四川師範大學
SICHUAN NORMAL UNIVERSITY

第 2 章　线 性 表

DATA STRUCTURE

计算机科学学院　刘 芳

第2章 线性表

2.1 线性表的定义

2.2 线性表的顺序表示和实现

2.3 线性表的链式表示和实现

2.4 典型示例：一元多项式的表示及相加

四川師範大學
SICHUAN NORMAL UNIVERSITY

2.3 线性表的链式表示和实现

2.3.1 单链表（线性链表）

2.3.2 循环链表

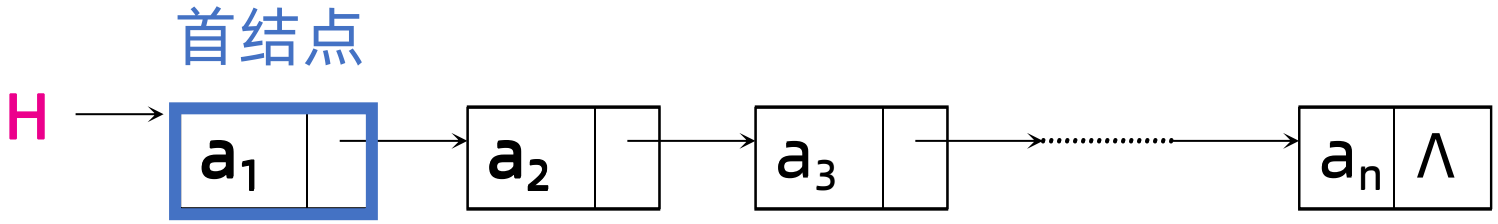
2.3.3 双向链表

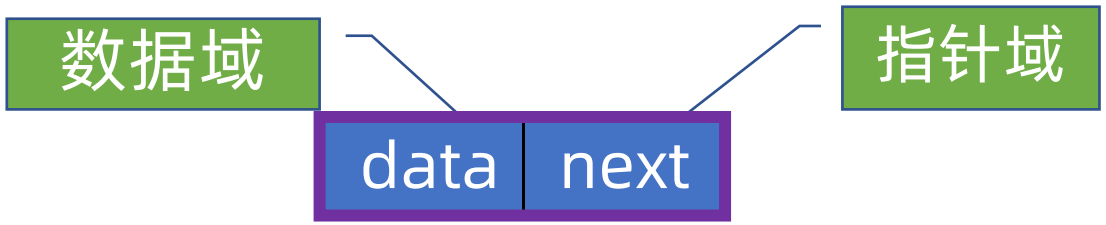
2.3.1 单链表（线性链表）

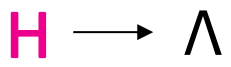
刘　芳 LiuFang

四川師範大學
SICHUAN NORMAL UNIVERSITY

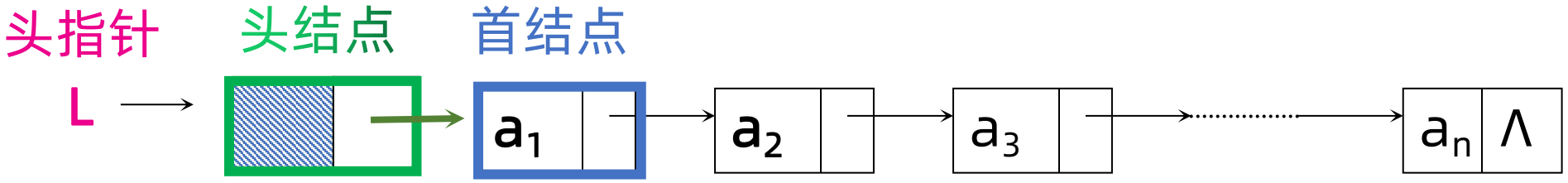
单链表的定义



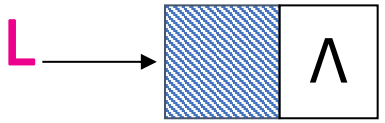




带头结点的单链表



带头结点的非空单链表



带头结点的空单链表

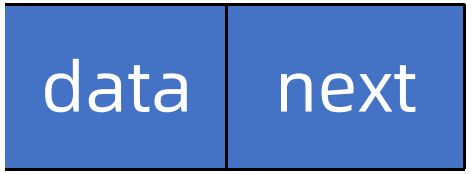
设置头结点的作用：

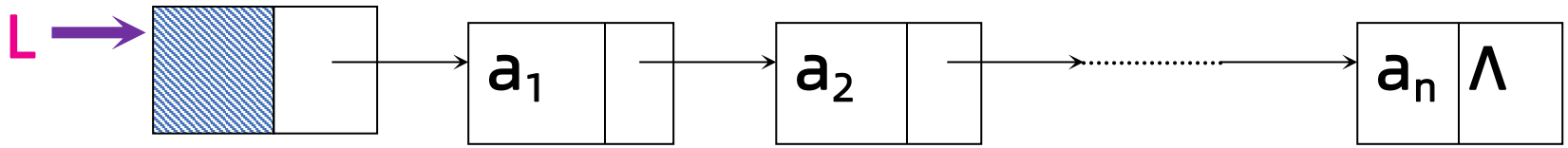
- 主要是使插入和删除等操作统一，在插入或删除第1个结点时不需另作判断。
- 另外，不论链表是否为空，头指针始终指向头结点。

单链表类型的C语言定义

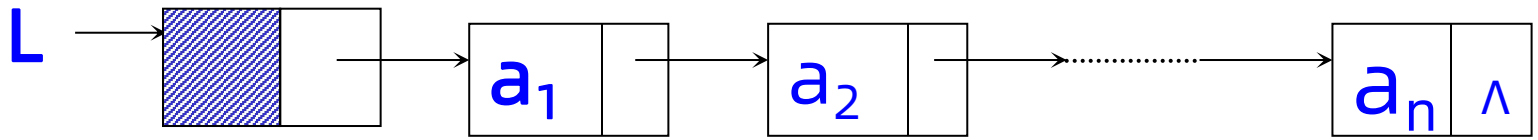
```
typedef  struct  LNode{
    ElemType data;
    struct  LNode *next;
  } LNode,  *  LinkList;

LinkList  L;
```





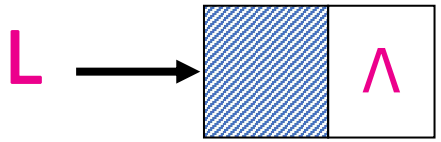
单链表的基本操作



1. 建立 InitList(&L)
2. 输出 OutputList(L)
3. 求长度 ListLength(L)
4. 查找 LocateElem(L,e)
   GetElem(L,i,&e)
5. 插入 ListInsert(&L,i,e)
6. 删除 ListDelete(&L,i,&e)
7. 判空 ListEmpty(L)
8. 表的合并 MergeList(La,Lb,&Lc)
9. 清空表 ClearList(&L)
10. 销毁表 DestroyList(&L)

1.单链表的建立（空表）

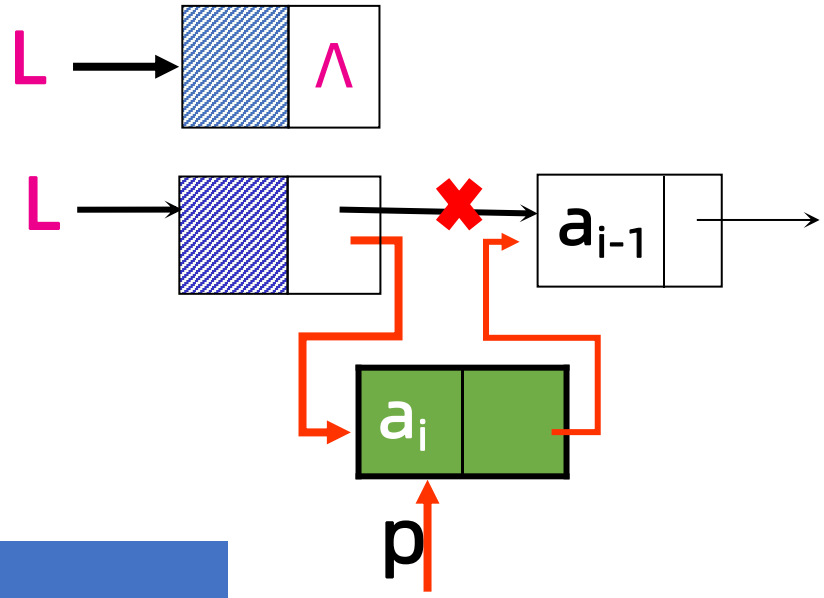
```
Status InitList(LinkList & L){
    L=(LinkList) malloc (sizeof(LNode));
    if (!L) exit(OVERFLOW);
    L->next=NULL;
    return OK;
}
```

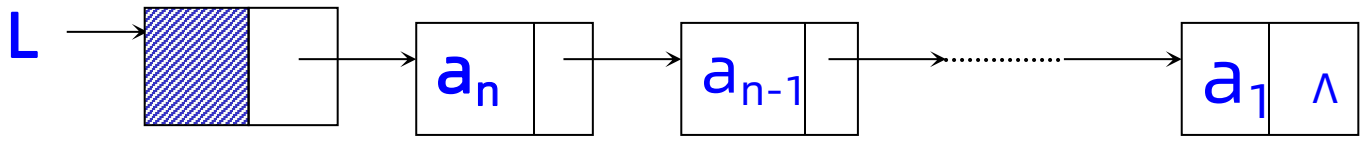


思考：如何建立一个具有n个结点的非空单链表？

1.单链表的建立（头插法建立非空单链表）

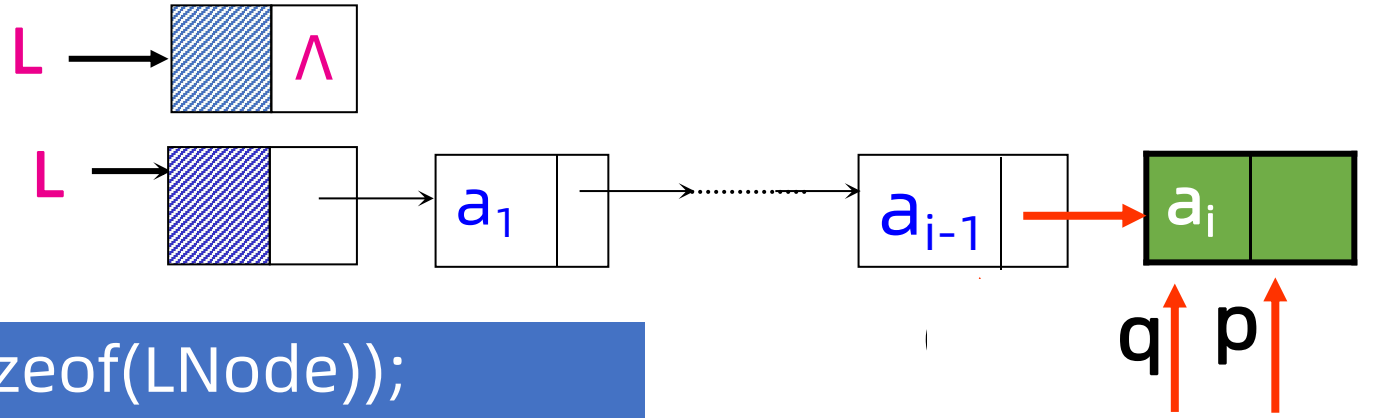
```
Status InitList(LinkList & L){
    L=(LinkList) malloc (sizeof(LNode));
    if (!L) exit(OVERFLOW);
    L->next=NULL;
   cin>>n;
   for (i=1;i<=n; i++){
        p=(LinkList)malloc(sizeof(LNode));
        cin>> p->data;
        p->next=L->next;          L->next=p;
    }
    return OK;
}
```

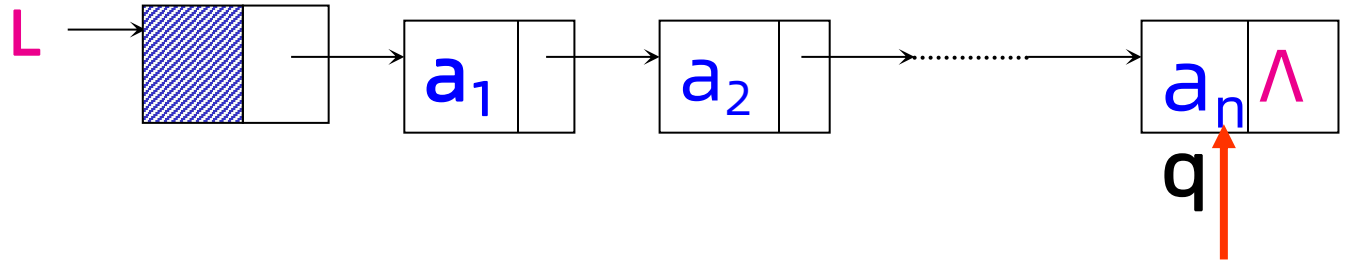




1.单链表的建立（尾插法建立非空单链表）

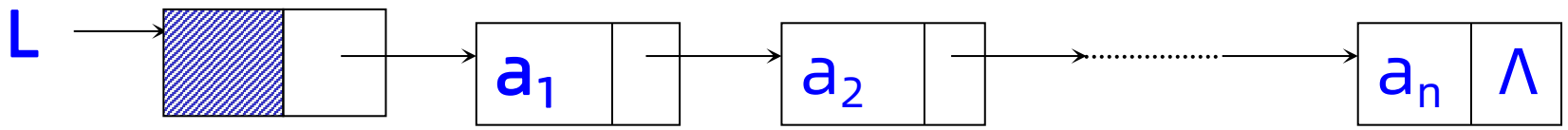
```
Status InitList(LinkList & L){
    L=(LinkList) malloc (sizeof(LNode));
    if (!L) exit(OVERFLOW);
    L->next=NULL;  q=L;
    cin>>n;
    for (i=1;i<=n; i++){
        p=(LinkList)malloc(sizeof(LNode));
        cin>> p->data;
        q->next=p;        q=p;
    }
    q->next=NULL;
    return OK;
    }
```





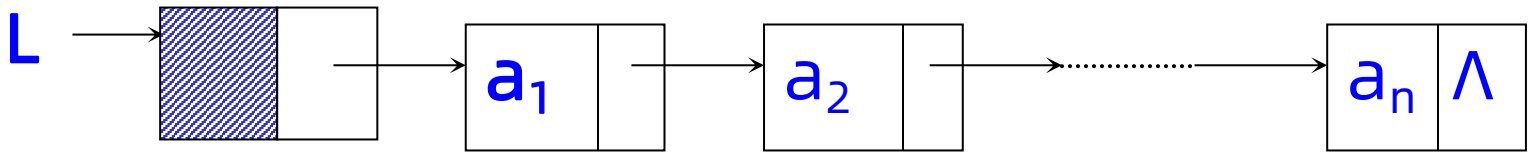
2.输出单链表

```
void OutPut(LinkList  L){
    //基本思想：从首元结点开始，依次输出结点的数据值，直到NULL。
    for (p=L->next ;  p  ; p=p->next)
        cout<<p->data;
    }
```



3.求单链表的长度

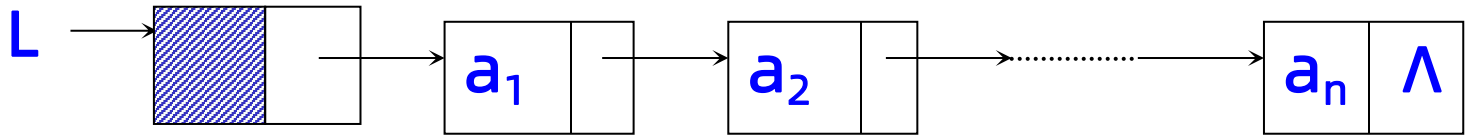
```
int  ListLength(LinkList L){
   //设计数器，初值为0，从首元结点开始遍历链表，并计数
   for (i=0, p=L->next ;  p  ; p=p->next,i++) ;
   return i;
}
```



4.单链表的查找

算法的时间复杂度为O(n)。

- 按值查找
- 按位置查找



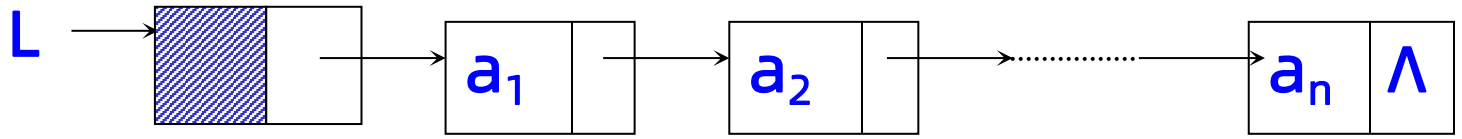

```
LinkList LocateElem(LinkList  L,ElemType e){
    /*从首元结点开始，进行比较操作，若结点的值等于待
    查找值，返回结点的指针，否则指针下移，继续比较。
    若结点为NULL，则查找失败，返回NULL。*/
    for (p=L->next ;  p  ; p=p->next)
        if (p->data==e)  return p;
    return NULL;
}
```

4.单链表的查找

算法的时间复杂度为O(n)。

按值查找

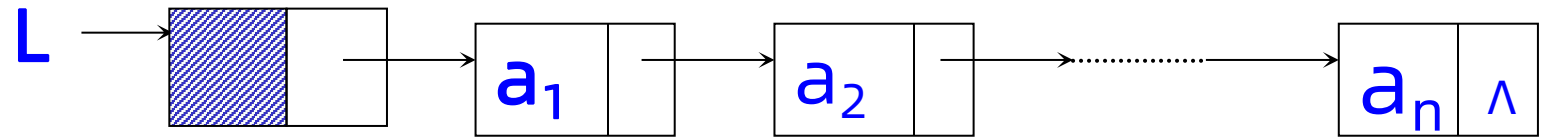
按位置查找



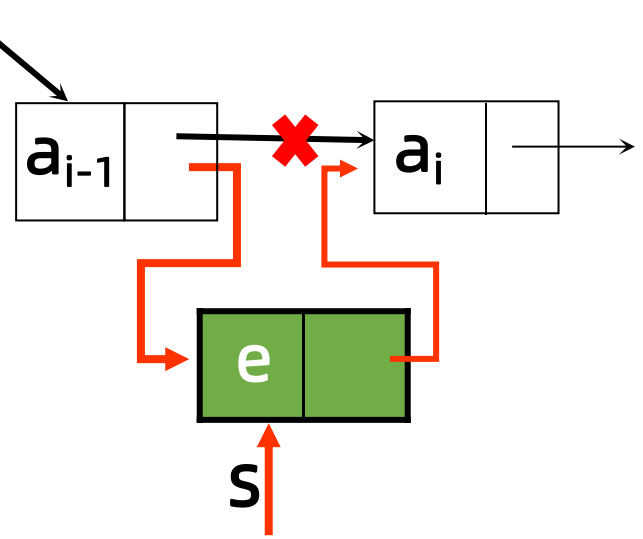

```
Status  GetElem(LinkList  L,int i,ElemType &e){
  /*从首元结点开始，顺链计数。若找到第i个结点，结点的
  值赋给e，函数返回OK，否则返回ERROR。*/
  for (p=L->next，j=1  ；  p &&  j<i   ; j++, p=p->next);
  if (! p ‖  j>i  ) return ERROR;
  e=p->data;
  return OK
}
```

5.单链表的插入

算法的时间复杂度为O(n)。

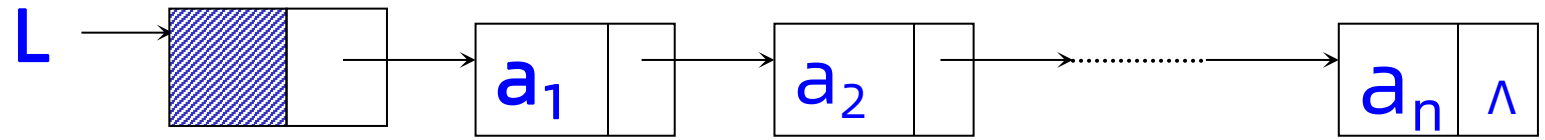



```
Status ListInsert(LinkList &L, int i,ElemType e){
    p=L;  j=0;
    while (p &&  j< i-1) {
        p=p->next;  j++;
    }
    if (!p || j>i-1)  return ERROR;
    s=(LinkList)malloc(sizeof(LNode));  s->data=e;
    s->next=p->next;          p->next=s;
    return OK
}
```

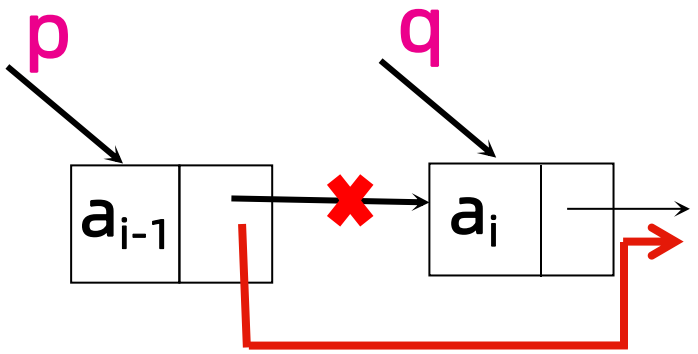


6.单链表的删除

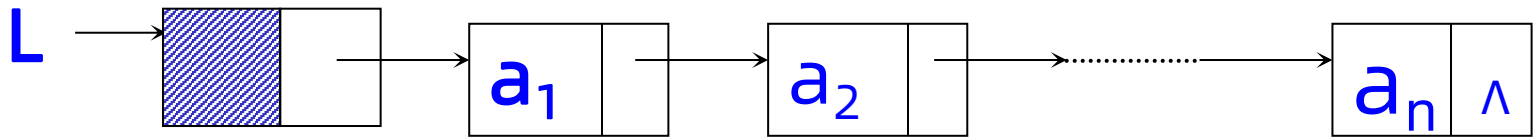
算法的时间复杂度为O(n)。




```
Status ListDelete(LinkList &L,int i,ElemType &e){
    p=L;  j=0;
    while (p->next &&  j< i-1) {
        p=p->next;  j++;
    }
    if (!(p->next) || j>i-1)  return ERROR;
    q=p->next;   e=q->data;   p->next=q->next;
    free (q);
    return OK
}
```



本节要点



1. 建立　　InitList(&L)
2. 输出　　OutputList(L)
3. 求长度　　ListLength(L)
4. 查找　　LocateElem(L,e)
   GetElem(L,i,&e)
5. 插入　　ListInsert(&L,i,e)
6. 删除　　ListDelete(&L,i,&e)
7. 判空　　ListEmpty(L)
8. 表的合并　　MergeList(La,Lb,&Lc)
9. 清空表　　ClearList(&L)
10. 销毁表　　DestroyList(&L)

四川師範大學
SICHUAN NORMAL UNIVERSITY

感谢聆听

业精于勤,荒于嬉;行成于思,毁于随.