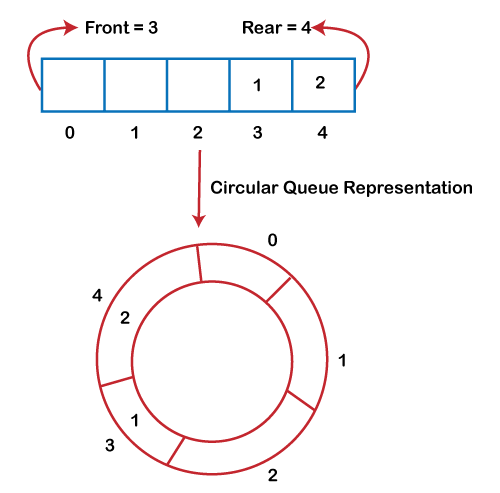
Circular Queue

There was one limitation in the array implementation of Queue. If the rear reaches to the end position of the Queue then there might be possibility that some vacant spaces are left in the beginning which cannot be utilized. So, to overcome such limitations, the concept of the circular queue was introduced.



What is a Circular Queue?

A circular queue is similar to a linear queue as it is also based on the FIFO (First In First Out) principle except that the last position is connected to the first position in a circular queue that forms a circle. It is also known as a Ring Buffer.

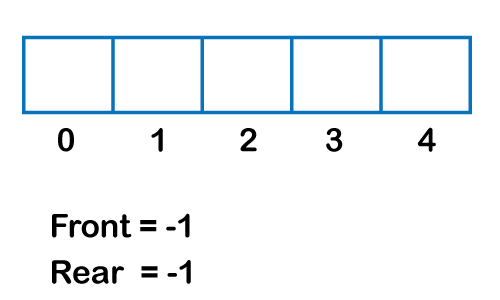
Applications of Circular Queue

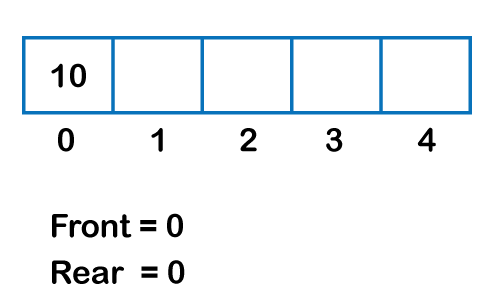
The circular Queue can be used in the following scenarios:

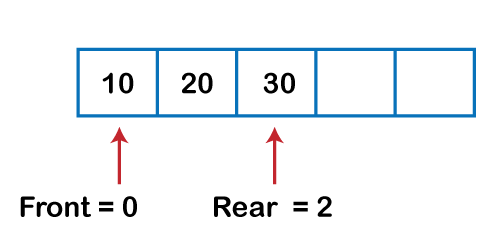
Memory management: The circular queue provides memory management. As we have already seen that in linear queue, the memory is not managed very efficiently. But in case of a circular queue, the memory is managed efficiently by placing the elements in a location which is unused.

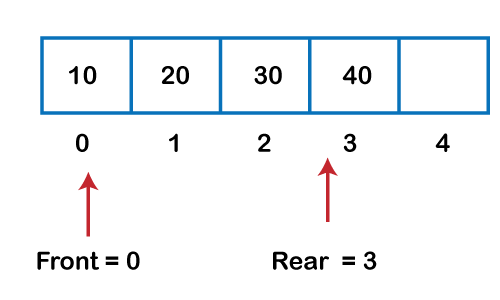
CPU Scheduling: The operating system also uses the circular queue to insert the processes and then execute them.

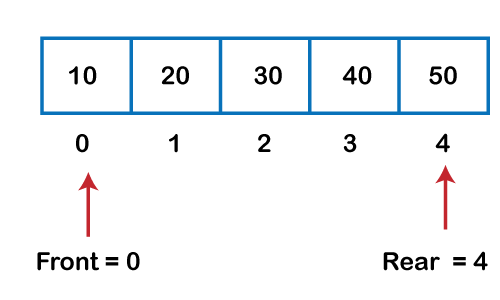
Traffic system: In a computer-control traffic system, traffic light is one of the best examples of the circular queue. Each light of traffic light gets ON one by one after every jinterval of time. Like red light gets ON for one minute then yellow light for one minute and then green light. After green light, the red light gets ON.

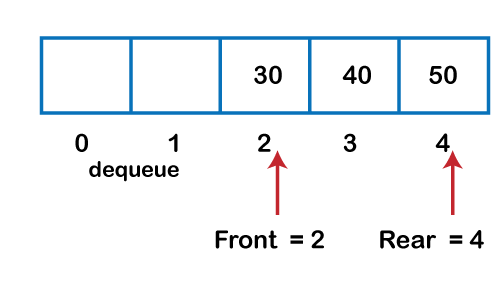


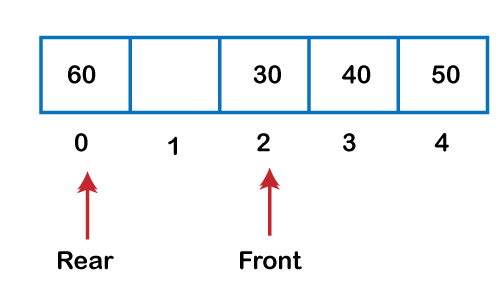


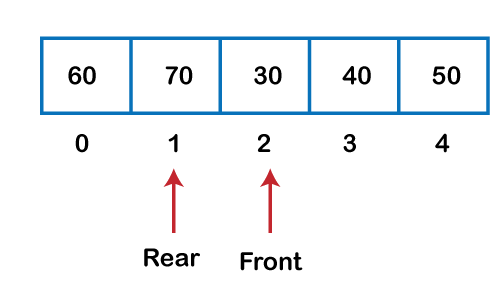












#include<stdlib.h>

#include<stdio.h>

#define max 5

int front=-1,rear=-1;       // global variable

int CQueue[max];

void insert();

int delete();

void display();

int main()

{

    int w,no;

    for(;;)

    {

        printf("\n1. Insert");

        printf("\n2. Delete");

        printf("\n3. Display");

        printf("\n4. EXIT");

        printf("\nEnter what you want :");

        scanf("%d",&w);

        switch(w)

        {

        case 1:

            insert();

            break;

        case 2:

            no=delete();

            break;

        case 3:

            display();

            break;

        case 4:

            exit(1);

        default:

            printf("\nInvalid Choice !!\n");

        }

    }

}

void insert()

{

    int no;

    if((front ==0 && rear == max-1) || front == rear+1)

    {

        printf("\nCircular Queue Is Full !\n");

        return;

    }

    printf("\nEnter a number to Insert :");

    scanf("%d",&no);

    if(front==-1)

        front=front+1;

    if(rear==max-1)

        rear=0;

    else rear=rear+1;

        CQueue[rear]=no;

}

int delete()

{

    int e;

    if(front==-1)

    {

        printf("\nThe Circular Queue is Empty !!\n");

    }

    e=CQueue[front];

    if(front==max-1)

        front=0;

    else if(front==rear)

    {

        front=-1;

        rear=-1;

    }

    else front=front+1;

    printf("\n%d was deleted !\n",e);

    return e;

}

void display()

{

    int i;

    if(front==-1)

    {

        printf("\nThe Circular Queue is Empty ! Nothing To Display !!\n");

        return;

    }

    i=front;

    if(front<=rear)

    {

        printf("\n\n");

        while(i<=rear)

            printf("%d ",CQueue[i++]);

        printf("\n");

    }

    else

    {

        printf("\n\n");

        while(i<=max-1)

           printf("%d ",CQueue[i++]) ;

        i=0;

        while(i<=rear)

            printf("%d ",CQueue[i++]);

        printf("\n");

    }

}

Perfect version:

#include<stdio.h>

#define MAX 5

int cque[20],front=-1,rear=-1;

void enqueue()

{

    int ele;

    if( (rear==MAX-1 && front==0) || (front==rear+1) )

        printf("Circular queue is full\n");

    else

    {

        printf("Enter new element to be enqueued:");

        scanf("%d",&ele);

        if(front==-1 && rear==-1)

        {

            front=0;

            rear=0;

        }

        else

        {

            if(rear==MAX-1 && front!=0)

               rear=0;

            else

                rear++;

        }

        cque[rear]=ele;

        printf("Element enqueued successfully\n");

    }

}

void dequeue()

{

    if(front==-1)

        printf("Circular queue is empty\n");

    else

    {

        printf("Deleted element is:%d\n",cque[front]);

        if(front==MAX-1 && rear!=MAX-1)

            front=0;

        else if(front==MAX && rear ==MAX-1)

            front=rear=-1;

        else if(front==rear)

            front=rear=-1;

        else

            front++;

    }

}

void peek()

{

int i;

if(front==-1)

    printf("Cqueue is empty\n");

else

{

if(front<=rear)

{

    for(i=front;i<=rear;i++)

        printf("%d\t",cque[i]);

}

else

{

    for(i=front;i<=MAX-1;i++)

        printf("%d\t",cque[i]);

    for(i=0;i<=rear;i++)

        printf("%d\t",cque[i]);

}

}

}

void isEmpty()

{

if(front==-1)

    printf("Circular queue is empty\n");

else

    printf("Circular queue is not empty\n");

}

void isFull()

{

if( (rear==MAX-1 && front==0) || (front==rear+1) )

        printf("Circular queue is full\n");

else

    printf("Circular queue is not full\n");

}

//main

void main()

{

    int ch;

    char rep='y';

    do

    {

        printf(".....MENU.....\n1.Enqueue\n2.Dequeue\n3.Peek\n");

        printf("4.IsEmpty\n5.IsFull\n");

        printf("Select one option:");

        scanf("%d",&ch);

        switch(ch)

        {

            case 1:

                enqueue();break;

            case 2:

                dequeue();break;

            case 3:

                peek();break;

            case 4:

                isEmpty();break;

            case 5:

                isFull();break;

            default:

                printf("Invalid choice\n");

        }

        printf("want to repeat y/n:");

        scanf(" %c",&ch);

    }while(ch=='y'||ch=='Y');

}

//version2

#include<stdio.h>

#define MAX 5

int cque[20],front=-1,rear=-1;

void enqueue()

{

    int ele;

    if( (rear==MAX-1 && front==0) || (front==rear+1) )

        printf("Circular queue is full\n");

    else

    {

        printf("Enter new element to be enqueued:");

        scanf("%d",&ele);

        if(front==-1 && rear==-1)

        {

            front=0;

            rear=0;

        }

        else

        {

            if(rear==MAX-1 && front!=0)

               rear=0;

            else

                rear++;

        }

        cque[rear]=ele;

        printf("Element enqueued successfully\n");

    }

}

void dequeue()

{

    if(front==-1)

        printf("Circular queue is empty\n");

    else

    {

        printf("Deleted element is:%d\n",cque[front]);

        if(front==MAX-1)

            front=0;

        else if(front==rear)

            front=rear=-1;

        else

            front++;

    }

}

void peek()

{

int i;

if(front==-1)

    printf("Cqueue is empty\n");

else

{

if(front<=rear)

{

    for(i=front;i<=rear;i++)

        printf("%d\t",cque[i]);

}

else

{

    for(i=front;i<=MAX-1;i++)

        printf("%d\t",cque[i]);

    for(i=0;i<=rear;i++)

        printf("%d\t",cque[i]);

}

}

}

void isEmpty()

{

}

void isFull()

{

}

//main

void main()

{

    int ch;

    char rep='y';

    do

    {

        printf(".....MENU.....\n1.Enqueue\n2.Dequeue\n3.Peek\n");

        printf("4.IsEmpty\n5.IsFull\n");

        printf("Select one option:");

        scanf("%d",&ch);

        switch(ch)

        {

            case 1:

                enqueue();break;

            case 2:

                dequeue();break;

            case 3:

                peek();break;

            case 4:

                isEmpty();break;

            case 5:

                isFull();break;

            default:

                printf("Invalid choice\n");

        }

        printf("want to repeat y/n:");

        scanf(" %c",&ch);

    }while(ch=='y'||ch=='Y');

}