数据结构（C++版）

（第2版）

陈宝平 张巨萍 孙宝军 阿雅娜 主编

清华大学出版社

数据结构（C++版）

（第2版）

陈宝平 张巨萍 孙宝军 阿雅娜 主编

清华大学出版社
北京

内 容 简 介

数据结构是计算机专业教学计划中的核心课程，也是计算机及相关专业考研和水平等级考试的必考科目。要从事和计算机科学与技术相关的工作，尤其是计算机应用领域的开发和研制工作，必须具备坚实的数据结构基础。本书介绍了学习数据结构所用到的预备知识，叙述了数据结构、算法以及抽象数据类型的概念，介绍了线性表、栈、队列和串、数组和广义表、树和二叉树、图等常用数据结构，讨论了常用的查找、排序和索引技术。

本书内容丰富，层次清晰，讲解深入浅出，可作为计算机及相关专业本专科数据结构课程的教材，也可供从事计算机软件开发和应用的工程技术人员阅读、参考。

图书在版编目(CIP)数据

数据结构：C++版/陈宝平等主编. --2 版. --北京：清华大学出版社，2016
21 世纪高等学校规划教材 · 计算机科学与技术
ISBN 978-7-302-42235-8

Ⅰ. ①数… Ⅱ. ①陈… 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(2015)第 278879 号

责任编辑：闫红梅
封面设计：傅瑞学
责任校对：时翠兰
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 装 者：三河市少明印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：16.25　　**字　　数**：393 千字
版　　次：2012 年 2 月第 1 版　2016 年 6 月第 2 版　　**印　　次**：2016 年 6 月第1 次印刷
印　　数：1～2000
定　　价：29.00 元

产品编号：067617-01

目 录

第1章 绪论

由唐纳德·克努特(Donald Ervin Knuth)所著的洋洋数百万言的多卷本《计算机程序设计的艺术》(*The Art of Computer Programming*)堪称计算机科学技术的经典巨著。其第一卷《基本算法》于1968年出版,全面、系统地阐述了数据的逻辑结构和存储结构,为数据结构奠定了基础。随后一些大学开始将"数据结构"作为一门课程讲授,并且列入了教学计划。迄今为止,"数据结构"的研究和发展已经经历了数十载。期间,尼克劳斯·沃斯(NikLaus Wirth)编写的《算法+数据结构=程序》(*Algorithms+Data Structure=Programs*)、约翰·霍普克罗夫特(John Edward Hopcroft)编写的《数据结构和算法》(*Data Structure and Algorithms*)等都为数据结构的发展做出了卓越的贡献。如今,虽然数据结构理论已经较为成熟,但随着计算机技术的发展,对于它的研究从未停止过。如何将数据结构应用于专业领域的研究还在不断发展中,如多维图形数据结构、动画数据结构、音频数据结构等。

1.1 为什么要学习数据结构

第一台得到应用的计算机ENIAC(Electrical Numerical Integrator And Calculator,电子数字积分计算机)于1946年在美国诞生,第一家计算机研发和生产的企业埃克特-莫奇莱计算机公司(EMCC)于1948年成立,第一个成功的高级程序设计语言FORTRAN在1957年开始商业发行,第一个商业成功的文字处理软件——WordStar在20世纪80年代风行于全球,第一个专门用于网格计算的数据库——Oracle数据库在1983年发布了具有可移植性的Oracle第3版,1991年开始,Internet真正开始商业化运行。计算机渐渐渗透到现实世界的各个领域当中,汹涌而来的信息数据使人无所适从,管理它们、使它们井井有条、易于使用变得非常困难。这些信息数据不再仅仅局限于简单的数值,还包括更为复杂的字符、图像、表格等。为了更好地用计算机处理这些具有一定组织结构的数据,就必须分析它们的特性、相互关系及相应的存储方式,在这个基础上设计出相应的算法和程序,这正是数据结构这门课程的研究内容。

从课程体系来说,数据结构是一门综合性很强的专业基础课,不仅是计算机专业教学计划中的核心课程之一,而且还是其他非计算机专业的主要选修课之一。它是介于数学、计算机硬件和计算机软件三者之间的一门核心课程,是计算机系统软件和应用软件研制者的必修课程。在数学方面,数据结构这个概念包含了图论、树、网络、集合论、关系等现在离散数学中讲述的内容;在硬件方面,编码理论、存储装置和存储方法等与数据结构有着密切的关

系；在软件方面，计算机应用软件和系统软件如操作系统、编译系统和数据库等都需要很多数据结构相关知识，如编译技术要使用栈，操作系统会用到队列、存储管理，数据库系统将运用线性表、散列表、链表以及索引技术等。

本书中介绍了一些数据结构中最常用的数据结构，阐明数据结构的逻辑关系，分析它们在计算机中的存储表示，并结合各种数据结构讨论可以实现的算法。通过对本书的学习，结合实际编程练习，读者将有能力根据实际问题选择恰当的数据结构以及控制相应算法的效率。

1.2 什么是数据结构

在日常生活中，人们会遇到各种信息，如在路上进行交通指挥的交警的手势，人们通过电子邮件或聊天工具交流的思想，顾客在商场购买物品后所做的记录等。这些信息必须转换成数据才能在计算机中进行处理。因此，在学习数据结构这门课程之前，需要定义和说明一些术语。

数据(data)是信息的载体，是描述客观事物的数、字符，以及所有能输入到计算机中并被计算机程序识别和处理的符号的集合。我们平时所遇到的数据主要有两类，一类是数值型数据，包括整数、实数、复数，主要在工程和科学计算，以及商业事务处理中使用；另一类是非数值数据，主要包括文本、表格、图形和语音等数据。

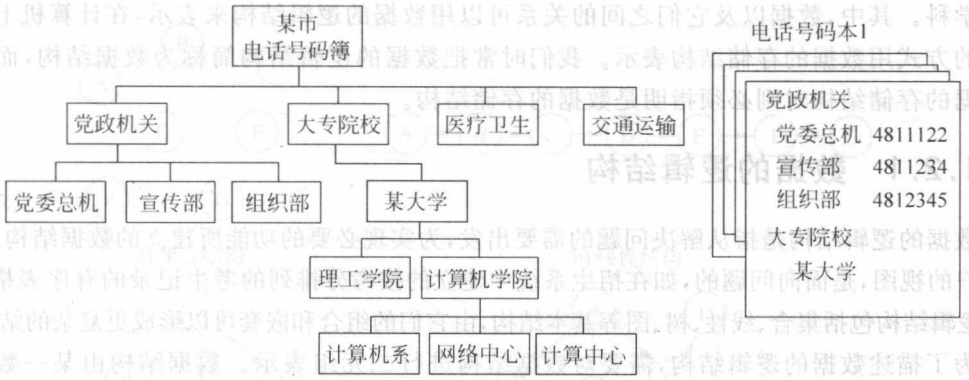
数据元素(data element)简称元素，在用计算机进行处理的时候通常将它作为一个相对独立的单位。如对于一个二维表格数据来说，每行信息就是它的数据元素；对于一个字符串数据来说，每个字符就是它的数据元素；对于一维数组来说，每个下标位置所存储的值就是它的数据元素。一个数据元素通常由一个或多个数据项组成，每个数据项可以是简单数据项，即不可再分，如一个数值、字符等；也可以是组合数据项，如一个字符串、数组、记录、对象等。数据和数据元素是相对而言的，如对于一条记录信息来说，它是所属文件的一个数据元素，而它相对于所含的数据项而言又可看成数据。

数据处理(data processing)是指利用计算机对数据进行处理的过程，这里所指的处理可以是存储、检索、插入、删除、排序、统计、输入和输出等。

1.1节提到随着计算机应用领域的扩大，计算机所处理数据的结构越来越复杂，非数值计算问题显得越来越重要。据统计，目前85%以上的机器时间用于处理非数值计算性问题。下面列举一些具体问题。

【例1.1】 通常情况下，在制作一个电话号码簿时，会按照组织机构来编排，如图1.1(a)所示。这种编排形式的优点是，在确定要查找的号码隶属于哪个部门的情况下，可以迅速地查找出电话号码。但是，知道某个电话号码，想查找出它属于哪个部门，就不容易了。所以电话号码簿的编排可以有另一种形式，即按照电话号码的大小进行排列，标注出其隶属部门，如图1.1(b)所示。数据的安排直接影响到工作效率。

【例1.2】 通信网络问题。假设要在n个城市之间建立通信联络网，如图1.2所示，图中用○表示城市，两个○之间连线上的数字表示在两个城市之间铺设通信线路所需要的代价。连通n个城市最少需要n−1条线路，在建立通信网络的时候需要在众多线路中选出最小代价并且能连通n个城市的n−1条线路。



(a) 按照组织机构来排列电话号码

(b) 按照电话号码大小来排列电话号码

图 1.1 电话号码簿排列图

【例 1.3】 人机对弈问题。假设 A 为玩家，B 为电脑，A 与 B 下棋对弈。首先 A 下棋时对应局面是 1，由此局面出发 B 有 n 种符合规则的方式下棋，走棋后对应的局面为 2，3，4，…，n+1；如果 B 选择了形成局面 2 的方式走棋，则 A 在下一步有 m 种符合规则的方式下棋，对应的局面为 I+1，I+2，…，I+m；如果 A 选择了形成局面 I+1 的方式走棋，B 也有若干种符合规则的走棋方式，以此类推。这样 A 和 B 轮流下棋，棋盘局面的发展变化如图 1.3 所示，这种关系可以由一种叫作树的数据结构来表示。

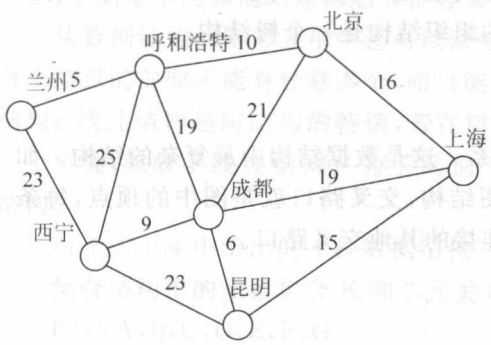


图 1.2 通信网络图

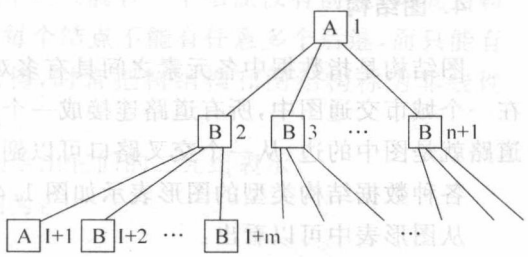


图 1.3 博弈树

通过以上 3 个实例可以清楚地认识到，用简单的数值和数学方程已经不足以表示和计算这类非数值问题，而要使用更为复杂的图、表和树之类的数据结构。所以，简单来讲，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操

作的学科。其中,数据以及它们之间的关系可以用数据的逻辑结构来表示,在计算机上进行存储的方式用数据的存储结构表示。我们时常把数据的逻辑结构简称为数据结构,而在讨论数据的存储结构时则必须指明是数据的存储结构。

1.2.1 数据的逻辑结构

数据的逻辑结构是指从解决问题的需要出发,为实现必要的功能所建立的数据结构,它属于用户的视图,是面向问题的,如在招生系统中建立的按考分排列的考生记录的有序表格。数据的逻辑结构包括集合、线性、树、图等基本结构,由它们的组合和嵌套可以形成更复杂的结构。

为了描述数据的逻辑结构,需要对数据结构进行二元组表示。数据结构由某一数据对象及该对象中所有数据成员之间的关系组成,记为:

$$DataStructure=\{D,R\}$$

其中,D 是某一数据对象,R 是该对象中所有数据成员之间的关系的有限集合。

1. 集合

集合结构是指数据中各元素之间没有任何次序。如一个容器中的所有乒乓球,一个俱乐部里的所有成员,可以认为它们之间没有任何次序,它们均为集合结构。

2. 线性结构

线性结构是指数据中各元素之间具有 1 对 1 的先后次序关系。如在一个列车时刻表中,各车次记录之间是按照发车时间的先后次序排列的;在一个人事职工表中,各职工记录之间是按照职工编号的先后次序排列的。所以,它们的表结构都是线性结构。

3. 树结构

树结构是指数据中各元素之间具有 1 对多的先后次序关系,并且只有一个元素称为树根结点,其余均为树枝结点和树叶结点。如在一个企业的组织机构中,总经理只有一个,相当于树根;其下属多个部门,每个部门又各有一个部门经理,相当于树枝;每个部门又有多名员工,归部门经理领导,相当于树叶。所以,企业的组织结构是一个树结构。

4. 图结构

图结构是指数据中各元素之间具有多对多的关系。这是数据结构中最复杂的结构。如在一个城市交通图中,所有道路连接成一个复杂的图结构,交叉路口就是图中的顶点,每条道路就是图中的边,从一个交叉路口可以到达与其连接的其他交叉路口。

各种数据结构类型的图形表示如图 1.4 所示。

从图形表中可以看出:

(1) 集合结构中的元素是各自独立的,元素之间没有联系。

(2) 线性结构中的元素是一个接一个串联起来的,它有一个头元素 A 和一个尾元素 G,其余为中间元素;每个中间元素既有前驱元素,又有后继元素,如 B 的前驱元素为 A,后继元素为 C;C 的前驱元素为 B,后继元素为 D,……,头元素 A 没有前驱元素,只有后继元素 B;尾元素 G 只有前驱元素 F,没有后继元素。

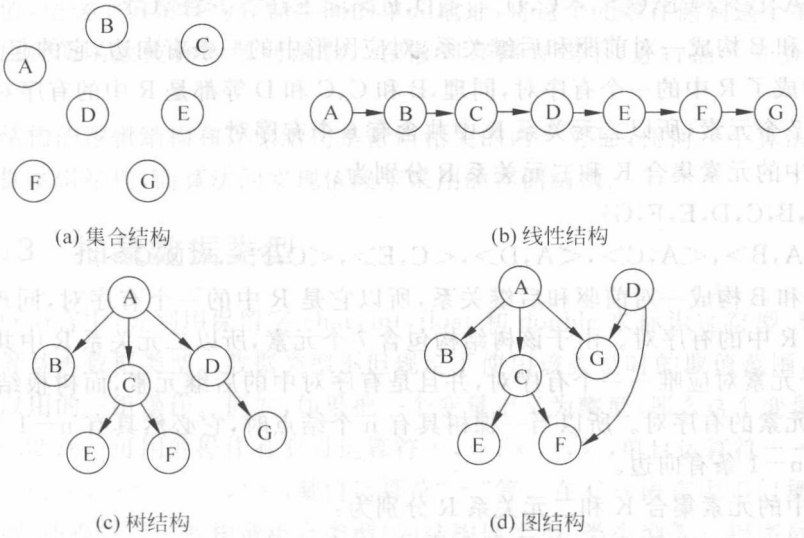


图 1.4 各种数据结构类型的图形表示

(3) 树结构的图形表示像倒着画的一棵树，树中有一个树根元素 A，它有 3 个后继元素，又称为 A 的孩子结点 B，C 和 D，C 结点有两个孩子 E 和 F，D 结点有一个孩子 G，由于 B，E，F，G 没有孩子，所以称它们为叶子结点，而 A，C，D 被称为树枝结点或分支结点，同时 A 又是唯一的一个树根结点。在树结构中，树根结点只有后继结点，而没有前驱结点。如 A 为树根结点，它没有前驱结点，或者说其前驱结点为空，它的后继结点为 B，C 和 D。除树根结点外，每个结点都有唯一一个前驱结点，又称为父结点或双亲结点。如 C 的前驱结点为 A，G 的前驱结点为 D，每个结点的前驱结点即双亲结点，从图形中都能够很容易得到。

(4) 在图结构中，每个结点或称顶点都可以有任意多个前驱结点和任意多个后继结点。如图 1.4(d)所示，顶点 A 没有前驱结点，或者说它有 0 个前驱结点，A 有 3 个后继结点 B，C 和 G；G 有 2 个前驱结点 A 和 D；E 有 1 个前驱结点 C 和 0 个后继结点，或者说，E 没有后继，对于图形中的其他结点都能够很容易得到其前驱和后继结点。

从数据结构的图形表示中还可以清楚地看出：树结构是图结构的特例，若在图结构中，每个结点的前驱不能有任意多个，而只能有一个，并且只能有一个结点没有前驱，则成为树结构；线性结构是树结构的特例，若在树结构中，每个结点不能有任意多个后继，而只能有一个后继，就成了线性结构。为了区别于线性结构，时常把树结构和图结构称为非线性结构。

对于图 1.4 中给出的 4 种数据结构，下面分别给出它们的二元组表示。

集合结构中的元素集合 K 和二元关系 R 分别为：

K={A,B,C,D,E,F,G}

R={ }

因为集合中的元素为孤立顶点，它们之间没有前驱和后继的关系，所以对应的二元关系为空。

线性结构中的元素集合 K 和二元关系 R 分别为：

K={A,B,C,D,E,F,G}

R={<A,B>,<B,C>,<C,D>,<D,E>,<E,F>,<F,G>}

因为A和B构成一对前驱和后继关系,对应图形中的一条有向边,它的起点为A,终点为B,它就构成了R中的一个有序对,同理,B和C、C和D等都是R中的有序对。由于该线性结构包含7个元素,所以二元关系R中共含有6个有序对。

树结构中的元素集合K和二元关系R分别为:

K={A,B,C,D,E,F,G}

R={<A,B>,<A,C>,<A,D>,<C,E>,<C,F>,<D,G>}

因为A和B构成一对前驱和后继关系,所以它是R中的一个有序对,同理,A和C、A和D等都是R中的有序对。由于该树结构包含7个元素,所以二元关系R中共含有6个有序对,即每个元素对应唯一一个有序对,并且是有序对中的后继元素,而树根结点没有对应的作为后继元素的有序对。所以当一棵树具有n个结点时,它必然具有n−1个有序对,对应图形中有n−1条有向边。

图结构中的元素集合K和二元关系R分别为:

K={A,B,C,D,E,F,G}

R={<A,B>,<A,C>,<A,G>,<D,G>,<D,F>,<C,E>,<C,F>,<G,F>}

在线性结构和树结构中,若元素个数为n,则有序对个数必然为n−1。而在图结构中不存在这种对应的关系,也就是说,其有序对的个数又称有向边数可能大于、等于或小于n−1。在这个图结构的例子中,元素个数为7,边数为8。

1.2.2 数据的存储结构

数据结构在计算机中的表示(又称映像)称为数据的存储结构,又称为物理结构。它包括数据元素的表示和关系的表示。对于二元组表示的数据逻辑结构{D,R}而言,其数据的存储结构就是建立一种由逻辑结构到物理存储空间的映射。同一种逻辑结构可以采用不同的存储结构来进行存储。常用的存储方法有顺序存储方法、链接存储方法、索引方法和散列方法等。

顺序存储方法是把逻辑上相邻的结点存储在物理位置相邻的存储单元里,结点间的逻辑关系由存储单元的邻接关系来体现,由此得到的存储表示称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法,通常借助于程序设计语言中的数组来实现。

链接存储方法不要求逻辑上相邻的结点在物理位置上亦相邻,结点间的逻辑关系是由附加的指针字段表示,由此得到的存储表示称为链式存储结构。链接存储结构中数据结点由数据域(用于存储结点本身的数据)和指针域(存放指向其后继结点或前驱结点的指针)组成。

索引方法是顺序存储结构的一种推广,通常在存储结点信息的同时要建立一个索引表,表中的每一项被称为索引项。索引项由关键字和地址组成,其中关键字用于唯一标识一个数据结点。索引方法要构造一个映射函数,即把关键字映射到存储地址的函数,这个函数被称为索引函数。索引表的存储空间是附加在结点存储空间之外的。索引方法在程序设计中是一种较为常用的方法,对于非顺序存储的结构来说,利用索引方法是可以快速地由关键字找到其对应数据结点的唯一方法。

散列存储的基本思想是,以集合或线性表中的每个元素的关键字为自变量,通过函数计

算出函数值，把这个值解释为存储空间的单元地址，将这个元素存储到这个单元中。在这个过程中所用到的函数被称为散列函数。在散列存储方式下，进行插入和查找操作是非常快的。

数据结构的逻辑结构和存储结构是密切相关的两个方面，任何一个算法的设计取决于选定的数据逻辑结构，而算法的实现依赖于采用的存储结构。

1.2.3 抽象数据类型

在C++语言中，分别用保留字char，int，float和double来标识字符型、整型、浮点型和双精度型等基本数据类型。数据类型不但规定了使用该类型时的取值范围，而且还规定了该类型可以用的一组操作。例如，如果把一个变量定义为整型，那么这个变量的取值范围是−32 768～32 767，可用的操作有双目运算符+，−，*，/，%，单目运算符++，−−，关系运算符<，>，<=，>=，==，!=，赋值运算符“=”等。在C++语言中不但规定了一些基本的数据类型，还提供了一些构造组合类型（如结构体类型、类类型等），程序员可以利用这些规则自行定义所需的数据类型来解决应用问题、确切地描述数据对象和正确地进行相关计算。

在一般数据类型的基础上，可以定义抽象数据类型（Abstract Data Type，ADT），它的含义比一般数据类型更广、更抽象，由一种数据结构和在其上进行的一组运算所组成。抽象数据类型包括定义和实现两部分，其特征是定义与实现分离，实行封装和信息隐蔽。首先根据问题的要求，定义该抽象数据类型需要包含的信息，确定可以进行的操作。使用者可以通过这些操作来使用该抽象数据类型。另一方面，抽象数据类型的物理实现部分封装在其实现模块内，使用者不能看到，也不能直接操作该类型所存储的数据。这样使用者可以在开发过程中抓住重点，集中精力考虑如何解决应用问题，使问题得到简化。把抽象数据类型的物理实现封装起来，有利于编码、测试，也有利于将来的修改，并且可以使得错误局部化，一旦出现错误，不至于影响其他模块。如果为了提高效率希望改进数据结构，可能需要改变抽象数据类型的物理实现，但只要该抽象数据类型所提供的使用方式不变，其他所有使用该抽象数据类型的程序都可以不变，从而大大提高系统的稳定性。

抽象数据类型在C++语言中是通过类类型来实现的，其数据部分通常定义为类的私有（private）或保护（protected）的数据成员，它只允许给该类或派生类直接使用。操作部分通常定义为类的公共（public）的成员函数，它既可以提供给该类或派生类使用也可以提供给另外的类或函数使用。操作部分只给出操作说明（即函数声明），操作的具体实现通常在一个单独文件中给出，使它与类的定义（即声明）相分离，当然在编译时将被连接在一起。类的声明通常存放在一个专门的头文件（其扩展名为.h）中，这样能够较好地实现信息的隐藏和封装，符合面向对象程序设计（Object-Oriented Programming，OOP）的思想。

在本书中，描述每一种抽象数据类型将采用如下书写格式：

```
ADT <抽象数据类型名>{
    Data:
        <数据元素集合>
    Structure:
        <数据关系集合>
```

```
    Operations:
        <操作声明>
};
```

【例 1.4】 把圆柱体定义及其运算设计成一种抽象数据类型,其数据部分包括圆柱体的半径和高,操作部分包括初始化圆柱体、求面积函数和求体积函数等。

假定该抽象数据类型名用 Cylinder(圆柱体)表示,定义圆柱体的半径和高度的数据用 radius 和 height 表示,并假定其类型为单精度浮点型(float),初始化圆柱体的函数名用 InitCylinder 表示,求圆柱体面积的函数名用 Area(面积)表示,求圆柱体体积的函数名用 Volume(体积)表示,则圆柱体的 ADT(抽象数据类型)描述如下:

```
ADT Cylinder{
  Data:
      radius, height;
  Structure:
      R = { DR }
      DR = { radius, height∈float 且通过 radius, height 唯一确定一个圆柱体 }
  Operations:
      void InitCylinder (struct Cylinder c, float r, float h);
      float Area (struct Cylinder c);
      float Volume (struct Cylinder c);
};
```

该类型用 C++语言定义如下:

```
class Cylinder
{
  float radius, height;
public:
  Cylinder ();
  Cylinder (float len,float wid);
  float Area ();
  float Volume ();
};
```

1.3 算法与算法分析

1.3.1 算法

算法是解决特定问题的方法,是一个有穷的指令集。可以用多种方式来描述算法,如流程图、自然语言、N-S 图和程序语言等。在本书中采用 C++语言描述,它的优点是类型丰富,语句精练,具有面向过程和面向对象的双重特点,编出的程序结构化程度高,可读性强。为了更好地说明算法的思路,有时还采用 C++语句与自然语言结合的方式来描述算法。一个算法应当具有以下特性。

(1) 有输入。一个算法必须有 0 个或多个输入。这些输入属于特定的对象集合,它们可以使用输入语句从外部输入,也可以在算法内用给定初值或赋值的方法给定。

(2) 有输出。当一个算法结束的时候,应有一个或多个输出,作为算法操作的结果。

(3) 确定性。算法的每一步都应具有确切的含义,没有歧义。

(4) 有穷性。一个算法应该在执行有穷步后结束,没有形成死循环的情况发生。

(5) 可行性。算法中每一条运算都应该是已经实现的基本运算,原则上都能精确地执行。

算法和程序不同,程序可以不满足上述特性。例如,一个操作系统在用户未使用前一直处于"等待"的循环中,直到出现新的用户事件为止。这样的系统可以无休止地运行,直到系统停工。

下面举一个查找的例子,说明如何把一个具体问题转变为一个算法。所使用的方法是自顶向下、逐步求精的结构化程序设计方法。

第一步,我们要搞清楚需要解决什么问题。在此例中,我们的任务是要从存放在一个整数数组中的 n 个已经按从小到大的顺序排列好的记录中查找等于给定值的记录的位置。

第二步,考虑问题解决方案。n 个数据存放在数组 a[0]到 a[n−1]中,要进行查找,可以采用二分查找算法。其基本查找过程是:首先取整个有序表 a[0]~a[n−1] 的中点 a[mid](其中 mid=(n−1)/2)的值与给定值进行比较,若相等,则查找成功,返回该记录的下标位置;若 a[mid]的值大于给定值,则表明待查值落在右子表 a[mid+1]~a[n−1]中,接着只需要在右子表中继续进行二分查找即可;若 a[mid]的值小于给定值,则表明待查值落在左子表 a[0]~a[mid−1]中,接着只需要在左子表中继续进行二分查找即可;这样,每经过一次比较,查找空间就会缩小一半,如此进行下去,直到找到等于给定值的数组元素,或者当前查找区间为空为止(即查找失败)。

第三步,根据以上思路,写出算法框架。因为在查找过程中会修改查找区间,因此令 low 表示区间下界,high 表示区间上界,mid 表示区间的中点,k 为待查找值。初始时,查找区间为整个数组空间,所以 low=0,high=n−1。每次通过 low 与 high 确定区间中点 mid,且令 a[mid]与 k 进行比较,根据判断结果做出不同的操作,包括修改查找区间或返回下标程序结束等操作。算法如下:

```
while(查找区间不为空)
{
  mid = (low + high)/2;
  if(a[mid] = k) 返回下标,算法结束;
  else if(a[mid]> k) 修改区间下界 low = mid + 1;
  else 修改区间上界 high = mid - 1;
}
```

第四步,将其细化。这里要解决的问题是,在操作过程中如何判断查找区间为空。我们知道,查找区间上界和下界之间是有限制关系的,即下界一定不大于上界。因此如果在操作过程中出现下界大于上界的情况,就知道查找区间已经不合法了,因此可用这个条件来断定查找区间是否为空。

综合以上做法,即可得到取名为 Binsch 的查找过程如下:

```
int Binsch ( int a[ ], int k) {
//从 n 个整数 a[0],a[1],…,a[n-1]中,查找等于给定值的元素,返回下标
    int low = 0, high = n - 1 ;
```