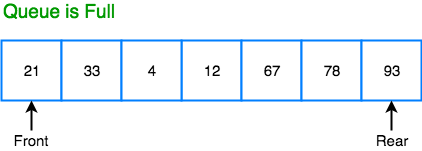
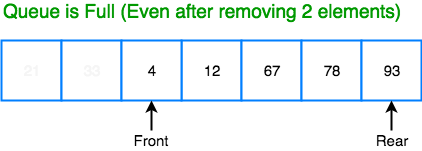
**Drawback in Queue-** In a Linear queue, once the queue is completely full, it's not possible to insert more elements. Even if we de-queue the queue to remove some of the elements, until the queue is reset, no new elements can be inserted. You must be wondering why?



When we **dequeue** any element to remove it from the queue, we are actually moving the **front** of the queue forward, thereby reducing the overall size of the queue. And we cannot insert new elements, because the **rear** pointer is still at the end of the queue.

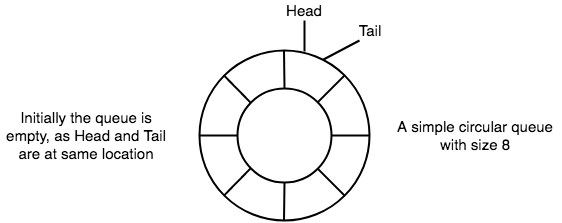


The only way is to reset the linear queue, for a fresh start.

**Circular Queue** is also a linear data structure, which follows the principle of **FIFO**(First In First Out), but instead of ending the queue at the last position, it again starts from the first position after the last, hence making the queue behave like a circular data structure.

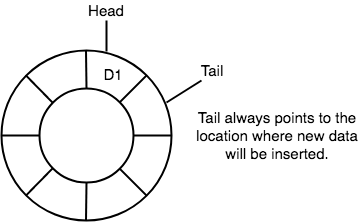
Basic features of Circular Queue

1. In case of a circular queue, head pointer will always point to the front of the queue, and tail pointer will always point to the end of the queue.
2. Initially, the head and the tail pointers will be pointing to the same location, this would mean that the queue is empty.

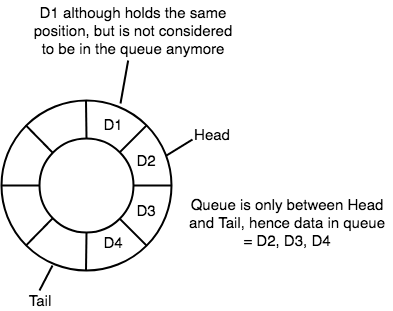


New data is always added to the location pointed by the tail pointer, and once the data is added, tail pointer is incremented to point to the next available location.

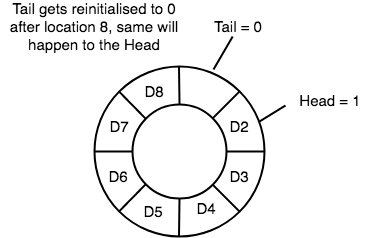
3. New data is always added to the location pointed by the tail pointer, and once the data is added, tail pointer is incremented to point to the next available location.



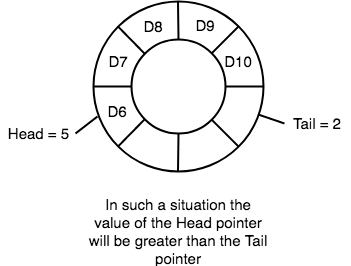
4.In a circular queue, data is not actually removed from the queue. Only the head pointer is incremented by one position when dequeue is executed. As the queue data is only the data between head and tail, hence the data left outside is not a part of the queue anymore, hence removed.



5.The head and the tail pointer will get reinitialised to 0 every time they reach the end of the queue.



6.Also, the head and the tail pointers can cross each other. In other words, head pointer can be greater than the tail. Sounds odd? This will happen when we dequeue the queue a couple of times and the tail pointer gets reinitialised upon reaching the end of the queue.



**#include<stdio.h>**

**#include<conio.h>**

**#define size 100**

**void insertion(int \*arr, int \*front, int \*rear);**

**void deletion(int \*front, int \*rear);**

**void display(int \*arr,int front,int rear);**

**void main()**

**{**

**int arr[size],choice,front,rear;**

**front=rear=-1;**

**while(1)**

**{**

**clrscr();**

**printf("\n\n\t\tThats the program of circular queue through ARRAY");**

**printf("\n\t\t(MAX Size = 100)\n\n\n\n");**

**printf("\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf("\n\n\t\t1. Insert the element OR");**

**printf("\n\n\t\t2. Withdraw the element OR");**

**printf("\n\n\t\t3. Display the contents OR");**

**printf("\n\n\t\t4. Exit.");**

**printf("\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**

**printf("\n\n\n\n\t\tWhich operation you wanna to perform =");**

**fflush(stdin);**

**scanf("%d",&choice);**

**switch(choice)**

**{**

**case 1: insertion(arr,&front,&rear);**

**break;**

**case 2: deletion(&front,&rear);**

**break;**

**case 3: display(arr,front,rear);**

**break;**

**case 4: printf("\n\n\n\n\t\tYou have successfully terminated from the program.");**

**getch();**

**exit(0);**

**default:printf("\n\n\n\n\t\tYour choice is not valid.");**

**printf("\n\n\t\tPress any key to rechoice.......");**

**getch();**

**continue;**

**}**

**}**

**}**

**void insertion(int \*arr, int \*front, int \*rear)**

**{**

**int num;**

**clrscr();**

**if(((\*rear)+1)%size==(\*front))**

**{**

**printf("\n\n\t\tArray is FULL FULL FULL.......");**

**printf("\n\n\t\tPlease first delete some element then try again.");**

**getch();**

**}**

**else**

**{**

**if((\*rear)+1==size)**

**(\*rear)=0;**

**else**

**(\*rear)++;**

**if((\*front)==-1)**

**(\*front)++;**

**printf("\n\n\t\tEnter the number to insert = ");**

**fflush(stdin);**

**scanf("%d",&num);**

**arr[\*rear]=num;**

**printf("\n\n\t\tYour element has been successfully inserted.");**

**getch();**

**}**

**}**

**void deletion(int \*front, int \*rear)**

**{**

**clrscr();**

**if((\*front)==-1)**

**{**

**printf("\n\n\t\tThere is no element in the array.");**

**printf("\n\n\t\tPlease insert some element and then try again.");**

**getch();**

**return;**

**}**

**else**

**if((\*front)==(\*rear))**

**{**

**(\*front)=(\*rear)=-1;**

**printf("\n\n\t\tYour element has been successfully deleted.");**

**printf("\n\n\t\tPress any key to continue.......");**

**getch();**

**return;**

**}**

**else**

**{**

**if((\*front)+1==size)**

**(\*front)=0;**

**else**

**(\*front)++;**

**printf("\n\n\t\tYour element has been successfully deleted.");**

**printf("\n\n\t\tPress any key to continue.......");**

**getch();**

**return;**

**}**

**}**

**void display(int \*arr,int front,int rear)**

**{**

**int count,i;**

**clrscr();**

**if(front==-1)**

**{**

**printf("\n\n\t\tThere is no element in the list to display.");**

**printf("\n\n\t\tReturn back and insert some elements first......");**

**getch();**

**return;**

**}**

**printf("\n\n\t\tYour element is as follows --\n\n");**

**if(rear<front)**

**count=size+rear-front+1;**

**else**

**count=rear-front+1;**

**for(i=0;i<count;i++)**

**{**

**printf("\n\t\t\t%d",arr[front]);**

**front++;**

**if(front==size)**

**front=0;**

**}**

**getch(); }**