

2.3 队列及实现

什么是队列

队列(Queue): 具有一定操作约束的线性表

□ 插入和删除操作: 只能在一端插入, 而在另一端删除。

- 数据插入: 入队列 (AddQ)
- 数据删除: 出队列 (DeleteQ)
- 先来先服务
- 先进先出: **FIFO**

队列的抽象数据类型描述

类型名称：队列(Queue)

数据对象集：一个有0个或多个元素的有穷线性表。

操作集：长度为MaxSize的队列 $Q \in \text{Queue}$ ，队列元素 $\text{item} \in \text{ElementType}$

- 1、**Queue CreatQueue(int MaxSize)**：生成长度为MaxSize的空队列；
- 2、**int IsFullQ(Queue Q, int MaxSize)**：判断队列Q是否已满；
- 3、**void AddQ(Queue Q, ElementType item)**：将数据元素item插入队列Q中；
- 4、**int IsEmptyQ(Queue Q)**：判断队列Q是否为空；
- 5、**ElementType DeleteQ(Queue Q)**：将队头数据元素从队列中删除并返回。

队列的顺序存储实现

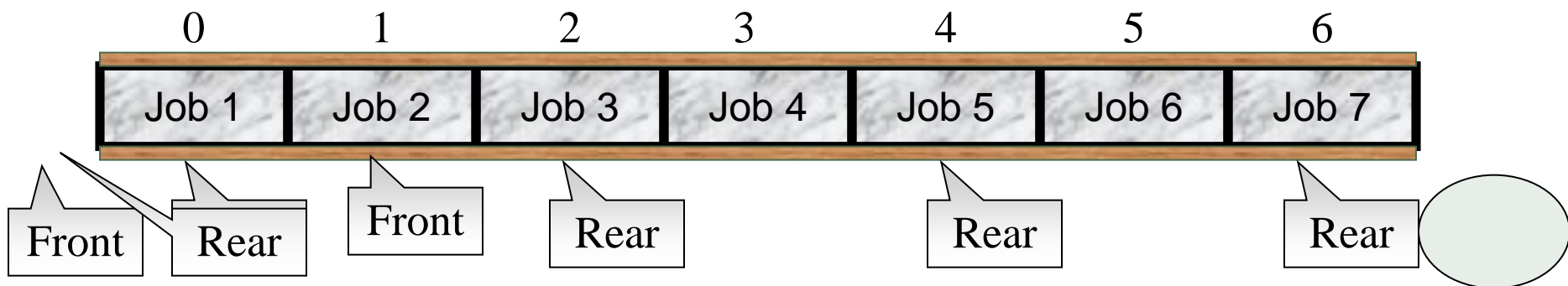
队列的顺序存储结构通常由一个一维数组和一个记录队列头元素位置的变量`front`以及一个记录队列尾元素位置的变量`rear`组成。

```
#define MaxSize <储存数据元素的最大个数>
typedef struct {
    ElementType Data[ MaxSize ];
    int rear;
    int front;
} Queue;
```

队列的顺序存储实现

队列的顺序存储结构通常由一个一维数组和一个记录队列头元素位置的变量`front`以及一个记录队列尾元素位置的变量`rear`组成。

【例】 一个工作队列



AddQ Job 1	AddQ Job 2	AddQ Job 3	DeleteQ Job 1
AddQ Job 4	AddQ Job 5	AddQ Job 6	DeleteQ Job 2
AddQ Job 7	AddQ Job 8		

顺环队列:

AddQ Job 1

AddQ Job 2

AddQ Job 3

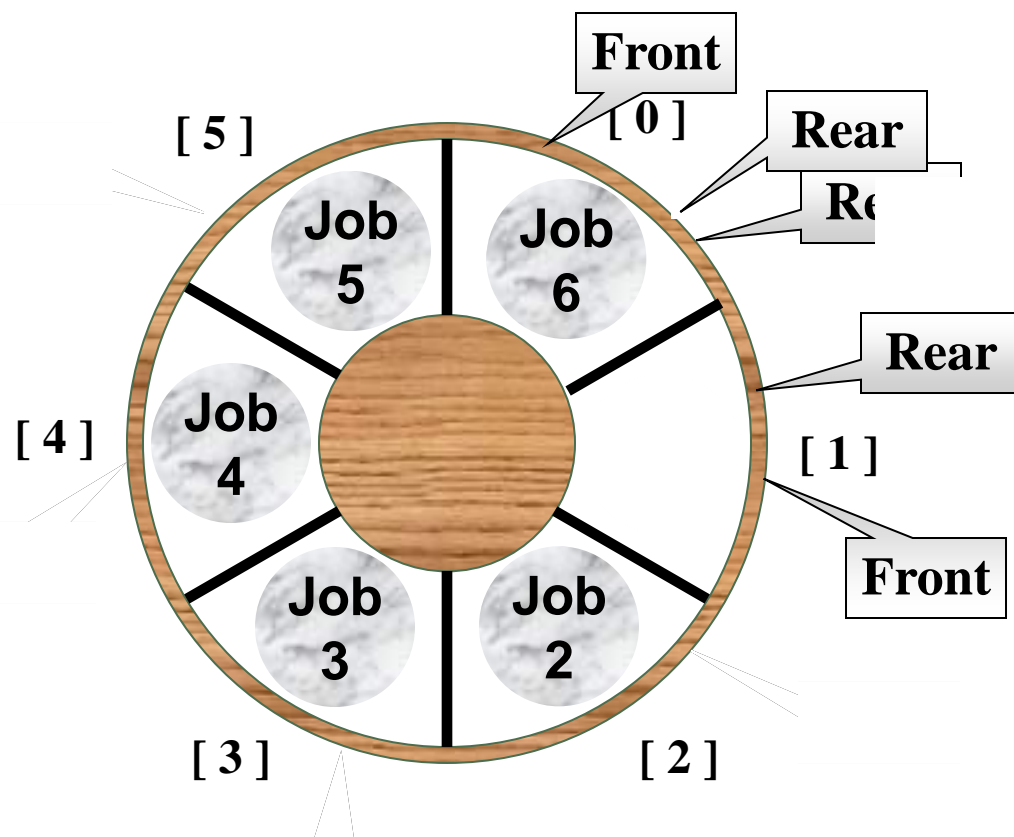
DeleteQ Job 1

AddQ Job 4

AddQ Job 5

AddQ Job 6

AddQ Job 7



思考:

- (1) 这种方案: 堆栈空和满的判别条件是什么?
- (2) 为什么会出现空、满无法区分? 根本原因?

(1)入队列

Front和**rear**
指针的移动
采用“**加1取
余**”法，体
现了顺序存
储的“**循环
使用**”。

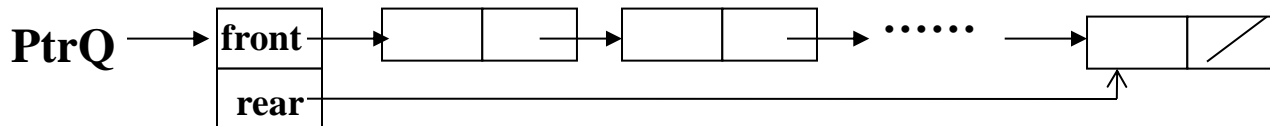
```
void AddQ( Queue *PtrQ, ElementType item)
{
    if ( (PtrQ->rear+1) % MaxSize == PtrQ->front ) {
        printf("队已满");
        return;
    }
    PtrQ->rear = (PtrQ->rear+1)% MaxSize;
    PtrQ->Data[PtrQ->rear] = item;
}
```

```
ElementType DeleteQ ( Queue *PtrQ )
{
    if ( PtrQ->front == PtrQ->rear ) {
        printf("队列空");
        return ERROR;
    } else {
        PtrQ->front = (PtrQ->front+1)% MaxSize;
        return PtrQ->Data[PtrQ->front];
    }
}
```

队列的链式存储实现

队列的链式存储结构也可以用一个单链表实现。插入和删除操作分别在链表的两头进行；**队列指针front和rear应该分别指向链表的哪一头？**

```
typedef struct Node{  
    ElementType Data;  
    struct Node *Next;  
}QNode;  
typedef struct {    /* 链队列结构 */  
    QNode *rear;    /* 指向队尾结点 */  
    QNode *front;   /* 指向队头结点 */  
} LinkQueue;  
LinkQueue *PtrQ ;
```



❖ 不带头结点的链式队列出队操作的一个示例:

```
ElementType DeleteQ ( LinkQueue *PtrQ )
{   Qnode *FrontCell;
    ElementType FrontElem;

    if ( PtrQ->front == NULL) {
        printf("队列空");   return ERROR;
    }
    FrontCell = PtrQ->front;
    if ( PtrQ->front == PtrQ->rear) /* 若队列只有一个元素 */
        PtrQ->front = PtrQ->rear = NULL; /* 删除后队列置为空 */
    else
        PtrQ->front = PtrQ->front->Next;
    FrontElem = FrontCell->Data;
    free( FrontCell );          /* 释放被删除结点空间 */
    return FrontElem;
}
```