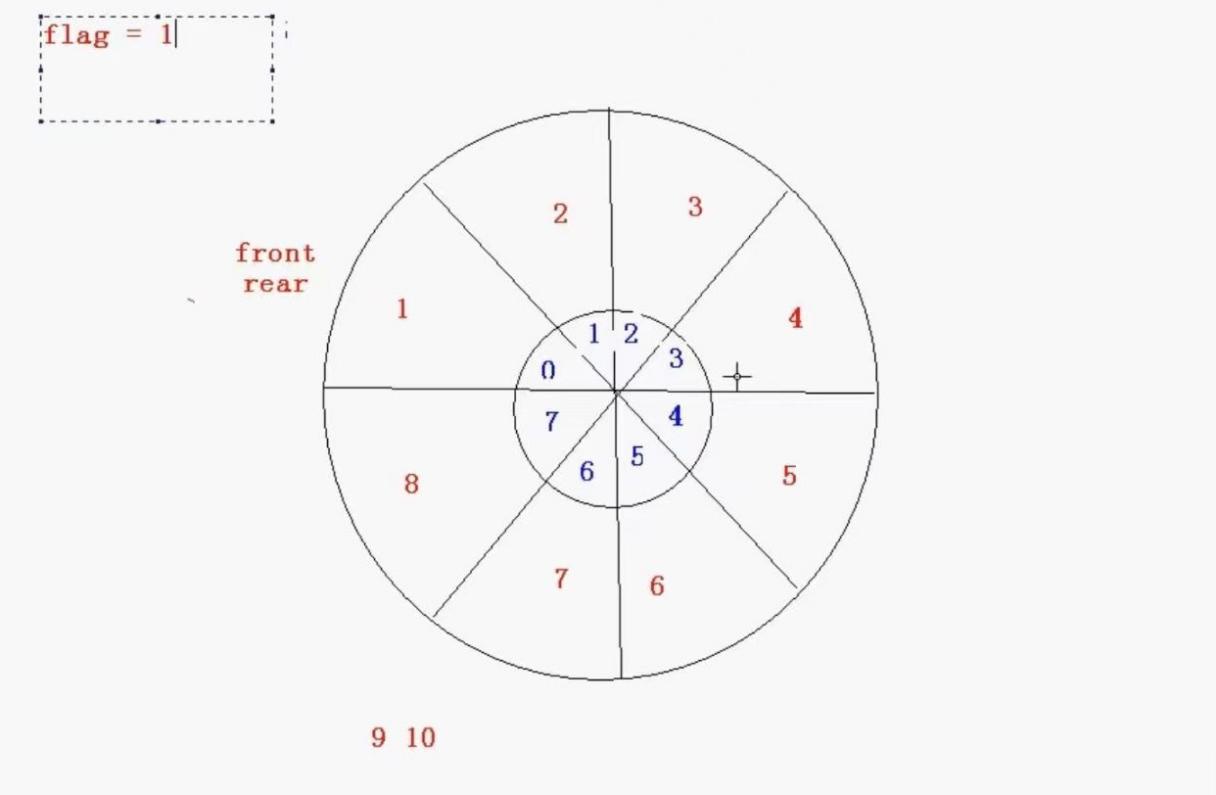
**队列之循环队列的实现**



**判断循环队列满不满的两种方法：**

**方法一**，如图所示，如果满则头尾指针重合，但为了和空区分下来，利用flag进行打标记，1即为满，0即为空。

**方法二**则是**空出一个位置出来**，即如果当前意义上的rear+1=front,则认为队列已满，此时rear所指的位置不能插入新的数据。（本次代码编写采取该种方式）

注：循环队列只是一种想象形式，计算机真实的表示形式仍为线性结构

#include <stdio.h> //标准输入输出的头文件

#include <malloc.h>

#include <assert.h>

typedef int ElemType;

#define MAXSIZE 8

typedef struct Queue {

ElemType\* base;

int front;

int rear;

}Queue;

void InitQueue(Queue\* Q) {

Q->base = (ElemType\*)malloc(sizeof(ElemType) \* MAXSIZE);

assert(Q->base != NULL);

Q->front = Q->rear = 0;

}

//修改

void EnQueue(Queue\* Q,ElemType x) {

if ((Q->rear + 1) % MAXSIZE==Q->front) { //入队判断队列满不满

return;

}

Q->base[Q->rear] = x;

Q->rear = (Q->rear + 1) % MAXSIZE;

}

//Show函数要修改

void ShowQueue(Queue\* Q) {

for (int i = Q->front; i != Q->rear;) {

printf("%d ", Q->base[i]);

i = (i + 1) % MAXSIZE;

}

printf("\n");

}

//修改

void DeQueue(Queue\* Q) {

if (Q->front == Q->rear) { //判断队列空不空的条件

return;

}

Q->front=(Q->front+1)% MAXSIZE;

}

void GetHead(Queue\* Q, ElemType\*v) {

if (Q->front == Q->rear) { //判断队列空不空的条件

return;

}

\*v = Q->base[Q->front];

}

int Length(Queue\* Q) {

return (Q->rear - Q->front);

}

void ClearQueue(Queue\* Q) {

Q->front = Q->rear = 0; //回归到初始设置

}

void DestroyQueue(Queue\* Q) {

ClearQueue(Q);

free(Q->base);

Q->base = NULL;

}

void main() {

Queue Q;

InitQueue(&Q);

for (int i = 1; i <= 7; i++) {

EnQueue(&Q, i);

}

ShowQueue(&Q);

DeQueue(&Q);

EnQueue(&Q, 10);

DeQueue(&Q);

EnQueue(&Q, 20);

ShowQueue(&Q);

}

**程序运行结果：**

