上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

第2章 线性表(一)

2.2 线性表的顺序实现

线性表的顺序表示又称为顺序存储结构。

顺序存储：把逻辑上相邻的数据元素存储在物理上相邻的存储单元中的存储结构。

简言之：　逻辑上相邻的元素，物理上也相邻

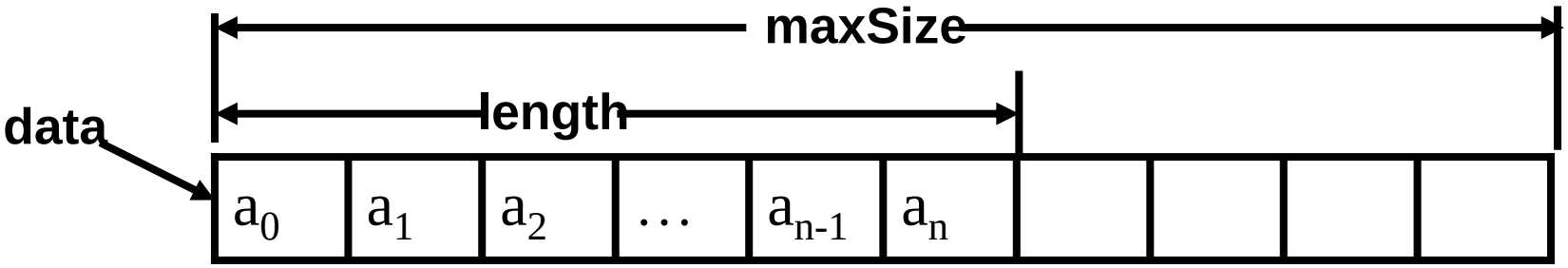
顺序存储方法：用一组地址连续的存储单元依次存储线性表的元素。

可以利用数组V[n]来实现。

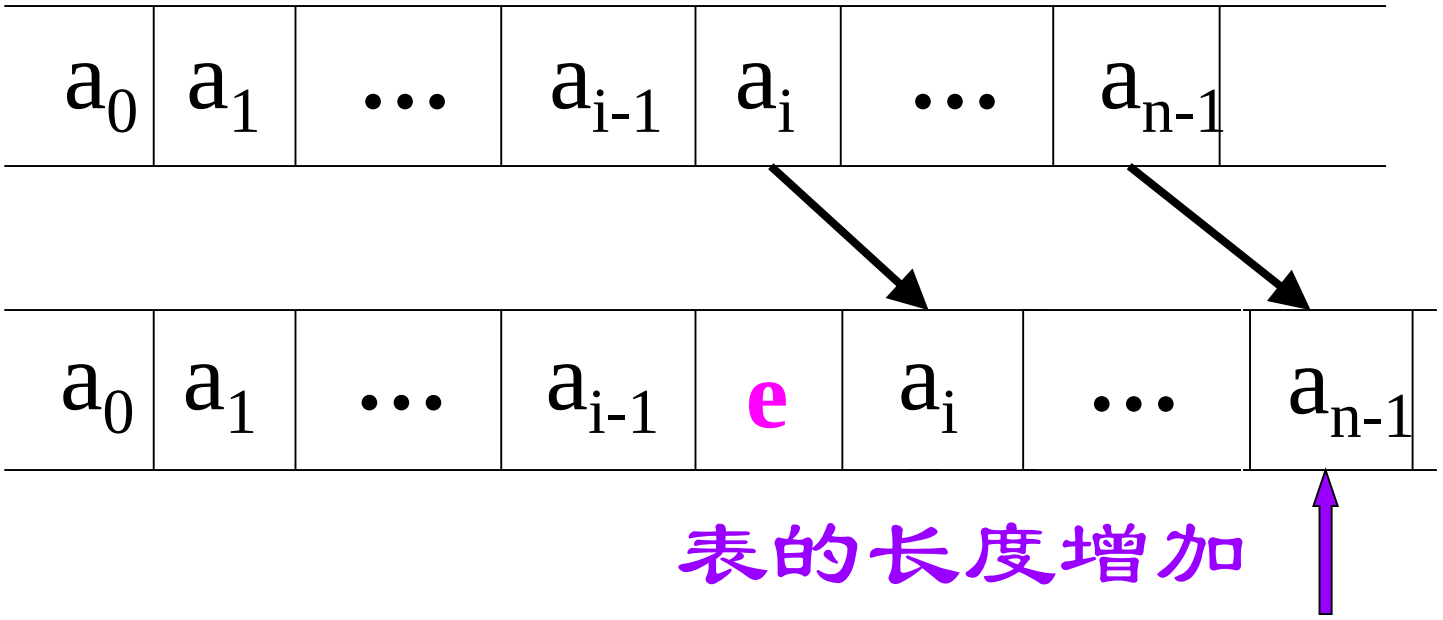
注意：在C++语言中数组的下标是从0开始，即：V[n]的有效范围是从 V[0]～V[n-1]

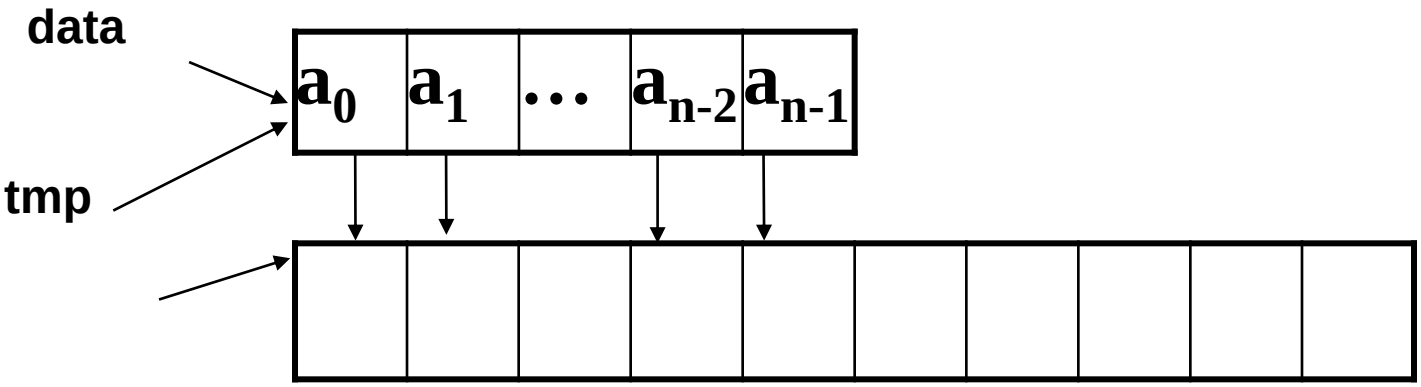
线性表的顺序存储

- 在程序设计语言中，一块连续的存储空间可以用一个数组实现。由于线性表中的元素个数是动态的，因此采用了动态数组。
- 保存一个动态数组，需要三个变量：指向数组起始位置的指针，数组规模（容量），数组中的元素个数（表长）。



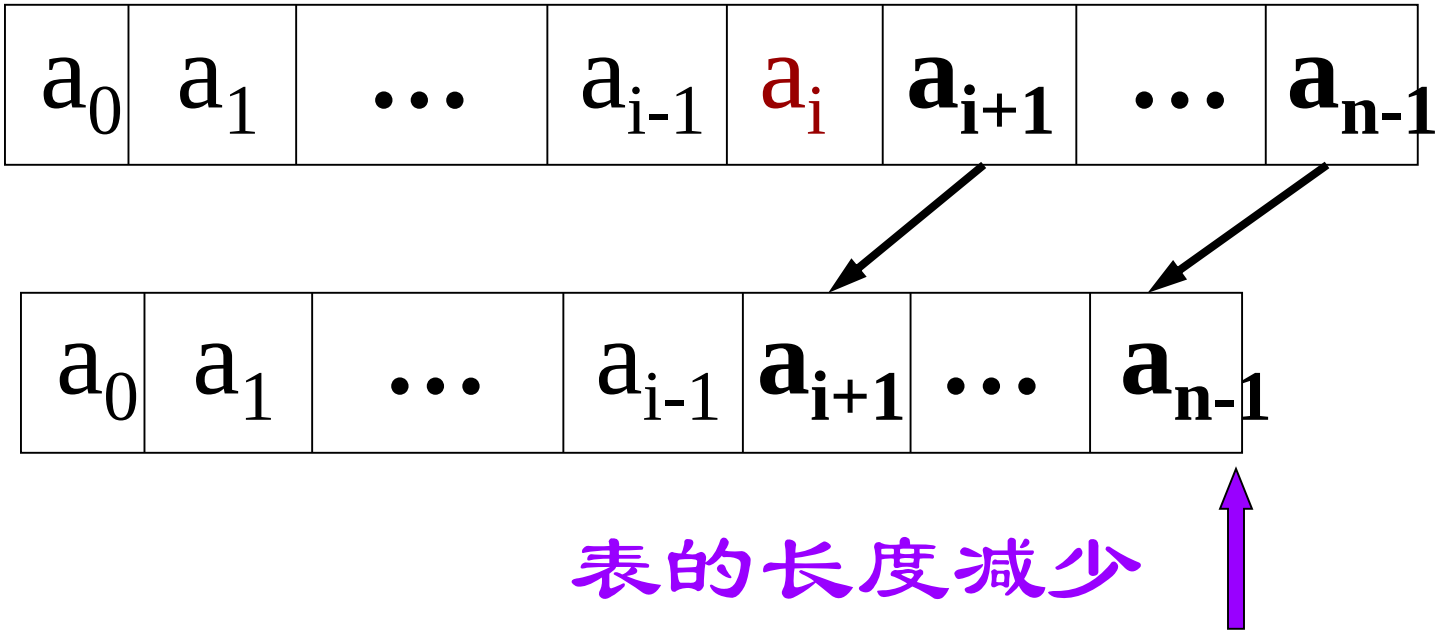
线性表存储结构的变化:






```
void seqList<elemType>::doubleSpace()
{
        elemType *tmp = data;
        maxSize *= 2;
        data = new elemType[maxSize];
        for (int i = 0; i < currentLength; ++i)
            data[i] = tmp[i];
        delete [] tmp;
}
```

线性表存储结构的变化:



顺序实现的算法分析

- length, visit和clear的实现与表中的元素个数无关，因此它们的时间复杂度是O(1)。
- traverse()操作遍历整个表中的所有元素，因此时间复杂度为O(n) 。
- create操作需要申请一块动态数组的空间，并设表为空。因此也是O(1)的时间复杂度。
- 插入操作，需要移动结点。当i等于n时，移动次数为0。当i等于0时，移动次数为n。
  - 最好情况下的时间复杂度为O(1)
  - 最坏情况下的时间复杂度为O(n)
  - 平均的时间复杂度：如果在每个位置上的插入都是等概率的，则插入算法的平均时间复杂度函数为

$$\sum_{i=0}^{n} \frac{1}{n+1}(n-i) = \frac{n}{2}$$

O(n)

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

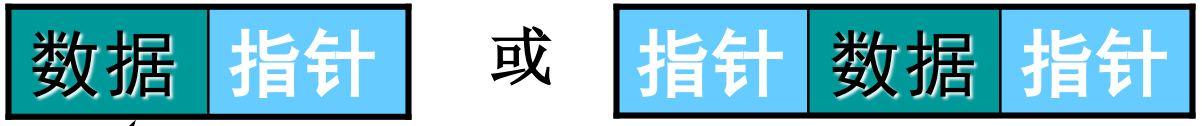
第2章 线性表(二)

2.3 线性表的链接实现

线性表的链式存储结构表示

让每个存储结点都包含两部分：**数据域**和**指针域**

样式：



数据域：存储元素的值

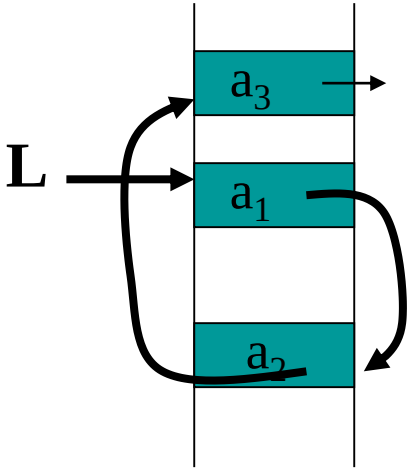
指针域：存储**直接后继**或**直接前驱**元素的存储地址

设计思想：牺牲空间效率换取时间效率

链式存储结构特点

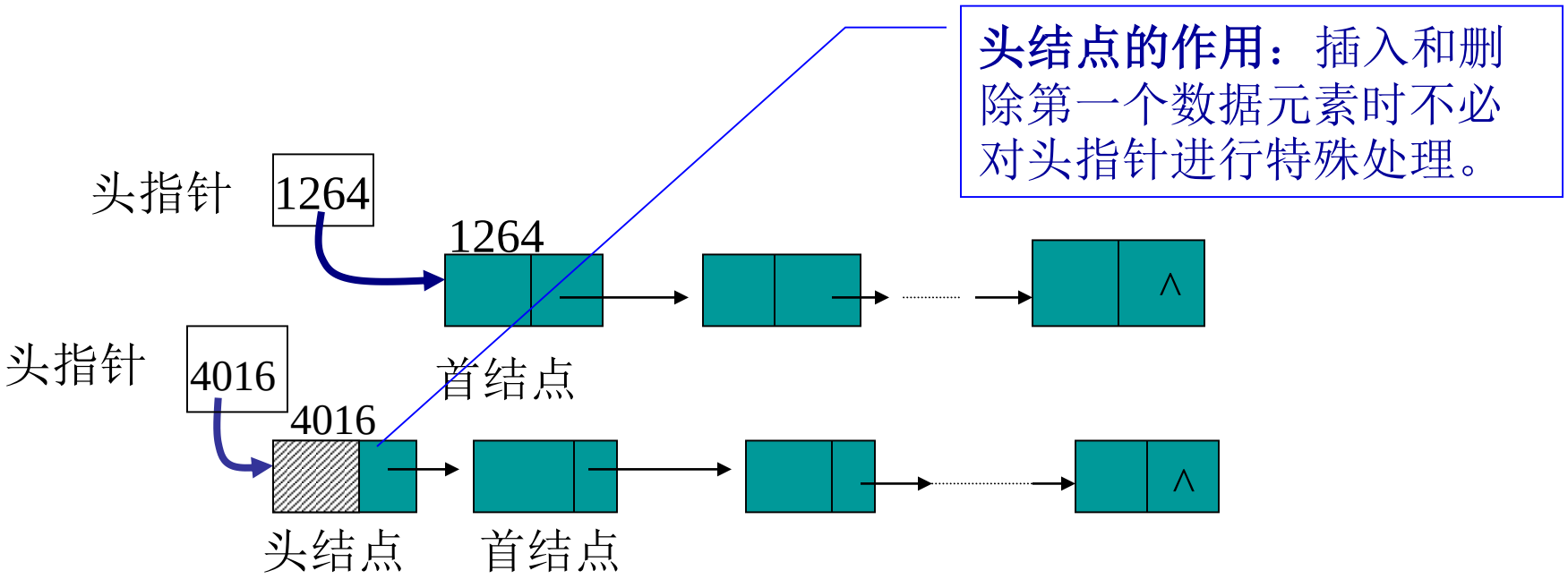
其结点在存储器中的位置是随意的，即逻辑上相邻的数据元素在物理上不一定相邻。

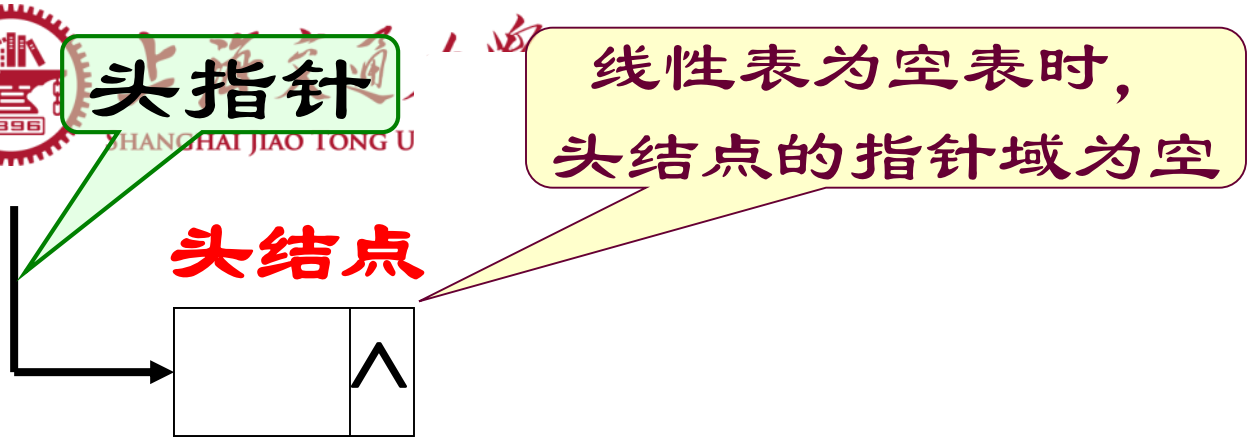
如何实现？ 通过**指针**来实现！



上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

与链式存储有关的术语





以线性表中第一个数据元素a_0的存储地址作为线性表的地址，称作线性表的头指针。

为了操作方便，在第一个结点之前虚加一个“头结点”，以指向头结点的指针为链表的头指针。

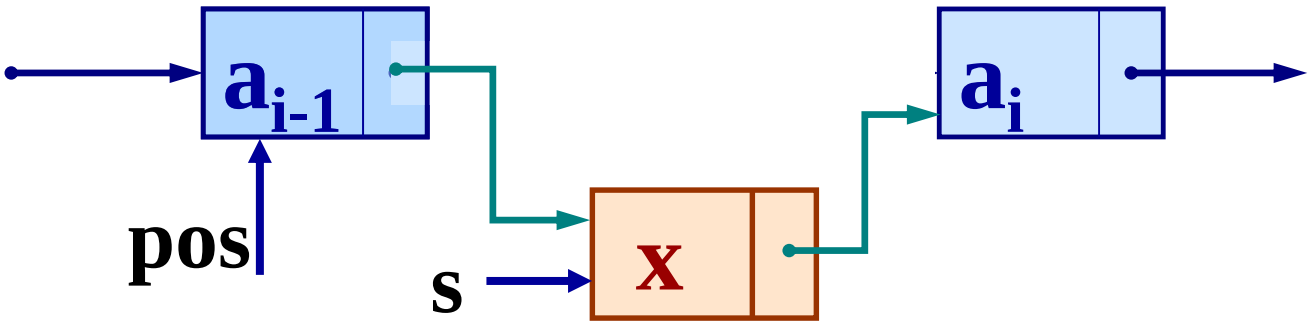
单链表插入insert

1)找到单链表的第i-1个结点并由指针pos指示。

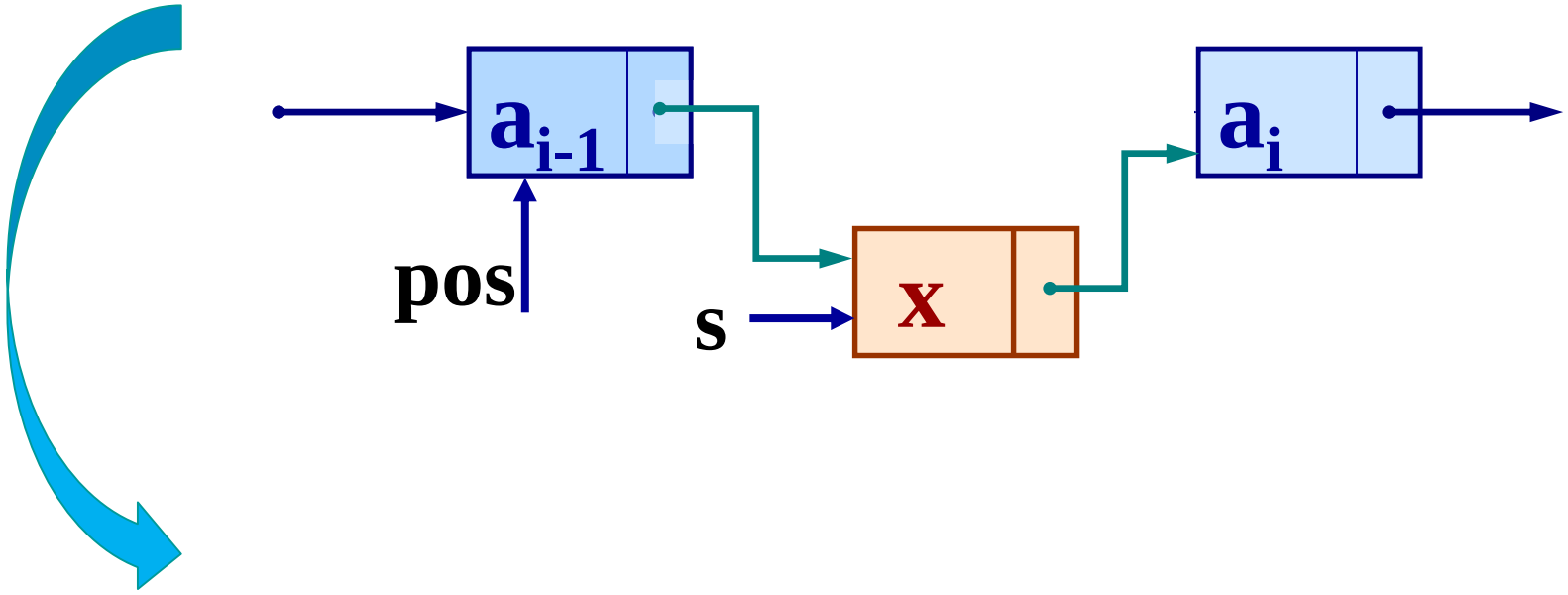
2)然后申请一个新结点并由指针s指示。

3)将pos指示结点的指针值赋给s指示结点的指针域。
(将第i个结点链接到新结点上)

4)然后修改pos指示结点的指针域，使它等于s。
(新结点链接到第i-1个结点上)



```
s=new node;                                    //申请新结点
s->data=x;                                     //将x赋给s的数据域
s->next=pos->next;    pos->next=s;             //链接
```



pos->next=new node(x,pos->next);

单链表插入insert

```
template <class T>
void LinkedList<T>::insert(int i,const T &x){
        node *p=move(i-1);
        p->next=new node(x,p->next);
        ++currentLength;
}
```

单链表删除第i个结点

操作为:找到线性表中第i-1个结点*pos,修改其指针域让它指向第i+1个结点。释放第i个结点的空间。

```
delp= pos->next; pos->next= delp->next;
delete delp;
```

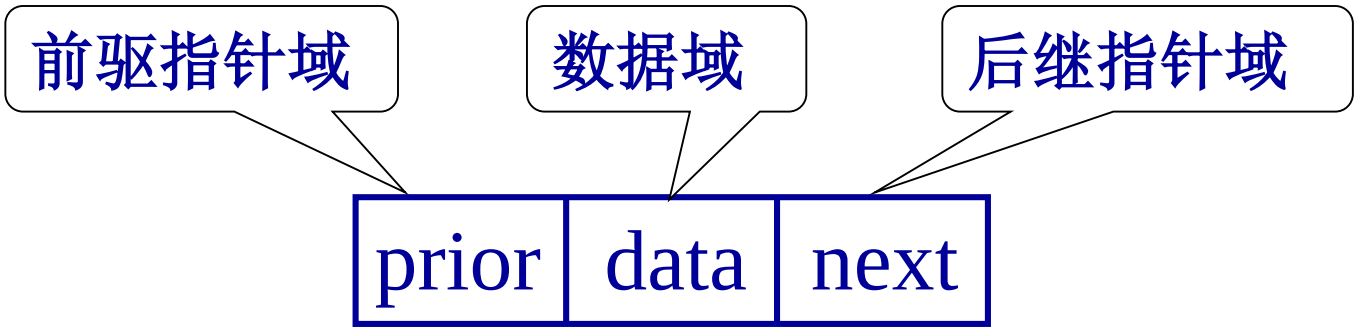
删除第i个位置的元素remove

```
template <class T>
void LinkedList<T>::remove(int i){
        node *pos=move(i-1);
        node *delp=pos->next;
        pos->next=delp->next;
        delete delp;
        --currentLength;
}
```

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

2.3.2 双链表

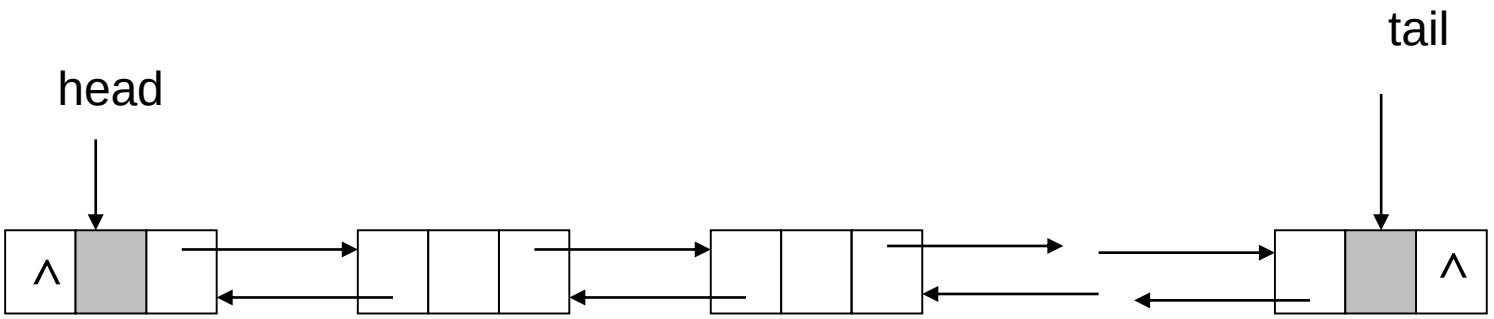
- 每个结点附加了两个指针字段，如prior和next
- prior字段给出直接前驱结点的地址
- next给出直接后继结点的地址。



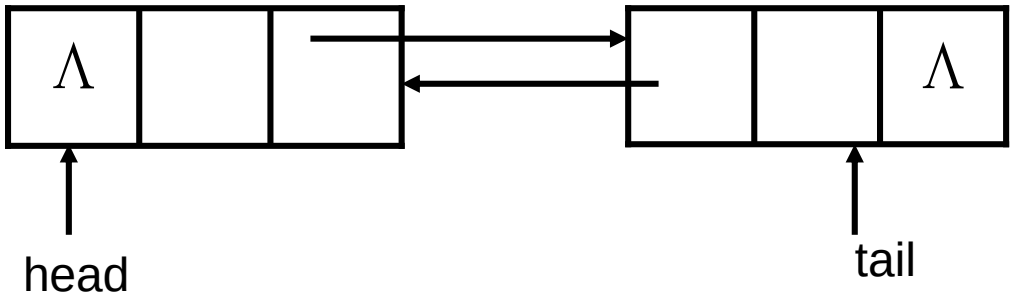
上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

双链表的头尾节点

- 为了消除在表头、表尾插入删除的特殊情况，通常双链表设一头结点，设一尾节点
- 头结点中prior字段为空，它的next字段给出线性表中的首结点的地址
- 尾结点中next字段为空，它的prior字段给出线性表中最后一个节点的地址



构造函数




```
template <class elemType>
linkList<elemType>::linkList()
{   head = new node;
    head->next = tail = new node;
    tail->prev = head;
    currentLength = 0;
}
```

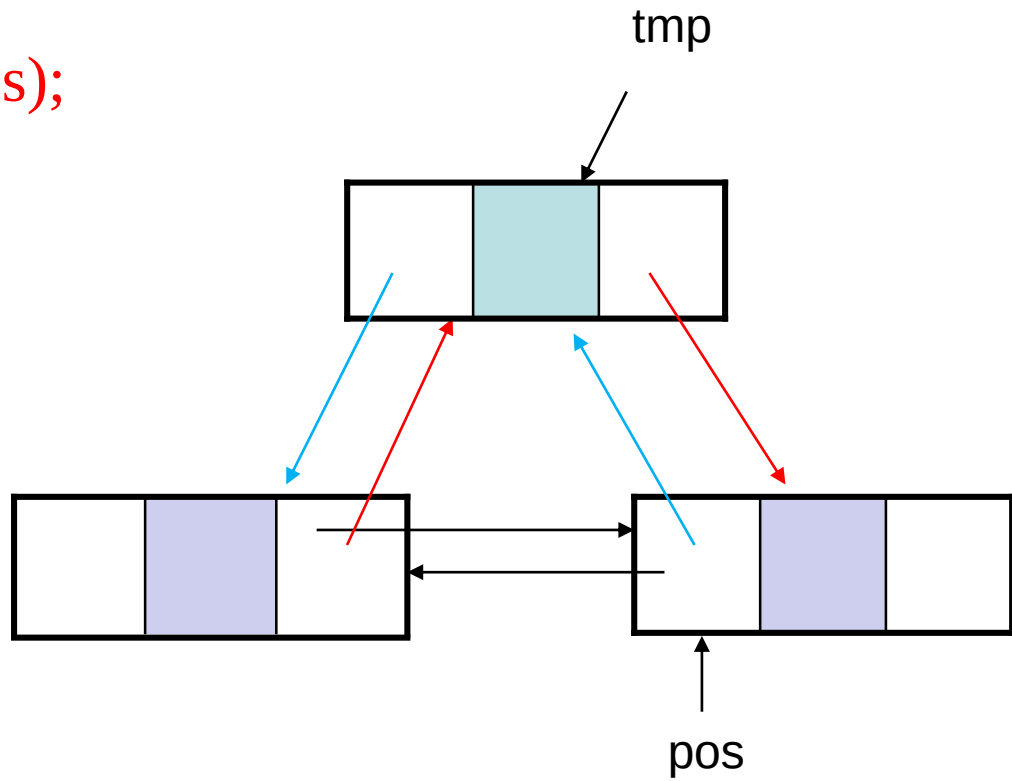
O(1)

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

insert

```
template <class elemType>
void linkList<elemType>::insert(int i,  const elemType &x)
{node *pos, *tmp;
 pos = move(i);
 tmp = new node(x, pos->prev, pos);
 pos->prev->next = tmp;
 pos->prev = tmp;
 ++currentLength;
}
```

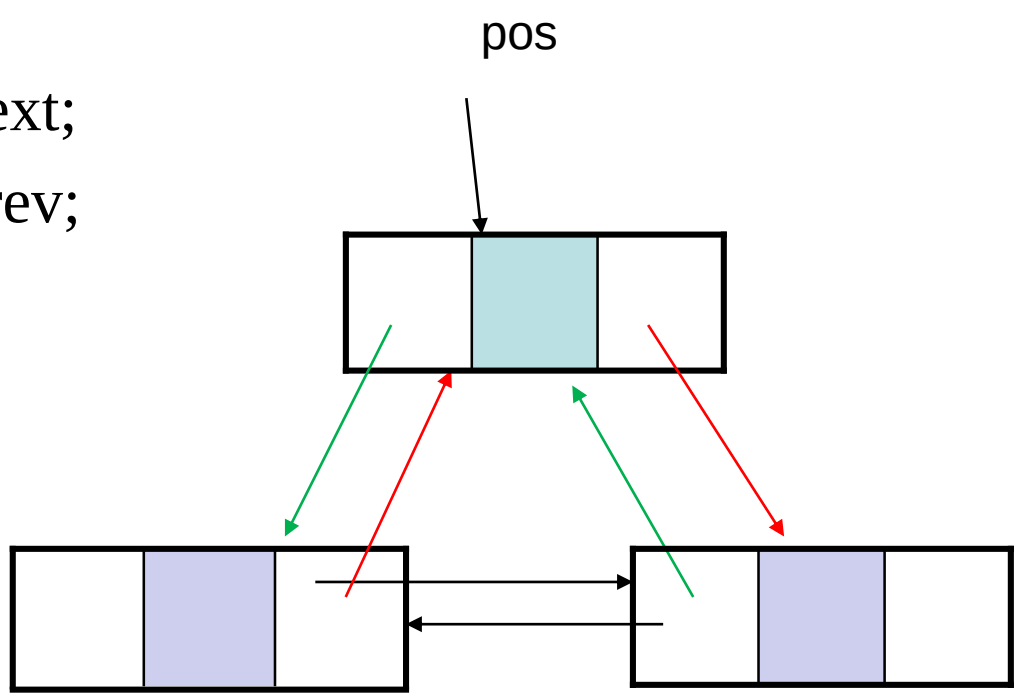
O(n)



remove

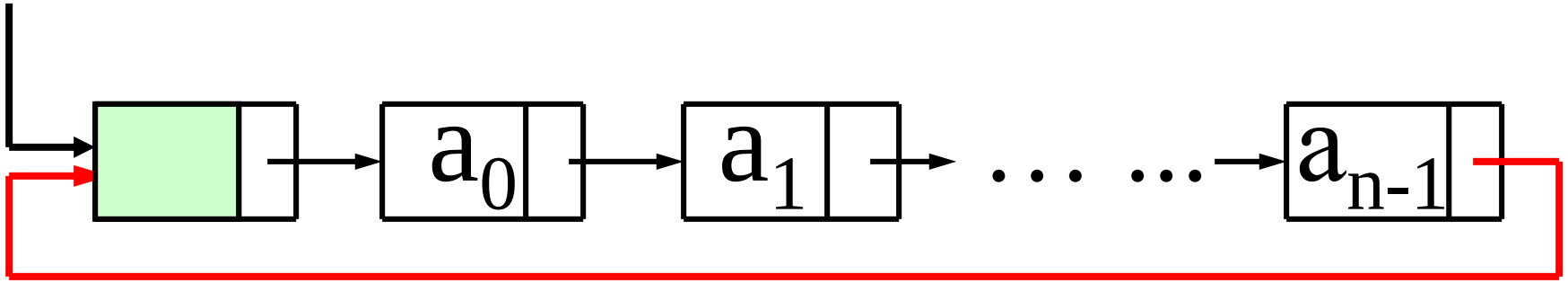
```
template <class elemType>
void linkList<elemType>::remove(int i)
{node *pos;
 pos = move(i);
 pos->prev->next = pos->next;
 pos->next->prev = pos->prev;
 delete pos;
 --currentLength;
}
```

O(n)



2.3.3 循环链表

最后一个结点的指针域的指针又指回第一个结点（或头结点）的链表



和单链表的差别：判别链表中尾结点的条件不再是“后继指针是否为空”，而是“后继指针是否为头指针”。

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

课堂练习

1. 在双向循环链表中，在p指针所指向的结点前插入一个指针q所指向的新结点，其修改指针的操作是_____D_____. [备注：双向链表的结点结构为（prev, data, next）]

A. p -> prev = q; q -> next = p; p -> prev -> next = q; q -> prev = q;

B. p -> prev = q; p -> prev -> next = q; q -> next = p; q -> prev = p -> prev;

C. q -> prev = p -> prev; q -> next = p; p -> prev = q; p -> prev = q;

D. q -> next = p; q -> prev = p -> prev; p -> prev -> next = q; p -> prev = q;

2. 在单链表中，已知其结点的指针场为next。现已知ptr所指结点之后至少存在两个结点；若要交换ptr所指结点之后的两个相邻结点，应执行操作：

ptr2 = ptr -> next -> next;
ptr -> next -> next = ptr2 -> next;
ptr2 -> next = ptr -> next ;
ptr -> next = ptr2;

3. 对于一个头指针为head的带头结点的单链表，判定为空表的条件是( ):

A. head==NULL　　B. head->next==NULL　　C. head->next=head　　D. head!=NULL

参考答案：B

4. 对于一个头指针为head的带头结点的循环链表，判定为空表的条件是( ):

A. head==NULL　　B. head->next==NULL　　C. head->next=head　　D. head!=NULL

参考答案：C

5. 在一个长度为n的顺序表中删除第i(1<= i <= n)个元素时,需向前移动( )个元素:

A. n-i　　B. n-i+1　　C. n-i-1　　D. i

参考答案：A

6. 对于顺序表的优缺点,以下说法错误的是( )
A. 无需为表示结点间的逻辑关系而增加额外的存储空间
B. 可以方便地随机存取表中的任一结点
C. 插入和删除运算较方便
D. 容易造成一部分空间长期闲置而得不到充分利用

参考答案：C

7. 将长度为N的单链表链接在长度为M的单链表之后的算法的时间复杂度为( ):

A. O(1)　　B. O(N)　　C. O(M)　　D. O(M+N)

参考答案：C

8. 以下算法的时间复杂度为( ):

```
void fun (int n){
    int i=1;
    while(I <=n)
        i= i*2;
}
```

A. O(n)　　B. O(n^2)　　C. O(nlogn)　　D. O(logn)

参考答案：D

9. 以下程序段：

```
for (i=n-1;i>0;i--)
    for (j=1;j<=i;j++)
        if (a[j]>a[j+1]){
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
```

其中n为正整数，则语句temp=a[j]在最坏情况下执行频度是( )

A. O(n)　　B. O(nlogn)　　C. O(n^3)　　D. O(n^2)

参考答案：D

10. 下列函数的时间复杂度是

```
int func ( int n){
    int i=0, sum=0;
    while(sum< n) sum += ++ i;
     return i ;
}
```

A. O(log n)　　B. O($n^{1/2}$)　　C. O(n)　　D. O(nlog n)

参考答案：B

11. 下列程序段的时间复杂度是（）

```
count=0;
for(k=1;k<=n;k*=2)
   for(j=1;j<=n;j++)
       count++;
```

A. O(logn)　　B. O(n)　　C. O(nlogn)　　D. O(n^2)

参考答案：C

12．已知表头元素为c的单链表在内存中的存储状态如下表所示。

地址	元素	链接地址
1000H	a	1010H
1004H	b	100CH
1008H	C	1000H
100CH	d	NULL
1010H	e	1004H
1014H		

现将f存放于1014H处并插入到单链表中，若f在逻辑上位于a和e之间，则a，e，f的“链接地 址(next指针)”依次是()

A．1010H，1014H，1004H　　B．1010H，1004H，1014H

C．1014H，1010H，1004H　　D．1014H，1004H，1010H

参考答案：D

13．已知一个带有表头结点的双向循环链表L，结点结构为 prev-data-next ，其中，prev和next分别是指向其直接前驱和直接后继结点的指针。现要删除指针p所指的结点，正确的语句序列是( )

A. p->next->prev=p->prev; p->prev->next=p->prev; delete p;

B. p->next->prev=p->next; p->prev-> next=p->next; delete p;

C. p->next->prev=p->next; p->prev->next=p->prev; delete p;

D. p-> next-> prev=p->prev; p->prev->next=p->next; delete p;

参考答案：D