这里的队列，在逻辑上的设置除了在队列的的创建时，Front 和 Rear 会相等。而在正式插入时。由于IsFull(Queue q)作为判断条件，【反正举个例子就知道了，设数组长度5，比如Front在0位且有元素，Real在4位且有元素，这时再插入元素】不会出现Front中有元素的时候，Rear还指向Front位置的情况。

注：

现在这里的 Front 、Read都是 int 类型，存放的是动态数组的下标。

# 链表的顺序存储实现（摘自中国慕课大学）

1. typedef int Position;
2. struct QNode {
3. ElementType \*Data;     /\* 存储元素的数组 \*/
4. Position Front, Rear;  /\* 队列的头、尾指针 \*/
5. int MaxSize;           /\* 队列最大容量 \*/
6. };
7. typedef struct QNode \*Queue;
9. Queue CreateQueue( int MaxSize )
10. {
11. Queue Q = (Queue)malloc(sizeof(struct QNode));
12. Q->Data = (ElementType \*)malloc(MaxSize \* sizeof(ElementType));
13. Q->Front = Q->Rear = 0;
14. Q->MaxSize = MaxSize;
15. return Q;
16. }
18. bool IsFull( Queue Q )
19. {
20. return ((Q->Rear+1)%Q->MaxSize == Q->Front);
21. }
23. bool AddQ( Queue Q, ElementType X )
24. {
25. if ( IsFull(Q) ) {
26. printf("队列满");
27. return false;
28. }
29. else {
30. Q->Rear = (Q->Rear+1)%Q->MaxSize;
31. Q->Data[Q->Rear] = X;
32. return true;
33. }
34. }
36. bool IsEmpty( Queue Q )
37. {
38. return (Q->Front == Q->Rear);
39. }
41. ElementType DeleteQ( Queue Q )
42. {
43. if ( IsEmpty(Q) ) {
44. printf("队列空");
45. return ERROR;
46. }
47. else  {
48. Q->Front =(Q->Front+1)%Q->MaxSize;
49. return  Q->Data[Q->Front];
50. }
51. }