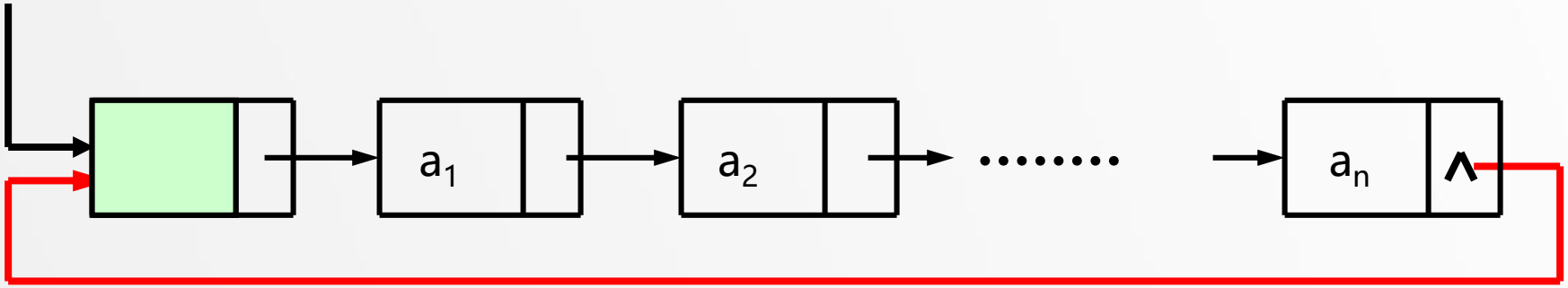
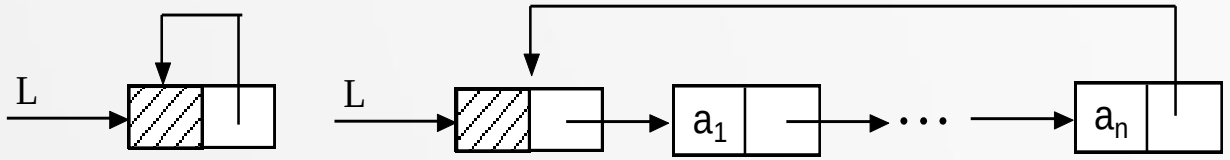
四、单链表的一些变形

1. 循环单链表

最后一个结点的指针域的指针又指回第一个结点的链表



和单链表的差别仅在于，**判别**链表中最后一个结点的**条件**不再是“后继是否为空”，而是“**后继是否为头结点**”。



(a)空表　　(b)非空表

四、单链表的一些变形

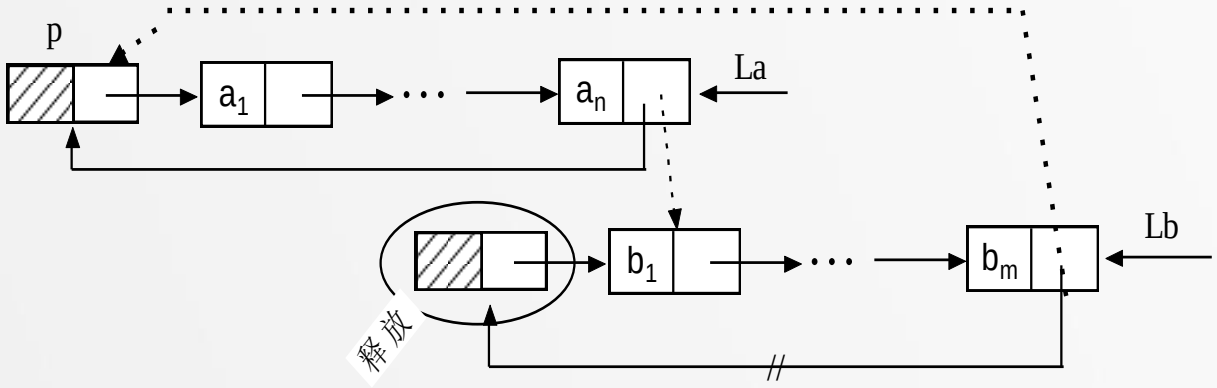
2. 带尾指针的循环单链表

什么用?

四、单链表的一些变形

2. 带尾指针的循环单链表

应用情形1：将两个循环单链表合并




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

四、单链表的一些变形

2. 带尾指针的循环单链表

应用情形1：将两个循环单链表合并

应用情形2：经常需要在最后一个元素后面添加新元素和

在第一个元素前面插入新元素

四、单链表的一些变形

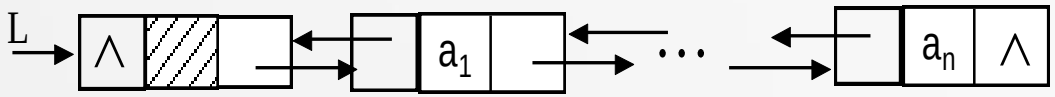
3. 双向链表

- 单链表找后继很方便，找前驱很复杂
- 另用一个空间/指针域来存放前驱的指针

```
typedef struct duNode{
    ElemType elem;
    struct duNode *prior,*next;
}DuNode, *DuNodePtr, **DuList;
```

(a)结点结构

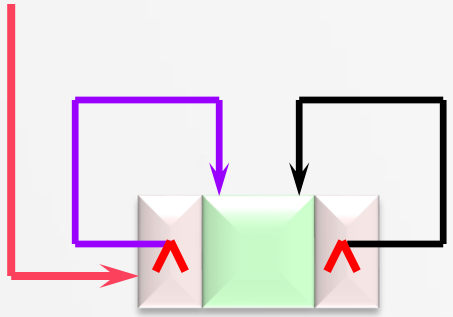
(b)空表



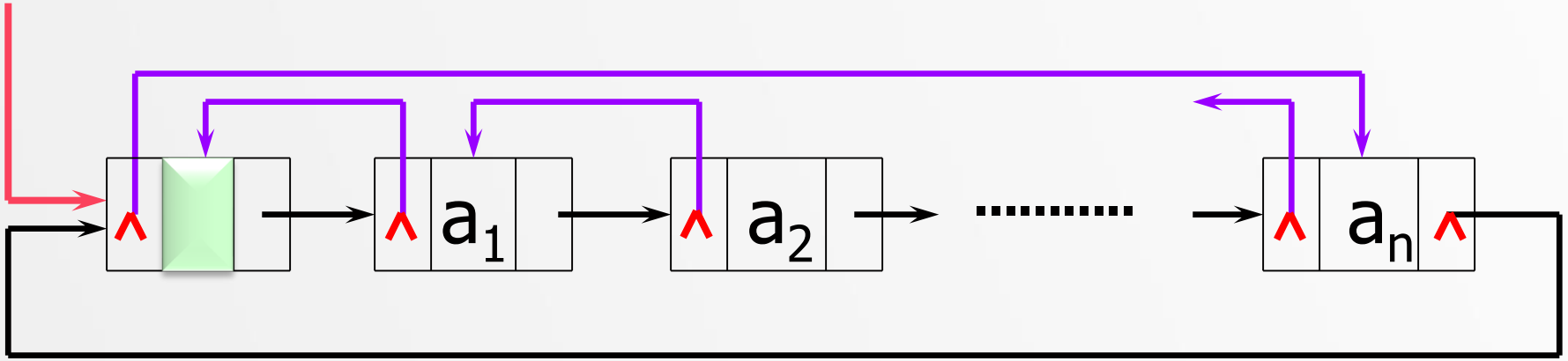
(c)非空表

3. 双向链表

空表



非空表

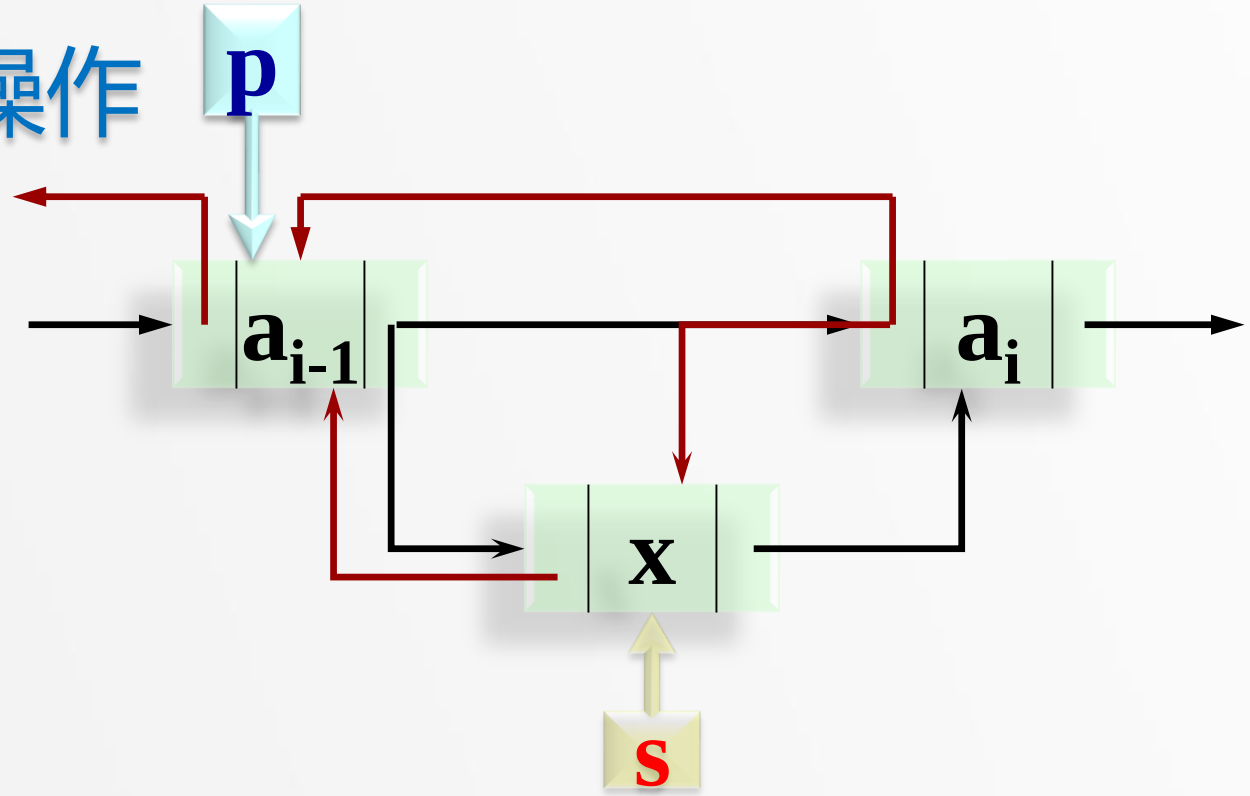


双向链表的操作特点：

“查询” 和单链表相同。

“插入” 和 “删除” 时需要同时修改两个方向上的指针。

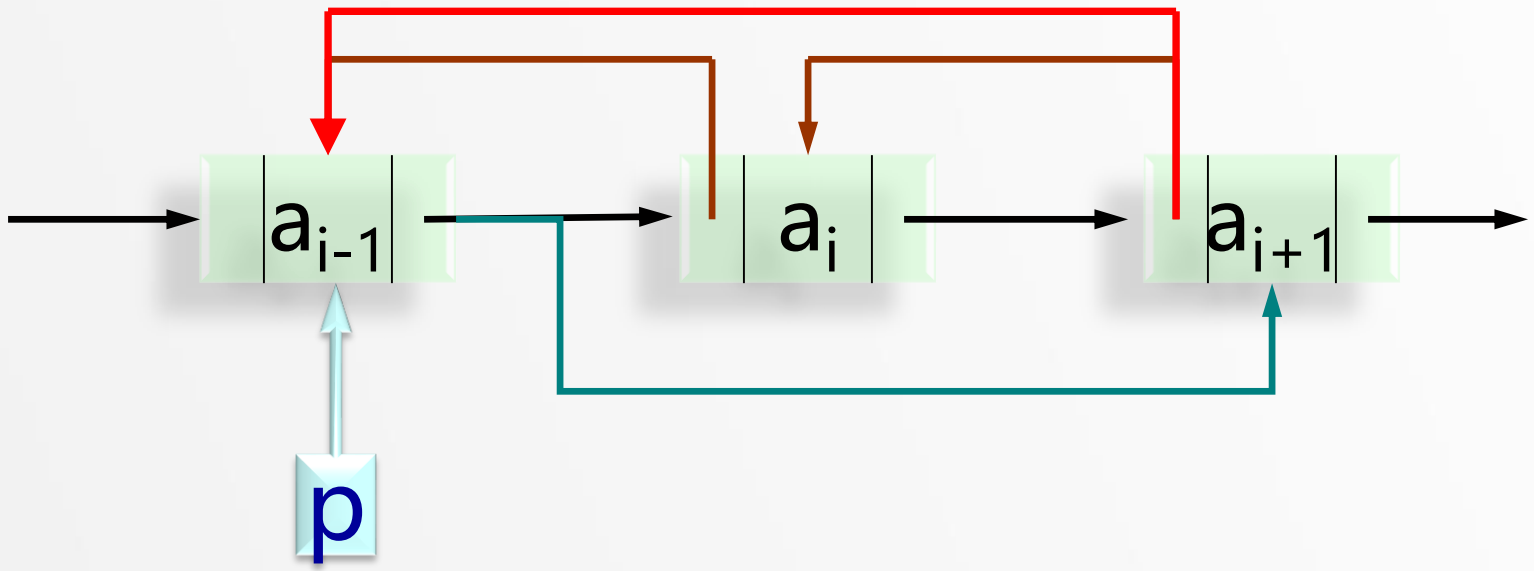
双向链表插入操作



s->next = p->next;　p->next = s;

s->next->prior = s;　s->prior = p;

双向链表删除操作




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

4.静态链表

- 某些语言中不提供指针，如Java和Visual BASIC等，则只能通过其他方式来模拟指针
- 采用数组模拟链表的指针，用以表示数据元素后继所存放位置
  - 数据元素的存储空间像顺序表一样是事先静态分配的
  - 数据元素之间的关系像链表一样是显示的

		data	Next
head=0	0		4
	1	a_4	5
	2		9
	3	a_3	1
	4	a_1	8
	5	a_5	-1
av=6	6		7
	7		2
	8	a_2	3
	9		10
	10		11
	11		-1