全国计算机等级考试指定教材配套辅导

新大纲

National Computer Rank Examination

全国计算机等级考试

考点解析与上机辅导

（二级C程序设计）

曹德胜 等 编著

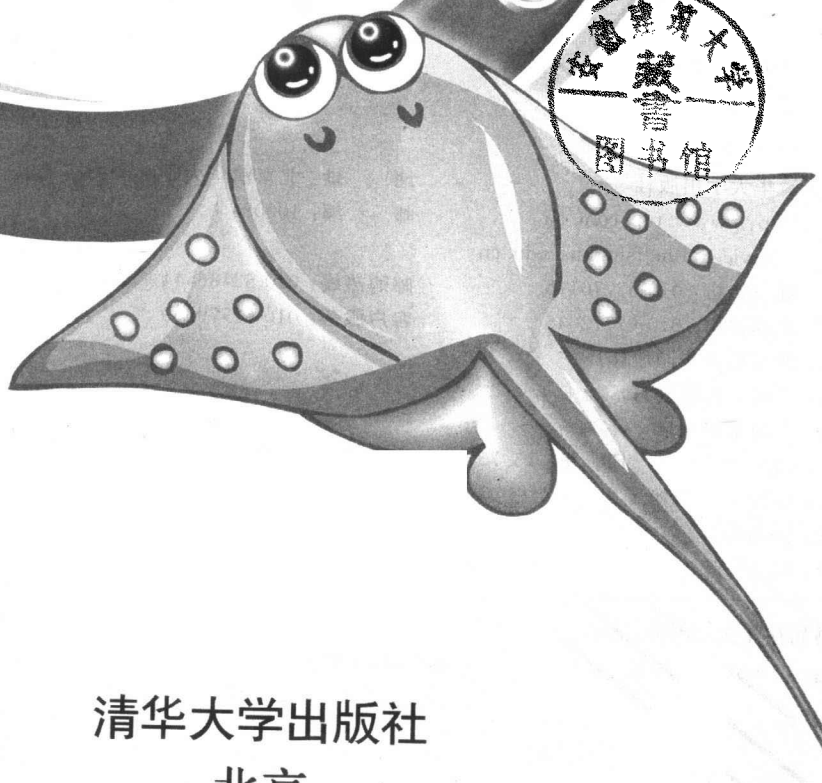
- 与指定教程**完全同步**
- 突出**考试重点、难点**及经验分析
- 大量历年**真题和典型习题**
- 总结**应试技巧**及高分策略

清华大学出版社

National Computer Rank Examination

全国计算机等级考试
考点解析与上机辅导
（二级C程序设计）

曹德胜 等 编著



清华大学出版社
北京

内 容 简 介

本书分为10章，其中，第1章为基本数据结构与算法，第2章为程序设计基础，第3章为软件工程基础，第4章为数据库设计基础，第5章为C语言基础知识，第6章为程序填空题，第7章为C程序修改，第8章为C程序编程题，第9章为全国计算机等级考试二级考试模拟试卷，第10章为上机模拟试题。

全书以2004年的新大纲为蓝本，针对等级考试的特点，精心策划，准确定位，概念清晰，例题丰富，深入浅出，它是等级考试者必备辅导丛书之一。书中每一部分都有大量的习题，这些习题都来自历年计算机等级考试笔试及上机题，而且所选题目精练，每道题都有相应的分析。全书结构如下：前4章为计算机基础知识，第5、9章为笔试部分，第6、7、8、10章为上机分析。另外，本书还总结了计算机基础知识、C语言基础知识，有助于参加全国计算机二级考试的考生有的放矢地复习，每部分都配有大量的练习题，以提高考生的笔试、上机能力。

本书可作为各类“全国计算机等级考试”培训班、补习班的辅导用书，也适于参加C语言考试的人员自学使用。

图书在版编目(CIP)数据

全国计算机等级考试考点解析与上机辅导（二级C程序设计）/曹德胜等编著. —北京：清华大学出版社，2007.2
ISBN 978-7-302-14581-3

Ⅰ. 全… Ⅱ. 曹… Ⅲ. ①电子计算机-水平考试-自学参考资料 ②C语言-程序设计-水平考试-自学参考资料
Ⅳ. TP3

中国版本图书馆CIP数据核字（2007）第013340号

责任编辑：冯志强　刘　霞
责任校对：张　剑
责任印制：何　芊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线**：010-62786544
投稿咨询：010-62772015　　**客户服务**：010-62776969
印 刷 者：北京市清华园胶印厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：210×285　　**印　　张**：15.75　　**字　　数**：502千字
版　　次：2007年2月第1版　　**印　　次**：2007年2月第1次印刷
印　　数：1～5000
定　　价：25.00元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。
联系电话：010-62770177转3103　　产品编号：024263-01

前言 Preface

随着社会经济的发展与科学技术的进步，特别是近几年来全球IT产业的迅猛发展，计算机技术对我国人民的生活和工作产生了越来越深的影响。培养计算机应用人才，仅靠大学里少数的计算机专业的学生是远远不够的，还要靠大量的非计算机专业人员。高等学校非计算机专业学生的计算机教育是高等教育的组成部分，是实现高等教育培养目标的一个重要环节。

很多人都希望取得全国计算机等级证书，以证明自己的计算机知识与应用能力。为了满足广大读者的愿望，作者通过对历年计算机等级考试试题的分析和研究，特别是针对2004年新大纲的出台及考试方式的改革，作者编写了本书希望对读者复习和应试有所帮助。

本书的内容主要包括以下几个部分。

第1章为基本数据结构与算法，包括算法、数据结构、线性表、栈和队列、线性表的链式存储结构、树、查找与排序等内容，最后有解析题范例和练习题。

第2章为程序设计基础，包括程序设计方法与风格介绍、结构化程序设计、面向对象的程序设计，最后有解析题范例和练习题。

第3章为软件工程基础，包括软件工程、结构化分析方法、结构化设计方法、软件测试的方法、程序调试的内容，最后有解析题范例和练习题。

第4章为数据库设计基础，包括数据库的基本概念、数据模型、关系代数、数据库设计等内容，最后有解析题范例和练习题。

第5章为C语言基础知识，包括C语言的数据类型，运算符、表达式及位运算，C程序的基本语句及结构，文件及预处理命令等内容，最后有解析题范例和练习题。

第6章为程序填空题，包括程序填空题的基本做法，程序填空题举例。

第7章为C程序修改，分基础知识、选择与循环、数组与指针、综合题几类介绍。

第8章为C程序编程题，分选择和循环结构、数组、字符串处理几类介绍。

第9章为全国计算机等级考试二级考试模拟试卷，包括新大纲后的历年全国计算机等级考试试卷模拟。

第10章为上机模拟试题，包括上机考试系统简介、上机考试应试技巧、全国计算机等级考试上机试题详解、上机模拟试卷。

在全书的最后附有全国计算机等级考试公共基础知识大纲和C语言考试大纲。

本书有利于考生在较短时间内强化自己的计算机知识，为通过等级考试增添信心和把握。

本书由曹德胜、任占营任主编，巫新建、时光、王晔任副主编，其中，任占营完成了第1、2章的编写任务，巫新建完成了第3章的编写任务，时光完成了第4章的编写任务，王长利完成了第5章的编写任务，王晔完成了第6、7章的编写任务，曹德胜完成了第8，9章和附录的编写任务，王长利完成了第10章的编写任务，全书最后由曹德胜统稿。赵爱华、曹达彬在全书的编写过程中给予了极大的支持和帮助，在此一并表示感谢。

由于作者水平有限，加之计算机技术发展的十分迅速，本书难免会有不足之处，敬请广大读者批评指正。

邮箱：cds@ncist.edu.cn。

编　者

2006.11

目录 Contents

第1章

基本数据结构与算法

1.1 算　法

1.1.1 算法的基本概念

算法（Algorithm）是对特定问题求解步骤的一种描述，它是指令的有限序列，其中，每一条指令表示一个或多个操作。一个算法通常具有下列5个重要特性。

（1）有穷性：一个算法必须总是（对任何合法的输入值）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：算法中每一条指令必须有确切的含义，读者理解时不会产生二义性。并且，在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。

（3）可行性：一个算法是可行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

（4）输入：一个算法有零个或多个输入，这些输入取自于某个特定的对象的集合。

（5）输出：一个算法有一个或多个输出。这些输出是同输入有着某些特定关系的量。

1.1.2 算法复杂度

1．时间复杂度

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；进行简单操作的次数越多，其运行时间也就相对地越多。因此，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度，它是一个算法运行时间的相对量度。

2．空间复杂度

算法在运行过程中所占用的存储空间的大小被定义为算法的空间复杂度。算法的空间复杂度比较容易计算，它包括局部变量所占用的存储空间和系统为了实现递归（如果是递归算法）所使用的堆栈这两个部分。

1.2 数据结构

1.2.1 数据结构的定义

数据结构是研究数据元素之间抽象化的相互关系和这种关系在计算机中的存储表示（即所谓数据的逻辑结构和物理结构），并对这种结构定义相适应的运算，设计出相应的算法，而且确保经过这些运算后所得到的新结构仍然是原来的结构类型。

通常把数据的逻辑结构统称为数据结构，把数据的物理结构统称为存储结构。

1.2.2 数据结构的图形表示

在任何问题中，数据元素都不是孤立存在的，它们之间存在着某种关系，这种数据元素相互之间的关系称为结构。根据数据元素之间关系的不同特性，通常有下列四类基本结构。

- 集合：结构中的数据元素之间除了“同属于一个集合”的关系外，别无其他关系。
- 线性结构：结构中的数据元素之间存在一个对一个的关系。

- 树形结构：结构中的数据元素之间存在一个对多个的关系。
- 图形结构或网状结构：结构中的数据元素之间存在多个对多个的关系。

图 1-1 所示为上述四类基本结构的关系图。

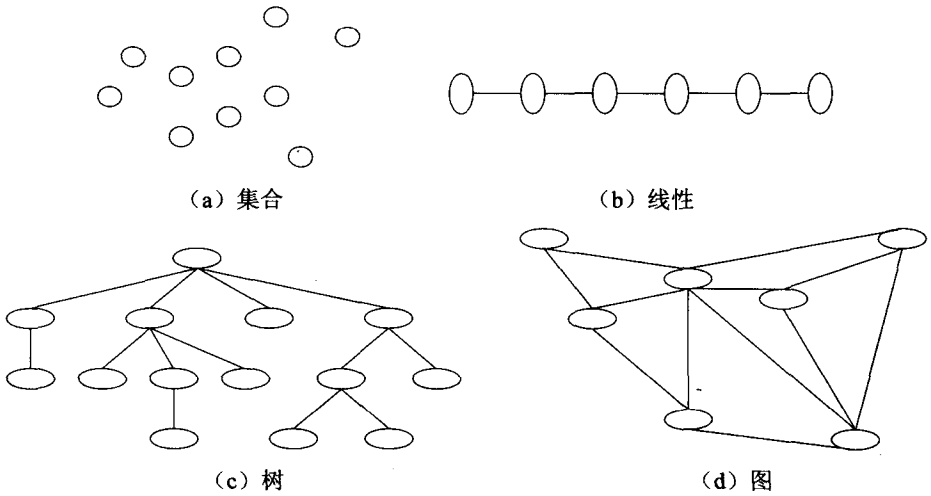


图 1-1　四类基本结构关系图

1.2.3　线性结构

线性结构有以下特点。

（1）在数据元素的非空有限集中存在唯一的一个被称做“第一个”的数据元素（即首结点）。

（2）存在唯一的一个被称做“最后一个”的数据元素（即尾结点）。

（3）除第一个之外，集合中的每个数据元素均只有一个前驱（在某元素之前的元素称之为前驱）。

（4）除最后一个之外，集合中每个数据元素均只有一个后继（在某元素之后的元素称之为后继）。

1.3　线　性　表

1.3.1　线性表的定义

线性表是最常用且最简单的一种数据结构。简言之，一个线性表是 n 个数据元素的有限序列。

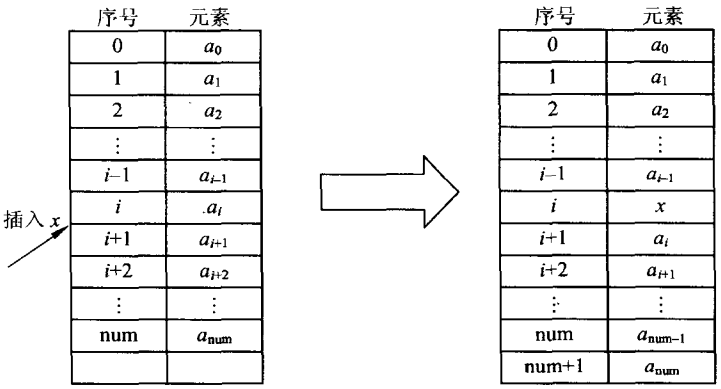
1.3.2　线性表的顺序存储结构

1．定义

在计算机中用一组地址连续的存储单元依次存储线性表的各个数据元素，称做线性表的顺序存储结构，简称顺序表。

2．顺序表的插入操作

在长度为 num($0 \leqslant num \leqslant MAXNUM-2$)(MAXNUM 为顺序表中长度的最大值)的顺序表 List 的第 i($0 \leqslant i \leqslant num+1$)个元素之前插入一个新的数据元素 x 时，需将最后一个即第 num 个至第 i 个元素（共 num–i+1 个元素）依次向后移动一个位置，空出第 i 个位置，然后把 x 插入第 i 个存储位置，插入结束后顺序表的长度增加 1，返回 TRUE 值；若 $i<0$ 或 $i>num+1$，则无法插入，返回 FALSE，如图 1-2 所示。

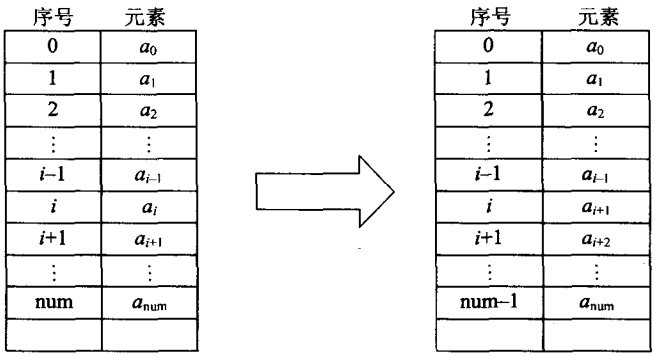


（a）插入 x 前　　（b）插入 x 后

图 1-2　在数组中插入元素

3．顺序表的删除操作

在长度为 num(0≤num≤MAXNUM–1)的顺序表 List 中删除第 i(0≤i≤num)个数据元素时，需将第 i 至第 num 个数据元素的存储位置(num–i+1)依次前移，并使顺序表的长度减 1，返回 TRUE 值，若 i<0 或 i>num，则无法删除，返回 FALSE 值，如图 1-3 所示。



（a）插入 a_i 前　　（b）插入 a_i 后

图 1-3　在数组中删除元素

1.4　栈和队列

1.4.1　栈

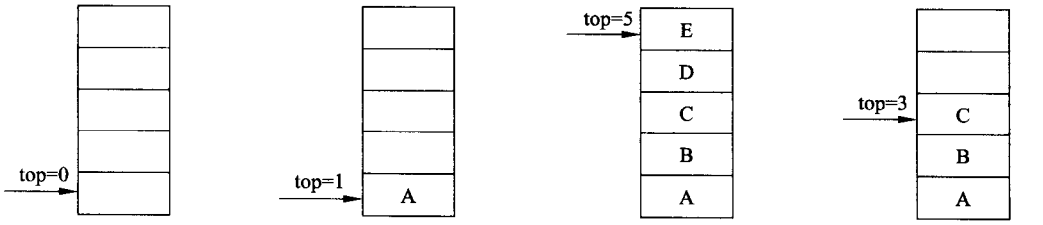
1．定义

栈（Stack）是一种只允许在一端进行插入和删除的线性表，它是一种操作受限的线性表。在表中只允许进行插入和删除的一端称为栈顶（top），另一端称为栈底（bottom）。栈的插入操作通常称为入栈（push），而栈的删除操作则称为出栈或退栈（top）。当栈中无数据元素时，称为空栈。

2．顺序存储结构

与一般的顺序存储结构的线性表一样，利用一组地址连续的存储单元依次存放自栈底到栈顶的数据元素，这种形式的栈也称为顺序栈。因此，可以使用一维数组来作为栈的顺序存储空间。设指针 top 指向栈顶元素的当前位置，以数组下标的一端作为栈底，通常以 top=0 时为空栈，在元素进栈时指针 top 不断地加 1，当 top 等于数组的最大下标值时则栈满。

图 1-4 展示了顺序栈中数据元素与栈顶指针的变化。



（a）空栈　（b）插入元素 A 后　（c）插入元素 B、C、D、E 后　（d）删除元素 E、D 后

图 1-4　栈的存储结构

1.4.2　队列

1．队列的定义

队列（Queue）是一种只允许在一端进行插入，而在另一端进行删除的线性表，它是一种操作受限的线性表。在表中只允许进行插入的一端称为队尾（rear），只允许进行删除的一端称为队头（front）。队列的插入操作通常称为入队列或进队列，而队列的删除操作则称为出队列或退队列。当队列中无数据元素时，称为空队列。

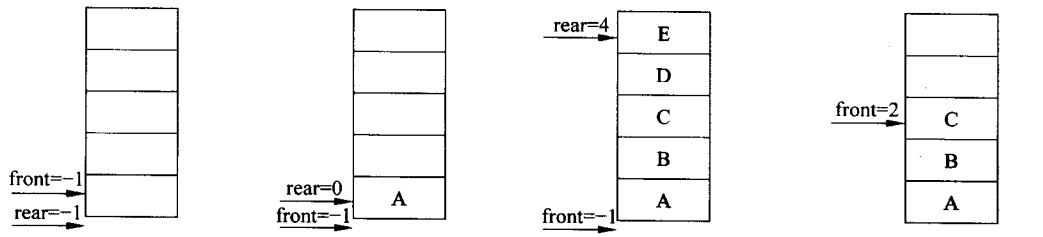
根据队列的定义可知，队头元素总是最先进队列，也总是最先出队列；队尾元素总是最后进队列，因而也是最后出队列。这种表是按照先进先出（FIFO，First In First Out）的原则组织数据的，因此，队列也被称为“先进先出”表。

2．队列的顺序存储结构

队列的顺序存储结构可以简称为顺序队列，也就是利用一组地址连续的存储单元依次存放队列中的数据元素。一般情况下，使用一维数组来作为队列的顺序存储空间，另外再设立两个指示器：一个为指向队头元素位置的指示器 front，另一个为指向队尾元素位置的指示器 rear。

C 语言中，数组的下标是从 0 开始的，因此，为了算法设计的方便，在此约定：初始化队列时，即空队列时，令 front=rear=−1，当插入新的数据元素时，尾指示器 rear 加 1，而当队头元素出队列时，队头指示器 front 加 1。另外还约定，在非空队列中，头指示器 front 总是指向队列中实际队头元素的前面一个位置，而尾指示器 rear 总是指向队尾元素。

图 1-5 给出了队列中头、尾指针的变化状态。



（a）空队列　（b）元素 A 入列后　（c）元素 B、C、D、E 入列后　（d）元素 E、D 出队列后

图 1-5　队列的存储结构

1.5　线性表的链式存储结构

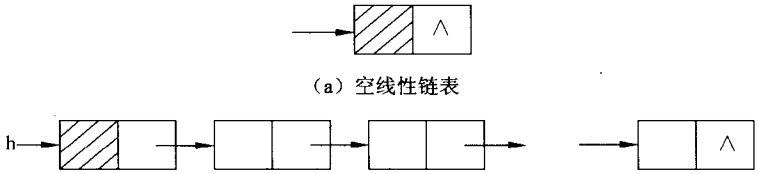
1.5.1　线性单链表

线性链表的存储结构

线性链表是线性表的链式存储结构，是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑

顺序是通过链表中的指针链接次序实现的。因此，在存储线性表中的数据元素时，一方面要存储数据元素的值，另一方面要存储各数据元素之间的逻辑顺序，为此，将每一个存储结点分为两部分：一部分用于存储数据元素的值，称为数据域；另一部分用于存放下一个数据元素的存储结点的地址，即指向后继结点，称为指针域。

此种形式的链表因为只含有一个指针域，又称为单向链表，简称单链表。图1-6（a）所示为一个空线性链表，图1-6（b）所示为一个非空线性链表（$a_0,a_1,a_2,\cdots,a_{n-1}$）。



（a）空线性链表

（a）非空线性链表

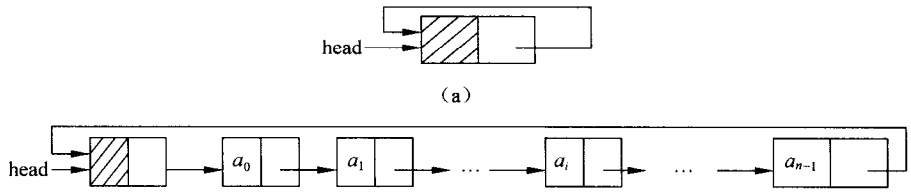
图1-6 线性链表的存储结构

图1-6中，通常在线性链表的第一个结点之前附设一个称为头结点的结点。头结点的数据域可以不存放任何数据，也可以存放链表的结点个数的信息。对空线性表，附加头结点的指针域为空（用NULL或0表示），在图示中用∧表示。头指针head指向链表附加头结点的存储位置。对于链表的各种操作必须从头指针开始。

1.5.2 循环链表

1．循环链表的结构

循环链表（Circular Linked List）是另一种形式的链式存储结构。它是将单链表的表中最后一个结点指针指向链表的表头结点，整个链表形成一个环，这样从表中任一结点出发都可找到表中其他的结点。图1-7（a）所示为带头结点的循环单链表的空表形式，图1-7（b）所示为带头结点的循环单链表的一般形式。



（a）

（b）

图1-7 循环链表

2．循环链表的基本运算

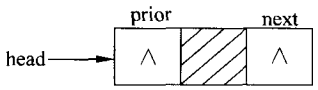
带头结点的循环链表的操作实现算法和带头结点的单链表的操作实现算法类同，差别在于算法中的条件不同：在单链表中为p!=null或p->next!=null，而在循环链表中应改为p!=head或p->next!=head。

在循环链表中，除了头指针head外，有时还加了一个尾指针rear，尾指针rear指向最后一结点，从最后一个结点的指针又可立即找到链表的第一个结点。在实际应用中，使用尾指针代替头指针来进行某些操作，往往更简单。

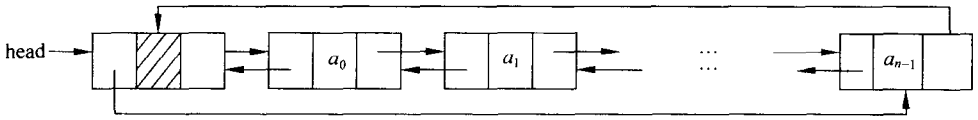
1.5.3 双向链表

在双向链表中，每一个结点除了数据域外，还包含两个指针域，一个指针（prior）指向该结点的后继结点，另一个指针（next）指向它的前驱结点。

和单链表的循环表类似，双向链表也可以有循环表。让表头结点的前驱指针指向链表的最后的一个结点，让最后一个结点的后继指针指向头结点。图1-8所示为双向链表示意图，其中，图（b）是一个循环双向链表。



（a）空双向链表



（b）非空的双向链表

图 1-8　双向链表

1.6　树

1.6.1　树的基本概念

树（tree）是 $n(n\geq 0)$个结点的有限集。在任意一棵非空树中，有且仅有一个特定的称为根（root）的结点，该结点没有前驱。当 $n>1$ 时，其余结点可分为 $m(m>0)$个互不相交的有限集 $T_1,T_2,\cdots,T_m$，其中，每一个集合本身又是一棵树，并且称为根的子树（Sub-tree）。这是一个递归的定义，即在定义树时又用到了树这个术语。

下面介绍与树有关的一些常用术语。

- 结点（node）：树中的元素，包含数据项及若干指向其子树的分支。
- 结点的度（degree）：结点拥有的子树数。
- 树的度：树内各结点的度的最大值。
- 叶子（leaf）：度为 0 的结点，又称终端结点。
- 分支结点：树中度不为 0 的结点，又称非终端结点。
- 孩子（child）：结点的子树的根称为该结点的孩子。
- 双亲（parents）：对应上述称为孩子结点的上层结点即为这些结点的双亲。
- 兄弟（sibling）：同一双亲的孩子之间互为兄弟。
- 堂兄弟：其双亲在同一层的结点互为堂兄弟。
- 结点的祖先：从根到该结点所经分支上的所有结点。
- 结点的子孙：以某结点为根的子树中的任一结点都称为该结点的子孙。
- 结点的层次（level）：从根开始定义，根为第一层，根的孩子为第二层；若某结点在 m 层，则该结点的子树的根在 $m+1$ 层。
- 深度（depth）：树中结点的最大层次数。
- 森林（forest）：是 $m(m>0)$棵互不相交的树的集合。对树中每个结点而言，其子树的集合即为森林。
- 有序树和无序树：如果各子树依次从左到右排列，不可对换，则称该子树为有序树，且把各子树分别称为第一子树，第二子树……；反之，称为无序树。

1.6.2　二叉树

二叉树（Binary Tree）又称为二分树或二元树，是一种重要的树形结构。

1．二叉树的定义

二叉树是一种度不大于 2 的有序树，它的特点是每个结点至多只有两棵子树（即二叉树中不存在度大于 2 的结点），并且二叉树的子树有左右之分，其次序不能任意颠倒。

也可以以递归的形式定义二叉树：二叉树是 $n(n\geqslant 0)$个结点的有限集，它或者为空树($n=0$)，或者由一个根结点和两棵分别称为左子树和右子树的互不相交的二叉树所构成。

2．二叉树的存储结构

树形结构是非线性结构，采用顺序存储有一定的困难。通常用具有两个指针域的链表作为二叉树的存储结构，其中，每个结点由数据域 data、左指针域 Lchild 和右指针域 Rchild 组成。两个指针域分别指向该结点的左、右孩子。若某结点没有左孩子或右孩子，则对应的指针域为空。当然，还需要一个链表的头指针指向根结点。二叉树的链接存储结构也叫二叉链表，如图 1-9 所示。

Lchild	data	Rchild

图 1-9　二叉链表

1.6.3　遍历二叉树

在二叉树的应用中，常常需要在树中搜索具有某种特征的结点，或者对树中全部的结点逐一进行处理，这就涉及一个遍历二叉树（Traversing binary tree）的问题。遍历（Traversing）是指循着某条搜索路线查找某数据结构中的结点，而且每个结点只被访问一次。

遍历操作让二叉树中各结点按指定的顺序线性排列，使得非线性的二叉树线性化。线性遍历规则是一种递归定义的规则。

由于一棵非空二叉树是由根结点、左子树和右子树三个基本部分组成，遍历二叉树时只要依次遍历这三部分即可。假定以 D、L、R 分别表示访问根结点、遍历左子树和遍历右子树，则可以有六种遍历形式：DLR、LDR、LRD、DRL、RDL、RLD，若规定先左后右，则上述六种形式可归并为以下三种形式。

（1）DLR：前序（根）遍历。

（2）LDR：中序（根）遍历。

（3）LRD：后序（根）遍历。

下面将分别具体介绍这样三种形式的遍历规则。

1．前序遍历

在前序遍历中，当二叉树非空时按以下顺序遍历，否则结束操作：

（1）访问根结点；

（2）按前序遍历规则遍历左子树；

（3）按前序遍历规则遍历右子树。

例如，对如图 1-10 所示的二叉树按前序遍历规则遍历，则遍历结果为：A B D E C F G。

2．中序遍历

在中序遍历中，当二叉树非空时按以下顺序遍历，否则结束操作：

（1）按中序遍历规则遍历左子树；

（2）访问根结点；

（3）按中序遍历规则遍历右子树。

例如，对如图 1-10 所示的二叉树按中序遍历规则遍历，则遍历结果为：D B E A F G C。

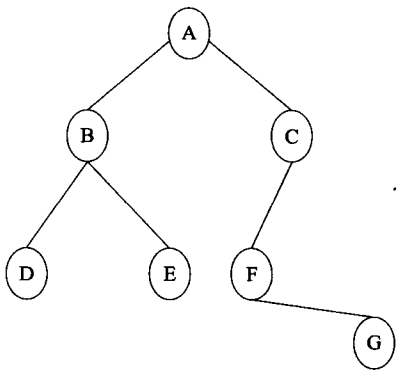


图 1-10　二叉树

3．后序遍历

在后序遍历中，当二叉树非空时按以下顺序遍历，否则结束操作：

（1）按后序遍历规则遍历左子树；

（2）按后序遍历规则遍历右子树；

（3）访问根结点。

例如，对如图 1-10 所示的二叉树按后序遍历规则遍历，则遍历结果为：D E B G F C A。

1.7 查找与排序

1.7.1 查找

本节所要介绍的查找不是一种数据结构，而是一种基于数据结构的辅助性运算。查找又称检索，是从大量的数据中找出所需要的数据来，也可以简单地说是确定一个已给的数据是否出现在某个数据表中。查找的方法很多，下面将重点讨论两种比较常用的方法。

1. 顺序查找

顺序查找是一种最基本和最简单的查找方法。它的思路是，从表中的第一个元素开始，将给定的值与表中逐个元素的关键字进行比较，直到两者相符，查找到所要的元素为止；否则就是表中没有要找的元素，查找不成功。对于表中记录的关键字是无序的表，只能采用这种方法。描述顺序查找的算法如下。

```
Void seqsrch (struct node r[], int n, int k)
{ int i=1;
  r[n+1]. Key=k;
  while (r[i]. Key!=k)
      i=i+1;
  if (i<=n)
    printf ("%3d, it is r[%2d]", k, I);
  else printf ("%3d not fount", k);
}
```

其中，n 是表 r 的长度，k 是要查找的元素的关键字，i 是查到的元素的序号。此算法中，在表尾增加了一个关键字为指定值 k 的记录，即 r[n+1]=k，用以起到标志的作用，以免每一步都要测试表是否结束，可节省大量时间。

顺序查找对短表合适，方法简单；对长表不合适，查找起来会很慢。

2. 二分法查找

二分查找又称折半查找，是针对有序表进行查找的简单、有效而又较常用的方法。所谓有序表，即要求表中的各元素按关键字的值有序（升序或降序）存放。

折半查找不像顺序查找那样，从第一个记录开始逐个顺序搜索，其基本思想是：首先选取表中间位置的记录，将其关键字与给定关键字 k 进行比较，若相等，则查找成功；若 k 值比该关键字值大，则要查找的元素一定在表的后半部分（或称右子表），应继续对右子表进行折半查找；若 k 值比该关键字值小，则要查找的元素一定在表的前半部分（左子表），同样应继续对左子表进行折半查找。每进行一次比较，要么找到要查找的元素，要么将查找的范围缩小一半，如此递推，直到查找成功或把要查找的范围缩小为空（查找失败）。

设表的长度为 n，表的被查找部分的头为 low，尾为 high，初始时，low=1，high=n，k 为关键字的值。查找步骤如下：

（1）计算中间记录的序号 mid=[(low+high)/2]，取整。

（2）若 k=r[mid].key，查找成功，否则：

若 k<r[mid].key，则 high=mid–1,重复步骤（1）。

若 k>r[mid].key，则 low=mid+1,重复步骤（1）。

直到查找成功或失败（此时 low>high）。