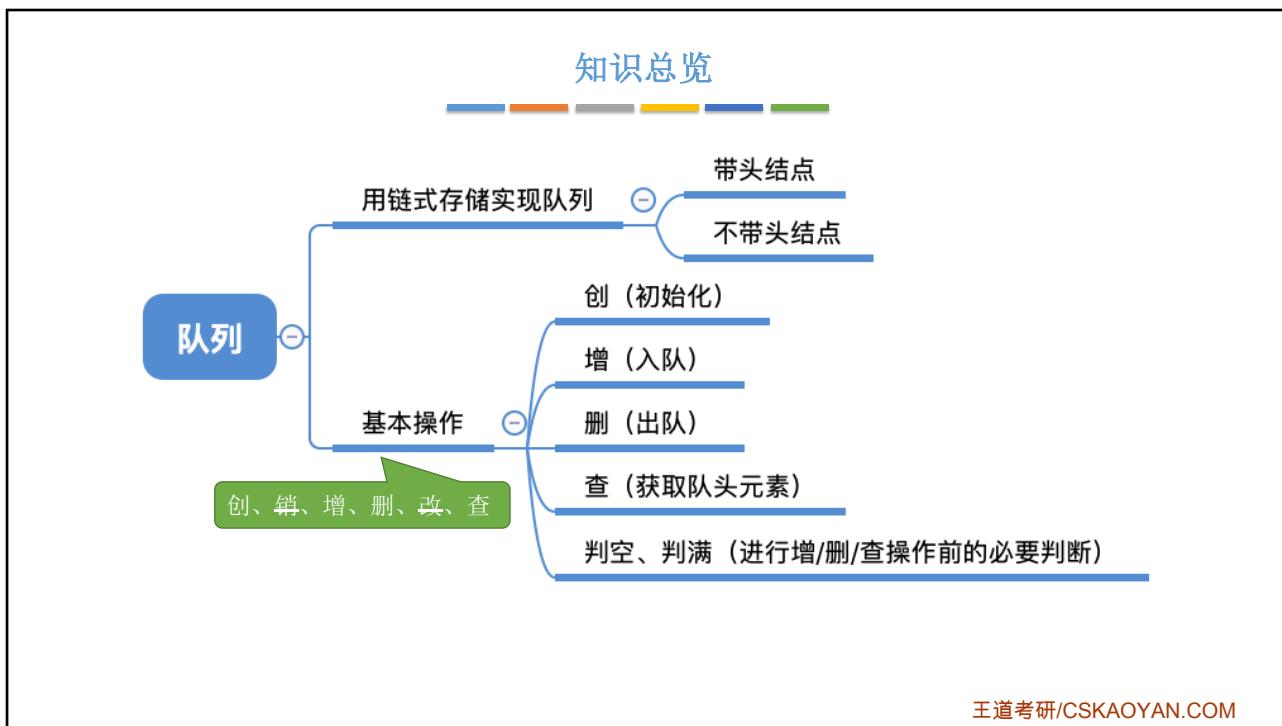


1



2

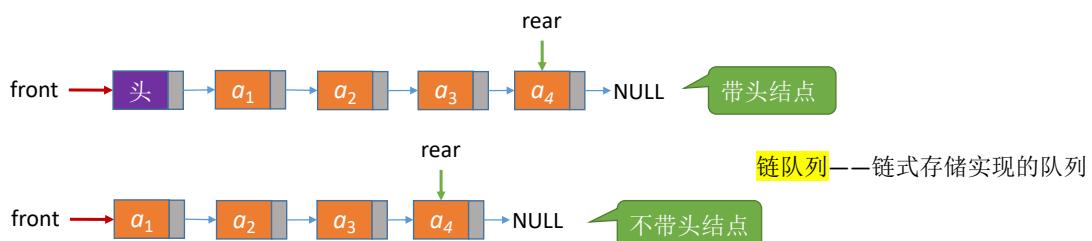
队列的链式实现

```

typedef struct LinkNode{      //链式队列结点
    ELEMTYPE data;
    struct LinkNode *next;
}LinkNode;

typedef struct{           //链式队列
    LinkNode *front,*rear;  //队列的队头和队尾指针
}LinkQueue;

```



王道考研/CSKAOYAN.COM

3

初始化（带头结点）

```

typedef struct LinkNode{      typedef struct{
    ELEMTYPE data;           LinkNode *front,*rear;
    struct LinkNode *next;   }LinkQueue;
}LinkNode;

//初始化队列(带头结点)
void InitQueue(LinkQueue &Q){
    //初始时 front、rear 都指向头结点
    Q.front=Q.rear=(LinkNode*)malloc(sizeof(LinkId));
    Q.front->next=NULL;
}

void testLinkQueue(){
    LinkQueue Q;           //声明一个队列
    InitQueue(Q);          //初始化队列
    //...后续操作...
}

```

带头结点空队列

```

bool IsEmpty(LinkQueue Q){
    if(Q.front==Q.rear)
        return true;
    else
        return false;
}

```

王道考研/CSKAOYAN.COM

4

初始化（不带头结点）

```
//初始化队列(不带头结点)
void InitQueue(LinkQueue &Q){
    //初始时 front、rear 都指向NULL
    Q.front=NULL;
    Q.rear=NULL;
}

//判断队列是否为空 (不带头结点)
bool IsEmpty(LinkQueue Q){
    if(Q.front==NULL)
        return true;
    else
        return false;
}
```

rear → NULL

front → NULL

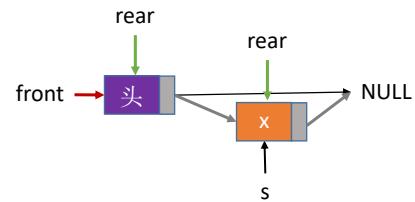
不带头结点的空队列

王道考研/CSKAOYAN.COM

5

入队（带头结点）

```
//新元素入队 (带头结点)
void EnQueue(LinkQueue &Q, ELEM_TYPE x){
    LinkNode *s=(LinkNode *)malloc(sizeof(LinkNode));
    s->data=x;
    s->next=NULL;
    Q.rear->next=s;    //新结点插入到rear之后
    Q.rear=s;          //修改表尾指针
}
```



王道考研/CSKAOYAN.COM

6

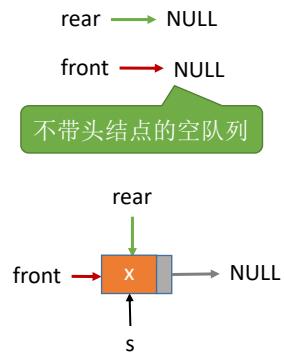
入队 (不带头结点)

```

//新元素入队 (不带头结点)
void EnQueue(LinkQueue &Q, ElemType x){
    LinkNode *s=(LinkNode *)malloc(sizeof(LinkNode));
    s->data=x;
    s->next=NULL;
    if (Q.front == NULL){
        Q.front = s;
        Q.rear=s;
    } else {
        Q.rear->next=s;
        Q.rear=s;
    }
}

```

//在空队列中插入第一个元素
 //修改队头队尾指针
 不带头结点的队列, 第一个元素入队时需要特别处理
 //新结点插入到 rear 结点之后
 //修改 rear 指针



王道考研/CSKAOYAN.COM

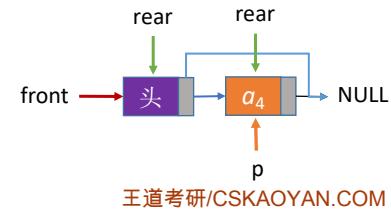
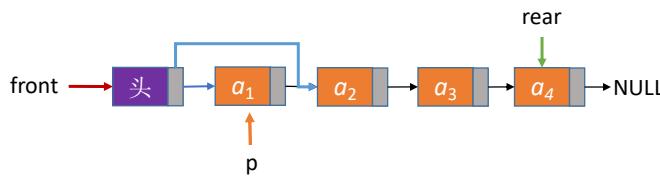
7

出队 (带头结点)

```

//队头元素出队 (不带头结点)
bool DeQueue(LinkQueue &Q, ElemType &x){
    if(Q.front==Q.rear)
        return false;          //空队
    LinkNode *p=Q.front->next;
    x=p->data;                //用变量x返回队头元素
    Q.front->next=p->next;    //修改头结点的 next 指针
    if(Q.rear==p)            //此次是最后一个结点出队
        Q.rear=Q.front;        //修改 rear 指针
    free(p);                  //释放结点空间
    return true;
}

```



王道考研/CSKAOYAN.COM

8

出队 (不带头结点)

```
//队头元素出队 (不带头结点)
bool DeQueue(LinkQueue &Q, ElemType &x){
    if(Q.front==NULL)
        return false; //空队
    LinkNode *p=Q.front; //p指向此次出队的结点
    x=p->data; //用变量x返回队头元素
    Q.front=p->next; //修改 front 指针
    if(Q.rear==p){ //此次是最后一个结点出队
        Q.front = NULL;
        Q.rear = NULL; //front 指向 NULL
        //rear 指向 NULL
    }
    free(p); //释放结点空间
    return true;
}
```

rear → NULL
front → NULL

不带头结点的空队列

王道考研/CSKAOYAN.COM

9

队列满的条件

链式存储——一般不会队满, 除非内存不足

顺序存储——预分配的空间耗尽时队满

王道考研/CSKAOYAN.COM

10

