数据结构(C 语言)

(第二版)

苏德富　杨　柳　华　蓓
钟　诚　薛弘晔　编著

重庆大学出版社

内容提要

本书以通俗易懂的语言,按照“加强基础”、“学以致用”、“重点突出”和“少而精”的原则编写,主要介绍了三个知识单元:一是数据类型、数据结构的基本概念,线性表、堆栈和队列、树、图的逻辑结构和存储结构及其基本操作;二是算法分析的基本概念,算法的地位、性质和表示方法,以及算法在程序设计中的作用;三是排序、查找技术和文件的基本知识。书中许多算法采用流行的C程序语言作较为详尽的描述,学生只需根据自己的计算机系统的特点,对算法做少量的修改或不做修改即可上机实现。各章间均配置经过精心选择的习题供读者练习,以巩固、加深对课程内容的理解,检验学习效果。

本书可作为普通高校、成人高校计算机和相关专业以及各种培训班的教材,也可以作为工程技术人员学习“数据结构”的参考书。

图书在版编目(CIP)数据

数据结构:C语言/苏德富等编著. —重庆:重庆大学出版社,2003.5

计算机专科系列教材

ISBN 7-5624-1341-X

Ⅰ.数… Ⅱ.苏… Ⅲ.①数据结构—高等学校—教材②C语言—程序设计

Ⅳ.①TP311.12②TP312

中国版本图书馆CIP数据核字(2003)第034052号

数据结构(C语言)

(第二版)

苏德富 杨 柳 华 蓓 钟 诚 薛弘晔 编著

责任编辑:曾令维 版式设计:曾令维

责任校对:蓝安梅 责任印制:秦 梅

*

重庆大学出版社出版发行

出版人:张鸽盛

社址:重庆市沙坪坝正街174号重庆大学(A区)内

邮编:400044

电话:(023) 65102378 65105781

传真:(023) 65103686 65105565

网址:http://www.cqup.com.cn

邮箱:fxk@cqup.com.cn(市场营销部)

全国新华书店经销

重庆大学建大印刷厂印刷

*

开本:787×1092 1/16 印张:12.5 字数:312千

1997年6月第1版 2003年5月第2版 2003年5月第6次印刷

印数:26 001 - 30 000

ISBN 7-5624-1341-X/TP·119 定价:12.00元

序

面对知识爆炸,社会学家们大都开出了一个相同的药方:计算机。计算机也深孚众望,以其强大的功能,对人类做出了巨大的贡献,取得了叹观止矣的成就。自它 1946 年 2 月 14 日在美国费城诞生以来,至今已过“知天命”的年龄了。现在,计算机已是一个庞大的家庭。如果说,它的成员占据了世界的每一个角落和每一个部门也并不过分,甚至找不到这样一个文明人,他的生活不直接或间接与计算机有关。目前,全世界计算机的总量已达数亿台,而且,现在正以每年几千万台的速度增长。

作为计算机在信息传递方面的应用,计算机加上网络,被认为是和能源、交通同等重要的基础设施。这种设施对信息的传递起着异常重要的作用。西方发达国家和我们国家对此都非常重视。例如,美国的信息高速公路计划,全球通讯的“铱”计划,我国也开始实行一系列“金”字头的国民经济管理信息化计划。这些计划中唱主角的设备便是计算机。计算机在各个方面的应用不胜枚举,我们每个人都自觉不自觉地处于计算机包围中。

计算机对社会生产来说是一个产业大户,对每个现代人来说是一种工具,对学生们来说,它是一个庞大的知识系统。面对计算机知识的膨胀,面对计算机及其应用产业的膨胀,计算机各个层次的从业人员的需要也在不断膨胀,计算机知识的教育也遍及从小学生到研究生的各个层次。

为了适应计算机教学的需要,重庆大学出版社近几年出版了大量的计算机教学用书,这一套教材就是一套适应专科层次的系列教材。我们将会看到,这一套教材以系列、配套、适用对路,便于教师和学生选用。如果再仔细研究一下,将会发现它的一系列编写特色:

1 这些书的作者们是一些长期从事计算机教学和科研的教师,不少作者在以前都有大量计算机方面的著作出版。例如本系列书中的《Visual Fox Pro 中文版教程》的作者,十年前回国后最早将狐狸软件介绍到祖国大陆,这一本书已是他的第八本著作了。坚实的作者基础,是这套书成功的最根本的保证。

2 .计算机科学是发展速度惊人的科学,内容的先进性、新颖性、科学性是衡量计算机图书质量的重要标准,这一套书的作者们在这方面花了极大的功夫,力求让读者既掌握计算机的基础知识,又让读者了解最新的计算机信息。

3 .在内容的深度和知识结构上,从专科学生的培养目标出发,在理论上,从实际出发,满足本课程及后续课程的需要,而不刻意追求理论的深度。在知识结构上,考虑到全书结构的整

体优化，而不过分强调单本书的系统性。这样，在学过这一套系列教材后，学生们就可在浩瀚的计算机知识中，建立起清晰的轮廓，就会知道这些知识的前因后果，就会了解这些知识的前接后续。使学生们能在今后的工作实践中得心应手。

4 .计算机是实践性很强的课程，仅靠坐而论道是学习不了这些知识的。所以从课程整体设置来讲，包括有最基本的操作技能的教材。对单本书来说，在技术基础课和专业课中，都安排有一定的上机实习或实验，这样可使学生既具备一定的理论知识以利今后发展和深造，又掌握实际的工作技能胜任今后的实际工作。

编写一套系列教材，这是一个巨大的工程。这一套书的作者们，重庆大学出版社的领导和编辑们，都为此付出了辛勤的劳动。作为计算机工作者，以此序赞赏他们的耕耘，弘扬他们的成绩。

1997 年 6 月 15 日

再版前言

计算机的应用日益广泛,计算机网络系统及其互联技术对于促进国民经济的发展起着越来越大的作用。国际计算机互联网络(Internet 网)把世界各地的计算机连成一个整体,实现相互之间的数据通信和资源的高度共享。人类将步入信息社会,人们的思维方式、工作方式和生活方式都在发生着深刻的变化。计算机正逐步成为人们工作和生活的必备工具。因此,计算技术水平的高低将逐步成为人们衡量人才的重要尺度。人们学习计算技术的需求越来越大。

《数据结构》是计算机有关专业的一门核心课程,也是非计算机专业学生进一步学习计算机知识的必备基础。它包含三个知识单元:一是数据类型、数据结构的基本概念,链表、树、图的逻辑结构和存储结构及其基本操作;二是算法的基本概念、算法的地位和表示方法,以及算法在程序设计中的作用;三是排序、查找技术的基本算法和文件的基本知识。通过这些知识的学习,掌握常用的数据结构的特点、用途及算法,当面对实际问题时能够选择合理的数据结构,组织好数据,选择或设计好的算法,从而使得编制的程序更合理、更简洁,运行效率更高。

“加强基础”、“学以致用”、“少而精”是本书的编写原则。书中大多数算法除了用文字作简明的描述外,还用 C 程序设计语言作较为详尽的描述。学生只需根据自己的计算机系统的特点,对算法做少量的修改或不做修改即可上机实现。《数据结构》是一门专业基础课,它的概念和算法较抽象,要深入地了解它,需要做一定的实验和练习。所以,我们安排了较多的实验,学习时,一定要认真地去做。

全书共分 9 章。第 1 章介绍数据结构的基本知识,数据结构与算法之间的关系,以及算法分析的一般方法。第 2 章介绍线性表的概念,线性表的逻辑结构、物理存储结构及其操作算法,此外,还介绍了数组的有关知识。第 3 章重点介绍栈和队列的概念、物理存储结构和算法,并介绍了栈和队列的应用。第 4 章专门讨论字符串的概念、物理存储结构和处理字符串的算法。第 5 章介绍树的一般概念,重点介绍二叉树的概念、性质、物理存储结构和遍历算法,线索二叉树及其操作,并讨论了树、二叉树和森林之间的转换,最后介绍哈夫曼树及其应用。第 6 章介绍有向图和无向图的概念、物理存储结构和遍历算法,最小生成树及其应用,以及拓扑排序、最短路径和关键路径等内容。第 7 章介绍各种排序方法,包括直接插入和折半插入排序、希尔排序、冒泡排序、快速排序、堆排序、两路合并排序和基数排序。第 8 章介绍查找技术,重点介绍顺序查找、二分查

找、分块查找、二叉树查找、B 树查找和 HASH 查找。第 9 章介绍文件的基本知识。

本书第 1、7、9 章由苏德富编写，第 2、3、4 章由杨柳和钟诚编写，第 5、6、8 章由华蓓和薛弘晔编写。全部实验由杨柳编写。

由于时间紧迫，加之编者的水平和见识有限，对于书中存在的缺陷和错误，热切希望广大读者批评指正，以臻完善。

编　者

2003 年 5 月 3 日

目录

第 1 章 绪 论

数据结构是随着电子计算机的发展而发展起来的一门计算机课程。它目前已成为各计算机类专业的核心课程,许多非计算机专业也把它作为必修课或选修课。它是设计和实现编译程序、操作系统、数据库系统以及其他系统程序和应用程序的重要基础。本章将讨论什么是数据结构以及相关的基本概念。

1.1 数据结构的基本概念

数据是人们利用文字符号、数字符号及其他规定的符号对现实世界的事物及其活动所做的描述。凡是能被计算机输入、存储、处理和输出的一切信息都叫数据。因此,数字、字符、图像、图形、声音等都属于数据的范畴。例如,财务账本中的数量、单价、金额等是数值数据,而科目、摘要、凭证号、日期等则是字符数据。会计凭证是进行会计处理的原始依据。

数据元素是数据中相对独立的、不可分的基本数据单位。例如,一张名片由姓名、职务、职称、电话、邮编和地址等信息组成。其中姓名和职务等是名片这个实体(称为记录)的数据元素。而对于由许多名片作为记录组成的数据文件来说,名片记录则是名片文件的数据元素。

数据对象是性质相同的数据元素的集合。例如,整型数据对象是集合{0,±1,±2,±3,±4,…},而扑克牌上点数的数据对象是{2,3,4,…,J,Q,K,A}。

数据处理是指对数据进行录入、查找、修改、合并、删除、统计、计算和输出等操作的过程。对会计业务的数据处理,就是将会计凭证转换成日记账、总账、明细账,最终输出会计报表的过程。

信息处理是把数据处理成为有用的资料。例如,把会计数据加工成为能综合反映企业财务状况和经济活动指标的会计报表,以满足管理和决策的需要。

数据和信息是相对的。经过数据处理得到的信息,通过反馈又成为下一次处理的数据。例如收、付款凭证登账处理后,经过月末结账与转账,又产生了转账凭证作为下次账务处理的数据。数据和信息之间的关系如图 1.1 所示。

数据结构是数据及其相互之间具有某种特定关系的数据元素的集合。数据元素之间的相互关系称为结构。

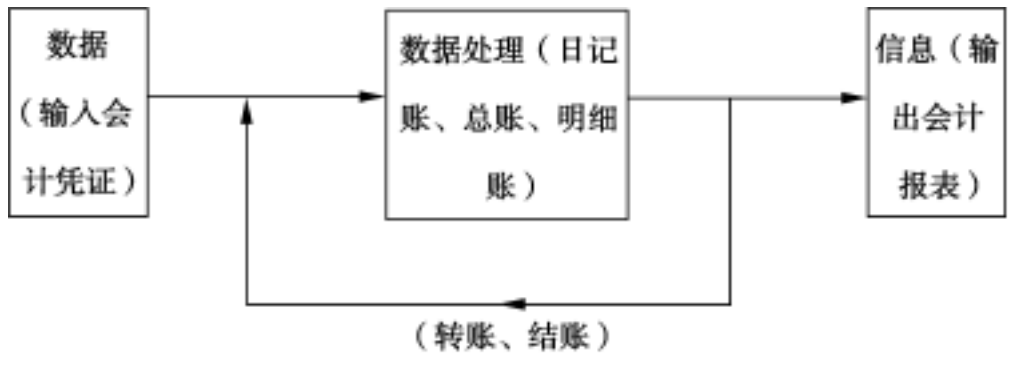

图 1 .1　数据与信息的关系

数据结构 DS 可用二元组作形式定义:

DS = (K, R)

这表示 DS 是一种数据结构,它由数据元素集合 K 和在 K 之上定义的一种二元关系的集合 R 所组成。其中:

$K = \{K_i \mid 0 \leq i \leq n, n \geq 0\}$

$R = \{R_j \mid 0 \leq j \leq m, m \geq 0\}$

K_i 表示第 i 个数据,n 为 DS 中数据元素的个数;R_j 表示数据元素之间的第 j 个关系,m 为在 K 之上关系的个数。

K 上的二元关系 R 是序偶的集合。对于 R 中的任一序偶 〈x, y〉 (x, y∈K),称 x 为序偶的第一元素(或 y 的前驱),称 y 为序偶的第二元素(或 x 的后继)。

数据结构还可以用图形直观地表示。图形中的每个结点对应于一个数据元素,两结点之间带箭头的连线(称为弧或有向边)对应于二元关系的一个序偶。其中序偶的第一元素为弧的起点(即弧尾),第二元素称为弧的终点(即弧头)。

下面用一个实际例子来说明数据结构的概念。例如,考虑一个生产蜂鸣器的厂家。众所周知,一种产品分为三类:成品、部件和零件。一个蜂鸣器由一个外壳、一片压电陶瓷、一块电路板和一条 5 英寸的双股铜线组成;而压电陶瓷片由一片不锈钢和一片陶瓷组成,一块电路板则由一块线路板、两个二极管、两个三极管和四个电阻组成。这个产品与成品、部件和零件之间的关系如图 1 .2 所示。

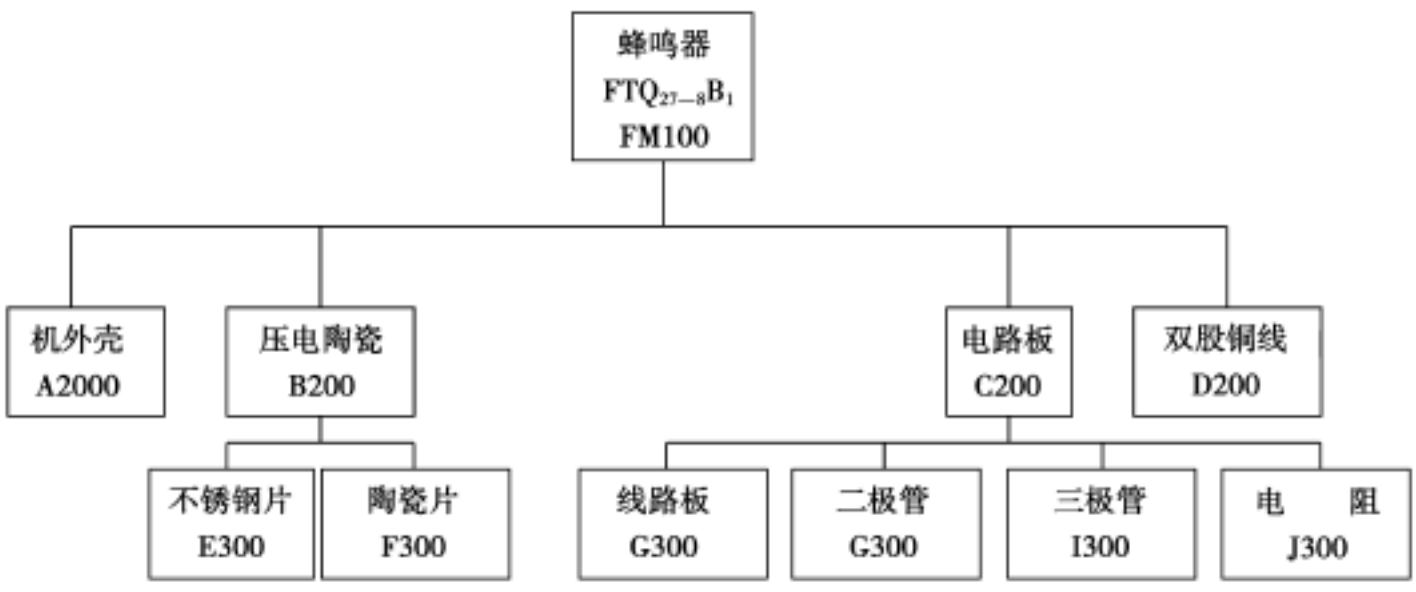

图 1 .2　蜂鸣器产品结构图

蜂鸣器所需零部件库存物料如表 1 .1 所示。

【例 1】　对蜂鸣器定义一种数据结构 LIST = (K, R),其中:

K = { FM100, A200, B200, C200, D200, E300, F300, G300, H300, I300, J300}

R = {〈FM100, H300〉,〈H300, I300〉,〈I300, B200〉,〈B200, G300〉,〈G300, C200〉,〈C200, J300〉,〈J300, E300〉,〈E300, A200〉,〈A200, F300〉,〈F300, D200〉}

相应地对蜂鸣器这一数据结构可表示为

FM100→H300→I300→B200→G300→C200→J300→E300→A200→F300→D200

不难看出,上述关系 R 反映零部件库存量从小到大的递增排列。

在 LIST 这一数据结构中,除第一个元素 FM100 和最后一个元素 D200 外,其他每个数据元素只有一个直接前驱元素且只有一个后继元素。数据元素之间的关系若是一对一的关系,则称这种关系为线性关系。具有这种特性的数据结构称为线性结构。

表 1.1　蜂鸣器零部件库存表

物料号	名称	库存量
FM100	蜂鸣器	10
A200	机外壳	440
B200	压电陶瓷片	150
C200	电路板	300
D200	双股铜线	1 500
E300	不锈钢片	400
F300	陶瓷片	460
G300	线路板	250
H300	二极管	80
I300	三极管	88
J300	电阻	330

【例 2】　定义另一种数据结构 TREE = (K, R),其中:

K = { FM100, A200, B200, C200, D200, E300, F300, G300, H300, I300, J300}

R = {〈FM100, A200〉,〈FM100, B200〉,〈FM100, C200〉,〈FM100, D200〉,〈B200, E300〉,〈B200, F300〉,〈C200, G300〉,〈C200, H300〉,〈C200, I300〉,〈C200, J300〉}

此时关系 R 所对应的图如图 1.2 所示,该结构反映了组装蜂鸣器时生产的工艺流程和零部件的组装过程,称它为产品结构树。

图 1.2 像一棵倒画的树,它的最上面的结点没有前驱只有后继结点,称这个结点为根结点。最下面一层的结点没有后继结点只有前驱结点,称为叶结点。除根结点和叶结点以外的其他结点称为树枝结点。除根结点外,每个结点有且只有一个前驱,但可以有多个后继。这种数据结构元素之间的关系是一对多的关系,即 1∶N 的关系(N≥0)。具有这种性质的数据结构称为树型结构或树结构。为区别于线性结构,又把诸如树结构这样的一类数据结构称为非线性结构。

数据结构定义中的“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。数据结构在计算机中的表示(又称为映像)称为数据的物理结构或存储结构。它包括数据

元素的表示和关系的表示。

数据元素之间的关系在计算机中有两种不同的表示方法:顺序映像和非顺序映像,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。

选择数据结构的目的是为了方便对数据进行操作,以有效地解决实际问题。因此,数据结构这一学科主要是研究面对实际问题时,如何选择好的数据的逻辑结构、数据的存储结构以及在此数据结构上进行各种操作的算法。

1.2 数据结构与算法

有专家曾经概括过这样一个公式:程序 = 算法 + 数据结构。这表明要设计出一个好的程序,首先选择好的数据结构,然后在此数据结构上设计出好的算法。本节将介绍算法的概念和分析算法优劣的方法。

1.2.1 算法的概念

解问题一步一步的过程称为算法。通常所说的算法是指在计算机上执行的计算过程的抽象描述,它和计算机程序不同,程序是在特定的计算机上执行算法。算法是抽象的,它与具体的计算机无关。

通常,解同一问题 P 可能有几种算法,因此要考虑:

(1)哪一个是解问题 P 的‘好’的算法?

为了回答这个问题,需要比较不同算法的有效性。

(2)什么是解问题 P 所需的基本运算的最少次数?

可以通过比较找到解问题 P 的所有算法的基本运算次数。

(3)在指定的适当时间内是否有解问题 P 的算法?

上述问题一直是计算机科学技术研究的核心问题之一。

1.2.2 算法评价和算法分析

评价、判断一个算法优劣的标准有:

1)正确性。这是评价算法优劣的前提。一个算法是正确的,其意思是指当输入合法数据时,能在有限的运行时间内得到正确的结果。

2)简明性。算法思路清楚、层次分明、易读易用、简单明了。

3)占用存储空间少。算法占用存储空间的大小称为算法的空间复杂性。

4)运行速度快。运行算法所需的时间多少称为算法的时间复杂性。

如何比较两个算法的运行速度呢?本节将着重讨论这个问题。

在 20 世纪 60 年代,对算法的有效性没有一个令人满意的客观标准。一个研究者可能会在杂志上发表一个算法及对于少数例题运行并得到一个执行时间;而几年后,第二个研究者又会给出一个改进了的算法以及对于同样例题运行并又得到另一个执行时间。新的算法肯定更快,因为相隔这几年里,计算机性能和程序设计语言都得到改善和提高。算法在不同的计算机上用程序设计语言编程运行,而且编程语言也可能不同,因此这种比较不能令人满意。一方

面,要弄清计算机性能提高与实现编程技术对算法执行时间的影响是很困难的。另一方面,有可能第二个研究者无意中将算法搞得对这些例题运行起来特别有效。可以想象,如果对于另外的例题再运行这两个算法,则可能第一个算法的运行速度更快。

因此,对算法的分析必须脱离具体的计算机和程序设计语言。

比较两个算法的好坏,首先是看其所需的运行时间,时间的长短是由算法所需的运算次数决定的。算法的运算次数又称为工作量或者称为算法的时间复杂性。任何一个算法都可能有多种运算(操作),但不可能对所有运算的量都作统计。因此,只能抓住其中影响算法运行时间最长的一种或几种运算作为基本运算。例如,在一个用户名字和电话号码的线性表中,对于查找用户 X 相应电话号码的问题,是将 X 和表中元素(用户名字)的比较运算作为算法的基本运算。又如在两个实数矩阵相乘的问题中,则是把两个实数相乘的运算作为算法的基本运算。

但是,即使是同一个问题,对于不同的输入,其基本运算的次数也可能不同,它与输入元素的个数有关。因此,引进问题大小的概念。例如,在一个用户名字和电话号码的线性表中,对于查找用户 X 相应电话号码的问题,问题的大小是用表中(用户名字,电话号码)的个数来表示。对于两个实数矩阵相乘的问题,问题的大小是用矩阵的阶来表示。

这样,一个算法的基本运算的次数就可以用问题的大小函数 f(n)来表示。f(n)是自然数 N→N 的一个函数。

对于算法的好坏可以进行最坏情形和平均情形分析。

(1)算法的平均情形复杂性

设 Dn 是问题大小为 n 的输入集合,I 是 Dn 的一个元素,P(I)是 I 出现的概率,而 T(I)是算法在输入 I 时所需的执行次数。则算法的平均情形复杂性定义为:

$$A(n) = \sum_{I \in Dn} P(I)T(I)$$

(2)算法的最坏情形复杂性

设 Dn 是问题大小为 n 的输入的集合,I 是 Dn 的一个元素,T(I)是算法输入 I 时所需的执行次数。则算法的最坏情形复杂性定义为:

$$W(n) = \underset{I \in Dn}{Max} T(I)$$

W(n)给出了算法时间复杂性的上界,用它来估算执行一个算法的时间界,这在实际应用中特别重要。因此,对于本书中给出的大多数算法将进行最坏情形分析,当说到一个算法的时间复杂性时,除非特别说明,否则均指最坏情形时间复杂性。

【例 3】　设 L 是一个包含 n 个整数元素的表,对某一指定元素 x,如果 x 在表中,则输出其下标;如果 x 不在表中,则输出零。用 C 语言描述的算法如下:

```
int Serial_Search(L,n)
  int L[];
  int n;
  {
    int j;
    j = 0;
    while((j< = n - 1) &&(L[j] ! = x))
      j = j + 1;
```

```
        if(j > n - 1)
           j = 0;
        return(j);
    }
```

上述算法所用到的运算有赋值、加法和比较。因此,其基本运算为 x 和表中元素的比较操作。

显然,在最坏的情形下,x 或者不出现在表中或者为表中最后一个元素。此时,算法所做的比较次数是 n 次,即 f(n) = n,也即算法的时间复杂性为 n 的线性函数。

【例 4】 考虑计算 f = 1 ! + 2 ! + 3 ! + … + n ! 的问题。用 C 语言描述的算法如下:

```
long int Factor_Sum( n)
int n;
{
   int i;
   long int f, w;
   f  = 0
   for (i = 1; i < = n; i + + )
   {
       w = 1;
       for (j = 1 ; j < = i; j + + )
           w = w * j;
       f = f + w;
    }
  return(f);
}
```

上述算法所用到的运算有乘法、加法、赋值和比较。其基本运算为乘法操作。

在上述算法的执行过程中,对外循环变量 i 的每次取值,内循环变量 j 循环 i 次。因为内循环每执行一次,内循环体语句 w = w * j 只做一次乘法操作,即当内循环变量 j 循环 i 次时,内循环语句 w = w * j 做 i 次乘法。所以,整个算法所做的乘法操作总数是:

$$f(n) = 1 + 2 + 3 + \cdots + n = n(n - 1)/ 2$$

衡量两个算法的好坏主要是比较它们工作量的大小,但当 n 较小时难以看出谁优谁劣。例如,对于某个问题 P,其第一个算法的工作量是 $w1 = 3n^2$,第二个算法的工作量是 w2 = 25n。当 n = 8 时,w1(8) = 192,w2(8) = 200,显然,w1(8) < w2(8),即 w1 的运算次数比 w2 的运算次数少。但是,不能据此立即断定第一个算法就一定比第二个算法好。这是因为,当 n = 9 时,w1(9) = 243,w2(9) = 225,且当 n > 9 时都有 w1(n) > w2(n),即第一个算法的工作量比第二个算法的大。

因此,衡量两个算法的好坏时,应当是在 n 足够大的情形下对算法的工作量进行比较,即对算法进行渐近性态分析。

设 f(n)和 g(n)是定义域为自然数 N 的非负函数,如果存在正整数 c 和 n_0,使得当 $n \geq n_0$ 时,有 $f(n) \leq cg(n)$,则称函数 f(n)的阶低于或等于函数 g(n) 的阶。记为“ f(n)是O(g(n)) ”或

“$f(n)\in O(g(n))$”,读做 $f(n)$是 $g(n)$的大 O。

【例 5】　设 $f(n)=3n^3+2n^2$, $g(n)=5n^3$;取 $c=5, n_0=0$,则当 $n\geq n_0$时,有 $3n^3+2n^2\leq 5n^3$。所以,$f(n)\in O(n^3)$。

【例 6】　设 $f(n)=3^n$, $g(n)=2^n$,则 $f(n)$不是 $O(2^n)$。因为存在常数 c 和 n_0,使得当 $n\geq n_0$时,有 $3^n\leq c2^n$,则有 $c\geq(3/2)^n$对于一切的 $n\geq n_0$成立。但是,实际上随着 n 的增大,$(3/2)^n$可以大于任意预先指定的常数 c,这说明所假设的常数 c 并不存在。

用定义判断两个算法时间复杂性函数 $f(n)$和 $g(n)$的阶比较繁琐,可以采用求极限的方法来比较:

$$\lim_{n\to\infty} f(n)/g(n)=c$$

则

(1)当 $c\neq 0$时,表示 $f(n)$和 $g(n)$同阶,记为 $f(n)=\Theta(g(n))$。

(2)当 $c=0$时,表示 $f(n)$比 $g(n)$低阶,记为 $f(n)\in O(g(n))$。

(3)当 $c=\infty$时,表示 $f(n)$比 $g(n)$高阶,记为 $f(n)=\Omega(g(n))$。

只要求出极限,就可以方便地判断任意两个时间复杂性函数阶之间的关系。

【例 7】　极限 $\lim_{n\to\infty}(n^2/2)/(307n^2)=1/614$,所以 $f(n)=n^2/2$ 与 $g(n)=307n^2$ 同阶。

【例 8】　极限 $\lim_{n\to\infty}(\log)_2 n/n=\lim_{n\to\infty}(\ln n)/(n\ln 2)=\frac{1}{\ln 2}\lim_{n\to\infty}(1/n)1n\ n=0$,所以 $f(n)=\log_2 n$ 比 $g(n)=n$ 低阶。

从上述分析看出,比较两个算法的优劣,并不是要精确统计其各种运算次数,而是要比较其工作量的阶。因此,把算法工作量 $f(n)$表示为 $O(g(n))$,其中大写字母 O 为英文单词 Order(即数量级)的第一个字母。

当 $f(n)$数量级与对数函数、幂函数或它们的乘积同阶时,算法的运行时间是可以接受的,称这些算法为有效算法。当 $f(n)$为指数函数或阶乘函数时,算法的运行时间随 n 的增大而变得不可接受,这类算法不是有效算法。

随着 n 的增大,对数函数的增长速度最慢,线性函数次之,其余类推。当 n 足够大后,各种不同数量级的 $f(n)$函数存在下列关系:

$$O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<\cdots<O(2^n)<O(n!)$$

算法时间复杂性除了与问题的大小 n 有关外,还与输入的具体数据和数据的输入次序即数据的分布有关。

分析算法的目的是为了改进算法。对上述例 4 的算法,在算法的第三步,每次都做 i 次乘法,这是不必要的。只要把 $(i-1)!$保存起来,下一次循环时只需做一次乘法就可以了。因此,改进的算法如下:

```
long int Update_Factor_Sum(n)
  int n;
   {
      int i ;
      long int f, w;
      f = 0;
      w = 1;
```

```
    for(i = 1;i < = n;i + + )
      {
       w = w * i;
       f = f + w;
      }
    return (f);
  }
```

这样,上述算法所做的乘法次数只有 n 次,即 f(n) = n,f(n)∈O(n);而例 4 中未改进的算法的时间复杂性为 f(n) = n(n - 1)/ 2∈O(n^2),它们相差整整一个数量级。

1.2.3 算法设计与程序设计

为了提高程序设计的质量和效率,可按下列三步进行程序设计:

1)题目分析:理解题意,明确要求,确定任务,建立解实际问题的数学模型。

2)算法设计:通过比较分析,选择出解此数学模型的最佳算法。

3)编程实现:选择适当的程序设计语言编写程序,然后上机调试以验证程序的正确性,最后运行获得结果。

【例 9】 编写一个求圆面积的 C 语言程序。

第一步,设计其算法如下:

1)确定圆的半径 r 的值。

2)利用公式 S = πr^2 进行计算。

3)输出显示结果。

第二步,将上述算法用 C 语言描述如下:

```
main()
  {
    float s,r;
    printf("r = ");
    scanf("%f",&r);
    printf("\n");
    s = 3.1415926536 * r * r;
    printf("The area = %f\n",s);
   }
```

习 题 一

1.举例说明什么叫数据、数据元素、数据处理和数据结构。

2.下列为一个用二元组表示的数据结构,试画出它们对应的图表示,并指出它们分别属于

何种结构。

(1) A = (K,R)

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

R = { ⟨01,02⟩ , ⟨02,03⟩ , ⟨03,04⟩ , ⟨04,05⟩ , ⟨05,06⟩ , ⟨07,08⟩ , ⟨08,09⟩ }

(2) B = (K,R)

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

R = { ⟨01,02⟩ , ⟨01,03⟩ , ⟨01,04⟩ , ⟨02,05⟩ , ⟨02,06⟩ , ⟨03,07⟩ , ⟨03,08⟩ , ⟨03,09⟩ }

(3) C = (K,R)

K = {a,b,c,d,e,f,g}

R = { ⟨a,b⟩ , ⟨b,c⟩ , ⟨c,d⟩ , ⟨d,e⟩ , ⟨e,f⟩ , ⟨f,g⟩ }

(4) D = (K,R)

K = {a,b,c,d,e,f,g}

R = { ⟨d,b⟩ , ⟨d,g⟩ , ⟨b,a⟩ , ⟨b,c⟩ , ⟨g,e⟩ , ⟨g,f⟩ }

3.对于如下的一元多项式

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

请编写出求其值的两种不同的算法,然后分析这两种算法的时间复杂性并指出哪一个算法更好。

4.设 n 为正整数,求出下列算法基本操作的执行时间 T(n)。

```
(1)   i = 1;
      k = 0;
      while(i < = n - 1)
      {   k = k + 10 * i;
          i = i + 1;
      }
(2)   x = 0;
      for(i = 1;i < = n;i + + )
          for(j = 1; j < = i; j + + )
              x = x + j;
(3)   x = 91;
      n = 100;
      while(n > 0)
      {
          if(x > 100)
          {   x = x - 10;
              y = y - 1;
          }
        else
            x = x + 10;
      }
```