**Practical No: 12(E)**

**Practical Title:** Perform operations on Double ended queue.

**Aim:** A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque..

**Pre-requisite:**

* + Knowledge of Queue
  + Types of queue
  + Knowledge of double ended queue and different operations that can be performed on it

**Objective:** To simulate deque with functions to add and delete elements from either end of the deque.

**Input :**

* Size of array.
* Elements in queue

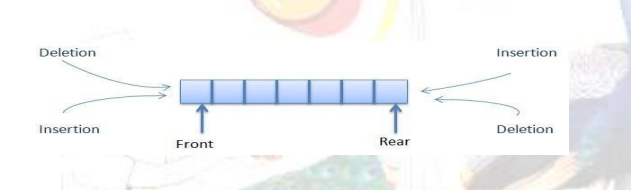
**Outcome:**

* Result of deque with functions to add and delete elements from either end of the deque

**Theory:**

**Double-Ended Queue**

A double-ended queue is an abstract data type similar to an simple queue, it allows you to insert and delete from both sides means items can be added or deleted from the front or rear end.



**Algorithm for Insertion at rear end**

Step -1: [Check for overflow]

if(rear==MAX)

Print("Queue is Overflow”);

return;

Step-2: [Insert element]

else

rear=rear+1

q[rear]=no;

[Set rear and front pointer]

if rear=0

rear=1;

if front=0

front=1;

Step-3: return

Implementation of Insertion at rear end

void add\_item\_rear()

{

int num;

printf("\n Enter Item to insert : ");

scanf("%d",&num);

if(rear==MAX)

{

printf("\n Queue is Overflow");

return;

}

else

{

rear++;

q[rear]=num;

if(rear==0)

rear=1;

if(front==0)

front=1;

}

}

**Algorithm for Insertion at front end**

Step-1 : [Check for the front position]

if(front<=1)

Print (“Cannot add item at front end”);

return;

Step-2 : [Insert at front]

else

front=front-1;

q[front]=no;

Step-3 : Return

**Implementation of Insertion at front end**

void add\_item\_front()

{

int num;

printf("\n Enter item to insert:");

scanf("%d",&num);

if(front<=1)

{

printf("\n Cannot add item at front end");

return;

}

else

{

front--;

q[front]=num;

}

}

Algorithm for Deletion from front end

Step-1 [ Check for front pointer]

if front=0

print(" Queue is Underflow”);

return;

Step-2 [Perform deletion]

else

no=q[front];

print(“Deleted element is”,no);

[Set front and rear pointer]

if front=rear

front=0;

rear=0;

else

front=front+1;

Step-3 : Return

**Implementation of Deletion from front end**

void delete\_item\_front()

{

int num;

if(front==0)

{

printf("\n Queue is Underflow\n");

return;

}

else

{

num=q[front];

printf("\n Deleted item is %d\n",num);

if(front==rear)

{

front=0;

rear=0;

}

else

{

}

}

**Algorithm for Deletion from rear end**

Step-1 : [Check for the rear pointer]

if rear=0

print(“Cannot delete value at rear end”);

return;

Step-2: [ perform deletion]

else

no=q[rear];

[Check for the front and rear pointer]

if front= rear

front=0;

rear=0;

else

rear=rear-1;

print(“Deleted element is”,no);

Step-3 : Return

**Implementation of Deletion from rear end**

void delete\_item\_rear()

{

int num;

if(rear==0)

{

printf("\n Cannot delete item at rear end\n");

return;

}

else

{

num=q[rear];

if(front==rear)

{

front=0;

rear=0;

}

else

{

rear--;

printf("\n Deleted item is %d\n",num);

}

}

}

Flowchart :

Draw flowchart for above algorithms.

**Conclusion:** By this way, we can perform operations on double ended queue

**Question Bank:**

1. What is queue?

2. Types of queue

3. What is double ended queue?

4. How to insert the new node in Doubly Link List?

5. How to delete the node from front of Doubly Link List ?

6. How to delete the node from end of Doubly Link List ?

7. How to delete the node in between of Doubly Link List