

tail

front

当前的这种模式会有一个问题,当tail==front时为空,但当放入四个数据时,此时的tail转了一圈又回到了第一个结点,此时tail依然等于front,但此时队列为满.循环链表的问题就在于无法判断空和满。

解决的方法就是：永远空出一个位置不存储数据

比如这里有4个空间，则只能存储3个数据

当tail的下一个结点是front就满了

然后进行pop删除数据,front向前走,走到front == tail是就为空了

代码：

typedef struct {

    int\* a;

    int k;  //队列最多能存多少个数据

    int front;  //头

    int tail;  //尾(队尾数据的下一个)

} MyCircularQueue;

MyCircularQueue\* myCircularQueueCreate(int k) {

    MyCircularQueue\* obj = (MyCircularQueue\*)malloc(sizeof(MyCircularQueue));

    obj->a = (int\*)malloc(sizeof(int)\*(k+1));

    obj->front = 0;

    obj->tail = 0;

    obj->k = k;

    return obj;

}

bool myCircularQueueIsEmpty(MyCircularQueue\* obj) {

    return obj->front == obj->tail;

}

bool myCircularQueueIsFull(MyCircularQueue\* obj) {

    int TailNext = obj->tail+1;

    if(TailNext == obj->k+1)

    {

        TailNext = 0;

    }

    return TailNext == obj->front;

}

bool myCircularQueueEnQueue(MyCircularQueue\* obj, int value) {

    if(myCircularQueueIsFull(obj))

        return false;

    else

    {

        obj->a[obj->tail] = value;

        ++obj->tail;

        if(obj->tail == obj->k+1)

        {

            obj->tail = 0;

        }

        return true;

    }

}

bool myCircularQueueDeQueue(MyCircularQueue\* obj) {

    if(myCircularQueueIsEmpty(obj))

    {

        return false;

    }

    else

    {

        ++obj->front;

        if(obj->front == obj->k+1)

        {

            obj->front = 0;

        }

        return true;

    }

}

int myCircularQueueFront(MyCircularQueue\* obj) {

    if(myCircularQueueIsEmpty(obj))

    {

        return -1;

    }

    else

    {

        return obj->a[obj->front];

    }

}

int myCircularQueueRear(MyCircularQueue\* obj) {

    if(myCircularQueueIsEmpty(obj))

    {

        return -1;

    }

    else

    {

       int tailPrev = obj->tail-1;

       if(tailPrev == -1)

       {

           tailPrev = obj->k;

       }

       return obj->a[tailPrev];

    }

}

void myCircularQueueFree(MyCircularQueue\* obj) {

    free(obj->a);

    free(obj);

}

/\*\*

 \* Your MyCircularQueue struct will be instantiated and called as such:

 \* MyCircularQueue\* obj = myCircularQueueCreate(k);

 \* bool param\_1 = myCircularQueueEnQueue(obj, value);

 \* bool param\_2 = myCircularQueueDeQueue(obj);

 \* int param\_3 = myCircularQueueFront(obj);

 \* int param\_4 = myCircularQueueRear(obj);

 \* bool param\_5 = myCircularQueueIsEmpty(obj);

 \* bool param\_6 = myCircularQueueIsFull(obj);

 \* myCircularQueueFree(obj);

\*/