

Name:

Div- 3 Year- SE Roll no- Batch- B

Experiment no 4

Aim: To implement Circular Queue ADT using array

Objective:

Circular Queue offer a quick and clean way to store FIFIO data with maximum size

Theory:

A Circular Queue is an extended version of a normal queue where the last element of the queue is connected to the first element of the queue forming a circle. The operations are performed based on FIFO (First In First Out) principle. It is also called 'Ring Buffer

Algorithm:

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

Initialize two variables front and rear to -1.

Enqueue: To enqueue an element x into the queue, do the following:

Increment rear 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.

Dequeue: 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.

Circular Queue implementation in C

CODE:

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<conio.h>

#define max 10

int cqueue\_arr[max];

int front=-1;

int rear=-1;

void display();

void insert(int item) ;

int del();

int peek();

int isEmpty();

int isFull();

int main()

{

int choice,item;

while(1)

{

printf("\n1.INSERT\n 2.DELETE\n 3.PEEK\n 4.DISPLAY\n 5.QUIT");

printf("\nENTER THE CHOICE:\n");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("ENTER THE ELEMENTFOR INSERTION:");

scanf("%d",&item);

insert(item);

break;

case 2: printf("ElEMENT DELETED is %d\n",del());

break;

case 3: printf(" ELEMENT AT THE FRONT:%d",peek());

break;

case 4: display();

break;

case 5: exit(1);

break;

default:printf("WRONG CHOICE");

}

}

}

void insert(int item)

{

if(isFull())

{

printf(" OVERFLOW");

return ;

}

if(front == -1)

front=0;

if(rear==max-1)

rear=0;

else

rear = rear+1;

cqueue\_arr[rear]=item;

}

int del()

{

int item;

if(isEmpty())

{

printf("UNDERFLOW");

}

item=cqueue\_arr[front];

if(front==rear)

{

front=rear=-1;

}

else if(front==max-1)

front =0;

else

front=front+1;

return item;

}

int isEmpty()

{

if(front==-1)

return 1;

else

return 0;

}

int isFull()

{

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

return 1;

else

return 0;

}

int peek()

{

if(isEmpty())

{

printf("UNDERFLOW");

exit(1) ;

}

return cqueue\_arr[front];

}

void display()

{

int i;

if(isEmpty())

{

printf("QUEUE IS EMPTY");

return ;

}

printf("QUEUE ELEMENTS:\n");

i=front;

if(front<=rear)

{

while(i<=rear)

printf("\n%d",cqueue\_arr[i++]);

}

else

{

while(i<=max-1)

printf("%d",cqueue\_arr[i++]);

i=0;

while(i<=rear)

printf("%d",cqueue\_arr[i++]);

}

}

Conclusion:HENCE WE HAVE SEE CONCEPT OF CIRCULAR QUEUE WHICH HELP

TO SAVE MEMORY .