AIM: - Circular queue using array

Algorithm:-

Step 1: Initialize an array queue of size n, where n is the maximum number of elements that the queue can hold.

Step 2: initialize two variables front and rear to -1.

Step 3: To enqueue an element x onto the queue, do the following:

Increment by 1

if rear is equal to n, set rear to 0.

if front is -1, set front to 0

set queue[rear] to x

Step 4: To dequeue an element from the queue, do the following:

check if the queue is empty by checking if front is -1. If it is, return an error message indicating that the queue is empty.

set x to queue[front]

if front is equal to rear, set front and rear to -1

otherwise, increment front by 1 and if front is equal to n, set front to 0

return x

Program: -

#include<stdio.h>//standard input output header file//

#include<stdlib.h>//standard library header file//

#define N 3//defining the size//

int front=-1,rear=-1,queue[N];//initializes the front and the rare value//

void enqueue()//enqueue function//

{

int val;//initializes the val//

printf("enter the value ::\n");//prints the statement//

scanf("%d",&val);//scans the value and the initializes the memory location//

if((rear+1)%N==front)//checks the condition//

{

printf("Over flow \n");//prints the statement//

return ;

}

else if ((front==-1)&&(rear==-1))//checks the condition//

{

front=0;

rear=0;

}

else if((rear==N-1)&&(front!=0))//checks the condition//

rear=0;

else

rear=(rear+1)%N;

queue[rear]=val;//queue rare value is initialized to the val//

printf("Value enqueued\n");//prints the statement//

}

void dequeue()//dequeue function//

{

int val;//initializes the val//

if ((front==-1)&&(rear==-1))//checks the condition//

{

printf("Underflow\n");//prints the statement//

return ;

}

val=queue[front];//val is initialized to queue of front//

if(front==rear)//checks the condition//

{

front=-1;

rear=-1;

}

else if(front==N-1)//checks the condition//

front=0;

else

front=front+1;//front value is incremented//

printf("value=%d Dequeued\n",val); //prints the statement//

}

void display()//display function//

{

int i=front;//initializes and declaring the I value to the front//

if ((front==-1)&&(rear==-1))//checks the condition//

{

printf("Queue is empty");//prints the statement//

return ;

}

else

{

printf("elements in a queue are::\n");//prints the statement//

while(i!=rear)//while loop//

{

printf("%d\t",queue[i]); //prints the statement//

i=(i+1)%N;

}

printf("%d",queue[i]); //prints the statement//

}

printf("\n");//prints the statement//

}

int main()//main function//

{

int choice;//initializes the choice//

while(1)//while loop//

{

printf("select the operation::\n");//prints the statement//

printf("1.insert\n2.delete\n3.display\n4.exit\n");//prints the statement//

printf("enter the number to be performed");//prints the statement//

scanf("%d",&choice);//scans the value and initializes the memory location//

switch(choice)//switch case//

{

case 1:enqueue();//enqueue function//

break;

case 2:dequeue();//dequeue function//

break;

case 3:display();//display function//

break;

case 4:exit(0);//exit function//

default:printf("invalid choice\n");//prints the statement//

}

}

}

Screenshot of the Output: -



