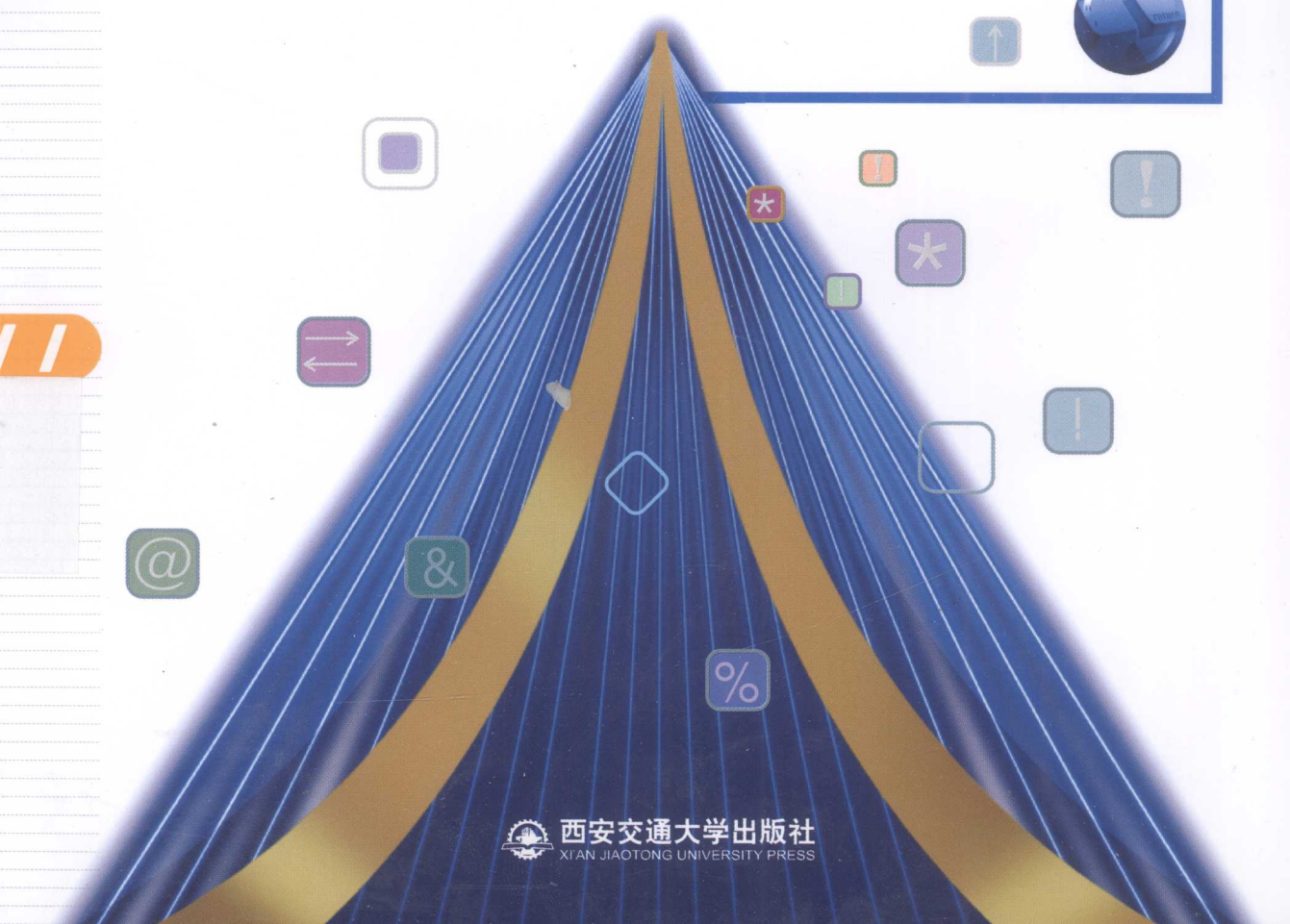
计算机专业核心课程辅导及考研攻略

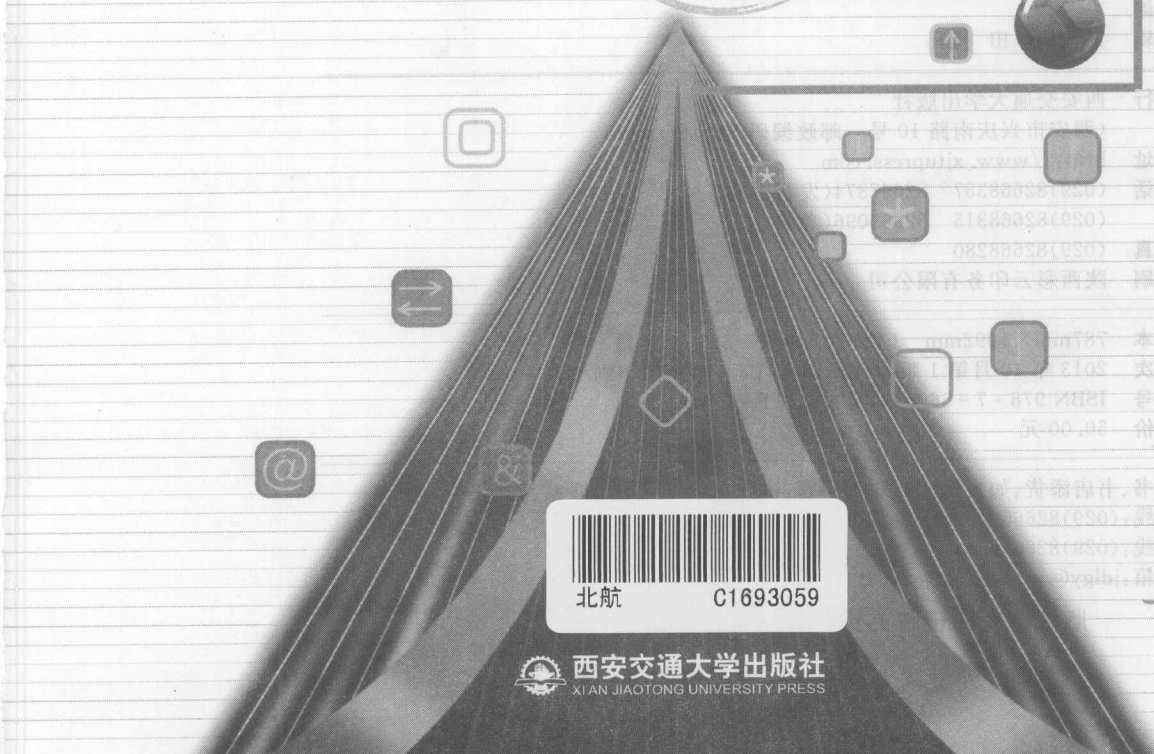
主　编　王曙燕
副主编　王小银　王晓婕　谢晓燕　王春梅



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

计算机专业核心课程辅导及考研攻略

主　编　王曙燕
副主编　王小银　王晓婕　谢晓燕　王春梅



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

内容简介

本书紧扣《计算机学科专业基础综合考试大纲》的基本要求编写。全书共分为4篇，第1篇数据结构，第2篇计算机组成原理，第3篇操作系统，第4篇计算机网络，每篇为一门课程的学习指导，包含重点难点分析、典型试题分析，习题精选并对所有精选的习题给出了参考答案，供读者参阅。

本书内容丰富，概念清晰，实用性及指导性强，列举的例题繁简得当，具有一定的代表性。本书可作为高等学校学生学习数据结构、组成原理、操作系统、计算机网络课程的配套用书，也可以作为计算机专业研究生入学考试的辅导用书。

图书在版编目(CIP)数据

计算机专业核心课程辅导及考研攻略/王曙燕主编.
—西安:西安交通大学出版社,2013.10
ISBN 978-7-5605-5349-8

Ⅰ.①计… Ⅱ.①王… Ⅲ.①电子计算机-研究生-入学考试-自学参考资料 Ⅳ.①TP3

中国版本图书馆CIP数据核字(2013)第139899号

书　　名　计算机专业核心课程辅导及考研攻略
主　　编　王曙燕
责任编辑　刘雅洁　田　华

出版发行　西安交通大学出版社
　　(西安市兴庆南路10号　邮政编码710049)
网　　址　http://www.xjtupress.com
电　　话　(029)82668357　82667874(发行中心)
　　(029)82668315　82669096(总编办)
传　　真　(029)82668280
印　　刷　陕西彩云印务有限公司

开　　本　787mm×1092mm　1/16　印张　32　字数　785千字
版次印次　2013年10月第1版　2013年10月第1次印刷
书　　号　ISBN 978-7-5605-5349-8/TP·582
定　　价　50.00元

读者购书、书店添货、如发现印装质量问题，请与本社发行中心联系、调换。
订购热线:(029)82665248　(029)82665249
投稿热线:(029)82664954
读者信箱:jdlgy@yahoo.cn

前言

2014年是全国硕士研究生统一入学考试计算机科学与技术学科的初试专业考试实行全国统一命题的第六年。《2009—2013年计算机学科专业基础综合考试大纲》反映了计算机专业考试的新变化。计算机学科专业基础综合考试涵盖数据结构、计算机组成原理、操作系统和计算机网络等学科专业基础课程。要求考生比较系统地掌握上述专业基础课程的概念、基本原理和方法，能够运用所学的基本原理和基本方法分析、判断和解决有关理论问题和实际问题。

从题型来看，统考题型有两种：即单项选择题和综合应用题。选择题40小题每小题2分，分值为80；应用题共7题，分值为70分。在题型方面删除了以往考研中经常出现的判断题，填空题等，加大了选择题的分值。从考试范围来看，《考试大纲》中确定的考试范围为四门课程。其中，传统的计算机考察热点科目数据结构和计算机组成原理都占了45分，仍然占据着优势，操作系统占了35分，计算机网络在以前很多学校的考研科目都没考察过，在统考大纲里面占了25分。由于一门考试涉及多门课程内容，而且考研大纲中每门课程的大纲几乎涵盖了课程的全部内容，考生很难把握重点。

本书紧扣考试大纲，结合2009—2013年全国统考真题，针对全国计算机学科专业考研大纲中的4大部分内容各有侧重地将数据结构、操作系统、计算机组成原理、计算机网络各课程中的知识点进行归纳、梳理，疑点诠释、难点辅导、综合复习；通过对大量例题的求解分析，力求帮助考生从容应试、提高考生分析问题与解决问题的能力。使读者把握重点难点，提高复习的效率。

同时，根据今年计算机专业研究生入学考试逐渐实行非统考的形式变化，书中对计算机核心课程大量非统考知识点也进行了讲解，使本书除覆盖统考大纲的所有内容外，还包括了各大自主命题高校所要求的知识点及相关课程学习的知识点。本书既可作为参加计算机专业研究生的入学考试辅导用书，也可作为高校学生或计算机爱好者学习计算机专业核心课程的辅导用书。

本书共分4篇，第1篇数据结构，第2篇计算机组成原理，第3篇操作系统，第4篇计算机网络，附录为2013年统考真题解析。各篇篇幅按照大纲中课程分值比例安排。前4篇中，每篇为一门课程的学习指导，每篇的第1章为“重点难点分析”，首先对该课程的大纲进行解析，然后对考试大纲规定的内容进行有重点地串讲，把相关知识点串起来，突出常考知识与核心知识，对考点、重点、难点内容进行解释与讲述，让考生掌握问题的本质，指出复习的方法。第2章是典型试题分析，精选常考题型，分析解题思路，对重点难点进行剖析，让考生掌握解题方法与技巧，增强考生的解题能力。第3章是习题精选，给出模拟试题或名校试题作为同步练习题，可以帮助考生系统地理解和掌握考试大纲中的各个考点，通过实战练习提高考生的应试能力。第4章是习题参考答案，供读者参阅。

本书是作者根据多年从事相关课程的教学经验和考研辅导的经验编写的，王曙燕任主编，

王小银、王晓婕、谢晓燕和王春梅任副主编。第1篇数据结构由王曙燕、孟彩霞、王春梅编写，其中王曙燕编写了第2章中2.1至2.3，孟彩霞编写了第1章、第2章中2.5和2.6，王春梅编写了第2章中的2.4、第3章和第4章。第2篇计算机组成原理由王晓婕和董梁编写，其中第5章由王晓婕编写，第6章由王晓婕和董梁共同编写，第7章和第8章由董梁编写。第3篇操作系统由王小银、舒新峰和梁琛编写，其中第9章由舒新峰编写，第10章中10.1—10.3由梁琛编写，10.4—10.7由王小银编写，第11章中习题11-1至11-45以及11-136至11-149由梁琛编写、习题11-46至11-135以及11-150至11-168由王小银编写，第12章由王小银和梁琛共同编写；第4篇计算机网络由谢晓燕编写，附录真题由王春梅、王晓婕、王小银和谢晓燕解析。全书由王曙燕和王小银统稿。作者在此一并向他们表示衷心的感谢。

由于编者水平有限，书中难免存在缺点和错误，恳请专家和读者批评指正。

作者联系方式：wsylxj@126.com。

编　者

2013.4

目　录

第1篇　数据结构

第2篇　计算机组成原理

第 3 篇　操作系统

第 4 篇　计算机网络

第1篇　数据结构

第1章　重点难点分析

1.1　大纲解析

1.1.1　统考大纲

【考查目标】

1. 掌握数据结构的基本概念、基本原理和基本方法。

2. 掌握数据的逻辑结构、存储结构及其基本操作的实现，能够对算法进行基本的时间复杂度与空间复杂度的分析。

3. 能够运用数据结构的基本原理和方法进行问题的分析和求解，具备采用C、C++或Java语言设计与实现算法的能力。

【考查内容】

一、线性表

(一)线性表的定义和基本操作

(二)线性表的实现

1. 顺序存储

2. 链式存储

二、栈、队列和数组

(一)栈和队列的基本概念

(二)栈和队列的顺序存储结构

(三)栈和队列的链式存储结构

(四)栈和队列的应用

(五)特殊矩阵的压缩存储

三、树与二叉树

(一)树的基本概念

(二)二叉树

1. 二叉树的定义及其主要特征

2. 二叉树的顺序存储结构和链式存储结构

(七)堆排序
(八)二路归并排序
(九)基数排序
(十)各种内部排序算法的比较
(十一)内部排序算法的应用

1.1.2　大纲解析

在统考大纲的考查目标中,数据结构的基本概念、数据的逻辑结构与存储结构及它们之间的区别在一般教材中的第 1 章绪论中都有较为详细的介绍;每种数据结构的逻辑定义、存储表示以及各种基本操作的实现算法体现在后面各章每种结构的讲解中;能够运用数据结构的基本原理和方法进行问题的分析和求解其实也是数据结构课程的学习目标。

笔者在多年讲授数据结构课程的过程中深切感到,学生理解课程的概念和书本知识并不困难,一旦涉及解决具体问题,特别是编制算法(程序),往往无从下手。但是从统考大纲和考题类型来看,除了编制程序外,10 道或 11 道选择题是考察考生对基本概念和基本理论的理解和掌握,涉及的内容不会超过大纲中提及的知识点。两道大题中肯定会有一题编写算法,描述算法的语言并不限制,可以使用 C、C++或 Java 语言;另外一题可以是算法,与算法思想有关的证明,或者综合应用题等。

统观 2009 到 2012 四年的统考考题发现,考试内容以树、图、哈希表、链表为重点,其他部分的分量相差不多。从难度上看,应属于较为容易。因为所有题目的解答都比较直接,几乎没有需要综合运用多种数据结构知识进行抽象提升才能解答的问题。也就是说,这些题目都属于局部性较强的题型。即在题目覆盖的知识点上,解答也几乎是直接将课堂或书本上的知识进行应用即可,没有什么太费脑筋的转折和变化。数据结构部分在整份考题中所占比例如下:2009 年考题中,选择题 10 题,计 20 分,综合应用题 2 题(图 1 题 10 分,链表 1 题 15 分),计 25 分,总计 45 分;2010 年,选择题 11 题,计 22 分,综合应用题 2 题(散列表 1 题 10 分,顺序表 1 题 13 分),计 23 分,总计 45 分;2011 年,选择题 11 题,计 22 分,综合应用题 2 题(图 1 题 8 分,线性表 1 题 15 分),计 23 分,总计 45 分;2012 年,选择题 11 题,计 22 分,综合应用题 2 题(排序 1 题 10 分,链表 1 题 13 分),计 23 分,总计 45 分。

1. 数据结构概述

概述部分不是考试的重点,但是后面每个知识点中的数据结构都是以这里介绍的数据结构三要素(数据的逻辑结构、存储结构和各种运算)为主线来理解的。一般教材中对于如何进行算法的时间和空间复杂度分析都是在概述中讲解,后面各章中的每种数据结构涉及的算法统一依照这里介绍的方法进行分析。2011 年和 2012 年考题中的选择题各有 1 题是进行算法时间复杂度计算。

2. 线性表

线性表是一种最简单最常用的数据结构,这部分内容的考试形式一般是以大题的形式出现,比如 2009 年到 2011 年的考题中都有一道 15 分的算法设计题,2012 年的考题中有一道 13 分的算法设计题。所以考生要灵活应用线性表适当的存储表示法解决具体问题,要清楚两种不同存储结构的线性表:顺序表和链表,尤其是线性表的应用。

3. 栈、队列和数组

栈、队列都是操作受限的特殊线性表，数组也可以看作是线性表的推广。这部分内容的考核多数是以选择题的形式出现，如2009年到2011年的考题中每年都有2道选择题，2012年是1道选择题。单独以该部分知识点作为大题的可能性比较小，但是在树、图等的应用(大题)中使用到栈、队列和数组的可能性比较大。

4. 树与二叉树

树型结构是一类重要的非线性结构，其中最常用的是树与二叉树。这部分内容的考核既可能以选择题的形式出现，也可能作为大题出现。如2009年和2010年的考题中是4道选择题，2011年的考题中是3道选择题，2012年的考题中是2道选择题。这部分重点要掌握几个重要的知识点：二叉树的性质、二叉树的存储、二叉树的遍历、二叉排序树、平衡二叉树、树(森林)与二叉树的转换、树的应用(如：哈夫曼树和哈夫曼编码)等。大题可能是算法题，也可能是有关的证明题，或者计算类的综合应用题。

5. 图

图是比树更一般、更复杂的非线性数据结构。同树一样，这部分内容的考核既可能以选择题的形式出现，也可能作为大题出现。如2009年的考题中是1道选择题和1道10分的大题，2010年的考题中是2道选择题，2011年的考题中是1道选择题和1道8分的大题，2012年的考题中是4道选择题。这部分重点要掌握几个重要的知识点：图的邻接矩阵和邻接表表示法、图的深度优先和广度优先搜索、图的几个典型应用。大题可能是算法题，也可能是有关的证明题，或者计算类的综合应用题。

6. 查找

查找是所有操作中最重要的操作之一，使用频率很高。这部分内容的考核可以选择题的形式出现，也可以大题的形式出现。如2009年的考题中是1道选择题，2010年的考题中是1道选择题和1道10分的大题，2011年的考题中是2道选择题，2012年的考题中是1道选择题。这部分重点要掌握几种不同的查找方法以及对不同的查找算法的评价，评价查找算法是用平均查找长度来衡量。

7. 内部排序

同查找一样，排序也是重要的操作之一，使用频率也很高。这部分内容的考核可以选择题的形式出现，也可以大题的形式出现。如2009年到2011年的考题中都是2道选择题，而2012年的考题中是2道选择题和1道10分的大题。这部分重点是要掌握几种典型的排序方法思想，并可以按照排序思想手动进行排序，以及各种不同排序方法在时间、空间和稳定性方面的区别。

1.2 知识点精讲

1.2.1 绪论

数据结构是相互之间存在一种或多种特定关系的数据元素的集合。根据数据元素之间关

系的不同特性，通常有四种基本结构：(1)集合结构；(2)线性结构；(3)树形结构；(4)图状结构。

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。这里的关系描述的是数据元素之间的逻辑关系，因此又称为数据的逻辑结构。

讨论数据结构的目的是为了在计算机中实现对它的操作，因此还需研究如何在计算机中表示它。数据结构在计算机中的表示(又称映像)称为数据的物理结构或者存储结构。它包括数据元素的表示和关系的表示。数据元素之间的关系在计算机中有两种表示方法：顺序映像和非顺序映像，并由此得到两种不同的存储结构：顺序存储和链式存储。顺序存储的特点是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系；链式存储的特点是借助指针表示数据元素之间的逻辑关系。

任何一个算法的设计(决定有什么样的操作或运算)取决于选定的数据(逻辑)结构，而算法的实现依赖于采用的存储结构。

数据结构不仅讨论数据在计算机中的组织和表示方法，同时也重在训练高效地解决复杂问题的能力。为了实现某类数据结构上的运算(操作)，就需要设计算法。

算法是指令的有限序列，是为解决特定问题而规定的一系列操作。一个算法具有 5 个重要特性：有穷性、确定性、可行性、零个或多个输入、一个或多个输出。

设计一个“好”的算法应考虑达到以下目标：(1)正确性，算法应当满足具体问题的需求；(2)可读性；(3)健壮性(鲁棒性)；(4)高效率和低存储量，效率指的是算法的运行时间，存储量是指算法在运行过程中所需要的最大存储空间，这两者都与问题的规模有关。

对算法效率的度量是采用事前分析估算的方法。一个特定算法的运行时间只依赖于问题的规模(通常用整数 n 表示)，或者说，它是问题规模的函数。具体地说，从算法中选取一种对于所研究问题(或算法类型)来说是基本操作的原操作，以该基本操作重复执行的次数作为算法的时间度量。被称做基本操作的原操作应是其重复执行次数和算法的执行时间成正比的原操作，多数情况下它是最深层循环内的语句中的原操作。

一般情况下，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 $f(n)$，算法的时间度量记作 $T(n)=O(f(n))$，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，称做算法的时间复杂度。

算法需要被描述，统考大纲中已经明确描述算法可以采用 C、C++或者 Java 语言，考生可以依据自己的熟悉程度选择语言。

在绪论部分考生必须清楚数据结构包括数据的逻辑结构、存储结构和运算集合三部分。逻辑结构定义了数据元素之间的逻辑关系；存储结构是逻辑结构在计算机中的表示。一种逻辑结构可以采用不同存储方式存放在计算机中，存储结构有顺序存储和链式存储。切记算法的实现是与所选用的存储结构有关，同一个问题存储表示不同，编写的算法(程序)也不同。

1.2.2　线性表

1. 线性表的定义

线性表是由 $n(n\geqslant 0)$ 个类型相同的数据元素组成的有限序列，记作 $(a_1,a_2,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n)$。数据元素 $a_i(1\leqslant i\leqslant n)$ 的具体含义在不同的情况下可以不同，它既可以是原子类型，也可以是结构类型。此外，线性表中相邻的数据元素之间存在着序偶关系。元素个数 n

为线性表的长度，$n=0$ 时称为空表。

线性表是一个相当灵活的数据结构，它的长度可根据需要增长或缩短。因此对线性表的数据元素不仅可以进行访问操作，还可进行插入和删除操作。在实际应用中对线性表的运算有很多，例如有时需要将多个线性表合并成一个线性表，以及在此问题基础之上进行的有条件合并等，还有如分拆、复制、排序等。各种运算（操作）的具体实现是与线性表具体采用哪种存储结构有关。

线性表是最基本的数据结构，相对比较简单，考生一定要深入理解线性表中数据元素之间的关系。

2. 线性表的顺序存储结构

线性表的顺序存储是指用一组地址连续的存储单元依次存储线性表中的各个数据元素，使得线性表中在逻辑结构上相邻的数据元素存储在连续的物理存储单元中，即通过数据元素物理存储的连续性来反映数据元素之间逻辑上的相邻关系。采用顺序存储结构存放的线性表通常简称为顺序表。

在顺序表中，只要确定了存储线性表的起始位置，线性表中的任一元素都可随机存取，所以线性表的顺序存储结构是一种随机存取的存储结构。假设线性表有 n 个元素，每个元素占 k 个存储单元，第一个元素的存储地址为 $\mathrm{loc}(a_1)$，则可以通过公式计算出第 i 个元素的地址 $\mathrm{loc}(a_i)=\mathrm{loc}(a_1)+(i-1)*k$。由于高级程序设计语言中的数组具有随机存取的特性，所以一般采用一维数组（向量）表示顺序表。

在这种存储结构中，容易实现线性表的某些操作，如随机存取第 i 个数据元素等。但是进行插入和删除操作时需要移动大量的数据元素，效率比较低。

这里考生需要深入理解线性表的顺序存储是一种随机存取结构，体会顺序表上基本操作实现的特征，不要与链式存储的操作实现混淆。

3. 线性表的链式存储结构

线性表的链式存储是指用一组任意的存储单元来存放线性表的数据元素，这组存储单元可以是连续的，也可以是不连续的，甚至是零散分布在内存的任何位置上。为了表示每个数据元素与其直接后继数据元素之间的逻辑关系，除了存储数据元素本身的信息外，还需存储一个指示其直接后继的信息（即直接后继的存储位置）。这两部分信息组成一个结点，即一个结点包括两个域：数据域（存储数据元素信息的域）和指针域（存储直接后继存储位置的域）。指针域中存储的信息称做指针或链。n 个结点链结成一个链表，即为线性表的链式存储结构。

如果链表中的每个结点只包含一个指针域，则称为线性链表或单链表。由于线性表中第一个结点无前驱，所以应设一个头指针指向第一个结点；线性表中最后一个结点无后继，因此单链表中最后一个结点的指针域为“空”（NULL）。

单链表的头指针标志着整个单链表的开始，给定头指针，即可顺着每个结点的指针域得到单链表中的每个元素。为了操作方便，常常在单链表的第一个结点之前附设一个头结点，头结点的数据域可以存储一些关于线性表长度的附加信息，也可以什么都不存，而指针域存储指向第一个结点的指针（即第一个结点的存储位置）。此时头指针就不再指向表中的第一个结点而是指向头结点。如果线性表为空表，则头结点的指针域为“空”。

由于单链表的操作必须从头指针开始，顺着链查找任意一个结点，所以单链表是一种顺序存取的存储结构，而不是随机存取的存储结构。

必须清楚各种运算的实现是与存储结构密切相关，因此在单链表上实现各种操作和顺序表中实行相同的操作所编写的算法是不同的。显然在单链表上的查找操作比顺序表上的查找效率低。但是单链表上的插入、删除操作由于不需要移动大量的数据元素，效率要高。

建立单链表的过程就是在一个空的单链表上不断插入结点的过程，有头插法建表和尾插法建表。插入或删除操作时要先查找到插入或删除的位置，然后实施插入或删除操作。

这里考生需要深入理解线性表的链式存储是一种顺序存取结构，切记不要根据元素位置直接访问数据元素。理解链式存储下的操作和顺序表操作的不同，千万不要混淆。需要熟练掌握各种基本操作算法的实现。

4. 线性表的应用

前边已经说过，线性表是一个相当灵活的数据结构，它的应用十分广泛。在实际应用中对线性表的运算有很多，各种运算的实现都与具体采用哪种存储结构有关。

考生主要是要能够灵活应用线性表适当的存储表示解决具体问题，编写相关算法。在编写算法时，如果使用链表，切记操作中千万不能使链意外“断开”。另外要知道链表中存储空间是动态分配的，需要时申请空间，使用完毕要记得释放存储空间。

1.2.3　栈、队列和数组

1. 栈和队列的基本概念

从数据结构角度讲，栈和队列属于线性结构，是一类操作受限的特殊线性表。其特殊性在于限制插入和删除等运算的位置。

栈是只准在一端进行插入和删除操作的线性表，该端称为栈顶(top)，另一端称为栈底(bottom)。栈顶的当前位置由栈顶指针指示。当栈中没有元素时称为空栈。栈的插入操作也称为进栈或入栈，删除操作也称为出栈或退栈。栈具有后进先出的特性，因此又称为后进先出(LIFO)表。假设栈 $S=(a_1,a_2,\cdots,a_n)$，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_n 的次序进栈，退栈的第一个元素应为栈顶元素 a_n。

和栈相反，队列是一种先进先出(FIFO)的线性表。它只允许在表的一端插入元素，而在另一端删除元素。允许插入的一端叫做队尾(rear)，允许删除的一端叫做队头(front)。假设队列为 $Q=(a_1,a_2,\cdots,a_n)$，则 a_1 就是队头元素，a_n 是队尾元素，队列中的元素是按照 a_1，a_2，…，a_n 的顺序进入的，退出队列时也必须按照同样的次序依次出队。

2. 栈的存储结构

栈有两种存储结构：顺序存储和链式存储。顺序存储的栈称为顺序栈，链式存储的栈称为链栈。

顺序栈是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，同时附设指针 top(栈顶指针)指示栈顶元素在顺序栈中的位置。顺序栈在高级程序设计语言中是用一维数组表示。通常习惯以 top=0 表示空栈，由于 C 语言中数组的下标从 0 开始，所以也可以 top=−1 表示空栈。在顺序栈中实现栈的操作时，要注意栈顶指针 top 的变化。

链栈是采用链表作为存储结构实现的栈。top 为栈顶指针，始终指向当前栈顶元素所在的结点，若 top=NULL 代表栈空。通常链栈不设头结点。链栈中各种基本操作的实现与单链表的操作类似，对于链栈，在使用完毕时，应释放其空间。

栈的操作较为简单，是线性表的子集。一般是在其他应用(算法)中使用到栈，使用时对于

栈的存储结构不作要求，既可以是顺序栈也可以是链栈。

3. 队列的存储结构

队列也有两种存储结构：顺序存储和链式存储。顺序存储的队列称为顺序队列，链式存储的队列称为链队列。

链队列中为了操作方便，可以采用带头结点的链表表示队列，并设置一个队头指针（头指针）和一个队尾指针（尾指针）。头指针始终指向头结点，尾指针指向当前最后一个元素结点。空的链队列的头指针和尾指针均指向头结点。通常将头指针和尾指针封装在一个结构体中，并将该结构体类型重新命名为链队列类型。链队列中各种操作的实现也和单链表类似，它的插入和删除操作是单链表的特殊情况。需要注意的是在链队列的删除操作中，对于仅有一个元素结点的特殊情况，删除后还需要修改尾指针。

循环队列是队列的一种顺序表示和实现方法，是用一组地址连续的存储单元依次存放从队头到队尾的元素，在高级程序设计语言中是用一维数组表示，如 Queue[MAXSIZE]，另外还需附设两个指针 front 和 rear 分别指示队头元素和队尾元素的位置。一般约定，初始化队列时，令 front = rear = 0；插入新的队尾元素（入队）时，直接将新元素送入尾指针 rear 所指的单元，然后尾指针增 1；删除队头元素（出队）时，直接取出队头指针 front 所指的元素，然后头指针增 1。这样队列中的头指针和尾指针都是动态变化的，在非空队列中，头指针始终指向队头元素位置，尾指针始终指向队尾元素的下一个位置。当 rear == MAXSIZE 时，认为队满，不能进行入队操作，而此时不一定真的队满，因为随着部分元素的出队，数组前面会出现一些空单元，但由于只能在队尾入队，使得这些空单元无法利用，这种现象称为假溢出。真正队满的条件是数组中的单元全部被占用，即：rear－front==MAXSIZE。

为了解决假溢出现象，一个较巧妙的办法是将顺序队列的数组看成一个环状的空间，即规定最后一个单元的后继为第一个单元，称之为循环队列。

循环队列中进行入队操作时，当 rear+1 == MAXSIZE 时，令 rear=0，即可求得最后一个单元 Queue[MAXSIZE－1]的后继 Queue[0]；出队操作时，头指针的变化类似。使用取模（求余）运算，可以自动实现尾指针、头指针的循环变化，即：rear =(rear+1)mod MAXSIZE；front =(front+1)mod MAXSIZE。

循环队列中，当数据元素相继入队时，尾指针不断追赶头指针，就有可能出现 front==rear，此时队列满；反之，当数据元素相继出队时，头指针不断追赶尾指针，也有可能出现 front==rear，此时队列空。可见，只凭 front==rear 无法判断队列的状态是“空”还是“满”。对于这个问题，可有两种处理方法。

一种方法是少用一个元素空间，当尾指针所指向的空单元的后继是队头元素所在的单元时，则停止入队，这样尾指针就永远追不上头指针，队满时不会有 front==rear。这时队列“满”的条件为：(rear+1)mod MAXSIZE == front；判断队列“空”的条件不变，仍为 front==rear。

另一种是增设一个标志位，以区别队列是“空”还是“满”。假设标志位 tag=0 表示队列空，tag=1 表示队列满。这样初始化循环队列时，令 front=rear=0 且 tag=0；入队时判断队列满的条件为：(front==rear) && (tag==1)；出队时判断队列空的条件为：(front==rear) && (tag==0)。

队列中链队列的操作类似于链表，较为简单，容易掌握。循环队列比较复杂，关键是要弄

清楚队满和队空的判断。入队时，先判断队列是否满，若不满才可以进行入队操作；出队时，先判断队列是否为空，若不空才能进行出队操作。在一些应用(算法)中也经常使用到队列，使用时对其存储结构不会作要求。队列单独出考题的话，考察循环队列的可能性大。

4. 栈和队列的应用

栈和队列是两种重要的线性结构，它们广泛应用在各种软件系统中。堆栈被广泛应用于编译软件和程序设计语言中，队列则被广泛应用于操作系统和事务管理中。

栈结构所具有的“后进先出”特性，使得栈成为程序设计语言中的有力工具。现实生活中也有很多“后进先出”的例子，典型的如铁路调度站等。下面是栈的一些典型应用。

(1)括号匹配问题：检验是否匹配时可设置一个栈，每读入一个括号，若是左括号，则入栈，等待相匹配的同类右括号；若是右括号，且与当前栈顶的左括号同类型，则二者匹配，将栈顶的左括号出栈，否则属于不合法的情况。另外，如果输入序列已读完，而栈中仍有等待匹配的左括号，或者读入了一个右括号，而栈中已无等待匹配的左括号，均属不合法的情况。当输入序列和栈同时变为空时，说明所有括号完全匹配。

(2)表达式求值：表达式求值是高级语言编译中的一个基本问题。任何一个表达式都是由运算数、运算符和界限符组成。实现表达式求值的算法思想：规定运算符的优先级表；设置两个栈，分别为 OVS(运算数栈)和 OPTR(运算符栈)；自左向右扫描，进行如下处理：遇到运算数进 OVS 栈，遇到运算符则与 OPTR 栈的栈顶运算符优先级比较后决定进栈还是退栈，如为退栈，且得到栈顶运算符 θ，那么 OVS 连续退栈两次，得到运算数 a、b，对 a、b 执行 θ 运算，得到的结果进 OPTR 栈。

(3)数制转换：十进制数和其他进制数的互相转换是计算机实现计算的基本问题，其解决方法很多，其中一种方法可以利用栈来完成。

(4)行编辑程序：一个简单的行编辑程序的功能是，接收用户从终端输入的程序或数据，并存入用户数据区。用户输入时不能保证不出差错，通常的做法是，设立一个输入缓冲区，接收用户输入的一行数据，然后逐行存入用户数据区。可设这个输入缓冲区为一个栈结构，当用户输错时，使用退格符则从栈中删除一个字符，使用退行符清空栈等。

(5)迷宫求解：计算机解迷宫时，通常是用“穷举求解”的方法，即从入口出发，顺某一方向向前探索，若能走通，则继续往前走；否则沿原路返回，换一个方向再继续探索，直至所有可能的通路都探索到为止。为了保证在任何位置上都能沿原路退回，显然要用具有后进先出的栈来保存从入口到当前位置的路径。只需大致了解一下算法的思想就可以了。

(6)栈与递归：栈还有一个非常重要的应用就是在程序设计语言中用来实现递归。当递归函数调用时，应按照“后调用先返回”的原则处理调用过程，因此函数之间的信息传递和控制转移必须通过栈来实现，递归工作栈是实现递归的核心技术。

队列最典型的例子就是操作系统中的作业排队，循环队列也经常用于操作系统的一些实用程序中。例如，当程序正在执行其他任务时，用户可以从键盘上不断键入所要输入的内容，很多字处理软件就是这样工作的。用户键入的字符存到缓冲区中，系统在处理完当前进程后，就从缓冲区中取出键入的字符，并按要求进行处理。这里的键盘输入缓冲区采用了循环队列。队列的特性保证了输入字符先输入、先保存、先处理的要求，循环结构又有效地限制了缓冲区的大小，并避免了假溢出问题。

栈和队列的应用很广泛，利用栈和队列可以控制解决问题的顺序。实际应用中，凡是对元