

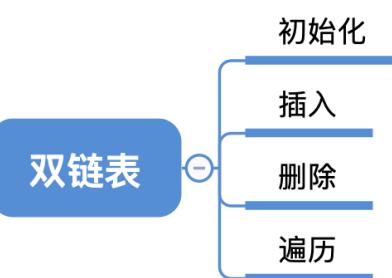
本节内容

双链表

王道考研/CSKAOYAN.COM

1

知识总览



王道考研/CSKAOYAN.COM

2

单链表 V.S. 双链表

The diagram illustrates the differences between singly and doubly linked lists. In a singly linked list, a node (x) has a pointer (p) to the next node in a 'secret known area'. In a doubly linked list, each node (a₁, a₂, a₃, a₄) contains pointers to both its previous ('prior') and next nodes. A person icon says: 'Singly linked list: cannot reverse search, sometimes inconvenient' and 'Doubly linked list: can go forward/backward, lower storage density loss'.

```

Singly Linked List:
    神秘不可知区域..... → x → .....神秘可知区域
    ↑ p

Doubly Linked List:
    NULL ← 头 ← a1 ← a2 ← a3 ← a4 → NULL
    ↑ p
  
```

typedef struct DNode{

ElemType data;	// 数据域
struct DNode *prior,*next;	// 前驱和后继指针

}DNode, *DLinklist;

prior 英 [ˈpreɪə(r)] 美 [ˈpreɪər] adj. (时间、顺序等) 先前的; 优先的
n. 男修道院院长; 托钵会会长; (非正式) 犯罪前科; 先验; n. (Prior) (美) 普廖尔 (人名)
[复数 priors]

王道考研/CSKAOYAN.COM

3

双链表的初始化（带头结点）

The diagram shows the initialization of a doubly linked list with a header node. It includes code examples, data structures, and a diagram of the header node.

```

// 初始化双链表
bool InitDLinkList(DLinklist &L){
    L = (DNode *) malloc(sizeof(DNode)); // 分配一个头结点
    if (L==NULL)
        return false;
    L->prior = NULL; // 头结点的 prior 永远指向 NULL
    L->next = NULL; // 头结点之后暂时还没有节点
    return true;
}

void testDLinkList() {
    // 初始化双链表
    DLinklist L;
    InitDLinkList(L);
    // 后续代码。。。
}
  
```

typedef struct DNode{

ElemType data;	// 数据域
struct DNode *prior,*next;	// 前驱和后继指针

}DNode, *DLinklist;

DLinklist ←→ **DNode ***

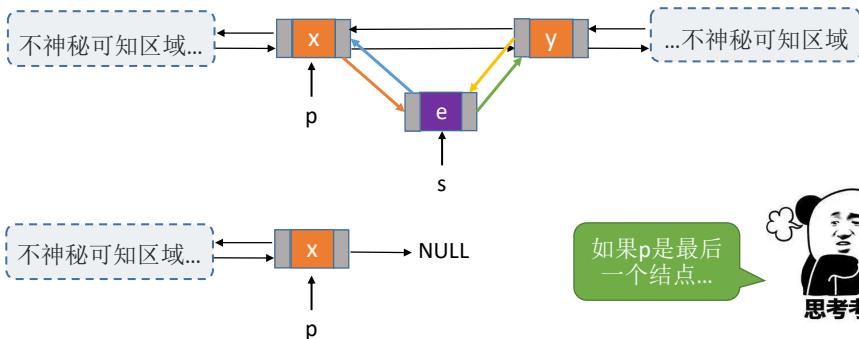
bool Empty(DLinklist L) {
if (L->next == **NULL**)
 return true;
 else
return false;
}

王道考研/CSKAOYAN.COM

4

双链表的插入

```
//在p结点之后插入s结点
bool InsertNextDNode(DNode *p, DNode *s){
    s->next=p->next; //将结点*s插入到结点*p之后
    p->next->prior=s;
    s->prior=p;
    p->next=s;
}
```



王道考研/CSKAOYAN.COM

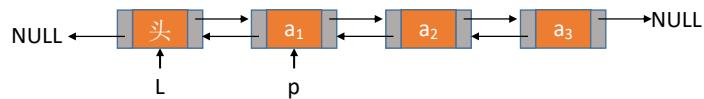
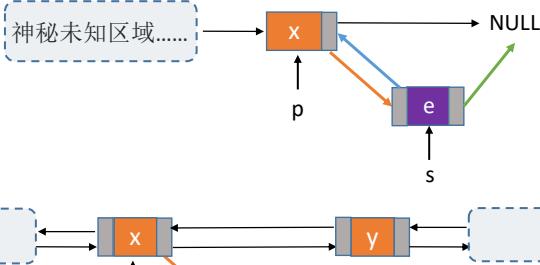
5

双链表的插入

```
//在p结点之后插入s结点
bool InsertNextDNode(DNode *p, DNode *s){
    if (p==NULL || s==NULL) //非法参数
        return false;
    ① s->next=p->next;
    ② if (p->next != NULL)
        p->next->prior=s;
    ③ s->prior=p;
    ④ p->next=s;
    return true;
}
```

修改指针时要注意顺序

用后插操作实现结点的插入有什么好处？

按位序插入
前插操作

王道考研/CSKAOYAN.COM

6

双链表的删除

王道考研/CSKAOYAN.COM

```

void DestoryList(DLinklist &L){
    //循环释放各个数据结点
    while (L->next != NULL)
        DeleteNextDNode(L);
    free(L); //释放头结点
    L=NULL; //头指针指向NULL
}

```

销毁表时才能删除头结点

//删除p结点的后继结点
bool DeleteNextDNode(DNode *p){
 if (p==NULL) return false;
 DNode *q = p->next; //找到p的后继结点q
 if (q==NULL) return false; //p没有后继
 p->next=q->next;
 if (q->next!=NULL) q->next->prior=p; //q结点不是最后一个结点
 free(q); //释放结点空间
 return true;
}

7

双链表的遍历

后向遍历

```

while (p!=NULL){
    //对结点p做相应处理, 如打印
    p = p->next;
}

```

前向遍历

```

while (p!=NULL){
    //对结点p做相应处理
    p = p->prior;
}

```

前向遍历(跳过头结点)

```

while (p-> prior != NULL){
    //对结点p做相应处理
    p = p->prior;
}

```

双链表不可随机存取, 按位查找、按值查找操作都只能用遍历的方式实现。时间复杂度 $O(n)$

王道考研/CSKAOYAN.COM

8

