HZ BOOKS
华章科技

全国计算机等级考试命题研究小组最新成果
2014年最新无纸化考试专用指导
赠送价值100元的希赛等考重点培训视频

2014年
全国计算机等级考试
3年真题精解与过关全真训练题

二级Java
语言程序设计

希赛教育等考学院 卢艳芝 主编

紧扣考试大纲，提炼必考点，解析重点难点
连续3年考题训练解析
全真模拟试题与解析
赠送权威专家与辅导名师考试培训教程
在线测试，考前训练，加深记忆

机械工业出版社
China Machine Press

2014年
全国计算机等级考试
3年真题精解与过关全真训练题

二级Java
语言程序设计

希赛教育等考学院 卢艳芝 主编

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

2014年全国计算机等级考试3年真题精解与过关全真训练题：二级Java语言程序设计 / 卢艳芝主编. —北京：机械工业出版社，2013.12

ISBN 978-7-111-44859-4

Ⅰ.2… Ⅱ.卢… Ⅲ.① 电子计算机－水平考试－题解 ② JAVA语言－程序设计－水平考试－题解 Ⅳ.TP3-44

中国版本图书馆CIP数据核字（2013）第276499号

本书由希赛教育等考学院组织编写，作为全国计算机等级考试二级的辅导和培训指定教程。书中内容紧扣全国计算机等级考试2014年考试大纲，通过对历年试题进行科学分析、研究、总结、提炼而成。书中内容全面实用，涵盖了考试大纲规定的所有知识点，对考试大纲规定的内容有重点地进行了细化和深化。阅读本书，就相当于阅读了一本详细的、带有知识注释的考试大纲。准备考试的人员可通过阅读本书掌握考试大纲规定的知识点，掌握考试重点和难点，熟悉内容的分布。

本书适合参加全国计算机等级考试的人员及广大计算机爱好者阅读。

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：高婧雅

北京瑞德印刷有限公司印刷

2014年1月第1版第1次印刷

185mm×260mm • 15.25印张

标准书号：ISBN 978-7-111-44859-4

ISBN 978-7-89405-191-2（光盘）

定 价：39.00元（附光盘）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991 88361066 投稿热线：（010）88379604

购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前 言

全国计算机等级考试（NCRE）由教育部考试中心主办，面向社会，用于考查非计算机专业人员的计算机应用知识与能力。本考试客观、公正，得到了社会的广泛认可。

本书根据全国计算机等级考试2014年考试大纲编写而成，在组织和写作上倾注了笔者的许多精力和心血，相信能够提高考生通过率，有效地为“考试过关”提供帮助。考生可通过阅读本书，迅速掌握考试所涉及的知识点，全面梳理和系统学习考试大纲中的内容。

作者权威，阵容强大

希赛教育（www.educity.cn）专业从事人才培养、教育产品开发、教育图书出版，在职业教育方面具有极高的权威性，特别在在线教育方面名列前茅，远程教育模式也得到了国家教育部门的认可和推广。

希赛教育等考学院是国内知名的大型计算机等级考试在线教育机构，在该领域成绩斐然。我们组织大纲制订者和阅卷组成员编写了考试辅导教材近50本，内容涵盖了计算机等级考试的主要级别。组织权威专家和辅导名师录制了考试培训视频教程，对历年考试进行了跟踪研究和比较研究，编写了权威的全真模拟试题。希赛教育的计算机等级考试培训采取统一教材、统一视频、统一认证教师的形式，采取线下培训与线上辅导相结合的方式，确保学员在通过考试的前提下能真正学到有用的知识。

本书由希赛教育等考学院的卢艳芝主编，参加编写工作的有彭雪阳、胡钊源、张友生、桂阳、王勇、何玉云、张丹、谢顺、刘洋波、余华山，都来自大学教学一线和企业研发团队，具有丰富的教学和辅导经验，对等级考试有深入的研究，具有极强的应试技巧、理论知识、实践经验和责任心。

在线测试，心中有数

希赛网在线测试平台（platform.educity.cn/oletest/）为考生提供了在线测试系统，其中有数十套全真模拟试题和考前密卷，考生可选择任何一套进行测试。

测试完毕，系统会自动判卷，并立即给出分数。对于考生做错的地方，系统会自动记忆，待考生第二次参加测试时，可选择“试题复习”功能。这样，系统会自动把考生原来做错的试题显示出来，供考生重新测试，以加强记忆。

因此，读者可利用希赛网在线测试平台的在线测试系统检查自己的实际水平，加强考前训练，做到心中有数，考试不慌。

随书光盘，物超所值

本书附带的光盘内容为希赛网等考学院（www.educity.cn/ncre/）培训视频教程的节选，本视频教程对考试所有的难点和重点知识进行了精深讲解，可以保证既不漏掉考试必需的知识点，又不加重考生备考负担，使考生轻松、愉快地掌握知识点并领悟考试的真谛。更完整的培训视频教程，请访问希赛教育视频教学平台（platform.educity.cn/v/）。

诸多帮助，诚挚感谢

在本书出版之际，要特别感谢教育部考试中心计算机等级考试办公室的命题专家们，编者在本书中引用了部分考试原题，使本书能够尽量方便读者的阅读。在本书的编写过程中，作者参考了许多相关的文献和书籍，在此对这些参考文献的作者表示感谢。

感谢机械工业出版社李华君老师，他在本书的策划、选题的申报、写作大纲的确定以及编辑、出版等方面付出了辛勤的劳动，给予了我们很多的支持和帮助。

感谢参加希赛教育计算机等级考试辅导和培训的学员，正是他们的想法汇成了本书的源动力，他们的意见使本书更加贴近读者。

由于编者水平有限，且本书涉及的内容面很广，书中难免存在疏漏和不妥之处，编者诚恳地期望各位专家和读者不吝指正和帮助，对此，我们将十分感激。

希赛教育等考学院

目 录

第1章 算法和数据结构

计算机已经被广泛用于数据处理。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操作，包括插入、删除、查找、更改等，也包括对数据元素进行分析。在进行数据处理时，实际需要处理的数据元素一般有很多，而这些大量的数据元素都需要存放在计算机中，因此，大量的数据元素在计算机中如何组织，以便提高数据处理的效率，并且节省计算机的存储空间，这是进行数据处理的关键问题。

算法是一个十分古老的研究课题，然而计算机的出现为这个课题注入了青春和活力，使算法的设计和分析成为计算机科学中最为活跃的研究热点之一。针对实际问题，例如要编制一个高效率的处理程序，那就需要解决如何合理地组织数据、建立合适的数据结构、设计好的算法，以提高程序执行的效率这样的问题。

1.1 考点精讲

根据考试大纲，本章主要考查以下知识点：

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；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）。

1.1.1 算法与数据结构概述

本节的主要考点集中在算法与数据结构的基本概念上，包括算法的基本特征、复杂度，以及数据结构的表示等。

1. 算法的概念

算法（Algorithm）是一系列解决问题的清晰指令，也就是说，能够对一定规范的输入，在有限时间内得到所要求的输出的步骤。如果一个算法有缺陷，或不适合某个问题，执行这个算法将不会解决这个问题。不同的算法完成同样的任务可能有不同的时间、空间或效率。

（1）算法的基本特征

1）有穷性（Finiteness）：算法必须能在执行有限个步骤之后终止。

2）确定性（Definiteness）：算法的每一步骤必须有确切的定义。

3）输入项（Input）：一个算法有0个或多个输入，以刻画运算对象的初始情况，所谓0个输入是指算法本身定出了初始条件。

4）输出项（Output）：一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

5）可行性（Effectiveness）：也称有效性，算法中执行的任何计算步都可以被分解为基本的、可执行的操作步，即每个计算步都可以在有限时间内完成。

（2）算法的基本要素

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。

计算机可以执行的基本操作是以指令的形式描述的。一个计算机系统能执行的所有指令的集合，称为该计算机系统的指令系统。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。

算法的控制结构：一个算法的功能不仅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。算法中各操作之间的执行顺序称为算法的控制结构。

（3）算法设计的基本方法

计算机算法不同于人工处理的方法，下面是工程上常用的几种算法设计。在实际应用时，各种方法之间往往存在着一定的联系。

1）递推法：递推法是利用问题本身所具有的一种递推关系求解问题的一种方法。它把问题分成若干步，找出相邻几步的关系，从而达到目的。

2）递归法：递归法指的是一个过程。函数不断引用自身，直到引用的对象已知。

3）穷举搜索法：穷举搜索法是对可能是解的众多候选解按某种顺序进行逐一枚举和检验，并从中找出那些符合要求的候选解作为问题的解。

4）贪婪法：贪婪法是一种不追求最优解，只希望得到较为满意解的方法。贪婪法一般可以快速得到满意的解，因为它省去了为找最优解要穷尽所有可能而必须耗费的大量时间。贪婪法常以当前情况为基础做最优选择，而不考虑各种可能的整体情况，所以贪婪法不要回溯。

5）分治法：分治法是把一个复杂的问题分成两个或更多相同或相似的子问题，再把子问题分成更小的子问题，直到最后子问题可以简单地直接求解，原问题的解即子问题的解的合并。

6）动态规划法：动态规划法是一种在数学和计算机科学中使用的，用于求解包含重叠子问题的最优化问题的方法。其基本思想是，将原问题分解为相似的子问题，在求解的过程中通过子问题的解求出原问题的解。动态规划的思想是多种算法的基础，被广泛应用于计算机科学和工程领域。

7）迭代法：迭代法是数值分析中通过从一个初始估计解出发寻找一系列近似解来解决问题（一般是解方程或者方程组）的过程，为实现这一过程所使用的方法统称为迭代法。

（4）良好的算法设计的要求

一个良好的算法应达到如下目标。

1）正确性（Correctness）：算法的计算结果必须是正确的。

2）可读性（Readability）：可读性好有助于用户对算法的理解；不易理解的程序容易隐藏较多错误，难以调试和修改。

3）健壮性（Robustness）：当输入数据非法时，算法也能适当地做出反应或进行处理，而不会产生莫名其妙的输出结果。

4）效率与低存储量需求：效率指的是程序执行时，对于同一个问题如果有多个算法可以解决，执行时间短的算法效率高；存储量需求指算法执行过程中所需要的最大存储空间。

2．算法的复杂度

算法复杂度分为空间复杂度和时间复杂度。

（1）算法的时间复杂度

算法的时间复杂度是指执行算法所需要的计算工作量。同一个算法用不同的语言实现，或者用不同的编译程序进行编译，或者在不同的计算机上运行，效率均不同。

（2）算法的空间复杂度

算法的空间复杂度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存储空间、输入的初始数据所占的存储空间，以及算法执行中所需要的额外空间。

3．数据结构的定义

数据结构（Data Structure）是指相互之间存在一种或多种特定关系的数据元素的集合。

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

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。

在一般情况下，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（即连续），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。

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

数据结构是指反映数据元素之间的关系的数据元素集合的表示。通俗地说，数据结构是指带有结构的数据元素的集合。所谓结构实际上就是指数据元素之间的前后件关系。

一个数据结构应包含两方面信息：表示数据元素的信息和表示各数据元素之间的前后件关系的信息。

数据的逻辑结果是对数据元素之间的逻辑关系的描述。它可以用一个数据元素的集合和在此集合中定义的若干关系来表示。用 D 表示数据元素的集合，用 R 表示数据元素之间的前后件关系，即一个数据结构可以表示为 B=(D, R)，这是一个二元关系的表示方式。

（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（也称为数据的物理结构）。

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，还需要存放各数据元素之间的前后件关系的信息。

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，而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是很重要的。

4．数据结构的表示

数据结构的表示除了用二元关系表示外，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，一般称之为数据节点，简称为节点。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节点指向后件节点。

在数据结构中，没有前件的节点称为根节点；没有后件的节点称为终端节点（也称为叶子节点）。

一个数据结构中的节点可能是在动态变化的。根据需要或在处理过程中，可以在一个数据结构中增加一个新节点（称为插入运算），也可以删除数据结构中的某个节点（称为删除运算）。插入与删除是对数据结构的两种基本运算。除此之外，对数据结构的运算还有查找、分类、合并、分解、复制和修改等。

5．线性结构与非线性结构

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。

线性结构满足以下条件：

1）有且只有一个根节点。

2）每一个节点最多有一个前件，也最多只有一个后件。

如果一个数据结构不是线性结构，则称之为非线性结构。如果在一个数据结构中一个数据元素都没有，则称该数据结构为空。线性结构与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。

1.1.2 线性表

本节主要考查线性表的基本概念，以及线性表的顺序存储方式。

1．线性表概述

线性表是一种常用的数据结构。

在实际应用中，线性表都是以栈、队列、字符串、数组等特殊线性表的形式来使用的。由于这些特殊线性表都具有各自的特性，因此，掌握这些特殊线性表的特性，对于数据运算的可靠性和提高操作效率都是至关重要的。

线性表是一个线性结构，它是一个含有n（$n\geqslant 0$）个节点的有限序列，对于其中的节点，有且仅有一个开始节点没有前驱（前件）节点但有一个后继（后件）节点，有且仅有一个终端节点没有后继节点但有一个前驱节点，其他的节点都有且仅有一个前驱节点和一个后继节

点。一般来说，一个线性表可以表示成一个线性序列：$k_1,k_2,\ldots,k_n$，其中 k_1 是开始节点，k_n 是终端节点。

线性结构的基本特征如下：

1）集合中必存在唯一的一个“第一元素”。

2）集合中必存在唯一的一个“最后元素”。

3）除最后一个元素之外，每个元素均有唯一的后继。

4）除第一个元素之外，每个元素均有唯一的前驱。

由 n（$n\geqslant 0$）个数据元素（节点）$a_1,a_2,\ldots,a_n$ 组成的有限序列，数据元素的个数 n 定义为表的长度。当 $n=0$ 时称为空表。常常将非空的线性表（$n>0$）记作：$(a_1,a_2,\ldots,a_n)$。

2．线性表的顺序存储

线性表的顺序存储指的是用一组地址连续的存储单元依次存储线性表的数据元素。线性表的顺序存储又叫作顺序表。

（1）线性表的顺序存储基本概念

线性表的顺序存储结构具备以下两个基本特征：

1）线性表中的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。

假设线性表的每个元素需要占用 k 个存储单元，并且所占的存储位置 ADR(a_i+1)和第 i 个数据元素的存储位置 ADR(a_i)之间满足下列关系：

$$\mathrm{ADR}(a_i+1)=\mathrm{ADR}(a_i)+k$$

线性表第 i 个元素 a_i 的存储位置为：

$$\mathrm{ADR}(a_i)=\mathrm{ADR}(a_1)+(i-1)\times k$$

公式中 ADR(a_1)是线性表的第一个数据元素的存储位置，通常称为线性表的起始位置或基址。

在 C 语言中，通常定义一个一维数组来表示线性表的顺序存储空间，如图 1-1 所示。

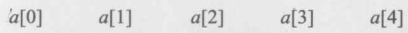

图 1-1　顺序存储

在用一维数组存放线性表时，该一维数组的长度通常要定义得比线性表的实际长度大一些，以便对线性表进行各种运算，特别是插入运算。

在线性表的顺序存储结构下，可以对线性表做以下运算：

1）在线性表的指定位置处加入一个新的元素（即线性表的插入）。

2）在线性表中删除指定的元素（即线性表的删除）。

3）在线性表中查找某个（或某些）特定的元素（即线性表的查找）。

4）对线性表中的元素进行排序（即线性表的排序）。

5）将一个线性表分解成多个线性表（即线性表的分解）。

6）将多个线性表合并成一个线性表（即线性表的合并）。

7）复制一个线性表（即线性表的复制）。

8）逆转一个线性表（即线性表的逆转）。

（2）顺序表的基本操作

顺序表的基本操作包括插入运算和删除运算。

1）顺序表的插入运算：线性表的插入运算是指在表的第 i（$1 \leqslant i \leqslant n+1$）个位置上，插入一个新节点 x，使长度为 n 的线性表（$a_1,\ldots,a_{i-1},a_i,\ldots,a_n$）变成长度为 $n+1$ 的线性表（$a_1,\ldots,a_{i-1},x,a_i,\ldots,a_{n+1}$）。

现在分析算法的复杂度。设它的值为 n。该算法的时间主要花费在循环节点后移语句上，该语句的执行次数（即移动节点的次数）是 $n-i+1$。由此可看出，所需移动节点的次数不仅依赖于表的长度，而且还与插入位置有关。

当 $i=n+1$ 时，由于循环变量的终值大于初值，节点后移语句将不执行。这是最好的情况，其时间复杂度为 $O(1)$。

当 $i=1$ 时，节点后移语句将循环执行 n 次，需移动表中所有节点。这是最坏的情况，其时间复杂度为 $O(n)$。

综合以上的情况，得出算法的平均时间复杂度为 $O(n)$。

2）顺序表的删除运算：线性表的删除运算是指将表的第 i（$1 \leqslant i \leqslant n$）个节点删除，使长度为 n 的线性表（$a_1,\ldots,a_{i-1},a_i,a_{i+1},\ldots,a_n$）变成长度为 $n-1$ 的线性表（$a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_{n-1}$）。

该算法的时间分析与插入算法相似，节点的移动次数也是由表长 n 和位置 i 决定的。

若 $i=n$，则由于循环变量的初值大于终值，前移语句将不执行，无需移动节点。

若 $i=1$，则前移语句将循环执行 $n-1$ 次，需移动表中除开始节点外的所有节点。这两种情况下算法的时间复杂度分别为 $O(1)$和 $O(n)$。

综合以上的情况得出，在顺序表上做删除运算，平均要移动表中约一半的节点，平均时间复杂度也是 $O(n)$。

1.1.3 栈和队列

栈和队列都是特殊的线性表，其定义符合线性表的定义，其操作也类似于线性表的操作，只不过增加了一些限定而已。

1. 栈的定义与操作

栈（Stack）是一种特殊的线性表。栈是只能在表的一端进行插入和删除运算的线性表。通常称插入、删除的这一端为栈顶（Top），另一端为栈底（Bottom），如图 1-2 所示。当表中没有元素时称为空栈。栈顶元素总是后插入的元素，也是最先被删除的元素；栈底元素总是最先被插入的元素，也是最后才能被删除的元素。

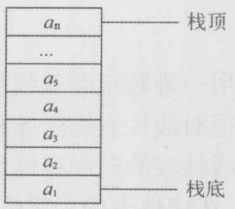

图 1-2 栈

假设栈 $S=(a_1,a_2,a_3,\ldots,a_n)$，则 a_1 为栈底元素，a_n 为栈顶元素。栈中元素按 $a_1,a_2,a_3,\ldots,a_n$ 的次序进栈，退栈的第一个元素应为栈顶元素，换句话说，栈的修改是按后进先出的原则进行的。因此，栈称为先进后出表（First In Last Out，FILO），或后进先出表（Last In First Out，LIFO）。

栈的操作主要有入栈运算、退栈运算（出栈）和读栈顶元素。

1）入栈运算：入栈运算是指在栈顶位置插入一个新元素。首先将栈顶指针加 1（即 Top 加 1），然后将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：退栈是指取出栈顶元素并赋给一个指定的变量。首先将栈顶元素（栈顶指针指向的元素）赋给一个指定的变量，然后将栈顶指针减 1（即 Top 减 1）。当栈顶指针为 0 时，说明栈空，不可进行退栈操作，这种情况称为栈的“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。这个运算不删除栈顶元素，只是将它赋给一个变量，因此栈顶指针不会改变。当栈顶指针为 0 时，说明栈空，读不到栈顶元素。

2. 队列的定义与操作

队列（Queue）是只允许在一端删除、在另一端插入的顺序表。允许删除的一端叫作队头（Front），允许插入的一端叫作队尾（Rear），如图 1-3 所示。

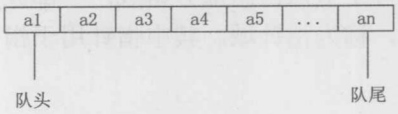

图 1-3　队列

（1）队列的运算

当队列中没有元素时称为空队列。在空队列中依次加入元素 $a_1,a_2,\ldots,a_n$ 之后，a_1 是队头元素，a_n 是队尾元素。显然退出队列的次序也只能是 $a_1,a_2,\ldots,a_n$，也就是说队列的修改是按先进先出的原则进行的。因此队列亦称为先进先出的线性表，或后进后出的线性表。

1）入队操作：往队列队尾插入一个元素称为入队运算。

2）出队操作：从队列队头删除一个元素称为出队运算。

（2）循环队列的运算

所谓循环队列，就是将队列存储空间的最后一个位置绕到第一个位置，形成逻辑上的环状空间。

在循环队列中，用队尾指针 Rear 指向队列中的队尾元素，用队头指针 Front 指向排头元素的前一个位置。因此，从排头指针 Front 指向的后一个位置直到队尾指针 Rear 指向的位置之间的所有元素均为队列中的元素。

在循环队列中进行出队、入队操作时，头尾指针仍要加 1，朝前移动。只不过当头尾指针指向向量上界（Queuesize-1）时，其加 1 操作的结果是指向向量的下界 0。

由于入队时尾指针向前追赶头指针，出队时头指针向前追赶尾指针，故队空和队满时头尾指针均相等。因此，我们无法通过 Front=Rear 来判断队列是空还是满。在实际使用循环队列时，为了能区分队列满还是队列空，通常还需增加一个标志值，其定义如下：当 s=0 时表示队列空，当 s=1 时表示队列非空。

入队运算：入队运算是指在循环队列的队尾加入一个新元素。首先将队尾指针进 1（即 Rear=Rear+1），且当 Rear=m+1 时置 Rear=1；然后将新元素插入到队尾指针指向的位置。当循环队列非空（s=1）且队尾指针等于队头指针时，说明循环队列已满，不能进行入队运算，这种情况称为“上溢”。

出队运算：出队运算是指在循环队列的队头位置退出一个元素并赋给指定的变量。首先将队头指针进 1（即 From=Front+1），且当 Front=m+1 时置 Front=1，然后将排头指针指向的元素赋给指定的变量。当循环队列为空（s=0）时，不能进行出队运算，这种情况称为“下溢”。

1.1.4 线性链表

本节主要考查线性链表的存储方式和基本操作。

1．线性表的链式存储

在定义的链表中，若只含有一个指针域来存放下一个元素地址，称这样的链表为单链表或线性链表，如图 1-4 所示。

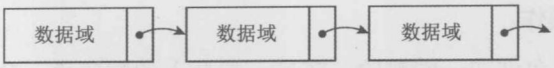

图 1-4　线性表链式存储

在链式存储方式中，要求每个节点由两部分组成：一部分用于存放数据元素值，称为数据域；另一部分用于存放指针，称为指针域。其中指针用于指向该节点的前一个或后一个节点（即前件节点或后件节点）。

（1）线性链表的存储结构

用一组任意的存储单元来依次存放线性表的节点，这组存储单元既可以是连续的，也可以是不连续的，甚至可以是零散分布在内存中的任意位置上的。因此，链表中节点的逻辑次序和物理次序不一定相同。为了能正确表示节点间的逻辑关系，在存储每个节点值的同时，还必须存储指示其后件节点的地址（或位置）信息，这个信息称为指针（Pointer）或链（Link）。这两部分组成了链表中的节点结构，链表正是通过每个节点的链域将线性表的 n 个节点按其逻辑次序链接在一起。由于上述链表的每一个节点只有一个链域，故将这种链表称为单链表（Single Linked）。

显然，单链表中每个节点的存储地址存放在其前驱节点 Next 域中，而开始节点无前驱节点，故应设头指针 HEAD 指向开始节点。同时，由于终端节点无后继节点，故终端节点的指针域为空，即 NULL。

（2）线性链表的基本运算

线性链表的基本运算包括在线性链表中查找指定元素、在线性链表中插入元素、在线性链表中删除元素。

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的运算中，总是首先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，在线性链表中寻找包含指定元素的前一个节点。

在链表中，查找是否有节点值等于给定值 x 的节点，若有的话，则返回首次找到的其值为 x 的节点的存储位置；否则返回 NULL。查找过程从开始节点出发，顺着链表逐个将节点的值与给定值 x 做比较。

2）在线性链表中插入元素：线性链表的插入是指在链式存储结构下的线性链表中插入一个新元素。

插入运算是将值为 x 的新节点插入到表的第 i-1 个节点和第 i 个节点之间。因此，必须首先找到 a_{i-1} 的存储位置 p，然后生成一个数据域为 x 的新节点*p，并令节点 p 的指针域指向新节点，新节点的指针域指向节点 a_i。

由线性链表的插入过程可以看出，由于插入的新节点取自于可利用栈，因此，只要可利