线性表

顺序表
链表

使用ADT来描述线性表

线性表是具有相同数据类型的n个数据元素的有限序列，n为表长

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

除了第一个元素之外，每个元素有且仅有一个直接前驱。除了最后一个元素外，每个元素有且仅有一个直接后继。

ADT List{

数据对象: $D=\{a_i \mid a_i \in ElemSet, i=1,2,\cdots,n, n\geqslant 0\}$

数据关系: $R=\{<a_{i-1}, a_i> \mid a_{i-1}, a_i \in D, i=2,\cdots,n\}$

基本操作（运算）

基本操作：

InitList(&L)

操作结果：构造一个空的线性表 L。

DestroyList(&L)

初始条件：线性表 L 已存在。

操作结果：销毁线性表 L。

ClearList(&L)

初始条件：线性表 L 已存在。

操作结果：将 L 重置为空表。

ListEmpty(L)

初始条件：线性表 L 已存在。

操作结果：若 L 为空表，则返回 true，否则返回 false。

ListLength(L)

初始条件：线性表 L 已存在。

操作结果：返回 L 中数据元素个数。

GetElem(L,i,&e)

初始条件：线性表 L 已存在，且 1≤i≤ListLength(L)。

操作结果：用 e 返回 L 中第 i 个数据元素的值。

LocateElem(L,e)

初始条件：线性表 L 已存在。

操作结果：返回 L 中第 1 个值与 e 相同的元素在 L 中的位置。若这样的数据元素不存在，则返回值为 0。

顺序表

顺序表：用顺序存储的方式来实现线性表

顺序存储：把逻辑上相邻的元素存储在物理位置上也相邻的存储单元

```
int main() {
    SeqList seqList;
    seqList.arr[0] = 1;
    seqList.arr[1] = 2;
    seqList.arr[2] = 3;
    for (int i = 0;i<3;i++)
        cout << &seqList.arr[i] << endl;
    return 0;
}
```

```
E:\kaoyanfudao\1\D
00EFFB88
00EFFB8C
00EFFB90
```

顺序表的结构体定义

```c
#define MaxSize 50
typedef struct
{
    int data[MaxSize];
    int length;
}SeqList;

void InitList(SeqList &L){
    for (int i=0;i<L.length;i++)
        L.data[i] = 0;
    L.length = 0;
}
```

```c
#define InitSize 100
typedef struct
{
    int *data;
    int MaxSize, length;
};

void InitList(SeqList &L){
    // L.data = new int[InitSize];
    L.data = (int *)malloc(sizeof(int)*InitSize);
    L.MaxSize = InitSize;
    L.length = 0;
}
```

动态扩充

```
//增加动态数组的长度
void IncreaseSize(SeqList &L, int len){
    int *p=L.data;
    L.data=(int *)malloc((L.MaxSize+len)*sizeof(int));
    for(int i=0; i<L.length; i++){
        L.data[i]=p[i];                    //将数据复制到新区域
    }
    L.MaxSize=L.MaxSize+len;               //顺序表最大长度增加 len
    free(p);                               //释放原来的内存空间
}
```

```
int insertElem(Sqlist &L,int p,int e) //L本身要发生改变，所以用引用型
{
    int i;
    if(p<0||p>L.length||L.length==maxSize)//位置错误或者表长已经达到
        return 0;                  //顺序表的最大允许值，此时插入不成功，返回0
    for(i=L.length-1;i>=p;--i)
        L.data[i+1]=L.data[i];  //从后往前，逐个将元素往后移动一个位置
    L.data[p]=e;                //将e放在插入位置p上
    ++(L.length);               //表内元素多了一个，因此表长自增1
    return 1;                   //插入成功，返回1
}
```

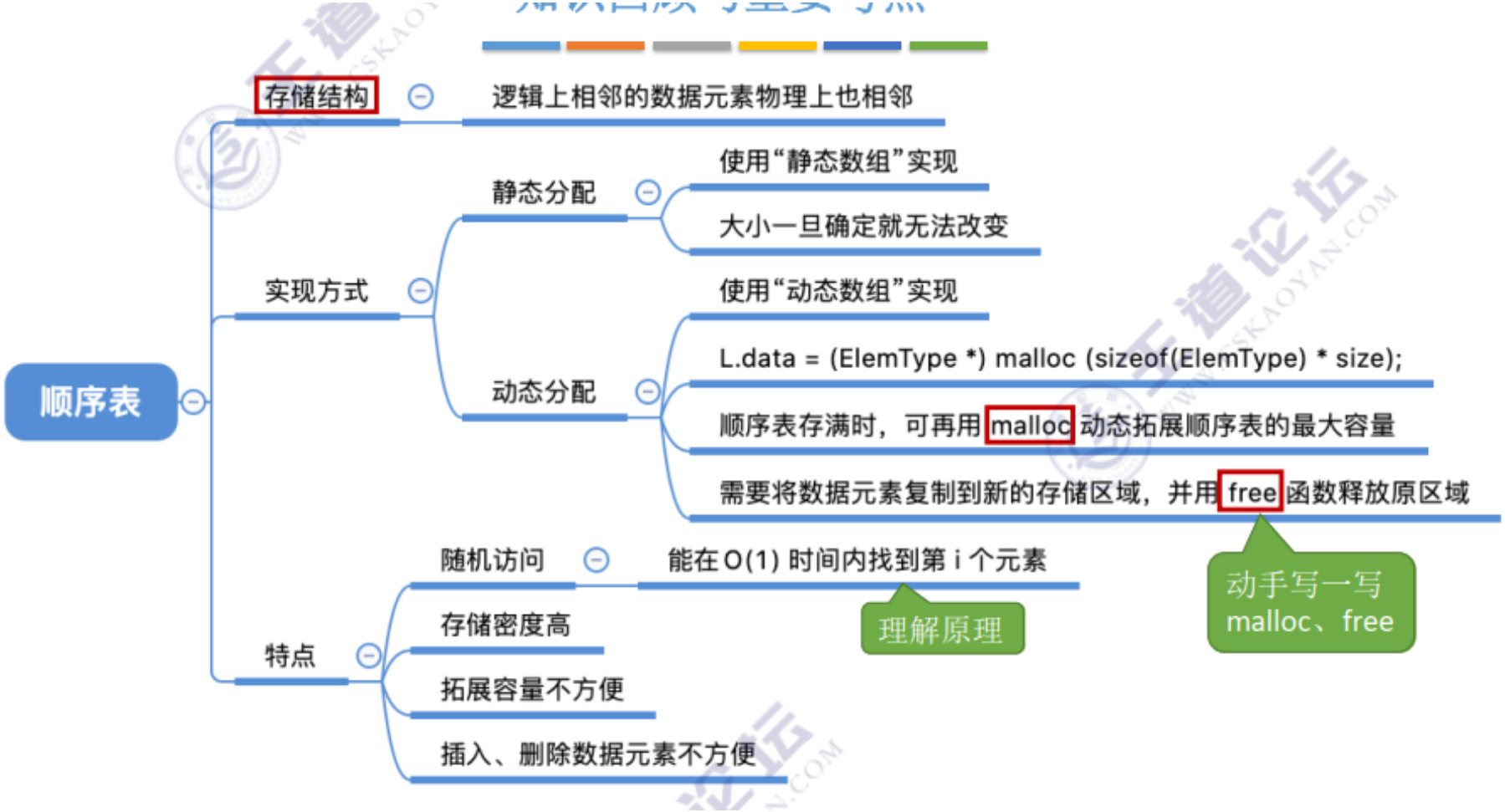
1.判断数据的合法性
2.模拟操作

补充

```
void demo2(){
    vector<int> vec;
    cout << vec.capacity() << endl;
    for (int i = 0;i<15;i++){
        vec.push_back(i);
        cout << i+1 << ' ';
        cout << vec.capacity() << endl;
    }
}
```

```
void demo3(){
    #define Equ(a, b) ((fabs((a) - (b))) < (eps))
    const double eps = 1e-8;
    double db = 1.23;
    if(Equ(db,  b: 1.23)) {
        printf( _Format: "true");
    } else {
        printf( _Format: "false");
    }
}
```

顺序表总结


顺序表
存储结构
逻辑上相邻的数据元素物理上也相邻
实现方式
静态分配
使用“静态数组”实现
大小一旦确定就无法改变
动态分配
使用“动态数组”实现
L.data = (ElemType *) malloc (sizeof(ElemType) * size);
顺序表存满时，可再用 malloc 动态拓展顺序表的最大容量
需要将数据元素复制到新的存储区域，并用 free 函数释放原区域
动手写一写 malloc、free
特点
随机访问
能在 O(1) 时间内找到第 i 个元素
理解原理
存储密度高
拓展容量不方便
插入、删除数据元素不方便

真题

(2014 填空1)在一个长度为k的顺序存储线性表中，向第i个元素(1<=i<=k+1)之前插入一个新元素时，需要从后向前依次后移____个元素。

(2016 填空1)若长度为n的顺序表第i个元素之前插入一个元素，则需要向后移动的元素个数是____

(2017 填空6)长度为n的顺序表中删除第i个元素(1<=i<=n)时，元素移动的次数为_____

(2017 填空3)给定一个有n个元素的线性表。若采用顺序存储结构，则在等概率前提下，向其插入一个元素需要移动的元素个数平均为____

真题

(2020 2)(1)在顺序表中查找第一个元素值等于x的元素，并返回其顺序。

(2)统计顺序表中x出现的次数。

练习

02. 设计一个高效算法，将顺序表 L 的所有元素逆置，要求算法的空间复杂度为 $O(1)$。

循环（遍历，迭代）
/递归

```
void reverseSeqListRecursion(SeqList &L, int i){
    if (i>=length/2){
        return ;
    }

    int temp = 0;
    temp = L.data[i];
    L.data[i] = L.data[length-i-1];
    L.data[length-i-1] = temp;
    reverseSeqListRecursion(L, i+1);
    return ;
}

reverseSeqListRecursion(L, 0);
```

练习

05. 从顺序表中删除其值在给定值 s 与 t 之间（包含 s 和 t，要求 $s<t$）的所有元素，若 s 或 t 不合理或顺序表为空，则显示出错信息并退出运行。

剑指 Offer 03. 数组中重复的数字

难度 简单 829

找出数组中重复的数字。

在一个长度为 n 的数组 nums 里的所有数字都在 0～n-1 的范围内。数组中某些数字是重复的，但不知道有几个数字重复了，也不知道每个数字重复了几次。请找出数组中任意一个重复的数字。

示例 1：

输入：
`[2, 3, 1, 0, 2, 5, 3]`
输出： `2 或 3`

1. 两数之和

难度 简单 14324

给定一个整数数组 `nums` 和一个整数目标值 `target`，请你在该数组中找出 **和为目标值** *target* 的那 **两个** 整数，并返回它们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素在答案里不能重复出现。

你可以按任意顺序返回答案。

示例 1：

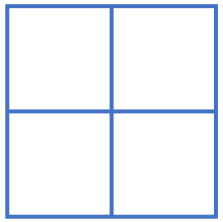
```
输入：nums = [2,7,11,15], target = 9
输出：[0,1]
解释：因为 nums[0] + nums[1] == 9 ，返回 [0, 1] 。
```

线性表推广

C语言中的二维/多维数组

```
void demo4() {
    int a[2][2];
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++)
            cout << &a[i][j] << endl;
}
```

a[2][2]



```
Demo ×
E:\kaoyanfudao\1\Demo\cmak
0095F9B8
0095F9BC
0095F9C0
0095F9C4
```

真题

(2016 填空8)若三维数组a[4][5][6]的基地址是100，每个元素占用2存储单元，则数组a中最后一个元素的存储地址是_____

(2018 问答题3)已知二维数组A[5][10]按行优先顺序存储在内存中，假设每个元素占三个存储单元，第一个元素的存储地址即LOC(A[0][0])=1000，计算出LOC(A[3][4])的值。

(2020 程序4) 设计算法使n*n的矩阵旋转180度

#define m 10

给定一个二维数组int A[m][m] 水平翻转

垂直翻转

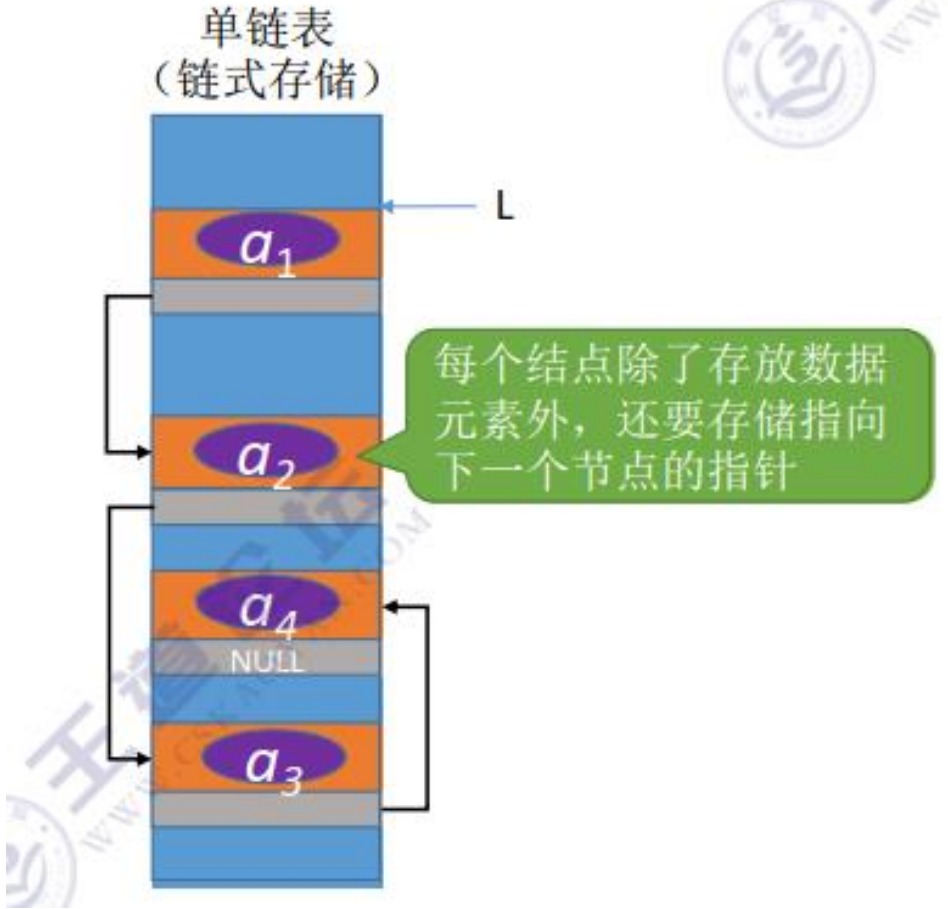
矩阵转置　　#define c 5, m 10, n 20

旋转90度　　矩阵乘法 int A[c][m] int B[m][n]

单链表

单链表的结构体定义

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



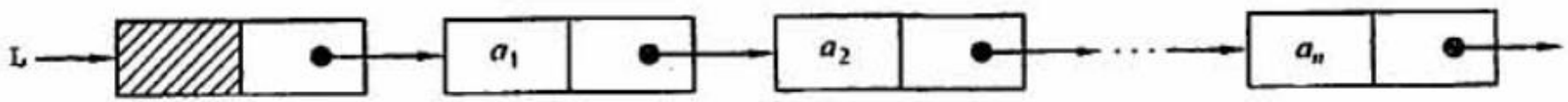
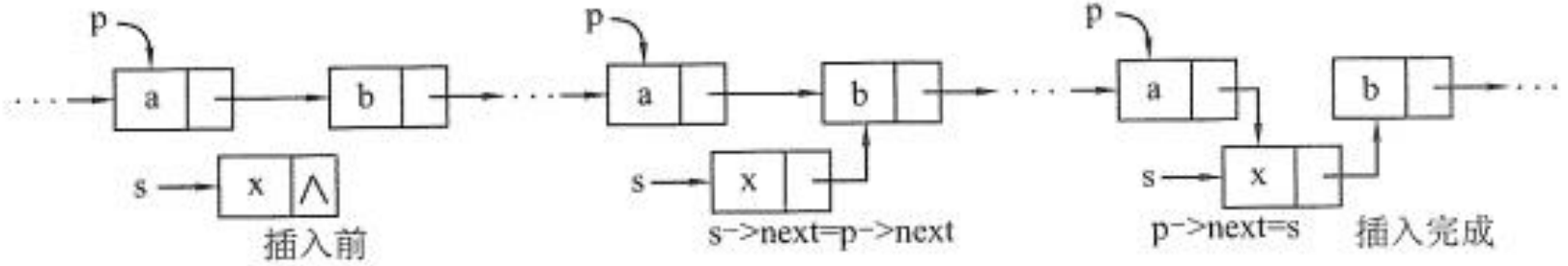


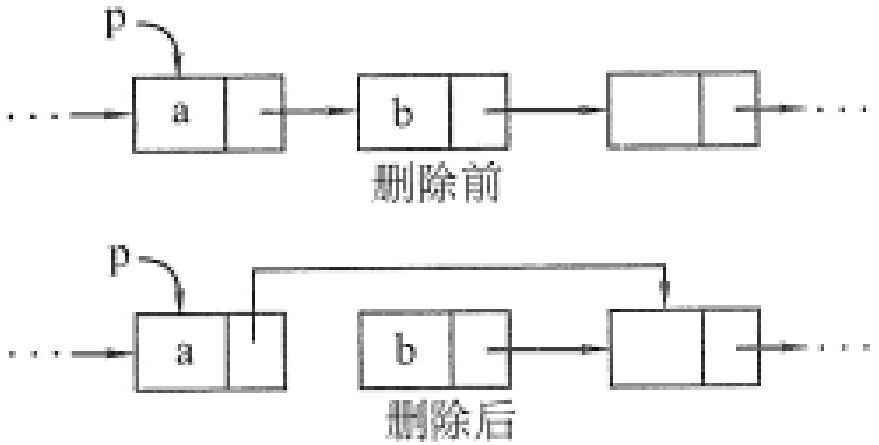
图 2.4　带头结点的单链表

按序号插入/按序号删除/按值删除

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




```
q=p->next;
p->next=p->next->next;
free(q);//调用函数 free()来释放 q 所指结点的内存空间
```



创建单链表

为什么是引用?

头插法创建一个带有头结点的单链表

```
void createlistF(LNode *&C,int a[],int n)
{
    LNode *s;
    int i;
    C=(LNode*)malloc(sizeof(LNode));
    C->next=NULL;
    for (i=0;i<n;++i)
    {
        s=(LNode*)malloc(sizeof(LNode));
        s->data=a[i];
        /*下边两句是头插法的关键步骤*/
        s->next=C->next;//s 所指新结点的指针域 next 指向 C 中的开始结点
        C->next=s;          //头结点的指针域 next 指向 s 结点，使得 s 成为新的开始结点
    }
}
```

创建单链表

尾插法创建一个带有*头结点*的单链表

```
void createList_tail(LNode *&C, int a[], int n){
    LNode *s;
    C = (LNode *)malloc(sizeof(LNode));
    C->next = NULL;
    LNode *tail = C;
    for (int i = 0;i<n;i++){
        s = (LNode *)malloc(sizeof(LNode));
        s->data = a[i];
        tail->next = s;
        tail = tail->next;
    }
    tail->next = NULL;
}
```

创建单链表

头插法创建一个无头结点的单链表 写我们讲的另外一种

```
void createList_head(LNode *&C, int a[], int n){
    LNode *s;
    C = (LNode *)malloc(sizeof(LNode));
    C->next = NULL;
    C->data = a[0];
    for (int i = 1;i<n;i++){
        s = (LNode *)malloc(sizeof(LNode));
        s->data = a[i];
        s->next = C;
        C = s;
    }
}
```

思考

两种创建方式有什么不同?

头插法创建的链表，其中元素和原来相比是倒序的。
（天勤P27页）（单链表的逆置 原地逆置，模拟头插）
反转链表： https://leetcode-cn.com/problems/reverse-linked-list/

- 头节点的数据域可以存放链表的长度。
- 一般情况下，都采用带有头结点的单链表

判空

空表判断：L==NULL。写代码不方便

空表判断：L->next==NULL。写代码更方便

求单链表长度时，头结点不算

单链表的查找

GetElem(L,i)：按位查找操作。获取表L中第i个位置的元素的值。

LocateElem(L,e)：按值查找操作。在表L中查找具有给定关键字值的元素。

```
//按值查找，找到数据域==e 的结点
LNode * LocateElem(LinkList L,ElemType e) {
    LNode *p = L->next;
    //从第1个结点开始查找数据域为e的结点
    while (p != NULL && p->data != e)
        p = p->next;
    return p;    //找到后返回该结点指针，否则返回NULL
}
```

双链表

单链表结点中只有一个指向其后继的指针，使得单链表只能从头结点依次顺序地向后遍历。要访问某个结点的前驱结点（插入、删除操作时），只能从头开始遍历，访问后继结点的时间复杂度为 $O(1)$，访问前驱结点的时间复杂度为 $O(n)$。

双链表的结构体定义

3．双链表结点定义

```
typedef struct DLNode
{
    int  data;                  //data 中存放结点数据域（默认是 int 型）
    struct DLNode  *prior;      //指向前驱结点的指针
    struct DLNode  *next;       //指向后继结点的指针
}DLNode;                        //定义双链表结点类型
```

说明：

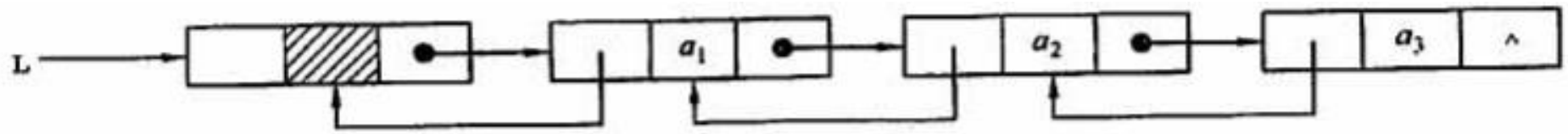


图 2.9　双链表示意图

双链表的插入

1．双链表的插入操作

在双链表中 p 所指的结点之后插入结点*s，其指针的变化过程如图 2.10 所示。

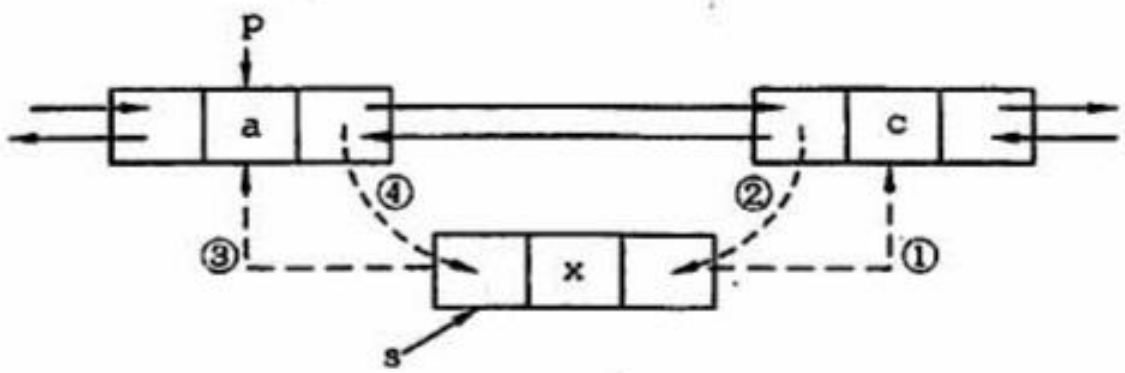


图 2.10　双链表插入结点过程

插入操作的代码片段如下：

```
①s->next=p->next;                    //将结点*s 插入到结点*p 之后
②p->next->prior=s;
③s->prior=p;
④p->next=s;
```

双链表的删除

2．双链表的删除操作

删除双链表中结点*p 的后继结点*q，其指针的变化过程如图 2.11 所示。

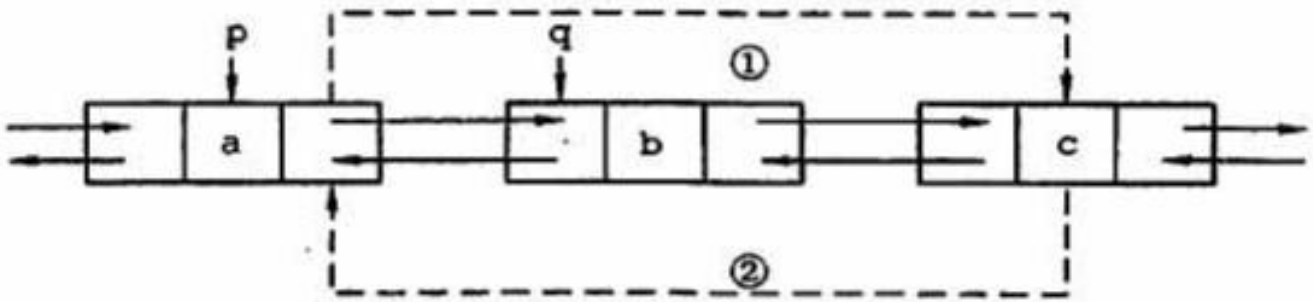


图 2.11　双链表删除结点过程

删除操作的代码片段如下：

```
p->next=q->next;              //图 2.11 中步骤①
q->next->prior=p;             //图 2.11 中步骤②
free(q);                      //释放结点空间
```

循环单链表

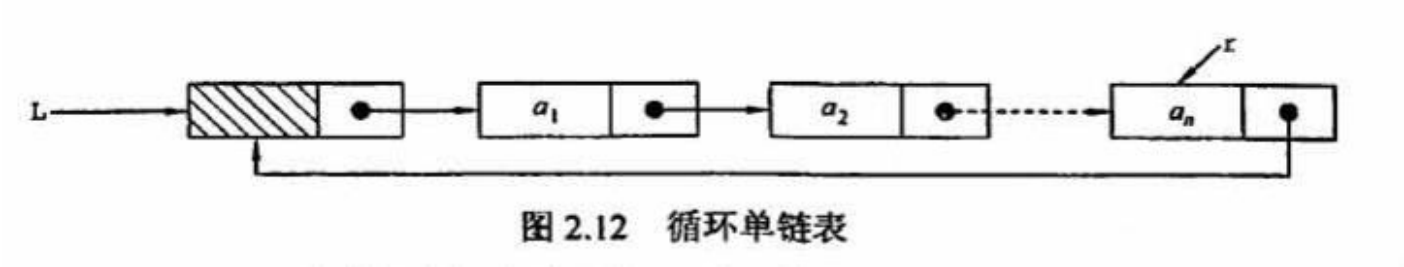
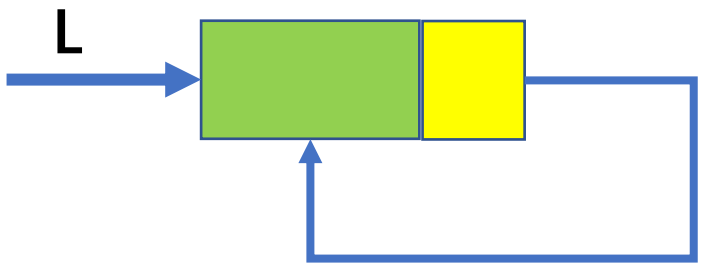


图 2.12　循环单链表

循环单链表的判空条件不是头结点的指针是否为空
而是它是否等于头结点



循环单链表

```
void InitList(LNode *L){
    L = (LNode *)malloc(sizeof(LNode));
    L->next = L;
}
```

```
bool isTail(LNode *L, LNode *p){
    return p->next == L;
}
```

循环双链表

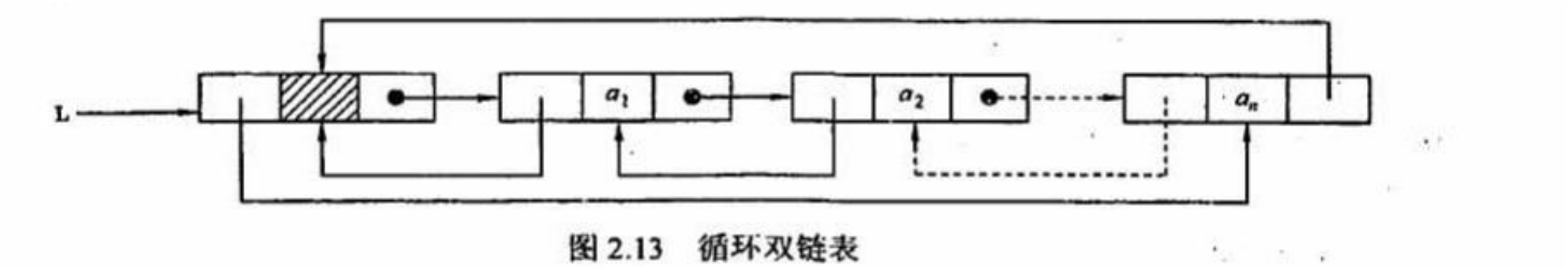
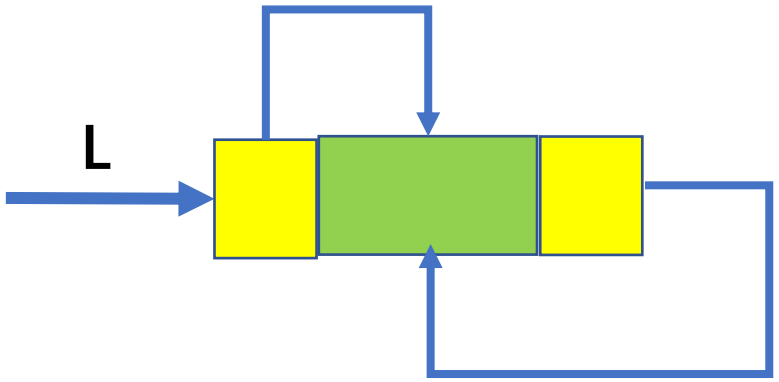


图 2.13　循环双链表

当循环双链表为空表时，其头结点的prior域和next域都等于L



循环双链表

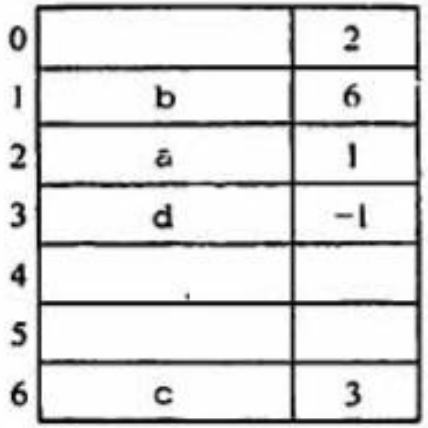
```
void InitList(DLNode *L){
    L = (DLNode *)malloc(sizeof(DLNode));
    L->next = L;
    L->prior = L;
}

bool isTail(DLNode *L, DLNode *p){
    return p->next == L;
}
```

静态链表

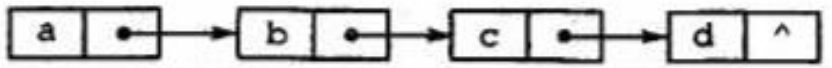
静态链表的结构体定义

```
#define MaxSize 50
typedef struct
{
    int data;
    int next;
}SLinkList[MaxSize];
```



0		2
1	b	6
2	a	1
3	d	-1
4		
5		
6	c	3

(a)静态链表示例

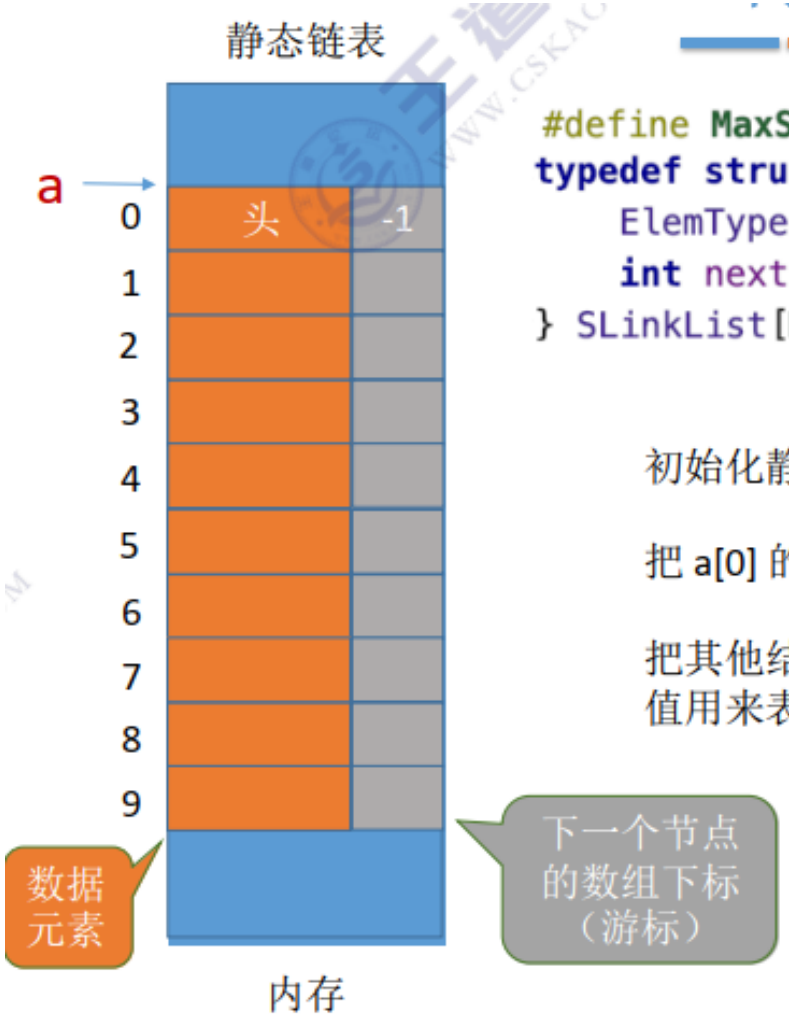


(b)静态链表对应的单链表

静态链表以next=-1作为其结束的标志。

静态链表为什么是链式存储?

静态链表




```
#define MaxSize 10          //静态链表的最大长度
typedef struct {            //静态链表结构类型的定义
    ElemType data;          //存储数据元素
    int next;               //下一个元素的数组下标
} SLinkList[MaxSize];
```

```
void testSLinkL
    SLinkList a
    //.....后续
}
```

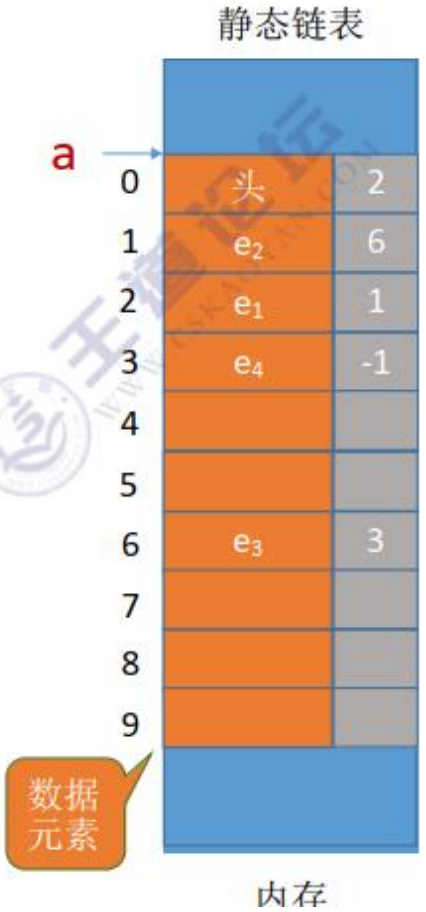
初始化静态链表：

把 a[0] 的 next 设为 -1

把其他结点的 next 设为一个特殊值用来表示结点空闲，如 -2

静态链表

简述基本操作的实现




```
#define MaxSize 10          //静态链表的最大长度
typedef struct {            //静态链表结构类型的定义
    ElemType data;          //存储数据元素
    int next;               //下一个元素的数组下标
} SLinkList[MaxSize];
```

```
void testSLinkList() {
    SLinkList a;
    //......后续代码
}
```

查找：
从头结点出发挨个往后遍历结点

插入位序为 i 的结点：

如何判断结点是否为空？

可让 next 为某个特殊值，如 -2

①找到一个空的结点，存入数据元素
②从头结点出发找到位序为 i-1 的结点
③修改新结点的 next
④修改 i-1 号结点的 next

删除某个结点：

①从头结点出发找到前驱结点
②修改前驱结点的游标
③被删除结点 next 设为 -2

适用场景：①不支持指针的低级语言；②数据元素数量固定不变的场景（如操作系统的文件分配表FAT）

顺序表和链表的比较

表 2.2　　顺序表和链表的比较

比较项目 \ 存储结构		顺序表	链表
空间	存储空间	预先分配，会导致空间闲置或溢出现象	动态分配，不会出现存储空间闲置或溢出现象
	存储密度	不用为表示结点间的逻辑关系而增加额外的存储开销，存储密度等于 1	需要借助指针来体现元素间的逻辑关系，存储密度小于 1
时间	存取元素	随机存取，按位置访问元素的时间复杂度为 O(1)	顺序存取，按位置访问元素时间复杂度为 O(n)
	插入、删除	平均移动约表中一半元素，时间复杂度为 O(n)	不需移动元素，确定插入、删除位置后，时间复杂度为 O(1)
适用情况		① 表长变化不大，且能事先确定变化的范围 ② 很少进行插入或删除操作，经常按元素位置序号访问数据元素	① 长度变化较大 ② 频繁进行插入或删除操作

线性表应用——线性表合并

集合并集

【问题描述】

已知两个集合 A 和 B，现要求一个新的集合 $A = A \cup B$。例如，设

$$A = (7, 5, 3, 11)$$

$$B = (2, 6, 3)$$

合并后

$$A = (7, 5, 3, 11, 2, 6)$$

【问题分析】

可以利用两个线性表 LA 和 LB 分别表示集合 A 和 B（即线性表中的数据元素为集合中的成员），这样只需扩大线性表 LA，将存在于 LB 中而不存在于 LA 中的数据元素插入到 LA 中去。只要从 LB 中依次取得每个数据元素，并依值在 LA 中进行查访，若不存在，则插入之。

上述操作过程可用算法 2.15 来描述。具体实现时既可采用顺序形式，也可采用链表形式。

线性表应用——线性表合并

【算法步骤】

① 分别获取 LA 表长 m 和 LB 表长 n。

② 从 LB 中第 1 个数据元素开始，循环 n 次执行以下操作：

- 从 LB 中查找第 i（$1 \leqslant i \leqslant n$）个数据元素赋给 e；
- 在 LA 中查找元素 e，如果不存在，则将 e 插在表 LA 的最后。

【算法描述】

```
void MergeList(List &LA,List LB)
{//将所有在线性表 LB 中但不在 LA 中的数据元素插入到 LA 中
   m=ListLength(LA); n=ListLength(LB);        //求线性表的长度
   for(i=1;i<=n;i++)
   {
      GetElem(LB,i,e);                        //取 LB 中第 i 个数据元素赋给 e
      if(!LocateElem(LA,e))                   //LA 中不存在和 e 相同的数据元素
         ListInsert(LA,++m,e);                //将 e 插在 LA 的最后
   }
}
```

1.调用基本操作
2.没有指明存储结构

线性表应用——线性表合并

有序表的合并

若线性表中的数据元素相互之间可以比较，并且数据元素在线性表中依值非递减或非递增有序排列，则称该线性表为**有序表**（Ordered List）。

天勤P27（链表）

(2018年 程序1)已知两个有序表A[0…n-1]和B[0…m-1]，试写一个算法，将他们归并为一个有序表C[0…m+n-1]

线性表应用——一元多项式

一般一元多项式

利用数组 p 表示：数组中每个分量 $p[i]$表示多项式每项的系数 p_i，数组分量的下标 i 即对应每项的指数。数组中非零的分量个数即为多项式的项数。

例如，多项式 $P(x) = 10 + 5x - 4x^2 + 3x^3 + 2x^4$ 可以用表 2.1 所示的数组表示。

表 2.1　　多项式的数组表示

指数（下标 i）	0	1	2	3	4
系数 $p[i]$	10	5	-4	3	2

显然，利用上述方法表示一元多项式，多项式相加的算法很容易实现，只要把两个数组对应的分量项相加就可以了。

```
typedef struct PNode
{
   float  coef;                          //系数
   int    expn;                          //指数
   struct PNode  *next;                  //指针域
}PNode,*Polynomial;
```

稀疏多项式

和顺序存储结构相比，利用链式存储结构更加灵活，更适合表示一般的多项式，合并过程的空间复杂度为 O(1)，所以较为常用。本节将给出如何利用单链表的基本操作来实现多项式的相加运算。

例如，图 2.22 所示两个链表分别表示多项式 $A(x)=7+3x+9x^8+5x^{17}$ 和多项式 $B(x)=8x+22x^7-9x^8$。从图中可见，每个结点表示多项式中的一项。

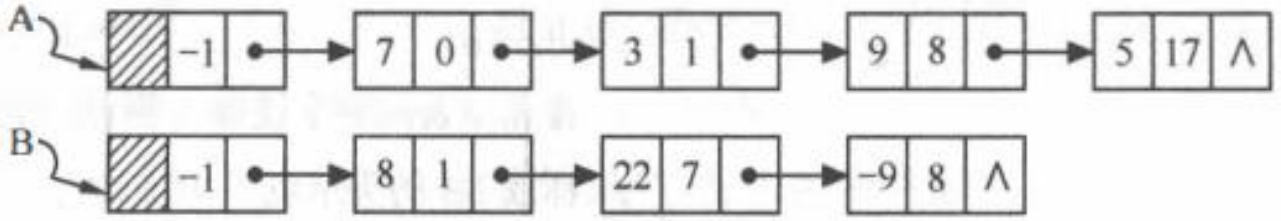


图 2.22　多项式的单链表存储结构

有序

算法 2.19　多项式的相加

【算法步骤】

时间复杂度为 $O(m+n)$

① 指针 p1 和 p2 初始化，分别指向 Pa 和 Pb 的首元结点。

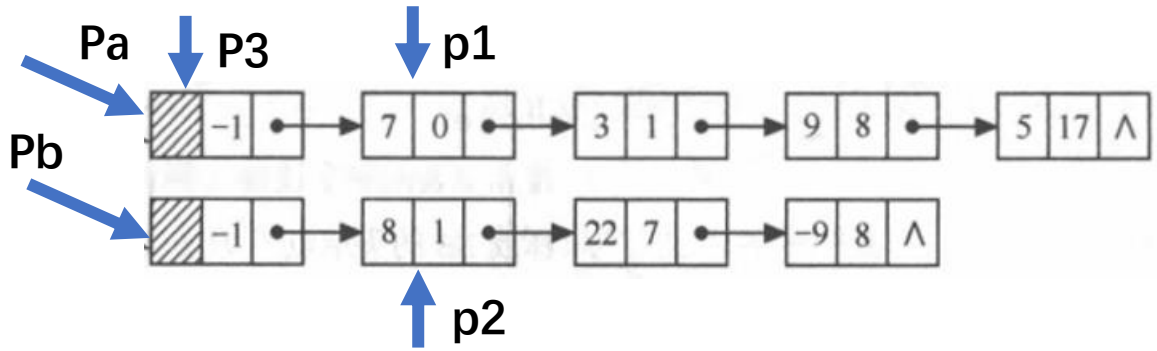
② p3 指向和多项式的当前结点，初值为 Pa 的头结点。

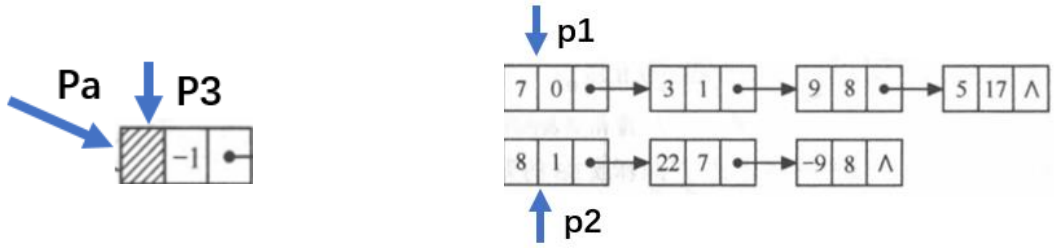
③ 当指针 p1 和 p2 均未到达相应表尾时，则循环比较 p1 和 p2 所指结点对应的指数值（p1->expn 与 p2->expn），有下列 3 种情况：

- 当 p1->expn 等于 p2->expn 时，则将两个结点中的系数相加，若和不为零，则修改 p1 所指结点的系数值，同时删除 p2 所指结点，若和为零，则删除 p1 和 p2 所指结点；
- 当 p1->expn 小于 p2->expn 时，则应摘取 p1 所指结点插入到“和多项式”链表中去；
- 当 p1->expn 大于 p2->expn 时，则应摘取 p2 所指结点插入到“和多项式”链表中去。

④ 将非空多项式的剩余段插入到 p3 所指结点之后。

⑤ 释放 Pb 的头结点。






```
void AddPolyn(Polynomial &Pa,Polynomial &Pb)
{//多项式加法：Pa=Pa+Pb，利用两个多项式的结点构成"
  p1=Pa->next; p2=Pb->next;              //p1
  p3=Pa;                                 //p3
  while(p1&&p2)                          //p1
  {
   if(p1->expn==p2->expn)                //指数
   {
      sum=p1->coef+p2->coef;             //sum
      if(sum!=0)                         //系数
      {
        p1->coef=sum;                    //修改
        p3->next=p1; p3=p1;              //将修
        p1=p1->next;                     //p1
        r=p2; p2=p2->next; delete r;     //删除
      }
      else                               //系数
      {
        r=p1; p1=p1->next; delete r;
        r=p2; p2=p2->next; delete r;
      }
   }
   else if(p1->expn<p2->expn)
   {
      p3->next=p1;
      p3=p1;
      p1=p1->next;
   }
   else
   {
      p3->next=p2;
      p3=p2;
      p2=p2->next;
   }
  }
  p3->next=p1?p1:p2;
  delete Pb;
}
```

真题

(2014 填空3)在一个带头结点的单循环链表中，p指向尾节点的直接前驱，则指向头结点的指针head可用p表示为head=_____

(2015 填空2)在单链表中某个节点后插入一个新节点，需要修改_____个节点指针域的值。

(2015 填空4)若带头节点的单链表的头指针为head，则判断链表是否为空的条件是_____

(2016 程序1)指针A，B分别指向两个带头结点的单链表。实现函数

int ListIsEqual(LinkList A, LinkList B)：若A，B中全部对应节点的data值相等，则返回1，否则返回0

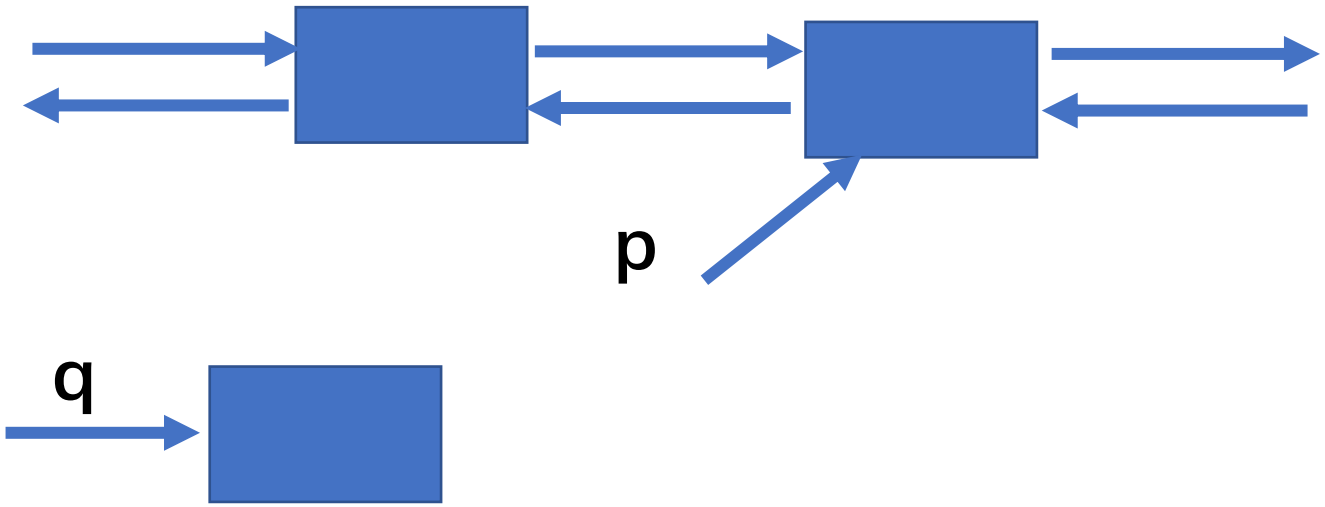
递归

真题（插入，删除）

如果是单链表呢?

(2012 简答1) 有以下双链表，其中有一p指针指向某节点，若有一新节点q要插入在p节点之前，请写出相关的插入语句。

P节点之后；表头；表尾；（作业）



指定节点的前插操作（小技巧）

```
//前插操作：在p结点之前插入元素 e
bool InsertPriorNode (LNode *p, ElemType e){
    if (p==NULL)
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    if (s==NULL)    //内存分配失败
        return false;
    s->next=p->next;
    p->next=s;                  //新结点 s 连到 p 之后
    s->data=p->data;            //将p中元素复制到s中
    p->data=e;                  //p 中元素覆盖为 e
    return true;
}
```

时间复杂度：O(1)

删除指定节点
（P不能为尾节点）

```
//删除指定结点 p
bool DeleteNode (LNode *p)
```

真题（合并）

其他集合运算呢？交，并（作业）

(2013 读程题1)已知无头单链表A和B表示两个集合，实现集合运算

A=A-B

A=AUB

C=A∩B A和B不能动

如果是C=A-B呢？不允许破坏链表A和链表B

删除链表的倒数第N个节点 https://leetcode-cn.com/problems/remove-nth-node-from-end-of-list/

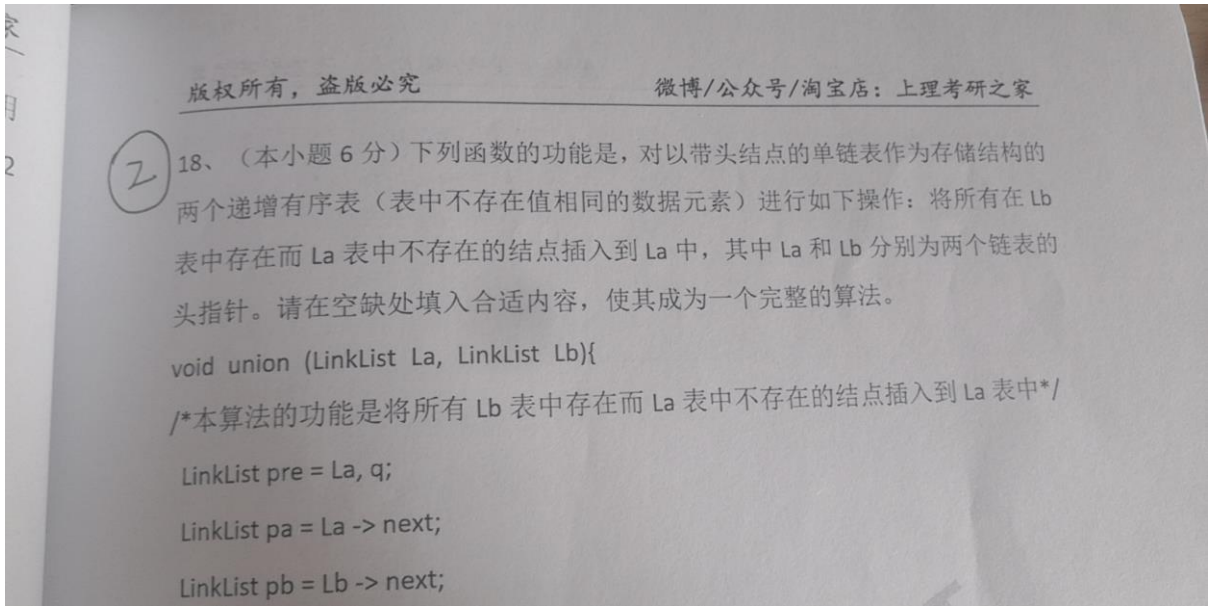
练习题（作业）

已知递增有序的两个单链表A，B分别存储了一个集合。设计算法
实现求两个集合的并集的运算（A∪B） LinkList &C

【例 2-3】 A 和 B 是两个单链表（带表头结点），其中元素递增有序。设计一个算法，将 A 和 B 归并成一个按元素值非递减有序的链表 C，C 由 A 和 B 中的结点组成。

如果是非递增的有序链表C呢?

2017



版权所有，盗版必究　　微博/公众号/淘宝店：上理考研之家

18、（本小题 6 分）下列函数的功能是，对以带头结点的单链表作为存储结构的两个递增有序表（表中不存在值相同的数据元素）进行如下操作：将所有在 Lb 表中存在而 La 表中不存在的结点插入到 La 中，其中 La 和 Lb 分别为两个链表的头指针。请在空缺处填入合适内容，使其成为一个完整的算法。

```
void union (LinkList La, LinkList Lb){
/*本算法的功能是将所有 Lb 表中存在而 La 表中不存在的结点插入到 La 表中*/
 LinkList pre = La, q;
 LinkList pa = La -> next;
 LinkList pb = Lb -> next;
```

合并两个有序链表 https://leetcode-cn.com/problems/merge-two-sorted-lists/

总结

顺序表逆序
二维数组逆序，翻转，转置，旋转
顺序表合并

链表逆序
合并集合
合并有序表
应用——一元多项式