计算机等级考试教程

（三级B）

数据结构

全国高等学校计算机教育研究会
教材与课程建设委员会 组编

李大友 主编

机械工业出版社

计 算 机 等 级 考 试 教 程

（三 级 B）

数 据 结 构

全国高等学校计算机教育研究会
教 材 与 课 程 建 设 委 员 会 组编

李大友 主编
徐孝凯 魏荣 编著

机 械 工 业 出 版 社

本书是根据国家教委制定的全国计算机等级考试三级 B 类考试大纲要求编写的，其深度和广度符合考试大纲要求。本书主要内容有：线性表、栈和队列的顺序存储与链接存储，相应的插入与删除运算的算法，栈在递归和非递归算法中的应用，树和二叉树的定义、性质、存储结构以及遍历算法，二叉排序树的建立、查找、插入和删除运算的算法，图的概念、存储结构和遍历等运算的算法，对数据进行顺序、二分、索引、分块、散列等查找运算的算法，对数据进行插入、选择、冒泡、希尔、堆、快速、归并等排序运算的算法。全书共分为 7 章，每章均配有丰富的选择题、填空题和其它类型的习题，书后附有习题参考答案。本书的突出特点是便于自学。

本书除作为全国计算机等级考试三级 B 类考生学习“数据结构与算法”内容的教材外，亦可作为大、中专院校开设数据结构课程的教材和教学参考书。

图书在版编目（CIP）数据

计算机等级考试教程（三级 B）：数据结构/李大友主编．—北京：机械工业出版社，1996.2
ISBN 7-111-04987-X

Ⅰ.计… Ⅱ.李… Ⅲ.①计算技术-基本知识-考试，等级-指导读物②数据结构-考试，等级-指导读物 Ⅳ.①TP3②TP311.12

中国版本图书馆 CIP 数据核字（95）第 22608 号

出 版 人：马九荣（北京市百万庄南街 1 号 邮政编码 100037）
责任编辑：何文军 蒋 克 版式设计：张世琴 责任校对：贾立萍
封面设计：郭景云 责任印制：卢子祥

三河永和印刷有限公司印刷 · 新华书店北京发行所发行
1996 年 2 月第 1 版第 1 次印刷
787mm×1092mm 1/16 · 15.25 印张 · 367 千字
0 001—6 000 册
定价：21.00 元

《计算机等级考试教程》

编　委　会

《计算机等级考试教程》序言

当前，在世界范围内，一个以微电子技术、计算机技术和通信技术为先导的，以信息技术和信息产业为中心的信息革命方兴未艾。信息技术和信息产业的发展，对国民经济的发展、国家经济信息化起着举足轻重的作用，并已成为衡量一个国家发展水平的重要标志。因此，实现国家经济信息化，已成为世界各国所追求的共同目标。

为了使我国尽快实现国家经济信息化，赶上发达国家的水平，必须加速发展我国的信息技术和信息产业。其中最关键的环节就是人才的培养，尤其是计算机应用人才的培养。有了人才，才能迅速提高全社会的计算机应用水平，促进国家经济信息化水平的提高。因此，解决全民普及计算机知识，尽快提高全民族整体的计算机应用水平，已成为当务之急。各行各业、各层次人员，不论年龄与知识背景如何，都应掌握和应用计算机，解决其各自专业领域的计算机应用问题，为本职工作或专业服务，使其与国家经济信息化的需要相适应。

国家教委考试中心为适应这一形势发展的需要，使所培养的计算机应用人才的水平有一个公正的、客观的统一标准，推出了全国计算机等级考试。这一考试，根据应试者所具有的计算机应用能力水平的不同，划分为不同等级，分别进行考核。

全国计算机等级考试共分为四级六类，其内容范围如下：

一级分为 A、B 两类，均面向文字处理和数据库应用系统操作人员。

一级 A 类要求掌握计算机基础知识、微机系统基本组成、操作系统功能和使用、字表处理软件的功能和使用、数据库应用系统的基本概念和操作。

一级 B 类要求掌握计算机基础知识、微机系统基本组成、DOS 操作系统基本知识及操作、文字处理软件 WPS 和数据库语言 FoxBASE 的操作。

二级面向使用高级语言进行程序设计的人员。要求掌握计算机基础知识、操作系统的功能和使用、数据库的基本概念及应用和具有使用一种高级语言（C 语言、PASCAL 语言、FORTRAN 语言、BASIC 语言或数据库语言）进行程序设计的能力。

三级分为 A、B 两类。

三级 A 类面向测控领域的应用人员。要求掌握微机原理、汇编语言程序设计、微机接口技术、软件技术基础以及微机在测控领域的应用。

三级 B 类面向软件方面的应用人员。要求掌握计算机基础知识、数据结构与算法、操作系统、软件工程方法以及具有微机在管理信息系统或数值计算或计算机辅助设计方面的应用能力。

四级要求达到相当于大学计算机专业本科毕业生水平，具有计算机软件和硬件系统的设计开发能力。要求掌握计算机系统原理、计算机体系结构、计算机网络与通信、离散数学、数据结构与算法、操作系统、软件工程和数据库系统原理等方面的基础理论知识。

为推动全国计算机等级考试的健康发展，满足社会上对等级考试教材的迫切要求，全国

高等学校计算机教育研究会课程与教材建设委员会组织了高等院校多年从事计算机教育的第一线专家教授，编写了《计算机等级考试教程》系列教材，并得到机械工业出版社的大力支持与合作，使得这套教程能够及时与广大读者见面。

这套教程严格按照各级各类考试大纲的要求编写，内容深入浅出、图文并茂，每本书均附有习题，便于自学。

由于计算机技术是一门迅速发展的学科及作者水平所限，这套教程肯定会有很多不足之处，衷心希望得到社会各界和广大读者的批评指正。

主编　李大友

1995 年 11 月

前　言

计算机是一种数据处理机，它能够存储大量的数据，并能够通过执行编制的程序来处理这些数据，使人们获得有用的信息。计算机存储的数据绝不是杂乱无章的，而是根据对数据处理的要求按照一定的结构组织起来的。数据的结构即数据的组织形式，从逻辑角度看，具有线性、树和图这三种基本结构，从存储角度看，具有顺序、链接、索引和散列这四种基本结构，但在实际应用中，数据的逻辑结构和存储结构往往是其基本结构的各种不同的组合。数据按照一定的存储结构存入计算机后，接着就是如何利用一种程序设计语言编写算法，对数据进行处理。对数据进行处理，通常是指对数据进行的非数值运算，它主要包括遍历运算、查找运算、插入运算、删除运算、排序运算等。对于同一种运算，数据的逻辑结构和存储结构不同，算法的设计思想不同，则可编写出不同的算法。本书就是研究数据结构与算法的一本教材，它是按照全国计算机等级考试三级B类考试大纲对“数据结构与算法”的具体要求编写的。

全书共分为7章。第1章主要讨论数据结构的基本概念，用“类PASCAL语言”描述算法规则与算法评价的一般原则，以及常用类型数据（如数组）的存储分配等；第2章和第3章主要讨论线性表、栈、队列等线性结构数据的顺序与链接存储结构，插入、删除等运算的算法，以及栈在递归和非递归算法中的应用等；第4章主要讨论一般树和二叉树的定义、性质、存储结构以及遍历算法，二叉排序树的建立、查找、插入和删除运算的算法，构造哈夫曼树等；第5章讨论图的基本概念，图的存储结构，图的遍历算法，求图的最小生成树，求图中顶点之间的最短路径，以及对图进行拓扑排序等；第6章讨论线性表的索引存储与散列存储，B树的结构，顺序、二分、索引、分块、散列、B树等查找运算的算法；第7章讨论直接插入排序、希尔排序、直接选择排序、堆排序、冒泡排序、快速排序、归并排序等排序运算的算法。

本书内容丰富、实用，叙述条理清楚、循序渐进，算法分析具体、描述详细，整体结构紧凑，风格前后一致，每一章后均配有丰富的选择题、填空题及其它习题，书后附有习题参考答案。本书充分考虑到广大自学读者的学习特点，使之便于自学。

本书除作为全国计算机等级考试三级B类考生学习“数据结构与算法”内容的教材外，亦可作为大、中专院校开设数据结构课程的教材和教学参考书。

本书由徐孝凯编写前四章，魏荣编写后三章。

由于作者水平有限，加之时间仓促，错误和不足之处在所难免，殷切希望授课老师和广大读者批评指正。

作者

1995年10月

目　录

第1章 绪 论

1.1 基本概念

数据(data)是人们利用文字符号、数字符号以及其它规定的符号对现实世界的事物及其活动所做的描述。因此,大到一本书、一篇文章、一张图表等是数据,小到一个句子、一个单词、一个算式,以至一个数值、一个字符等都是数据。在计算机领域,人们把能够被计算机加工的对象,或者说能够被计算机输入、存储、处理和输出的一切信息都叫做数据。

数据元素(data element)是一个数据整体中相对独立的单位。如对于一个文件来说,每个记录都是它的数据元素;对于一个字符串来说,每个字符都是它的数据元素;对于一个数组来说,每个下标变量都是它的数据元素。数据和数据元素是相对而言的,是整体与个体之间的关系。如对于一个记录来说,若把它所在的文件看作为整体数据的话,则它就是整体数据里的元素,而它相对于所含的数据项又可以看作为数据。

数据记录(data record)简称为**记录**。它是数据处理领域组织数据的基本单位。它又由更小的单位——**数据项**(item)所组成,一个记录一般包括 一个或若干个固定的数据项(当然每一个数据项还可以是记录的形式)。记录是数据的基本组织单位,数据元素经常组织成记录的形式,所以,记录和元素这两个术语往往不加区别地使用。

表 1-1-1 是一个图书目录表,表中第一行为表目行,又称目录行,它给出了该表中每条记录的结构;从表目行向下的每一行为一条记录,每条记录给出了一本图书的相关信息。表中的每一列为一个数据项,每个数据项描述记录的一个属性。每条记录都由 6 个数据项(属性)所组成,依次为:登录号、书号、书名、作者、出版社和定价。当记录不同时,所对应的同一数据项的值(又称属性值)可能相同,也可能不同。如对于第 4 条和第 6 条记录来说,出版社数据项的值就相同,同为"清华大学";对于第 1 条和第 3 条记录来说,书名数据项的值就不同,一个为《计算方法》,另一个为《PASCAL 程序设计》。

表 1-1-1 图书目录表

登录号	书号	书名	作者	出版社	定价
00001	ISBN 7-04-003907-9/TP · 103	计算方法	唐珍	高等教育	4.80
00002	ISBN 7-111-03247-0/TP · 158	计算机辅助制造	李德庆	机械工业	3.30
00003	ISBN 7-302-01842-1/TP · 828	PASCAL 程序设计	谭浩强	清华大学	14.00
00004	ISBN 7-302-01663-1/TP · 712	数据结构简明教程	徐孝凯	清华大学	13.80
00005	ISBN 7-304-00543-2/TP · 27	计算机应用基础	王利	中央电大	8.05
00006	ISBN 7-302-00860-4/TP · 312	C 程序设计	谭浩强	清华大学	7.30
00007	ISBN 7-80046-822-4/O · 026	应用概率统计	武继玉	航空工业	8.50
⋮	⋮	⋮	⋮	⋮	⋮

在一个表或文件中，若所有记录的某一数据项的值都不同；或者说，每个值能够唯一地标识一个记录时，则可把这个数据项作为记录的**关键数据项**，简称为**关键项**(key item)，关键项中的每一个值称为所在记录的**关键字**(key word 或 key)。在表 1-1-1 中，登录号数据项的值都不同，所以可把登录号作为记录的关键项，其中的每一个值都是所在记录的关键字。如 00002 为第 2 条记录的关键字，00005 为第 5 条记录的关键字等。

在一个表或文件中，能作为关键项的数据项可能没有，可能只有一个，也可能多于一个。当没有或有具体需要时，可把若干个非关键数据项联合起来，构成一个组合关键项，用组合关键项中的每一个值来唯一地标识一个记录。

引入了记录的关键项和关键字后，为简便起见，在以后的讨论中，经常利用关键项来代表所有记录，利用关键字来代表对应的记录。

数据处理(data processing)是指对数据进行查找、插入、删除、合并、排序、统计、简单计算、输入、输出等的操作过程。在早期，计算机主要用于科学和工程计算，而现在，计算机主要用于数据处理。数据处理又常称作信息管理。象办公室自动化系统、情报检索系统、经济信息管理系统、物资调配系统、财务管理系统等都是计算机在数据处理方面的具体应用。数据结构与算法是进行数据处理的软件基础，因此，该门课程是学习计算机知识的主干课程之一。

数据结构(data structure)是指数据以及数据之间的联系。数据之间的联系可以是固有的，也可以是根据数据处理的需要人为定义的。数据结构又称作数据的**逻辑结构**，数据结构(即数据及其联系)在计算机存储器上的存储表示称作为数据的**物理结构**或**存储结构**。由于存储表示的方法有顺序、链接、索引、散列等多种，所以一种数据结构可以根据需要表示成一种或多种物理结构。在有的书中，把数据的逻辑结构、物理结构以及相关运算合称为数据结构，而在本教材中，为了叙述清楚起见，所说的数据结构专指数据的逻辑结构，它与物理结构和运算无关。

为了更确切地定义数据结构，通常采用二元组表示：

$$B=(K,R)$$

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。其中：

$$K=\{k_i \mid 1\leqslant i\leqslant n,\ n\geqslant 0\}$$

$$R=\{r_j \mid 1\leqslant j\leqslant m,\ m\geqslant 1\}$$

k_i 表示第 i 个数据元素，n 为 B 中数据元素的个数，特别是，若 n=0，则 K 是一个空集，因而 B 也就无结构可言，或者说它具有任何结构；r_j 表示第 j 个二元关系(以后简称为关系)，m 为 K 上关系的个数。

在本书所讨论的数据结构中，一般只讨论 m=1 的情况，即 R 中只包含一个关系(即 R={r})的情况。对于包含有多个关系的数据结构，可分别对每一个关系进行讨论。

K 上的一个关系 r 是序偶的集合。对于 r 中的任一序偶〈x,y〉(x,y∈K)，我们把 x 叫做序偶的第一元素，把 y 叫做序偶的第二元素，又称序偶的第一元素为第二元素的**前驱**或**前驱元素**，称第二元素为第一元素的**后继**或**后继元素**。如在〈x,y〉序偶中，x 为 y 的前驱，y 为 x 的后继。

数据结构还能够利用图形形象地表示出来，图形中的每个顶点(又称结点)表示一个数据元素，两顶点之间带箭头的连线(又称作有向边或弧)表示关系中的一个序偶，其中序偶的第一元素为有向边的起始顶点，第二元素为有向边的终止顶点。

下面通过表 1-1-2 构造出一些典型的数据结构，从而加深对数据结构这一重要概念的理

解。

表 1-1-2 教务处人事简表

职工号	姓名	性别	出生年月	职务	单位
01	刘一刚	男	1956 年 8 月	处长	
02	王 平	男	1954 年 2 月	科长	教材科
03	赵玉华	女	1973 年 1 月	科员	教材科
04	孙小明	男	1965 年 10 月	科员	教材科
05	马天敏	女	1968 年 5 月	主任	办公室
06	刘 明	男	1957 年 3 月	科员	办公室
07	魏 杰	男	1946 年 12 月	科长	考务科
08	王启星	男	1964 年 2 月	科员	考务科
09	何 晨	男	1973 年 4 月	科员	考务科
10	沈 红	女	1962 年 3 月	科员	考务科

表 1-1-2 中共有 10 条记录，每条记录都由 6 个数据项所组成，由于每条记录的职工号各不相同，所以可把每条记录的职工号作为该记录的关键字，并在下面的举例中，用记录的关键字来代表整个记录。

例 1-1 一种数据结构 linearity＝(K,R)，其中：

K＝{01,02,03,04,05,06,07,08,09,10}

R＝{r}

r＝{〈07,02〉,〈02,01〉,〈01,06〉,〈06,10〉,〈10,08〉,〈08,04〉,〈04,05〉,〈05,03〉,〈03,09〉}

对应的图形如图 1-1-1 所示。

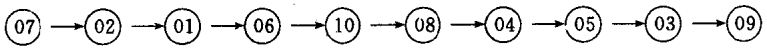

图 1-1-1 线性结构示意图

结合表 1-1-2，细心的读者不难看出：r 是按职工年龄从大到小排列的关系。

在 linearity 数据结构中，每个数据元素有且只有一个前驱元素(除结构中第一个元素 07 外)。有且只有一个后继元素(除结构中最后一个元素 09 外)。这种数据结构的特点是数据元素之间的 1 对 1 联系，即线性关系，把具有这种特点的数据结构叫做线性数据结构，简称为**线性结构**。

例 1-2 一种数据结构 tree＝(K,R)，其中：

K＝{01,02,03,04,05,06,07,08,09,10}

R＝{r}

r＝{〈01,02〉,〈02,03〉,〈02,04〉,〈01,05〉,〈05,06〉,〈01,07〉,〈07,08〉,〈07,09〉,〈07,10〉}

对应的图形如图 1-1-2 所示。

结合表 1-1-2，细心的读者不难看出：r 是人员之间领导与被领导的关系。

图 1-1-2 像倒着画的一棵树，在这棵树中，最上面的一个没有前驱只有后继的结点叫做树

根结点，最下面一层只有前驱没有后继的结点叫做树叶结点，树叶以外的结点叫做树枝结点。在一棵树中，每个结点有且只有一个前驱结点（除树根结点外），但可以有任意多个后继结点（含 0 个、1 个或若干个）。这种数据结构的特点是数据元素之间的 1 对 N 联系（N≥0），即**层次关系**，把具有这种特点的数据结构叫做**树型结构**，简称**树**。

例 1-3 一种数据结构 graph＝(K,R)，其中：

K＝{01,02,04,05,08,09,10}

R＝{r}

r＝{〈01,02〉,〈02,01〉,〈01,04〉,〈04,01〉,〈01,05〉,〈05,01〉,〈02,08〉,〈08,02〉,〈04,09〉〈09,04〉,〈05,09〉,〈09,05〉,〈08,09〉,〈09,08〉,〈09,10〉,〈10,09〉}

对应的图形如图 1-1-3 所示。

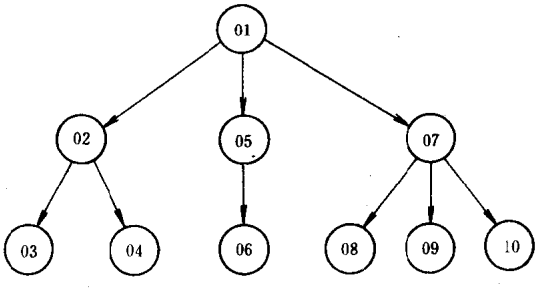

图 1-1-2 树型结构示意图

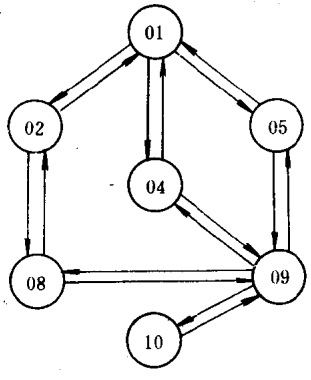

图 1-1-3 有向图

从图 1-1-3 可以看出，r 是 K 上的对称关系，为了简化起见，把〈x,y〉和〈y,x〉这两个对称序偶用一个无序对(x,y)或(y,x)来代替；在图形表示中，把 x 结点和 y 结点之间两条相反的有向边用一条无向边来代替。这样 r 关系可改写为

r＝{(01,02),(01,04),(01,05),(02,08),(04,09),(05,09),(08,09),(09,10)}

对应的图形如图 1-1-4 所示。

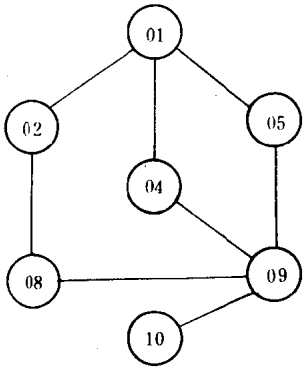

图 1-1-4 无向图

如果说 r 中每个序偶里的两个元素所代表的人员是好友的话，那么 r 关系就是人员之间的好友关系。

从图 1-1-3 或 1-1-4 可以看出，结点之间的联系是 M 对 N 联系（M≥0,N≥0），即**网状关系**，也就是说，每个结点可以有任意多个前驱结点和任意多个后继结点。把具有这种特点的数据结构叫做**图型结构**，简称**图**。

从图型结构、树型结构和线性结构的定义可知，树型结构是图型结构的特殊情况（即 M＝1 的情况），线性结构是树型结构的特殊情况（即 N＝1 的情况）。为了区别于线性结构，把树型结构和图型结构统称为**非线性结构**。

例 1-4 一种数据结构 B＝(K,R)，其中：

$K=\{k_1,k_2,k_3,k_4,k_5,k_6\}$

$R=\{r_1,r_2\}$

$r_1=\{\langle k_3,k_2\rangle,\langle k_3,k_5\rangle,\langle k_2,k_1\rangle,\langle k_5,k_4\rangle,\langle k_5,k_6\rangle\}$

$r_2=\{\langle k_1,k_2\rangle,\langle k_2,k_3\rangle,\langle k_3,k_4\rangle,\langle k_4,k_5\rangle,\langle k_5,k_6\rangle\}$

若用实线表示关系 r_1，虚线表示关系 r_2，则对应的图型如图 1-1-5 所示。

从图 1-1-5 可以看出，数据结构 B 是一种非线性的图型结构。但是，若只考虑关系 r_1 则为树型结构，若只考虑关系 r_2 则为线性结构。

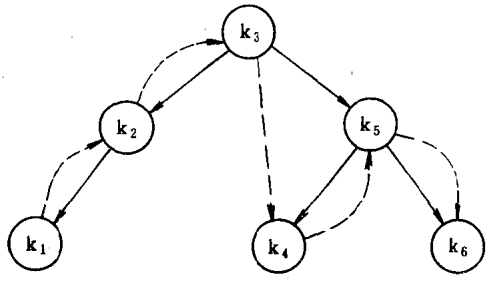

图 1-1-5　带有两个关系的数据结构示意图

算法(algorithm)就是解决特定问题的方法。描述一个算法可以采用某一种计算机程序设计语言，也可以采用流程图、N-S 图或 PAD 图等。作为一个算法应具备如下 5 个特性：

1)有穷性　一个算法必须在执行有穷步之后结束。

2)确定性　算法中的每一步都必须具有确切的含义，无二义性。

3)可行性　算法中的每一步都必须是可行的，也就是说，每一步都能够通过手工或机器在有限时间内完成。

4)输入　一个算法可以有 0 个、1 个或多个输入量，在算法被执行之前提供给算法。

5)输出　一个算法执行结束后至少要有一个输出量。

待解决的特定问题可分为数值的和非数值的两类。解决数值问题的算法叫做数值算法。科学和工程计算方面的算法都属于数值算法，如求解数值积分、线性方程组、代数方程、微分方程等。解决非数值问题的算法叫做非数值算法。数据处理方面的算法都属于非数值算法，如各种排序算法、查找算法、插入算法、删除算法、遍历算法等。数值算法和非数值算法并没有严格的区别，一般说来，在数值算法中主要进行算术运算，而在非数值算法中，则主要进行比较和逻辑运算。另一方面，特定的问题可能是递归的，也可能是非递归的，因而解决它们的算法就有递归算法和非递归算法之分。当然，从理论上讲，任何递归算法都可以通过循环、堆栈等技术转化为非递归算法。

在计算机领域，一个算法实质上是针对所处理问题的需要，在数据的逻辑结构和存储结构的基础上施加的一种运算。由于数据的逻辑结构和存储结构不是唯一的，在很大程度上可以由用户自行选择和设计，所以处理同一个问题的算法也不是唯一的。另外，即使对于具有相同的逻辑结构和存储结构而言，其算法的设计思想和技巧不同，编写出的算法也大不相同。学习数据结构与算法这门课程的目的，就是要学会根据数据处理问题的需要，为待处理的数据选择合适的逻辑结构和存储结构，进而设计出比较满意的算法。

1.2　算法描述

本书在进行算法描述时，采用“类 Pascal 语言”作为工具。类 Pascal 语言是在标准 Pascal 语言的基础上所做的修改，它忽略了标准 Pascal 语言中语法规则的一些细节，同时增加了一些功能较强的语句，这样使得描述出来的算法清晰、直观，便于阅读和分析。读者根据这些算法不难改写为符合任一种 Pascal 语言（如标准 Pascal、IBM-PC Pascal、Turbo Pascal 等）或其它任一种计算机语言语法规则的算法来。

类 Pascal 语言所包含的语句如下：

1. 赋值语句

变量名：＝表达式；

2. 转向语句

goto 语句标号；

3. 调用过程或函数语句

过程名（参数表）；或者　函数名（参数表）；

4. 返回语句

return；

用于过程或函数体中，该语句被执行时，将立即退出当前过程或函数，返回到调用前所处的位置继续向下执行，它是执行过程或函数的一个非正常出口。

在函数体中使用该语句还可以书写成 return（表达式）的形式，表示把表达式的值赋给函数名后再退出当前被执行的函数，它相当于如下两条语句：

函数名：＝表达式；return；

5. 出错处理语句

error（字符串）；

在算法中为了避免非法操作，需要进行出错处理时统一使用该语句，它表示此算法的执行到此中止。在符合上机要求的实际算法中，它可能是转去执行一个错误处理程序，也可能是简单地打印出错误信息并停止运行等。该语句括号内的字符串用以说明出错的原因，如使用字符串'overflow'表示溢出，'out of range'表示超出范围或越界等。

6. 复合语句

begin 语句 1；语句 2；…语句 n end；

或简记为

[语句 1；语句 2；…语句 n]

7. 条件语句

if 条件 then 语句 1 [else 语句 2]；

其中，中括号部分可以省略，这里的语句 1 和语句 2 可以是任何语句，包括是复合语句。

8. 情况语句

格式 1：

```
case 变量名 of
  常量 1：语句 1；
  常量 2：语句 2；
  ⋮      ⋮
  常量 n：语句 n
end；
```

格式 2：

```
case
  条件 1：语句 1；
  条件 2：语句 2；
```

⋮　　⋮

条件 n：语句 n；

else 语句 n+1

end；

此语句的功能是从所含的几条语句中选择出相匹配的一条语句加以执行，对于格式 1，若没有相匹配的语句，则什么也不执行，对于格式 2，若没有相匹配的语句，则执行 else 后面的语句 n+1。当然该语句中所含的每一条语句都可以是包含复合语句在内的任何语句。

9. for 循环语句

格式 1：

for 变量名：＝初值 to 终值 do 语句；

格式 2：

for 变量名：＝初值 downto 终值 do 语句；

若格式 1 中的初值大于终值，或者格式 2 中的初值小于终值，则什么也不执行，否则循环体（即 do 后面的语句）被反复执行的次数为：终值减初值后取绝对值再加 1。

10. while 循环语句

while 条件 do 语句；

当条件成立时执行循环体（即 do 后面的语句），直到条件不成立为止。

11. repeat 循环语句

repeat 一组语句 until 条件；

反复执行循环体（即 repeat 和 until 之间的语句序列），直到 until 后面的条件成立为止。

在上面三种循环语句中，作为循环体的语句可以是包含复合语句在内的任何语句。

12. 退出循环语句

exit；

用于各种循环语句中，该语句被执行时，将立即退出当前循环，相当于 goto 到该循环语句后的第一条语句上，它是各种循环语句的一个非正常出口。

在这 12 种语句中，第 4、5、12 种语句和第 8 种格式 2 语句是标准 pascal 语言所没有的，但在其它的 pascal 语言中具有类似的语句。

采用类 pascal 语言描述算法的书写规则如下：

1)所有算法均以过程或函数说明的形式给出，算法的输入数据一般来自参数表，输出数据一般也由函数名或参数表中的变参带回，这样就使得每个算法成为功能相对独立的一个模块。

2) 对参数表中的参数一般不进行类型说明，若参数是变参，则规定以大写字母或大写字母开头的标识符表示，若参数是值参，则规定以小写字母或小写字母开头的标识符表示，当然作为指定说明的参数除外。

3) 过程和函数体中的定义和说明部分一般被省略，语句部分通常按内容层次分别对语句进行编号，以便分析。第一层编号用（1）、（2）、（3）、……表示，第二层编号用（Ⅰ）、（Ⅱ）、（Ⅲ）、……表示，第三层编号用（a）、（b）、（c）、……表示，并规定除最外面的语句括号 begin 和 end 不省略外，其余层次均省略。

例如，对于一维整型数组 A（1：n）或表示为 A［1.. n］，其中 1 和 n 分别表示该数组下标的下界和上界，若要求从此数组中分别查找出具有最大值和最小值的元素，并把它们的

值分别赋给变参 Max 和 Min，则算法描述为：

```
procedure find(A,n,Max,Min);
  begin
    (1)Max:=A[1];Min:=A[1];  {赋初值}
    (2)for i:=2 to n do  {顺序查找过程}
      (I)if A[i]>Max then Max:= A[i];
      (Ⅱ)if A[i]<Min then Min:= A[i]
  end;
```

此算法的描述完全符合上面所述的三条规则。

4)当两个变量 x 和 y 需要相互交换值时，可简记为：

x↔y

实际上它对应下面三条赋值语句：

```
t:=x;x:=y;y:=t;  {t 为临时工作变量}
```

此外，在利用类 pascal 语言描述算法时，还需要使用标准 pascal 语言中的所有数据类型、标准过程和函数等。

1.3 算法评价

对于解决同一个问题，往往能够编写出许多不同的算法。例如，对于一批数据的排序问题，在第七章中将介绍多种算法。进行算法评价的目的，既在于从解决同一问题的不同算法中选择出较为合适的一种，也在于知道如何对现有算法进行改进，从而有可能设计出更好的算法。

一般从以下四个方面对算法进行评价。

1.3.1 正确性

正确性是设计和评价一个算法的首要条件，如果一个算法不正确，其它方面就无从谈起。一个正确的算法是指在合理的数据输入下，能在有限的运行时间内得出正确的结果。通过对数据输入的所有可能情况的分析和上机调试可以证明算法是否正确。当然，要从理论上证明一个算法的正确性，并不是一件容易的事，也不属于本书所研究的范围，故不作讨论。

1.3.2 运行时间

运行时间是指一个算法在计算机上运行所花费的时间。它大致等于计算机执行一种简单操作(如赋值、转向、比较、输入、输出操作等)所需的平均时间与算法中进行简单操作次数的乘积。因为执行一种简单操作所需的平均时间随机器而异，它是由机器本身硬软件环境决定的，与算法无关，所以只需讨论影响运行时间的另一个因素——算法中进行简单操作的次数。

不管一个算法是简单的还是复杂的，最终都是被分解成简单操作来具体执行的，因此，每个算法都对应着一定的简单操作的次数。显然，在一个算法中，进行简单操作的次数越少，其运行时间也就相对地越少；次数越多，其运行时间也就相对地越多。所以，通常把算法中包含简单操作次数的多少叫做算法的时间复杂性，它是一个算法运行时间的相对量度。

若解决一个问题的规模为 n，如在排序问题中，n 表示待排序的数据元素的个数，在查找问题中，n 表示查找表中数据元素的个数，在图的运算中，n 表示图中的顶点数，那么，算法的时间复杂性就是 n 的一个函数，假定记作为 T(n)。下面通过两个算法来分析其时间复杂性。

算法 1：累加求和

```
function sum(A,n):real;
                {A 表示一维实型数组,n 表示数组的大小,即所含数据元素的个数}
  begin
    (1) s:=0;{给累加变量 s 赋初值}
    (2) for i:=1 to n do{进行累加求和}
          s:=s+A[i];
    (3) sum:=s{把 s 值(即累加和)赋给函数名}
  end;
```

计算机执行这个算法时,第(1)步和第(3)步都只需一次赋值操作,为了分析第(2)步包含多少简单操作的次数,可改写为

```
(2)  i:=1;                    1 次
   1:if i>n then goto 2;      n+1 次
     s:=s+A[i];               n 次
     i:=i+1;                  n 次
     goto 1;                  n 次
(3)2:sum:=s
```

把第(2)步分解后的每一条语句的执行次数加起来,就得到了它包含的简单操作的次数,即为 4n+2。因此,算法(1)的时间复杂性为

$$T(n)=4n+4$$

算法 2:矩阵相加

```
procedure matrixadd(A,B,C,n);{A,B,C 分别为 n×n 阶矩阵,A,B 表示两个加数,C
                              表示和}
  begin
    for i:=1 to n do
      for j:=1 to n do
        c[i,j]:=A[i,j]+B[i,j]
  end
```

通过与算法 1 相似的分析过程可得到算法 2 的时间复杂性为

$$T(n)=4n^2+5n+2$$

算法 1 和算法 2 的时间复杂性还比较容易计算,因为算法比较简单,同时 for 循环中的循环次数是固定的。但是当算法较复杂,同时包含有 while 循环或 repeat 循环时,其时间复杂性的计算就相当困难了。实际上,一般也没有必要精确地计算出算法的时间复杂性,只要大致计算出相应的**数量级**(Order)即可。下面接着讨论时间复杂性 T(n)的数量级表示。

设 T(n)的一个辅助函数为 f(n),定义为当 n 大于等于某一足够大的正整数 n_0 时,存在着两个正的常数 a 和 b(a<b),使得 a<T(n)/f(n)<b 均成立,则称 f(n)是 T(n)的同数量级函数。把T(n)表示成数量级的形式为

$$T(n)=O(f(n))$$

其中,大写字母 O 为英文 Order(即数量级)一词的第一个字母。这种表示的意思是指 f(n)同 T(n)只相差一个常数倍。