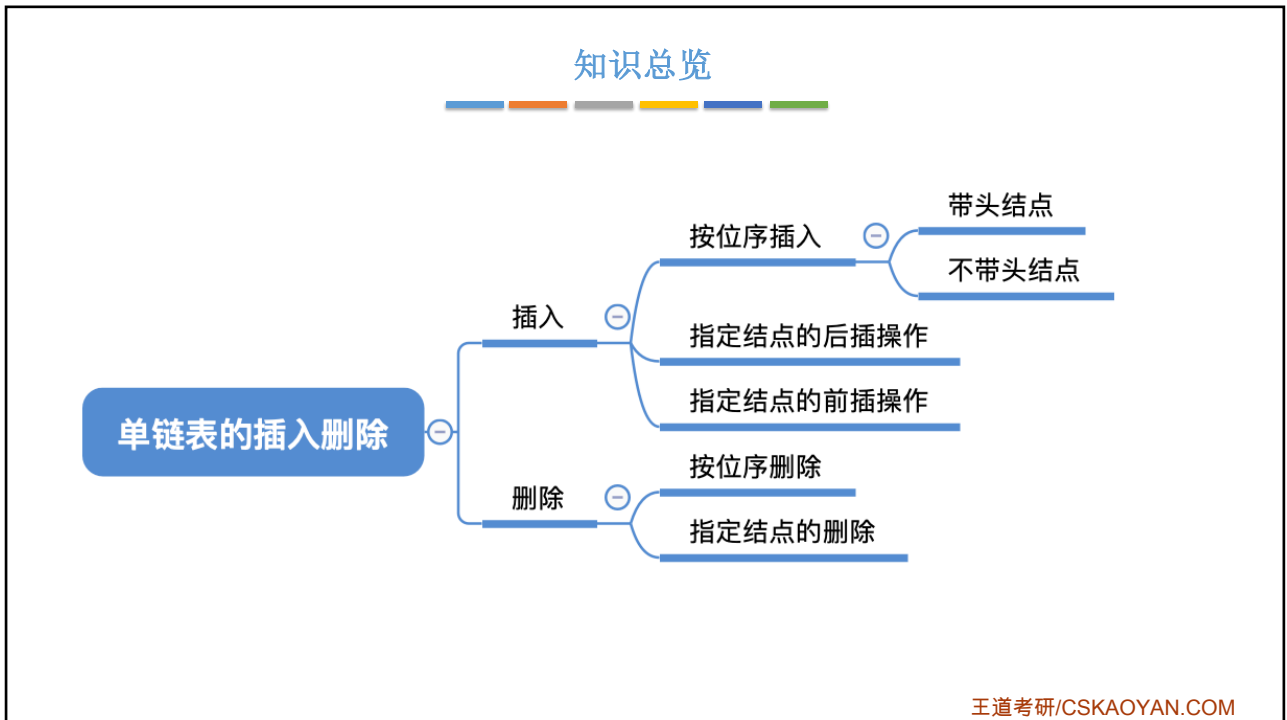


本节内容

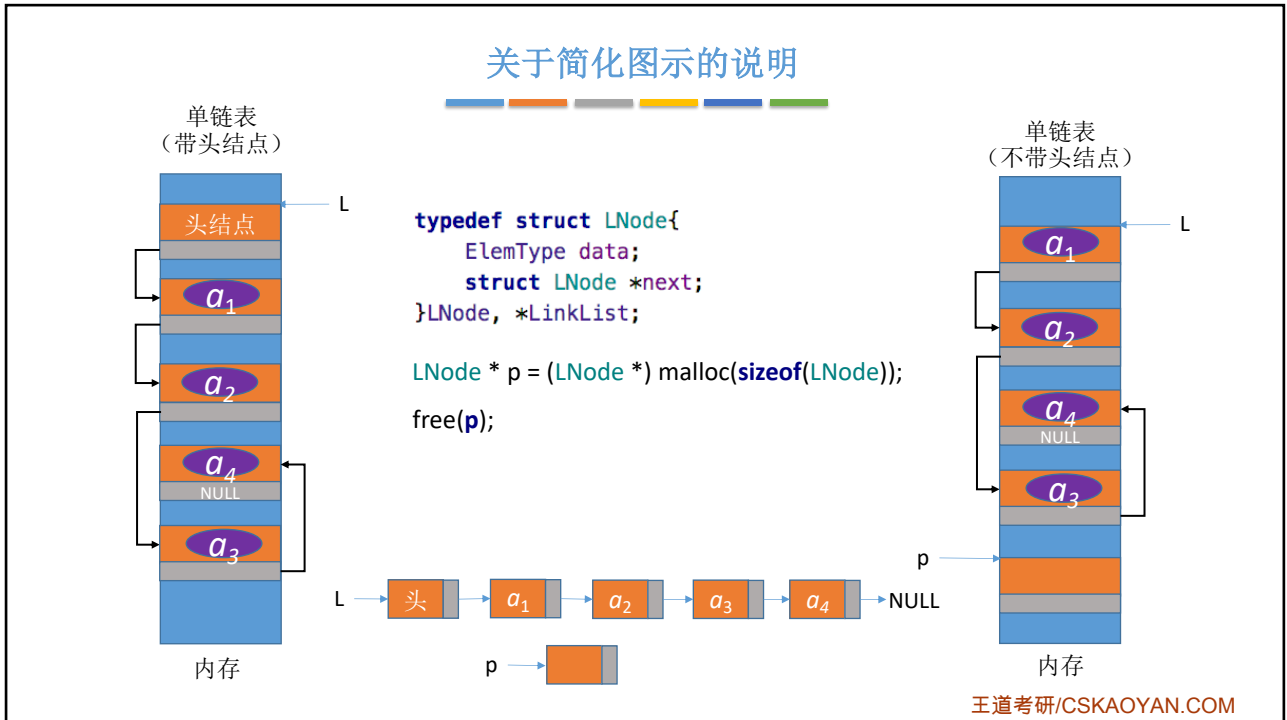
单链表
插入和删除

王道考研/CSKAOYAN.COM

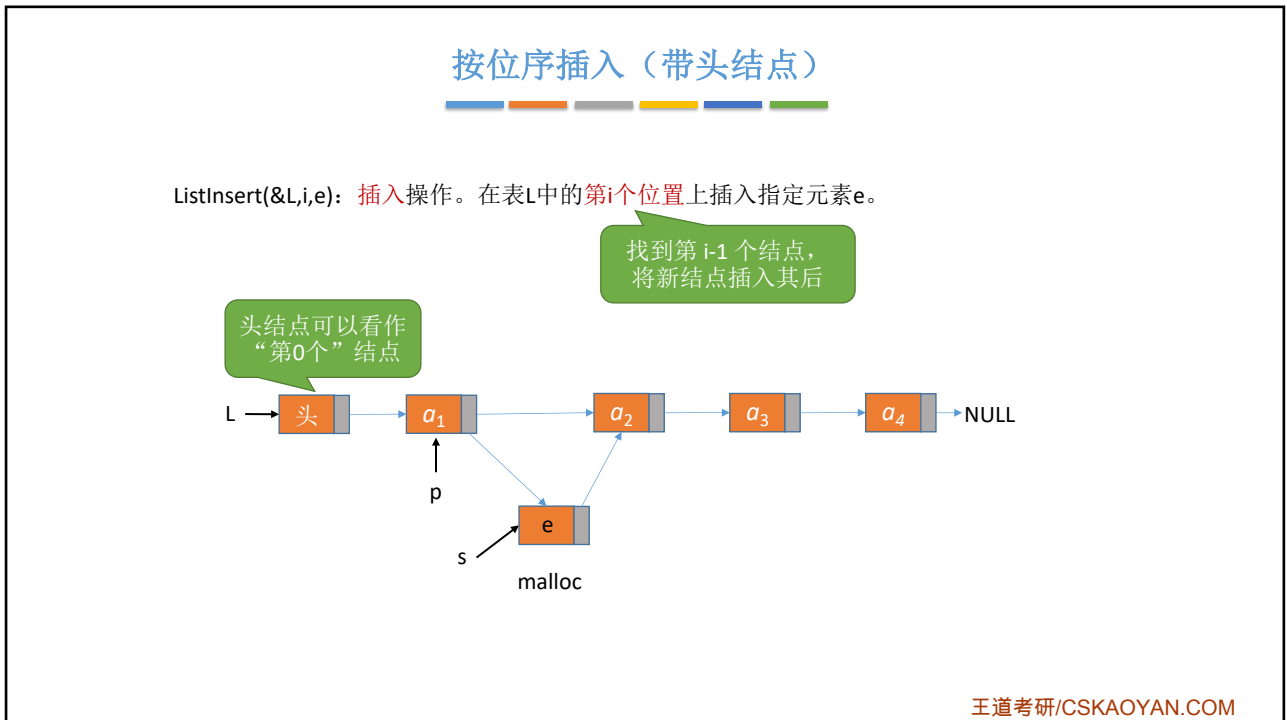
1



2



3



4

按位序插入（带头结点）

//在第 i 个位置插入元素 e （带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p;    //指针p指向当前扫描到的结点
    int j=0;      //当前p指向的是第几个结点
    p = L;        //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL)    //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;      //将结点s连到p之后
    return true;    //插入成功
}

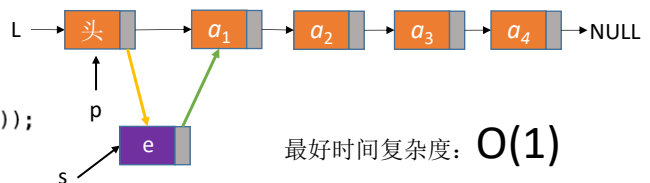
```

```

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

```

分析:

①如果 $i=1$ （插在表头）最好时间复杂度: $O(1)$

注意: 绿绿和黄黄顺序不能颠倒鸭!

王道考研/CSKAOYAN.COM

5

按位序插入（带头结点）

//在第 i 个位置插入元素 e （带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p;    //指针p指向当前扫描到的结点
    int j=0;      //当前p指向的是第几个结点
    p = L;        //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL)    //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;      //将结点s连到p之后
    return true;    //插入成功
}

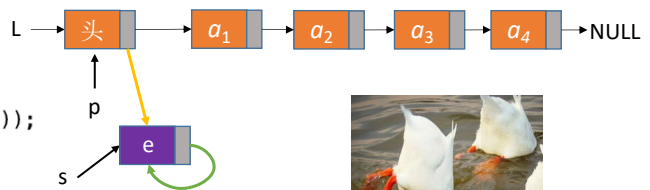
```

```

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

```

分析:

①如果 $i=1$ （插在表头）

注意: 绿绿和黄黄顺序不能颠倒鸭!



王道考研/CSKAOYAN.COM

6

按位序插入（带头结点）

//在第 i 个位置插入元素 e （带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p;    //指针p指向当前扫描到的结点
    int j=0;      //当前p指向的是第几个结点
    p = L;        //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;    //将结点s连到p之后
    return true;  //插入成功
}

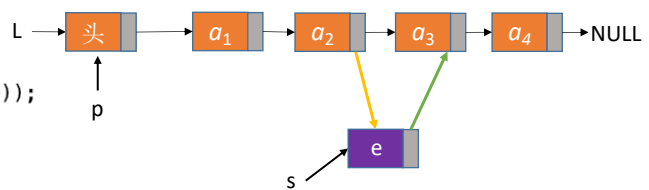
```

```

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

```

分析：
②如果 $i=3$ （插在表中）



王道考研/CSKAOYAN.COM

7

按位序插入（带头结点）

//在第 i 个位置插入元素 e （带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p;    //指针p指向当前扫描到的结点
    int j=0;      //当前p指向的是第几个结点
    p = L;        //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;    //将结点s连到p之后
    return true;  //插入成功
}

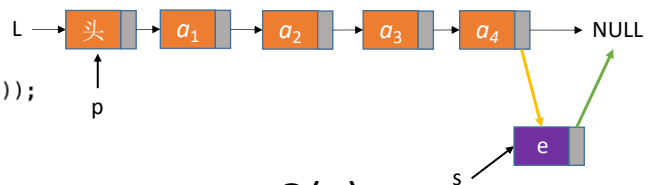
```

```

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

```

分析：
③如果 $i=5$ （插在表尾）

最坏时间复杂度: $O(n)$

王道考研/CSKAOYAN.COM

8

按位序插入（带头结点）

//在第 i 个位置插入元素 e （带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p;    //指针p指向当前扫描到的结点
    int j=0;      //当前p指向的是第几个结点
    p = L;        //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL)    //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;      //将结点s连到p之后
    return true;    //插入成功
}

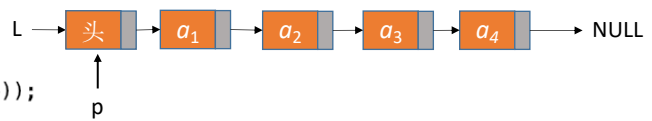
```

```

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

```

分析:

④如果 $i = 6$ ($i > \text{Length}$)

王道考研/CSKAOYAN.COM

9

按位序插入（带头结点）

//在第 i 个位置插入元素 e （带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p;    //指针p指向当前扫描到的结点
    int j=0;      //当前p指向的是第几个结点
    p = L;        //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL)    //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;      //将结点s连到p之后
    return true;    //插入成功
}

```

```

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

```

平均时间复杂度: $O(n)$

原来如此，简单！

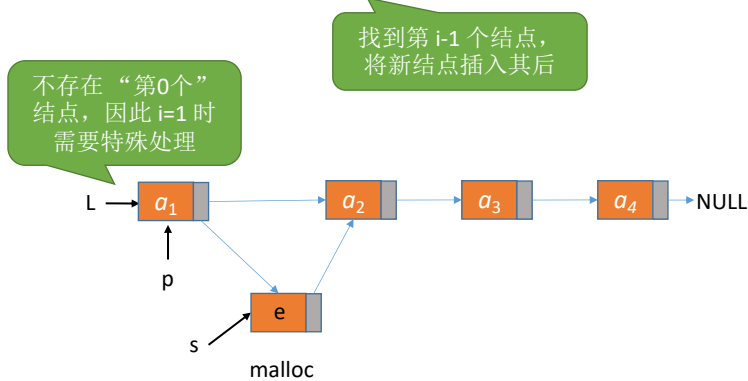


王道考研/CSKAOYAN.COM

10

按位序插入（不带头结点）

ListInsert(&L,i,e): 插入操作。在表L中的第*i*个位置上插入指定元素e。



王道考研/CSKAOYAN.COM

11

按位序插入（不带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    if(i==1){ //插入第1个结点的操作与其他结点操作不同
        LNode *s = (LNode *)malloc(sizeof(LNode));
        s->data = e;
        s->next=L;
        L=s; //头指针指向新结点
        return true;
    }
    LNode *p; //指针p指向当前扫描到的结点
    int j=1; //当前p指向的是第几个结点
    p = L; //p指向第1个结点（注意：不是头结点）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;
    return true; //插入成功
}

```

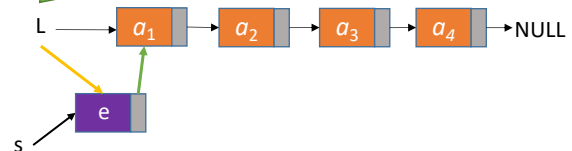
```

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

```

分析：
①如果 $i = 1$ （插在表头）

如果不带头结点，则插入、删除第1个元素时，需要更改头指针L



王道考研/CSKAOYAN.COM

12

按位序插入（不带头结点）

```

bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    if(i==1){ //插入第1个结点的操作与其他结点操作不同
        LNode *s = (LNode *)malloc(sizeof(LNode));
        s->data = e;
        s->next=L;
        L=s; //头指针指向新结点
        return true;
    }
    LNode *p; //指针p指向当前扫描到的结点
    int j=1; //当前p指向的是第几个结点
    p = L; //p指向第1个结点(注意:不是头结点)
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值不合法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s;
    return true; //插入成功
}

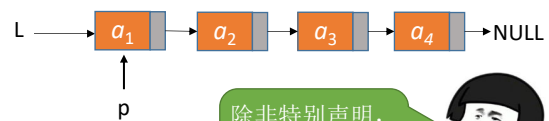
```

```

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

```

分析:
②如果 $i > 1...$



后续逻辑和带头结点的一样

除非特别声明，之后的代码默认带头结点



结论: 不带头结点写代码更不方便, 推荐用带头结点
注意: 考试中带头、不带头都有可能考察, 注意审题

王道考研/CSKAOYAN.COM

13

指定结点的后插操作

```

//后插操作: 在p结点之后插入元素 e
bool InsertNextNode (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->data = e; //用结点s保存数据元素e
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true;
}

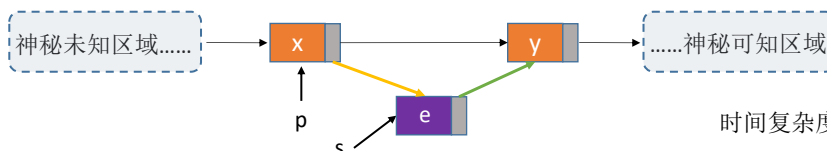
```

```

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

```

某些情况下有可能分配失败 (如内存不足)



时间复杂度: $O(1)$

王道考研/CSKAOYAN.COM

14

指定结点的后插操作

```
//在第 i 个位置插入元素 e (带头结点)
bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p; //指针p指向当前扫描到的结点
    int j=0; //当前p指向的是第几个结点
    p = L; //L指向头结点, 头结点是第0个结点 (不存数据)
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值不合法
        return InsertNextNode(p, e);
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true; //插入成功
}
```

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

```
//后插操作: 在p结点之后插入元素 e
bool InsertNextNode (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->data = e; //用结点s保存数据元素e
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true;
}
```

王道考研/CSKAOYAN.COM

15

指定结点的前插操作

```
//前插操作: 在p结点之前插入元素 e
bool InsertPriorNode (LNode *p, ElemType e)
```

如何找到 p 结点的前驱结点?

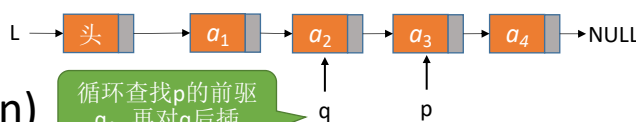


笑容被打上马赛克



```
//前插操作: 在p结点之前插入元素 e
bool InsertPriorNode (LinkList L, LNode *p, ElemType e)
```

传入头指针

时间复杂度: $O(n)$

循环查找p的前驱q, 再对q后插

高清无码
拒绝神秘

王道考研/CSKAOYAN.COM

16

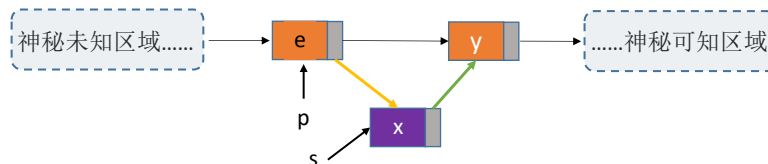
指定结点的前插操作

//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;
    s->data=p->data;
    p->data=e;
    return true;
}

```

时间复杂度: $O(1)$ 

王道考研/CSKAOYAN.COM

17

指定结点的前插操作

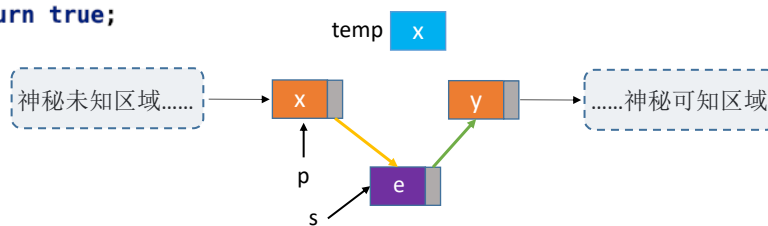
//前插操作: 在p结点之前插入结点 s

```

bool InsertPriorNode (LNode *p, LNode *s){
    if (p==NULL || s==NULL)
        return false;
    s->next=p->next;
    p->next=s;
    ElemType temp=p->data; //s连到p之后
    p->data=s->data; //交换数据域部分
    s->data=temp;
    return true;
}

```

王道书版本

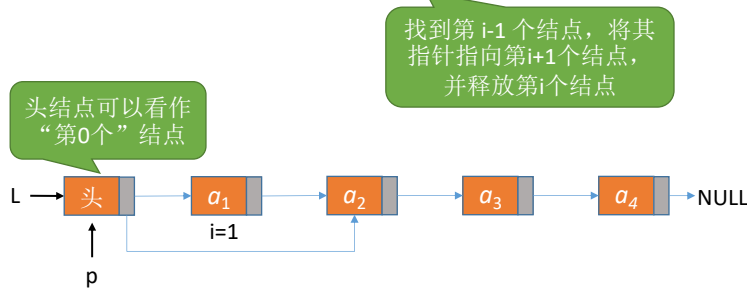


王道考研/CSKAOYAN.COM

18

按位序删除（带头结点）

ListDelete(&L,i,&e): 删除操作。删除表L中第*i*个位置的元素，并用e返回删除元素的值。



王道考研/CSKAOYAN.COM

19

按位序删除（带头结点）

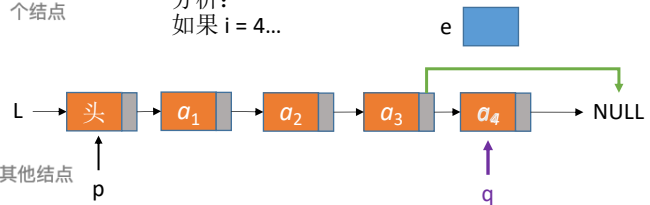
```

bool ListDelete(LinkList &L, int i, ElemType &e){
    if(i<1)
        return false;
    LNode *p; //指针p指向当前扫描到的结点
    int j=0; //当前p指向的是第几个结点
    p = L; //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值不合法
        return false;
    if(p->next == NULL) //第i-1个结点之后已无其他结点
        return false;
    LNode *q=p->next; //令q指向被删除结点
    e = q->data; //用e返回元素的值
    p->next=q->next; //将*q结点从链中“断开”
    free(q); //释放结点的存储空间
    return true; //删除成功
}
  
```

```

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

分析：
如果 $i = 4 \dots$



最坏、平均时间复杂度: $O(n)$
最好时间复杂度: $O(1)$

如果不带头结点，删除第1个元素，是否需要特殊处理？

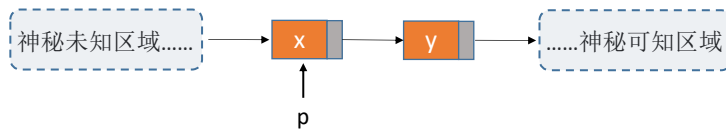


王道考研/CSKAOYAN.COM

20

指定结点的删除

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



我这暴脾气

删除结点p，需要修改其前驱结点的 next 指针

方法1：传入头指针，循环寻找 p 的前驱结点
方法2：偷天换日（类似于结点前插的实现）

王道考研/CSKAOYAN.COM

21

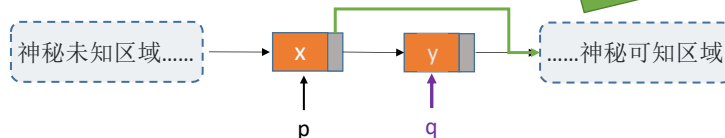
指定结点的删除

```
//删除指定结点 p
bool DeleteNode (LNode *p){
    if (p==NULL)
        return false;
    LNode *q=p->next; //令q指向*p的后继结点
    p->data=p->next->data; //和后继结点交换数据域
    p->next=q->next; //将*q结点从链中“断开”
    free(q); //释放后继结点的存储空间
    return true;
}
```



我可真聪明

可能是指向一个结点，
也可能是指向NULL



时间复杂度：O(1)

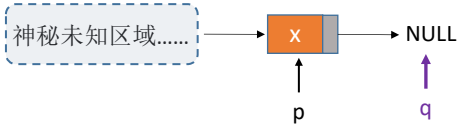
王道考研/CSKAOYAN.COM

22

指定结点的删除

```
//删除指定结点 p
bool DeleteNode (LNode *p){
    if (p==NULL)
        return false;
    LNode *q=p->next; //令q指向*p的后继结点
    p->data=p->next->data; //和后继结点交换数据域
    p->next=q->next; //将*q结点从链中“断开”
    free(q); //释放后继结点的存储空间
    return true;
}
```

单链表的局限性：
无法逆向检索，有
时候不太方便



如果p是最后
一个结点...

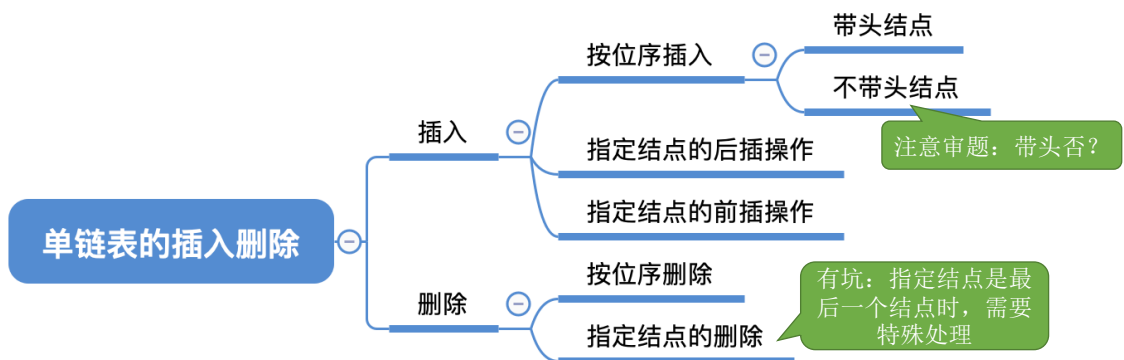
只能从表头开始依
次寻找p的前驱，
时间复杂度 $O(n)$



王道考研/CSKAOYAN.COM

23

知识回顾与重要考点



Tips:

1. 这些代码都要会写，都重要
2. 打牢基础，慢慢加速
3. 体会带头结点、不带头结点的代码区别
4. 体会“封装”的好处

王道考研/CSKAOYAN.COM

24

封装的好处

小功能模块化，
代码逻辑清晰

//在第 i 个位置插入元素 e (带头结点)

```
bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p; //指针p指向当前扫描到的结点
    int j=0; //当前p指向的是第几个结点
    p = L; //L指向头结点，头结点是第0个结点（不存数据）
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值不合法
        return InsertNextNode(p, e);
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true; //插入成功
}
```

指针p指向第 i-1 个结点

p后插入新元素e

也可以称为“基本操作”。对
书本的概念理解不要教条化

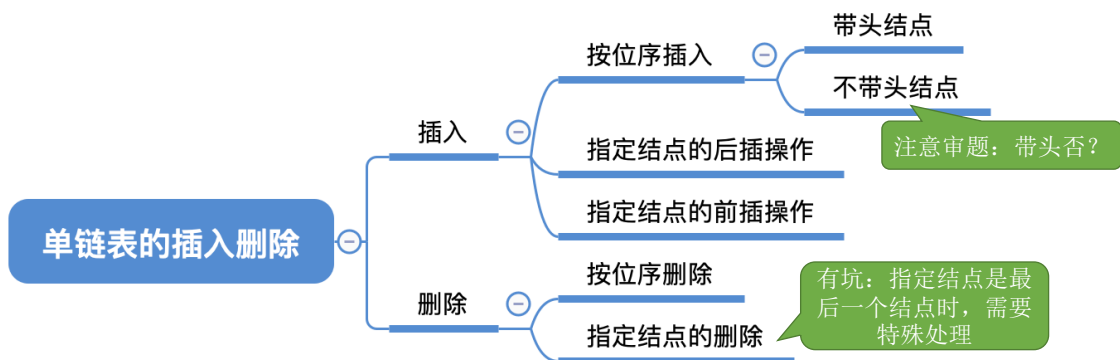
//后插操作：在p结点之后插入元素 e

```
bool InsertNextNode (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->data = e; //用结点s保存数据元素e
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true;
}
```

王道考研/CSKAOYAN.COM

25

知识回顾与重要考点



Tips:

1. 这些代码都要会写，都重要
2. 打牢基础，慢慢加速
3. 体会带头结点、不带头结点的代码区别
4. 体会“封装”的好处

王道考研/CSKAOYAN.COM

26



@王道论坛



等撩

@王道计算机考研备考
@王道咸鱼老师-计算机考研
@王道楼楼老师-计算机考研



等撩



@王道计算机考研



@王道计算机考研



微信视频号

@王道计算机考研



微信公众平台

@王道在线