西安电子科技大学
XIDIAN UNIVERSITY

第二章
线性表

LINEAR LIST

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 1

内容大纲

- 2-1 线性表的定义与运算
- 2-2 线性表的顺序存储
- 2-3 线性表的链式存储结构
- 小 结
- 实验2 线性表子系统
- 习题2

数据结构

- 集合结构
- 线性结构 → 线性表
- 树形结构
- 图形结构

多项式的表示

- 【例】一元多项式及其运算

一元多项式：$f(x)=a_0+a_1x+\cdots+a_{n-1}x^{n-1}+a_nx^n$

主要运算：多项式相加、相减、相乘等

【分析】如何表示多项式？

多项式的关键数据：

- 多项式项数n
- 各项系数a_i及指数i

方法1：顺序存储结构直接表示

数组各分量对应多项式各项：

a[i]: 项x^i的系数a_i

例如： $f(x)=4x^5-3x^2+1$

表示成：

下标i	0	1	2	3	4	5	……
a[i]	1	0	–3	0	0	4	……
	1		$-3x^2$			$4x^5$	

两个多项式相加：两个数组对应分量相加

问题：如何表示多项式 $x+3x^{2000}$?

单选题 1分

用顺序存储结构直接表示多项式$x+3x^{2000}$，数组大小需要多少?

A 2

B 2000

C 2001

D 2002

6

方法2：顺序存储结构表示非零项

每个非零项$a_i x^i$涉及两个信息：系数a_i和指数i

可以将一个多项式看成是一个(a_i, i)二元组的集合。

用结构数组表示：数组分量是由系数a_i、指数i组成的结构，对应一个非零项

例如：

$$P_1(x) = 9x^{12} + 15x^8 + 3x^2 \text{ 和 } P_2(x) = 26x^{19} - 4x^8 - 13x^6 + 82$$

下标i	0	1	2	……
系数a^i	9	15	3	–
指数i	12	8	2	–

(a) $P_1(x)$

下标i	0	1	2	3	……
系数a^i	26	–4	–13	82	–
指数i	19	8	6	0	–

(b) $P_2(x)$

按指数大小有序存储！

方法2：顺序存储结构表示非零项

相加过程：从头开始，比较两个多项式当前对应项的指数

P1： (9,12), (15,8), (3,2)

P2： (26,19), (-4,8), (-13,6), (82,0)

P3： (26,19), (9,12), (11,8), (-13,6), (3,2), (82,0)

$$P_3(x) = 26x^{19} + 9x^{12} + 11x^8 - 13x^6 + 3x^2 + 82$$

方法3：链表结构存储非零项

链表中每个结点存储多项式中的一个非零项，包括系数和指数两个数据域以及一个指针域

coef	expon	link

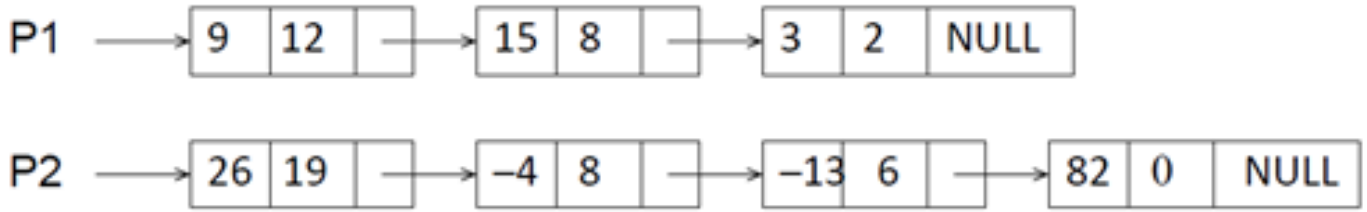
```
typedef struct PolyNode *Polynomial;
struct PolyNode {
        int coef;
        int expon;
        Polynomial link;
}
```

例如：

$P_1(x) = 9x^{12} + 15x^8 + 3x^2$

$P_2(x) = 26x^{19} - 4x^8 - 13x^6 + 82$

链表存储形式为：



GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 9

什么是线性表

多项式表示问题的启示

1. 同一个问题可以有不同的表示（存储）方法

2. 有一类共性问题：有序线性序列的组织和管理

线性表(linear list):具有相同数据类型的 n（n>=0）个数据元素的有限序列，通常记为:

$$(a_1, a_2, \ldots a_{i-1}, a_i, a_{i+1}, \ldots a_n)$$

- 表中元素个数称为线性表的长度
- 线性表没有元素时，称为空表
- 表起始位置称表头，表结束位置称表尾

线性表的定义

在线性表(a_1，a_2，…a_{i-1}，a_i，a_{i+1}，…a_n)中相邻元素之间存在着**顺序关系**。对于元素a_i而言，a_{i-1}称为a_i的**直接前趋**，a_{i+1}称为a_i的**直接后继**。即：

（1）有且仅有一个开始结点（a_1），它没有直接前趋；

（2）有且仅有一个终端结点（a_n），它没有直接后继；

（3）除了开始结点和终端结点以外，其余的结点都有且仅有一个直接前驱和一个直接后继。

线性表举例

（1）简单的线性表

例如一年12个月：

（1，2，3，4，5，6，7，8，9，10，11，12）

在C或C++语言中我们可以把它们定义为数值型。

又例如26个英文字母表：

（a，b，c，d，e，f，g，……，x，y，z）

在C或C++语言中我们可以把它们定义为字符型。

线性表举例

（2）复杂的线性表

例如我们曾经在绪论中引用的一个学生入学情况表（表1-1）可以是用户自定义的学生类型（如C语言中的结构体或数据库管理系统中的记录）。

由于表格中各记录之间存在“一对一”的关系，所以它也是一种线性表。

线性表的二元组表示

- Linearity =（D，R）
- 数据对象：D= {a_i | 1<=i<=n n>=0}
- 数据关系：{<a_{i-1},a_i > | 2<=i<=n} a_{i-1}, a_i∈D

关系中< a_{i-1}, a_i >是一个序偶的集合，它表示线性表中数据元素的相邻关系，即 a_{i-1}领先a_i，a_i领先 a_{i+1}。

- 基本操作：

(1) 创建线性表：CreateList()

初始条件：表不存在

操作结果：构造一个空的线性表

线性表的基本操作

■ 基本操作：

(2) 求线性表的长度：LengthList(L)

初始条件：表L存在

操作结果：返回线性表中的所含元素的个数

(3) 按值查找：SearchList(L,x)， x 是给定的一个数据元素。

初始条件：线性表L存在

操作结果：在表L中查找值为 x 的数据元素，其结果返回在L中首次出现的值为 x 的那个元素的序号或地址，称为查找成功； 否则，在L中未找到值为 x 的数据元素，返回一个特殊值表示查找失败。

(4) 显示操作：ShowList(L)

初始条件：线性表L存在，且非空。

操作结果：显示线性表L中的所有元素。

线性表的基本操作(续)

- 基本操作：

(5) 插入操作：InsList(L,i,x)

初始条件：线性表L存在，插入位置正确 (1<=i<=n+1, n 为插入前的表长)。

操作结果：在线性表L的第 i 个位置上插入一个值为 x 的新元素，这样使原序号为 i , i+1, … , n 的数据元素的序号变为 i+1,i+2, … , n+1，插入后表长=原表长+1。

(6) 删除操作：DelList(L,i)

初始条件：线性表L存在，1<=i<=n。

操作结果：在线性表L中删除序号为i的数据元素，删除后使序号为 i+1, i+2,…, n 的元素变为序号为i, i+1,…,n-1，新表长＝原表长- 1 。

线性表的抽象数据类型描述

类型名称：线性表（List）

数据对象集：线性表是 n (≥0)个元素构成的有序序列($a_1, a_2, \ldots, a_n$)

操作集：线性表L ∈ List，整数i表示位置，元素X ∈ ElementType，
线性表基本操作主要有：

1、List MakeEmpty()：初始化一个空线性表L；

2、ElementType FindKth(int K, List L)：根据位序K，返回相应元素；

3、int Find(ElementType X, List L)：在线性表L中查找X的第一次出现位置；

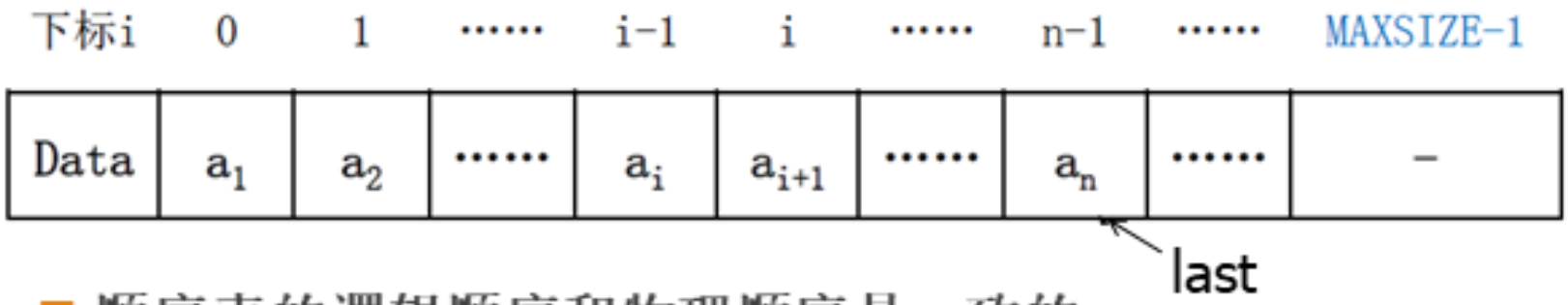
4、void Insert(ElementType X, int i, List L)：在位序i前插入一个新元素X；

5、void Delete(int i, List L)：删除指定位序i的元素；

6、int Length(List L)：返回线性表L的长度n。

线性表的顺序存储

利用数组的连续存储空间顺序存放线性表的各元素



- 顺序表的逻辑顺序和物理顺序是一致的。
- 只要知道顺序表首地址和每个数据元素所占存储单元的个数，就可以求出第i个数据元素的存储地址

设 a_1 的存储地址 $LOC(a_1)$ 为首地址B，每个数据元素占d个存储单元，则第i个数据元素的地址为：

$$LOC(a_i)=LOC(a_1)+(i-1)*d \quad 1<=i<=n$$

即： $LOC(a_i)=B+(i-1)*d$

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 18

线性表的顺序存储

- 在程序设计语言中，一维数组在内存中占用的存储空间就是一组连续的存储区域，可以用一维数组来表示：

```
typedef struct LNode *List;
struct LNode{
        ElementType  Data[MAXSIZE];
        int  Last;
};
struct LNode L;
List  PtrL;
```

- 从结构性上考虑，通常将 Data 和 Last 封装成一个结构作为顺序表的类型
- List为指向该抽象结构Lnode的指针

单选题 1分

如果使用以下结构表示线性表，PtrL为线性表结构的指针，那么该线性表"长度"怎么表示?

A PtrL.Last

B PtrL.Last+1

C PtrL->Last

D PtrL->Last+1

```
typedef struct LNode *List;
struct LNode{
    ElementType  Data[MAXSIZE];
    int Last;
};
List PtrL;
```

20

线性表的顺序存储

- 在程序设计语言中，一维数组在内存中占用的存储空间就是一组连续的存储区域，可以用一维数组来表示：

```
typedef struct LNode *List;
struct LNode{
        ElementType  Data[MAXSIZE];
        int  Last;
};
struct LNode L;
List  PtrL;
```

- 从结构性上考虑，通常将 Data 和 Last 封装成一个结构作为顺序表的类型
- 访问下标为i的元素：L.Data[i] 或 PtrL->Data[i]
- 线性表的长度：L.Last+1 或 PtrL->Last+1

主要操作的实现

1. 初始化（建立空的顺序表）

```
List MakeEmpty( )
{    List PtrL;
     PtrL = (List )malloc( sizeof(struct LNode) );
     PtrL->Last = -1;
     return PtrL;
}
```

查找成功的平均次数为(n+1)/2,平均时间复杂度O(n)

2. 查找

```
int Find( ElementType X, List PtrL )
{   int i = 0;
    while( i <= PtrL->Last && PtrL->Data[i]!= X )
        i++;
    if (i > PtrL->Last)  return -1; /* 如果没找到，返回-1 */
    else  return i;                 /* 找到后返回的是存储位置 */
}
```

主要操作的实现

3. 插入（第$i(1 \le i \le n+1)$个位置上插入一个值为X的新元素）

下标 i	0	1	……	i-1	i	……	n-1	……	MAXSIZE-1
Data	a_1	a_2	……	a_i	a_{i+1}	……	a_n	……	–

Last

先移动，再插入

下标 i	0	1	……	i-1	i	i+1	……	n	……	SIZE -1
Data	a_1	a_2	……	X	a_i	a_{i+1}	……	a_n	……	–

Last

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 23

3. 插入操作实现

```
void Insert( ElementType X, int i, List  PtrL )
{     int j;
      if ( PtrL->Last == MAXSIZE-1 ){      /* 表空间已满，不能插入*/
            printf( " 表满 " );
            return;
      }
      if ( i < 1 || i > PtrL->Last+2) {
            printf( " 位置不合法 " );
            return;
      }
      for ( j = PtrL->Last; j >= i-1; j-- )
            PtrL->Data[j+1] = PtrL->Data[j];  /*将 a_i～ a_n倒序向后移动*/
      PtrL->Data[i-1] = X;                    /*新元素插入*/
      PtrL->Last++;                           /*Last仍指向最后元素*/
      return;
}
```

平均移动次数为n/2,平均时间复杂度为O(n)

单选题 1分

如果把后移数组元素的循环

```
for ( j = PtrL->Last; j >= i-1; j-- )
    PtrL->Data[j+1]=PtrL->Data[j];
```

改为

```
for ( j = i-1; j <= PtrL->Last; j++ )
    PtrL->Data[j+1]=PtrL->Data[j];
```

那会是什么后果？

A 效果一样，没区别

B 分量Data[i-1]到Data[Ptrl->Last+1]都是同一个值，即移之前Data[i-1]的值

C 分量Data[i-1]到Data[Ptrl->Last+1]都是同一个值，即移之前Data[Ptrl->Last]的值

D 说不清楚，要看具体数据

25

主要操作的实现

4. 删除（删除表的第$i(1 \le i \le n)$个位置上的元素）

下标 i	0	1	……	i-1	i	……	n-1	……	MAXSIZE-1
Data	a_1	a_2	……	a_i	a_{i+1}	……	a_n	……	–

Last

后面的元素依次前移

下标 i	0	1	……	i-1	……	n-2	n-1	……	MAXSIZE-1
Data	a_1	a_2	……	a_{i+1}	……	a_n	a_n	……	–

Last

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 26

4. 删除操作实现

```
void Delete( int  i, List  PtrL )
{     int  j;
      if( i < 1 || i > PtrL->Last+1 ) {       /*检
          printf ("不存在第%d个元素", i );
          return ;
      }
      for ( j = i; j <= PtrL->Last; j++ )
          PtrL->Data[j-1] = PtrL->Data[j];    /*将 a(i+1)～ an顺序向前移动*/
      PtrL->Last--;                           /*Last仍指向最后元素*/
      return;
}
```

平均移动次数为(n-1)/2,平均时间复杂度为O(n)

顺序存储的优缺点

■ 顺序存储的优点：

- 可以随机存取表中任意一个元素；
- 存储位置可以用公式：B+(i-1)*d 计算；
- 节约存储空间。

■ 顺序存储的缺点：

- 对顺序表作插入、删除时需要通过移动大量的数据元素，影响了运行效率。
- 线性表预先分配空间时，必须按最大空间分配，存储空间得不到充分的利用。
- 表的容量难以扩充（对有些高级语言而言）。

线性表的链式存储

不要求逻辑上相邻的两个元素物理上也相邻；通过“链”建立起数据元素之间的逻辑关系

- 用一组任意的存储单元存储线性表的数据元素。
- 单链表的每个结点由一个数据域和一个指针域组成：

结点中存放数据元素信息的域称为数据域；存放其后继地址的域称为指针域。

- 单链表的存取必须从头指针开始，最后一个结点的指针为NULL
  - 头指针——指向链表中第一个结点的指针。
  - 头结点——在开始结点之前附加的一个结点。
  - 开始结点——在链表中，存储第一个数据元素（a_1）的结点。

线性表的链式存储

- 在C（或C++）中可以用“结构体指针”来描述：

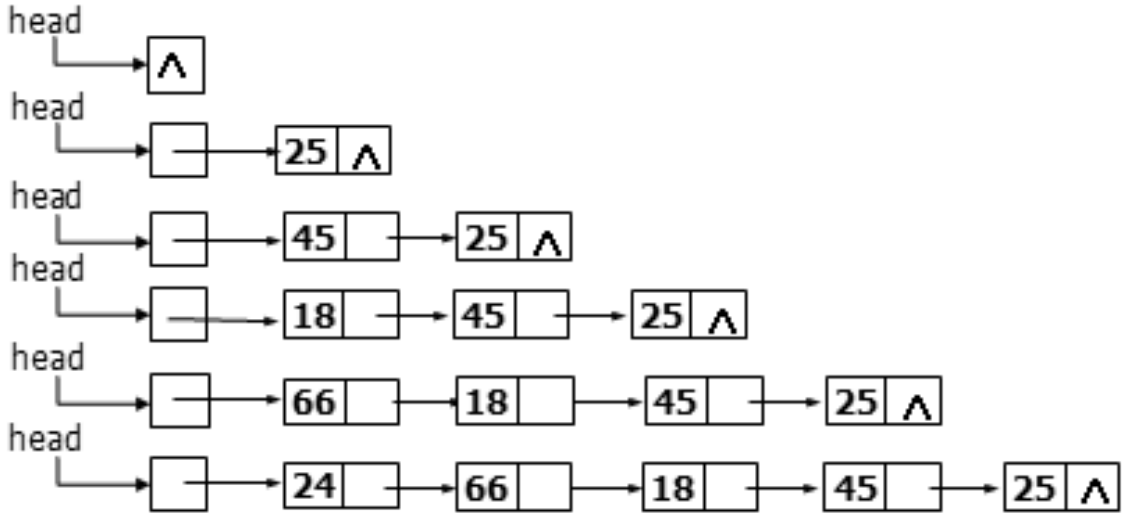
```
typedef struct LNode *List;
struct LNode{
    ElementType  Data;
    List  Next;
};

struct LNode L;
List   PtrL;
```

- 从结构性上考虑，通常将Data和指向struct Lnode结点的指针Next封装成一个结构作为单链表结点的类型
- 头指针定义List PtrL; 当PtrL有定义时，值要么为NULL，则表示一个空表；要么为第一个结点的地址，即链表的头指针

主要操作的实现

1. 建立线性链表

（1）在链表的头部建立线性表的算法（头插法）



主要操作的实现

（1）在链表的头部建立线性表的算法（头插法）

```
void CreateList()                    // 建立线性表
{ node *head,*p,*s;
  char x;
  int z=1,n=0;                              // n用来存储表长，不需要也可略去
  head = NULL;
  printf("\n\t\t建立一个线性表");
  printf("\n\t\t说明：请逐个输入字符，结束标记为"x"！\n");
  while(z)
 {   printf("\t\t输入： ");
     scanf("%c",&x);
     getchar();
     if(x!='x')                       // 输入"x"完成建立
            {
          s=new node;
             n++;                     // n为表长
             s->data=x;
             s->next=head;
             head=s;
            }
            else z=0;                           // 输入循环结束
     }
 }
```

主要操作的实现

（2）在链表的尾插入结点建立线性表算法（尾插法）

```
void CreateList()                    // 建立线性表
{  node *head,*p,*s;
   char x;
   int z=1,n=0;                                  // n用来存储表长，不需要也可略去
   head=NULL;
   p=head;
   printf("\n\t\t建立一个线性表");
   printf("\n\t\t说明：请逐个输入字符，结束标记为"x"！ \n");
   while(z)
    {  printf("\t\t输入： ");
       scanf("%c",&x);
       getchar();
       if(x!='x')                                // 输入"x"完成建立
 {  s=new node;
            n++;                          // n为表长
            s->data=x;
            if(!head)
                  head=s;
            else
                  p=s;
            s->next=NULL;
            p=s;
        }
       else z=0;                                 // 输入循环结束
     }
   }
```

主要操作的实现

2. 求表长

```
int  Length ( List  PtrL )
{    List  p = PtrL;      /* p指向表的第一个结点*/
     int  j = 0;
     while ( p ) {
          p = p->Next;
          j++;                /* 当前p指向的是第 j 个结点*/
     }
     return  j;
}
```

时间复杂度为
O(n)

主要操作的实现

3. 查找

（1）按序号查找FindKth

```
List   FindKth( int K, List  PtrL )
{     List   p = PtrL;
      int  i = 1;
      while (p !=NULL && i < K ){
            p = p->Next;
            i++;
      }
      if ( i == K ) return p;
                  /* 找到第K个，返回指针 */
      else  return NULL;
                        /* 否则返回空*/
}
```

（2）按值查找Find

```
List  Find( ElementType X, List
 PtrL )
{
    List   p = PtrL;
   while ( p!=NULL  && p->Data  != X )
         p = p->Next;
   return p;
}
```

平均时间复杂度为O(n)

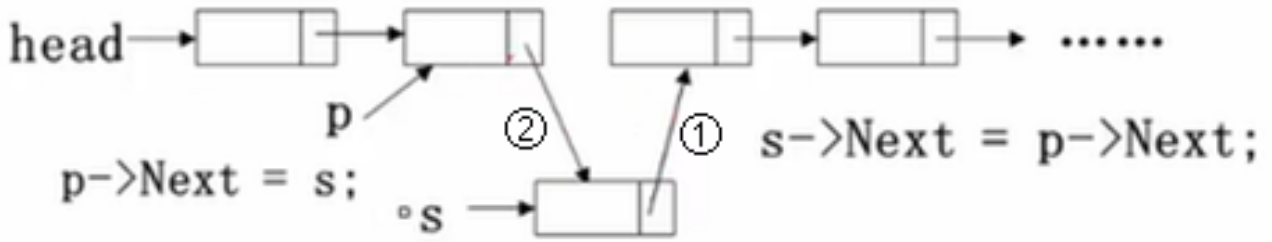
主要操作的实现

4. 插入（在第$i-1(1 \le i \le n+1)$个结点后插入一个值为X的新元素）

（1）先**构造**一个新节点，用s指向；**(malloc)**

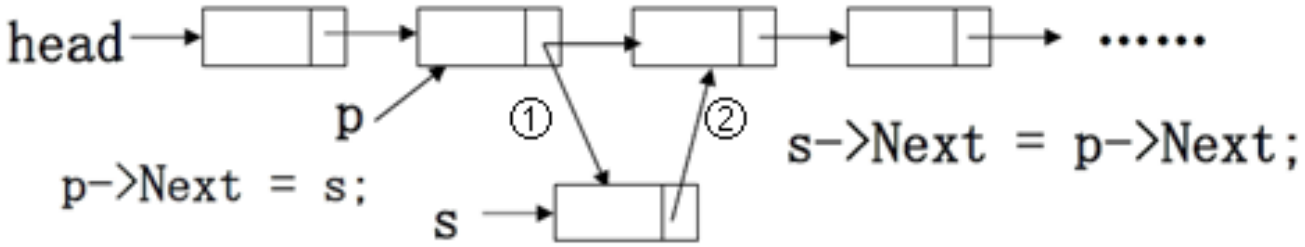
（2）再找到链表的第i-1个结点，用p指向；

（3）然后修改指针，插入结点（p之后插入新结点是s）



单选题 1分

如果语句执行顺序为：(1) p->Next=s; (2) s->Next=p->Next;
那么后果是什么？



A 也能正确插入s结点

B s->Next指向s,从而不能正确完成插入

C p->Next指向p,从而不能正确完成插入

D 说不清楚，直接告诉我答案吧

37

4. 插入操作实现

```
List Insert( ElementType X, int i, List PtrL )
{     List  p, s;
      if ( i == 1 ) {                                   /* 新结点插入在表头 */
          s = (List)malloc(sizeof(struct LNode));       /*申请、填装结点*/
          s->Data = X;
          s->Next = PtrL;
          return s;
      }
      p = FindKth( i-1, PtrL );
      if ( p == NULL ) {
            printf( " 参数i错 " );
            return NULL;
      }else {
            s = (List)malloc(sizeof(struct LNode));     /*申请、填装结点*/
            s->Data = X;
            s->Next = p->Next;                          /*新结点插入在第i-1个结点的后面*/
            p->Next = s;
            return PtrL;
      }
}
```

平均查找次数为n/2,平均时间复杂度为O(n)

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 38

主要操作的实现

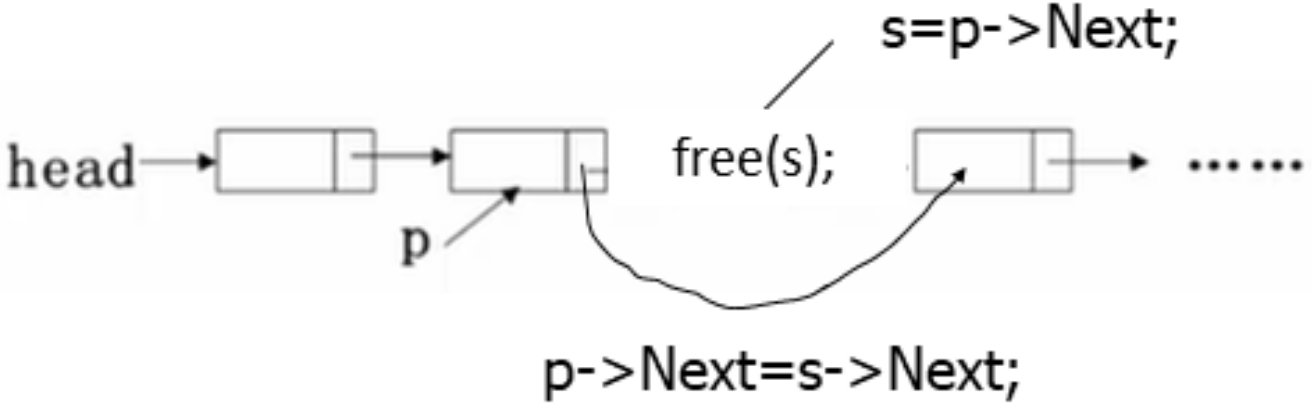
5. 删除（删除链表的第$i(1 \le i \le n)$个位置上的结点）

（1）先找到链表的第i-1个节点，用p指向；

（2）再用指针s指向要被删除的结点（p的下一个结点）；

（3）然后修改指针，删除s所指结点；

（4）最后**释放**s所指结点的空间。**(free)**



5. 删除操作实现

```
List Delete( int  i, List PtrL )
{     List  p, s;
      if ( i == 1 ) {                          /* 若要删除的是表的第一个结点 */
           s = PtrL;                                    /*s指向第1个结点*/
           if (PtrL!=NULL)  PtrL = PtrL->Next;          /*从链表中删除*/
           else return NULL;
           free(s);
           return PtrL;
      }
      p = FindKth( i-1, PtrL );
      if ( p == NULL ) {
           printf("第%d个结点不存在", i-1);     return NULL;
      } else  if ( p->Next == NULL ){
           printf("第%d个结点不存在", i);       return NULL;
      } else  {
           s = p->Next;                                 /*s指向第i个结点*/
           p->Next = s->Next;                           /*从链表中删除*/
           free(s);                                     /*释放被删除结点 */
           return PtrL;
      }
}
```

平均查找次数为n/2,平均时间复杂度为O(n)

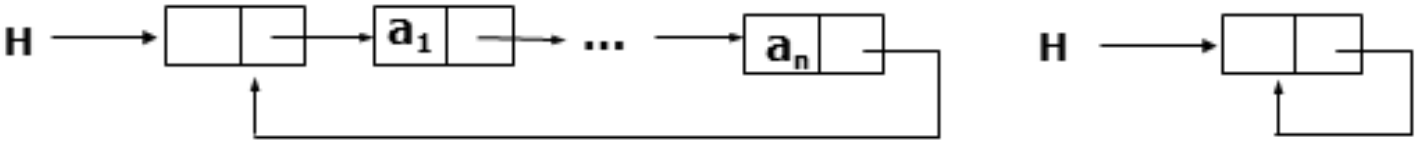
指针的小结

假设p是一个pointer类型，应正确区分指针型变量、指针、指针所指的结点和结点的内容这四个密切相关的不同概念：

- p的值（如果有的话）是一个指针，即是一个所指结点的地址 。
- 该指针（若不是NULL）指向的某个node型结点用*p来标识。
- 结点 *p是由两个域组成的记录，这两个域分别用p→data域和p→next域来标识，它们各有自己的值，p→data的值是一个数据元素，p→next的值是一个指针。

循环链表

■ 单循环链表：将线性单链表中最后一个结点的指针域指向头结点，整个链表头尾结点相连形成一个环，就构成了单循环链表。



（a）非空表

（b）空表

■循环链表上的操作和非循环链表上的操作基本相同，差别在于算法中循环条件不是判断指针是否为空（P->NEXT==NULL），而是判断指针是否为头指针：

P->NEXT==head;

循环链表

- 循环链表设尾指针可以简化某些操作

➢线性链表只能从头结点开始遍历整个链表，而对于单循环链表则可以从表中任意结点开始遍历整个链表

➢对链表常做的操作是在表尾、表头进行，此时可以改变一下链表的标识方法，不用头指针而用一个指向尾结点的指针T来标识，可以使得操作效率得以提高。

举例?

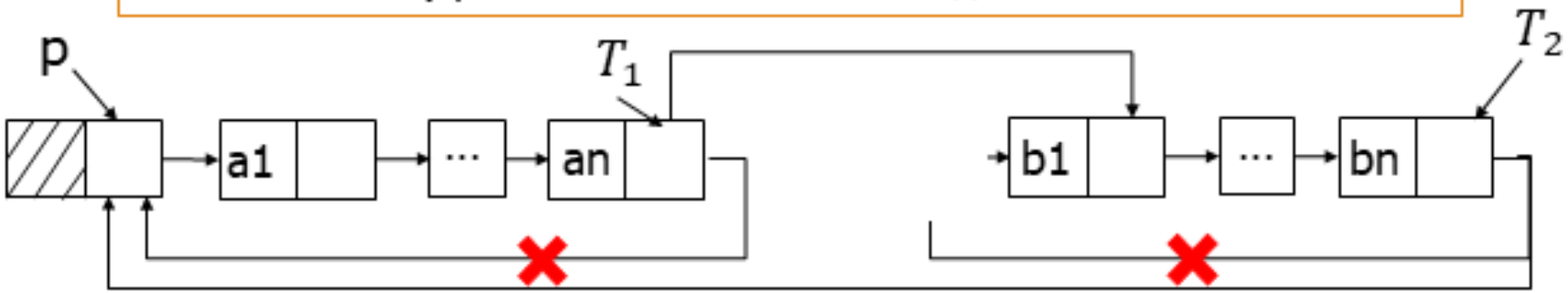
填空题 2分

对两个单循环链表H1 、H2的连接操作，是将H2的第一个数据结点接到H1的尾结点，如用头指针标识，则需要找到第一个链表的尾结点，其时间复杂性为 [填空1] ，而链表若用尾指针T1 、T2来标识，则时间性能为 [填空2] .

44

- 【例】对两个单循环链表H1 、H2的连接操作，是将H2的第一个数据结点接到H1的尾结点，如用头指针标识，则需要找到第一个链表的尾结点，其时间复杂性为O(n)，而链表若用尾指针T1 、T2来标识，则时间性能为O(1)。操作如下：

```
p= T1 ->next;                          //保存T1 的头结点指针
T1->next=T2->next->next;               // 头尾连接
free(T2->next);                        // 释放第二个表的头结点
T2->next=p;                            // 组成循环链表
```



存储密度

（1）存储密度是指结点数据本身所占的存储空间和整个结点结构所占的存储空间之比。即：

$$\text{存储密度 } d = \frac{\text{结点数据占的存储位}}{\text{整个结点实际分配的存储位}}$$

顺序表的存储密度等于1，而链表的存储密度小于1。

（2）采用链式存储比采用顺序存储占用更多的存储空间，是因为链式存储结构增加了存储其后继结点地址的指针域。

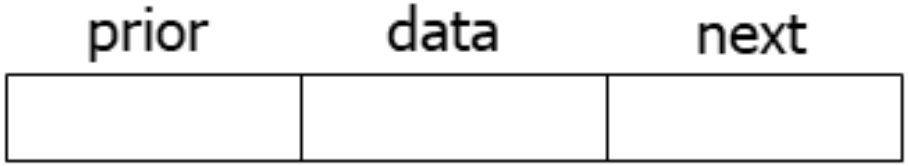
（3）存储空间完全被结点值占用的存储方式称为紧凑存储；否则称为非紧凑存储。显然，顺序存储是紧凑存储，而链式存储是非紧凑存储。存储密度d值越大，表示数据结构所占的存储空间越少。

双向链表

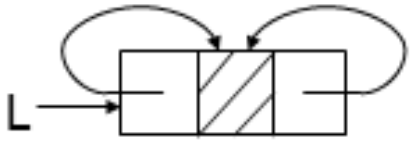
- 单向链表的缺点

单向链表只能顺指针往后寻找其它结点。若要寻找结点的前驱，则需要从表头指针出发。克服上述缺点可以采用双向链表。

- 双向链表：由一个数据域和两个指针域组成。

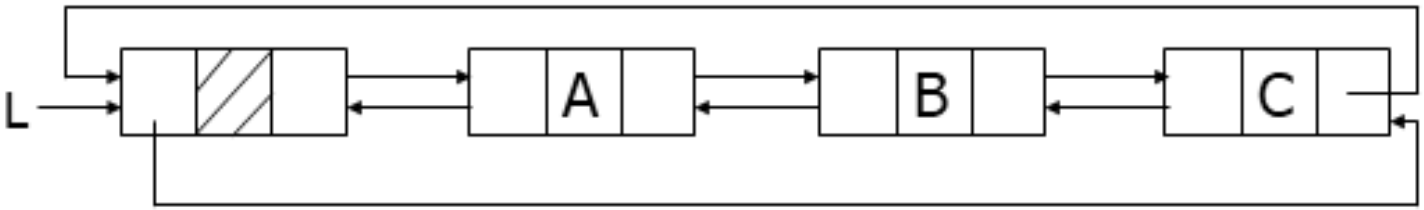


- 空双向链表



双向链表

- 非空双向链表



- 双链表的C（或C++）语言描述

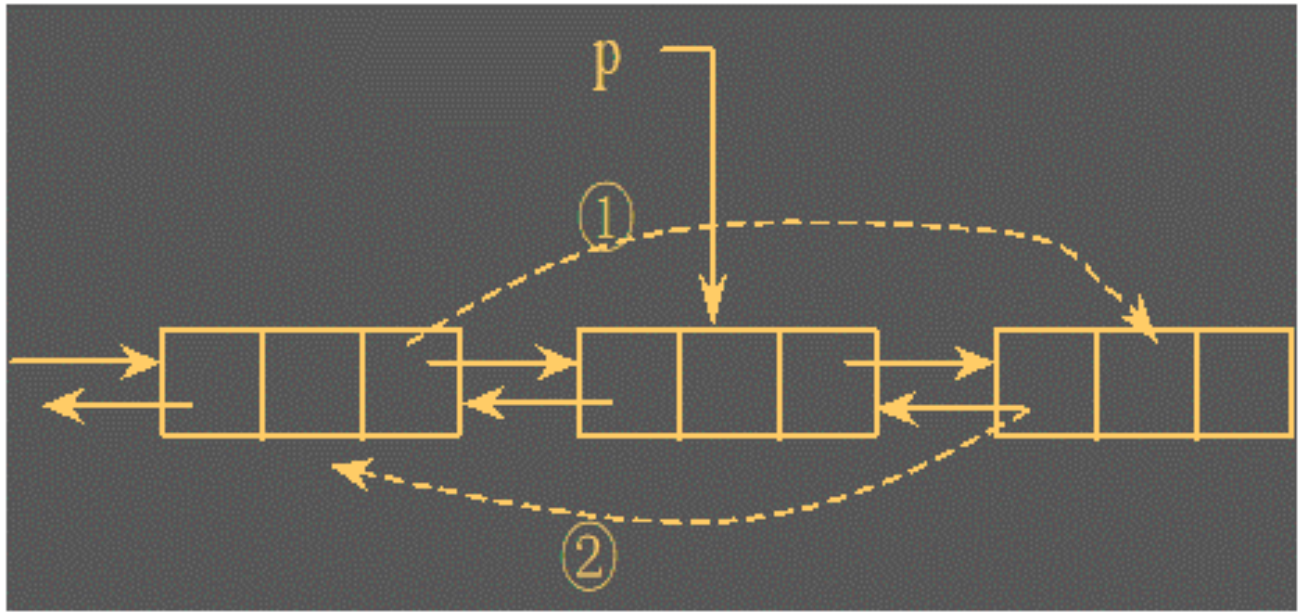
```
typedef struct cdlist *DuLinkList
struct cdlist
  {  datatype data;              //结点数据
     struct cdlist *prior;       //指向先前结点的指针
     struct cdlist *next; }      //指向后继结点的指针
DuLinkList d
```

- d->next->prior=d->prior->next=d

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 48

双向链表的操作

（1）删除结点具体操作描述：

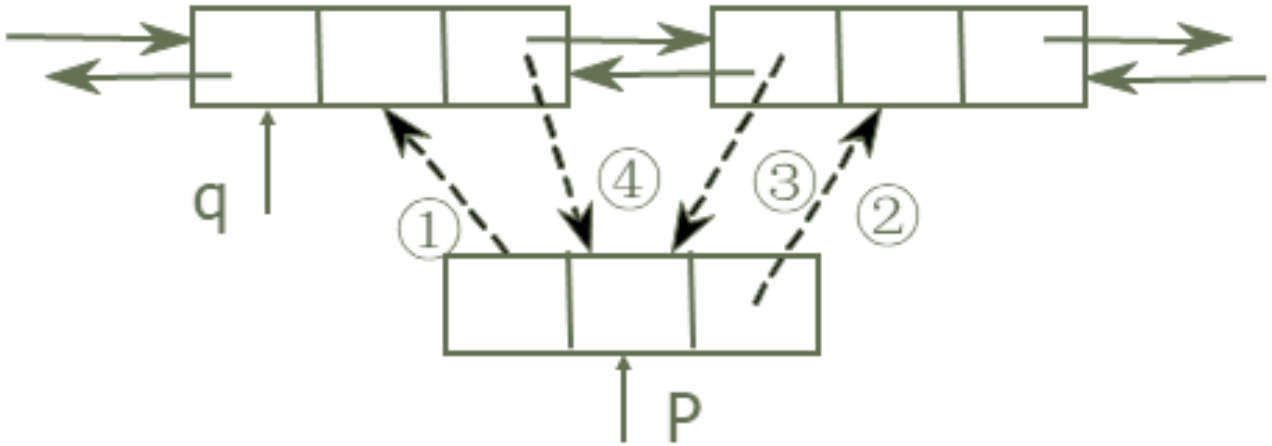


① p->prior->next=p->next;

② p->next->prior=p->prior;

③ delete p;

（2）插入结点具体操作描述：



① p->prior=q;
② p->next=q->next;
③ q->next->prior=p;
④ q->next=p;

小结

- 线性表是一种最简单的数据结构，数据元素之间存在着一对一的关系。其存储方法通常采用顺序存储和链式存储。

- 线性表的顺序存储可以采用结构体的形式，含有两个域。一个整型的长度域，用以存放表中元素的个数；另一个数组域，用来存放元素，其类型可以根据需要而定。顺序存储的最大优点是可以随机存取，且存储空间比较节约，而缺点是表的扩充困难，插入、删除要做大量的元素移动。

- 线性表的链式存储是通过结点之间的链接而得到的。根据链接方式又可以分为：单链表、双链表和循环链表等。

小结

- 单链表有一个数据域（data）和一个指针域（next）组成，数据域用来存放结点的信息；指针域指出表中下一个结点的地址。在单链表中，只能从某个结点出发找它的后继结点。单链表最大的优点是表的扩充容易、插入和删除操作方便，而缺点是存储空间比较浪费。

- 双链表有一个数据域（data）和两个指针域（prior和next）组成，它的优点是既能找到结点的前趋，又能找到结点的后继。

- 循环链表使最后一个结点的指针指向头结点（或开始结点）的地址，形成一个首尾链接的环。利用循环链表将使某些运算比单链表更方便。

广义表

- 【例】我们知道了一元多项式的表示，那么二元多项式又该如何表示？

比如，给定二元多项式：

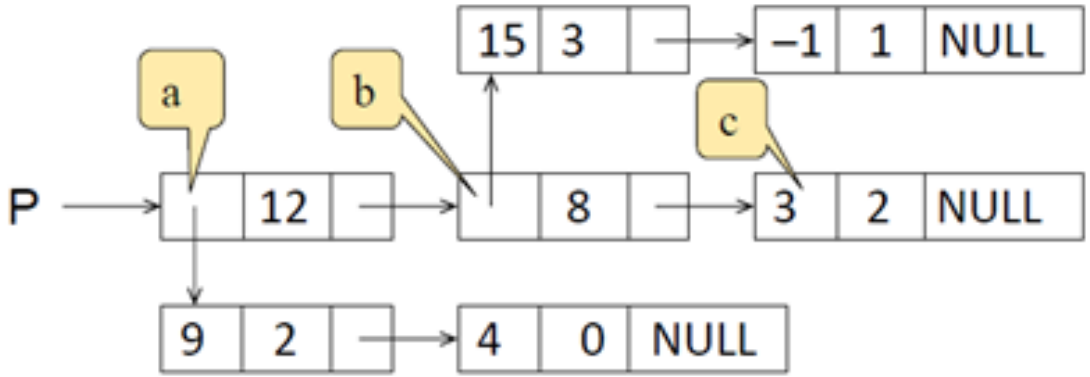
$$P(x,y)=9x^{12}y^2+4x^{12}+15x^8y^3-x^8y+3x^2$$

【分析】可以将上述二元多项式看成关于x的一元多项式

$$P(x,y)=(9y^2+4)x^{12}+(15y^3-y)x^8+3x^2$$

$$ax^{12}+bx^8+cx^2$$

于是，上述二元多项式可以用“复杂”链表表示为：



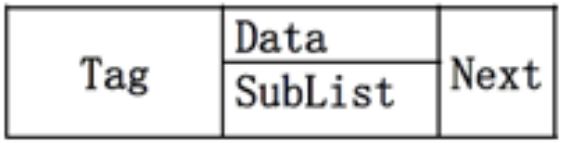
GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 53

广义表

广义表（generalized list）

- 广义表是线性表的推广；
- 对于线性表而言，n个元素都是基本的单元素；
- 广义表中，这些元素不仅可以是单元素也可以是另一个广义表。

```
typedef struct GNode *GList;
struct GNode{
    int Tag;              /*标志域：0表示结点是单元素，1表示结点是广义表 */
    union {               /*子表指针域Sublist与单元素数据域Data复用，即共用存储空间*/
      ElementType  Data;
      GList  SubList;
    } URegion;
    GList Next;           /* 指向后继结点 */
};
```



多重链表

多重链表：链表中的节点可能同时隶属于多个链

- 多重链表中结点的指针域会有多个，如前面例子包含了Next和SubList两个指针域；
- 但包含两个指针域的链表并不一定是多重链表，比如双向链表不是多重链表。
- 多重链表有广泛的用途：

  基本上如树、图这样相对复杂的数据结构都可以采用多重链表方式实现存储。

多重链表

- 【例】矩阵可以用二维数组表示，但二维数组表示有两个缺陷：
  - 一是数组的大小需要事先确定
  - 对于“稀疏矩阵”，将造成大量的存储空间浪费。

$$A=\begin{bmatrix}18 & 0 & 0 & 2 & 0\\ 0 & 27 & 0 & 0 & 0\\ 0 & 0 & 0 & -4 & 0\\ 23 & -1 & 0 & 0 & 12\end{bmatrix} \qquad B=\begin{bmatrix}0 & 2 & 11 & 0 & 0 & 0\\ 3 & -4 & -1 & 0 & 0 & 0\\ 0 & 0 & 0 & 9 & 13 & 0\\ 0 & -2 & 0 & 0 & 10 & 7\\ 6 & 0 & 0 & 5 & 0 & 0\end{bmatrix}$$

【分析】采用一种典型的多重链表——**十字链表**来存储稀疏矩阵

- 只存储矩阵非零元素项

  结点的**数据域**：行坐标Row、列坐标Col、数值Value

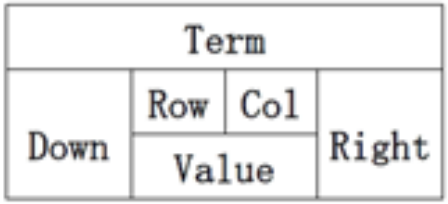
- 每个结点通过两个**指针域**，把同行、同列串起来；

  行指针(或称为向右指针)Right；列指针（或称为向下指针）Down

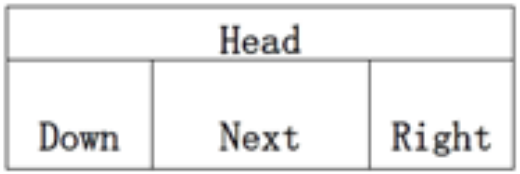
多重链表

- 矩阵A的多重链表
  - 用一个标识域Tag来区分头结点和非0元素结点；
  - 头节点的标识值为“Head”，矩阵非0元素结点的标识值为“Term”。

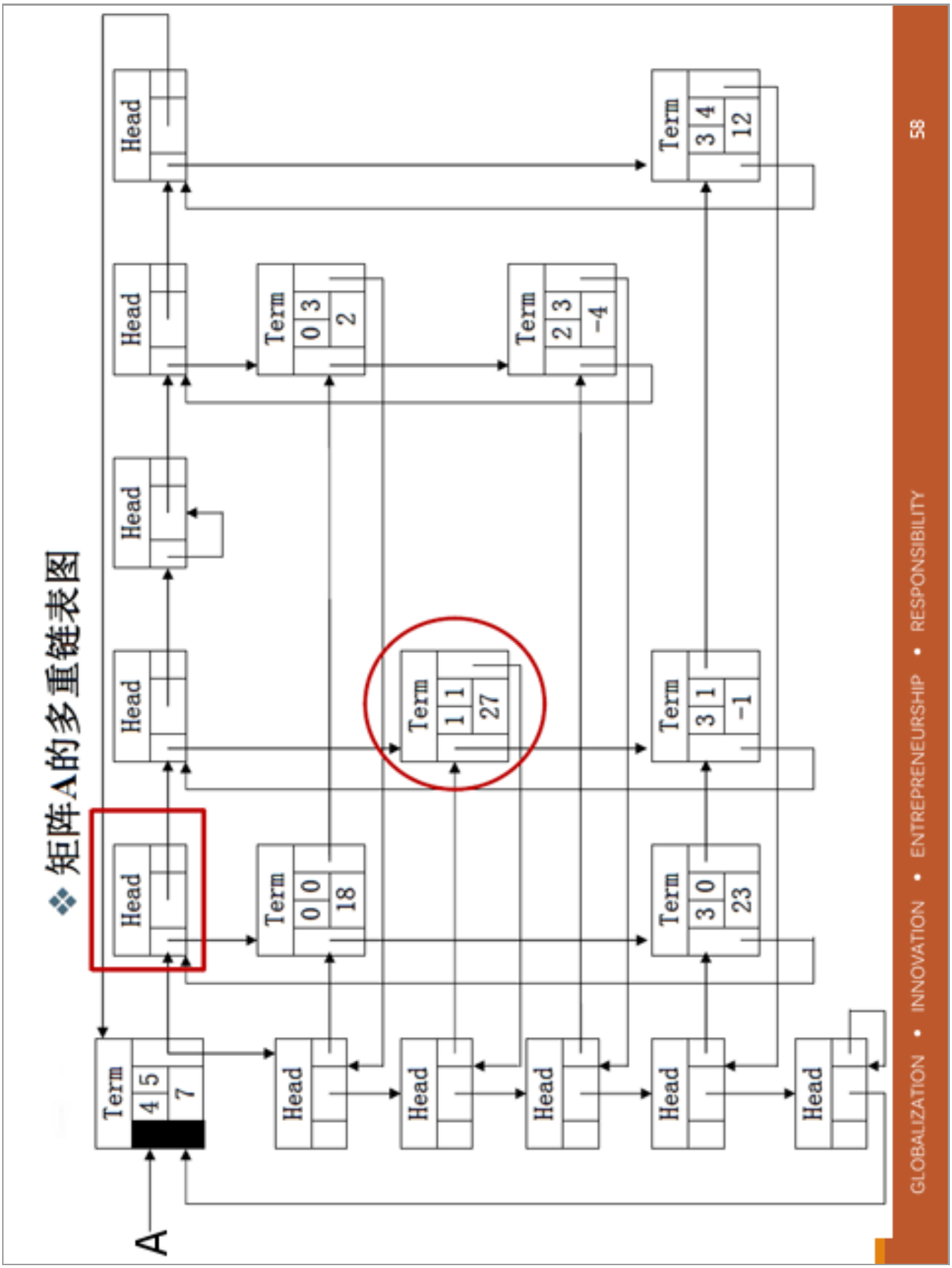
(a) 结点的总体结构



(b) 矩阵非0元素结点



(c) 头结点


矩阵A的多重链表图
A
Term 4 5 7
Head
Term 0 0 18
Term 0 3 2
Term 1 1 27
Term 2 3 -4
Term 3 0 23
Term 3 1 -1
Term 3 4 12
GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY
58

单选题 1分

对于线性表，在顺序存储结构和链式存储结构中查找第k个元素，其时间复杂性分别是多少?

A 都是O(1)

B 都是O(k)

C O(1)和O(k)

D O(k)和O(1)

59

单选题 1分

在顺序结构表示的线性表中，删除第i个元素（数组下标为i-1），需要把后面的所有元素都往前挪一位，相应的语句是：

for (_____)

PtrL->Data[j-1]=PtrL->Data[j];

其中空缺部分的内容应该是

A j = i; j< = PtrL->Last; j++

B j =PtrL->Last; j>= i; j--

C j = i-1; j< = PtrL->Last; j++

D j =PtrL->Last; j>= i-1; j--

60

单选题 1分

下列函数试图求链式存储的线性表的表长，是否正确?

```
int  Length ( List  *PtrL )
{    List  *p = PtrL;
     int  j = 0;
     while ( p ) {
          p++;
          j++;
     }
     return  j;
}
```

A 对

B 错

61

实验2 线性表子系统

1. 实验目的

（1）掌握线性表的特点

（2）掌握线性表顺序存储结构和链式存储结构的基本运算。

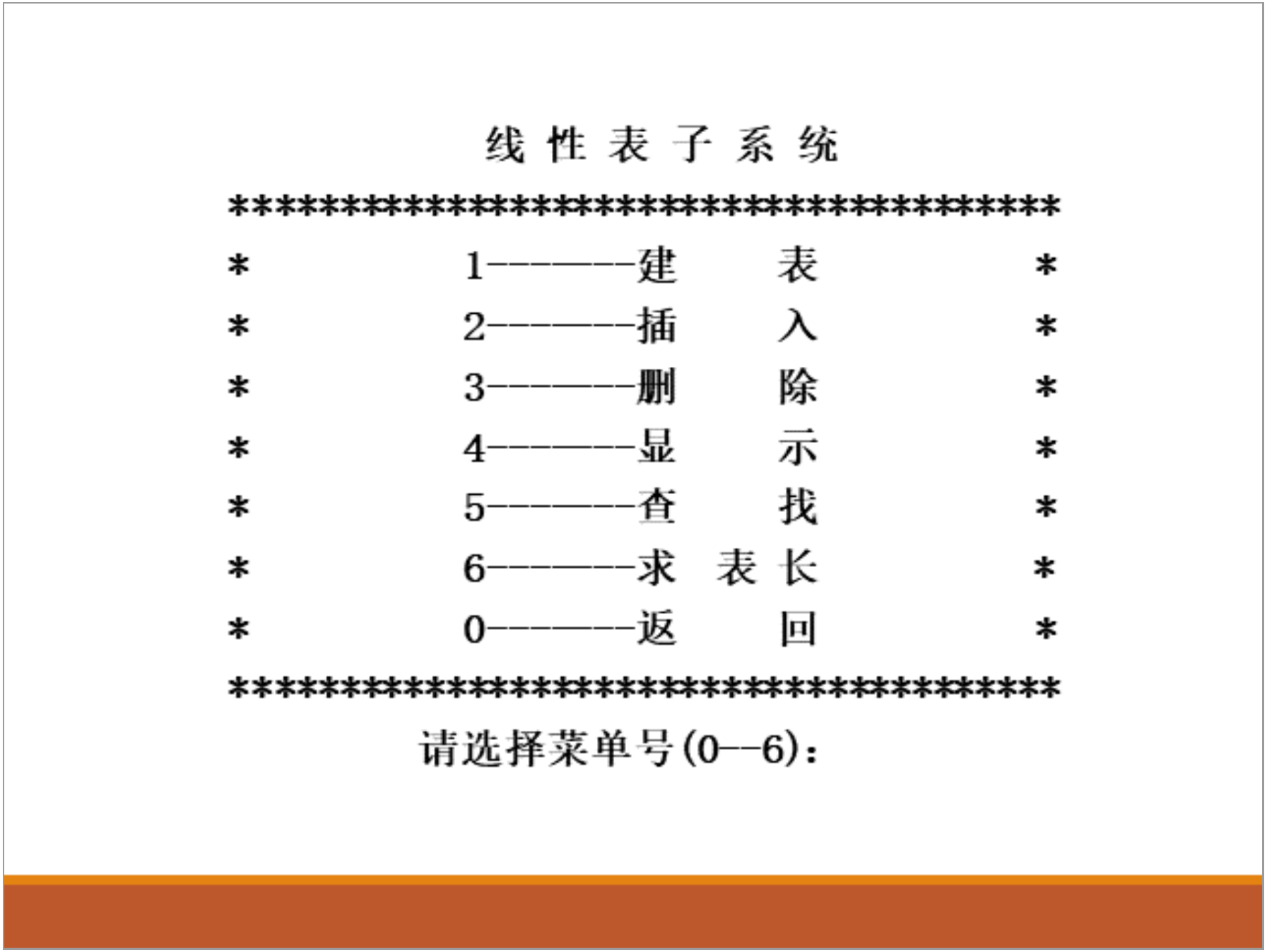
（3）掌握线性表的创建、插入、删除和显示线性表中元素等基本操作。

2. 实验内容

（1）用结构体描述一个字符形的单链表。

（2）创建线性表；在线性表中插入元素、删除元素；显示线性表中所有元素等基本操作。

（3）用if语句设计一个选择式菜单。


线 性 表 子 系 统
****
* 1--------建 表 *
* 2--------插 入 *
* 3--------删 除 *
* 4--------显 示 *
* 5--------查 找 *
* 6--------求 表 长 *
* 0--------返 回 *
****
请选择菜单号(0--6):

习题2

P13 2. 4(4) (7) ;

P15 2. 8 b, d, e

P18 2. 22