数据结构，通常采用二元组表示：

$$B=(K,R)$$

(1)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

(2)$K=\{k_i \mid 1\leqslant i\leqslant n, n\geqslant 0\}$。

(3)$R=\{r_j \mid 1\leqslant j\leqslant m, m\geqslant 0\}$。

式中　• k_i——表示集合 K 中的第 i 个结点或数据元素。

• n——K—中结点的个数，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——集合 R 中的第 j 个二元关系(后面均简称关系)。

• m——R 中关系的个数，特别地，若 m=0，则 R 是一个空集，表明集合 K 中的元结点间不存在任何关系，彼此是独立的。

序偶<x,y>(x,y∈K)，x 为第一结点，y 为第二结点。x 为 y 的直接前驱结点，y 为 x 的直接后继结点。若某个结点没有前驱结点，则称该结点为开始结点；若某个结点没有后继结点，则称该结点为终端结点。

注意：<x,y>表示有向关系，(x,y)表示无向关系。

例如有如下数据即一个矩阵：

$$\begin{bmatrix} 3 & 8 & 5 & 3 \\ 9 & 3 & 12 & 6 \\ 7 & 11 & 25 & 1 \end{bmatrix}$$

对应的二元组表示为 B=(K,R)，其中：

$$K=\{3,8,5,3,9,3,12,6,7,11,25,1\}$$

$$R=\{r_1,r_2\}$$

其中，r_1 表示行关系，r_2 表示列关系。

$r_1=\{<3,8>,<8,5>,<5,3>,<9,3>,<3,12>,<12,6>,<7,11>,<11,25>,<25,1>\}$

$r_2=\{<3,9>,<9,7>,<8,3>,<3,11>,<5,12>,<12,25>,<3,6>,<6,1>\}$

数据的逻辑结构主要分为线性结构和非线性结构，数据逻辑结构层次关系如图 1-8 所示。

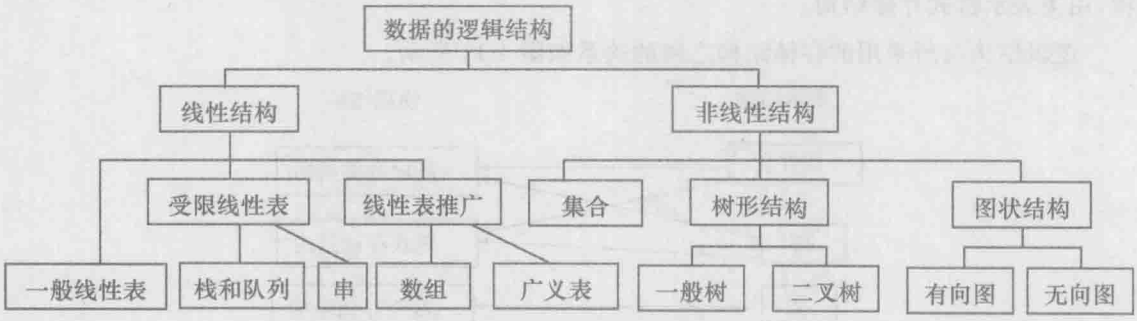


图 1-8 数据逻辑结构层次关系图

1.2.2 数据的物理结构

讨论数据结构的目的是为了在计算机中实现对它的操作，因此还需研究其如何在计算机中表示。数据结构在计算机中的表示称为数据的物理结构，又称为存储结构。在计算机中表示信息的最小单位是二进制数位(bit)。在计算机中，可以用一个由若干位组合起来形成的一个位串表示一个数据元素，通常称这个位串为元素(Element)或结点(Node)。当数据元素由若干数据项组成时，位于串中对应于各个数据项的子位串称为数据域(Data Field)。因此，元素或结点可看成是数据元素在计算机中的映象。

数据元素之间的关系在计算机中有两种不同的表示方法：顺序映象和非顺序映象，并由此得到两种不同的存储结构：顺序存储结构和链式存储结构。顺序存储结构的特点是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系，如图 1-9 所示。

序号	data
1	A
2	B
3	C
4	D

图 1-9 顺序存储结构

链式存储结构通过在结点域增加对象引用来找到下一个结点，逻辑上相邻的结点在物理上不用相邻，如图 1-10 所示。

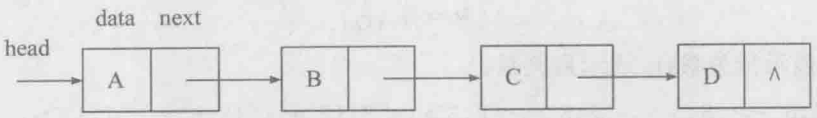


图 1-10 链式存储结构

数据的逻辑结构和物理结构是密不可分的两个方面，一个算法的设计取决于所选定的逻辑结构，而算法的实现依赖于所采用的存储结构。在 JAVA 语言中，用一维数组表示顺序存储结构，用类表示链式存储结构。

逻辑结构与所采用的存储结构之间的关系如图 1-11 所示。

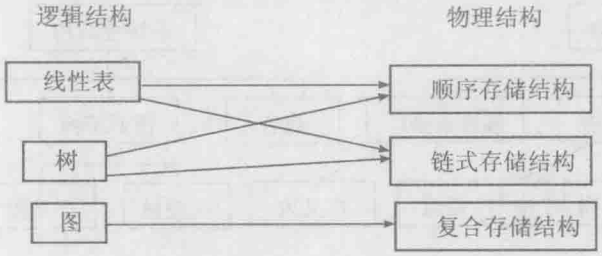


图 1-11 逻辑结构与存储结构关系图

1.2.3 数据类型

如何描述存储结构呢，存储结构涉及数据元素及其关系在存储器中的物理位置，现在不直接以内存地址来描述存储结构，可以借用高级程序语言中提供的数据类型来描述。

数据类型(Data Type)是和数据结构密切相关的一个概念，它最早出现在高级程序语言中，用以刻画操作对象的特性。在用高级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。类型明显或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些值上允许进行的操作。因此数据类型是一个值的集合和定义在这个值集上的一组操作的总称。例如 JAVA 语言中的整型变量，其值集为某个区间上的整数，定义在其上的操作为：加、减、乘、除和取模等算术运算。按值的不同特性，高级程序语言中的数据类型可分为两类：一类是非结构的原子类型，原子类型的值是不可分解的。如 JAVA 语言中的基本类型，整型、实型、字符型和布尔类型等。另一类是结构类型。结构类型的值是由若干成分按某种结构组成的，因此是可以分解的，并且它的成分可以是非结构的，也可以是结构的。例如数组的值由若干分量组成，每个分量可以是整数，也可以是数组等。在某种意义上，数据结构可以看成是一组具有相同结构的值，则结构类型可以看成由一种数据结构和定义在其上的一组操作组成。

在计算机中，数据类型的概念并非局限于高级语言中，每个处理器都提供了一组原子类型或结构类型。例如一个计算机硬件系统通常含有位、字节、字等原子类型，它们的操作通过计算

机设计的一套指令系统直接由电路系统完成，而高级程序语言提供的数据类型，其操作需通过编译器或解释器转化成低层即汇编语言或机器语言的数据类型来实现。引入数据类型，从硬件的角度看，是作为解释计算机内存中信息含义的一种手段，而对使用数据类型的用户来说，实现了信息的隐蔽，即将一切用户不必了解的细节都封装在类型中。例如，用户在使用整数类型时，既不需要了解整数在计算机内部是如何表示的，也不需要知道其操作是如何实现的。如两整数求和，程序设计者注重的仅仅是其数学上求和的抽象特性，而不是其硬件的位操作如何进行。

抽象数据类型(Abstract Data Type，ADT)是指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部使用。

抽象数据类型和数据类型实质上是一个概念，例如各个计算机都拥有的整数类型是一个抽象数据类型，尽管它们在不同处理器上实现的方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，抽象的意义在于数据类型的数学抽象特性：

抽象数据类型＝数据元素集合＋抽象运算

另一方面抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的数据类型，还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上，定义一组数据和施于这些数据上的一组操作，并在模块内部给出这些数据的表示及其操作的细节，而在模块外部使用的只是抽象的数据和抽象的操作。显然，所定义数据类型的抽象层次越高，含有该抽象数据类型的软件模块的复用程度也就越高。一个含抽象数据类型的软件模块通常应包含定义、表示和实现三个部分。

ADT 的一般定义形式是：

ADT ＜抽象数据类型名＞{

数据对象：＜数据对象的定义＞

数据关系：＜数据关系的定义＞

基本操作：＜基本操作的定义＞

}

其中数据对象和数据关系的定义用伪码描述。基本操作的定义是：

＜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＞

(1)初始条件：描述操作执行之前数据结构和参数应满足的条件；若不满足，则操作失败，返回相应的出错信息。

(2)操作结果:描述操作正常完成之后,数据结构的变化状况和 应返回的结果。

例如,抽象数据类型复数的定义:

ADT Complex {

数据对象:D={e1,e2|e1,e2 均为实数}

数据关系:R1={<e1,e2>| e1 是复数的实数部分,e2 是复数的虚数部分}

基本操作:

AssignComplex(&Z,v1,v2):构造复数 Z。

DestroyComplex(&Z):复数 Z 被销毁。

GetReal(Z,&real):返回复数 Z 的实部值。

GetImag(Z,&Imag):返回复数 Z 的虚部值。

Add(z1,z2,&sum):返回两个复数 z1,z2 的和。

}

如前所述,抽象数据类型的定义由一个值域和定义在该值域上的一组操作组成。若按其值的不同特性,可细分为下列三种类型:

(1)原子类型(Atomic Data Type)属于原子类型的变量的值是不可分解的。这类抽象数据类型较少,因为一般情况下,已有的固有数据类型足以满足需求。但有时也有必要定义新的原子数据类型,例如数位为 100 的整数。

(2)固定聚合类型(Fixed-aggregate Data Type)属于该类型的变量,其值由确定数目的成分按某种结构组成。例如复数是由两个实数依确定的次序关系构成。

(3)可变聚合类型(Variable-Aggregate Data Type)和固定聚合类型相比较,构成可变聚合类型"值'的成分的数目不确定。例如,可定义一个"有序整数序列"的抽象数据类型,其中序列的长度是可变的。

显然,后两种类型可统称为结构类型。和数据结构的形式定义相对应,抽象数据类型可用三元组表示(D,S,P),其中,D 是数据对象,S 是 D 上的关系集,P 是对 D 的基本操作集。

1.3 算法和算法分析

1.3.1 算法简介

1.3.1.1 算法的特点

曾获得图灵奖的著名计算机科学家 D. Knuth(克努特)对算法做过一个为学术界广泛接收

的定义：一个算法(Algorithm)，就是一个有穷规则的结合，其规则确定了一个解决某一特定类型问题的操作序列。一个算法具有下面五个重要特性：

(1)有穷性。对任何合法的输入值，一个算法必须总是在执行有穷步之后结束，且每一步都可在有穷时间内完成。

(2)确定性。算法中每一条指令必须有确切的含义，理解时不会产生二义性。并且在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。

(3)可行性。一个算法是能行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

(4)输入。一个算法有零个或多个的输入，这些输入取自于特定的对象的集合。

(5)输出。一个算法有一个或多个的输出。这些输出是同输入有某个特定关系的量。

一个算法可以用多种方法描述，主要有：使用自然语言描述；使用形式语言描述；使用计算机程序设计语言描述。

算法和程序是两个不同的概念，一个计算机程序是对一个算法使用某种程序设计语言的具体实现。算法用抽象的语言描述解决特定问题的每一步的操作。程序是计算机能理解和执行的指令序列，一个程序实现一个算法。算法的执行是有穷的，而程序的执行可以是无限的。例如操作系统程序，只要系统不受破坏，操作系统的运行将不会停止。

在本书中使用JAVA语言来描述算法。

1.3.1.2 算法设计的要求

通常设计一个好的算法应考虑达到以下目标：

(1)正确性(Correctness)：算法应当满足具体问题的需求。通常一个大型问题的需求，要以特定的规格说明方式给出，而一个实习问题或练习题，往往就不那么严格，目前多数是用自然语言描述需求，它至少应当包括对于输入、输出和加工处理等的明确的无歧义性的描述。设计或选择的算法应当能正确地反映这种需求，否则，算法的正确与否的衡量准则就不存在了。

“正确”一词的含义在通常的用法中有很大差别，可分为以下四个层次：1)程序不含语法错误。2)程序对于几组输入数据能够得出满足规格说明要求的结果。3)程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据能够得出满足规格说明要求的结果。4)程序对于一切合法的输入数据都能产生满足规格说明要求的结果。显然，达到第4)层意义的正确是极为困难的，所有不同输入数据的数量大得惊人，逐一验证的方法是不现实的。对于大型软件需要进行专业测试，而一般情况下，通常以第3)层意义的正确作为衡量一个程序是否合格的标准。

(2)可读性(Readability)：算法首先是为了人的阅读与交流，其次才是机器执行。可读性好有助于人对算法的理解，并且晦涩难懂的程序易于隐藏较多错误难以调试和修改。

(3)健壮性(Robustness)：当输入数据非法时，算法也能适当地作出反应或进行处理而不会