PROLOG ACADEMY

- **DATA STRUCTURE**

By- MASEERA ALI

- Book followed - Data structures by Seymour Lipschutz (Schaum Series)

LET’S START!

QUEUE

Like Stack, Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.
The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Operations on Queue:

Enqueue: Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
Dequeue: Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
Front: Get the front item from queue.
Rear: Get the last item from queue.

Applications of Queue Data Structure

Queue is used when things don’t have to be processed immediatly, but have to be processed in First In First Out order like Breadth First Search.

1) When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.

2) When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

QINSERT

Procedure to insert an element in ITEM into a queue

1.[Queue already filled]

If (Front=1 and Rear=N) OR (Front =Rear+1)

Write: Overflow and return

2.[Find the new value of Rear]

If(Front=NULL) then Set Front=1,Rear=1

Else if Rear=N then Set Rear=1

Else set Rear=Rear+1

[End of If strucuture]

3.Set Queue[Rear]=Item

4.Return

QDELETE

Procedure to delete an element from Queue

1.[Queue already empty]

if(Front=NULL) then write Underflow return

2.Set Item=queue[Front]

3.[Find new Value of Front]

If (Front=Rear) Set Front=NULL, Rear=NULL

Else if Front=N then Set Front=1

Else set Front=Front+1

[End of If structure]

4.Return

CODE IN C

```
 #include <stdio.h>
#include <stdlib.h>
#define MAX 5

int cqueue[MAX];
int front = -1;
int rear = -1;


void insert( )
{
        int item;
        if((front==0 && rear==MAX-1) || (front==rear+1))
        {
                printf("Queue is Overflow\n");
                return;
        }
        if (front==-1)/*If queue is empty*/
        {
                front = 0;
                rear = 0;
        }
        else
        if (rear==MAX-1) /*rear is at last position of queue*/
                rear = 0;
        else
                rear = rear + 1;
                printf("Input the element for insertion :");
                scanf("%d", &item);
                cqueue[rear] = item;
}
```

```
void del( )
{
        if (front == -1)
        {
                printf("Queue Underflow\n");
                return;
        }
        printf ("Deleted element from queue is : %d\n", cqueue[front]);
        if(front == rear)/* queue has only one element */
        {
                front = -1;
                rear = -1;
        }
        else
        if(front==MAX-1)
                front = 0;
        else
                front = front + 1;
}

  void display( )
    {
            int front_pos = front, rear_pos = rear;
            if(front == -1)
            {
                    printf("Queue is empty\n");
                    return;
            }
            printf ("Queue elements are:\n");
            if(front_pos <= rear_pos)
            while(front_pos <= rear_pos)
            {
                    printf(" %d\t", cqueue[front_pos]);
                    front_pos++;
            }
```

```
        else
        {
                while(front_pos <= MAX-1)
                {
                        printf(" %d\t", cqueue[front_pos]);
                        front_pos++;
                }
                front_pos = 0;
                while(front_pos <= rear_pos)
                {
                        printf(" %d\t", cqueue[front_pos]);
                        front_pos++;
                }
        }
        printf("\n");
  }
void main ( )
  {
        int choice;
        while (1)
        {
                printf ("1.Insert\n2.Delete\n3.Display\n4.Exit\n");
                printf ("Enter your choice :");
                scanf ("%d", &choice);
                switch(choice)
                {
                        case 1 :insert( );
                        break;
                        case 2 :del( );
                        break;
                        case 3 :display( );
                        break;
                        case 4 :exit(1);
                        default:printf("Wrong choice\n");
                }
        }
  }
```

DEQUE

Deque or Double Ended Queue is a generalized version of Queue data structure that allows insert and delete at both ends.

Operations on Deque:

insetFront(): Adds an item at the front of Deque.

insertLast(): Adds an item at the rear of Deque.

deleteFront(): Deletes an item from front of Deque.

deleteLast(): Deletes an item from rear of Deque.

Implementation:

A Deque can be implemented either using a doubly linked list or circular array.

TYPES OF DEQUE

1. Input restricted Deque

   A deque which allows insertion at only one end of the list but allows deletion at both ends of the list.

2.Output restricted Deque

   A deque which allows deletions at only one end of the list but allows insertions at both ends of the list.