全国计算机等级考试（新大纲）应试用书

全国计算机等级考试

二级教程

——计算机公共基础

■ 管群　主编　■ 曾新　副主编　■ 管斌 刘淑东 贺丹 葛宇　编著

计算机等级考试

人民邮电出版社
POSTS & TELECOM PRESS

全国计算机等级考试（新大纲）应试用书

全国计算机等级考试

二级教程
——计算机公共基础

管 群 主 编

曾 新 副主编

管 斌 刘淑东 贺 丹 葛 宇 编著

人民邮电出版社

图书在版编目（CIP）数据

全国计算机等级考试二级教程．计算机公共基础／管群主编．
—北京：人民邮电出版社，2005.8
全国计算机等级考试（新大纲）应试用书

ISBN 7-115-13967-9

Ⅰ．全… Ⅱ．管… Ⅲ．电子计算机－水平考试－教材 Ⅳ．TP3

中国版本图书馆 CIP 数据核字（2005）第 096156 号

内 容 提 要

本书按照全国计算机等级考试最新大纲（2004 年版）的要求，系统介绍全国计算机等级考试二级公共基础的知识点。主要包括算法与数据结构、程序设计、软件基础和数据库设计等 4 个方面。本书内容简明扼要，讲解深入浅出，并精选范例以辅助读者理解相关的知识。

公共计算机基础是对现行大学计算机基础的补充和扩展，本书适合欲参加全国计算机等级二级考试的各类考生学习参考，可作为本科、专科院校学生学习大学计算机应用技术基础的辅助教材，也可作各类社会计算机应用技术基础培训的教材，还可作为计算机从业人员的计算机技术学习参考用书。

本书配有电子教案供教学参考，读者可与作者联系索取。

全国计算机等级考试（新大纲）应试用书

全国计算机等级考试

二级教程——计算机公共基础

◆ 主　　编　管　群
副 主 编　曾　新
编　　著　管　斌　刘淑东　贺　丹　葛　宇
责任编辑　邹文波

◆ 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　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京通州大中印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：8.25
字数：190 千字　　2005 年 8 月第 1 版
印数：3 001－5 000 册　　2006 年 2 月北京第 2 次印刷

ISBN 7-115-13967-9/TP・4952

定价：16.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223

前　言

本书按照最新全国计算机等级二级考试大纲要求，系统地介绍了全国计算机等级考试二级公共基础的知识点。

全书由4个相互独立和关联的章节组成，力求做到结构合理，概念准确，范例典型，步骤详尽，通俗易懂。第1章是“算法与数据结构基础”，包括算法的基本概念、数据结构基础等内容。主要讲述算法的时间复杂度与空间复杂度的概念和意义、数据的逻辑结构与存储结构、线性链表、栈和队列、树的基础等。第2章是“程序设计基础”，包括计算机语言和程序的基本概念、程序设计方法与风格等内容。主要讲解结构化程序设计和面向对象的程序设计基础等。第3章是“软件工程基础”，包括软件工程基本概念、结构化分析方法和结构化设计方法、软件测试的方法和程序的调试方法等内容。主要讲解软件生命周期、软件工具与软件开发环境的概念，以及数据流图、数据字典、总体设计与详细设计、白盒测试与黑盒测试、单元测试、集成测试和系统测试、静态调试与动态调试等。第4章是“数据库设计基础”，包括数据库的基本概念、关系代数运算和数据库设计方法基础等内容。主要讲解数据库管理系统、数据库系统、数据模型、E-R图，还有集合运算及选择、投影、连接运算，以及数据库规范化理论和数据库设计的需求分析、概念设计、逻辑设计和物理设计的相关知识等。

本书范例典型，图文并茂、深入浅出地分析讲解各个知识点；各章后还配有适量习题供读者选择练习，并附有参考答案以方便自学。

本书可作为大学本科、专科各类学生学习大学计算机基础的辅助教材，也可作为全国计算机等级考试二级公共基础的培训教材，还可作为其他相关技术或就业考试的计算机基础学习的参考书。为便于本书的学习使用，下面给出一个12～18学时的参考教学安排表，教师在讲课时可根据学生的具体情况和学时数来安排自己的课时，适当增减。另外，本书配有电子教案供教学参考，读者可与作者联系索取，作者联系方式：guanqun168@126.com。

本书由管群任主编，曾新任副主编，管斌、刘淑东、贺丹和葛宇编著。由于时间仓促，作者水平有限，书中难免存在疏漏，希望各位同行和读者不吝赐教。

编　者

2005年7月

课时分配表

学时安排	教 学 内 容
2～3	第 1 章　算法与数据结构基础（1） 包括算法的基本概念、数据结构基础、线性表、栈和队列
2～4	第 1 章　算法与数据结构基础（2） 包括树的基本概念、排序和查找基础算法
2～3	第 2 章　程序设计基础 包括程序设计方法与风格、结构化程序设计和面向对象的程序设计方法等知识
2～3	第 3 章　软件工程基础 包括软件工程的基本概念、结构化分析方法、软件设计、软件测试和程序的调试
2～3	第 4 章　数据库设计基础（1） 包括数据库的基础知识、数据模型、E-R 图
2	第 4 章　数据库设计基础（2） 包括关系运算、数据库规范化理论和数据库设计方法和步骤

目　录

第 1 章　算法与数据结构基础 …… 1

1.1　算法的基本概念 …… 1
1.2　数据结构基础 …… 2
1.3　线性表 …… 4
1.3.1　线性表的顺序存储结构 …… 4
1.3.2　线性表的链式存储结构 …… 5
1.4　栈和队列 …… 7
1.4.1　栈及其基本操作 …… 7
1.4.2　队列及其基本操作 …… 8
1.5　树 …… 10
1.5.1　树的基本概念 …… 10
1.5.2　二叉树的基本概念 …… 11
1.5.3　二叉树的存储结构 …… 12
1.5.4　二叉树的遍历 …… 13
1.6　排序 …… 16
1.6.1　选择排序 …… 16
1.6.2　插入排序 …… 18
1.6.3　交换排序 …… 18
1.7　查找 …… 20
1.7.1　顺序查找 …… 20
1.7.2　二分查找 …… 21
本章小结 …… 21
习题 1 …… 22

第 2 章　程序设计基础 …… 26

2.1　程序设计方法与风格 …… 26
2.1.1　计算机程序与程序设计语言 …… 26
2.1.2　编码风格 …… 27
2.2　结构化程序设计 …… 29
2.3　面向对象的程序设计方法 …… 29
本章小结 …… 35
习题 2 …… 35

第 3 章　软件工程 …… 37

3.1 软件工程的基本概念 …… 37
3.2 结构化分析方法 …… 39
3.2.1 结构化分析方法概述 …… 39
3.2.2 数据流图 …… 40
3.2.3 数据字典 …… 42
3.2.4 软件需求规格说明书 …… 44
3.3 软件设计 …… 45
3.3.1 总体设计 …… 45
3.3.2 详细设计 …… 48
3.4 软件测试 …… 51
3.4.1 软件测试方法 …… 51
3.4.2 测试用例的设计 …… 52
3.5 程序的调试 …… 57
3.5.1 静态调试 …… 57
3.5.2 动态调试 …… 57
本章小结 …… 58
习题 3 …… 58

第 4 章 数据库设计基础 …… 61

4.1 数据库的基础知识 …… 61
4.1.1 数据库 …… 61
4.1.2 数据库管理系统 …… 62
4.1.3 数据库系统 …… 63
4.2 数据模型 …… 64
4.2.1 数据模型概述 …… 64
4.2.2 实体联系模型及 E-R 图 …… 67
4.2.3 从 E-R 图导出关系数据模型 …… 69
4.3 关系运算 …… 71
4.3.1 关系代数 …… 71
4.3.2 集合运算 …… 72
4.3.3 关系运算 …… 73
4.4 数据库规范化理论 …… 77
4.4.1 范式 …… 77
4.4.2 规范化的优缺点 …… 78
4.5 数据库设计方法和步骤 …… 79
4.5.1 数据库设计概述 …… 79
4.5.2 数据库设计的步骤 …… 81
本章小结 …… 89
习题 4 …… 90

附录　参考答案…… 104

习题 1 参考答案…… 104
习题 2 参考答案…… 105
习题 3 参考答案…… 106
习题 4 参考答案…… 111

参考文献…… 122

第 1 章　算法与数据结构基础

本章知识要点：

- 算法的基本概念、算法复杂度的概念和意义（时间复杂度与空间复杂度）。
- 数据结构的定义、数据的逻辑结构与存储结构、数据结构的图形表示、线性结构与非线性结构的概念。
- 线性表的定义、线性表的顺序存储结构及其插入与删除运算。
- 栈和队列的定义、栈和队列的顺序存储结构及其基本运算。
- 线性单链表、双向链表与循环链表的结构及其基本运算。
- 树的基本概念、二叉树的定义及其存储结构，二叉树的前序、中序和后序遍历。
- 顺序查找与二分法查找算法、基本排序算法（交换类排序、选择类排序和插入类排序）。

1.1　算法的基本概念

算法的定义：一个有穷的指令集，这些指令为解决某一特定问题规定了一个运算序列，即方法和步骤，在计算机学科中，算法就是计算机解决问题的过程或步骤。

比如，结构化程序算法的特性如下。

（1）输入：有 0 个或多个输入。

（2）输出：有一个或多个输出（处理结果）。

（3）确定性：每步定义都是确切、无歧义的。

（4）有穷性：算法应在执行有限的操作步骤后结束。

（5）有效性：每一条语句都能够有效实现。

常用自然语言、伪代码、流程图、N-S 图和 PAD 图等方法对算法进行描述。此处不详述，读者有兴趣，可以参看相关书籍。

评价程序的算法采用算法复杂度来描述，算法的复杂性是算法效率的度量，是评价算法优劣的重要依据。一个算法的复杂性的高低体现在运行该算法所需要的计算机资源的多少上面，所需的资源越多，就认为该算法的复杂性越高；反之，所需的资源越低，则该算法的复杂性越低。算法复杂度通常采用由德国数学家 Paul Bachmann 在 1892 年提出的“大 O 表达式”表述，该符号以大写字母 O 带一对小括号括起的一个表达式构成。一般描述算法复杂度时括号里用 N 的简单函数表示，如 $O(N^2)$（读作：ON 大平方）。注意，使用大 O 表达式的目的是了解算法性能如何随 N 的变化而变化，它是质的描述，不是量的描述。主要考虑用怎样的一个量来表达一个算法的复杂性；对于给定的一个算法，怎样具体计算它的复杂性。

计算机的资源，最重要的是时间和空间（即存储器）资源。所以，算法的复杂性有时间

复杂性和空间复杂性之分。

算法的时间复杂度指算法的时间耗费，算法时间是由控制结构和原操作的决定的。算法中基本操作重复执行的次数是问题规模 n 的某个函数 f(n)，记作：

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

它表示随问题规模 n 的增大，算法执行时间的增长率和 f(n)的增长率相同。

算法的空间复杂度描述算法的存储空间需求，运行完一个程序所需要的内存大小是问题规模 n 的某个函数 g(n)，记作：

$$S(n) = O(g(n))$$

它表示随着问题规模 n 的增大，算法运行所需存储空间的增长率 S(n)与 g(n)的增长率相同。

例 1-1 和例 1-2 是关于算法具体分析方法的实例。

【例 1-1】下列程序段的时间复杂度为（　　）。

```
for(i=1;i<n;i++){
    y=y+1;                    ①
    for(j=0;j<=(2*n);j++)
        x++;                  ②
}
```

（A）O(n−1)　　（B）O(2n)

（C）$O(n^2)$　　（D）O(2n+1)

分析：语句的频度指的是该语句重复执行的次数。一个算法中所有语句的频度之和构成了该算法的运算时间。在本例算法中，其中语句①的频度是 n−1，语句②的频度是 $(n-1)(2n+1)=2n^2-n-1$，则该程序段的时间复杂度 $T(n)=2n^2-n-1=O(n^2)$。综上所述，答案选 C。

参考答案是 C。

【例 1-2】下列程序段的时间复杂度为（　　）。

```
a=0;b=1;          ①
for(i=2;i<=n;i++){    ②
    s=a+b;        ③
    b=a;          ④
    a=s;          ⑤
}
```

（A）O(1)　　（B）O(n)

（C）$O(\log_2^n)$　　（D）$O(n^2)$

分析：语句①的频度是 2；语句②的频度是 n；语句③的频度是 n−1；语句④的频度是 n−1；语句⑤的频度是 n−1。则该程序段的时间复杂度 T(n)=2+n+3*(n−1)=4n−1=O(n)。

参考答案是 B。

1.2 数据结构基础

数据：数据是信息的载体，是描述客观事物的数、字符，以及所有能输入到计算机中，

被计算机程序识别和处理的符号的集合。数据包括数值性数据和非数值性数据。

数据元素（也称为元素、接点、顶点、记录）：数据元素是数据的基本单位。一个数据元素可以由若干个数据项组成。

数据项：数据项是具有独立含义的最小标识单位。

数据对象：数据的子集，是具有相同性质的数据成员（数据元素）的集合。

例如，整数数据对象 N = { 0，±1，±2，…}。

数据结构的定义：数据结构是指数据之间的相互关系，即数据的组织形式，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：Data_Structure={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。它包括逻辑结构、存储结构和数据的运算 3 个方面的内容。

数据的逻辑结构：用来描述数据元素之间的逻辑关系。

数据的存储结构：用来描述数据元素及数据元素之间的关系在存储器中的存储形式。

数据的运算：即对数据元素施加的操作。

数据结构的图形表示：用图形来直观地表示数据及其之间的关系。例如，一个学生选课系统中相关的数据结构可以表示为如图 1-1 所示。

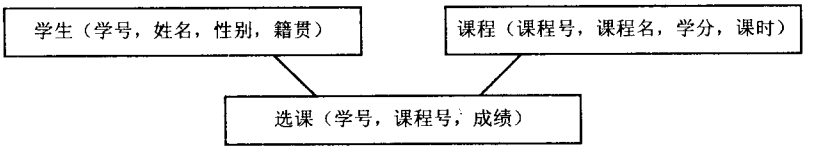

图 1-1　数据结构的图形表示

【例 1-3】_____是信息的载体，它能够被计算机识别、存储和加工处理。

（A）数据　　（B）数据元素

（C）结点　　（D）数据项

分析：数据元素是数据的基本单位。数据项是具有独立含义的最小标识单位。而数据则是各种存储信息的总称。因此选 A。

参考答案是 A。

数据的逻辑结构分为线性结构和非线性结构两类：

线性结构：数据元素之间构成一种顺序的线性关系，如图 1-2 所示。线性结构包括线性表、堆栈、队列和串。

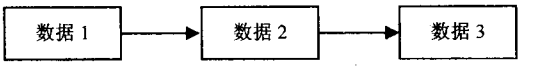

图 1-2　数据元素的线性关系

线性结构的特征有：

（1）集合中必存在惟一的一个“第一元素”；

（2）集合中必存在惟一的一个“最后元素”；

（3）除最后的元素之外，均有惟一的后继；

（4）除第一元素之外，均有惟一的前驱。

非线性结构是指不满足以上条件的存储结构。非线性结构包括树、二叉树、 图（或网络）和广义表。

1.3 线 性 表

1.3.1 线性表的顺序存储结构

线性表是最常用且最简单的一种数据结构。一个线性表是 n 个数据元素的有限序列。在同一线性表中的数据元素必定具有相同的特性。表中的数据元素可以是单个字母字符，也可以是整数、字符串，甚至更复杂的任何类型，通常在程序中用 ElemType 来表示相应的类型。

采用顺序存储结构的线性表也叫做顺序表，如 n 个元素的线性表可以记为：$L=(a_1, a_2, \cdots, a_n)$。其中，i=2，3，…，n 时，$a_{i-1}$ 是 a_i 的直接前驱，a_{i+1} 是 a_i 的直接后继；n 定义为线性表 L 的长度，n=0 时称为空表。假如 n=5，则线性表 L 的顺序存储结构可以表示为如图 1-3 所示。

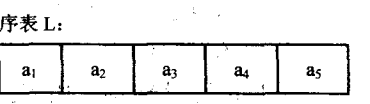

图 1-3 顺序表

由于高级程序设计语言中的数组类型具有随即存取的特性，因此，通常都用数组来描述顺序表。

【例 1-4】 一个顺序表的第一个元素的存储地址是 100，每个元素的长度为 2，那么第 5 个元素的地址为（　　）。

（A）110　　（B）108

（C）100　　（D）120

分析：在顺序表中，每个元素在存储器中占用的空间大小相同，若第一个元素存放的位置是 $LOC(a_1)$，每个元素占用的空间大小为 c，则元素 a_i 的存放位置为：$LOC(a_i)= LOC(a_1)+(i-1)*c$ （$1 \leqslant i \leqslant n$）。把题中各值代入公式可得第 5 个元素的地址为 108。综上所述，答案选 B。

参考答案是 B。

在 C 语言中顺序表可以描述如下：

```
typedef    struct{
ElemType   *elem;      //存储空间的基地址
int        length;     //当前表长度
int        listsize;   //当前分配的存储容量
}SqList;
```

顺序表有插入和删除两种基本操作。

插入操作定义为 Insert_Sq(L;i;b)，其含义为，在给定的线性表 L 中的第 i 个元素之前插入元素 b，其中 i 要满足 $1 \leqslant i \leqslant length+1$。插入元素时，顺序表的逻辑结构会发生变化，表的长度增加 1。例如，在 i 位置插入元素 e 后，$(a_1, \cdots, a_{i-1}, a_i, \cdots, a_n)$ 改变为 $(a_1, \cdots, a_{i-1}, e, a_i, \cdots, a_n)$。插入的具体方法是，将 a_n（L.elem[n−1]）至 a_i（L.elem[i−1]）的元素依次向后移一位，然后在原来 a_n（L.elem[i−1]）的位置放入 e，如图 1-4 所示。

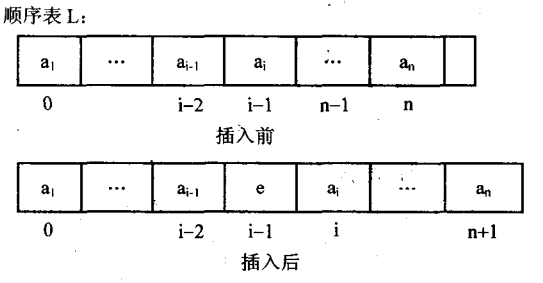

图 1-4 顺序表的插入

该算法在最好的情况下的时间复杂度是 O(1)（即在表尾插入），最坏时间复杂度是 O(n)（即在表头插入），因此平均时间复杂度为 O(n)。

删除操作定义为 Delete_Sq(L,i,&e)，其含义为，在给定的线性表 L 中删除第 i 个元素，i 的范围是 1<i<L.length，并将被删除元素存储到 e 所指的地址处。删除后，顺序表的逻辑结构发生变化，表的长度减少 1。例如，在 i 位置删除元素 a_i 后，（a_1，…，a_{i-1}，a_i，a_{i+1}，…，a_n）改变为（a_1，…，a_{i-1}，a_{i+1}，…，a_n）。具体操作方法：将第 i 个元素 a_i 复制到 e 中，并将 a_{i+1} 至 a_n 依次向前移动一位。

同样，该算法在最好的情况下时间复杂度是 O(1)，最坏时间复杂度是 O(n)，平均时间复杂度为 O(n)。

顺序表具有以下缺点。

（1）在插入或删除操作时，需要移动大量的元素。

（2）在给长度变化较大的顺序表预先分配空间时，必须按最大空间分配，造成一定程度上的的浪费。

（3）表的容量难以扩充。

为解决这些问题，引入了线性链表。

1.3.2　线性表的链式存储结构

链式存储是指用一组地址任意的存储单元存放线性表中的数据元素。在链式存储结构中，表中各元素的存储位置不一定相邻，它们可以随意分散在内存中，从而克服了顺序表的弱点，也就是插入和删除时，不需要移动大量的元素，而且，也不需要预先分配足够大的存储空间。链式存储的缺点在于不能像顺序表那样可以随即存取。

链式存储采用结点来表示数据元素。一个结点由两个部分构成：数据域和指针域。其中数据域存储数据元素，指针域指示后继或前驱元素的存储位置，通常有一个头结点，不含数据，其指针域的指针指向第一个数据结点。常见的链表有单链表、双向链表和循环链表。

（1）单链表及其基本操作

单链表是链式存储结构中最简单最基本的结构，其指针域包括指向该接点的直接后继指针。单链表的结构如图 1-5 所示。

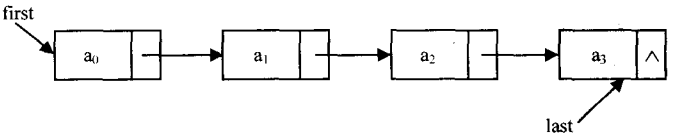

图 1-5　单链表

单链表在 C 语言中可以如下描述：

```
Typedef struct LNode{
ElemType   data;          //数据域
struct Lnode   * next;    //指针域
}LNode,* LinkList;
LinkList  L;              //L 为单链表的头指针。
```

单链表有查找、插入和删除 3 种基本操作。

由于在单链表中，任何两个元素的存储位置之间没有固定的联系，为查找到第 i 个数据元素，必须先找到第 i-1 个数据元素。其具体查找过程为：设定一个计数器 j=1，一个指针 p 指向表的第一个位置，指针每向后移动一位，j 增加 1，直到 j=i。算法时间复杂度为 O(ListLength(L))，ListLength(L)表示 L 的长度。

要在位置 i 处插入元素 x，有序对<a_{i-1}，a_i>改变为<a_{i-1}，x>和<x，a_i>。在连表中插入结点只需要修改指针。具体操作为：找到第 i-1 个结点，修改后继指针使其指向新结点，并使新结点的后继指针指向第 i 个结点。其插入示意图如图 1-6 所示，其算法的时间复杂度为 O(ListLength(L))。

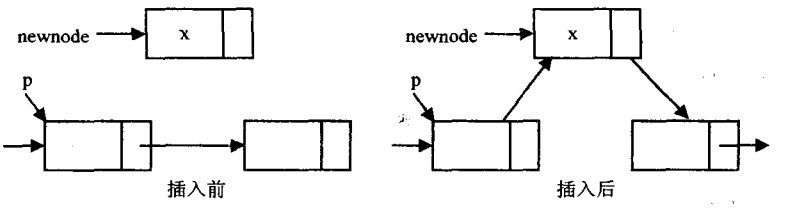

图 1-6　单链表的插入

删除位置 i 处的元素。具体操作为：找到单链表中第 i-1 个结点，使其后继指针指向第 i+1 个结点。其删除示意图如图 1-7 所示，其算法的时间复杂度为 O(ListLength(L))。

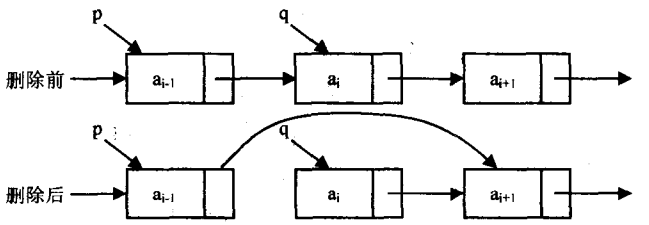

图 1-7　单链表的删除

（2）双向链表及其基本操作

指针域包括该结点的直接前驱和直接后继两个指针的存储结构链表，称为双向链表。

双向链表用 C 语言描述如下：

```
Typedef struct DuLNode{
 ElemType data;              //数据域
 struct   DuLNode * prior;   //指向前驱的指针
 struct   DuLNode * next;    //指向后继的指针
}DulNode,* DuLinkList;
```

在双向链表中查找和单链表相同。而插入和删除则需要同时修改两个方向上的指针。在第 i 个结点前插入 1 个新结点的具体操作为：找到表中第 i-1 个结点，使其后继指针指向新结点，修改新结点的前驱指针使其指向第 i-1 个结点；使新结点的后继指针指向第 i 个结点，使第 i 个结点的前驱指针指向新结点。其插入示意图如图 1-8 所示，其算法的时间复杂度为 O(ListLength(L))。

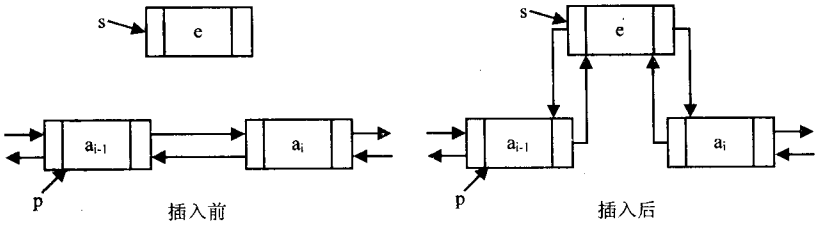

图 1-8　双向链表的插入

删除第 i 个元素的具体操作为：找到表中第 i-1 个元素，修改使其后继指针指向第 i+1 个元素，将第 i+1 个元素的前驱指针修改为指向第 i-1 个元素。算法的时间复杂度为 O(ListLength(L))。

（3）循环链表

循环链表是一种首尾相接的链表，它的特点是表中最后一个结点的指针域指向头结点，整个链表形成一个环。在循环链表中，可以从任何一个结点出发查找表中其他的结点，而非循环链表则是从头结点处出发查找其他结点。循环链表有单向循环链表和双向循环链表等。

1.4　栈 和 队 列

栈和队列是两种常用的数据类型。通常称栈和队列是限定插入和删除操作只能在表的“端点”进行的线性表。

1.4.1　栈及其基本操作

栈是规定只能在表的一端进行插入和删除的线性表。允许插入和删除的一端称为栈顶，另一端称为栈底。当表中没有元素时称为空栈。由于栈的插入和删除运算仅在栈顶一端进行，后进栈的元素必定先被删除，所以又把栈称为后进先出（Last_In_First_Out，简称 LIFO）表。栈有插入（即进栈）和删除（即出栈）两种基本操作。

顺序栈类似于线性表的顺序结构，是利用一组地址连续的存储单元依次存放从栈底到栈顶的若干数据元素。指向表尾的指针可以作为栈顶指针。

栈的顺序存储定义如下：

```
#define STACK_INIT_SIZE 100; //默认初始栈的大小
typedef struct{
  SElemType * base;      //栈底指针
  SElemType * top;       //栈顶指针
  int stacksize;         //栈的大小
}SqStack;
```

顺序栈的插入算法可以描述如下：将待插元素 e 存入栈顶指针指向的位置，然后将栈顶指针加 1。插入也叫入栈或进栈。

顺序栈的删除算法可以描述如下：如需保存栈顶元素 e，则将 e 复制到目标变量，然后直接将栈顶指针减 1。

顺序栈的存储及其操作如图 1-9 所示。

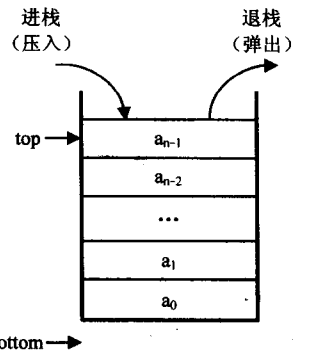

图 1-9　顺序栈的存储及其操作图示

【例 1-5】若已知一个栈的入栈序列是 1，2，3，…，n，其输出序列为 P_1，P_2，P_3，…，P_n，若 P_1=n，则 P_i 为（　　）。

（A）i　　（B）n−1

（C）n−i+1　　（D）不确定

分析：当 P_1=n，即 n 是最先出栈的，根据栈的原理，n 必定是最后入栈的，那么输入序列必定是 1，2，3，…，n，则出栈的序列是 n，…，3，2，1。综上所述，答案选 C。

参考答案是 C。

【例 1-6】对于一个栈，给出输入项 a，b，c。如果输入项序列由 a，b，c 所组成，则不可能产生的输出序列是（　　）。

（A）cab　　（B）abc

（C）bac　　（D）cba

分析，本题根据栈的“后进先出”的特点，有以下几种情况：

① a 进 a 出 b 进 b 出 c 进 c 出，产生输出序列 abc；

② a 进 a 出 b 进 c 进 c 出 b 出，产生输出序列 acb；

③ a 进 b 进 b 出 a 出 c 进 c 出，产生输出序列 bac；

④ a 进 b 进 b 出 c 进 c 出 a 出，产生输出序列 bca；

⑤ a 进 b 进 c 进 c 出 b 出 a 出，产生输出序列 cba。不可能产生的输出序列为 cab。综上所述，答案选 A。

参考答案是 A。

1.4.2　队列及其基本操作

队列可以看作是插入在一端进行，删除在另一端进行的线性表。允许插入的一端称为队尾，允许删除的一端称为队头。在队列中，只能删除队头元素，队列的最后一个元素一定是最新入队的元素，因此队列又称为先进先出（First_In_First_Out,FIFO）表。队列的基本操作有插入（即入队）和删除（即出队）两种，如图 1-10 所示。

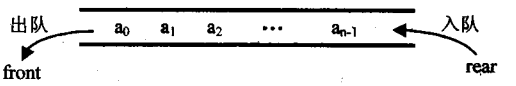

图 1-10　顺序队列

在队列的顺序存储结构中，除了用一组地址连续的存储单元依次存放从队列头到队列尾的数据元素之外，还要设置两个指针 front 和 rear 分别指示队头元素和队尾元素的位置。空队列时，front=rear=0；在非空队列中，头指针 front 始终指向队列头元素，而尾指针 rear 始终指向队列尾元素的下一个位置。为了充分利用存储空间，可以让队列首尾相连，构成一个循环队列，这样当队列的最后一个单元已占用时（rear=MAXNUM），只要第一个单元是空的就可以加入欲入队的元素。

图 1-11 为循环队列的删除和插入示意图。

循环队列的存储表示为：

```
#define MAXQSIZE 100      //最大队列长度
```

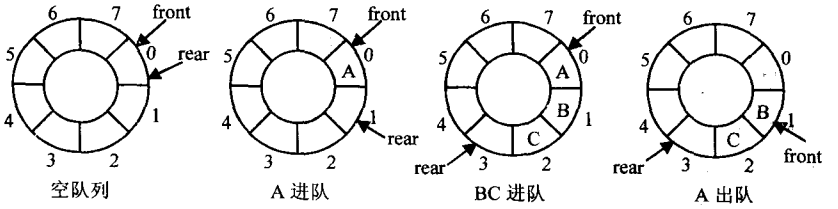

图 1-11　循环队列的插入和删除

```
Typedef struct{
QElemType * base;          //动态分配存储空间
int * front;               //头指针，若队列不空，则指向队列头元素
int * rear;                //尾指针，若队列不空，则指向队尾元素的下一个位置
int queuesize;             //队列长度
}SqQueue
```

循环队列的插入算法：

```
Status EnQueue(SqQueue & Q,QElemType e){
//插入元素 e 为 Q 的队尾元素
if((Q.rear+1)%Q.queuesize==Q.front)
return ERROR;                          //队列满
Q.base[Q.rear]=e;
Q.rear=(Q.rear+1)%Q.queuesize;         //元素插入到队尾
return OK;
}
```

循环队列的删除算法：

```
Status DeQueue(SqQueue &Q,QElemType &e){
//若队列不空，则删除 Q 的队头元素，用 e 返回其值
if(Q.front==Q.rear)
return EMPTYQUEUE;                     //队列为空
e=Q.base[Q.front];
Q.front=(Q.front+1)%Q.queuesize;
return OK;
}
```

【例 1-7】以下哪一个不是队列的基本运算（　　）。

（A）从队尾插入一个新元素　　（B）从队列中删除第 i 个元素

（C）判断一个队列是否为空　　（D）读取队头元素的值

分析：队列是一种特殊的线性表，这种表规定只能在表的一端进行插入，而在另一端进入删除运算，它不可能在队列中间进行插入，删除操作。综上所述，答案选 B。

参考答案是 B。

【例 1-8】设栈 S 和队列 Q 的初始状态为空，元素 e_1，e_2，e_3，e_4，e_5 和 e_6 依次通过栈 S，一个元素出栈后即进入队列 Q，若 6 个元素出队的顺序是 e_2，e_4，e_3，e_6，e_5，e_1，则栈 S 的