

全国硕士研究生 入学统一考试

计算机学科专业基础综合

考题集与精解（第2版）

希赛教育研究生院 施游 黄少年 主编

新大纲
权威辅导



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

全国硕士研究生 入学统一考试

计算机学科专业基础综合

考题集与精解（第2版）

希赛教育研究生院 施游 黄少年 主编

电子工业出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书由希赛教育研究生院组织编写,紧密围绕考试大纲,把可能要考的知识点采用习题、习题分析与解答的形式呈现给读者,使读者的学习更具有针对性。根据经典的题型,精辟地指出题型的特点,阐述解题思路,使读者更好地了解考试题型,以及试题的解答方法和技巧。

本书是为备考全国硕士研究生入学统一考试计算机学科专业基础综合课考试的考生而量身定做的。

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

全国硕士研究生入学统一考试计算机学科专业基础综合考题集与精解 / 施游, 黄少年主编. —2 版. —北京: 电子工业出版社, 2010.10

(全国硕士研究生入学统考专用辅导丛书)

ISBN 978-7-121-11897-5

I. ①全… II. ①施… ②黄… III. ①电子计算机—研究生—入学考试—解题 IV. ①TP3-44

中国版本图书馆 CIP 数据核字 (2010) 第 187904 号

责任编辑: 李利健

印 刷: 北京中新伟业印刷有限公司

装 订:

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: 860×1092 1/16 印张: 25.25 字数: 576 千字 插页: 1

印 次: 2010 年 10 月第 1 次印刷

印 数: 4000 册 定价: 49.80 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 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

前 言

根据教育部文件要求,全国硕士研究生入学统一考试计算机科学与技术学科联考计算机学科专业基础,初试科目调整为4门,分别是政治理论(100分)、外语(100分)、数学一(150分)、计算机专业基础综合(150分)。其中,计算机专业基础综合考试内容涵盖数据结构、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,要求考生比较系统地掌握上述专业基础课的概念、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。

内容超值,针对性强

为了帮助考生熟练掌握考试大纲所规定的知识点,使考生能举一反三,希赛教育研究生院组织有关专家,在电子工业出版社的大力支持下,编写和出版了本书。本书紧密围绕考试大纲,把可能要考的知识点采用习题、习题分析与解答的形式呈现给读者,使读者的学习更具有针对性。根据经典的题型,精辟地指出题型的特点,阐述解题思路,使读者更好地了解考试题型,以及试题的解答方法和技巧。

由于编写组成员均为研究生考试第一线的辅导专家,负责和参与了历年的考试培训与辅导、教程编写、考试阅卷等方面的工作,因此,本书凝聚了专家们的知识、经验、心得和体会,集成了专家们的精力和心血。

根据希赛教育研究生院学员反馈的经验,通过习题形式来学习知识,能更加容易地掌握知识。同时,通过阅读本书,考生还可以清晰地把握命题思路,掌握知识点在试题中的变化,以便在研究生入学统一考试中洞察先机。

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,在职业教育和基础教育方面具有极高的权威性。特别是在在线教育方面,稳居国内首位,希赛教育的远程教育模式得到了国家教育部门的认可和推广。

希赛教育研究生院是全国计算机专业考研专业课辅导的权威机构,多年对计算机考研专业课考试跟踪与分析。根据考试大纲,组织权威专家编写和出版了考试教材、考试串讲、习题解答、考前冲刺与全真模拟等4个系列的辅导书籍,录制了考试培训视频教程、历年考试真题解析视频教程等2个系列的考研视频。在辅导方面,希赛教育研究生院实行个性化辅导,家教式服务,名师亲自制订辅导计划和批改作业,博士团队在线辅导。希赛教育研究生院具有自成体系的辅导资料,使学习更具系统性,复习更具针对性。实时的网络课堂和答疑系统,学员能与名师在线交流。希赛教育研究生院组织相关专家编写了高质量的模拟试题,能有的放矢地帮助学员备考。

本书由希赛教育研究生院施游和黄少年主编，由希赛教育首席专家张友生博士负责统稿。参加编写的人员有张友生、王勇、朱小平、胡钊源、刘毅、施游、梁英、史湘宁、王冀和谢顺。

在线测试，心中有数

上学吧（www.shangxueba.com）在线测试平台为考生准备了在线测试，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，系统自动判卷，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”。这样，系统就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。

如此，读者可利用上学吧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。

诸多帮助，诚挚致谢

在本书的编写过程中，参考了许多相关的文献和书籍，编者在此对这些参考文献的作者表示感谢。

感谢电子工业出版社的田小康和孙学瑛老师，他们在本书的策划、选题的申报、写作大纲的确定，以及编辑、出版等方面，付出了辛勤的劳动和智慧，给予了我们很多的支持和帮助。

感谢参加希赛教育研究生院辅导和培训的学员，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，且本书涉及的内容很广，书中难免存在错漏和不妥之处，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。

互动讨论，专家答疑

希赛教育研究生院（www.csaiky.com）是中国最大的计算机专业考研在线教育网站，该网站论坛是国内人气最旺的计算机专业考研社区，在这里，读者可以和数十万考生进行在线交流，讨论有关学习和考试的问题。希赛教育研究生院拥有强大的师资队伍，为读者提供全程的答疑服务，在线回答读者的提问。

有关本书的意见反馈和咨询，读者可在希赛教育研究生院考研论坛（bbs.csaiky.com）“考研教材”版块中的“希赛教育研究生院”栏目与我们交流，我们会及时地在线解答读者的疑问。

希赛教育研究生院

目 录

第 1 章 线性表	1
1.1 顺序存储结构的存储结构和实现	1
1.2 链式存储结构的存储结构和实现	6
第 2 章 栈、队列和数组	19
2.1 栈的概念、实现及应用	19
2.2 队列的概念、实现及应用	26
2.3 数组的概念和实现	29
第 3 章 树和二叉树	35
3.1 树的基本概念与性质	35
3.2 二叉树	36
3.3 森林的基本概念与性质	47
3.4 树的应用	50
第 4 章 图	54
4.1 图的概念和相关术语	54
4.2 图的存储及基本操作	57
4.3 图的遍历	63
4.4 图的基本应用及其复杂度分析	67
第 5 章 查找	78
5.1 静态查找法	78
5.2 动态查找法	82
5.3 散列 (Hash) 表及其查找	84
第 6 章 内部排序	89
6.1 排序的基本概念	89
6.2 插入排序	90
6.3 起泡排序	90
6.4 快速排序	91
6.5 选择排序	93

6.6 各种内部排序算法比较及应用	94
第 7 章 操作系统概述	97
7.1 操作系统的概念	97
7.2 操作系统的发展与分类	99
第 8 章 进程管理	103
8.1 进程与线程	103
8.2 进程同步	108
8.3 处理机调度	151
8.4 死锁	157
第 9 章 内存管理	164
9.1 内存管理基础	164
9.2 虚拟内存管理	167
第 10 章 文件管理	176
10.1 文件系统基础与实现	176
10.2 磁盘组织与管理	182
第 11 章 输入/输出 (I/O) 管理	187
第 12 章 计算机系统概述	190
12.1 计算机发展历程	190
12.2 计算机系统层次结构	192
12.3 计算机性能指标	196
第 13 章 数据的表示和运算	197
13.1 数制与编码	197
13.2 定点数的表示和运算	201
13.3 浮点数的表示和运算	205
13.4 算术逻辑单元	211
第 14 章 存储器层次结构	216
14.1 存储器的分类	216
14.2 存储器的层次化结构	220
14.3 半导体随机存取存储器	222
14.4 只读存储器	225
14.5 主存储器与 CPU 的连接	227

14.6	双口 RAM 和多模块存储器	231
14.7	高速缓冲存储器 (Cache)	233
14.8	虚拟存储器	237
第 15 章	指令系统	240
15.1	指令格式	240
15.2	指令的寻址方式	244
15.3	CISC 和 RISC 的基本概念	252
第 16 章	中央处理器 (CPU)	254
16.1	CPU 的功能和基本结构	254
16.2	指令执行过程	258
16.3	数据通路的功能和基本结构	260
16.4	控制器的功能和工作原理	262
第 17 章	总线	270
17.1	总线概述	270
17.2	总线仲裁	275
17.3	总线操作和定时	278
17.4	总线标准	279
第 18 章	输入/输出 (I/O) 系统	282
18.1	I/O 系统基本概念	282
18.2	外部设备	283
18.3	I/O 接口 (I/O 控制器)	285
18.4	I/O 方式	287
第 19 章	计算机网络体系结构	292
19.1	计算机网络概述	292
19.2	计算机网络体系结构与参考模型	297
第 20 章	物理层	304
20.1	通信基础	304
20.2	传输介质和物理设备	311
第 21 章	数据链路层	320
21.1	数据链路层功能、组帧	320
21.2	差错控制	323
21.3	流量控制与可靠传输机制、多路复用	332

21.4 局域网、CSMA 协议	339
21.5 广域网	347
21.6 数据链路层设备	352
第 22 章 网络层	359
22.1 路由算法、路由协议、网络层设备	359
22.2 IPv4、IPv6、组播	370
第 23 章 传输层	379
第 24 章 应用层	387
主要参考文献	394

第1章

线性表

1.1 顺序存储结构的存储结构和实现

一、选择题

1. 选择题题目部分

● 线性表是具有 n 个 (1) 的有限序列 ($n>0$)。

- (1) A. 表元素 B. 字符 C. 数据元素 D. 数据项

● 下述 (2) 是顺序存储结构的优点？

- (2) A. 存储密度大 B. 插入运算方便
C. 删除运算方便 D. 可方便地用于各种逻辑结构的存储表示

● 若长度为 n 的线性表采用顺序存储结构，在其第 i 个位置插入一个新元素的算法的时间复杂度为 (3) ($1 \leq i \leq n+1$)。

- (3) A. $O(0)$ B. $O(1)$ C. $O(n)$ D. $O(n^2)$

● 线性表的顺序存储结构，设其长度为 n ，在任何位置插入和删除操作都是等概率的。删除一个元素时平均大约需要移动表中的 (4) 个元素。

- (4) A. $n/2$ B. $(n-1)/2$ C. $n+1$ D. $n-1$

● 下面的叙述不正确的是 (5)。

- (5) A. 线性表在链式存储时，查找第 i 个元素的时间同 i 的值成正比
B. 线性表在链式存储时，查找第 i 个元素的时间同 i 的值无关
C. 线性表在顺序存储时，查找第 i 个元素的时间同 i 的值成正比
D. 线性表在顺序存储时，查找第 i 个元素的时间同 i 的值无关

2. 选择题的答案与分析

题号	(1)
答案	C

习题分析：

这道题主要考查对线性表定义的理解。线性表是一个有 n 个数据元素的有限序列。数据元素是数据

的基本单位，在计算机程序中，通常把数据元素作为基本单位进行处理。在复杂的数据结构中，一个数据元素又可以由若干个数据项组成，数据项是数据的最小单位。

题号	(2)
答案	A

习题分析：

这是一道基本概念题，主要考查对线性表顺序存储结构基本概念的理解。线性表的顺序存储结构就是指将线性表中的节点依次存放在计算机内存中一组地址连续的存储单元内。用 C 语言的数组即可实现线性表的顺序存储结构。在这种存储方式下，对线性表进行插入和删除运算需要移动大量的元素。对于一些比较复杂的逻辑结构，如树、图等，用顺序存储结构来实现时并不方便。因此，不难看出，正确答案为 A。

题号	(3)
答案	C

习题分析：

这道题主要考查对于线性表顺序存储结构的插入操作的理解。假定在线性表的任何位置上插入元素的概率 p_i 是相等的，则在长度为 n 的线性表中插入一个元素时需要移动元素的平均次数为 $\sum_{i=1}^{n+1} p_i(n-i+1) = \frac{1}{n+1} \sum_{i=1}^{n+1} (n-i+1) = \frac{n}{2}$ 。由此可见，在长度为 n 的线性表中插入元素的时间复杂度为 $O(n)$ 。

同理，可以计算出在长度为 n 的线性表中删除一个元素的时间复杂度也为 $O(n)$ 。对于这种题目，大家在复习时要能够计算出进行插入或删除操作时，平均需要移动元素的次数。

题号	(4)
答案	B

习题分析：

这道题与前一道题目类似，考查对于线性表的顺序存储结构的删除操作的理解。根据前一题的分析可知，在长度为 n 的线性表中删除一个元素时需要移动元素的平均次数为 $\frac{1}{n} \sum_{i=1}^n (n-i) = \frac{n-1}{2}$ 。

题号	(5)
答案	B、C

习题分析：

这道题考查对于线性表的顺序存储结构和链式存储结构特点的理解。顺序存储结构的特点是“顺序存储、随机存取”，也就是说，线性表在顺序存储时，查找第 i 个元素的时间与 i 的值无关。

而链式存储结构的特点则是“随机存储，顺序存取”，即链式存储结构的数据元素可以随机地存储在内存单元中，但访问其中的任意一个数据元素时，都必须从其头指针开始逐个进行访问。因此，正确答案为 B 和 C。

3. 训练自测表（如表 1-1 所示）

表 1-1 选择题练习自测表

题 号	考 查 点	得 分
(1)	线性表的定义	
(2)	顺序存储结构的特点	

续表

题 号	考 查 点	得 分
(3)	顺序存储结构的插入操作	
(4)	顺序存储结构的删除操作	
(5)	对线性表的理解	

二、综合应用题

1. 综合应用题题目部分

● 习题 1: 设顺序表 va 中的数据元素递增有序, 试编写一个算法, 将 x 插入到顺序表的适当位置上, 以保持该表的有序性。

● 习题 2: 试写一个算法, 实现顺序表的就地逆置, 即利用原表的存储空间将线性表 $(a_1, a_2, \dots, a_n)$ 逆置为 $(a_n, a_{n-1}, \dots, a_1)$ 。

● 习题 3: 已知长度为 n 的线性表 A 采用顺序存储结构, 请编写一个时间复杂度为 $O(n)$ 、空间复杂度为 $O(1)$ 的算法, 该算法删除线性表中所有值为 $item$ 的数据元素。 $(O(1)$ 表示算法的辅助空间为常量)。

● 习题 4: 顺序存储的线性表 A , 其数据元素为整型, 试编写一个算法, 将表 A 拆成 B 和 C 两个表, 使表 A 中元素值大于等于 0 的元素放入 B , 小于 0 的元素放入 C 中, 要求:

(1) 表 B 和 C 另外设置存储空间;

(2) 表 B 和 C 不另外设置, 而利用表 A 的空间。

● 习题 5: 已知 A 、 B 、 C 为 3 个递增有序的线性表, 现要求对表 A 做如下操作: 删去那些既在表 B 又在表 C 中出现的元素。试对顺序表编写实现上述操作的算法。

2. 综合应用题的答案与分析

若无明确说明, 综合应用题中涉及的线性表顺序存储结构, 假定其数据结构均为:

```
#define LIST_INIT_SIZE    100
#define LISTINCREMENT    10
typedef struct{
    ElemType *elem;
    int      length;
    int      listsize;
}Sqlist;
```

习题 1 分析:

这道题目考查线性表顺序存储结构的插入操作, 这里的重点是如何找到元素 x 应该插入的位置。要注意题中给出的已知条件为数据元素有序递增, 利用这一已知条件, 对线性表顺序存储结构的插入操作稍做修改即可得出相应的算法。

算法的基本思想是: 从表的最后一个元素开始, 从后往前进行扫描, 若该元素值大于等于 x , 则将该元素后移一位, 这样就可找到元素 x 应该插入的位置。该题的算法如下:

```
Status Insert_Sqlist(Sqlist &va, ElemType x) //把 x 插入递增有序表 va 中
{
    if(va.length+1>va.listsize) return ERROR;
    va.length++;
    for(i=va.length-1; va.elem[i]>=x&&i>=0; i--) //寻找元素 x 的插入位置
```

```

        va.elem[i+1]=va.elem[i];
    va.elem[i]=x;
    return OK;
}

```

习题2 分析:

这一题的特色在于在实现原表的逆置时, 需要利用原表的存储空间。实现方法也比较简单, 即将原表的第一个元素和最后一个元素相交换, 第二个元素和倒数第二个元素相交换……依此类推, 直到最后一个元素。

```

void Reverse(Sqlist & L)//顺序表的就地逆置
{
    Elemtype t;
    For(i=0,j=L.length-1; i<j; i++, j--)
    {
        t=L.elem[i]; L.elem[i]=L.elem[j]; L.elem[j]=t;
    }
}

```

习题3 分析:

该题目考查对线性表删除操作的理解, 一般情况下, 对线性表的元素进行删除操作时, 都涉及了元素移动的问题, 因此, 在线性表中删除一个元素的时间复杂度为 $O(n)$ 。

在本题中, 要删除表中所有值为 item 的元素, 即对表中的每个元素都要进行一次扫描, 且时间复杂度必须为 $O(n)$, 此时必然不能进行元素移动的操作, 而只能用另外的元素来替换被删除的元素。这里可以考虑两种情况: 一种是不考虑元素的相对顺序, 即被删除后的线性表中的元素的相对位置可以发生变化; 另一种是被删除后元素的相对位置不变。

考虑相对位置可以发生变化的情况, 在一趟扫描过程中, 若找到值为 item 的元素, 则用表中的最后一个元素替换。其算法如下:

```

void Delete(Sqlist &L)
{
    i=0;j=L.length-1;//设立数组的头、尾指针
    while(i<j)
    {
        while(i<j && L.elem[i]!=item)
            i++; //若值不为 item, 头指针右移
        if(i<j)while(i<j && A[j]==item)j--; //若右端元素值为 item, 尾指针左移
        if(i<j)A[i++]=A[j--]; //用尾指针指向的元素替换被删除的元素
    }
    L.length=j+1;
}

```

删除操作后其余元素的相对位置不变的情况应如何考虑呢? 请读者自己思考并写出相应的算法。

习题4 分析:

本题依旧考查对线性表基本操作的理解。将表 A 拆分成表 B 和表 C, 若表 B 和表 C 不利用原来的存储空间, 则此题比较简单。可以对表 A 的所有元素依次进行扫描, 若元素值大于等于 0, 则将该元素放入线性表 B; 若元素值小于 0, 则将其放入线性表 C。算法如下:

```

void Rearrange(Sqlist A, Sqlist &B, Sqlist &C)
{
    i=0;                                //i, j, k 是工作指针, 分别指向 A、B 和 C 表的当前元素
    j=k=-1;                             //j, k 初始化为-1
    while(i<A.length)
    {
        if(A[i]<0)                      //将小于零的元素放入 C 表
            C[++k]=A[i++];
        else                            //将大于零的元素放入 B 表
            B[++j]=A[i++];
    }
    If(j!=-1)
        B.length=j;
    else
        B.length=0;
    If(k!=-1)
        C.length=k;
    else
        C.length=0;
}

```

若将表 A 拆分成表 B 和表 C 时, 需要利用原有的存储空间, 此时又该如何考虑呢? 事实上, 此时就是对表 A 重新进行排序, 且将所有小于 0 的元素放在表 A 的前半部分, 而所有大于等于 0 的元素放在表 A 的后半部分。此时可采用快速排序的思想, 但枢轴记录的值并不作为元素比较的标准, 而是用元素值是否为负数作为比较的标准, 只需一趟快速排序即可完成本题的要求。其算法如下:

```

void Rearrange1(Sqlist &A)
{
    i=0; j=A.length-1
    t=A[0];    //暂存枢轴元素
    while(i<j)
    {
        while(i<j && A[j]>=0) j--;    //若当前元素为大于等于零, 则指针前移
        if(i<j)
            {A[i]=A[j]; i++;}        //将负数前移
        while(i<j && A[i]<0) i++;    //当前元素为负数时指针后移
        if(i<j)
            A[j--]=A[i];            //正数后移
    }
    A[i]=t;    //将原第一元素放到最终位置
}

```

作为对本题的扩展, 若线性表中有负数、零和正数 3 种类型, 并按照负数、零、正数的顺序重排线性表时应该如何处理呢? 请读者自己思考并写出相应的算法。

习题 5 分析:

本题考查对于线性表操作的综合运用。本题的前提条件依旧是 3 个线性表递增有序, 要完成题目的要求, 必须首先依次找出在表 B 和表 C 中同时出现的元素值, 再将表 A 中的该元素删除。要特别注意的是, 在本题目中没有特别指明表中的元素值各不相同, 因此这里要考虑表 A 中有连续多个要被删除元素的情况。

```
void SqList_Intersect_Delete(SqList &A, SqList B, SqList C)
{
    i=0;j=0;k=0;m=0;  //i 指示 A 中元素原来的位置, m 为移动后的位置
    while(i<A.length&&j<B.length&& k<C.length)
    {
        if(B.elem[j]<C.elem[k]) j++;
        else if(B.elem[j]>C.elem[k]) k++;
        else
        {
            same=B.elem[j];                                //找到了相同元素 same
            while(B.elem[j]==same) j++;
            while(C.elem[k]==same) k++;                    //j, k 后移到新的元素
            while(i<A.length&&A.elem[i]<same)
                A.elem[m++]=A.elem[i++];                  //需保留的元素移动到新位置
            while(i<A.length&&A.elem[i]==same) i++;        //跳过相同的元素
        }
    }
    while(i<A.length)
        A.elem[m++]=A.elem[i++];                          //A 的剩余元素重新存储
    A.length=m;
}
```

3. 训练自测表（如表 1-2 所示）

表 1-2 应用题练习自测表

题 号	考 查 点	得 分
(1)	顺序存储结构的插入操作	
(2)	顺序存储结构的就地逆置	
(3)	顺序存储结构的删除操作	
(4)	顺序存储结构的合并	
(5)	顺序存储结构中特定元素的删除操作	

1.2 链式存储结构的存储结构和实现

考试大纲涉及本节的知识点有：线性链表、循环链表和双向链表。

一、选择题

1. 选择题题目部分

- 线性表的链式存储结构，其地址 （1）。
 - (1) A. 必须是连续的 B. 一定是不连续的
 - C. 部分地址必须是连续的 D. 连续与否都可以
- 线性表的链式存储结构不具备的特点是 （2）。
 - (2) A. 插入和删除不需要移动元素 B. 可以随机地访问任意节点
 - C. 不必事先估计存储空间 D. 所需空间与线性长度成正比

- 下面关于线性表的叙述中, 错误的是 (3)。
- (3) A. 线性表采用顺序存储, 必须占用一片连续的存储单元
B. 线性表采用顺序存储, 便于进行插入和删除操作
C. 线性表采用链接存储, 不必占用一片连续的存储单元
D. 线性表采用链接存储, 便于插入和删除操作
- 线性表的链式存储结构中增加一个头节点是为了 (4)。
- (4) A. 使链式存储结构中至少有一个节点
B. 标识表节点中首元节点的位置
C. 方便运算的实现
D. 说明单链表是线性表的链式存储
- 将长度为 n 的单链表接到长度为 m 的单链表后面的算法的时间复杂度为 (5)。
- (5) A. $O(1)$
B. $O(n)$
C. $O(m)$
D. $O(n+m)$
- 若某线性表最常用的操作是存取任意指定序号的元素和在线性表尾进行插入和删除运算, 则利用 (6) 存储方式最节省时间。
- (6) A. 顺序存储线性表
B. 双链表
C. 带头节点的双循环链表
D. 循环链表
- 7. 以下关于静态链表的描述中, 正确的是 (7)。
- (7) A. 静态链表既有顺序存储的优点, 又有动态链表的优点。所以, 它存取表中第 i 个元素的时间与 i 无关
B. 静态链表不需要一开始就分配所有的存储空间, 可以在插入数据元素时再申请
C. 静态链表与动态链表在元素的插入、删除上类似, 不需要移动元素
D. 静态链表的指针域存储的是数据元素的内存地址
- 非空的循环单链表 L 的尾节点 p 满足 (8)。
- (8) A. $p \rightarrow \text{next} = L$
B. $p \rightarrow \text{next} = \text{NULL}$
C. $p = \text{NULL}$
D. $p = L$
- 设一个链表最常用的操作是在末尾插入节点和删除尾节点, 则选用 (9) 最节省时间。
- (9) A. 带头节点的双循环链表
B. 单循环链表
C. 带尾指针的单循环链表
D. 单链表
- 双向循环链表中, 在 p 所指向的节点之后插入 s 指向的节点, 其修改指针的操作是 (10), 其中 p 指向的不是最后一个节点。
- (10) A. $p \rightarrow \text{next} = s; s \rightarrow \text{prev} = p; p \rightarrow \text{next} \rightarrow \text{prev} = s; s \rightarrow \text{next} = p \rightarrow \text{next};$
B. $p \rightarrow \text{next} \rightarrow \text{prev} = s; p \rightarrow \text{next} = s; s \rightarrow \text{prev} = p; s \rightarrow \text{next} = p \rightarrow \text{next};$
C. $s \rightarrow \text{prev} = p; s \rightarrow \text{next} = p \rightarrow \text{next}; p \rightarrow \text{next} = s; p \rightarrow \text{next} \rightarrow \text{prev} = s;$
D. $s \rightarrow \text{prev} = p; s \rightarrow \text{next} = p \rightarrow \text{next}; p \rightarrow \text{next} \rightarrow \text{prev} = s; p \rightarrow \text{next} = s;$
- 对于一个头指针为 head 的带头节点的双向循环链表, 可以作为判定该线性表为空表的条件是 (11)。
- (11) A. $\text{head} = \text{NULL}$
B. $\text{head} \rightarrow \text{next} = \text{NULL}$
C. $\text{head} \rightarrow \text{next} = \text{head}$
D. $\text{head} \neq \text{NULL}$
- 以下关于线性表采用链式存储时的删除节点运算描述, 正确的是 (12)。
- (12) A. 带头节点的线性链表删除节点时, 不需要更改头指针
B. 带头节点的线性链表删除第一个节点时, 需要更改头指针
C. 不带头节点的线性链表删除节点时, 需要更改头指针
D. 不带头节点的线性链表删除第一个节点时, 不需要更改头指针

● 若某线性表最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素, 则采用 (13) 存储方式最节省时间。

- (13) A. 单链表
B. 仅有头指针的单循环链表
C. 双向链表
D. 仅有尾指针的单循环链表

● 静态链表中指针表示的是 (14)。

- (14) A. 内存地址 B. 数组下标 C. 下一元素地址 D. 左、右孩子地址

● 在单链表指针为 p 的节点之后插入指针为 s 的节点, 正确的操作是 (15)。

- (15) A. $p \rightarrow next = s; s \rightarrow next = p \rightarrow next$
B. $s \rightarrow next = p \rightarrow next; p \rightarrow next = s$
C. $p \rightarrow next = s; p \rightarrow next = s \rightarrow next$
D. $p \rightarrow next = s \rightarrow next; p \rightarrow next = s$

2. 选择题的答案与分析

题号	(1)
答案	D

习题分析:

这道题考查对于线性表的链式存储结构定义的理解。链式存储结构的特点是用一组任意的存储单元存储线性表的数据元素。也就是说存储单元之间可以是连续的, 也可以是不连续的。

题号	(2)
答案	B

习题分析:

顺序表可以随机访问表中的任意节点, 而链表必须从第一个数据节点出发, 按照节点在表中的位置才能查找。即顺序表是随机访问, 而链表是顺序访问。因此答案 B 是链表不具备的特点。

题号	(3)
答案	B

习题分析:

线性表采用顺序存储时, 进行插入和删除操作都需要移动大量元素, 因此显然 B 是错误的。

题号	(4)
答案	C

习题分析:

头节点的好处有两个: 第一, 有头节点后, 插入和删除数据元素的算法统一了, 不再需要判断是否在第一个元素之前插入和删除一个元素; 第二, 不论链表是否为空, 链表的头指针不变。因此可以看出, 设置头节点的目的就是为了方便运算的实现。

题号	(5)
答案	C

习题分析:

这道题考查对于线性表链式存储结构的操作的理解。设长度为 n 的单链表头指针为 L_n , 长度 m 的单链表的头指针为 L_m , 首先从指针 L_m 出发找到该链表的最后一个节点, 若该节点的指针为 p , 则使用语句 $p \rightarrow next = L_n$ 即可实现将链表 L_n 链接在链表 L_m 的后面。因此该操作所需要的时间主要集中在查找链表 L_m 的最后一个节点上, 即其时间复杂度为 $O(m)$ 。