全国计算机等级考试

考点解析、例题精解与实战练习

——二级公共基础知识

NCRE研究组

高等教育出版社
Higher Education Press

全国计算机等级考试考点解析、例题精解与实战练习

——二级公共基础知识

NCRE 研究组

高等教育出版社

内容提要

本书是按照教育部考试中心颁布的最新考试大纲和指定教材编写的。全书分5章来讲解计算机等级考试二级公共基础的考点与试题，章节安排与最新教育部考试中心指定教材（2008年版）同步，主要从考试大纲要求、考试要点、典型例题分析和专项习题训练几个方面来对该部分内容进行系统的阐释。涉及的内容主要有数据结构与算法、程序设计基础、软件工程基础、数据库设计基础、笔试模拟试卷及答案分析等。

本书具有考点浓缩、例题典型、讲解细致等特点，非常适合参加全国计算机等级考试（二级公共基础知识）的人员考前复习使用，也适合其他相关人员及等级考试培训班使用。

图书在版编目(CIP)数据

全国计算机等级考试考点解析、例题精解与实战练习：二级公共基础知识／NCRE研究组. —北京：高等教育出版社，2008.3
ISBN 978-7-04-023865-5

Ⅰ.全… Ⅱ.N… Ⅲ.电子计算机—水平考试—自学参考资料 Ⅳ.TP3

中国版本图书馆CIP数据核字(2008)第028969号

策划编辑 何新权 **责任编辑** 彭立辉 **封面设计** 张志奇 **版式设计** 陆瑞红
责任校对 王效珍 **责任印制** 韩 刚

出版发行 高等教育出版社
社 址 北京市西城区德外大街4号
邮政编码 100011
总 机 010—58581000

经 销 蓝色畅想图书发行有限公司
印 刷 北京民族印刷厂

开 本 787×1092 1/16
印 张 11.25
字 数 280 000

购书热线 010-58581118
免费咨询 800-810-0598
网 址 http://www.hep.edu.cn
http://www.hep.com.cn
网上订购 http://www.landraco.com
http://www.landraco.com.cn
畅想教育 http://www.widedu.com

版 次 2008年4月第1版
印 次 2008年4月第1次印刷
定 价 22.00元

物料号 23865-00

郑重声明

前　言

全国计算机等级考试自1994年举办以来，得到了社会各界的广泛认可，在推广、普及计算机应用知识和技术，为用人单位录用和考核工作人员提供评价标准等方面发挥了重要的作用。全国计算机等级考试是目前国内参加人数最多、影响最大的计算机类考试。

为适应当前信息技术的飞速发展，国家教育部考试中心对全国计算机等级考试的考试科目及内容进行了重大调整，对考试大纲进行了全面修订。为了更好地服务于考生，引导考生尽快掌握计算机的先进技术，并顺利通过计算机等级考试，配合新考试大纲的推出，我们特别编写了本书。

本书分5章来讲解计算机等级考试二级公共基础知识的考点与试题，章节安排与教育部考试中心最新指定教材（2008年版）同步，主要从考试大纲要求、考试要点、典型例题分析和专项习题训练几个方面来对该部分内容进行系统的阐释。涉及的内容主要有数据结构与算法、程序设计基础、软件工程基础、数据库设计基础、笔试模拟试卷及答案分析等。

与目前已出版的同类图书相比，本书具有如下特色：

● 考点浓缩精解，重点突出。本书将指定的考试内容进行浓缩，用言简意赅的语言精讲考试要点、重点和难点，从而使考生更易于理解全国计算机等级考试的要求和范围，能在较短的时间内取得较大的收获。

● 例题选取精心，分析到位。书中的例题一部分选自近年全国计算机等级考试的真题，一部分是根据最新考试要求精心设计而成，具有典型性和针对性。所有例题均给出了详尽的分析，便于考生掌握完整的解题思路，以达举一反三、触类旁通之功效。

● 实战练习丰富，附有答案。本书立足考试实战，每个章节均配有练习题，这些练习题帮助考生逐段巩固、提高所学内容。最后，还提供了15套模拟试题，便于读者检测自己的总体水平。所有练习题、模拟题均配有答案，便于自测使用。

本书非常适合参加全国计算机等级考试（二级公共基础知识）的人员考前复习使用，也适合其他相关人员及等级考试培训班使用。

为方便读者学习，书中将重要考点或高频考点用“*”标记。

编　者

目　录

第1章

数据结构与算法

大纲要求

① 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）。
② 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。
③ 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。
④ 栈和队列的定义、栈和队列的顺序存储结构及其基本运算。
⑤ 线性单链表、双向链表与循环链表的结构及其基本运算。
⑥ 树的基本概念、二叉树的定义及其存储结构以及二叉树的前序、中序和后序遍历。
⑦ 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、插入类排序）。

重要考点提示

根据对历年真题的分析可知，本章考核内容约占13%，主要包括以下几个方面：
① 算法复杂度。
② 栈、队列、线性链表的基本概念。
③ 二叉树的存储结构。
④ 线性表、树的结点计算和遍历。
⑤ 冒泡排序的最坏次数计算。

1.1 算　法

考点1　算法的基本概念

所谓算法是指对解题方案的准确而完整的描述。

对于一个问题，如果可以通过一个计算机程序，在有限的存储空间内运行有限长的时间而得到正确的结果，则称这个问题是算法可解的。但算法不等于程序，也不等于计算方法。当然，程序也可以作为算法的一种描述，但通常还需考虑很多与方法和分析无关的细节问题，这是因为在编写程序时要受到计算机系统运行环境的限制。通常，程序的编制不可能优于算法的设计。

1. 算法的基本特征

作为一个算法，一般应具有以下4个基本特征：

（1）可行性（effectiveness）

针对实际问题设计的算法，人们总是希望能够得到满意的结果。但一个算法又总是在某个特定的计算工具上执行的，因此算法在执行过程中往往要受到计算工具的限制，使执行结果产生偏差。算法与计算公式是有差别的，在设计一个算法时，必须考虑它的可行性，否则将得不到满意的结果。

（2）确定性（definiteness）

算法的确定性是指算法中的每一个步骤必须有明确的定义，不能产生歧义。这一性质也反映了算法与数学公式的明显差别。

（3）有穷性（finiteness）

算法的有穷性，是指算法必须能在有限的时间内做完，即算法必须能在执行有限个步骤之后终止。算法的有穷性还包括合理的执行时间的含义，因为如果一个算法需要执行千万年，显然失去了价值。

（4）拥有足够的情报

一个算法是否有效，还取决于为算法所提供的情报是否足够。一般来说，当算法拥有足够的情报时，此算法才是有效的，而当提供的情报不够时，算法可能无效。

2. 算法的基本要素

一个算法通常由两种基本要素组成：一是对数据对象的运算和操作，二是算法的控制结构。

（1）算法中对数据的运算和操作

每个算法实际上是按解题要求，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。因此，计算机算法就是计算机能处理的操作所组成的指令序列。

通常，计算机可以执行的基本操作以指令形式来描述，所有指令的集合构成计算机指令系统。计算机程序就是按解题要求从计算机指令系统中选择合适的指令所组成的指令序列。在一般的计算机系统中，基本的运算和操作有以下4类：

① 算术运算：主要包括加、减、乘、除等运算。

② 逻辑运算：主要包括与、或、非等运算。

③ 关系运算：主要包括大于、小于、等于、不等于等运算。

④ 数据传输：主要包括赋值、输入、输出等操作。

（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，而且还与各操作之间的执行顺序有关。算法中各个操作间的执行顺序称为算法的控制结构。

算法的控制结构给出了算法的基本框架，它不仅决定了算法中各操作的执行顺序，而且也直接反映了算法的设计是否符合结构化原则。描述算法的工具通常有传统流程图、N－S结构化流程图、算法描述语言等。一个算法一般都可以用顺序、选择（也称分支）、循环（也称重复）3种基本控制结构组合而成。

3. 算法设计的基本方法

计算机解题的过程实际上是实施某种算法，这种算法称为计算机算法。计算机算法不同于人工处理的方法。

本节介绍工程上常用的几种算法设计方法，在实际应用时，各种方法之间往往存在着一定的联系。

（1）列举法

列举法的基本思想是根据提出的问题，列举所有可能的情况，并用问题中给定的条件检验哪些是需要的，哪些是不需要的。

列举法的特点是算法比较简单。但当列举的可能情况较多时，执行列举算法的工作量将会很大。因此，在用列举法设计算法时，应该重点注意方案的优化，尽量减少运算的工作量。

列举原理是计算机应用领域中十分重要的原理。许多实际问题，若采用人工列举是不可想

像的，但由于计算机的运算速度快，擅长重复操作，因此可以很方便地进行大量列举。列举算法虽然是一种比较笨拙而原始的方法，但在有些实际问题中，局部使用列举法却是很有效的，因此列举算法是计算机算法中的一个基础算法。

（2）归纳法

归纳法的基本思想是通过列举少量的特殊情况，经过分析最后找出一般的关系。从本质上讲，归纳就是通过观察一些简单而特殊的情况，最后总结出一般性的结论。

归纳是一种抽象，即从特殊现象中找出一般的关系。但由于在归纳过程中不可能对所有的情况进行列举，因此最后由归纳得到的结论还只是一种猜测，还需要对这种猜测加以必要的证明。实际上，通过精心观察而得到的猜测得不到证实或最后证明猜测是错误的情况也很常见。

（3）递推法

所谓递推，是指从已知的初始条件出发，逐次推出所要求的各中间结果和最后结果。其中，初始条件或者问题本身已经给定，或者通过对实际问题的分析与化简而确定。递推本质上也属于归纳法，工程上许多递推关系式实际上是通过对实际问题的分析与归纳而得到的，因此递推关系式往往是归纳的结果。

递推算法在数值计算中是极为常见的，但是对于数值型的递推算法必须要注意数值计算的稳定性问题。

（4）递归法

递归的基本思想是在解决复杂问题时，将问题逐层分解，降低问题的复杂程度，直到最后那些最简单的问题被解决后，再沿着原来分解的逆过程逐步进行综合。由此可以看出，递归的基础也是归纳。在工程实际中，有许多问题就是用递归来定义的，数学中的许多函数也是用递归来定义的。递归在可计算性理论和算法设计中占有很重要的地位。

递归分为直接递归与间接递归两种方式。如果一个算法 P 显式地直接调用自己则称为直接递归。如果算法 P 调用另一个算法 Q，而算法 Q 又调用算法 P，则称为间接递归调用。

递归是很重要的算法设计方法之一。实际上，递归过程能将一个复杂的问题归结为若干个比较简单的问题，然后将这些比较简单的问题再归结为更简单的问题，此过程可以一直做下去，直到出现最简单的问题为止。

有些实际问题，既可以归纳为递推算法，又可以归纳为递归算法，但递推与递归的实现方法是不一样的。递推是从初始条件出发，逐次推出所需求的结果；而递归则是从算法本身到达递归边界的。通常，递归算法要比递推算法清晰易读，其结构比较简练。特别是在许多比较复杂的问题中，很难找到从初始条件推出所需结果的全过程，此时设计递归算法要比递推算法容易得多。但递归算法的执行效率比较低。

（5）减半递推技术

实际问题的复杂程度往往与问题的规模有密切的联系。因此，利用分治法解决这类问题是有效的。所谓分治法，就是对问题分而治之。工程上常用的分治法是减半递推技术。

所谓减半，是指将问题的规模减半，而问题的性质不变；所谓递推，是指重复减半的过程。

下面举例说明利用减半递推技术设计算法的基本思想。

设方程 $f(x)=0$ 在区间 $[a, b]$ 上有实根，且 $f(a)$ 与 $f(b)$ 异号。利用二分法求方程实根的减半递推过程如下：

首先，取给定区间的中点 $c=(a+b)/2$。

然后，判断$f(c)$是否为0。若$f(c)=0$，则说明c即为所求的根，求解过程结束；如果$f(c)\neq 0$，则根据以下原则将原区间减半：

- 若$f(a)f(c)<0$，则取原区间的前半部分。
- 若$f(b)f(c)<0$，则取原区间的后半部分。

最后，判断减半后的区间长度是否已经很小：

- 若$|a-b|<\varepsilon$，则过程结束，取$(a+b)/2$为根的近似值。
- 若$|a-b|\geq\varepsilon$，则重复上述的减半过程。

(6) 回溯法

前面讨论的递推和递归算法本质上是对实际问题进行归纳的结果，而减半递推技术也是归纳法的一个分支。在工程上，有些实际问题很难归纳出一组简单的递推公式或直观的求解步骤，并且也不可能进行无限的列举。对于这类问题，一种有效的方法是“试”。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，若试探成功，就得到问题的解；若试探失败，就逐步退回，换别的路线再逐步试探。这种方法称为回溯法，回溯法在处理复杂数据结构方面有着广泛的应用。

*考点2 算法的复杂度

算法复杂度主要包括时间复杂度和空间复杂度。

1. 算法的时间复杂度

所谓算法的时间复杂度，是指执行算法所需要的计算工作量。

为了能够比较客观地反映出一个算法的效率，在度量一个算法的工作量时，不仅应该与所使用的计算机、程序设计语言以及程序编制者无关，而且还应该与算法实现过程中的许多细节无关。为此，可以用算法在执行过程中所需基本运算的执行次数来度量算法的工作量。

算法所执行的基本运算次数还与问题的规模有关。在同一个问题规模下，如果算法执行所需的基本运算次数取决于某一特定输入，可以用以下两种方法来分析算法的工作量。

(1) 平均性态 (average behavior)

所谓平均性态分析，是指用各种特定输入条件下的基本运算次数的加权平均值来度量算法的工作量。设x是所有可能输入中的某个特定输入，$p(x)$是x出现的概率（即输入为x的概率），$t(x)$是算法在输入为x时所执行的基本运算次数，则算法的平均性态定义为：

$$A(n)=\sum_{x\in D_n}p(x)*t(x)$$

其中，D_n表示当规模为n时，执行算法时所有可能输入的集合；$t(x)$可以通过分析算法来加以确定；而$p(x)$必须由经验或用算法中有关的一些特定信息来确定，通常不能解析地加以计算。如果确定$p(x)$比较困难，会给平均性态的分析带来困难。

(2) 最坏情况复杂性 (worst-case complexity)

所谓最坏情况分析，是指在规模为n时，算法所执行的基本运算的最大次数，定义为：

$$W(n)=\max_{x\in D_n}\{t(x)\}$$

显然，$W(n)$的计算要比$A(n)$的计算方便得多。由于$W(n)$实际上是给出了算法工作量的一个上界，因此它比$A(n)$更具有实用价值。

2. 算法的空间复杂度

一个算法的空间复杂度，一般是指执行这个算法所需要的内存空间。

一个算法所占用的存储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需要的额外空间。其中，额外空间包括算法程序执行过程中的工作单元以及某种数据结构所需要的附加存储空间。如果额外空间相对于问题规模来说是常数，则称该算法是原地（in place）工作的。在许多实际问题中，为了减少算法所占的存储空间，通常采用压缩存储技术，以便尽量减少不必要的额外空间。

1.2 数据结构的基本概念

考点3 什么是数据结构

计算机已经被广泛用于数据处理，实际问题中的各数据元素之间总是相互关联的。所谓数据处理，是指对数据集合中的各元素以各种方式进行运算，包括插入、删除、查找、更改等运算，也包括对数据元素进行分析。

- 数据元素：在数据处理领域中，每一个需要处理的对象都可以抽象成数据元素。数据元素一般简称为元素。
- 数据对象：性质相同的数据元素的集合是数据的一个子集。
- 数据结构（data structure）：简单地说，数据结构是指相互关联的数据元素的集合，即数据组织形式。所谓结构，就是指数据元素之间的前后件关系。

数据结构包含两方面信息：一是表示数据元素的信息；二是表示各数据元素之间的前后件关系。

一般情况下，在具有相同特征的数据元素集合中，各个数据元素之间存在某种关系（即联系），这种关系反映了该集合中的数据元素所固有的一种结构。在数据处理领域中，通常把数据元素之间这种固有的关系简单地用前后件关系（或直接前驱与直接后继关系）来描述。

前后件关系是数据元素之间的一个基本关系，但前后件关系所表示的实际意义随具体对象的不同而不同。一般来说，数据元素之间的任何关系都可以用前后件关系来描述。

1. 数据的逻辑结构

一种数据的逻辑结构根据需要可以表示成多种存储结构，常用的存储结构有顺序、链接、索引等结构。采用不同的存储结构，其数据处理的效率是不同的。因此，在进行数据处理时，选择合适的存储结构尤为重要。

所谓数据的逻辑结构，是指反映数据元素之间逻辑关系的数据结构。数据的逻辑结构有两个要素：一是数据元素的集合，通常记为 D；二是 D 上的关系，通常记为 R。即一个数据结构可以表示成：

$$B=(D, R)$$

2. 数据的存储结构

数据的逻辑结构在计算机存储空间中的存放形式称为数据的存储结构（也称数据的物理结构）。在进行数据处理时，被处理的各数据元素总是被存放在计算机的存储空间中，而且各数据元素在计算机存储空间中的位置关系与它们的逻辑关系可能不同。

由于数据元素在计算机存储空间中的位置关系可能与逻辑关系不同，因此为了表示存放在计算机存储空间中的各数据元素之间的逻辑关系（即前后件关系），在数据的存储结构中，不仅要存放各数据元素的信息，还需存放各数据元素之间的前后件关系的信息。

一般来说，一种数据的逻辑结构根据需要可以表示成多种存储结构。而采用不同的存储结构，其数据处理的效率是不同的。

考点4 数据结构的图形表示

一个数据结构除了可用二元关系表示外，还可以直观地用图形表示。在数据结构的图形表示中，对于数据集合 ***D*** 中的每一个数据元素用中间标有元素值的方框表示，一般称之为数据结点，简称为结点；为了进一步表示各数据元素之间的前后件关系，对于关系 ***R*** 中的每一个二元组，用一条有向线段从前件结点指向后件结点。在数据结构中，没有前件的结点称为根结点；没有后件的结点称为终端结点（也称为叶子结点）。

例如，一年四季的数据结构可以用如图 1.1 所示的图形来表示。

通常，一个数据结构中的元素结点可能是动态变化的。根据需要或者在数据结构进行处理的过程中，不仅数据结构中的结点（即数据元素）个数在动态地变化，而且各数据元素之间的关系也有可能在动态地变化。

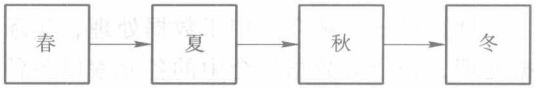


图 1.1 一年四季的数据结构的图形表示

*考点5 线性结构与非线性结构

如果在一个数据结构中没有一个数据元素，则称该数据结构为空的数据结构。在一个空的数据结构中插入一个新的元素后就变为非空；在只有一个数据元素的数据结构中，将该元素删除后就变为空的数据结构。

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两大类型：线性结构与非线性结构。

如果一个非空的数据结构满足下列两个条件：第一，有且只有一个根结点；第二，每一个结点最多有一个前件，也最多有一个后件，则称该数据结构为线性结构。线性结构又称线性表。

特别需要说明的是，在一个线性结构中插入或删除任何一个结点后还应是线性结构。图 1.2 所示为一个线性结构的实例。

如果一个数据结构不是线性结构，则称之为非线性结构。对非线性结构的存储与处理比线性结构要复杂得多。线性结构与非线性结构都可以是空的数据结构。一个空的数据结构究竟是属于线性结构还是属于非线性结构需要根据具体情况来确定。图 1.3 所示为一个非线性结构的实例。

图 1.2 线性结构实例

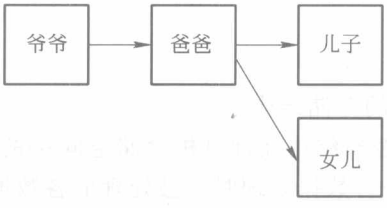


图 1.3 非线性结构实例

1.3 线性表及其顺序存储结构

考点6 线性表的基本概念

线性表（linear list）是最简单、最常用的一种数据结构。

线性表由一组数据元素构成。矩阵是一个比较复杂的线性表。

线性表是（$n \geqslant 0$）个元素 a_1，a_2，…，a_n 组成的一个有限序列，表中的每一个数据元素除了第一个以外的每一个元素，有且只有一个前件；除最后一个元素外，有且只有一个后件。即线性表要么是一个空表，或者可以表示为：

$$(a_1, a_2, \cdots, a_n)$$

其中，$a_i(i=1, 2, \cdots, n)$ 是属于数据对象的元素，通常也称其为线性表中的一个结点。

显然，线性表是一个线性结构。数据元素在线性表中的位置只取决于它们自己的序号，即数据元素之间的相对位置是线性的。

非空线性表有如下一些结构特征：

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。线性表中结点的个数 n 称为线性表的长度。当 $n=0$ 时，称为空表。

*考点7 线性表的顺序存储结构

在计算机中存放线性表，一种最简单的方法是顺序存储，也称为顺序分配。

线性表的顺序存储结构具有以下两个基本特点：

① 线性表中所有元素所占的存储空间是连续的。

② 线性表中各数据元素在存储空间中是按逻辑顺序依次存放的。

图 1.4 所示为线性表顺序结构的示意图。

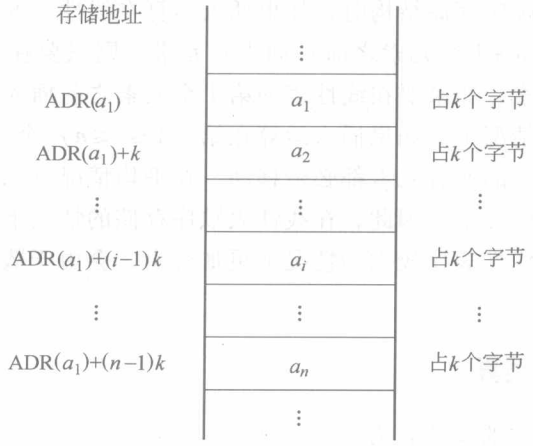


图 1.4 线性表的顺序存储结构

在用一维数组存放线性表时，该数组的长度通常要定义得比线性表的实际长度大一些。在一般情况下，如果线性表的长度在处理过程中是动态变化的，则在开辟线性表的存储空间时要考虑到线性表在动态变化过程中可能达到的最大长度。

在线性表的顺序存储结构下，可以对线性表进行各种处理。主要的运算如下：

① 在线性表的指定位置处加入一个新的元素（线性表的插入）。

② 在线性表中删除指定位置处的数据元素（线性表的删除）。

③ 在线性表中查找某个（或某些）特定的元素（线性表的查找）。

④ 对线性表中的元素进行排序（线性表的排序）。

⑤ 按要求将一个线性表分解成多个线性表（线性表的分解）。

⑥ 按要求将多个线性表合并成一个线性表（线性表的合并）。

⑦ 复制一个线性表（线性表的复制）。

⑧ 逆转一个线性表（线性表的逆转）等。

考点8 顺序表的插入运算

一般来说，设长度为 n 的线性表为：

$$(a_1, a_2, \cdots, a_i, \cdots, a_n)$$

现要在线性表的第 i 个元素 a_i 之前插入一个新元素 b，插入后得到长度为 $n+1$ 的线性表为：

$$(a'_1, a'_2, \cdots, a'_j, a'_{j+1}, \cdots, a'_n, a'_{n+1})$$

则插入前后的两线性表中的元素满足如下关系：

$$a'_j = \begin{cases} a_j & 1 \leqslant j \leqslant i-1 \\ b & j = i \\ a_{j-1} & i+1 \leqslant j \leqslant n+1 \end{cases}$$

在一般情况下，要在第 $i(1 \leqslant i \leqslant n)$ 个元素之前插入一个新元素时，首先要从最后一个（即第 n 个）元素开始，直到第 i 个元素之间共 $n-i+1$ 个元素依次向后移动一个位置，移动结束后，第 i 个位置就被空出，然后将新元素插入到第 i 项。插入结束后，线性表的长度就增加了1。

显然，在线性表采用顺序存储结构时，如果插入运算在线性表的末尾进行，即在第 n 个元素之后（可以认为是在第 $n+1$ 个元素之前）插入新元素，则只要在表的末尾增加一个元素即可，不需要移动表中的元素；如果要在线性表的第1个元素之前插入一个新元素，则需要移动表中所有的元素。在一般情况下，如果插入运算在第 $i(1 \leqslant i \leqslant n)$ 个元素之前进行，则原来第 i 个之后（包括第 i 个元素）的所有元素都必须移动。在平均情况下，要在线性表中插入一个新元素，需要移动表中一半的元素。因此，在线性表顺序存储的情况下，要插入一个新元素，其效率是很低的，特别是在线性表比较大的情况下更加突出，会由于数据元素的移动而消耗较多的处理时间。

考点9 顺序表的删除运算

一般来说，设长度为 n 的线性表为：

$$(a_1, a_2, \cdots, a_i, \cdots, a_n)$$

现要删除第 i 个元素 a_i，删除后得到长度为 $n-1$ 的线性表为：

$$(a'_1, a'_2, \cdots, a'_j, \cdots, a'_{n-1})$$

则删除前后的两线性表中的元素满足如下关系：

$$a'_j = \begin{cases} a_j & 1 \leqslant j \leqslant i-1 \\ a_{j+1} & 1 \leqslant j \leqslant n-1 \end{cases}$$

在一般情况下，要删除第 $i(1 \leqslant i \leqslant n)$ 个元素时，则要从第 $i+1$ 个元素开始，直到第 n 个元素之间共 $n-i$ 个元素依次向前移动一个位置。删除结束后，线性表的长度就减小了 1。

在线性表采用顺序存储结构时，如果删除运算在线性表的末尾进行，即删除第 n 个元素，则不需要移动表中的元素；如果要删除线性表中的第 1 个元素，则需要移动表中所有的元素。在一般情况下，如果要删除第 $i(1 \leqslant i \leqslant n)$ 个元素，则原来第 i 个元素之后的所有元素都必须依次向前移动一个位置。在平均情况下，要在线性表中删除一个元素，需要移动表中一半的元素。因此，在线性表顺序存储的情况下，要删除一个元素，其效率也是很低的，特别是在线性表比较大的情况下更加突出，会由于数据元素的移动而消耗较多的处理时间。

1.4 栈和队列

*考点 10 栈及其基本运算

1. 栈的定义

栈（stack）实际上也是线性表，只不过是一种特殊的线性表。在这种特殊的线性表中，其插入或者删除运算都只能在表的一端进行。

栈是限定在一端进行插入与删除的线性表。

在栈中，允许插入与删除的一端称为栈顶，而不允许插入与删除的另一端称为栈底，当表中没有元素时称为空栈。栈顶元素总是最后被插入的元素，也是最早被删除的元素；栈底元素是最早被插入的元素，也是最晚被删除的元素。即栈的修改原则是先进后出（first in last out，FILO）或后进先出（last in first out，LIFO）。

通常，用指针 top 来指示栈顶的位置，用指针 bottom 来指示栈底的位置。

图 1.5 所示为栈的示意图。

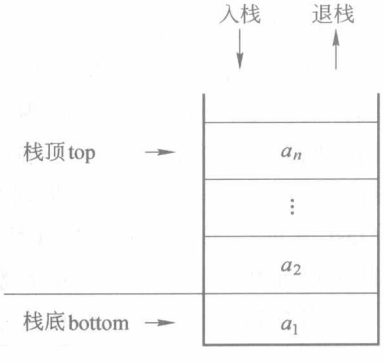
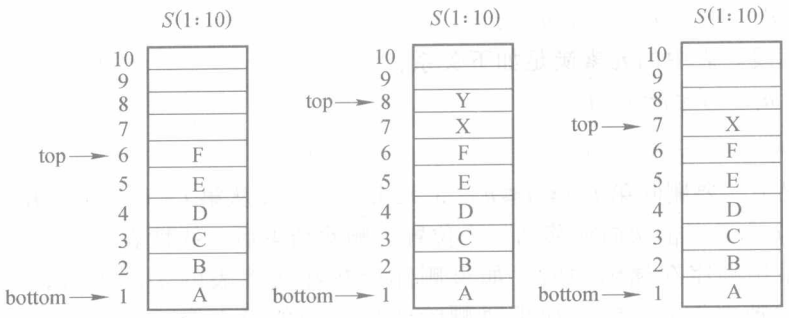


图 1.5 栈示意图

2. 栈的顺序存储及运算

与一般的线性表一样，在程序设计语言中，用一维数组 $S(1:m)$ 作为栈的顺序存储空间，其中 m 为栈的最大容量。通常，栈底指针指向栈空间的低地址一端（即数组的起始地址这一端）。图 1.6（a）所示为容量为 10 的栈顺序存储空间，栈中已有 6 个元素；图 1.6（b）与图 1.6（c）分别为入栈与退栈后的状态。



(a)有6个元素的栈　　(b)插入X与Y后的栈　　(c)退出一个元素后的栈

图 1.6　栈在顺序存储结构下的运算

栈的基本运算有 3 种：入栈、退栈与读栈顶元素。下面分别介绍在顺序存储结构下栈的这 3 种基本运算。

（1）入栈运算

入栈运算是指在栈顶位置插入一个新元素。此运算有两个基本操作：首先将栈顶指针进 1（即 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，说明栈空间已满，不可能再进行入栈操作。这种情况称为栈“上溢”错误。

（2）退栈运算

退栈运算是指取出栈顶元素并赋给一个指定的变量。此运算有两个基本操作：首先将栈顶元素（栈顶指针指向的元素）赋予一个指定的变量，然后将栈顶指针退 1（即 top 减 1）。

当栈顶指针为 0 时，说明栈空，不可能再进行退栈操作。这种情况称为栈“下溢”错误。

（3）读栈顶元素

读栈顶元素是指将栈顶元素赋给一个指定的变量。必须注意，此运算不删除栈顶元素，只是将其赋给一个变量，因此在这个运算中，栈顶指针不会改变。

当栈顶指针为 0 时，说明栈空，读不到栈顶元素。

*考点 11　队列及其基本运算

1. 队列的定义

队列（queue）是指允许在一端进行插入、而在另一端进行删除的线性表。允许插入的一端称为队尾，通常用一个称为尾指针（rear）的指针指向队尾元素，即尾指针总是指向最后被插入的元素；允许删除的一端称为排头（也称为队头），通常也用一个排头指针（front）指向排头元素的前一个位置。队列又称为“先进先出”（first in first out，FIFO）或“后进后出”（last in last out，LILO）的线性表，它体现了“先来先服务”的原则。在队列中，队尾指针与排头指针共同反映了队列中元素动态变化的情况。

当队列中没有元素时称为空队列。图 1.7 所示为具有 6 个元素的队列示意图。

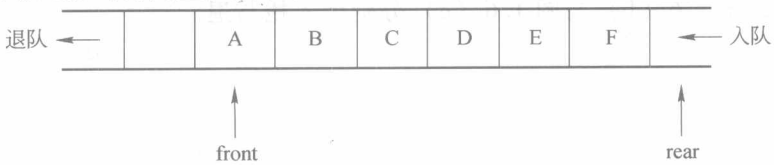


图 1.7　具有 6 个元素的队列示意图