Queue and Circular Queue using Array

Dr. Rahul Das Gupta

Queue using Array

```
#include<stdio.h>
#define MAX_QUEUE_SIZE 10

typedef struct queue
{
   int elements[MAX_QUEUE_SIZE];
   int front, rear;
}QUEUE;

void initialiseQueue (QUEUE *);
int addQueue (int , QUEUE *);
int deleteQueue (int *, QUEUE *);
int isEmptyQueue (QUEUE );
int isOverflow (QUEUE );
```

```
void initialiseQueue (QUEUE *aq)
/*aq = Address of a structure defined as QUEUE
*/
{
      aq->front =-1;
      aq->rear =-1;
}

int addQueue (int item, QUEUE *aq)
{
/*There is no space in Queue for the new Item.
Item can not be added in the Queue.*/
   if (aq->rear == MAX_QUEUE_SIZE-1)
            return 0; /* Unsuccessful Addition*/
/* Spaces are available in Queue for the new
Item. */
   else
   {
            aq->elements [++(aq->rear)]=item;
            return 1;   /* Successful Addition*/
    }
}
```

```
int deleteQueue (int *data, QUEUE *aq)
{
   if( aq->front = = aq->rear) /* Condition for
empty queue.*/
    {
       printf("\n Empty Queue…");
       aq->front = -1;
       aq->rear = -1;
       /*Alternatively: initialiseQueue (aq);*/
       data = NULL;
       return 0;
    }
*data = aq->elements[++(aq->front)];
 return 1;
}
```

```
int isEmptyQueue (QUEUE q)
{
    return (q.front = = q.rear);
}
```

```
int isOverflow (QUEUE q)
{
   return (q.rear = = MAX_QUEUE_SIZE -1);
}
void main( )
{
   QUEUE Q;
   int data;
   initialiseQueue (&Q);
   if(isEmptyQueue (Q))
        printf (“\n Empty Queue…”);

   addQueue (10, &Q);
   addQueue (11, &Q);
   addQueue (12, &Q);
   addQueue (13, &Q);
   addQueue (14, &Q);
   addQueue (15, &Q);
   addQueue (16, &Q);

    while (deleteQueue (&data, &Q)==1)
          printf(“\n Data :%d”, data);
    printf (“\n Empty Queue…”);
  }
```

Circular Queue using Array

```
#include<stdio.h>
#define MAX_QUEUE_SIZE 10

typedef struct queue
{
   int elements[MAX_QUEUE_SIZE];
   int front, rear;
}CQUEUE;

void initialiseCQueue (CQUEUE *);
int addCQueue (int , CQUEUE *);
int deleteCQueue (int *, CQUEUE *);
int checkEmpty (CQUEUE );
int checkFull (CQUEUE );

void initialiseCQueue (CQUEUE *aq)
{
      aq->front =-1;
      aq->rear =-1;
}



int checkFull (CQUEUE q)
{
 if((q.front == q.rear+1)
 || (q.front==0 && q.rear == MAX_QUEUE_SIZE-1))
/*In both these cases the location of the front is
exactly next to the location of the rear.*/
        return 1;
else
        return 0;
}

int checkEmpty(CQUEUE q)
{
      if (q.front == -1)
            return 1;
       else
            return 0;
}

int addCQueue (int item, CQUEUE *aq)
```

```
{

/*There is no space in Queue for the new Item.
Item can not be added in the Queue.*/
 if (checkFull (*aq))
{
       printf("\n Queue Overflow…");
       return 0; /* Unsuccessful Addition*/
}
/* Spaces are available in Queue for the new
Item. */
else
{
   if (aq->front = = -1) /*When the first item is
inserted in the Queue. */
       aq->front=0;
  aq->rear=(aq->rear+1)% MAX_QUEUE_SIZE;
  aq->elements [aq->rear]=item;
  return 1;   /* Successful Addition*/
}
}
```

```
int deleteCQueue (int *data, CQUEUE *aq)
{
/* Empty Queue*/
 if (checkEmpty(*aq))
 {
      printf (“\n Empty Queue…”);
      data=NULL;
      return 0; /* Empty Queue*/
 }
/* Non-empty Queue*/
else
{
    *data= aq->elements [aq->front];
    if (q->front == q->rear)
            initialiseCQueue (aq);
   else
            aq->front=(aq->front+1)% MAX_QUEUE_SIZE;
    return 1; /* Successful retrieval of data*/
 }
}

void main( )
{
    QUEUE cq;
    int data;
    initialiseCQueue (&cq);
    addCQueue (10, &cq);
    addCQueue (11, &cq);
    addCQueue (12, &cq);
    addCQueue (13, &cq);
    addCQueue (14, &cq);
    addCQueue (15, &cq);
    addCQueue (16, &cq);
    while (deleteCQueue (&data, &cq)==1)
            printf("\n Data :%d", data);
     printf ("\n Empty Queue…");
}
```