四川師範大學
SICHUAN NORMAL UNIVERSITY

第3章 栈和队列

DATA STRUCTURE

计算机科学学院 刘 芳

四川師範大學
SICHUAN NORMAL UNIVERSITY

第3章 栈和队列

3.1 栈

3.2 栈的应用举例

3.3 栈与递归

3.4 队列

计算机科学学院　刘 芳

四川師範大學
SICHUAN NORMAL UNIVERSITY

3.4 队列

刘　芳 LiuFang

- 3.4.1 队列的定义
- 3.4.2 队列的链式表示和实现
- 3.4.3 队列的顺序表示和实现

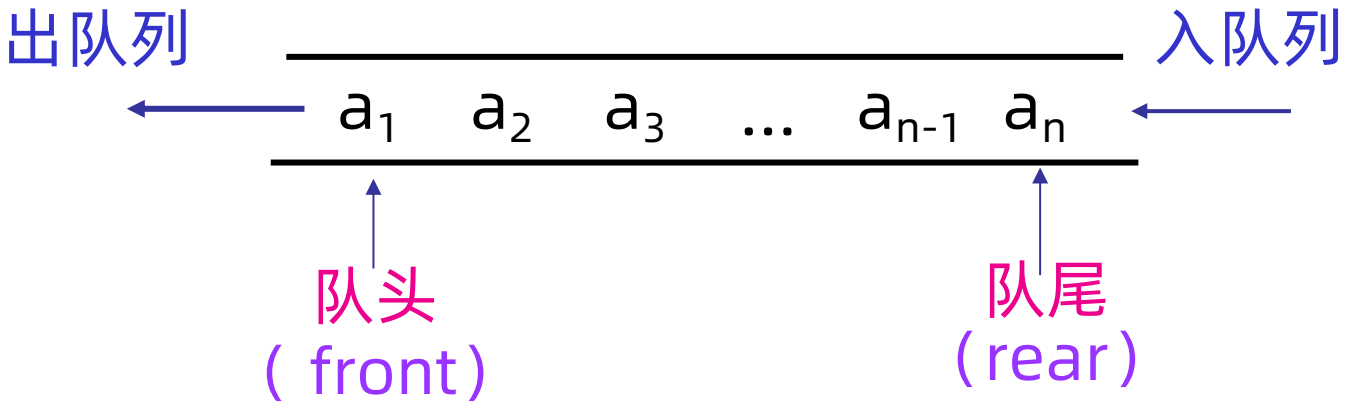
3.4.1 队列的定义

刘　芳 LiuFang

四川師範大學
SICHUAN NORMAL UNIVERSITY

队列的定义

- 队列(Queue)是只允许在表首删除，在表尾插入的线性表。



- 队列的特性：

  先进先出（FIFO，First In First Out)

队列的应用

- 生活中的排队现象
  - 银行业务
  - 汽车加油
  - ……
- 操作系统
  - 作业调度
  - 主机与外设的速度匹配问题等
- 队列的类型
  - FIFO队列
  - 优先级队列
  - 双端队列

队列的抽象数据类型定义

ADT Queue{

数据对象：$D=\{a_i|a_i\in ElemSet;1\le i\le n,n\ge 0;\}$

数据关系：$R=\{<a_i,a_{i+1}>|\ a_i,\ a_{i+1}\in D,i=1,2,...,n-1\}$

基本操作：

InitQueue(&Q)

DestroyQueue(&Q)

QueueEmpty(Q)

QueueLength(Q)

GetHead(Q,&e)

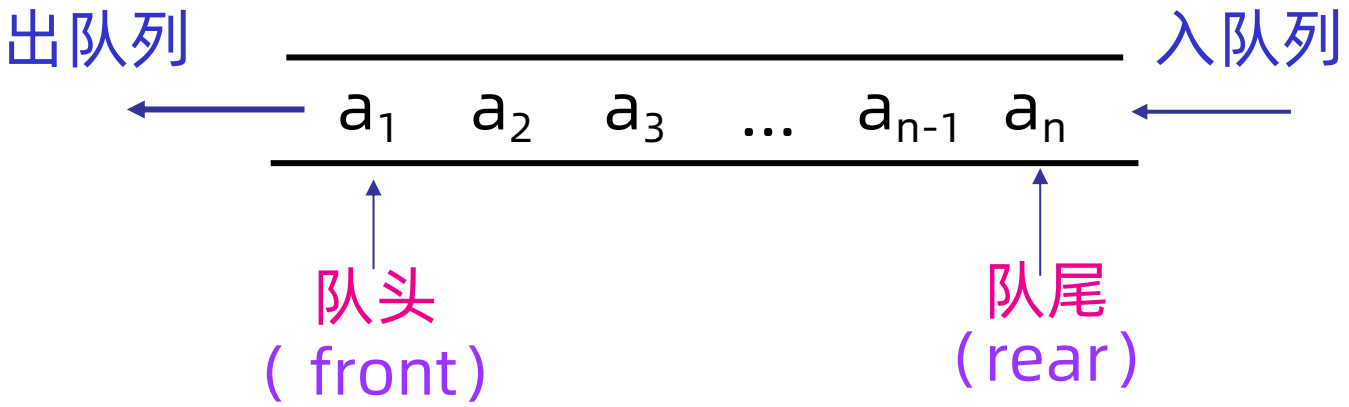
EnQueue(&Q,e)

DeQueue(&Q,&e)

} ADT Queue

本节要点

- 队列是一种特殊的线性表，它只能表尾插入，表首删除。



	线性表	栈	队列
逻辑结构	线性结构	线性结构	线性结构
存储结构	顺序表、链表	顺序栈、链栈	顺序队列、链队列
运算规则	任意位置插入删除	后进先出（LIFO）	先进先出(FIFO)

四川師範大學
SICHUAN NORMAL UNIVERSITY

感谢聆听

业精于勤,荒于嬉;行成于思,毁于随.

四川師範大學
SICHUAN NORMAL UNIVERSITY

第3章 栈和队列

3.1 栈

3.2 栈的应用举例

3.3 栈与函数

3.4 队列

计算机科学学院　刘 芳

四川師範大學
SICHUAN NORMAL UNIVERSITY

3.4 队列

刘　芳 LiuFang

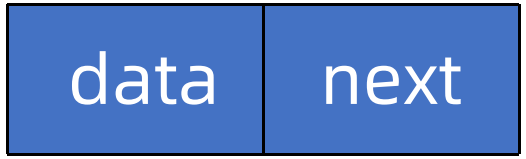
- 3.4.1 队列的定义
- 3.4.2 队列的链式表示和实现
- 3.4.3 队列的顺序表示和实现

3.1.2 队列的链式表示与实现

刘　芳 LiuFang

四川師範大學
SICHUAN NORMAL UNIVERSITY

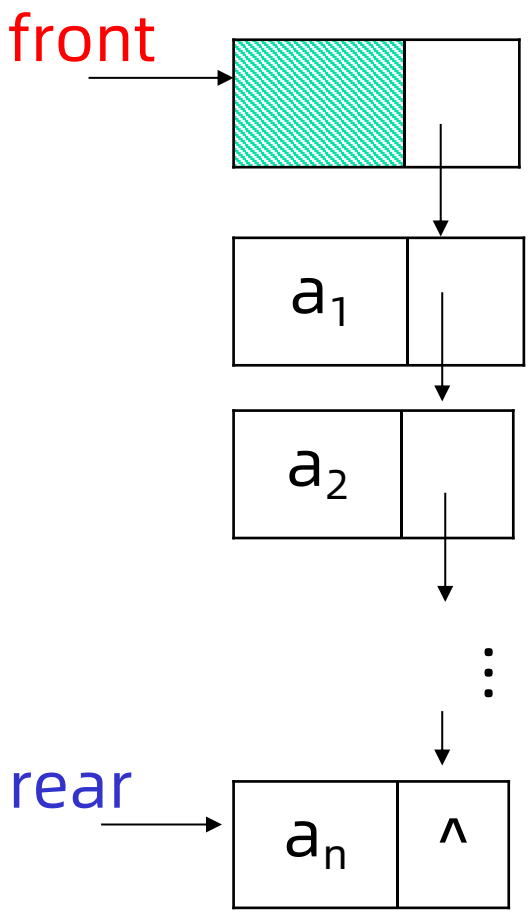
链队列的结构、示意图及C语言定义




```
typedef  struct QNode{
    QElemType data;
    struct QNode *next;
} QNode,* QueuePtr;

typedef struct {
    QueuePtr front;
    QueuePtr rear;
} LinkQueue;

LinkQueue  Q;
```

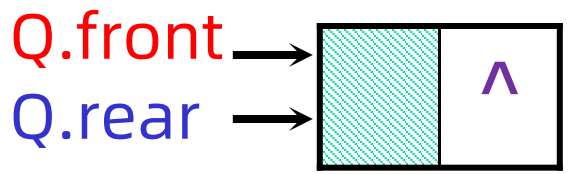


链队列的运算

1. 初始化空队列 InitQueue(&Q)
2. 判断队列空 QueueEmpty(Q)
3. 求队列的长度 QueueLength(Q)
4. 取队首 GetHead(Q,&e)
5. 出队列 DeQueue(&Q,&e)
6. 入队列 EnQueue(&Q,e)
7. 队列的销毁 DestroyQueue(&Q)

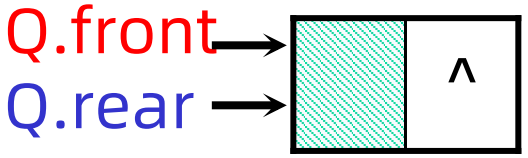
1.初始化空队列

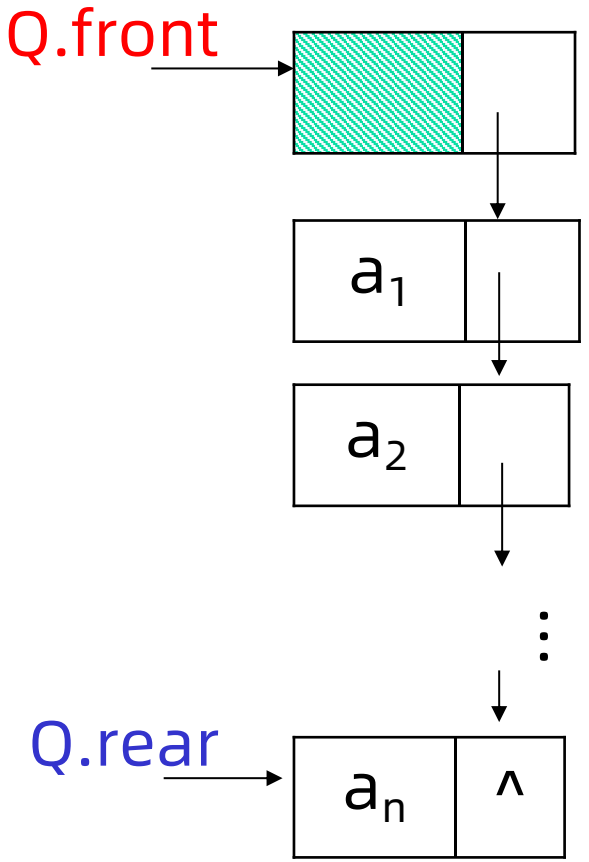
```
Status  InitQueue(LinkQueue &Q)
    Q.front=Q.rear= (QueuePtr)malloc(sizeof(QNode));
    if  (!Q.front) exit OVERFLOW;
    Q.front->next=NULL;
    return OK;
}
```



2. 判断队空

```
Status QueueEmpty( LinkQueue Q){
    return  (Q.front==Q.rear);
}
```



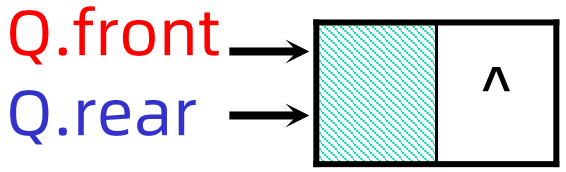


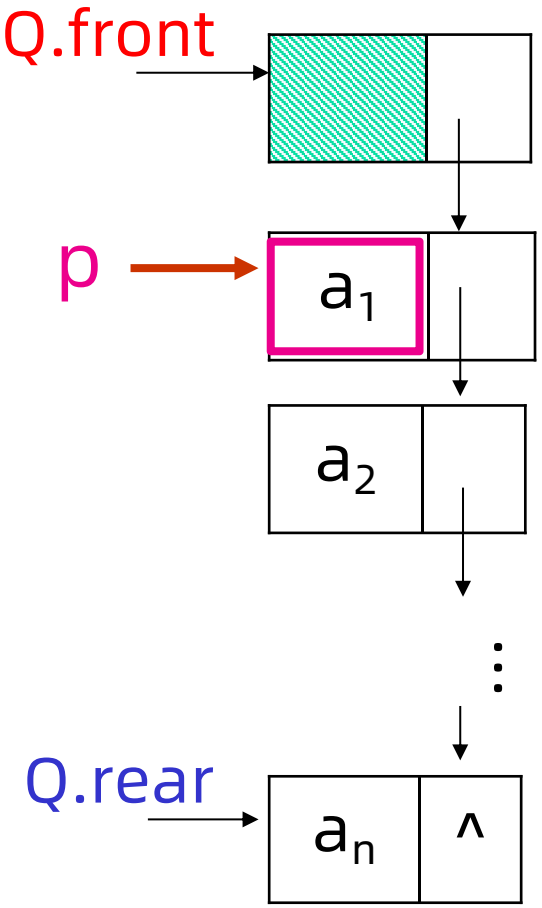
3. 求队列的长度

```
int QueueLength(LinkQueue Q){
   for (i=0,p=  Q.front->next; p; i++,p=p->next）；
   return i;
}
```

4. 取队首

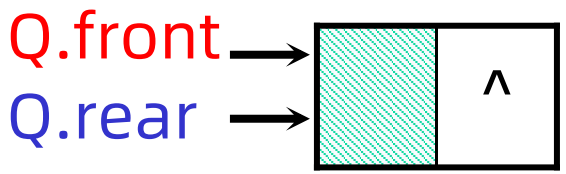
```
Status GetHead(LinkQueue  Q, QElemType &e){
        if (Q.front==Q.rear) return ERROR;
        p=Q.front->next;
        e=p->data;
        return OK;
}
```

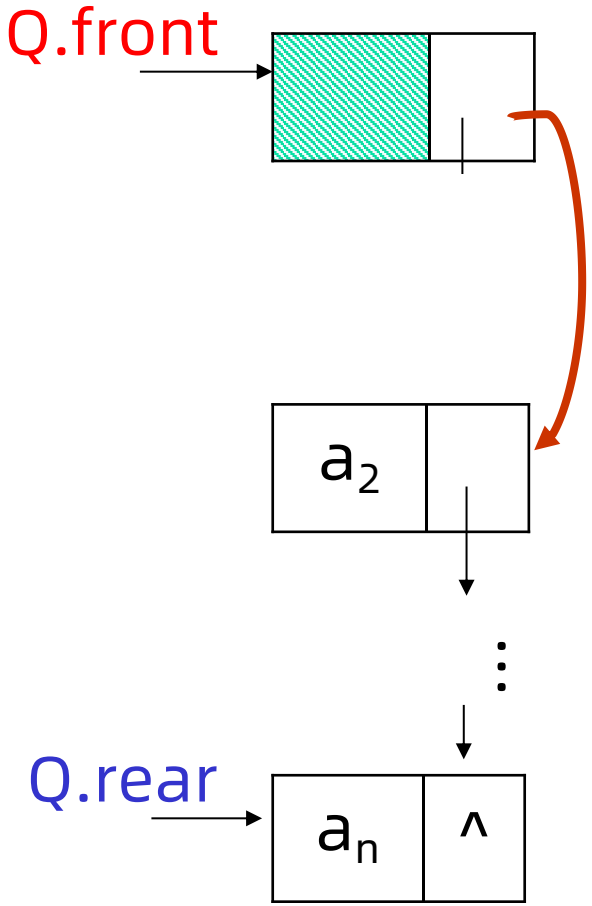




5. 出队列

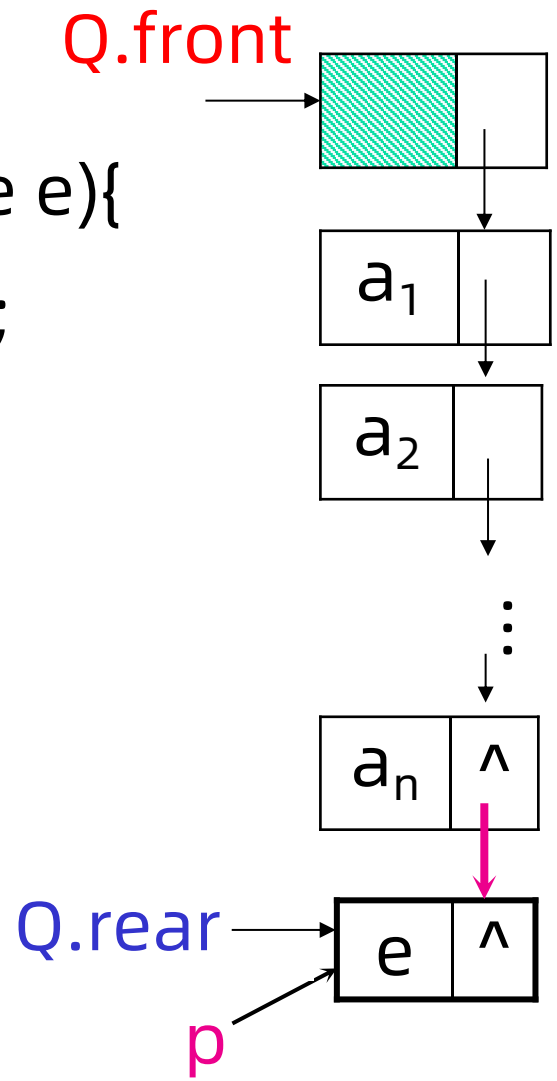
```
Status DeQueue(LinkQueue &Q, QElemType &e){
    if (Q.front==Q.rear) return ERROR;
    p=Q.front->next;
    e=p->data;
    Q.front->next=p->next;
    if (Q.rear==p) Q.rear=Q.front;
    free(p);
    return OK;
}
```





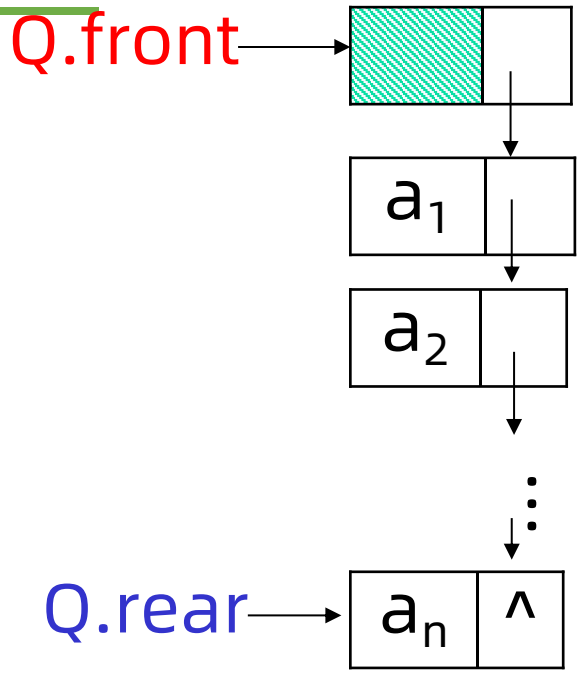
6. 入队列

```
Status EnQueue(LinkQueue &Q, QElemType e){
    p=(QueuePtr)malloc (sizeof(QNode));
    if (!p) exit  (OVERFLOW);
    p->data=e;
    p->next=NULL;
    Q.rear->next=p;
    Q.rear=p;
    return OK;
}
```



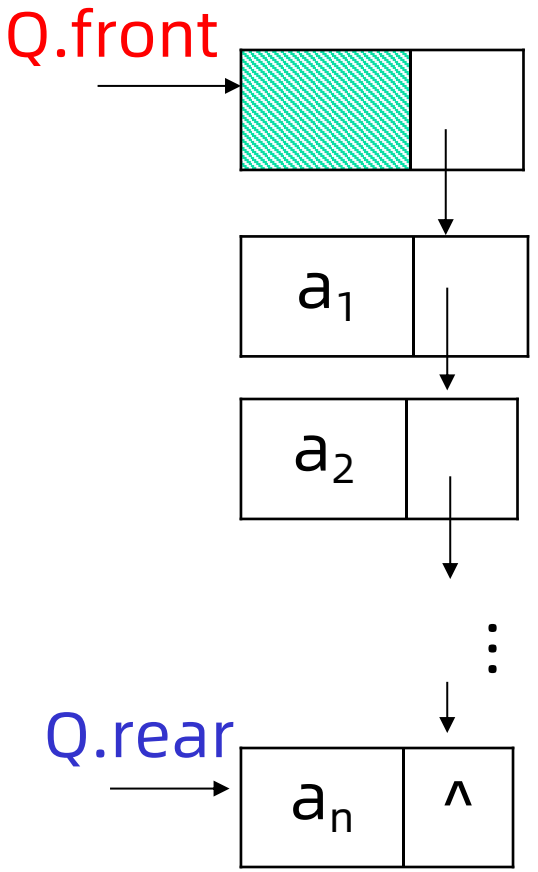
7. 队列的销毁

```
Status DestroyQueue(LinkQueue &Q){
        while (  Q.front ) {
            Q.rear =Q.front->next;
             free(Q.front);
             Q.front= Q.rear;
          }
          return OK;
}
```



本节要点：队列的链式表示与实现

1. 初始化空队列　　InitQueue(&Q)
2. 判断队列空　　QueueEmpty(Q)
3. 求队列的长度　　QueueLength(Q)
4. 取队首　　GetHead(Q,&e)
5. 出队列　　DeQueue(&Q,&e)
6. 入队列　　EnQueue(&Q,e)
7. 队列的销毁　　DestroyQueue(&Q)



四川師範大學
SICHUAN NORMAL UNIVERSITY

感谢聆听

业精于勤,荒于嬉;行成于思,毁于随.

四川師範大學
SICHUAN NORMAL UNIVERSITY

第3章 栈和队列

3.1 栈

3.2 栈的应用举例

3.3 栈与函数

3.4 队列

计算机科学学院　刘 芳

四川師範大學
SICHUAN NORMAL UNIVERSITY

3.4 队列

刘　芳 LiuFang

- 3.4.1 队列的定义
- 3.4.2 队列的链式表示和实现
- 3.4.3 队列的顺序表示和实现

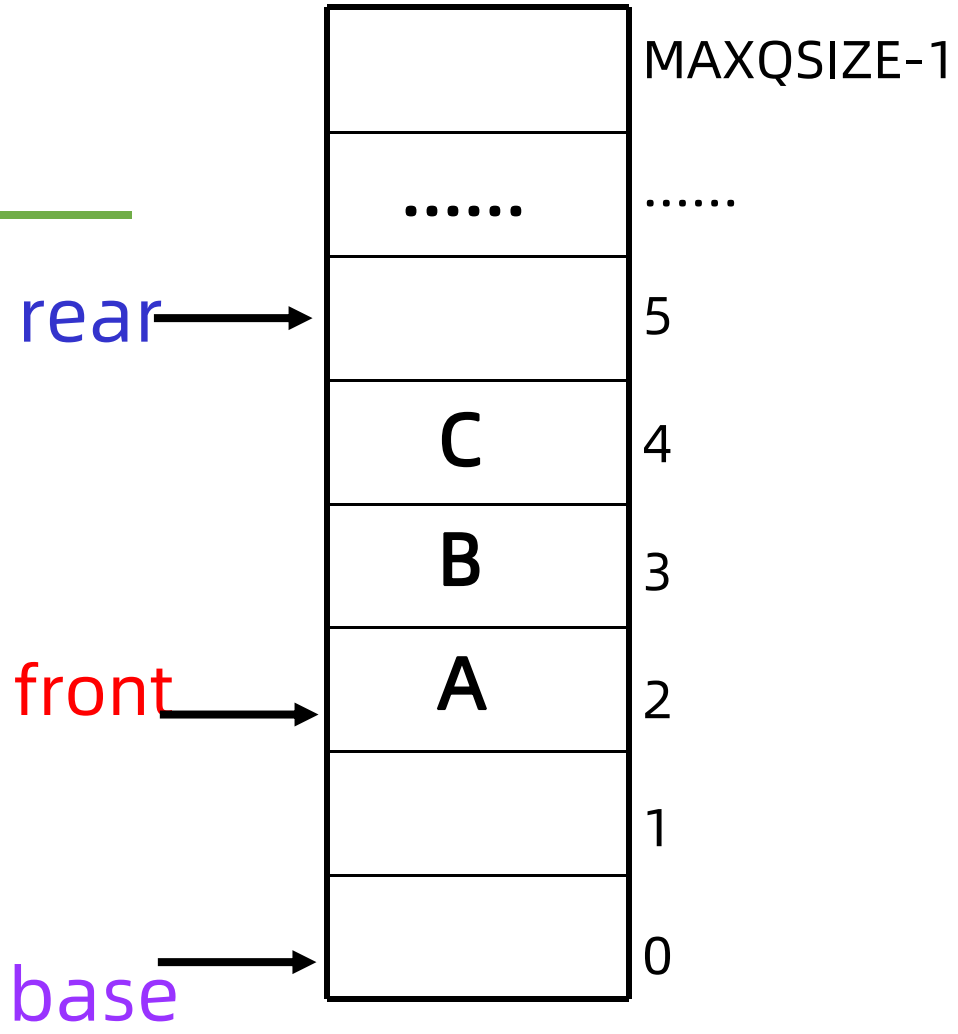
3.4.2 队列的顺序表示与实现

刘　芳 LiuFang

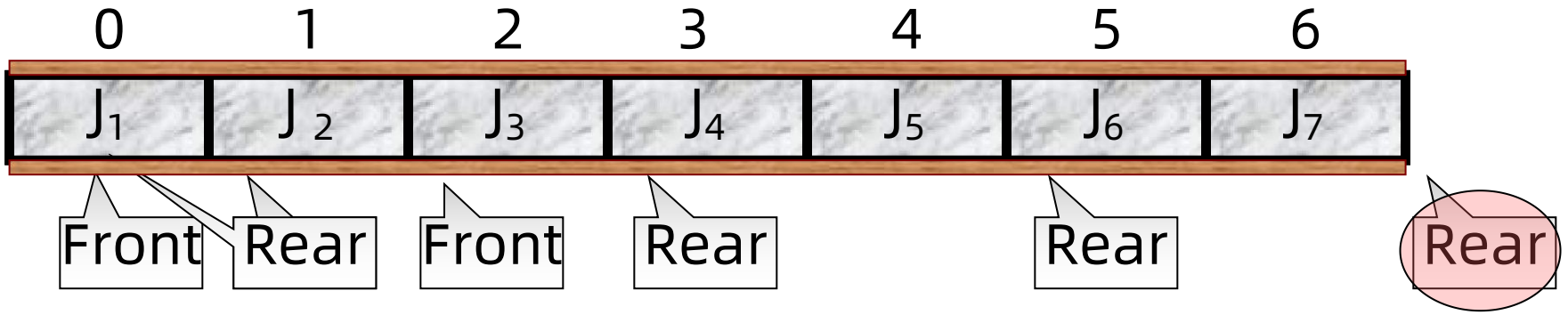
四川師範大學
SICHUAN NORMAL UNIVERSITY

顺序队列的C语言描述

```
#define MAXQSIZE  100
typedef  struct{
            QElemType * base;
            int front;
            int rear;
         }  SqQueue;
SqQueue Q;
```



顺序队列的运算演示

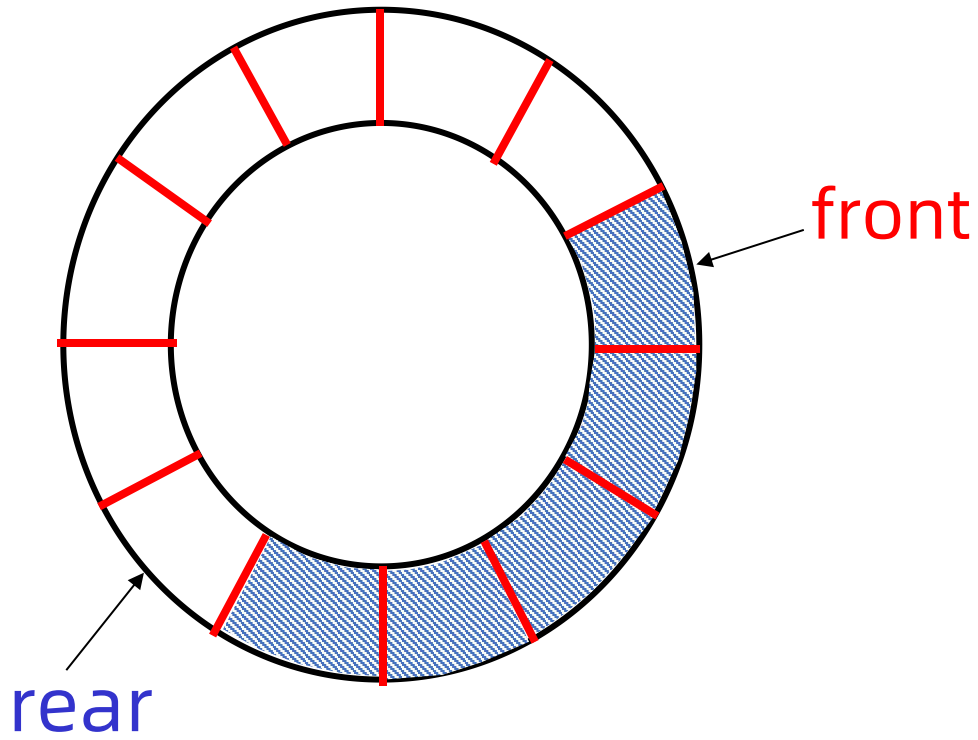


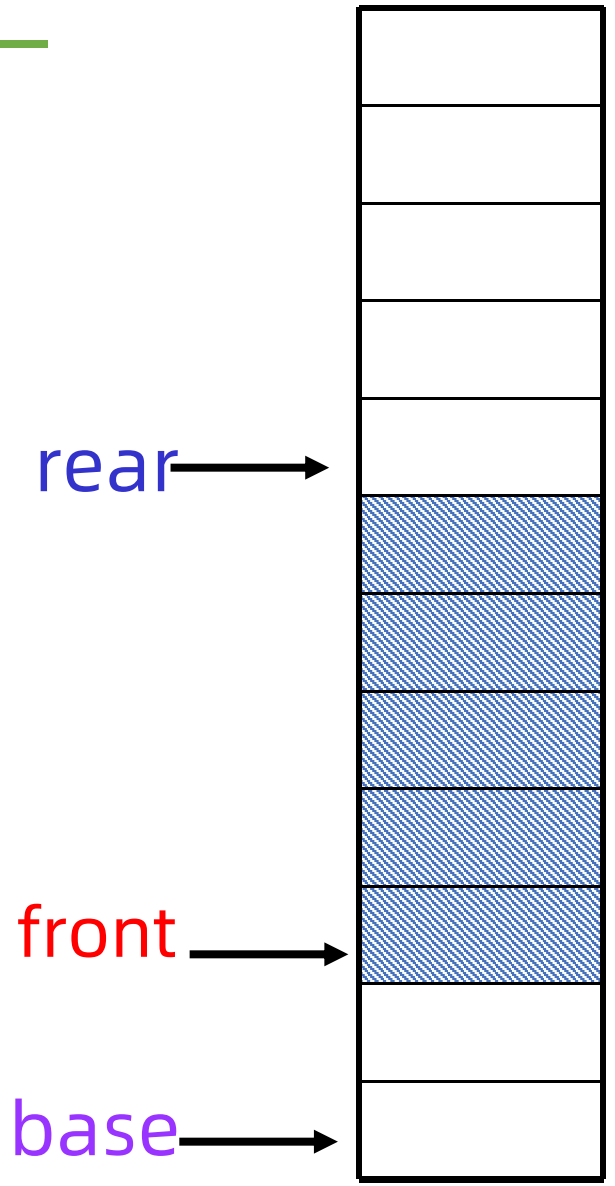
EnQueue J_1	EnQueue J_2	EnQueue J_3	DeQueue
EnQueue J_4	EnQueue J_5	EnQueue J_6	DeQueue
EnQueue J_7	EnQueue J_8		

队列溢出
- 真溢出
- 假溢出

1.修改MAXQSIZE
2.修改出队算法
3.修改入队算法
4.循环队列

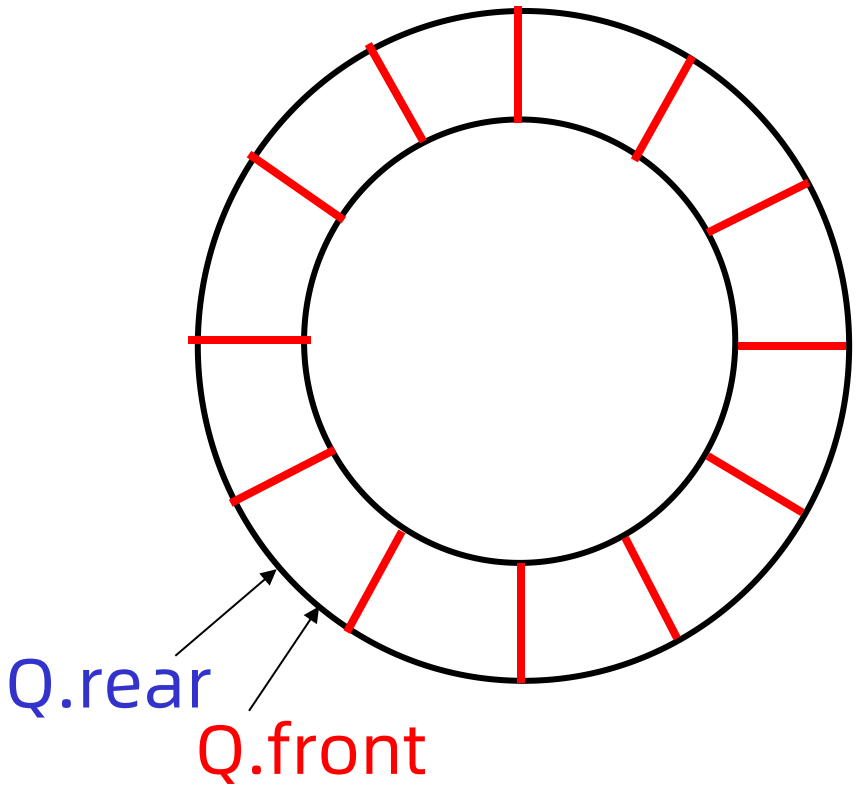
循环队列

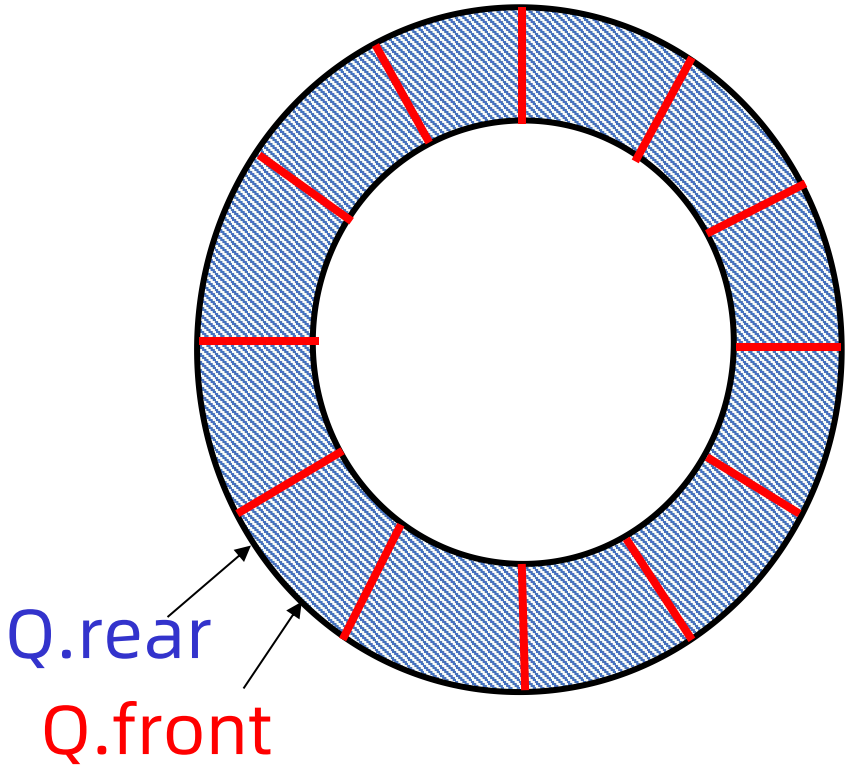




循环队列

二义性！？

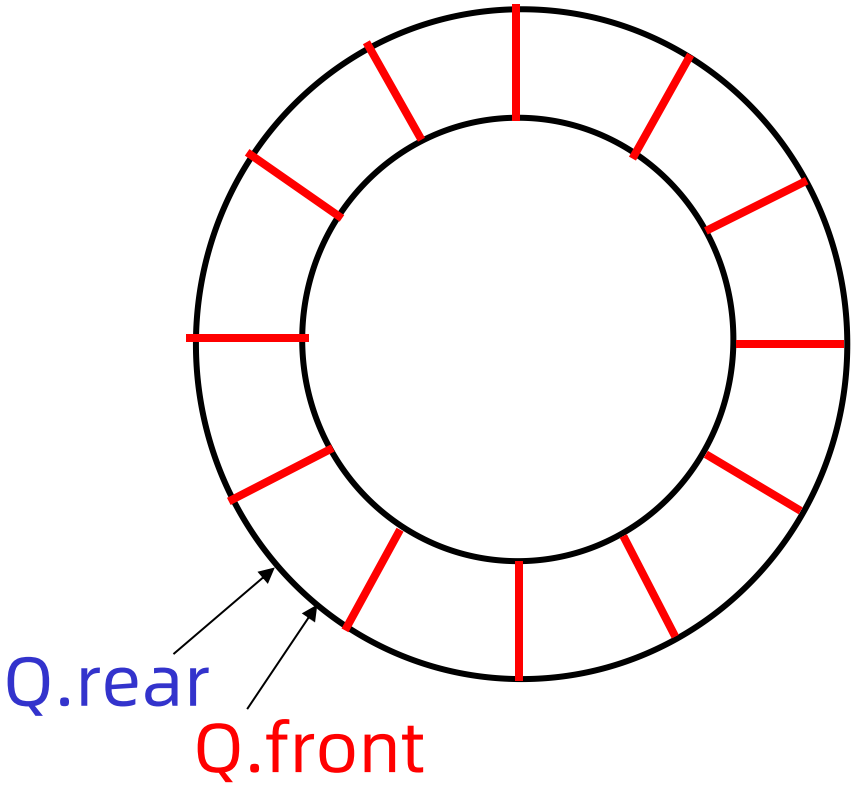


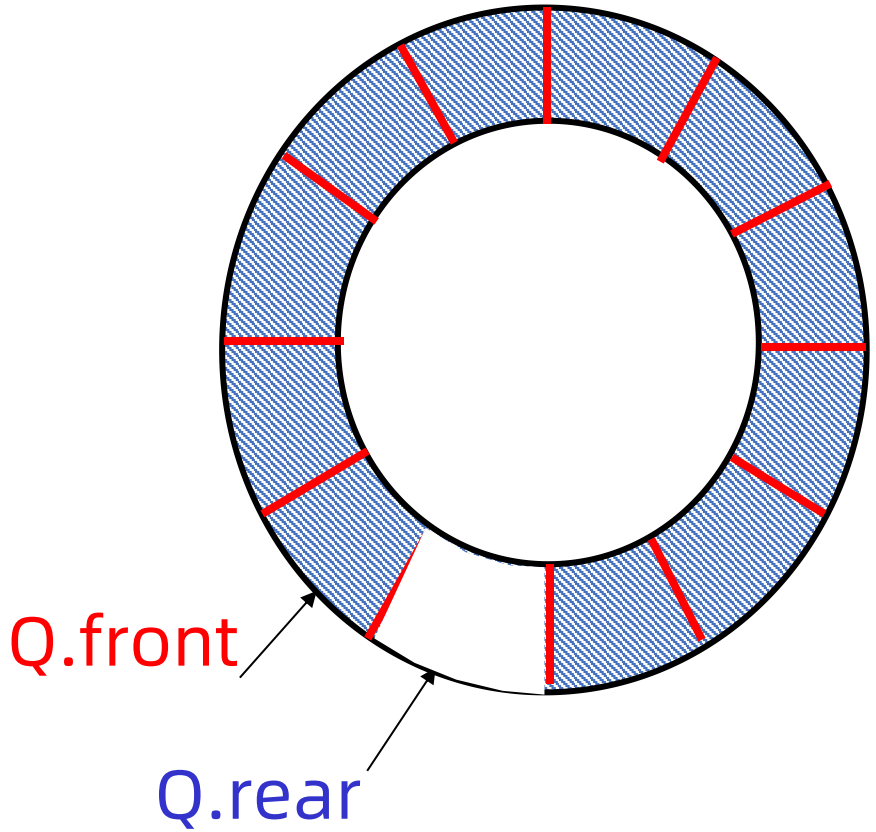


队列空：Q.front == Q.rear

队列满：Q.front == Q.rear

循环队列空、满状态二义性的解决方案　少用一个单元！





队列空：Q.front == Q.rear

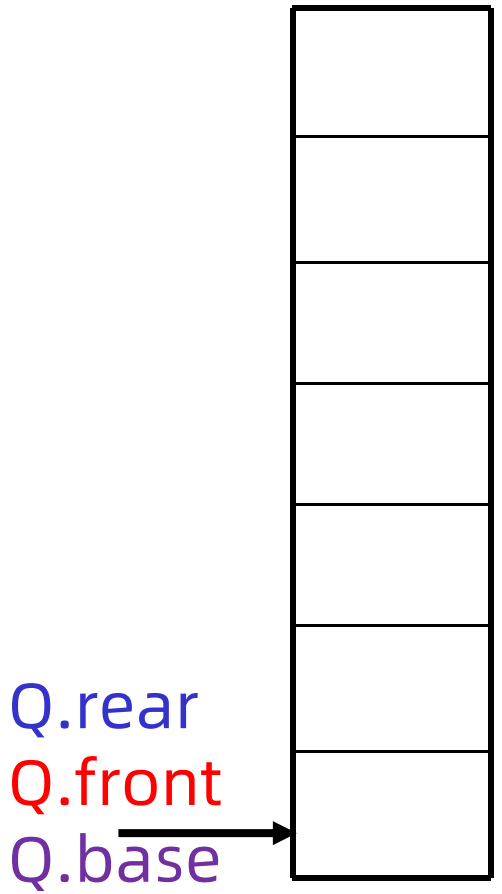
队列满：(Q.rear+1)%MAXQSIZE ==Q.front

循环队列的运算

1. 初始化空队列　　InitQueue(&Q)
2. 判断队列空　　QueueEmpty(Q)
3. 求队列的长度　　QueueLength(Q)
4. 取队首　　GetHead(Q,&e)
5. 出队列　　DeQueue(&Q,&e)
6. 入队列　　EnQueue(&Q,e)
7. 队列的销毁　　DestroyQueue(&Q)

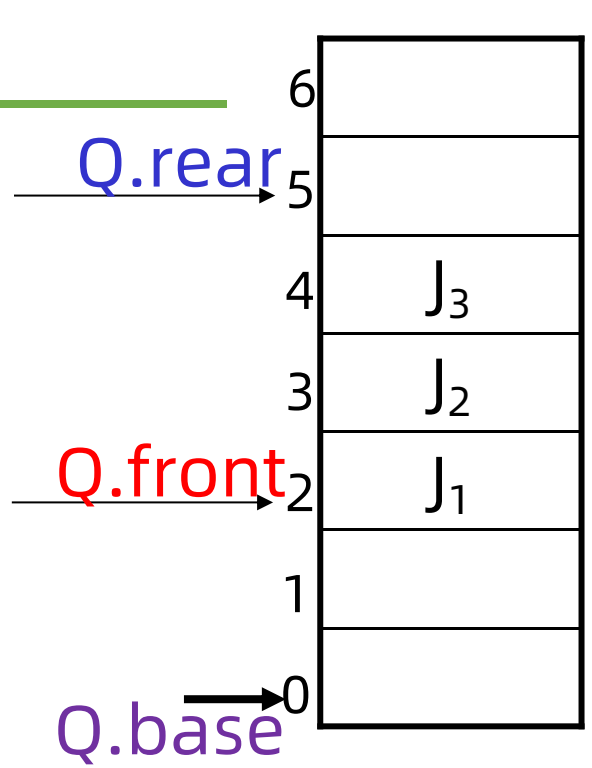
1.初始化空队列

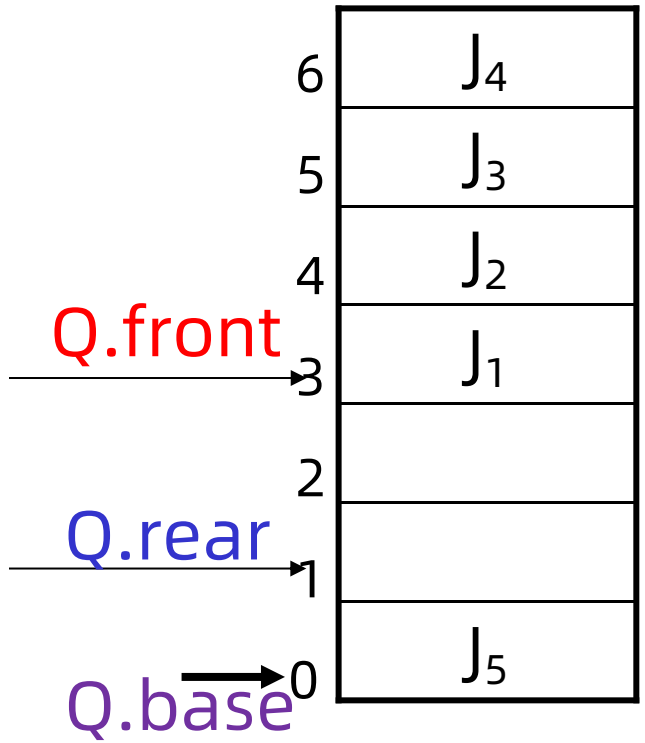
```
Status  InitQueue(SqQueue &Q)
    Q.base=(QElemType*)
     malloc(MAXQSIZE*sizeof(QElemType));
    if (!Q.base) exit (OVERFLOW);
    Q.front=Q.rear=0;
    return OK;
}
```



2. 判断队空

```
Status QueueEmpty( SqQueue Q){
    return  (Q.front==Q.rear);
}
```



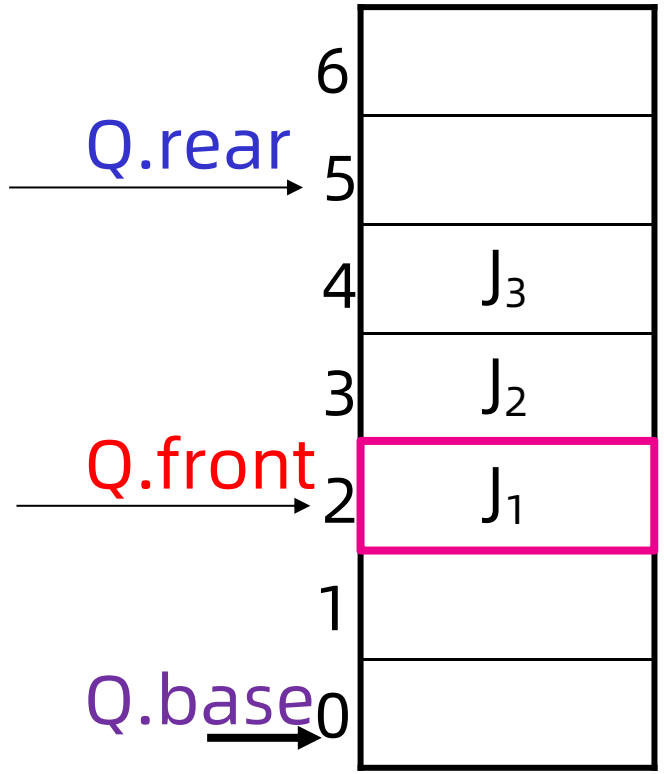


3. 求队列的长度

```
int QueueLength(SqQueue Q){
    return (Q.rear-Q.front+MAXQSIZE) % MAXQSIZE;
}
```

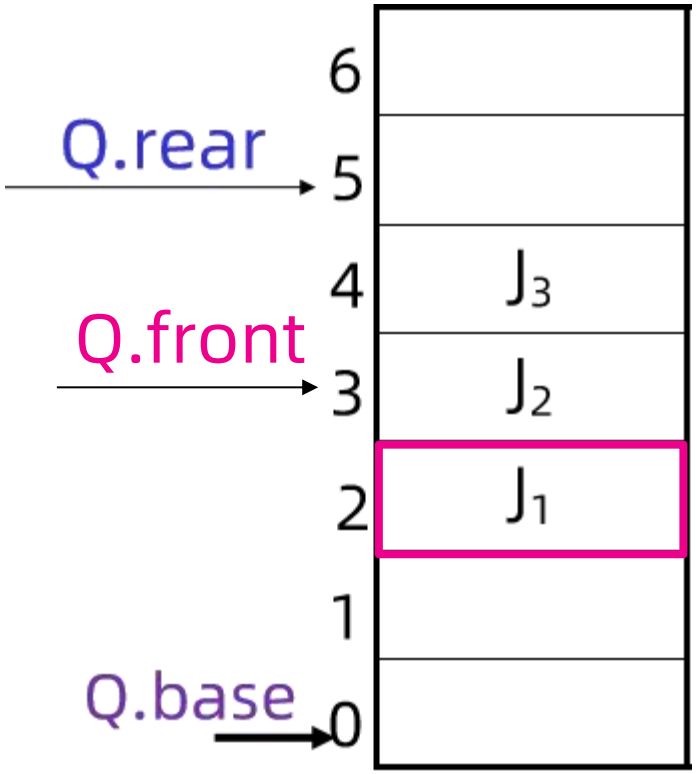
4. 取队首

```
Status GetHead(SqQueue  Q, QElemType &e){
        if (Q.front==Q.rear) return ERROR;
        e=Q.base[Q.front];
        return OK;
}
```



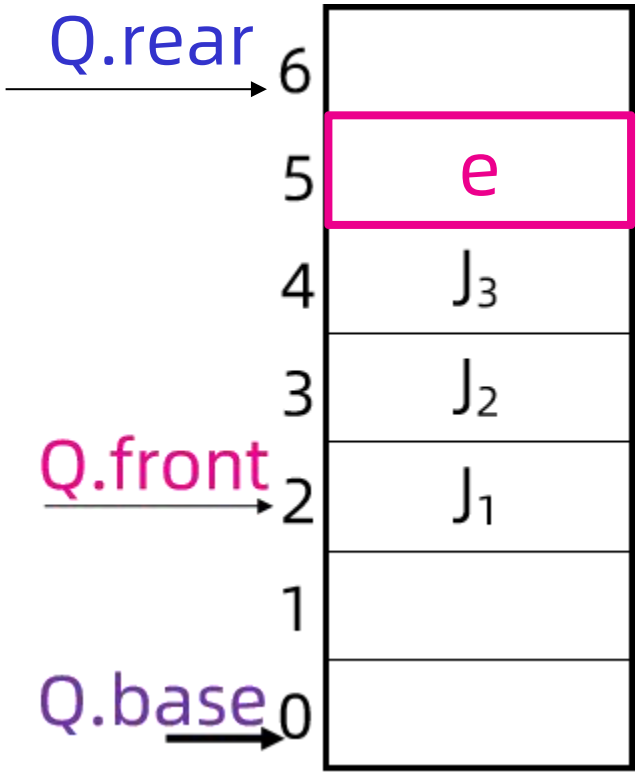
5. 出队列

```
Status DeQueue(SqQueue &Q, QElemType &e){
        if (Q.front==Q.rear) return ERROR;
        e=Q.base[Q.front];
        Q.front=(Q.front+1)%MAXQSIZE;
        return OK;
}
```



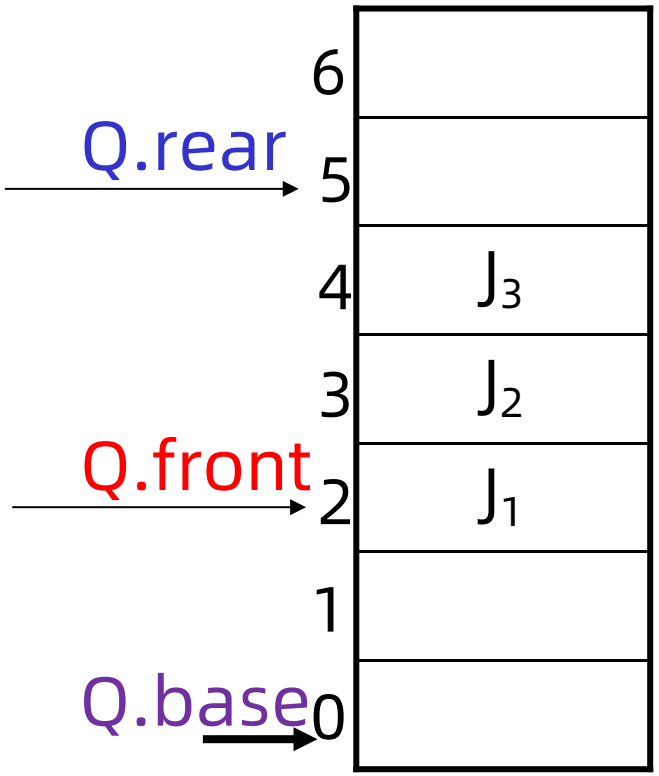
6. 入队列

```
Status EnQueue(SqQueue &Q, QElemType e){
    if ((Q.rear+1)% MAXQSIZE==Q.front)
        return ERROR;
    Q.base[Q.rear]=e;
    Q.rear=(Q.rear+1)%MAXQSIZE;
    return OK;
}
```



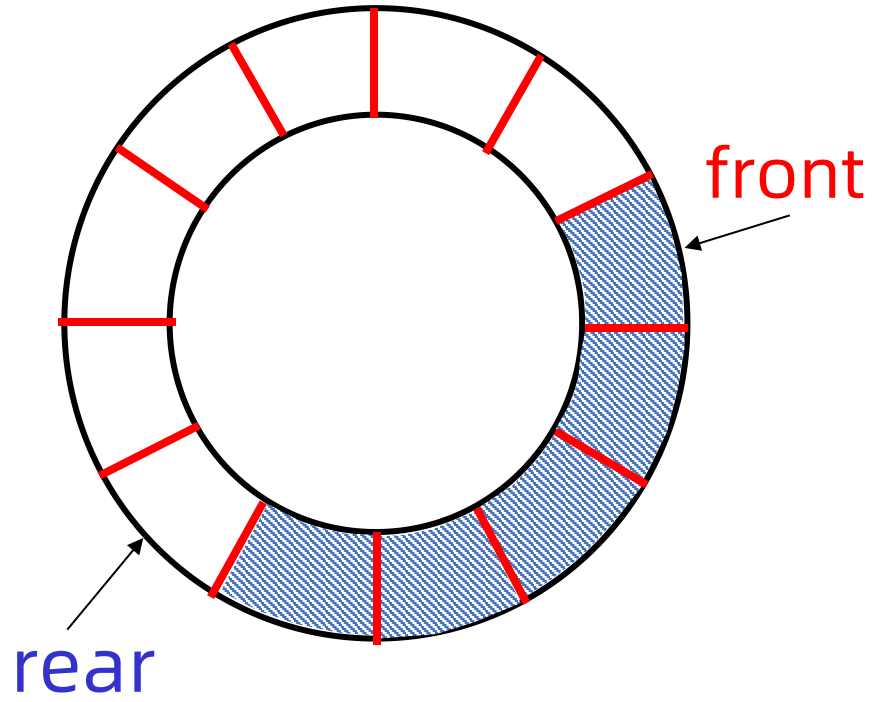
7. 队列的销毁

```
Status DestroyQueue(SqQueue &Q){
        free (Q.base);
        Q.base=NULL;
        Q.front=Q.rear=0;
        return OK;
}
```

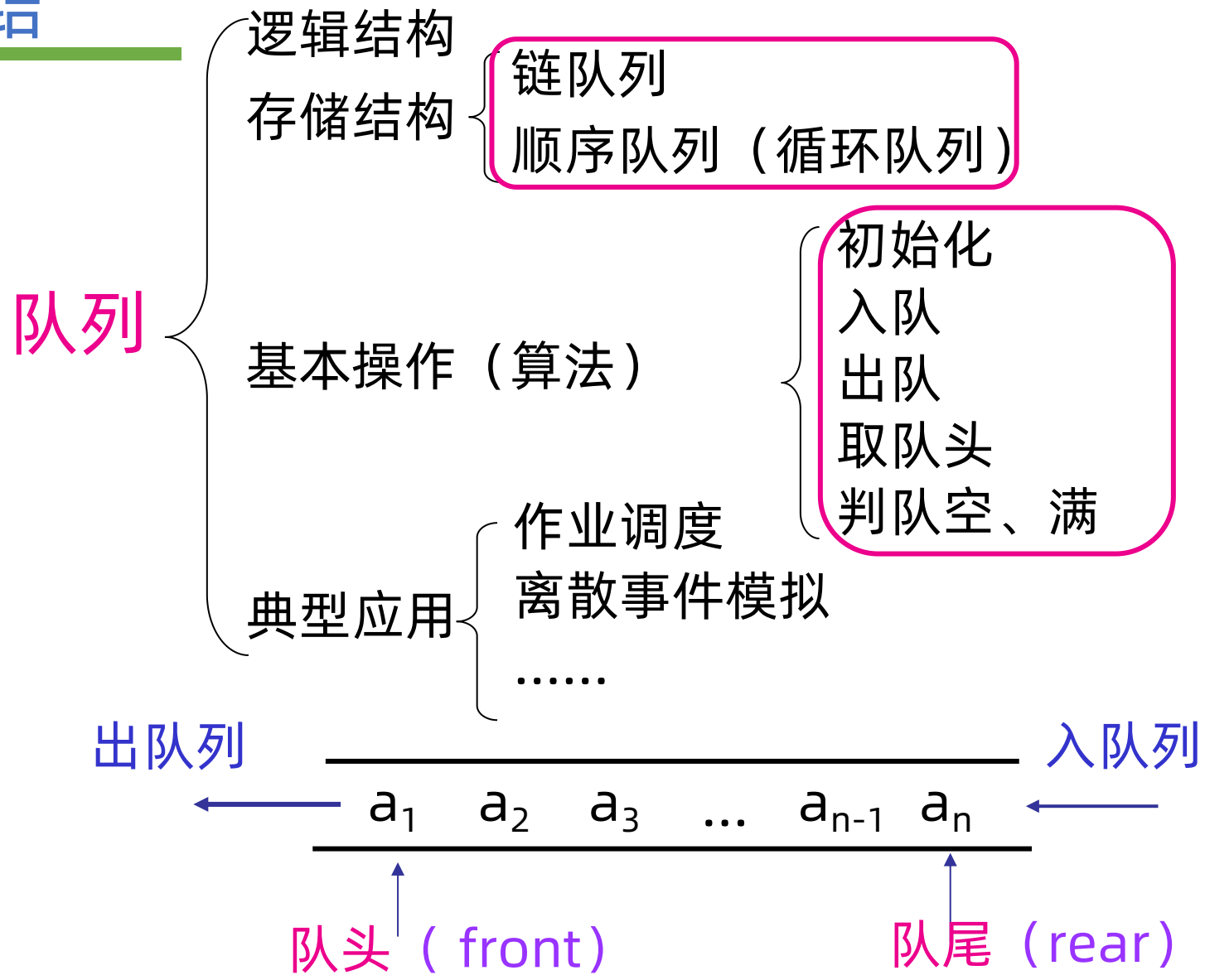


本节要点：队列的顺序表示与实现——循环队列

1. 初始化空队列　　InitQueue(&Q)
2. 判断队列空　　QueueEmpty(Q)
3. 求队列的长度　　QueueLength(Q)
4. 取队首　　GetHead(Q,&e)
5. 出队列　　DeQueue(&Q,&e)
6. 入队列　　EnQueue(&Q,e)
7. 队列的销毁　　DestroyQueue(&Q)



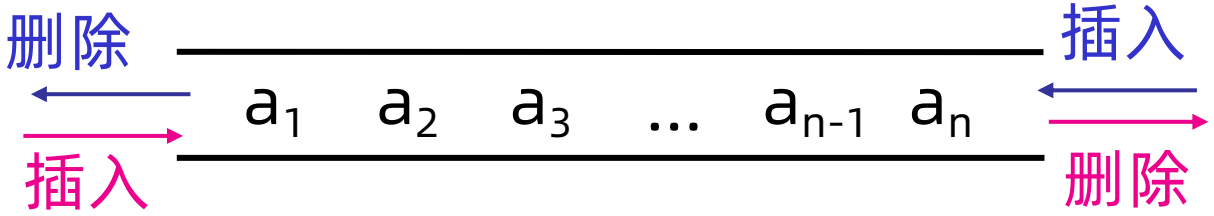
队列的小结


队列
逻辑结构
存储结构
链队列
顺序队列（循环队列）
基本操作（算法）
初始化
入队
出队
取队头
判队空、满
典型应用
作业调度
离散事件模拟
……
出队列
入队列
a_1 a_2 a_3 … a_{n-1} a_n
队头（front）
队尾（rear）

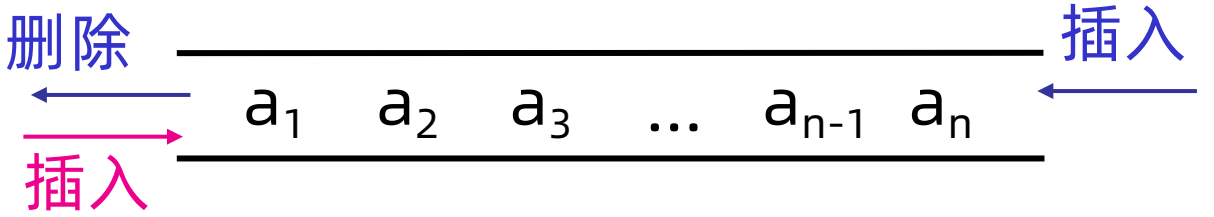
拓展——其他队列

1.优先级队列

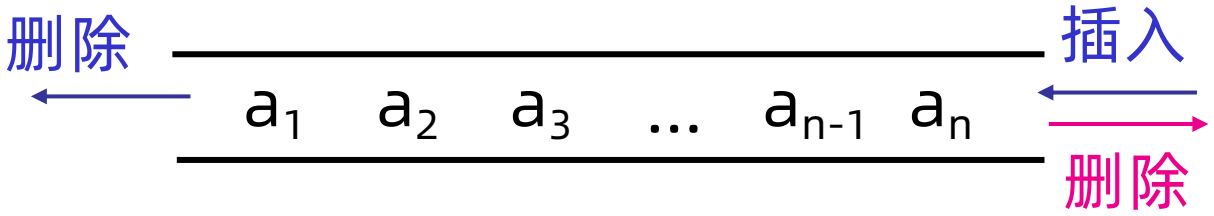
2.双端队列



（1）输出受限的双端队列



（2）输入受限的双端队列



四川師範大學
SICHUAN NORMAL UNIVERSITY

感谢聆听

业精于勤,荒于嬉;行成于思,毁于随.