

2008 年清华大学计算机考研

数据结构练习题与详细答案

说明：这份练习题目是从 60 多所院校历年考研试卷中精选出 1800 道真题，附详细参考答案

目 录

第 1 章	绪论	2.....12
第 2 章	线性表	13.....68
第 3 章	栈和队列	69.....101
第 4 章	串	102.....116
第 5 章	数组和广义表	117.....156
第 6 章	树和二叉树	157.....243
第 7 章	图	244.....270
第 8 章	动态存储管理	271.....275
第 9 章	集合	276.....301
第 10 章	排序	302.....356
第 11 章	文件	357.....362

第 1 章 绪论

一、选择题

1. 算法的计算量的大小称为计算的 ()。【北京邮电大学 2000 二、3 (20/8 分)】
A. 效率 B. 复杂性 C. 现实性 D. 难度
2. 算法的时间复杂度取决于 () 【中科院计算所 1998 二、1 (2 分)】
A. 问题的规模 B. 待处理数据的初态 C. A 和 B
3. 计算机算法指的是 (1)，它必须具备 (2) 这三个特性。
(1) A. 计算方法 B. 排序方法 C. 解决问题的步骤序列 D. 调度方法
(2) A. 可执行性、可移植性、可扩充性 B. 可执行性、确定性、有穷性
C. 确定性、有穷性、稳定性 D. 易读性、稳定性、安全性
【南京理工大学 1999 一、1 (2 分)】 【武汉交通科技大学 1996 一、1 (4 分)】
4. 一个算法应该是 ()。【中山大学 1998 二、1 (2 分)】
A. 程序 B. 问题求解步骤的描述 C. 要满足五个基本特性 D. A 和 C.
5. 下面关于算法说法错误的是 () 【南京理工大学 2000 一、1 (1.5 分)】
A. 算法最终必须由计算机程序实现
B. 为解决某问题的算法同为该问题编写的程序含义是相同的
C. 算法的可行性是指指令不能有二义性 D. 以上几个都是错误的
6. 下面说法错误的是 () 【南京理工大学 2000 一、2 (1.5 分)】
(1) 算法原地工作的含义是指不需要任何额外的辅助空间
(2) 在相同的规模 n 下，复杂度 $O(n)$ 的算法在时间上总是优于复杂度 $O(2^n)$ 的算法
(3) 所谓时间复杂度是指最坏情况下，估算算法执行时间的一个上界
(4) 同一个算法，实现语言的级别越高，执行效率就越低
A. (1) B. (1), (2) C. (1), (4) D. (3)
7. 从逻辑上可以把数据结构分为 () 两大类。【武汉交通科技大学 1996 一、4 (2 分)】
A. 动态结构、静态结构 B. 顺序结构、链式结构
C. 线性结构、非线性结构 D. 初等结构、构造型结构
8. 以下与数据的存储结构无关的术语是 ()。【北方交通大学 2000 二、1 (2 分)】
A. 循环队列 B. 链表 C. 哈希表 D. 栈
9. 以下数据结构中，哪一个是线性结构 ()？【北方交通大学 2001 一、1 (2 分)】
A. 广义表 B. 二叉树 C. 稀疏矩阵 D. 串
10. 以下那一个术语与数据的存储结构无关？ () 【北方交通大学 2001 一、2 (2 分)】
A. 栈 B. 哈希表 C. 线索树 D. 双向链表
11. 在下面的程序段中，对 x 的赋值语句的频度为 () 【北京工商大学 2001 一、10 (3 分)】
FOR $i:=1$ TO n DO
FOR $j:=1$ TO n DO
 $x:=x+1$;
A. $O(2n)$ B. $O(n)$ C. $O(n^2)$ D. $O(\log_2^n)$
12. 程序段 FOR $i:=n-1$ DOWNTO 1 DO
FOR $j:=1$ TO i DO
 IF $A[j]>A[j+1]$
 THEN $A[j]$ 与 $A[j+1]$ 对换;
其中 n 为正整数，则最后一行的语句频度在最坏情况下是 ()

- A. $O(n)$ B. $O(n \log n)$ C. $O(n^3)$ **D. $O(n^2)$** 【南京理工大学 1998 一、1 (2 分)】
13. 以下哪个数据结构不是多型数据类型 () 【中山大学 1999 一、3 (1 分)】
A. 栈 B. 广义表 C. 有向图 **D. 字符串**
14. 以下数据结构中, () 是非线性数据结构 【中山大学 1999 一、4】
A. 树 B. 字符串 C. 队 D. 栈
15. 下列数据中, () 是非线性数据结构。【北京理工大学 2001 六、1 (2 分)】
A. 栈 B. 队列 **C. 完全二叉树** D. 堆
16. 连续存储设计时, 存储单元的地址 ()。【中山大学 1999 一、1 (1 分)】
A. **一定连续** B. 一定不连续 C. 不一定连续 D. 部分连续, 部分不连续
17. 以下属于逻辑结构的是 ()。【西安电子科技大学应用 2001 一、1】
A. 顺序表 B. 哈希表 **C. 有序表** D. 单链表

二、判断题

1. 数据元素是数据的最小单位。()
【北京邮电大学 1998 一、1 (2 分)】【青岛大学 2000 一、1 (1 分)】
【上海交通大学 1998 一、1】 【山东师范大学 2001 一、1 (2 分)】
2. 记录是数据处理的最小单位。() 【上海海运学院 1998 一、5 (1 分)】
3. 数据的逻辑结构是指数据的各数据项之间的逻辑关系; () 【北京邮电大学 2002 一、1 (1 分)】
4. 算法的优劣与算法描述语言无关, 但与所用计算机有关。()
【大连海事大学 2001 一、10 (1 分)】
5. 健壮算法不会因非法的输入数据而出现莫名其妙的状态。()
【大连海事大学 2001 一、11 (1 分)】
6. 算法可以用不同的语言描述, 如果用 C 语言或 PASCAL 语言等高级语言来描述, 则算法实际上就是程序了。() 【西安交通大学 1996 二、7 (3 分)】
7. 程序一定是算法。() 【燕山大学 1998 二、2 (2 分) 并改错】
8. 数据的物理结构是指数据在计算机内的实际存储形式。() 【山东师范大学 2001 一、2 (2 分)】
9. 数据结构的抽象操作的定义与具体实现有关。() 【华南理工大学 2002 一、1 (1 分)】
10. 在顺序存储结构中, 有时也存储数据结构中元素之间的关系。()
【华南理工大学 2002 一、2 (1 分)】
11. 顺序存储方式的优点是存储密度大, 且插入、删除运算效率高。()
【上海海运学院 1999 一、1 (1 分)】
12. 数据结构的基本操作的设置的最重要的准则是, 实现应用程序与存储结构的独立。()
【华南理工大学 2002 一、5 (1 分)】
13. 数据的逻辑结构说明数据元素之间的顺序关系, 它依赖于计算机的储存结构。()
【上海海运学院 1998 一、1 (1 分)】

三、填空

1. 数据的物理结构包括_____的表示和_____的表示。【燕山大学 1998 一、1 (2 分)】
2. 对于给定的 n 个元素, 可以构造出的逻辑结构有_____(1)____, _____(2)____, _____(3)____, _____(4)____四种。
【中科院计算所 1999 二、1 (4 分)】

3. 数据的逻辑结构是指_____。【北京邮电大学 2001 二、1 (2 分)】
4. 一个数据结构在计算机中_____称为存储结构。【华中理工大学 2000 一、1 (1 分)】
5. 抽象数据类型的定义仅取决于它的一组_(1)_, 而与_(2)_无关, 即不论其内部结构如何变化, 只要它的_(3)_不变, 都不影响其外部使用。【山东大学 2001 三、3 (2 分)】
6. 数据结构中评价算法的两个重要指标是_____【北京理工大学 2001 七、1 (2 分)】
7. 数据结构是研讨数据的_(1)_和_(2)_, 以及它们之间的相互关系, 并对与这种结构定义相应的_(3)_, 设计出相应的_(4)_. 【西安电子科技大学 1998 二、2 (3 分)】
8. 一个算法具有 5 个特性:_(1)_、_(2)_、_(3)_, 有零个或多个输入、有一个或多个输出。

【华中理工大学 2000 一、2 (5 分)】 【燕山大学 1998 一、2 (5 分)】

9. 已知如下程序段

```
FOR i:=n DOWNTO 1 DO      {语句 1}
  BEGIN
    x:=x+1;                {语句 2}
    FOR j:=n DOWNTO i DO    {语句 3}
      y:=y+1;              {语句 4}
  END;
```

语句 1 执行的频度为_(1)_; 语句 2 执行的频度为_(2)_; 语句 3 执行的频度为_(3)_; 语句 4 执行的频度为_(4)_. 【北方交通大学 1999 二、4 (5 分)】

10. 在下面的程序段中, 对 x 的赋值语句的频度为_____ (表示为 n 的函数)

```
FOR i:=1 TO n DO
  FOR j:=1 TO i DO
    FOR k:=1 TO j DO
      x:=x+delta;
```

【北京工业大学 1999 一、6 (2 分)】

11. 下面程序段中带下划线的语句的执行次数的数量级是: _____ 【合肥工业大学 1999 三、1 (2 分)】

```
i:=1; WHILE i<n DO i:=i*2;
```

12. 下面程序段中带下划线的语句的执行次数的数量级是(). 【合肥工业大学 2000 三、1 (2 分)】

```
i:=1;
WHILE i<n BEGIN FOR j:=1 TO n DO x:=x+1; i:=i*2 END;
```

13. 下面程序段中带有下划线的语句的执行次数的数量级是() 【合肥工业大学 2001 三、1 (2 分)】

```
i:=n*n WHILE i>1 DO i:=i div 2;
```

14. 计算机执行下面的语句时, 语句 s 的执行次数为 _____. 【南京理工大学 2000 二、1 (1.5 分)】

```
FOR(i=1; i<n-1; i++)
  FOR(j=n; j>=i; j--)
    s;
```

15. 下面程序段的时间复杂度为_____. ($n>1$)

```
sum=1;
for (i=0; sum<n; i++) sum+=1; 【南京理工大学 2001 二、1 (2 分)】
```

16. 设 m, n 均为自然数, m 可表示为一些不超过 n 的自然数之和, f(m, n) 为这种表示方式的数目。例 f(5, 3)=5, 有 5 种表示方式: 3+2, 3+1+1, 2+2+1, 2+1+1+1, 1+1+1+1+1。

①以下是该函数的程序段，请将未完成的部分填入，使之完整

```
int f(m,n)
    int m,n;
    { if(m==1)
        return (1);
      if(n==1){
        return (2);
      }
      if(m<n)
        {return f(m,m);
      }
      if (m==n)
        {return 1+(3);
      }
      return f(m,n-1)+f(m-n,(4));
    }
```

②执行程序， $f(6,4)=$ _____。【中科院软件所 1997 二、1 (9 分)】

17. 在有 n 个选手参加的单循环赛中，总共将进行_____场比赛。【合肥工业大学 1999 三、8 (2 分)】

四、应用题

1. 数据结构是一门研究什么内容的学科？【燕山大学 1999 二、1 (4 分)】

2. 数据元素之间的关系在计算机中有几种表示方法？各有什么特点？【燕山大学 1999 二、2 (4 分)】

3. 数据类型和抽象数据类型是如何定义的。二者有何相同和不同之处，抽象数据类型的主要特点是什么？使用抽象数据类型的主要好处是什么？【北京邮电大学 1994 一 (8 分)】

4. 回答问题（每题 2 分）【山东工业大学 1997 一 (8 分)】

(1) 在数据结构课程中，数据的逻辑结构，数据的存储结构及数据的运算之间存在着怎样的关系？

(2) 若逻辑结构相同但存储结构不同，则为不同的数据结构。这样的说法对吗？举例说明之。

(3) 在给定的逻辑结构及其存储表示上可以定义不同的运算集合，从而得到不同的数据结构。这样说法对吗？举例说明之。

(4) 评价各种不同数据结构的标准是什么？

5. 评价一个好的算法，您是从哪几方面来考虑的？

【大连海事大学 1996 二、3 (2 分)】【中山大学 1998 三、1 (5 分)】

6. 解释和比较以下各组概念【华南师范大学 2000 一 (10 分)】

(1) 抽象数据类型及数据类型 (2) 数据结构、逻辑结构、存储结构

(3) 抽象数据类型【哈尔滨工业大学 2000 一、1 (3 分)】

(4) 算法的时间复杂性【河海大学 1998 一、2 (3 分)】

(5) 算法【吉林工业大学 1999 一、1 (2 分)】

(6) 频度【吉林工业大学 1999 一、2 (2 分)】

7. 根据数据元素之间的逻辑关系，一般有哪几类基本的数据结构？

【北京科技大学 1998 一、1】【同济大学 1998】

8. 对于一个数据结构，一般包括哪三个方面的讨论？【北京科技大学 1999 一、1 (2 分)】

9. 当你为解决某一问题而选择数据结构时，应从哪些方面考虑？【西安电子北京科技大学 2000】

10. 若将数据结构定义为一个二元组 (D, R) ，说明符号 D, R 应分别表示什么？

【北京科技大学 2001 一、1 (2 分)】

11. 数据结构与数据类型有什么区别? 【哈尔滨工业大学 2001 三、1 (3 分)】

12. 数据的存储结构由哪四种基本的存储方法实现? 【山东科技大学 2001 一、1 (4 分)】

13. 若有 100 个学生, 每个学生有学号, 姓名, 平均成绩, 采用什么样的数据结构最方便, 写出这些结构?

【山东师范大学 1996 二、2 (2 分)】

14. 运算是数据结构的一个重要方面。试举一例, 说明两个数据结构的逻辑结构和存储方式完全相同, 只是对于运算的定义不同。因而两个结构具有显著不同的特性, 是两个不同的结构。

【北京大学 1998 一、1 (5 分)】

15. 在编制管理通讯录的程序时, 什么样的数据结构合适? 为什么? 【长沙铁道学院 1998 四、3 (6 分)】

16. 试举一例, 说明对相同的逻辑结构, 同一种运算在不同的存储方式下实现, 其运算效率不同。

【北京理工大学 2000 三、1 (4.5 分)】

17. 有实现同一功能的两个算法 A1 和 A2, 其中 A1 的时间复杂度为 $T_1=O(2^n)$, A2 的时间复杂度为 $T_2=O(n^2)$, 仅就时间复杂度而言, 请具体分析这两个算法哪个好。【北京航空航天大学 2000 二 (10 分)】

18. 设计一数据结构, 用来表示某一银行储户的基本信息: 账号、姓名、开户年月日、储蓄类型、存入累加数、利息、帐面总数。【浙江大学 1994 一、3 (5 分)】

19. 写出下面算法中带标号语句的频度。

```
TYPE ar=ARRAY[1..n] OF datatype;
PROCEDURE perm (a: ar; k, n: integer);
VAR x: datatype; i: integer;
BEGIN
    (1) IF k=n
        THEN BEGIN
            (2) FOR i:=1 TO n DO
                (3) write (a[i]);
                    writeln;
            END
        ELSE BEGIN
            (4) FOR i:=k TO n DO
                (5) a[i]:=a[i]+i*i;
                (6) perm (a, k+1, n);
            END;
        END;
END;
```

设 k 的初值等于 1。

【北京邮电大学 1997 二 (10 分)】

20. 分析下面程序段中循环语句的执行次数。

```
i:=0; s:=0; n:=100;
REPEAT
    i:=i+1;
    s:=s+10*i;
UNTIL NOT((i<n) AND (s<n));
```

【北京邮电大学 1998 四、1 (5 分)】

21. 下列算法对一 n 位二进制数加 1, 假如无溢出, 该算法的最坏时间复杂性是什么? 并分析它的平均时间复杂性。

```

TYPE num=ARRAY [1..n] of [0..1];
PROCEDURE Inc (VAR a: num);
VAR i: integer;
BEGIN i:=n;
      WHILE A[i]=1 DO
        BEGIN A[i]:=0; i:=i-1; END;
      END;
      A[i]:=1;
END Inc;
```

【东南大学 1998 三 (8 分) 1994 二 (15 分)】

22. 阅读下列算法, 指出算法 A 的功能和时间复杂性

```

PROCEDURE A (h,g:pointer);
(h,g 分别为单循环链表 (single linked circular list) 中两个结点指针)
PROCEDURE B(s,q:pointer);
  VAR p:pointer;
  BEGIN
    p:=s;
    WHILE p^.next<>q DO p:=p^.next;
    p^.next:=s;
  END; (of B)
BEGIN
  B(h,g); B(g,h);
END; (of A)
```

【东南大学 1999 二 (10 分)】

23. 调用下列 C 函数 $f(n)$ 或 PASACAL 函数 $f(n)$ 回答下列问题:

- (1) 试指出 $f(n)$ 值的大小, 并写出 $f(n)$ 值的推导过程;
- (2) 假定 $n=5$, 试指出 $f(5)$ 值的大小和执行 $f(5)$ 时的输出结果。

```

C 函数: int f(int n)
{ int i, j, k, sum= 0;
  for(i=1; i<n+1; i++)
    {for(j=n; j>i-1; j--)
      for(k=1; k<j+1; k++)
        sum++;
      printf("sum=%d\n", sum);
    }
  return (sum);
}
```

【华中理工大学 2000 六 (10 分)】

24. 设 n 是偶数, 试计算运行下列程序段后 m 的值并给出该程序段的时间复杂度。

```

m:=0;
FOR i:=1 TO n DO
  FOR j:=2*i TO n DO
    m:=m+1;
```

【南京邮电大学 2000 一、1】

25. 有下列运行时间函数：

$$(1) T_1(n)=1000; \quad (2) T_2(n)=n^2+1000n; \quad (3) T_3(n)=3n^3+100n^2+n+1;$$

分别写出相应的大 O 表示的运算时间。

【吉林工业大学 1999 二 (12 分)】

26. 试给出下面两个算法的运算时间。

```
(1)  for i ← 1 to n do
      x ← x+1
    END
```

```
(2)  for i ← 1 to n do
      for j ← 1 to n do
        x ← x+1
      end
    end
```

【中科院自动化研究所 1995 二、2 (6 分)】

27. 斐波那契数列 F_n 定义如下

$$F_0=0, \quad F_1=1, \quad F_n=F_{n-1}+F_{n-2}, \quad n=2,3,\dots$$

请就此斐波那契数列，回答下列问题。

(1) (7 分) 在递归计算 F_n 的时候，需要对较小的 $F_{n-1}, F_{n-2}, \dots, F_1, F_0$ 精确计算多少次？

(2) (5 分) 如果用大 O 表示法，试给出递归计算 F_n 时递归函数的时间复杂度录多少？

【清华大学 2000 二 (12 分)】

28. 将下列函数，按它们在 $n \rightarrow \infty$ 时的无穷大阶数，从小到大排序。

$$n, n-n^3+7n^5, n \log n, 2^{n/2}, n^3, \log n, n^{1/2}+\log n, (3/2)^n, \binom{2n}{n}, n!, n^2+\log n$$

【中科院计算所 1995 】

第1章 绪论

一、选择题

1.B	2.C	3.1C	3.2B	4.B	5.D	6.C	7.C	8.D	9.D	10.A	11.C
12.D	13.D	14.A	15.C	16.A	17.C						

二、判断题

1. ×	2. ×	3. ×	4. ×	5. √	6. ×	7. ×	8. √	9. ×	10. ×	11. ×	12. √
13. ×											

三. 填空题

1. 数据元素 数据元素间关系 2. 集合 线性结构 树形结构 图状结构或网状结构。

3. 数据的组织形式，即数据元素之间逻辑关系的总体。而逻辑关系是指数据元素之间的关联方式或称“邻接关系”。

4. 表示（又称映像）。 5. (1) 逻辑特性 (2) 在计算机内部如何表示和实现 (3) 数学特性。

6. 算法的时间复杂度和空间复杂度。 7. (1) 逻辑结构 (2) 物理结构 (3) 操作（运算） (4) 算法。

8. (1) 有穷性 (2) 确定性 (3) 可行性。

9. (1) $n+1$ (2) n (3) $n(n+3)/2$ (4) $n(n+1)/2$ 。

10. $1 + (1+2) + (1+2+3) + \dots + (1+2+\dots+n) = n(n+1)(n+2)/6$ $O(n^3)$

11. $\log_2 n$ 12. $n \log_2 n$ 13. $\log_2 n^2$ 14. $(n+3)(n-2)/2$ 15. $O(n)$

16. ① (1) 1 (2) 1 (3) $f(m, n-1)$ (4) n ② 9 17. $n(n-1)/2$

四. 应用题

1. 数据结构是一门研究在非数值计算的程序设计问题中，计算机的操作对象及对象间的关系和施加于对象的操作等的学科。

2. 四种表示方法

(1) 顺序存储方式。数据元素顺序存放，每个存储结点只含一个元素。存储位置反映数据元素间的逻辑关系。存储密度大，但有些操作（如插入、删除）效率较差。

(2) 链式存储方式。每个存储结点除包含数据元素信息外还包含一组（至少一个）指针。指针反映数据元素间的逻辑关系。这种方式不要求存储空间连续，便于动态操作（如插入、删除等），但存储空间开销大（用于指针），另外不能折半查找等。

(3) 索引存储方式。除数据元素存储在一地址连续的内存空间外，尚需建立一个索引表，索引表中索引指示存储结点的存储位置（下标）或存储区间端点（下标），兼有静态和动态特性。

(4) 散列存储方式。通过散列函数和解决冲突的方法，将关键字散列在连续的有限的地址空间内，并将散列函数的值解释成关键字所在元素的存储地址，这种存储方式称为散列存储。其特点是存取速度快，只能按关键字随机存取，不能顺序存取，也不能折半存取。

3. 数据类型是程序设计语言中的一个概念，它是一个值的集合和操作的集合。如 C 语言中的整型、实型、字符型等。整型值的范围（对具体机器都应有整数范围），其操作有加、减、乘、除、求余等。实际上数据类型是厂家提供给用户的已实现了的数据结构。“抽象数据类型（ADT）”指一个数学模型及定义在该模型上的一组操作。“抽象”的意义在于数据类型的数学抽象特性。抽象数据类型的定义仅取决于它的逻辑特性，而与其在计算机内部如何表示和实现无关。无论其内部结构如何变化，只要它的数学特性不变就不影响它的外部使用。抽象数据类型和数据类型实质上是一个概念。此外，抽象数据类型的范围更广，它已不再局限于机器已定义和实现的数据类型，还包括用户在设计软件系统时自行定义的数据类型。使用抽象数据类型定义的软件模块含定义、表示和实现三部分，封装在一起，对用户透明（提供接口），而不必了解实现细节。抽象数据类型的出现使程序设计不再是“艺术”，而是向“科学”迈进了一步。

4. (1) 数据的逻辑结构反映数据元素之间的逻辑关系（即数据元素之间的关联方式或“邻接关系”），数据的存储结构是数据结构在计算机中的表示，包括数据元素的表示及其关系的表示。

数据的运算是对于数据定义的一组操作，运算是定义在逻辑结构上的，和存储结构无关，而运算的实现则是依赖于存储结构。

(2) 逻辑结构相同但存储不同，可以是不同的数据结构。例如，线性表的逻辑结构属于线性结构，采用顺序存储结构为顺序表，而采用链式存储结构称为线性链表。

(3) 栈和队列的逻辑结构相同，其存储表示也可相同（顺序存储和链式存储），但由于其运算集合不同而成为不同的数据结构。

(4) 数据结构的评价非常复杂，可以考虑两个方面，一是所选数据结构是否准确、完整的刻画了问题的基本特征；二是是否容易实现（如对数据分解是否恰当；逻辑结构的选择是否适合于运算的功能，是否有利于运算的实现；基本运算的选择是否恰当。）

5. 评价好的算法有四个方面。一是算法的正确性；二是算法的易读性；三是算法的健壮性；四是算法的时空效率（运行）。

6. (1) 见上面题 3 (2) 见上面题 4 (3) 见上面题 3

(4) 算法的时间复杂性是算法输入规模的函数。算法的输入规模或问题的规模是作为该算法输入的数据所含数据元素的数目，或与此数目有关的其它参数。有时考虑算法在最坏情况下的时间复杂度或平均时间复杂度。

(5) 算法是对特定问题求解步骤的描述，是指令的有限序列，其中每一条指令表示一个或多个操作。算法具有五个重要特性：有穷性、确定性、可行性、输入和输出。

(6) 频度。在分析算法时间复杂度时，有时需要估算基本操作的原操作，它是执行次数最多的一个操作，该操作重复执行的次数称为频度。

7. 集合、线性结构、树形结构、图形或网状结构。 8. 逻辑结构、存储结构、操作（运算）。

9. 通常考虑算法所需要的存储空间量和算法所需要的时间量。后者又涉及到四方面：程序运行时所需输入的数据总量，对源程序进行编译所需时间，计算机执行每条指令所需时间和程序中指令重复执行的次数。

10. D 是数据元素的有限集合，S 是 D 上数据元素之间关系的有限集合。

11. “数据结构”这一术语有两种含义，一是作为一门课程的名称；二是作为一个科学的概念。作为科学概念，目前尚无公认定义，一般认为，讨论数据结构要包括三个方面，一是数据的逻辑结构，二是数据的存储结构，三是对数据进行的操作（运算）。而数据类型是值的集合和操作的集合，可以看作是已实现了的数据结构，后者是前者的一种简化情况。

12. 见上面题 2。

13. 将学号、姓名、平均成绩看成一个记录（元素，含三个数据项），将 100 个这样的记录存于数组中。因一般无增删操作，故宜采用顺序存储。

```
typedef struct
{
    int num; //学号
    char name[8]; //姓名
    float score; //平均成绩
} node;

node student[100];
```

14. 见上面题 4 (3)。

15. 应从两方面进行讨论：如通讯录较少变动（如城市私人电话号码），主要用于查询，以顺序存储较方便，既能顺序查找也可随机查找；若通讯录经常有增删操作，用链式存储结构较为合适，将每个人的情况作为一个元素（即一个结点存放一个人），设姓名作关键字，链表安排成有序表，这样可提高查询速度。

16. 线性表中的插入、删除操作，在顺序存储方式下平均移动近一半的元素，时间复杂度为

$O(n)$; 而在链式存储方式下, 插入和删除时间复杂度都是 $O(1)$ 。

17. 对算法 A1 和 A2 的时间复杂度 T1 和 T2 取对数, 得 $n\log^2$ 和 $2\log^n$ 。显然, 算法 A2 好于 A1。

18. **struct** node
 {**int** year, month, day; };
typedef struct
 {**int** num; // 帐号
char name[8]; // 姓名
struct node date; // 开户年月日
int tag; // 储蓄类型, 如: 0- 零存, 1- 一年定期……
float put; // 存入累加数;
float interest; // 利息
float total; // 帐面总数
 } count;

19. (1) n (2) $n+1$ (3) n (4) $(n+4)(n-1)/2$ (5) $(n+2)(n-1)/2$ (6) $n-1$

这是一个递归调用, 因 k 的初值为 1, 由语句 (6) 知, 每次调用 k 增 1, 故第 (1) 语句执行 n 次。(2) 是 FOR 循环语句, 在满足 (1) 的条件下执行, 该语句进入循环体 (3) n 次, 加上最后一次判断出界, 故执行了 $n+1$ 次。(4) 也是循环语句, 当 $k=1$ 时判断 $n+1$ 次 (进入循环体 (5) n 次), $k=2$ 时判断 n 次, 最后一次 $k=n-1$ 时判断 3 次, 故执行次数是 $(n+1) + n + \dots + 3 = (n+4)(n-1)/2$ 次。语句 (5) 是 (4) 的循环体, 每次比 (4) 少一次判断, 故执行次数是 $n + (n-1) + \dots + 2 = (n+2)(n-1)/2$ 次。注意分析时, 不要把 (2) 分析成 n 次, 更不是 1 次。

20. 4 (这时 $i=4$, $s=100$) REPEAT 语句先执行循环体, 后判断条件, 直到条件为真时退出循环。

21. 算法在最好情况下, 即二进制数的最后一位为零时, 只作一次判断, 未执行循环体, 赋值语句 $A[i]$ 执行了一次; 最坏情况出现在二进制数各位均为 1 (最高位为零, 因题目假设无溢出), 这时循环体执行了 $n-1$ 次, 时间复杂度是 $O(n)$, 循环体平均执行 $n/2$ 次, 时间复杂度仍是 $O(n)$ 。

22. 该算法功能是将原单循环链表分解成两个单循环链表: 其一包括结点 h 到结点 g 的前驱结点; 另一个包括结点 g 到结点 h 的前驱结点。时间复杂度是 $O(n)$ 。

23. 第一层 FOR 循环判断 $n+1$ 次, 往下执行 n 次, 第二层 FOR 执行次数为 $(n+(n-1)+(n-2)+\dots+1)$, 第三层循环体受第一层循环和第二层循环的控制, 其执行次数如下表:

$i=$	1	2	3	...	n
$j=n$	n	n	n	...	n
$j=n-1$	$n-1$	$n-1$	$n-1$...	
...		
$j=3$	3	3			
$j=2$	2	2			
$j=1$	1				

执行次数为 $(1+2+\dots+n) + (2+3+\dots+n) + \dots + n = n(n+1)/2 - n(n^2-1)/6$ 。在 $n=5$ 时, $f(5)=55$, 执行过程中, 输出结果为: $\text{sum}=15, \text{sum}=29, \text{sum}=41, \text{sum}=50, \text{sum}=55$ (每个 $\text{sum} =$ 占一行, 为节省篇幅, 这里省去换行)。

24. $O(n^2)$, m 的值等于赋值语句 $m:=m+1$ 的运行次数, 其计算式为 $\sum_{i=1}^{n/2} (n-2i+1) = \frac{n^2}{4}$

25. (1) $O(1)$ (2) $O(n^2)$ (3) $O(n^3)$

26. (1) $O(n)$ (2) $O(n^2)$

27. (1) 由斐波那契数列的定义可得:

$$\begin{aligned} F_n &= F_{n-1} + F_{n-2} \\ &= 2F_{n-2} + F_{n-3} \\ &= 3F_{n-3} + 2F_{n-4} \\ &= 5F_{n-4} + 3F_{n-5} \\ &= 8F_{n-5} + 5F_{n-6} \\ &\dots\dots \\ &= pF_1 + qF_0 \end{aligned}$$

设 F_m 的执行次数为 B_m ($m=0, 1, 2, \dots, n-1$), 由以上等式可知, F_{n-1} 被执行一次, 即 $B_{n-1}=1$; F_{n-2} 被执行两次, 即 $B_{n-2}=2$; 直至 F_1 被执行 p 次、 F_0 被执行 q 次, 即 $B_1=p$, $B_0=q$. B_m 的执行次数为前两等式第一因式系数之和, 即 $B_m=B_{m-1}+B_{m-2}$, 再有 $B_{n-1}=1$ 和 $B_{n-2}=2$, 这也是一个斐波那契数列。可以解得:

$$B_m = \frac{\sqrt{5}}{5} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{n-m+2} - \left(\frac{1-\sqrt{5}}{2} \right)^{n-m+2} \right] \quad (m=0, 1, 2, \dots, n-1)$$

(2) 时间复杂度为 $O(n)$

28. 从小到大排列为: $\log n$, $n^{1/2} + \log n$, n , $n \log n$, $n^2 + \log n$, n^3 , $n - n^3 + 7n^5$, $2^{n/2}$, $(3/2)^n$, $n!$,

$$\binom{2n}{n}$$

第2章 线性表

一 选择题

1. 下述哪一条是顺序存储结构的优点? () 【北方交通大学 2001 一、4 (2分)】
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
2. 下面关于线性表的叙述中, 错误的是哪一个? () 【北方交通大学 2001 一、14 (2分)】
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。
3. 线性表是具有 n 个 () 的有限序列 ($n > 0$)。 【清华大学 1998 一、4 (2分)】
A. 表元素 B. 字符 C. 数据元素 D. 数据项 E. 信息项
4. 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。【哈尔滨工业大学 2001 二、1 (2分)】
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
5. 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。【南开大学 2000 一、3】
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
6. 设一个链表最常用的操作是在末尾插入结点和删除尾结点, 则选用 () 最节省时间。
A. 单链表 B. 单循环链表 C. 带尾指针的单循环链表 D. 带头结点的双循环链表
【合肥工业大学 2000 一、1 (2分)】
7. 若某表最常用的操作是在最后一个结点之后插入一个结点或删除最后一个结点。则采用 () 存储方式最节省运算时间。【北京理工大学 2000 一、1 (2分)】
A. 单链表 B. 双链表 C. 单循环链表 D. 带头结点的双循环链表
8. 静态链表中指针表示的是 ()。 【北京理工大学 2001 六、2 (2分)】
A. 内存地址 B. 数组下标 C. 下一元素地址 D. 左、右孩子地址
9. 链表不具有的特点是 () 【福州大学 1998 一、8 (2分)】
A. 插入、删除不需要移动元素 B. 可随机访问任一元素
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. 下面的叙述不正确的是 () 【南京理工大学 1996 一、10 (2分)】
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
11. 线性表的表元存储方式有 (1) 和链接两种。试指出下列各表中使用的是何种存储方式: 表 1 是 (2) 存储方式; 表 2 是 (3) 存储方式; 表 3 是 (4) 存储方式; 表 4 是 (5) 存储方式。表左的 s 指向起始表元。

表元编号	货号	数量	表元间联系
1	618	40	2
2	205	2	3
3	503	25	4

5	781	17	6
6	910	24	0

表 1

s→

表元编号	货号	数量	表元间联系
1	618	40	5
2	205	2	1
3	103	15	4
4	501	20	2
5	781	17	6
6	910	24	3

表 2

s→

表元编号	货号	数量	表元间联系
1	618	40	5
2	205	2	1
3	103	15	4
4	501	20	0
5	781	17	6
6	910	24	3

表 3

s→

表元编号	货号	数量	表元间联系	
			1	2
1	618	40	5	2
2	205	2	1	0
3	103	15	4	6
4	501	20	0	3
5	781	17	6	1
6	910	24	3	5

表 4

s→

供选择的答案：

- A. 连续 B. 单向链接 C. 双向链接 D. 不连接 E. 循环链接
F. 树状 G. 网状 H. 随机 I. 顺序 J. 顺序循环

【上海海运学院 1995 二、1 (5 分)】

12. (1) 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。
(2) 静态链表中能容纳的元素个数的最大数在表定义时就确定了，以后不能增加。
(3) 静态链表与动态链表在元素的插入、删除上类似，不需做元素的移动。

以上错误的是 () 【南京理工大学 2000 一、3 (1.5 分)】

- A. (1), (2) B. (1) C. (1), (2), (3) D. (2)

13. 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 () ($1 \leq i \leq n+1$)。【北京航空航天大学 1999 一、1 (2 分)】

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

14. 对于顺序存储的线性表，访问结点和增加、删除结点的时间复杂度为 ()。

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

【青岛大学 2000 五、1 (2 分)】

15. 线性表 (a_1, a_2, \dots, a_n) 以链接方式存储时, 访问第 i 位置元素的时间复杂性为 ()
 A. $O(i)$ B. $O(1)$ C. $O(n)$ D. $O(i-1)$ 【中山大学 1999 一、2】
16. 非空的循环单链表 head 的尾结点 p 满足 ()。【武汉大学 2000 二、10】
 A. $p \uparrow .link = head$ B. $p \uparrow .link = NIL$ C. $p = NIL$ D. $p = head$
17. 循环链表 H 的尾结点 P 的特点是 ()。【中山大学 1998 二、2 (2 分)】
 A. $P \wedge .NEXT = H$ B. $P \wedge .NEXT = H \wedge .NEXT$ C. $P = H$ D. $P = H \wedge .NEXT$
18. 在一个以 h 为头的单循环链中, p 指针指向链尾的条件是 () 【南京理工大学 1998 一、15 (2 分)】
 A. $p \wedge .next = h$ B. $p \wedge .next = NIL$ C. $p \wedge .next \wedge .next = h$ D. $p \wedge .data = -1$
19. 完成在双循环链表结点 p 之后插入 s 的操作是 ()；【北方交通大学 1999 一、4 (3 分)】
 A. $p \wedge .next := s$; $s \wedge .priou := p$; $p \wedge .next \wedge .priou := s$; $s \wedge .next := p \wedge .next$;
 B. $p \wedge .next \wedge .priou := s$; $p \wedge .next := s$; $s \wedge .priou := p$; $s \wedge .next := p \wedge .next$;
 C. $s \wedge .priou := p$; $s \wedge .next := p \wedge .next$; $p \wedge .next := s$; $p \wedge .next \wedge .priou := s$;
 D. $s \wedge .priou := p$; $s \wedge .next := p \wedge .next$; $p \wedge .next \wedge .priou := s$; $p \wedge .next := s$;
20. 在双向循环链表中, 在 p 指针所指向的结点前插入一个指针 q 所指向的新结点, 其修改指针的操作是 ()。【北京邮电大学 1998 二、2 (2 分)】
 注: 双向链表的结点结构为 (llink, data, rlink)。 供选择的答案:
 A. $p \uparrow .llink := q$; $q \uparrow .rlink := p$; $p \uparrow .llink \uparrow .rlink := q$; $q \uparrow .llink := q$;
 B. $p \uparrow .llink := q$; $p \uparrow .llink \uparrow .rlink := q$; $q \uparrow .rlink := p$; $q \uparrow .llink := p \uparrow .llink$;
 C. $q \uparrow .rlink := p$; $q \uparrow .llink := p \uparrow .llink$; $p \uparrow .llink \uparrow .rlink := q$; $p \uparrow .llink := q$;
 D. $q \uparrow .llink := p \uparrow .llink$; $q \uparrow .rlink := p$; $p \uparrow .llink := q$; $p \uparrow .llink := q$; (编者按: 原题如此)
21. 在非空双向循环链表中 q 所指的结点前插入一个由 p 所指的链结点的过程依次为:
 $rlink(p) \leftarrow q$; $llink(p) \leftarrow llink(q)$; $llink(q) \leftarrow p$; ()
 A. $rlink(q) \leftarrow p$ B. $rlink(llink(q)) \leftarrow p$ C. $rlink(llink(p)) \leftarrow p$ D. $rlink(rlink(p)) \leftarrow p$
 【北京航空航天大学 2000 一、1 (2 分)】
22. 双向链表中有两个指针域, llink 和 rlink, 分别指回前驱及后继, 设 p 指向链表中的一个结点, q 指向一待插入结点, 现要求在 p 前插入 q , 则正确的插入为 () 【南京理工大学 1996 一、1 (2 分)】
 A. $p \wedge .llink := q$; $q \wedge .rlink := p$; $p \wedge .llink \wedge .rlink := q$; $q \wedge .llink := p \wedge .llink$;
 B. $q \wedge .llink := p \wedge .llink$; $p \wedge .llink \wedge .rlink := q$; $q \wedge .rlink := p$;
 $p \wedge .llink := q \wedge .rlink$;
 C. $q \wedge .rlink := p$; $p \wedge .rlink := q$; $p \wedge .llink \wedge .rlink := q$; $q \wedge .rlink := p$;
 D. $p \wedge .llink \wedge .rlink := q$; $q \wedge .rlink := p$; $q \wedge .llink := p \wedge .llink$; $p \wedge .llink := q$;
23. 在双向链表指针 p 的结点前插入一个指针 q 的结点操作是 ()。【青岛大学 2000 五、2 (2 分)】
 A. $p \rightarrow Llink = q$; $q \rightarrow Rlink = p$; $p \rightarrow Llink \rightarrow Rlink = q$; $q \rightarrow Llink = q$;
 B. $p \rightarrow Llink = q$; $p \rightarrow Llink \rightarrow Rlink = q$; $q \rightarrow Rlink = p$; $q \rightarrow Llink = p \rightarrow Llink$;
 C. $q \rightarrow Rlink = p$; $q \rightarrow Llink = p \rightarrow Llink$; $p \rightarrow Llink \rightarrow Rlink = q$; $p \rightarrow Llink = q$;
 D. $q \rightarrow Llink = p \rightarrow Llink$; $q \rightarrow Rlink = q$; $p \rightarrow Llink = q$; $p \rightarrow Llink = q$;

24. 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. $p \rightarrow next = s; s \rightarrow next = p \rightarrow next;$ B. $s \rightarrow next = p \rightarrow next; p \rightarrow next = s;$
C. $p \rightarrow next = s; p \rightarrow next = s \rightarrow next;$ D. $p \rightarrow next = s \rightarrow next; p \rightarrow next = s;$

【青岛大学 2001 五、3 (2 分)】

25. 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. $head \rightarrow next == NULL$ C. $head \rightarrow next == head$ D. $head != NULL$

【北京工商大学 2001 一、5 (3 分)】

26. 在双向链表存储结构中, 删除 p 所指的结点时须修改指针 ()。

- A. $(p \rightarrow llink) \rightarrow rlink = p \rightarrow rlink$ $(p \rightarrow rlink) \rightarrow llink = p \rightarrow llink;$
B. $p \rightarrow llink = (p \rightarrow llink) \rightarrow llink$ $(p \rightarrow llink) \rightarrow rlink = p;$
C. $(p \rightarrow rlink) \rightarrow llink = p$ $p \rightarrow rlink = (p \rightarrow rlink) \rightarrow rlink$
D. $p \rightarrow rlink = (p \rightarrow llink) \rightarrow llink$ $p \rightarrow llink = (p \rightarrow rlink) \rightarrow rlink;$

【西安电子科技大学 1998 一、1 (2 分)】

27. 双向链表中有两个指针域, llink 和 rlink 分别指向前趋及后继, 设 p 指向链表中的一个结点, 现要求删去 p 所指结点, 则正确的删除是 () (链中结点数大于 2, p 不是第一个结点)

- A. $p \rightarrow llink \rightarrow rlink = p \rightarrow llink;$ $p \rightarrow llink \rightarrow rlink = p \rightarrow rlink;$ $dispose(p);$
B. $dispose(p);$ $p \rightarrow llink \rightarrow rlink = p \rightarrow llink;$ $p \rightarrow llink \rightarrow rlink = p \rightarrow rlink;$
C. $p \rightarrow llink \rightarrow rlink = p \rightarrow llink;$ $dispose(p);$ $p \rightarrow llink \rightarrow rlink = p \rightarrow rlink;$
D. 以上 A, B, C 都不对。 【南京理工大学 1997 一、1 (2 分)】

二、判断

1. 链表中的头结点仅起到标识的作用。() 【南京航空航天大学 1997 一、1 (1 分)】

2. 顺序存储结构的主要缺点是不利于插入或删除操作。() 【南京航空航天大学 1997 一、2 (1 分)】

3. 线性表采用链表存储时, 结点和结点内部的存储空间可以是不连续的。()

【北京邮电大学 1998 一、2 (2 分)】

4. 顺序存储方式插入和删除时效率太低, 因此它不如链式存储方式好。()

【北京邮电大学 2002 一、2 (1 分)】

5. 对任何数据结构链式存储结构一定优于顺序存储结构。() 【南京航空航天大学 1997 一、3 (1 分)】

6. 顺序存储方式只能用于存储线性结构。()

【中科院软件所 1999 六、1-2 (2 分)】 【上海海运学院 1997 一、1 (1 分)】

7. 集合与线性表的区别在于是否按关键字排序。() 【大连海事大学 2001 一、5 (1 分)】

8. 所谓静态链表就是一直不发生变化的链表。() 【合肥工业大学 2000 二、1 (1 分)】

9. 线性表的特点是每个元素都有一个前驱和一个后继。() 【合肥工业大学 2001 二、1 (1 分)】

10. 取线性表的第 i 个元素的时间同 i 的大小有关。() 【南京理工大学 1997 二、9 (2 分)】

11. 循环链表不是线性表。() 【南京理工大学 1998 二、1 (2 分)】

12. 线性表只能用顺序存储结构实现。() 【青岛大学 2001 四、2 (1 分)】

13. 线性表就是顺序存储的表。() 【青岛大学 2002 一、1 (1 分)】

14. 为了很方便的插入和删除数据, 可以使用双向链表存放数据。()

【上海海运学院 1995 一、1 (1 分)】 【上海海运学院 1997 一、2 (1 分)】

15. 顺序存储方式的优点是存储密度大, 且插入、删除运算效率高。()

【上海海运学院 1996 一、1 (1 分)】 【上海海运学院 1999 一、1 (1 分)】

16. 链表是采用链式存储结构的线性表, 进行插入、删除操作时, 在链表中比在顺序存储结构中效率高。 () 【上海海运学院 1998 一、2 (1 分)】

三、填空

1. 当线性表的元素总数基本稳定，且很少进行插入和删除操作，但要求以最快的速度存取线性表中的元素时，应采用_____存储结构。【北方交通大学 2001 二、4】
2. 线性表 $L = (a_1, a_2, \dots, a_n)$ 用数组表示，假定删除表中任一元素的概率相同，则删除一个元素平均需要移动元素的个数是_____。【北方交通大学 2001 二、9】
3. 设单链表的结点结构为 $(data, next)$ ， $next$ 为指针域，已知指针 px 指向单链表中 $data$ 为 x 的结点，指针 py 指向 $data$ 为 y 的新结点，若将结点 y 插入结点 x 之后，则需要执行以下语句：_____； _____；【华中理工大学 2000 一、4 (2 分)】
4. 在一个长度为 n 的顺序表中第 i 个元素 $(1 \leq i \leq n)$ 之前插入一个元素时，需向后移动_____个元素。

【北京工商大学 2001 二、4 (4 分)】

5. 在单链表中设置头结点的作用是_____。【哈尔滨工业大学 2000 二、1 (1 分)】
6. 对于一个具有 n 个结点的单链表, 在已知的结点 $*p$ 后插入一个新结点的时间复杂度为_____, 在给定值为 x 的结点后插入一个新结点的时间复杂度为_____。【哈尔滨工业大学 2001 一、1 (2 分)】
7. 根据线性表的链式存储结构中每一个结点包含的指针个数, 将线性链表分成_____和_____；而又根据指针的连接方式, 链表又可分成_____和_____。【西安电子科技大学 1998 二、4 (3 分)】
8. 在双向循环链表中, 向 p 所指的结点之后插入指针 f 所指的结点, 其操作是_____, _____、_____, _____。【中国矿业大学 2000 一、1 (3 分)】
9. 在双向链表结构中, 若要求在 p 指针所指的结点之前插入指针为 s 所指的结点, 则需执行下列语句:

$$\hat{s}.next := p; \quad \hat{s}.prior := \text{nil}; \quad \hat{p}.prior := s; \quad \text{nil}.prior := s;$$

【福州大学 1998 二、7 (2 分)】

10. 链接存储的特点是利用_____来表示数据元素之间的逻辑关系。【中山大学 1998 一、1 (1 分)】

11. 顺序存储结构是通过_____表示元素之间的关系的;链式存储结构是通过_____表示元素之间的关系的。【北京理工大学 2001 七、2 (2 分)】

12. 对于双向链表, 在两个结点之间插入一个新结点需修改的指针共 _____ 个, 单链表为 _____ 个。

【南京理工大学 2000 二、2 (3 分)】

13. 循环单链表的最大优点是：_____。【福州大学 1998 二、3 (2 分)】

14. 已知指针 p 指向单链表 L 中的某结点, 则删除其后继结点的语句是:

【合肥工业大学 1999 三、2 (2 分)】

15. 带头结点的双循环链表 L 中只有一个元素结点的条件是:

【合肥工业大学 1999 三、3 2000 三、2 (2 分)】

16. 在单链表 L 中, 指针 p 所指结点有后继结点的条件是: ____【合肥工业大学 2001 三、3 (2 分)】

17. 带头结点的双循环链表 L 为空表的条件是: _____。

【北京理工大学 2000 二、1 (2 分)】 【青岛大学 2002 三、1 (2 分)】

18. 在单链表 p 结点之后插入 s 结点的操作是: _____。【青岛大学 2002 三、2 (2 分)】

19. 请在下列算法的横线上填入适当的语句。【清华大学 1994 五 (15 分)】

```
FUNCTION inclusion(ha,hb:linklisttp):boolean;
```

{以 ha 和 hb 为头指针的单链表分别表示有序表 A 和 B, 本算法判别表 A 是否包含在表 B 内, 若是, 则返回 “true”, 否则返回 “false” }

```
BEGIN
```

```
    pa:=ha^.next; pb:=hb^.next; (1)_____;
```

```
    WHILE (2)_____ DO
```

```
        IF pa^.data=pb^.data THEN (3)_____ ELSE (4)_____;
```

```
    (5)_____
```

```
END;
```

20. 完善算法: 已知单链表结点类型为:

```
TYPE ptr=^node;
```

```
node=RECORD
```

```
    data: integer; next: ptr
```

```
END;
```

过程 create 建立以 head 为头指针的单链表。

```
PROCEDURE create ( (1) );
```

```
VAR p,q: ptr; k: integer;
```

```
BEGIN
```

```
    new (head); q:=head; read (k);
```

```
    WHILE k>0 DO
```

```
        BEGIN
```

```
            (2); (3); (4); (5);
```

```
            read (k)
```

```
        END;
```

```
        q^.next:=NIL;
```

```
END; 【北京师范大学 1999 三】
```

21. 已给如下关于单链表的类型说明:

```
TYPE
```

```
    list=^node ;
```

```
    node=RECORD
```

```
        data: integer; next: list;
```

```
    END;
```

以下程序采用链表合并的方法, 将两个已排序的单链表合并成一个链表而不改变其排序性 (升序), 这里两链表的头指针分别为 p 和 q.

```
PROCEDURE mergelink(VAR p,q:list):
```

```
    VAR h,r: list;
```

```
    BEGIN
```

```
        (1)_____
```

```
        h^.next:= NIL; r:=h;
```

```
        WHILE ((p<>NIL) AND (q<>NIL)) DO
```

```
            IF (p^.data<=q^.data)
```

```
                THEN BEGIN (2)_____; r:=p; p:=p^.next; END
```

```
                ELSE BEGIN (3)_____; r:=q; q:=q^.next; END;
```

```
IF (p=NIL) THEN r^.next:=q;
  (4);
  p:=h^.next; dispose(h);
```

END; 【厦门大学 2000 三、2 (8 分)】

22. 假设链表 p 和链表 q 中的结点值都是整数, 且按结点值的递增次序链接起来的带表头结点的环形链表。各链表的表头结点的值为 max, 且链表中其他结点的值都小于 max, 在程序中取 max 为 9999。在各个链表中, 每个结点的值各不相同, 但链表 p 和链表 q 可能有值相同的结点 (表头结点除外)。下面的程序将链表 q 合并到链表 p 中, 使得合并后的链表是按结点值递增次序链接起来的带表头结点的环形链表, 且链表中各个结点的值各不相同。请在划线处填上适当内容, 每个框只填一个语句或一个表达式, 链表的结点类型如下

```
TYPE nodeptr=^nodetype;
  nodetype=RECORD
    data: integer; link: nodeptr;
  END;
CONST max=9999;
PROCEDURE merge (VAR p:nodeptr;q:nodeptr);
  VAR r,s: nodeptr;
  BEGIN
    r:=p;
    WHILE (A) DO
      BEGIN
        WHILE r^.link^.data<q^.link^.data DO (B);
        IF r^.link^.data>q^.link^.data
          THEN BEGIN s:=(C); (D):=s^.link; s^.link:=(E); (F):=s; (G);
        ELSE BEGIN (H); s:=q^.link; (I); dispose(s) END
      END;
      dispose(q)
    END; 【复旦大学 1997 五 (18 分)】
```

23. PROC ins__linklist(la:linkisttp; i:integer; b:elemtp);
 {la 为指向带头结点的单链表的头指针, 本算法在表中第 i 个元素之前插入元素 b}
 p:=(1); j:=(2); {指针初始化, j 为计数器}
 WHILE (p<>NIL) AND ((3)) DO [p:=(4); j:=j+1;]
 {寻找第 i-1 个结点}
 IF (p=NIL) OR ((5))
 THEN error ('No this position')
 ELSE [new(s); s↑.data:=b; s↑.next:=p↑.next; p↑.next:=s;]
 ENDP; {ins-linklist} 【燕山大学 1998 四、1 (15 分)】

24. 已知双链表中结点的类型定义为:

```
TYPE dpointer=^list;
  list=RECORD
    data:integer; left,right:dpointer;
  END;
```

如下过程将在双链表第 i 个结点 (i>=0) 之后插入一个元素为 x 的结点, 请在答案栏给出题目中

_____处应填入的语句或表达式，使之可以实现上述功能。

```

PROCEDURE insert(VAR head:dpointer;i,x:integer);
VAR s,p:dpointer; j: integer;
BEGIN
    new(s); s^.data:=x;
    IF(i=0)THEN BEGIN s^.right:=head; (1)_____ head:=s END{如果 i=0,则将 s 结点插入到
    表头后返回}
    ELSE BEGIN p:=head; (2)_____ ; {在双链表中查找第 i 个结点,由 p 所指向}
    WHILE ((p<>NIL) AND (j<i)) DO BEGIN j:=j+1; (3)_____ END;
    IF p<>NIL THEN
        IF (p^.right=NIL)
            THEN BEGIN p^.right:=s; s^.right:=NIL; (4)_____ END
            ELSE BEGIN s^.right:=p^.right; (5)_____ ;p^.right:=s; (6)_____ END
        ELSE writeln( 'can not find node!' )
    END
END;

```

END;【厦门大学 2002 二 (12 分)】

25. 阅读以下算法，填充空格，使其成为完整的算法。其功能是在一个非递减的顺序存储线性表中，删除所有值相等的多余元素。

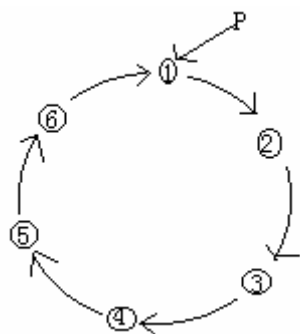
```

CONST maxlen=30
TYPE sqlisttp=RECORD
    elem:ARRAY[1..maxlen] OF integer;
    last:0..maxlen
END;
PROC exam21(VAR L:sqlisttp);
j:=1; i:=2;
WHILE (1)_____ DO
    [ IF L.elem[i]<>L.elem[j] THEN [ (2)_____ ; (3)_____ ];
    i:=i+1 ]
    (4)_____ ;

```

ENDP;【同济大学 2000 二、1 (10 分)】

26. 在本题的程序中，函数过程 Create_link_list(n)建立一个具有 n 个结点的环形链表;程序过程 josephus(n,i,m)对由 Create_link_list(n)所建立的具有 n 个结点的环形链表按一定的次序逐个输出并删除链表中的所有结点，参数 n(n>0)指明环形链表的结点个数，参数 i(1<=i<=n)指明起始结点，参数 m(m>0)是步长，指明从起始结点或前次被删除并输出的结点之后的第 m 个结点作为本次被输出并删除的结点。例如，对于下图中具有 6 个结点的环形链表，在调用 josephus(6,3,2)后，将输出 5,1,3,6,4,2 请在横线处填上适当内容，每空只填一个语句。



```

TYPE nodeptr = ^nodetype;
nodetype = RECORD
    data: integer; link: nodeptr
END;
VAR n, i, m: integer;
FUNCTION Create_link_list(n: integer): nodeptr;
VAR head, p, q: nodeptr; i: integer;
BEGIN head := NIL;
    IF n > 0 THEN
        BEGIN new(head); p := head;
            FOR i := 1 TO n-1 DO
                BEGIN p^.data := i; new(q); (A)____; (B)____ END;
                p^.data := n; (C)____;
            END;
        Creat_link_list := head
    END;
PROCEDURE josephus(n, i, m: integer);
VAR p, q: nodeptr; j: integer;
BEGIN p := Creat_link_list(n);
    WHILE i > 1 DO BEGIN p := p^.link; i := i-1 END;
    (D)____;
    WHILE j < n DO
        BEGIN
            FOR i := 1 TO m-1 DO p := p^.link;
                (E)____; write(q^.data:8); (F)____;
                dispose(q); j := j+1
            END
        END;
END; 【复旦大学 1997 四 (12 分)】

```

27. 对于给定的线性链表 head，下面的程序过程实现了按结点值非降次序输出链表中的所有结点，在每次输出一个结点时，就把刚输出的结点从链表中删去。请在划线处填上适当的内容，使之成为一个完整的程序过程，每个空框只填一个语句。

```

TYPE nodeptr = ^ nodetype;
nodetype = RECORD
    data : integer; link : nodeptr
END;
VAR head : nodeptr;
PROCEDURE sort_output_delete (head : nodeptr);
VAR p, q, r, s: nodeptr;
BEGIN WHILE head <> NIL DO
    BEGIN p := NIL ; q := head; r := q ; s := q^.link ;
        WHILE s <> NIL DO
            BEGIN IF s^.data < q^.data THEN BEGIN (1)____; (2)____ END ;
                r := s ; (3)____
            END;
        END;
    END;

```

```

        write(q^.data : 5) ;
        IF p=NIL THEN (4) ELSE (5) ;
        dispose (q) ;
    END;
    writeln

```

END; 【复旦大学 1996 七 (20 分) 1995 一 (12 分) 与本题相似】

28. 下面函数的功能是在一个按访问频度不增有序的, 带头结点的双向链环上检索关键值为 x 的结点, 对该结点访问频度计数, 并维护该链环有序。若未找到, 则插入该结点。所有结点的频度域初值在建表时都为零。请将程序中四处空缺补写完整。

```

TYPE
    link=^node
    node=RECORD
        key:char; freq:integer; pre,next:link;
    END;
VAR    l:link;
FUNCTION loc(l:link;x:char):link;
    VAR    p,q:link;
    BEGIN
        p:=l^.next; (1);
        WHILE p^.key<>x DO p:=p^.next;
        IF p=l THEN [ new(q); q^.key:=x; q^.freq:=0 ]
        ELSE {找到}
            [ p^.freq:=p^.freq+1; q:=p; (2);
              WHILE q^.freq>p^.pre^.freq DO p:=p^.pre;
              IF p<>q THEN [ (3) ]
            ];
        IF (4) THEN [q^.next:=p; q^.pre:=p^.pre; p^.pre^.next:=q; p^.pre:=q]
        return(q);

```

END; 【北京工业大学 1999 五 (12 分)】

29. 循环链表 a 和 b 的结点值为字母, 其中 a 表非递减有序, 下面的程序欲构造一个递增有序的循环链表 c , 其中结点的值为同时在 a, b 两链表中出现的字母, 且 c 中字母不重复, 请补上程序中空缺的部分, 并估计算法的时间复杂度。(设 a, b 的结点数分别为 m, n)

```

TYPE
    link=^node;
    node=RECORD
        key:char; next:link
    END;
PROC    jj(a,b:link; VAR c:link);
VAR    p,q,r,s:link;
BEGIN
    new(c);c^.next:=c; q:=a; p:=a^.next;
    WHILE p<>a DO
        [(1);
        WHILE p^.key=p^.next^.key DO [q:=p; p:=p^.next]; {跳过相同字母}

```

```

r:=b^.next ; (2) _____;
WHILE r^.key <>p^.key DO r:=r^.next;
IF r<>b THEN
    [ s:=p; q^.next:=p^.next; (3) _____;
      s^.next:=c^.next; c^.next:=s; c:=s ]
ELSE [ q:=p; p:=p^.next ]
]; c:=c^.next;
END;

```

算法时间复杂度为 O (4) _____ 【北京工业大学 2000 四 (15 分)】

30. 以下程序的功能是实现带附加头结点的单链表数据结点逆序连接, 请填空完善之。

```

void reverse(pointer h)
/* h 为附加头结点指针; 类型 pointer 同算法设计第 3 题*/
{ pointer p,q;
  p=h->next; h->next=NULL;
  while((1) _____)
  {q=p; p=p->next; q->next=h->next; h->next=(2) _____; }
} 【西南交通大学 2000 一、9】

```

31. 下面是用 c 语言编写的对不带头结点的单链表进行就地逆置的算法, 该算法用 L 返回逆置后的链表的头指针, 试在空缺处填入适当的语句。

```

void reverse (linklist &L) {
  p=null; q=L;
  while (q!=null)
  { (1) _____; q->next=p; p=q; (2) _____; }
  (3) _____;
} 【北京理工大学 2001 九、1 (6 分)】

```

32. 下面程序段是逆转单向循环链表的方法, p_0 是原链表头指针, 逆转后链表头指针仍为 p_0 。
(可以根据需要增加标识符)

```

p:= p0; q0:=NIL;
WHILE (1) _____ DO
  BEGIN (2) _____; (3) _____; (4) _____; (5) _____ END;
  p^.next:= q0; p0^.next:=p; p0:=p; 【中国人民大学 2000 二、1 (4 分)】

```

33. 一个无头结点的线性链表(不循环)有两个域。数据域 data, 指针域 next, 链首 head, 下面算法用 read(num) 读入数据, 当 num 小于 0 时, 输入结束。建立一个数据以递增序组成的链表。

```

PROC insert( head, x);
{在链首为 head 的表中按递增序插入 x}
new(r); r^.data:=x;
IF head=NIL
  THEN[ head:=(1) _____; r^.next:= (2) _____ ]
  ELSE IF (3) _____ THEN [r^.next:=head; head:=r]
  ELSE [p:=head;
        WHILE (4) _____ AND (p^.next ≠ NIL ) DO[q:=p;
(5) _____ ];
        IF (6) _____ THEN [ q^.next:=(7) _____; r^.next:=
(8) _____; ]

```

```

ELSE [p ^ .next:=(9)____; r ^ .next:=
(10)____; ]
]

```

ENDP;

PROC creat(head);

head:= (11)____; read(num);

WHILE num>0 DO

[insert(head,num); read(num)]

ENDP; 【南京理工大学 1999 三、4 (11 分)】

34. 一元稀疏多项式以循环单链表按降幂排列，结点有三个域，系数域 coef，指数域 exp 和指针域 next；现对链表求一阶导数，链表的头指针为 ha，头结点的 exp 域为 -1。

derivative(ha)

{ q=ha; pa=ha->next;

while((1)____)

{ if ((2)____) { ((3)____); free(pa); pa= ((4)____); }

else{ pa->coef ((5)____); pa->exp((6)____); q=((7)____); }

pa=((8)____);

}

} 【南京理工大学 2000 三、3 (10 分)】

35. 下面是删除单链表 L 中最大元素所在结点的类 PASCAL 语言算法，请在横线填上内容，完成其功能。

TYPE pointer = ↑ node;

node=RECORD

data:integer; next:pointer

END;

PROCEDURE delmax (L:pointer);

VAR p,q,r:pointer; m:integer;

BEGIN

r:=L; p:=L ↑ .next;

IF p<>NIL THEN

[m:=p ↑ .data; (1)____; p:=p ↑ .next;

WHILE p<>NIL DO

[IF (2)____ THEN [(3)____; m:=p ↑ .data;]

(4)____; p:=p ↑ .next;

]]

q:=r ↑ .next; (5)____; dispose(q);

]]

END; 【北京科技大学 1998 二】

36. 对单链表中元素按插入方法排序的 C 语言描述算法如下, 其中 L 为链表头结点指针。请填充算法中标出的空白处, 完成其功能。

```
typedef struct node
{
    int data; struct node *next;
}linknode,*link;

void Insertsort(link L)
{
    link p,q,r,u;
    p=L->next; (1)_____;
    while((2)_____)
    {
        r=L; q=L->next;
        while((3)_____ && q->data<=p->data) {r=q; q=q->next;}
        u=p->next; (4)_____; (5)_____; p=u;
    }
}
```

【北京科技大学 2001 二 (10 分)】

37. 下面是一个求两个集合 A 和 B 之差 $C=A-B$ 的程序, 即当且仅当 e 是 A 的一个元素, 但不是 B 中的一个元素时, e 才是 C 中的一个元素。集合用有序链表实现, 初始时, A, B 集合中的元素按递增排列, C 为空; 操作完成后 A, B 保持不变, C 中元素按递增排列。下面的函数 append(last, e) 是把值为 e 的新结点链接在由指针 last 指向的结点的后面, 并返回新结点的地址; 函数 difference(A, B) 实现集合运算 $A-B$, 并返回表示结果集合 C 的链表的首结点的地址。在执行 $A-B$ 运算之前, 用于表示结果集合的链表首先增加一个附加的表头结点, 以便新结点的添加, 当 $A-B$ 运算执行完毕, 再删除并释放表示结果集合的链表的表头结点。

程序(a) (编者略去这个 PASCAL 程序)

程序(b)

```
typedef struct node{ int element; struct node *link;
}NODE;

NODE *A, *B, *C;
NODE *append (NODE *last,int e)
{
    last->link=(NODE*) malloc (sizeof(NODE));
    last->link->element=e;
    return(last->link);
}

NODE *difference(NODE *A,NODE *B)
{
    NODE *C,*last;
    C=last=(NODE*) malloc (sizeof(NODE));
    while (1)_____
        if (A->element<B->element) { last=append(last,A->element); A=A->link; }
        else if (2)_____ { A=A->link; B=B->link; } ELSE (3)_____ ;
    while (4)_____
        { last=append(last,A->element); A=A->link; }
    (5)_____; last=C; C=C->link; free (last); return (C);
}

/*call form:C=difference(A,B);*/ 【上海大学 2000 一、4 (10 分)】
```

四 应用题

1. 线性表有两种存储结构：一是顺序表，二是链表。试问：

(1) 如果有 n 个线性表同时并存，并且在处理过程中各表的长度会动态变化，线性表的总数也会自动地改变。在此情况下，应选用哪种存储结构？为什么？

(2) 若线性表的总数基本稳定，且很少进行插入和删除，但要求以最快的速度存取线性表中的元素，那么应采用哪种存储结构？为什么？【西安电子科技大学 1999 软件 二、1 (5 分)】

2. 线性表的顺序存储结构具有三个弱点：其一，在作插入或删除操作时，需移动大量元素；其二，由于难以估计，必须预先分配较大的空间，往往使存储空间不能得到充分利用；其三，表的容量难以扩充。线性表的链式存储结构是否一定都能够克服上述三个弱点，试讨论之。【重庆大学 2000 二、5】

3. 若较频繁地对一个线性表进行插入和删除操作，该线性表宜采用何种存储结构？为什么？

【北京航空航天大学 1998 一、2 (4 分)】

4. 线性结构包括_____、_____和_____。线性表的存储结构分成_____和_____。请用类 PASCAL 语言描述这两种结构。【华北计算机系统工程研究所 1999 一、2 (10 分)】

5. 线性表 (a_1, a_2, \dots, a_n) 用顺序映射表示时， a_i 和 a_{i+1} ($1 \leq i < n$) 的物理位置相邻吗？链接表示时呢？

【东南大学 1996 一、1 (5 分)】

6. 说明在线性表的链式存储结构中，头指针与头结点之间的根本区别；头结点与首元结点的关系。

【厦门大学 2000 五、1 (14%/3 分)】

7. 试述头结点，首元结点，头指针这三个概念的区别。

【武汉交通科技大学 1996 二、2 (3 分)】【西安电子科技大学 2001 计应用 二、1 (5 分)】

8. 已知有如下定义的静态链表：

```
TYPE component=RECORD
```

```
    data:elemtp;
```

```
    next:0..maxsize
```

```
END
```

```
VAR stalist:ARRAY[0..maxsize] OF component;
```

以及三个指针:av 指向头结点，p 指向当前结点，pre 指向前驱结点，现要求修改静态链表中 next 域中的内容，使得该静态链表有双向链表的功能，从当前结点 p 既能往后查找，也能往前查找：

(1) 定义 next 域中的内容。(用老的 next 域中的值表示)；

(2) 如何得到当前结点 p 的前驱 (pre) 的前驱，给出计算式；

(3) 如何得到 p 的后继，给出计算式；【中科院计算所 2000 四 (10 分)】

9. 在单链表和双向链表中，能否从当前结点出发访问到任何一个结点？

【西安电子科技大学 1999 计应用一、1 (5 分)】

10. 如何通过改链的方法，把一个单向链表变成一个与原来链接方向相反的单向链表？

【中国人民大学 2001 二、4 (2 分)】

11. 下面是一算法的核心部分，试说明该算法的功能。

```
pre:=L↑.next;
```

```
{L 是一单链表，结点有数据域 data 和指针域 next}
```

```
IF pre<>NIL THEN
```

```
    WHILE pre↑.next<>NIL DO
```

```
        BEGIN p:=pre↑.next; IF p↑.data>pre↑.data THEN pre:=p ELSE return(false)
```

```
END;
```

```
return(true); 【燕山大学 2000 七、1 (7 分)】
```

12. 设单链表结点指针域为 next, 试写出删除链表中指针 p 所指结点的直接后继的 C 语言语句。

【北京科技大学 2000 一、3】

13. 设单链表中某指针 p 所指结点 (即 p 结点) 的数据域为 data, 链指针域为 next, 请写出在 p 结点之前插入 s 结点的操作 (PASCAL 语句)。【北京科技大学 1999 一、2 (2 分)】

14. 有线性表 (a_1, a_2, \dots, a_n) , 采用单链表存储, 头指针为 H, 每个结点中存放线性表中一个元素, 现查找某个元素值等于 X 的结点。分别写出下面三种情况的查找语句。要求时间尽量少。

(1) 线性表中元素无序。(2) 线性表中元素按递增有序。(3) 线性表中元素按递减有序。

【北京邮电大学 1994 七 (7 分)】

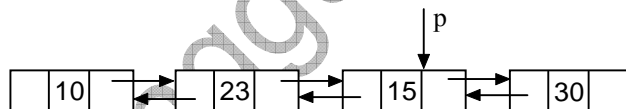
15. 设 pa, pb 分别指向两个带头结点的有序 (从小到大) 单链表。仔细阅读如下的程序, 并回答问题:

(1) 程序的功能。(2) s1, s2 中值的含义。(3) pa, pb 中值的含义。

```
PROCEDURE exam(pa, pb)
BEGIN
  p1:=pa↑.next; p2:=pb↑.next; pa↑.next:=∧; s1:=0; s2:=0;
  WHILE p1≠∧ AND p2≠∧ DO
  [ CASE p1↑.data<p2↑.data: [p:=p1; p1:=p1↑.next; s2:=s2+1; dispose(p) ];
    p1↑.data>p2↑.data: p2:=p2↑.next;
    p1↑.data=p2↑.data: [p:=p1; p1:=p1↑.next; p↑.next:= pa↑.next;
                        pa↑.next:= p; p2:= p2↑.next;s1:=s1+1; ];
  ]
  END
];
  WHILE p1≠∧ DO [ p:=p1; p1:=p1↑.next; dispose(p); s2:=s2+1 ]
END; 【南京航空航天大学 1995 十 (9 分)】
```

16. 写出下图双链表中对换值为 23 和 15 的两个结点相互位置时修改指针的有关语句。

结点结构为: (llink, data, rlink) 【北京邮电大学 1992 三、4 (25/4 分)】



17. 按照下列题目中的算法功能说明, 将算法描述片段中的错误改正过来。

(1) (4 分) 下面的算法描述片段用于在双链表中删除指针变量 p 所指的结点:

```
p^.rlink←p^.llink^.rlink;
p^.llink←p^.rlink^.llink
dispose(p);
```

(2) (6 分) 下面的算法描述片段用于在双链表中指针变量 p 所指结点后插入一个新结点:

```
new(q);
q^.llink←p;
p^.rlink←q;
q^.rlink←p^.rlink;
q←p^.rlink^.llink; 【山东大学 1999 八 (10 分)】
```

18. 已知 L 是一个数据类型 linkedlist 的单循环链表, pa 和 pb 是指向 L 中结点的指针。简述下列程序段的功能。【山东科技大学 2001 一、2 (5 分)】

```
TYPE linkedlist=↑node;
node=RECORD
  data:datatype; next:linkedlist
```

```

        END;
PROC  Mp(pa,pb:linkedList);
PROC  subp(s,q:linkedList);
    p:=s;
    WHILE  p↑.next<>q  DO  p:=p↑.next;
    p↑.next:=s
ENDP;
subp(pa,pb);
subp(pb,pa);
ENDP;

```

19. 设双向循环链表中结点的数据域、前驱和后继指针域分别为 data,pre 和 next,试写出在指针 p 所指结点之前插入一 s 结点的 C 语言描述语句。【北京科技大学 2001 一、3 (2 分)】

20. 本题给出一个子程序的框图,如图 2,试填空完善此算法框图。该子程序用来寻找第一个均出现在三个整数单向链表 f1, f2, f3 中的相同整数。假定在调用该子程序前,这三个整数链表已按从小到大的次序排序,单向链表的形式如下图 1 的例子所示。

图 1:

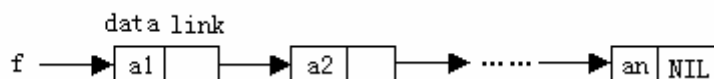
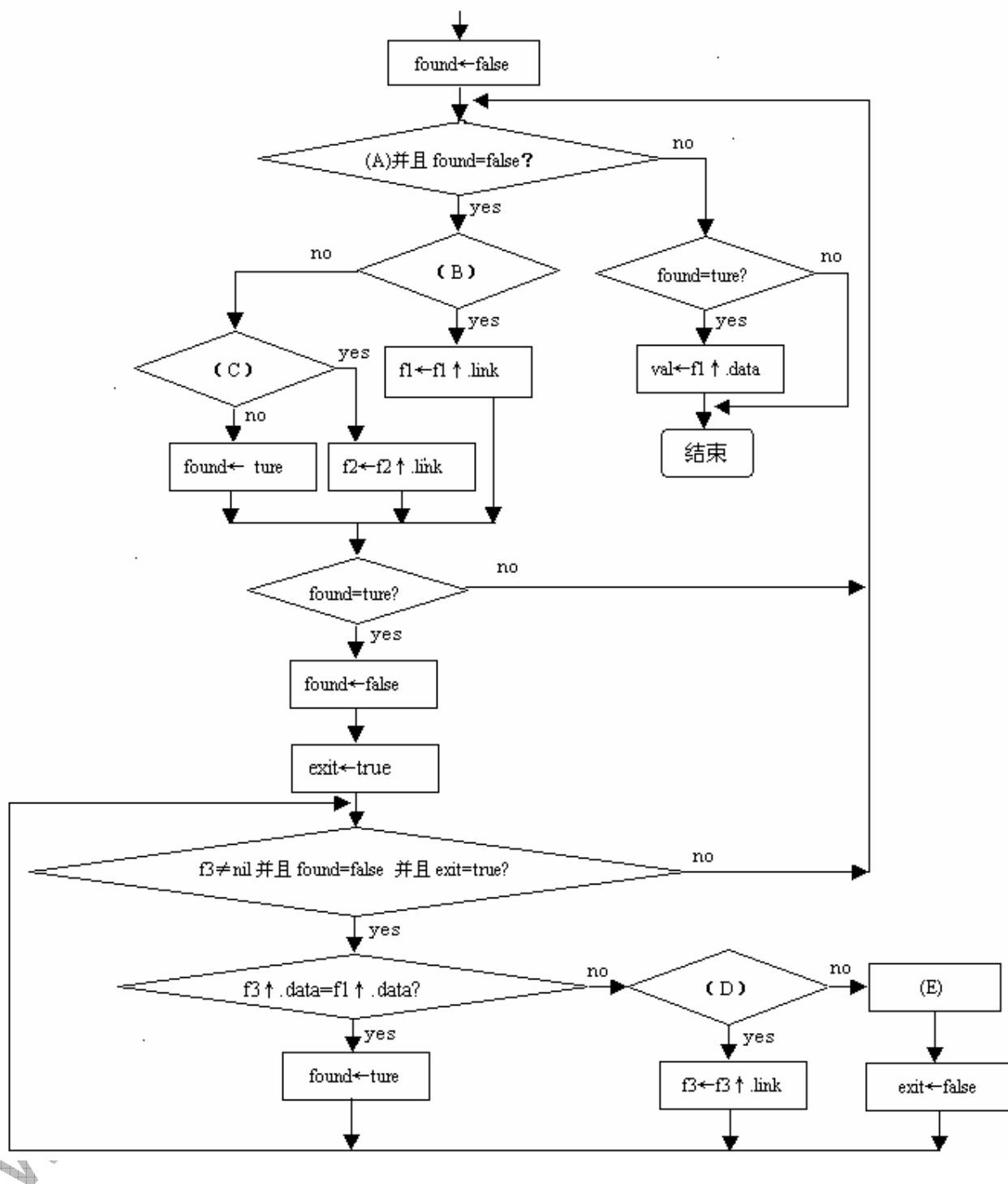


图 2:



注：在图 2 的框图中：found 和 exit 均为布尔型的变量，可取值为 true 和 false。val 是整型变量，用来存放第一个均出现在 f1, f2, f3 中的相同整数。若 f1, f2 和 f3 中无相同的整数，found 的值为 false，否则 found 的值为 true。f1 → link 表示访问 f1 所指结点的 link 域。

【哈尔滨工业大学 1999 三 (15 分)】

21. 一线性表存储在带头结点的双向循环链表中，L 为头指针。如下算法：

(1) 说明该算法的功能。(2) 在空缺处填写相应的语句。

```

void unknown (BNODETP *L)
{
    ...
}
    
```

```

p=L->next; q=p->next; r=q->next;
while (q!=L)
{ while (p!=L) && (p->data>q->data) p=p->prior;
  q->prior->next=r; (1)_____ ;
  q->next=p->next; q->prior=p;
  (2)_____ ; (3)_____ ; q=r; p=q->prior;
  (4)_____ ;
}
} 【北京理工大学 1999 第二部分 数据结构 [7] (8 分)】

```

五、算法设计题

1. 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递减次序排列的单链表，并要求利用原来两个单链表的结点存放归并后的单链表。

【北京大学 1998 三、1 (5 分)】

类似本题的另外叙述有：

(1) 设有两个无头结点的单链表，头指针分别为 ha, hb, 链中有数据域 data, 链域 next, 两链表的数据都按递增序存放，现要求将 hb 表归到 ha 表中，且归并后 ha 仍递增序，归并中 ha 表中已有的数据若 hb 中也有，则 hb 中的数据不归并到 ha 中，hb 的链表在算法中不允许破坏。【南京理工大学 1997 四、3 (15 分)】

PROCEDURE merge(ha, hb);

(2) 已知头指针分别为 la 和 lb 的带头结点的单链表中，结点按元素值非递减有序排列。写出将 la 和 lb 两链表归并成一个结点按元素值非递减有序排列的单链表（其头指针为 lc），并计算算法的时间复杂度。【燕山大学 1998 五 (20 分)】

2. 图（编者略）中带头结点且头指针为 ha 和 hb 的两线性表 A 和 B 分别表示两个集合。两表中的元素皆为递增有序。请写一算法求 A 和 B 的并集 AUB。要求该并集中的元素仍保持递增有序。且要利用 A 和 B 的原有结点空间。【北京邮电大学 1992 二 (15 分)】

类似本题的另外叙述有：

(1) 已知递增有序的两个单链表 A, B 分别存储了一个集合。设计算法实现求两个集合的并集的运算 $A := A \cup B$ 【合肥工业大学 1999 五、1 (8 分)】

(2) 已知两个链表 A 和 B 分别表示两个集合，其元素递增排列。编一函数，求 A 与 B 的交集，并存放于 A 链表中。【南京航空航天大学 2001 六 (10 分)】

(3) 设有两个从小到大排序的带头结点的有序链表。试编写求这两个链表交运算的算法（即 $L1 \cap L2$ ）。要求结果链表仍是从小到大排序，但无重复元素。【南京航空航天大学 1996 十一 (10 分)】

(4) 已知两个线性表 A, B 均以带头结点的单链表作存储结构，且表中元素按值递增有序排列。设计算法求出 A 与 B 的交集 C，要求 C 另开辟存储空间，要求 C 同样以元素值的递增序的单链表形式存贮。

【西北大学 2000 五 (8 分)】

(5) 已知递增有序的单链表 A, B 和 C 分别存储了一个集合，设计算法实现 $A := A \cup (B \cap C)$ ，并使求解结构 A 仍保持递增。要求算法的时间复杂度为 $O(|A| + |B| + |C|)$ 。其中， $|A|$ 为集合 A 的元素个数。

【合肥工业大学 2000 五、1 (8 分)】

3. 知 L1、L2 分别为两循环单链表的头结点指针，m, n 分别为 L1、L2 表中数据结点个数。要求

设计一算法,用最快速度将两表合并成一个带头结点的循环单链表。【东北大学 1996 二 (12 分)】

类似本题的另外叙述有:

(1) 试用类 Pascal 语言编写过程 PROC join (VAR la: link; lb: link) 实现连接线性表 la 和 lb(lb 在后)的算法,要求其时间复杂度为 $O(1)$, 占用辅助空间尽量小。描述所用结构。

【北京工业大学 1997 一、1 (8 分)】

(2) 设有两个链表, ha 为单向链表, hb 为单向循环链表。编写算法,将两个链表合并成一个单向链表,要求算法所需时间与链表长度无关。【南京航空航天大学 1997 四 (8 分)】

4. 顺序结构线性表 LA 与 LB 的结点关键字为整数。LA 与 LB 的元素按非递减有序,线性表空间足够大。试用类 PASCAL 语言给出一种高效算法,将 LB 中元素合到 LA 中,使新的 LA 的元素仍保持非递减有序。高效指最大限度的避免移动元素。【北京工业大学 1997 一、2 (12 分)】

5. 已知不带头结点的线性链表 list,链表中结点构造为 (data、link),其中 data 为数据域,link 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。【北京航空航天大学 1998 五 (15 分)】

6. 设 L 为单链表的头结点地址,其数据结点的数据都是正整数且无相同的,试设计利用直接插入的原则把该链表整理成数据递增的有序单链表的算法。【东北大学 1996 六 (14 分)】

类似本题的另外叙述有:

(1) 设一单向链表的头指针为 head,链表的记录中包含着整数类型的 key 域,试设计算法,将此链表的记录按照 key 递增的次序进行就地排序。【中科院计算所 1999 五、1 (10 分)】

7. 设 Listhead 为一单链表的头指针,单链表的每个结点由一个整数域 DATA 和指针域 NEXT 组成,整数在单链表中是无序的。编一 PASCAL 过程,将 Listhead 链中结点分成一个奇数链和一个偶数链,分别由 P, Q 指向,每个链中的数据按由小到大排列。程序中不得使用 NEW 过程申请空间。【山东大学 1993 六 (15 分)】

类似本题的另外叙述有:

(1) 设计算法将一个带头结点的单链表 A 分解为两个具有相同结构的链表 B、C,其中 B 表的结点为 A 表中值小于零的结点,而 C 表的结点为 A 表中值大于零的结点(链表 A 的元素类型为整型,要求 B、C 表利用 A 表的结点)。【北京理工大学 2000 四、2 (4 分)】

(2) 设 L 为一单链表的头指针,单链表的每个结点由一个整数域 data 和指针域 NEXT 组成,整数在单链表中是无序的。设计算法,将链表中结点分成一个奇数链和一个偶数链,分别由 P, Q 指向,每个链中的数据按由小到大排列,算法中不得申请新的结点空间。【青岛海洋大学 1999 三 (12 分)】

(3) 将一个带头结点的单链表 A 分解为两个带头结点的单链表 A 和 B,使得 A 表中含有原表中序号为奇数的元素,而 B 表中含有原表中序号为偶数的元素,且保持其相对顺序不变。

1) 写出其类型定义:

2) 写出算法。【山东大学 1998 九 (9 分)】 【山东工业大学 2000 九 (9 分)】

8. 已知线性表 $(a_1 a_2 a_3 \cdots a_n)$ 按顺序存于内存,每个元素都是整数,试设计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 $(-x, -x, -x \cdots x, x, x, x)$ 。

【东北大学 1998 二 (15 分)】

类似本题的另外叙述有:

(1) 设有一元素为整数的线性表 $L=(a_1, a_2, a_3, \cdots, a_n)$,存放在一维数组 $A[N]$ 中,设计一个算法,以表中 a_n 作为参考元素,将该表分为左、右两部分,其中左半部分每个元素小于等于 a_n ,右半部分每个元素都大于 a_n , a_n 位于分界位置上(要求结果仍存放在 $A[N]$ 中)。【北京理工大学 1999 八 (6 分)】

(2) 顺序存储的线性表 A, 其数据元素为整型, 试编写一算法, 将 A 拆成 B 和 C 两个表, 使 A 中元素值大于等于 0 的元素放入 B, 小于 0 的放入 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 A 的空间. 【山东大学 2001 九、1 (12 分)】

(3) 知线性表 $(a_1, a_2, a_3, \dots, a_n)$ 按顺序存储, 且每个元素都是整数均不相同, 设计把所有奇数移到所有偶数前边的算法。(要求时间最少, 辅助空间最少) 【东北大学 1997 三 (15 分)】

(4) 编写函数将一整数序列中所有负数移到所有正数之前, 要求时间复杂度为 $O(n)$

【南京航空航天大学 2001 八 (10 分)】

(5) 已知一个由 n (设 $n=1000$) 个整数组成的线性表, 试设计该线性表的一种存储结构, 并用标准 pascal 语言描述算法, 实现将 n 个元素中所有大于等于 19 的整数放在所有小于 19 的整数之后。要求算法的时间复杂度为 $O(n)$, 空间复杂度 $O(1)$ 。【西安交通大学 1996 六 (11 分)】

9. 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。void delete(Linklist &L)

【北京理工大学 2001 九、3 (8 分)】

10. 已知非空线性链表由 list 指出, 链结点的构造为 (data, link). 请写一算法, 将链表中数据域值最小的那个链结点移到链表的最前面。要求: 不得额外申请新的链结点。【北京航空航天大学 2001 四 (10 分)】

11. 已知 p 指向双向循环链表中的一个结点, 其结点结构为 data、llink、rlink 三个域, 写出算法 change(p), 交换 p 所指向的结点和它的前继结点的顺序。【首都经贸大学 1997 二、2 (15 分)】

12. 线性表 $(a_1, a_2, a_3, \dots, a_n)$ 中元素递增有序且按顺序存储于计算机内。要求设计一算法完成:

(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。【东北大学 1996 三 (12 分)】

13. 设单链表的表头指针为 h, 结点结构由 data 和 next 两个域构成, 其中 data 域为字符型。写出算法 dc(h, n), 判断该链表的前 n 个字符是否中心对称。例如 xyx, xyyx 都是中心对称。【首都经贸大学 1998 三、9 (15 分)】

14. 已知两个单链表 A 和 B, 其头指针分别为 heada 和 headb, 编写一个过程从单链表 A 中删除自第 i 个元素起的共 len 个元素, 然后将单链表 A 插入到单链表 B 的第 j 个元素之前。

【中国矿业大学 2000 三 (10 分)】

类似本题的另外叙述有:

(1) h1、h2 为两个链表的表头指针, 结点结构为 data 和 link 两个域组成。写出算法 inde(h1, h2, i, j, l), 将链表 h1 从第 i 个结点起的 l 个结点删除, 并插入到 h2 表的第 j 个结点之前。

【首都经贸大学 1998 三、10 (20 分)】

15. 设线性表存于 $A[1..size]$ 的前 num 分量中, 且递增有序。请设计一个算法, 将 x 插入到线性表的适当位置上, 以保持线性表的有序性, 并在设计前说明设计思想, 最后说明所设计算法的时间复杂度。

【西安电子科技大学 1999 计应用 1997 二 (10 分)】

类似本题的另外叙述有:

(1) 试编制在线性表 $L=\{12, 13, 21, 24, 28, 30, 42, \dots\}$ 中插入数据元素 26 的程序。(要求该程序用 turbo Pascal 语言编制并能在计算机上运行, 结点类型为链式结构)【大连海事大学 1996 二、1 (16 分)】

16. 假设一个单循环链表, 其结点含有三个域 pre、data、link。其中 data 为数据域; pre 为指

针域，它的值为空指针（NIL）；link 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。

【西安电子科技大学 1999 软件 五（10 分）】

17. 已知递增有序的单链表 A, B 分别存储了一个集合，请设计算法以求出两个集合 A 和 B 的差集 A-B（即仅由在 A 中出现而不在 B 中出现的元素所构成的集合），并以同样的形式存储，同时返回该集合的元素个数。

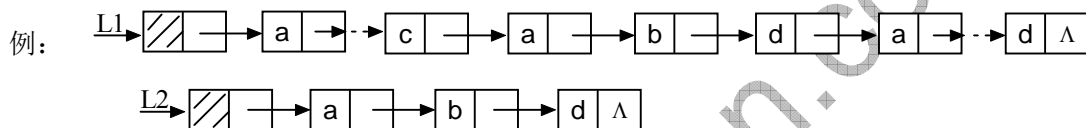
【西安电子科技大学 2000 计应用 1997 二（10 分）】

18. 已知一个单链表中每个结点存放一个整数，并且结点数不少于 2，请设计算法以判断该链表中第二项起的每个元素值是否等于其序号的平方减去其前驱的值，若满足则返回 true，否则返回 false。

【西安电子科技大学 2000 软件 1997 二（10 分）】

19. 两个整数序列 $A=a_1, a_2, a_3, \dots, a_m$ 和 $B=b_1, b_2, b_3, \dots, b_n$ 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。【东北大学 1999 二（10 分）】

20. L1 与 L2 分别为两单链表头结点地址指针，且两表中数据结点的数据域均为一个字母。设计把 L1 中与 L2 中数据相同的连续结点顺序完全倒置的算法。【东北大学 1997 四（15 分）】



类似本题的另外叙述有：

(1) 知 L 为链表的头结点地址，表中共有 $m(m>3)$ 个结点，从表中第 i 个结点 ($1<i<m$) 起到第 m 个结点构成一个循环部分链表，设计将这部分循环链表中所有结点顺序完全倒置的算法。



【东北大学 1998 三（15 分）】

21. 请写一个算法将顺序存储结构的线性表 $(a_1 \dots a_n)$ 逆置为 $(a_n \dots a_1)$ 。【大连海事大学 1996 八（6 分）】

类似本题的另外叙述有：

(1) 设有一带头结点的单链表，编程将链表颠倒过来。要求不用另外的数组或结点完成。

【南京航空航天大学 1999 八（10 分）】

(2) 设有一个带头结点的单向链表，数据项递减有序。写一算法，重新排列链表，使数据项递增有序，要求算法时间复杂度为 $O(n)$ 。（注：用程序实现）【南京航空航天大学 1997 七（12 分）】

(3) 试编写求倒排循环链表元素的算法。【南京航空航天大学 1995 十二（10 分）】

(4) 请设计算法将不带头结点的单链表就地逆置。【北方交通大学 2001 三（12 分）】

(5) 试编写算法，将不设表头结点的、不循环的单向链表就地逆转。【北方交通大学 1997 五（10 分）】

(6) 有一个单链表 L（至少有 1 个结点），其头结点指针为 head，编写一个过程将 L 逆置，即最后一个结点变成第一个结点，原来倒数第二个结点变成第二个结点，如此等等。【燕山大学 2001 四、2（8 分）】

22. 设有一个由正整数组成的无序（向后）单链表，编写完成下列功能的算法：

(1) 找出最小值结点，且打印该数值；

(2) 若该数值是奇数，则将其与直接后继结点的数值交换；

(3) 若该数值是偶数，则将其直接后继结点删除。【东北大学 2000 二（15 分）】

23. 已知 L 为没有头结点的单链表中第一个结点的指针, 每个结点数据域存放一个字符, 该字符可能是英文字母字符或数字字符或其它字符, 编写算法构造三个以带头结点的单循环链表表示的线性表, 使每个表中只含同一类字符。(要求用最少的时间和最少的空间)【东北大学 2002 三 (15 分)】

24. 在一个递增有序的线性表中, 有数值相同的元素存在。若存储方式为单链表, 设计算法去掉数值相同的元素, 使表中不再有重复的元素。例如: (7, 10, 10, 21, 30, 42, 42, 42, 51, 70) 将变作 (7, 10, 21, 30, 42, 51, 70), 分析算法的时间复杂度。【北京工业大学 1996 三 (15 分)】

25. 在输入数据无序的情况下, 建立一个数据值为整型的递增有序的顺序存储线性表 L, 且要求当输入相同数据值时, 线性表中不能存在数据值相同的数据元素, 试写出其算法。

顺序存储结构的线性表描述为:

```
CONST maxlen={线性表可能达到的最大长度};
TYPE sqllisttp=RECORD
    elem:array[1..maxlen] of integer;
    last :0..maxlen
END;
```

VAR L: sqllisttp;【同济大学 1998 二 (12 分)】

26. 设有一个正整数序列组成的有序单链表 (按递增次序有序, 且允许有相等的整数存在), 试编写能实现下列功能的算法: (要求用最少的时间和最小的空间)

(1) 确定在序列中比正整数 x 大的数有几个 (相同的数只计算一次, 如序列 {20, 20, 17, 16, 15, 15, 11, 10, 8, 7, 7, 5, 4} 中比 10 大的数有 5 个);

(2) 在单链表将比正整数 x 小的数按递减次序排列;

(3) 将正整数 (比) x 大的偶数从单链表中删除。【东北大学 2001 二 (17 分)】

27. 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。【吉林大学 2001 二、1 (7 分)】

类似本题的另外叙述有:

(1) 已知非空线性链表第一个结点由 List 指出, 请写一算法, 交换 p 所指的结点与其下一个结点在链表中的位置 (设 p 指向的不是链表最后那个结点)。【北京航空航天大学 1999 五 (10 分)】

(2) 已知任意单链表如图所示 (编者略去图)。Head 为表头指针, 指向表的第一个元素, p 为指向表中任意结点的指针, 试设计一个算法, 将 p 指向的结点和其后面结点交换位置 (可采用任何高级语言描述算法)。

【山东大学 1992 二 (12 分)】

28. 设键盘输入 n 个英语单词, 输入格式为 n, w₁, w₂, ..., w_n, 其中 n 表示随后输入英语单词个数, 试编一程序, 建立一个单向链表, 实现: (10 分)

(1) 如果单词重复出现, 则只在链表上保留一个。(单考生做)。

(2) 除满足 (1) 的要求外。链表结点还应有一个计数域, 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 ≤ n) 个单词 (统考生做)。【南京航空航天大学 1998 九 (10 分)】

29. 已知一双向循环链表, 从第二个结点至表尾递增有序, (设 a₁ < x < a_n) 如下图 (“第二个结点至表尾”指 a₁..a_n, 因篇幅所限, 编者略去图)。试编写程序, 将第一个结点删除并插入表中适当位置, 使整个链表递增有序。【南京航空航天大学 1998 八 (10 分)】

30. 已知长度为 n 的线性表 A 采用顺序存储结构, 请写一时间复杂度为 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)。

【北京航空航天大学 2000 五 (10 分)】

31. 设民航公司有一个自动预订飞机票的系统, 该系统有一张用双重链表示的乘客表, 表中结

点按乘客姓氏的字母序相链。例如，下面是张某个时刻的乘客表。试为该系统写出一个当任一乘客要订票时修改乘客表的算法。

序号	data	Llink	Rlink
1	Liu	6	5
2	Chan	4	9
3	Wang	5	7
4	Bao	0	2
5	Mai	1	3
6	Dong	8	1
7	Xi	3	0
8	Deng	9	6
9	Cuang	2	8

【北方交通大学 2000 六 (17 分)】

32. 设有一头指针为 L 的带有表头结点的非循环双向链表，其每个结点中除有 pred (前驱指针)，data (数据) 和 next (后继指针) 域外，还有一个访问频度域 freq。在链表被起用前，其值均初始化为零。每当在链表中进行一次 Locate(L, x) 运算时，令元素值为 x 的结点中 freq 域的值增 1，并使此链表中结点保持按访问频度非增 (递减) 的顺序排列，同时最近访问的结点排在频度相同的结点的最后，以便使频繁访问的结点总是靠近表头。试编写符合上述要求的 Locate(L, x) 运算的算法，该运算为函数过程，返回找到结点的地址，类型为指针型。【清华大学 1997 二 (10 分)】

33. 给定 (已生成) 一个带表头结点的单链表，设 head 为头指针，结点的结构为 (data, next)，data 为整型元素，next 为指针，试写出算法：按递增次序输出单链表中各结点的数据元素，并释放结点所占的存储空间。(要求：不允许使用数组作辅助空间) 【华中理工大学 2000 八、2 (13 分)】

34. 已知三个带头结点的线性链表 A、B 和 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 A 表进行如下操作：使操作后的链表 A 中仅留下三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 分别为三个表的长度。【清华大学 1995 一 (15 分)】

第 2 章 线性表

一. 选择题

1. A	2. B	3. C	4. A	5. D	6. D	7. D	8. C	9. B	10. B, C	11. 1I	11. 2I	11. 3E
11. 4B	11. 5C	12. B	13. C	14. C	15. C		16. A	17. A	18. A	19. D	20. C	21. B
22. D	23. C	24. B	25. B	26. A	27. D							

二. 判断题

1. ×	2. √	3. √	4. ×	5. ×	6. ×	7. ×	8. ×	9. ×	10. ×	11. ×	12. ×
13. ×	14. √	15. ×	16. √								

部分答案解析如下。

1. 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。
4. 两种存储结构各有优缺点，应根据实际情况选用，不能笼统说哪个好。
7. 集合中元素无逻辑关系。
9. 非空线性表第一个元素无前驱，最后一个元素无后继。

13. 线性表是逻辑结构，可以顺序存储，也可链式存储。

三. 填空题

1. 顺序
2. $(n-1)/2$
3. $py \rightarrow next = px \rightarrow next; px \rightarrow next = py$
4. $n-i+1$
5. 主要是使插入和删除等操作统一，在第一个元素之前插入元素和删除第一个结点不必另作判断。另外，不论链表是否为空，链表指针不变。
6. $O(1), O(n)$
7. 单链表，多重链表，(动态)链表，静态链表
8. $f \rightarrow next = p \rightarrow next; f \rightarrow prior = p; p \rightarrow next \rightarrow prior = f; p \rightarrow next = f;$
9. $p \rightarrow prior \quad s \rightarrow prior \rightarrow next$
10. 指针
11. 物理上相邻
12. 4
13. 从任一结点出发都可访问到链表中每一个元素。
14. $u = p \rightarrow next; p \rightarrow next = u \rightarrow next; free(u);$
15. $L \rightarrow next \rightarrow next = L$
16. $p \rightarrow next \neq null$
17. $L \rightarrow next = L \ \&\& \ L \rightarrow prior = L$
18. $s \rightarrow next = p \rightarrow next; p \rightarrow next = s;$
19. (1) IF $pa = NIL$ THEN return(true);
 (2) $pb \neq NIL$ AND $pa \rightarrow data = pb \rightarrow data$
 (3) return(inclusion(pa, pb));
 (4) $pb := pb \rightarrow next;$
 (5) return(false);
 非递归算法:
 (1) $pre := pb;$ (2) $pa \neq NIL$ AND $pb \neq NIL$ AND $pb \rightarrow data = pa \rightarrow data$ (3) $pa := pa \rightarrow next;$
 $pb := pb \rightarrow next;$
 (4) $pb := pre \rightarrow next; pre := pb; pa := pa \rightarrow next;$ (5) IF $pa = NIL$ THEN return(true) ELSE return(false);
- [注]: 本题是在链表上求模式匹配问题。非递归算法中用指针 pre 指向主串中开始结点(初始时为第一元素结点)。若主串与子串对应数据相等，两串工作指针 pa 和 pb 后移; 否则，主串工作指针从 pre 的下一结点开始(这时 pre 又指向新的开始结点)，子串工作指针从子串第一元素开始，比较一直继续到循环条件失败。若 pa 为空，则匹配成功，返回 true，否则，返回 false。
20. A. VAR head:ptr B. new(p) C. $p \rightarrow data := k$ D. $q \rightarrow next := p$ E. $q := p$ (带头结点)
21. (1) new(h); //生成头结点，以便于操作。
 (2) $r \rightarrow next := p;$ (3) $r \rightarrow next := q;$ (4) IF $(q = NIL)$ THEN $r \rightarrow next := p;$
22. A: $r \rightarrow link \rightarrow data \neq max$ AND $q \rightarrow link \rightarrow data \neq max$
 B: $r := r \rightarrow link$ C: $q \rightarrow link$ D: $q \rightarrow link$ E: $r \rightarrow link$ F: $r \rightarrow link$
 G: $r := s$ (或 $r := r \rightarrow link$) H: $r := r \rightarrow link$ I: $q \rightarrow link := s \rightarrow link$
23. (1) la (2) 0 (3) $j < i-1$ (4) $p \uparrow \rightarrow next$ (5) $i < 1$
24. (1) $head \rightarrow left := s$ //head 的前驱指针指向插入结点
 (2) $j := 1;$
 (3) $p := p \rightarrow right$ //工作指针后移
 (4) $s \rightarrow left := p$
 (5) $p \rightarrow right \rightarrow left := s;$ //p 后继的前驱是 s
 (6) $s \rightarrow left := p;$
25. (1) $i \leq L \rightarrow last$ //L.last 为元素个数
 (2) $j := j+1$ //有值不相等的元素

- (3) $L.elem[j] := L.elem[i]$ // 元素前移
- (4) $L.last := j$ // 元素个数
26. (A) $p.link := q$; // 拉上链, 前驱指向后继
- (B) $p := q$; // 新的前驱
- (C) $p.link := head$; // 形成循环链表
- (D) $j := 0$; // 计数器, 记被删结点
- (E) $q := p.link$ // 记下被删结点
- (F) $p.link := q.link$ // 删除结点
27. (1) $p := r$; // r 指向工作指针 s 的前驱, p 指向最小值的前驱。
- (2) $q := s$; // q 指向最小值结点, s 是工作指针
- (3) $s := s.link$ // 工作指针后移
- (4) $head := head.next$; // 第一个结点值最小;
- (5) $p.link := q.link$; // 跨过被删结点 (即删除一结点)
28. (1) $l.key := x$; // 头结点 l 这时起监视哨作用
- (2) $l.freq := p.freq$ // 头结点起监视哨作用
- (3) $q \rightarrow pre \rightarrow next = q \rightarrow next$; $q \rightarrow next \rightarrow pre = q \rightarrow pre$; // 先将 q 结点从链表上摘下
- $q.next := p$; $q.pre := p.pre$; $p.pre \rightarrow next := q$; $p.pre := q$; // 结点 q 插入结点 p 前
- (4) $q.freq = 0$ // 链表中无值为 x 的结点, 将新建结点插入到链表最后 (头结点前)。
29. (1) $a.key := '@'$ // a 的头结点用作监视哨, 取不同于 a 链表中其它数据域的值
- (2) $b.key := p.key$ // b 的头结点起监视哨作用
- (3) $p := p.next$ // 找到 a, b 表中共同字母, a 表指针后移
- (4) $0(m*n)$
30. C 部分: (1) $p! = null$ // 链表未到尾就一直作
- (2) q // 将当前结点作为头结点后的第一元素结点插入
31. (1) $L = L \rightarrow next$; // 暂存后继
- (2) $q = L$; // 待逆置结点
- (3) $L = p$; // 头指针仍为 L
32. (1) $p.next \neq p_0$ (2) $r := p.next$ (3) $p.next := q_0$;
- (4) $q_0 := p$; (5) $p := r$
33. (1) r (2) NIL (3) $x < head.data$ (4) $p.data < x$
- (5) $p := p.next$ (6) $p.data > x$; (7) r (8) p
- (9) r (10) NIL (11) NIL
34. (1) $pa \neq ha$ // 或 $pa \rightarrow exp \neq -1$
- (2) $pa \rightarrow exp = 0$ // 若指数为 0, 即本项为常数项
- (3) $q \rightarrow next = pa \rightarrow next$ // 删常数项
- (4) $q \rightarrow next$ // 取下一元素
- (5) $= pa \rightarrow coef * pa \rightarrow exp$
- (6) $--$ // 指数项减 1
- (7) pa // 前驱后移, 或 $q \rightarrow next$
- (8) $pa \rightarrow next$ // 取下一元素
35. (1) $q := p$; // q 是工作指针 p 的前驱
- (2) $p.data > m$ // p 是工作指针
- (3) $r := q$; // r 记最大值的前驱,

- ```

(4) q:=p; // 或 q:=q^.next;
(5) r^.next:=q^.next; // 或 r^.next:=r^.next^.next 删最大值结点
36. (1) L->next=null // 置空链表, 然后将原链表结点逐个插入到有序表中
 (2) p!=null // 当链表尚未到尾, p 为工作指针
 (3) q!=null // 查 p 结点在链表中的插入位置, 这时 q 是工作指针。
 (4) p->next=r->next // 将 p 结点链入链表中
 (5) r->next=p // r 是 q 的前驱, u 是下个待插入结点的指针。
37. 程序 (a) PASCAL 部分 (编者略)
 程序 (b) C 部分
 (1) (A!=null && B!=null) // 两均未空时循环
 (2) A->element==B->element // 两表中相等元素不作结果元素
 (3) B=B->link // 向后移动 B 表指针
 (4) A!=null // 将 A 表剩余部分放入结果表中
 (5) last->link=null // 置链表尾

```

#### 四、应用题

1. (1) 选链式存储结构。它可动态申请内存空间, 不受表长度 (即表中元素个数) 的影响, 插入、删除时间复杂度为  $O(1)$ 。

(2) 选顺序存储结构。顺序表可以随机存取, 时间复杂度为  $O(1)$ 。

2. 链式存储结构一般说克服了顺序存储结构的三个弱点。首先, 插入、删除不需移动元素, 只修改指针, 时间复杂度为  $O(1)$ ; 其次, 不需要预先分配空间, 可根据需要动态申请空间; 其三, 表容量只受可用内存空间的限制。其缺点是因为指针增加了空间开销, 当空间不允许时, 就不能克服顺序存储的缺点。

3. 采用链式存储结构, 它根据实际需要申请内存空间, 而当不需要时又可将不用结点空间返还给系统。在链式存储结构中插入和删除操作不需要移动元素。

4. 线性表 栈 队列 串 顺序存储结构和链式存储结构。

顺序存储结构的定义是:

```

CONST maxlen=线性表可能达到的最大长度;
TYPE sqlisttp=RECORD
 elem:ARRAY[1..maxlen] OF ElemType;
 last:0..maxlen;
END;

```

链式存储结构的定义是:

```

TYPE pointer=↑nodetype;
 nodetype=RECORD
 data:ElemType;
 next:pointer;
 END;
 linklisttp=pointer;

```

5. 顺序映射时,  $a_i$  与  $a_{i+1}$  的物理位置相邻; 链表表示时  $a_i$  与  $a_{i+1}$  的物理位置不要求相邻。

6. 在线性表的链式存储结构中, 头指针指链表的指针, 若链表有头结点则是链表的头结点的指针, 头指针具有标识作用, 故常用头指针冠以链表的名字。头结点是为了操作的统一、方便而设立的, 放在第一元素结点之前, 其数据域一般无意义 (当然有些情况下也可存放链表的长度、

用做监视哨等等), 有头结点后, 对在第一元素结点前插入结点和删除第一结点, 其操作与对其它结点的操作统一了。而且无论链表是否为空, 头指针均不为空。首元结点也就是第一元素结点, 它是头结点后边的第一个结点。

7. 见上题 6。

8. (1) 将 next 域变为两个域: pre 和 next, 其值域均为  $0..maxsize$ 。初始化时, 头结点 (下标为 0 的元素) 其 next 域值为 1, 其 pre 域值为 n (设 n 是元素个数, 且  $n < maxsize$ )

(2) `stalist[stalist[p].pre].pre;`

(3) `stalist[p].next;`

9. 在单链表中不能从当前结点 (若当前结点不是第一结点) 出发访问到任何一个结点, 链表只能从头指针开始, 访问到链表中每个结点。在双链表中求前驱和后继都容易, 从当前结点向前到第一结点, 向后到最后结点, 可以访问到任何一个结点。

10. 本题是链表的逆置问题。设该链表带头结点, 将头结点摘下, 并将其指针域置空。然后从第一元素结点开始, 直到最后一个结点为止, 依次前插入头结点的后面, 则实现了链表的逆置。

11. 该算法的功能是判断链表 L 是否是递减有序, 若是则返回 “true”; 否则返回 “false”。

pre 指向当前结点, p 指向 pre 的后继。

12. `q=p->next; p->next=q->next; free(q);`

13. 设单链表的头结点的头指针为 head, 且 `pre=head;`

`while(pre->next!=p) pre=pre->next;`

`s->next=p; pre->next=s;`

14. 设单链表带头结点, 工作指针 p 初始化为 `p=H->next;`

(1) `while(p!=null && p->data!=X) p=p->next;`

`if(p==null) return(null); // 查找失败`

`else return(p); // 查找成功`

(2) `while(p!=null && p->data<X) p=p->next;`

`if(p==null || p->data>X) return(null); // 查找失败`

`else return(p);`

(3) `while(p!=null && p->data>X) p=p->next;`

`if(p==null || p->data<X) return(null); // 查找失败`

`else return(p); // 查找成功`

15. 本程序段功能是将 pa 和 pb 链表中的值相同的结点保留在 pa 链表中 (pa 中与 pb 中不同结点删除), pa 是结果链表的头指针。链表中结点值与从前逆序。S1 记结果链表中结点个数 (即 pa 与 pb 中相等的元素个数)。S2 记原 pa 链表中删除的结点个数。

16. 设 `q:=p^.llink;` 则

`q^.rlink:=p^.rlink; p^.rlink^.llink:=q; p^.llink:=q^.llink;`

`q^.llink^.rlink:=p; p^.rlink:=q; q^.llink:=p`

17. (1) 前两个语句改为:

`p^.llink^.rlink<- p^.rlink;`

`p^.rlink^.llink<- p^.llink;`

(2) 后三个语句序列应改为:

`q^.rlink<- p^.rlink; // 以下三句的顺序不能变`

`p^.rlink^.llink<- q;`

`p^.rlink<- q;`

18. mp 是一个过程, 其内嵌套有过程 subp。

subp(s, q) 的作用是构造从 s 到 q 的循环链表。

subp(pa, pb)调用结果是将 pa 到 pb 的前驱构造为循环链表。

subp(pb, pa)调用结果是将 pb 到 pa 的前驱（指在 L 链表中，并非刚构造的 pa 循环链表中）构造为循环链表。

总之，两次调用将 L 循环链表分解为两个。第一个循环链表包含从 pa 到 pb 的前驱，L 中除刚构造的 pa 到 pb 前驱外的结点形成第二个循环链表。

19. 在指针 p 所指结点前插入结点 s 的语句如下：

```
s->pre=p->pre; s->next=p; p->pre->next=s; p->pre=s;
```

20. (A)  $f1 \neq \text{NIL}$  并且  $f2 \neq \text{NIL}$

(B)  $f1 \uparrow .\text{data} < f2 \uparrow .\text{data}$

(C)  $f2 \uparrow .\text{data} < f1 \uparrow .\text{data}$

(D)  $f3 \uparrow .\text{data} < f1 \uparrow .\text{data}$

(E)  $f1 \leftarrow f1 \uparrow .\text{link}$  或  $f2 = f2 \uparrow .\text{link}$ ;

21. 1) 本算法功能是将双向循环链表结点的数据域按值从小到大排序，成为非递减（可能包括数据域值相等的结点）有序双向循环链表。

2) (1)  $r \rightarrow \text{prior} = q \rightarrow \text{prior}$ ; // 将 q 结点摘下，以便插入到适当位置。

(2)  $p \rightarrow \text{next} \rightarrow \text{prior} = q$ ; // (2) (3) 将 q 结点插入

(3)  $p \rightarrow \text{next} = q$ ;

(4)  $r = r \rightarrow \text{next}$ ; 或  $r = q \rightarrow \text{next}$ ; // 后移指针，再将新结点插入到适当位置。

## 五、 算法设计题

1. [题目分析]因为两链表已按元素值递增次序排列，将其合并时，均从第一个结点起进行比较，将小的链入链表中，同时后移链表工作指针。该问题要求结果链表按元素值递减次序排列。故在合并的同时，将链表结点逆置。

LinkedList Union(LinkedList la, lb)

// la, lb 分别是带头结点的两个单链表的头指针，链表中的元素值按递增序排列，本算法将两链表合并成一个按元素值递减次序排列的单链表。

```
{ pa=la->next; pb=lb->next; // pa, pb 分别是链表 la 和 lb 的工作指针
```

```
la->next=null; // la 作结果链表的头指针，先将结果链表初始化为空。
```

```
while(pa!=null && pb!=null) // 当两链表均不为空时作
```

```
if(pa->data <= pb->data)
```

```
{ r=pa->next; // 将 pa 的后继结点暂存于 r。
```

```
pa->next=la->next; // 将 pa 结点链于结果表中，同时逆置。
```

```
la->next=pa;
```

```
pa=r; // 恢复 pa 为当前待比较结点。
```

```
}
```

```
else
```

```
{r=pb->next; // 将 pb 的后继结点暂存于 r。
```

```
pb->next=la->next; // 将 pb 结点链于结果表中，同时逆置。
```

```
la->next=pb;
```

```
pb=r; // 恢复 pb 为当前待比较结点。
```

```
}
```

```
while(pa!=null) // 将 la 表的剩余部分链入结果表，并逆置。
```

```
{r=pa->next; pa->next=la->next; la->next=pa; pa=r; }
```

```
while(pb!=null)
```



```
{r=pb->next; pb->next=la->next; la->next=pb; pb=r; }
```

} // 算法 Union 结束。

[算法讨论] 上面两链表均不为空的表达式也可简写为 **while**(pa&&pb), 两递增有序表合并成递减有序表时, 上述算法是边合并边逆置。也可先合并完, 再作链表逆置。后者不如前者优化。算法中最后两个 **while** 语句, 不可能执行两个, 只能二者取一, 即哪个表尚未到尾, 就将其逆置到结果表中, 即将剩余结点依次前插入到结果表的头结点后面。

与本题类似的其它题解答如下:

(1) [问题分析] 与上题类似, 不同之处在于: 一是链表无头结点, 为处理方便, 给加上头结点, 处理结束再删除之; 二是数据相同的结点, 不合并到结果链表中; 三是 hb 链表不能被破坏, 即将 hb 的结点合并到结果链表时, 要生成新结点。

```
LinkedList Union(LinkedList ha, hb)
```

// ha 和 hb 是两个无头结点的数据域值递增有序的单链表, 本算法将 hb 中并不出现在 ha 中的数据合并到 ha 中, 合并中不能破坏 hb 链表。

```
{LinkedList la;
```

```
la=(LinkedList)malloc(sizeof(LNode));
```

```
la->next=ha; // 申请头结点, 以便操作。
```

```
pa=ha; // pa 是 ha 链表的工作指针
```

```
pb=hb; // pb 是 hb 链表的工作指针
```

```
pre=la; // pre 指向当前待合并结点的前驱。
```

```
while(pa&&pb)
```

```
if(pa->data<pb->data) // 处理 ha 中数据
```

```
{pre->next=pa;pre=pa;pa=pa->next;}
```

```
else if(pa->data>pb->data) // 处理 hb 中数据。
```

```
{r=(LinkedList)malloc(sizeof(LNode)); // 申请空间
```

```
r->data=pb->data; pre->next=r;
```

```
pre=r; // 将新结点链入结果链表。
```

```
pb=pb->next; // hb 链表中工作指针后移。
```

```
}
```

```
else // 处理 pa->data=pb->data;
```

```
{pre->next=pa; pre=pa;
```

```
pa=pa->next; // 两结点数据相等时, 只将 ha 的数据链入。
```

```
pb=pb->next; // 不要 hb 的相等数据
```

```
}
```

```
if(pa!=null)pre->next=pa; // 将两链表中剩余部分链入结果链表。
```

```
else pre->next=pb;
```

```
free(la); // 释放头结点. ha, hb 指针未被破坏。
```

```
} // 算法 nion 结束。
```

(2) 本题与上面两题类似, 要求结果指针为 lc, 其核心语句段如下:

```
pa=la->next;pb=hb->next;
```

```
lc=(LinkedList)malloc(sizeof(LNode));
```

```
pc=lc; // pc 是结果链表中当前结点的前驱
```

```
while(pa&&pb)
```

```
if(pa->data<pb->data)
```

```
{pc->next=pa;pc=pa;pa=pa->next;}
```

```
else {pc->next=pb;pc=pb;pb=pb->next;}
if(pa)pc->next=pa; else pc->next=pb;
free(la);free(lb); // 释放原来两链表的头结点。
```

算法时间复杂度为  $O(m+n)$ ，其中  $m$  和  $n$  分别为链表  $la$  和  $lb$  的长度。

2. [题目分析]本组题有 6 个，本质上都是链表的合并操作，合并中有各种条件。与前组题不同的是，叙述上是用线性表代表集合，而操作则是求集合的并、交、差 ( $A \cup B$ ,  $A \cap B$ ,  $A - B$ ) 等。

本题与上面 1. (2) 基本相同，不同之处 1. (2) 中链表是“非递减有序”，(可能包含相等元素)，本题是元素“递增有序”(不准有相同元素)。因此两表中合并时，如有元素值相等元素，则应删掉一个。

```
LinkedList Union(LinkedList ha, hb)
```

// 线性表 A 和 B 代表两个集合，以链式存储结构存储，元素递增有序。ha 和 hb 分别是其链表的头指针。本算法求 A 和 B 的并集  $A \cup B$ ，仍用线性表表示，结果链表元素也是递增有序。

```
{ pa=ha->next;pb=hb->next; // 设工作指针 pa 和 pb。
pc=ha; // 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 ha 表未空，则链入结果表。
 else pc->next=pb; // 若 hb 表未空，则链入结果表。
 free(hb); // 释放 hb 头结点
 return(ha);
} // 算法 Union 结束。
```

与本题类似的其它几个题解答如下：

(1) 解答完全同上 2。

(2) 本题是求交集，即只有同时出现在两集合中的元素才出现在结果表中。其核心语句段如下：

```
pa=la->next;pb=lb->next; // 设工作指针 pa 和 pb;
pc=la; // 结果表中当前合并结点的前驱的指针。
while(pa&&pb)
 if(pa->data==pb->data) // 交集并入结果表中。
 { pc->next=pa;pc=pa;pa=pa->next;
 u=pb;pb=pb->next;free(u);}
 else if(pa->data<pb->data) {u=pa;pa=pa->next;free(u);}
 else {u=pb; pb=pb->next; free(u);}
while(pa) { u=pa; pa=pa->next; free(u);} // 释放结点空间
while(pb) { u=pb; pb=pb->next; free(u);} // 释放结点空间
pc->next=null; // 置链表尾标记。
```

free(lb); //注: 本算法中也可对 B 表不作释放空间的处理

(3) 本题基本与 (2) 相同, 但要求无重复元素, 故在算法中, 待合并结点数据要与其前驱比较, 只有在与前驱数据不同时才并入链表。其核心语句段如下。

pa=L1->next; pb=L2->next; // pa、pb 是两链表的工作指针。

pc=L1; // L1 作结果链表的头指针。

while(pa&&pb)

if(pa->data<pb->data) {u=pa; pa=pa->next; free(u);} // 删除 L1 表多余元素

else if (pa->data>pb->data) pb=pb->next; // pb 指针后移

else // 处理交集元素

{if(pc==L1) {pc->next=pa; pc=pa; pa=pa->next;} // 处理第一个相等的元素。

else if(pc->data==pa->data) { u=pa; pa=pa->next; free(u);} // 重复元素不进入 L1 表。

else { pc->next=pa; pc=pa; pa=pa->next;} // 交集元素并入结果表。

} // while

while(pa) {u=pa; pa=pa->next; free(u);} // 删 L1 表剩余元素

pc->next=null; // 置结果链表尾。

注: 本算法中对 L2 表未作释放空间的处理。

(4) 本题与上面 (3) 算法相同, 只是结果表要另辟空间。

(5) [题目分析] 本题首先求 B 和 C 的交集, 即求 B 和 C 中共有元素, 再与 A 求并集, 同时删除重复元素, 以保持结果 A 递增。

LinkedList union(LinkedList A, B, C)

// A, B 和 C 均是带头结点的递增有序的单链表, 本算法实现  $A = A \cup (B \cap C)$ , 使求解结构保持递增有序。

{pa=A->next; pb=B->next; pc=C->next; // 设置三个工作指针。

pre=A; // pre 指向结果链表中当前待合并结点的前驱。

if(pa->data<pb->data || pa->data<pc->data) // A 中第一个元素为结果表的第一元素。

{pre->next=pa; pre=pa; pa=pa->next;}

else {while(pb&&pc) // 找 B 表和 C 表中第一个公共元素。

if(pb->data<pc->data) pb=pb->next;

else if(pb->data>pc->data) pc=pc->next;

else break; // 找到 B 表和 C 表的共同元素就退出 while 循环。

if(pb&&pc) // 因共同元素而非 B 表或 C 表空而退出上面 while 循环。

if(pa->data>pb->data) // A 表当前元素值大于 B 表和 C 表的公共元素, 先将 B 表元素链入。

{pre->next=pb; pre=pb; pb=pb->next; pc=pc->next;} // B, C 公共元素为结果表第一元素。

} // 结束了结果表中第一元素的确定

while(pa&&pb&&pc)

{while(pb&&pc)

if(pb->data<pc->data) pb=pb->next;

else if(pb->data>pc->data) pc=pc->next;

else break; // B 表和 C 表有公共元素。

if(pb&&pc)

{while(pa&&pa->data<pb->data) // 先将 A 中小于 B, C 公共元素部分链入。

```

 {pre->next=pa;pre=pa;pa=pa->next;}
 if(pre->data!=pb->data){pre->next=pb;pre=pb;pb=pb->next;pc=pc->next;}
 else{pb=pb->next;pc=pc->next;} // 若 A 中已有 B, C 公共元素, 则不再存入结果表。
}
} // while(pa&&pb&&pc)
if(pa) pre->next=pa; // 当 B, C 无公共元素 (即一个表已空), 将 A 中剩余链入。
} // 算法 Union 结束

```

[算法讨论]本算法先找结果链表的第一个元素, 这是因为题目要求结果表要递增有序 (即删除重复元素)。这就要求当前待合并到结果表的元素要与其前驱比较。由于初始  $pre=A$  (头结点的头指针), 这时的  $data$  域无意义, 不能与后继比较元素大小, 因此就需要确定第一个元素。当然, 不要这样作, 而直接进入下面循环也可以, 但在链入结点时, 必须先判断  $pre$  是否等于  $A$ , 这占用了过多的时间。因此先将第一结点链入是可取的。

算法中的第二个问题是要求时间复杂度为  $O(|A|+|B|+|C|)$ 。这就要求各个表的工作指针只能后移 (即不能每次都从头指针开始查找)。本算法满足这一要求。

最后一个是问题, 当  $B, C$  有一表为空 (即  $B$  和  $C$  已无公共元素时), 要将  $A$  的剩余部分链入结果表。

3. [题目分析]循环单链表  $L1$  和  $L2$  数据结点个数分别为  $m$  和  $n$ , 将二者合成一个循环单链表时, 需要将一个循环链表的结点 (从第一元素结点到最后一个结点) 插入到另一循环链表的第一元素结点前即可。题目要求 “用最快速度将两表合并”, 因此应找结点个数少的链表查其尾结点。

```

LinkedList Union(LinkedList L1,L2;int m,n)
//L1 和 L2 分别是两循环单链表的头结点的指针, m 和 n 分别是 L1 和 L2 的长度。
// 本算法用最快速度将 L1 和 L2 合并成一个循环单链表。
{if(m<0||n<0){printf("表长输入错误\n"); exit(0);}
if(m<n) // 若 m<n, 则查 L1 循环单链表的最后一个结点。
{if(m==0)return(L2); // L1 为空表。
else{p=L1;
while(p->next!=L1) p=p->next; // 查最后一个元素结点。
p->next=L2->next; // 将 L1 循环单链表的元素结点插入到 L2 的第一元素结点前。
L2->next=L1->next;
free(L1); // 释放无用头结点。
}
} // 处理完 m<n 情况
else // 下面处理 L2 长度小于等于 L1 的情况
{if(n==0)return(L1); // L2 为空表。
else{p=L2;
while(p->next!=L2) p=p->next; // 查最后元素结点。
p->next=L1->next; // 将 L2 的元素结点插入到 L1 循环单链表的第一元素结点前。
L1->next=L2->next;
free(L2); // 释放无用头结点。
}
} // 算法结束。
}

```

类似本题叙述的其它题解答如下:

(1) [题目分析] 本题将线性表 1a 和 1b 连接, 要求时间复杂度为  $O(1)$ , 且占用辅助空间尽量小。应该使用只设尾指针的单循环链表。

LinkedList Union(LinkedList 1a, 1b)

// 1a 和 1b 是两个无头结点的循环单链表的尾指针, 本算法将 1b 接在 1a 后, 成为一个单循环链表。

```
{ q=1a->next; //q 指向 1a 的第一个元素结点。
 1a->next=1b->next; //将 1b 的最后元素结点接到 1b 的第一元素。
 1b->next=q; //将 1b 指向 1a 的第一元素结点, 实现了 1b 接在 1a 后。
 return(1b); //返回结果单循环链表的尾指针 1b。
} // 算法结束。
```

[算法讨论] 若循环单链表带有头结点, 则相应算法片段如下:

```
q=1b->next; //q 指向 1b 的头结点;
1b->next=1a->next; //1b 的后继结点为 1a 的头结点。
1a->next=q->next; //1a 的后继结点为 1b 的第一元素结点。
free(q); //释放 1b 的头结点
return(1b); //返回结果单循环链表的尾指针 1b。
```

(2) [题目分析] 本题要求将单向链表 ha 和单向循环链表 hb 合并成一个单向链表, 要求算法所需时间与链表长度无关, 只有使用带尾指针的循环单链表, 这样最容易找到链表的首、尾结点, 将该结点序列插入到单向链表第一元素之前即可。

其核心算法片段如下 (设两链表均有头结点)

```
q=hb->next; //单向循环链表的表头指针
hb->next=ha->next; //将循环单链表最后元素结点接在 ha 第一元素前。
ha->next=q->next; //将指向原单链表第一元素的指针指向循环单链表第一结点
free(q); //释放循环链表头结点。
```

若两链表均不带头结点, 则算法片段如下:

```
q=hb->next; //q 指向 hb 首元结点。
hb->next=ha; //hb 尾结点的后继是 ha 第一元素结点。
ha=q; //头指针指向 hb 的首元结点。
```

4. [题目分析] 顺序存储结构的线性表的插入, 其时间复杂度为  $O(n)$ , 平均移动近一半的元素。线性表 LA 和 LB 合并时, 若从第一个元素开始, 一定会造成元素后移, 这不符合本题“高效算法”的要求。另外, 题中叙述“线性表空间足够大”也暗示出另外合并方式, 即应从线性表的最后一个元素开始比较, 大者放到最终位置上。设两线性表的长度各为 m 和 n, 则结果表的最后一个元素应在 m+n 位置上。这样从后向前, 直到第一个元素为止。

PROC Union(VAR LA:SeqList;LB:SeqList)

//LA 和 LB 是顺序存储的非递减有序线性表, 本算法将 LB 合并到 LA 中, 元素仍非递减有序。

m:=LA.last;n:=LB.last; //m, n 分别为线性表 LA 和 LB 的长度。

k:=m+n; //k 为结果线性表的工作指针 (下标)。

i:=m;j:=n; //i, j 分别为线性表 LA 和 LB 的工作指针 (下标)。

WHILE (i>0) AND (j>0) DO

IF LA.elem[i]>=LB.elem[j]

THEN [LA.elem[k]:=LA.elem[i];k:=k-1;i:=i-1;]

ELSE [LA.elem[k]:=LB.elem[j];k:=k-1;j:=j-1;]

WHILE (j>0) DO [LA.elem[k]:=LB.elem[j];k:=k-1;j:=j-1;]

LA.last:=m+n;

ENDP;

[算法讨论]算法中数据移动是主要操作。在最佳情况下(LB的最小元素大于LA的最大元素), 仅将LB的 $n$ 个元素移(拷贝)到LA中, 时间复杂度为 $O(n)$ , 最差情况, LA的所有元素都要移动, 时间复杂度为 $O(m+n)$ 。因数据合并到LA中, 所以在退出第一个**WHILE**循环后, 只需要一个**WHILE**循环, 处理LB中剩余元素。第二个循环只有在LB有剩余元素时才执行, 而在LA有剩余元素时不执行。本算法利用了题目中“线性表空间足够大”的条件, “最大限度的避免移动元素”, 是“一种高效算法”。

5. [题目分析]本题实质上是一个排序问题, 要求“不得使用除该链表结点以外的任何链结点空间”。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。

LinkedList LinkListSort(LinkedList list)

//list 是不带头结点的线性链表, 链表结点构造为 data 和 link 两个域, data 是数据域, link 是指针域。本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。

```
{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。
}
}
```

[算法讨论]算法时间复杂度的分析与用顺序存储结构时的情况相同。但顺序存储结构将第 $i$  ( $i>1$ ) 个元素插入到前面第1至第 $i-1$ 个元素的有序表时, 是将第 $i$ 个元素先与第 $i-1$ 个元素比较。而在链表最佳情况均是和第一元素比较。两种存储结构下最佳和最差情况的比较次数相同, 在链表情况下, 不移动元素, 而是修改结点指针。

另一说明是, 本题中线性链表 list 不带头结点, 而且要求“不得使用除该链表以外的任何链结点空间”, 所以处理复杂, 需要考虑当前结点元素值比有序链表第一结点的元素值还小的情况, 这时要修改链表指针 list。如果 list 是头结点的指针, 则相应处理要简单些, 其算法片段如下:

```
p=list->link; //p 指向第一元素结点。
list->link=null; //有序链表初始化为空
while(p!=null)
{r=p->link; //保存后继
q=list;
while(q->link!=null && q->link->data<p->data)q=q->link;
p->link=q->link;
```

```
q->link=p;
q=r;
}
```

6. [题目分析] 本题明确指出单链表带头结点，其结点数据是正整数且不相同，要求利用直接插入原则把链表整理成递增有序链表。这就要求从第二结点开释，将各结点依次插入到有序链表中。

LinkedList LinkedListInsertSort(LinkedList la)

// la 是带头结点的单链表，其数据域是正整数。本算法利用直接插入原则将链表整理成递增的有序链表。

```
{if(la->next!=null) // 链表不为空表。
{p=la->next->next; // p 指向第一结点的后继。
la->next->next=null; // 直接插入原则认为第一元素有序，然后从第二元素起依次插入。
while(p!=null)
{r=p->next; // 暂存 p 的后继。
q=la;
while(q->next!=null&&q->next->data<p->data) q=q->next; // 查找插入位置。
p->next=q->next; // 将 p 结点链入链表。
q->next=p;
p=r;
}
```

与本题有类似叙述的题的解答：

(1) 本题也是链表排序问题，虽没象上题那样明确要求“利用直接插入的原则”来排序，仍可用上述算法求解，这里不再赘述。

7. [题目分析] 本题要求将一个链表分解成两个链表，两个链表都要有序，两链表建立过程中不得使用 NEW 过程申请空间，这就是要利用原链表空间，随着原链表的分解，新建链表随之排序。

PROC discreat (VAR listhead, P, Q: linkedlist)

// listhead 是单链表的头指针，链表中每个结点由一个整数域 DATA 和指针域 NEXT 组成。本算法将链表 listhead 分解成奇数链表和偶数链表，分解由 P 和 Q 指向，且 P 和 Q 链表是有序的。

P:=NIL; Q:=NIL; // P 和 Q 链表初始化为空表。

```
s:=listhead;
WHILE (s<>NIL) DO
[r:=s^.NEXT; // 暂存 s 的后继。
IF s^.DATA DIV 2=0 // 处理偶数。
THEN IF P=NIL THEN [P:=s; P^.NEXT:=NIL;] // 第一个偶数链结点。
ELSE [pre:=P;
IF pre^.DATA>s^.DATA THEN [s^.NEXT:=pre; P:=s; // 插入当前最小值结点修改
头指针]
ELSE [WHILE pre^.NEXT<>NIL DO
IF pre^.NEXT^.DATA<s^.DATA THEN pre:=pre^.NEXT; // 查找插入位
置。
s^.NEXT:=pre^.NEXT; // 链入此结点。
pre^.NEXT:=s;
]
]
```

```

ELSE//处理奇数链。
 IF Q=NIL THEN[Q:=s;Q^.NEXT:=NIL;] //第一奇数链结点。
 ELSE[pre:=Q;
 IF pre^.DATA>s^.DATA THEN[s^.NEXT:=pre; Q:=s;]//修改头指针。
 ELSE[WHILE pre^.NEXT<>NIL DO //查找插入位置。
 IF pre^.NEXT^.DATA<s^.DATA THEN pre:=pre^.NEXT;
 s^.NEXT:=pre^.NEXT; //链入此结点。
 pre^.NEXT:=s;
]
]//结束奇数链结点
 s:=r; //s 指向新的待排序结点。
]//结束“WHILE(s<>NIL)DO”
ENDP; //结束整个算法。

```

[算法讨论]由于算法要求“不得使用 NEW 过程申请空间，也没明确指出链表具有头结点，所以上述算法复杂些，它可能需要在第一个结点前插入新结点，即链表的头指针会发生变化。如有头结点，算法不必单独处理在第一个结点前插入结点情况，算法会规范统一，下面的(1)是处理带头结点的例子。算法中偶数链上结点是靠数据整除 2 等于 0 (DATA DIV 2=0) 判断的。

类似本题的其它题解答如下：

(1) [题目分析]本题基本类似于上面第 7 题，不同之处有二。一是带头结点，二是分解后的两个链表，一个是数据值小于 0，另一个是数据值大于 0。由于没明确要求用类 PASCAL 书写算法，故用 C 书写如下。

```
void DisCreat1(LinkedList A)
```

//A 是带头结点的单链表，链表中结点的数据类型为整型。本算法将 A 分解成两个单链表 B 和 C，B 中结点的数据小于零，C 中结点的数据大于零。

```

{B=A;
C=(LinkedList)malloc(sizeof(LNode)); //为 C 申请结点空间。
C->next=null //C 初始化为空表。
p=A->next; //p 为工作指针。
B->next=null; //B 表初始化。
while(p!=null)
{r=p->next; //暂存 p 的后继。
if (p->data<0) //小于 0 的放入 B 表。
{p->next=B->next; B->next=p; } //将小于 0 的结点链入 B 表。
else {p->next=C->next; C->next=p; }
p=r; //p 指向新的待处理结点。
}
} //算法结束。

```

[算法讨论]因为本题并未要求链表中结点的数据值有序，所以算法中采取最简单方式：将新结点前插到头结点后面（即第一元素之前）。

(2) 本题同上面第 7 题，除个别叙述不同外，本质上完全相同，故不再另作解答。

(3) [题目分析]本题中的链表有头结点，分解成表 A 和表 B，均带头结点。分解后的 A 表含有原表中序号为奇数的元素，B 表含有原 A 表中序号为偶数的元素。由于要求分解后两表中元素结点的相对顺序不变，故采用在链表尾插入比较方便，这使用一指向表尾的指针即可方便实现。

```
void DisCreat3(LinkedList A)
```



//A 是带头结点的单链表, 本算法将其分解成两个带头结点的单链表, A 表中含原表中序号为奇数

//的结点, B 表中含原表中序号为偶数的结点。链表中结点的相对顺序同原链表。

{i=0; // i 记链表中结点的序号。

B=(LinkedList)malloc(sizeof(LNode)); // 创建 B 表表头。

B->next=null; // B 表的初始化。

LinkedList ra,rb; // ra 和 rb 将分别指向将创建的 A 表和 B 表的尾结点。

ra=A;rb=B;

p=A->next; // p 为链表工作指针, 指向待分解的结点。

A->next=null; // 置空新的 A 表

while(p!=null)

{r=p->next; // 暂存 p 的后继。

i++;

if(i%2==0) // 处理原序号为偶数的链表结点。

{p->next=rb->next; // 在 B 表尾插入新结点;

rb->next=p; rb=p; // rb 指向新的尾结点;

}

else // 处理原序号为奇数的结点。

{p->next=ra->next; ra->next=p; ra=p; }

p=r; // 将 p 恢复为指向新的待处理结点。

} // 算法结束

8. [题目分析] 题目要求重排  $n$  个元素且以顺序存储结构存储的线性表, 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。

int Rearrange (SeqList 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) {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。

}

[算法讨论] 本算法时间复杂度为  $O(n)$ 。算法只是按题目要求把正负数分开, 如要求统计负数和大于等于零的个数, 则最后以  $t$  来定。如  $t$  为负数, 则 0 至  $i$  共  $i+1$  个负数,  $n-1-i$  个正数 (包括零)。另外, 题目并未提及零的问题, 笔者将零放到正数一边。对此问题的扩充是若元素包含正数、负数和零, 并要求按负数、零、正数的顺序重排线性表, 统计负数、零、正数的个数。请读者利用上面解题思想自行解答。

类似本题的选了 5 个题, 其解答如下:

(1) 与上面第 8 题不同的是, 这里要求以  $a_n$  为参考元素, 将线性表分成左右两部分。左半部分的元素都小于等于  $a_n$ , 右半部分的元素都大于  $a_n$ ,  $a_n$  位于分界位置上。其算法主要片段语句如下:

```
i=1; j=n;
t=a[n]; // 暂存参考元素。
while(i<j)
{
 while(i<j && a[i]<=t) i++; // 当前元素不大于参考元素时, 指针 i 后移。
 if(i<j) a[j--]=a[i]; // 将大于参考元素的元素后移。
 while(i<j && a[j]>t) j--; // 当前元素大于参考元素时指针前移。
 if(i<j) a[i++]=a[j]; // 将小于参考元素的当前元素前移。
}
a[i]=t; // 参考元素置于分界位置。
```

(2) [题目分析] 本题要求将线性表 A 分成 B 和 C 两个表, 表 B 和表 C 不另占空间, 而是利用表 A 的空间, 其算法与第 8 题相同。这里仅把表 B 和表 C 另设空间的算法解答如下:

```
void Rearrange2(int A[], B[], C[])
// 线性表 A 有 n 个整型元素, 顺序存储。本算法将 A 拆成 B 和 C 两个表, B 中存放大于
// 等于零的元素, C 中存放小于零的元素。
{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<n)
 {
 if(A[i]<0) C[++k]=A[i++]; // 将小于零的元素放入 C 表。
 else B[++j]=A[i++]; // 将大于零的元素放入 B 表。
```

[算法讨论] 本题用一维数组存储线性表, 结果线性表 B 和 C 中分别有  $j+1$  和  $k+1$  个元素。若采用教材中的线性表, 则元素的表示作相应改变, 例如  $A.elem[i]$ , 而最后 B 和 C 表应置上表的长度, 如  $B.length=j$  和  $C.length=k$ 。

(3) 本题与第 8 题本质上相同, 第 8 题要求分开正数和负数, 这里要求分开奇数和偶数, 判别方式是  $a[i]\%2==0$ , 满足时为偶数, 反之为奇数。

(4) 本题与第 8 题相同, 只是叙述不同。

(5) 本题与第 8 题基本相同, 不同之处在于这里的分界元素是整数 19 (链表中并不要求一定有 19)。本题要求用标准 pascal 描述算法, 如下所示。

```
TYPE arr=ARRAY[1..1000] OF integer;
VAR a: arr;
PROCEDURE Rearrange5 (VAR a: arr);
// a 是 n (设 n=1000) 个整数组成的线性表, 用一维数组存储。本算法将 n 个元素中所有
// 大于等于 19 的整数放在所有小于 19 的整数之后。
VAR i, j, t: integer;
BEGIN
 i:=1; j:=n; t:=a[1]; // i, j 指示顺序表的首尾元素的下标, t 暂存分界元素
 WHILE (i<j) DO
 BEGIN
 WHILE (i<j) AND (a[j]>=19) DO j:=j-1;
 IF (i<j) THEN BEGIN A[i]:=a[j]; i:=i+1 END;
 WHILE (i<j) AND (a[i]<19) DO i:=i+1;
 IF (i<j) THEN BEGIN A[j]:=a[i]; j:=j-1 END;
```

```

 END;
 a[i]:=t;
END;

```

[算法讨论] 分界元素  $t$  放入  $a[i]$ ，而不论它的值如何。算法中只用了一个  $t$  中间变量，符合空间复杂度  $O(1)$  的要求。算法也满足时间复杂度  $O(n)$  的要求。

9. [题目分析] 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。

```
LinkedList Delete (LinkedList 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) {pre=p; q=p->next; } //查最小值结点
```

```
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
```

```
} //结束算法 delete。
```

[算法讨论] 算法中函数头是按本教材类 C 描述语言书写的。原题中 `void delete(linklist &L)`，是按 C++ 的“引用”来写的，目的是实现变量的“传址”，克服了 C 语言函数传递只是“值传递”的缺点。

10. [题目分析] 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），再插入到链表的最前面。

```
LinkedList delinsert (LinkedList list)
```

//list 是非空线性链表，链结点结构是 (data, link)，data 是数据域，link 是链域。

//本算法将链表中数据域值最小的那个结点移到链表的最前面。

```
{p=list->link; //p 是链表的工作指针
```

```
pre=list; //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->link!=null)
```

```
{if (p->link->data<q->data) {pre=p; q=p->link; } //找到新的最小值结点;
```

```
p=p->link;
```

```
}
```

```
if (q!=list->link) //若最小值是第一元素结点，则不需再操作
```

```
{pre->link=q->link; //将最小值结点从链表上摘下;
```

```
q->link= list->link; //将 q 结点插到链表最前面。
```

```
list->link=q;
```

```
}
```

```
} //算法结束
```

[算法讨论] 算法中假定 list 带有头结点，否则，插入操作变为 `q->link=list; list=q`。

11. [题目分析] 知道双向循环链表中的一个结点，与前驱交换涉及到四个结点（p 结点，

前驱结点, 前驱的前驱结点, 后继结点) 六条链。

```
void Exchange (LinkedList p)
//p 是双向循环链表中的一个结点, 本算法将 p 所指结点与其前驱结点交换。
{q=p->llink;
 q->llink->rlink=p; //p 的前驱的前驱之后继为 p
 p->llink=q->llink; //p 的前驱指向其前驱的前驱。
 q->rlink=p->rlink; //p 的前驱的后继为 p 的后继。
 q->llink=p; //p 与其前驱交换
 p->rlink->llink=q; //p 的后继的前驱指向原 p 的前驱
 p->rlink=q; //p 的后继指向其原来的前驱
} // 算法 exchange 结束。
```

12. [题目分析] 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。

```
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
 {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。
```

[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。第三, 本算法可以写成三个函数, 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, 易读。

13. [题目分析] 判断链表中数据是否中心对称, 通常使用栈。将链表的前一半元素依次进栈。在处理链表的另一半元素时, 当访问到链表的一个元素后, 就从栈中弹出一个元素, 两元素比较, 若相等, 则将链表中下一元素与栈中再弹出元素比较, 直至链表到尾。这时若栈是空栈, 则得出链表中心对称的结论; 否则, 当链表中一元素与栈中弹出元素不等时, 结论为链表非中心对称, 结束算法的执行。

```
int dc (LinkedList h, int n)
// h 是带头结点的 n 个元素单链表, 链表中结点的数据域是字符。本算法判断链表是否是
// 中心对称。
{char s[]; int i=1; //i 记结点个数, s 字符栈
```

```
p=h->next; //p 是链表的工作指针，指向待处理的当前元素。
for (i=1; i<=n/2; i++) // 链表前一半元素进栈。
{ s[i]=p->data; p=p->next; }
i--; //恢复最后的 i 值
if (n%2==1) p=p->next; } // 若 n 是奇数，后移过中心结点。
while (p!=null && s[i]==p->data) { i--; p=p->next; } // 测试是否中心对称。
if (p==null) return (1); // 链表中心对称
else return (0); // 链表不中心对称
} // 算法结束。
```

[算法讨论] 算法中先将“链表的前一半”元素（字符）进栈。当  $n$  为偶数时，前一半和后一半的个数相同；当  $n$  为奇数时，链表中心结点字符不必比较，移动链表指针到下一字符开始比较。比较过程中遇到不相等时，立即退出 **while** 循环，不再进行比较。

14. [题目分析] 在单链表中删除自第  $i$  个元素起的共  $len$  个元素，应从第 1 个元素起开始计数，记到第  $i$  个时开始数  $len$  个，然后将第  $i-1$  个元素的后继指针指向第  $i+len$  个结点，实现了在 A 链表中删除自第  $i$  个起的  $len$  个结点。这时应继续查到 A 的尾结点，得到删除元素后的 A 链表。再查 B 链表的第  $j$  个元素，将 A 链表插入之。插入和删除中应注意前驱后继关系，不能使链表“断链”。另外，算法中应判断  $i$ ， $len$  和  $j$  的合法性。

```
LinkedList DelInsert (LinkedList heada, headb, int i, j, len)
```

//heada 和 headb 均是带头结点的单链表。本算法删除 heada 链表中自第  $i$  个元素起的共  $len$  个元素，然后将单链表 heada 插入到 headb 的第  $j$  个元素之前。

```
{ if (i<1 || len<1 || j<1) { printf ("参数错误\n"); exit (0); } // 参数错，退出算法。
p=heada; //p 为链表 A 的工作指针，初始化为 A 的头指针，查到第 i 个元素时，p 指向第 i-1 个元素
k=0; // 计数
while (p!=null && k<i-1) // 查找第 i 个结点。
{ k++; p=p->next; }
if (p==null) { printf ("给的%d 太大\n", i); exit (0); } // i 太大，退出算法
q=p->next; // q 为工作指针，初始指向 A 链表第一个被删结点。
k=0;
while (q!=null && k<len) { k++; u=q, q=q->next; free (u); } // 删除结点，后移指针。
if (k<len) { printf ("给的%d 太大\n", len); exit (0); }
p->next=q; // A 链表删除了 len 个元素。
if (heada->next!=null) // heada->next=null 说明链表中结点均已删除，无需往 B 表插入
{ while (p->next!=null) p= p->next; // 找 A 的尾结点。
q=headb; // q 为链表 B 的工作指针。
k=0; // 计数
while (q!=null && k<j-1) // 查找第 j 个结点。
{ k++; q= q->next; } // 查找成功时，q 指向第 j-1 个结点
if (q==null) { printf ("给的%d 太大\n", j); exit (0); }
p->next=q->next; // 将 A 链表链入
q->next=heada->next; // A 的第一元素结点链在 B 的第 j-1 个结点之后
} // if
free (heada); // 释放 A 表头结点。
```

} // 算法结束。

与本题类似的题的解答如下：

(1) 本题与第 14 题基本相同，不同之处仅在于插入 B 链表第 j 个元素之前的，不是删除了 len 个元素的 A 链表，而是被删除的 len 个元素。按照上题，这 len 个元素结点中第一个结点的指针 p→next，查找从第 i 个结点开始的第 len 个结点的算法修改为：

```
k=1; q=p->next; //q 指向第一个被删除结点
while (q!=null && k<len) // 查找成功时，q 指向自 i 起的第 len 个结点。
{ k++; q= q->next; }
if (k<len) {printf ("给的%d 太大\n", len); exit (0); }
```

15. [题目分析] 在递增有序的顺序表中插入一个元素 x，首先应查找待插入元素的位置。因顺序表元素递增有序，采用折半查找法比顺序查找效率要高。查到插入位置后，从此位置直到线性表尾依次向后移动一个元素位置，之后将元素 x 插入即可。

```
void Insert (ElemType A[], int size, ElemType x)
```

// A 是有 size 个元素空间目前仅有 num (num<size) 个元素的线性表。本算法将元素 x 插入到线性表中，并保持线性表的有序性。

```
{ low=1; high=num; // 题目要求下标从 1 开始
while (low<=high) // 对分查找元素 x 的插入位置。
{ mid= (low+high) /2;
 if (A[mid]==x) {low=mid+1; break; }
 else if (A[mid]>x) high=mid-1 ; else low=mid+1 ;
}
for (i=num; i>=low; i--) A[i+1]=A[i]; // 元素后移。
A[i+1]=x; // 将元素 x 插入。
} // 算法结束。
```

[算法讨论] 算法中当查找失败（即线性表中无元素 x）时，变量 low 在变量 high 的右面（low=high+1）。移动元素从 low 开始，直到 num 为止。特别注意不能写成 for (i=low; i<=num; i++) A[i+1]=A[i]，这是一些学生容易犯的错误。另外，题中未说明若表中已有值为 x 的元素时不再插入，故安排在 A[mid]==x 时，用 low (=mid+1) 记住位置，以便后面统一处理。查找算法时间复杂度为  $O(\log n)$ ，而插入时的移动操作时间复杂度为  $O(n)$ ，若用顺序查找，则查找的时间复杂度亦为  $O(n)$ 。

类似本题的其它题的解答：

(1) [题目分析] 本题与上面 15 题类似，不同之处是给出具体元素值，且让编写 turbo pascal 程序，程序如下：

```
PROGRAM example (input, output);
TYPE pointer=^node;
 node=RECORD
 data: integer;
 next: pointer;
 END;
VAR head, q: pointer;
PROCEDURE create (VAR la: pointer);
VAR x: integer;
 p, q: pointer;
BEGIN
```

```

new (la); la^.next:=NIL; {建立头结点。}
read (x); q:=la; {q 用以指向表尾。}
WHILE NOT EOF DO {建立链表}
 BEGIN
 new (p); p^.data:=x; p^.next:=q^.next; q^.next:=p; q:=p; read(x);
 END;
END;

PROCEDURE insert (VAR la: pointer; s: pointer);
VAR p,q: pointer; found: boolean;
BEGIN
 p:= la^.next; {p 为工作指针。}
 q:=la; {q 为 p 的前驱指针。}
 found:=false;
 WHILE (p<>NIL) AND NOT found
 IF (p^.data<x) THEN BEGIN q:=p; p:= p^.next; END
 ELSE found:=true;
 s^.next:=p; {将 s 结点插入链表}
 q^.next:=s;
END;
BEGIN {main}
 writeln (“请按顺序输入数据，建立链表”)
 create (head);
 writeln (“请输入插入数据”)
 new (q);
 readln (q^.data);
 insert (head, q);
END. {程序结束}

```

[程序讨论] 在建立链表时，输入数据依次为 12, 13, 21, 24, 28, 30, 42，键入 CTRL-Z，输入结束。“插入数据”输 26 即可。本题编写的是完整的 pascal 程序。

16. [题目分析] 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。

```
void StoDouble (LinkedList la)
```

//la 是结点含有 pre, data, link 三个域的单循环链表。其中 data 为数据域, pre 为空指针域, link 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->link->pre==null)
 {la->link->pre=la; //将结点 la 后继的 pre 指针指向 la。
 la=la->link; //la 指针后移。
 }
} //算法结束。

```

[算法讨论] 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。

17. [题目分析] 求两个集合 A 和 B 的差集 A-B，即在 A 中删除 A 和 B 中共有的元素。由于集合用单链表存储，问题变成删除链表中的结点问题。因此，要记住被删除结点的前驱，以便顺利删除被删结点。两链表均从第一元素结点开始，直到其中一个链表到尾为止。



```
void Difference (LinkedList A, B, *n)
```

//A 和 B 均是带头结点的递增有序的单链表，分别存储了一个集合，本算法求两集合的差集，存储于单链表 A 中，\*n 是结果集合中元素个数，调用时为 0

```
{p=A->next; // p 和 q 分别是链表 A 和 B 的工作指针。
q=B->next; pre=A; // pre 为 A 中 p 所指结点的前驱结点的指针。
while (p!=null && q!=null)
 if (p->data<q->data) {pre=p; p=p->next; *n++; } // A 链表中当前结点指针后移。
 else if (p->data>q->data) q=q->next; // B 链表中当前结点指针后移。
 else {pre->next=p->next; // 处理 A, B 中元素值相同的结点，应删除。
 u=p; p=p->next; free (u); } // 删除结点
```

18. [题目分析] 本题要求对单链表结点的元素值进行运算，判断元素值是否等于其序号的平方减去其前驱的值。这里主要技术问题是结点的序号和前驱及后继指针的正确指向。

```
int Judge (LinkedList la)
// la 是结点的元素为整数的单链表。本算法判断从第二结点开始，每个元素值是否等于其序号
// 的平方减去其前驱的值，如是返回 true；否则，返回 false。
{p=la->next->next; // p 是工作指针，初始指向链表的第二项。
pre=la->next; // pre 是 p 所指结点的前驱指针。
i=2; // i 是 la 链表中结点的序号，初始值为 2。
while (p!=null)
 if (p->data==i*i-pre->data) {i++; pre=p; p=p->next; } // 结点值间的关系符合题目要求
 else break; // 当前结点的值不等于其序号的平方减去前驱的值。
if (p!=null) return (false); // 未查到表尾就结束了。
else return (true); // 成功返回。
} // 算法结束。
```

[算法讨论] 本题不设头结点也无影响。另外，算法中还可节省前驱指针 pre，其算法片段如下：

```
p=la; // 假设无头结点，初始 p 指向第一元素结点。
i=2;
while (p->next!=null) // 初始 p->next 指向第二项。
 if (p->next->data==i*i-p->data)
 {i++; p=p->next; }
 if (p->next!=null) return (false); // 失败
 else return (true); // 成功
```

19. [题目分析] 本题实质上是一个模式匹配问题，这里匹配的元素是整数而不是字符。因两整数序列已存入两个链表中，操作从两链表的第一个结点开始，若对应数据相等，则后移指针；若对应数据不等，则 A 链表从上次开始比较结点的后继开始，B 链表仍从第一结点开始比较，直到 B 链表到尾表示匹配成功。A 链表到尾 B 链表未到尾表示失败。操作中应记住 A 链表每次的开始结点，以便下趟匹配时好从其后继开始。

```
int Pattern (LinkedList A, B)
```

//A 和 B 分别是数据域为整数的单链表，本算法判断 B 是否是 A 的子序列。如是，返回 1；否则，返回 0 表示失败。

```
{p=A; // p 为 A 链表的工作指针，本题假定 A 和 B 均无头结点。
```



```
pre=p; //pre 记住每趟比较中 A 链表的开始结点。
q=B; //q 是 B 链表的工作指针。
while (p && q)
 if (p->data==q->data) {p=p->next; q=q->next; }
 else{pre=pre->next; p=pre; //A 链表新的开始比较结点。
 q=B; } //q 从 B 链表第一结点开始。
if (q==null) return (1); //B 是 A 的子序列。
else return (0); //B 不是 A 的子序列。
} //算法结束。
```

20. [题目分析] 本题也是模式匹配问题，应先找出链表 L2 在链表 L1 中的出现，然后将 L1 中的 L2 倒置过来。设 L2 在 L1 中出现时第一个字母结点的前驱的指针为 p，最后一个字母结点在 L1 中为 q 所指结点的前驱，则在保存 p 后继结点指针(s)的情况下，执行 p->next=q。之后将 s 到 q 结点的前驱依次插入到 p 结点之后，实现了 L2 在 L1 中的倒置。

LinkedList PatternInvert (LinkedList L1, L2)

//L1 和 L2 均是带头结点的单链表，数据结点的数据域均为一个字符。本算法将 L1 中与 L2 中数据域相同的连续结点的顺序完全倒置过来。

```
{p=L1; //p 是每趟匹配时 L1 中的起始结点前驱的指针。
q=L1->next; //q 是 L1 中的工作指针。
s=L2->next; //s 是 L2 中的工作指针。
while (p!=null && s!=null)
 if (q->data==s->data) {q=q->next; s=s->next;} //对应字母相等，指针后移。
 else {p=p->next; q=p->next; s=L2->next;} //失配时，L1 起始结点后移，L2 从首
 结点开始。
 if (s==null) //匹配成功，这时 p 为 L1 中与 L2 中首字母结点相同数据域结点的前驱，q 为 L1
 中与 L2 最后一个结点相同数据域结点的后继。
 {r=p->next; //r 为 L1 的工作指针，初始指向匹配的首字母结点。
 p->next=q; //将 p 与 q 结点的链接。
 while (r!=q); //逐结点倒置。
 {s=r->next; //暂存 r 的后继。
 r->next=p->next; //将 r 所指结点倒置。
 p->next=r;
 r=s; //恢复 r 为当前结点。
 }
 else printf ("L2 并未在 L1 中出现");
} //算法结束。
```

[算法讨论] 本算法只讨论了 L2 在 L1 至多出现一次(可能没出现)，没考虑在 L1 中多次出现的情况。若考虑多次出现，可在上面算法找到第一次出现后的 q 结点作 L1 中下次比较的第一个字母结点，读者可自行完善之。

类似本题的另外叙述题的解答：

(1) [题目分析] 本题应先查找第 i 个结点，记下第 i 个结点的指针。然后从第 i+1 个结点起，直至第 m (1<i<m) 个结点止，依次插入到第 i-1 个结点之后。最后将暂存的第 i 个结点的指针指向第 m 结点，形成新的循环链表，结束了倒置算法。

LinkedList PatternInvert1 (LinkedList L, int i, m)

//L 是有 m 个结点的链表的头结点的指针。表中从第 i ( $1 < i < m$ ) 个结点到第 m 个结点构成循环部分链表, 本算法将这部分循环链表倒置。

```
{if (i<1 || i>m || m<4) {printf ("%d,%d 参数错误\n", i, m); exit (0); }
p=L->next->next; // p 是工作指针, 初始指向第二结点 (已假定 i>1)。
pre=L->next; // pre 是前驱结点指针, 最终指向第 i-1 个结点。
j=1; // 计数器
while (j<i-1) // 查找第 i 个结点。
 {j++; pre=p; p=p->next; } // 查找结束, p 指向第 i 个结点。
q=p; // 暂存第 i 个结点的指针。
p=p->next; // p 指向第 i+1 个结点, 准备逆置。
j+=2; // 上面 while 循环结束时, j=i-1, 现从第 i+1 结点开始逆置。
while (j<=m)
 {r=p->next; // 暂存 p 的后继结点。
 p->next=pre->next; // 逆置 p 结点。
 pre->next=p;
 p=r; // p 恢复为当前待逆置结点。
 j++; // 计数器增 1。
 }
q->next=pre->next; // 将原第 i 个结点的后继指针指向原第 m 个结点。
```

[算法讨论] 算法中未深入讨论 i, m, j 的合法性, 因题目的条件是  $m > 3$  且  $1 < i < m$ 。因此控制循环并未用指针判断 (如一般情况下的  $p != \text{null}$ ), 结束循环也未用指针判断。注意最后一句  $q \rightarrow \text{next} = \text{pre} \rightarrow \text{next}$ , 实现了从原第 i 个结点到原第 m 个结点的循环。最后  $\text{pre} \rightarrow \text{next}$  正是指向原第 m 个结点, 不可用  $p \rightarrow \text{next}$  代替  $\text{pre} \rightarrow \text{next}$ 。

21. [题目分析] 顺序存储结构的线性表的逆置, 只需一个变量辅助空间。算法核心是选择循环控制变量的初值和终值。

```
void SeqInvert (ElemType a[], int n)
// a 是具有 n 个元素用一维数组存储的线性表, 本算法将其逆置。
{for (i=0; i<= (n-1)/2; i++)
 {t=a[i]; a[i]=a[n-1-i]; a[n-1-i]=t; }
} // 算法结束
```

[算法讨论] 算法中循环控制变量的初值和终值是关键。C 中数组从下标 0 开始, 第 n 个元素的下标是 n-1。因为首尾对称交换, 所以控制变量的终值是线性表长度的一半。当 n 为偶数, “一半” 恰好是线性表长度的二分之一; 若 n 是奇数, “一半” 是小于  $n/2$  的最大整数, 这时取大于  $n/2$  的最小整数的位置上的元素, 恰是线性表中间位置的元素, 不需要逆置。另外, 由于 pascal 数组通常从下标 1 开始, 所以, 上下界处理上略有不同。这点请读者注意。

类似本题的其它题的解答:

这一组又选了 6 个题, 都是单链表 (包括单循环链表) 的逆置。链表逆置的通常作法是: 将工作指针指向第一个元素结点, 将头结点的指针域置空。然后将链表各结点从第一结点开始直至最后一个结点, 依次前插至头结点后, 使最后插入的结点成为链表的第一结点, 第一个插入的结点成为链表的最后结点。

(1) 要求编程实现带头结点的单链表的逆置。首先建立一单链表, 然后逆置。

```
typedef struct node
{int data; // 假定结点数据域为整型。
struct node *next;
```

```

 }node,*LinkedList;
LinkedList creat ()
{LinkedList head, p
 int x;
 head= (LinkedList) malloc (sizeof (node));
 head->next=null; /*设置头结点*/
 scanf ("%d", &x);
 while (x!=9999) /*约定输入 9999 时退出本函数*/
 {p= (LinkedList) malloc (sizeof (node));
 p->data=x;
 p->next=head->next; /* 将新结点链入链表*/
 head->next=p;
 scanf ("%d", &x);
 }
 return (head);
} //结束 creat 函数。
LinkedList invert1 (LinkedList head)
/*逆置单链表*/
{LinkedList p=head->next; /*p 为工作指针*/
 head->next=null;
 while (p!=null)
 {r=p->next; /*暂存 p 的后继*/
 p->next=head->next;
 head->next=p;
 p=r;
 }
 return (head);
} /*结束 invert1 函数*/
main ()
{LinkedList la;
 la=creat (); /*生成单链表*/
 la=invert1 (la); /*逆置单链表*/
}

```

(2) 本题要求将数据项递减有序的单链表重新排序，使数据项递增有序，要求算法复杂度为  $O(n)$ 。虽没说要求将链表逆置，这只是叙述不同，本质上是将单链表逆置，现编写如下：

```

LinkedList invert2 (LinkedList la)
//la 是带头结点且数据项递减有序的单链表，本算法将其排列成数据项递增有序的单链表。
{p=la->next; /*p 为工作指针*/
 la->next=null;
 while (p!=null)
 {r=p->next; /*暂存 p 的后继。*/
 p->next=la->next; /*将 p 结点前插入头结点后。*/
 la->next=p; p=r;
 }
}

```

} // 结束算法

(3) 本题要求倒排循环链表, 与上面倒排单链表处理不同之处有二: 一是初始化成循环链表而不是空链表; 二是判断链表尾不用空指针而用是否是链表头指针。算法中语句片段如下:

```
p=la->next; //p 为工作指针。
la->next=la; //初始化成空循环链表。
while (p!=la) //当 p=la 时循环结束。
{
 r=p->next; //暂存 p 的后继结点
 p->next=la->next; //逆置
 la->next=p; p=r;
}
```

(4) 不带头结点的单链表逆置比较复杂, 解决方法可以给加上头结点:

```
la= (LinkedList) malloc (sizeof (node));
la->next=L;
```

之后进行如上面 (2) 那样的逆置, 最后再删去头结点:

```
L=la->next; //L 是不带头结点的链表的指针。
free (la); //释放头结点。
```

若不增加头结点, 可用如下语句片段:

```
p=L->next; //p 为工作指针。
L->next=null; //第一结点成为尾结点。
while (p!=null)
{
 r=p->next;
 p->next=L; //将 p 结点插到 L 结点前面。
 L=p; //L 指向新的链表“第一”元素结点。
 p=r;
}
```

(5) 同 (4), 只是叙述有异。

(6) 同 (2), 差别仅在于叙述不同。

22. [题目分析] 在无序的单链表上, 查找最小值结点, 要查遍整个链表, 初始假定第一结点是最小值结点。当找到最小值结点后, 判断数据域的值是否是奇数, 若是, 则“与其后继结点的值相交换”即仅仅交换数据域的值, 用三个赋值语句即可交换。若与后继结点交换位置, 则需交换指针, 这时应知道最小值结点的前驱。至于删除后继结点, 则通过修改最小值结点的指针域即可。

[算法设计]

```
void MiniValue (LinkedList la)
```

// la 是数据域为正整数且无序的单链表, 本算法查找最小值结点且打印。若最小值结点的数值是奇数, 则与后继结点值交换; 否则, 就删除其直接后继结点。

```
{p=la->next; //设 la 是头结点的头指针, p 为工作指针。
pre=p; //pre 指向最小值结点, 初始假定首元结点值最小。
while (p->next!=null) //p->next 是待比较的当前结点。
{
 if (p->next->data<pre->data) pre=p->next;
 p=p->next; //后移指针
}
printf (“最小值=%d\n”, pre->data);
if (pre->data%2!=0) //处理奇数
```

```

 if (pre->next!=null) //若该结点没有后继, 则不必交换
 {t= pre->data; pre->data=pre->next->data; pre->next->data=t; } //交换完毕
else //处理偶数情况
 if (pre->next!=null) //若最小值结点是最后一个结点, 则无后继
 {u=pre->next; pre->next=u->next; free (u); } //释放后继结点空间

```

23. [题目分析] 将一个结点数据域为字符的单链表, 分解成含有字母字符、数字字符和其它字符的三个循环链表, 首先要构造分别含有这三类字符的表头结点。然后从原链表第一个结点开始, 根据结点数据域是字母字符、数字字符和其它字符而分别插入到三个链表之一的链表。注意不要因结点插入新建链表而使原链表断链。另外, 题目并未要求链表有序, 插入采用“前插法”, 每次插入的结点均成为所插入链表的第一元素的结点即可。

```

void OneToThree (LinkedList L, la, ld, lo)

```

//L 是无头结点的单链表第一个结点的指针, 链表中的数据域存放字符。本算法将链表 L 分解成含有英文字母字符、数字字符和其它字符的带头结点的三个循环链表。

```

{la= (LinkedList) malloc (sizeof (LNode)); //建立三个链表的头结点
ld= (LinkedList) malloc (sizeof (LNode));
lo= (LinkedList) malloc (sizeof (LNode));
la->next=la; ld->next=ld; lo->next=lo; //置三个循环链表为空表
while (L!=null) //分解原链表。
{r=L; L=L->next; //L 指向待处理结点的后继
if (r->data>= 'a' && r->data<= 'z' || r->data>= 'A' && r->data<= 'Z')
 {r->next=la->next; la->next=r; } //处理字母字符。
else if (r->data>= '0' && r->data<= '9')
 {r->next=ld->next; ld->next=r; } //处理数字字符
else {r->next=lo->next; lo->next=r; } //处理其它符号。
} //结束 while (L!=null)。
} //算法结束

```

[算法讨论] 算法中对 L 链表中每个结点只处理一次, 时间复杂度  $O(n)$ , 只增加了必须的三个表头结点, 符合题目“用最少的时间和最少的空间”的要求。

24. [题目分析] 在递增有序的线性表中, 删除数值相同的元素, 要知道被删除元素结点的前驱结点。

```

LinkedList DelSame (LinkedList la)

```

//la 是递增有序的单链表, 本算法去掉数值相同的元素, 使表中不再有重复的元素。

```

{pre=la->next; //pre 是 p 所指向的前驱结点的指针。
p=pre->next; //p 是工作指针。设链表中至少有一个结点。
while (p!=null)
 if (p->data==pre->data) //处理相同元素值的结点
 {u=p; p=p->next; free (u); } //释放相同元素值的结点
 else {pre->next=p; pre=p; p=p->next; } //处理前驱, 后继元素值不同
pre->next=p; //置链表尾。
} //DelSame

```

[算法讨论] 算法中假设链表至少有一个结点, 即初始时 pre 不为空, 否则 p->next 无意义。算法中最后 pre->next=p 是必须的, 因为可能链表最后有数据域值相同的结点, 这些结点均被删除, 指针后移使 p=null 而退出 while 循环, 所以应有 pre->next=p 使链表有尾。若链表尾部没数据域相同的结点, pre 和 p 为前驱和后继, pre->next=p 也是对的。

顺便提及，题目应叙述为非递减有序，因为“递增”是说明各结点数据域不同，一个值比一个值大，不会存在相同值元素。

25. [题目分析] 建立递增有序的顺序表，对每个输入数据，应首先查找该数据在顺序表中的位置，若表中没有该元素则插入之，如已有该元素，则不再插入，为此采用折半查找方法。

**FUNC** BinSearch (VAR a: sqliстtp; x: integer): integer;

// 在顺序表 a 中查找值为 x 的元素，如查找成功，返回 0 值，如 x 不在 a 中，则返回查找失败时的较大下标值。

low:=1; high:=a.last; found:=false;

**WHILE** (low<=high) **AND NOT** found **DO**

  [mid:=(low+high) DIV 2;

**IF** a.elem[mid]=x **THEN** found:=true

**ELSE IF** a.elem[mid]>x **THEN** high:=mid-1 **ELSE** low:=mid+1;

  ]

**IF** found=true **THEN return** (0)

**ELSE return** (low); // 当查找失败时，low=high+1。

**ENDF**; // 结束对分查找函数。

**PROC** create (VAR L: sqliстtp)

  // 本过程生成顺序表 L。

L.last:=0;           // 顺序表 L 初始化。

read (x);

**WHILE** x<>9999 **DO**           // 设 x=9999 时退出输入

  [k:=binsearch (L, x); // 去查找 x 元素。

**IF** k<>0                   // 不同元素才插入

**THEN** [FOR i:=L.last DOWNT0 k **DO** L.elem[i+1]:=L.elem[i];

    L.elem[k]=x; L.last:= L.last+1; // 插入元素 x，线性表长度增 1

  ]

  read (x);

  ]

**ENDP**; // 结束过程 creat

26. [题目分析] 在由正整数序列组成的有序单链表中，数据递增有序，允许相等整数存在。确定比正整数 x 大的数有几个属于计数问题，相同数只计一次，要求记住前驱，前驱和后继值不同时移动前驱指针，进行计数。将比正整数 x 小的数按递减排序，属于单链表的逆置问题。比正整数 x 大的偶数从表中删除，属于单链表中结点的删除，必须记住其前驱，以使链表不断链。算法结束时，链表中结点的排列是：小于 x 的数按递减排列，接着是 x (若有的话)，最后是大于 x 的奇数。

**void** exam (LinkedList la, int x)

  // la 是递增有序单链表，数据域为正整数。本算法确定比 x 大的数有几个；将比 x 小的数按递减排序，并将比 x 大的偶数从链表中删除。)

  {p=la->next; q=p; // p 为工作指针 q 指向最小值元素，其可能的后继将是 >=x 的第一个元素。

  pre=la;           // pre 为 p 的前驱结点指针。

  k=0;           // 计数 (比 x 大的数)。

  la->next=null; // 置空单链表表头结点。

**while** (p && p->data<x)       // 先解决比 x 小的数按递减次序排列

```

 {r=p->next; // 暂存后继
 p->next=la->next; // 逆置
 la->next=p;
 p=r; // 恢复当前指针。退出循环时，r 指向值>=x 的结点。
 }
 q->next=p; pre=q; // pre 指向结点的前驱结点
 while (p->data==x) {pre=p; p=p->next;} // 从小于 x 到大于 x 可能经过等于 x
 while (p) // 以下结点的数据域的值均大于 x
 {k++; x=p->data; // 下面仍用 x 表示数据域的值，计数
 if (x % 2==0) // 删偶数
 {while (p->data==x)
 {u=p; p=p->next; free(u); }
 pre->next=p; // 拉上链
 }
 else // 处理奇数
 while (p->data==x) // 相同数只记一次
 {pre->next=p; pre=p; p=p->next; }
 } // while(p)
 printf ("比值%d 大的数有%d 个\n", x, k);
} // 算法 exam 结束

```

[算法讨论] 本题“要求用最少的时间和最小的空间”。本算法中“最少的时间”体现在链表指针不回溯，最小空间是利用了几个变量。在查比 x 大的数时，必须找到第一个比 x 大的数所在结点（因等于 x 的数可能有，也可能多个，也可能没有）。之后，计数据的第一次出现，同时删去偶数。

顺便指出，题目设有“按递增次序”的“有序单链表”，所给例子序列与题目的论述并不一致。

27. [题目分析] 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。

另外，若无特别说明，为了处理的方便统一，单链表均设头结点，链表的指针就是头结点的指针。并且由于链表指针具有标记链表的作用，也常用指针名冠以链表名称。如“链表 head”既指的是链表的名字是 head，也指出链表的头指针是 head。

LinkedList Exchange (LinkedList HEAD, 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。

p->next=q->next; // p 的后继指向原 p 后继的后继。

```
q->next=p ; //原 p 后继的后继指针指向 p。
}
```

} //算法结束。

类似本题的其它题目的解答:

(1) 与上面第 27 题基本相同, 只是明确说明“p 指向的不是链表最后那个结点。”

(2) 与上面第 27 题基本相同, 仅叙述不同, 故不再作解答。

28. [题目分析] 本题链表结点的数据域存放英文单词, 可用字符数组表示, 单词重复出现时, 链表中只保留一个, 单词是否相等的判断使用 strcmp 函数, 结点中增设计数域, 统计单词重复出现的次数。

```
typedef struct node
{
 int freq; //频度域, 记单词出现的次数。
 char word[maxsize]; //maxsize 是单词中可能含有的最多字母个数。
 struct node *next;
}node, *LinkedList;

(1)LinkedList creat ()
//建立有 n (n>0) 个单词的单向链表, 若单词重复出现, 则只在链表中保留一个。
{LinkedList la;
la=(LinkedList) malloc (sizeof (node)); //申请头结点。
la->next=null; //链表初始化。
for (i=1; i<=n; i++) //建立 n 个结点的链表
{scanf ("%s", a); //a 是与链表中结点数据域同等长度的字符数组。
p=la->next; pre=p; //p 是工作指针, pre 是前驱指针。
while (p!=null)
 if (strcmp(p->data, a)==0) {p->freq++; break; } //单词重复出现, 频度增 1。
 else {pre=p; p=p->next; } //指针后移。
if (p==null) //该单词没出现过, 应插入。
{p=(LinkedList) malloc (sizeof (node));
strcpy (p->data, a); p->freq=1; p->next=null; pre->next=p;
} //将新结点插入到链表最后。
} //结束 for 循环。
return (la);
} //结束 creat 算法。
```

(2) void CreatOut ( )

//建立有 n 个单词的单向链表, 重复单词只在链表中保留一个, 最后输出频度最高的 k 个单词。

```
{LinkedList la;
la=(LinkedList) malloc (sizeof (node)); //申请头结点。
la->next=null; //链表初始化。
for (i=1; i<=n; i++) //建立 n 个结点的链表
{scanf ("%s", a); //a 是与链表中结点数据域同等长度的字符数组。
p=la->next; pre=p; //p 是工作指针, pre 是前驱指针。
while (p!=null)
 if (strcmp(p->data, a)==0)
 {p->freq++; //单词重复出现, 频度增 1。

```



```

 pre->next=p->next; // 先将 p 结点从链表上摘下, 再按频度域值插入到合适位置
 pre=la; q=la->next;
 while(q->freq>p->freq) (pre=q; q=q->next;)
 pre->next=p; p->next=q; // 将 p 结点插入到合适位置
}
else {pre=p; p=p->next; } // 指针后移。
if (p==null) // 该单词没出现过, 应插入到链表最后。
{p= (LinkedList) malloc (sizeof (node));
strncpy (p->data, a); p->freq=1; p->next=null; pre->next=p;
} // if 新结点插入。
} // 结束 for 循环建表。
int k, i=0;
scanf(“输入要输出单词的个数%d”, &k);
p=la->next;
while (p && i<k) // 输出频度最高的 k 个单词
{printf(“第%d 个单词%s 出现%d 次\n”, ++i, p->data, p->freq);
p=p->next;
}
if (!p)
printf(“给出的%d 值太大\n”, k);
} // 结束算法

```

29. [题目分析] 双向循环链表自第二结点至表尾递增有序, 要求将第一结点插入到链表中, 使整个链表递增有序。由于已给条件 ( $a_1 < x < a_n$ ), 故应先将第一结点从链表上摘下来, 再将其插入到链表中相应位置。由于是双向链表, 不必象单链表那样必须知道插入结点的前驱。

```

void DInsert (DLinkedList dl)
// dl 是无头结点的双向循环链表, 自第二结点起递增有序。本算法将第一结点 ($a_1 < x < a_n$)
// 插入到链表中, 使整个链表递增有序。
{s=la; // s 暂存第一结点的指针。
p=la->next; p->prior=la->prior; p->prior->next=p; // 将第一结点从链表上摘下。
while (p->data<x) p=p->next; // 查插入位置
s->next=p; s->prior=p->prior; p->prior->next=s; p->prior=s; // 插入原第一结点 s
} // 算法结束。

```

[算法讨论] 由于题目已给  $a_1 < x < a_n$ , 所以在查找第一结点插入位置时用的循环条件是  $p->data < x$ , 即在  $a_1$  和  $a_n$  间肯定能找到第一结点的插入位置。若无此条件, 应先看第一结点数据域值  $x$  是否小于等于  $a_1$ , 如是, 则不作任何操作。否则, 查找其插入位置, 循环控制要至多查找完  $a_1$  到  $a_n$  结点。

```

if (p->data<x) p=p->next; else break;

```

30. [题目分析] 在顺序存储的线性表上删除元素, 通常要涉及到一系列元素的移动 (删第  $i$  个元素, 第  $i+1$  至第  $n$  个元素要依次前移)。本题要求删除线性表中所有值为  $item$  的数据元素, 并未要求元素间的相对位置不变。因此可以考虑设头尾两个指针 ( $i=1, j=n$ ), 从两端向中间移动, 凡遇到值  $item$  的数据元素时, 直接将右端元素左移至值为  $item$  的数据元素位置。

```

void Delete (ElemType A[], int n)
// A 是有 n 个元素的一维数组, 本算法删除 A 中所有值为 item 的元素。
{i=1; j=n; // 设置数组低、高端指针 (下标)。

```

```

while (i<j)
{while (i<j && A[i]!=item) i++; //若值不为 item, 左移指针。
 if (i<j) while (i<j && A[j]==item) j--; //若右端元素值为 item, 指针左移
 if (i<j) A[i++]=A[j--];
}

```

[算法讨论] 因元素只扫描一趟, 算法时间复杂度为  $O(n)$ 。删除元素未使用其它辅助空间, 最后线性表中的元素个数是  $j$ 。若题目要求元素间相对顺序不变, 请参见本章三、填空题 25 的算法。

31. [题目分析] 本题所用数据结构是静态双向链表, 其结构定义为:

```

typedef struct node
{char data[maxsize]; //用户姓名, maxsize 是可能达到的用户名的最大长度。
 int Llink, Rlink; //前向、后向链, 其值为乘客数组下标值。
}unode;

unode user[max]; //max 是可能达到的最多客户数。

```

设  $av$  是可用数组空间的最小下标, 当有客户要订票时, 将其姓名写入该单元的  $data$  域, 然后在静态链表中查找其插入位置。将该乘客姓名与链表中第一个乘客姓名比较, 根据大于或小于第一个乘客姓名, 而决定沿第一个乘客的右链或左链去继续查找, 直到找到合适位置插入之。

```
void Insert (unode user[max], int av)
```

//user 是静态双向链表, 表示飞机票订票系统, 元素包含  $data$ 、 $Llink$  和  $Rlink$  三个域, 结点按来客姓名排序。本算法处理任一乘客订票申请。

```

{scanf ("%s", s); //s 是字符数组, 存放乘客姓名。
 strcpy (user[av].data, s);
 p=1; //p 为工作指针 (下标)
 if(strcmp (user[p].data, s)<0) //沿右链查找
 {while (p!=0 && strcmp (user[p].data, s)<0) {pre=p; p=user[p].Rlink; }
 user[av].Rlink=p; user[av].Llink=pre; //将新乘客链入表中
 user[pre].Rlink=av; user[p].Llink=av;
 }
 else //沿左右链查找
 {while (p!=0 && strcmp (user[p].data, s)>0) {pre=p; p=user[p].Llink; }
 user[av].Rlink=pre; user[av].Llink=p; //将新乘客链入表中
 user[pre].Llink=av; user[p].Rlink=av;
 }
} //算法结束

```

[算法讨论] 本算法只讨论了乘客订票情况, 未考虑乘客退票。也未考虑从空开始建立链表。增加乘客时也未考虑姓名相同者 (实际系统姓名不能做主关键字)。完整系统应有 (1) 初始化, 把整个数组空间初始化成双向静态链表, 全部空间均是可利用空间。(2) 申请空间。当有乘客购票时, 要申请空间, 直到无空间可用为止。(3) 释放空间。当乘客退票时, 将其空间收回。由于空间使用无优先级, 故可将退票释放的空间作为下个可利用空间, 链入可利用空间表中。

32. [题目分析] 首先在双向链表中查找数据值为  $x$  的结点, 查到后, 将结点从链表上摘下, 然后再顺结点的前驱链查找该结点的位置。

```
DLinkedList locate(DLinkedList L, ElemType x)
```

//  $L$  是带头结点的按访问频度递减的双向链表, 本算法先查找数据  $x$ , 查找成功时结点的访问频度增 1, 最后将该结点按频度递减插入链表中适当位置。

```

{ DLinkedList 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(“不存在值为 x 的结点\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
} // 算法结束

```

33. [题目分析] 题目要求按递增次序输出单链表中各结点的数据元素, 并释放结点所占存储空间。应对链表进行遍历, 在每趟遍历中查找出整个链表的最小值元素, 输出并释放结点所占空间; 再查次最小值元素, 输出并释放空间, 如此下去, 直至链表为空, 最后释放头结点所占存储空间。当然, 删除结点一定要记住该结点的前驱结点的指针。

```

void MiniDelete (LinkedList head)
//head 是带头结点的单链表的头指针, 本算法按递增顺序输出单链表中各结点的数据元素, 并
//释放结点所占的存储空间。
{while (head->next!=null) //循环到仅剩头结点。
 {pre=head; //pre 为元素最小值结点的前驱结点的指针。
 p=pre->next; //p 为工作指针
 while (p->next!=null)
 {if (p->next->data<pre->next->data) pre=p; //记住当前最小值结点的前驱
 p=p->next;
 }
 printf (pre->next->data); //输出元素最小值结点的数据。
 u=pre->next; pre->next=u->next; free (u); //删除元素值最小的结点, 释放结点空
 间
 } // while (head->next!=null)
 free (head); } //释放头结点。

```

[算法讨论] 算法中使用的指针变量只有 pre, p 和 u 三个, 请读者细心体会。要注意没特别记最小值结点, 而是记其前驱。

34. [题目分析] 留下三个链表中公共数据, 首先查找两表 A 和 B 中公共数据, 再去 C 中找有无该数据。要消除重复元素, 应记住前驱, 要求时间复杂度  $O(m+n+p)$ , 在查找每个链表时, 指针不能回溯。

```

LinkedList Common (LinkedList A, B, C)
//A, B 和 C 是三个带头结点且结点元素值非递减排列的有序表。本算法使 A 表仅留下三个表均
//包含的结点, 且结点值不重复, 释放所有结点。
{pa=A->next; pb=B->next; pc=C->next; //pa, pb 和 pc 分别是 A, B 和 C 三个表的工作指针。
 pre=A; //pre 是 A 表中当前结点的前驱结点的指针。
 while (pa && pb && pc) //当 A, B 和 C 表均不空时, 查找三表共同元素
 { while (pa && pb)

```

if (pa->data<pb->data) {u=pa; pa=pa->next; free (u); } // 结点元素值小时, 后移指针。

```
else if (pa->data> pb->data) pb=pb->next;
else if (pa && pb) // 处理 A 和 B 表元素值相等的结点
{while (pc && pc->data<pa->data) pc=pc->next;
if(pc)
{if(pc->data>pa->data)//pc 当前结点值与 pa 当前结点值不等, pa 后移指针。
{u=pa; pa=pa->next; free (u); }
else // pc, pa 和 pb 对应结点元素值相等。
{if(pre==A) { pre->next=pa; pre=pa; pa=pa->next} // 结果表中第一个结
点。
```

```
else if(pre->data==pa->data) // (处理) 重复结点不链入 A 表
{u=pa; pa=pa->next; free (u); }
else {pre->next=pa; pre=pa; pa=pa->next; } // 将新结点链入 A 表。
pb=pb->next; pc=pc->next; // 链表的工作指针后移。
} } // else pc, pa 和 pb 对应结点元素值相等
if (pa==null) pre->next=null; // 原 A 表已到尾, 置新 A 表表尾
else // 处理原 A 表未到尾而 B 或 C 到尾的情况
{pre->next=null; // 置 A 表表尾标记
while (pa!=null) // 删除原 A 表剩余元素。
{u=pa; pa=pa->next; free (u); }
```

[算法讨论] 算法中 A 表、B 表和 C 表均从头到尾 (严格说 B、C 中最多一个到尾) 遍历一遍, 算法时间复杂度符合  $O(m+n+p)$ 。算法主要由 while (pa && pb && pc) 控制。三表有一个到尾则结束循环。算法中查到 A 表与 B 表和 C 表的公共元素后, 又分三种情况处理: 一是三表中第一个公共元素值相等的结点; 第二种情况是, 尽管不是第一结点, 但与前驱结点元素值相同, 不能成为结果表中的结点; 第三种情况是新结点与前驱结点元素值不同, 应链入结果表中, 前驱指针也移至当前结点, 以便与以后元素值相同的公共结点进行比较。算法最后要给新 A 表置结尾标记, 同时若原 A 表没到尾, 还应释放剩余结点所占的存储空间。

## 第3章 栈和队列

### 一 选择题

1. 对于栈操作数据的原则是 ( )。【青岛大学 2001 五、2 (2 分)】

- A. 先进先出 B. 后进先出 C. 后进后出 D. 不分顺序

2. 在作进栈运算时,应先判别栈是否 ( ① ),在作退栈运算时应先判别栈是否 ( ② )。当栈中元素为  $n$  个,作进栈运算时发生上溢,则说明该栈的最大容量为 ( ③ )。

为了增加内存空间的利用率和减少溢出的可能性,由两个栈共享一片连续的内存空间时,应将两栈的 ( ④ ) 分别设在这片内存空间的两端,这样,当 ( ⑤ ) 时,才产生上溢。

①, ②: A. 空 B. 满 C. 上溢 D. 下溢

③: A.  $n-1$  B.  $n$  C.  $n+1$  D.  $n/2$

④: A. 长度 B. 深度 C. 栈顶 D. 栈底

⑤: A. 两个栈的栈顶同时到达栈空间的中心点.

B. 其中一个栈的栈顶到达栈空间的中心点.

C. 两个栈的栈顶在栈空间的某一位置相遇.

D. 两个栈均不空,且一个栈的栈顶到达另一个栈的栈底.

【上海海运学院 1997 二、1 (5 分)】【上海海运学院 1999 二、1 (5 分)】

3. 一个栈的输入序列为  $123\cdots n$ ,若输出序列的第一个元素是  $n$ ,输出第  $i$  ( $1 \leq i \leq n$ ) 个元素是 ( )。

- A. 不确定 B.  $n-i+1$  C.  $i$  D.  $n-i$

【中山大学 1999 一、9 (1 分)】

4. 若一个栈的输入序列为  $1, 2, 3, \cdots, n$ ,输出序列的第一个元素是  $i$ ,则第  $j$  个输出元素是 ( )。

- A.  $i-j-1$  B.  $i-j$  C.  $j-i+1$  D. 不确定的

【武汉大学 2000 二、3】

5. 若已知一个栈的入栈序列是  $1, 2, 3, \cdots, n$ ,其输出序列为  $p_1, p_2, p_3, \cdots, p_n$ ,若  $p_n$  是  $n$ ,则  $p_i$  是 ( )。

- A.  $i$  B.  $n-i$  C.  $n-i+1$  D. 不确定

【南京理工大学 2001 一、1 (1.5 分)】

6. 有六个元素  $6, 5, 4, 3, 2, 1$  的顺序进栈,问下列哪一个不是合法的出栈序列? ( )

- A.  $5\ 4\ 3\ 6\ 1\ 2$  B.  $4\ 5\ 3\ 1\ 2\ 6$  C.  $3\ 4\ 6\ 5\ 2\ 1$  D.  $2\ 3\ 4\ 1\ 5\ 6$

【北方交通大学 2001 一、3 (2 分)】

7. 设栈的输入序列是  $1, 2, 3, 4$ ,则 ( ) 不可能是其出栈序列。【中科院计算所 2000 一、10 (2 分)】

- A.  $1, 2, 4, 3,$  B.  $2, 1, 3, 4,$  C.  $1, 4, 3, 2,$   
D.  $4, 3, 1, 2,$  E.  $3, 2, 1, 4,$

8. 一个栈的输入序列为  $1\ 2\ 3\ 4\ 5$ ,则下列序列中不可能是栈的输出序列的是 ( )。

- A.  $2\ 3\ 4\ 1\ 5$  B.  $5\ 4\ 1\ 3\ 2$  C.  $2\ 3\ 1\ 4\ 5$  D.  $1\ 5\ 4\ 3\ 2$

【南开大学 2000 一、1】【山东大学 2001 二、4 (1 分)】【北京理工大学 2000 一、2 (2 分)】

9. 设一个栈的输入序列是  $1, 2, 3, 4, 5$ ,则下列序列中,是栈的合法输出序列的是 ( )。

- A.  $5\ 1\ 2\ 3\ 4$  B.  $4\ 5\ 1\ 3\ 2$  C.  $4\ 3\ 1\ 2\ 5$  D.  $3\ 2\ 1\ 5\ 4$

【合肥工业大学 2001 一、1 (2 分)】

10. 某堆栈的输入序列为  $a, b, c, d$ ,下面的四个序列中,不可能是它的输出序列的是 ( )。

- A. a, c, b, d      B. b, c, d, a      C. c, d, b, a      D. d, c, a, b

【北京航空航天大学 2000 一、3 (2 分)】【北京邮电大学 1999 一、3 (2 分)】

11. 设 abcdef 以所给的次序进栈, 若在进栈操作时, 允许退栈操作, 则下面得不到的序列为 ( )。

- A. fedcba      B. bcafed      C. dcefba      D. cabdef

【南京理工大学 1996 一、9 (2 分)】

12. 设有三个元素 X, Y, Z 顺序进栈 (进的过程中允许出栈), 下列得不到的出栈排列是 ( )。

- A. XYZ      B. YZX      C. ZXY      D. ZYX

【南京理工大学 1997 一、5 (2 分)】

13. 输入序列为 ABC, 可以变为 CBA 时, 经过的栈操作为 ( ) 【中山大学 1999 一、8 (1 分)】

- A. push, pop, push, pop, push, pop      B. push, push, push, pop, pop, pop  
C. push, push, pop, pop, push, pop      D. push, pop, push, push, pop, pop

14. 若一个栈以向量  $V[1..n]$  存储, 初始栈顶指针  $top$  为  $n+1$ , 则下面  $x$  进栈的正确操作是 ( )。

- A.  $top:=top+1; V[top]:=x$       B.  $V[top]:=x; top:=top+1$   
C.  $top:=top-1; V[top]:=x$       D.  $V[top]:=x; top:=top-1$

【南京理工大学 1998 一、13 (2 分)】

15. 若栈采用顺序存储方式存储, 现两栈共享空间  $V[1..m]$ ,  $top[i]$  代表第  $i$  个栈 ( $i=1, 2$ ) 栈顶, 栈 1 的底在  $v[1]$ , 栈 2 的底在  $V[m]$ , 则栈满的条件是 ( )。

- A.  $|top[2]-top[1]|=0$       B.  $top[1]+1=top[2]$       C.  $top[1]+top[2]=m$       D.  $top[1]=top[2]$

【南京理工大学 1999 一、14 (1 分)】

16. 栈在 ( ) 中应用。【中山大学 1998 二、3 (2 分)】

- A. 递归调用      B. 子程序调用      C. 表达式求值      D. A, B, C

17. 一个递归算法必须包括 ( )。【武汉大学 2000 二、2】

- A. 递归部分      B. 终止条件和递归部分      C. 迭代部分      D. 终止条件和迭代部分

18. 执行完下列语句段后,  $i$  值为: ( ) 【浙江大学 2000 一、6 (3 分)】

```
int f(int x)
{ return ((x>0) ? x*f(x-1):2); }
int i ;
i =f(f(1));
```

- A. 2      B. 4      C. 8      D. 无限递归

19. 表达式  $a*(b+c)-d$  的后缀表达式是 ( )。【南京理工大学 2001 一、2 (1.5 分)】

- A. abcd\*+-      B. abc+\*-d      C. abc\*+d-      D. -+\*abcd

20. 表达式  $3*2^{(4+2*2-6*3)}-5$  求值过程中当扫描到 6 时, 对象栈和算符栈为 ( ), 其中  $^$  为乘幂。

- A. 3, 2, 4, 1, 1;  $(*^{+*-}$       B. 3, 2, 8;  $(*^{--}$       C. 3, 2, 4, 2, 2;  $(*^{(-}$       D. 3, 2, 8;  $(*^{(-$

【青岛大学 2000 五、5 (2 分)】

21. 设计一个判别表达式中左, 右括号是否配对出现的算法, 采用 ( ) 数据结构最佳。

- A. 线性表的顺序存储结构      B. 队列      C. 线性表的链式存储结构      D. 栈

【西安电子科技大学 1996 一、6 (2 分)】

22. 用链接方式存储的队列, 在进行删除运算时 ( )。【北方交通大学 2001 一、12 (2 分)】

- A. 仅修改头指针      B. 仅修改尾指针      C. 头、尾指针都要修改      D. 头、尾指针可能

都要修改

23. 用不带头结点的单链表存储队列时, 其队头指针指向队头结点, 其队尾指针指向队尾结点, 则在进行删除操作时( )。【北京理工大学 2001 六、3 (2 分)】

- A. 仅修改队头指针      B. 仅修改队尾指针  
C. 队头、队尾指针都要修改      D. 队头、队尾指针都可能要修改

24. 递归过程或函数调用时, 处理参数及返回地址, 要用一种称为( )的数据结构。

- A. 队列      B. 多维数组      C. 栈      D. 线性表

【福州大学 1998 一、1 (2 分)】

25. 假设以数组  $A[m]$  存放循环队列的元素, 其头尾指针分别为  $front$  和  $rear$ , 则当前队列中的元素个数为( )。【北京工商大学 2001 一、2 (3 分)】

- A.  $(rear-front+m)\%m$       B.  $rear-front+1$       C.  $(front-rear+m)\%m$   
D.  $(rear-front)\%m$

26. 循环队列  $A[0..m-1]$  存放其元素值, 用  $front$  和  $rear$  分别表示队头和队尾, 则当前队列中的元素数是( )。【南京理工大学 2001 一、5 (1.5 分)】

- A.  $(rear-front+m)\%m$       B.  $rear-front+1$       C.  $rear-front-1$       D.  $rear-front$

27. 循环队列存储在数组  $A[0..m]$  中, 则入队时的操作为( )。【中山大学 1999 一、6 (1 分)】

- A.  $rear=rear+1$       B.  $rear=(rear+1) \bmod (m-1)$   
C.  $rear=(rear+1) \bmod m$       D.  $rear=(rear+1) \bmod (m+1)$

28. 若用一个大小为 6 的数组来实现循环队列, 且当前  $rear$  和  $front$  的值分别为 0 和 3, 当从队列中删除一个元素, 再加入两个元素后,  $rear$  和  $front$  的值分别为多少? ( ) 【浙江大学 1999 四、1 (4 分)】

- A. 1 和 5      B. 2 和 4      C. 4 和 2      D. 5 和 1

29. 已知输入序列为  $abcd$  经过输出受限的双向队列后能得到的输出序列有( )。

- A.  $dacb$       B.  $cadb$       C.  $dbca$       D.  $bdac$       E. 以上答案都不对

【西安交通大学 1996 三、3 (3 分)】

30. 若以 1234 作为双端队列的输入序列, 则既不能由输入受限的双端队列得到, 也不能由输出受限的双端队列得到的输出序列是( )。【西安电子科技大学 1996 一、5 (2 分)】

- A. 1234      B. 4132      C. 4231      D. 4213

31. 最大容量为  $n$  的循环队列, 队尾指针是  $rear$ , 队头是  $front$ , 则队空的条件是( )。

- A.  $(rear+1) \bmod n = front$       B.  $rear = front$   
C.  $rear+1 = front$       D.  $(rear-1) \bmod n = front$

【南京理工大学 1999 一、16 (2 分)】

32. 栈和队列的共同点是( )。【燕山大学 2001 一、1 (2 分)】

- A. 都是先进先出      B. 都是先进后出  
C. 只允许在端点处插入和删除元素      D. 没有共同点

33. 栈的特点是( ① ), 队列的特点是( ② ), 栈和队列都是( ③ )。若进栈序列为 1, 2, 3, 4 则( ④ )不可能是一个出栈序列(不一定全部进栈后再出栈); 若进队列的序列为 1, 2, 3, 4 则( ⑤ )是一个出队列序列。【北方交通大学 1999 一、1 (5 分)】

- ①, ②: A. 先进先出      B. 后进先出      C. 进优于出      D. 出优于进

- ③: A. 顺序存储的线性结构      B. 链式存储的线性结构  
C. 限制存取点的线性结构      D. 限制存取点的非线性结构

- ④, ⑤: A. 3, 2, 1, 4      B. 3, 2, 4, 1      C. 4, 2, 3, 1      D. 4, 3, 2, 1      F. 1, 2, 3, 4      G. 1, 3, 2, 4



34. 栈和队都是 ( ) 【南京理工大学 1997 一、3 (2 分)】

- A. 顺序存储的线性结构      B. 链式存储的非线性结构  
C. 限制存取点的线性结构      D. 限制存取点的非线性结构

35. 设栈 S 和队列 Q 的初始状态为空, 元素  $e_1, e_2, e_3, e_4, e_5$  和  $e_6$  依次通过栈 S, 一个元素出栈后即进队列 Q, 若 6 个元素出队的序列是  $e_2, e_4, e_3, e_6, e_5, e_1$  则栈 S 的容量至少应该是 ( )。

- A. 6      B. 4      C. 3      D. 2

【南京理工大学 2000 一、6 (1.5 分)】

36. 用单链表表示的链式队列的队头在链表的 ( ) 位置。【清华大学 1998 一、1 (2 分)】

- A. 链头      B. 链尾      C. 链中

37. 依次读入数据元素序列 {a, b, c, d, e, f, g} 进栈, 每进一个元素, 机器可要求下一个元素进栈或弹栈, 如此进行, 则栈空时弹出的元素构成的序列是以下哪些序列? 【哈尔滨工业大学 2000 七 (8 分)】

- A. {d, e, c, f, b, g, a}      B. {f, e, g, d, a, c, b}  
C. {e, f, d, g, b, c, a}      D. {c, d, b, e, f, a, g}

## 二 判断题

1. 消除递归不一定需要使用栈, 此说法 ( )

【中科院计算所 1998 二、2 (2 分)】【中国科技大学 1998 二、2 (2 分)】

2. 栈是实现过程和函数等子程序所必需的结构。( ) 【合肥工业大学 2000 二、2 (1 分)】

3. 两个栈共用静态存储空间, 对头使用也存在空间溢出问题。( ) 【青岛大学 2000 四、2 (1 分)】

4. 两个栈共享一片连续内存空间时, 为提高内存利用率, 减少溢出机会, 应把两个栈的栈底分别设在这片内存空间的两端。( ) 【上海海运学院 1998 一、4 (1 分)】

5. 即使对不含相同元素的同一输入序列进行两组不同的合法的入栈和出栈组合操作, 所得的输出序列也一定相同。( ) 【北京邮电大学 1999 二、4 (2 分)】

6. 有  $n$  个数顺序 (依次) 进栈, 出栈序列有  $C_n$  种,  $C_n = [1/(n+1)] * (2n)! / [(n!) * (n!)]$ 。( )

【北京邮电大学 1998 一、3 (2 分)】

7. 栈与队列是一种特殊操作的线性表。( ) 【青岛大学 2001 四、3 (1 分)】

8. 若输入序列为 1, 2, 3, 4, 5, 6, 则通过一个栈可以输出序列 3, 2, 5, 6, 4, 1。( )

【上海海运学院 1995 一、2 (1 分)      1997 一、3 (1 分)】

9. 栈和队列都是限制存取点的线性结构。( ) 【中科院软件所 1999 六、(5) (2 分)】

10. 若输入序列为 1, 2, 3, 4, 5, 6, 则通过一个栈可以输出序列 1, 5, 4, 6, 2, 3。( )

【上海海运学院 1999 一、3 (1 分)】

11. 任何一个递归过程都可以转换成非递归过程。( ) 【上海交通大学 1998 一、3 (1 分)】

12. 只有那种使用了局部变量的递归过程在转换成非递归过程时才必须使用栈。( )

【上海交通大学 1998 一、4 (1 分)】

13. 队列是一种插入与删除操作分别在表的两端进行的线性表, 是一种先进后出型结构。( )

【上海海运学院 1998 一、3 (1 分)】

14. 通常使用队列来处理函数或过程的调用。( ) 【南京航空航天大学 1997 一、5 (1 分)】

15. 队列逻辑上是一个下端和上端既能增加又能减少的线性表。( ) 【上海交通大学 1998 一、2】

16. 循环队列通常用指针来实现队列的头尾相接。( ) 【南京航空航天大学 1996 六、1 (1 分)】



17. 循环队列也存在空间溢出问题。( )【青岛大学 2002 一、2 (1分)】
18. 队列和栈都是运算受限的线性表, 只允许在表的两端进行运算。( )【长沙铁道学院 1997 一、5(1分)】
19. 栈和队列都是线性表, 只是在插入和删除时受到了一些限制。( )【北京邮电大学 2002 一、3(1分)】
20. 栈和队列的存储方式, 既可以是顺序方式, 又可以是链式方式。( )  
【上海海运学院 1996 一、2 (1分) 1999 一、2 (1分)】

### 三 填空题

1. 栈是\_\_\_\_\_的线性表, 其运算遵循\_\_\_\_\_的原则。【北京科技大学 1997 一、3】
2. \_\_\_\_\_是限定仅在表尾进行插入或删除操作的线性表。【燕山大学 1998 一、3 (1分)】
3. 一个栈的输入序列是: 1, 2, 3 则不可能的栈输出序列是\_\_\_\_\_。【中国人民大学 2001 一、1(2分)】
4. 设有一个空栈, 栈顶指针为 1000H(十六进制), 现有输入序列为 1, 2, 3, 4, 5, 经过 PUSH, PUSH, POP, PUSH, POP, PUSH, PUSH 之后, 输出序列是\_\_\_\_\_, 而栈顶指针值是\_\_\_\_\_H。设栈为顺序栈, 每个元素占 4 个字节。【西安电子科技大学 1998 二、1 (4分)】
5. 当两个栈共享一存储区时, 栈利用一维数组  $stack(1, n)$  表示, 两栈顶指针为  $top[1]$  与  $top[2]$ , 则当栈 1 空时,  $top[1]$  为\_\_\_\_\_, 栈 2 空时,  $top[2]$  为\_\_\_\_\_, 栈满时为\_\_\_\_\_。  
【南京理工大学 1997 三、1 (3分)】
6. 两个栈共享空间时栈满的条件\_\_\_\_\_。【中山大学 1998 一、3 (1分)】
7. 在作进栈运算时应先判别栈是否 (1); 在作退栈运算时应先判别栈是否 (2); 当栈中元素为  $n$  个, 作进栈运算时发生上溢, 则说明该栈的最大容量为 (3)。  
为了增加内存空间的利用率和减少溢出的可能性, 由两个栈共享一片连续的空间时, 应将两栈的 (4) 分别设在内存空间的两端, 这样只有当 (5) 时才产生溢出。【山东工业大学 1994 一、1 (5分)】
8. 多个栈共存时, 最好用\_\_\_\_\_作为存储结构。【南京理工大学 2001 二、7 (2分)】
9. 用 S 表示入栈操作, X 表示出栈操作, 若元素入栈的顺序为 1234, 为了得到 1342 出栈顺序, 相应的 S 和 X 的操作串为\_\_\_\_\_。【西南交通大学 2000 一、5】
10. 顺序栈用  $data[1..n]$  存储数据, 栈顶指针是  $top$ , 则值为  $x$  的元素入栈的操作是\_\_\_\_\_。  
【合肥工业大学 2001 三、2 (2分)】
11. 表达式  $23 + ((12 * 3 - 2) / 4 + 34 * 5 / 7) + 108 / 9$  的后缀表达式是\_\_\_\_\_。【中山大学 1998 一、4 (1分)】
12. 循环队列的引入, 目的是为了克服\_\_\_\_\_。【厦门大学 2001 一、1 (14/8分)】
13. 用下标 0 开始的  $N$  元数组实现循环队列时, 为实现下标变量  $M$  加 1 后在数组有效下标范围内循环, 可采用的表达式是:  $M :=$  \_\_\_\_\_ (填 PASCAL 语言, C 语言的考生不填);  $M =$  \_\_\_\_\_ (填 C 语言, PASCAL 语言的考生不填)。【西南交通大学 2000 一、7】
14. \_\_\_\_\_又称作先进先出表。【重庆大学 2000 一、7】
15. 队列的特点是\_\_\_\_\_。【北京理工大学 2000 二、2 (2分)】
16. 队列是限制插入只能在表的一端, 而删除在表的另一端进行的线性表, 其特点是\_\_\_\_\_。  
【北方交通大学 2001 二、5】
17. 已知链队列的头尾指针分别是  $f$  和  $r$ , 则将值  $x$  入队的操作序列是\_\_\_\_\_。  
【合肥工业大学 2000 三、3 (2分)】
18. 区分循环队列的满与空, 只有两种方法, 它们是\_\_\_\_\_和\_\_\_\_\_。【北京邮电大学 2001 二、2 (4分)】

19. 设循环队列用数组  $A[1..M]$  表示, 队首、队尾指针分别是 FRONT 和 TAIL, 判定队满的条件为\_\_\_\_\_。

【山东工业大学 1995 一、1 (1 分)】

20. 设循环队列存放在向量  $sq.data[0:M]$  中, 则队头指针  $sq.front$  在循环意义下的出队操作可表示为\_\_\_\_\_, 若用牺牲一个单元的办法来区分队满和队空 (设队尾指针  $sq.rear$ ), 则队满的条件为\_\_\_\_\_。

【长沙铁道学院 1997 二、4 (4 分)】

21. 表达式求值是\_\_\_\_\_应用的一个典型例子。【重庆大学 2000 一、10】

22. 循环队列用数组  $A[0..m-1]$  存放其元素值, 已知其头尾指针分别是 front 和 rear, 则当前队列的元素个数是\_\_\_\_\_。【厦门大学 2000 六、1 (16%/3 分)】

23. 设  $Q[0..N-1]$  为循环队列, 其头、尾指针分别为 P 和 R, 则队 Q 中当前所含元素个数为\_\_\_\_\_。

【北京科技大学 1997 一、4】

24. 完善下面算法。【中山大学 1998 四、2 (6 分)】

后缀表达式求值, 表达式  $13/25+61$  的后缀表达式格式为: 13, 25/61, +  
FUNC compute(a):real; 后缀表达式存储在数组  $a[1..m]$  中。

BEGIN

setnull(s); i:=1; ch:= (1)\_\_\_\_\_;

WHILE ch<>'@' DO

BEGIN

CASE ch OF

'0' .. '9' : x:=0;

WHILE ch<>',' DO

BEGIN

x:=x\*10+ord(ch)-ord('0');

i:=i+1; ch:= (2)\_\_\_\_\_;

END

'+' : x:=pop(s)+pop(s);

'-' : x:=pop(s); x:=pop(s)-x;

'\*' : x:=pop(s)\*pop(s);

'/' : x:=pop(s); x:=pop(s)/x;

ENDCASE

push(s,x); i:=i+1; ch:=a[i];

END;

comput:= (3)\_\_\_\_\_;

END;

25. 算术表达式求值的流程, 其中 OPTR 为算术符栈, OPND 为操作数栈, precede(oper1, oper2) 是比较运算符优先级别的函数, operate(opnd1, oper, opnd2) 为两操作数的运算结果函数。( # 表示运算起始和终止符号) 【西北工业大学 1999 六、2 (7 分)】

FUNCTION exp\_reduced:operandtype;

INITSTACK(OPTR); PUSH(OPTR"#"); INITSTACK(OPND); read(w);

WHILE NOT((w='#' ) AND (GETTOP(OPTR)=' #')) DO

IF NOT w in op THEN PUSH(OPND, w);

ELSE CASE precede(GETTOP(OPTR), w) OF

'<': [(1)\_\_\_\_\_]; read(w);]

```

'=' : [(2)_____ ; read(w) ;];
'>' : [theta:=POP(OPTR) ; b:=POP(OPND) ; a:=POP(OPND) ; (3)_____ ;]
ENDC;
RETURN(GETTOP(OPND));
ENDF;

```

26. 根据需要, 用适当的语句填入下面算法的\_\_\_\_\_中:

问题: 设有  $n$  件物品, 重量分别为  $w_1, w_2, w_3, \dots, w_n$  和一个能装载总重量为  $T$  的背包。能否从  $n$  件物品中选择若干件恰好使它们的重量之和等于  $T$ 。若能, 则背包问题有解, 否则无解。解此问题的算法如下:

```

FUNCTION kanp_stack(VAR stack,w:ARRAY[1..n] OF real; VAR top:integer;
T:real):boolean;

```

{w[1: n] 存放  $n$  件物品的重量, 依次从中取出物品放入背包中, 检查背包重量, 若不超过  $T$ , 则装入, 否则弃之, 取下一个物品试之。若有解则返回函数值 true, 否则返回 false}

BEGIN

top:=0; i:=1; { i 指示待选物品}

WHILE (1)\_\_\_\_\_ AND (2)\_\_\_\_\_ DO

[IF (3)\_\_\_\_\_ OR (4)\_\_\_\_\_ AND (i<n)

THEN [top := (5)\_\_\_\_\_ ; stack[top] := i; {第 i 件物品装入背包}

T:=T-w[i];

IF T=0 THEN RETURN ((6)\_\_\_\_\_ ) {背包问题有解}

ELSE [IF (i=n ) AND (top>0)

THEN [i:=(7)\_\_\_\_\_ ; {取出栈顶物品}

top:= (8)\_\_\_\_\_ ; T:= (9)\_\_\_\_\_ ]; {恢复 T 值}

i:=i+1 {准备挑选下一件物品}

];

];

RETURN((10)\_\_\_\_\_ ) {背包无解}

END;

【北京邮电大学 1996 四 (10 分)】

#### 四 应用题

1. 名词解释: 栈。【燕山大学 1999 一、1 (2 分)】【吉林工业大学 1999 一、3 (2 分)】

2. 名词解释: 队列【大连海事大学 1996 一、6 (1 分)】

3. 什么是循环队列?【哈尔滨工业大学 2001 三、2 (3 分)】【河南大学 1998 一、4 (3 分)】

4. 假设以 S 和 X 分别表示入栈和出栈操作, 则对初态和终态均为空的栈操作可由 S 和 X 组成的序列表示 (如 SXSX)。

(1) 试指出判别给定序列是否合法的一般规则。

(2) 两个不同合法序列 (对同一输入序列) 能否得到相同的输出元素序列? 如能得到, 请举例说明。

【东南大学 1992 二 (10 分)】

5. 有 5 个元素, 其入栈次序为: A, B, C, D, E, 在各种可能的出栈次序中, 以元素 C, D 最先出栈 (即 C 第一个且 D 第二个出栈) 的次序有哪几个?【西南交通大学 2000 二、1】

6. 如果输入序列为 1 2 3 4 5 6, 试问能否通过栈结构得到以下两个序列: 4 3 5 6 1 2 和 1 3 5 4 2 6; 请说明为什么不能或如何才能得到。【武汉交通科技大学 1996 二、3 (3 分)】

7. 若元素的进栈序列为: A、B、C、D、E, 运用栈操作, 能否得到出栈序列 B、C、A、E、D 和 D、B、A、C、E? 为什么? 【北京科技大学 1998 一、2】

8. 设输入序列为 a, b, c, d, 试写出借助一个栈可得到的两个输出序列和两个不能得到的输出序列。

【北京科技大学 2001 一、4 (2 分)】

9. 设输入序列为 2, 3, 4, 5, 6, 利用一个栈能得到序列 2, 5, 3, 4, 6 吗? 栈可以用单链表实现吗?

【山东师范大学 1996 五、4 (2 分)】

10. 试证明: 若借助栈由输入序列  $1, 2, \dots, n$  得到输出序列为  $P_1, P_2, \dots, P_n$  (它是输入序列的一个排列), 则在输出序列中不可能出现这样的情形: 存在着  $i < j < k$ , 使  $P_j < P_k < P_i$ 。【上海交通大学 1998 二 (15 分)】

11. 设一数列的输入顺序为 123456, 若采用堆栈结构, 并以 A 和 D 分别表示入栈和出栈操作, 试问通过入出栈操作的合法序列。

(1) 能否得到输出顺序为 325641 的序列。(5 分)

(2) 能否得到输出顺序为 154623 的序列。(5 分) 【北方交通大学 1995 一 (10 分)】

12. (1) 什么是递归程序?

(2) 递归程序的优、缺点是什么?

(3) 递归程序在执行时, 应借助于什么来完成?

(4) 递归程序的入口语句、出口语句一般用什么语句实现? 【大连海事大学 1996 二、4 (4 分)】

13. 设有下列递归算法:

```
FUNCTION vol(n:integer):integer;
VAR x :integer;
BEGIN IF n=0 THEN vol:=0
 ELSE BEGIN read(x); vol:=vol(n-1)+x; END;
END;
```

如该函数被调用时, 参数 n 值为 4, 读入的 x 值依次为 5, 3, 4, 2, 函数调用结束时返回值 vol 为多少? 用图示描述函数执行过程中, 递归工作栈的变化过程。【北京工业大学 1998 四 (10 分)】

14. 当过程 P 递归调用自身时, 过程 P 内部定义的局部变量在 P 的 2 次调用期间是否占用同一数据区? 为什么? 【山东师范大学 1999 一、4 (4 分)】

15. 试推导出当总盘数为 n 的 Hanoi 塔的移动次数。【北京邮电大学 2001 四、3 (5 分)】

16. 对下面过程写出调用 P(3) 的运行结果。

```
PROCEDURE p (w: integer);
BEGIN
 IF w>0 THEN
 BEGIN
 p(w-1);
 writeln(w); {输出 W}
 p(w-1)
 END;
END;
```

【西北大学 2001 三、7】

17. 用一个数组 S (设大小为 MAX) 作为两个堆栈的共享空间。请说明共享方法, 栈满/栈空的判断条件, 并用 C 或 PASCAL 设计公用的入栈操作 push (i, x), 其中 i 为 0 或 1, 用于表示栈号,

x 为入栈值。

【浙江大学 1998 五、2 (7 分)】

18. 简述下列程序段的功能。

```
PROC algo(VAR S : stack; k:integer);
VAR T: stack; temp: integer;
 WHILE NOT empty(S) DO
 [temp:=POP(S); IF temp<>k THEN PUSH(T,temp)];
 WHILE NOT empty(T) DO [temp:=POP(T);PUSH(S,temp)];
```

【山东科技大学 2002 一、1 (4 分)】

19. 用栈实现将中缀表达式  $8-(3+5)*(5-6/2)$  转换成后缀表达式，画出栈的变化过程图。

【南京航空航天大学 2001 五 (10 分)】

20. 在表达式中，有的运算符要求从右到左计算，如  $A**B**C$  的计算次序应为  $(A**(B**C))$ ，这在中缀生成后缀的算法中是怎样实现的？(以  $**$  为例说明)【东南大学 1993 一、2 (6 分) 1997 一、1 (8 分)】

21. 有递归算法如下：

```
FUNCTION sum (n:integer):integer;
BEGIN
 IF n=0 THEN sum:=0
 ELSE BEGIN read(x); sum:=sum(n-1)+x END;
END;
```

设初值  $n=4$ ，读入  $x=4, 9, 6, 2$

问：(1) 若  $x$  为局部变量时；该函数递归结束后返回调用程序的  $sum=?$  并画出在递归过程中栈状态的变化过程；

(2) 若  $x$  为全程变量递归结束时返回调用程序的  $sum=?$ 【北京邮电大学 1997 一 (10 分)】

22. 画出对算术表达式  $A-B*C/D-E \uparrow F$  求值时操作数栈和运算符栈的变化过程。

【东南大学 2000 一、3 (6 分)】

23. 计算算术表达式的值时，可用两个栈作辅助工具。对于给出的一个表达式，从左向右扫描它的字符，并将操作数放入栈  $S_1$  中，运算符放入栈  $S_2$  中，但每次扫描到运算符时，要把它同  $S_2$  的栈顶运算符进行优先级比较，当扫描到的运算符的优先级不高于栈顶运算符的优先级时，取出栈  $S_1$  的栈顶和次栈顶的两个元素，以及栈  $S_2$  的栈顶运算符进行运算将结果放入栈  $S_1$  中（得到的结果依次用  $T_1$ 、 $T_2$  等表示）。为方便比较，假设栈  $S_2$  的初始栈顶为  $\oplus$  ( $\oplus$  运算符的优先级低于加、减、乘、除中任何一种运算)。现假设要计算表达式： $A-B*C/D+E/F$ 。写出栈  $S_1$  和  $S_2$  的变化过程。【山东科技大学 2001 一、4 (7 分)】

24. 有字符串次序为  $3*-y-a/y^2$ ，利用栈，给出将次序改为  $3y-*ay^2/-$  的操作步骤。(可用  $X$  代表扫描该字符串过程中顺序取一个字符进栈的操作，用  $S$  代表从栈中取出一个字符加入到新字符串尾的出栈操作。例如， $ABC$  变为  $BCA$  的操作步骤为  $XXSXSS$ )【东北大学 2001 一、4 (4 分)】

25. 内存中一片连续空间（不妨假设地址从 1 到  $m$ ）提供给两个栈  $S_1$  和  $S_2$  使用，怎样分配这部分存储空间，使得对任一个栈，仅当这部分空间全满时才发生上溢。【东北大学 2000 一、1 (3 分)】

26. 将两个栈存入数组  $V[1..m]$  应如何安排最好？这时栈空、栈满的条件是什么？【东南大学 1998 一、5】

27. 在一个算法中需要建立多个堆栈时可以选用下列三种方案之一，试问：这三种方案之间相比较各有什么优缺点？

(1) 分别用多个顺序存储空间建立多个独立的堆栈；



- (2) 多个堆栈共享一个顺序存储空间;
- (3) 分别建立多个独立的链接堆栈。【北京航空航天大学 1998 一、6 (4 分)】
28. 在某程序中, 有两个栈共享一个一维数组空间  $SPACE[N]$ 、 $SPACE[0]$ 、 $SPACE[N-1]$  分别是两个栈的栈底。
- (1) 对栈 1、栈 2, 试分别写出 (元素  $x$ ) 入栈的主要语句和出栈的主要语句。
- (2) 对栈 1、栈 2, 试分别写出栈满、栈空的条件。【北京理工大学 1999 二、2 (8 分)】
29. 简述顺序存储队列的假溢出的避免方法及队列满和空的条件。【山东大学 2000 一、2 (4 分)】
30. 举例说明顺序队的“假溢出”现象, 并给出解决方案。【福州大学 1998 三、5 (6 分)】
31. 怎样判定循环队列的空和满? 【燕山大学 1999 二、3 (4 分)】
32. 简要叙述循环队列的数据结构, 并写出其初始状态、队列空、队列满时的队首指针与队尾指针的值。
- 【南京航空航天大学 1995 七 (5 分)】
33. 利用两个栈  $s_1, s_2$  模拟一个队列时, 如何用栈的运算实现队列的插入, 删除以及判队空运算。请简述这些运算的算法思想。【北京邮电大学 1992 一、1】【东南大学 1999 一、1 (7 分)】
34. 一个循环队列的数据结构描述如下:
- ```

TYPE sequeuetp=RECORD
    elem: ARRAY[1..MAXSIZE] OF elemtp;
    front, rear: 0..MAXSIZE;
END;
```
- 给出循环队列的队空和队满的判断条件, 并且分析一下该条件对队列实际存储空间大小的影响, 如果为了不损失存储空间, 你如何改进循环队列的队空和队满的判断条件? 【西北工业大学 1999 三 (8 分)】
35. 如果用一个循环数组 $q[0..m-1]$ 表示队列时, 该队列只有一个队列头指针 $front$, 不设队列尾指针 $rear$, 而改置计数器 $count$ 用以记录队列中结点的个数。
- (1) 编写实现队列的三个基本运算: 判空、入队、出队 (3 分)
- (2) 队列中能容纳元素的最多个数是多少? (1 分) 【东北大学 2002 一、1】
36. 给出循环队列中元素个数的计算式 (设队最大长度为 N , 队首指针 $FRONT$, 队尾指针 $REAR$)
- 【西北大学 2000 二、7 (5 分)】
37. 顺序队列一般应该组织成为环状队列的形式, 而且一般队列头或尾其中之一应该特殊处理。例如, 队列为 $listarray[0..n-1]$, 队列头指针为 $front$, 队列尾指针为 $rear$, 则 $listarray[rear]$ 表示下一个可以插入队列的位置。请解释其原因。【北京大学 1999 一、3 (20/3 分)】
38. 设一个双端队列, 元素进入该队列的次序为 a, b, c, d 。求既不能由输入受限的双端队列得到, 又不能由输出受限的双端队列得到的输出序列。【中山大学 1999 一、4 (3 分)】
39. 若以 1、2、3、4 作为双端队列的输入序列, 试分别求出以下条件的输出序列:
- (1) 能由输入受限的双端队列得到, 但不能由输出受限的双端队列得到的输出序列;
- (2) 能由输出受限的双端队列得到, 但不能由输入受限的双端队列得到的输出序列;
- (3) 既不能由输入受限的双端队列得到, 也不能由输出受限的双端队列得到的输出序列。
- 【山东科技大学 2001 一、3 (6 分)】
40. 假设以数组 $sq[0..7]$ 存放循环队列元素, 变量 f 指向队头元素的前一位置, 变量 r 指向队尾元素, 如用 A 和 D 分别表示入队和出队操作, 请给出:
- (1) 队空的初始条件;
- (2) 执行操作序列 $A^3D^1A^5D^2A^1D^2A^4$ 时的状态, 并作必要的说明。【北方交通大学 1993 四 (12 分)】
41. 设输入元素为 1、2、3、 P 和 A , 输入次序为 123PA, 如图 (编者略)。元素经过栈后达输出

序列, 当所有元素均到达输出序列后, 有哪些序列可以作为高级语言的变量名。【中山大学 1997】

五 算法设计题

1. 设有两个栈 S_1, S_2 都采用顺序栈方式, 并且共享一个存储区 $[0..maxsize-1]$, 为了尽量利用空间, 减少溢出的可能, 可采用栈顶相向, 迎面增长的存储方式。试设计 S_1, S_2 有关入栈和出栈的操作算法。

【哈尔滨工业大学 2001 七 (12 分)】

2. 设从键盘输入一整数的序列: $a_1, a_2, a_3, \dots, a_n$, 试编写算法实现: 用栈结构存储输入的整数, 当 $a_i \neq -1$ 时, 将 a_i 进栈; 当 $a_i = -1$ 时, 输出栈顶整数并出栈。算法应对异常情况 (入栈满等) 给出相应的信息。

【南京航空航天大学 1998 六 (10 分)】

3. 设表达式以字符形式已存入数组 $E[n]$ 中, ‘#’ 为表达式的结束符, 试写出判断表达式中括号 (‘(’ 和 ‘)’) 是否配对的 C 语言描述算法: EXYX(E); (注: 算法中可调用栈操作的基本算法。)

【北京科技大学 2001 九、1 (10 分)】

4. 从键盘上输入一个逆波兰表达式, 用伪码写出其求值程序。规定: 逆波兰表达式的长度不超过一行, 以 \$ 符作为输入结束, 操作数之间用空格分隔, 操作符只可能有 +、-、*、/ 四种运算。例如: 234 34+2*\$

【山东师范大学 1999 七 (10 分)】

5. 假设以 I 和 O 分别表示入栈和出栈操作。栈的初态和终态均为空, 入栈和出栈的操作序列可表示为仅由 I 和 O 组成的序列, 称可以操作的序列为合法序列, 否则称为非法序列。

(1) 下面所示的序列中哪些是合法的?

A. IOIOIOIO B. IOOIIOIO C. IIIIOIOIO D. IIIIOOIOO

(2) 通过对 (1) 的分析, 写出一个算法, 判定所给的操作序列是否合法。若合法, 返回 true, 否则返回 false (假定被判定的操作序列已存入一维数组中)。【武汉大学 2000 五、2】

6. 设计一个算法, 判断一个算术表达式中的括号是否配对。算术表达式保存在带头结点的单循环链表中, 每个结点有两个域: ch 和 link, 其中 ch 域为字符类型。【南京邮电大学 2000 五】

7. 请利用两个栈 S_1 和 S_2 来模拟一个队列。已知栈的三个运算定义如下: PUSH(ST, x): 元素 x 入 ST 栈; POP(ST, x): ST 栈顶元素出栈, 赋给变量 x; Empty(ST): 判 ST 栈是否为空。那么如何利用栈的运算来实现该队列的三个运算: enqueue: 插入一个元素入队列; dequeue: 删除一个元素出队列; queue_empty: 判队列为空。(请写明算法的思想及必要的注释)

【西安电子科技大学 2001 软件五 (10 分)】【上海交通大学 1999 二 (12 分)】【河海大学 1998 三 (12 分)】

类似本题的另外叙述有:

(1) 有两个长度相同的栈 S_1, S_2 , 已知以下入栈、出栈、判栈满和判栈空操作:

PROCEDURE push(Stack:Stacktype;x:Datatype);

FUNCTION Pop(Stack:Stacktype):Datatype;

FUNCTION Full (Stack:Stacktype):Boolean;

FUNCTION Empty (Stack:Stacktype) Boolean;

现用此二栈构成一个队列, 试写出下面入队列、出队列操作算法:

PROCEDURE EnQueue(x:Datatype);

FUNCTION DeQueue: Datatype; 【北京邮电大学 2000 六 (10 分)】

8. 设结点结构为 (data, link), 试用一个全局指针 p 和某种链接结构实现一个队列, 画出示意图, 并给出入队 addq 和出队 deleteq 过程, 要求它们的时间复杂性都是 $O(1)$ (不计 new 和 dispose 时间)

【东南大学 1996 二 (10 分)】

9. 假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾结点，但不设头指针，如图所示(编者略)，请写出相应的入队列和出队列算法。【西安电子科技大学 1999 计应用 六 (10 分)】

10. 如果允许在循环队列的两端都可以进行插入和删除操作。要求：

(1) 写出循环队列的类型定义；

(2) 写出“从队尾删除”和“从队头插入”的算法。【北方交通大学 1994 三 (12 分)】

11. 在一个循环链队中只有尾指针(记为 rear，结点结构为数据域 data，指针域 next)，请给出这种队列的入队和出队操作的实现过程。【山东科技大学 2002 一、2 (6 分)】

12. 双端队列(duque)是一个可以在任一端进行插入和删除的线性表。现采用一个一维数组作为双端队列的数据存储结构，使用类 PASCAL 语言描述如下：

```
CONST maxsize=32;           {数组中可容纳的元素个数}
TYPE duque=RECORD
    elem: ARRAY[0..MAXSIZE-1] OF datatype;   {环形队列的存放数组}
    end1, end2: 0..MAXSIZE;                   {环形数组的两端}
END;
```

试编写两个算法 add (Qu:duque;x:datatype;tag:0..1) 和 delete (Qu:duque; var x:datatype; tag:0..1) 用以在此双端队列的任一端进行插入和删除。当 tag=0 时在左端 end1 端操作，当 tag=1 时在右端 end2 端操作。【清华大学 1998 二 (10 分)】

13. 一个双端队列 deque 是限定在两端 end1, end2 都可进行插入和删除的线性表。队空条件是 end1=end2。若用顺序方式来组织双端队列，试根据下列要求，定义双端队列的结构，并给出在指定端 i (i=1, 2) 的插入 enq 和删除 deq 操作的实现。

(1) 当队满时，最多只能有一个元素空间可以是空的。

(2) 在做两端的插入和删除时，队列中其它元素一律不动。【清华大学 1999 六 (12 分)】

14. 已知 Q 是一个非空队列，S 是一个空栈。仅用队列和栈的 ADT 函数和少量工作变量，使用 Pascal 或 C 语言编写一个算法，将队列 Q 中的所有元素逆置。栈的 ADT 函数有：

```
makeEmpty(s:stack);           置空栈
push(s:stack;value:datatype); 新元素 value 进栈
pop(s:stack):datatype;         出栈，返回栈顶值
isEmpty(s:stack):Boolean;      判栈空否
```

队列的 ADT 函数有：

```
enqueue(q:queue:value:datatype); 元素 value 进队
deQueue(q:queue):datatype;        出队列，返回队头值
isEmpty(q:queue):boolean;         判队列空否 【清华大学 2000 六 (12 分)】
```

15. 将 n 个队列顺序映射到数组 v[1..m] 中，每一队列在 v 中表示为一循环队列。试画出其示意图并写出对应这种表示的 addq 和 deleteq 过程。【东南大学 1993 二 (20 分)】

16. 设整数序列 a_1, a_2, \dots, a_n ，给出求解最大值的递归程序。【南京航空航天大学 2000 六】

17. 线性表中元素存放在向量 A (1, ..., n) 中，元素是整型数。试写出递归算法求出 A 中的最大和最小元素。【北京邮电大学 1994 八 (10 分)】

18. 已知求两个正整数 m 与 n 的最大公因子的过程用自然语言可以表述为反复执行如下动作：第一步：若 n 等于零，则返回 m；第二步：若 m 小于 n，则 m 与 n 相互交换；否则，保存 m，然后将 n 送 m，将保存的 m 除以 n 的余数送 n。

(1) 将上述过程用递归函数表达出来(设求 x 除以 y 的余数可以用 x MOD y 形式表示)。

(2) 写出求解该递归函数的非递归算法。【北京航空航天大学 2001 五 (15 分)】

19. 写出和下列递归过程等价的非递归过程。

```
PROCEDURE test(VAR sum:integer);
VAR a:integer;
BEGIN
    read(a);
    IF a=0 THEN sum:=1
        ELSE BEGIN test(sum); sum:=sum*a; END;
    write(sum);
END;
```

【清华大学 1996 二】

20. 试将下列递归过程改写为非递归过程。

```
void test(int &sum)
{ int x;
  scanf(x);
  if(x=0) sum=0 else {test(sum); sum+=x;}
  printf(sum);
}
```

【北京轻工业学院 2000 三 (15 分)】

21. 已知 Ackermann 函数定义如下：

$$Ack(m,n)=\begin{cases} n+1 & \text{当 } m=0 \text{ 时} \\ Ack(m-1,1) & \text{当 } m \neq 0, n=0 \text{ 时} \\ Ack(m-1, Ack(m,n-1)) & \text{当 } m \neq 0, n \neq 0 \text{ 时} \end{cases}$$

(1) 写出 Ack(2, 1) 的计算过程。

(2) 写出计算 Ack(m, n) 的非递归算法。【北京航空航天大学 1999 六 (15 分)】

22. 设计算法以求解从集合 {1..n} 中选取 k (k ≤ n) 个元素的所有组合。例如，从集合 {1..4} 中选取 2 个元素的所有组合的输出结果为：1 2, 1 3, 1 4, 2 3, 2 4, 3 4。

【合肥工业大学 2000 五、5 (8 分)】

第三章 栈和队列答案

一、选择题

1. B	2. 1B	2. 2A	2. 3B	2. 4D	2. 5. C	3. B	4. D	5. D	6. C	7. D	8. B
9. D	10. D	11. D	12. C	13. B	14. C	15. B	16. D	17. B	18. B	19. B	20. D
21. D	22. D	23. D	24. C	25. A	26. A	27. D	28. B	29. BD	30. C	31. B	32. C
33. 1B	33. 2A	33. 3C	33. 4C	33. 5F	34. C	35. C	36. A	37. AD			

二、判断题

1. √	2. √	3. √	4. √	5. ×	6. √	7. √	8. √	9. √	10. ×	11. √	12. ×
13. ×	14. ×	15. √	16. ×	17. √	18. ×	19. √	20. √				

部分答案解释如下。

1、尾递归的消除就不需用栈

2、这个数是前序序列为 1,2,3,...,n，所能得到的不相似的二叉树的数目。

三、填空题

1、操作受限（或限定仅在表尾进行插入和删除操作） 后进先出

2、栈 3、3 1 2 4、23 100CH 5、0 $n+1$ $top[1]+1=top[2]$

6、两栈顶指针值相减的绝对值为 1（或两栈顶指针相邻）。

7、(1)满 (2)空 (3)n (4)栈底 (5)两栈顶指针相邻（即值之差的绝对值为 1）

8、链式存储结构 9、 $S \times SS \times S \times \times$ 10、 $data[++top]=x$;

11、23. 12. $3*2-4/34. 5*7/++108. 9/+$ （注：表达式中的点(.)表示将数隔开，如 23. 12. 3 是三个数）

12、假溢出时大量移动数据元素。

- 13、 $(M+1) \bmod N$ $(M+1) \% N$; 14、队列 15、先进先出 16、先进先出
 17、 $s = (\text{LinkedList})\text{malloc}(\text{sizeof}(\text{LNode}))$; $s \rightarrow \text{data} = x$; $s \rightarrow \text{next} = r \rightarrow \text{next}$; $r \rightarrow \text{next} = s$; $r = s$;
 18、牺牲一个存储单元 设标记
 19、 $(\text{TAIL}+1) \bmod M = \text{FRONT}$ (数组下标 0 到 M-1, 若一定使用 1 到 M, 则取模为 0 者, 值改取 M)
 20 、 $\text{sq.front} = (\text{sq.front} + 1) \% (M+1)$; **return**($\text{sq.data}(\text{sq.front})$) ;
 $(\text{sq.rear} + 1) \% (M+1) == \text{sq.front}$;
 21、栈 22、 $(\text{rear} - \text{front} + m) \% m$; 23、 $(R - P + N) \% N$;
 24、(1) $a[i]$ 或 $a[1]$ (2) $a[i]$ (3) $\text{pop}(s)$ 或 $s[1]$;
 25、(1) $\text{PUSH}(\text{OPTR}, w)$ (2) $\text{POP}(\text{OPTR})$ (3) $\text{PUSH}(\text{OPND}, \text{operate}(a, \text{theta}, b))$
 26、(1) $T > 0$ (2) $i < n$ (3) $T > 0$ (4) $\text{top} < n$ (5) $\text{top} + 1$ (6) true (7) $i - 1$ (8) $\text{top} - 1$ (9) $T + w[i]$
 (10) false

四、应用题

1、栈是只准在一端进行插入和删除操作的线性表, 允许插入和删除的一端叫栈顶, 另一端叫栈底。最后插入的元素最先删除, 故栈也称后进先出 (LIFO) 表。

2、队列是允许在一端插入而在另一端删除的线性表, 允许插入的一端叫队尾, 允许删除的一端叫队头。最先插入队的元素最先离开 (删除), 故队列也常称先进先出 (FIFO) 表。

3、用常规意义下顺序存储结构的一维数组表示队列, 由于队列的性质 (队尾插入和队头删除), 容易造成“假溢出”现象, 即队尾已到达一维数组的高下标, 不能再插入, 然而队中元素个数小于队列的长度 (容量)。循环队列是解决“假溢出”的一种方法。通常把一维数组看成首尾相接。在循环队列下, 通常采用“牺牲一个存储单元”或“作标记”的方法解决“队满”和“队空”的判定问题。

4、(1) 通常有两条规则。第一是给定序列中 S 的个数和 X 的个数相等; 第二是从给定序列的开始, 到给定序列中的任一位置, S 的个数要大于或等于 X 的个数。

(2) 可以得到相同的输出元素序列。例如, 输入元素为 A, B, C, 则两个输入的合法序列 ABC 和 BAC 均可得到输出元素序列 ABC。对于合法序列 ABC, 我们使用本题约定的 $S \times S \times S \times$ 操作序列; 对于合法序列 BAC, 我们使用 $SS \times \times S \times$ 操作序列。

5、三个: CDEBA, CDBEA, CDBAE

6、输入序列为 123456, 不能得出 435612, 其理由是, 输出序列最后两元素是 12, 前面 4 个元素 (4356) 得到后, 栈中元素剩 12, 且 2 在栈顶, 不可能栈底元素 1 在栈顶元素 2 之前出栈。

得到 135426 的过程如下: 1 入栈并出栈, 得到部分输出序列 1; 然后 2 和 3 入栈, 3 出栈, 部分输出序列变为: 13; 接着 4 和 5 入栈, 5, 4 和 2 依次出栈, 部分输出序列变为 13542; 最后 6 入栈并退栈, 得最终结果 135426。

7、能得到出栈序列 B、C、A、E、D, 不能得到出栈序列 D、B、A、C、E。其理由为: 若出栈序列以 D 开头, 说明在 D 之前的入栈元素是 A、B 和 C, 三个元素中 C 是栈顶元素, B 和 A 不可能早于 C 出栈, 故不可能得到 D、B、A、C、E 出栈序列。

8、借助栈结构, n 个入栈元素可得到 $1/(n+1) ((2n)! / (n! * n!))$ 种出栈序列。本题 4 个元素, 可有 14 种出栈序列, abcd 和 dcba 就是其中两种。但 dabc 和 adbc 是不可能得到的两种。

9、不能得到序列 2, 5, 3, 4, 6。栈可以用单链表实现, 这就是链栈。由于栈只在栈顶操作, 所以链栈通常不设头结点。

10、如果 $i < j$, 则对于 $p_i < p_j$ 情况, 说明 p_i 在 p_j 入栈前先出栈。而对于 $p_i > p_j$ 的情况, 则说明要将 p_j 压到 p_i 之上, 也就是在 p_j 出栈之后 p_i 才能出栈。这就说明, 对于 $i < j < k$, 不可能出现 $p_j < p_k < p_i$ 的输出序列。换句话说, 对于输入序列 1, 2, 3, 不可能出现 3, 1, 2 的输出序列。

11、(1) 能得到 325641。在 123 依次进栈后, 3 和 2 出栈, 得部分输出序列 32; 然后 4, 5 入栈, 5 出栈, 得部分出栈序列 325; 6 入栈并出栈, 得部分输出序列 3256; 最后退栈, 直到栈空。得输出序列 325641。其操作序列为 AAADDAADADDD。

(2) 不能得到输出顺序为 154623 的序列。部分合法操作序列为 ADAAAADDAD, 得到部分输出序列 1546 后, 栈中元素为 23, 3 在栈顶, 故不可能 2 先出栈, 得不到输出序列 154623。

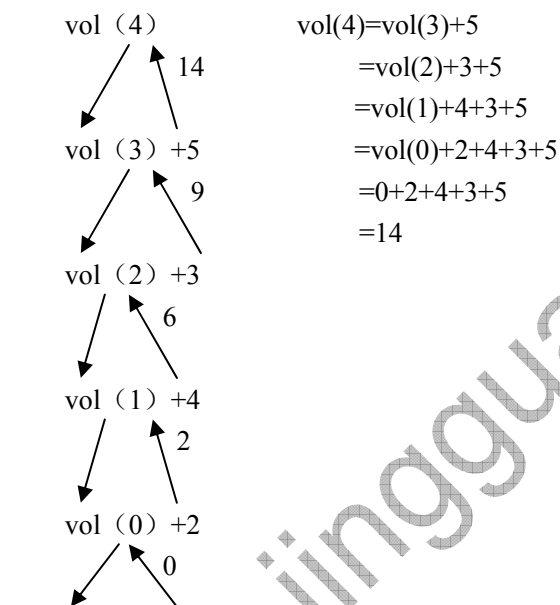
12、(1) 一个函数在结束本函数之前, 直接或间接调用函数自身, 称为递归。例如, 函数 f 在执行中, 又调用函数 f 自身, 这称为直接递归; 若函数 f 在执行中, 调用函数 g , 而 g 在执行中, 又调用函数 f , 这称为间接递归。在实际应用中, 多为直接递归, 也常简称为递归。

(2) 递归程序的优点是程序结构简单、清晰, 易证明其正确性。缺点是执行中占内存空间较多, 运行效率低。

(3) 递归程序执行中需借助栈这种数据结构来实现。

(4) 递归程序的入口语句和出口语句一般用条件判断语句来实现。递归程序由基本项和归纳项组成。基本项是递归程序出口, 即不再递归即可求出结果的部分; 归纳项是将原来问题化成简单的且与原来形式一样的问题, 即向着“基本项”发展, 最终“到达”基本项。

13、函数调用结束时 $vol=14$ 。执行过程图示如下:



14、过程 p 递归调用自身时, 过程 p 由内部定义的局部变量在 p 的 2 次调用期间, 不占同一数据区。每次调用都保留其数据区, 这是递归定义所决定, 用“递归工作栈”来实现。

15、设 H_n 为 n 个盘子的 Hanoi 塔的移动次数。(假定 n 个盘子从钢针 X 移到钢针 Z , 可借助钢针 Y)

则 $H_n = 2H_{n-1} + 1$ // 先将 $n-1$ 个盘子从 X 移到 Y , 第 n 个盘子移到 Z , 再将那 $n-1$ 个移到 Z

$$\begin{aligned}
 &= 2(2H_{n-2} + 1) + 1 \\
 &= 2^2 H_{n-2} + 2 + 1 \\
 &= 2^2 (2H_{n-3} + 1) + 2 + 1 \\
 &= 2^3 H_{n-3} + 2^2 + 2 + 1 \\
 &\quad \cdot \\
 &\quad \cdot \\
 &\quad \cdot \\
 &= 2^k H_{n-k} + 2^{k-1} + 2^{k-2} + \cdots + 2^1 + 2^0
 \end{aligned}$$

$$=2^{n-1} H_1 + 2^{n-2} + 2^{n-3} + \dots + 2^1 + 2^0$$

因为 $H_1=1$ ，所以原式 $H_n=2^{n-1}+2^{n-2}+\dots+2^1+2^0=2^n-1$

故总盘数为 n 的 Hanoi 塔的移动次数是 2^n-1 。

16、运行结果为：1 2 1 3 1 2 1（注：运行结果是每行一个数，为节省篇幅，放到一行。）

17、两栈共享一向量空间（一维数组），栈底设在数组的两端，两栈顶相邻时为栈满。设共享数组为 $S[\text{MAX}]$ ，则一个栈顶指针为 -1，另一个栈顶指针为 MAX 时，栈为空。

用 C 写的入栈操作 $\text{push}(i, x)$ 如下：

const MAX =共享栈可能达到的最大容量

typedef struct node

{**elemtype s**[MAX];

int top[2];

}**anode;**

anode ds;

int push(int i, elemtype x)

//ds 为容量有 MAX 个类型为 elemtype 的元素的一维数组，由两个栈共享其空间。 i 的值为 0 或 1， x 为类型为 elemtype 的元素。本算法将 x 压入栈中。如压栈成功，返回 1；否则，返回 0。

{**if** ($\text{ds.top}[1]-\text{ds.top}[0]==1$) {**printf** (“栈满\n”); **return** (0); }

switch (i)

{**case** 0: $\text{ds.s}[\text{++ds.top}[i]]=x$; **break**;

case 1: $\text{ds.s}[\text{--ds.top}[i]]=x$;

return (1); }//入栈成功。

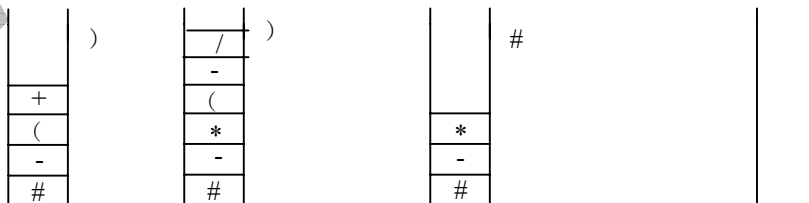
}

18、本程序段查找栈 S 中是否有整数为 k 的元素，如有，则删除。采用的办法使用另一个栈 T 。在 S 栈元素退栈时，若退栈元素不是整数 k ，则压入 T 栈。遇整数 k ， k 不入 T 栈，然后将 T 栈元素全部退栈，并依次压入栈 S 中，实现了在 S 中删除整数 k 的目的。若 S 中无整数 k ，则在 S 退成空栈后，再将 T 栈元素退栈，并依次压入 S 栈。直至 T 栈空。这后一种情况下 S 栈内容操作前后不变。

19、中缀表达式 $8-(3+5)*(5-6/2)$ 的后缀表达式是：8 3 5 + 5 6 2 / - * -

栈的变化过程图略（请参见 22 题），表达式生成过程为：

(1) 8 3 5 (2) 8 3 5 + 5 6 2 (3) 8 3 5 + 5 6 2 / - (4) 8 3 5 + 5 6 2 / - * -



中缀表达式 exp1 转为后缀表达式 exp2 的规则如下：

设操作符栈 s ，初始为空栈后，压入优先级最低的操作符 ‘#’。对中缀表达式从左向右扫描，遇操作数，直接写入 exp2 ；若是操作符（记为 w ），分如下情况处理，直至表达式 exp1 扫描完毕。

(1) w 为一般操作符（‘+’，‘-’，‘*’，‘/’等），要与栈顶操作符比较优先级，若 w 优先级高于栈顶操作符，则入栈；否则，栈顶运算符退栈到 exp2 ， w 再与新栈顶操作符作上述比较处理，

直至 w 入栈。

(2) w 为左括号 ('('), w 入栈。

(3) w 为右括号 (')'), 操作符栈退栈并进入 exp2 , 直到碰到左括号为止, 左括号退栈 (不能进入 exp2), 右括号也丢掉, 达到 exp2 中消除括号的目的。

(4) w 为 '#', 表示中缀表达式 exp1 结束, 操作符栈退栈到 exp2 , 直至碰到 '#', 退栈, 整个操作结束。

这里, 再介绍一种简单方法。中缀表达式转为后缀表达式有步骤: 首先, 将中缀表达式中所有的子表达式按计算规则用嵌套括号括起来; 接着, 顺序将每对括号中的运算符移到相应括号的后面; 最后, 删除所有括号。

例如, 将中缀表达式 $8-(3+5)*(5-6/2)$ 转为后缀表达式。按如上步骤:

执行完上面第一步后为: $(8-((3+5)*(5-(6/2))))$;

执行完上面第二步后为: $(8((35)+(5(62)/)-)*-)$;

执行完上面第三步后为: $8\ 3\ 5\ +\ 5\ 6\ 2\ /\ -\ *\ -$ 。

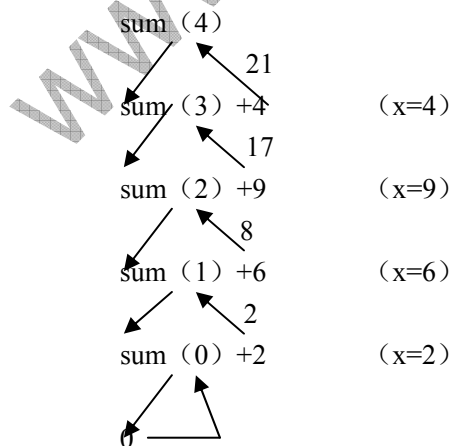
可用类似方法将中缀表达式转为前缀表达式。

20、中缀表达式转为后缀表达式的规则基本上与上面 19 题相同, 不同之处是对运算符**优先级的规定。在算术运算中, 先乘除后加减, 先括号内后括号外, 相同级别的运算符按从左到右的规则运算。而对**运算符, 其优先级同常规理解, 即高于加减乘除而小于左括号。为了适应本题中“从右到左计算”的要求, 规定栈顶运算符**的级别小于正从表达式中读出的运算符**, 即刚读出的运算符**级别高于栈顶运算符**, 因此也入栈。

下面以 $A**B**C$ 为例说明实现过程。

读入 A , 不是操作符, 直接写入结果表达式。再读入 $*$, 这里规定在读入 $*$ 后, 不能立即当乘号处理, 要看下一个符号, 若下个符号不是 $*$, 则前个 $*$ 是乘号。这里因为下一个待读的符号也是 $*$, 故认为 $**$ 是一个运算符, 与运算符栈顶比较 (运算符栈顶初始化后, 首先压入 '#' 作为开始标志), 其级别高于 '#', 入栈。再读入 B , 直接进入结果表达式。接着读入 $**$, 与栈顶比较, 均为 $**$, 我们规定, 后读入的 $**$ 级别高于栈顶的 $**$, 因此 $**$ 入栈。接着读入 C , 直接到结果表达式。现在的结果 (后缀) 表达式是 ABC 。最后读入 '#', 表示输入表达式结束, 这时运算符栈中从栈顶到栈底有两个 $**$ 和一个 '#', 两个运算符 $**$ 退栈至结果表达式, 结果表达式变为 $ABC****$ 。运算符栈中只剩 '#', 退栈, 运算结束。

21、(1) $\text{sum}=21$ 。当 x 为局部变量时, 每次递归调用, 都要给局部变量分配存储单元, 故 x 数值 4, 9, 6 和 2 均保留, 其递归过程示意图如下:



(2) $\text{sum}=8$, 当 x 为全局变量时, 在程序的整个执行期间, x 只占一个存储单元, 先后读入的 4

个数(4,9,6,2),仅最后一个起作用。当递归调用结束,逐层返回时 $\text{sum}:=\text{sum}(n-1)+x$ 表达式中, x 就是 2, 所以结果为 $\text{sum}=8$ 。

22、设操作数栈是 opnd, 操作符栈是 optr, 对算术表达式 $A-B*C/D-E \uparrow F$ 求值, 过程如下:

步骤	opnd 栈	optr 栈	输入字符	主要操作
初始		#	$A-B*C/D-E \uparrow F\#$	PUSH(OPTR, ' #')
1	A	#	$A-B*C/D-E \uparrow F\#$	PUSH(OPND, A)
2	A	# -	$A-B*C/D-E \uparrow F\#$	PUSH(OPTR, ' -')
3	AB	# -	$A-B*C/D-E \uparrow F\#$	PUSH(OPND, B)
4	AB	# - *	$A-B*C/D-E \uparrow F\#$	PUSH(OPTR, ' *')
5	ABC	# - *	$A-B*C/D-E \uparrow F\#$	PUSH(OPND, C)
6	AT ($T=B*C$)	# - /	$A-B*C/D-E \uparrow F\#$	PUSH(OPND, POP(OPND)*POP(OPND)) PUSH(OPTR, ' /')
7	ATD	# - /	$A-B*C/D-E \uparrow F\#$	PUSH(OPND, D)
8	AT ($T=T/D$) T ($T=A-T$)	# - # -	$A-B*C/D-E \uparrow F\#$	$x=\text{POP(OPND)}; y=\text{POP(OPND)}$ PUSH(OPND, y/x); $x=\text{POP(OPND)}; y=\text{POP(OPND)}$; PUSH(OPND, $y-x$) PUSH(OPTR, ' -')
9	TE	# -	$A-B*C/D-E \uparrow F\#$	PUSH(OPND, E)
10	TE	# - \uparrow	$A-B*C/D-E \uparrow F\#$	PUSH(OPTR, ' \uparrow ')
11	TEF	# - \uparrow	$A-B*C/D-E \uparrow F\#$	PUSH(OPND, F)
12	TE TS ($S=E \uparrow F$) R ($R=T-S$)	# # #	$A-B*C/D-E \uparrow F\#$	$X=\text{POP(OPND)} \quad Y=\text{POP(OPND)}$ POP(OPTR) PUSH(OPND, $y \uparrow x$) $x=\text{POP(OPND)} \quad y=\text{POP(OPND)}$ POP(OPTR) PUSH(OPND, $y-x$)

23、

步骤	栈 S1	栈 S2	输入的算术表达式 (按字符读入)
初始		®	$A-B*C/D+E/F\textcircled{R}$
<u>1</u>	A	®	$A-B*C/D+E/F\textcircled{R}$
<u>2</u>	A	®-	$A-B*C/D+E/F\textcircled{R}$
<u>3</u>	AB	®-	$A-B*C/D+E/F\textcircled{R}$
<u>4</u>	AB	®*	$A-B*C/D+E/F\textcircled{R}$
<u>5</u>	ABC	®*	$A-B*C/D+E/F\textcircled{R}$
<u>6</u>	AT ₁ (注: $T_1=B*C$)	®/	$A-B*C/D+E/F\textcircled{R}$
<u>7</u>	AT ₁ D	®/	$A-B*C/D+E/F\textcircled{R}$
<u>8</u>	AT ₂ (注: $T_2=T_1/D$)	®-	$A-B*C/D+E/F\textcircled{R}$

	T_3 (注: $T_3=A-T_2$)	$\textcircled{R}+$	
<u>9</u>	T_3E	$\textcircled{R}+$	$\underline{E}/F^{\textcircled{R}}$
<u>10</u>	T_3E	$\textcircled{R}+/\textcircled{R}$	$\underline{E}/F^{\textcircled{R}}$
<u>11</u>	T_3EF	$\textcircled{R}+/\textcircled{R}$	$\underline{F}^{\textcircled{R}}$
<u>12</u>	T_3T_4 (注: $T_4=E/F$) T_5 (注: $T_5= T_3+ T_4$)	$\textcircled{R}+$ \textcircled{R}	$\underline{\textcircled{R}}$

24、XSXXXSSSSXSXXXSSSS

25、S1 和 S2 共享内存中一片连续空间 (地址 1 到 m)，可以将 S1 和 S2 的栈底设在两端，两栈顶向共享空间的中心延伸，仅当两栈顶指针相邻 (两栈顶指针值之差的绝对值等于 1) 时，判断为栈满，当一个栈顶指针为 0，另一个栈顶指针 m+1 时为两栈均空。

26、设栈 S1 和栈 S2 共享向量 $V[1..m]$ ，初始时，栈 S1 的栈顶指针 $\text{top}[0]=0$ ，栈 S2 的栈顶指针 $\text{top}[1]=m+1$ ，当 $\text{top}[0]=0$ 为左栈空， $\text{top}[1]=m+1$ 为右栈空；当 $\text{top}[0]=0$ 并且 $\text{top}[1]=m+1$ 时为全栈空。当 $\text{top}[1]-\text{top}[0]=1$ 时为栈满。

27、(1) 每个栈仅用一个顺序存储空间时，操作简便，但分配存储空间小了，容易产生溢出，分配空间大了，容易造成浪费，各栈不能共享空间。

(2) 多个栈共享一个顺序存储空间，充分利用了存储空间，只有在整个存储空间都用完时才能产生溢出，其缺点是当一个栈满时要向左、右栈查询有无空闲单元。如果有，则要移动元素和修改相关的栈底和栈顶指针。当接近栈满时，查询空闲单元、移动元素和修改栈底栈顶指针的操作频繁，计算复杂并且耗费时间。

(3) 多个链栈一般不考虑栈的溢出 (仅受用户内存空间限制)，缺点是栈中元素要以指针相链接，比顺序存储多占用了存储空间。

28、设 top1 和 top2 分别为栈 1 和 2 的栈顶指针

(1) 入栈主要语句

```
if(top2-top1==1) {printf("栈满\n"); exit(0);}
case1:top1++; SPACE[top1]=x; //设 x 为入栈元素。
case2:top2--; SPACE[top2]=x;
```

出栈主要语句

```
case1: if (top1==1) {printf("栈空\n"); exit(0);}
        top1--; return (SPACE[top1+1]); //返回出栈元素。
case2: if (top2==N) {printf("栈空\n"); exit(0);}
        top2++; return (SPACE[top2-1]); //返回出栈元素。
```

(2) 栈满条件: $\text{top2}-\text{top1}=1$

栈空条件: $\text{top1}=-1$ 并且 $\text{top2}=N$ // $\text{top1}=-1$ 为左栈空， $\text{top2}=N$ 为右栈空

29、设顺序存储队列用一维数组 $q[m]$ 表示，其中 m 为队列中元素个数，队列中元素在向量中的下标从 0 到 $m-1$ 。设队头指针为 front ，队尾指针是 rear ，约定 front 指向队头元素的前一位置， rear 指向队尾元素。当 front 等于 -1 时队空， rear 等于 $m-1$ 时为队满。由于队列的性质 (删除在队头而插入在队尾)，所以当队尾指针 rear 等于 $m-1$ 时，若 front 不等于 -1，则队列中仍有空闲单元，所以队列并不是真满。这时若再有入队操作，会造成假“溢出”。其解决办法有二，一是将队列元素向前“平移” (占用 0 至 $\text{rear}-\text{front}-1$)；二是将队列看成首尾相连，即循环队列 (0.. $m-1$)。在循环队列下，仍定义 $\text{front}=\text{rear}$ 时为队空，而判断队满则用两种办法，一是用“牺牲一个单元”，即 $\text{rear}+1=\text{front}$ (准确记是 $(\text{rear}+1)\%m=\text{front}$ ， m 是队列容量) 时为队满。另一种解法是“设标记”方法，如设标记 tag ， tag 等于 0 情况下，若删除时导致 $\text{front}=\text{rear}$ 为队空； $\text{tag}=1$ 情况下，若因插入导致 $\text{front}=\text{rear}$ 则为队满。

30、见上题 29 的解答。 31、参见上面 29 题。

32、typedef struct node

```
{elemtype elemcq[m]; //m 为队列最大可能的容量。
    int front ,rear; //front 和 rear 分别为队头和队尾指针。
}cqnode;
```

cqnode cq;

(1) 初始状态

cq.front=cq.rear=0;

(2) 队列空

cq.front==cq.rear;

(3) 队列满

(cq.rear+1)%m==cq.front;

33、栈的特点是后进先出，队列的特点是先进先出。初始时设栈 s1 和栈 s2 均为空。

(1) 用栈 s1 和 s2 模拟一个队列的输入：设 s1 和 s2 容量相等。分以下三种情况讨论：若 s1 未滿，则元素入 s1 栈；若 s1 满，s2 空，则将 s1 全部元素退栈，再压栈入 s2，之后元素入 s1 栈；若 s1 满，s2 不空（已有出队列元素），则不能入队。

(2) 用栈 s1 和 s2 模拟队列出队（删除）：若栈 s2 不空，退栈，即是队列的出队；若 s2 为空且 s1 不空，则将 s1 栈中全部元素退栈，并依次压入 s2 中，s2 栈顶元素退栈，这就是相当于队列的出队。若栈 s1 为空并且 s2 也为空，队列空，不能出队。

(3) 判队空 若栈 s1 为空并且 s2 也为空，才是队列空。

讨论：s1 和 s2 容量之和是队列的最大容量。其操作是，s1 栈满后，全部退栈并压栈入 s2（设 s1 和 s2 容量相等）。再入栈 s1 直至 s1 满。这相当队列元素“入队”完毕。出队时，s2 退栈完毕后，s1 栈中元素依次退栈到 s2，s2 退栈完毕，相当于队列中全部元素出队。

在栈 s2 不空情况下，若要求入队操作，只要 s1 不满，就可压入 s1 中。若 s1 满和 s2 不空状态下要求队列的入队时，按出错处理。

34、(1) 队空 s.front=s.rear; //设 s 是 sequenuep 类型变量

(2) 队满：(s.rear+1) MOD MAXSIZE=s.front //数组下标为 0.. MAXSIZE-1

具体参见本章应用题第 29 题

35、typedef struct

```
{elemtp q[m];
    int front,count; //front 是队首指针，count 是队列中元素个数。
}cqnode; //定义类型标识符。
```

(1) 判空：int Empty(cqnode cq) //cq 是 cqnode 类型的变量

{if(cq.count==0) return(1); else return(0); //空队列}

入队：int EnQueue(cqnode cq, elemtp x)

{if(count==m){printf(“队满\n”); exit(0); }

cq.q[(cq.front+count)%m]=x; //x 入队

count++; return(1); //队列中元素个数增加 1，入队成功。

}

出队：int DelQueue(cqnode cq)

{if (count==0){printf(“队空\n”); return(0);}

printf(“出队元素”，cq.q[cq.front]);

x=cq.q[cq.front];

cq.front=(cq.front+1)%m; //计算新的队头指针。

return(x)

}

(2) 队列中能容纳的元素个数为 m 。队头指针 $front$ 指向队头元素。

36、循环队列中元素个数为 $(REAR - FRONT + N) \% N$ 。其中 $FRONT$ 是队首指针，指向队首元素的前一位置； $REAR$ 是队尾指针，指向队尾元素； N 是队列最大长度。

37、循环队列解决了用向量表示队列所出现的“假溢出”问题，但同时又出现了如何判断队列的满与空问题。例如：在队列长 10 的循环队列中，若假定队头指针 $front$ 指向队头元素的前一位置，而队尾指针指向队尾元素，则 $front=3, rear=7$ 的情况下，连续出队 4 个元素，则 $front==rear$ 为队空；如果连续入队 6 个元素，则 $front==rear$ 为队满。如何判断这种情况下的队满与队空，一般采取牺牲一个单元的做法或设标记法。即假设 $front==rear$ 为队空，而 $(rear+1) \% \text{表长} == front$ 为队满，或通过设标记 tag 。若 $tag=0, front==rear$ 则为队空；若 $tag=1$ ，因入队而使得 $front==rear$ ，则为队满。

本题中队列尾指针 $rear$ ，指向队尾元素的下一位置， $listarray[rear]$ 表示下一个入队的元素。在这种情况下，我们可规定，队头指针 $front$ 指向队首元素。当 $front==rear$ 时为队空，当 $(rear+1) \% n == front$ 时为队满。出队操作（在队列不空情况下）队头指针是 $front = (front+1) \% n$ ，

38、既不能由输入受限的双端队列得到，也不能由输出受限的双端队列得到的输出序列是 $dbca$ 。

39、(1) 4132 (2) 4213 (3) 4231

40、(1) 队空的初始条件： $f=r=0$ ；

(2) 执行操作 A^3 后， $r=3$ ；// A^3 表示三次入队操作

执行操作 D^1 后， $f=1$ ；// D^1 表示一次出队操作

执行操作 A^5 后， $r=0$ ；

执行操作 D^2 后， $f=3$ ；

执行操作 A^1 后， $r=1$ ；

执行操作 D^2 后， $f=5$ ；

执行操作 A^4 后，按溢出处理。因为执行 A^3 后， $r=4$ ，这时队满，若再执行 A 操作，则出错。

41. 一般说，高级语言的变量名是以字母开头的字母数字序列。故本题答案是：

AP321, PA321, P3A21, P32A1, P321A。

五、算法设计题

1、[题目分析]两栈共享向量空间，将两栈栈底设在向量两端，初始时， $s1$ 栈顶指针为 -1 ， $s2$ 栈顶为 $maxsize$ 。两栈顶指针相邻时为栈满。两栈顶相向，迎面增长，栈顶指针指向栈顶元素。

#define maxsize 两栈共享顺序存储空间所能达到的最多元素数

#define elemtp int //假设元素类型为整型

typedef struct

{elemtp stack[maxsize]; //栈空间

int top[2]; //top 为两个栈顶指针

}stk;

stk s; //s 是如上定义的结构类型变量，为全局变量。

(1) 入栈操作：

int push(int i, int x)

//入栈操作。i 为栈号， $i=0$ 表示左边的栈 $s1$ ， $i=1$ 表示右边的栈 $s2$ ， x 是入栈元素。入栈成功返回 1，否则返回 0。

{if ($i < 0$ || $i > 1$) {printf(“栈号输入不对”); exit(0);}}

```

if(s.top[1]-s.top[0]==1) {printf(“栈已满\n”);return(0);}
switch(i)
{case 0: s.stack[++s.top[0]]=x; return(1); break;
 case 1: s.stack[--s.top[1]]=x; return(1);
 }
} //push

```

(2) 退栈操作

```

elemtp pop(int i)
//退栈算法。i 代表栈号，i=0 时为 s1 栈，i=1 时为 s2 栈。退栈成功返回退栈元素，否则返回-1。
{if(i<0 || i>1) {printf(“栈号输入错误\n”); exit(0);}
 switch(i)
 {case 0: if(s.top[0]==-1) {printf(“栈空\n”); return (-1);}
           else return(s.stack[s.top[0]--]);
 case 1: if(s.top[1]==maxsize {printf(“栈空\n”); return(-1);}
           else return(s.stack[s.top[1]++]);
 }
} //算法结束

```

[算法讨论] 请注意算法中两栈入栈和退栈时的栈顶指针的计算。两栈共享空间示意图略，s1 栈是通常意义下的栈，而 s2 栈入栈操作时，其栈顶指针左移（减 1），退栈时，栈顶指针右移（加 1）。

2、#define maxsize 栈空间容量

```

void InOutS(int s[maxsize])
//s 是元素为整数的栈，本算法进行入栈和退栈操作。
{int top=0; //top 为栈顶指针，定义 top=0 时为栈空。
 for(i=1; i<=n; i++) //n 个整数序列作处理。
 {scanf(“%d”,&x); //从键盘读入整数序列。
  if(x!=-1) //读入的整数不等于-1 时入栈。
   if(top==maxsize-1) {printf(“栈满\n”);exit(0);} else s[++top]=x; //x 入栈。
  else //读入的整数等于-1 时退栈。
   {if(top==0) {printf(“栈空\n”);exit(0);} else printf(“出栈元素是 %d\n”,s[top--]); }
 } //算法结束。
}

```

3、[题目分析] 判断表达式中括号是否匹配，可通过栈，简单说是左括号时进栈，右括号时退栈。退栈时，若栈顶元素是左括号，则新读入的右括号与栈顶左括号就可消去。如此下去，输入表达式结束时，栈为空则正确，否则括号不匹配。

```

int EXYX(char E[], int n)
//E[] 是有 n 字符的字符数组，存放字符串表达式，以 ‘#’ 结束。本算法判断表达式中圆括号是否匹配。
{char s[30]; //s 是一维数组，容量足够大，用作存放括号的栈。
 int top=0; //top 用作栈顶指针。
 s[top]= ‘#’ ; // ‘#’ 先入栈，用于和表达式结束符号 ‘#’ 匹配。
 int i=0; //字符数组 E 的工作指针。
 while(E[i] != ‘#’ ) //逐字符处理字符表达式的数组。

```

```

switch(E[i])
{case '(' : s[++top]= '(' ; i++ ; break ;
 case ')' : if(s[top]== '(' {top--; i++; break;}
              else{printf(“括号不配对”);exit(0);}
 case '#' : if(s[top]== '#') {printf(“括号配对\n”);return (1);}
              else {printf(“ 括号不配对\n”);return (0);} //括号不配对
 default : i++; //读入其它字符，不作处理。
}
} //算法结束。

```

[算法讨论]本题是用栈判断括号匹配的特例：只检查圆括号的配对。一般情况是检查花括号（‘{’，‘}’）、方括号（‘[’，‘]’）和圆括号（‘(’，‘)’）的配对问题。编写算法中如遇左括号（‘{’，‘[’，或‘(’）就压入栈中，如遇右括号（‘}’，‘]’，或‘)’），则与栈顶元素比较，如是与其配对的括号（左花括号，左方括号或左圆括号），则弹出栈顶元素；否则，就结论括号不配对。在读入表达式结束符‘#’时，栈中若应只剩‘#’，表示括号全部配对成功；否则表示括号不匹配。

另外，由于本题只是检查括号是否匹配，故对从表达式中读入的不是括号的那些字符，一律未作处理。再有，假设栈容量足够大，因此入栈时未判断溢出。

4、[题目分析]逆波兰表达式(即后缀表达式)求值规则如下：设立运算数栈 OPND, 对表达式从左到右扫描(读入)，当表达式中扫描到数时，压入 OPND 栈。当扫描到运算符时，从 OPND 退出两个数，进行相应运算，结果再压入 OPND 栈。这个过程一直进行到读出表达式结束符\$，这时 OPND 栈中只有一个数，就是结果。

```

float expr( )
//从键盘输入逆波兰表达式，以‘$’表示输入结束，本算法求逆波兰式表达式的值。
{float OPND[30]; // OPND 是操作数栈。
 init(OPND); //两栈初始化。
 float num=0.0; //数字初始化。
 scanf ( “%c”, &x); //x 是字符型变量。
 while(x!=' $')
 {switch
 {case '0' <=x<=' 9' :while((x>=' 0' &&x<=' 9' )||x==' .') //拼数
 if(x!=' .') //处理整数
 {num=num*10+ (ord(x)-ord( '0' )); scanf( “%c”, &x);}
 else //处理小数部分。
 {scale=10.0; scanf( “%c”, &x);
 while(x>=' 0' &&x<=' 9' )
 {num=num+(ord(x)-ord( '0' ))/scale;
 scale=scale*10; scanf( “%c”, &x); }
 } //else
 push(OPND, num); num=0.0; //数压入栈，下个数字初始化
 case x= ' ' :break; //遇空格，继续读下一个字符。
 case x= '+' :push(OPND, pop(OPND)+pop(OPND)); break;
 case x= '-' :x1=pop(OPND); x2=pop(OPND); push(OPND, x2-x1); break;
 case x= '*' :push(OPND, pop(OPND)*pop(OPND)); break;
 case x= '/' :x1=pop(OPND); x2=pop(OPND); push(OPND, x2/x1); break;

```

```

        default:          //其它符号不作处理。
    } //结束 switch
    scanf( "%c" , &x); //读入表达式中下一个字符。
    } //结束 while (x != '$')
    printf( "后缀表达式的值为%f" , pop(OPND));
} //算法结束。

```

[算法讨论]假设输入的后缀表达式是正确的,未作错误检查。算法中拼数部分是核心。若遇到大于等于‘0’且小于等于‘9’的字符,认为是数。这种字符的序号减去字符‘0’的序号得出数。对于整数,每读入一个数字字符,前面得到的部分数要乘上10再加新读入的数得到新的部分数。当读到小数点,认为数的整数部分已完,要接着处理小数部分。小数部分的数要除以10(或10的幂数)变成十分位,百分位,千分位数等等,与前面部分数相加。在拼数过程中,若遇非数字字符,表示数已拼完,将数压入栈中,并且将变量num恢复为0,准备下一个数。这时对新读入的字符进入‘+’、‘-’、‘*’、‘/’及空格的判断,因此在结束处理数字字符的case后,不能加入break语句。

5、(1) A和D是合法序列, B和C是非法序列。

(2) 设被判定的操作序列已存入一维数组A中。

```

int Judge(char A[])
//判断字符数组A中的输入输出序列是否是合法序列。如是,返回true,否则返回false。
{
    i=0;          //i为下标。
    j=k=0;        //j和k分别为I和字母O的个数。
    while(A[i]!='\0') //当未到字符数组尾就作。
    {
        switch(A[i])
        {
            case 'I': j++; break; //入栈次数增1。
            case 'O': k++; if(k>j){printf("序列非法\n"); exit(0);}
        }
        i++; //不论A[i]是'I'或'O',指针i均后移。
    }
    if(j!=k){printf("序列非法\n"); return(false);}
    else {printf("序列合法\n"); return(true);}
} //算法结束。

```

[算法讨论]在入栈出栈序列(即由‘I’和‘O’组成的字符串)的任一位置,入栈次数(‘I’的个数)都必须大于等于出栈次数(即‘O’的个数),否则视作非法序列,立即给出信息,退出算法。整个序列(即读到字符数组中字符串的结束标记‘\0’),入栈次数必须等于出栈次数(题目中要求栈的初态和终态都为空),否则视为非法序列。

6、[题目分析]表达式中的括号有以下三对:‘(’、‘)’、‘[’、‘]’、‘{’、‘}’,使用栈,当为左括号时入栈,右括号时,若栈顶是其对应的左括号,则退栈,若不是其对应的左括号,则结论为括号不配对。当表达式结束,若栈为空,则结论表达式括号配对,否则,结论表达式括号不配对。

```

int Match(LinkedList la)
//算术表达式存储在以la为头结点的单循环链表中,本算法判断括号是否正确配对
{
    char s[];          //s为字符栈,容量足够大
    p=la->link;        //p为工作指针,指向待处理结点
    StackInit(s);       //初始化栈s
    while (p!=la)       //循环到头结点为止
    {
        switch (p->ch)
        {
            case '(': push(s, p->ch); break;

```

```

case '(' : if(StackEmpty(s) || StackGetTop(s) != '(' )
            {printf( "括号不配对\n" ); return(0);} else pop(s);break;
case '[' : push(s, p->ch); break;
case ']' : if(StackEmpty(s) || StackGetTop(s) != '[' )
            {printf( "括号不配对\n" ); return(0);} else pop(s);break;
case '{' : push(s, p->ch); break;
case '}' : if(StackEmpty(s) || StackGetTop(s) != '{' )
            {printf( "括号不配对\n" ); return(0);} else pop(s);break;
} p=p->link; 后移指针
} //while
if (StackEmpty(s)) {printf( "括号配对\n" ); return(1);}
else {printf( "括号不配对\n" ); return(0);}
} //算法 match 结束

```

[算法讨论] 算法中对非括号的字符未加讨论。遇到右括号时，若栈空或栈顶元素不是其对应的左圆（方、花）括号，则结论括号不配对，退出运行。最后，若栈不空，仍结论括号不配对。

7、[题目分析] 栈的特点是后进先出，队列的特点是先进先出。所以，用两个栈 s1 和 s2 模拟一个队列时，s1 作输入栈，逐个元素压栈，以此模拟队列元素的入队。当需要出队时，将栈 s1 退栈并逐个压入栈 s2 中，s1 中最先入栈的元素，在 s2 中处于栈顶。s2 退栈，相当于队列的出队，实现了先进先出。显然，只有栈 s2 为空且 s1 也为空，才算是队列空。

(1) int enqueue(stack s1, elemtp x)

//s1 是容量为 n 的栈，栈中元素类型是 elemtp。本算法将 x 入栈，若入栈成功返回 1，否则返回 0。

```

{if(top1==n && !Empty(s2)) //top1 是栈 s1 的栈顶指针，是全局变量。
    {printf( "栈满" );return(0);} //s1 满 s2 非空,这时 s1 不能再入栈。
if(top1==n && Empty(s2)) //若 s2 为空，先将 s1 退栈,元素再压栈到 s2。
    {while(!Empty(s1)) {POP(s1, x); PUSH(s2, x);}
    PUSH(s1, x); return(1); //x 入栈，实现了队列元素的入队。
}

```

(2) void dequeue(stack s2, s1)

//s2 是输出栈，本算法将 s2 栈顶元素退栈，实现队列元素的出队。

```

{if(!Empty(s2)) //栈 s2 不空，则直接出队。
    {POP(s2, x); printf( "出队元素为", x); }
else //处理 s2 空栈。
    if(Empty(s1)) {printf( "队列空" );exit(0);} //若输入栈也 为空，则判定队
空。
else //先将栈 s1 倒入 s2 中，再作出队操作。
    {while(!Empty(s1)) {POP(s1, x); PUSH(s2, x);}
    POP(s2, x); //s2 退栈相当队列出队。
    printf( "出队元素", x);
}
} //结束算法 dequeue。

```

(3) int queue_empty()

//本算法判用栈 s1 和 s2 模拟的队列是否为空。

```

{if(Empty(s1)&&Empty(s2)) return(1); //队列空。
}

```

```

else return(0);           //队列不空。
}

```

[算法讨论]算法中假定栈 s1 和栈 s2 容量相同。出队从栈 s2 出，当 s2 为空时，若 s1 不空，则将 s1 倒入 s2 再出栈。入队在 s1，当 s1 满后，若 s2 空，则将 s1 倒入 s2，之后再入队。因此队列的容量为两栈容量之和。元素从栈 s1 倒入 s2，必须在 s2 空的情况下才能进行，即在要求出队操作时，若 s2 空，则不论 s1 元素多少（只要不空），就要全部倒入 s2 中。

类似本题叙述的其它题的解答：

(1) 该题同上面题本质相同，只有叙述不同，请参考上题答案。

8、[题目分析]本题要求用链接结构实现一个队列，我们可用链表结构来实现。一般说，由于队列的先进先出性质，所以队列常设队头指针和队尾指针。但题目中仅给出一个“全局指针 p”，且要求入队和出队操作的时间复杂性是 O(1)，因此我们用只设尾指针的循环链表来实现队列。

```

(1) PROC addq(VAR p:linkedList,x:elemtp);
    //p 是数据域为 data、链域为 link 的用循环链表表示的队列的尾指针，本算法是入队操作。
    new(s);           //申请新结点。假设有内存空间，否则系统给出出错信息。
    s↑.data:=x; s↑.link:=p↑.link; //将 s 结点入队。
    p↑.link:=s; p:=s;           //尾指针 p 移至新的队尾。
ENDP;

(2) PROC deleq(VAR p:linkedList,VAR x:elemtp);
    // p 是数据域为 data、链域为 link 的用循环链表表示的队列的尾指针，本算法实现队列元素的出队，若出队成功，返回出队元素，否则给出失败信息。
    IF (p↑.link=p) THEN[writeln(“空队列”);return(0);]//带头结点的循环队列。
    ELSE[s:=p↑.link↑.link;           //找到队头元素。
        p↑.link↑.link:=s↑.link;       //删队头元素。
        x:=s↑.data;                   //返回出队元素。
        IF (p=s) THEN p:=p↑.link;     //队列中只有一个结点，出队后成为空队列。
        dispose(s);                   //回收出队元素所占存储空间。
    ]
ENDP;

```

[算法讨论]上述入队算法中，因链表结构，一般不必考虑空间溢出问题，算法简单。在出队算法中，首先要判断队列是否为空，另外，对出队元素，要判断是否因出队而成为空队列。否则，可能导致因删除出队结点而将尾指针删掉成为“悬挂变量”。

9、本题与上题本质上相同，现用类 C 语言编写入队和出队算法。

```

(1) void EnQueue (LinkedList rear, ElemType x)
    // rear 是带头结点的循环链队列的尾指针，本算法将元素 x 插入到队尾。
    { s= (LinkedList) malloc (sizeof(LNode)); //申请结点空间
      s->data=x; s->next=rear->next;           //将 s 结点链入队尾
      rear->next=s; rear=s;                   //rear 指向新队尾
    }

(2) void DeQueue (LinkedList rear)
    // rear 是带头结点的循环链队列的尾指针，本算法执行出队操作，操作成功输出队头元素；否则给出出错信息。
    { if (rear->next==rear) { printf(“队空\n”); exit(0);}
      s=rear->next->next;                       //s 指向队头元素，
    }

```

```

    rear->next->next=s->next;        //队头元素出队。
    printf(“出队元素是”, s->data);
    if (s==rear) rear=rear->next;    //空队列
    free(s);
}

```

10、[题目分析] 用一维数组 $v[0..M-1]$ 实现循环队列，其中 M 是队列长度。设队头指针 $front$ 和队尾指针 $rear$ ，约定 $front$ 指向队头元素的前一位置， $rear$ 指向队尾元素。定义 $front=rear$ 时为队空， $(rear+1)\%m=front$ 为队满。约定队头端入队向下标小的方向发展，队尾端入队向下标大的方向发展。

(1) #define M 队列可能达到的最大长度

```

typedef struct
{
    elemtp data[M];
    int front, rear;
} cycqueue;

```

(2) elemtp delqueue (cycqueue Q)

//Q 是如上定义的循环队列，本算法实现从队尾删除，若删除成功，返回被删除元素，否则给出出错信息。

```

{ if (Q.front==Q.rear) {printf(“队列空”); exit(0);}
  Q.rear=(Q.rear-1+M)%M;        //修改队尾指针。
  return(Q.data[(Q.rear+1+M)%M]); //返回出队元素。
} //从队尾删除算法结束

```

void enqueue (cycqueue Q, elemtp x)

// Q 是顺序存储的循环队列，本算法实现“从队头插入”元素 x 。

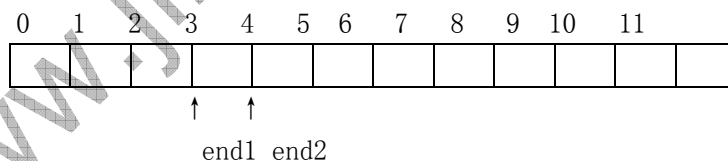
```

{if (Q.rear==(Q.front-1+M)%M) {printf(“队满”); exit(0);}
  Q.data[Q.front]=x;        //x 入队列
  Q.front=(Q.front-1+M)%M;   //修改队头指针。
} // 结束从队头插入算法。

```

11、参见 9。

12、[题目分析] 双端队列示意图如下（设 $maxsize = 12$ ）



用上述一维数组作存储结构，把它看作首尾相接的循环队列。可以在任一端（ $end1$ 或 $end2$ ）进行插入或删除。初始状态 $end1+1=end2$ 被认为是队空状态； $end1=end2$ 被认为是队满状态。（左端队列） $end1$ 指向队尾元素的前一位置。 $end2$ 指向（右端队列）队尾元素的下一位置。入队时判队满，出队（删除）时判队空。删除一个元素时，首先查找该元素，然后，从队尾将该元素前的元素依次向后或向前（视 $end1$ 端或 $end2$ 端而异）移动。

FUNC add (Qu:deque; var x:datatype;tag 0..1):integer;

//在双端队列 Qu 中插入元素 x ，若插入成功，返回插入元素在 Qu 中的下标；插入失败返回-1。

tag=0 表示在 $end1$ 端插入；tag=1 表示在 $end2$ 端插入。

IF Qu. $end1$ =Qu. $end2$ **THEN** [writeln(“队满”);return(-1);]

CASE tag **OF**

0: //在 $end1$ 端插入


```

[Qu.end1:=x;                                //插入 x
Qu.end1:=(Qu.end1-1) MOD maxsize; //修改 end1
RETURN(Qu.end1+1) MOD maxsize); //返回插入元素的下标。
1: //在 end2 端插入
[Qu.end2:=x;
Qu.end2:=(Qu.end2+1) MOD maxsize;
RETURN(Qu.end2-1) MOD maxsize);
]
ENDC; //结束 CASE 语句
ENDF; //结束算法 add
FUNC delete (Qu: deque; VAR x:datatype; tag:0..1):integer;
//本算法在双端队列 Qu 中删除元素 x, tag=0 时从 end1 端删除, tag=1 时从 end2 端删除。删除
成功返回 1, 否则返回 0。
IF (Qu.end1+1) MOD maxsize=Qu.end2 THEN [writeln(“队空”);return(0);]
CASE tag OF
0: //从 end1 端删除
[i:=(Qu.end1+1) MOD maxsize; //i 是 end1 端最后插入的元素下标。
WHILE(i<>Qu.end2) AND (Qu.elem[i]<>x) DO
i=(i+1) MOD maxsize; //查找被删除元素 x 的位置
IF (Qu.elem[i]=x) AND (i<>Qu.end2) THEN
[ j:=i;
WHILE((j-1+maxsize) MOD maxsize <>Qu.end1) DO
[Qu.elem[j]:=Qu.elem[(j-1+maxsize) MOD maxsize];
j:=(j-1+maxsize) MOD maxsize;
]//移动元素, 覆盖达到删除
Qu.end1:=(Qu.end1+1) MOD maxsize; //修改 end1 指针
RETURN(1);
]
ELSE RETURN(0);
]//结束从 end1 端删除。
1: //从 end2 端删除
[i:=(Qu.end2-1+maxsize) MOD maxsize; //i 是 end2 端最后插入的元素下标。
WHILE(i<>Qu.end1) AND (Qu.elem[i]<>x) DO
i=(i-1+maxsize) MOD maxsize; //查找被删除元素 x 的下标
IF (Qu.elem[i]=x) AND (i<>Qu.end1) THEN //被删除元素找到
[ j:=i;
WHILE((j+1) MOD maxsize <>Qu.end2) DO
[Qu.elem[j]:=Qu.elem[(j+1) MOD maxsize];
j:=(j+1) MOD maxsize;
]//移动元素, 覆盖达到删除
Qu.end2:=(Qu.end2-1+maxsize) MOD maxsize; //修改 end2 指针
RETURN(1); //返回删除成功的信息
]
ELSE RETURN(0); //删除失败

```

]//结束在 end2 端删除。

ENDC;//结束 CASE 语句

ENDF;//结束 delete

[算法讨论] 请注意下标运算。 $(i+1) \text{ MOD } \text{maxsize}$ 容易理解, 考虑到 $i-1$ 可能为负的情况, 所以求下个 i 时用了 $(i-1+\text{maxsize}) \text{ MOD } \text{maxsize}$ 。

13、[题目分析] 本题与上面 12 题基本相同, 现用类 C 语言给出该双端队列的定义。

```
#define maxsize 32
```

```
typedef struct
```

```
{datatype elem[maxsize];
```

```
int end1, end2; //end1 和 end2 取值范围是 0..maxsize-1
```

```
} deque;
```

14、[题目分析] 根据队列先进先出和栈后进先出的性质, 先将非空队列中的元素出队, 并压入初始为空的栈中。这时栈顶元素是队列中最后出队的元素。然后将栈中元素出栈, 依次插入到初始为空的队列中。栈中第一个退栈的元素成为队列中第一个元素, 最后退栈的元素(出队时第一个元素)成了最后入队的元素, 从而实现了原队列的逆置。

```
void Invert(queue Q)
```

//Q 是一个非空队列, 本算法利用空栈 S 和已给的几个栈和队列的 ADT 函数, 将队列 Q 中的元素逆置。

```
{makeempty(S); //置空栈
```

```
while (!isEmpty(Q)) // 队列 Q 中元素出队
```

```
{value=deQueue(Q); push(S, value); } // 将出队元素压入栈中
```

```
while(!isEmpty(S)) //栈中元素退栈
```

```
{value=pop(S); enQueue(Q, value); } //将出栈元素入队列 Q
```

```
} //算法 invert 结束
```

15、为运算方便, 设数组下标从 0 开始, 即数组 $v[0..m-1]$ 。设每个循环队列长度(容量)为 L , 则循环队列的个数为 $n=\lceil m/L \rceil$ 。为了指示每个循环队列的队头和队尾, 设如下结构类型

```
typedef struct
```

```
{int f, r;
```

```
} scq;
```

```
scq q[n];
```

(1) 初始化的核心语句

```
for(i=1; i<=n; i++) q[i].f=q[i].r=(i-1)*L; //q[i]是全局变量
```

(2) 入队 `int addq(int i; elemtp x)`

//n 个循环队列共享数组 $v[0..m-1]$ 和保存各循环队列首尾指针的 $q[n]$ 已经定义为全局变量, 数组元素为 elemtp 类型, 本过程将元素插入到第 i 个循环队列中。若入队成功, 返回 1, 否则返回队满标记 0 (入队失败)。

```
{ if (i<1||i>n) {printf("队列号错误");exit(0);}
```

```
if (q[i].r+1)%L+(i-1)*L==q[i].f) {printf("队满\n");exit(0);}
```

```
q[i].r=(q[i].r+1)%L+(i-1)*L; // 计算入队位置
```

```
v[q[i].r]=x; return(1); //元素 x 入队
```

```
}
```

(3) 出队 `int deleteq (int i)`

// n 个循环队列共享数组 $v[0..m-1]$ 和保存各循环队列首尾指针的 $q[n]$ 已经定义为全局变量, 数组元素为 elemtp 类型, 本过程将第 i 个循环队列出队。若出队成功, 打印出队列元素,

并返回 1 表示成功；若该循环队列为空，返回 0 表示出队失败。

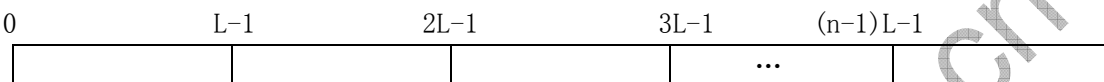
```

if (<1||>n) {printf("队列号错误\n");exit(0);}
if (q[i].r==q[i].f) {printf("队空\n");return(0);}
q[i].f=(q[i].f+1)%L+(i-1)*L;
printf("出队元素",q[i].f);return(1);
}

```

(4) 讨论，上述算法假定最后一个循环队列的长度也是 L ，否则要对最后一个循环队列作特殊处理。另外，未讨论一个循环队列满而相邻循环队列不满时，需修改个循环队列首尾指针的情况（即各循环队列长度不等）。

n 个循环队列共享数组 $v[0..m-1]$ 的示意图如下：



第 i 个循环队列从下标 $(i-1)L$ 开始，到 $iL-1$ 为止。设每个循环队列均用牺牲一个单元的办法来判断队满，即为 $(q[i].r+1) \% L + (i-1) * L = q[i].f$ 时，判定为队满。

16、int MaxValue (int a[],int n)

//设整数序列存于数组 a 中，共有 n 个，本算法求解其最大值。

```

if (n==1) max=a[1];
else if a[n]>MaxValue(a,n-1) max=a[n];
    else max=MaxValue(a,n-1);
return(max);
}

```

17、本题与上题类似，只是这里是同时求 n 个数中的最大值和最小值的递归算法。

int MinMaxValue(int A[],int n,int *max,int *min)

//一维数组 A 中存放有 n 个整型数，本算法递归的求出其中的最小数。

```

if (n>0)
{if(*max<A[n]) *max=A[n];
  if(*min>A[n]) *min=A[n];
  MinMaxValue(A,n-1,max,min);
} //算法结束

```

[算法讨论]调用本算法的格式是 $\text{MinMaxValue}(\text{arr},n,\&\text{max},\&\text{min})$ ；其中， arr 是具有 n 个整数的一维数组， $\text{max}=-32768$ 是最大数的初值， $\text{min}=32767$ 是最小数的初值。

18、[题目分析] 求两个正整数 m 和 n 的最大公因子，本题叙述的运算方法叫辗转相除法，也称欧几里德定理。其函数定义为：

$$\text{gcd}(m,n) = \begin{cases} m & \text{若 } n = 0 \\ \text{gcd}(n,m\%n) & \text{若 } n > 0 \end{cases}$$

int gcd (int m,n)

//求正整数 m 和 n 的最大公因子的递归算法

```

if(m<n) return(gcd(n,m)); //若 m<n, 则 m 和 n 互换
if(n==0) return(m); else return(gcd(n,m%n));
} //算法结束

```

使用栈，消除递归的非递归算法如下：

int gcd(int m,n)

{int s[max][2]; //s 是栈，容量 max 足够大

```

top=1; s[top][0]=m; s[top][1]=n;
while (s[top][1]!=0)
    if (s[top][0]<s[top][1]) //若 m<n, 则交换两数
        {t=s[top][0]; s[top][0]=s[top][1]; s[top][1]=t;}
    else {t=s[top][0]%s[top][1]; top++; s[top][0]=s[top-1][1]; s[top][1]=t; }
return(s[top][0]);
} //算法结束

```

由于是尾递归, 可以不使用栈, 其非递归算法如下

```

int gcd (int m,n)
//求正整数 m 和 n 的最大公因子
{if (m<n) {t=m;m=n;n=t;}} // 若 m<n, 则 m 和 n 互换
while (n!=0) {t=m; m=n; n=t%n;}
return(m);
} //算法结束

```

- 19、[题目分析] 这是以读入数据的顺序为相反顺序进行累乘问题, 可将读入数据放入栈中, 到输入结束, 将栈中数据退出进行累乘。累乘的初值为 1。

```

PROC test;
CONST maxsize=32;
VAR s:ARRAY[1..maxsize] OF integer, top,sum,a:integer;
[ top:=0; sum:=1; //
read(a);
WHILE a<>0 DO
    [ top:=top+1; s[top]:=a; read(a); ]
write(sum:5);
WHILE top>0 DO
    [ sum:=sum*s[top]; top:=top-1; write(sum:5); ]
ENDP;

```

- 20、[题目分析] 本题与第 19 题基本相同, 不同之处就是求和, 另外用 C 描述。

```

int test;
{int x, sum=0, top=0, s[];
scanf("%d",&x)
while (x<>0)
    {s[++top]:=a; scanf("%d",&x); }
printf(sum:5);
while (top)
    {sum+=s[top--]; printf(sum:5); }
};

```

- 21、int Ack(int m,n)
 {if (m==0) return(n+1);
 else if (m!=0&& n==0) return(Ack(m-1,1));
 else return(Ack(m-1,Ack(m,m-1)));
 } //算法结束

- (1) Ack(2,1) 的计算过程

Ack(2,1)=Ack(1,Ack(2,0)) //因 m<>0, n<>0 而得

```

=Ack(1, Ack(1, 1))           //因  $m < 0$ ,  $n = 0$  而得
=Ack(1, Ack(0, Ack(1, 0)))    // 因  $m < 0$ ,  $n < 0$  而得
= Ack(1, Ack(0, Ack(0, 1)))    // 因  $m < 0$ ,  $n = 0$  而得
=Ack(1, Ack(0, 2))           // 因  $m = 0$  而得
=Ack(1, 3)                   // 因  $m = 0$  而得
=Ack(0, Ack(1, 2))           //因  $m < 0$ ,  $n < 0$  而得
= Ack(0, Ack(0, Ack(1, 1)))    //因  $m < 0$ ,  $n < 0$  而得
= Ack(0, Ack(0, Ack(0, Ack(1, 0)))) //因  $m < 0$ ,  $n < 0$  而得
= Ack(0, Ack(0, Ack(0, Ack(0, 1)))) //因  $m < 0$ ,  $n = 0$  而得
= Ack(0, Ack(0, Ack(0, 2)))    //因  $m = 0$  而得
= Ack(0, Ack(0, 3))           //因  $m = 0$  而得
= Ack(0, 4)                   //因  $n = 0$  而得
=5                             //因  $n = 0$  而得

```

(2) `int Ackerman(int m, int n)`

```

{int akm[M][N]; int i, j;
for(j=0; j<N; j++) akm[0][j]=j+1;
for(i=1; i<m; i++)
{akm[i][0]=akm[i-1][1];
for(j=1; j<N; j++)
akm[i][j]=akm[i-1][akm[i][j-1]];
}
return(akm[m][n]);
} //算法结束

```

22、[题目分析]从集合 $(1..n)$ 中选出 k (本题中 $k=2$) 个元素，为了避免重复和漏选，可分别求出包括 1 和不包括 1 的所有组合。即包括 1 时，求出集合 $(2..n)$ 中取出 $k-1$ 个元素的所有组合；不包括 1 时，求出集合 $(2..n)$ 中取出 k 个元素的所有组合。将这两种情况合到一起，就是题目的解。

`int A[], n;` //设集合已存于数组 A 中。

`void comb(int P[], int i, int k)`

//从集合 $(1..n)$ 中选取 k ($k \leq n$) 个元素的所有组合

`{if (k==0) printf(P);`

`else if(k<=n) {P[i]=A[i]; comb(P, i+1, k-1); comb(P, i+1, k); }`

`} //算法结束`

第四章 串

一、选择题

- 下面关于串的叙述中,哪一个是不正确的? () 【北方交通大学 2001 一、5 (2分)】
A. 串是字符的有限序列 B. 空串是由空格构成的串
C. 模式匹配是串的一种重要运算 D. 串既可以采用顺序存储,也可以采用链式存储
- 若串 $S_1 = 'ABCDEFG'$, $S_2 = '9898'$, $S_3 = '###'$, $S_4 = '012345'$, 执行 $\text{concat}(\text{replace}(S_1, \text{substr}(S_1, \text{length}(S_2), \text{length}(S_3)), S_3), \text{substr}(S_4, \text{index}(S_2, '8'), \text{length}(S_2)))$ 其结果为 () 【北方交通大学 1999 一、5 (25/7分)】
A. ABC###G0123 B. ABCD###2345 C. ABC###G2345 D. ABC###2345
E. ABC###G1234 F. ABCD###1234 G. ABC###01234
- 设有两个串 p 和 q , 其中 q 是 p 的子串, 求 q 在 p 中首次出现的位置的算法称为 ()
A. 求子串 B. 联接 C. 匹配 D. 求串长
【北京邮电大学 2000 二、4 (20/8分)】 【西安电子科技大学 1996 一、1 (2分)】
- 已知串 $S = 'aaab'$, 其 Next 数组值为 ()。 【西安电子科技大学 1996 一、7 (2分)】
A. 0123 B. 1123 C. 1231 D. 1211
- 串 $'ababaaababaa'$ 的 next 数组为 ()。 【中山大学 1999 一、7】
A. 012345678999 B. 012121111212 C. 011234223456 D. 0123012322345
- 字符串 $'ababaabab'$ 的 nextval 为 ()
A. (0, 1, 0, 1, 0, 1, 0, 1)
B. (0, 1, 0, 1, 0, 2, 1, 0, 1)
C. (0, 1, 0, 1, 0, 0, 0, 1, 1)
D. (0, 1, 0, 1, 0, 1, 0, 1, 1)
【北京邮电大学 1999 一、1 (2分)】
- 模式串 $t = 'abcaabbcaabab'$, 该模式串的 next 数组的值为 (), nextval 数组的值为 ()。
A. 0 1 1 1 2 2 1 1 1 2 3 4 5 6 7 1 2 B. 0 1 1 1 2 1 2 1 1 2 3 4 5 6 1 1 2
C. 0 1 1 1 0 0 1 3 1 0 1 1 0 0 7 0 1 D. 0 1 1 1 2 2 3 1 1 2 3 4 5 6 7 1 2
E. 0 1 1 0 0 1 1 1 0 1 1 0 0 1 7 0 1 F. 0 1 1 0 2 1 3 1 0 1 1 0 2 1 7 0 1
【北京邮电大学 1998 二、3 (2分)】
- 若串 $S = 'software'$, 其子串的数目是 ()。 【西安电子科技大学 2001 应用 一、2 (2分)】
A. 8 B. 37 C. 36 D. 9
- 设 S 为一个长度为 n 的字符串, 其中的字符各不相同, 则 S 中的互异的非平凡子串 (非空且不同于 S 本身) 的个数为 ()。 【中科院计算所 1997】
A. $2n-1$ B. n^2 C. $(n^2/2) + (n/2)$ D. $(n^2/2) + (n/2) - 1$ E. $(n^2/2) - (n/2) - 1$ F. 其他情况
- 串的长度是指 () 【北京工商大学 2001 一、6 (3分)】
A. 串中所含不同字母的个数 B. 串中所含字符的个数
C. 串中所含不同字符的个数 D. 串中所含非空格字符的个数

二、判断题

- KMP 算法的特点是在模式匹配时指示主串的指针不会变小。 () 【北京邮电大学 2002 一、4 (1分)】
- 设模式串的长度为 m , 目标串的长度为 n , 当 $n \approx m$ 且处理只匹配一次的模式时, 朴素的匹配 (即子串定位函数) 算法所花的时间代价可能会更为节省。 () 【长沙铁道学院 1998 一、1 (1分)】

3. 串是一种数据对象和操作都特殊的线性表。() 【大连海事大学 2001 1、L (1 分)】

二、填空题

1. 空格串是指 (1), 其长度等于 (2)。 【西安电子科技大学 2001 软件 一、4 (2 分)】
2. 组成串的数据元素只能是_____。 【中山大学 1998 一、5 (1 分)】
3. 一个字符串中_____称为该串的子串。 【华中理工大学 2000 一、3 (1 分)】
4. INDEX ('DATASTRUCTURE', 'STR') = _____。 【福州大学 1998 二、4 (2 分)】
5. 设正文串长度为 n, 模式串长度为 m, 则串匹配的 KMP 算法的时间复杂度为_____。

【重庆大学 2000 一、4】

6. 模式串 P = 'abaabacac' 的 next 函数值序列为_____。 【西安电子科技大学 2001 软件 一、6 (2 分)】
7. 字符串 'ababaaab' 的 nextval 函数值为_____。 【北京邮电大学 2001 二、4 (2 分)】
8. 设 T 和 P 是两个给定的串, 在 T 中寻找等于 P 的子串的过程称为 (1), 又称 P 为 (2)。

【西安电子科技大学 1998 二、5 (16/6 分)】

9. 串是一种特殊的线性表, 其特殊性表现在 (1); 串的两种最基本的存储方式是 (2)、(3); 两个串相等的充分必要条件是 (4)。 【中国矿业大学 2000 一、3 (4 分)】
10. 两个字符串相等的充分必要条件是_____。 【西安电子科技大学 1999 软件 一、1 (2 分)】

11. 知 U = 'xyxyxyxyxy'; t = 'xyx';

ASSIGN (S, U);

ASSIGN (V, SUBSTR (S, INDEX (s, t), LEN (t) + 1));

ASSIGN (m, 'ww')

求 REPLACE (S, V, m) = _____。 【东北大学 1997 一、1 (5 分)】

12. 实现字符串拷贝的函数 strcpy 为:

```
void strcpy(char *s, char *t) /*copy t to s*/
```

```
{ while (_____)
```

```
} 【浙江大学 1999 一、5 (3 分)】
```

13. 下列程序判断字符串 s 是否对称, 对称则返回 1, 否则返回 0; 如 f("abba") 返回 1, f("abab") 返回 0;

```
int f((1)_____)
```

```
{int i=0, j=0;
```

```
while (s[j]) (2)_____;
```

```
for(j--; i<j && s[i]==s[j]; i++, j--);
```

```
return((3)_____)
```

【浙江大学 1999 一、6 (3 分)】

14. 下列算法实现求采用顺序结构存储的串 s 和串 t 的一个最长公共子串。

程序 (a)

```
PROCEDURE maxcomstr(VAR s, t : orderstring; VAR index, length : integer);
```

```
VAR i, j, k, length1: integer; con: boolean;
```

```
BEGIN
```

```
index := 0; length := 0; i := 1;
```

```
WHILE(i<=s.len) DO
```

```
[j:=1;
```

```
WHILE (j<=t.len) DO
```

```

[ IF (s[i]=t[j]) THEN
  [ k:=1; length1:=1; con:=true;
    WHILE con DO
      IF (1) THEN [length1:=length1+1;k:=k+1;] ELSE (2);
      IF (length1>length) THEN [index:=i; length:=length1; ]
        (3);
    ]
  ELSE (4);
]
(5);
]
END;

```

程序(b)

```

void maxcomstr(orderstring *s,*t; int index, length)
{int i, j, k, length1, con;
  index=0; length=0; i=1;
  while (i<=s.len)
  {j=1;
    while(j<=t.len)
    { if (s[i]= t[j])
      { k=1; length1=1; con=1;
        while(con)
          if (1) { length1=length1+1;k=k+1; } else (2);
          if (length1>length) { index=i; length=length1; }
          (3);
        }
      else (4);
    }
    (5);
  } }

```

【上海大学 2000 一、2 (10 分)】

15. 完善算法：求 KMP 算法中 next 数组。

```

PROC get_next(t:string, VAR next:ARRAY[1..t.len] OF integer);
BEGIN
  j:=1; k:=(1); next[1]:=0;
  WHILE j<t.len DO
    IF k=0 OR t.ch[j]=t.ch[k] THEN BEGIN j:=j+1; k:=k+1; next[j]:=k;END
      ELSE k:=(2);
  END;

```

【中山大学 1998 四、1 (4 分)】

16. 下面函数 index 用于求 t 是否为 s 的子串，若是返回 t 第一次出现在 s 中的序号(从 1 开始计)，否则返回 0。

例如:s= 'abcdefcdek', t= 'cde' ,则 indse(s,t)=3, index(s, 'aaa')=0 。已知 t, s 的串长分别是 mt, ms

```

FUNC index(s, t, ms, mt);

```



```
i:=1;j:=1;
WHILE (i<ms) AND (j<mt) DO
  IF s[i]=t[j] THEN [ (1) ]; (2) ]
                ELSE [ (3) ]; (4) ]
  IF j>mt THEN return (5); ELSE return (6)
ENDF;
```

【南京理工大学 1999 三、2 (6分)】

第四章 串

一、选择题

1. B	2. E	3. C	4. A	5. C	6. A	7. 1D	7. 2F	8. B 注	9. D	10. B
------	------	------	------	------	------	-------	-------	--------	------	-------

注：子串的定义是：串中任意个连续的字符组成的子序列，并规定空串是任意串的子串，任意串是其自身的子串。若字符串长度为 n ($n>0$)，长为 n 的子串有 1 个，长为 $n-1$ 的子串有 2 个，长为 $n-2$ 的子串有 3 个，……，长为 1 的子串有 n 个。由于空串是任何串的子串，所以本题的答案为： $8*(8+1)/2+1=37$ 。故选 B。但某些教科书上认为“空串是任意串的子串”无意义，所以认为选 C。为避免考试中的二意性，编者认为第 9 题出得好。

二、判断题

1. √	2. √	3. √								
------	------	------	--	--	--	--	--	--	--	--

三、填空题

- (1) 由空格字符 (ASCII 值 32) 所组成的字符串 (2) 空格个数 2. 字符
- 任意个连续的字符组成的子序列 4. 5 5. $0(m+n)$
- 01122312 7. 01010421 8. (1) 模式匹配 (2) 模式串
- (1) 其数据元素都是字符 (2) 顺序存储 (3) 和链式存储 (4) 串的长度相等且两串中对应位置的字符也相等
- 两串的长度相等且两串中对应位置的字符也相等。
- 'xyxyxywwy' 12. $*s++=*t++$ 或 $(*s++=*t++) != '\0'$
- (1) **char** s[] (2) $j++$ (3) $i >= j$
- [题目分析] 本题算法采用顺序存储结构求串 s 和串 t 的最大公共子串。串 s 用 i 指针 ($1 \leq i \leq s.len$)。串 t 用 j 指针 ($1 \leq j \leq t.len$)。算法思想是对每个 i ($1 \leq i \leq s.len$ ，即程序中第一个 **WHILE** 循环)，来求从 i 开始的连续字符串与从 j ($1 \leq j \leq t.len$ ，即程序中第二个 **WHILE** 循环) 开始的连续字符串的最大匹配。程序中第三个 (即最内层) 的 **WHILE** 循环，是当 s 中某字符 ($s[i]$) 与 t 中某字符 ($t[j]$) 相等时，求出局部公共子串。若该子串长度大于已求出的最长公共子串 (初始为 0)，则最长公共子串的长度要修改。

程序 (a): (1) $(i+k \leq s.len) \text{ AND } (j+k \leq t.len) \text{ AND } (s[i+k]=t[j+k])$

//如果在 s 和 t 的长度内，对应字符相等，则指针 k 后移 (加 1)。

(2) $con:=false$ // s 和 t 对应字符不等时置标记退出

(3) $j:=j+k$ //在 t 串中，从第 $j+k$ 字符再与 $s[i]$ 比较

(4) $j:=j+1$ // t 串取下一字符

(5) $i:=i+1$ // s 串指针 i 后移 (加 1)。

程序 (b): (1) $i+k \leq s.len \text{ \&\& } j+k \leq t.len \text{ \&\& } s[i+k]=t[j+k]$ //所有注释同上 (a)

(2) $con=0$ (3) $j+=k$ (4) $j++$ (5) $i++$

15. (1) 0 (2) $next[k]$

16. (1) $i:=i+1$ (2) $j:=j+1$ (3) $i:=i-j+2$ (4) $j:=1$; (5) $i-mt$ (或 $i:=i-j+1$) (6) 0

17. 程序中递归调用

- (1) $ch1 < midch$ //当读入不是分隔符&和输入结束符\$时, 继续读入字符
- (2) $ch1 = ch2$ //读入分隔符&后, 判 $ch1$ 是否等于 $ch2$, 得出真假结论。
- (3) $answer := true$
- (4) $answer := false$
- (5) $read(ch)$
- (6) $ch = endch$

18. (1) $initstack(s)$ // 栈 s 初始化为空栈。

- (2) $setnull(exp)$ //串 exp 初始化为空串。
- (3) $ch \in opset$ //判取出字符是否是操作符。
- (4) $push(s, ch)$ //如 ch 是运算符, 则入运算符栈 s 。
- (5) $empty(s)$ //判栈 s 是否为空。
- (6) $succ := false$ //若读出 ch 是操作数且栈为空, 则按出错处理。
- (7) exp (8) ch //若 ch 是操作数且栈非空, 则形成部分中缀表达式。
- (9) exp (10) $gettop(s)$ //取栈顶操作符。
- (11) $pop(s)$ //操作符取出后, 退栈。
- (12) $sempy(s)$ //将 pre 的最后一个字符 (操作数) 加入到中缀式 exp 的最后。

四. 应用题

1. 串是零个至多个字符组成的有限序列。从数据结构角度讲, 串属于线性结构。与线性表的特殊性在于串的元素是字符。

2. 空格是一个字符, 其 ASCII 码值是 32。空格串是由空格组成的串, 其长度等于空格的个数。空串是不含任何字符的串, 即空串的长度是零。

3. 最优的 $T(m, n)$ 是 $O(n)$ 。串 S_2 是串 S_1 的子串, 且在 S_1 中的位置是 1。开始求出最大公共子串的长度恰是串 S_2 的长度, 一般情况下, $T(m, n) = O(m \cdot n)$ 。

4. 朴素的模式匹配 (Brute-Force) 时间复杂度是 $O(m \cdot n)$, KMP 算法有一定改进, 时间复杂度达到 $O(m + n)$ 。本题也可采用从后面匹配的方法, 即从右向左扫描, 比较 6 次成功。另一种匹配方式是从左往右扫描, 但是先比较模式串的最后一个字符, 若不等, 则模式串后移; 若相等, 再比较模式串的第一个字符, 若第一个字符也相等, 则从模式串的第二个字符开始, 向右比较, 直至相等或失败。若失败, 模式串后移, 再重复以上过程。按这种方法, 本题比较 18 次成功。

5. KMP 算法主要优点是主串指针不回溯。当主串很大不能一次读入内存且经常发生部分匹配时, KMP 算法的优点更为突出。

6. 模式串的 next 函数定义如下:

$$next[j] = \begin{cases} 0 & \text{当 } j = 1 \text{ 时} \\ \max\{k \mid 1 < k < j \text{ 且 } 'p_1 \dots p_{k-1}' = 'p_{j-k+1} \dots p_{j-1}'\} & \text{当此集合不空时} \\ 1 & \text{其它情况} \end{cases}$$

根据此定义, 可求解模式串 t 的 next 和 nextval 值如下:

j	1	2	3	4	5	6	7	8	9	10	11	12
t 串	a	b	c	a	a	b	b	a	b	c	a	b
next[j]	0	1	1	1	2	2	3	1	2	3	4	5

nextval[j]	0	1	1	0	2	1	3	0	1	1	0	5
------------	---	---	---	---	---	---	---	---	---	---	---	---

7. 解法同上题 6, 其 next 和 nextval 值分别为 0112123422 和 0102010422。
8. 解法同题 6, t 串的 next 和 nextval 函数值分别为 0111232 和 0110132。
9. 解法同题 6, 其 next 和 nextval 值分别为 011123121231 和 011013020131。
10. p1 的 next 和 nextval 值分别为:0112234 和 0102102;p2 的 next 和 nextval 值分别为:0121123 和 0021002。
11. next 数组值为 011234567 改进后的 next 数组信息值为 010101017。
12. 011122312。
13. next 定义见题上面 6 和下面题 20。串 p 的 next 函数值为: 01212345634。
14. (1) S 的 next 与 nextval 值分别为 012123456789 和 002002002009, p 的 next 与 nextval 值分别为 012123 和 002003。

(2) 利用 BF 算法的匹配过程:

第一趟匹配: aabaabaabaac
aabaac (i=6, j=6)

第二趟匹配: aabaabaabaac
aa (i=3, j=2)

第三趟匹配: aabaabaabaac
a (i=3, j=1)

第四趟匹配: aabaabaabaac
aabaac (i=9, j=6)

第五趟匹配: aabaabaabaac
aa (i=6, j=2)

第六趟匹配: aabaabaabaac
a (i=6, j=1)

第七趟匹配: aabaabaabaac
(成功) aabaac (i=13, j=7)

利用 KMP 算法的匹配过程:

第一趟匹配: aabaabaabaac
aabaac (i=6, j=6)

第二趟匹配: aabaabaabaac
(aa)baac

第三趟匹配: aabaabaabaac
(成功) (aa)baac

15. (1) p 的 nextval 函数值为 0110132。(p 的 next 函数值为 0111232)。

(2) 利用 KMP(改进的 nextval)算法, 每趟匹配过程如下:

第一趟匹配: abcaabbabcabaacbacba
abcab (i=5, j=5)

第二趟匹配: abcaabbabcabaacbacba
abc (i=7, j=3)

第三趟匹配: abcaabbabcabaacbacba
a (i=7, j=1)

第四趟匹配: abcaabbabcabaac bacba
(成功) abcabaa (i=15, j=8)

16. KMP 算法的时间复杂性是 $O(m+n)$ 。

p 的 next 和 nextval 值分别为 01112212321 和 01102201320。

17. (1) p 的 nextval 函数值为 01010。(next 函数值为 01123)

(2) 利用所得 nextval 数值, 手工模拟对 s 的匹配过程, 与上面 16 题类似, 为节省篇幅, 故略去。

18. 模式串 T 的 next 和 nextval 值分别为 0121123 和 0021002。

19. 第 4 行的 $p[J]=p[K]$ 语句是测试模式串的第 J 个字符是否等于第 K 个字符, 如是, 则指针 J 和 K 均增加 1, 继续比较。第 6 行的 $p[J]=p[K]$ 语句的意义是, 当第 J 个字符在模式匹配中失配

时, 若第 K 个字符和第 J 个字符不等, 则下个与主串匹配的字符是第 K 个字符; 否则, 若第 K 个字符和第 J 个字符相等, 则下个与主串匹配的字符是第 K 个字符失配时的下一个 (即 $\text{NEXTVAL}[K]$)。

该算法在最坏情况下的时间复杂度 $O(m^2)$ 。

20. (1) 当模式串中第一个字符与主串中某字符比较不等 (失配) 时, $\text{next}[1]=0$ 表示模式串中已没有字符可与主串中当前字符 $s[i]$ 比较, 主串当前指针应后移至下一字符, 再和模式串中第一个字符进行比较。

(2) 当主串第 i 个字符与模式串中第 j 个字符失配时, 若主串 i 不回溯, 则假定模式串第 k 个字符与主串第 i 个字符比较, k 值应满足条件 $1 < k < j$ 并且 ' $p_1 \cdots p_{k-1}$ ' = ' $p_{j-k+1} \cdots p_{j-1}$ ', 即 k 为模式串向后移动的距离, k 值有多个, 为了不使向右移动丢失可能的匹配, k 要取大, 由于 $\max\{k\}$ 表示移动的最大距离, 所以取 $\max\{k\}$, k 的最大值为 $j-1$ 。

(3) 在上面两种情况外, 发生失配时, 主串指针 i 不回溯, 在最坏情况下, 模式串从第 1 个字符开始与主串第 i 个字符比较, 以便不致丢失可能的匹配。

21. 这里失败函数 f , 即是通常讲的模式串的 next 函数, 其定义见本章应用题的第 6 题。

进行模式匹配时, 若主串第 i 个字符与模式串第 j 个字符发生失配, 主串指针 i 不回溯, 和主串第 i 个字符进行比较的是模式串的第 $\text{next}[j]$ 个字符。模式串的 next 函数值, 只依赖于模式串, 和主串无关, 可以预先求出。

该算法的技术特点是主串指针 i 不回溯。在经常发生“部分匹配”和主串很大不能一次调入内存时, 优点特别突出。

22. 失败函数 (即 next) 的值只取决于模式串自身, 若第 j 个字符与主串第 i 个字符失配时, 假定主串不回溯, 模式串用第 k (即 $\text{next}[j]$) 个字符与第 i 个相比, 有 ' $p_1 \cdots p_{k-1}$ ' = ' $p_{j-k+1} \cdots p_{j-1}$ ', 为了不因模式串右移与主串第 i 个字符比较而丢失可能的匹配, 对于上式中存在的多个 k 值, 应取其中最大的一个。这样, 因 $j-k$ 最小, 即模式串向右滑动的位数最小, 避免因右移造成的可能匹配的丢失。

23. 仅从两串含有相等的字符, 不能判定两串是否相等, 两串相等的充分必要条件是两串长度相等且对应位置上的字符相同 (即两串串值相等)。

24. (1) s_1 和 s_2 均为空串; (2) 两串之一为空串; (3) 两串串值相等 (即两串长度相等且对应位置上的字符相同); (4) 两串中一个串长是另一个串长 (包括串长为 1 仅有一个字符的情况) 的数倍, 而且长串就好像是由数个短串经过连接操作得到的。

25. 题中所给操作的含义如下:

//: 连接函数, 将两个串连接成一个串

$\text{substr}(s, i, j)$: 取子串函数, 从串 s 的第 i 个字符开始, 取连续 j 个字符形成子串

$\text{replace}(s_1, i, j, s_2)$: 置换函数, 用 s_2 串替换 s_1 串中从第 i 个字符开始的连续 j 个字符
本题有多种解法, 下面是其中的一种:

- (1) $s_1 = \text{substr}(s, 3, 1)$ //取出字符: 'y'
- (2) $s_2 = \text{substr}(s, 6, 1)$ //取出字符: '+'
- (3) $s_3 = \text{substr}(s, 1, 5)$ //取出子串: '(xyz)'
- (4) $s_4 = \text{substr}(s, 7, 1)$ //取出字符: '*'
- (5) $s_5 = \text{replace}(s_3, 3, 1, s_2)$ //形成部分串: '(x+z)'
- (6) $s = s_5 // s_4 // s_1$ //形成串 t 即 '(x+z)*y'

五、算法设计

1、[题目分析]判断字符串 t 是否是字符串 s 的子串, 称为串的模式匹配, 其基本思想是对串 s 和 t 各设一个指针 i 和 j , i 的值域是 $0..m-n$, j 的值域是 $0..n-1$ 。初始值 i 和 j 均为 0。

模式匹配从 s_0 和 t_0 开始, 若 $s_0=t_0$, 则 i 和 j 指针增加 1, 若在某个位置 $s_i \neq t_j$, 则主串指针 i 回溯到 $i=i-j+1$, j 仍从 0 开始, 进行下一轮的比较, 直到匹配成功 ($j>n-1$), 返回子串在主串的位置 ($i-j$)。否则, 当 $i>m-n$ 则为匹配失败。

```
int index(char s[], t[], int m, n)
//字符串 s 和 t 用一维数组存储, 其长度分别为 m 和 n。本算法求字符串 t 在字符串 s 中的
//第一次出现, 如是, 输出子串在 s 中的位置, 否则输出 0。
{int i=0, j=0;
 while (i<=m-n && j<=n-1)
   if (s[i]==t[j]) {i++;j++;} //对应字符相等, 指针后移。
   else {i=i-j+1;j=0;} //对应字符不相等, i 回溯, j 仍为 0。
   if (i<=m-n && j==n) {printf("t 在 s 串中位置是%d", i-n+1);return(i-n+1);} //匹配成功
   else return(0); //匹配失败
} //算法 index 结束
main () //主函数
{char s[], t[]; int m, n, i;
 scanf("%d%d", &m, &n); //输入两字符串的长度
 scanf("%s", s); //输入主串
 scanf("%s", t); //输入子串
 i=index(s, t, m, n);
} //程序结束
```

[程序讨论] 因用 C 语言实现, 一维数组的下标从 0 开始, $m-1$ 是主串最后一个字符的下标, $n-1$ 是 t 串最后一个字符的下标。若匹配成功, 最佳情况是 s 串的第 0 到第 $n-1$ 个字符与 t 匹配, 时间复杂度为 $O(n)$; 匹配成功的最差情况是, 每次均在 t 的最后一个字符才失败, 直到 s 串的第 $m-n$ 个字符成功, 其时间复杂度为 $O((m-n)*n)$, 即 $O(m*n)$ 。失败的情况是 s 串的第 $m-n$ 个字符比 t 串某字符比较失败, 时间复杂度为 $O(m*n)$ 。之所以串 s 的指针 i 最大到 $m-n$, 是因为在 $m-n$ 之后, 所剩子串长度已经小于子串长度 n , 故不必再去比较。算法中未讨论输入错误 (如 s 串长小于 t 串长)。

另外, 根据子串的定义, 返回值 $i-n+1$ 是子串在主串中的位置, 子串在主串中的下标是 $i-n$ 。

2. [问题分析] 在一个字符串内, 统计含多少整数的问题, 核心是如何将数从字符串中分离出来。从左到右扫描字符串, 初次碰到数字字符时, 作为一个整数的开始。然后进行拼数, 即将连续出现的数字字符拼成一个整数, 直到碰到非数字字符为止, 一个整数拼完, 存入数组, 再准备下一整数, 如此下去, 直至整个字符串扫描到结束。

```
int CountInt ()
// 从键盘输入字符串, 连续的数字字符算作一个整数, 统计其中整数的个数。
{int i=0, a[]; // 整数存储到数组 a, i 记整数个数
 scanf("%c", &ch); // 从左到右读入字符串
 while (ch!='#') // '#' 是字符串结束标记
   if (isdigit(ch)) // 是数字字符
   {num=0; // 数初始化
    while (isdigit(ch) && ch!='#') // 拼数
    {num=num*10+'ch'-'0';
     scanf("%c", &ch);
    }
   }
```

```
a[i]=num; i++;
if (ch!= '#') scanf ("%c", &ch); // 若拼数中输入了 '#', 则不再输入
} // 结束 while (ch!= '#')
printf ("共有%d 个整数, 它们是:" i);
for (j=0; j<i; j++)
{printf ("%6d", a[j]);
if ((j+1) %10==0) printf ("\n"); } // 每 10 个数输出在一行上
} // 算法结束
```

[算法讨论] 假定字符串中的数均不超过 32767, 否则, 需用长整型数组及变量。

3、[题目分析] 设以字符数组 s 表示串, 重复子串的含义是由一个或多个连续相等的字符组成的子串, 其长度用 max 表示, 初始长度为 0, 将每个局部重复子串的长度与 max 相比, 若比 max 大, 则需要更新 max, 并用 index 记住其开始位置。

```
int LongestString(char s[], int n)
//串用一维数组 s 存储, 长度为 n, 本算法求最长重复子串, 返回其长度。
{int index=0, max=0; //index 记最长的串在 s 串中的开始位置, max 记其长度
int length=1, i=0, start=0; //length 记局部重复子串长度, i 为字符数组下标
while(i<n-1)
if(s[i]==s[i+1]) {i++; length++;}
else //上一个重复子串结束
{if(max<length) {max=length; index=start;} //当前重复子串长度大, 则更新
max
i++; start=i; length=1; //初始化下一重复子串的起始位置和
长度
}
printf("最长重复子串的长度为%d, 在串中的位置%d\n", max, index);
return(max);
} //算法结束
```

[算法讨论] 算法中用 $i < n-1$ 来控制循环次数, 因 C 数组下标从 0 开始, 故长度为 n 的串, 其最后一个字符下标是 $n-1$, 当 i 最大为 $n-2$ 时, 条件语句中 $s[i+1]$ 正好是 $s[n-1]$, 即最后一个字符。子串长度的初值数为 1, 表示一个字符自然等于其身。

算法的时间复杂度为 $O(n)$, 每个字符与其后继比较一次。

4、[题目分析] 教材中介绍的串置换有两种形式: 第一种形式是 $\text{replace}(s, i, j, t)$, 含义是将 s 串中从第 i 个字符开始的 j 个字符用 t 串替换, 第二种形式是 $\text{replace}(s, t, v)$, 含义是将 s 串中所有非重叠的 t 串用 v 代替。我们先讨论第一种形式的替换。因为已经给定顺序存储结构, 我们可将 s 串从第 $(i+j-1)$ 到串尾 (即 $s.\text{curlen}$) 移动 $t.\text{curlen}-j$ 绝对值个位置 (以便将 t 串插入): 若 $j > t.\text{curlen}$, 则向左移; 若 $j < t.\text{curlen}$, 则向右移动; 若 $j = t.\text{curlen}$, 则不必移动。最后将 t 串复制到 s 串的合适位置上。当然, 应考虑置换后的溢出问题。

```
int replace(strtp s, t, int i, j)
//s 和 t 是用一维数组存储的串, 本算法将 s 串从第 i 个字符开始的连续 j 个字符用 t 串置
换, 操作成功返回 1, 否则返回 0 表示失败。
{if(i<1 || j<0 || t.curlen+s.curlen-j>maxlen)
{printf("参数错误\n"); exit(0);} //检查参数及置换后的长度的合法性。
if(j<t.curlen) //若 s 串被替换的子串长度小于 t 串长度, 则 s 串部分右移,
for(k=s.curlen-1; k>=i+j-1; k--) s.ch[k+t.curlen-j]=s.ch[k];
```



```

else if (j>t.curlen)    //s 串中被替换子串的长度小于 t 串的长度。
    for(k=i-1+j;k<=s.curlen-1;k++) s.ch[k-(j-t.curlen)]=s.ch[k];
    for(k=0;k<t.curlen;k++) s.ch[i-1+k]=t.ch[k]; //将 t 串复制到 s 串的适当位置
if(j>t.curlen)          s.curlen=s.curlen-(j-t.curlen);else
s.curlen=s.curlen+(t.curlen-j);
} //算法结束

```

[算法讨论]若允许使用另一数组,在检查合法性后,可将 s 的第 i 个(不包括 i)之前的子串复制到另一子串如 s1 中,再将 t 串接到 s1 串后面,然后将 s 的第 i+j 直到尾的部分加到 s1 之后。最后将 s1 串复制到 s。主要语句有:

```

for(k=0;k<i;k++) s1.ch[k]=s.ch[k]; //将 s1 第 i 个字符前的子串复制到 s1, 这时 k=i-1
for(k=0;k<t.curlen;k++) s1.ch[i+k]=t.ch[k] //将 t 串接到 s1 的尾部
l=s.curlen+t.curlen-j-1;
for(k=s.curlen-1;k>i-1+j;k--) //将子串第 i+j-1 个字符以后的子串复制到 s1
    s1.ch[l--]=s.ch[k]
for(k=0;k<s.curlen+t.curlen-j;k++) s.ch[k]=s1.ch[k]; //将结果串放入 s。

```

下面讨论 replace(s, t, v) 的算法。该操作的意义是用串 v 替换所有在串 s 中出现的和非空串 t 相等的非重叠的子串。本算法不指定存储结构,只使用串的基本运算。

```

void replace(string s, t, v)
//本算法是串的置换操作,将串 s 中所有非空串 t 相等且不重复的子串用 v 代替。
{i=index(s, t); //判断 s 是否有和 t 相等的子串
if(i!=0) //串 s 中包含和 t 相等的子串
{creat(temp, " "); //creat 操作是将串常量(此处为空串)赋值给 temp。
m=length(t); n=length(s); //求串 t 和 s 的长度
while(i!=0)
{assign(temp, concat(temp, substr(s, 1, i-1), v)); //用串 v 替换 t 形成部分结果
assign(s, substr(s, i+m, n-i-m+1)); //将串 s 中串后的部分形成新的
s 串
n=n-(i-1)-m; //求串 s 的长度
i=index(s, t); //在新 s 串中再找串 t 的位置
}
assign(s, contact(temp, s)); //将串 temp 和剩余的串 s 连接后再赋值给 s
} //if 结束
} //算法结束

```

5、[题目分析]本题是字符串的插入问题,要求在字符串 s 的 pos 位置,插入字符串 t。首先应查找字符串 s 的 pos 位置,将第 pos 个字符到字符串 s 尾的子串向后移动字符串 t 的长度,然后将字符串 t 复制到字符串 s 的第 pos 位置后。

对插入位置 pos 要验证其合法性,小于 1 或大于串 s 的长度均为非法,因题目假设给字符串 s 的空间足够大,故对插入不必判溢出。

```

void insert(char *s, char *t, int pos)
//将字符串 t 插入字符串 s 的第 pos 个位置。
{int i=1, x=0; char *p=s, *q=t; //p, q 分别为字符串 s 和 t 的工作指针
if(pos<1) {printf("pos 参数位置非法\n"); exit(0);}
while(*p!='\0' && i<pos) {p++; i++;} //查 pos 位置
//若 pos 小于串 s 长度,则查到 pos 位置时, i=pos。

```

```

if(*p == '/0') {printf("%d 位置大于字符串 s 的长度", pos); exit(0);}
else //查找字符串的尾
    while(*p != '/0') {p++; i++;} //查到尾时, i 为字符 '\0' 的下标, p 也指向 '\0'。
while(*q != '\0') {q++; x++;} //查找字符串 t 的长度 x, 循环结束时 q 指向 '\0'。
for(j=i; j>=pos; j--) {*(p+x)=*p; p--;} //串 s 的 pos 后的子串右移, 空出串 t 的位置。
q--; //指针 q 回退到串 t 的最后一个字符
for(j=1; j<=x; j++) *p--=*q--; //将 t 串插入到 s 的 pos 位置上
[算法讨论] 串 s 的结束标记('\0')也后移了, 而串 t 的结尾标记不应插入到 s 中。

```

6. [题目分析] 本题属于查找, 待查找元素是字符串 (长 4), 将查找元素存放在一维数组中。二分检索 (即折半查找或对分查找), 是首先用一维数组的“中间”元素与被检索元素比较, 若相等, 则检索成功, 否则, 根据被检索元素大于或小于中间元素, 而在中间元素的右方或左方继续查找, 直到检索成功或失败 (被检索区间的低端指针大于高端指针)。下面给出类 C 语言的解法

```

typedef struct node
{char data[4]; //字符串长 4
}node;
非递归过程如下:
int binsearch(node string [], int n; char name[4])
//在有 n 个字符串的数组 string 中, 二分检索字符串 name。若检索成功, 返回 name 在 string
中的下标, 否则返回 -1。
{int low = 0, high = n - 1; //low 和 high 分别是检索区间的下界和上界
while(low <= high)
{mid = (low + high) / 2; //取中间位置
if(strcmp(string[mid], name) == 0) return (mid); //检索成功
else if(strcmp(string[mid], name) < 0) low = mid + 1; //到右半部分检索
else high = mid - 1; //到左半部分检索
}
return 0; //检索失败
} //算法结束
最大检索长度为 log2n。

```

7. [题目分析] 设字符串存于字符数组 X 中, 若转换后的数是负数, 字符串的第一个字符必为 '-', 取出的数字字符, 通过减去字符零 ('0') 的 ASCII 值, 变成数, 先前取出的数乘上 10 加上本次转换的数形成部分数, 直到字符串结束, 得到结果。

```

long atoi(char X[])
//一数字字符串存于字符数组 X 中, 本算法将其转换成数
{long num=0;
int i=1; //i 为数组下标
while (X[i] != '\0') num=10*num+(X[i++]-'0'); //当字符串未到尾, 进行数的转换
if(X[0]=='-') return (-num); //返回负数
else return ((X[0]-'0')*10+num); //返回正数, 第一位若不是负号, 则是数字
} //算法 atoi 结束

```

[算法讨论] 如是负数, 其符号位必在前面, 即字符数组的 x[0], 所以在作转换成数时下标 i 从 1 开始, 数字字符转换成数使用 X[i]-'0', 即字符与 '0' 的 ASCII 值相减。请注意对返回正整数的处理。

8. [题目分析] 本题要求字符串 s1 拆分成字符串 s2 和字符串 s3, 要求字符串 s2 “按给定长度 n 格式化两端对齐的字符串”, 即长度为 n 且首尾字符不得为空格字符。算法从左到右扫描字符串 s1, 找到第一个非空格字符, 计数到 n, 第 n 个拷入字符串 s2 的字符不得为空格, 然后将余下字符复制到字符串 s3 中。

```
void format (char *s1,*s2,*s3)
//将字符串 s1 拆分成字符串 s2 和字符串 s3, 要求字符串 s2 是长 n 且两端对齐
{char *p=s1, *q=s2;
int i=0;
while(*p!= '\0' && *p== ' ') p++; //滤掉 s1 左端空格
if(*p== '\0') {printf("字符串 s1 为空串或空格串\n"); exit(0); }
while( *p!= '\0' && i<n) { *q=*p; q++; p++; i++; } //字符串 s1 向字符串 s2 中复制
if(*p == '\0') { printf("字符串 s1 没有%d 个有效字符\n", n); exit(0); }
if(*(--q)==' ') //若最后一个字符为空格, 则需向后找到第一个非空格字符
{p-- ; //p 指针也后退
while(*p==' ' && *p!= '\0') p++; //往后查找一个非空格字符作串 s2 的尾字符
if(*p== '\0') {printf("s1 串没有%d 个两端对齐的字符串\n", n); exit(0); }
*q=*p; //字符串 s2 最后一个非空字符
*(++q)= '\0'; //置 s2 字符串结束标记
}
*q=s3;p++; //将 s1 串其余部分送字符串 s3.
while (*p!= '\0') { *q=*p; q++; p++; }
*q= '\0'; //置串 s3 结束标记
}
```

9. [题目分析] 两个串的相等, 其定义为两个串的值相等, 即串长相等, 且对应字符相等是两个串相等的充分必要条件。因此, 首先比较串长, 在串长相等的前提下, 再比较对应字符是否相等。

```
int equal(strtp s, strtp t)
//本算法判断字符串 s 和字符串 t 是否相等, 如相等返回 1, 否则返回 0
{if (s.curlen!=t.curlen) return (0);
for (i=0; i<s.curlen;i++) //在类 C 中, 一维数组下标从零开始
if (s.ch[i]!= t.ch[i]) return (0);
return (1); //两串相等
} //算法结束
```

10. [问题分析] 由于字母共 26 个, 加上数字符号 10 个共 36 个, 所以设一长 36 的整型数组, 前 10 个分量存放数字字符出现的次数, 余下存放字母出现的次数。从字符串中读出数字字符时, 字符的 ASCII 代码值减去数字字符 ‘0’ 的 ASCII 代码值, 得出其数值(0..9), 字母的 ASCII 代码值减去字符 ‘A’ 的 ASCII 代码值加上 10, 存入其数组的对应下标分量中。遇其它符号不作处理, 直至输入字符串结束。

```
void Count ()
//统计输入字符串中数字字符和字母字符的个数。
{int i, num[36];
char ch;
for (i=0; i<36; i++) num[i]= 0; // 初始化
while ((ch=getchar ()) != '#') // ‘#’ 表示输入字符串结束。
```

```

        if ('0' <= ch <= '9') {i = ch - 48; num[i]++;}           // 数字字符
        else if ('A' <= ch <= 'Z') {i = ch - 65 + 10; num[i]++;} // 字母字符

    for (i = 0; i < 10; i++) // 输出数字字符的个数
        printf("数字%d的个数=%d\n", i, num[i]);
    for (i = 10; i < 36; i++) // 求出字母字符的个数
        printf("字母字符%c的个数=%d\n", i + 55, num[i]);
} // 算法结束。

```

11. [题目分析]实现字符串的逆置并不难，但本题“要求不另设串存储空间”来实现字符串逆序存储，即第一个输入的字符最后存储，最后输入的字符先存储，使用递归可容易做到。

```

void InvertStore(char A[])
//字符串逆序存储的递归算法。
{
    char ch;
    static int i = 0; //需要使用静态变量
    scanf("%c", &ch);
    if (ch != '.') //规定'.'是字符串输入结束标志
    {
        InvertStore(A);
        A[i++] = ch; //字符串逆序存储
    }
    A[i] = '\0'; //字符串结尾标记
} //结束算法 InvertStore。

```

12. 串 s'' 可以看作由以下两部分组成：'caabcbca...a' 和 'ca...a'，设这两部分分别叫串 s1 和串 s2，要设法从 s, s' 和 s'' 中得到这两部分，然后使用联接操作联接 s1 和 s2 得到 s''。

```

i = index(s, s'); //利用串 s' 求串 s1 在串 s 中的起始位置
s1 = substr(s, i, length(s) - i + 1); //取出串 s1
j = index(s, s''); //求串 s'' 在串 s 中的起始位置，s 串中' bcb' 后是' ca...a'
s2 = substr(s, j + 3, length(s) - j - 2); //形成串 s2
s3 = concat(s1, s2);

```

13. [题目分析]对读入的字符串的第奇数个字符，直接放在数组前面，对第偶数个字符，先入栈，到读字符串结束，再将栈中字符出栈，送入数组中。限于篇幅，这里编写算法，未编程序。

```

void RearrangeString()
//对字符串改造，将第偶数个字符放在串的后半部分，第奇数个字符前半部分。
{
    char ch, s[], stk[]; //s 和 stk 是字符数组（表示字符串）和字符栈
    int i = 1, j; //i 和 j 字符串和字符栈指针
    while((ch = getchar()) != '#' ) // '#' 是字符串结束标志
    {
        s[i++] = ch; //读入字符串
        s[i] = '\0'; //字符数组中字符串结束标志
        i = 1; j = 1;
        while(s[i]) //改造字符串
        {
            if(i % 2 == 0) stk[j++] = s[i]; else s[j++] = s[i];
            i++; } //while
        i--; i = i / 2; //i 先从 '\0' 后退，是第偶数字符的个数
        while(i > 0) s[j++] = stk[i--] //将第偶数个字符逆序填入原字符数组
    }
}

```

14. [题目分析]本题是对字符串表达式的处理问题，首先定义 4 种数据结构：符号的类码，符号的 TOKEN 表示，变量名表 NAMEL 和常量表 CONSL。这四种数据结构均定义成结构体形式，数据部分用一维数组存储，同时用指针指出数据的个数。算法思想是从左到右扫描表达式，对读出的字符，先查出其符号类码：若是变量或常量，就到变量名表和常量表中去查是否已有，若无，则在相应表中增加之，并返回该字符在变量名表或常量表中的下标；若是操作符，则去查其符号类码。对读出的每个符号，均填写其 TOKEN 表。如此下去，直到表达式处理完毕。先定义各数据结构如下。

```

struct // 定义符号类别数据结构
{
    char data[7]; //符号
    char code[7]; //符号类码
} TYPL;

typedef struct //定义 TOKEN 的元素
{
    int typ; //符号码
    int addr; //变量、常量在名字表中的地址
} cmp;

struct {cmp data[50]; //定义 TOKEN 表长度<50
        int last; //表达式元素个数
    } TOKEN;

struct {char data[15]; //设变量个数小于 15 个
        int last; //名字表变量个数
    } NAMEL;

struct {char data[15]; //设常量个数小于 15 个
        int last; //常量个数
    } CONSL;

int operator (char cr)
//查符号在类码表中的序号
{
    for (i=3; i<=6; i++)
        if (TYPL.data[i]==cr) return (i);
}

void PROCeString ()
//从键盘读入字符串表达式 (以 '#' 结束)，输出其 TOKEN 表示。
{NAMEL.last=CONSL.last=TOKEN.last=0; //各表元素个数初始化为 0
  TYPL.data[3]='*'; TYPL.data[4]='+'; TYPL.data[5]='(';
  TYPL.data[6]=')'; //将操作符存入数组
  TYPL.code[3]='3'; TYPL.code[4]='4'; TYPL.code[5]='5';
  TYPL.code[6]='6'; //将符号的类码存入数组
  scanf ("%c", &ch); //从左到右扫描 (读入) 表达式。
  while (ch != '#') // '#' 是表达式结束符
  {switch (ch) of
    {case 'A' : case 'B' : case 'C' : //ch 是变量
      TY=0; //变量类码为 0
      for (i=1; i<=NAMEL.last; i++)
        if (NAMEL.data[i]==ch) break; //已有该变量, i 记住其位置
      if (i>NAMEL.last) {NAMEL.data[i]=ch; NAMEL.last++; } //变量加入
    }
  }
}

```

```

        case '0' : case '1' : case '2' : case '3' : case '4' : case '5' : //处理常
量
        case '6' : case '7' : case '8' : case '9' : TY=1; //常量类码为 1
            for (i=1; i<=CONSL.last; i++)
                if (CONSL.data[i]==ch) break; ////已有该常量, i 记住其位置
            if (i>CONSL.last) {CONSL.data[i]=ch; CONSL.last++;} //将新常量
加入
        default: //处理运算符
            TY=operator (ch); //类码序号
            i=' \0'; //填入 TOKEN 的 addr 域 (期望输出空白)
        } //结束 switch, 下面将 ch 填入 TOKEN 表
        TOKEN.data [++TOKEN.last] .typ=TY; TOKEN.data [TOKEN.last] .addr=i;
        scanf ("%c", &ch); //读入表达式的下一符号。
    } //while
} //算法结束

```

[程序讨论]为便于讨论, 各一维数组下标均以 1 开始, 在字符为变量或常量的情况下, 将其类码用 TY 记下, 用 i 记下其 NAMEL 表或 CONSL 表中的位置, 以便在填 TOKEN 表时用。在运算符 ('+', '*', '(', ')') 填入 TOKEN 表时, TOKEN 表的 addr 域没意义, 为了程序统一, 这里填入了 ' \0'。本题是表达式处理的简化情况 (只有 3 个单字母变量, 常量只有 0..9, 操作符只 4 个), 若是真实情况, 所用数据结构要相应变化。

第 5 章 数组和广义表

一、选择题

1. 设有一个 10 阶的对称矩阵 A, 采用压缩存储方式, 以行序为主存储, a_{11} 为第一元素, 其存储地址为 1, 每个元素占一个地址空间, 则 a_{85} 的地址为 ()。【燕山大学 2001 一、2 (2 分)】

- A. 13 B. 33 C. 18 D. 40

2. 有一个二维数组 A[1:6, 0:7] 每个数组元素用相邻的 6 个字节存储, 存储器按字节编址, 那么这个数组的体积是 (①) 个字节。假设存储数组元素 A[1, 0] 的第一个字节的地址是 0, 则存储数组 A 的最后一个元素的第一个字节的地址是 (②)。若按行存储, 则 A[2, 4] 的第一个字节的地址是 (③)。若按列存储, 则 A[5, 7] 的第一个字节的地址是 (④)。就一般情况而言, 当 (⑤) 时, 按行存储的 A[I, J] 地址与按列存储的 A[J, I] 地址相等。供选择的答案:【上海海运学院 1998 二、2 (5 分)】

- ①-④: A. 12 B. 66 C. 72 D. 96 E. 114 F. 120
G. 156 H. 234 I. 276 J. 282 K. 283 L. 288

- ⑤: A. 行与列的上界相同 B. 行与列的下界相同
C. 行与列的上、下界都相同 D. 行的元素个数与列的元素个数相同

3. 设有数组 A[i, j], 数组的每个元素长度为 3 字节, i 的值为 1 到 8, j 的值为 1 到 10, 数组从内存首地址 BA 开始顺序存放, 当用以列为主存放时, 元素 A[5, 8] 的存储首地址为 ()。

- A. BA+141 B. BA+180 C. BA+222 D. BA+225

【南京理工大学 1997 一、8 (2 分)】

4. 假设以行序为主序存储二维数组 A=array[1..100, 1..100], 设每个数据元素占 2 个存储单元, 基地址为 10, 则 LOC[5, 5]= ()。【福州大学 1998 一、10 (2 分)】

- A. 808 B. 818 C. 1010 D. 1020

5. 数组 A[0..5, 0..6] 的每个元素占五个字节, 将其按列优先次序存储在起始地址为 1000 的内存单元中, 则元素 A[5, 5] 的地址是 ()。【南京理工大学 2001 一、13 (1.5 分)】

- A. 1175 B. 1180 C. 1205 D. 1210

6. 有一个二维数组 A[0:8, 1:5], 每个数组元素用相邻的 4 个字节存储, 存储器按字节编址, 假设存储数组元素 A[0, 1] 的第一个字节的地址是 0, 存储数组 A 的最后一个元素的第一个字节的地址是 (①)。若按行存储, 则 A[3, 5] 和 A[5, 3] 的第一个字节的地址是 (②) 和 (③)。若按列存储, 则 A[7, 1] 和 A[2, 4] 的第一个字节的地址是 (④) 和 (⑤)。【上海海运学院 1996 二、1 (5 分)】

- ①-⑤: A. 28 B. 44 C. 76 D. 92 E. 108 F. 116 G. 132 H. 176 I. 184
J. 188

7. 将一个 A[1..100, 1..100] 的三对角矩阵, 按行优先存入一维数组 B[1..298] 中, A 中元素 A_{6665} (即该元素下标 $i=66, j=65$), 在 B 数组中的位置 K 为 ()。供选择的答案:

- A. 198 B. 195 C. 197 【北京邮电大学 1998 二、5 (2 分)】

8. 二维数组 A 的元素都是 6 个字符组成的串, 行下标 i 的范围从 0 到 8, 列下标 j 的范围从 1 到 10。从供选择的答案中选出应填入下列关于数组存储叙述中 () 内的正确答案。

- (1) 存放 A 至少需要 () 个字节;
(2) A 的第 8 列和第 5 行共占 () 个字节;
(3) 若 A 按行存放, 元素 A[8, 5] 的起始地址与 A 按列存放时的元素 () 的起始地址一致。

供选择的答案:

- (1) A. 90 B. 180 C. 240 D. 270 E. 540

- (2) A. 108 B. 114 C. 54 D. 60 E. 150
 (3) A. A[8, 5] B. A[3, 10] C. A[5, 8] D. A[0, 9]

【山东工业大学 2000 三、1 (4分)】 【山东大学 1998 三、1 (4分)】

9. 二维数组 A 的每个元素是由 6 个字符组成的串, 其行下标 $i=0, 1, \dots, 8$, 列下标 $j=1, 2, \dots, 10$ 。若 A 按行先存储, 元素 A[8, 5] 的起始地址与当 A 按列先存储时的元素 () 的起始地址相同。设每个字符占一个字节。【西安电子科技大学 1998 一、2 (2分)】

- A. A[8, 5] B. A[3, 10] C. A[5, 8] D. A[0, 9]

10. 若对 n 阶对称矩阵 A 以行序为主序方式将其下三角形的元素 (包括主对角线上所有元素) 依次存放于一维数组 B $[1..(n(n+1))/2]$ 中, 则在 B 中确定 a_{ij} ($i < j$) 的位置 k 的关系为 ()。

- A. $i*(i-1)/2+j$ B. $j*(j-1)/2+i$ C. $i*(i+1)/2+j$ D. $j*(j+1)/2+i$

【北京航空航天大学 2000 一、2 (2分)】

11. 设 A 是 $n \times n$ 的对称矩阵, 将 A 的对角线及对角线上方的元素以列为主的次序存放在一维数组 B $[1..n(n+1)/2]$ 中, 对上述任一元素 a_{ij} ($1 \leq i, j \leq n$, 且 $i \leq j$) 在 B 中的位置为 ()。

- A. $i(i-1)/2+j$ B. $j(j-1)/2+i$ C. $j(j-1)/2+i-1$ D. $i(i-1)/2+j-1$

【南京理工大学 1999 一、9 (2分)】

12. A[N, N] 是对称矩阵, 将下面三角 (包括对角线) 以行序存储到一维数组 T $[N(N+1)/2]$ 中, 则对任一上三角元素 $a[i][j]$ 对应 T[k] 的下标 k 是 ()。【青岛大学 2002 二、6 (2分)】

- A. $i(i-1)/2+j$ B. $j(j-1)/2+i$ C. $i(j-i)/2+1$ D. $j(i-1)/2+1$

13. 设二维数组 A $[1..m, 1..n]$ (即 m 行 n 列) 按行存储在数组 B $[1..m*n]$ 中, 则二维数组元素 A[i, j] 在一维数组 B 中的下标为 ()。【南京理工大学 1998 一、2 (2分)】

- A. $(i-1)*n+j$ B. $(i-1)*n+j-1$ C. $i*(j-1)$ D. $j*m+i-1$

14. 有一个 100×90 的稀疏矩阵, 非 0 元素有 10 个, 设每个整型数占 2 字节, 则用三元组表示该矩阵时, 所需的字节数是 ()。【南京理工大学 1999 二、8 (2分)】

- A. 60 B. 66 C. 18000 D. 33

15. 数组 A $[0..4, -1..-3, 5..7]$ 中含有元素的个数 ()。【中山大学 1998 二、5 (2分)】

- A. 55 B. 45 C. 36 D. 16

16. 用数组 r 存储静态链表, 结点的 next 域指向后继, 工作指针 j 指向链中结点, 使 j 沿链移动的操作为 ()。【南京理工大学 2001 一、16 (1.5分)】

- A. $j=r[j].next$ B. $j=j+1$ C. $j=j \rightarrow next$ D. $j=r[j] \rightarrow next$

17. 对稀疏矩阵进行压缩存储目的是 ()。【北京工商大学 2001 一、1 (3分)】

A. 便于进行矩阵运算 B. 便于输入和输出 C. 节省存储空间 D. 降低运算的时间复杂度

18. 已知广义表 $L=((x, y, z), a, (u, t, w))$, 从 L 表中取出原子项 t 的运算是 ()。

- A. head(tail(tail(L))) B. tail(head(head(tail(L))))
 C. head(tail(head(tail(L)))) D. head(tail(head(tail(tail(L))))))

【北京邮电大学 1998 二、4 (2分)】

19. 已知广义表 $LS=((a, b, c), (d, e, f))$, 运用 head 和 tail 函数取出 LS 中原子 e 的运算是 ()。

- A. head(tail(LS)) B. tail(head(LS))
 C. head(tail(head(tail(LS)))) D. head(tail(tail(head(LS))))

【西安电子科技大学 2001 应用一、3 (2分)】

20. 广义表 $A=(a, b, (c, d), (e, (f, g)))$, 则下面式子的值为 ()。【北京邮电大学 1999 一、2 (2分)】

Head(Tail(Head(Tail(Tail(A)))))

- A. (g) B. (d) C. c D. d
21. 已知广义表: $A=(a, b)$, $B=(A, A)$, $C=(a, (b, A), B)$, 求下列运算的结果:
 $\text{tail}(\text{head}(\text{tail}(C))) = (\quad)$ 。【长沙铁道学院 1998 三、4 (2分)】
 A. (a) B. A C. a D. (b) E. b F. (A)
22. 广义表运算式 $\text{Tail}(((a, b), (c, d)))$ 的操作结果是 ()。【西安电子科技大学 1998 一、4 (2分)】
 A. (c, d) B. c, d C. ((c, d)) D. d
23. 广义表 $L=(a, (b, c))$, 进行 $\text{Tail}(L)$ 操作后的结果为 ()。【中山大学 1999 一、10】
 A. c B. b, c C. (b, c) D. ((b, c))
24. 广义表 $((a, b, c, d))$ 的表头是 (), 表尾是 ()。【青岛大学 2002 二、7 (2分)】
 A. a B. () C. (a, b, c, d) D. (b, c, d)
25. 广义表 $(a, (b, c), d, e)$ 的表头为 ()。【中山大学 1998 二、6 (2分)】
 A. a B. a, (b, c) C. (a, (b, c)) D. (a)
26. 设广义表 $L=((a, b, c))$, 则 L 的长度和深度分别为 ()。【武汉大学 2000 二、9】
 A. 1 和 1 B. 1 和 3 C. 1 和 2 D. 2 和 3
27. 下面说法不正确的是 ()。【南京理工大学 2001 一、3 (1.5分)】
 A. 广义表的表头总是一个广义表 B. 广义表的表尾总是一个广义表
 C. 广义表难以用顺序存储结构 D. 广义表可以是一个多层次的结构

二、判断题

1. 数组不适合作为任何二叉树的存储结构。() 【南京航空航天大学 1995 五、2 (1分)】
2. 从逻辑结构上看, n 维数组的每个元素均属于 n 个向量。()
 【东南大学 2001 一、2 (1分)】 【中山大学 1994 一、2 (2分)】
3. 稀疏矩阵压缩存储后, 必会失去随机存取功能。() 【中科院软件所 1997 一、1 (1分)】
4. 数组是同类型值的集合。() 【上海海运学院 1996 一、3 (1分) 1999 一、4 (1分)】
5. 数组可看成线性结构的一种推广, 因此与线性表一样, 可以对它进行插入, 删除等操作。()
 【上海交通大学 1998 一、5】
6. 一个稀疏矩阵 $A_{m \times n}$ 采用三元组形式表示, 若把三元组中有关行下标与列下标的值互换, 并把 m 和 n 的值互换, 则就完成了 $A_{m \times n}$ 的转置运算。() 【西安交通大学 1996 二、8 (3分)】
7. 二维以上的数组其实是一种特殊的广义表。() 【北京邮电大学 2002 一、5 (1分)】
8. 广义表的取表尾运算, 其结果通常是个表, 但有时也可是个单元素值。()
 【南京航空航天大学 1996 六、2 (1分)】
9. 若一个广义表的表头为空表, 则此广义表亦为空表。()
 【中科院软件所 1997 一、8 (1分)】 【长沙铁道学院 1998 一、8 (1分)】
10. 广义表中的元素或者是一个不可分割的原子, 或者是一个非空的广义表。()
 【合肥工业大学 2000 二、3 (1分)】
11. 所谓取广义表的表尾就是返回广义表中最后一个元素。() 【合肥工业大学 2001 二、3 (1分)】
12. 广义表的同级元素 (直属于同一个表中的各元素) 具有线性关系。()
 【华南理工大学 2002 一、9 (1分)】
13. 对长度为无穷大的广义表, 由于存储空间限制, 不能在计算机中实现。()
 【华南理工大学 2002 一、10 (1分)】
14. 一个广义表可以为其它广义表所共享。() 【山东大学 2001 一、2 (1分)】

三、 填空题

1. 数组的存储结构采用_____存储方式。【中山大学 1998 一、6 (1 分)】
2. 设二维数组 $A[-20..30, -30..20]$, 每个元素占有 4 个存储单元, 存储起始地址为 200. 如按行优先顺序存储, 则元素 $A[25, 18]$ 的存储地址为_____(1)____; 如按列优先顺序存储, 则元素 $A[-18, -25]$ 的存储地址为_____(2)____。【北方交通大学 1999 二、3 (4 分)】
3. 设数组 $a[1..50, 1..80]$ 的基地址为 2000, 每个元素占 2 个存储单元, 若以行序为主序顺序存储, 则元素 $a[45, 68]$ 的存储地址为_____(1)____; 若以列序为主序顺序存储, 则元素 $a[45, 68]$ 的存储地址为_____(2)____。
【华中理工大学 2000 一、5 (2 分)】
4. 将整型数组 $A[1..8, 1..8]$ 按行优先次序存储在起始地址为 1000 的连续的内存单元中, 则元素 $A[7, 3]$ 的地址是:_____。【合肥工业大学 1999 三、4 (2 分)】
5. 二维数组 $a[4][5][6]$ (下标从 0 开始计, a 有 $4*5*6$ 个元素), 每个元素的长度是 2, 则 $a[2][3][4]$ 的地址是_____。(设 $a[0][0][0]$ 的地址是 1000, 数据以行为主方式存储)
【南京理工大学 2000 二、11 (1.5 分)】
6. 设有二维数组 $A[0..9, 0..19]$, 其每个元素占两个字节, 第一个元素的存储地址为 100, 若按列优先顺序存储, 则元素 $A[6, 6]$ 存储地址为_____。【北京工商大学 2001 二、5 (4 分)】
7. 已知数组 $A[0..9, 0..9]$ 的每个元素占 5 个存储单元, 将其按行优先次序存储在起始地址为 1000 的连续的内存单元中, 则元素 $A[6, 8]$ 的地址为_____。【合肥工业大学 2001 三、4 (2 分)】
8. 已知二维数组 $A[1..10, 0..9]$ 中每个元素占 4 个单元, 在按行优先方式将其存储到起始地址为 1000 的连续存储区域时, $A[5, 9]$ 的地址是:_____。【厦门大学 2002 六、5 (4 分)】
9. 用一维数组 B 与列优先存放带状矩阵 A 中的非零元素 $A[i, j]$ ($1 \leq i \leq n, i-2 \leq j \leq i+2$), B 中的第 8 个元素是 A 中的第_____(1)____行, 第_____(2)____列的元素。【北京邮电大学 2001 二、3 (4 分)】
10. 设数组 $A[0..8, 1..10]$, 数组中任一元素 $A[i, j]$ 均占内存 48 个二进制位, 从首地址 2000 开始连续存放在主内存里, 主内存字长为 16 位, 那么
(1) 存放该数组至少需要的单元数是_____;
(2) 存放数组的第 8 列的所有元素至少需要的单元数是_____;
(3) 数组按列存储时, 元素 $A[5, 8]$ 的起始地址是_____。【中国矿业大学 2000 一、4 (4 分)】
11. 设 n 行 n 列的下三角矩阵 A 已压缩到一维数组 $B[1..n*(n+1)/2]$ 中, 若按行为主序存储, 则 $A[i, j]$ 对应的 B 中存储位置为_____。【武汉大学 2000 一、1】
12. n 阶对称矩阵 a 满足 $a[i][j]=a[j][i], i, j=1..n$, 用一维数组 t 存储时, t 的长度为_____(1)____, 当 $i=j, a[i][j]=t[(2)], i>j, a[i][j]=t[(3)], i<j, a[i][j]=t[(4)]$ 。【青岛大学 2001 六、1 (3 分)】
13. 已知三对角矩阵 $A[1..9, 1..9]$ 的每个元素占 2 个单元, 现将其三条对角线上的元素逐行存储在起始地址为 1000 的连续的内存单元中, 则元素 $A[7, 8]$ 的地址为_____。【合肥工业大学 2000 三、4 (2 分)】
14. 设有一个 10 阶对称矩阵 A 采用压缩存储方式 (以行为主序存储: $a_{11}=1$), 则 a_{85} 的地址为_____。
【西安电子科技大学 1999 软件 一、3 (2 分)】
15. 所谓稀疏矩阵指的是_____。【厦门大学 2001 一、2 (14%/5 分)】
16. 对矩阵压缩是为了_____。【北京理工大学 2000 二、3 (2 分)】
17. 上三角矩阵压缩的下标对应关系为:_____。【福州大学 1998 二、6 (2 分)】【南京大学

1999】

18. 假设一个 15 阶的上三角矩阵 A 按行优先顺序压缩存储在一维数组 B 中, 则非零元素 $A_{9,9}$ 在 B 中的存储位置 $k=$ _____。(注: 矩阵元素下标从 1 开始)【北京工商大学 2001 二、1 (4 分)】

$$A = \begin{bmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & \dots & a_{nn} \end{bmatrix}$$

19. 设下三角矩阵 $A=$

如果按行序为主序将下三角元素 A_{ij} (i, j) 存储在一个一维数组 $B[1..n(n+1)/2]$ 中, 对任一个三角矩阵元素 A_{ij} , 它在数组 B 中的下标为_____。【北方交通大学 2001 二、3】

20. 当广义表中的每个元素都是原子时, 广义表便成了_____。【长沙铁道学院 1998 二、8 (2 分)】

21. 广义表的表尾是指除第一个元素之外, _____。【中山大学 1998 一、7 (1 分)】

22. 广义表简称表, 是由零个或多个原子或子表组成的有限序列, 原子与表的差别仅在于 (1)_____。为了区分原子和表, 一般用 (2)_____表示表, 用 (3)_____表示原子。一个表的长度是指 (4)_____, 而表的深度是指 (5)_____。【山东工业大学 2000 一、3 (3 分)】 【山东大学 1998 一、2 (3 分)】

23. 广义表的_____ 定义为广义表中括弧的重数。【重庆大学 2000 一、5】

24. 设广义表 $L=((), ())$, 则 $head(L)$ 是 (1)_____; $tail(L)$ 是 (2)_____; L 的长度是 (3)_____; 深度是 (4)_____。

【中科院计算所 1998 一、2 (4 分)】 【中国科技大学 1998 一、2 (4 分)】

25. 已知广义表 $A=(9, 7, (8, 10, (99)), 12)$, 试用求表头和表尾的操作 $Head()$ 和 $Tail()$ 将原子元素 99 从 A 中取出来。【西安交通大学 1996 四、5 (5 分)】

26. 广义表的深度是_____。【北京轻工业学院 2000 一、1 (2 分)】

27. 广义表 $(a, (a, b), d, e, ((i, j), k))$ 的长度是 (1)_____, 深度是 (2)_____。【山东大学 2001 三、9 (2 分)】

【西安电子科技大学 2001 软件 一、5 (2 分)】 【哈尔滨工业大学 2001 一、2 (2 分)】

28. 已知广义表 $LS=(a, (b, c, d), e)$, 运用 $head$ 和 $tail$ 函数取出 LS 中原子 b 的运算是_____。【燕山大学 2001 二、2 (3 分)】

29. 广义表 $A=((((a, b), (c, d, e))))$, 取出 A 中的原子 e 的操作是: _____。

【合肥工业大学 1999 三、5 (2 分)】

30. 设某广义表 $H=(A, (a, b, c))$, 运用 $head$ 函数和 $tail$ 函数求出广义表 H 中某元素 b 的运算式_____。

【北京科技大学 1997 一、5】

31. 广义表 $A((((), (a, (b, c))))$, $head(tail(head(tail(head(A)))))$ 等于_____。

【合肥工业大学 2000 三、5 (2 分)】

32. 广义表运算式 $HEAD(TAIL(((a, b, c), (x, y, z))))$ 的结果是_____。

【西安电子科技大学 1999 软件 一、9 (2 分)】

33. 已知广义表 $A=((((a, b), (c), (d, e))))$, $head(tail(tail(head(A))))$ 的结果是_____。

【合肥工业大学 2001 三、5 (2 分)】

34. 利用广义表的 $GetHead$ 和 $GetTail$ 操作, 从广义表 $L=((apple, pear), (banana, orange))$ 中分离出原子 banana 的函数表达式是_____。【山东大学 2001 三、6 (2 分)】

35. 已知 a 数组元素共 5 个, 依次为 12, 10, 5, 3, 1; b 数组元素共 4 个, 依次为 4, 6, 8, 15,

则执行如下所示的过程语句 sort 后得到 c 数组各元素依次为 15, 12, 10, 8, 6, 5, 4, 3, 1; 数组 a, b, c 的长度分别为 l=5, m=4, n=9 请在程序中方框内填入正确的成分, 完成上述要求。

```
PROCEDURE sort;
VAR i, j, k, x: integer; d: ARRAY[1..m] OF integer;
BEGIN
  FOR i:=1 TO m DO d[i]:= (1) _____;
  i:=1; j:=1; k:=1;
  WHILE (i<=l) AND (j<=m) DO
    BEGIN
      IF a[i]>d[j] THEN BEGIN (2) _____; (3) _____ END
      ELSE BEGIN (4) _____; (5) _____ END;
      c[k]:=x; (6) _____
    END;
    WHILE (7) _____ DO
      BEGIN c[k]:=a[i]; k:=k+1; i:=i+1; END;
    WHILE (8) _____ DO
      BEGIN c[k]:=d[j]; k:=k+1; j:=j+1; END;
  END. {sort} 【上海交通大学 1998 七 (12 分)】
```

36. 下列程序段 search(a, n, k) 在数组 a 的前 n (n>=1) 个元素中找出第 k (1<=k<=n) 小的值。这里假设数组 a 中各元素的值都不相同。

```
#define MAXN 100
int a[MAXN], n, k;
int search_c(int a[], int n, int k)
{int low, high, i, j, m, t;
  k--; low=0; high=n-1;
  do {i=low; j=high; t=a[low];
    do {while (i<j && t<a[j]) j--;
      if (i<j) a[i++] = a[j];
      while (i<j && t>=a[i]) i++;
      if (i<j) a[j--] = a[i];
    } while (i<j);
    a[i]=t;
    if (1) _____;
    if (i<k) low= (2) _____; else high= (3) _____;
  } while (4) _____;
  return(a[k]);
} 【上海大学 1999 一、1 (8 分)】
```

37. 完善下列程序, 每小题在 PASCAL 语言 (a) 和 C 语言 (b) 中任选一题。下面是一个将广义表逆置的过程。例如原来广义表为 ((a, b), c, (d, e)), 经逆置后为: ((e, d), c, (b, a))。

(a) 算法的 PASCAL 语言过程描述 (编者略): (b) 算法的 C 语言过程描述:

```
typedef struct glistnode
{int tag;
  struct glistnode *next;
  union {char data;
```

```

        struct{struct glistnode *hp,*tp;}ptr;
    }val;
}*glist,gnode;
glist reverse(p)
glist p;
{glist q,h,t,s;
  if(p==NULL) q=NULL;
  else
    {if(1) { q=(glist)malloc(sizeof(gnode)); q->tag=0;
            q->val.data=p->val.data; }
      else { (2)
            if (3)
              {t=reverse(p->val.ptr.tp); s=t;
               while(s->val.ptr.tp!=NULL) s=s->val.ptr.tp;
               s->val.ptr.tp=(glist)malloc(sizeof(gnode));
               s=s->val.ptr.tp;s->tag=1;s->val.ptr.tp=NULL;
               s->val.ptr.hp=h; (4) }
              else {q=(glist)malloc(sizeof(gnode));q->tag=1;
                    q->val.ptr.tp=NULL; (5); }
            }
          }
    return(q);
  }
}

```

【上海大学 2002 六、3 (10 分)】

38. 完善下列程序，每小题在 PASCAL 语言 (a) 和 C 语言 (b) 中任选一题。下面的程序将数列 1, 2, 3, ..., n*n, 依次按蛇型方式存放在二维数组 A[1..n, 1..n] 中。即 (示意图编者略)。

(a) 算法的 PASCAL 语言程序描述 (编者略): (b) 算法的 C 语言程序描述:

```

#define NMAX 10
#include "stdio.h"
main()
{ int i,j,n,k,p,q,m;
  int a[NMAX][NMAX];
  scanf("%d",&n);
  m=1;
  for(k=1; (1); k++)
    {if (k<n) q=k; else(2);
     for(p=1;p<=q;p++)
       {if(3) {i=q-p+1; j=p; }
        else{i=p; j=q-p+1; }
        if(4){i=i+n-q; j=j+n-q; }
        a[i][j]=m; (5);
       }
     for (i=1; i<=n;i++)
       { for (j=1; j<=n; j++)

```

```
printf ("%4d", a[i][j]); printf( "\n" );
}
}
```

【上海大学 2002 六、1 (10 分)】

39. 约瑟夫环问题：设有 n 个人围坐一圈，并按顺时针方向 $1 \sim n$ 编号。从第 s 个人开始进行报数，报数到第 m 个人，此人出圈，再从他的下一个人重新开始从 1 到 m 的报数进行下去，直到所有的人都出圈为止。

PROCEDURE Josef (A:ARRAY [1..n] OF integer; s,m:integer);

BEGIN

FOR i:= 1 TO n DO A[i]:=i;

s1:=s;

FOR i:=n DOWNT0 2 DO

BEGIN s1:= (1); //计算出圈人 s1

IF s1=0 THEN (2);

w:=A[s1]; //A[s1]出圈

FOR j:= (3) DO A[j]:=A[j+1];

A[i]:=w;

END;

write(' 出圈序列为: '); //输出出圈序列

FOR i:=n DOWNT0 1 DO write(A[i]); writeln ;

END; 【华南师范大学 2000 五、2 (9 分)】

40. 设有一个背包可以放入的物品重量为 S ，现有 n 件物品，重量分别为 W_1, W_2, \dots, W_n 。问能否从这 n 件物品中选择若干件放入背包，使得放入的重量之和正好是 S 。设布尔函数 $Knap(S, n)$ 表示背包问题的解， $W_i (i=1, 2, \dots, n)$ 均为正整数，并已顺序存储在数组 W 中。请在下列算法的下划线处填空，使其正确求解背包问题。

Knap(S, n)

若 $S=0$

则 $Knap \leftarrow true$

否则若 $(S < 0)$ 或 $(S > 0$ 且 $n < 1)$

则 $Knap \leftarrow false$

否则若 $Knap(1), \dots = true$

则 print($W[n]$); $Knap \leftarrow true$

否则 $Knap \leftarrow Knap(2), \dots$

【山东工业大学 1996 五 (10 分) 1998 二、1 (4 分)】

四 应用题

1. 数组 $A[1..8, -2..6, 0..6]$ 以行为主序存储，设第一个元素的首地址是 78，每个元素的长度为 4，试求元素 $A[4, 2, 3]$ 的存储首地址。【厦门大学 1998 五、1 (5 分)】

2. 已知 b 对角矩阵 $(a_{ij})_{n \times n}$ ，以行主序将 b 条对角线上的非零元存储在一维数组中，每个数据元素占 L 个存储单元，存储基地址为 S ，请用 i, j 表示出 a_{ij} 的存储位置。【北方交通大学 1996 三 (10 分)】

3. 数组 A 中，每个元素 $A[i, j]$ 的长度均为 32 个二进制，行下标从 -1 到 9，列下标从 1 到 11，从首地址 S 开始连续存放主存储器中，主存储器字长为 16 位。求：

(1) 存放该数组所需多少单元？

- (2) 存放数组第 4 列所有元素至少需多少单元?
- (3) 数组按行存放时, 元素 $A[7, 4]$ 的起始地址是多少?
- (4) 数组按列存放时, 元素 $A[4, 7]$ 的起始地址是多少? 【大连海事大学 1996 四、1 (6 分)】
4. 假设按低下标优先存储整型数组 $A(-3:8, 3:5, -4:0, 0:7)$ 时, 第一个元素的字节存储地址是 100, 每个整数占 4 个字节, 问 $A(0, 4, -2, 5)$ 的存储地址是什么? 【清华大学 1996 三】
5. 设有三维数组 $A[-2:4, 0:3, -5:1]$ 按列序存放, 数组的起始地址为 1210, 试求 $A(1, 3, -2)$ 所在的地址。【长沙铁道学院 1997 三、1 (3 分)】
6. 三维数组 $A[1..10, -2..6, 2..8]$ 的每个元素的长度为 4 个字节, 试问该数组要占多少个字节的存储空间? 如果数组元素以行优先的顺序存贮, 设第一个元素的首地址是 100, 试求元素 $A[5, 0, 7]$ 的存贮首地址。
- 【上海海运学院 1995 三 (6 分) 1997 三 (8 分)】
7. 设有五对角矩阵 $A=(a_{ij})_{20 \times 20}$, 按特殊矩阵压缩存储的方式将其五条对角线上的元素存于数组 $A[-10:m]$ 中, 计算元素 $A[15, 16]$ 的存储位置。【东北大学 1999 一、2 (4 分)】
8. 数组 $A[0..8, 1..10]$ 的元素是 6 个字符组成的串, 则存放 A 至少需要多少个字节? A 的第 8 列和第 5 行共占多少个字节? 若 A 按行优先方式存储, 元素 $A[8, 5]$ 的起始地址与当 A 按列优先方式存储时的哪个元素的起始地址一致? 【厦门大学 2000 五、3 (14%/3 分)】
9. 若按照压缩存储的思想将 $n \times n$ 阶的对称矩阵 A 的下三角部分 (包括主对角线元素) 以行序为主序方式存放于一维数组 $B[1..n(n+1)/2]$ 中, 那么, A 中任一个下三角元素 $a_{ij}(i \geq j)$, 在数组 B 中的下标位置 k 是什么? 【北京航空航天大学 1998 一、4 (4 分)】
10. 设 $m \times n$ 阶稀疏矩阵 A 有 t 个非零元素, 其三元组表表示为 $LTMA[1..(t+1), 1..3]$, 试问: 非零元素的个数 t 达到什么程度时用 LTMA 表示 A 才有意义? 【北京航空航天大学 1998 一、5 (4 分)】
11. 利用三元组存储任意稀疏数组时, 在什么条件下才能节省存储空间。【西北工业大学 1998 三、2 (5 分)】
12. 对一个有 t 个非零元素的 A_m 矩阵, 用 $B[0..t][1..3]$ 的数组来表示, 其中第 0 行的三个元素分别为 m, n, t, 从第一行开始到最后一行, 每行表示一个非零元素; 第一列为矩阵元素的行号, 第二列为其列号, 第三列为其值。对这样的表示法, 如果需要经常进行该操作——确定任意一个元素 $A[i][j]$ 在 B 中的位置并修改其值, 应如何设计算法可以使时间得到改善? 【长沙铁道学院 1998 四、4 (6 分)】
13. 有一个二维数组 $A[0:8, 1:5]$, 每个数组元素用相邻的 4 个字节存储, 存储器按字节编址, 假设存储数组元素 $A[0, 1]$ 的第一个字节的地址是 0, 那么存储数组的最后一个元素的第一个字节的地址是多少? 若按行存储, 则 $A[3, 5]$ 和 $A[5, 3]$ 的第一个字节的地址是多少? 若按列存储, 则 $A[7, 1]$ 和 $A[2, 4]$ 的第一个字节的地址是多少? 【上海海运学院 1999 三 (10 分)】
14. 设有三对角矩阵 $(a_{i,j})_{m \times n}$, 将其三条对角线上的元素逐行的存于数组 $B(1:3n-2)$ 中, 使得 $B[k]=a_{i,j}$, 求:
- (1) 用 i, j 表示 k 的下标变换公式;
- (2) 若 $n=10^3$, 每个元素占用 L 个单元, 则用 $B[K]$ 方式比常规存储节省多少单元。
- 【西安电子科技大学 1996 二、4 (5 分)】
15. 已知 A 为稀疏矩阵, 试从空间和时间角度, 比较采用两种不同的存储结构 (二维数组和三元组表) 完成求 $\sum_{i=1}^n a_{ii}$ 运算的优缺点。【西安电子科技大学 1996 二、6 (5 分)】
16. 特殊矩阵和稀疏矩阵哪一种压缩存储后失去随机存取的功能? 为什么?

【北京邮电大学 2001 三、1 (5 分)】

17. 试叙述一维数组与有序表的异同。【西安电子科技大学 1999 计应用一、2 (5 分)】

18. 一个 $n \times n$ 的对称矩阵, 如果以行或列为主序存入内存, 则其容量为多少?

【西安电子科技大学 1999 计应用 一、3 (5 分)】

19. 给出数组 $A: \text{ARRAY}[3..8, 2..6] \text{ OF INTEGER}$; 当它在内存中按行存放和按列存放时, 分别写出数组元素 $A[i, j]$ 地址计算公式 (设每个元素占两个存储单元)。【南开大学 1998 一 (8 分)】

20. 已知 n 阶下三角矩阵 A (即当 $i < j$ 时, 有 $a_{ij} = 0$), 按照压缩存储的思想, 可以将其主对角线以下所有元素 (包括主对角线上元素) 依次存放于一维数组 B 中, 请写出从第一列开始采用列序为主序分配方式时在 B 中确定元素 a_{ij} 的存放位置的公式。【北京航空航天大学 1999 二 (10 分)】

【中山大学 1999 三 2】

21. 设有三对角矩阵 $(a_{ij})_{n \times n}$, 将其三条对角线上的元素逐行地存于数组 $B(1:3n-2)$ 中, 使得 $B[k] = a_{ij}$, 求: (1) 用 i, j 表示 k 的下标变换公式;

(2) 用 k 表示 i, j 的下标变化公式。

【山东科技大学 2001 一、6 (6 分)】【长沙铁道学院 1997 五、1 (10 分)】

【东北大学 2002 一 (4 分)】【北京工业大学 2000 二、1 (9 分)】【南京航空航天大学 2000

四】

22. 算法 Print 及所引用的数组 A 的值如下, 写出调用 Print(1) 的运行结果 (其中 $n=15$)。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	E	F	G	O	O	H	O	I	J	K	L

PROCEDURE print (i: integer);

BEGIN

IF ($i \leq n$) AND ($A[i] \neq '0'$) THEN

BEGIN Print (2*i); write (A[i]); Print (2*i+1); END;

END;

【合肥工业大学 1999 四、1 (5 分)】

23. 设数组 A 的长度为 $2N$, 前 N 个元素 $A[1..N]$ 递减有序, 后 N 个元素 $A[N+1..2N]$ 递增有序, 且 $2N$ 是 2 的整数次幂, 即 $k = \log_2 2N$ 为整数。例如 $A[1..8] = [90, 85, 50, 10, 30, 65, 80, 100]$ 满足上述要求, 这里 $N=4, k=3$, A 的前 4 个元素和后 4 个元素分别递减和递增有序。用此例调用如下的 Demo 过程, 并要求:

(1) 给出 for 循环中每次执行 PerfectShuffle (A, N) 和 CompareExchange (A, N) 的结果;

(2) 解释 Demo 的功能; (3) 给出 Demo 的时间复杂度。

PROCEDURE PerfectShuffle(VAR A:arraytype; N:integer)

[i:=1; j:=1;

WHILE i<=N DO

[B[j]:=A[i]; B[j+1]:=A[i+N]; i:=i+1; j:=j+2;]

A[1..2N]:=B[1..2N]; //B copy to A

]

PROCEDURE CompareExchange(VAR A:arraytype; N:integer)

[j:=1;

WHILE j<2N DO

[IF A[j]>A[j+1] THEN A[j] \leftrightarrow A[j+1]; //交换 A[j] 和 A[j+1]

j:=j+2;]

]

PROCEDURE Demo (VAR A:arraytype;N:integer)

//A 的长度为 $2N$, $k=\log_2 2N$ 为整数

```
[ FOR i:=1 TO log22N DO
  [ PerfectShuffle(A,N); CompareExchange(A,N); ]
]
```

【中科院计算所 1998 四 (15 分)】 【中国科技大学 1998 4 (15 分)】

24. 现有算法及正整数 n 和数组 A 如下, 求数组 C 的值。

```
VAR A,B,C:Array[1..100] of integer;
FUNC AAA(s,t:integer):integer;
  IF s=t THEN IF B[s]=0 THEN AAA:=S ELSE AAA:=-s ELSE
  BEGIN
    l:=AAA(s, (s+t) DIV 2);
    r:=AAA((s+t) DIV 2+1, t);
    IF l>0 THEN AAA:=l ELSE AAA:=r;
    IF (r>0) AND (A[l]>A[r]) THEN AAA:=r
  END
ENDF;
PROC BBB;
  FOR i:=1 TO n DO B[i]:=0;
  FOR i:=1 TO n DO B[AAA(1,n)]:=i;
  FOR i:=1 TO n DO C[B[i]]:=A[i];
ENDP;
```

初始值: $n=10$, $A=\{121, 22, 323, 212, 636, 939, 828, 424, 55, 262\}$;

【北京邮电大学 2002 五、1 (10 分)】

25. 设有矩阵 a 且 $a = \begin{bmatrix} 2 & 1 & 3 \\ 3 & 3 & 1 \\ 1 & 2 & 1 \end{bmatrix}$ 执行下列语句后, 矩阵 c 和 a 的结果分别是什么?

```
(1) FOR i:=1 TO 3 DO
      FOR j:=1 TO 3 DO c[i,j]:=a[a[i,j],a[j,i]]
(2) FOR i:=1 TO 3 DO
      FOR j:=1 TO 3 DO a[i,j]:=a[a[i,j],a[j,i]]
```

【浙江大学 1997 三 (8 分)】

26. 设矩阵 A 为 $\begin{bmatrix} 2 & 1 & 3 \\ 3 & 3 & 1 \\ 1 & 2 & 1 \end{bmatrix}$

(1) 执行语句

```
FOR i:=1 TO 3 DO
  FOR j:=1 TO 3 DO ①
    C[i,j]:=A[A[i,j],A[j,i]]
```

结果 C 矩阵的值是什么?

(2) 所选择的下标 i, j 的次序有关系吗?

(3) 在语句①中, 用 A 代替 C , A 的结果值是什么?

(4) 对 i, j 这对下标取反序, 即

$(3, 3), (3, 2), (3, 1), \dots, (1, 3), (1, 2), (1, 1)$

重复执行 (3), 把所得结果与 (3) 中所得结果作比较。【吉林大学 1995 二 (15 分)】

27. 指出下列算法中错误、低效之处, 并将其改成一个正确且高效的算法。

```
PROCEDURE delk(A, m, last, i, k);
{从数组 A[1..last]中删除第 i 个元素起的 k 个元素, m 为 A 上限}
BEGIN
  IF (i < 1) OR (i > last) OR (k < 0) OR (last > m)
  THEN write ('error')
  ELSE FOR count := 1 TO k TO
    [FOR j := last DOWNTO i+1 DO
      A[j-1] := A[j];
      last := last-1]
```

ENDP; {delk} 【北方交通大学 1997 一 (10 分)】

28. 设输入为整数数组 $A[1..n]$, 其中 $1 \leq A[i] \leq k$ ($1 \leq i \leq n$); 输出数组为 $b[1..n]$; $c[1..k]$ 是临时工作空间; 阅读下述算法后, 回答下列问题:

```
PROC Demo(A, B, k) {
  (1) FOR i := 1 TO k DO C[i] := 0;
  (2) FOR j := 1 TO n DO C[A[j]] := C[A[j]] + 1;
  (3) FOR i := 2 TO k DO C[i] := C[i] + C[i-1]
  (4) FOR j := n DOWNTO 1 DO {
  (5) B[C[A[j]]] := A[j];
  (6) C[A[j]] := C[A[j]] - 1 } }
```

(a). 当标号 (2) 行的循环执行完后, $C[i]$ ($1 \leq i \leq n$) 的值有何意义?

(b). 当标号 (3) 行的循环执行完后, $C[i]$ ($1 \leq i \leq n$) 的值有何意义?

(c). 算法执行后, B 的内容有何特点?

(d). 当 $k=0(n)$ 时, 算法的时间复杂度是多少? 【中科院软件所 1997 二、2 (9 分)】

29. 上三角阵 $A(N \times N)$ 按行主序压缩存放在数组 B 中, 其中 $A[i, j] = B[k]$. 写出用 i, j 表示的 k .

【北京工业大学 2001 二、1 (5 分)】

30. 设有上三角矩阵 $(a_{ij})_{n \times n}$, 将其上三角中的元素按先行后列的顺序存于数组 B (1:m) 中, 使得 $B[k] = a_{ij}$ 且 $k = f_1(i) + f_2(j) + c$, 请推导出函数 f_1, f_2 和常数 c , 要求 f_1 和 f_2 中不含常数项。

【中科院自动化所 1999】 【山东科技大学 2002 一、5 (6 分)】

31. 设矩阵 $A = \begin{bmatrix} 2 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 \\ 0 & 3 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix}$

(1) 若将 A 视为对称矩阵, 画出对其压缩存储的存储表, 并讨论如何存取 A 中元素 a_{ij} ($0 \leq i, j < 4$);

(2) 若将 A 视为稀疏矩阵, 画出 A 的十字链表结构。【北京科技大学 1997 三 (10 分)】

32. 设对称矩阵 $A = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 5 \\ 2 & 0 & 5 & 0 \end{bmatrix}$

(1) 若将 A 中包括主对角线的下三角元素按列的顺序压缩到数组 S 中, 即 S:

1	0	0	2	3	0	0	0	5	0
下标: 1	2	3	4	5	6	7	8	9	10

试求出 A 中任一元素的行列下标 $[i, j]$ ($1 \leq i, j \leq 4$) 与 S 中元素的下标 K 之间的关系.

(2) 若将 A 视为稀疏矩阵时, 画出其三元组表形式压缩存储表。【北京科技大学 1999 三 (10 分)】

$$A = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix}$$

33. 设对角线矩阵 $A = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix}$ (行列下标 i, j 满足: $1 \leq i, j \leq 5$)

(1) 若将矩阵 A 压缩存储到数组 S 中:

1	2	1	0	1	2	1	0	0	0	1	3	5
下标: 1	2	3	4	5	6	7	8	9	10	11	12	13

试求出 A 中已存储之元素的行列下标 (i, j) 与 S 中元素的下标 K 之间的关系

(2) 若将 A 视为稀疏矩阵时, 请画出其行逻辑链接顺序表。【北京科技大学 2000 三 (10 分)】

34. 设有一个三维数组 $a[c1:d1, c2:d2, c3:d3]$, 其中 $ci:di$ 是第 i 维的界偶, 如该数组在内存中按行排列, 且 $a[c1, c2, c3]$ 的地址为 a_0 , 那么请导出下标变量 $a[i1, i2, i3]$ 的地址, 假设每个元素占 L 个单元。

【山东师范大学 1999 四、1 (5 分)】

$$A = \begin{bmatrix} a_{11} & 0 & \dots & \dots & \dots & 0 & a_{1n} \\ 0 & a_{22} & \dots & \dots & \dots & a_{2,n-1} & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & a_{\frac{n+1}{2}, \frac{n+1}{2}} & \dots & 0 & 0 \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & 0 & \dots & \dots & \dots & 0 & a_{nn} \end{bmatrix}$$

35. 假定有下列 $n \times n$ 矩阵 (n 为奇数) $A = \begin{bmatrix} a_{11} & 0 & \dots & \dots & \dots & 0 & a_{1n} \\ 0 & a_{22} & \dots & \dots & \dots & a_{2,n-1} & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & a_{\frac{n+1}{2}, \frac{n+1}{2}} & \dots & 0 & 0 \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & 0 & \dots & \dots & \dots & 0 & a_{nn} \end{bmatrix}$

如果用一维数组 B 按行主次序存储 A 的非零元素, 问:

(1) A 中非零元素的行下标与列下标的关系;

(2) 给出 A 中非零元素 a_{ij} 的下标 (i, j) 与 B 中的下标 R 的关系;

(3) 假定矩阵中每个元素占一个存储单元且 B 的起始地址为 A_0 , 给出利用 a_{ij} 的下标 (i, j) 定位在 B 中的位置公式。【上海交通大学 1998 三 (12 分)】

36. 对于一个对称矩阵采用压缩存储, 只存放它的上三角部分, 并按列存放。例如对于一个 $n \times n$ 的对称矩阵 A (如右图), 用一个一维数组 B 来存放它的上三角部分:

$$B = [A_{11}, A_{12}, A_{22}, A_{13}, A_{23}, A_{33}, A_{14}, \dots, A_{1n}, A_{2n}, \dots, A_{nn}]$$

同时有两个函数: $\text{MAX}(i, j)$ 和 $\text{MIN}(i, j)$, 分别计算下标 i 和 j 中的大者与 smaller。试利用它们给出求任意一个 A_{ij} 在 B 中存放位置的公式。(若式中没有 $\text{MAX}(I, j)$ 和 $\text{MIN}(i, j)$ 则不给分)

【清华大学 1997 五 (10 分)】

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & \dots & A_{1n} \\ & A_{22} & A_{23} & & \dots & A_{2n} \\ & & A_{33} & \dots & \dots & A_{3n} \\ & & & \dots & \dots & \dots \\ & & & & \dots & \dots \\ & & & & & A_{nn} \end{bmatrix}$$

37. 用三元组表示稀疏矩阵的转置矩阵, 并简要写出解题步骤。【山东工业大学 1995 五 (10 分)】

38. 简述广义表属于线性结构的理由。【西北大学 2000 一、5 (3 分)】
39. 数组, 广义表与线性表之间有什么样的关系? 【西北工业大学 1998 一、2 (4 分)】
40. 什么是广义表? 请简述广义表和线性表的主要区别。【北京大学 1997 二、2 (5 分)】
41. 求下列广义表的运算结果。【南京航空航天大学 1998 三 (10 分)】

- (1) CAR(CDR((a, b), (c, d), (e, f)))
- (2) CDR(CAR((a, b), (c, d), (e, f)))
- (3) CAR(CDR(CAR((a, b), (e, f))))
- (4) CDR(CAR(CDR((a, b), (e, f))))
- (5) CDR(CDR(CAR((a, b), (e, f))))

注：CAR 运算相当于有些教材中的 Head 运算，CDR 运算相当于 Tail 运算。

42. 利用广义表的 Head 和 Tail 运算, 把原子 d 分别从下列广义表中分离出来, L1 = (((((a), b), d), e)); L2 = (a, (b, ((d)), e))。【北方交通大学 1996 一、2 (5 分)】

类似本题的另外叙述有:

- (1) 已知广义表 $L = (((a)), (b)), (c), d)$ ，试利用 head 和 tail 运算把原子项 c 从 L 中分离出来。

【北京邮电大学 1992 三、2 (25/4 分)】【青岛海洋大学 1999 一、6 (5 分)】

- (2) 画出下列广义表的存储结构图, 并利用取表头和取表尾的操作分离出原子 e 。

$$(a, ((b, ((c, ((d, e)))))).$$

【清华大学 1995 二 (10 分)】

- (3) 已知广义表 $A=((a, b, c), (d, e, f))$ 试写出从表 A 中取出原子元素 e 的运算。

【西安电子科技大学 1996 二、3 (5 分)】

- (4) 请将香蕉 banana 用工具 H()—Head(), T()—Tail() 从 L 中取出。

L=(apple, (orange, (strawberry, (banana)), peach), pear)

【北京邮电大学 2000 三、2 (5 分)】

- (5) 试利用广义表取表头 $\text{head}(ls)$ 和取表尾 $\text{tail}(ls)$ 的基本运算, 将原子 d 从下列表中分解出来, 请写出每一步的运算结果。

$L = ((a, (b)), ((c, d)), (e, f))$ 【北京工商大学 2001 三 (10 分)】

- (6) 画出广义表 $A = (a, (b, ()), (((), c)))$ 的第一种存储结构（表结点第二指针指向余表）图，并用取首元（head()）和取尾元（tail()）函数表示原子 c 。【北京工业大学 2001 二、2 (5 分)】

43. 画出下列广义表的两种存储结构图 $(((), A, (B, (C, D))), (E, F))$ 。【南京航空航天大学 1999 三 (10 分)】

44. 假设采用以下两种结点的链表作为广义表的存贮结构,表结点: (tag=1, hp, tp), 元素结点: (tag=0, data)。请画出下列广义表的存储结构图,并求它的深度和长度。

$$(\quad), \quad (\quad), \quad (\quad)$$

【北方交通大学 1998 一 (13 分)】

45. 知广义表 $A = (((a)), (b), c, (a), (((d, e))))$

- (1) 画出其一种存贮结构图:

- (2) 写出表的长度与深度:

- (3) 用求头部, 尾部的方式求出 e 。【东北大学 1997 一、2 (5 分)】

46. 画出具有共享结构广义表 $((b, c), d), (a), ((a), ((b, c), d)), e, ()$ 的存贮表示。

【北京工业大学 1996 一、3 (6 分)】

47. 广义表的结点结构如下: (TAG, DATA, LINK)。其中 LINK 为指向表中下一元素的指针; TAG 为标志域; DATA 为数据域, 具体含义如下: TAG=0 表示该结点为原子结点, DATA 为其数据; TAG=1

表示该结点为一个子表，DATA 为指向该子表的指针。

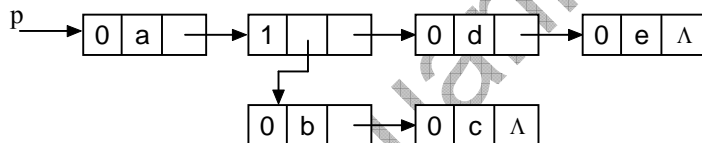
(1) 说明下列算法 A 的功能(注:算法中 p, t, m, n, r, q 为指针;算法中的 NIL 对应图中的 Λ)

```

PROCEDURE A(p, t)
BEGIN
  q:=NIL;
  WHILE p<>NIL DO
  BEGIN
    IF p^.TAG<>0 THEN
    BEGIN
      m:=p^.DATA;
      A(m, n);
      p^.DATA:=n;
    END;
    r:=p^.LINK;
    p^.LINK:=q;
    q:=p;
    p:=r;
  END;
  t:=q;
END.

```

(2) 对于 p 所指的广义表,画出执行算法 A 后的表结构以及 p, t 的值:



【北方交通大学 1993 六 (20 分)】

类似本题的另外叙述有:

题目基本相同,差别仅在于子表(b, c)与原子 d 的前后顺序颠倒。【浙江大学 1994 六 (7 分)】

48. 写出对广义表 $A=(x, ((a, b), c, d))$ 作运算 $\text{head}(\text{head}(\text{tail}(A)))$ 后的结果。

【西安电子科技大学 2000 计应用 一、5 (5 分)】

49. 写出广义表的运算结果: $\text{TAIL}[\text{HEAD}[\text{TAIL}[(a, b), (c, d)]]]=?$

【武汉交通科技大学 1996 二、7 (3 分)】

50. 广义表中原子个数是否即为广义表的长度? 【西安电子科技大学 2000 计应用一、9 (5 分)】

51. 给出下列所示的 3 元多项式的广义表表示 (分别以 X_1, X_2, X_3 第一到第三层变元)

$$P(X_1X_2X_3) = X_1^5X_2^3X_3+2X_1^5X_2^2X_3^4+5X_1^5X_2^3X_3^3+3X_1X_2^4X_3^2+X_2X_3+6$$

【华南理工大学 2001 一、2 (4 分)】

52. 设某表 H 如下:

A		B	C		X
a1	a2	b1	c1	c2	

其中 A, B, C 为子表名, a1, a2, b1, c1, c2, x 为其元素。

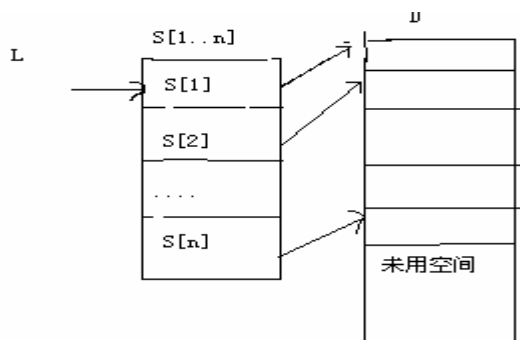
(1) 试用广义表形式表示 H, 并写出运算 $\text{HEAD}(H)$ 和 $\text{TAIL}(H)$ 函数从 H 中取出单元素 a2 的运算;

(2) 画出表 H 的链式存储结构;【北京科技大学 1998 三 (10 分)】

五 算法设计题

1. 设有大小不等的 n 个数据组 (n 个数据组中数据的总数为 m)，顺序存放在空间区 D 内, 每个数据占一个存储单元，数据组的首地址由数组 S 给出, (如下图所示)，试编写将新数据 x 插入到第 i 个数据组的末尾且属于第 i 个数据组的算法，插入后，空间区 D 和数组 S 的相互关系仍保持正确。

【东北大学 2000 六 (15 分)】



2. 以三元组表存储的稀疏矩阵 A , B 非零元个数分别为 m 和 n 。试用类 PASCAL 语言编写时间复杂度为 $O(m+n)$ 的算法将矩阵 B 加到矩阵 A 上去。 A 的空间足够大，不另加辅助空间。要求描述所用结构。

【北京工业大学 1997 三 (10 分)】

3. 设整数 x_1, x_2, \dots, x_n 已存放在数组 A 中，编写一 PASCAL 递归过程，输出从这 n 个数中取出所有 k 个数的所有组合 ($k \leq n$)。例：若 A 中存放的数是 1, 2, 3, 4, 5, k 为 3, 则输出结果应为：543, 542, 541, 532, 531, 521, 432, 431, 421, 321。【山东大学 1992 五 (13 分)】

类似本题的另外叙述有：

(1) 题目基本相同，只是叙述不同，要求用 PASCAL 语言。【东南大学 2001 三 (10 分)】

4. 编写一个过程，对一个 $n \times n$ 矩阵，通过行变换，使其每行元素的平均值按递增顺序排列。

【中科院软件所 1996】

5. 设原来将 N 个自然数 $1, 2, \dots, N$ 按某个顺序存于数组 A 中，经过下面的语句计算，使 $A[I]$ 的值变为 $A[1]$ 到 $A[I-1]$ 中小于原 $A[I]$ 值的个数。

```
FOR I:=N DOWNT0 1 DO
BEGIN
  C:=0;
  FOR J:=1 TO I-1 DO
    IF A[J]<A[I] THEN C:=C+1;
  A[I]:=C
END;
```

编程将经过上述处理后的 A 还原成原来的 A 。【上海大学 1996 二 (16 分)】

6. 请编写完整的程序。如果矩阵 A 中存在这样的一个元素 $A[i, j]$ 满足条件： $A[i, j]$ 是第 i 行中值最小的元素，且又是第 j 列中值最大的元素，则称之为该矩阵的一个马鞍点。请编程计算出 $m \times n$ 的矩阵 A 的所有马鞍点。【上海大学 2000 三 (20 分)】【中科院自动化所 1997】

7. 给定一个整数数组 $b[0..N-1]$ ， b 中连续的相等元素构成的子序列称为平台。试设计算法，求出 b 中最长平台的长度。【中科院计算所 1999 五、2 (20 分)】

8. 给定 $n \times m$ 矩阵 $A[a..b, c..d]$ ，并设 $A[i, j] \leq A[i, j+1]$ ($a \leq i \leq b, c \leq j \leq d-1$) 和 $A[i, j] \leq$

$A[i+1, j]$ ($a \leq i \leq b-1, c \leq j \leq d$). 设计一算法判定 X 的值是否在 A 中, 要求时间复杂度为 $O(m+n)$ 。

【东南大学 2001 六(13 分)】

类似本题的另外叙述有:

(1) 给定整型数组 $B[0..m, 0..n]$ 。已知 B 中数据在每一维方向上都按从小到大的次序排列，且整型变量 x 在 B 中存在。试设计一个程序段找出一对满足 $B[i, j]=x$ 的 (i, j) 值，要求比较次数不超过 $m+n$ 。

【清华大学 1998 六 (10 分)】

(2) 给定 $n \times m$ 矩阵 $A[a..b, c..d]$, 并设 $A[i, j] \leq A[i, j+1]$ ($a \leq i \leq b, c \leq j \leq d-1$) 知 $A[i, j] \leq A[i+1, j]$, ($a \leq i \leq b-1, c \leq j \leq d$)。设计一算法以比 $O(n*m)$ 小的最坏时间复杂性判定值 x 是否在 A 中。【东南大学 1994 三 (17 分)】

9. 编写算法, 将自然数 $1 \sim n^2$ 按“蛇形”填入 $n \times n$ 矩阵中。例 ($1 \sim 4^2$) 如图所示: (用程序实现)

1	3	4	10
---	---	---	----

1	3	4	10
2	5	9	11
6	8	12	15
7	13	14	16

【南京航空航天大学 1997 八 (12 分)】 【中科院计算所 1996】

10. 设二维数组 $a[1..m, 1..n]$ 含有 $m*n$ 个整数。

(1) 写出算法(pascal 过程或 c 函数): 判断 a 中所有元素是否互不相同?输出相关信息(yes/no):

(2) 试分析算法的时间复杂度。【华中理工大学 1999 五 (10 分)】

11. 二项式 $(a+b)^n$ 展开式的系数为

$$C(n, 0) = 1, C(n, n) = 1, \text{ 对于 } n \geq 0$$

$C(n, k) = C(n-1, k) + C(n-1, k-1)$, 对于 $0 < k < n$ 形成著名的杨辉三角形, 如图所示。

(1) 试写一个递归算法，根据以上公式生成 $C(n, k)$ 。(6 分)

(2) 试画出计算 $C(6, 4)$ 的递归树。(4 分)

(3) 试写一个非递归算法，既不用数组也不用栈，对于任意的 $0 \leq k \leq n$ 计算 $C(n, k)$ (6 分)

				1					n=0
				1		1			n=1
			1	2	1				n=2
		1	3	3	1				n=3
	1	4	6	4	1				n=4
	1	5	10	10	5	1			n=5
	1	6	15	20	15	6	1		n=6
	1	7	21	35	35	21	7	1	n=7
1	8	28	56	70	56	28	8	1	n=8

【清华大学 1999 五 (16 分)】

12. 设任意 n 个整数存放于数组 $A(1:n)$ 中, 试编写程序, 将所有正数排在所有负数前面 (要求算法复杂性为 $O(n)$)。【山东大学 1993 三 (12 分)】.

类似本题的另外叙述有:

(1) 已知数组 $A[1..n]$ 的元素类型为整型, 设计算法调整 A , 使其左边的所有元素小于零, 右边的所有元素大于等于零。(要求算法的时间复杂度和空间复杂度均为 $O(n)$)

【北京理工大学 2000 四、1 (4 分)】

(2) 设计一个算法, 把整数数组中所有的偶数放到所有的奇数之前。要求时间、空间效率尽可能高。

【华南师范大学 1999 六、1 (10 分)】

(3) 设一系列正整数存放在一个数组中, 试设计算法, 将所有奇数存放在数组的前半部分, 将所有的偶数存放在数组的后半部分。要求尽可能少用临时存储单元并使时间最少。请试着分析你实现的算法的时间复杂度和空间复杂度。 【南开大学 2000 三、2】

(4) 设计算法将数组 $A[1..n]$ 调整为左右两部分, 使的左边所有的元素小于右边的所有元素, 并给出这一划分的分界位置。要求算法的时间复杂度为 $O(n)$ 。 【合肥工业大学 2001 五、3 (8 分)】

13. 若 S 是 n 个元素的集合, 则 S 的幂集 $P(S)$ 定义为 S 所有子集的集合。例如, $S = \{a, b, c\}$, $P(S) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$ 给定 S , 写一递归算法求 $P(S)$ 。

【东南大学 1993 五 (15 分)】【东南大学 1997 五 (15 分)】

14. 编写算法打印出由指针 Hm 指向总表头的以十字链表形式存储的稀疏矩阵中每一行的非零元的个数。注意: 行、列及总表头结点的形式为:

row	col	val
down	right	

它们已用 val 域链接成循环链表。非零元的结点形式也同上, 每一行(列)的非零元由 $right(down)$ 域把它们链接成循环链表, 该行(列)的表头结点即为该行(列)循环链表的表头。

【上海大学 1998 五 (16 分)】

15. 试编写建立广义表存储结构的算法, 要求在输入广义表的同时实现判断、建立。设广义表按如下形式输入 $(a_1, a_2, a_3, \dots, a_n)$ $n \geq 0$, 其中 a_i 或为单字母表示的原子或为广义表, $n=0$ 时为只含空格字符的空表。(注: 算法可用类 pascal 或类 c 书写) 【北京工业大学 1998 十 (15 分)】

16. 广义表是 $n(n \geq 0)$ 个数据元素 $a_1, a_2, a_3, \dots, a_n$ 的有限序列。其中 a_i ($1 \leq i \leq n$) 或者是单个数据元素(原子), 或仍然是一个广义表。广义表的结点具有不同的结构, 即原子结点和子表元素结点, 为了将两者统一, 用了一个标志 tag , 当其为 0 时表示是原子结点, 其 $data$ 域存储结点值, $link$ 域指向下一个结点, 当其 tag 为 1 时表示是子表结点, 其 $sublist$ 为指向子表的指针。因此, 广义表可采用如下结构存储:

```

TYPE glist = ^gnode;
gnode = RECORD
    link: glist;
    CASE tag: 0..1 OF
        0: (data: char);
        1: (sublist: glist)
    END;

```

(1) 画出广义表 $((a, b), c)$ 的存储结构;

(2) 写出计算一个广义表的原子结点个数的递归算法表示式;

(3) 编写实现上述算法的过程或函数程序。【厦门大学 2002 三 (12 分)】

17. 广义表 $GL = (a_1, a_2, \dots, a_n)$, 其中 a_k ($k=1, 2, \dots, n$) 或是单个数据元素(原子), 或仍然是个

广义表。给定如下有关广义表的类型定义：

```

TYPE tagtype=0..1;
glist=^gnode;
gnode=RECORD
    link:glist;           {link 域指向下一个结点}
CASE tag:tagtype OF      {tag=0 表示原子结点}
    0: (data :integer);
    1:(sublist: glist)
END;
```

编写一个过程或函数计算一个广义表的所有原子结点数据域之和，例如对广义表 (3, (2, 4, 5), (6, 3)) 数据域之和为 23。【厦门大学 2000 四、2 (9 分)】

18. 数组 $H[1:1000]$ 中存放着 1000 个大小不同的正整数；

(1) 选择一分类算法使能最快地得到其中 10 个最大的数，简要说明理由；

(2) 编写一程序 seek()，执行该程序时，在命令行中提供二个参数：

seek a n<enter> 表示需打印 $H[]$ 中 n 个最大数。

seek I n<enter> 表示需打印 $H[]$ 中 n 个最小数。【浙江大学 1994 八 (18 分)】

19. 已知两个定长数组，它们分别存放两个非降序有序序列，请编写程序把第二个数组序列中的数逐个插入到前一个数组序列中，完成后两个数组中的数分别有序（非降序）并且第一数组中所有的数都不大于第二个数组中的任意一个数。注意，不能另开辟数组，也不能对任意一个数组进行排序操作。例如，

第一个数组为：4, 12, 28

第二个数组为：1, 7, 9, 29, 45

输出结果为：1, 4, 7-----第一个数组

9, 12, 28, 29, 45-----第二个数组 【上海大学 1998 四 (20 分)】

20. 设数组 $A[1..n]$ 中， $A[n-2k+1..n-k]$ 和 $[n-k+1..n]$ 中元素各自从小到大排好序，试设计一个算法使 $A[n-2k+1..n]$ 按从小到大次序排好序。并分析算法所需的计算时间。【福州大学 1998 四、3 (10 分)】

21. 设 $A[1..100]$ 是一个记录构成的数组， $B[1..100]$ 是一个整数数组，其值介于 1 至 100 之间，现要求按 $B[1..100]$ 的内容调整 A 中记录的次序，比如当 $B[1]=11$ 时，则要求将 $A[1]$ 的内容调整到 $A[11]$ 中去。规定可使用的附加空间为 $O(1)$ 。【中科院计算所 2000 七 (15 分)】

22. 给定有 m 个整数的递增有序数组 $a[1..m]$ 和有 n 个整数的递减有序数组 $b[1..n]$ ，试写出算法：将数组 a 和 b 归并为递增有序数组 $c[1..m+n]$ 。(要求：算法的时间复杂度为 $O(m+n)$)

【华中理工大学 2000 八、1 (10 分)】

23. 在数组 $A[1..n]$ 中有 n 个数据，试建立一个带有头结点的循环链表，头指针为 h ，要求链中数据从小到大排列，重复的数据在链中只保存一个。【南京理工大学 1998 七、2 (7 分)】

第五章 数组和广义表

一、选择题

1. B	2. 1L	2. 2J	2. 3C	2. 4I	2. 5C	3. B	4. B	5. A	6. 1H	6. 2C	6. 3E
6. 4A	6. 5F	7. B	8. 1E	8. 2A	8. 3B	9. B	10. B	11. B	12. B	13. A	14. B
15. B	16. A	17. C	18. D	19. C	20. D	21. F	22. C	23. D	24. C	25. A	26. C
27. A											

二、判断题

1. ×	2. √	3. √	4. ×	5. ×	6. ×	7. √	8. ×	9. ×	10. ×	11. ×	12. √
13. √	14. √										

部分答案解释如下。

- 错误。对于完全二叉树，用一维数组作存储结构是效率高的（存储密度大）。
- 错误。数组是具有相同性质的数据元素的集合，数据元素不仅有值，还有下标。因此，可以说数组是元素值和下标构成的偶对的有穷集合。
- 错误。数组在维数和界偶确定后，其元素个数已经确定，不能进行插入和删除运算。
- 错误。稀疏矩阵转置后，除行列下标及行列数互换外，还必须确定该元素转置后在新三元组中的位置。
- 错误。广义表的取表尾运算，是非空广义表除去表头元素，剩余元素组成的表，不可能是原子。
- 错误。广义表的表头就是广义表的第一个元素。只有非空广义表才能取表头。
- 错误。广义表中元素可以是原子，也可以是表（包括空表和非空表）。
- 错误。广义表的表尾，指去掉表头元素后，剩余元素所组成的表。

三、填空题

- 顺序存储结构
- (1) 9572 (2) 1228
- (1) 9174 (2) 8788
- 1100
- 1164 公式： $LOC(a_{ijk}) = LOC(a_{000}) + [v_2 * v_3 * (i - c_1) + v_3 * (j - c_2) + (k - c_3)] * 1$ (1 为每个元素所占单元数)
- 232
- 1340
- 1196
- 第 1 行第 3 列
- (1) 270 (2) 27 (3) 2204
- $i(i-1)/2 + j$ ($1 \leq i, j \leq n$)
- (1) $n(n+1)/2$ (2) $i(i+1)/2$ (或 $j(j+1)/2$) (3) $i(i-1)/2 + j$ (4) $j(j-1)/2 + i$ ($1 \leq i, j \leq n$)
- 1038 三对角矩阵按行存储： $k = 2(i-1) + j$ ($1 \leq i, j \leq n$)
- 33 ($k = i(i-1)/2 + j$) ($1 \leq i, j \leq n$)
- 非零元很少 ($t \ll m * n$) 且分布没有规律
- 节省存储空间。
- 上三角矩阵中，主对角线上第 r ($1 \leq r \leq n$) 行有 $n-r+1$ 个元素， a_{ij} 所在行的元素数是 $j-i+1$ 。

所以，元素在一维数组的下标 k 和二维数组下标关系： $k=((i-1)*(2n-i+2))/2+(j-i+1)=(i-1)(2n-i)/2+j$ ($i \leq j$)

18. 93 19. $i(i-1)/2+j$ 20. 线性表 21. 其余元素组成的表

22. (1) 原子(单元素)是结构上不可再分的，可以是一个数或一个结构；而表带结构，本质就是广义表，因作为广义表的元素故称为子表。

(2) 大写字母 (3) 小写字母 (4) 表中元素的个数 (5) 表展开后所含括号的层数

23. 深度 24. (1) () (2) (()) (3) 2 (4) 2

25. head(head(tail(tail(head(tail(tail(A)))))))

26. 表展开后所含括号的层数 27. (1) 5 (2) 3

28. head(head(tail(LS))) 29. head(tail(tail(head(tail(head(A))))))

30. head(tail(head(tail(H)))) 31. (b) 32. (x, y, z) 33. (d, e)

34. GetHead(GetHead(GetTail(L)))

35. 本算法中，首先数组 b 中元素以逆置顺序放入 d 数组中，然后数组 a 和数组 d 的元素比较，将大者拷贝到数组 c 。第一个 WHILE 循环到数组 a 或数组 d 结尾，第二个和第三个 WHILE 语句只能执行其中的一个。

(1) $b[m-i+1]$ (2) $x:=a[i]$ (3) $i:=i+1$ (4) $x:=d[j]$ (5) $j:=j+1$ (6) $k:=k+1$ (7) $i \leq 1$ (8) $j \leq m$

36. (1) $(i==k)$ return (2) $i+1$ (3) $i-1$ (4) $i!=k$

本算法利用快速排序思想，找到第 k 个元素的位置(下标 $k-1$ 因而开初有 $k--$)。内层 do 循环以 $t(t=a[low])$ 为“枢轴”找到其应在 i 位置。这时若 i 等于 k ，则算法结束。(即第一个空格处 $if(i==k)$ return)。否则，若 $i < k$ ，就在 $i+1$ 至 $high$ 中间去查；若 $i > k$ ，则在 low 到 $i-1$ 间去找，直到找到 $i=k$ 为止。

37. 逆置广义表的递归模型如下

$$f(LS) = \begin{cases} \text{null} & \text{若 } LS \text{ 为空} \\ LS & \text{若 } LS \text{ 为原子，且 } tail(LS) \text{ 为空} \\ \text{append}(f(tail(LS)), head(LS)) & \text{若 } LS \rightarrow tag = 0, \text{ 且 } LS \rightarrow val.ptr.tp \neq \text{null} \\ \text{append}(f(tail(LS)), f(head(LS))) & \text{若 } LS \rightarrow tag = 1 \end{cases}$$

上面 appEND(a, b) 功能是将广义表 a 和 b 作为元素的广义表连接起来。

- (1) $(p \rightarrow tag == 0)$ //处理原子
- (2) $h = reverse(p \rightarrow val.ptr.hp)$ //处理表头
- (3) $(p \rightarrow val.ptr.tp)$ //产生表尾的逆置广义表
- (4) $s \rightarrow val.ptr.tp = t$; //连接
- (5) $q \rightarrow val.ptr.hp = h$ //头结点指向广义表

38. 本题要求将 $1, 2, \dots, n*n$ 个自然数，按蛇型方式存放在二维数组 $A[n][n]$ 中。“蛇型”方式，即是按“副对角线”平行的各对角线，从左下到右上，再从右上到左下，存放 n^2 个整数。对角线共 $2n-1$ 条，在副对角线上方的对角线，题目中用 k 表示第 k 条对角线(最左上角 $k=1$)，数组元素 x 和 y 方向坐标之和为 $k+1$ (即题目中的 $i+j=k+1$)。副对角线下方第 k 条对角线与第 $2n-k$ 条对角线对称，其元素的下标等于其对称元素的相应坐标各加 $(k-n)$ 。

- (1) $k \leq 2*n-1$ //共填 $2*n-1$ 条对角线
- (2) $q=2*n-k$ //副对角线以下的各条对角线上的元素数
- (3) $k \% 2 \neq 0$ // k 为偶数时从右上到左下，否则从左下向右上填数。(本处计算下标 i 和 j)

(4) $k \geq n$ //修改副对角线下方的下标 i 和 j 。

(5) $m++$; 或 $m=m+1$ //为填下个数据作准备, m 变化范围 $1..n*n$ 。

本题解法的另一种思路见本章算法设计题第 9 题。

39. 本题难点有二: 一是如何求下一出圈人的位置, 二是某人出圈后对该人的位置如何处理。

按题中要求, 从第 s 个人开始报数, 报到第 m 个人, 此人出圈。 n 个人围成一圈, 可看作环状, 则下个出圈人, 其位置是 $(s+m-1) \% n$ 。 n 是人数, 是个变量, 出圈一人减 1, 算法中用 i 表示。对第二个问题, 算法中用出圈人后面人的位置依次前移, 并把出圈人的位置 (下标) 存放到当时最后一个人的位置 (下标)。算法最后打印出圈人的顺序。

(1) $(s+m-1) \% i$ //计算出圈人 $s1$

(2) $s1:=i$ //若 $s1=0$, 说明是第 i 个人出圈 ($i \% i=0$)

(3) $s1 \rightarrow i-1$ //从 $s1$ 到 i 依次前移, 使人数减 1, 并将出圈人放到当前最后一个位置 $A[i]=w$ 。

40. 若第 n 件物品能放入背包, 则问题变为能否再从 $n-1$ 件物品中选出若干件放入背包 (这时背包可放入物品的重量变为 $s-w[n]$)。若第 n 件物品不能放入背包, 则考虑从 $n-1$ 件物品选若干件放入背包 (这时背包可放入物品仍为 s)。若最终 $s=0$, 则有一解; 否则, 若 $s < 0$ 或虽然 $s > 0$ 但物品数 $n < 1$, 则无解。

(1) $s-w[n], n-1$ //Knap($s-w[n], n-1$)=true

(2) $s, n-1$ // Knap \leftarrow Knap($s, n-1$)

四、应用题

1、958 三维数组以行为主序存储, 其元素地址公式为:

$$LOC(A_{ijk}) = LOC(A_{1c_1c_3}) + [(i-c_1)V_2V_3 + (j-c_2)V_3 + (k-c_3)] * L + 1$$

其中 c_i, d_i 是各维的下界和上界, $V_i = d_i - c_i + 1$ 是各维元素个数, L 是一个元素所占的存储单元数。

2. b 对角矩阵的 b 条对角线, 在主对角线上方和下方各有 $\lfloor b/2 \rfloor$ 条对角线 (为叙述方便, 下面设 $a = \lfloor b/2 \rfloor$)。从第 1 行至第 a 行, 每行上的元素数依次是 $a+1, a+2, \dots, b-2, b-1$, 最后的 a 行上的元素个数是 $b-1, b-2, \dots, a+1$ 。中间的 $(n-2a)$ 行, 每行元素个数都是 b 。故 b 条对角线上元素个数为 $(n-2a)b + a*(a+b)$ 。存放在一维数组 $V[1..nb-a(b-a)]$ 中, 其下标 k 与元素在二维数组中下标 i 和 j 的关系为:

$$k = \begin{cases} \frac{(i-1)(i+2a)}{2} + j & \text{当 } 1 \leq i \leq a+1 \\ a(2i-a) + j & \text{当 } a+1 < i \leq n-a+1 \\ a(2i-a) - \frac{(i-n+a)(i-n+a-1)}{2} + j & \text{当 } n-a+1 < i \leq n \end{cases}$$

3. 每个元素 32 个二进制位, 主存字长 16 位, 故每个元素占 2 个字长, 行下标可平移至 1 到 11。

(1) 242 (2) 22 (3) $s+182$ (4) $s+142$

4. 1784 (公式: $Loc(A_{ijk1}) = 100$ (基地址) $+ [(i-c_1)v_2v_3v_4 + (j-c_2)v_3v_4 + (k-c_3)v_4 + (1-c_4)] * 4$

5. $1210+108L$ ($LOC(A[1, 3, -2]) = 1210 + [(k-c_3)v_2v_1 + (j-c_2)v_1 + (i-c_1)] * L$ (设每个元素占 L 个存储单元)

6. 数组占的存储字节数 $= 10 * 9 * 7 * 4 = 2520$; $A[5, 0, 7]$ 的存储地址 $= 100 + [4 * 9 * 7 + 2 * 7 + 5] * 4 = 1184$

7. 五对角矩阵按行存储, 元素在一维数组中下标 (从 1 开始) k 与 i, j 的关系如下:

$$k = \begin{cases} 4(i-1) + j & (\text{当 } i = 1 \text{ 时}) \\ 4(i-1) + j - 1 & (\text{当 } 1 < i < n \text{ 时}) \\ 4(i-1) + j - 2 & (\text{当 } i = n \text{ 时}) \end{cases}$$

$A[15, 16]$ 是第 71 个元素, 在向量 $[-10:m]$ 中的存储位置是 60。

8. (1) 540 (2) 108 (3) $i=3, j=10$, 即 $A[3, 10]$ 9. $k=i(i-1)/2+j$
10. 稀疏矩阵 A 有 t 个非零元素, 加上行数 m_u 、列数 n_u 和非零元素个数 t_u (也算一个三元组), 共占用三元组表 LTMA 的 $3(t+1)$ 个存储单元, 用二维数组存储时占用 $m*n$ 个单元, 只有当 $3(t+1) < m*n$ 时, 用 LTMA 表示 A 才有意义。解不等式得 $t < m*n/3 - 1$ 。
11. 参见 10。
12. 题中矩阵非零元素用三元组表存储, 查找某非零元素时, 按常规要从第一个元素开始查找, 属于顺序查找, 时间复杂度为 $O(n)$ 。若使查找时间得到改善, 可以建立索引, 将各行行号及各行第一个非零元素在数组 B 中的位置 (下标) 偶对放入一向量 C 中。若查找非零元素, 可先在数组 C 中用折半查找到该非零元素的行号, 并取出该行第一个非零元素在 B 中的位置, 再到 B 中顺序 (或折半) 查找该元素, 这时时间复杂度为 $O(\log n)$ 。
13. (1) 176 (2) 76 和 108 (3) 28 和 116。
14. (1) $k = 3(i-1)$ (主对角线左下角, 即 $i=j+1$)
 $k = 3(i-1)+1$ (主对角线上, 即 $i=j$)
 $k = 3(i-1)+2$ (主对角线上, 即 $i=j-1$)
 由以上三式, 得 $k=2(i-1)+j$ ($1 \leq i, j \leq n; 1 \leq k \leq 3n-2$)
 (2) $10^3 * 10^3 - (3 * 10^3 - 2)$
15. 稀疏矩阵 A 采用二维数组存储时, 需要 $n*n$ 个存储单元, 完成求 $\sum a_{ii} (1 \leq i \leq n)$ 时, 由于 $a[i][i]$ 随机存取, 速度快。但采用三元组表时, 若非零元素个数为 t , 需 $3(t+1)$ 个存储单元 (第一个分量中存稀疏矩阵 A 的行数, 列数和非零元素个数, 以后 t 个分量存各非零元素的行值、列值、元素值), 比二维数组节省存储单元; 但在求 $\sum a_{ii} (1 \leq i \leq n)$ 时, 要扫描整个三元组表, 以便找到行列值相等的非零元素求和, 其时间性能比采用二维数组时差。
16. 特殊矩阵指值相同的元素或零元素在矩阵中的分布有一定规律, 因此可以对非零元素分配单元 (对值相同元素只分配一个单元), 将非零元素存储在向量中, 元素的下标 i 和 j 和该元素在向量中的下标有一定规律, 可以用简单公式表示, 仍具有随机存取功能。而稀疏矩阵是指非零元素和矩阵容量相比很小 ($t \ll m*n$), 且分布没有规律。用十字链表作存储结构自然失去了随机存取的功能。即使用三元组表的顺序存储结构, 存取下标为 i 和 j 的元素时, 要扫描三元组表, 下标不同的元素, 存取时间也不同, 最好情况下存取时间为 $O(1)$, 最差情况下是 $O(n)$, 因此也失去了随机存取的功能。
17. 一维数组属于特殊的顺序表, 和有序表的差别主要在于有序表中元素按值排序 (非递增或非递减), 而一维数组中元素没有按元素值排列顺序的要求。
18. $n(n+1)/2$ (压缩存储) 或 n^2 (不采用压缩存储)
19. $LOC(A[i, j]) = LOC(A[3, 2]) + [(i-3)*5 + (j-2)] \times 2$ (按行存放)
 $LOC(A[i, j]) = LOC(A[3, 2]) + [(j-2)*6 + (i-3)] \times 2$ (按列存放)
20. n 阶下三角矩阵元素 $A[i][j] (1 \leq i, j \leq n, i \geq j)$ 。第 1 列有 n 个元素, 第 j 列有 $n-j+1$ 个元素, 第 1 列到第 $j-1$ 列是等腰梯形, 元素数为 $(n+(n-j+2))(j-1)/2$, 而 a_{ij} 在第 j 列上的位置是为 $i-j+1$ 。所以 n 阶下三角矩阵 A 按列存储, 其元素 a_{ij} 在一维数组 B 中的存储位置 k 与 i 和 j 的关系为:
 $k = (n+(n-(j-1)+1))(j-1)/2 + (i-j+1) = (2n-j)(j-1)/2 + i$
21. 三对角矩阵第一行和最后一行各有两个非零元素, 其余每行均有三个非零元素, 所以共有

$3n-2$ 个元素。

(1) 主对角线左下对角线上的元素下标间有 $i=j+1$ 关系, k 与 i 和 j 的关系为 $k=3(i-1)$; 主对角线上元素下标间有关系 $i=j$, k 与 i 和 j 的关系为 $k=3(i-1)+1$; 主对角线右上那条对角线上元素下标间有关系 $i=j-1$, k 与 i 和 j 的关系为 $k=3(i-1)+2$ 。综合以上三等式, 有 $k=2(i-1)+j$ ($1 \leq i, j \leq n, |i-j| \leq 1$)

(2) $i=k/3+1$; ($1 \leq k \leq 3n-2$) // $k/3$ 取小于 $k/3$ 的最大整数。下同

$j=k-2(i-1)=k-2(k/3)=k\%3+k/3$

22. 这是一个递归调用问题, 运行结果为: DBHEAIFJCKGL

23. (1) FOR 循环中, 每次执行 PerfectShuffle(A, N) 和 CompareExchange(A, N) 的结果:

第 1 次: $A[1..8]=[90, 30, 85, 65, 50, 80, 10, 100]$

$A[1..8]=[30, 90, 65, 85, 50, 80, 10, 100]$

第 2 次: $A[1..8]=[30, 50, 90, 80, 65, 10, 85, 100]$

$A[1..8]=[30, 50, 80, 90, 10, 65, 85, 100]$

第 3 次: $A[1..8]=[30, 10, 50, 65, 80, 85, 90, 100]$

$A[1..8]=[10, 30, 50, 65, 80, 85, 90, 100]$

(2) Demo 的功能是将数组 A 中元素按递增顺序排序。

(3) PerfectShuffle 中 WHILE 循环内是赋值语句, 共 $2N$ 次, WHILE 外成组赋值语句, 相当 $2N$ 个简单赋值语句; CompareExchange 中 WHILE 循环内是交换语句, 最好情况下不发生交换, 最差情况下发生 N 次交换, 相当于 $3N$ 个赋值语句; Demo 中 FOR 循环循环次数 $\log_2 2N$, 故按赋值语句次数计算 Demo 的时间复杂度为: 最好情况: $O(4N \cdot \log_2 2N) \approx O(N \log(2 \cdot N))$; 最差情况: $O((4N+3N) \cdot \log_2 2N) \approx O(N \log(2 \cdot N))$ 。

24. 这是一个排序程序。运行后 B 数组存放 A 数组各数在排序后的位置。结果是:

$A=\{121, 22, 323, 212, 636, 939, 828, 424, 55, 262\}$

$B=\{3, 1, 6, 4, 8, 10, 9, 7, 2, 5\}$

$C=\{22, 55, 121, 212, 262, 323, 424, 639, 828, 939\}$

25. (1) $c = \begin{bmatrix} 3 & 3 & 1 \\ 1 & 1 & 1 \\ 3 & 3 & 2 \end{bmatrix}$ (2) $a = \begin{bmatrix} 3 & 3 & 1 \\ 1 & 1 & 3 \\ 3 & 3 & 3 \end{bmatrix}$

26. (1) 同上面 26 题 (1)

(2) 对 c 数组的赋值同所选择的下标 i 和 j 的次序 (指外层循环用 j 内层用 i) 没有关系

(3) 同上题 26 (2)

(4) 对 i, j 下标取反序后, 重复执行第 (3) 步, A 数组所有元素均变为 2。(在机器上验证, 反复循环 3 次后, 所有元素均变为 2)

27. 错误有以下几处:

(1) 过程参数没有类型说明; (2) 出错条件判断: 缺少 OR ($i+k > \text{last}+1$);

(3) 删除元素时 FOR 循环应正向, 不应用反向 DOWNT0; (4) count 没定义;

低效体现在两处:

(1) 删除 k 个元素时, 不必一个一个元素前移, 而应一次前移 k 个位置;

(2) last 指针不应一次减 1, 而应最后一次减 k 。

正确的高效算法如下:

const m=64;

TYPE ARR=ARRAY[1..m] OF integer;

PROCEDURE delk (VAR A:ARR; VAR last:integer;i,k: integer);

{从数组 A[1..last]中删除第 i 个元素起的 k 个元素, m 为 A 的上限}

VAR count: integer;

BEGIN

IF (i<0) OR (i>last) OR (k<0) OR (last>m) OR (i+k>last+1)

THEN write (' error')

ELSE[FOR count:= i+k TO last DO A[count-k]:=A[count];

last:=last-k;]

END;

28. 这是计数排序程序。

(a) c[i] (1<=i<=n)中存放 A 数组中值为 i 的元素个数。

(b) c[i] (1<=i<=n)中存放 A 数组中小于等于 i 的个数。

(c) B 中存放排序结果, B[1..n]已经有序。

(d) 算法中有 4 个并列 for 循环语句, 算法的时间复杂度为 O(n)。

29. 上三角矩阵第一行有 n 个元素, 第 i-1 行有 n-(i-1)+1 个元素, 第一行到第 i-1 行是等腰梯形, 而第 i 行上第 j 个元素 (即 a_{ij}) 是第 i 行上第 j-i+1 个元素, 故元素 A_{ij} 在一维数组中的存储位置 (下标 k) 为:

$$k=(n+(n-(i-1)+1))(i-1)/2+(j-i+1)=(2n-i+2)(i-1)/2+j-i+1$$

30. 将上面 29 题的等式进一步整理为:

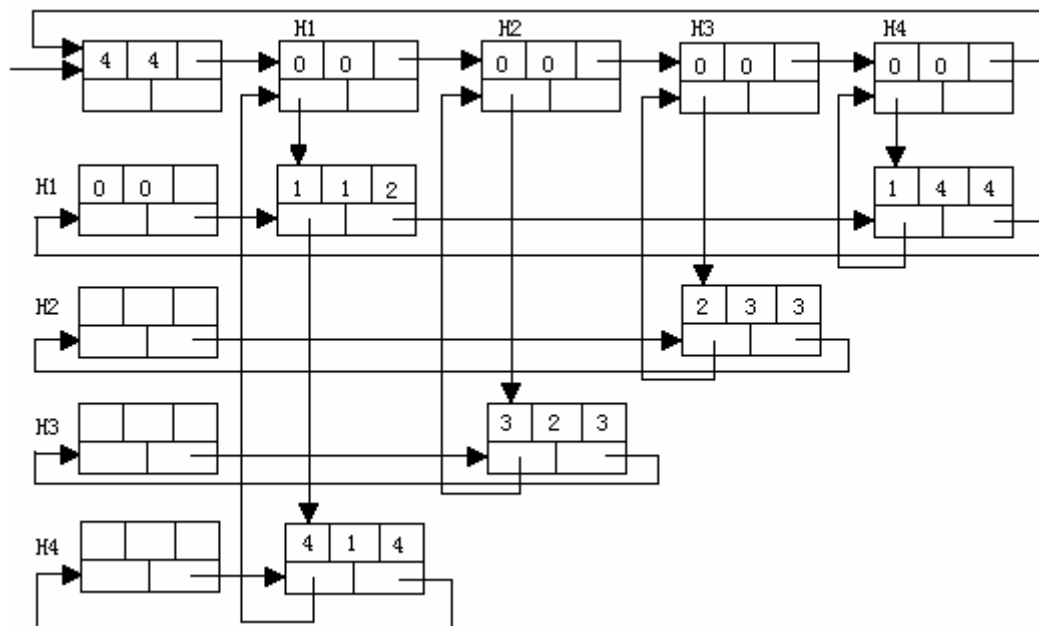
$$k=(n+1/2)i-i^2/2+j-n,$$

则得 $f_1(i)=(n+1/2)i-i^2/2$, $f_2(j)=j$, $c=-n$ 。

31. (1) 将对称矩阵对角线及以下元素按行序存入一维数组中, 结果如下:

	2	0	0	0	3	0	4	0	0	0
下标	1	2	3	4	5	6	7	8	9	10

(2) 因行列表头的“行列域”值用了 0 和 0, 下面十字链表中行和列下标均从 1 开始。



注: 上侧列表头 H_i 和左侧行表头 H_i 是一个 (即 H_1 、 H_2 、 H_3 和 H_4), 为了清楚, 画成了两个。

32. (1) $k=(2n-j+2)(j-1)/2+i-j+1$ (当 $i \geq j$ 时, 本题 $n=4$)

$k=(2n-i+2)(i-1)/2+j-i+1$ (当 $i < j$ 时, 本题 $n=4$)

(2) 稀疏矩阵的三元组表为：
 $s=((4, 4, 6), (1, 1, 1), (1, 4, 2), (2, 2, 3), (3, 4, 5), (4, 1, 2), (4, 3, 5))$ 。其中第一个三元组是稀疏矩阵行数、列数和非零元素个数。其它三元组均为非零元素行值、列值和元素值。

33. (1) $k=2(i-1)+j$ ($1 \leq i, j \leq n, |i-j| \leq 1$)

$i=\text{floor}(k/3)+1$ // $\text{floor}(a)$ 是取小于等于 a 的最大整数

$j=k-2(i-1)$

推导过程见上面第 25 题。

(2) 行逻辑链接顺序表是稀疏矩阵压缩存储的一种形式。为了随机存取任意一行的非零元，需要知道每一行第一个非零元在三元组表中的位置。为此，除非零元的三元组表外，还需要一个向量，其元素值是每行第一个非零元在三元组表中的位置。其类型定义如下：

typedef struct

{ **int** mu, nu, tu; //稀疏矩阵的行数、列数和非零元素个数

int rpos[maxrow+1]; //行向量，其元素值是每行第一个非零元在三元组表中的位置。

Triple data[maxsize];

}SparsMatrix;

因篇幅所限，不再画出行逻辑链接顺序表。

34. 各维的元素数为 $d_i - c_i + 1$ ，则 $a[i_1, i_2, i_3]$ 的地址为：

$a_0 + [(i_1 - c_1)(d_3 - c_3 + 1)(d_2 - c_2 + 1) + (i_2 - c_2)(d_3 - c_3 + 1) + (i_3 - c_3)] * L$

35. 主对角线上元素的坐标是 $i=j$ ，副对角线上元素的坐标 i 和 j 有 $i+j=n+1$ 的关系

(1) $i=j$ 或 $i=n+1-j$ ($1 \leq i, j \leq n$)

(2) 非零元素分布在两条主、副对角线上，除对角线相交处一个元素（下称“中心元素”）外，其余每行都有两个元素。主对角线上的元素，在向量 B 中存储的下标是 $k=2i-1$ ($i=j, 1 \leq i, j \leq n, 1 \leq k \leq 2n-1$)。

副对角线上的元素，在中心元素前，在向量 B 中存储的下标是 $k=2i$ ($i < j, 1 \leq i, j \leq n/2$)；在中心元素后，其下标是 $k=2(i-1)$ ($i < j, n/2+1 \leq i, j \leq n, 1 \leq k \leq 2n-1$)。

(3) a_{ij} 在 B 中的位置 $k = \begin{cases} A_0 + 2(i-1) & (i=j, 1 \leq i, j \leq n) \\ A_0 + 2(i-1) + 1 & (i < j, 1 \leq i, j \leq n/2) \\ A_0 + 2(i-1) - 1 & (i < j, n/2+1 \leq i, j \leq n) \end{cases}$

36. 由于对称矩阵采用压缩存储，上三角矩阵第一列一个元素，第二列两个元素，第 j 列 j 个元素。上三角矩阵共有 $n(n+1)/2$ 个元素。我们将这些元素存储到一个向量 $B[n(n+1)/2+1]$ 中。可以看到 $B[k]$ 和矩阵中的元素 a_{ij} 之间存在着——对应关系：

$$k = \begin{cases} \frac{j(j-1)}{2} + i & \text{当 } i \leq j \\ \frac{i(i-1)}{2} + j & \text{当 } i > j \end{cases}$$

则其对应关系可表示为： $k = \frac{\text{MAX}(i, j) * (\text{MAX}(i, j) - 1)}{2} + \text{MIN}(i, j)$ ($1 \leq i, j \leq n$,

$1 \leq k \leq n(n+1)/2$)

```
int MAX(int x, int y)
{ return (x > y ? x : y);
```

```

    }
    int MIN(int x, int y)
    { return(x<y?x:y);
    }

```

37. 设用 mu, nu 和 tu 表示稀疏矩阵行数, 列数和非零元素个数, 则转置矩阵的行数, 列数和非零元素的个数分别是 nu, mu 和 tu 。转置可按转置矩阵的三元组表中的元素顺序进行, 即按稀疏矩阵的列序, 从第 1 列到第 nu 列, 每列中按行值递增顺序, 找出非零元素, 逐个放入转置矩阵的三元组表中, 转时行列值互换, 元素值复制。按这种方法, 第 1 列到第 1 个非零元素一定是转置后矩阵的三元组表中的第 1 个元素, 第 1 列非零元素在第 2 列非零元素的前面。这种方法时间复杂度是 $O(n \cdot P)$, 其中 p 是非零元素个数, 当 p 和 $m \cdot n$ 同量级时, 时间复杂度为 $O(n^3)$ 。

另一种转置方法称作快速转置, 使时间复杂度降为 $O(m \cdot n)$ 。它是按稀疏矩阵三元组表中元素的顺序进行。按顺序取出一个元素, 放到转置矩阵三元组表的相应位置。这就要求出每列非零元素个数和每列第一个非零元素在转置矩阵三元组表中的位置, 设置了两个附加向量。

38. 广义表中的元素, 可以是原子, 也可以是子表, 即广义表是原子或子表的有限序列, 满足线性结构的特性: 在非空线性结构中, 只有一个称为“第一个”的元素, 只有一个成为“最后一个”的元素, 第一元素有后继而没有前驱, 最后一个元素有前驱而没有后继, 其余每个元素有唯一前驱和唯一后继。从这个意义上说, 广义表属于线性结构。

39. 数组是具有相同性质的数据元素的集合, 同时每个元素又有唯一下标限定, 可以说数组是值和下标偶对的有限集合。 n 维数组中的每个元素, 处于 n 个关系之中, 每个关系都是线性的, 且 n 维数组可以看作其元素是 $n-1$ 维数组的一个线性表。而广义表与线性表的关系, 见上面 38 题的解释。

40. 线性表中的元素可以是各种各样的, 但必须具有相同性质, 属于同一数据对象。广义表中的元素可以是原子, 也可以是子表。其它请参见 38。

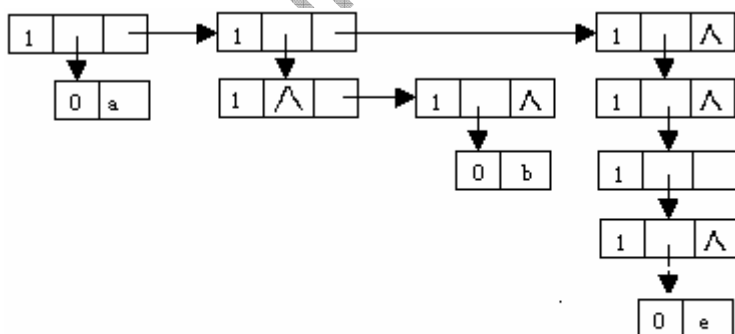
41. (1) (c, d) (2) (b) (3) b (4) (f) (5) $()$

42. $Head(Tail(Head(Head(L1))))$

$Head(Head(Head(Tail(Head(Tail(L2))))))$

类似本题的另外叙述的几个题解答如下:

- (1) $head(head(tail(tail(L))))$, 设 $L = (a, (c), b), (((e)))$
 (2) $head(head(head(head(tail(tail(L))))))$

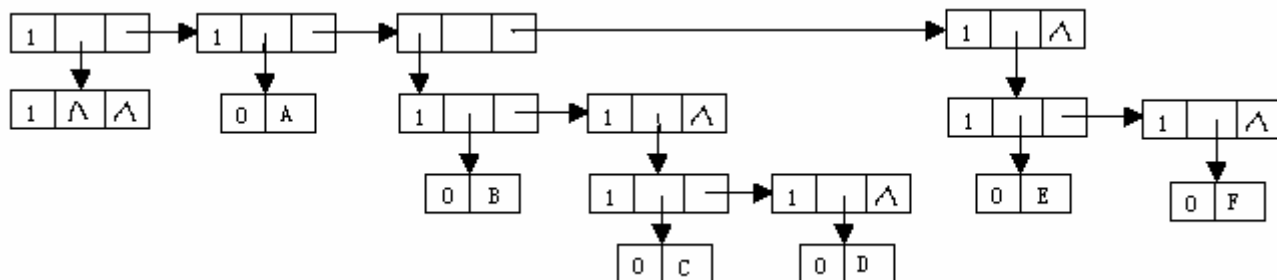


- (3) $head(tail(head(tail(A))))$
 (4) $H(H(T(H(T(H(T(L)))))))$
 (5) $tail(L) = (((c, d)), (e, f))$
 $head(tail(L)) = ((c, d))$

head (head (tail (L))) = (c, d)
tail (head (head (tail (L)))) = (d)
head (tail (head (head (tail (L)))) = d

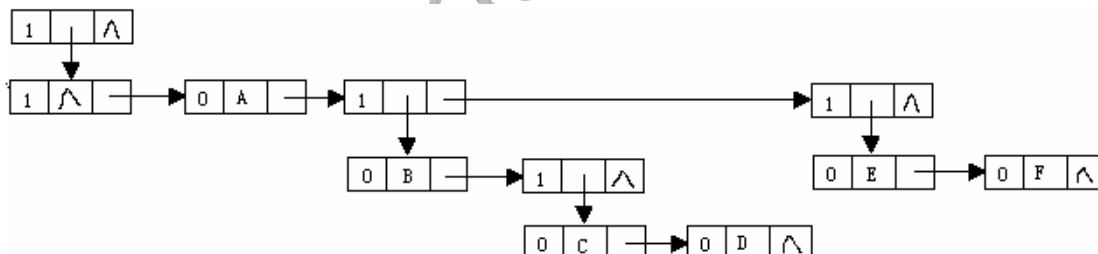
(6) head (tail (head (head (tail (tail (A))))))

43. 广义表的第一种存储结构的理论基础是，非空广义表可唯一分解成表头和表尾两部分，而由表头和表尾可唯一构成一个广义表。这种存储结构中，原子和表采用不同的结点结构（“异构”，即结点域个数不同）。



原子结点两个域：标志域 tag=0 表示原子结点，域 DATA 表示原子的值；子表结点三个域：tag=1 表示子表，hp 和 tp 分别是指向表头和表尾的指针。在画存储结构时，对非空广义表不断进行表头和表尾的分解，表头可以是原子，也可以是子表，而表尾一定是表（包括空表）。上面是本题的第一种存储结构图。

广义表的第二种存储结构的理论基础是，非空广义表最高层元素间具有逻辑关系：第一个元素无前驱有后继，最后一个元素无后继有前驱，其余元素有唯一前驱和唯一后继。有人将这种结构看作扩充线性结构。这种存储结构中，原子和表均采用三个域的结点结构（“同构”）。结点中都有一个指针域指向后继结点。原子结点中还包括标志域 tag=0 和原子值域 DATA；子表结点还包括标志域 tag=1 和指向子表的指针 hp。在画存储结构时，从左往右一个元素一个元素的画，直至最后一个元素。下面是本题的第二种存储结构图。



由于存储结构图占篇幅较大，下面这类题均不再解答。

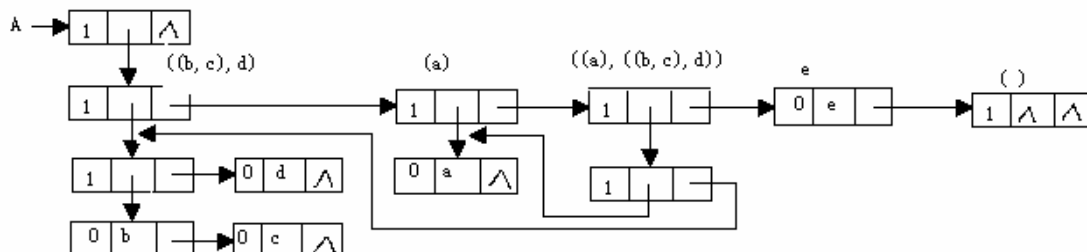
44. 深度为 5，长度为 2

45. (1) 略

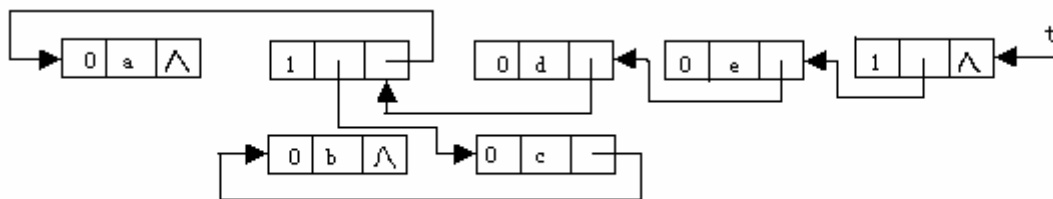
(2) 表的长度为 5，深度为 4

(3) head (tail (head (head (head (tail (tail (tail (tail (A))))))))))

46. 共享结构广义表 A = (((b, c), d), (a), ((a), ((b, c), d)), e, ()) 的存储表示：



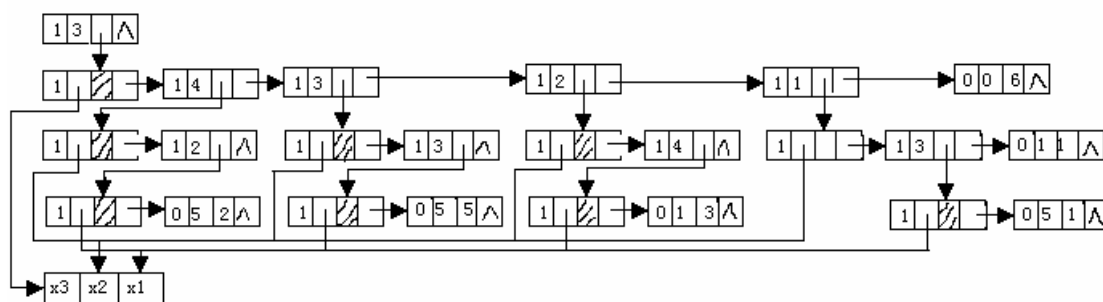
47. (1) 算法 A 的功能是逆置广义表 p (即广义表由 p 指针所指)。逆置后的广义表由 t 指向。
(2) 逆置后的广义表由 t 指向, 这时 p=nil。



48. (a, b) 49. (d)

50. 否。广义表的长度不是广义表中原子个数，而是指广义表中所含元素的个数，广义表中的元素可以是原子，也可以是子表。广义表元素多于 1 个时，元素间用逗号分开。

51. $p(x_1, x_2, x_3) = 2x_1^5x_2^2x_3^4 + 5x_1^5x_2^3x_3^3 + 3x_1x_2^4x_3^2 + (x_1^5x_2^3 + x_2)x_3 + 6$



52. (1) $H(A(a_1, a_2), B(b_1), C(c_1, c_2), x)$

$$\text{HEAD}(\text{TAIL}(\text{HEAD}(H))) = a_2$$

(2) 略

五. 算法设计题

1. [题目分析]本题是在向量 D 内插入元素问题。首先要查找插入位置, 数据 x 插入到第 i 个数据组的末尾, 即是第 i+1 个数据组的开始, 而第 i ($1 \leq i \leq n$) 个数据组的首地址由数组 s (即数组元素 s[i]) 给出。其次, 数据 x 插入后, 还要维护数组 s, 以保持空间区 D 和数组 s 的正确的相互关系。

```
void Insert (int s[], datatype D[], x, int i,m)
```

//在 m 个元素的 D 数据区的第 i 个数据组末尾，插入新数据 x，第 i 个数据组的首址由数组 s 给出。

```
{if (i<1 || i>n) {printf (“参数错误”); exit (0);}}
```

```
if (i==n) D[m]=x; // 在第 n 个数据组末尾插入元素。
```

```
else {for (j=m-1;j>=s[i+1];j--) D[j+1]=D[j]; // 第 i + 1 个数据组及以后元素后移
      D[s[i+1]]=x; // 将新数据 x 插入
```

```
for(j=i+1;j<=n;j++) s[j]++; // 维护空间区 D 和数组 s 的关系。
```

```
} //结束元素插入
```

```
m++; //空间区 D 的数据元素个数增 1。
```

```
}// 算法 Insert 结束
```

〔算法讨论〕数据在空间区从下标 0 开始, 最后一个元素的下标是 $m-1$ 。设空间区容量足够大, 未考虑空间溢出问题。数组 s 随机存数, 而向量 D 数据插入, 引起数组元素移动, 时间复杂度是 $O(n)$ 。

2. [题目分析] 设稀疏矩阵的非零元素的三元组以行序为主存储在三元组表中。矩阵的相加是对

应元素的相加。对两非零元素相加，若行号不等，则行号大者是结果矩阵中的非零元素。若行号相同，则列号大者是结果中一非零元素；若行号列号相同，若对应元素值之和为零，不予存储，否则，作为新三元组存到三元组表中。题目中要求时间复杂度为 $O(m+n)$ 。因此需从两个三元组表的最后一个元素开始相加。第一个非零元素放在 A 矩阵三元组表的第 $m+n$ 位置上。结果的三元组至多是 $m+n$ 个非零元素。最后若发生对应元素相加和为零的情况，对三元组表中元素要进行整理，以便使第一个三元组存放在下标 1 的位置上。

CONST maxnum=大于非零元素数的某个常量

TYPE tuple=RECORD

i, j: integer; v: elemtp;

END;

sparmattp=RECORD

mu, nu, tu: integer;

data: ARRAY[1..maxnum] OF tuple;

END;

PROC AddMatrix (VAR A: sparmattp; B: sparmattp);

// 稀疏矩阵 A 和 B 各有 m 和 n 个非零元素，以三元组表存储。A 的空间足够大，本算法实现两个稀疏矩阵相加，结果放到 A 中。

L:=m; p:=n; k:=m+n; // L, p 为 A, B 三元组表指针，k 为结果三元组表指针（下标）。

A.tu:=m+n; // 暂存结果矩阵非零元素个数

WHILE (L≥1) AND (p≥1) DO

[CASE // 行号不等时，行号大者的三元组为结果三元组表中一项。

A.data[L].i>B.data[p].i: A.data[k]:=A.data[L]; L:=L-1; // A 中当前项为结果项

A.data[L].i<B.data[p].i: A.data[k]:=B.data[p]; p:=p-1; //B 中当前项为结果当前项

A.data[L].i=B.data[p].i;

CASE //行号相等时，比较列号

A.data[L].j>B.data[p].j: A.data[k]:=A.data[L]; L:=L-1;

A.data[L].j<B.data[p].j: A.data[k]:=B.data[p]; p:=p-1;

A.data[L].j=B.data[p].j: IF A.data[L].v+B.data[p].v≠0 THEN
[A.data[L].v=A.data[L].v+

B.data[p].v;

A.data[k]:=A.data[L];]

L:=L-1; p:=p-1;

ENDC; //结束行号相等时的处理

ENDC; //结束行号比较处理。

k:=k-1; //结果三元组表的指针前移（减1）

]//结束 WHILE 循环。

WHILE p>0 DO[A.data[k]:=B.data[p]; k:=k-1; p:=p-1;] //处理 B 的剩余部分。

WHILE L>1 DO[A.data[k]:=A.data[L]; k:=k-1; L:=L-1;] //处理 A 的剩余部分。

IF k>1 THEN //稀疏矩阵相应元素相加时，有和为零的元素，因而元素总数<m+n。

[FOR p:=k TO m+n DO A[p-k+1]:=A[p]; // 三元组前移，使第一个三元组的下标为 1。

A.tu=m+n-k+1;]

// 修改结果三元组表中非零元素个数。

ENDP; // 结束 addmatrix

[算法讨论]算法中三元组的赋值是“成组赋值”，可用行值、列值和元素值的三个赋值句代

替。A 和 B 的三元组表的当前元素的指针 L 和 p，在每种情况处理后均有修改，而结果三元组表的指针 k 在 CASE 语句后统一处理 ($k:=k-1$)。算法在 B 的第一个元素“大于”A 的最后一个元素时，时间复杂度最佳为 $O(n)$ ，最差情况是每个元素都移动（赋值）了一次，且出现了和为零的元素，致使最后 $(m+n-k+1)$ 个元素向前平移一次，时间复杂度最差为 $O(m+n)$ 。

3. [题目分析] 从 n 个数中，取出所有 k 个数的所有组合。设数已存于数组 $A[1..n]$ 中。为使结果唯一，可以分别求出包括 $A[n]$ 和不包括 $A[n]$ 的所有组合。即包括 $A[n]$ 时，求出从 $A[1..n-1]$ 中取出 $k-1$ 个元素的所有组合，不包括 $A[n]$ 时，求出从 $A[1..n-1]$ 中取出 k 个元素的所有组合。

```
CONST n=10; k=3;
TYPE ARR=ARRAY[1..n] OF integer;
VAR A, B: ARR; // A 中存放 n 个自然数, B 中存放输出结果。
PROC outresult; //输出结果
    FOR j:=1 TO k DO write (B[j]); writeln;
ENDP;
PROC nkcombination (i, j, k: integer);
//从 i 个数中连续取出 k 个数的所有组合, i 个数已存入数组 A 中, j 为结果数组 B 中的下标。
    IF k=0 THEN outresult
    ELSE IF (i-k>=0) THEN [ B[j]:=A[i]; j:=j+1;
                           nkcombination (i-1, k-1, j);
                           nkcombination (i-1, k, j-1); ]
    ENDP;
```

[算法讨论] 本算法调用时， i 是数的个数（题目中的 n ）， $k \leq i$ ， j 是结果数组的下标。按题中例子，用 $nkcombination(5, 1, 3)$ 调用。若想按正序输出，如 123, 124, ...，可将条件表达式 $i-k \geq 0$ 改为 $i+k-1 \leq n$ ，其中 n 是数的个数， i 初始调用时为 1，两个调用语句中的 $i-1$ 均改为 $i+1$ 。

4. [题目分析] 题目中要求矩阵两行元素的平均值按递增顺序排序，由于每行元素个数相等，按平均值排列与按每行元素之和排列是一个意思。所以应先求出各行元素之和，放入一维数组中，然后选择一种排序方法，对该数组进行排序，注意在排序时若有元素移动，则与之相应的行中各元素也必须做相应变动。

```
void Translation (float *matrix, int n)
//本算法对  $n \times n$  的矩阵 matrix，通过行变换，使其各行元素的平均值按递增排列。
{int i, j, k, l;
 float sum, min; //sum 暂存各行元素之和
 float *p, *pi, *pk;
 for(i=0; i<n; i++)
 {sum=0.0; pk=matrix+i*n; //pk 指向矩阵各行第 1 个元素.
  for (j=0; j<n; j++) {sum+=*(pk); pk++;} //求一行元素之和.
  *(p+i)=sum; //将一行元素之和存入一维数组.
 } //for i
 for(i=0; i<n-1; i++) //用选择法对数组 p 进行排序
 {min=*(p+i); k=i; //初始设第 i 行元素之和最小.
  for(j=i+1; j<n; j++) if(p[j]<min) {k=j; min=p[j];} //记新的最小值及行号.
  if(i!=k) //若最小行不是当前行, 要进行交换(行元素及行元素之和)
  {pk=matrix+n*k; //pk 指向第 k 行第 1 个元素.
```

```

pi=matrix+n*i;    //pi 指向第 i 行第 1 个元素.
for(j=0;j<n;j++) //交换两行中对应元素.
{sum=*(pk+j); *(pk+j)=*(pi+j); *(pi+j)=sum;}
sum=p[i]; p[i]=p[k]; p[k]=sum; //交换一维数组中元素之和.
} //if
} //for i
free(p); //释放 p 数组.
} // Translation

```

[算法分析] 算法中使用选择法排序, 比较次数较多, 但数据交换(移动)较少. 若用其它排序方法, 虽可减少比较次数, 但数据移动会增多. 算法时间复杂度为 $O(n^2)$.

5. [题目分析] 因为数组中存放的是从 1 到 N 的自然数, 原程序运行后, 数组元素 $A[i]$ ($1 \leq i \leq N$) 中存放的是 $A[1]$ 到 $A[i-1]$ 中比原 $A[i]$ 小的数据元素的个数. 易见 $A[N]+1$ 就是原 $A[N]$ 的值(假定是 j , $1 \leq j \leq N$). 设一元素值为 1 的辅助数组 flag, 采用累加, 确定一个值后, flag 中相应元素置零. 下面程序段将 A 还原成原来的 A:

```

VAR flag:ARRAY[1..N] OF integer;
FOR i:=1 TO N DO flag[i]:=1; //赋初值
FOR i:=N DOWNTO 1 DO
  BEGIN sum:=0; j:=1; found:=false;
    WHILE j<=N AND NOT found DO
      BEGIN sum:=sum+flag[j];
        IF sum=A[i]+1 THEN BEGIN flag[j]:=0; found:=true; END;
      END;
    A[i]:=j;
  END;
END;

```

6. [题目分析] 寻找马鞍点最直接的方法, 是在一行中找出一个最小值元素, 然后检查该元素是否是元素所在列的最大元素, 如是, 则输出一个马鞍点, 时间复杂度是 $O(m*(m+n))$. 本算法使用两个辅助数组 max 和 min, 存放每列中最大值元素的行号和每行中最小值元素的列号, 时间复杂度为 $O(m*n+m)$, 但比较次数比前种算法会增加, 也多使用向量空间.

```

int m=10, n=10;
void Saddle(int A[m][n])
//A 是 m*n 的矩阵, 本算法求矩阵 A 中的马鞍点.
{int max[n]={0}, //max 数组存放各列最大值元素的行号, 初始化为行号 0;
  min[m]={0}, //min 数组存放各行最小值元素的列号, 初始化为列号 0;
  i, j;
for(i=0;i<m;i++) //选各行最小值元素和各列最大值元素.
  for(j=0;j<n;j++)
    {if(A[max[j]][j]<A[i][j]) max[j]=i; //修改第 j 列最大元素的行号
      if(A[i][min[i]]>A[i][j]) min[i]=j; //修改第 i 行最小元素的列号.
    }
for (i=0;i<m;i++)
  {j=min[i]; //第 i 行最小元素的列号
    if(i==max[j])printf( "A[%d][%d]是马鞍点, 元素值是%d", i, j, A[i][j]); //是马
鞍点
  }
}

```

```
 }// Saddle
```

[算法讨论] 以上算法假定每行(列)最多只有一个可能的马鞍点, 若有多个马鞍点, 因为一行(或一列)中可能的马鞍点数值是相同的, 则可用二维数组 min2, 第一维是行向量, 是各行行号, 第二维是列向量, 存放一行中最大值的列号。对最大值也同样处理, 使用另一二维数组 max2, 第一维是列向量, 是各列列号, 第二维存该列最大值元素的行号。最后用类似上面方法, 找出每行(i)最小值元素的每个列号(j), 再到 max2 数组中找该列是否有最大值元素的行号(i), 若有, 则是马鞍点。

7. [题目分析] 我们用 l 代表最长平台的长度, 用 k 指示最长平台在数组 b 中的起始位置(下标)。用 j 记住局部平台的起始位置, 用 i 指示扫描 b 数组的下标, i 从 0 开始, 依次和后续元素比较, 若局部平台长度(i-j)大于 l 时, 则修改最长平台的长度 k(l=i-j)和其在 b 中的起始位置(k=j), 直到 b 数组结束, l 即为所求。

```
void Platform (int b[ ], int N)
//求具有 N 个元素的整型数组 b 中最长平台的长度。
{ l=1; k=0; j=0; i=0;
while(i<n-1)
{ while(i<n-1 && b[i]==b[i+1]) i++;
if(i-j+1>l) { l=i-j+1; k=j; } //局部最长平台
i++; j=i; } //新平台起点
printf("最长平台长度%d, 在 b 数组中起始下标为%d", l, k);
} // Platform
```

8. [题目分析] 矩阵中元素按行和按列都已排序, 要求查找时间复杂度为 $O(m+n)$, 因此不能采用常规的二维循环的查找。可以先从右上角 ($i=a, j=d$) 元素与 x 比较, 只有三种情况: 一是 $A[i, j]>x$, 这情况下向 j 小的方向继续查找; 二是 $A[i, j]<x$, 下步应向 i 大的方向查找; 三是 $A[i, j]=x$, 查找成功。否则, 若下标已超出范围, 则查找失败。

```
void search(datatype A[ ][ ], int a, b, c, d, datatype x)
//n*m 矩阵 A, 行下标从 a 到 b, 列下标从 c 到 d, 本算法查找 x 是否在矩阵 A 中。
{ i=a; j=d; flag=0; //flag 是成功查到 x 的标志
while(i<=b && j>=c)
if(A[i][j]==x) { flag=1; break; }
else if (A[i][j]>x) j--; else i++;
if(flag) printf("A[%d][%d]=%d", i, j, x); //假定 x 为整型。
else printf("矩阵 A 中无%d 元素", x);
} //算法 search 结束。
```

[算法讨论] 算法中查找 x 的路线从右上角开始, 向下 (当 $x>A[i, j]$) 或向左 (当 $x<A[i, j]$)。向下最多是 m, 向左最多是 n。最佳情况是在右上角比较一次成功, 最差是在左下角 ($A[b, c]$), 比较 $m+n$ 次, 故算法最差时间复杂度是 $O(m+n)$ 。

9. [题目分析] 本题的一种算法前面已讨论 (请参见本章三、填空题 38)。这里给出另一中解法。分析数的填法, 是按“从右上到左下”的“蛇形”, 沿平行于副对角线的各条对角线上, 将自然数从小到大填写。当从右上到左下时, 坐标 i 增加, 坐标 j 减小, 当 j 减到小于 0 时结束, 然后 j 从 0 开始增加, 而 i 从当前值开始减少, 到 $i<0$ 时结束。然后继续如此循环。当过副对角线后, 在 $i>n-1$ 时, $j=j+2$, 开始从左下向右上填数; 而当 $j>n-1$ 时 $i=i+2$, 开始从右上向左下的填数, 直到 $n*n$ 个数填完为止。

```
void Snake_Number(int A[n][n], int n)
//将自然数 1..n*n, 按“蛇形”填入 n 阶方阵 A 中。
```

```
{i=0; j=0; k=1;           //i, j 是矩阵元素的下标, k 是要填入的自然数.
while(i<n && j<n)
{while(i<n && j>-1)        //从右上向左下填数,
{A[i][j]=k++; i++ ;j--;}
if((j<0)&&(i<n)) j=0;      //副对角线及以上部分的新 i, j 坐标.
else {j=j+2; i=n-1;}      // 副对角线以下的新的 i, j 坐标.
while(i>-1 && j<n)        //从左下向右上
{A[i][j]=k++; i--; j++;}
if(i<0 && j<n) i=0;
else{i=i+2; j=n-1;}
} //最外层 while
} //Snake_Number
```

10. [题目分析]判断二维数组中元素是否互不相同, 只有逐个比较, 找到一对相等的元素, 就可结论为不是互不相同。如何达到每个元素同其它元素比较一次且只一次? 在当前行, 每个元素要同本行后面的元素比较一次 (下面第一个循环控制变量 p 的 for 循环), 然后同第 i+1 行及以后各行元素比较一次, 这就是循环控制变量 k 和 p 的二层 for 循环。

```
int JudgEqual (ing a[m][n], int m, n)
//判断二维数组中所有元素是否互不相同, 如是, 返回 1; 否则, 返回 0。
{for(i=0; i<m; i++)
for(j=0; j<n-1; j++)
{ for(p=j+1; p<n; p++) //和同行其它元素比较
if(a[i][j]==a[i][p]) {printf("no"); return(0); }
//只要有一个相同的, 就结论不是互不相同
for(k=i+1; k<m; k++) //和第 i+1 行及以后元素比较
for(p=0; p<n; p++)
if(a[i][j]==a[k][p]) {printf("no"); return(0); }
} // for(j=0; j<n-1; j++)
printf("yes"); return(1); //元素互不相同
} //算法 JudgEqual 结束
```

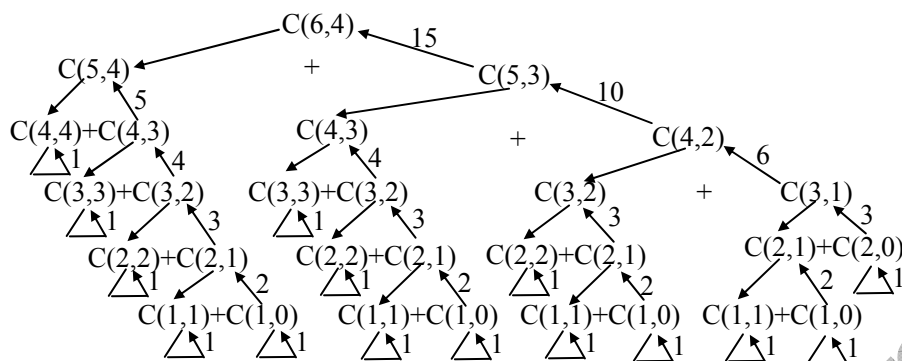
(2) 二维数组中的每一个元素同其它元素都比较一次, 数组中共 $m \times n$ 个元素, 第 1 个元素同其它 $m \times n - 1$ 个元素比较, 第 2 个元素同其它 $m \times n - 2$ 个元素比较, …… , 第 $m \times n - 1$ 个元素同最后一个元素 ($m \times n$) 比较一次, 所以在元素互不相等时总的比较次数为 $(m \times n - 1) + (m \times n - 2) + \dots + 2 + 1 = (m \times n)(m \times n - 1) / 2$ 。在有相同元素时, 可能第一次比较就相同, 也可能最后一次比较时相同, 设在 $(m \times n - 1)$ 个位置上均可能相同, 这时的平均比较次数约为 $(m \times n)(m \times n - 1) / 4$, 总的时间复杂度是 $O(n^4)$ 。

11. 二项式 $(a+b)^n$ 展开式的系数的递归定义为:

$$C(n, k) = \begin{cases} 1 & \text{当 } k = 0 \text{ 或 } k = n (n \geq 0) \\ C(n-1, k) + C(n-1, k-1) & \text{当 } (0 < k < n) \end{cases} \quad C(n, k) = C_n^k = \frac{n(n-1)\dots(n-k+1)}{1*2*\dots*(k-1)*k}$$

```
(1) int BiForm(int n, k)           //二项式展开式的系数的递归算法
{if(n<0 || k<0 || k>=n) {printf("参数错误\n"); exit(0);}
if(k==0 || k==n) return(1);
else return(BiForm(n-1, k)+BiForm(n-1, k-1);
}
```


(2) $C(6, 4)$ 的递归树



(3) 计算 $C(n, k)$ ($0 \leq k \leq n$) 的非递归算法

```
int cnk(int n, int k)
{
    int i; long x=1, y=1;
    for (i=1; i<=k; i++) x*=i;
    for (i=n-k+1; i<=n; i++) y*=i;
    return(y/x)
} //cnk
```

12. [题目分析] 本题属于排序问题，只是排出正负，不排出大小。可在数组首尾设两个指针 i 和 j ， i 自小至大搜索到负数停止， j 自大至小搜索到正数停止。然后 i 和 j 所指数据交换，继续以上过程，直到 $i=j$ 为止。

```
void Arrange(int A[], int n)
//n 个整数存于数组 A 中，本算法将数组中所有正数排在所有负数的前面
{
    int i=0, j=n-1, x; //用类 C 编写，数组下标从 0 开始
    while(i<j)
    {
        while(i<j && A[i]>0) i++;
        while(i<j && A[j]<0) j--;
        if(i<j) {x=A[i]; A[i++]=A[j]; A[j--]=x; } //交换 A[i] 与 A[j]
    }
} //算法 Arrange 结束.
```

[算法讨论] 对数组中元素各比较一次，比较次数为 n 。最佳情况(已排好，正数在前，负数在后)不发生交换，最差情况(负数均在正数前面)发生 $n/2$ 次交换。用类 c 编写，数组界偶是 $0..n-1$ 。空间复杂度为 $O(1)$ 。

类似本题的其它题的解答::

(1) 与上面 12 题同，因要求空间复杂度也是 $O(n)$ ，可另设一数组 C，对 A 数组从左到右扫描，小于零的数在 C 中从左(低下标)到右(高下标)存，大于等于零的数在 C 中从右到左存。

(2) 将 12 题中判定正数 ($A[i]>0$) 改为判偶数 ($A[i]\%2==0$)，将判负数 ($A[j]<0$) 改为 ($A[j]\%2!=0$)。

(3) 同 (2)，只是要求奇数排在偶数之前。

(4) 利用快速排序思想，进行一趟划分。

```
int Partition(int A[], int n)
//将 n 个元素的数组 A 调整为左右两部分，且左边所有元素小于右边所有元素，返回分界位置。
{
    int i=0, j=n-1, rp=A[0]; //设数组元素为整型
```

```

while(i<j)
{while(i<j &&A[j]>=rp) j--;
 while(i<j &&A[i]<=rp) i++;
 if(i<j) { x=A[i];A[i]=A[j]; A[j]=x; }
}
A[i]=rp; return(i); //分界元素
} // Partition

```

13. [题目分析] 设 n 个元素存放在数组 $A[1..n]$ 中。设 S 初始为空集，可依次将数组 A 的每一个元素并入 S ，产生了含一个元素的若干集合，再以含一个元素的集合为初始集合，依次并入 A 的第二个（异于 S 的那个元素）元素并入 S ，形成了含两个元素的若干集合，……，如此下去，直至 $A[i]$ 的全部元素并入。

```

CONST n=10;
TYPE datatype=char;
VAR A: array[1..n] OF datatype;
PROC powerset(s:set OF datatype)
[outset(s); //输出集合 S
 FOR i:=1 TO n DO powerset(S+A[i]);
]
ENDP;

```

调用本过程时，参数 S 为空集 $[\]$ 。

14. [题目分析] 设稀疏矩阵是 $A_{m \times n}$, H_m 是总表头指针。设 rch 是行列表头指针，则 $rch \rightarrow right = rch$ 时该行无非零元素，用 i 记行号，用一维数组元素 $A[i]$ 记第 i 行非零元个数。（为方便输出，设元素是整数。）

```

int MatrixNum(Olink Hm)
//输出由 Hm 指向的十字链表中每一行的非零元素个数
{Olink rch=Hm->uval.next, p;
 int A[]; i=1; //数组 A 记各行非零元个数, i 记行号
 while(rch!=Hm) //循环完各行列表头
 {p=rch->right; num=0; //p 是稀疏矩阵行内工作指针, num 记该行非零个数
  while(p!=rch) //完成行内非零元的查找
  {printf( "M[%d][%d]=%d", p->row, p->col, p->uval.e);
   num++; p=p->right; printf( "\n"); //指针后移 }
  A[i++]=num; //存该行非零元个数
  rch=rch->uval.next; //移到下一行列表头
 }
 num=0
 for(j=1; j<i; j++) //输出各行非零元个数
 {num+=A[j]; printf( "第%d 行非零元个数为%d\n", j, A[j]); }
 return(num); //稀疏矩阵非零元个数
} 算法结束

```

15. [题目分析] 广义表的元素有原子和表。在读入广义表“表达式”时，遇到左括号 ‘(’ 就递归的构造子表，否则若是原子，就建立原子结点；若读入逗号 ‘,’，就递归构造后续子表；若 $n=0$ ，则构造含空格字符的空表，直到碰到输入结束符号（‘#’）。设广义表的形式定义如下：

```

typedef struct node

```



```

{int tag;                //tag=0 为原子, tag=1 为子表
 struct node *link;      //指向后继结点的指针
 union {struct node *slink; //指向子表的指针
        char data;        //原子
        }element;
}Glist;
Glist *creat ()
//建立广义表的存储结构
{char ch; Glist *gh;
 scanf( "%c" ,&ch);
 if(ch==' ' ) gh=null;
 else {gh=(Glist*)malloc(sizeof(Glist));
        if(ch==' ( ' ) {gh->tag=1; //子表
                        gh->element.slink=creat(); } //递归构造子表
        else {gh->tag=0; gh->element.data=ch;} //原子结点
      }
 scanf( "%c" ,&ch);
 if(gh!=null) if(ch==' , ' ) gh->link=creat(); //递归构造后续广义表
               else gh->link=null;
 return(gh);
}

```

}算法结束

16、(1)略

(2)求广义表原子个数的递归模型如下

$$f(p) = \begin{cases} 1 + f(p^{\wedge}.link) & \text{如果 } p^{\wedge}.tag = 0 \\ f(p^{\wedge}.sublist) + f(p^{\wedge}.link) & \text{如果 } p^{\wedge}.tag = 1 \end{cases}$$

```

PROC Number(p:glist; VAR n: integer)
VAR m:integer;
n:=0;
IF p<>NIL THEN
  [IF p^ .tag=0 THEN n:=1 ELSE Number(p^ .sublist,m)
   n:=n+m; Number(p^ .link,m); n:=n+m; ]
ENDP;

```

17. int Count(glist *gl)

//求广义表原子结点数据域之和, 原子结点数据域定义为整型

```

{if(gl==null) return(0);
 else if (gl->tag==0) return((p->data)+count(gl->link));
      else return(count(gl->sublist)+count(gl->link)); }
} // Count

```

18. (1) 在 n 个正整数中, 选出 k ($k \leq m$) 个最大的数, 应使用堆排序方法。对深度为 h 的堆, 筛选算法中关键字的比较次数至多为 $2(h-1)$ 次。建堆总共进行的关键字比较次数不超过 $4n$, 堆排序在最坏情况下的时间复杂度是 $O(n \log n)$ 。

int r[1000]; // r[1000]是整型数组

(2) **void sift(int r[], int k, m, tag)**

//已知 $r[k+1..m]$ 是堆, 本算法将 $r[k..m]$ 调整成堆, $tag=1$ 建立大根堆, $tag=2$ 建立小根堆
 $\{i=k; j=2*i; x=r[k];$

while ($j \leq m$)

if ($tag==2$) //建立小根堆

$\{ \text{if } (j < m \ \&\& \ r[j] > r[j+1]) \ j++; // \text{沿关键字小的方向筛选}$

$\text{if } (r[j] < x) \ \{ r[i] = r[j]; i = j; j = 2*i; \}$

else break;

else //建立大根堆

$\{ \text{if } (j < m \ \&\& \ r[j] < r[j+1]) \ j++; // \text{沿关键字小的方向筛选}$

$\text{if } (r[j] > x) \ \{ r[i] = r[j]; i = j; j = 2*i; \}$

else break;

}

$r[i] = x;$

} //sift

main(int argc, char *argv[])

//根据命令行中的输入, 从 1000 个数中选取 n 个最大数或 n 个最小数

{int $m=1000, i, j;$

$n=argv[2];$ //从命令行输入的第二个参数是需要输出的数的个数

if ($n > m$) {**printf**(“参数错误\n”); **exit**(0);}

for ($i=0; i < m; i++$) **scanf**(“%d”, $\&r[i]$); //输入 1000 个大小不同的正整数

if ($argv[1] == 'a'$) //输出 n 个最大数, 要求建立大根堆

{for ($i=m/2; i > 0; i--$) **sift**($r, i, m, 1$)

printf(“%d 个最大数依次为\n”, n);

for ($i=m; i > m-n+1; i--$) //输出 n 个最大数

$\{ \text{printf}(\text{"%5d"}, r[i]); \ j++; \text{if}((j+1)\%5==0) \text{printf}(\text{"\n"}); // \text{一行打印 } 5$

个数

$\text{sift}(r, 1, i-1, 1); \}$ //调堆

}

else //($argv[1] == 'i'$) //输出 n 个最小数, 要求建立小根堆

{for ($i=m/2; i > 0; i--$) **sift**($r, i, m, 2$)

printf(“%d 个最小数依次为\n”, n);

for ($i=m; i > m-n+1; i--$) //输出 n 个最小数

$\{ \text{printf}(\text{"%5d"}, r[i]); \ j++; \text{if}((j+1)\%5==0) \text{printf}(\text{"\n"}); // \text{一行打印 } 5$

个数

$\text{sift}(r, 1, i-1, 2); \}$ //调堆

}

} //main

[算法讨论] 算法讨论了建堆, 并输出 n (n 小于等于 m) 个最大(小)数的情况, 由于要求输出 n 个最大数或最小数, 必须建立极大化堆和极小化堆。注意输出时的 **for** 循环控制到变量 i 从 m 变化到 $m-n+1$, 这是堆的性质决定的, 只有堆顶元素才是最大(小)的。要避免使 i 从 1 到 n 来输出 n 个最大(小)数的错误。

19、[题目分析] 题目要求调整后第一数组 (A) 中所有数均不大于第二个数组 (B) 中所有数。因两数组分别有序, 这里实际是要求第一数组的最后一个数 $A[m-1]$ 不大于第二个数组的第一个

数 $B[0]$ 。由于要求将第二个数组的数插入到第一个数组中。因此比较 $A[m-1]$ 和 $B[0]$ ，如 $A[m-1] > B[0]$ ，则交换。交换后仍保持 A 和 B 有序。重复以上步骤，直到 $A[m-1] \leq B[0]$ 为止。

```
void Rearranger (int A[], B[], m, n)
```

//A 和 B 是各有 m 个和 n 个整数的非降序数组，本算法将 B 数组元素逐个插入到 A 中，使 A 中各元素均不大于 B 中各元素，且两数组仍保持非降序排列。

```
{ while (A[m-1]>B[0])
    {x=A[m-1];A[m-1]=B[0]; //交换 A[m-1]和 B[0]
      j=1;
      while(j<n && B[j]<x) B[j-1]=B[j++]; //寻找 A[m-1]的插入位置
      B[j-1]=x;
      x=A[m-1];i=m-2;
      while(i>=0 && A[i]>x) A[i+1]=A[i--]; //寻找 B[0]的插入位置
      A[i+1]=x;
    }
```

} 算法结束

20、[题目分析]本题中数组 A 的相邻两段分别有序，要求将两段合并为一段有序。由于要求附加空间为 $O(1)$ ，所以将前段最后一个元素与后段第一个元素比较，若正序，则算法结束；若逆序则交换，并将前段的最后一个元素插入到后段中，使后段有序。重复以上过程直到正序为止。

```
void adjust(int A[], int n)
```

//数组 $A[n-2k+1..n-k]$ 和 $[n-k+1..n]$ 中元素分别升序, 算法使 $A[n-2k+1..n]$ 升序

$$\{i=n-k; j=n-k+1;$$

```
while(A[i]>A[j])
```

```
{x=A[i]; A[i]=A[j]; //值小者左移, 值大者暂存于 x
```

 $k = i + 1:$

```
while (k<n && x>A[k]) A[k+1]=A[k++]; //调整后段有序
```

$$A[k-1]=x:$$

```
i--; j--; //修改前段最后元素和后段第一元素的指针
```

}

} 算法结束

〔算法讨论〕最佳情况出现在数组第二段 $[n-k+1..n]$ 中值最小元素 $A[n-k+1]$ 大于等于第一段值最大元素 $A[n-k]$ ，只比较一次无须交换。最差情况出现在第一段的最小值大于第二段的最大值，两段数据间发生了 k 次交换，而且每次段交换都在段内发生了平均 $(k-1)$ 次交换，时间复杂度为 $O(n^2)$ 。

21、[题目分析]题目要求按 B 数组内容调整 A 数组中记录的次序，可以从 $i=1$ 开始，检查是否 $B[i]=i$ 。如是，则 $A[i]$ 恰为正确位置，不需再调；否则， $B[i]=k \neq i$ ，则将 $A[i]$ 和 $A[k]$ 对调， $B[i]$ 和 $B[k]$ 对调，直到 $B[i]=i$ 为止。

```
void CountSort (rectype A[], int B[])
```

//A 是 100 个记录的数组，B 是整型数组，本算法利用数组 B 对 A 进行计数排序

```
{int i, j, n=100;
```

```
i=1;
```

```
while (i<n)
```

```
{if(B[i]!=i) //若 B[i]=i 则 A[i]正好在自己的位置上，则不需要调整
```

$$\{j=i;$$

```
while (B[i] != i)
```

```

    { k=B[j]; B[j]=B[k]; B[k]=k;    // B[j]和B[k]交换
      r0=A[j];A[j]=A[k]; A[k]=r0; } //r0 是数组 A 的元素类型,A[j]和A[k]交换
    i++;} //完成了一个小循环,第 i 个已经安排好
  } //算法结束

```

22、[题目分析]数组 A 和 B 的元素分别有序,欲将两数组合并到 C 数组,使 C 仍有序,应将 A 和 B 拷贝到 C,只要注意 A 和 B 数组指针的使用,以及正确处理一数组读完数据后将另一数组余下元素复制到 C 中即可。

```

void union(int A[],B[],C[],m,n)
//整型数组 A 和 B 各有 m 和 n 个元素,前者递增有序,后者递减有序,本算法将 A 和 B 归并
//为递增有序的数组 C。
{i=0; j=n-1; k=0; // i, j, k 分别是数组 A,B 和 C 的下标,因用 C 描述,下标从 0 开始
while(i<m && j>=0)
  if(a[i]<b[j]) c[k++]=a[i++] else c[k++]=b[j--];
while(i<m) c[k++]=a[i++];
while(j>=0) c[k++]=b[j--];
}算法结束

```

[算法讨论]若不允许另辟空间,而是利用 A 数组(空间足够大),则初始 $k=m+n-1$,请参见第 2 章算法设计第 4 题。

23、[题目分析]本题要求建立有序的循环链表。从头到尾扫描数组 A,取出 $A[i]$ ($0 \leq i < n$),然后到链表中去查找值为 $A[i]$ 的结点,若查找失败,则插入。

```

LinkedList creat(ElemType A[],int n)
//由含 n 个数据的数组 A 生成循环链表,要求链表有序并且无值重复结点
{LinkedList h;
h=(LinkedList)malloc(sizeof(LNode)); //申请结点
h->next=h; //形成空循环链表
for(i=0;i<n;i++)
{pre=h;
p=h->next;
while(p!=h && p->data<A[i])
  {pre=p; p=p->next;} //查找 A[i]的插入位置
if(p==h || p->data!=A[i]) //重复数据不再输入
  {s=(LinkedList)malloc(sizeof(LNode));
  s->data=A[i]; pre->next=s; s->next=p; //将结点 s 链入链表中
  }
} //for
return(h);
}算法结束

```