

## 第2章 线性表

### 一. 填空题

- 1、顺序存储结构，链式存储结构
- 2、也，不一定
- 3、链式，顺序
- 4、 $n-i+1$ ， $n-i$
- 5、 $O(1)$
- 6、插入和删除元素不需要移动元素，小
- 7、头指针，头结点的 next 域，其前驱的 next 域
- 8、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。
- 9、 $f \rightarrow next = p \rightarrow next$ ， $p \rightarrow next \rightarrow prior = f$ ， $f \rightarrow prior = p$ ， $p \rightarrow next = f$
- 10、a. 在 p 结点后插入 s 结点的语句序列是：（4）、（1）  
 b. 在 p 结点前插入 s 结点的语句序列是：（8）、（12）、（9）、（4）、（1）  
 c. 在表首插入 s 结点的语句序列是：（5）、（13）  
 d. 在表尾插入 s 结点的语句序列是：（12）、（10）、（1）、（7）

### 二. 单项选择题

题号	1	2	3	4	5	6	7	8	9	10
答案	A	B	A	B	B	A	B	A	D	B
题号	11	12	13	14	15	16	17	18	19	20
答案	C	C	C	A	B	B	A	B	C	C

### 三. 判断题

题号	1	2	3	4	5	6	7	8	9	10
答案	×	×	√	×	×	×	√	√	×	×

### 四. 综合题

- 1、答：顺序存储结构可以实现对表中元素的随机存取，但在进行插入或删除操作时，需要移动大量元素。  
 链表在存储空间的合理利用、插入删除操作不需要移动大量元素方面优于顺序表，但链表不能像顺序表那样实现元素的随机存取。
- 2、答：

头指针：指向链表第一个结点（或为头结点或为首元结点）的指针，单链表可由头指针唯一地确定。

头结点：在链表的第一个结点之前附设的一个结点，头结点的指针域指向表中的第一个结点。

首元结点：链表中存放第一个数据元素  $a_1$  的结点，在带头结点的链表中，它是头结点的下一个结点。

3、

05	U	17	X	23	V	31	Y	47	Z
100	112	104	116	108	104	112	0	116	100

X: 116

Y: 0

Z: 100

首结点的起始地址为：108

末结点的起始地址为：112

4、

```
int insertx(int a[],int R, int x)
{
    int i,j;
    for(i=0; i<R&& a[i]<x; i++);    /*找到 x 的插入位置*/
    for(j=R-1;j>=i;j--)    a[j+1]=a[j];    /*插入位置之后的元素后移*/
    a[i]=x;    /*插入 x*/
    R=R+1;
    return R;    /*返回新的 R 值*/
}
```

5、（1）以一维数组作存储结构，设线性表存于  $a[n]$  中。

```
void invert1(int a[], int n)
{
    int j,temp;
    for(j=0; j<=(n-1)/2; j++)
    {
        temp = a[j];
        a[j] = a[n-j-1];
        a[n-j-1] = temp;
    } // for
} // invert
```

（2）以单链表作存储结构。

算法思路：依次取原链表中的每个结点，将其作为第一个结点插入到新链表中去，指针  $p$  用来指向当前结点， $p$  为空时结束。

```
void invert2(LNode *L)    /*L 是带头结点的单链表，本算法将其逆置*/
{
    LNode *p,*q;
```

```

p=L->next;          /*初始时 p 指向第一个数据结点*/
L->next=NULL;       /*先建一个只有头结点的空链表*/
while (p!=NULL)
{
    q=p;   p=p->next;
    q->next=L->next    /*将当前结点前插入头结点之后*/
    L->next=q;
}
}

```

6、

```

void DeleteMin(LNode *L) /*L 是带头结点的单链表，本算法删除其最小值结点*/
{
    LNode *p,*pre,*q;
    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*/
}

```

7、

```

void LocateNode(DuLNode *L, int x)
{
    DuLNode *p,*q;
    p=L->next;
    while(p&& p->data!=x) p=p->next;
    if(p==NULL) printf("No node of value x.");
    else{
        p->freq++;
        p->prior->next=p->next; p->next->prior=p->prior; /*删除 p*/
        q=L->next;
        while(q->freq>=p->freq) q=q->next; /*找到插入位置*/
        q->prior->next=p; /*将 p 插入到 q 前*/
        p->prior=q->prior;
        q->prior=p;
        p->next=q;
    }
}

```