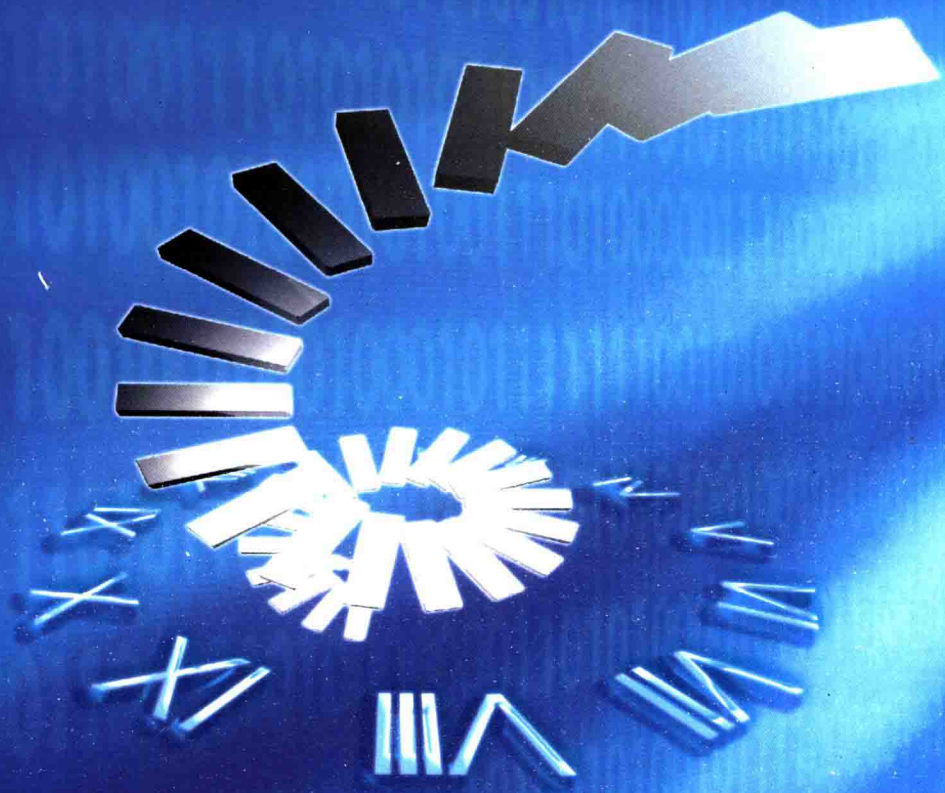


数据结构

（C语言版）

孟祥瑞 汤文兵
胡胜利 葛 斌 编著

华东理工大学出版社
EAST CHINA UNIVERSITY OF SCIENCE AND TECHNOLOGY PRESS

数 据 结 构

(C 语言版)

孟祥瑞 汤文兵
胡胜利 葛 斌 编著

华东理工大学出版社

内 容 提 要

《数据结构》(C语言版)是为"数据结构"课程编写的教材,同时也可作为学习"数据结构及算法"课程的参考教材。

本书系统地介绍了各种常用的数据结构和排序、查找的各种算法,阐述了各种数据结构内在的逻辑关系、存储表示和运算操作。本书概念表达严谨,注重理论与实践的结合,内容丰富,通俗易懂,既便于教学,又可用于自学。

本书可作为普通高等院校计算机类专业的教材,也可作为信息类相关专业的本专科教材。同时对于从事计算机工程与应用工作的科技工作者,本书也是一本实用的参考手册。

图书在版编目(CIP)数据

数据结构(C语言版)/(孟祥瑞、汤文兵、胡胜利、葛斌 编著.
—上海:华东理工大学出版社,2004.7
ISBN 7-5628-1553-4

Ⅰ.数... Ⅱ.孟... Ⅲ.①数据结构-高等学校-教材
②C语言-程序设计-高等学校-教材 Ⅳ.①TP311.12
②TP312

中国版本图书馆CIP数据核字(2004)第051002号

数 据 结 构
(C语言版)
孟祥瑞 汤文兵 胡胜利 葛 斌 编著

出版	华东理工大学出版社	开本	787×1092 1/16
社址	上海市梅陇路130号	印张	15.25
邮编	200237 电话(021)64250306	字数	357千字
网址	www.hdlgpress.com.cn	版次	2004年7月第1版
发行	新华书店上海发行所	印次	2004年7月第1次
印刷	上海复旦四维印刷有限公司	印数	1-4050册

ISBN 7-5628-1553-4/TP·127 定价:24.00元

前　言

“数据结构”是公认的计算机学科的核心专业基础课程，它所讨论的知识内容和提倡的方法为计算机程序设计提供了重要的理论基础，对于软件工程的开发具有不可替代的作用。本书正是针对这一背景编写的教材。

本书共分为9章。第1章综述了数据结构中的一些基本概念，如数据、算法等。第2～4章分别介绍了几种基本的线性结构，即线性表、队列和串、多维数组、广义表。第5章和第6章讨论了常用的非线性结构，即树和图。第7章和第8章介绍了信息处理中常用的技术——排序和查找。第9章讨论了常用的文件结构。本书的每一章都配有一定的习题，供读者复习提高之用。

本书在内容表达上注重原理与实践的结合，配备大量解释详尽的例题和插图，以便读者能更好地对各种数据结构加深认识和理解。对于知识点和相关概念的叙述，本着从易到难的学习规律。对各种算法均首先讨论其设计思想和方法，然后对其逐步求精，给出完整的C语言描述，以使学生能抓住算法的本质和基本思想。在每章的开始列出了学习要点，以方便读者的学习。

本书力求深入浅出，通俗易懂。既可作为计算机类的本科教材，也可作为信息类相关专业的教材。本书教授学时可为50～80学时，同时应预留一定的上机时间。教师可根据学时及专业，酌情删去一些内容。在学习本书时，至少应掌握一门高级程序设计语言，如C语言。同时还应具备离散数学的相关知识，这样可更好地理解书中的某些内容。

本书由孟祥瑞、汤文兵、胡胜利和葛斌共同编写，由汤文兵最后定稿。在本书的写作过程中，陈鸣、朱晓娟、刘向举、徐辉、郭家荣、张洁、张柱等老师参加了相关程序的调试工作，在此，本书作者对所有参与者表示感谢，是他们的辛勤工作使这本书得以尽快与读者见面。

限于作者的水平和时间，本书难免会有谬误之处，还请各位老师、同学不吝赐教。

作者

2004年4月

目　录

第 1 章　绪　　论

【学习要点】

1. 理解数据、数据对象、数据元素、数据类型、数据结构等基本概念，特别是数据的逻辑结构与物理(存储)结构间的关系及在这种结构上所定义的操作。

2. 了解算法的定义、算法的特性、算法的时间代价、算法的空间代价。

3. 掌握计算语句频度和估算算法时间复杂度的方法。

自 1946 年第一台计算机问世以来，计算机产业的发展突飞猛进，计算机功能与速度在不断提高。如今，计算机已深入到人类社会的各个领域。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也由纯粹的数值发展到字符、表格和图像等各种具有一定结构的数据，这就给程序设计带来一些新的问题。在处理这些数据之前，不仅要研究处理数据的特性还要研究处理数据间的相互关系及其对应的存储表示。因此解决非数据性问题的关键是利用这些特性和关系能设计出合适的数据结构。为了编写出一个“好”的程序，必须分析待处理的对象的特性以及各处理对象之间存在的关系，这就是“数据结构”这门学科形成和发展的背景。

1.1　数据结构的定义

在计算机发展的初期，人们面临的许多问题基本上可以用数学工具进行描述，那时使用计算机主要是处理数值计算问题，由于当时所涉及的运算对象是简单的整型、实型或布尔类型数据，所以程序设计者的主要精力是集中于程序设计的技巧上，而无须重视数据结构。但随着我们遇到的问题不断地增多并且更加复杂，许多方面是无法用数学工具或数学方程来描述的，因此我们必须选择其它的方法加以解决。这样就引入了数据结构的知识概念。

现在当我们使用计算机来解决一个具体问题时，一般需要经过下列几个步骤：首先分析实际问题，从中抽象出一个适当的数学模型，然后设计一个解决此数学模型的算法，最后编程、调试、测试，直至得到最终的解答。下面我们来看列举的几个例子。

例 1.1　学校人事信息检索系统。如果想获取学校教务处某个教职员工的有关信息，我们可以通过计算机来实现自动搜索。很显然，对应于我们选取的教职员工其姓名、职务、部门等应是一一对应的，各个教职员工的信息集合实际上是一种查找表的结构。因此，我们可以根据它们之间的这种关系建立数据结构，再按照某种算法编写出相关程序，就能够满足我们的需要选择相应的查询。由此，可以在学校人事信息检索系统中建立一张按教务处教职员工顺序号排列的教职员工信息表和分别按姓名、职务、部门(科室)顺序排列的索引表，如图 1.1 所示。

由这四张表构成的文件便是教务处教职员工信息检索的数学模型，这类数学模型可称为线性的数据结构，即对象之间通常存在着的是一种简单的线性关系。诸如此类的还有电

话自动查号系统、排课系统、仓库库存管理系统等。

职工号	姓名	性别	出生日期	职务	部门
0001	曾能自	男	1965.03.20	处长	教务处
0002	吕建莉	女	1966.04.16	科长	考务科
0003	许　晴	女	1967.12.09	科员	教材科
0004	张国宝	男	1969.07.28	科员	考务科
0005	陈小东	男	1970.09.14	科员	考务科
0006	程玉莲	女	1975.11.27	科员	教材科
0007	高　磊	男	1965.06.25	科长	教材科
0008	汪立斌	男	1970.10.19	主任	办公室
0009	时从政	男	1973.07.03	科员	考务科
0010	许　晴	女	1974.01.21	科员	办公室

（a）教职员工信息表

陈小东	5
程玉莲	6
高　磊	7
吕建莉	2
时从政	9
汪立斌	8
许　晴	3, 10
曾能自	1
张国宝	4

（b）姓名索引表

处长	1
科长	2，7
科员	3, 4, 5, 6, 9, 10

（c）职务索引表

教务处	1
考务科	2, 4, 5, 9
教材科	3, 6, 7
办公室	8, 10

（d）部门索引表

图 1.1　人事信息查询系统中的数据结构

例 1.2　计算机系统组成结构，如图 1.2 所示。

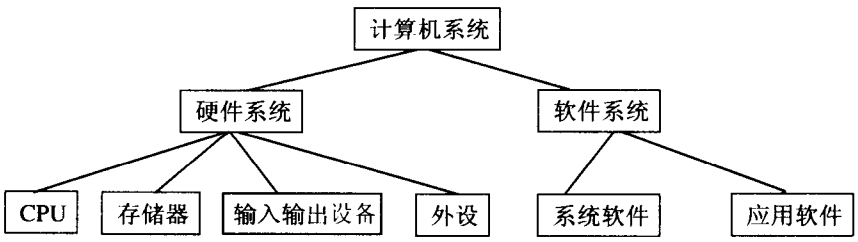


图 1.2　计算机系统组成结构图

由此可以直接看出，计算机系统是由硬件系统和软件系统这两大系统组成，硬件系统是由 CPU、存储器、输入输出设备、外设组成，而软件系统又是由系统软件、应用软件组成。如

果把它们视为数据元素，这些元素之间所呈现的是一种层次关系，从上到下按层进行展开形成一棵倒立的“树”，最上层是“树根”，依层向下射出“结点”和“树叶”。这种“树”型结构的模型在我们生活中接触得也比较多，比如国家行政区域规划、书籍目录等。

例 1.3 比赛编排问题。有 6 支球队进行足球比赛，我们分别用 $v_1, v_2, \cdots, v_6$ 表示这 6 支球队，它们之间的比赛情况也可以用一个称作图的数据结构来反映，有向图中的每个顶点表示一个球队，如果从顶点 v_i 到 v_j 之间存在有向边 $<v_i, v_j>$，则表示球队 i 战胜球队 j。如 v_1 队战胜 v_2 队，v_2 队战胜 v_3 队，v_3 队战胜 v_5 队，如此等等，这种胜负情况我们可以用图 1.3 表示。

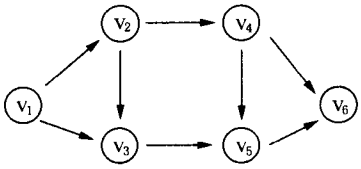


图 1.3 比赛胜负图

由此可以看出，用点和点与点之间的线所构成的图可反映实际生产和生活中的某些特定对象之间的特定关系。诸如此类有铁路交通图、教学编排等。

由以上三个例子可见，描述这类非数值计算问题的数学模型不再是数学方程，而是诸如表、树、图之类的数据结构。因此，可以说数据结构课程主要是研究非数值计算的程序设计问题中所出现的计算机操作对象以及它们之间的关系和操作的学科。

1.2 数据结构的发展及其目的

回顾数据结构的形成与发展可以帮助我们认识和理解数据结构的内容及其重要性。国外把“数据结构”作为一门独立的课程而设立是在 1968 年。在此之前是没有专门的“数据结构”课程的。但在当时(20 世纪 60 年代初期)，在“编译原理”和“操作系统”等课程中已经存在了有关“数据结构”的内容。稍后的一些时间里，有些国家的高校中开始设立有关课程，但当时的称谓是“表处理语言”而非“数据结构”。它的主要任务是分析当时的一些表处理系统，如 SLIP 系统(简单表处理语言)、IPL-V 系统(信息处理语言)、SNOBOL 系统(串处理语言)等。它们的数据对象的结构是表或者树。1968 年，美国的一些大学开始明确规定“数据结构”为一门课程，但对课程的内容范围并未明确界定。在随后的一些文章和书籍中，“数据结构”的范围被扩大到了图、集合、代数结构等方面，从而最终演变成了现在称为“数据结构”的内容，它与现在称为“数据结构”的某些内容混合在一起，总称为“数据结构”。

由于数据必须在计算机中进行处理，因此不能局限于数据本身的数学问题的研究，还必须考虑到数据的物理结构即数据在计算机中的存储结构，这进一步扩大了“数据结构”的内容范围。

1968 年，美国著名计算机科学家 D · E · 克努特教授所著的《The Art of Computer Programming》Volume I(《计算机程序设计技巧》第一卷)出版，对计算机的发展作出了重大贡献，在书中作者论证了任何语言都可以采用表处理语言那样的技术。书中系统全面地论

述了若干种数据的逻辑结构及物理结构。随后，数据的逻辑结构、存储结构及对应每种结构的操作被独立出来，形成了“数据结构”的主要内容。20 世纪 70 年代以后，由于人们在软件开发领域对数据结构的重要性的认识，数据结构开始越来越广泛地得到研究，各种论著相继出现。

近年来，由于数据库、情报检索系统的发展，“数据结构”课程中又增加了文件结构，特别是大型文件的组织等内容。

然而，“数据结构”还处于高速发展的时代。一方面，从抽象和具体的两种观点研究“数据结构”正在成为趋势；另一方面，计算机硬件系统及其它各学科的不断发展也必然对数据结构产生重大影响。

“数据结构”是一门涉及广泛的专业基础课，它与数学、计算机软件、计算机硬件之间的关系如图 1.4所示。

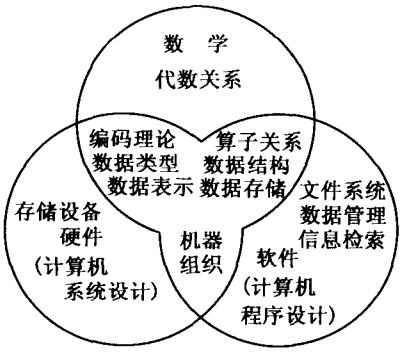


图 1.4 “数据结构”所处的地位

由图 1.4 可以看出，数据结构是介于数学、计算机软件和计算机硬件之间的一门计算机专业的核心课程，它是计算机程序设计、数据库、操作系统、编译原理及人工智能等的重要基础，广泛地应用于信息学、系统工程等各种领域。

学习数据结构的目的是为了了解计算机处理对象的特性，将实际问题中所涉及的处理对象在计算机中表示出来并对它们进行处理。与此同时，通过算法训练来提高学生的思维能力，通过程序设计的技能训练来促进学生的综合应用能力和专业素质的提高。

1.3 基本概念及术语

为了便于以后各章的学习，本节主要讲述数据结构中常用的基本概念和术语。

数据(Data)：数据是客观事物的符号表示，是对现实生活中的客观事物所采用的，能被计算机识别、存储并处理的形式描述的符号的集合，是计算机进行程序处理的必要的“原料”信息。它的形式是多种多样的。比如：表格处理软件中的各类表格信息，多媒体处理软件中的各种媒体信息包括音频、视频信息等等。总之，数据是通过编码的各种客观事物的信息。

数据元素(Data Element)：数据元素是数据的基本单位，在计算机程序中通常把它作为一个整体来处理。一个数据元素又可以由若干个初等数据项或组合项组成。初等数据项，简称数据项是数据不可分的最小单位。几个初等数据项又可以组成一个组合数据项。如图 1.5 所示教师基本情况调查表中的姓名、性别、年龄等都分别为一个数据项，而且是初等数据项。工资状况是组合数据项。

数据对象(Data Object)：数据对象是性质相同的数据元素组成的集合，是数据集合的一个子集。数据元素是数据对象的数据成员，也是数据集合的元素。如图 1.5 所示的教师基本情况调查表就是一个数据对象。正整数的数据对象是 $N=\{1,2,3,4,\cdots\}$ 等等。

姓 名	性别	年龄	籍 贯	工龄	工 资 状 况			
					基本工资	岗位津贴	职 补	其他
孔 军	男	35	合肥	11	645	127	234	2.7
李 梅	女	24	广州	1	395	97	157	2.7
钱安川	男	36	上海	13	645	127	285	2.7
张智明	男	30	蚌埠	7	535	114	210	0
周春蕾	女	27	杭州	3	465	97	157	2.7

图 1.5 教师基本情况调查表

上面的“教师基本情况调查表”就是一个数据对象，每位教师的数据占一行，为一个数据元素。每个数据元素由姓名、性别、年龄、籍贯、工龄和工资状况等数据项组成，其中，姓名、性别、年龄、籍贯、工龄为初等数据项，工资状况为组合数据项，由基本工资、岗位津贴、职补和其它四个初等数据项组成。

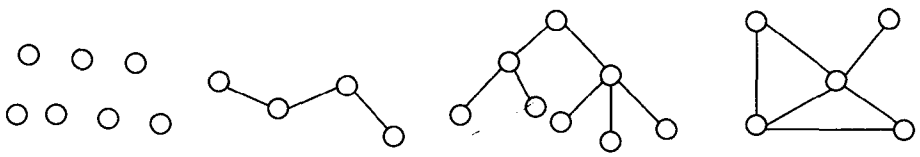
数据结构(Data Structure)：在任何实际问题中，各数据元素之间都不可能是孤立的，它们之间总是存在着这样或那样的相互关系。这种数据元素之间的相互关系就称为结构。根据数据元素之间关系的不同，可以把结构分成以下四种形式。

集合：在集合结构中各元素之间，除了“同属于某一个数据对象”的关系外，再别无其它的关系。如图 1.6(a)所示。

线性结构：线性结构中的各数据元素之间存在一一对应的关系，即一个元素最多只有一个前驱，一个后继。其原形是线性表，如图 1.6(b)所示。

树形结构：树形结构中各数据元素之间存在一对多的关系，即一个元素只有一个前驱，但是可能有多个后继。如图 1.6(c)所示。

网状结构或图状结构：在这种结构中对数据元素的前驱和后继不做任何的限制，它们之间存在着多对多的关系。如图 1.6(d)所示。



(a) 集合 (b) 线性结构 (c) 树形结构 (d) 图状结构

图 1.6 几种结构示意图

正因为对于任何数据对象中各数据元素之间都存在特定的关系，而这些存在着相互关系的数据元素的集合就是数据结构。更简单地说，数据结构就是带有结构的数据元素的集合。数据结构可以用二元组的形式来进行数学意义上的形式定义：

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

其中，D：是数据元素的有限集，即数据对象。

R：为数据对象 D 上所有数据元素之间的关系的有限集。

以上是一种数学意义上的定义，也是一种数学的描述。它是从操作对象中抽象出来的数学模型，仅仅描述了数据元素之间的逻辑关系，即数据的逻辑结构。如何在计算机中实现对数据结构的操作，还需要考虑如何在计算机中表示与存储它。在研究数据结构在计算机中的实现时，一般应该考虑以下三点：

数据集合中各数据元素之间的逻辑关系，即数据的逻辑结构。

数据元素之间的逻辑关系在计算机中的存储表示，即数据的存储结构或称为物理结构。

数据结构中各数据元素之间的运算，即数据元素之间的操作。

其中，数据元素之间的逻辑结构是根据实际问题的需要而选择设计的，因此是面向问题而与计算机本身无关的，或者说是不依赖于机器的。而数据元素的存储结构是指数据在计算机中的物理表示和存储的方式，涉及数据元素及其关系在存储器中的物理位置、机器响应的速度等方面因素，因而物理结构的设计是与计算机本身密切相关的。另一方面，数据的逻辑结构与物理结构又是密不可分的两个方面，对于任何一种算法的设计都取决于选定的数据的逻辑结构，而算法的实现则依赖于所采用的存储结构。

例如，对于线性表，除第一个元素外，其它元素只有一个前驱；除最后一个元素外，其它元素只有一个后继，这就是线性表的逻辑结构。至于它在计算机中是怎样表示和存储的，是用连续的存储单元存储还是用分散的存储单元存储而用指针来连接，这则是线性表实现的存储方式。另外，涉及到线性表的插入、删除、查询、排序等操作，就是数据结构的操作。

1.4 数据类型和抽象数据类型

数据类型(Data Type)是一个值的集合和定义在这个值集上的一组操作的总称。例如整型变量允许在计算机硬件中能表示最大正整数和最小负整数之间的所有整数值。在典型的 16 位计算机中整型变量一般取－32768 到＋32767 之间的整数值，而一个布尔型的变量仅允许取 true 和 false 这两个值。

每一种程序设计语言都有一组固有的或基本的数据类型，如 C 中的基本类型是 int、float、long int、double、char、enum 和指针。但很多现代的程序设计语言，例如 PASCAL、C、JAVA 等都允许定义新的类型，这样的类型仅仅是对基本类型范围的限制，例如在 1 与 10 之间的整型或基本类型的组合；由整型的学生号码、字符串类型的学生姓名和实型的平均学习成绩组成的记录等。

按“值”的不同特性，高级程序设计语言中的数据类型可分为两类。一类是每一个对象仅由单值构成的类型，称为原子类型。如整型、字符型，其值是不可分解的，程序设计语言提供的基本算术操作，如加、减、乘、除和数学函数大部分是在这种类型上进行的；另一类是每一个对象可由若干成分按某种结构构成的类型，称为结构类型，每个结构数据可以再分。如数组就是一种结构类型，它由固定个数的同一类型顺序排列而成，数组类型中每一个数组值包含有固定个数的同一类型数据，每个数据都可以通过下标运算符直接访问。记录也是一种结构类型，它由固定个数的不同(也可以相同)类型顺序排列而成，记录类型中的每一个记录值包含有固定个数的不同类型数据，每个数据也都可以直接访问。另外，字符串和文件也都是结构类型。由此可见，结构类型数据值可进一步分解为组成元素，即数据元素，并且它

的成分可以是非结构的也可以是结构的。

抽象数据类型(Abstract Data Type,简称 ADT)是一种数据类型及在这个类型上定义的一组合法的操作。抽象数据类型不仅包括数据类型的定义,而且为这个新类型说明了一个有效的操作集合。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,即不论抽象数据类型内部结构如何变化,只要它的数学特性不变,都不影响其外部的使用。因此,抽象数据类型和数据类型实质上是同一个概念。

除此之外,抽象数据类型的范畴更广。它不仅仅局限于高级语言中已实现了的数据类型,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,近代程序设计方法学指出,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一个相对独立的模块中,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实现细节,而在模块外部,只使用抽象数据和抽象操作。由此可见,该抽象类型定义的层次越高,含有该抽象数据类型的软件模块的复用率也就越高。

我们知道,数据结构是一个二元组,它的逻辑定义是结点集合以及结点间关系的集合。而在我们研究数据结构时更为重要的是要设计定义在其上的一组操作的算法、种类和数目不同,数据结构所起的作用也不同。这样,如果能把一个数据结构以及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。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组操作组成,而数据结构又包括数据元素及元素间的关系,因此抽象数据类型一般可以由元素、关系及操作三种要素来定义。抽象数据类型的特征是使用与实现相分离,实行封装和信息隐蔽。就是说,在抽象数据类型设计时,把类型的定义与其实现分离开来。

和数据结构的形式定义相对应,抽象数据类型可用以下三元组表示:

$$ADT=(D,S,P)$$

其中 D 是数据对象,用结点的有限集合表示;

S 是 D 上的关系集,用结点间序偶的集合表示;

P 是对 D 的基本操作集。其操作的定义格式为:

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>

例 1.4 构造三元组表

ADT Triplet

{ 对 象:$D=\{e_1,e_2,e_3 \mid e_1,e_2,e_3 \in ElemSet\}$

关 系:$R1=\{<e_1,e_2>,<e_2,e_3>\}$

操 作:InitTriplet(&T,v_1,v_2,v_3)

操作结果:构造三元组 T,元素 e_1,e_2,e_3 分别被赋以参数 v_1,v_2,v_3 的值

操 作:Get(T,i,&e)取表中元素

}

至于上述抽象类型如何具体实现,读者可以根据所学过的知识来思考一下,在这里就不赘述了。

1.5 算法描述

算法(Algorithm)是一个有穷规则(或语句、指令)的有序集合。通俗地说,就是计算机解题的过程。在这个过程中,无论是形成解题思路还是编写程序,都是在实施某种算法。前者是推理实现的算法,后者是操作实现的算法。作为一个完整的算法应具有以下几个重要的特性。

输入:一个算法有零个或者多个的输入,这些输入取自于某个特定的对象的集合。

输出:一个算法至少有一个输出,这些输出是同输入有着某些特定关系的量。没有输出的算法是没有意义的。

有穷性: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。

确定性:算法中的每条指令的含义都必须明确,无二义性。对相同的输入,必须有相同的执行结果。

可行性:算法中的每条指令的执行时间都是有限的。

为了解决特定的问题,我们可以借助各种工具来描述算法,通常有以下几种描述工具:自然语言、流程图、形式化语言、具体的程序设计语言等。一般来说,用自然语言来描述的算法,易于理解,但不直观,对于复杂的算法难以表达;用流程图来描述的算法比较直观,但移植性差;用形式化语言来描述的算法能方便地表达思想,但不能直接在机器上运行;用具体的程序设计语言来描述算法,编写的程序能直接运行,但受具体语言语法的限制。因此,在实际中我们要根据它们的各自特点灵活选择。

用 C 语言描述算法的一般形式如下:

```
void      函数名(<参数表>)
   {
   <语句组>
   }
函数类型     函数名(<参数表>)
   {
   <语句组>
   }
```

例如计算两数中的较大数,可以如下描述:

```
int bigger(int a,int b)
{if a>=b return (a);
 return (b);
}
void main()
{int a=3,b=4,c;
 c=bigger(a,b);
printf("The bigger is %d",c);
```