21世纪高等教育计算机规划教材　国家级双语教学示范课程配套教材

COMPUTER

数据结构与算法

Data Structures and Algorithm

彭军 向毅　主编
熊茜 裴仰军　副主编

- 依据计算机教指委专业规范编写
- 以数据的逻辑结构、存储结构和相应算法为主线
- 附有习题和上机实验

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

数据结构与算法

Data Structures and Algorithm

彭军 向毅 主编

熊茜 裴仰军 副主编

人民邮电出版社

北京

图书在版编目（CIP）数据

数据结构与算法 / 彭军，向毅主编. -- 北京 : 人民邮电出版社，2013.1
21世纪高等教育计算机规划教材
ISBN 978-7-115-28770-0

Ⅰ. ①数… Ⅱ. ①彭… ②向… Ⅲ. ①数据结构－高等学校－教材②算法分析－高等学校－教材 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2012)第227459号

内 容 提 要

本书是国家级双语教学示范课程《数据结构》的配套教材，根据教育部高等学校计算机科学与技术教学指导委员会制定的《高等学校计算机科学与技术专业发展战略研究报告暨专业规范》编写。全书每章均以数据的逻辑结构、存储结构和相应的算法实现为主线，并对算法的运算效率进行分析。全书分为8章，涵盖了各种常见数据结构。第1章主要介绍数据结构和算法分析的基本概念，第2～6章主要介绍典型的线性结构、树型结构和图型结构，第7～8章分别介绍查找和排序操作。

另外，每章后面附有习题和上机实验内容，上机实验提供了完整的、可运行的程序上机实验供读者参考，以加深读者对所学知识的理解和应用。

本书既可作为高等院校计算机及相关专业数据结构课程的教学用书，也可作为从事计算机工程与应用的广大读者的参考书。

21世纪高等教育计算机规划教材

数据结构与算法

◆ 主　　编　彭　军　向　毅
副 主 编　熊　茜　裴仰军
责任编辑　刘　博

◆ 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
中国铁道出版社印刷厂印刷

◆ 开本：787×1092　1/16
印张：16　　2013年1月第1版
字数：403千字　　2013年1月北京第1次印刷

ISBN 978-7-115-28770-0

定价：34.00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”是一门关于非数值数据在计算机中表示、变换及处理的课程。在非数值计算中，处理对象已从简单数值发展到具有一定结构的数据，这就需要讨论如何有效地组织计算机的存储，并在此基础上有效地实现对象间的运算，数据结构就是研究与解决这些问题的重要基础。

数据结构课也是计算机科学与技术专业的一门必修的、重要的专业基础课，是计算机程序设计的重要理论技术基础。通过数据结构课的学习，使学生掌握数据结构的基本内容、典型算法和使用方法，培养学生应用数据结构相关知识编写高效应用程序解决具体应用问题的能力。

本书的作者是重庆科技学院国家级双语教学示范课程《数据结构》课程组成员，他们对该课程的教学理念和教学方法进行了长期深入的探索，结合本科教育教学改革的成果，共同完成了本书的编写。全书分为 8 章，涵盖了数据结构的主要内容。

第 1 章主要介绍数据结构中的逻辑结构、存储结构和运算的相关概念，对类 C 的语法进行简介，并详细介绍了算法分析的概念。

第 2 章主要介绍线性表的存储结构及相应的运算算法，对两种存储结构的算法效率进行比较。

第 3 章主要介绍栈的概念、基本运算及利用栈进行表达式求值、递归过程实现，介绍队列的基本概念，讨论顺序队列、链队列和循环队列的基本运算及实现。

第 4 章主要介绍串的顺序和链接存储结构及基本运算算法介绍数组的顺序存储结构及特殊矩阵的压缩存储。

第 5 章主要介绍树的有关概念和树的存储结构介绍二叉树、满二叉树、完全二叉树、线索二叉树的有关概念讨论二叉树的 3 种遍历方式的递归算法和前（或中）序遍历的非递归算法哈夫曼树的建立算法、哈夫曼编码的算法；树、树林和二叉树的转换方法。

第 6 章主要介绍图的有关概念；邻接矩阵、邻接表的存储结构；深度优先和广度优先遍历的算法；介绍了 Prim 算法和 Kruskal 算法思想求最小生成树；通过有向无环图的讨论，介绍拓扑排序和求关键路径的算法思想；利用 Floyd 算法求最短路径。

第 7 章主要介绍查找的有关概念，讨论了静态查找表中顺序查找、折半查找、分块查找的算法实现；动态查找表中排序二叉树的算法实现和平衡二叉树的基本概念；介绍了散列表的构造方法和处理冲突的方法。并对上述各种查找方法的效率用平均查找长度进行分析。

第 8 章主要介绍几种常用的内部排序算法，并对各种内部排序方法进行比较和讨论。

本书的编写工作由彭军教授主持，由彭军、向毅担任主编，由熊茜、裴仰军担任副主编。其中，第 1 章由彭军编写，第 3 章、第 5 章由向毅编写，第 4 章、

第7章由熊茜编写，第2章、第6章由裴仰军编写，第8章由黄永文编写。在本书编写过程中，得到了许多同行专家、教师的支持与帮助，在此，对他们表示衷心的感谢。

本书的编写参考了近年来国内外出版的大量书籍及相关技术资料，吸取了许多同仁和专家的宝贵经验。尽管我们付出了很大的努力，但由于作者学识水平有限，书中难免出现不妥之处，恳请广大读者批评指正。

编者

2012年5月

目 录

第1章 绪论

随着计算机应用领域的不断扩大，大量出现的非数值计算问题，如图书检索系统、交通灯管理系统、工艺过程最优控制、模式识别、机器视觉等，占据了当今计算机应用的绝大多数。这类问题涉及的数据元素之间的关系更为复杂，一般很难用数学方程式加以描述。因此，我们必须对数据的特性、数据之间的关系（结构）及其操作进行系统研究，而解决问题的关键就是要设计一个“好”的数据结构。目前，广泛使用的各类计算机系统软件和应用软件都要用到各种复杂的数据结构。

在国外，“数据结构”作为一门独立的课程，是从1968年才开始设立的。1968年美国唐·欧·克努斯（D·E·Knuth）教授[1]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《基本算法》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。目前在我国，“数据结构”是计算机及其相关专业中一门重要的专业基础课，主要学习数据的逻辑结构、数据的物理结构以及针对不同数据结构的算法。掌握好数据结构的知识，在帮助我们提高解决实际问题的能力的同时也为后续专业课程（如数据库原理、操作系统、编译原理）的学习打下坚实基础。

1.1 数据及其逻辑结构

本节，我们将对一些基本概念如数据、数据元素、数据对象等进行定义，并对数据的逻辑结构进行论述。

1.1.1 基本概念

（1）数据（Data）是信息的载体，是对客观事物的符号表示。在计算机科学中，数据是指所有能输入到计算机中并被程序处理的符号的总称。数据能够被计算机识别、存储和加工处理，是计算机程序加工的“原料”。随着计算机应用领域的不断扩大，数据的含义也随之扩展，其范畴包括整数、实数等数值数据，也包括字符串、图像和声音等非数值数据。例如，一个代数方程求解程序中所用的数据是整数和实数，而一个图像处理系统如Adobe Photoshop所处理的数据是图像。

（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。一个数据元素可由若干个数据项（Data Item）组成，并且数据项是不可分割的、含有独立意义的最小数据单位。

[1] D·E·Knuth教授因在算法分析和编程语言设计方面的突出贡献，1974荣获美国计算机协会图灵奖，1975年当选为美国国家科学院院士。

（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{0，±1，±2，±3，...}，工科类专业数据对象是{石油工程，冶金工程，机械工程，自动化，...}。

1.1.2 逻辑结构

逻辑结构（Logical Structure）是指数据元素之间的逻辑关系。逻辑结构是从逻辑关系上描述数据，与数据的存储无关，是独立于计算机的。数据的逻辑结构可以看作是从具体问题抽象出来的数学模型。

根据数据元素之间关系的不同特性，通常有下列四类基本逻辑结构。

（1）集合结构。该结构中的数据元素之间除了“属于同一个集合”的关系之外，别无其他关系，是极为松散的一种结构。

（2）线性结构。该结构中的数据元素之间存在一对一的关系（线性关系），即除了开始元素和结束元素之外，每一个元素都有且仅有一个直接前驱和一个直接后继。例如，线性表就是一种典型的线性结构。

（3）树状结构。该结构中的数据元素之间存在一对多的关系，即每个元素最多只有一个直接前驱，但可以有多个直接后继。例如，二叉树就是一种典型的树状结构。

（4）网状结构。该结构中的数据元素之间存在多对多的关系，即数据元素之间连接成网状，每个元素的直接前驱和直接后继的个数都可以是任意的。

图 1.1 所示为上述四类基本结构的关系图。

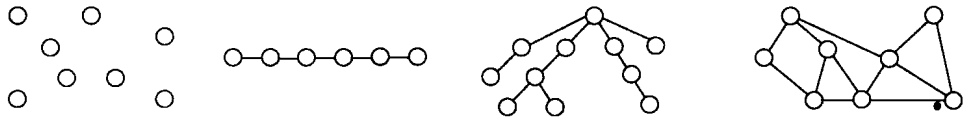


图 1.1　四类逻辑结构示意图

1.2 数据结构

1.2.1 什么是数据结构

数据结构（Data Structure）只是在整个计算机科学与技术领域上广泛被使用的术语，目前对它还没有一个公认的定义，不同的人根据自身的理解有不同的论述。

美国 Florida 大学的 Sartaj Sahni 教授在他的《数据结构、算法与应用》一书中称：“数据结构是数据对象，以及存在于该对象的实例和组成实例的数据元素之间的各种联系。这些联系可以通过定义相关的函数来给出。”

美国 Virginia Tech 大学的 Clifford A.Shaffer 教授在《数据结构与算法分析》一书中的定义是：“数据结构是 ADT（抽象数据类型）的物理实现。”

Robert L.Kruse 在《数据结构与程序设计——C++语言描述》一书中，将一个数据结构的设计过程分成抽象层、数据结构层和实现层。其中，抽象层是指抽象数据类型层，它讨论数据的逻辑

结构及其运算，数据结构层和实现层讨论一个数据结构的表示和在计算机内的存储细节以及运算的实现。

Richard F.Gilberg 和 Behrouz A.Forouzan 在《Data Structures：A Pseudocode Approach with C》一书中，描述数据结构为数据元素的集合，每一个数据元素可以是一种数据类型或另一个数据结构，以及存在于数据元素之间的一组关系（即结构）。

在本书中，数据结构定义为相互之间存在一种或多种特定关系的数据元素的集合，一般包括如下三个方面：

（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）施加在数据上的操作，即数据的运算。

从学科角度来讲，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运算等的学科。

1.2.2　数据结构的二元组描述

为了更确切地描述一种数据结构，通常采用二元组表示，即

```
Data_Structure = (D, S)
```

式中，D 是数据元素的有限集合，S 是 D 上关系的有限集。D 上关系可以用一有序对 $<x,y>$ 表示，其中 $x,y \in D$，x 称作 y 的直接前驱；y 称作 x 的直接后继。对于对称序对，即若 $<x,y> \in S$，则 $<y,x> \in S$，可以用 $(x,y) \in S$ 表示。下面给出几个例子加以说明。

【例 1.1】　在计算机科学中，复数可取如下定义：复数是一种数据结构

```
Complex = (C, R)
```

式中，C 是含两个实数的集合{c1,c2}；R={P}，而 P 是定义在集合 C 上的一种关系{<c1,c2>}，其中有序对<c1,c2>表示 c1 是复数的实部，c2 是复数的虚部。

【例 1.2】　有一种数据结构 Linear_List=(D, S)，式中

```
D={1, 3, 5, 7, 2, 4, 6}
S={<2,5>, <5,3>, <3,6>, <6,1>, <1,4>, <4,7>}
```

对应的逻辑结构图如图 1.2 所示。

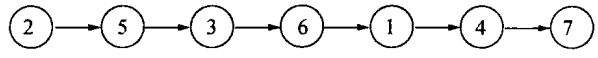


图 1.2　数据结构 Linear_List 对应的逻辑结构图

从该图可以看出，数据结构 Linear_List 表达的是一种线性结构。

【例 1.3】　有一种数据结构 Tree=(D, S)，式中

```
D={a, b, c, d, e, f, g}
S={(b,a), (b,c), (a,d), (a,e), (c,f), (e,g)}
```

对应的逻辑结构图如图 1.3 所示。

从该图可以看出，数据结构 Tree 表达的是一种树状结构。

【例 1.4】　有一种数据结构 Graph=(D, S)，式中

```
D={a, b, c, d, e, f, g}
S={S1, S2}
S1={<b,a>, <b,c>, <a,d>, <a,e>, <c,f>, <e,g>}
S2={<a,d>, <d,b>, <b,e>, <e,g>, <g,c>, <c,f>}
```

对应的逻辑结构图如图 1.4 所示。

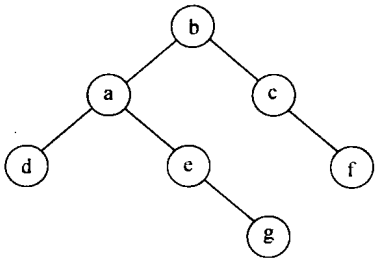


图 1.3　数据结构 Tree 对应的逻辑结构图

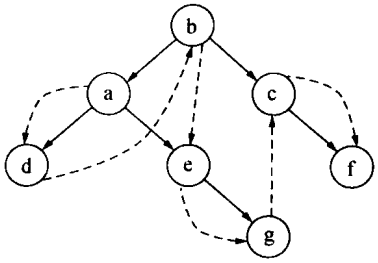


图 1.4　数据结构 Graph 对应的逻辑结构图

从该图可以看出，数据结构 Graph 表达的是一种带有两个关系的网状结构，其中 S_1 为树状结构（图中实线部分）；S_2 为线性结构（图中虚线部分）。

1.2.3　数据结构和数据类型

在高级程序设计语言中，每个常量、变量或表达式都有各自确定的数据类型。数据类型明显或隐含地规定了在程序执行期间各变量或表达式的取值范围、存储方式以及允许进行的运算。可以认为，数据类型是一个值的集合和定义在此集合上的一组操作的总称。例如整数类型（int），它的取值范围是[−32768, 32767]，定义在其上的一组操作为加、减、乘、除、取模等。

数据类型分为简单（或基本）数据类型和结构类型两种。简单类型中的每个数据不可再分，而结构类型是由简单类型按照一定规则构造而成，并且结构类型中还可以包含结构类型。因此，结构类型中的每种数据（即结构数据）可以分解为若干个简单数据或结构数据，每个结构数据仍可再分。

数据类型是和数据结构密切相关的一个概念。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”，而数据类型则可以看成是在程序设计中已经实现了的数据结构。在程序设计过程中，当需要引入某种新的数据结构时，就要借助编程语言所提供的数据类型来描述数据的存储结构。

1.3　存储实现与运算实现

存储实现的基本目的就是建立数据结构在计算机内部的表示或实现，包括数据以及逻辑关系的物理表达，也被称为存储结构或物理结构。存储结构不同于逻辑结构，它与所使用的计算机语言相关。归纳起来，数据结构有以下四种常用的存储结构类型。

1.3.1　顺序存储结构

该结构是将逻辑上相邻的结点存储在物理位置上也相邻的存储单元里，结点之间的逻辑关系则由存储单元的邻接关系来表达。由此得到的存储表示称为顺序存储结构（Sequential Storage Structure），主要用于线性的数据结构。而某些非线性的数据结构也可以通过线性化处理后进行顺序存储。顺序存储结构可以借助于计算机程序设计语言（例如 C/C++）提供的数组类型来描述。

顺序存储结构的主要特点有以下 3 点。

（1）结点只存储数据本身，结点之间的逻辑关系没有占用额外空间，因此存储空间利用率高。

（2）数据结构中的每个结点的存储地址可以通过公式直接计算。第 i 个结点的存储地址为：

$$L_i = L_0 + (i-1) * B$$

式中，L_0 为第一个结点的存储位置，B 为每个结点所占用的存储单位个数。

（3）对结点进行插入、删除运算时，会引起大量的结点移动，操作不便。

【例 1.5】　有一种数据结构 DS=(D, S)，式中

```
D={C, O, M, P, U, T, E, R}
S={<C,O>, <O,M>, <M,P>, <P,U>, <U,T>, <T,E>, <E,R>}
```

设第一个结点的存储地址为 3000，每一个结点所占用的存储单元个数为 1，该数据结构的存储结构如图 1.5 所示。

存储单元	C	O	M	P	U	T	E	R
地址	3000	3001	3002	3003	3004	3005	3006	3007

图 1.5　顺序存储结构

1.3.2　链式存储结构

该结构不要求逻辑上相邻的结点在物理上也相邻，结点间的逻辑关系是由附加的指针来表达的，指针指向结点的邻接结点，于是将所有的结点串联在一起。由此得到的存储表示称为链式存储结构（Linked Storage Structure），通常可以借助于计算机程序设计语言（例如 C/C++）提供的指针类型来描述。链式存储结构由两部分组成，一个是存储结点本身的值，称为数据域；另一个是存储该结点的各个后续结点的存储单元地址，称为指针域（可以有多个指针）。

链式存储结构的主要特点介绍如下。

（1）结点除了存储数据本身的值之外，还分配一部分空间用于存储结点之间的逻辑关系，因此相比顺序存储结构其存储空间利用率较低。

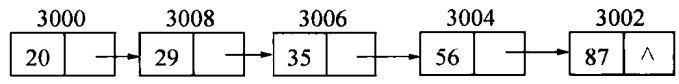
（2）逻辑上相邻的结点，物理上不必邻接，可用于线性表、树、图等多种逻辑结构的存储表达。

（3）对结点进行删除和插入操作比较灵活，不必移动结点，只需要改变结点中指针域的值即可。

【例 1.6】　假设有一种线性结构的结点集 D={20,87,56,35,29}，且第一个结点的存储地址为 3000，每一个结点所占用的存储单元个数为 2。现以结点的升序为逻辑关系 S={<20,29>, <29,35>, <35,56>, <56,87>}，则其对应的链接结构如图 1.6 所示。

地址	数据	指针
3000	20	3008
3002	87	∧
3004	56	3002
3006	35	3004
3008	29	3006

(a) 链式存储结构



(b) 线性结构的链式存储

图 1.6　线性结构的链式存储表达

1.3.3 索引存储结构

该结构是在存储结点数据的同时，建立一个附加的索引表。索引表中的每一项称作索引项，利用索引项的值来确定结点的实际存储单元地址，由此得到的存储表示称为索引存储结构（Index Storage Structure）。索引项的一般形式为（关键字，地址），关键字能唯一标识一个结点，如图 1.7 所示。

地址	关键字	指针
4000	023	220
4001	010	140
4002	027	300
4003	028	180
4004	024	260

(a) 索引表

地址	城市名	区号	说明
140	Beijing	010	首都
180	Chengdu	028	四川省省会
220	Chongqing	023	重庆直辖市
260	Shenyang	024	辽宁省省会
300	Wuhan	027	湖北省省会

(b) 结点表

图 1.7 索引存储结构

该图是城市的一种索引存储表示。在进行关键字查找时，可以先在索引表中快速查找到相应的关键字，然后通过指针（指向结点表中的地址）找到结点表中对应的结点。

索引存储结构的特点是检索速度快，可以直接对结点进行随机访问。在进行插入、删除操作时，只需要移动存储在索引表中对应结点的存储地址，而无需移动存储在结点表中的结点数据，因此效率较高。但由于增加了附加的索引表，导致存储空间利用率降低。

1.3.4 散列存储结构

该结构的基本思想是根据结点的关键字直接计算出结点的存储地址。把结点的关键字作为自变量，通过一个称作散列函数（Hash Function）的计算规则，确定出该结点的存储单元地址，然后将结点存入到此存储单元中，由此得到的存储表示称为散列存储结构（Hash Storage Structure）。该结构也可以用于检索。

我们仍以城市结构为例，以城市名为自变量 key，选用散列函数

```
H(key) = 1000+Sum(ASC(key)) mod 128
```

计算结点的存储地址，其中 Sum(ASC(key))表示对 key 的所有字符的 ASCII 值进行求和，mod 表示取模运算。计算结果如下：

key	Beijing	Chengdu	Chongqing	Shenyang	Wuhan
Sum(ASC(key))	696	702	926	829	515
H(key)	1056	1062	1030	1061	1003

于是得到如图 1.8 所示的城市结构的散列存储表示。

地址	城市名	区号	说明
1003	Wuhan	027	湖北省省会
≈			
1030	Chongqing	023	重庆直辖市
≈			
1056	Beijing	010	首都
≈			
1061	Shenyang	024	辽宁省省会
1062	Chengdu	028	四川省省会

图 1.8　散列存储结构

散列存储结构的特点是检索、增加和删除结点的操作速度快。由于该结构的核心是散列函数的设计，因此如果采用了设计不好的散列函数，有可能会出现结点单元的碰撞或冲突（如不同的结点关键字得到了相同的存储单元地址），此时就需要附加时间和空间开销（如进行二次散列处理），这样就会导致效率降低。

散列函数具有单向不可逆性，因此散列存储结构只适合于存储结点的数据，而不存储结点之间的逻辑关系。该存储方法一般只适合要求对数据能够进行快速查找、插入和删除等操作的场合。

1.3.5　运算实现

数据的运算是定义在数据的逻辑结构之上的，每一种逻辑结构都有一个运算的集合，例如插入、删除、修改等。这些运算实际上是在数据元素上施加的一系列抽象的操作。我们只知道这些操作要“做什么”，而无需考虑“如何做”，只有在确定了具体的存储结构之后，才会考虑如何去具体实现这些运算。也就是说，数据的运算是对数据定义的一组操作，是定义在逻辑结构上的，和存储结构无关。而运算的实现则依赖于存储结构。

下面简要地介绍几种数据的运算。

（1）查找：查找就是在数据结构里查找满足一定条件的结点，通常是给定某个字段的值，查找有该字段的结点。

（2）插入：向数据结构中增加新的结点。

（3）删除：从数据结构中删除指定的结点。

（4）修改：改变指定结点一个或多个字段的值。

（5）排序：按某种指定的顺序重新排列数据结构中的结点。

我们以栈为例，列出其常用的基本运算《具体实现参见第 3 章》。栈又称为堆栈，它是一种运算受限的特殊线性表，所受限制是仅允许在表的一端进行插入和删除运算。

（1）栈的初始化（InitStack）：建立一个空栈。

（2）判栈空（StackEmpty）：判断栈是否为空。

（3）入栈（Push）：向栈顶插入一个新元素。

（4）出栈（Pop）：删除栈顶元素。

（5）读取栈元素（GetTop）：读栈顶元素，但栈顶位置不变。

（6）置空栈（SetNull）：清空栈内元素。

运算是数据结构的一个重要方面，数据运算的实现是通过算法来描述的。讨论任何一种数据结构都离不开对该结构上的数据运算及其实现算法的分析。通常，我们在对一个数据结构进行某种运算时需要选择一个算法，如对于顺序存储的线性表的查找，我们可以采取顺序查找，也可以采取折半查找等。

1.3.6 进一步讨论

数据的逻辑结构、存储结构及运算这三个方面是一个整体，不能单一地去理解某一个方面，而应该注重它们之间的联系。

（1）存储结构是数据结构不可缺少的一个方面，同一个逻辑结构的不同存储结构可冠以不同的数据结构名称来标识。

【例 1.7】 线性表是一种逻辑结构，若采用顺序存储方法，可称其为顺序表；若采用链式存储方法，则可称其为链表；若采用散列存储方法，则可称为散列表。

（2）数据的运算也是数据结构不可分割的一个方面。在给定了数据的逻辑结构和存储结构之后，按定义的运算集合及其运算的性质不同，也可能导致完全不同的数据结构。

【例 1.8】 若对线性表上的插入、删除运算限制在表的一端进行，则该线性表称为栈；若对插入限制在表的一端进行，而删除限制在表的另一端进行，则该线性表称为队列。此外，若线性表采用顺序表或链表作为存储结构，则对插入和删除运算做了上述限制之后，可分别得到顺序栈或链栈，顺序队列或链队列。

1.4 算法描述和算法分析

1.4.1 什么是算法

算法一词，在中国最早出现在《周简》、《九章算术》中，其英文名称（Algorithm）出现在公元 825 年左右[2]，但算法思想早在 3500 年以前就诞生了。人们最熟悉的经典算法是公元前 300 年欧几里得关于求两个数的最大公约数的辗转相除法。1936 年英国数学家图灵（A.M.Turing）通过引进“图灵机”概念，给出算法概念严格的数学表达方法，并指出“算法可计算函数”，即“用图灵机可计算的函数”。

算法是计算机科学中的重要概念之一，是程序设计的精髓。程序设计的实质就是对要解决问题选择一个“好”的数据结构，同时在此结构上运用一种“好”的算法，并将其转化为计算机语言。也就是说，数据结构+算法=程序[3]。

关于算法的定义有多种表述方法。曾获图灵奖的著名科学家唐·欧·克努斯（D.E.Knuth）对算法的概念给出了一种描述：一个算法，就是一个有穷规则的集合，其规则确定了一个解决某一特定类型问题的运算序列。

[2] 算法英文一词最早来自公元 825 年左右，波斯数学家比阿勒·霍瓦里松（Al-Khwarizmi）的一本影响深远的著作《代数对话录》，算法之所以被拼写成“algorithm”，也是由于和算术（arithmetic）相关联。

[3] 见瑞士著名科学家尼古拉斯·沃斯（Niklaus Wirth）撰写的《Algorithms + Data Structures = Programs》一书。N.Wirth 因在程序设计语言方面的杰出贡献于 1984 年获得图灵奖。

本书中，认为算法是一组有穷的且定义明确的规则，它们规定了解决某一特定类型问题的一系列运算，是对解题方案的准确与完整的描述。通常一个问题可以有多个算法，一个给定的算法解决一个特定问题。作为一个算法，一般应具备下列五个基本特征。

（1）有穷性（Finiteness）[4]：一个算法必须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（Definiteness）：算法中的每一步都必须有确切的含义，且无多义性。

（3）可行性（Effectiveness）：算法中的每一步都是可行的，即都可以通过已经实现的基本操作执行有限次得以实现。

（4）输入（Input）：一个算法有零个或多个输入，在运算开始之前给出算法所需数据的初始值，这些输入取自特定的对象集合。另外，有些输入可以在算法执行过程中提供。

（5）输出（Output）：作为算法运算的结果，一个算法产生一个或多个输出，这些输出是同输入有某种特定关系的量。

要设计一个“好”的算法，除了具备上述五个基本特征之外，通常还应考虑如下几个要求。

（1）正确性（Correctness）：一个正确的算法是指在合理的数据输入下，能够正确地执行预先规定的功能和性能要求，在有限的运行时间内得出正确的结果。正确性是设计和评价一个算法的首要条件。

（2）可读性（Readability）：一个算法应该便于阅读和交流。好的可读性有助于人们对算法的理解，也有助于程序的调试和维修。在保证算法正确性的前提下，应强调算法的可读性。为了达到这个目的，需要有一个清晰的算法逻辑和保持良好的编程风格。

（3）健壮性（Robustness）：一个健壮（鲁棒性强）的算法应该能对不合理的输入进行检查和异常处理，以提高算法的容错性，减少出现异常中断或死机现象的概率。

（4）高效性（Efficiency）：指算法的执行效率高，包括时间效率和空间效率两个方面。时间效率是指算法执行所需要的时间，空间效率是指执行算法所需要的存储量。对于同一个算法，如果执行时间越短，所需存储量越小，则算法效率越高。效率和存储量这两者都与问题的规模有关。

1.4.2　算法描述

设计出来的算法需要用一种语言来描述，可以清楚地表达问题的求解步骤，一般分为自然语言、流程图、程序设计语言、伪代码等。

下面我们以判断一个任意给定的自然数 n 是否为素数为例，列出几种算法描述的说明。

数学依据：如果一个数 n 不能被任何从 2 到 n+1 的平方根之间的任何一个数整除，那么它就是一个素数。

1. 自然语言

用自然语言描述算法，最大的优点是容易理解，缺点是算法表达不够精准，容易出现二义性。算法描述如下：

（1）输入自然数 n

（2）判断 n 能否被 2 到 n+1 的平方根之间任意一个数整除

（3）如果能被整除，输出“非素数”信息；否则，输出“素数”信息

[4] 算法中有穷的概念不是纯数学的，而是指在实际应用中是合理的、可接受的。

（4）程序结束

2. 流程图

用流程图来描述算法，就是使用国际标准的流程图图形符号来表示算法的求解步骤，通常有ANSI 标准和 ISO 标准。图 1.9 所示为采用 ANSI 标准的流程图格式。

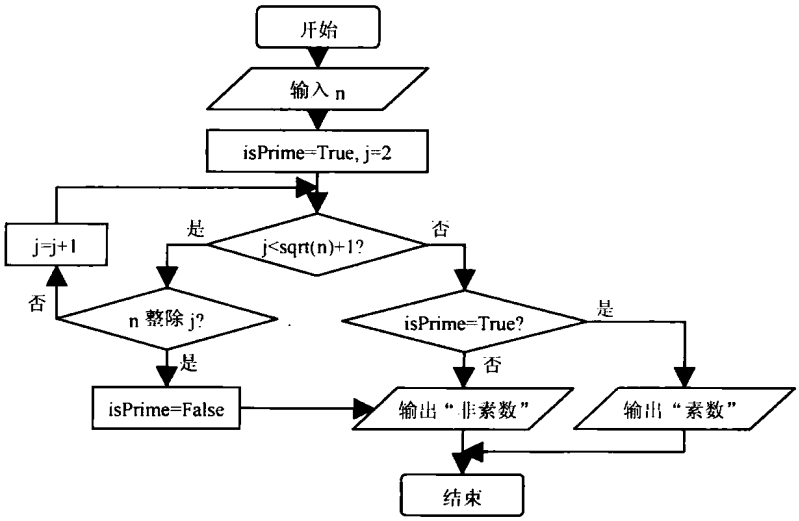


图 1.9 流程图描述算法

3. 程序设计语言

用程序设计语言来描述算法，就是直接用某种程序设计语言来表达算法的求解过程，其特点是算法描述准确、严谨、结构化程度高，通过编译器编译和链接生成机器代码后就可以直接运行，不足是细节过多，直观性差，常需借助程序注释才能明白算法的含义。常用于算法描述的程序设计语言有 Pascal，C，C++，Java 等，本书采用 C 语言作为描述算法的主要工具。本例的算法描述如下：

```
#include <stdio.h>
#include <math.h>
void main()
{
    int n, isPrime, j;
    printf("input a number n:");              /*输入一个自然数*/
    scanf("%d",&n);
    isPrime=1;                                /*素数标志初始化*/
    for(j=2;j<(int)sqrt(n)+1;j++) {           /*循环以便判断能否被整除*/
        if(n%j==0) {                          /*n 能被 j 整除*/
            isPrime=0;                        /*置非素数标志*/
            break;
        }
    }
    if(isPrime==1)
        printf("%d is a prime number.\n",n);        /*输出素数*/
    else
        printf("%d is not a prime number.\n",n);    /*输出非素数*/
}
```

4. 伪代码

伪代码介于程序设计语言和自然语言之间，它忽略了程序设计语言中的一些严格语法规则与描述细节，可以采用英文和中英文混合书写，以便把注意力主要集中在算法处理步骤的描述上。