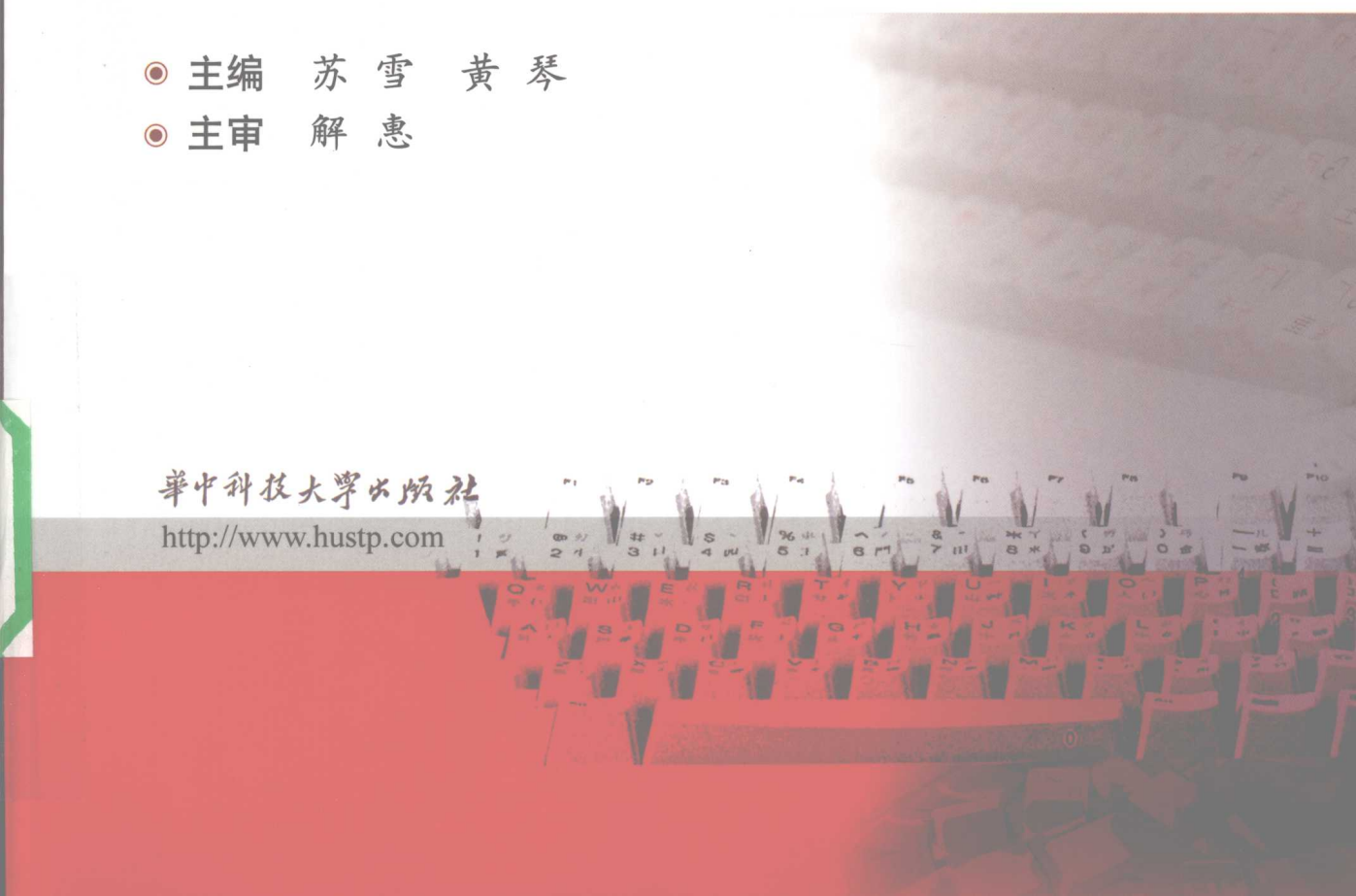
应用型电子信息类专业“十一五”规划教材

二级Visual FoxPro程序设计与综合训练

二级公共基础知识＋笔试＋机试抓图

◉ 主编　苏　雪　黄　琴

◉ 主审　解　惠



華中科技大學出版社

http://www.hustp.com

应用型电子信息类专业“十一五”规划教材

二级 Visual FoxPro 程序设计与综合训练

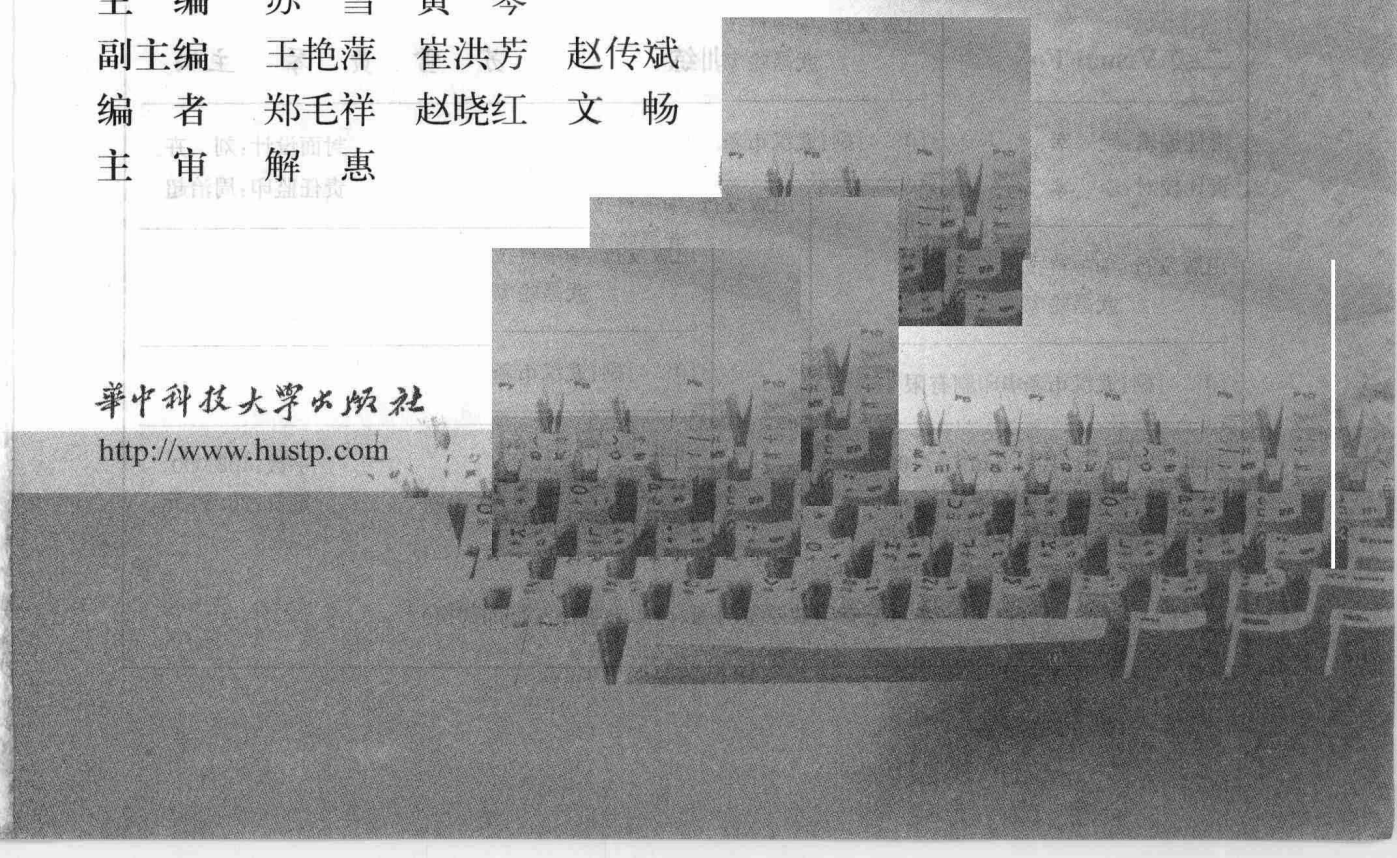
二级公共基础知识＋笔试＋机试抓图

主　编　苏　雪　黄　琴

副主编　王艳萍　崔洪芳　赵传斌

编　者　郑毛祥　赵晓红　文　畅

主　审　解　惠



華中科技大學出版社

http://www.hustp.com

图书在版编目(CIP)数据

二级 Visual FoxPro 程序设计与综合训练/苏　雪　黄　琴　主编. —武汉:华中科技大学出版社,2008 年 9 月
ISBN 978-7-5609-4513-2

Ⅰ. 二…　Ⅱ. ①苏…　②黄…　Ⅲ. 关系数据库-数据库管理系统,Visual FoxPro-程序设计-水平考试-自学参考资料　Ⅳ. TP311.138

中国版本图书馆 CIP 数据核字(2008)第 138230 号

二级 Visual FoxPro 程序设计与综合训练　　苏　雪　黄　琴　主编

责任编辑:姚　幸　　封面设计:刘　卉
责任校对:李　琴　　责任监印:周治超

出版发行:华中科技大学出版社(中国·武汉)
武昌喻家山　邮编:430074　电话:(027)87557437

印　刷:武汉市新华印刷有限责任公司

开本:787 mm×1092 mm　1/16　印张:17　字数:408 000
版次:2008 年 9 月第 1 版　印次:2008 年 9 月第 1 次印刷　定价:26.80 元
ISBN 978-7-5609-4513-2/TP·664

内容提要

本书以教育部考试中心最新颁布的全国计算机等级考试大纲(2008版)为依据,以对考生进行综合指导为原则,综合了近3年连续5次考试题和考前辅导班教师的实际教学经验编写而成。

全书共分三部分。第一部分在深入研究考试大纲的基础上,总结提炼出二级公共基础知识的主要考点,针对各考点组织了相关内容并编写了相关习题,主要包括基本数据结构和算法、程序设计基础、软件工程基础及数据库基础等内容。第二部分为二级Visual FoxPro程序设计笔试部分,主要综合了近3年连续5次考试真题及其详解,同时融汇考前辅导班教师的实际教学经验编写了模拟试题及详细解析。第三部分为二级Visual FoxPro程序设计上机考试部分,主要从基本操作题、简单应用题和综合应用题3个方面总结了历次上机考试常考的知识点;并通过图示详细讲解真题,总结重要的考点、答题思路和应试技巧。

本书适合准备报考全国计算机等级考试二级Visual FoxPro语言程序设计的考生的考前自学,同时也可作为普通高校、成人教育及各类培训学校举办的考前二级辅导班的培训教材。

前　言

为了引导考生有条不紊地进行复习，提高复习效率和应试能力，我们以教育部考试中心最新颁布的全国计算机等级考试《二级 Visual FoxPro 程序设计》大纲（2008 版）为依据，以对考生进行综合指导为原则，综合了近 3 年连续 5 次考试题和考前辅导班教师的实际教学经验共同编写了此书。

本书主要有以下几个特点：

1. 根据考试题型归纳考试内容，以重点知识为突破口，明确复习思路，提高考生应试能力。

不少考生在复习考试时会感到内容零散，知识点之间的跳跃性大，似乎没有连续性，学习和掌握起来很困难。鉴于此种情况，本书以对考生进行综合指导为原则，坚持“试题虽然千变万化，但命题角度有规律可循”的编写思路，通过深入研究历年真题，从不同命题角度归纳考试中常见的出题类型，使考生不仅了解哪些是要考的，而且了解会怎么考。在每一类题型下，设有 2 个栏目：真题演练、热点试题与答案。通过近 3 年连续 5 次考试题来摸清考试规律，总结出各知识点的出现概率，从而提炼出考点的重要程度。根据考试大纲中真题频发的相关知识点精确选取、编写模拟试题详细解析。明确复习思路，提高综合应试能力。

2. 内容全面、新颖，结构严谨、重点突出，有很强的实用性。

本书包括全国计算机等级考试《二级 Visual FoxPro 程序设计》中涉及的所有内容：二级公共基础知识、笔试部分和机试部分。内容新颖，所有知识点的选编都是以教育部考试中心最新颁布的全国计算机等级考试《二级 Visual FoxPro 程序设计》大纲（2008 版）为依据，以对考生进行综合指导为原则，综合了近 3 年连续 5 次考试题和考前辅导班教师的实际教学经验。结构严谨、重点突出，有很强的实用性，可以说考生只要有一书在手，考试无忧。

3. 图文并茂，通俗易懂，适合考前自学。

本书机试部分图文并茂，所有操作都依据屏幕实际显示的样式一步一步讲述，读者可以边看边上机操作，通过范例和具体操作，理解基本概念和学会操作方法。即使是计算机基础薄弱的考生也能轻松理解。为考生顺利通过机试提供了方便。

本书由武汉铁路职业技术学院苏雪、黄琴主编，解惠主审。参与本书编写的院校和人员还有：郑州铁路职业技术学院（王艳萍）、湖北经济学院（崔洪芳）、武汉理工大学华夏学院（赵传斌）、武汉铁路职业技术学院（郑毛祥、赵晓红）、长江大学（文畅）。在本书的编写和出版过程中，还得到了华中科技大学出版社有关人员的大力支持和帮助，在此一并表示感谢。

由于作者水平有限，书中难免存在错漏或不妥之处，敬请读者批评指正。

编　者

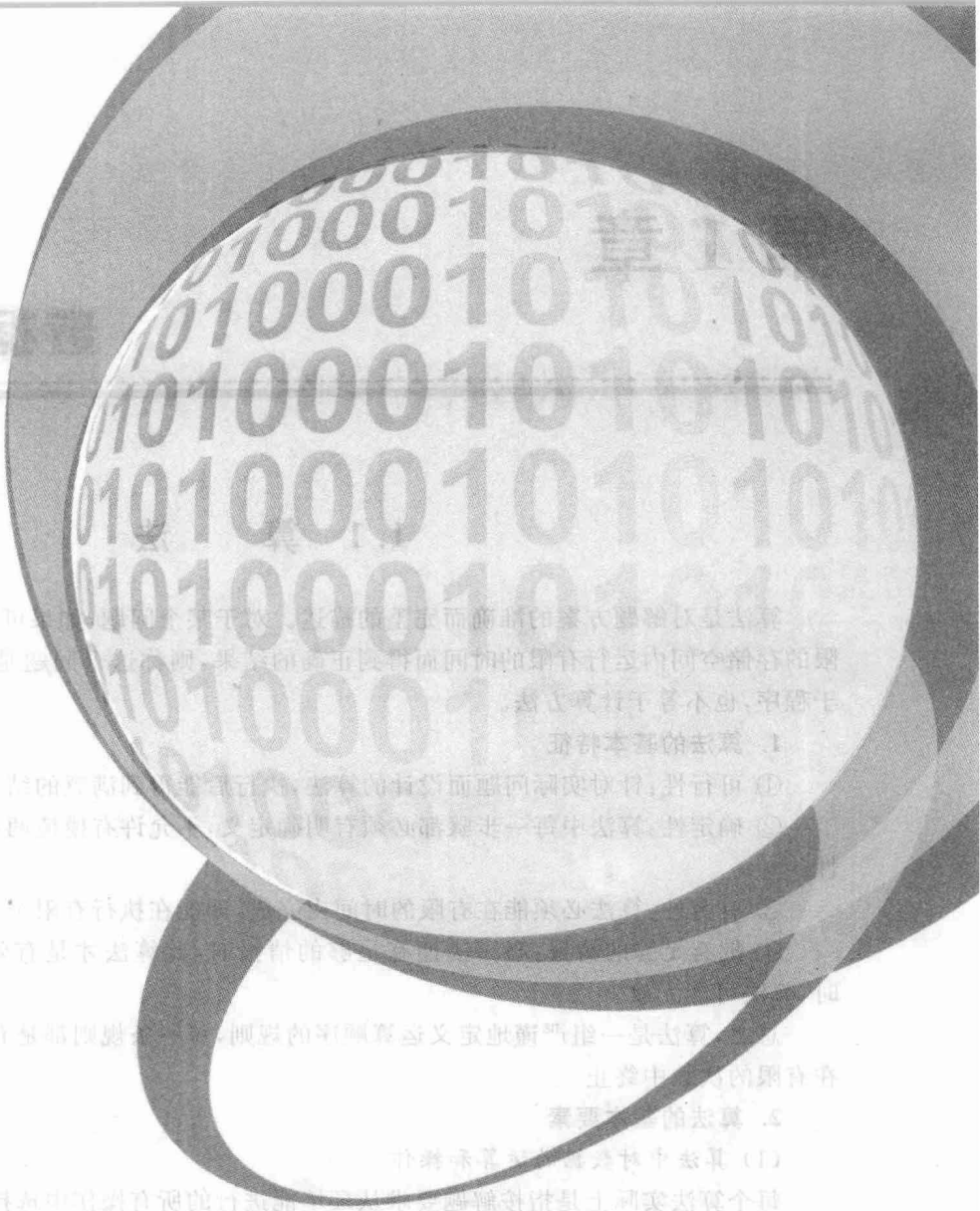
2008 年 6 月

目　录

第 1 部分　二级公共基础知识

第 2 部分 二级 Visual FoxPro 笔试试题解析

第 3 部分　二级 Visual FoxPro 机试试题解析



第1部分

二级公共基础知识

第 1 章

数据结构与算法

1.1 算　　法

算法是对解题方案的准确而完整的描述。对于某个问题，如果可以通过计算机程序，在有限的存储空间内运行有限的时间而得到正确的结果，则称这个问题是算法可解的。算法不等于程序，也不等于计算方法。

1. 算法的基本特征

① 可行性：针对实际问题而设计的算法，执行后能得到满意的结果。

② 确定性：算法中每一步骤都必须有明确定义，不允许有模棱两可的解释，不允许有多义性。

③ 有穷性：算法必须能在有限的时间内完成，即能在执行有限个步骤后终止。

④ 拥有足够的情报：当算法拥有足够的情报时，此算法才是有效的；当提供的情报不足时，算法可能无效。

总之，算法是一组严谨地定义运算顺序的规则，每一条规则都是有效的、明确的，此顺序将在有限的次数中终止。

2. 算法的基本要素

(1) 算法中对数据的运算和操作

每个算法实际上是指按解题要求从环境能进行的所有操作中选择合适的操作所组成的一组指令序列。其基本的运算和操作包括：算术运算(加、减、乘、除等)，逻辑运算(与、或、非等)，关系运算(大于、小于、等于、不等于等)，数据传输(赋值、输入、输出等)。

(2) 算法的控制结构

一个算法的功能不仅取决于所选用的操作，而且与各操作之间的执行顺序有关。算法中各操作之间的执行顺序称为算法的控制结构。算法的控制结构一般有顺序结构、选择结构、循环结构等。

3. 算法设计的基本方法

常用算法及其思想如表 1-1-1 所示。

表 1-1-1

方　法	思　想
列举法	根据提出的问题，列举所有可能的情况，并用问题中给定的条件检测哪些是需要的，哪些是不需要的
归纳法	通过一些简单而特殊的事例，最后得出一般性的结论

续表

方　法	思　想
递推法	从已知的初始条件出发，逐步推出所要求的中间结果和最后结果。其中，初始条件或是问题本身已经给定，或是通过对问题的分析与化简而得到
递归法	为了降低问题的复杂程度，将问题逐层分解，最后归结为一些最简单的问题。解决了那些最简单的问题后，再沿着原来分解的逆过程逐步进行求解
减半递推法	“减半”是指将问题的规模减半，“递推”是指重复减半的过程
回溯法	通过对问题的分析，找出一个解决问题的线索，然后沿着这个线索逐步试探。试探成功就得到问题的解，试探失败则逐步回退，换其他路线再试探

4. 算法复杂度

算法的复杂度包括算法的时间复杂度和算法的空间复杂度。算法的时间复杂度是指执行算法所需要的计算工作量，它取决于问题的规模。算法的空间复杂度是指执行这个算法所需要的内存空间，包括算法程序所占的空间，输入的初始数据所占的存储空间，以及算法执行过程中所要的额外空间。

1.2　数据结构的基本概念

1. 数据

数据(Data)是对客观事物的符号表示。在计算机科学中，数据是所有输入到计算机中并被计算机程序处理的符号的总称。

2. 数据元素

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。数据元素可由若干个数据项(Data Item)组成，例如，学生信息中每一项(如学号、姓名、性别、年龄、籍贯等)都可作为一个数据项。数据项是数据不可分割的最小单位。

3. 数据结构

研究数据结构的目的是提高数据处理的效率。提高数据处理效率包括两个方面的内容：一是提高数据处理的速度，二是尽量节省在数据处理过程中所占用的计算机存储空间。

数据结构研究以下三个方面的问题：

① 数据集合中各数据元素之间所固有的逻辑关系，即数据的逻辑结构；

② 在对数据进行处理时，各数据元素在计算机中的存储关系，即数据的存储结构；

③ 对各种数据结构进行的运算。

(1) 数据的逻辑结构

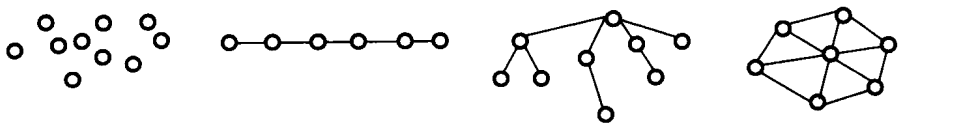
数据的逻辑结构是对数据元素之间的逻辑关系的描述，可以用数据元素的集合和定义在此集合中的若干关系来表示。数据元素间有如下四类基本逻辑结构，如图 1-1-1 所示。

① 集合：结构中数据元素同属于一个集合，别无其他关系。

② 线性结构：结构中数据元素存在一对一的关系。

③ 树形结构：结构中数据元素存在一对多的关系。

④ 图状结构或网状结构：结构中数据元素存在多个对多个的关系。



(a) 集合　(b) 线性结构　(c) 树形结构　(d) 图状结构或网状结构

图 1-1-1

数据结构的形式定义为

$$Data\_Structure=(D,S)$$

其中，D 是数据元素的有限集，S 是 D 上关系的有限集。

(2) 数据的存储结构

数据的存储结构是数据的逻辑结构在计算机存储空间中的存放形式，也称数据的物理结构。一种数据结构可根据需要采用不同的存储结构。一个存储结构包括以下三个主要部分：

① 存储结点，每个存储结点存放一个数据元素；

② 数据元素之间的关联方式表示，也就是逻辑结构的计算机内部表示；

③ 附加设施。

存储结点之间可以有如下四种关联方式(称为四种基本存储方式)。

① 顺序存储：每个存储结点只含一个数据元素，所有存储结点相继存放在一个连续的存储区域。

② 链式存储：在存储结点上附加一个指针域来表示结点之间的逻辑关系，每个指针指向一个与本结点逻辑相关的结点。

③ 索引存储：每个存储结点只含一个数据元素，所有存储结点连续存放。此外，增设一个索引表，索引表中的索引指示各存储结点的存储位置或位置区间端点。

④ 散列存储：每个存储结点含一个数据元素，均匀分布在存储区，用散列函数指示各结点的存储位置或位置区间端点。

4. 线性结构与非线性结构

如果一个数据结构中一个数据元素也没有，则称该数据结构为空的数据结构。

根据数据结构中各数据元素之间前后件关系的复杂程度，一般将数据结构分为线性结构与非线性结构两类。

线性结构的条件如下。

① 有且只有一个根结点(没有前件的结点)；

② 每个结点最多有一个前件，也最多有一个后件。

线性结构又称为线性表，本书将要介绍的线性表、栈、队列和线性链表等都是线性结构。

非线性结构是指不满足线性结构条件的数据结构，如树、二叉树、图。

线性结构和非线性结构都可以是空的数据结构。对于空的数据结构，如果对该数据结构的运算是按线性结构的规则来处理的，则属于线性结构，否则属于非线性结构。

1.3　线性表及其顺序存储结构

1. 线性表的基本概念

线性表是由 $n(n\geqslant 0)$ 个数据元素组成的有限序列。通常，将含 n 个结点的线性表表示成

$(a_1,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n)$的有序集合。其中，$a_i(0<i\leqslant n)$表示一个结点。$a_1$ 称为起始结点，a_n 称为终端结点，i 为 a_i 在线性表中的序号或位置。对任意对相邻结点$<a_i,a_{i+1}>(1\leqslant i<n)$，称 a_i 为 a_{i+1}的前件(或前驱结点)，a_{i+1}为 a_i 的后件(或后继结点)。

非空线性表的结构特征如下。

① 有且只有一个根结点 a_1，它无前件。

② 有且只有一个终端结点 a_n，它无后件。

③ 除根结点与终端结点外，其他所有结点有且只有一个前件、也有且只有一个后件。结点个数 n 称为线性表的长度，当 n=0 时，称为空表。

2. 线性表的顺序存储结构——顺序表

顺序表是线性表的顺序存储结构，即用一组地址连续的存储单元依次存储线性表的数据元素。所有的存储结点按相应数据元素间的逻辑关系(即一对一的邻接关系)决定的次序依次排列。每一个数据元素的存储位置都和线性表的起始位置相差一个和数据元素在线性表中位置成正比的常数(见图 1-1-2)。由此，只要确定了存储线性表的起始位置，线性表中任一数据元素的存储地址都可表示。a_i 的存储地址为：$ADR(a_i)=ADR(a_1)+(i-1)k$。其中，$ADR(a_1)$为第一个元素的地址，k 为每个元素所占的字节数。maxsize 为顺序表的容量，从最后一个结点 a_n 到 maxsize−1 为空闲区。

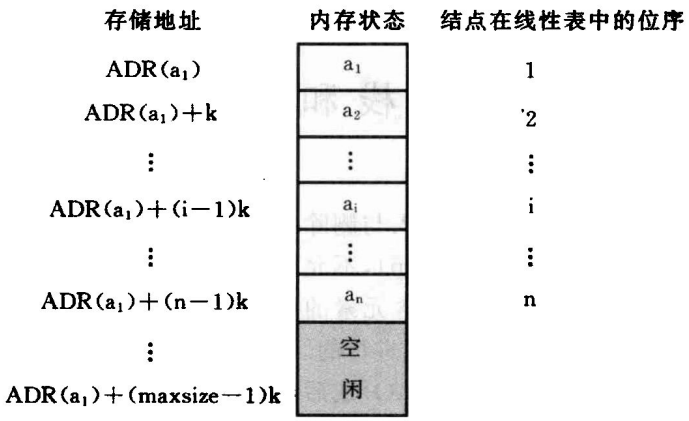


图 1-1-2

注意：顺序表的容量是指线性表实际达到的最大长度；表长是指结点的个数 n，表长小于等于表的容量。

3. 线性表的插入和删除运算

(1) 插入运算

线性表的插入运算是指在表的第 $i(1\leqslant i\leqslant n)$个结点的位置上，插入一个新结点 b，使长度为 n 的线性表$(a_1,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n)$变成长度为 n+1 的线性表$(a_1,\cdots,a_{i-1},b,a_i,a_{i+1},\cdots,a_n)$。插入算法的基本步骤是：

① 将结点 $a_i,\cdots,a_n$ 依次后移一个位置，空出第 i 个结点的位置；

② 将新结点 b 插入第 i 个结点的位置；

③ 表长加 1。

(2) 删除运算

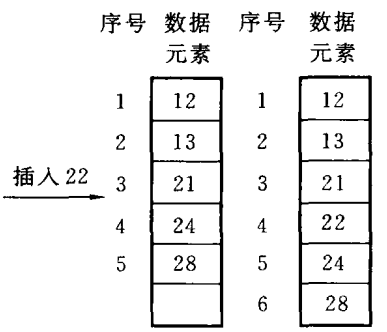
线性表的删除运算是指将表的第 $i(1\leqslant i\leqslant n)$个结点删除，使长度为 n 的线性表$(a_1,\cdots,$

a_{i-1}，a_i，a_{i+1}，…，a_n）变成长度为 n－1 的线性表（a_1，…，a_{i-1}，a_{i+1}，…，a_n）。删除算法的基本步骤如下。

① 将结点 a_{i+1}，…，a_n 依次前移一个位置，覆盖被删结点 a_i；

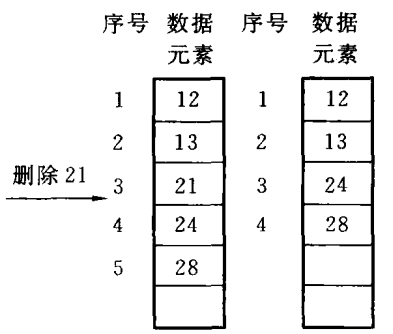
② 表长减 1。

图 1-1-3 和图 1-1-4 分别演示了一个线性表在插入、删除前后，其数据元素在存储空间中的位置变化。



（a）插入前　（b）插入后

图 1-1-3



（a）删除前　（b）删除后

图 1-1-4

1.4　栈和队列

1. 栈

栈（Stack）是限定只能在一端进行插入与删除的线性表。允许插入与删除的一端称为栈顶（Top），不允许插入与删除的另一端称为栈底（Bottom）。不含元素的空表称为空栈。栈的修改是按照后进先出的原则进行的。因此，栈又称为先进后出（First In Last Out，FILO）或后进先出（Last In First Out，LIFO）的线性表，如图 1-1-5 所示。

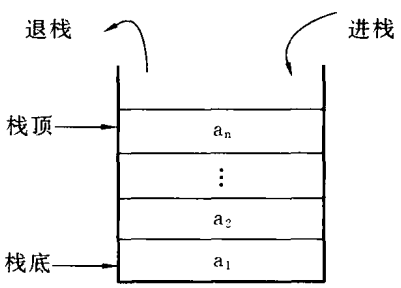


图 1-1-5

2. 栈的顺序存储

栈的顺序存储结构称为顺序栈。顺序栈被定义为一个结构类型，它由两部分组成：数据数组 data 和栈顶指针 top。top 的取值为 0～栈容量－1。当 top＝0 时，栈空；当 top＝栈容量－1 时，栈满。

3. 栈的基本运算

① 入栈运算：在栈顶位置插入一个新元素。首先将栈顶指针 top 加 1，然后将新元素插入到栈顶指针指向的位置。当栈顶指针已经指向存储空间的最后一个位置时，说明栈满，不能再进行入栈运算，这种情况称为“上溢”。

② 退栈运算：取出栈顶元素并赋予它一个指定的变量。首先给栈顶元素赋予一个指定的变量，然后将栈顶指针减 1，当栈顶指针为 0 时，说明栈空，不能再进行退栈运算，这种情况称为“下溢”。

③ 读栈顶元素：将栈顶元素赋给一个指定的变量，栈顶指针 top 无变化。当栈顶指针为 0 时，说明栈空，读不到栈顶元素。

4. 队列

队列是指允许在一端(队尾)进行插入,而在另一端(队首)进行删除的线性表。rear 指针指向队尾,front 指针指向队首。因此,队列是先进先出(First In First Out)或后进后出(LILO,Last In Last Out)的线性表,如图 1-1-6 所示。

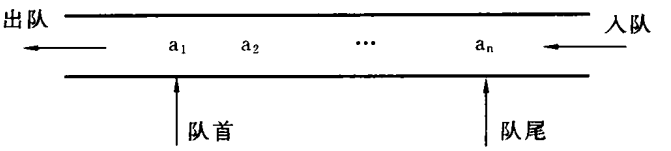


图 1-1-6

队列运算包括入队运算(往队列的队尾插入一个元素)和退队运算(从队列的队首删除一个元素)。入队运算只涉及队尾指针(rear=rear+1),退队运算只涉及队首指针(front=front+1)。

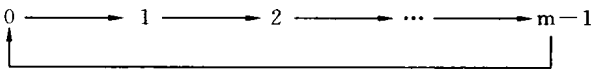
5. 循环队列及其运算

在队列的入队和出队操作中,其对应的队尾和队首指针(下标)都是进 1 操作(否则,出队操作需要移动所有的数据元素)。随着入队和出队操作的进行,队尾和队首指针都在逐步增大,所以队列若用普通的顺序存储结构来实现,很容易上溢。因此,实际中一般使用循环队列。

通过求余运算(%),可以实现下标的循环累进:

$$index=(index+1)\ \%\ m$$

即



因此,对队首和队尾进行如下操作就可以实现循环队列:

front=(front+1) % maxsize　　队首指针循环进 1

rear=(rear+1) % maxsize　　队尾指针循环进 1

其中,maxsize 表示数组的最大容量,front 和 rear 的最大取值为 maxsize−1。

图 1-1-7 和图 1-1-8 分别是循环队列逆时针运行和循环队列满的示意图。通过队首指针 front、队尾指针 rear 可知,判断队列为空的条件为 front==rear,判断队满条件为 front==(rear+1)% maxsize。

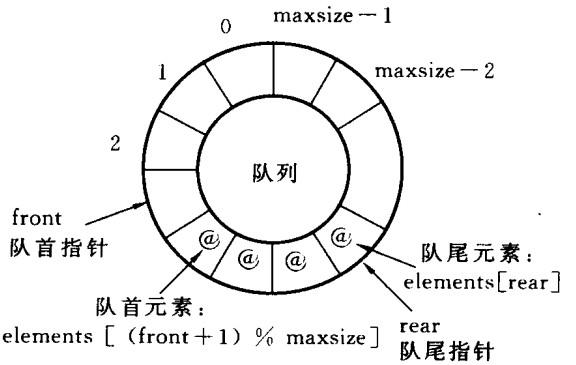


图 1-1-7

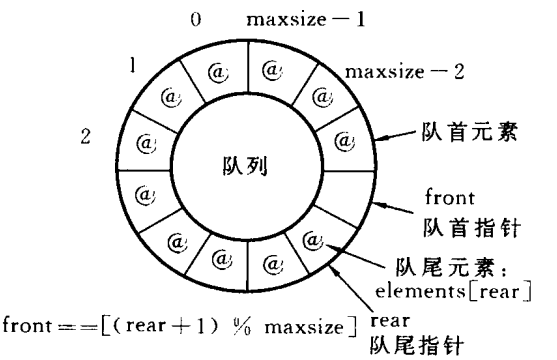


图 1-1-8

由图 1-1-8 可知:队满时队首指针所指的单元不能利用,此时再入队会使队列变成空队列(front==rear)。

1.5　线 性 链 表

1. 单链表

单链表是线性表的链式存储结构。其基本思想是，用指针表示结点间的逻辑关系。单链表的结点表现形式为

可以看出，单链表的结点由数据域(data)和指针域(next)两部分组成。数据域用于存储一个数据元素的信息；指针域用于存储一个指针，该指针指向本结点所含数据元素的后继结点。如此，所有结点通过指针的链接而组织成单链表，如图 1-1-9 所示。

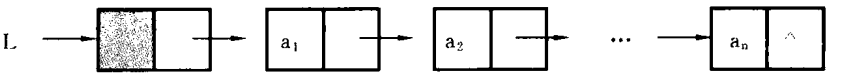


图 1-1-9

在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致。各数据元素之间的逻辑关系是由指针域来确定的。

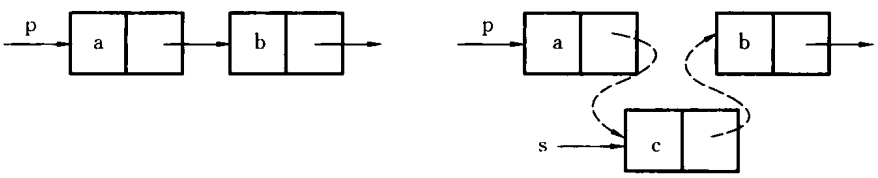
(1) 在线性链表中查找指定元素

在对线性链表进行插入或删除操作之前，需要找到插入或删除的位置，这就需要对线性链表进行扫描查找。例如，要查找第 i 个元素，不能直接访问 i 个结点，而只能从链表的头指针出发，顺着指针 next 逐个往下搜索。

(2) 插入运算

在单链表中插入一个结点 s 的过程如图 1-1-10 所示。用 C 语言描述，则为

```
s->next=p->next;p->next=s;
```



(a) 插入前　　(b) 插入后

图 1-1-10

(3) 删除运算

在单链表中删除一个结点 s 的过程如图 1-1-11 所示。用 C 语言描述，则为

```
p->next=s->next;free(s);
```

其中，free(s)的功能是释放 s 结点所占的内存空间。

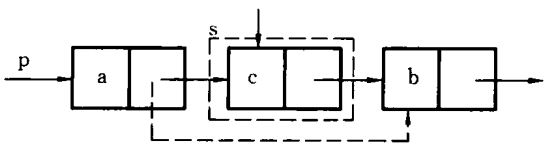


图 1-1-11

2. 循环链表

循环链表的组织和操作基本上和单链表一致，差别仅仅在于其尾结点的指针指向头结点。非空的循环链表如图 1-1-12 所示。空的循环链表如图 1-1-13 所示。

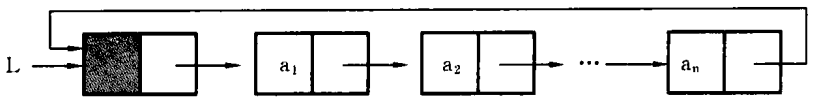


图 1-1-12

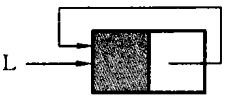
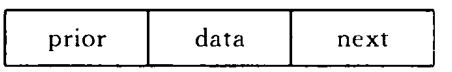


图 1-1-13

3. 双链表（双向循环链表）

和单链表一样，双链表也有一个数据域，用于存放元素的数据信息；与单链表不同的是，双链表有两个指针域，一个指针（prior）指向前驱结点，另一个指针（next）指向后继结点。双链表结点的表现形式为



其中，指针（prior 和 next）分别指向本结点数据域 data 所含数据元素的前驱结点和后继结点。所有结点通过前驱指针 prior 和后继指针 next 链接在一起，再加上起标识作用的头指针，就得到双向循环链表，简称双链表。非空的双链表如图 1-1-14 所示。空的双链表如图 1-1-15 所示。

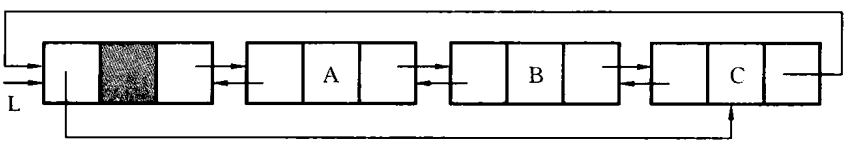


图 1-1-14

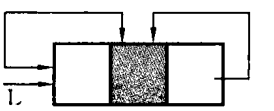


图 1-1-15

（1）插入结点

在双链表中插入结点 s 时，其指针变化过程如图 1-1-16 所示，用 C 语言描述则为

① s->next=p->next;

② p->next->prior=s;

③ p->next=s;

④ s->prior=p;

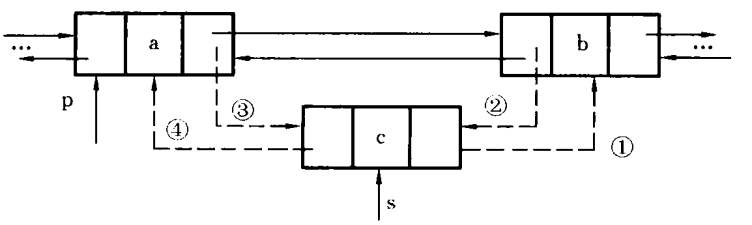


图 1-1-16

（2）删除结点

在双链表中删除 s 结点时，其指针变化过程如图 1-1-17 所示，用 C 语言描述则为

① p->next=s->next;

② s->next->prior=p;