

软件技术基础

第三讲



上讲主要内容

- 线性表

- 顺序表

- 运算

 - ◆ 插入

 - ◆ 删除

 - ◆ 定位

本讲主要内容

链表

单链表

双链表

循环链表

链表

用一组任意的存储单元存放线性表的元素，用指针表示元素间的逻辑关系

存储的数据元素的映像由两部分组成，称为**结点** (node)

Data

next


数据域：存放数据
元素信息

指针域：存放相关
元素的存储位置



❖ **N个结点** ($a_i(1 \leq i \leq n)$ 的存储映像) 链结成一个**链表**，也称为**链式**存储结构

链表的分类

- 
- ❖ 单链表：每个结点只包含一个指针域的链表。该指针指向该结点数据元素的直接后继。
 - ❖ 循环链表：单链表中最后一结点的指针域指向头结点，整个链表形成一个环。
 - ❖ 双链表：每个结点包含两个指针域的链表。这两个指针分别指向该结点数据元素的直接前驱和直接后继。

单链表

❖ 右图是线性表（zhao, qian, sun, li, zhou, wu, zheng, wang）的单链表示意图。

❖ 下图是另一种链表的表示形式——称为静态链表

	数据域	指针域
	⋮	⋮
110	zhou	200
	⋮	⋮
130	qian	135
135	sun	170
	⋮	⋮
160	wang	NULL
165	zhao	130
170	li	110
	⋮	⋮
200	wu	205
205	zheng	160
	⋮	⋮

头指针 head 165



用C语言描述单链表

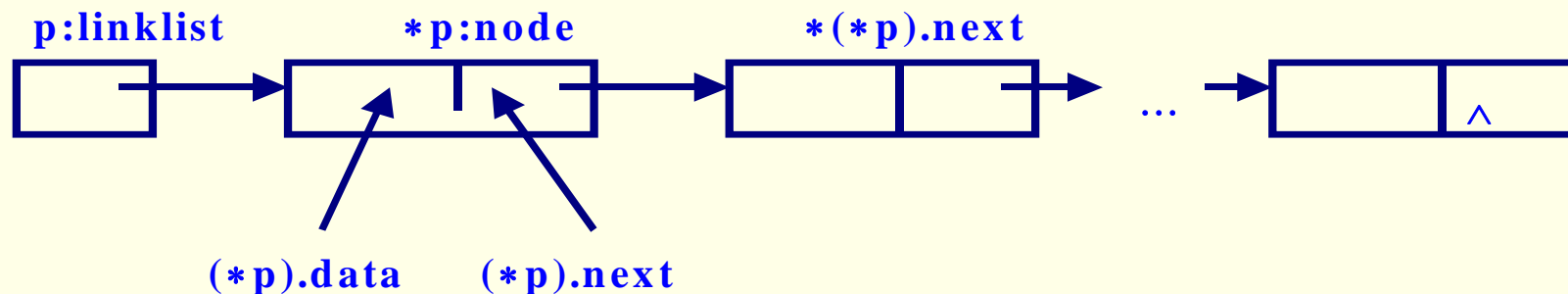
❖ 数据结构定义

```
typedef struct LNode  
{ ElemType data;  
  struct LNode *next;  
} LNode, *linklist;
```

❖ 单链表可由头指针唯一确定

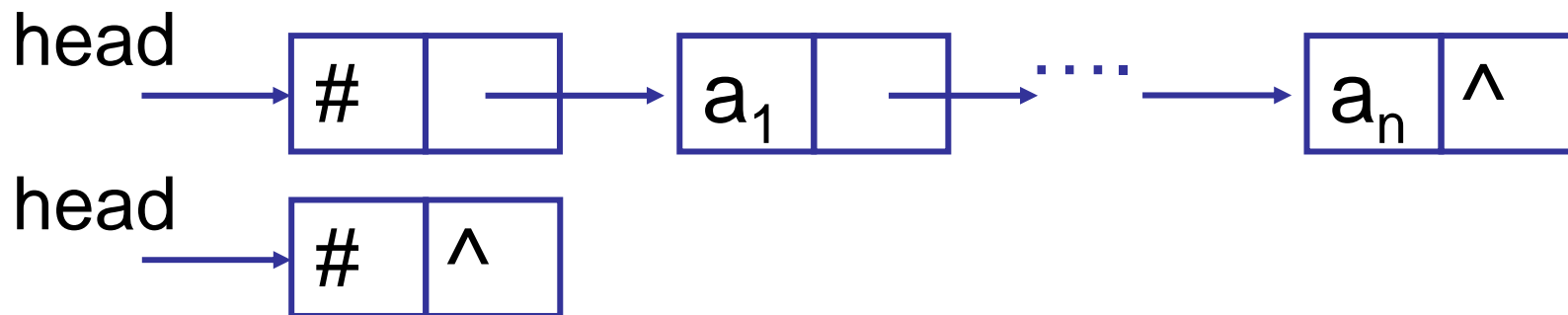
❖ 若L为空，则所表示的线性表为空表，长度为0。

❖ 取得第i个数据必须从头指针出发寻找，它是非随机的存储结构。



带头结点的单链表

- 在单链表的第一个结点之前附设一个结点，称之为头结点。
- 头结点的数据域可以不放任何信息，也可存储如线性表的长度等附加信息。
- 空表的头结点指针域为空



算法2.8GetElem

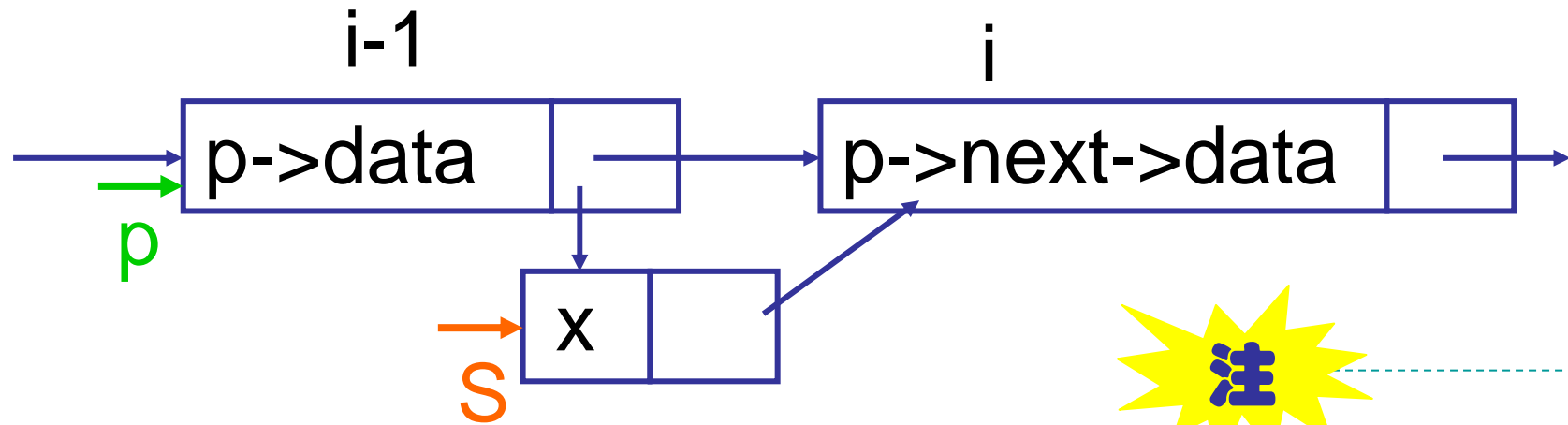
Status GetElem_L(LinkList L,int i,ElemType &e)

/ L为带头结点的单链表的头指针，当第i个元素存在时，其值赋给e并返回OK，否则返回ERROR*/*

```
{ p=L->next; j=1; //初始化，p指向第一个结点，j为计数器
  while (p&& j<i) //顺指针向后查找，直到p指向第i个元素或p为空。
  { p=p->next;
    ++j;
  }
  if (!p||j>i) return ERROR; //第i个元素不存在
  e=p->data; //取第i个元素
  return OK;
} //GetElem_L
```

时间复杂度： $O(n)$

插法图示



[步骤]

1. 找到第 $i-1$ 个结点
2. 生成新结点
3. 输入数据放入新结点数据域
4. 新结点的指针指向 p 结点的直接后继
5. p 结点的指针指向新生产结点

注

3, 4步能否互换

算法2.9 插入

```
Status ListInsert_L(LinkList &L,int i,ElemType e)
```

```
    /* 在带头结点的单链表L的第i个位置之前插入元素e*/
```

```
{ p=L; j=0; //初始化, p指向头结点, j为计数器
```

```
    while (p&& j<i-1) //寻找第i-1个结点
```

```
        { p=p->next;
```

```
          ++j;
```

```
        }
```

```
    if (!p||j>i-1) return ERROR; //i小于1或大于表长
```

```
        s=(LinkList *)malloc(sizeof(Lnode)); //生成新结点
```

```
        s->data=e; s->next=p->next; //插入L中
```

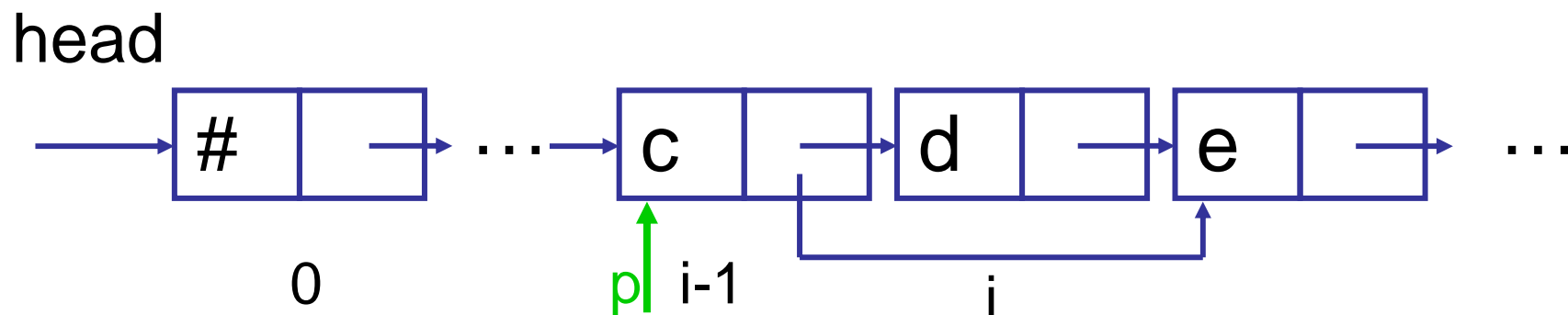
```
        p->next=s;
```

```
        return OK;
```

```
}//ListInsert_L
```

时间复杂度: $O(n)$

第i个结点删除示意图



[步骤]

1. 找到第 $i-1$ 个结点`p`
2. 删除结点`i`, `p->next=p->next->next;`
3. 释放第 i 结点 ;

算法2.10 删除

Status ListDelete_L(LinkList &L,int i,ElemType &e)

/ 在带头结点的单链表L中，删除第i个元素，并由e返回其值*/*

{ p=L; j=0; //初始化，p指向头结点，j为计数器

while (p&& j<i-1) //寻找第i-1个结点

{ p=p->next;

++j;

}

if (!p||j>i-1) return ERROR; //i小于1或大于表长

q=p->next; p->next=q->next; //插入L中

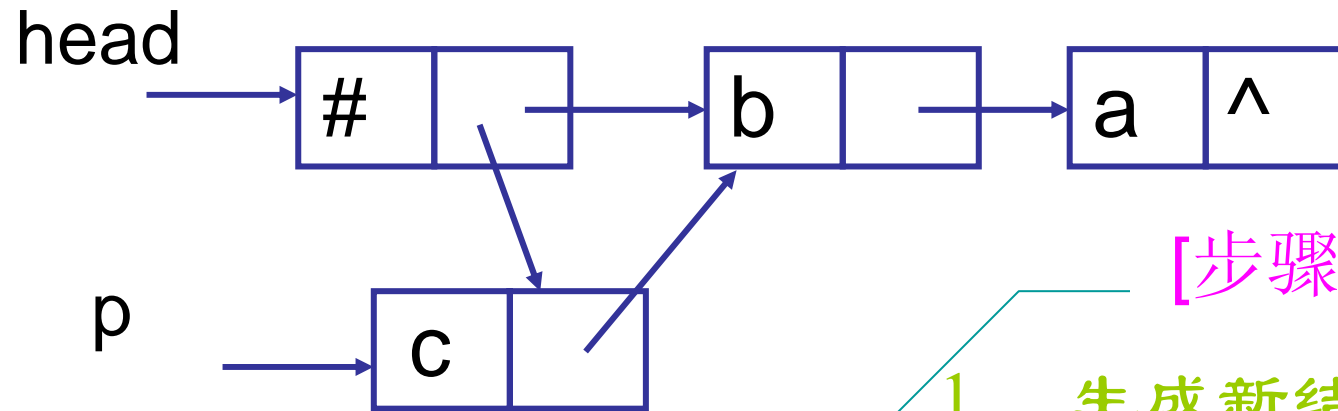
e=q->data; free(q);

return OK;

}//ListDelete_L

时间复杂度： $O(n)$

头插法建表图示



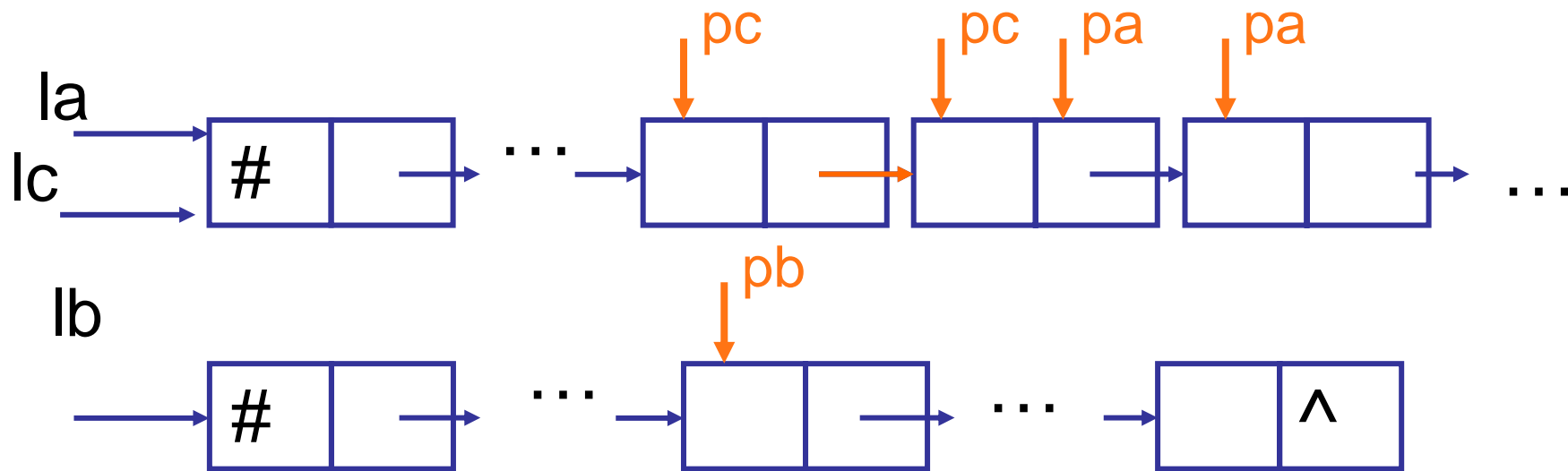
[步骤]

1. 生成新结点
2. 输入数据放入新结点数据域
3. 新结点插入表头

算法2.11 建立单链表

```
void CreatList_L(LinkList &L,int n)
    /* 逆位序输入n个元素的值，建立带表头结点的单链表Le*/
{   L=(LinkList *)malloc(sizeof(Lnode));
    L->next=NULL; //先建立一个带头结点的单链表点
    for (i=n;i>0;--i)
    {   p= (LinkList *)malloc(sizeof(Lnode)); //生成新结点
        scanf(&p->data); //输入元素值
        p->next=L->next; //插入到表头
        L->next=p;
    }
} //ListInsert_L
```

递增单链表合并示意图



[情况1] $*pa$ 结点数据小于 $*pb$ 结点数据

1. pc 的指针域指向 $*pa$ 结点;
2. pc 指针 后移
3. pa 指向 $*pa$ 的直接后继;

算法2.12 合并有序链表

```
void Mergelist_ L(LinkList &La,  
    LinkList &Lb, List &Lc)  
  
{ pa=La->next;  
  pb=Lb->next;  
  Lc=pc=La;  
  while (pa&&pb)  
      { if (pa->data<=pb->data)  
          { pc->next<=pa;  
            pc=pa;  
            pa=pa->next;  
          }  
      }
```

/* 已知线性表**la**和**lb**中的数据元素按值非递减排列。归并**la**和**lb**得到新的线性表**Lc**, **Lc**的数据元素也按值非递减排列。*/

算法2.2

合并有序线性表

```
else
    { pc->next<=pb;
      pc=pb;
      pb=pb->next;
    }
pc->next=pa?pa:pb;
free(Lb);
} //MergeList_Sq
```

//插入的剩余元素
//释放占用单元

用一维数组描述单链表

❖ 用于没有指针的语言

❖ 数据结构定义

```
#define MAXSIZE 1000  
  
typedef struct  
{ ElemType data;  
  int cur;  
}component,  
SLinklist [MAXSIZE];
```

// 链表可能的最大长度

//结点在直接后继在数组中的位置

静态链表示

0		1
1	zhao	2
2	qian	3
3	sun	4
4	li	5
5	zhou	6
6	wu	7
7	zheng	8
8	wang	0
9		
10		

修改前状态

0		1
1	zhao	2
2	qian	3
3	sun	4
4	li	9
5	zhou	6
6	wu	7
7	zheng	8
8	wang	0
9	shi	5

在li后插入shi

0		1
1	zhao	2
2	qian	3
3	sun	4
4	li	9
5	zhou	6
6	wu	8
7	zheng	8
8	wang	0
9	shi	5

删除zheng

算法2.13 定位

int LocateElem_S L(SLinkList S, ElemType e) /*在静态单链表L中查找第1个值为e的元素的位置，若找到，则返回其在L中的序位，否则返回0*/

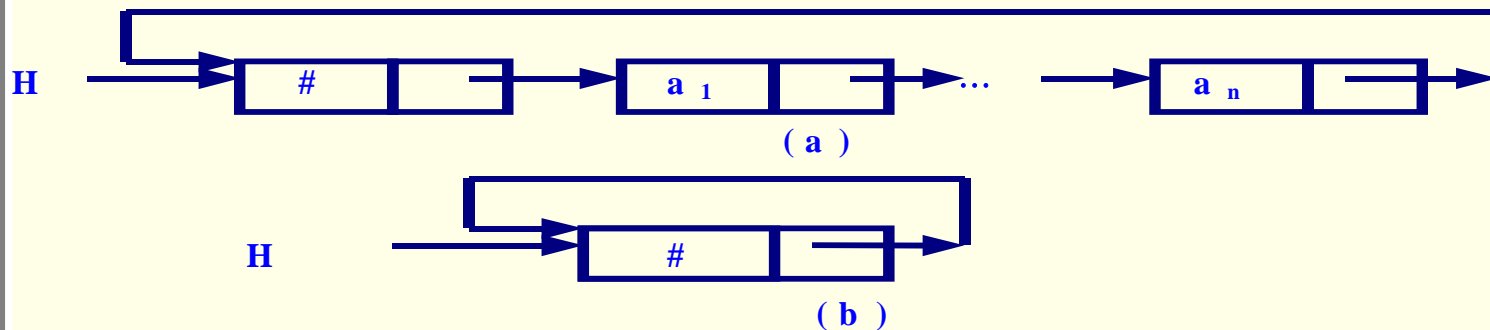
```
{ i=S[0].cur; //i指示表中的第1个结点
  while(i&& S[i].data!=e) i=S[i].cur;
  i;
} //LocateElem_SL
```

循环链表

循环链表是单链表中最后一结点的指针域指向头结点，整个链表形成一个环。

特点：从表中任意结点出发均可找到表中其它结点。

循环链表和单链表的**区别**：循环链表的运算和单链表基本一致，差别仅在于算法中对**最后一个结点**的循环处理上有所不同。



算法 GetElem

Status GetElem_LoopL(LinkList L,int i,ElemType &e)

/* L为带头结点的单链表的头指针，当第i个元素存在时，其值赋给e并返回OK，否则返回ERROR*/

{ p=L->next; j=1; //初始化，p指向第一个结点，j为计数器

while (p!=L &&j<i) /*顺指针向后查找，直到p指向第i个元素或p为空。*/

{ p=p->next;

++j;

}

if (P==L ||j>i) return ERROR; //第i个元素不存在

e=p->data; //取第i个元素

return OK;

}//GetElem_L

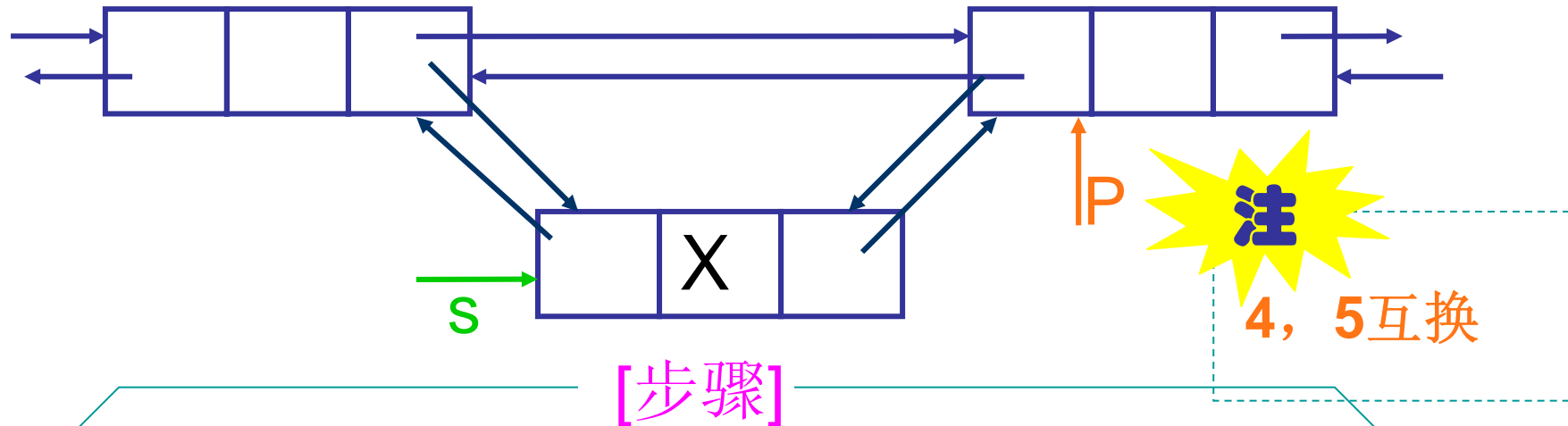
双向链表

定义

在双向链表的结点中有两个指针域，其一指向直接后继，另一指向直接前趋，可描述如下：

```
typedef struct DulNode
{
    ElemType data;
    struct DulNode *prior, *next;
} DulNode *DulLinkList;
```


双链表插入图示



[步骤]

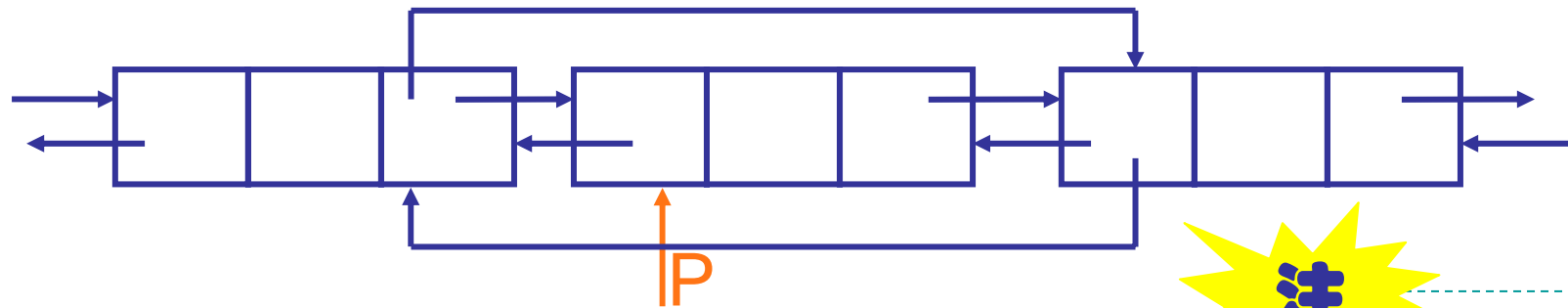
1. 生成新结点；数据放到新结点的数据域中。
2. $*s$ 结点的直接前驱指向 $*p$ 的直接前驱；
3. $*s$ 结点的直接后继指向 $*p$ 的直接后继。
4. $*p$ 结点的直接前驱的直接后继指向 $*s$ 结点。
5. $*p$ 结点的直接前驱指向 $*s$ 结点；

算法2.18 插入

```
Status ListInsert_DuL(DuLinkList L,int i,ElemType e)  
    /* 在带头结点的双链表L的第i个位置之前插入元素e*/  
{ if (!(p=GetElemP_DuL(L,i))) return ERROR;  
    //确定插入位置的指针p  
if (! s=(LinkList *)malloc(sizeof(Lnode)))) return ERROR;  
    s->data=e;  
    s->prior=p->prior;  
    s->prior->next=s;  
    s->next=p; //插入L中  
    p->next=s;  
    return OK;  
}//ListInsert_DuL
```

时间复杂度： $O(n)$

双链表删除图示



注

1, 2互换

[步骤]

1. $*p$ 结点的直接前驱指向 $*p$ 的直接后继
2. $*p$ 结点的直接后继指向 $*p$ 的直接前驱

算法2.19 删除

Status ListDelete_DuL(LinkList L,int i,ElemType &e)

/ 在带头结点的单链表L中，删除第i个元素，并由e返回其值*/*

{ if (!(p=GetElemP_DuL(L,i))) return ERROR;

//确定插入位置的指针p

e=p->data; *//i小于1或大于表长*

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

p->next->prior=p->prior; *//插入L中*

free(p);

return OK;

}//ListDelete_DuL

时间复杂度： $O(n)$

一元多项式的表示及相加

❖ 一般情况下的一元 n 次多项式如右图。其中， p_i 是指数为 e_i 的项的非零系数，且满足：
 $0 \leq e_1 < e_2 < \dots < e_m = n$

$$p_n(x) = p_1x^{e_1} + p_2x^{e_2} + \dots + p_mx^{e_m}$$

❖ 结点类型说明如下：

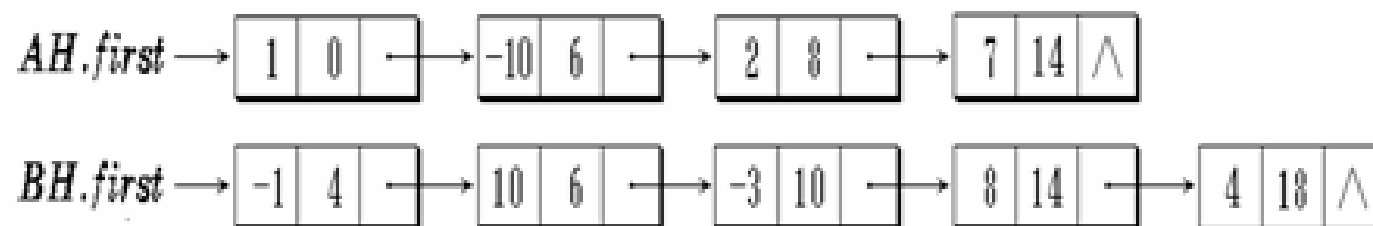
```
typedef struct pnode  
{ float coef;  
  int expn;  
  struct pnode *next;  
}polynode;
```

```
/* 系数 */  
/* 指数 */
```

例 多项式链表的相加

$$AH = 1 - 10x^6 + 2x^8 + 7x^{14}$$

$$BH = -x^4 + 10x^6 - 3x^{10} + 8x^{14} + 4x^{18}$$



(a) 两个相加的多项式



(b) 相加结果的多项式

多项式相加的运算过程

- 结果多项式另存
- 扫描两个相加多项式，若都未检测完：
 - 若当前被检测项指数相等，系数相加。若未变成 **0**，则将结果加到结果多项式。
 - 若当前被检测项指数不等，将指数小者加到结果多项式。
- 若有一个多项式已检测完，将另一个多项式剩余部分复制到结果多项式。

小结

链表

线性表的链式存储结构

单链表

只有一个指针域的链表

运算

建立

查找

插入

删除

用一维数组描述单链表

双链表

循环链表

作业

P16

**10; 11; 13; 14; 15;
17; 19; 20; 24; 26**