第二章 线性表

一、选择题

1.一个线性表第一个元素的存储地址是 100,每个元素的长度为 2,则第 5 个元素的地址是( )

（A）110 （B）108（C）100 （D）120

2. 向一个有 127 个元素的顺序表中插入一个新元素并保持原来顺序不变，平均要移动（ ）个元素。

（A）64（B）63 （C）63.5 （D）7

3.线性表采用链式存储结构时，其地址（ ）。

(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) 连续与否均可以

4. 在一个单链表中，若 p 所指结点不是最后结点，在 p 之后插入 s 所指结点，则执行（ ）

（A）s->next=p;p->next=s; （B） s->next=p->next;p->next=s;

（C）s->next=p->next;p=s; （D）p->next=s;s->next=p;

5.在一个单链表中，若删除 p 所指结点的后续结点，则执行（ ）

（A）p->next=p->next->next; （B）p=p->next;

p->next=p->next->next;

（C）p->next=p->next; （D）p =p->next->next;

6.下列有关线性表的叙述中，正确的是（ ）

（A）线性表中的元素之间隔是线性关系

（B）线性表中至少有一个元素

（C）线性表中任何一个元素有且仅有一个直接前趋

（D）线性表中任何一个元素有且仅有一个直接后继

7.线性表是具有 n 个（ ）的有限序列（n≠0)

（A）表元素 （B）字符 （C）数据元素 （D）数据项

二、判断题

1.线性表的链接存储，表中元素的逻辑顺序与物理顺序一定相同。（ ）

2.如果没有提供指针类型的语言，就无法构造链式结构。（ ）

3.线性结构的特点是只有一个结点没有前驱，只有一个结点没有后继，其余的结点只有一个前驱和后继。（ ）

4.语句 p=p->next 完成了指针赋值并使 p 指针得到了 p 指针所指后继结点的数据域值。（ ）

5.要想删除 p 指针的后继结点，我们应该执行 q=p->next ；

p->next=q->next； free(q)。（ ）

三、填空题

1.已知 P 为单链表中的非首尾结点，在 P 结点后插入 S 结点的语句为：

_____ 。

2.顺序表中逻辑上相邻的元素物理位置( )相邻， 单链表中逻辑上相邻的元素物理位置_____相邻。

3.线性表 L = （a1，a2，…，an）采用顺序存储，假定在不同的 n + 1 个位置上插入的概率相同，则插入一个新元素平均需要移动的元素个数是

_____

4.在非空双向循环链表中，在结点 q 的前面插入结点 p 的过程如下：

p->prior=q->prior;

q->prior->next=p;

p->next=q;

_____;

5.已知 L 是无表头结点的单链表，是从下列提供的答案中选择合适的语句序列，分别实现：

（1）表尾插入 s 结点的语句序列是_____

(2) 表尾插入 s 结点的语句序列是_____

1. p->next=s;
2. p=L;
3. L=s;
4. p->next=s->next;
5. s->next=p->next;
6. s->next=L;
7. s->next=null;
8. while(p->next!= Q)? p=p-next;
9. while(p->next!=null) p=p->next;

四、算法设计题

1.试编写一个求已知单链表的数据域的平均值的函数（数据域数据类型为整型）。

2.已知带有头结点的循环链表中头指针为 head,试写出删除并释放数据域值

为 x 的所有结点的 c 函数。

3.某百货公司仓库中有一批电视机，按其价格从低到高的次序构成一个循环链表，每个结点有价格、数量和链指针三个域。现出库（销售）m 台价格为 h 的电视机，试编写算法修改原链表。

4.某百货公司仓库中有一批电视机，按其价格从低到高的次序构成一个循环链表，每个结点有价格、数量和链指针三个域。现新到 m 台价格为 h 的电视机，试编写算法修改原链表。

5.线性表中的元素值按递增有序排列，针对顺序表和循环链表两种不同的存储方式，分别编写 C 函数删除线性表中值介于 a 与 b（a≤b）之间的元素。

6.设 A=(a0,a1,a2,...,an-1),B=(b0,b1,b2,...,bm-1)是两个给定的线性表,它们的结点个数分别是 n 和 m,且结点值均是整数。

若 n=m，且 ai= bi （0≤i<n ），则 A=B;

若 n<m ，且 ai=bi （0≤i<n ），则 A<B;

若存在一个 j， j<m ，j<n ，且 ai=bi （0≤i<j ）， 若 aj<bj，则 A<B，否则 A>B。

试编写一个比较 A 和 B 的 C 函数,该函数返回 -1 或 0 或 1,分别表示 A<B 或 A=B 或 A>B。

7.试编写算法,删除双向循环链表中第 k 个结点。

8.线性表由前后两部分性质不同的元素组成

(a0,a1,...,an-1,b0,b1,...,bm-1),m 和 n 为两部分元素的个数,若线性表分别采用数组和链表两种方式存储,编写算法将两部分元素换位成

(b0,b1,...,bm-1,a0,a1,...,an-1),分析两种存储方式下算法的时间和空间复杂

度。

9.用循环链表作线性表(a0,a1,...,an-1)和（b0,b1,...,bm-1）的存储结构,头指针分别为 ah 和 bh，设计 C 函数，把两个线性表合并成形如（a0,b0,a1,b1,...）的线性表，要求不开辟新的动态空间,利用原来循环链表的结点完成合并操作,结构仍为循环链表,头指针为 head,并分析算法的时间复杂度。

10.试写出将一个线性表分解为两个带有头结点的循环链表，并将两个循环链表的长度放在各自的头结点的数据域中的 C 函数。其中,线性表中序号为偶数的元素分解到第一个循环链表中,序号为奇数的元素分解到第二个循环链表中。

11.试写出把线性链表改为循环链表的 C 函数。

12.已知非空线性链表中 x 结点的直接前驱结点为 y,试写出删除 x 结点的 C 函数。

参考答案：

一、选择题

1. B 2.C 3. D 4. B 5. A 6.A 7、C

二、判断题:

参考答案：1、×2、√3、×4、×5、√

三、填空题

1、s->next=p->next; p->next=s; 2、一定；不一定 3、n/2 4、q->prior=p;

5、(1)6) 3)

(2) 2) 9) 1) 7)

四、算法设计题

1、

```
#include "stdio.h"
#include "malloc.h"
typedef struct node
{int data;
struct node *link;
}NODE;
int aver(NODE *head)
```

```
{int i=0,sum=0,ave; NODE *p;
p=head;
while(p!=NULL)
{p=p->link;++i;
sum=sum+p->data;}
ave=sum/i;
return (ave);}
```

2、

```
#include "stdio.h"
#include "malloc.h"
typedef struct node
{
int data; /*  假设数据域为整型  */
struct node *link;
}NODE;
void del_link(NODE *head,int x) /*  删除数据域为 x 的结点*/
{
NODE *p,*q,*s;
p=head;
q=head->link;
while(q!=head)
{if(q->data==x)
```

```
{p->link=q->link;
s=q;
q=q->link;
free(s);}
else
{
p=q;
q=q->link;
}
}
}
```

3、

```
void del(NODE *head,float price,int num)
{
NODE *p,*q,*s;
p=head;q=head->next;
while(q->price<price&&q!=head)
{
p=q;
q=q->next;
}
if(q->price==price)
```

```
q->num=q->num-num;
else
printf("无此产品");
if(q->num==0)
{
p->next=q->next;
free(q);
}
}
```

4、

```
#include "stdio.h"
#include "malloc.h"
typedef struct node
{
float price;
int num;
struct node *next;
}NODE;
void ins(NODE *head,float price,int num)
{
NODE *p,*q,*s;
p=head;q=head->next;
```

```
while(q->price<price&&q!=head)
{
p=q;
q=q->next;
}
if(q->price==price)
q->num=q->num+num;
else
{
s=(NODE *)malloc(sizeof(NODE));
s->price=price;
s->num=num;
s->next=p->next;
p->next=s;
}
}
```

5、顺序表：

算法思想：从 0 开始扫描线性表，用 k 记录下元素值在 a 与 b 之间的元素个数，对于不满足该条件的元素，前移 k 个位置，最后修改线性表的长度。

```
void del（elemtype list[]，int *n，elemtype a，elemtype b）
{
int i=0，k=0；
```

```
while（i<n)
{
if(list[i]>=a&&list[i]<=b) k++;
else
list[i-k]=list[i];
i++;
}
*n=*n-k; /* 修改线性表的长度*/
}
```

循环链表:

```
void del(NODE *head,elemtype a,elemtype b)
{
NODE *p,*q;
p= head;q=p->link; /* 假设循环链表带有头结点 */
while(q!=head && q->data<a)
{
p=q;
q=q->link;
}
while(q!=head && q->data<b)
{
r=q;
```

```
q=q->link;
free(r);
}
if(p!=q)
p->link=q;
}
```

6、

```
#define MAXSIZE 100
int listA[MAXSIZE],listB[MAXSIZE];
int n,m;
int compare(int a[],int b[])
{
int i=0;
while(a[i]==b[i]&&i<n&&i<m)
i++;
if(n==m&&i==n) return(0);
if(n<m&&i==n) return(-1);
if(n>m&&i==m) return(1);
if(i<n&&i<m)
if(a[i]<b[i]) return(-1);
else if(a[i]>b[i]) return(1);
}
```

7、

```
void del(DUNODE   **head,int i)
{
DUNODE *p;
if(i==0)
{
*head=*head->next;
*head->prior=NULL;
return(0);
}
Else
{for(j=0;j<i&&p!=NULL;j++)
p=p->next;
if(p==NULL||j>i) return(1);
p->prior->next=p->next;
p->next->prior=p->proir;
free(p);
return(0);
}
```

8.

顺序存储:

```
void convert(elemtype list[],int l,int h) /*  将数组中第 l 个到第 h 个元素
```

```
逆置*/
{
int i;
elemtype temp;
for(i=h;i<=(l+h)/2;i++)
{
temp=list[i];
list[i]=list[l+h-i];
list[l+h-i]=temp;
}
}
void exchange(elemtype list[],int n,int m);
{
convert(list,0,n+m-1);
convert(list,0,m-1);
convert(list,m,n+m-1);
}
```

该算法的时间复杂度为 O(n+m),空间复杂度为 O(1)

链接存储:(不带头结点的单链表)

```
typedef struct node
{
elemtype data;
```

```
struct node *link;
}NODE;
void convert(NODE **head,int n,int m)
{
NODE *p,*q,*r;
int i;
p=*head;
q=*head;
for(i=0;i<n-1;i++)
q=q->link; /*q 指向 an-1 结点 */
r=q->link;
q->link=NULL;
while(r->link!=NULL)
r=r->link; /*r 指向最后一个 bm-1 结点 */
*head=q;
r->link=p;
}
```

该算法的时间复杂度为 O(n+m),但比顺序存储节省时间(不需要移动元素,只需改变指针),空间复杂度为 O(1)

9.

```
typedef struct node
{
```

```
elemtype data;
struct node *link;
}NODE;
NODE   *union(NODE   *ah,NODE *bh)
{
NODE   *a,*b,*head,*r,*q;
head=ah;
a=ah;
b=bh;
while(a->link!=ah&&b->link!=bh)
{
r=a->link;
q=b->link;
a->link=b;
b->link=r;
a=r;
b=q;
}
if(a->link==ah) /*a 的结点个数小于等于 b 的结点个数  */
{
a->link=b;
while(b->link!=bh)
```

```
b=b->link;
b->link=head;
}
if(b->link==bh) /*b 的结点个数小于 a 的结点个数 */
{
r=a->link;
a->link=b;
b->link=r;
}
return(head);
}
```

该算法的时间复杂度为 O(n+m),其中 n 和 m 为两个循环链表的结点个数.

10.

```
typedef struct node
{
elemtype data;
struct node *link;
}NODE;
void analyze(NODE *a)
{
NODE  *rh，*qh，*r,*q,*p;
int i=0，j=0；/*i 为序号是奇数的结点个数 j 为序号是偶数的结点个数 */
```

```
p=a;
rh=（NODE *）malloc（sizeof（NODE））；/*rh 为序号是奇数的链表头
指针 */
qh=(NODE *)malloc(sizeof(NODE)); /*qh 为序号是偶数的链表头指针 */
r=rh;
q=qh;
while(p!=NULL)
{
r->link=p;
r=p;
i++;
p=p->link;
if(p!=NULL)
{
q->link=p;
q=p;
j++;
p=p->link;
}
}
rh->data=i;
r->link=rh;
```

```
qh->data=j;
q->link=qh;
}
```

11.

```
typedef struct node
{
elemtype data;
struct node *link;
}NODE;
void change(NODE   *head)
{
NODE   *p;
p=head;
if(head!=NULL)
{
while(p->link!=NULL)
p=p->link;
p->link=head;
}
}
```

12.

```
typedef struct node
```

```
{
elemtype data;
struct node *link;
}NODE;
void del(NODE *x,NODE *y)
{
NODE *p,*q;
elemtype d1;
p=y;
q=x;
while(q->next!=NULL) /*  把后一个结点数据域前移到前一个结点*/
{
p->data=q->data;
q=q->link;
p=q;
p->link=NULL; /* 删除最后一个结点*/
free(q);
}
```