Navlakhi®

Circular Queues

Methodology and Program

By Abhishek Navlakhi

Semester 3: Data Structures

This document is for private circulation for the students of Navlakhi's.
More educational content can be found on **www.navlakhi.com** and **navlakhi.mobi**
Contact Numbers 9820246760/9769479368/9820009639/23548585/23868356

Program

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>

struct node
{
	int data;
	struct node *link;
};


struct queue
{
	int count;
	struct node *front;
	struct node *rear;
}*q;

void destroyQueue( )
{
	struct node *pLoc;
	while (q->count!=0)
	{
		pLoc=q->front;
		q->front=q->front->link; /* OR q->front=pLoc->link */
		q->count-=1;
		free(pLoc);
	}
	free(q);
}
```

```
int enqueue(int dataIn)
{
	struct node *pNew;

	pNew=(struct node *)malloc(sizeof(struct node));

	if (pNew!=NULL)
	{
		pNew->data=dataIn;
		//pNew->link=NULL;
		if (q->count==0)
			q->front=pNew;
		else
			q->rear->link=pNew;

		pNew->link=q->front;
		q->rear=pNew;
		q->count+=1;
		return 1;
	}
	else
		return 0;
}
```

```
int dequeue(int *dataPtr)
{
	struct node *pLoc;
	if (q->count==0) return 0;

	pLoc=q->front;
	*dataPtr=pLoc->data; /****OR *dataPtr=q->front->data	***/

	if (q->count==1)
	{
		q->rear=NULL;
		q->front=NULL;
	}
	else
	{
		q->front=q->front->link; /*OR q->front=pLoc->link */
		q->rear->link=q->front;
	}

	q->count-=1;
	free(pLoc);
	return 1;
}
```

```
int queueFront(int *dataPtr)
{
	if (q->count==0) return 0;

	*dataPtr=q->front->data;
	return 1;
}

int queueRear(int *dataPtr)
{
	if (q->count==0) return 0;

	*dataPtr=q->rear->data;
	return 1;
}

int fullQueue( )
{
	struct node *temp;
	temp=(struct node*)malloc(sizeof(struct node*));
	if (temp==NULL)	return 1;
	else
	{
		free(temp);
		return 0;
	}
}
```

```
int menu( )
{
int choice;
printf("\n\n\t\t M E N U \n");
printf("\n\n\t\t ========\n");
printf("1. Add Data To Circular Queue (Enqueue)\n");
printf("2. Remove Data From Circular Queue (Dequeue)\n");
printf("3. View Front element\n");
printf("4. View Rear element\n");
printf("5. View Count\n");
printf("6. Check if Memory FULL\n");
printf("7. Exit\n");

printf("\n Feed in your choice: ");
scanf("%d",&choice);

return choice;
}


void  createQueue( )
{
      q =(struct queue *)malloc(sizeof(struct queue));
      if (q ==NULL)
      {
            printf("Insufficient memory.....\n");
            exit(1);
      }
      q ->front= NULL;
      q ->rear=NULL;
      q ->count=0;
}
```

```
void main( )
{
int choice,dataIn,dataOut,success;
createQueue( );
do
{
choice=menu( );
switch(choice)
{
case 1:     printf("feed in Data to insert: ");
            scanf("%d",&dataIn);
            success=enqueue(dataIn);
            if(success) printf("Data inserted successfully\n");
            else printf("data Insertion Failed.. Insufficient memory..\n");
            break;
case 2:     success=dequeue(&dataOut);
            if (success) printf("Data successfully Deleted = %d\n",dataOut);
            else printf("Data Not present\n");
            break;
case 3: success=queueFront(&dataOut);
            if (success) printf("Data at the front of queue= %d\n",dataOut);
            else printf("No data in queue\n");
            break;
case 4: success=queueRear(&dataOut);
            if (success) printf("Data at the front of queue= %d\n",dataOut);
            else printf("No data in queue\n");
            break;
case 5: printf("Number of Data Items in Queue = %d\n",q->count);
            break;
case 6:     success=fullQueue();
            if (success) printf("Memory FULL !!!! ");
            else printf("Memory not FULL...\n");
            break;
}
}while(choice!=7);
destroyQueue( );
}
```

HOME OF EDUCATION

Navlakhi®

www.navlakhi.com
Home of Education

DATA Structures@ Navlakhi's