06

Queue Data Structure

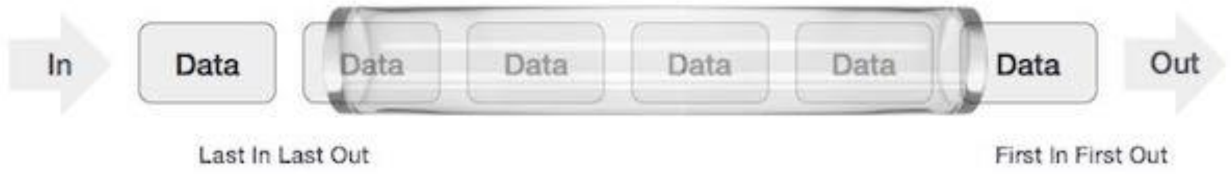
DATA STRUCTURE LAB

SESSION - 06

CSE135 | DAFFODIL INTERNATIONAL UNIVERSITY

Queue:

- Linear data structure
- First in first out (FIFO/LILO)
- Priority queue



Queue

Basic Operations:

enque() : add (store)/ insert an item to the queue

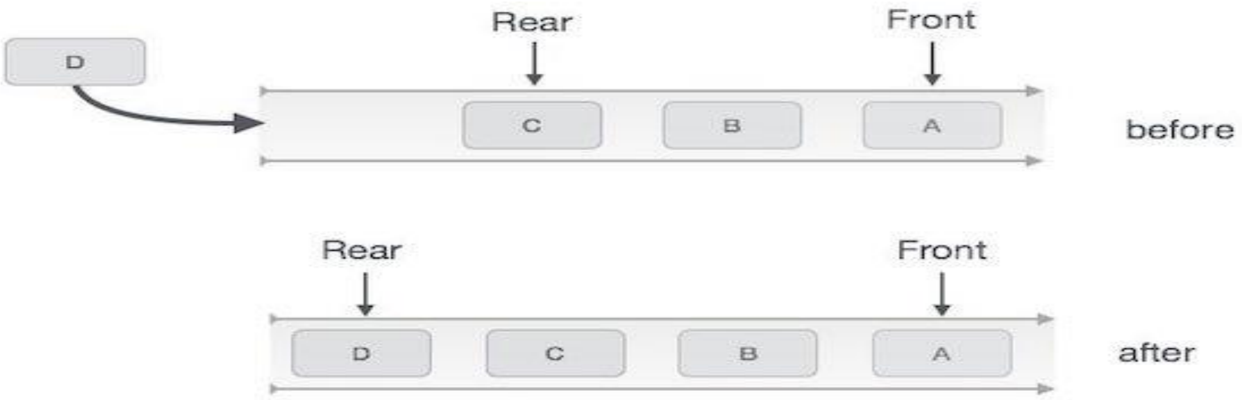
dequeuer() : remove an item from the queue

Enque Operations:

Step – 1: Check if queue is full

Step – 2: If queue is full procedure overflow error

Step –3: If queue is not full create a temp node and assign node at last



Queue Enqueue

Enque Implementation:

```
void enqueue(int data)
{
        if(rear==full)
        {
                  printf("Queue is full!\n");
                  return;
        }
        queue[rear]=data;
                rear++;
        printf("%d is enqueue!\n",data);
}
```

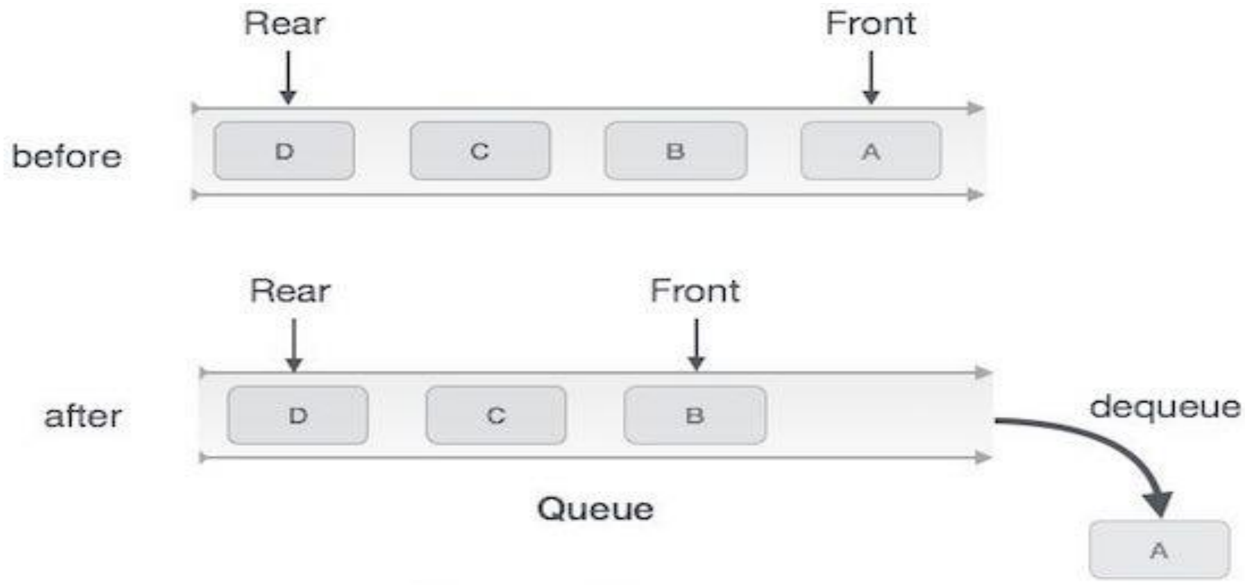
Dequeue Operation:

Step 1 – Check if the queue is empty.

Step 2 – If empty then produce underflow

Step 3 – If not then access the data where front is pointing.

Step 4 – Increment front pointer to point to the next available data element.



Queue Dequeue

Dequeue Implementation:

```
int dequeue()
{
        int temp;
        if(front==rear)
        {
                        printf("Queue is empty!\n");
                        return;
        }
        int data=queue[front];
        front++;
        return data;
}
```

Queue Implementation Using Linklist:

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
        int data;
        struct Node *next;
};
typedef struct Node queue;
void enque(queue *q,int data)
{
        queue *temp;
        temp=(queue*)malloc(sizeof(queue));
        temp->data=data;
        temp->next=NULL;
        while(q->next!=NULL)
        {
                       q=q->next;
        }
        q->next=temp;
        printf("%d is enqueued!\n",data);
}
int dequeue(queue *q)
{
        queue *temp;
        int data;
        if(q->next==NULL)
        {
                       printf("Queue is empty!\n");
                       return -1;
        }
        temp=q->next;
        data=temp->data;
        q->next=temp->next;
        free(temp);
        return data;
}
int main()
{
        queue *q;
        q=(queue*)malloc(sizeof(queue));
        q->next=NULL;
        enque(q,10);
        enque(q,5);
        enque(q,11);
        printf("%d\n",dequeue(q));

        printf("%d\n",dequeue(q));

        printf("%d\n",dequeue(q));

        return 0;

}
```

#EXERCISE:

QUEUE:

1.Convert the following infix expression to prefix and postfix expression:

(a) (5+6)*(6-4)/(8+2)
(b) A*B+C-D/E-F*G-H

3. Do enqueue 3,8,9,5,13,7 respectively in an empty queue. Now dequeue 9 then print the present queue. Now insert 1 and print the present queue. Now enqueue 11 and 20 and count total number of items in the present queue then do summation of them and print.

3.Write a C program to create a Queue data structure. This Queue data structure is to store the integer values. Your program should display a menu of choices to operate the Queue data structure. The menu given below.

a. Add items
b. Delete items
c. Show the number of items
d. Show min and max items
e. Find an item
f. Print all items
g. Exit