严格依据2015年考研计算机专业基础综合考试大纲编写

高教版
2015

考研计算机专业基础综合考试大纲解析配套1000题

全国考研计算机大纲配套
教材专家委员会

最佳搭配：大纲解析+配套1000题+历年真题标准解析

登录中国教育考试在线www.eduexam.com.cn分享资源、课程和冲刺密卷

高等教育出版社
HIGHER EDUCATION PRESS

2015 KAOYAN JISUANJI ZHUANYE JICHU ZONGHE
KAOSHI DAGANG JIEXI PEITAO 1000 TI

高教版
2015

考研计算机专业基础综合考试大纲解析配套1000题

全国考研计算机大纲配套
教材专家委员会

最佳搭配：大纲解析+配套1000题+历年真题标准解析

登录中国教育考试在线www.eduexam.com.cn分享资源、课程和冲刺密卷

高等教育出版社·北京
HIGHER EDUCATION PRESS BEIJING

内容简介

《2015 考研计算机专业基础综合考试大纲解析配套 1000 题》依据 2015 年考研大纲和大纲解析将知识点、考点与试题结合，使考生通过难易适度的练习题达到巩固基础、掌握重点、提高解题能力的目的，真正实现记、练、用的结合。 通过学习《2015 考研计算机专业基础综合考试大纲解析配套 1000 题》考生能掌握真实考试的题型和难易度及答题技巧，使考生在正式考试时有种似曾相识的熟悉感。 考生最好看一章《大纲解析》内容做一章练习，以便巩固知识，提高学习效率。

图书在版编目(CIP)数据

2015 考研计算机专业基础综合考试大纲解析配套 1000 题 / 全国考研计算机大纲配套教材专家委员会编. - -北京:高等教育出版社,2014.8

ISBN 978 - 7 - 04 - 040521 - 7

Ⅰ.①2… Ⅱ.①全… Ⅲ.①电子计算机 - 研究生 - 入学考试 - 习题集 Ⅳ.①TP3 - 44

中国版本图书馆 CIP 数据核字(2014)第 178223 号

策划编辑	张耀明	责任编辑	何新权	封面设计	王 洋	版式设计	王艳红
插图绘制	杜晓丹	责任校对	王 雨	责任印制	田 甜		

出版发行	高等教育出版社	咨询电话	400 - 810 - 0598
社 址	北京市西城区德外大街 4 号	网 址	http://www.hep.edu.cn
邮政编码	100120		http://www.hep.com.cn
印 刷	北京铭成印刷有限公司	网上订购	http://www.landraco.com
开 本	787mm × 1092mm 1/16		http://www.landraco.com.cn
印 张	17	版 次	2014 年 8 月第 1 版
字 数	470 千字	印 次	2014 年 8 月第 1 次印刷
购书热线	010 - 58581118	定 价	33.00 元

物 料 号 40521 - 00

出版前言

一、教育部制定的**《2015年全国硕士研究生招生考试计算机科学与技术学科联考计算机学科专业基础综合考试大纲》**（以下简称《考试大纲》）规定了2015年全国硕士研究生招生考试计算机科目的考试范围、考试要求、考试形式、试卷结构等。它既是2015年全国硕士研究生招生考试计算机专业命题的唯一依据，也是考生复习备考必不可少的工具书。

二、**《2015全国硕士研究生招生考试计算机学科专业基础综合考试大纲解析》**（以下简称《大纲解析》）根据《考试大纲》的要求和最新精神，深入研究考研命题的特点及动态，结合作者多年阅卷工作总结，特别注重与考生的实际相结合，注重与考研的要求相结合。

本书由数据结构、计算机组成原理、操作系统、计算机网络四部分组成。其中各部分均包括以下三部分内容：

（一）复习要点——使考生明确各章的重点、难点及常考点，弄清各知识点之间的相互联系，以及多年考试中本章节的出题情况，以便对本章内容有一个全局性的认识和把握。

（二）考点精讲——本部分参考当前国内最权威的大学教材，对大纲所要求的知识点进行了全面、准确的阐述，以加深考生对基本概念和原理等重点内容的理解和正确应用。本部分讲解考点明确、重点突出、层次清晰、简明实用。

（三）例题与练习——通过对经典例题的分析教会考生分析问题、解决问题的方法和技巧；通过大量练习题使考生学练结合，更好地巩固所学知识，提高实战能力。

三、**《2015考研计算机专业基础综合考试大纲解析配套1000题》**由经验丰富的考研辅导专家根据全面调整后的《考试大纲》、《大纲解析》编写，将大纲和大纲解析中的考点、重点和难点与试题结合，使考生在学习《大纲解析》后通过难易适度的练习题达到检测复习效果、巩固基础、掌握重点、提高解题能力的目的，真正实现记、练、用的结合。

在开始复习时，最好把本书对照《大纲解析》复习，看一章即做一章相应的练习，以检测复习效果，帮助理解和掌握考点。本书可贯穿复习始终，前期可以作为同步训练，后期用于强化训练。

为了给考生提供更多的增值服务，凡购买正版考研大纲配套系列用书的考生都可以登录“中国教育考试在线”www.eduexam.com.cn做考研全真模拟试卷。

高等教育出版社

2014年8月

目　录

第一部分　数据结构

内容综述

一、内容体系

数据结构是主要研究数据的各种逻辑结构和在计算机中的存储结构，还研究对数据进行的插入、查找、删除、排序、遍历等基本运算或操作以及这些运算在各种存储结构上具体实现的算法。

数据结构部分在研究生考试中，主要考查的内容包括以下几个部分：

第一章线性表，主要考点包括线性表的基本定义、线性表的顺序存储结构、链式存储结构以及在这两种结构上实现插入、删除、查询等基本操作。

第二章栈、队列和数组，主要考点包括栈、队列、数组的基本定义，栈和队列的顺序存储结构和链式存储结构以及在各种存储结构中进行的操作，数组的压缩存储等。

第三章是树与二叉树，主要考点包括树的基本概念，二叉树的定义及其主要特性、存储结构、遍历过程，线索二叉树的基本概念和构造，树、森林与二叉树的转换，二叉排序树，平衡二叉树，哈夫曼(Huffman)树和哈夫曼编码等。这一章节在考试中占的分值较大，是重点内容，要引起重视。

第四章是图，主要考点包括图的概念、图的存储及基本操作、图的遍历以及图的基本应用。这一章可以出综合题考查，尤其是最小(代价)生成树、最短路径、拓扑排序、关键路径这几个知识点，务必掌握并熟练运用。

第五章是查找，主要考点包括查找的基本概念、顺序和折半查找方法、B-树与B+树的基本概念以及二者的区别、B-树的基本操作、散列表以及散列表的查找等。

第六章排序，主要考点包括内部排序的基本概念、各种内部排序算法及其时间复杂度和空间复杂度、各种内部排序算法的比较、外部排序、排序的应用等。

二、知识要点

数据结构在计算机基础综合考试中共占45分，所占的比例较高。考研试题中，共考查11道单项选择题和2道综合应用题。

根据对历年试卷的分析可知，主要考点分布在树与二叉树、图和排序三章中。其中，单项选择题中常见考点主要分布在时间复杂度的计算，栈的基本操作(入栈和出栈)，队列的基本操作(入队和出队)，循环队列，二叉树的基本性质，树与二叉树的遍历，树、森林与二叉树的转换，二叉排序树，平衡二叉树，图的基本概念，图的存储及基本操作，图的遍历，最小(代价)生成树，最短路径，拓扑排序，关键路径，查找的基本概念，顺序和折半查找方法，B-树与B+树的基本概念以及二者的区别，B-树的基本操作，各种内部排序算法的比较等。综合应用题的第一题(即考试中的第41题)约10分，通常考查各种数据结构的应用。常见的考点分布在树与二叉树的基本应用，图的基本应用，散列表的查找，各个内部排序算法及外部排序等。每年考题的第42题为算法设计类题目，约13分，主要考点分布在线性表的章节内，考生要熟练掌握线性表的逻辑结构、存储结构以及在各种存储结构上操作的实现。本题的考查要求通常分为三个部分：(1) 给出算法的基本设计思想；(2) 根据设计思想，采用C或C++或Java语言描述算法，关键之处给出注释；(3) 说明你所设计算法的时间复杂度。根据题目的要求可知，此类题目要求考生能够设计算法，熟练掌握C或C++或Java语言中的一种，并利用该语言完成编码。如果考生在这方面的基础很薄弱，建议多用精力复习C或C++或Java语言，并且多做练习。

三、复习建议

数据结构在计算机基础综合考试中相对难度较大，所占分值较高，涉及的考点众多且不易理解，在复习过程中应该引起足够的重视。数据结构的复习需要一定的基础，如果考生对 C 或 C++或 Java 语言不熟悉，建议先复习 C 或 C++或 Java 语言，将两门课程结合起来复习效果会更好一些。

一般情况下，数据结构的复习可以从基础阶段（每年的 5 月—7 月）开始。在数据结构的基础阶段中，要把重点放在对基本概念的理解和应用上，多看教材，将各个数据结构的逻辑结构和存储结构分清楚，对各种操作的实现过程要一清二楚。这一阶段主要的复习资料是教材，还要配合一定的基础习题练习，这样会达到事半功倍的效果。在基础阶段的复习结束后，可以进入强化阶段的复习（每年的 8 月—10 月）。强化阶段复习的目标是将考试中的重点和难点完全掌握并熟练运用。在这一阶段的复习过程中，要把历年考研真题仔细研究一遍，归纳出重点和难点，并总结命题规律。同时，可以选择一些典型的题目进行训练。强化阶段的复习结束后，可以进入冲刺阶段的复习（每年的 11 月—次年 1 月），这一阶段复习方法是做总结，尤其是考生不熟悉、模棱两可的考点，一定要在考试之前总结一遍，这样可以在考试过程中思路更清晰。冲刺阶段也可以选择做一些模拟题，但不要过多，重点还是放在对知识点的总结上。

当然，上述的方法并不是绝对地适用于每一个人，考生可以根据自身的基础情况灵活调整。

第一章　线性表

一、单项选择题

1. 若某线性表中最常用的操作是在最后一个结点之后插入一个结点和删除第一个结点，则下面最节省运算时间的存储方式是(　　)。

A. 单链表　　B. 带有头指针的单循环链表

C. 双链表　　D. 带有尾指针的单循环链表

2. 已知两个长度分别为 l 和 s 的降序链表，若将它们合并为一个长度为 $l+s$ 的升序链表，则最坏情况下的时间复杂度是(　　)。

A. $O(l)$　　B. $O(ls)$　　C. $O(\min(l,s))$　　D. $O(\max(l,s))$

3. 线性表中存放的主要是(　　)。

A. 整型常量　　B. 字符　　C. 数据元素　　D. 信息元素

4. 下面的叙述中正确的是(　　)。

Ⅰ. 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

Ⅱ. 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

Ⅲ. 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

A. 仅Ⅰ　　B. 仅Ⅱ　　C. 仅Ⅲ　　D. Ⅰ、Ⅱ、Ⅲ

5. 对于某线性表来说，主要的操作是存取任一指定序号的元素和在最后进行插入运算，那么应该选择(　　)存储方式最节省时间。

A. 顺序表　　B. 双链表

C. 带头结点的双循环链表　　D. 单循环链表

6. 若线性表最常用的运算是查找第 i 个元素及其前驱的值，则下列存储方式中最节省时间的是(　　)。

A. 单链表　　B. 双链表　　C. 单循环链表　　D. 顺序表

7. 如果线性表中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(　　)存储方式最节省运算时间。

A. 单链表　　B. 仅有头指针的单循环链表

C. 双链表　　D. 仅有尾指针的单循环链表

8. 算法的时间复杂度取决于(　　)。

A. 问题的规模　　B. 待处理数据的初态

C. A 和 B　　D. 以上都不正确

9. 关于链表的特点，下面的叙述中不正确的是(　　)。

A. 插入、删除运算方便　　B. 可实现随机访问任一元素

C. 不必事先估计存储空间　　D. 所需空间与线性长度成正比

10. 设线性表中有 $2n$ 个元素，以下操作中，在单链表上实现要比在顺序表上实现效率更高的是(　　)。

A. 删除指定元素

B. 在最后一个元素的后面插入一个新元素

C. 顺序输出前 k 个元素

D. 交换第 i 个元素和第 $2n-i-1$ 个元素的值($i=0,1,\cdots,n-1$)

11. 下面的算法实现的是带附加头结点的单链表数据结点逆序连接，空缺处应当填入(　　)。

```
void reverse(pointer h) {      //h 为附加头结点指针
    pointer p, q;
```

```
p=h->next; h->next=NULL;
while(p != null){
    q=p;
    p=p->next;
    q->next=h->next;
    h->next=(____);
}
}
```

A. h　　B. p　　C. q　　D. q->next

12. 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(　　)($1 \leqslant i \leqslant n+1$)。

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

13. 线性表($a_1, a_2, \cdots, a_n$)以链式存储方式存储时，访问第 i 位置元素的时间复杂度为(　　)。

A. $O(i)$　　B. $O(1)$　　C. $O(n)$　　D. $O(i-1)$

14. 非空的循环单链表 head 的尾结点 p 满足(　　)。

A. p->next=head　　B. p->next=NULL　　C. p=NULL　　D. p=head

15. 双向链表中有两个指针域，即 prior 和 next，分别指向前驱及后继，设 p 指向链表中的一个结点，q 指向一个待插入结点，现要求在 p 前插入 q，则正确的插入为(　　)。

A. p->prior=q;q->next=p;p->prior->next=q;q-> prior=p->prior;

B. q->prior = p->prior; p->prior->next=q;q->next=p;p->prior =q;

C. q->next=p;p->next=q;p->prior->next=q;q->next=p;

D. p->prior->next = q;q->next=p;q->prior=p->prior; p->prior=q;

16. 静态链表中指针表示的是(　　)。

A. 内存地址　　B. 数组下标

C. 下一元素数组下标　　D. 左、右孩子地址

17. 在单链表指针为 p 的结点之后插入指针为 s 的结点，正确的操作是(　　)。

A. p->next=s;s->next = p->next;　　B. s->next=p->next;p->next=s;

C. p->next=s;p->next=s->next;　　D. p->next=s->next;p->next=s;

18. 对于一个头指针为 head 的带头结点的单链表，判定该表为空表的条件是(　　)。

A. head==NULL　　B. head->next==NULL

C. head->next ==head　　D. head!=NULL

19. 以下与数据的存储结构无关的术语是(　　)。

A. 循环队列　　B. 链表　　C. 哈希表　　D. 栈

20. 以下数据结构中，(　　)是线性数据结构。

A. 广义表　　B. 二叉树　　C. 稀疏矩阵　　D. 串

二、综合应用题

1. 有两个集合 A 和 B，利用带头结点链表表示，设头指针分别为 la 和 lb。两集合的链表元素皆为递增有序。设计一个算法，将 A 与 B 合并，合并后仍然保持整个链表中的数据依次递增。不得利用额外的结点空间，只能在 A 和 B 的原有结点空间上完成。要求：

(1) 给出算法的基本设计思想。

(2) 根据设计思想，采用 C 或 C++或 Java 语言描述算法，关键之处给出注释。

(3) 分别给出算法各部分的时间复杂度。

2. 线性表($a_1, a_2, a_3, \cdots, a_n$)中元素递增有序且按顺序存储于计算机内。要求设计一算法用最少时间在表中查找数值为 x 的元素，并将其与后继元素位置相交换。如果线性表中找不到该元素，则将该元素插

入表中并使表中元素仍递增有序。

(1) 给出算法的基本设计思想。

(2) 根据设计思想,采用 C 或 C++或 Java 语言描述算法,关键之处给出注释。

(3) 分别给出算法各部分的时间复杂度。

3. 已知顺序表 A,在不改变顺序表中奇数号元素与偶数号元素相对位置的前提下,设计算法,将所有奇数号元素移到所有偶数号元素前。

(1) 给出算法的基本设计思想。

(2) 根据设计思想,采用 C 或 C++或 Java 语言描述算法,关键之处给出注释。

(3) 说明你所设计算法的时间复杂度和空间复杂度。

4. 已知单链表 L 是一个递增有序表,试写一高效算法,删除表中值大于 min 且小于 max 的结点(若表中有这样的结点),同时释放被删结点的空间,这里 min 和 max 是两个给定的参数。

5. 线性表$(a_1,a_2,a_3,\cdots,a_n)$中元素递增有序且按顺序存储于计算机内。要求设计算法完成以下内容:

(1) 用最少的时间在表中查找数值为 x 的元素。

(2) 若找到将其与后继元素位置相交换。

(3) 若找不到将其插入表中并使表中元素仍递增有序。

6. 设有一个双链表 L,每个结点中除有 prior、data 和 next 这 3 个域外,还有一个访问频度域 freq,在链表被启用之前,其值均初始化为零。每当在链表进行一次 LocateNode(L, x)运算时,令元素值为 x 的结点中 freq 域的值加 1,并调整表中结点的次序,使其按访问频度的递减排列,以便使频繁访问的结点总是靠近表头。试写一符合上述要求的 LocateNode 运算的算法。

7. 有一个不带头结点的单链表 list,链表中结点都有两个域:数据域 data 和指针域 link。已知初始时该单链表无序,请设计一个算法将该链表按结点数据域的值的大小,将其从小到大依次重新链接,在链接过程中不得使用除该链表以外的任何链结点空间。要求:

(1) 给出算法的基本设计思想。

(2) 根据设计思想,采用 C 或 C++或 Java 语言描述算法,关键之处给出注释。

8. 如果以单链表表示集合,设集合 A 用单链表 LA 表示,集合 B 用单链表 LB 表示,设计算法求两个集合的差,即 $A-B$。

9. 已知 3 个带头结点的线性链表 A、B、C 中的结点均依元素值自小至大非递减排列(可能存在两个以上值相同的结点),编写算法对链表 A 进行如下操作:使操作后的链表 A 中仅留下 3 个表中均包含的数据元素的结点,且没有值相同的结点,并释放所有无用结点。限定算法的时间复杂度为 $O(m+n+p)$,其中 m、n 和 p 分别为 3 个表的长度。

10. 已知非空链表 A,其指针是 list,链表中的结点由两部分组成:数据域 data 和指针域 link。设计一个算法,将链表中数据域值最小的那个链结点移到链表的最前面,在不额外申请新的链结点的情况下,使得算法时间复杂度和空间复杂度尽可能低。要求:

(1) 给出算法的基本设计思想。

(2) 根据设计思想,采用 C 或 C++或 Java 语言描述算法,关键之处给出注释。

11. 已知一个双向链表,其结点结构为数据域 data、左指针域 llink、右指针域 rlink;设指针 P 指向双向链表中的某个结点。写出一个算法,实现 P 所指向的结点和它的前缀结点之间顺序的互换。要求:

(1) 给出算法的基本设计思想。

(2) 根据设计思想,采用 C 或 C++或 Java 语言描述算法,关键之处给出注释。

12. 两个整数序列 $A=a_1,a_2,a_3,\cdots,a_m$ 和 $B=b_1,b_2,b_3,\cdots,b_n$ 已经存入两个单链表中,设计一个算法,判断序列 B 是否是序列 A 的子序列。

13. 已知一个带有头结点的单链表 L,其结点结构由两部分组成:数据域 data,指针域 link。设计一个算法,以最高效的方法实现在单链表中删除数据域最小值结点。要求:

(1) 给出算法的基本设计思想。

(2) 根据设计思想,采用 C 或 C++或 Java 语言描述算法,关键之处给出注释。

14. 设有集合 A 和集合 B,要求设计生成集合 $C=A\cap B$ 的算法,其中集合 A、集合 B 和集合 C 用链式存储结构表示。

参考答案及解析

一、单项选择题

1.【答案】 D。在链表中的最后一个结点之后插入一个结点要知道终端结点的地址，所以，单链表、带有头指针的单循环链表、双链表都不合适。考虑在带有尾指针的单循环链表中删除第一个结点，其时间性能是 $O(1)$，所以答案是 D。

2.【答案】 D。在合并过程中，最坏的情况是两个链表中的元素依次进行比较，比较的次数最少是 m 和 n 中的最大值。

3.【答案】 C。线性表中主要存放的是数据元素，而数据元素可以是整型也可以是字符型，但对于一个线性表来说，所有的数据元素的类型必须相同。

4.【答案】 A。在线性表链式存储结构中，查找第 i 个元素的时间与 i 的位置成正比。而在顺序存储结构中查找第 i 个元素的时间与 i 的位置无关。

5.【答案】 A。线性表中要想最省时间地存取某一指定序号的元素，那么就要利用顺序表这种存储方式。但顺序表不利于插入和删除运算，可是题目中强调是在最后进行插入运算，因此，本题最合适的选项是顺序表。

6.【答案】 D。本题的考点是线性表的存储结构及其特点。在线性表中主要的存储结构有顺序表和链表两种，其特点如下：

(1) 顺序表可以实现随机存取，其时间复杂度为 $O(1)$。但在顺序表中，进行插入和删除操作需要移动大量的元素，其时间复杂度为 $O(n)$；

(2) 链表中只能实现顺序查找，其时间复杂度为 $O(n)$。但链表中进行插入和删除操作不需要移动元素，只需要修改指针，其时间复杂度为 $O(1)$。

本题中，线性表中常用的操作是取第 i 个元素，所以应选择随机存取结构，即顺序表；同时在顺序表中查找第 i 个元素的前驱也很方便。单链表和单循环链表既不能实现随机存取，查找第 i 个元素的前驱也不方便；双链表虽然能快速查找第 i 个元素的前驱，但不能实现随机存取。

7.【答案】 D。最常用的操作是最后一个元素之后插入一个元素和删除第一个元素，则采用尾指针的单循环链表。

8.【答案】 C。此题考查的知识点是算法时间复杂度的定义。算法的时间复杂度取决于输入问题的规模和待处理数据的初态，所以选 C。A 和 B 都不全面。

9.【答案】 B。链表的特点包括：事先不需要申请存储空间，插入和删除运算方便，但不能实现随机存取。

10.【答案】 A。对于 A，删除指定元素，在顺序表中需要移动较多元素，而在单链表上执行同样的操作不需要移动元素，因此单链表的效率要高一些。

对于 B，在最后一个元素的后面插入一个新元素不需要移动元素，顺序表的效率和单链表相同。

对于 C，顺序输出前 k 个元素，单链表和顺序表的效率几乎相同。

对于 D，交换第 i 个元素和第 $2n-i-1$ 个元素的值($i=0,1,\cdots,n-1$)，由于顺序表可以实现随机查找，因此顺序表的效率会更高一些。

11.【答案】 C。h->next=q；表示将当前结点作为头结点后的第一元素结点插入。

12.【答案】 C。此题考查的知识点是线性表基本操作的时间复杂度。顺序存储的线性表插入元素时需要从插入位置开始向后移动元素，腾出位置以便插入，平均移动次数为 $(n+1)/2$，所以复杂度为 $O(n)$，选 C。

13.【答案】 C。此题考查的知识点是线性表基本操作的时间复杂度。链式存储的线性表访问第 i 个位置的元素时需要从头开始向后查找，平均查找次数为 $(n+1)/2$，所以时间复杂度为 $O(n)$，选 C。

14.【答案】 A。此题考查的知识点是循环单链表的存储定义。非空的循环单链表的尾结点的指针指

向头结点，所以选 A。B、C、D 均不能构成非空的循环单链表。

15.【答案】 A。此题考查的知识点是双向链表的插入操作。在 p 前插入，要修改 p 的 prior 指针、p 的 prior 所指结点的 next 指针，所以选 A。B、C、D 都将使地址丢失，连接失败。

16.【答案】 C。静态链表中指针表示的是下一元素的数组下标。

17.【答案】 B。此题考查的知识点是单链表的插入操作。要先保存 p 的后继结点，再连入 s 结点，所以选 B。A、C、D 都将使地址丢失，连接失败。

18.【答案】 B。此题考查的知识点是带头结点的单链表操作。带头结点的单链表空的时候表示只有一个结点存在，但没有存信息。所以选 B。A 表示没有结点，C 表示循环单链表，D 表示有一个指针不为空，所以都不对。

19.【答案】 D。此题考查的知识点是对数据结构和存储结构的理解。A、B、C 描述的均为物理结构即数据的存储结构，D 是逻辑结构，所以选 D。

20.【答案】 D。此题考查的知识点是线性结构的定义。线性结构的定义可简单地理解为元素只有一个前导、一个后继，而 A、B、C 有多个后继，均错，所以选 D。

二、综合应用题

1.【参考答案】

(1) 算法的基本设计思想：分别从 A、B 的头结点开始，依次比较 A、B 中元素的内容，如果 A 中的元素值大于 B 中的元素值，则将 B 中的结点插入结果链表，反之将 A 中的结点插入结果链表。由于题目中要求将结果链表中的结点按元素值的大小依次递增地排列。因此，如果 A、B 中两个元素值相同，只将其中的一个加入结果链表。

(2) 算法的设计如下：

```
typedef struct LNode{
    int data;
    struct LNode  * next;
} * Linkedlist;

LinkedList Union( LinkedList la, lb) {
    pa = la -> next;
    pb = lb -> next;                    //设工作指针 pa 和 pb
    pc = la;                            //pc 为结果链表当前结点的前驱指针
    while( pa&&pb) {
        if ( pa->data<pb ->data) {
            pc->next=pa;
            pc=pa;
            pa=pa->next;
        }
        else if ( pa->data>pb ->data) {
                pc->next=pb;
                pc=pb;
                pb=pb->next;
            }
            else{                       //处理 pa->data = pb ->data.
                pc->next=pa;
                pc=pa;
                pa=pa->next;
```

```
            u=pb;
            pb=pb->next;
            free(u);
        }
    }
    if(pa) pc->next=pa;                    //若 la 表未空,则链入结果表
    else pc->next=pb;                      //若 lb 表未空,则链入结果表
    free(lb);                              //释放 lb 头结点
    return(la);
}
```

(3) 本题中的主要操作是依次比较 A、B 链表中的数据元素值的大小,因此时间复杂度为 $O(n)$。

2.【参考答案】

(1) 顺序存储的线性表递增有序,可以顺序查找,也可折半查找。题目要求"用最少的时间在表中查找数值为 x 的元素",这里应使用折半查找方法。

(2) 算法的设计如下:

```
void SearchExchangeInsert(ElemType a[],ElemType x){
    int low=0; int high=n-1; int mid;          //low 和 high 指向线性表下界和上界的下标
    while(low<= high){
        mid = ( low + high) /2 ;               //找中间位置
        if(a[mid] == x) break;                 //找到 x,退出 while 循环
        else if (a[mid]<x) low = mid + 1;      //到中点 mid 的右部去查
            else high = mid - 1;               //到中点 mid 的左部去查
    }
    if(a[mid] == x && mid!= n){                //若最后一个元素与 x 相等,
                                               //则不存在与其后继交换的操作
        t = a[mid] ;
        a[mid] = a[mid+1];
        a[mid+1]= t;
    }                                          //数值 x 与其后继元素位置交换
    if(low> high){                             //查找失败,插入数据元素 x
        int i;
        for(i=n-1;i>high;i--)  a[i+1]=a[i];    //后移元素
        a[low] = x;                            //插入 x
    }                                          //结束插入
}
```

(3) 在利用折半查找的方法查找 x 的过程中时间复杂度为 $O(n\log_2 n)$;交换元素位置时的时间复杂度为 $O(1)$;当查找不成功时,插入元素时的时间复杂度为 $O(n)$。

3.【参考答案】

(1) 基本的设计思想:先将偶数号元素复制到一个辅助空间,然后整理数组剩下的奇数号元素,最后将辅助空间中的元素复制到数组的后半部分,但这种思路的空间复杂度为 $O(n)$。

另一种思路:

① 在数组尾部从后往前找到第一个奇数号元素,将此元素与其前面的偶数号元素交换。这样,就形成了两个前后相连且相对顺序不变的奇数号元素"块"。

② 暂存①中"块"前面的偶数号元素,将"块"内奇数号结点依次前移,然后将暂存的偶数号结点复制到

空出来的数组单元中。就形成了三个连续的奇数号元素“块”。

③ 暂存②中“块”前面的偶数号元素，将“块”内奇数号结点依次前移，然后将暂存的偶数号结点复制到空出来的数组单元中。就形成了四个连续的奇数号元素“块”。

④ 如此继续，直到前一步的“块”前没有元素为止。

(2) 算法的设计如下：

```
void Swap(ElemType A[ ], int n){
    int i=n, v=1;                  //i 为工作指针,初始假设 n 为奇数,v 为“块”的大小
    ElemType temp;                 //辅助变量
    if(n%2==0) i=n-1;              //若 n 为偶数,则令 i 为 n-1
    while(i>1){                    //假设数组从 1 开始存放。当 i=1 时,气泡浮出水面
        temp=A[i-1];               //将“块”前的偶数号元素暂存
        for(int j=0; j<v; j++)     //将大小为 v 的“块”整体向前平移
            A[i-1+j]=A[i+j]        //从前往后依次向前平移
        A[i+v-1]=temp;             //暂存的奇数号元素复制到平移后空出的位置
        i--; v++;                  //指针向前,块大小增 1
    }//while
}
```

(3) 一共进行了 $n/2$ 次交换，每次交换的元素个数从 $1 \sim n/2$，因此时间复杂度为 $O(n^2)$。虽然时间复杂度为 $O(n^2)$，但因 n^2 前的系数很小，实际达到的效率是很高的。算法的空间复杂度为 $O(1)$。

4.【参考答案】

```
struct node{
    Datatype  data;
    struct node * next;
}ListNode;
typedef ListNode  * LinkList;

void DeleteList(LinkList L,DataType min,DataType max){
    ListNode  * p,  * q,  * h;
    p=L->next;                              //采用代表头结点的单链表
    while(p && p->data < = min){            //找比 min 大的前一个元素位置
        q=p;
        p=p->next;
    }
    p=q;                                    //保存这个元素位置
    while(q && q->data <max) q=q->next;     //找比 max 小的最后一个元素位置
    while(p->next!=q){
        h=p->next;
        p=p->next;
        free(h);                            //释放空间
    }
    p->next = q;                            //把断点链上
}
```

提示：首先想到的是拿链表中的元素一个个地与 *max* 和 *min* 比较，然后删除这个结点。其实因为已知其是有序链表，所以只要找到大于 *min* 的结点的直接前趋结点，再找到小于 *max* 的结点，然后一并把中间的

全部摘掉就可以了。

5.【参考答案】

(1) 顺序存储的线性表递增有序,可以顺序查找,也可折半查找。题目要求"用最少的时间在表中查找数值为 x 的元素",这里应使用折半查找方法。

```
void SearchExchangeInsert(ElemType a[];ElemType x)
//a 是具有 n 个元素的递增有序线性表,顺序存储。本算法在表中查找数值为 x 的
//元素,如查到则与其后继交换位置;如查不到,则插入表中,且使表仍递增有序
{
    low=0;
    high=n-1;                          //low 和 high 指向线性表下界和上界的下标
    while(low<=high)
    {
        mid=(low+high)/2;              //找中间位置
        if(a[mid]==x) break;           //找到 x,退出 while 循环
        else if (a[mid] <x) low=mid+1; //到中点 mid 的右半去查
        else high=mid-1;               //到中点 mid 的左半去查
    }
    if(a[mid]==x && mid!  =n)  //若最后一个元素与 x 相等,则不存在与其后继交换的操作
    {
        t=a[mid];
        a[mid]=a[mid+1];
        a[mid+1]=t;
    }                                  //数值 x 与其后继元素位置交换
    if(low>high)                       //查找失败,插入数据元素 x
    {
        for(i=n-1;i>high;i--)
            a[i+1]=a[i];               //后移元素
        a[i+1]=x;                      //插入 x
    }   //结束插入
}   //结束本算法
```

(2) 算法讨论

首先是线性表的描述。算法中使用一维数组 a 表示线性表,未使用包含数据元素的一维数组和指示线性表长度的结构体。若使用结构体,对元素的引用应使用 a. elem[i]。另外,元素类型就假定是 ElemType,未指明具体类型。其次,C 中一维数组下标从 0 开始,若说有 n 个元素的一维数组,其最后一个元素的下标应是 $n-1$。最后,本算法可以写成三个函数,即查找函数、交换后继函数与插入函数,写成三个函数显得逻辑清晰、易读。

6.【参考答案】

```
typedef struct DuLNode{
    ElemType data;
    int freq;
    struct DuLNode *pred, *next;
} *DList;

DList locate ( DList L, ElemType x) {          //L 是带头结点的按访问频度递减的双向链表
```

```
    DList p=L->next, q;                    //p 为 L 表的工作指针,q 为 p 的前驱,用于查找插入位置
    while(p && p->data !=x) p=p->next;     //查找值为 x 的结点
    if(! p){printf("不存在所查结点\n"); exit(0);}
    else{
        p->freq++;                         //令元素值为 x 的结点的 freq 域加 1
        p->next->pred=p->pred;             //将 p 结点从链表上摘下
        p->pred->next= p->next;
        q=p->pred;                         //以下查找 p 结点的插入位置
        while(q !=L && q->freq<p->freq) q=q->pred;
        p->next=q->next; q->next->pred=p; //将 p 结点插入
        p->pred=q;q->next=p;
    }
    return (p) ;      //返回值为 x 的结点的指针
}
```

提示:在算法中先查找数据 x,查找成功时结点的访问频度域增 1,最后将该结点按频度递减插入链表中。

7.【参考答案】

(1) 算法的基本设计思想:本题实质上是一个排序问题。链表上的排序采用直接插入排序比较方便,即首先假定第一个结点有序,然后,从第二个结点开始,依次插入到前面有序链表中,最终达到整个链表有序。

(2) 算法设计如下:

```
typedef struct LNode{
    int data;
    struct LNode *link;
} *linkedlist;

LinkedList LinkListSort(LinkedList list){
    Lnode *p, *q;
    p= list-> link;                    //p 是工作指针,指向待排序的当前元素
    list->link = null;                 //假定第一个元素有序,即链表中现只有一个结点
    while(p!=null){
            r = p->link;               //r 是 p 的后继
            q = list;
            if(q->data>p ->data){      //处理待排序结点 p 比第一个元素结点小的情况
                p->link=list;
                list = p;              //链表指针指向最小元素
            }
            else{                      //查找元素值最小的结点
                while(q->link==null&&q ->link->data<p -> data) q=q->link;
                p->link = q->link;     //将当前排序结点链入有序链表中
                q->link=p;
            }
            p=r;                       //p 指向下个待排序结点
    }
```