



全国计算机等级考试命题研究中心 / 编著
飞思数字创意出版中心 / 监制

全国计算机等级考试

National Computer Rank Examination

飞思考试中心
Fecit Examination Center



考点分析、 题解与模拟

二级公共基础知识（最新版）

含

无纸化真考试题库

新大纲 新题型 新题库

考点精编 ▶ 专家分析最新大纲与真考试题库，权威揭秘各考点考核概率

真题题解 ▶ 精选真考试题，权威分析真考出题角度与考试难度

同步练习 ▶ 习题源自真考试题库，练习即为真考，以练促学步步通关

真考试题库 ▶ 全套真考试题库，与真考考题完全一致



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

全国计算机等级考试命题研究中心 / 编著
飞思数字创意出版中心 / 监制

全国计算机等级考试

National Computer Rank Examination

飞思考试中心
Fecit Examination Center

考点分析、 题解与模拟

二级公共基础知识（最新版）



电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书依据教育部考试中心最新发布的《全国计算机等级考试考试大纲》，在《全国计算机等级考试考点分析、题解与模拟（2012版）》的基础上修订而成。在编写过程中，一方面结合最新大纲和数套真卷，对重要考点进行了分析、讲解，并选取经典考题进行了深入剖析；另一方面配有同步练习、无纸化考试试题，以逐步向考生详尽透析考试中的所有知识要点。“一书在手，通关无忧”。

本书适合作为全国计算机等级考试考前培训班辅导用书，也可作为应试人员的自学用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

全国计算机等级考试考点分析、题解与模拟：最新版. 二级公共基础知识 / 全国计算机等级考试命题研究中心编著.
北京：电子工业出版社，2013.1

（飞思考试中心）

ISBN 978-7-121-19121-3

I. ①全… II. ①全… III. ①电子计算机—水平考试—自学参考资料 IV. ①TP3

中国版本图书馆 CIP 数据核字（2012）第 286212 号

责任编辑：王树伟

特约编辑：赵树刚

印 刷：涿州市京南印刷厂

装 订：涿州市京南印刷厂

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本：880×1230 1/16 印张：10.75 字数：412.8 千字

印 次：2013 年 1 月第 1 次印刷

定 价：22.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zltts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

丛书编委会

主编 张明涛

编委 (排名不分先后)

丁海艳	万克星	马立娟	朱爱彬
王 伟	王 亮	王强国	王 磊
卢文毅	卢继军	任海艳	乔 影
刘之夫	刘金丽	刘春波	孙小稚
张 迪	张仪凡	张海刚	李 静
范二朋	李志红	杨 力	杨 闯
杨生喜	花 英	陈秋彤	周 辉
孟祥勇	欧海升	武 杰	范海双
郑 新	姜 涛	姜文宾	胡 杨
胡天星	赵 亮	赵东红	赵苡萱
王倩愉	倪海宇	高志军	高雪轩
董国明	谢公义	韩峻余	樊 钰

Preface

前言

全国计算机等级考试自 1994 年由国家教育部考试中心推出以来,为评测全社会非计算机专业人员的计算机知识与技能,培养各行业的计算机应用人才开辟了一条新的道路,受到了用人单位和学习人员的热烈欢迎。全国计算机等级考试通过数年的发展,已经成为我国最大型的计算机类考试。

为了帮助更多的学习者顺利地通过考试,并掌握相应的操作技能,我们在深入调研、详尽分析考试大纲的基础上,组织国内著名高校的计算机专家和一线教师编写了本书。

本书共分为两大部分。

※ 考点分析/真题题解/同步练习

“考点分析”结合最新考试大纲、教材,对教材中考核的重点和难点进行了讲解,内容涵盖了大纲中所有的考点。

“真题题解”选取极具代表性的经典真题作为例题。例题的选择力求突出考试考核重点,对题目的讲解深入、透彻,循序渐进,极有条理。

“同步练习”提供了大量习题,对前面所学的理论知识进行温习和巩固,以练促学、学练结合。

※ 无纸化真考试题

本部分试题全部选自无纸化真考试题库二级公共基础试题,配有详细文字解析。

由于时间仓促,书中难免有不当之处,敬请指正。

全国计算机等级考试命题研究中心
飞思数字创意出版中心
未来教育教学与研究中心

第 1 章 数据结构与算法

1.1 算法	1	1.6 树与二叉树	20
1.2 数据结构的基本概念	4	1.7 查找技术	29
1.3 线性表及顺序存储结构	7	1.8 排序技术	32
1.4 栈和队列	10	1.9 同步练习	41
1.5 线性链表	14	1.10 参考答案	48

第 2 章 程序设计基础

2.1 程序设计方法与风格	50	2.4 同步练习	60
2.2 结构化程序设计	52	2.5 参考答案	62
2.3 面向对象的程序设计	54		

第 3 章 软件工程基础

3.1 软件工程的基本概念	63	3.5 程序的调试	92
3.2 结构化分析方法	70	3.6 软件工程管理	94
3.3 结构化设计方法	77	3.7 同步练习	95
3.4 软件的测试	88	3.8 参考答案	108

第 4 章 数据库设计基础

4.1 数据库系统的基本概念	109	4.4 数据库的设计与管理	135
4.2 数据模型	121	4.5 同步练习	145
4.3 关系代数	132	4.6 参考答案	157

第 5 章 无纸化考试试题

5.1 无纸化考试试题(1)	158	5.4 无纸化考试试题(4)	160
5.2 无纸化考试试题(2)	159	5.5 无纸化考试试题(5)	161
5.3 无纸化考试试题(3)	160	5.6 参考答案及解析	162

第1章 数据结构与算法

考核知识点

- 算法的基本概念
- 数据结构的基本概念
- 线性表的定义
- 栈和队列
- 线性链表
- 循环链表
- 树的基本概念
- 查找技术
- 排序技术

重要考点提示

- 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
- 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、插入类排序)

1.1 算 法

考点 1 算法的基本概念

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。

算法(algorithm)是一组严谨地定义运算顺序的规则,并且每一个规则都是有效的,同时是明确的;此顺序将在有限的次数后终止。算法是对特定问题求解步骤的一种描述,它是指令的有限序列,其中每一条指令表示一个或多个操作。

1. 算法的基本特征

- (1)可行性(effectiveness):针对实际问题而设计的算法,执行后能够得到满意的结果。
- (2)确定性(definiteness):算法中的每一个步骤都必须有明确的定义,不允许有模棱两可的解释和多义性。
- (3)有穷性(finiteness):算法必须在有限时间内做完,即算法必须能在执行有限个步骤之后终止。
- (4)拥有足够的情报:要使算法有效必须为算法提供足够的情报。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

(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。

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。

(2)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。

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一定的联系。

(1)列举法

列举法是计算机算法中的一个基础算法。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,使方案优化,尽量减少运算工作量,是应该重点注意的。

(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。

(3)递推

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。递推本质上属于归纳法,工程上许多递推关系式实际上是通过分析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。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

人们在解决一些复杂问题时,为了降低问题的复杂程度(如问题的规模等),一般总是将问题逐层分解,最后归结为一些最简单的问题。这种将问题逐层分解的过程,实际上并没有对问题进行求解,而只是当解决了最后那些最简单的问题后,再沿着原来分解的逆过程逐步进行综合,这就是递归的基本思想。

递归分为直接递归与间接递归两种。

(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联系。因此,利用分治法解决这类实际问题是有效的。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

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,若试探成功,就得到问题的解;若试探失败,就逐步回退,换别的路线再逐步试探。

4. 算法设计的要求

通常一个好的算法应达到如下目标:

(1) 正确性(correctness)

正确性大体可以分为以下4个层次:

- ①程序不含语法错误;
- ②程序对于几组输入数据能够得出满足规格说明要求的结果;
- ③程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据能够得出满足规格说明要求的结果;
- ④程序对于一切合法的输入数据都能产生满足规格说明要求的结果。

(2) 可读性(readability)

算法主要是为了方便人的阅读与交流,其次才是其执行。可读性好有助于用户对算法的理解;晦涩难懂的程序易于隐藏较多错误,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地输出结果。

(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。

考点 2 算法的复杂度**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。这表明使用绝对的时间单位衡量算法的效率是不合适的。撇开这些与计算机硬件、软件有关的因素,可以认为一个特定算法“运行工作量”的大小,只依赖于问题的规模(通常用整数 n 表示),它是问题的规模函数。即

$$\text{算法的工作量} = f(n)$$

例如,在 $N \times N$ 矩阵相乘的算法中,整个算法的执行时间与该基本操作(乘法)重复执行的次数 n^3 成正比,也就是时间复杂度为 n^3 ,即

$$f(n) = O(n^3)$$

在有的情况下,算法中的基本操作重复执行的次数还随问题的输入数据集不同而不同。例如,在起泡排序的算法中,当要排序的数组 a 初始序列为自小至大有序时,基本操作的执行次数为0;当初始序列为自大至小有序时,基本操作的执行次数为 $n(n-1)/2$ 。对这类算法的分析,可以采用以下两种方法来分析。

(1) 平均性态(Average Behavior)

所谓平均性态是指利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,算法执行的所有可能输入的集合。

(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。

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

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空间、输入的初始数据所占的存储空间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及某种数据结构所需要的附加存储空间。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(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。

真题题解

【例1】算法的时间复杂度取决于()。

- A) 问题的规模
- C) 问题的难度

- B) 待处理的数据的初态
- D) A) 和 B)

【命题目的】影响算法时间复杂度的主要因素。

【解題要点】算法的时间复杂度不仅与问题的规模有关,而且与输入数据有关,即输入数据所有的可能取值范围及输入各种数据或数据集的概率有关。

【错解分析】算法的时间复杂度与问题的难度无关,而是与问题的规模和数据初态有关,所以 C) 选项错误, A) 选项和 B) 选项正确。

【考点链接】算法的空间复杂度及算法时间复杂度和空间复杂度的计算方法。

【答案】D)

1.2 数据结构的基本概念

考点 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,即数据的组织形式。

数据结构作为计算机的一门学科,主要研究和讨论以下 3 个方面:

- (1) 数据集中各数据元素之间所固有的逻辑关系,即数据的逻辑结构;
- (2) 在对数据元素进行处理时,各数据元素在计算机中的存储关系,即数据的存储结构;
- (3) 对各种数据结构进行的运算。

讨论以上问题的目的是提高数据处理的效率,所谓提高数据处理的效率有两个方面:

- (1) 提高数据处理的速度;
- (2) 尽量节省在数据处理过程中所占用的计算机存储空间。

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。

数据对象(data object):是性质相同的数据元素的集合,是数据的一个子集。

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。

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

数据结构是指反映数据元素之间的关系的的数据元素集合的表示。更通俗地说,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以下两方面信息:

- (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。

数据的逻辑结构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定义在此集合中的若干关系来表示。

数据的逻辑结构包括集合、线性结构、树型结构和图形结构四种。

线性结构:数据元素之间构成一种顺序的线性关系,如图 1-1 所示。

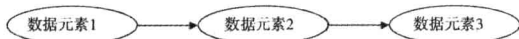


图 1-1 线性结构

树型结构:数据元素之间形成一种树型的关系,如图 1-2 所示。

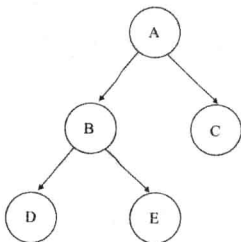


图 1-2 树型结构

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,它反映了数据元素之间的前后件关系,通常记为 R 。一个数据结构可以表示成:

$$B = (D, R)$$

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。

例如,复数是一种数据结构,在计算机科学中,复数可取如下定义:

$$B = (C, R)$$

其中, C 是含有两个实数的集合 $\{c_1, c_2\}$; R 是定义在集合 C 上的一种关系 $\{ \langle c_1, c_2 \rangle \}$, 其中有序偶 $\{ \langle c_1, c_2 \rangle \}$ 表示 c_1 是复数的实部, c_2 是复数的虚部。

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)。

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。因此,在进行数据处理时,选择合适的存储结构是很重要的。

考点 4 数据结构的图形表示

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,一般称为数据结点,并简称为结点;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。

例如,反映家庭成员间辈分关系的数据结构可以用如图 1-3 所示。

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)。

一个数据结构中的结点可能是动态变化的。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。

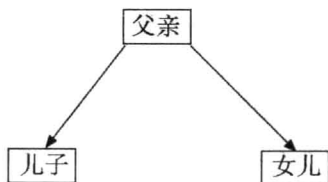


图 1-3 家庭成员辈分关系数据结构的图形表示

考点 5 线性结构与非线性结构

如果在一个数据结构中一个数据元素都没有,则称该数据结构为空的数据结构。

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。

非空数据结构满足:

- (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。线性结构又称为线性表。一个线性表是 n 个数据元素的有限序列。至于每个元素的具体含义,在不同的情况下各不相同,它可以是一个数或一个符号,也可以是一页书,甚至其他更复杂的信息。如果一个数据结构不是线性结构,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。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。

真题题解

【例 1】数据在计算机内存中的表示是指()。

- A) 数据的存储结构
- B) 数据结构
- C) 数据的逻辑结构
- D) 数据元素之间的关系

【命题目的】考查数据存储结构的定义。

【解题要点】数据的存储结构是数据结构在计算机内存中的表示,它既保存数据元素也保存数据元素之间的关系。

【错解分析】数据结构是指相互之间存在一种或者多种特定关系的数据元素的集合,它包括数据的逻辑结构和存储结构,因此 B) 选项错误。数据的逻辑结构描述的是数据之间的逻辑关系,因此 C) 选项错误。数据元素之间的关系就是指数据的逻辑结构,因此 D) 选项错误。

【考点链接】数据结构和数据逻辑结构的定义。

【答案】A)

【例 2】在数据结构中,与所使用的计算机无关的数据结构是()。

- A) 逻辑
- B) 存储
- C) 逻辑和存储
- D) 物理

【命题目的】考查数据的逻辑结构和存储结构的特点。

【解题要点】存储结构、物理结构是同一概念的两个术语,都是数据结构在计算机内存中的表示,逻辑结构是数据元素间关系的描述,与所用的计算机无关。

【错解分析】数据的存储结构又称为数据的物理结构,是指数据在计算机内存中的表示,与所使用的计算机密切相关,因此 B) 选项、C) 选项和 D) 选项都是错误的。

【考点链接】数据的存储结构和逻辑结构的定义及特点。

【答案】A)

【例 3】在数据结构中,从逻辑上可以把数据结构分成()。

- A) 动态结构和静态结构

- B) 紧凑结构和非紧凑结构
- C) 线性结构和非线性结构
- D) 内部结构和外部结构

【命题目的】考查数据结构从逻辑上的分类。

【解题要点】逻辑结构反映数据元素之间的逻辑关系,线性结构表示数据元素之间为一对一的关系,非线性结构表示数据元素之间为一对多或者多对一的关系,所以答案为 C)。

【错解分析】数据的逻辑结构就是指数据元素之间的相互关系,A) 选项、B) 选项和 D) 选项均不能表示数据元素之间的关系,只有 C) 选项可以。线性结构和非线性结构是根据数据元素之间关系的不同特性加以分类的,即从逻辑上加以分类。

【考点链接】数据存储结构的分类。

【答案】C)

1.3 线性表及顺序存储结构

考点 6 线性表的定义

线性表是 $n(n \geq 0)$ 个元素构成的有限序列 $(a_1, a_2, \dots, a_n)$ 。表中的每一个数据元素,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。即线性表是一个空表,或可以表示为

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

其中 $a_i(i=1, 2, \dots, n)$ 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。

其中,每个元素可以简单到是一个字母或是一个数据,也可能是比较复杂的由多个数据项组成的。在复杂的线性表中,由若干数据项组成的数据元素称为记录(record),而由多个记录构成的线性表又称为文件(file)。在非空表中的每个数据元素都有一个确定的位置,如 a_1 是第一个元素, a_n 是最后一个数据元素, a_i 是第 i 个数据元素,称 i 为数据元素 a_i 在线性表中的位序。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

线性表的顺序表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。

线性表的顺序存储结构具备如下两个基本特征:

- (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,并以所占的存储位置 $ADR(a_{i+1})$ 和第 i 个数据元素的存储位置 $ADR(a_i)$ 之间满足下列关系:

$$ADR(a_{i+1}) = ADR(a_i) + k$$

线性表第 i 个元素 a_i 的存储位置为

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

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

线性表的这种表示称做线性表的顺序存储结构或顺序映像,这种存储结构的线性表为顺序表。表中每一个元素的存储位置都和线性表的起始位置相差一个和数据元素在线性表中的位序成正比例的常数。如图 1-4 所示。由此只要确定了存储线性表的起始位置,线性表中任一数据元素都可以随机存取,所以线性表

的顺序存储结构是一种随机存取的存储结构。

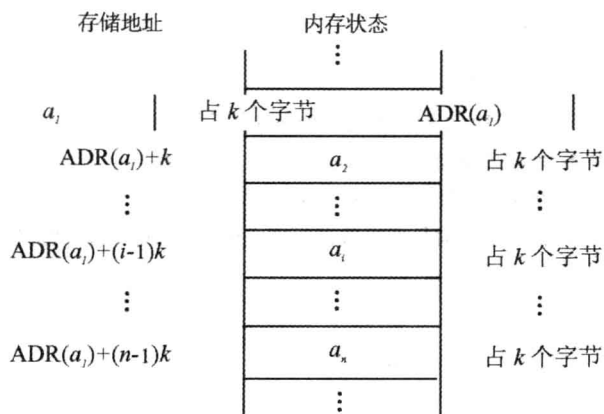


图 1-4 线性表顺序存储结构示意图

在程序设计语言中,通常定义一个一维数组来表示线性表的顺序存储空间。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整序(即线性表的排序);
- (5) 按要求将一个线性表分解成多个线性表(即线性表的分解);
- (6) 按要求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)等。

考点 8 顺序表的插入运算

线性表的插入运算是指在表的第 $i(1 \leq i \leq n+1)$ 个位置上,插入一个新结点 x ,使长度为 n 的线性表

$$(a_1, \dots, a_{i-1}, a_i, \dots, a_n)$$

变成长度为 $n+1$ 的线性表

$$(a_1, \dots, a_{i-1}, x, a_i, \dots, a_n)$$

现在分析算法的复杂度。这里的问题规模是表的长度,设它的值为 n 。该算法的时间主要花费在循环的结点后移语句上,该语句的执行次数(即移动结点的次数)是 $n-i+1$ 。由此可看出,所需移动结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,结点后移语句将不进行;这是最好情况,其时间复杂度为 $O(1)$;

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

由于插入可能在表中任何位置上进行,因此需分析算法的平均复杂度。

在长度为 n 的线性表中第 i 个位置上插入一个结点,令 $Eis(n)$ 表示移动结点的期望值(即移动的平均次数),则在第 i 个位置上插入一个结点的移动次数为 $n-i+1$ 。故

$$Eis(n) = \sum_{i=1}^{n+1} p_i (n-i+1)$$

不失一般性,假设在表中任何位置($1 \leq i \leq n+1$)上插入结点的机会是均等的,则

$$p_1 = p_2 = p_3 = \dots = p_{n+1} = 1/(n+1)$$

因此,在等概率插入的情况下,

$$Eis(n) = \sum_{i=1}^{n+1} p_i (n-i+1) / (n+1) = n/2$$

也就是说,在顺序表上做插入运算,平均要移动表上一半的结点。当表长 n 较大时,算法的效率相当低。虽然 $Eis(n)$ 中 n 的系数较小,但就数量级而言,它仍然是线性级的。因此算法的平均时间复杂度为 $O(n)$,如图 1-5 所示。

插入位置	移动元素个数	内存状态	
1	n	<table><tr><td>a_1</td></tr></table>	a_1
a_1			
2	$n-1$	<table><tr><td>a_2</td></tr></table>	a_2
a_2			
$\vdots$	$\vdots$	<table><tr><td>$\vdots$</td></tr></table>	$\vdots$
$\vdots$			
i	$n-i+1$	<table><tr><td>a_i</td></tr></table>	a_i
a_i			
$\vdots$	$\vdots$	<table><tr><td>$\vdots$</td></tr></table>	$\vdots$
$\vdots$			
n	1	<table><tr><td>a_n</td></tr></table>	a_n
a_n			
$n+1$	0	<table><tr><td></td></tr></table>	

图 1-5 插入操作时间复杂度示意图

考点 9 顺序表的删除运算

线性表的删除运算是指将表的第 $i (1 \leq i \leq n)$ 个结点删除,使长度为 n 的线性表:

$$(a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n)$$

变成长度为 $n-1$ 的线性表:

$$(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$$

该算法的时间分析与插入算法相似,结点的移动次数也是由表长 n 和位置 i 决定。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,无须移动结点;若 $i=1$,则前移语句将循环执行 $n-1$ 次,需移动表中除开始结点外的所有结点。这两种情况下算法的时间复杂度分别为 $O(1)$ 和 $O(n)$ 。

删除算法的平均性能分析与插入算法相似。在长度为 n 的线性表中删除一个结点,令 $Ede(n)$ 表示所需移动结点的平均次数,删除表中第 i 个结点的移动次数为 $n-i$,故

$$Ede(n) = \sum_{i=1}^n p_i (n-i)$$

式子中, p_i 表示删除表中第 i 个结点的概率。在等概率的假设下,

$$p_1 = p_2 = p_3 = \dots = p_n = 1/n$$

由此可得:

$$Ede(n) = \sum_{i=1}^n p_i (n-i) / n = (n-1)/2$$

即在顺序表上做删除运算,平均要移动表中约一半的结点,平均时间复杂度也是 $O(n)$ 。

删除元素位置	移动元素个数	内存状态	
1	$n-1$	<table><tr><td>a_1</td></tr></table>	a_1
a_1			
2	$n-2$	<table><tr><td>a_2</td></tr></table>	a_2
a_2			
$\vdots$	$\vdots$	<table><tr><td>$\vdots$</td></tr></table>	$\vdots$
$\vdots$			
i	$n-i$	<table><tr><td>a_i</td></tr></table>	a_i
a_i			
$\vdots$	$\vdots$	<table><tr><td>$\vdots$</td></tr></table>	$\vdots$
$\vdots$			
n	0	<table><tr><td>a_n</td></tr></table>	a_n
a_n			

图 1-6 删除操作时间复杂度示意图

真题题解

【例1】用数组表示线性表的优点是()。

- A) 便于插入和删除操作
- B) 便于随机存取
- C) 可以动态地分配存储空间
- D) 不需要占用一片相邻的存储空间

【命题目的】考查数组的存储方式。

【解题要点】由于数组的存储方式是采用顺序存储的,即占用连续的存储空间,所以可以用数组的下标直接存取。

【错解分析】对顺序存储的线性表进行插入和删除操作时,需要移动大量的元素,因此不便于进行插入删除操作,A)选项错误;数组的存储空间大小一开始就已经分配好,不能动态分配存储空间,所以C)选项错误;数组正是占用一片相邻的存储空间,所以D)选项也错误。

【考点链接】线性表的存储方式。

【答案】B)

1.4 栈和队列

考点10 栈及其基本运算

1. 什么是栈

栈实际也是线性表,只不过是一种特殊的线性表。栈(Stack)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)。当表中没有元素时称为空栈。栈顶元素总是后被插入的元素,从而也是最先被删除的元素;栈底元素总是最先被插入的元素,从而也是最后才能被删除的元素。

假设栈 $S = (a_1, a_2, a_3, \dots, a_n)$, 则 a_1 称为栈底元素, a_n 为栈顶元素。栈中元素按 $a_1, a_2, a_3, \dots, a_n$ 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(FILO, First In Last Out),或“后进先出”表(LIFO, Last In First Out),如图1-7所示。

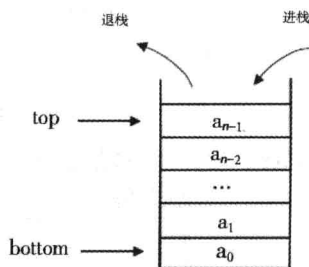


图1-7 进栈出栈示意图

2. 栈的顺序存储及其运算

(1)入栈运算:入栈运算是指在栈顶位置插入一个新元素。首先将栈顶指针加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,如图1-8所示。

(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,如图1-8所示。