全国计算机等级考试教材系列

National Computer Rank Examination

二级

公共基础知识教程与考前辅导

毕 超 主 编

Computer
National
Rank
Examination

中国水利水电出版社
www.waterpub.com.cn

最新大纲

全国计算机等级考试教材系列

二级公共基础知识教程与考前辅导

毕　超　主编

中国水利水电出版社

内 容 提 要

本书是根据 2004 年教育部考试中心公布的《全国计算机等级考试考试大纲》对二级公共基础知识内容的要求编写而成的。本书的内容包括：数据结构与算法、程序设计基础、软件工程基础以及数据库设计基础。在每章的后面安排了历年的习题解析和练习题，最后通过四套模拟试题来巩固读者学习到的知识。

本书内容翔实，操作步骤清晰，图文并茂，具有极强的可操作性和针对性，完全符合全国计算机等级考试对二级考试公共基础部分的要求。考生通过本书的学习可完全掌握等级考试二级公共基础知识部分的内容，并通过做大量练习题达到巩固和提高的目的。

本书是中国水利水电出版社计算机等级考试教程一套书中二级公共基础知识部分的教程，适合参加二级考试的各科读者作为公共基础知识部分的教材和考前参考用书。

图书在版编目（CIP）数据

二级公共基础知识教程与考前辅导 / 毕超主编. —北京：中国水利水电出版社，2006

（全国计算机等级考试教材系列）

ISBN 7-5084-3850-7

Ⅰ. 二… Ⅱ. 毕… 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（2006）第 073594 号

书　　名	**二级公共基础知识教程与考前辅导**
作　　者	毕 超 主 编
出版 发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787mm×1092mm　16 开本　10 印张　242 千字
版　　次	2006 年 7 月第 1 版　2006 年 7 月第 1 次印刷
印　　数	0001—5000 册
定　　价	15.00 元

前　言

从2005年上半年开始，教育部对全国计算机等级考试进行了较大调整。二级考试的笔试包括基础知识和程序设计两部分，其中基础知识占30分。二级基础知识考核内容的大变化令相当多的考生措手不及，事实上这次改革将基础知识的内容由计算机常识（一级难度）调整为程序开发基础（三级难度），很多考生由于掌握考试信息不及时因此没有在2005年的考试中取得好成绩。

本书是按照教育部考试中心最新修订的《全国计算机等级考试考试大纲》中对二级考试公共基础知识部分的要求而编写的。

全书主要内容包括：数据结构与算法、程序设计基础、软件工程基础以及数据库设计基础等基本知识点、出题要点、出题趋势和应试指导。在每一章后结合本章知识点设计并分析了历年真题和实战习题。在第5章中设计了5套笔试模拟试题，并对答案进行了分析。

本书内容详实，操作步骤清晰，图文并茂，涉及面广泛，具有极强的可操作性和针对性，完全针对和适用于参加全国计算机等级考试二级公共基础知识的考生。通过本书的学习可轻松掌握计算机的基本知识，达到等级考试对二级公共基础知识考试的要求。

本书内容紧扣考试大纲，对参加全国计算机等级考试二级公共基础知识的应试者，是必备的考前应试辅导教材。

读者通过认真的学习，相信能够轻松掌握相应的计算机知识，并顺利通过教育部的计算机等级考试。

本书由毕超主编，参加编写工作的还有贾凤波、封素洁、陈艳华、郝思嘉、李强、黄卓、王敬栋、杜波、赵应丁、王进、童剑、方春明、王克杰、张晋宝、张勇、王克杰、马路、项天一等。

由于编者水平有限，加之时间仓促，谬误之处实属难免。敬请读者不吝指正，以期日后修订时改进。如果读者在阅读本书的过程中遇到问题，或有其他意见和建议，请发电子邮件至：xinyuanxuan@263.net。我们将竭诚为您提供帮助，并努力改进今后的工作，奉献给读者更高品质的图书。

编　者

2006年1月

目　　录

第 1 章　数据结构与算法

本章考点和学习目标

考点：

- 具有数据结构及算法的基本知识

学习目标：

- 了解算法的基本知识
- 了解数据结构的基本概念
- 了解线性表及存储结构
- 了解栈、队列及线性链表的概念及运算
- 了解树与二叉树
- 了解查找技术及排序技术

本章系统地介绍软件设计中常用的几种数据结构和相应的存储结构和算法以及常用的几种查找和排序算法。该章为后续课程的学习以及软件设计水平的提高打下良好基础。

计算机解决具体问题的步骤：从具体问题中抽象出数学模型——数据表示设计或选择一个求解此数学模型的算法——数据处理编程、运行。

1.1　数据结构的基本概念

在计算机进行数据处理的时候，因为计算机中的数据元素很多，所以这些数据元素在计算机中如何组织才能提高数据处理的效率并且节省计算机的存储空间，这是数据处理所面临的关键问题。数据结构这个概念正是因为这些问题而产生的，针对以上提出的问题，数据结构需要研究以下 3 个方面的问题：

1）数据集合中各数据元素之间的固有逻辑关系，即数据的逻辑结构。

2）在进行数据处理时，计算机中各数据元素的存储关系，即数据的存储结构。

3）各种数据结构之间的运算。

本节主要讨论工程上常用的一些基本数据结构，它们是软件设计的基础。

1.1.1　数据、数据元素和数据项的概念

1. 数据

数据是对客观事物的符号表示，在计算机科学中是指能输入到计算机中并被计算机存储、

加工的符号总称。计算机加工处理的数据已从早期的数值、布尔值等扩展到字符串、表格、语音、图片和图像等。

2. 数据元素

一般来说，现实世界中客观存在的一切个体都可以是数据元素，甚至每一个客观存在的事件，如一次演出、一次借书、一次比赛等也可以作为数据元素。准确的定义，数据元素是数据的基本单位，在程序中作为一个整体而加以考虑和处理。数据元素具有完整确定的实际意义，有时也称为元素、结点、顶点或记录等。

3. 数据项

数据项是数据的不可分割的最小标识单位。数据元素是由若干个数据项组成的。

1.1.2 数据结构概述

数据结构是由一个逻辑结构 S，一个定义在 S 上的基本运算集△和 S 的一个存储实现 D 所构成的整体（S，△，D）。数据结构包括逻辑结构和基本运算两部分，逻辑结构是用来完成数据表示的，基本运算是用来完成数据处理的。因此，数据结构涉及数据表示和数据处理两个方面。

1.1.3 数据的逻辑结构

1. 逻辑结构的基本概念

由上所述，一个数据结构包含以下两方面的信息：

1）表示数据元素的信息。

2）表示各数据元素之间的前后件关系。

在以上所述的数据结构中，其中数据元素之间的前后件关系是指它们的逻辑关系，而与它们在计算机中的存储位置无关。因此，上面所述的数据结构实际上是数据的逻辑结构。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，它反映了 D 中各数据元素之间的前后件关系，通常记为 R。即一个数据结构可表示为 B=(D,R)，其中 B 表示数据结构。为了反映 D 中各数据元素之间的前后件关系，一般用二元组来表示。

2. 逻辑结构的图形表示

数据元素间有 4 种基本逻辑结构：集合、线性结构、树型结构和图状结构。它们的结构可以用图形表示，一个数据结构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，一般称之为数据结点，简称结点；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。

通常一个数据结构中的元素结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结点（称为插入运算），也可以删除数据结构中的某个结点（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。在对数据结构的处理过程中，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。

3. 线性结构和非线性结构

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以下两个条件：

1）有且只有一个根结点。

2）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。如果一个数据结构不是线性结构，则称为非线性结构。

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，要根据具体情况来确定。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。

1.1.4　数据的存储结构

数据按逻辑结构规定的关系在计算机存储器中的存放方式称为数据的存储结构，又称数据的物理结构。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，常用的数据存储结构有顺序、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。

1.2　算法

本节要求理解算法、算法的时间复杂度和空间复杂度等概念；熟悉常用算法的时间复杂度次序和时间复杂度分析方法。

1.2.1　算法的基本概念

算法（Algorithm）一词源于算术（Algorism），均起源于古老的年代。在古代，人们把采用算术的方法求解未知问题的运算过程称为算法，如解题过程。在近代，人们把采用科学的方法完成某项事务的执行过程称为算法，如乐谱、菜谱、工作计划等。在现代，特别是计算机诞生之后，人们把计算机解题步骤称为计算机算法，如程序是一类计算机算法。现在谈到的算法实际是计算机算法的代名词，没有特别说明，算法指计算机算法。

算法是指解题方案的准确而完整的描述。算法规定了求解给定类型问题所需的所有“处理步骤”及其执行顺序，使得给定类型的任何问题能通过有限的指令序列、在有限的时间内被求解。其中每条指令表示一个或多个操作。每一个算法具有下列 5 个特性：

1）有穷性。算法必须能在有限的时间内做完，即能在执行有限个步骤后终止，包括合理

的执行时间的含义。

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）可行性。算法中的操作都必须是可行的，即必须是能具体实现的基本操作。每条指令都应在有限时间内完成。

4）输入。一个算法有零个或多个输入。

5）输出。一个算法有一个或多个有效信息的输出。

1.2.2 算法的基本要素

一个算法通常包括两种基本要素：一是对数据对象的运算和操作；二是算法的控制结构。

（1）对数据对象的运算和操作。

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，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。

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：加、减、乘、除等。

2）逻辑运算：“与”、“或”、“非”等。

3）关系运算：“大于”、“小于”、“等于”、“不等于”等。

4）数据传输：赋值、输入、输出等。

计算机程序也可以作为算法的一种描述，但由于在编制计算机程序时通常要考虑很多与方法和分析无关的细节问题（如语法规则），因此，在设计算法的一开始，通常并不直接用计算机程序来描述算法，而是用别的描述工具（如流程图、专门的算法描述语言甚至用自然语言）来描述算法。不管用哪种工具来描述算法，算法的设计一般都应从上述 4 种基本操作考虑，按解题要求从这些基本操作中选择合适的操作组成解题的操作序列。

（2）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、循环 3 种基本控制结构组合而成。

（3）算法的评价标准。

在算法设计中，只强调算法特性是不够的。一个算法除了满足 5 个特性之外，还应该有一个质量问题。一个问题可有若干个不同的求解算法，一个算法又可有若干个不同的程序来实现。在不同算法中有好算法，也有差算法，如针对同一问题，执行 10 分钟的算法要比执行 10 小时的算法好得多。设计高质量算法是设计高质量程序的基本前提。如何评价算法的质量呢？评价的标准是什么？不同时期、不同环境、不同情况其评价标准可能不同，会有差异，但一些基本评价标准是相同的。目前，评价算法质量有 4 个基本标准：

- 正确性：一个好算法必须保证运行结果正确。算法正确性，不能主观臆断，必须经过严格验证，一般不能说绝对正确，只能说正确性高低。目前程序正确性很难给出严格的数学证明，程序正确性证明尚处于研究阶段。要多选用现有的、经过时间考验的算法或采用科学规范的算法设计方法，是保证算法正确性的有效途径。

- 可读性：一个好算法应有良好的可读性，好的可读性有助于保证正确性。科学、规范的程序设计方法（结构化和面向对象方法）可提高算法的可读性。
- 通用性：一个好算法要尽可能通用，可适用一类问题的求解。如：设计求解一元二次方程 $2x^2+3x+1=0$ 的算法，该算法最好设计成求解一元二次方程 $ax^2+bx+c=0$ 的算法。
- 高效率：效率包括时间和空间两个方面。一个好的算法应执行速度快、运行时间短、占用内存少。效率和可读性往往是矛盾的，可读性要优先于效率。目前，在计算机速度比较快，内存比较大的情况下，高效率已处于次要地位。

1.2.3　算法设计基本方法

计算机解题的过程实际上是在实施某种算法，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集中算法设计方法：列举法、归纳法、递推、递归、减半递推技术、回溯法。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。因此，列举法常用于解决“是否存在”或“有多少种可能”等类型的问题，例如求解不定方程的问题。

（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。显然，归纳法要比列举法更能反映问题的本质，并且可以解决列举量为无限的问题。但是，从一个实际问题中总结归纳出一般的关系，并不是一件容易的事情，尤其是要归纳出一个数学模型更为困难。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。

（3）递推。

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。其中初始条件或是问题本身已经给定，或是通过对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。

（4）递归。

递归分为直接递归与间接递归两种。如果一个算法 P 显式地调用自己则称为直接递归调用。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。实际上，递归过程能将一个复杂的问题归结为若干个较简单的问题，然后将这些较简单的问题再归结为更简单的问题，这个过程可以一直做下去，直到最简单的问题为止。

（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着密切的关系。因此，利用分治法解决这类实际问题是有效的。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。

（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，对于每一步的试探，若试探成功，就得到问题的解；若试探失败，就逐步回退，换别的路线再进行试探，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。

1.2.4 算法复杂度

算法复杂度主要包括时间复杂度和空间复杂度。

1. 算法时间复杂度

算法时间复杂度是指执行算法所需要的计算工作量。算法的工作量用算法所执行的基本运算次数来度量，而算法所执行的基本运算次数是问题规模的函数，即：

$$\text{算法工作量}=f(n)$$

其中 n 是问题的规模。例如，两个 n 阶矩阵相乘所需要的基本运算（即两个实数的乘法）次数为 n^3，即计算工作量为 n^3，也就是时间复杂度为 n^3。

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时，可以用以下两种方法来分析算法的工作量。

（1）平均性态。

所谓平均性态分析，是指用各种特定输入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（用 A(n)表示）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，算法执行时所有可能输入的集合。这个等式中的 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值来确定，通常是不能解析地加以计算的。如果确定 p(x) 比较困难，则会给平均性态的分析带来困难。

（2）最坏情况复杂性。

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

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

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. 算法空间复杂度

算法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的附加存储空间。如果额外空间量相对于问题规模来说是常数，则称该算法是原地工作的。在许多实际问题中，为了减少算法所占的存储空间，通过采用压缩存储技术，以便尽量减少不必要的额外空间。

1.3　线性表及其顺序存储结构

1.3.1　线性结构与线性表的基本概念

1. 线性结构

线性结构是 n（n>0）个数据元素（结点）的有穷序列。一个数据元素可以由若干个数据项组成，但同一个线性结构中的元素必定具有相同的特性，属于同一数据对象，相邻数据元素之间存在着序偶关系。

通常，线性结构是由 n 个数据元素 a_1，a_2，…，a_n 组成的一个有限序列，其中每一个数据元素，除了第一个外，有且只有一个前件，除了最后一个外，有且只有一个后件，可以表示为：a_1，a_2，…，a_i，…，a_n

其中 a_i（i=1,2, …,n）是属于数据对象的元素，通常也称其为线性结构中的一个结点。

2. 线性结构的基本特征

线性结构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。结点个数 n 称为线性表的长度，当 n=0 时，称为空表。

在线性结构中，这种邻接关系是一对一的，即每个结点至多只有一个前件，并且至多也只有一个后件。而所有结点按一对一的邻接关系构成的整体就是线性结构。

3. 线性表的概念

线性表的逻辑结构是线性结构，所含结点的个数称为线性表的长度（简称表长）。表长为 0 的线性表称为空表。

1.3.2　顺序表的基本概念

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中的数据元素。所有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。顺序表具有以下两个基本特点：

1）顺序表中所有元素的所占的存储空间是连续的。

2）顺序表中各数据元素在存储空间中是按逻辑顺序依次存放的。

由此可以看出，在顺序表中，其前后件两个元素在存储空间中是紧邻的，且前件元素一定存储在后件元素的前面。

在顺序表中，如果顺序表中各数据元素所占的存储空间（字节数）相等，则要在该顺序表中查找某一个元素是很方便的。

假设顺序表中的第一个数据元素的存储地址（指第一个字节，即首地址）为 ADR（a_1），每一个数据元素占 k 个字节，则顺序表中第 i 个元素 a_i 在计算机存储空间中的存储地址为

$$ADR(a_i)=ADR(a_1)+(i-1)k$$

即在顺序表中每一个数据元素在计算机存储空间中的存储地址由该元素在顺序表中的位

置序号惟一确定。

一般来说，长度为 n 的顺序表

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

在计算机中的顺序存储结构如图 1-1 所示。

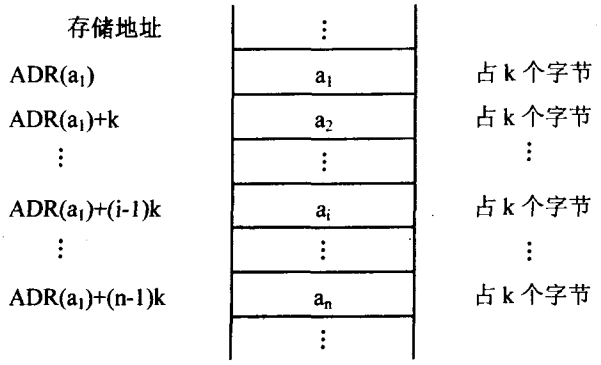

图 1-1 顺序表的顺序存储结构

在程序设计语言中，通常定义一个一维数组来表示顺序表的存储空间。因为程序设计语言中的一维数组与计算机中实际的存储空间是类似的，这就便于用程序设计语言对顺序表进行各种运算处理。

在用一维数组存放顺序表时，该一维数组的长度通常要定义得比顺序表的实际长度大一些，以便对顺序表进行各种运算，特别是插入运算。在一般情况下，如果顺序表的长度在处理过程中是动态变化的，则在开辟顺序表的存储空间时要考虑到顺序表在动态变化过程中可能达到的最大长度。如果开始时所开辟的存储空间太大，而实际上又用不着那么大的存储空间，则会造成存储空间的浪费。在实际应用中，可以根据顺序表动态变化过程中的一般规模来决定开辟的存储空间量。

在顺序表的顺序存储结构中，可以对顺序表进行各种处理。主要运算有以下几种：

- 在顺序表的指定位置处加入一个新的元素（即顺序表的插入）。
- 在顺序表中删除指定的元素（即顺序表的删除）。
- 在顺序表中查找某个（或某些）特定的元素（即顺序表的查找）。
- 对顺序表中的元素进行整序（即顺序表的排序）。
- 按要求将一个顺序表分解成多个顺序表（即顺序表的分解）。
- 按要求将多个顺序表合并成一个顺序表（即顺序表的合并）。
- 复制一个顺序表（即顺序表的复制）。
- 逆转一个顺序表（即顺序表的逆转）等。

下面一节主要讨论顺序表在顺序存储结构下的插入与删除问题。

1.3.3 顺序表的插入、删除运算

1. 插入运算

顺序表的插入运算是指在长度为 n 的顺序表（$a_1, a_2, \cdots, a_i, \cdots, a_n$）中，在顺序表的第 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 的顺序表（$a_1, a_2, \cdots, b, a_i, \cdots, a_n$）。

在一般情况下，要在第 i 个元素之前插一个新元素时，首先要从最后一个元素开始，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，顺序表的长度就增加了 1。

显然，在顺序表采用顺序存储结构时，如果插入运算在顺序表的末尾进行，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；如果要在顺序表的第 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 个元素之前进行，则原来第 i 个元素之后的所有元素都必须移动。在平均情况下，要在顺序表中插入一个新元素，需要移动表中一半的元素。因此，在顺序表顺序存储的情况下，要插入一个新元素，其效率是很低的，特别是在顺序表比较大的情况下更为突出，由于数据元素的移动而消耗较多的处理时间。

2. 顺序表的删除运算

顺序的删除运算是指在长度为 n 的顺序表（a_1，a_2，…，a_i，…，a_n）中，将顺序表中的第 i 个元素 a_i 删去，使得长度为 n 的顺序表变为长度为 n-1 的顺序表（a_1，a_2，…，a_{i-1}，a_{i+1}，…，a_n）。

在一般情况下，要删除第 i 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；顺序表的长度就减小了 1。

显然，在顺序表采用顺序存储结构时，如果删除运算在顺序表的末尾进行，即删除第 n 个元素，则不需要移动表中的元素；如果要删除顺序表中的第 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 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，要在顺序表中删除一个元素，需要移动表中一半的元素。因此，在顺序表顺序存储的情况下，要删除一个元素，其效率也是很低的，特别是在顺序表比较大的情况下更为突出，由于数据元素的移动而消耗较多的处理时间。

由顺序表在顺序存储结构下的插入与删除运算可以看出，顺序表的顺序存储结构对于小顺序表或者其中元素不常变动的顺序表来说是合适的，因为顺序结构的结构比较简单。但这种顺序存储的方式对于元素经常需要变动的大顺序表就不太合适了，因为插入与删除对效率比较低。

1.4　线性链表、循环链表及其基本运算

1.4.1　线性链表及其基本运算

1. 线性链表的基本概念

前面主要讨论了顺序表（线性表的顺序存储结构）以及在顺序存储结构下的运算。线性表的顺序存储结构具有简单、运算方便等优点，特别是对于小线性表或长度固定的线性表，采用顺序存储结构的优越性更突出。

但是，线性表的顺序存储结构在某些情况下就显得不那么方便，运算效率也不高。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，而数据元素之间的逻辑关系是由指针域来确定的。链式存储方式即可用于表示线性结构，也可用于表示非线性结构。在用链式结构表示较复杂的非线性结构时，其指针域的个数要多一些。