文都教育 | 备考2011年计算机专业研究生考试通用教材
文都考研命题研究中心策划

2011
全国硕士研究生入学统考
计算机学科专业基础综合
辅导讲义同步练习

2011NIANQUANGUOSHUOSHIYANJIUSHENGRUXUETONGKAOJISUANJIXUEKEZHUANYEJICHUZONGHEFUDAOJIANGYITONGBULIANXI

编著◎崔巍　卫真　白龙飞等

严格依据最新考试大纲范围及要求
无限度贴近统考真题难度
题解详尽到位 全面揭示高分策略
名师指点迷津 攻克难关轻松获胜

原子能出版社

文都教育 | 备考2011年计算机专业研究生考试通用教材
文都考研命题研究中心策划

2011
全国硕士研究生入学统考
计算机学科专业基础综合
辅导讲义同步练习

编著◎崔巍　卫真　白龙飞等

原子能出版社

图书在版编目(CIP)数据

计算机学科专业基础综合辅导讲义同步练习/崔巍等编著. —北京:原子能出版社,2009.10
(2010.3重印)

ISBN 978—7—5022—4717—1

Ⅰ.计… Ⅱ.崔… Ⅲ.电子计算机—研究生—入学考试—自学参考资料 Ⅳ.TP3

中国版本图书馆CIP数据核字(2009)第186647号

计算机学科专业基础综合辅导讲义同步练习

总 编 辑 杨树录
策　　划 文都考研命题研究中心
责任编辑 谭　俊
特约策划 宫　静
印　　刷 北京市增富印刷有限责任公司
出版发行 原子能出版社(北京市海淀区阜成路43号　100048)
经　　销 全国新华书店
开　　本 787mm×1092mm　1/16
印　　张 21　　**字　数** 350千字
版　　次 2009年10月第1版　2010年3月第2次印刷
书　　号 ISBN 978—7—5022—4717—1　　**定　　价** 35.00元

网址:http://www.aep.com.cn　　**E-mail:atomep123@126.com**
发行电话:010-68452845

前　言

为了帮助考生更好地掌握计算机学科专业基础综合的复习要点和解题技巧，编者对最新的《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试大纲》规定的考试内容和考试要求进行了深入分析，结合对近年计算机统考命题规律的准确把握及多年来对计算机专业课程的潜心研究编写此书。

《计算机学科专业基础综合辅导讲义同步练习》一书自出版之后引起全国广大考生的热烈反响，本书作为《2011年全国硕士研究生入学统考计算机学科专业基础综合辅导讲义》的配套同步练习题集，旨在加深考生对基本概念、基本知识的理解和认知，帮助考生掌握数据结构、计算机组成原理、操作系统和计算机网络课程的解题思路和技巧，在解题过程中举一反三，触类旁通，显著提高分析问题、解决问题的能力。在题目的选取方面，编者也对历年高校研究生入学考试的考题及2009年、2010年真题进行了分析，帮助考生通过实战练习进一步提高应试能力。

全书分为四个部分：第一部分数据结构，第二部分计算机组成原理，第三部分操作系统，第四部分计算机网络。每章内容包括单项选择题、综合应用题，同时提供习题答案与详细解答。

本书的亮点：

严格依据最新考纲规定，全方位逼近真题难度；与辅导讲义完全同步，实现知识梳理与解题训练的双方面强化，有利于考生在复习与练习同步进行的过程中巩固对知识点的理解和记忆。

参与本书编写的教师均为来自国家重点院校的长期从事计算机科学与技术学科相应本科生及研究生课程教学的一线教授和副教授，在相关课程中均具有十年以上的教学经历，并先后编写过多本教材和教学参考书。本书数据结构部分由崔巍老师编写，计算机组成原理部分由蒋本珊老师编写，操作系统部分由卫真老师编写，计算机网络部分由白龙飞老师编写。全书由崔巍老师统稿。

在本书的编写过程中，参考了一些相关的书籍和资料，在此向这些书的作者表示深深的谢意。由于编者水平有限，时间也比较仓促，尽管经过反复校对与修改，书中难免还存在错漏和不妥之处，敬请广大读者和专家批评指正。

衷心地希望本书能帮助考生在考试中取得理想的成绩！

编　者

2010年2月

目　录

第一部分　数据结构

第二部分　计算机组成原理

第三部分　操作系统

第四部分　计算机网络

第一部分

数据结构

第一章 线性表

一、单项选择题

1. 对于数据结构下列结论不正确的是(　　)。
   A. 相同的逻辑结构,对应的存储结构也必相同
   B. 数据结构由逻辑结构、存储结构和基本操作3个方面组成
   C. 数据存储结构就是数据逻辑结构的机内的实现
   D. 对数据基本操作的实现与存储结构有关
2. 下面算法的时间复杂度是(　　)。

```
int y=0;
while(n>=(y+1)*(y+1)){
  y++;
}
```

   A. $O(n)$　　B. $O(n^2)$　　C. $O(1)$　　D. $O(\sqrt{n})$
3. 若长度为 n 的线性表采用顺序存储结构,在表的第 i 个位置插入一个数据元素,i 的合法值应该是(　　)。
   A. $i>0$　　B. $i\leqslant n$　　C. $1\leqslant i\leqslant n$　　D. $1\leqslant i\leqslant n+1$
4. 若长度为 n 的非空线性表采用顺序存储结构,删除表的第 i 个数据元素,i 的合法值应该是(　　)。
   A. $i>0$　　B. $i\leqslant n$　　C. $1\leqslant i\leqslant n$　　D. $1\leqslant i\leqslant n+1$
5. 若长度为 n 的非空线性表采用顺序存储结构,删除表的第 i 个数据元素,需要移动表中(　　)个数据元素。
   A. $n-i$　　B. $n+i$　　C. $n-i+1$　　D. $n-i-1$
6. 若长度为 n 的线性表采用顺序存储结构,在表的第 i 个位置插入一个数据元素,需要移动表中(　　)个数据元素。
   A. i　　B. $n+i$　　C. $n-i+1$　　D. $n-i-1$
7. 采用顺序存储结构的线性表的插入算法中,当 n 个空间已满时,可申请再增加分配 m 个空间。若申请失败,则说明系统没有(　　)可分配的存储空间。
   A. m 个　　B. m 个连续的　　C. $n+m$ 个　　D. $n+m$ 个连续的
8. 线性链表中各链结点之间的地址(　　)。
   A. 必须连续　　B. 一定不连续
   C. 部分地址必须连续　　D. 连续与否无所谓
9. 给定 n 个数据元素,逐个输入这些元素,建立一个有序单链表的时间复杂度是(　　)。
   A. $O(1)$　　B. $O(n)$　　C. $O(n^2)$　　D. $O(n\log n)$
10. 线性表$(a_1,a_2,\cdots,a_n)$以链接方式存储时,访问第 i 个位置元素的时间复杂度为(　　)。
    A. $O(i)$　　B. $O(1)$　　C. $O(n)$　　D. $O(i-1)$
11. 将长度为 n 的单链表链接在长度为 m 的单链表之后的算法的时间复杂度为(　　)。
    A. $O(1)$　　B. $O(n)$　　C. $O(m)$　　D. $O(m+n)$

12. 将两个各有 $n1$ 和 $n2$ 个元素的有序表(递增)归并成一个有序表,仍保持其递增顺序,则最少的比较次数是(　　)。

A. $n1$　　B. $n2$　　C. $n1+n2-1$　　D. $\min(n1,n2)$

13. 已知 L 是带头结点的单链表,结点 p 既不是首结点(第一个结点),也不是尾结点,删除 p 结点的直接后继结点的语句序列是(　　)。

A. p = p- > next;　　B. p- > next = p;

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

14. 设双向链表中结点的结构为(prior,data,next),在双向链表中删除指针 p 所指的结点时需要修改指针(　　)。

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

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

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

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

15. 设双向循环链表中结点的结构为(prior,data,next),且不带表头结点。若想在指针 p 所指结点之后插入指针 s 所指结点,则应执行下列哪一个操作(　　)。

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

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

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

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

16. 下面关于线性表的叙述中,错误的是哪一个(　　)。

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

17. 以下关于链式存储结构的叙述中,(　　)是不正确的。

A. 结点除自身信息外还包括指针域,因此存储密度小于顺序存储结构

B. 逻辑上相邻的结点物理上不必相邻

C. 可以通过计算直接确定第 i 个结点的存储地址

D. 插入、删除运算操作方便,不必移动结点

18. 若某线性表最常用的操作是存取任一指定序号的元素和在最后进行插入和删除运算,则利用(　　)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

19. 某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

20. 设线性表非空,采用下列(　　)所描述的链表可以在 O(1)时间内在表尾插入一个新结点。

A. 带表头结点的单链表,一个链表指针指向表头结点

B. 带表头结点的单循环链表,一个链表指针指向表头结点

C. 不带表头结点的单链表,一个链表指针指向表的第一个结点

D. 不带表头结点的单循环链表,一个链表指针指向表的第一个结点

21. 线性表的静态链表存储结构与顺序存储结构相比优点是(　　)。

A. 所有的操作算法实现简单　　B. 便于随机存储

C. 便于插入和删除　　D. 便于利用零散的存储空间

22. 下面说法正确的是(　　)。

A. 集合与线性表的区别在于是否按关键字排序

B. 对于任何数据结构,链式存储结构一定优于顺序存储结构

C. 链表中的头结点仅起到标识作用

D. 在单链表中设置头结点的作用主要是使插入和删除等操作统一,在第一个元素之前插元素和删除第一个节点不必另作判断。另外,不论链表是否为空,链表指针不变

二、综合应用题

1. 线性表$(a_1,a_2,a_3,\cdots,a_n)$采用顺序存储,每个元素都是整数,试设计算法用最少时间把所有值为负数的元素移到全部正数值元素前边的算法。例:$(x,-x,-x,x,x,-x\cdots x)$变为$(-x,-x,-x\cdots x,x,x)$。

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,要求设计算法完成下述功能:

(1)用最少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。

3. 已知一个带有表头结点的单链表,假设该链表只给出了头指针 L,在不改变链表的前提下,请设计一个尽可能高效的算法,查找链表中倒数第 k 个位置上的结点(k 为正整数),若查找成功,算法输出该结点的 data 域的值,并返回 1;否则,只返回 0。要求:

(1)描述算法的基本设计思想;

(2)描述算法的详细实现步骤;

(3)根据设计思想和实现步骤,采用程序设计语言描述算法,关键之处请给出简要注释。

4. 假设一个单循环链表,其结点含有三个域(pre,data,next)。其中 data 为数据域;pre 为指针域,它的值为空指针;next 为指针域,它指向后继结点。请设计算法,将此表改成双向循环链表。

5. 试编写在带头结点的单链表中删除(一个)最小值结点的(高效)算法。函数原型如下:

void delete(LinkList &L);

6. 设计一个算法,将带有头结点的非空单链表中数据域值最小的那个结点移到链表的最前面。要求:不得额外申请新的链结点。函数原型如下:

void delinsert(LinkList &L);

7. 编写一个算法来交换单链表中指针 p 所指结点与其后继结点,head 是该链表的头指针,p 指向该链表中某一结点。

8. 假设有两个按元素值递增次序排列的线性表,均以单链表形式存储。请编写算法将这两个单链表归并为一个按元素值非递增次序排列的单链表,并要求利用原来两个单链表的结点存放归并后的单链表。

9. 编写一个算法,设有两个无头结点的单链表,头指针分别为 ha、hb,两个链表的数据都按递增序存放。要求将 hb 表归到 ha 表中,且归并后 ha 仍递增序,在归并中对于 ha 表中已有的数据若 hb 中也有,则 hb 中的这部分数据不归并到 ha 中,hb 的链表在算法中不允许破坏。

10. 已知 L1、L2 分别为两个单循环链表的头结点指针，m，n 分别为 L1、L2 表中数据结点个数。要求设计一个算法，用最快速度将两表合并成一个带头结点的单循环链表。
11. 已知不带头结点的线性链表 list，链表中结点构造为(data，next)，其中 data 为数据域，next 为指针域。请写一算法，将该链表按结点数据域的值的大小从小到大重新链接。要求链接过程中不得使用除该链表以外的任何链结点空间。
12. 已知线性链表第一个链结点指针为 list，请写一算法，将该链表分解为两个带有头结点的循环链表，并将两个循环链表的长度分别存放在各自头结点的数据域中。其中，线性表中序号为奇数的元素分解到第一个循环链表中，序号为偶数的元素分解到第二个循环链表中。(要求用最少的时间和最少的空间)
13. 设键盘输入 n 个英语单词，输入格式为 $n,w_1,w_2,\cdots,w_n$，其中 n 表示随后输入英语单词个数，试编一程序，建立一个单链表，实现：
    (1)如果单词重复出现，则只在链表上保留一个；
    (2)除满足(1)的要求外，链表结点还应有一个频度域，记录该单词重复出现的次数，然后输出出现次数最多的前 $k(k\leqslant n)$ 个单词。

一、单项选择题

1. 【答案】A
   选项 A 错误的原因是相同的逻辑结构可以由不同的存储结构来实现，例如线性表可以用顺序存储结构和链式存储结构来实现。
2. 【答案】D
   本题程序中每次循环将 y 的值增加 1，然后比较 n 与 $(y+1)^2$ 大小，所以总共要进行 $\sqrt{n}$ 次比较。
3. 【答案】D
   在线性表的第 1 至第 n 个位置插入一个数据元素显然是允许的，紧挨在线性表最后那个数据元素后面插入一个数据元素也是允许的，因此，i 的合法值应该是 $1\leqslant i\leqslant n+1$。
4. 【答案】C
   只有删除非空线性表的第 1 至第 n 个数据元素才是可能的，因此，i 的合法值应该是 $1\leqslant i\leqslant n$。
5. 【答案】A
   删除线性表的第 i 个数据元素，需要将线性表的第 $i+1$ 个数据元素至第 n 个数据元素依次前移 1 个位置，一共需要移动 $n-i$ 个数据元素。
6. 【答案】C
   在线性表的第 i 个位置插入一个新的数据元素之前，需要先将线性表的第 i 个数据元素至第 n 个数据元素依次后移 1 个位置，一共需要移动 $n-i+1$ 个数据元素。
7. 【答案】D
   本题主要考查线性表的顺序存储结构的特点。
8. 【答案】D
   若线性表采用链式存储结构，各链结点之间的地址连续可以，不连续也可以(但每一个链

结点所占用的存储空间必须连续)。

9.【答案】C

对于有序单链表，最坏情况下需要遍历整个链表才能找到当前新结点需要插入的位置。

10.【答案】C

链接方式存储时访问某个元素需要进行遍历，时间复杂度为O(n)。

11.【答案】C

由于将长度为n的单链表链接在长度为m的单链表之后的操作，需要把长度为m的单链表遍历一遍，找到最后的一个结点，所以时间复杂度为O(m)。

12.【答案】D

比较次数最少的情况是一个表中的元素完全大于另外一个表中的元素，这时需要比较的次数是min($n1$，$n2$)。

13.【答案】C

选项A是删除了当前p结点；选项B是把p结点之后的所有结点都丢失了，同时在p结点本身形成了一个环；选项C正确；选项D是把p和p的后继结点都删除了。

14.【答案】A

本题主要考查如何在双向链表中删除一个结点，与单链表上的插入和删除操作不同的是，在双向链表中插入和删除必须同时修改两个方向上的指针。

15.【答案】D

本题主要考查如何在双向链表中插入一个结点。根据双向链表的结构特点，选项D所提供的操作顺序是正确的，其关键之一是操作p->next->prior=s；要出现在p->next=s；之后。选项A、B、C均造成s的prior指向其自身。

16.【答案】B

线性表采用顺序存储，便于进行查找操作，不便于进行插入和删除操作。

【归纳总结】关于顺序表和链表的比较，请看表1-1-1：

表1-1-1 顺序表和链表的比较

具体要求	顺序表	链表
基于空间	适于线性表长度变化不大,易于事先确定其大小时采用。	适于当线性表长度变化大,难以估计其存储规模时采用。
基于时间	顺序表是一种随机存取的存储结构,当线性表的操作主要是查找时,宜采用顺序表。	链表是一种顺序存取的存储结构。链表中对任何位置进行插入和删除都只需修改指针,所以这类操作为主的线性表宜采用链表做存储结构。若插入和删除主要发生在表的首尾两端,则宜采用尾指针表示的单循环链表。

17.【答案】C

选项C错误的原因是链式存储结构的地址不一定是连续的，所以不能通过计算直接确定第i个结点的存储地址。

18.【答案】A

只有顺序表才能存取任一指定序号的元素，其他存储方式都需要遍历才能到达相应元素。顺序表同样可以在末尾进行插入和删除元素。

19.【答案】D

本题显然应在选项B和选项D中选择正确答案，考虑到需要在最后一个元素之后插入和删除第一个元素，所以最好可以直接得到链表尾指针。如果只有头指针，必须遍历所有

链表才能得到尾指针。

20.【答案】B

本题可以在O(1)时间内插入的方法是，在表头结点后面插入一个新表头，然后把原来表头改成新的结点。

21.【答案】C

本题主要考查线性表的静态链表的特点。

22.【答案】D

选项A错误是因为集合中元素无逻辑关系；选项B中链式存储结构与顺序存储结构各有优缺点，应根据实际情况选用，不能笼统说哪一个好；选项C中头结点并不仅起标识作用，并且使操作统一，另外，头结点数据域可写入链表长度或作监视哨。

二、综合应用题

1.【分析】题目中要求重排以顺序存储结构存储的线性表的 n 个元素，使得所有值为负数的元素移到正数元素的前面。这可采用快速排序的思想来实现，只是提出暂存的第一个元素(枢轴)并不作为以后的比较标准，比较的标准是元素是否为负数。

【算法如下】

```
int Rearrange(int a[], int n) {
/*a是具有n个元素的线性表，以顺序存储结构存储，线性表的元素是整数。本算法重排线性表a，使所有值为负数的元素移到所有值为正数的数的前面*/
  i=0;
  j=n-1;          //i，j为工作指针(下标)，初始指向线性表a的第1个和第n个元素
  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) {                              //负数前移
      a[i]=a[j];
      i++;
    }
    while(i<j && a[i]<0) i++;              //当前元素为负数时指针后移
    if(i<j) {                              //正数后移
      a[j]=a[i];
      j--;
    }
  }
  a[i]=t;                                  //将原第一元素放到最终位置
}                                          //算法结束
```

本算法时间复杂度为O(n)。

2.【分析】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。

【算法如下】

```
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 = 1;
  high = n;                                  //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 的右半部去查
         else high = mid-1;                  //到中点 mid 的左半部去查
  }
  if(a[mid] = = x && mid! = n){              //若最后一个元素与 x 相等，则不与其后继交换
    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; i > high; i - -) a[i + 1] = a[i]; //后移元素
    a[i + 1] = x;                            //插入 x
  }                                          //结束插入
}                                            //算法结束
```

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]。另外元素类型假定是 ElemType，未指明具体类型。

本算法也可以写成三个函数，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。

3. **【分析】** 算法思路：定义两个指针变量 p 和 q，初始时均指向头结点的下一个结点。p 指针沿链表移动；当 p 指针移动到第 k 个结点时，q 指针开始与 p 指针同步移动；当 p 指针移动到链表最后一个结点时，q 指针所指元素为倒数第 k 个结点。以上过程对链表仅进行一遍扫描。

算法的详细实现步骤：

①count = 0，p 和 q 指向链表表头结点的下一个结点；

②若 p 为空，转⑤；

③若 count 等于 k，则 q 指向下一个结点；否则，count = count + 1；

④p 指向下一个结点，转步骤②；

⑤若 count 等于 k，则查找成功，输出该结点的 data 域的值，返回 1；否则，查找失败，返回 0；

⑥算法结束。

【算法如下】

```
typedef struct LNode {
    int     data;                            // 数据域
    struct LNode   * next;                   // 指针域
} * LinkList;
int Get_ LinkList1 (LinkList L, int k) {
```

```
LinkList p, q;
int count = 0;                          //计数器赋初值
p = q = L - > next;                     //p 和 q 指向链表表头结点的下一个结点
while (p! = NULL) {
  if (count < k) count + + ;            //计数器 + 1
  else q = q - > next;                  //q 移到下一个结点
  p = p - > next;                       //p 移到下一个结点
}
if (count < k) return 0;                //如果链表的长度小于 k，查找失败
else {                                  // 查找成功
  printf ("%d", p - > data);
  return 1;
}
}                                       //算法结束
```

4. 【分析】将具有两个链域的单循环链表，改造成双向循环链表，关键是控制给每个结点均置上指向前驱的指针，而且每个结点的前驱指针置且仅置一次。

【算法如下】

```
void  ToDouble(LinkList &la){
/*la 是结点含有 pre，data，next 三个域的单循环链表。其中 data 为数据域，pre 为空指针域，next 是指向后继的指针域。本算法将其改造成双向循环链表。*/
  while(la- > next- > pre = = null){
    la- > next- > pre = la;    //将结点 la 后继的 pre 指针指向 la
    la = la- > next;           //la 指针后移
  }
}                              //算法结束
```

算法中没有设置变量记住单循环链表的起始结点，至少省去了一个指针变量。当算法结束时，la 恢复到指向刚开始操作的结点，这是本算法的优点所在。

5. 【分析】本题要求在单链表中删除最小值结点。单链表中删除结点，为使结点删除后不出现“断链”，应知道被删结点的前驱。而“最小值结点”是在遍历整个链表后才能知道。所以算法应首先遍历链表，求得最小值结点及其前驱。遍历结束后再执行删除操作。

【算法如下】

```
void delete(LinkList &L){              //L 是带头结点的单链表，本算法删除其最小值结点
  p = L- > next;                       //p 为工作指针，指向待处理的结点，假定链表非空
  pre = L;                             //pre 指向最小值结点的前驱
  q = p;                               //q 指向最小值结点，初始假定第一元素结点是最小值结点
  while(p- > next! = null){
    if(p- > next- > data < q- > data){  //查最小值结点
      pre = p;
      q = p- > next;
    }
    p = p- > next;                     //指针后移
```

```
    }
    pre->next = q->next;                    //从链表上删除最小值结点
    free q;                                 //释放最小值结点空间
}                                           //算法结束
```

6.【分析】本题要求将链表中数据域值最小的结点移到链表的最前面。首先要查找最小值结点，将其移到链表最前面，实质上是将该结点从链表上摘下(不是删除并回收空间)，在插入到链表的最前面。

【算法如下】

```
void delinsert(LinkList &L) {
  p = L->next;                              //p 是链表的工作指针
  pre = L;                                  //pre 指向链表中数据域最小值结点的前驱
  q = p;                                    //q 指向数据域最小值结点，初始假定是第一结点
  while(p->next! = NULL) {
    if(p->next->data < q->data) {           //找到新的最小值结点
      pre = p;
      q = p->next;
    }
    p = p->next;
  }
  if(q! = L->next) {                        //若最小值是第一元素结点，则不需再操作
    pre->next = q->next;                    //将最小值结点从链表上摘下
    q->next = L->next;                      //将 q 结点插到链表最前面
    L->next = q;
  }
}                                           //算法结束
```

7.【分析】单链表中查找任何结点，都必须从头指针开始。本题要求将指针 p 所指结点与其后继结点交换，这不仅要求知道 p 结点，还应知道 p 的前驱结点。这样才能在 p 与其后继结点交换后，由原 p 结点的前驱来指向原 p 结点的后继结点。

【算法如下】

```
void Exchange(LinkList &head, LNode *p){
/*head 是单链表头结点的指针，p 是链表中的一个结点。本算法将 p 所指结点与其后继结点交换*/
  q = head->next;                           //q 是工作指针，指向链表中当前待处理结点
  pre = head;                               //pre 是前驱结点指针，指向 q 的前驱
  while(q! = null && q! = p){
    pre = q;
    q = q->next;                            //未找到 p 结点，后移指针
  }
  if(p->next = =null)printf("p 无后继结点\n"); //p 是链表中最后一个结点，无后继
  else {                                    //处理 p 和后继结点交换
    q = p->next;                            //暂存 p 的后继
    pre->next = q;                          //p 前驱结点的后继指向 p 的后继
```