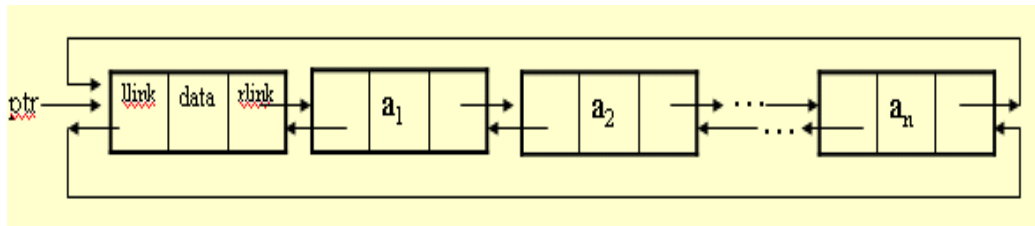


## 书面作业 4:《数据结构》Chapter 4 Linked Lists

### 一、选择题

1. 链表是一种采用( **B** )存储结构存储的线性表。  
A. 顺序                  B. 链式                  C. 星式                  D. 网状
2. 线性表若采用链式存储结构, 要求内存中可用存储单元的地址( **D** )。  
A. 必须是连续的                  B. 部分地址必须是连续的  
C. 一定是不连续的                  D. 连续或不连续都可以
3. 一般单链表(无头结点, 结点结构为 (data,next)), 为空的判定条件是( **A** )。  
A. head==NULL                  B. head->next==NULL  
C. head->next==head                  D. head!=NULL
4. 循环单链表(带头结点, 结点结构为 (data,next)) 为空的判定条件是( **C** )。  
A. head==NULL                  B. head->next==NULL  
C. head->next==head                  D. head!=NULL
5. 链表的一个主要缺点在于( **D** )。  
A. 插入元素需要移动后面一系列元素    B. 删除元素需要移动后面一系列元素  
C. 查找某个元素的效率远低于顺序表    D. 访问指定位置元素的效率远低于顺序表
6. 如果程序中要维护一个序列, 经常需要**遍历**序列中的元素, 以及在序列**尾部加入和删除**元素, 则选择下面哪种数据结构最为合适( **B** )。  
A. 标准数组    B. 动态数组    C. 单链表    D. 双向循环链表
7. 下面叙述中**不正确**的是( **BC** )。  
A. 线性表在链式存储时, 读取第  $i$  个元素的时间同  $i$  的值成正比。  
B. 线性表在链式存储时, 读取第  $i$  个元素的时间同  $i$  的值无关。  
C. 线性表在顺序存储时, 读取第  $i$  个元素的时间同  $i$  的值成正比。  
D. 线性表在顺序存储时, 读取第  $i$  个元素的时间同  $i$  的值无关。
8. 单链表和双向循环链表表示线性表 List 时常常增加一个头结点, 其目的是( **C** )。  
A. 为了简化查找算法                  B. 为了简化归并算法  
C. 为插入删除操作时不需要做特殊的处理                  D. 以上说法都不对。
9. 带头结点的非循环单链表 head 的尾结点(由  $p$  所指向, 结点的数据域为 Element, 指针域为 Next) 满足( **D** )。  
A.  $p == head$     B.  $p == NULL$     C.  $p->Next == head$                   D.  $p->Next == NULL$
10. 若非空单链表的数据域为 Element, 指针域为 Next, 指针  $p$  指向单链表中第  $i$  个结点,  $s$  指向已生成的新结点, 现要将  $s$  结点插入到单链表中  $p$  结点后, 使其成为第  $i+1$  个结点, 则下列算法段能正确完成上述要求的是( **A** )。  
A.  $s->Next = p->Next; p->Next = s;$                   B.  $p->Next = s; s->Next = p->Next;$   
C.  $p = s; s->Next = p;$                   D.  $s->Next = p; p->Next = s;$
11. 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域有(        )个, 单链表中需修改( **C** )个。  
A. 2, 4    B. 2, 4    C. 4, 2    D. 4, 4
12. 设双向循环链表 ptr 如图所示, 将指针  $q$  所指的结点插入到双向循环链表尾部(即数据元素  $a_n$  之后)的正确的算法段为( **AC** )。注: 第一个结点为头结点, 该题有多于一个的答案。



A).

...

```
q->llink = ptr->llink;
q->rlink = ptr;
ptr->llink = q;
q->llink->rlink = q;
```

...

B). ...

```
ptr->llink = q;
q->rlink = ptr;
ptr->llink->rlink = q;
q->llink = ptr->llink;
```

...

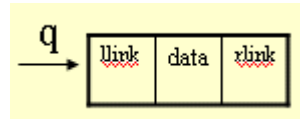
C). ...

```
q->llink = ptr->llink;
q->rlink = ptr;
ptr->llink->rlink = q;
ptr->llink = q;
```

...

D)...

```
ptr->llink = q;
q->rlink = ptr;
q->llink = ptr->llink;
q->llink->rlink = q;
```



13. 已知两个长度分别为  $m$  和  $n$  的升序单链表，若将它们合并为一个长度为  $m+n$  的降序链表，则最好合并算法的时间复杂度是（ **D** ）

A.  $O(n)$       B.  $O(mn)$       C.  $O(\min(m,n))$       D.  $O(m+n)$

14. 用不带头结点的单链表存储队列，其队头指针指向单链表第一个结点，队尾指针指向链尾结点，则在进行出队操作时（ **D** ）。

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

15. 链栈和顺序栈相比，有一个比较明显的优点是（ **A** ）。

A. 通常不会出现栈满情况      B. 通常不会出现栈空情况

- C. 插入操作更方便      D. 删除操作更方便
16. 用不带头结点的循环链表表示的队列长度为  $n$ , 若只设尾指针(指向第一个结点的指针), 则出队和入队的时间复杂度分别是( **D** )。
- A.  $O(n)$   $O(n)$       B.  $O(1)$   $O(n)$       C.  $O(n)$   $O(1)$       D.  $O(1)$   $O(1)$

## 二、简答题

1. 《数据结构题集》P14, 2.7

答: a. (11), (3), (14)

b. (10), (12), (8), (11), (3), (14)

c. (10), (12), (7), (3), (14)

d. (12), (11), (3), (14)

e. (9), (11), (3), (14)

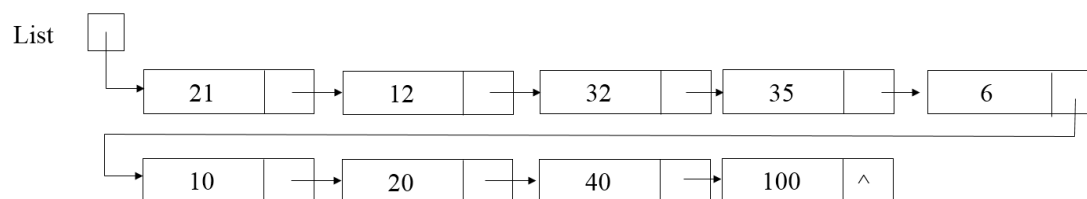
2. 给定由整数元素组成的线性表  $List=(21, 12, 32, 35, 6, 10, 20, 40, 100)$ , 请分别画出其顺序存储结构、单链表、循环链表和双向循环链表的存储示意图。

答:

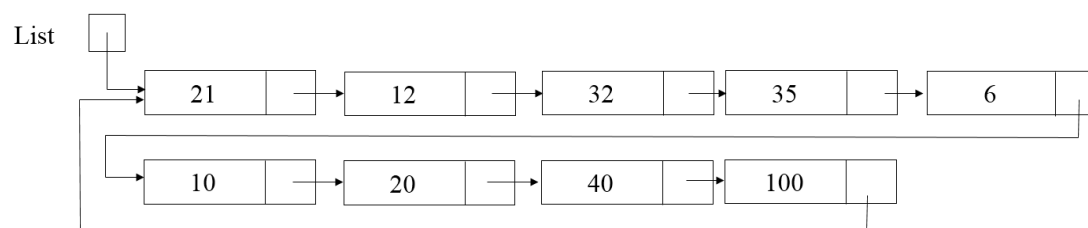
(1) 顺序存储结构:

0	1	2	3	4	5	6	7	8
21	12	32	35	6	10	20	40	100

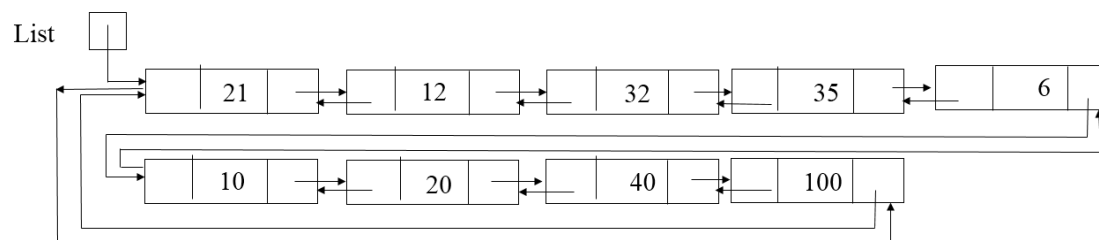
(2) 单链表 (不带头结点)



(3) 循环链表 (不带头结点)



(4) 双向循环链表 (不带头结点)



3. 用循环链表表示的队列长度为  $n$ , 若只设头指针 (指向第一个结点的指针), 则出队和入队的时间复杂度分别是多少? 若只设尾指针 (指向最后一个结点的指针), 则出队和入队的时间复杂度呢? 请说明原因。

答:

(1) 假设循环链表带头结点:

若头指针, 出队时间复杂度为  $O(1)$ , 入队时间复杂度为  $O(n)$

若尾指针, 出队时间复杂度为  $O(1)$ , 入队时间复杂度为  $O(1)$

(2) 假设循环链表不带头结点:

若头指针, 出队时间复杂度为  $O(n)$ , 入队时间复杂度为  $O(n)$

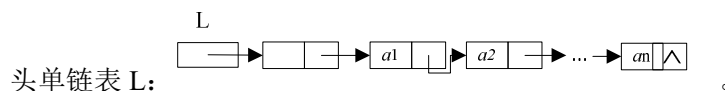
若尾指针, 出队时间复杂度为  $O(1)$ , 入队时间复杂度为  $O(1)$

### 三、算法题

下列 1-4 小题的单链表结点定义为:

```
struct listNode;  
typedef struct listNode *listPointer;  
typedef struct listNode {  
    int data;  
    listPointer link;  
};
```

1. 依次从键盘输入  $n$  个整数组成的线性序列  $(a_1, a_2, \dots, a_n)$ , 编写函数 CreateLink 创建带表



函数原型为: `listPointer CreateLink ( );`

输入示例:

7

2 4 20 35 50 60 86

答:

```
listPointer CreatList ( int n)  
{  
    listPointer p;  
    MALLOC(p, sizeof(listNode), listPointer);  
    scanf("%d", & p->data);  
    p->next = NULL;  
    if (n)  
        p->next = CreatList(n-1);  
    return p;  
}
```

2. 给定一个带表头结点的单链表 Front, 请编写函数 DeleteBack 实现删除链表尾节点, 函数原型为: `void DeleteBack (listPointer front)`, 其中 front 为链表的头指针。

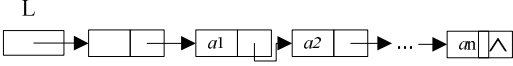
答:

```
void DeleteBack (listPointer front)  
{  
    listPointer rear;  
    rear = front->link;
```

```

while (rear->link) {rear = rear->link; front = front->link;}
front->link = NULL;
FREE(rear);
}

```

3. 给定带表头结点的单链表  , 编写函数 ClearLink 将单链表 L 清空为空表。

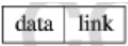
函数原型为: `void ClearLink (listPointer L);`

答:

```

void DestroyList(listPointer L)
{
    listPointer p = L->link;
    while (p)
    {
        L->link = p->link;
        FREE(p);
        p = L->link;
    }
}

```

4. 已知一个带有表头结点的单链表, 结点结构为  , 假设该链表只给出了头指针 list。在不改变链表的前提下, 请设计一个尽可能高效的算法, 查找链表中倒数第 k 个位置上的结点 (k 为正整数)。若查找成功, 算法输出该结点的 data 值, 并返回 TRUE; 否则, 只返回 FALSE。要求: (1) 写出该算法的实现函数 FindK; (2) 说明你所设计算法的时间复杂度和空间复杂度。

函数原型为: `int FindK (listPointer list, int k, int &data);`

```

int FindK (listPointer list, int k, int &data)
{
    listPointer front, rear;
    int i;
    front = list->link;
    rear = list;
    i = 1;
    while( front ){
        front = front->link; i++;
        if(i > k) rear = rear->link;
    }
    if( rear == list ) return 0;
    else{ data = rear->data ; return 1;}
}

```

- (3) 假设单链表中有 n 个元素, 则  $T(n) = O(n)$ ,  $S(n) = O(1)$ 。

下列 5-6 小题的双向循环链表结点定义为:

```

struct node;

```

```

typedef struct node *nodePointer;
typedef struct node {
    nodePointer llink;
    element data;
    nodePointer rlink;
};

```

5. 依次从键盘输入  $n$  个整数组成的线性序列  $(a_1, a_2, \dots, a_n)$ ，编写函数 CreateLink\_D 创建带表头结点的双向循环链表 H:。

函数原型为: nodePointer CreateLink\_D ();

输入示例:

```

7
2 4 20 35 50 60 86

```

答:

```

nodePointer CreateLink_D ()
{
    nodePointer L, node, newnode;
    int n; int item;
    MALLOC(L, sizeof(node), nodePointer);
    L->llink = L; L->rlink = L;
    node = L;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &item);
        MALLOC(newnode, sizeof(node), nodePointer);
        newnode->data = item;
        dinsert(node, newnode);
        node = node->rlink;
    }
}

void dinsert(nodePointer node, nodePointer newnode)
{
    newnode->llink = node;
    newnode->rlink = node->rlink;
    node->rlink->llink = newnode;
    node->rlink = newnode;
}

```

6. 给定一个非空的带表头结点的双向循环链表 H，编写函数 DeleteNum 实现删除链表中第  $i$  ( $1 \leq i \leq n$ ) 个元素，函数原型为: int DeleteNum\_D (listPointer H, int i, int &item)，其中 H 为链表的头指针，若第  $i$  个元素不存在则返回 FALSE，存在并删除成功返回 TRUE。

答:

```
int DeleteNum_D (listPointer H, int i, int &item)
{
    listPointer p, deleted; int j;
    p = H;  j=0;
    while (p->link && j < i - 1){
        p = p->link; ++j;
    }//寻找第 i-1 个结点，并令 p 指向其前趋
    if (!(p->link) || j > i - 1) return FALSE; //若第 i 个元素不存在
    deleted = p->link;//删除并释放结点
    deleted->llink->rlink = deleted->rlink;
    deleted->rlink->llink = deleted->llink;
    item = deleted->data;
    FREE(deleted );
}
```