

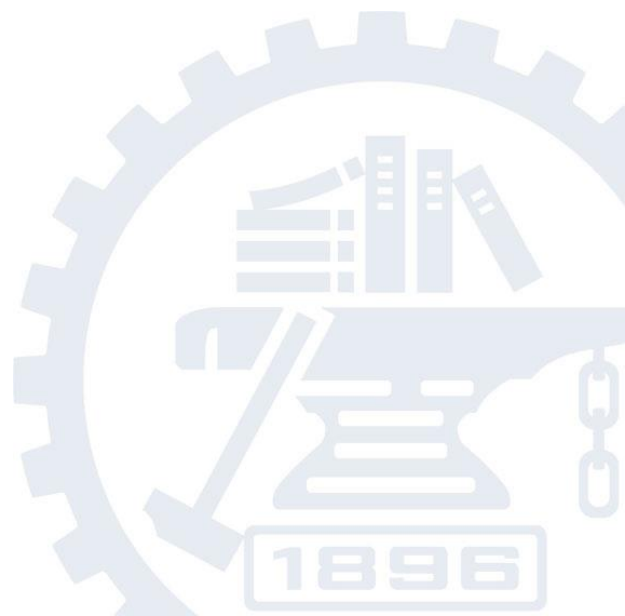


北京交通大学  
BEIJING JIAOTONG UNIVERSITY



# 高级程序设计训练

## SPT-02 线性表





- 2.1 线性表的概念和特点
- 2.2 顺序表的结构体定义及基本操作算法
- 2.3 链表的结构体定义及基本操作算法
- 2.4 线性表的应用**



## 例1——线性表合并问题



### 1.1 集合合并

#### 问题描述：

假设利用两个线性表LA和LB分别表示两个集合A和B，  
现要求一个新的集合

$$A = A \cup B$$

LA=(3, 5, 8, 11)

LB=(2, 6, 8, 9, 11, 15, 20)



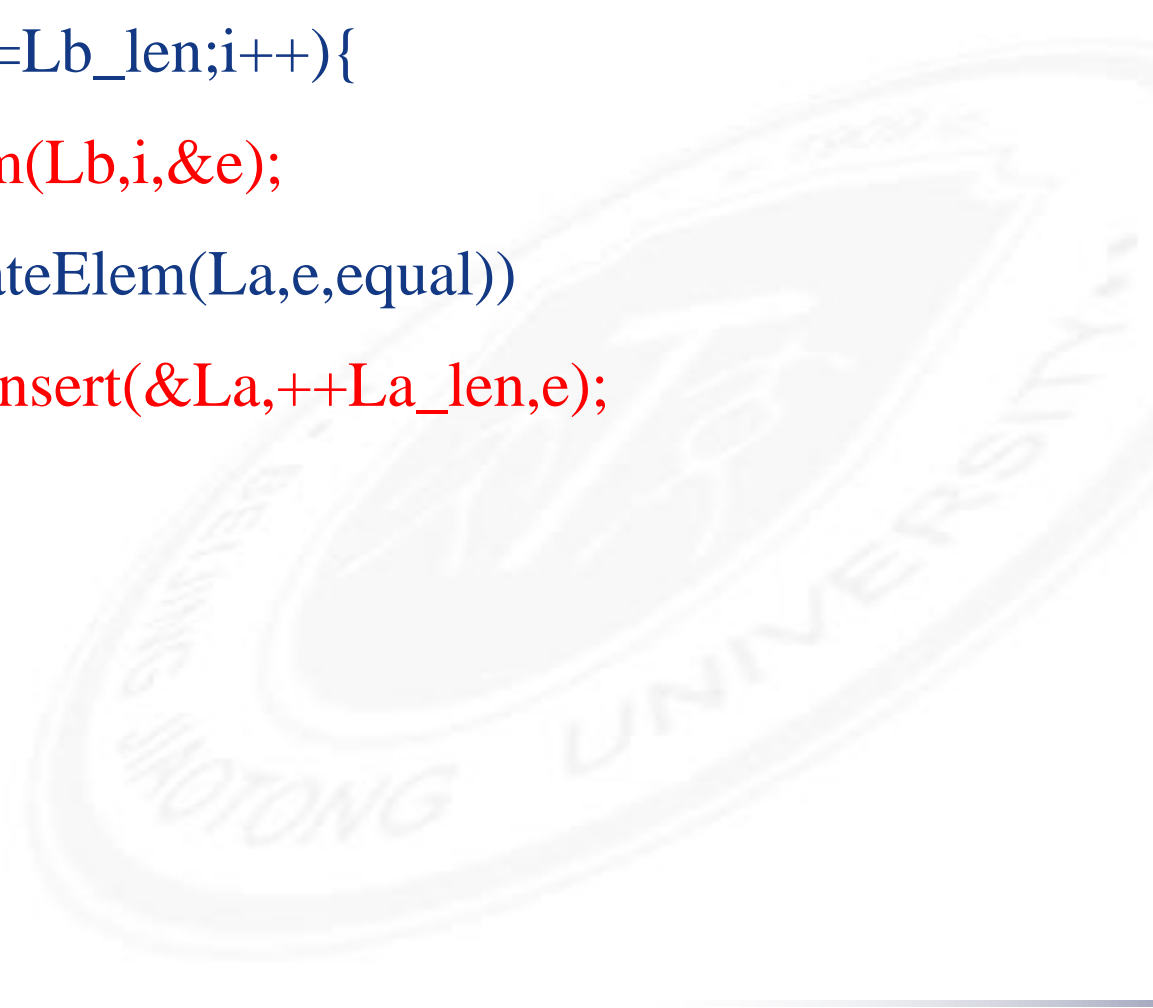
LA=(3, 5, 8, 11, 2, 6, 9, 15, 20)



## 例1——线性表合并问题

- ❶ 任务1：用自然语言或伪码编写算法
- ❷ 任务2：识别算法中所用到的线性表基本操作

```
void union(List &La, List Lb){  
    La_len=ListLength(La);  
    Lb_len=ListLength(Lb);  
    for(i=1;i<=Lb_len;i++){  
        GetElem(Lb,i,&e);  
        if(!LocateElem(La,e,equal))  
            ListInsert(&La,++La_len,e);  
    }  
} //union
```

A large, faint, diagonal watermark of the Wuyang University logo is visible in the background. The logo is oval-shaped and contains the university's name in both Chinese and English, along with the founding year 1982.

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

```
LA=(3, 5, 8, 11)
```

```
LB=(2, 6, 8, 9, 11, 15, 20)
```

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

```
LA=(3, 5, 8, 11)
```

```
LB=(2, 6, 8, 9, 11, 15, 20)
```

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=1



```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=1

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
**i=1**

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=1

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=2

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=2

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=2

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=2

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=3



```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=3

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=3

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=4

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=4

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=4

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6, 9)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=4

```
void union(List &La, List Lb){
```

```
    La_len=ListLength(La);
```

```
    Lb_len=ListLength(Lb);
```

```
    for(i=1;i<=Lb_len;i++){
```

$O(ListLength(LA) \times ListLength(LB))$

```
        GetElem(Lb,i, e);
```

```
        if(!LocateElem(La,e,equal))
```

```
            ListInsert(La,++La_len,e);
```

```
    }
```

```
} //union
```

LA=(3, 5, 8, 11, 2, 6, 9, 15, 20)

LB=(2, 6, 8, 9, 11, 15, 20)

↑  
i=8



## 例1——线性表合并问题

- ❶ 任务1：用自然语言或伪码编写算法
- ❷ 任务2：识别算法中所用到的线性表基本操作
- ❸ 任务3：利用实验1的顺序表基本操作，编写顺序表库文件，实现你的算法
- ❹ 任务4：利用实验2的链表基本操作，编写链表库文件，实现你的算法





## 例1——线性表合并问题

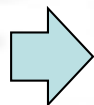
### 1.2 有序表的合并

#### 问题描述：

已知线性表LA 和LB中的数据元素按值非递减有序排列,现要求将LA和LB归并为一个新的线性表LC,且LC中的数据元素仍按值非递减有序排列.

LA=(3, 5, 8, 11)

LB=(2, 6, 8, 9, 11, 15, 20)



LA=(2, 3, 5, 6, 8, 8, 9, 11, 11, 15, 20)



## 例1——线性表合并问题

- ❶ 任务1：用自然语言或伪码编写算法
- ❷ 任务2：识别算法中所用到的线性表基本操作
- ❸ 任务3：利用你的顺序表库文件，实现你的算法
- ❹ 任务4：利用你的链表库文件，实现你的算法

```
void MergeList(List La,List Lb,List &Lc){
```

```
    InitList(Lc);    i=j=1;k=0;
```

```
    La_len=ListLength(La); Lb_len=ListLength(Lb);
```

```
    while((i<=La_len)&&(j<=Lb_len)){
```

```
        GetElem(La, i, ai); GetElem(Lb, j, bj);
```

```
        if(ai<=bj) {ListInsert(Lc, ++k, ai); ++i;}
```

```
        else{ListInsert(Lc,++k, bj); ++j;}
```

```
    }
```

```
    while(i<=La_len){
```

```
        GetElem(La, i++, ai); ListInsert(Lc, ++k, ai);
```

```
    }
```

```
    while(j<=Lb_len){
```

```
        GetElem(Lb, j++, bj); ListInsert(Lc, ++k, bj);
```

```
} // MergeList
```

LA=(3, 5, 8, 11)

LB=(2, 6, 8, 9, 11, 15, 20)

LC=(2, 3, 5, 6, 8, 8, 9, 11, 11, 15, 20)

**$O(\text{ListLength(LA)} + \text{ListLength(LB)})$**



## 实验3-库文件的编写和使用

将实验1和实验2的基本操作封装成库文件，调用自己的库文件完成以下题目。

1. 有序顺序表的合并。已知线性表LA 和LB中的数据元素按值非递减有序排列,线性表 $LA=\{3, 5, 8, 11\}$ ,  $LB=\{2, 6, 8, 9, 11, 15, 20\}$ 。现要求将LA和LB归并为一个新的线性表LC,且LC中的数据元素仍按值非递减有序排列。
  - (1) 创建两个顺序表LA, LB（各链表按升序排列），分别打印显示；
  - (2) 将两个顺序表合并成一个新的有序表（升序排列），打印显示；



## 实验3-库文件的编写和使用

将实验1和实验2的基本操作封装成库文件，调用自己的库文件完成以下题目。

2. 将线性表中最小的元素移到线性表的第一个位置。使用单链表实现。

(1) 从键盘顺序输入任意个数据，新建单链表。

(2) 找出单链表中数值最小的元素，并打印输出其数值和位置。格式为 “The minimum element is %d, and it's position in the list is %d.”

(3) 将该结点移到链表的表头位置，并打印输出新的链表。



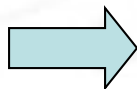
### 例3 合并有序表

#### 问题描述：

假设头指针为La和Lb的单链表分别为线性表LA和LB的存储结构，归并La和Lb得到单链表Lc

LA=(5, 8, 12)

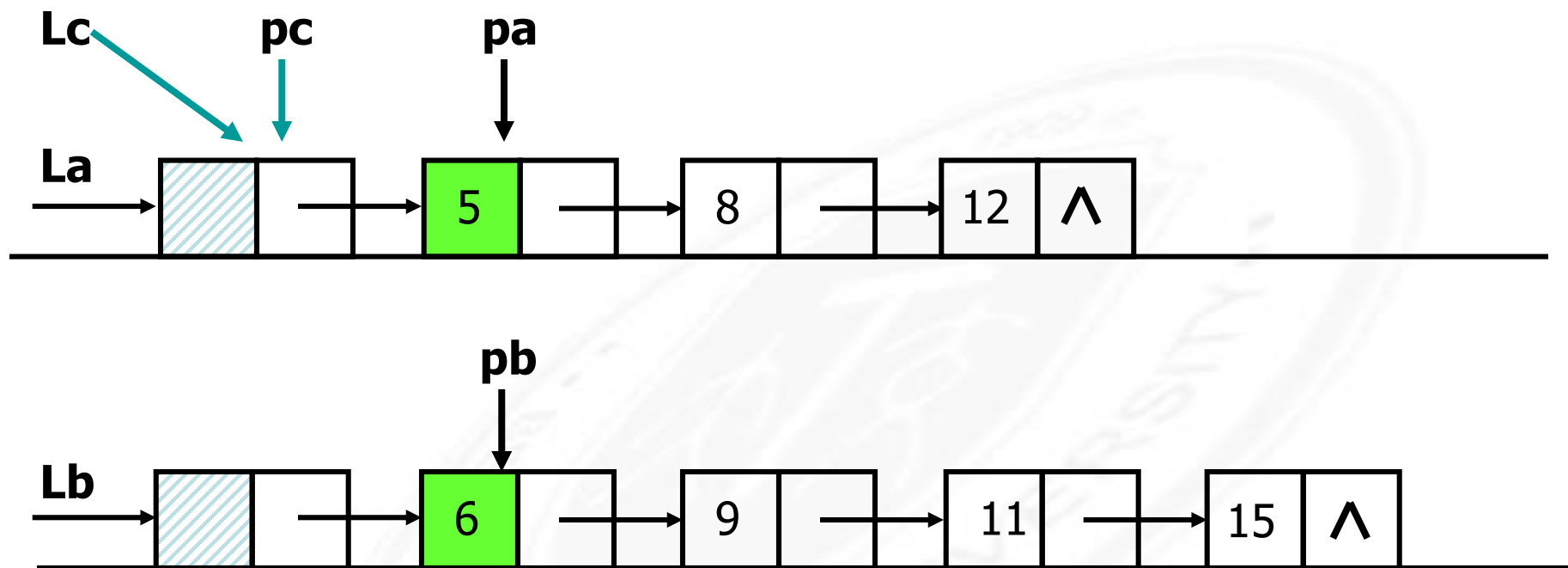
LB=(6, 9, 11, 15)



LC=(5, 6, 8, 9, 11, 12, 15)



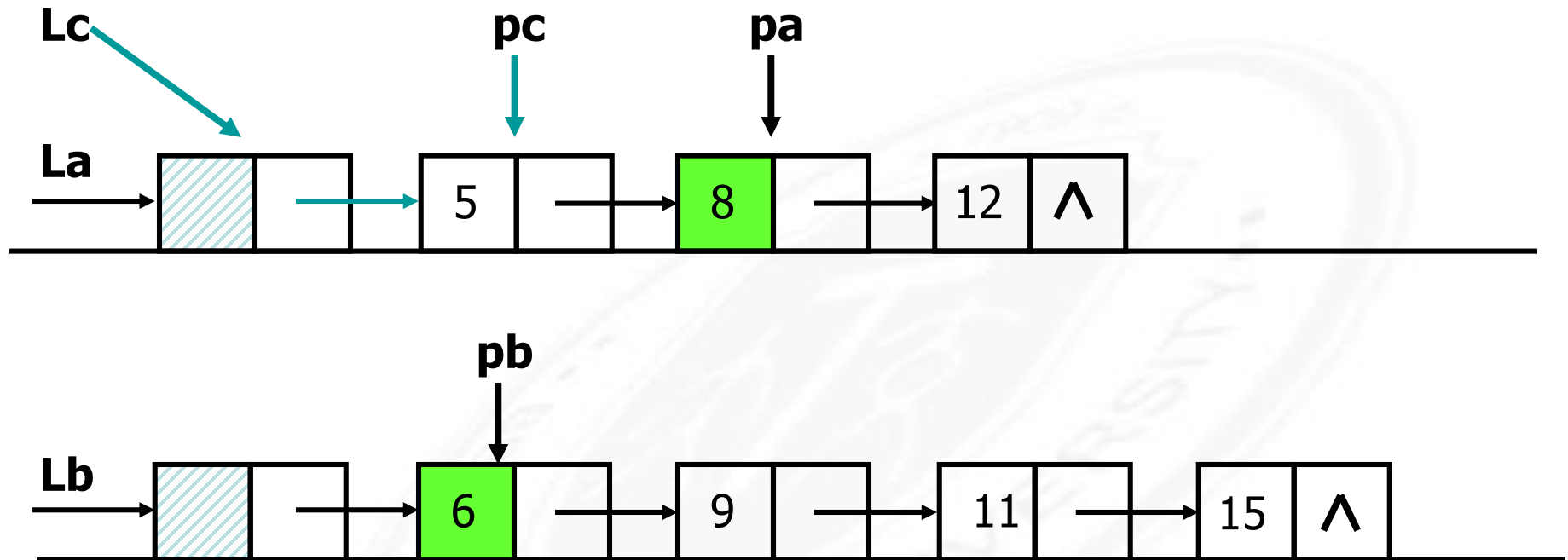
### 例3 合并有序表



```
pa=La->next;  
pb=Lb->next;  
pc=La=Lc;
```



### 例3 合并有序表



**`pc->next=pa;`**

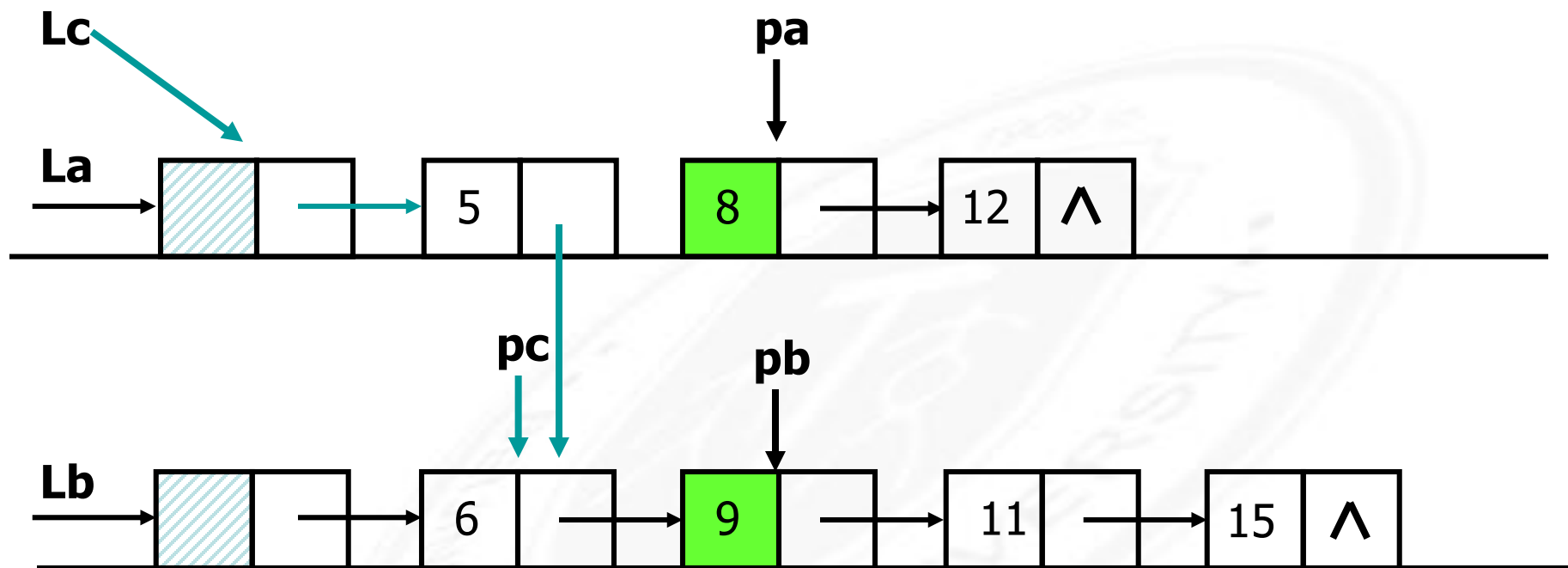
**`pc=pa`**

**`pa=pa->next;`**





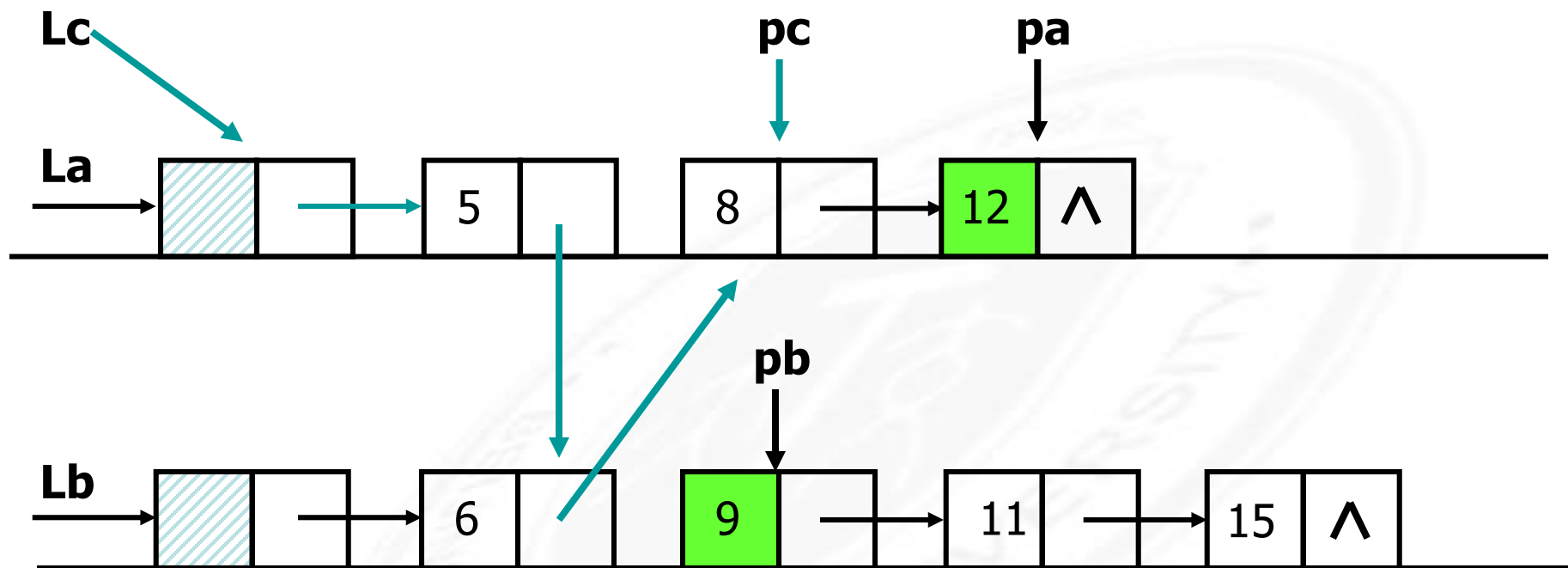
### 例3 合并有序表



**`pc->next=pb;`**  
**`pc=pb`**  
**`pb=pb->next;`**



### 例3 合并有序表



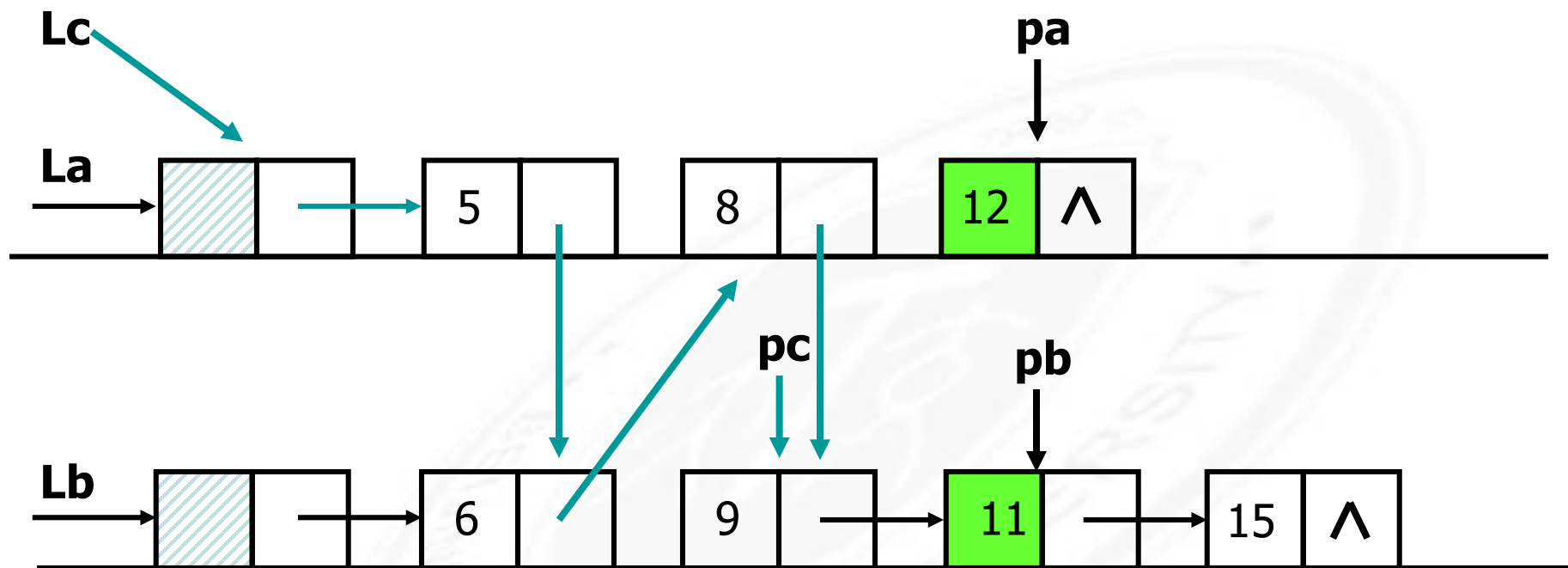
**pc->next=pa;**

**pc=pa**

**pa=pa->next;**



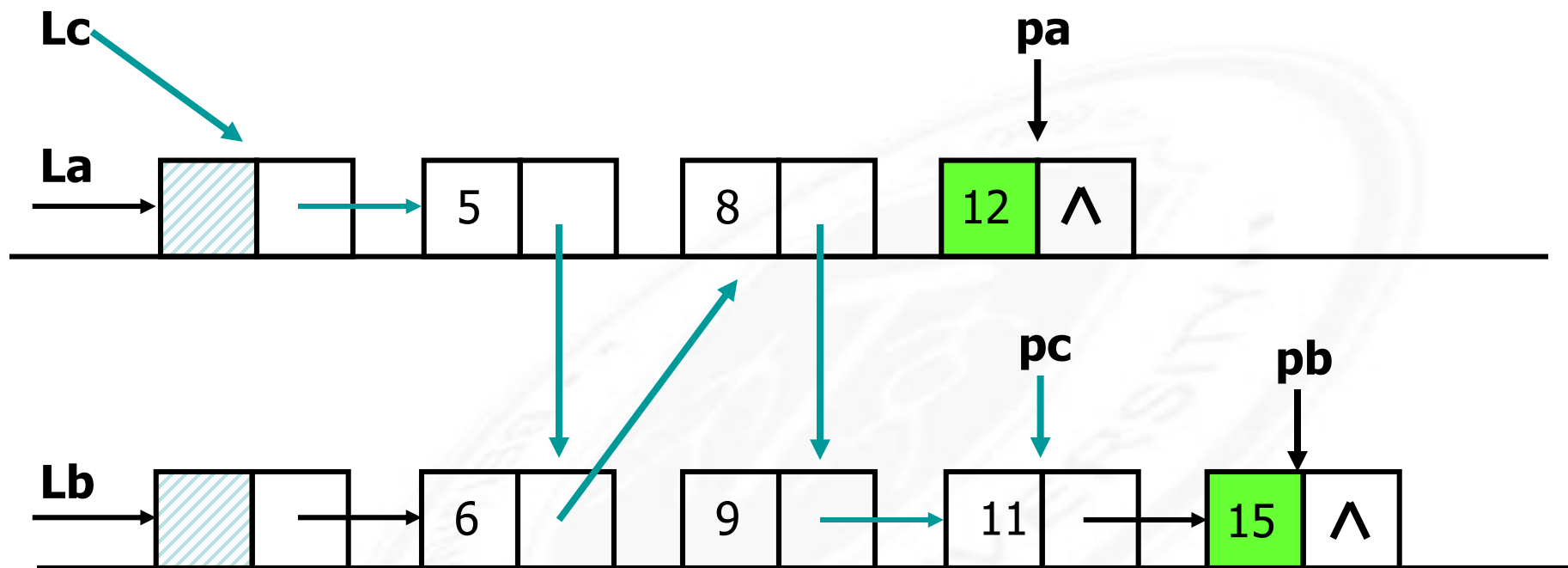
### 例3 合并有序表



**$pc \rightarrow next = pb;$**   
 **$pc = pb$**   
 **$pb = pb \rightarrow next;$**



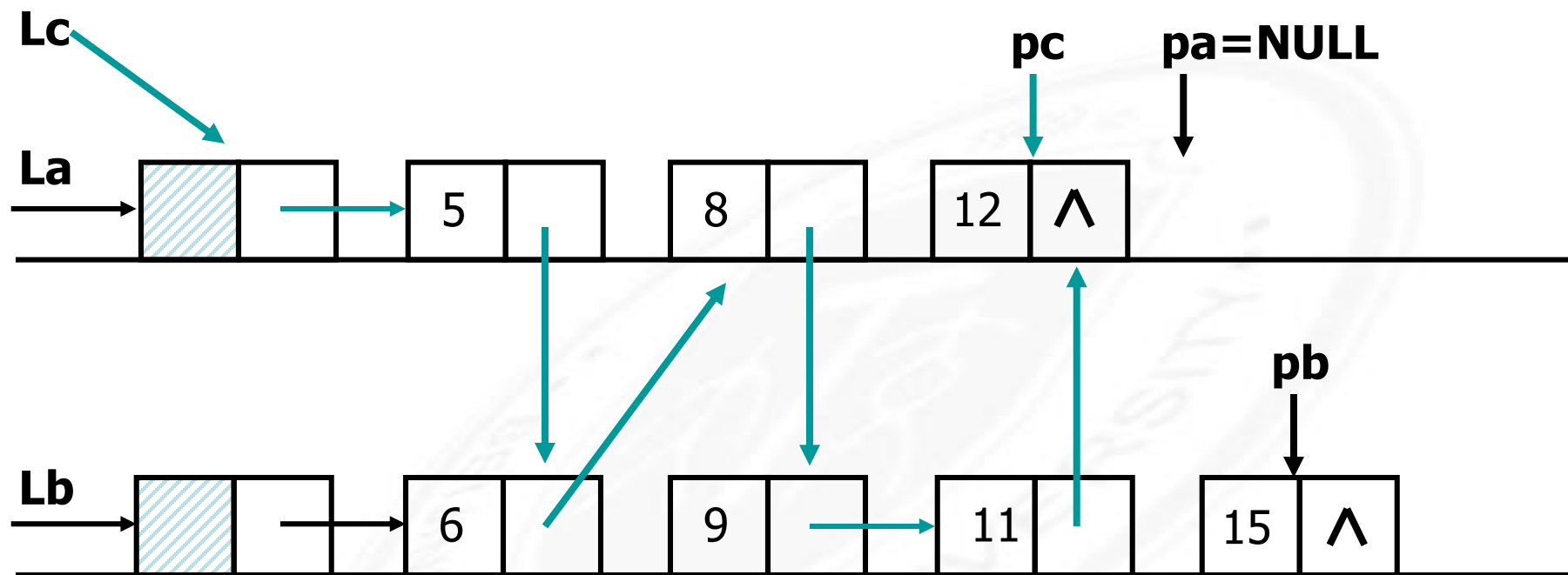
### 例3 合并有序表



```
pc->next=pb;  
pc=pb  
pb=pb->next;
```



### 例3 合并有序表



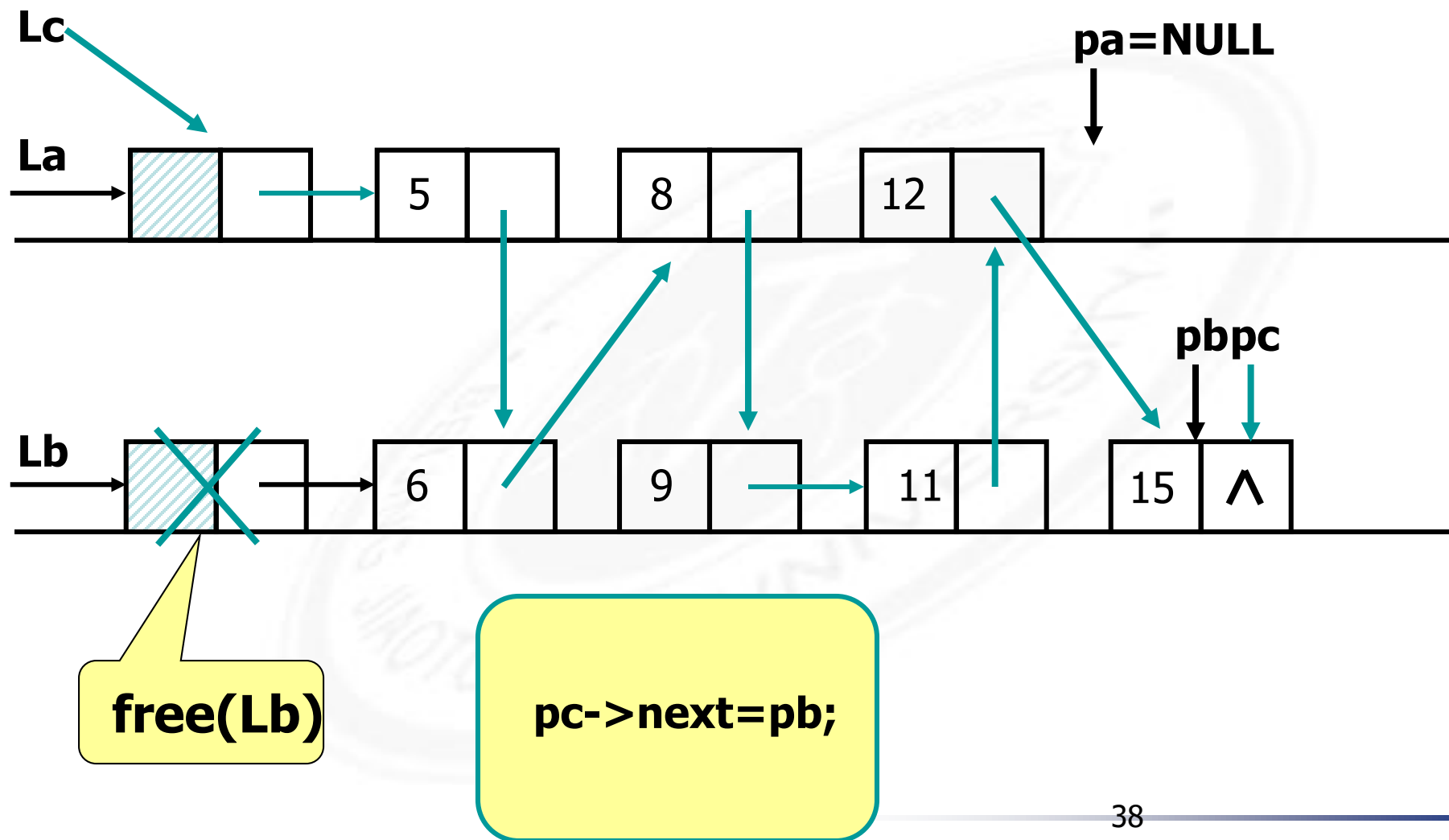
$pc \rightarrow next = pa;$

$pc = pa$

$pa = pa \rightarrow next;$



### 例3 合并有序表



```
void MergeList_L(LinkList &La,LinkList &Lb,LinkList &Lc){
```

```
//已知单链线性表La和Lb的元素按值非递减排列
```

```
//归并La和Lb得到新的单链线性表Lc,Lc的元素也按值非递减排列
```

```
pa=La->next; pb=Lb->next;
```

```
Lc=pc=La;          //用La的头结点作为Lc的头结点
```

```
while(pa && pb){
```

```
    if(pa->data<=pb->data){
```

```
        pc->next=pa; pc=pa; pa=pa->next;
```

```
    }
```

```
    else{pc->next=pb; pc=pb; pb=pb->next;}
```

```
pc->next= pa? pa: pb;  //插入剩余段
```

```
free(Lb);            //释放Lb的头结点
```

```
}//MergeList_L
```