**循环队列中判断队满与队空**

在引用循环队列前，我们需要了解队列是如何线性实现的。

简单地讲，便是当队列为空时，front = rear = 0,每当插入元素尾指针+1，删除元素是头指针-1。但是，我们会发现一个问题，如上面的第四个图，0，1，2三个空间并没有使用。因此，为了占用该空间，我们使用了循环队列来实现。

循环队列原理图：

我们可以发现，当循环队列属于上图的d1情况时，是无法判断当前状态是队空还是队满。为了达到判断队列状态的目的，可以通过牺牲一个存储空间来实现。

如上图d2所示，

**队头指针在队尾指针的下一位置时，队满。** Q.front == (Q.rear + 1) % MAXSIZE 因为队头指针可能又重新从0位置开始，而此时队尾指针是MAXSIZE - 1，所以需要求余。

**当队头和队尾指针在同一位置时，队空。** Q.front == Q.rear;

以下是实现的代码：

#include <stdio.h>
#include <malloc.h>
#define MAXSIZE 100 //最大队列长度
#define OK 1
#define ERROR 0
typedef int ElemType;
typedef int Status;
typedef struct {
ElemType \*base; *//队列空间*
int front; *//队头指针*
int rear; *//队尾指针，若队尾不为空，则指向队尾元素的下一个位置*
}SqQueue;
*//初始化循环队列*
Status initQueue(SqQueue &Q) {
Q.base = (ElemType \*) malloc(MAXSIZE \* sizeof(ElemType)); *//申请空间*
Q.front = Q.rear = 0; *//队空*
return OK;
}
*//入队*
Status enQueue(SqQueue &Q, ElemType e) {
if ((Q.rear + 1) % MAXSIZE == Q.front) return ERROR; *//队满，无法添加*
Q.base[Q.rear] = e; *//插入元素*
Q.rear = (Q.rear + 1) % MAXSIZE; *//队尾指针+1*
return OK;
}
*//出队*
Status deQueue(SqQueue &Q, ElemType &e) {
if (Q.front == Q.rear) return ERROR; *//队空，无法删除*
e = Q.base[Q.front];
Q.front = (Q.front + 1) % MAXSIZE; *//队头指针+1*
return OK;
}
*//返回队列长度*
Status length(SqQueue &Q) {
return (Q.rear - Q.front + MAXSIZE) % MAXSIZE;
}