第三章 栈和队列

3.1 栈

3.2 栈的应用举例

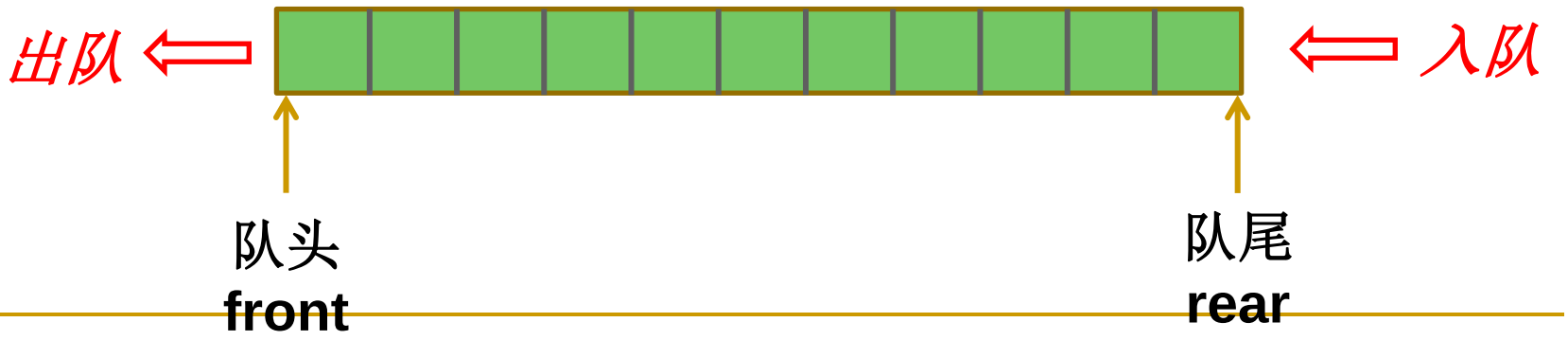
3.3 栈与递归的实现

3.4 队列

3.4 队列

一. 队列的概念

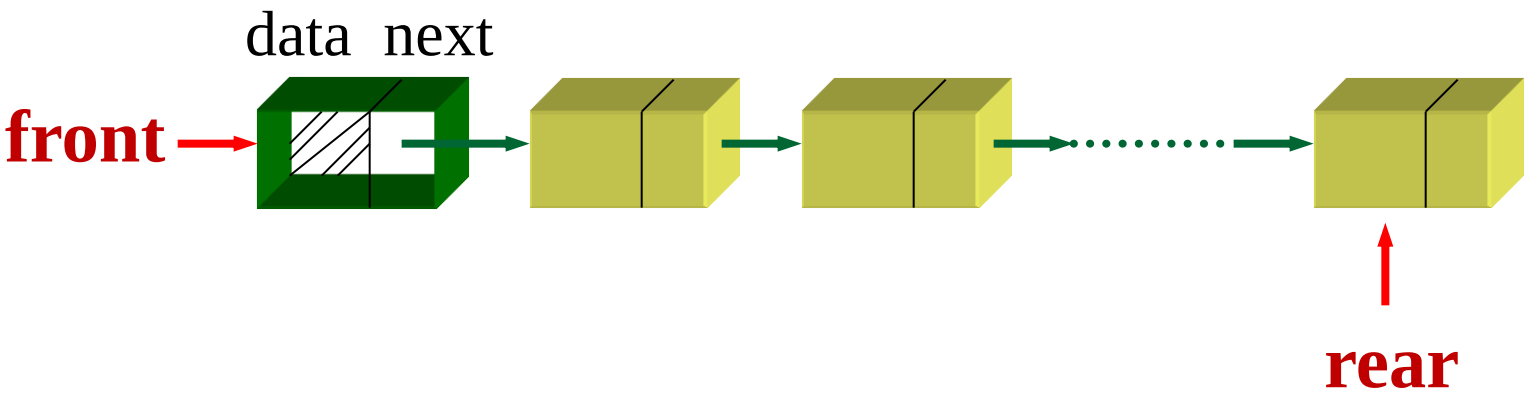
- 队列是只允许在表的一端进行插入，而在另一端删除元素的线性表。
- 在队列中，允许插入的一端叫队尾（rear），允许删除的一端称为队头(front)。
- 特点：先进先出 (FIFO)



3.4 队列

二. 单链队列

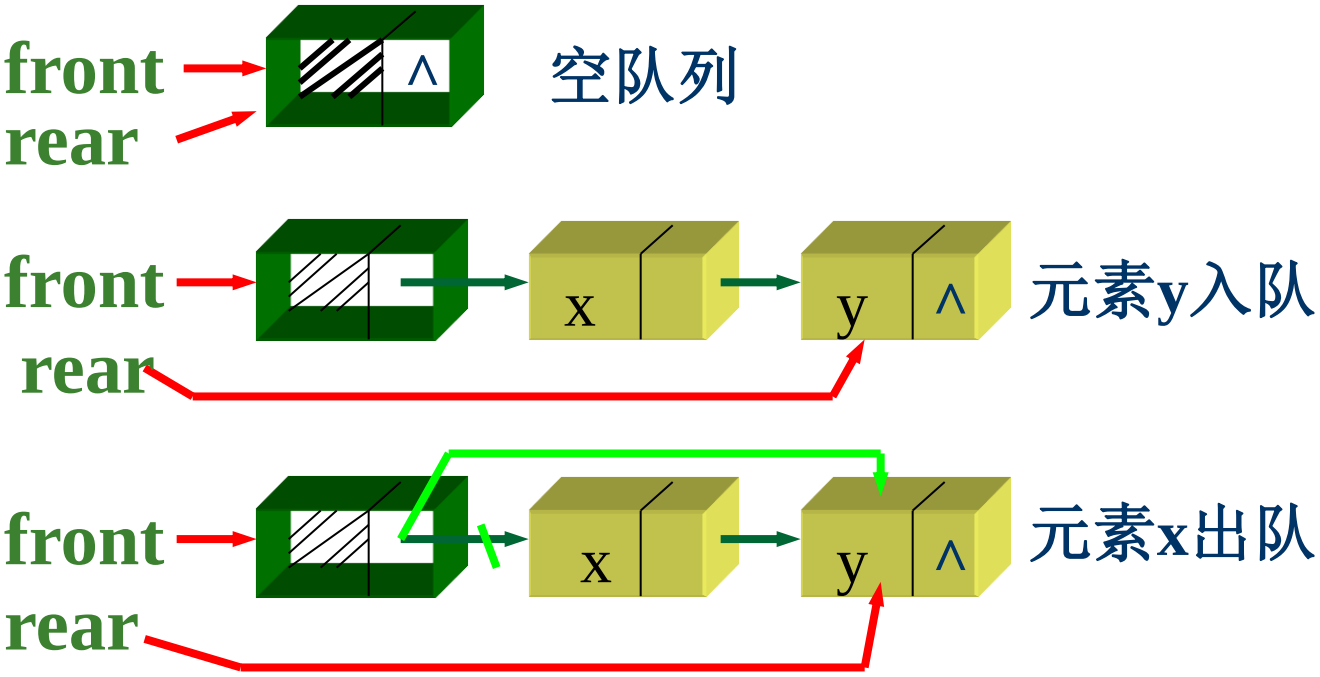
- 链队列采用链表存储表示
- 链队列中，有两个分别指示队头和队尾的指针
- 链式队列在进队时无队满问题，但有队空问题



3.4 队列

二. 单链队列

- 链队列操作实质和链表操作一样，只是多了front和rear的操作



3.4 队列

二. 单链队列

- 单链队列的定义

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

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

3.4 队列

二. 单链队列

- 单链队列的初始化

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

3.4 队列

二. 单链队列

- 单链队列的销毁

```
void DestoryQueue(LinkQueue &Q)
{     while(Q.front)
      {
         Q.rear = Q.front->next;
         free(Q.front);
         Q.front = Q.rear;
      }
}
```

3.4 队列

二. 单链队列

- 单链队列的插入

```
void EnQueue(LinkQueue &Q,  QElemType e)
{     if( !Q.front )     exit(ERROR);
      QNode *p;
      p = (QueuePtr)malloc(sizeof(QNode));
      if( !p )    exit(OVERFLOW);
      p->date=e;
      p->next=NULL;
      Q.rear->next=p;
      Q.rear=p;
}
```

3.4 队列

二. 单链队列

- 单链队列的删除

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

3.4 队列

三. 顺序队列

- 顺序队列是队列的一种实现
- 顺序队列采用一组地址连续的存储单元依次存储从队列头到队列尾的元素
- 顺序队列有两个指针：队头指针front和队尾指针rear在队列中，允许插入的一端叫队尾（rear），允许删除的一端称为队头(front)

3.4 队列

三. 顺序队列

- 顺序队列的进队和出队原则：
  - 进队时，新元素按rear指针位置插入，然后队尾指针增一，即

    `rear = rear + 1`
  - 出队时，将队头指针位置的元素取出，然后队头指针增一，即

    `front = front + 1`
  - 队头指针始终指向队列头元素
  - 队尾指针始终指向队列尾元素的下一个位置

3.4 队列

三. 顺序队列

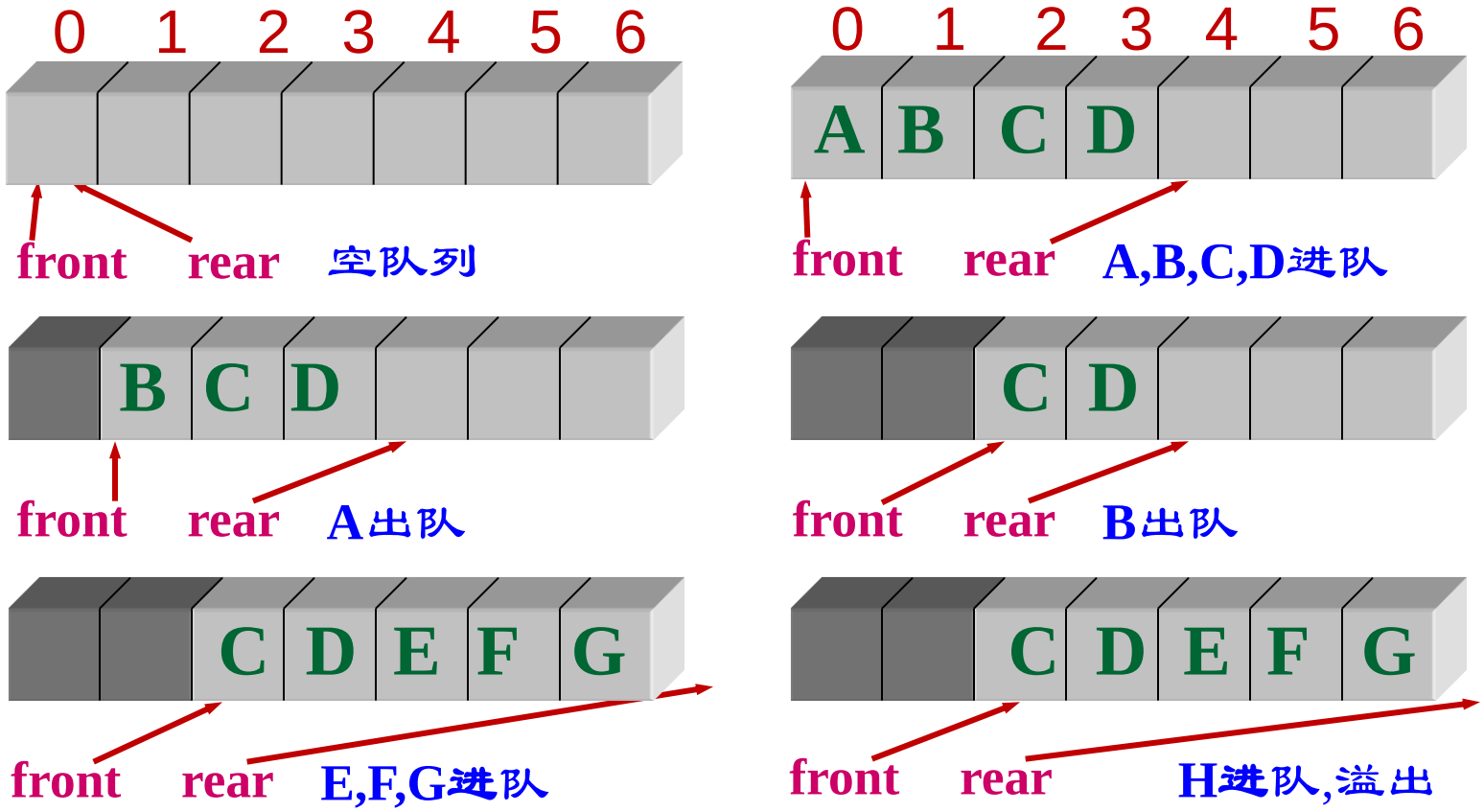
- 顺序队列的定义

```
#define STACK_INIT_SIZE     100        // 队列存储空间的初始分配量
#define STACKINCREMENT    10           //队列存储空间的分配增量
Typedef struct {
      QElemType *base   // 队列存储空间的基址
      int   front;      // 头指针（非空队列，指向队列头元素）
      int   rear;       // 尾指针（非空队列，指向队列尾元素的下一个位置）
} SqQueue;
```

3.4 队列

三. 顺序队列

- 顺序队列的进出队举例：



3.4 队列

三. 顺序队列

- 顺序队列的问题：
  - 当队尾指针指向队列存储结构中的最后单元时，如果再继续插入新的元素，则会产生溢出
  - 当队列发生溢出时，队列存储结构中可能还存在一些空闲位置（已被取走元素的位置）—假溢出

✓ 解决办法之一：设置一个标志位，表示队列是“空”还是“满”

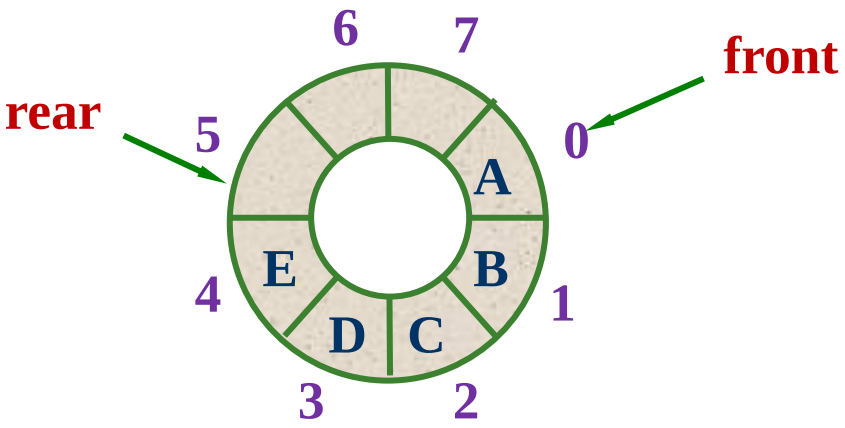
✓ 解决办法之二：队列用链表实现

✓ 解决办法三：将队列存储结构首尾相接，形成循环（环形）队列

3.4 队列

四. 循环队列

- 循环队列采用一组地址连续的存储单元
- 将整个队列的存储单元首尾相连



3.4 队列

四. 循环队列

- 循环队列的定义

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

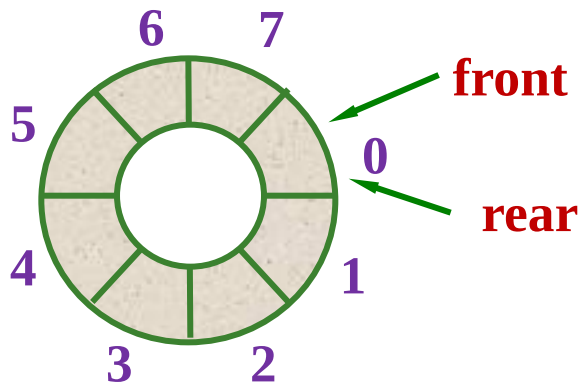
3.4 队列

四. 循环队列

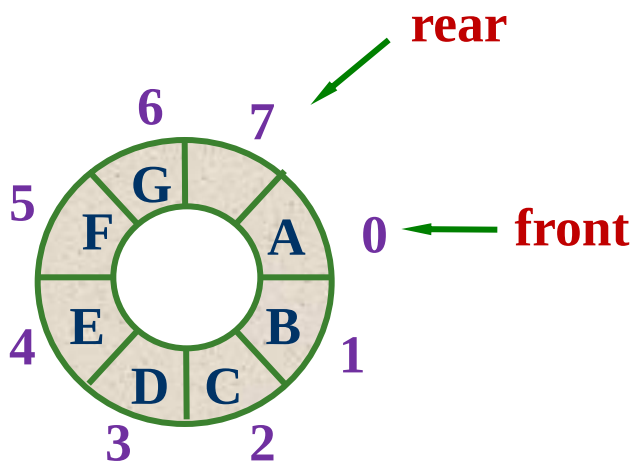
front, rear的取值范围是
【0，MAXQSIZE-1】

- 循环队列的空与满
  - 循环队列空： front = rear
  - 循环队列满 ??

(rear+1) % MAXQSIZE = front　　少用一个元素空间



队列空



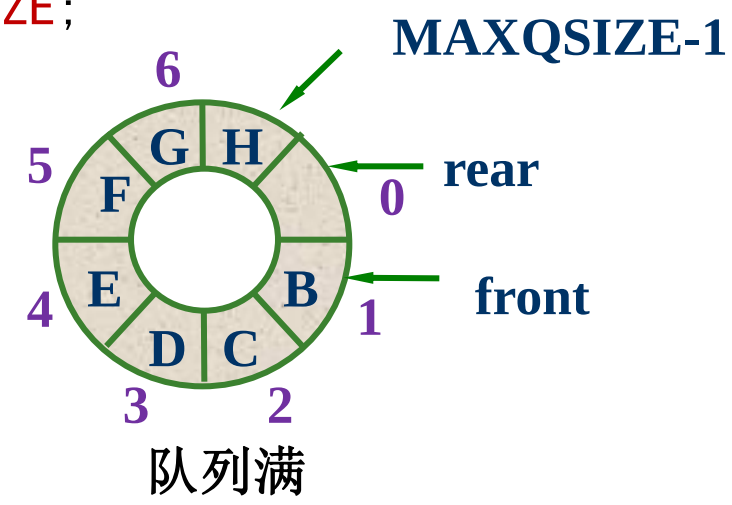
队列满

3.4 队列

四. 循环队列

- 循环队列的插入

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



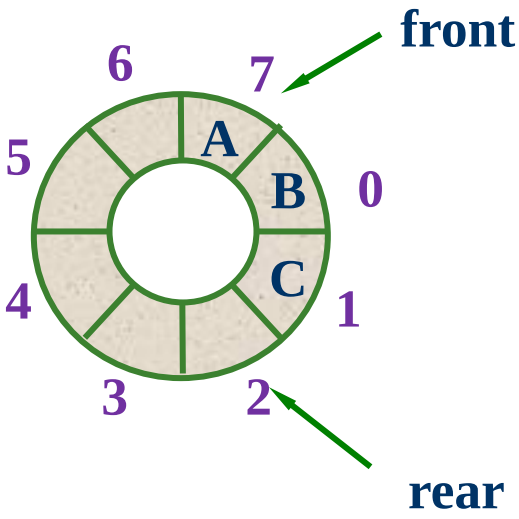
队列满

3.4 队列

四. 循环队列

- 循环队列的删除

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

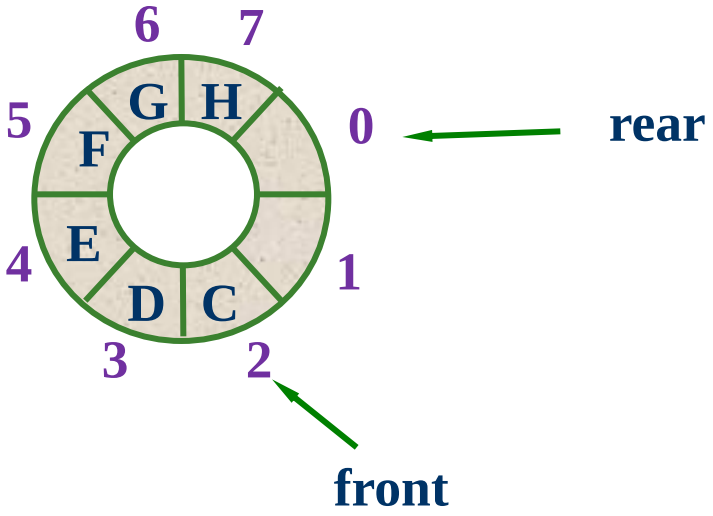


3.4 队列

四. 循环队列

- 循环队列的长度

```
int QueueLength(SqQueue &Q) {
    //返回Q的元素个数，即队列的长度
    return ( Q.rear - Q.front + MAXQSIZE) % MAXQSIZE ;
    }
```



3.4 队列

四. 循环队列

- 队列的数组被当作首尾相接的表处理。
- 队头、队尾指针加1时从*maxSize* −1直接进到0，可用取模(余数)运算实现。
- 队头指针进1: Q.*front* = (Q.*front* + 1)% *MAXSIZE*
- 队尾指针进1: Q.*rear* = (Q.*rear* + 1)% *MAXSIZE*;
- 队列初始化：Q.*front* = Q.*rear* = 0;
- 队空条件：Q.*front* == Q.*rear*;
- 队满条件：(Q.*rear* + 1) % *MAXSIZE* == *Q.front*

  少用了一个空间,以区别队空的状态
- 队列长度：(Q.rear-Q.front+MAXSIZE)%MAXSIZE

小结

- 队列是只允许在表的一端插入元素，在另一端删除元素，特点是先进先出
  - 队列包括队头front和队尾rear，队头只允许删除和访问，队尾只允许插入
    - 插入就是进队 rear+1
    - 删除就是出队 front+1
  - 链表队列，无队列满的问题，动态分配空间
  - 循环队列，通过取模运算解决顺序队列的假溢出问题，只浪费1个空间

练习

1、一个队列的入队序列是1，2，3，4，则队列的输出序列是（）

a) 4，3，2，1

b) 1，4，2，3

c) 1，2，3，4

d) 3，2，4，1

练习

2、假定一个顺序循环队列的队首和队尾指针分别用front和rear表示，则判断队空的条件为（）。

a) **front +1==rear**

b) **rear+1==front**

c) **front==0**

d) **front==rear**

练习

3、假定一个顺序循环队列存储于数组a[N]中，其队首和队尾指针分别用front和rear表示，则判断队满的条件为（）。

a) **（rear-1）%N==front**

b) **（rear+1）%N==front**

c) **（front-1）%N== rear**

d) **（front+1）%N==rear**

练习

4、循环队列用数组a[m]存放其元素值，已知其头尾指针分别用front和rear表示，则队列中的元素个数为（）。

a) （rear-front+m）%m

b) rear-front+1

c) rear-front-1

d) rear-front

练习

5、假定一个链队列的队首和队尾指针分别用front和rear表示，每个结点包含data和next两个域，出队时所进行的指针操作为（）。

a) **front=front-〉data**

b) **front=front-〉next**

c) **rear=rear-〉next**

d) **rear=rear-〉date**

练习

6、设栈S和队列Q的初始状态为空，元素e1，e2，e3，e4，e5，e6依次通过栈S，一个元素出栈后即进入队列Q，若出队的顺序为e2，e4，e3，e6，e5，e1，则栈S的容量至少应该为_____。

a) 2

b) 3

c) 4

d) 5

练习

7、循环队列的队首和队尾下标分别为f、r，最大长度为n，若采用少用一个元素空间的方式，

① 写出判断队空和队满的条件；

② 若n=8，f=3，r=2，请判断队列是否满或空；

③ 若n=9，f=3，r=6，队列从0开始编号，请指出哪几个位置为空？

练习

8、 假设Q[11](下标为从0到10)是一个循环队列,初始状态为front=rear=0；画出分别做完下列操作后队列的头尾指针的装填变化情况，若不能入队，请指出其元素，说明理由..(采用少用一个元素空间的方式)

① d,e,b,g,h入队
② d,e出队
③ i,j,k,l,m入队
④ b 出队
⑤ n,o,p,q,r 入队

练习

9、设长度为n的链队列用单循环链表表示，若只设头指针，则怎样进行入队和出队操作；若只设尾指针呢？