链式队列

什么是队列?

队列（**Queue**）是一种特殊的线性表，**插入**和**删除**分别在**队头（Front）**和**队尾（Rear）**进行，没有元素的队列称为空队列。

先进先出，插入元素称为**add**，删除元素称为**Pop**。

抽象数据类型ADT

接口

```
package DS;
public interface Queue<T> {
    public abstract boolean isEmpty();
    public abstract boolean add(T x);//入队返回布尔值
    public abstract T peek();//返回队头元素或者null
    public abstract T poll();//出队返回对头元素或者null
}
```

单链表队列

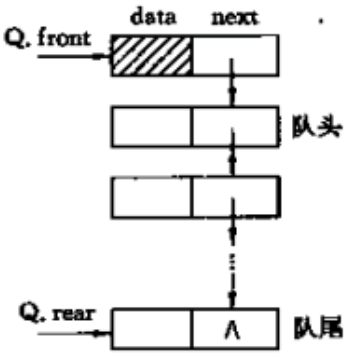

图 3.10 链队列示意图

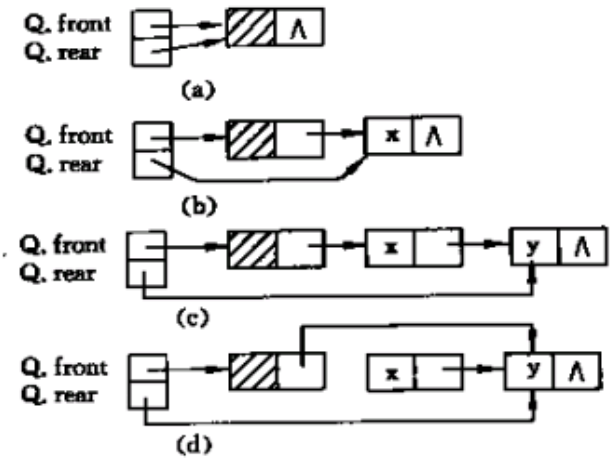

图 3.11 队列运算指针变化状况
(a) 空队列; (b) 元素 x 入队列;
(c) 元素 y 入队列; (d) 元素 x 出队列。

```
package DS;

class LinkedQueue<T> implements Queue<T> {
```

```
    protected Node<T> front,rear;
    public LinkedQueue(){
        this.rear = this.front = null;
    }

    public LinkedQueue(T[] values){
        this();
        for(int i=0;i<values.length;i++){
            this.add(values[i]);
        }
    }

    public boolean isEmpty(){
        return this.front == null&&this.rear == null;
    }

    public boolean add(T x){
        if(x == null){
            return false;
        }
        Node<T> q = new Node<T>(x,null);
        if(this.front == null){
            this.front = this.rear = q;
        }else{
            this.rear.next = q;
        }
        this.rear = q;
        return true;
    }

    public T peek(){
        return this.front.data;
    }

    public T poll(){
        if(this.isEmpty()){
            return null;
        }
        T x = this.front.data;
        this.front = this.front.next;
        if(this.front == null){
            this.rear = null;
            //记住了front和rear是两个指向相同的指针,需要两个都进行修改.
        }
        return (T)x;
    }
}
```

> 这也就相当于有头结点的单链表了，只是限制了操作，缺点是入队效率低，我们还需要额外分配一个rear来记载队尾的位置。

双链表队列

```
package DS;

final class DoubleLinkedQueue<T> implements Queue<T> {
    protected DoubleNode<T> front;
```

```
    public DoubleLinkedQueue() {
        this.front = new DoubleNode<T>();
    }

    public boolean isEmpty(){
        return this.front.next == null && this.front.prev == null;
    }

    public boolean add(T x){
        if(x == null){
            return false;
        }

        if(this.front.prev == null){
            DoubleNode<T> newRear = new DoubleNode<T>
(x,this.front,this.front);
            this.front.next = this.front.prev = newRear;
        }else{
            DoubleNode<T> newRear = new DoubleNode<T>
(x,this.front.prev,this.front);
            this.front.prev.next = newRear;
            this.front.prev = newRear;
        }
        return true;
    }

    public T peek(){
        return this.front.prev.data;
    }

    public T poll(){
        T polledValue = this.front.prev.data;
        this.front.prev.prev.next = this.front;
        this.front.prev = this.front.prev.prev;
        return polledValue;
    }
}

```

> 这里我们就不需要一个额外的指针来记载**rear**的位置了，因为**front**的**prev**就是**rear**。