



目录

CONTENTS

01

【 顺序队列的存储结构 】

02

【 构造一个空队列 】

03

【 入队列 】

04

【 出队列 】

05

【 求队列的长度 】

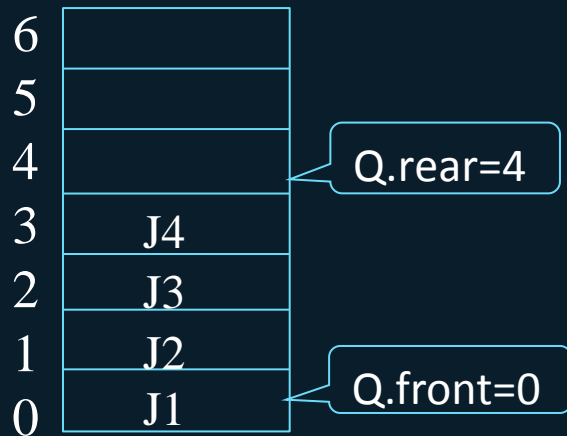
01

顺序队列的存储结构



顺序队列的存储结构

用一组地址连续的存储单元依次存放从队头到队尾的元素，另外还需附设两个指针front 和rear分别指示队列头元素和队列尾元素的位置。



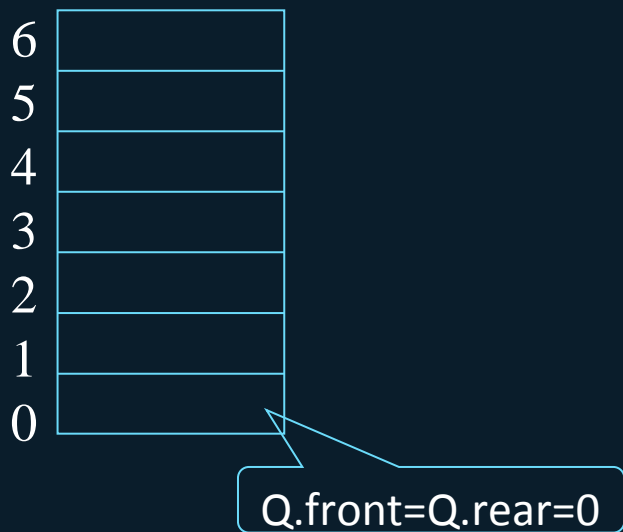
类型定义：

```
typedef struct {  
    QElemType *base; // 初始化的分配空间MAXQSIZE个  
    int front; // 头指针，指向队头元素  
    int rear; // 尾指针，指向队尾元素的下一个位置  
}SqQueue;
```

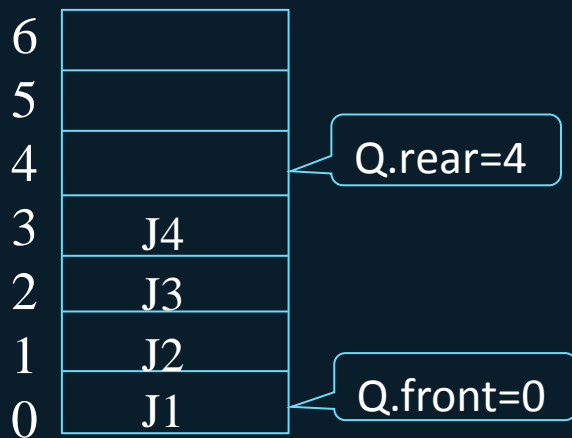
注意: front、rear 不是真正的指针变量，而是整型变量，分别存放队头队尾元素的位置(下标)

约定：在初始化建空队列时，令 $Q.front=Q.rear=0$
当插入新的队尾元素时，尾指针增1
当删除队头元素时，头指针增1

空队列
 $MAXQSIZE=7$

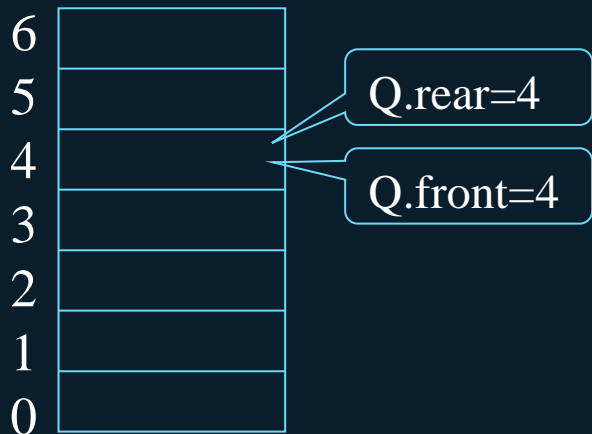


J1 J2 J3 J4相继入队
 $Q.rear=Q.rear+1$

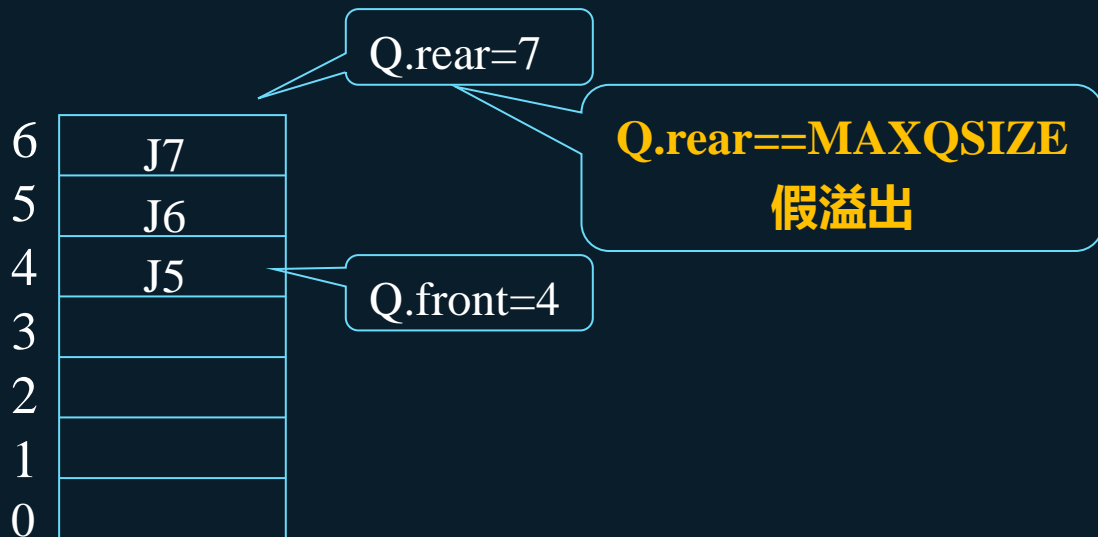


J1 J2 J3 J4 相继出队

$Q.front = Q.front + 1$

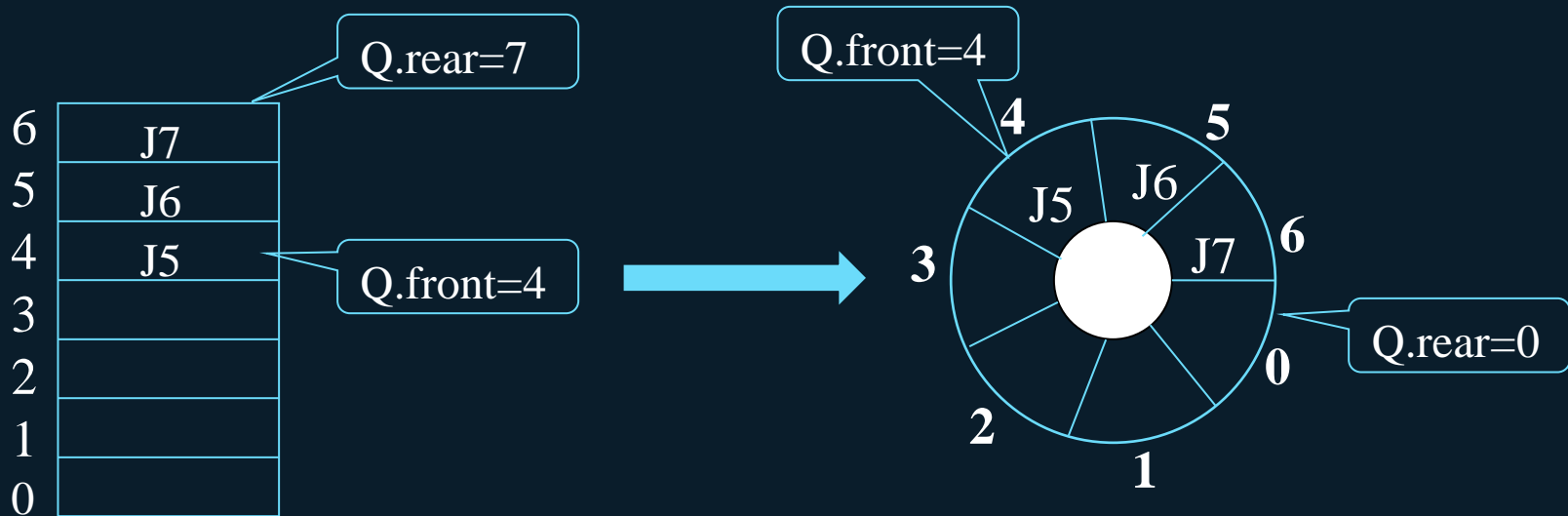


J5 J6 J7 相继入队



注意：在非空顺序队列中，头指针始终指向队头元素，尾指针始终指向队尾元素的下一个位置。

解决方法：把队列设想为一个循环的表，将Q.base[0]接在Q.base[MAXQSIZE-1]之后，即如果Q.rear=MAXQSIZE则令Q.rear=0

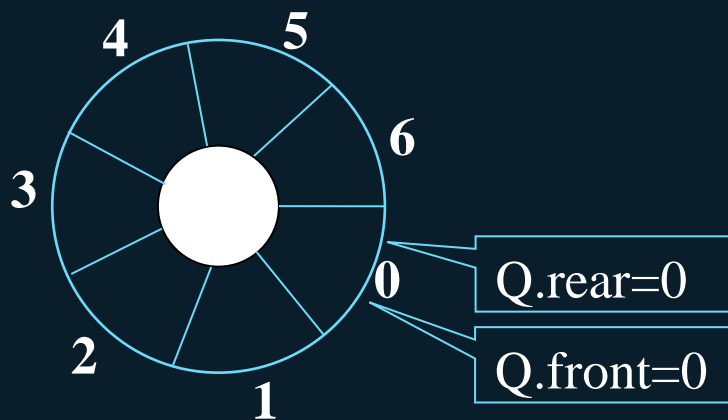


因此，入队： $Q.rear = (Q.rear + 1) \% MAXQSIZE$

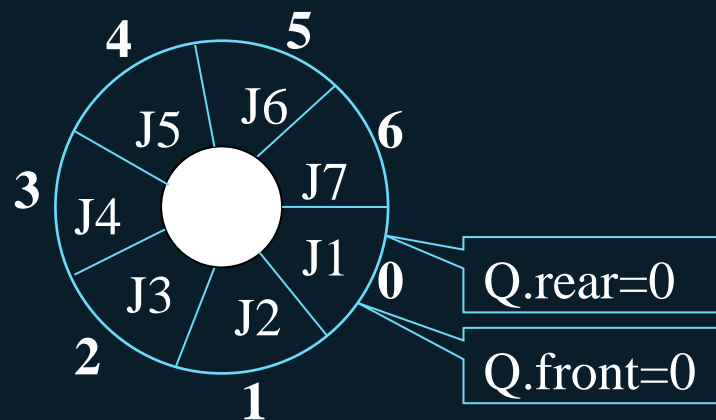
出队： $Q.front = (Q.front + 1) \% MAXQSIZE$

例如：

空队列Q

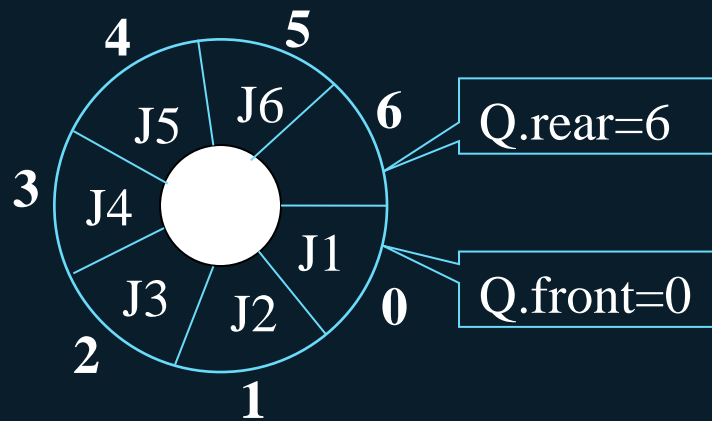


J1 - J7相继入队



如何区分队列的满和空呢？

- 1、设一标志位来区别队列是空还是满
- 2、少用一个元素的空间，**约定**：以队列头指针在队列尾指针的下一个位置 (指环状的下一个位置) 上作为队列呈“满”状态的标志。



J1 ---J6相继入队后，Q.rear=6定为队满

即： $(Q.rear+1)\%7 == Q.front$

因此：队列满的条件是： $(Q.rear+1)\% MAXQSIZE == Q.front$

注意：最大值为MAXQSIZE队列，最多存MAXQSIZE-1个数据元素。



02

构造一个空队列

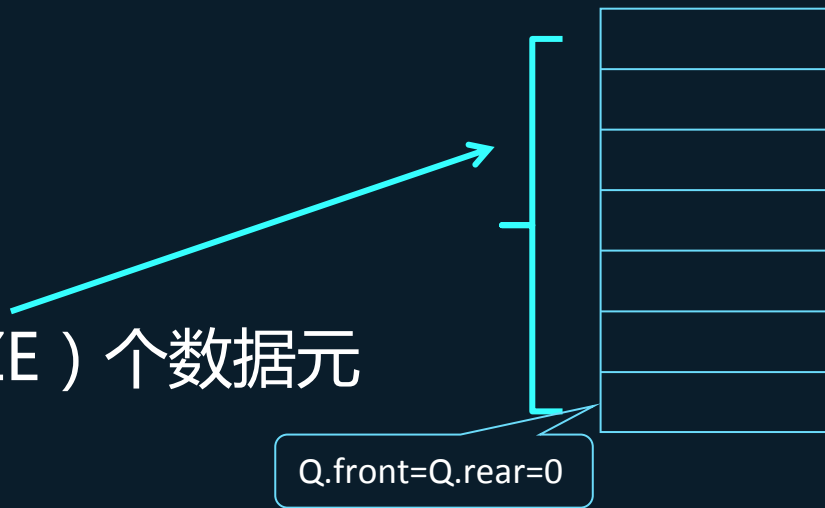
构造一个空队列

1. 问题分析

- 操作名称：InitQueue (&Q)
- 操作结果：构造一个空队列Q

算法思想：

- 1) 申请最大队列长度 (MAXQSIZE) 个数据单元的存储空间
- 2) 队头队尾指针都指向0号存储单元



构造一个空队列

2. 算法描述

//构造一个空队列

Status InitQueue (SqQueue &Q)

{ // 申请最大队列长度的存储空间

Q.base = (QElemType *) **malloc** (MAXQSIZE ***sizeof** (QElemType));

if (!Q.base) **exit** (OVERFLOW); // 存储分配失败

Q.front = Q.rear = 0; //队头队尾指向0号存储单元

return OK;

}

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/595310114142011121>