

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #define que_size 3
4  int item,front=0,rear=-1,q[que_size],
   count=0;
5  void insertrear()
6  {
7      if(count==que_size)
8      {
9          printf("queue overflow");
10         return;
11     }
12     rear=(rear+1)%que_size;
13     q[rear]=item;
14     count++;
15 }
16 int deletefront()
17 {
18     if(count==0) return -1;
19     item = q[front];
20     front=(front+1)%que_size;
21     count=count-1;
22     return item;
23 }
24 void displayq()
25 {
26     int i,f;
27     if(count==0)
28     {
29         printf("queue is empty");
30         return;
31     }
32     f=front;
33     printf("contents of queue \n");
34     for(i=0;i<count;i++)
35     {
36         printf("%d\n",q[f]);
37         f=(f+1)%que_size;
38     }
39 }
40 int main()
41 {

```

```
42     int choice;
43     for(;;)
44     {
45         printf("\n1.Insert rear \n2.Delete front
\n3.Display \n4.exit \n ");
46         printf("Enter the choice : ");
47         scanf("%d",&choice);
48         switch(choice)
49         {
50             case 1:printf("Enter the item to be
inserted :");
51                 scanf("%d",&item);
52                 insertrear();
53                 break;
54             case 2:item=deletefront();
55                 if(item==-1)
56                     printf("queue is empty\n");
57                 else
58                     printf("item deleted is %d \n",
item);
59                 break;
60             case 3:displayq();
61                 break;
62             default:exit(0);
63         }
64     }
65     return 0;
66 }
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 1
Enter the item to be inserted :10
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 1
Enter the item to be inserted :20
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 1
Enter the item to be inserted :30
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 1
Enter the item to be inserted :40
queue overflow
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 3
contents of queue
10
20
30
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 2
item deleted is 10
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 1
Enter the item to be inserted :40
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 3
contents of queue
20
30
40
```

```
1.Insert rear
2.Delete front
3.Display
4.exit
```

# CIRCULAR QUEUE

```
#include <stdio.h>
#include <stdlib.h>
#define que_size 3
int item, front = 0, rear = -1, a[que_size], count = 0;
void insert_rear()
{
    if (count == que_size)
    {
        printf("Queue Overflow\n");
        return;
    }
    rear = (rear + 1) % que_size;
    a[rear] = item;
    count++;
}
int delete_front()
{
    if (count == 0)
        return -1;
    item = a[front];
    front = (front + 1) % que_size;
    count--;
    return item;
}
void display()
{
    int i, f;
    if (count == 0)
    {
        printf("Queue is empty\n");
    }
}
```

```

return;
}
if = front;
printf("contents of Queue \n");
for (i=0; i<=count; i++)
{
    printf("%d * \n", q[i]);
    f=(f+1)%q;
}
}
}

```

```

int main()
{

```

```

    int choice;

```

```

    for(;;)
    {

```

```

        printf("\n 1. Insert rear \n 2. Delete front \n 3. Display \n 4. Exit \n");

```

```

        printf("Enter choice");

```

```

        scanf("%d", &choice);

```

```

        switch(choice)
        {

```

```

            case 1: printf("Enter the item to be inserted:");

```

```

                scanf("%d", &item);

```

```

                Insert = insert();

```

```

                break;

```

```

            case 2: item = delete-front();

```

```

                if (item == -1)

```

```

                    printf("Queue is empty \n");

```

```

                else

```

```

                    printf("item deleted = %d \n", item);

```

```

                break;

```

```

            case 3: display();

```



break;

~~for i:~~

default :: exit(0);

}

return 0;

}

```
1  #include<stdio.h>
2  #include<conio.h>
3  #define qsize 5
4  int f=0,r=-1,ch;
5  int item,q[10];
6
7  int isfull()
8  {
9      return(r==qsize-1)?1:0;
10 }
11 int isempty()
12 {
13     return(f>r)?1:0;
14 }
15 void insert_rear()
16 {
17     if(isfull())
18     {
19         printf("queue overflow\n");
20         return;
21     }
22     r=r+1;
23     q[r]=item;
24 }
25 void delete_front()
26 {
27     if(isempty())
28     {
29         printf("queue empty\n");
30         return;
31     }
32     printf("item deleted is %d\n",q[(f)++]);
33     if(f>r)
34     {
35         f=0;
36         r=-1;
```

```

37     }
38 }
39 void insert_front()
40 {
41     if(f!=0)
42     {
43         f=f-1;
44         q[f]=item;
45         return;
46     }
47     else if((f==0)&&(r==-1))
48     {
49         q[++(r)]=item;
50         return;
51     }
52     else
53         printf("insertion not possible\n");
54 }
55 void delete_rear()
56 {
57     if(isempty())
58     {
59         printf("queue is empty\n");
60         return;
61     }
62     printf("item deleted is %d\n",q[(r)--]);
63     if(f>r)
64     {
65         f=0;
66         r=-1;
67     }
68 }
69 void display()
70 {
71     int i;
72     if(isempty())

```



```

73     {
74         printf("queue empty\n");
75         return;
76     }
77     for(i=f;i<=r;i++)
78         printf("%d\n",q[i]);
79 }
80 int main()
81 {
82     for(;;)
83     {
84         printf("1.insert_rear\n2.insert_front\n3.
delete_rear\n4.delete_front\n5.display\n6.
exit\n");
85         printf("enter choice\n");
86         scanf("%d",&ch);
87         switch(ch)
88         {
89             case 1:printf("enter the item\n");
90                     scanf("%d",&item);
91                     insert_rear();|
92                     break;
93             case 2:printf("enter the item\n");
94                     scanf("%d",&item);
95                     insert_front();
96                     break;
97             case 3:delete_rear();
98                     break;
99             case 4:delete_front();
100                    break;
101             case 5:display();
102                    break;
103             default:_exit(0);
104         }
105     }
106     return 0;
107 }

```

```
4.delete_front
5.display
6.exit
enter choice
1
enter the item
30
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
5
10
20
30
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
```



TAB



```
enter choice
2
enter the item
10
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
2
enter the item
0
insertion not possible
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
5
10
20
3
4
5
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
3
item deleted is 5
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
3
item deleted is 4
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
3
item deleted is 3
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
```



TAB



```
enter the item
6
queue overflow
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
4
item deleted is 1
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
4
item deleted is 2
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
6
queue overflow
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
5
3
4
5
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
2
enter the item
20
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
```



TAB



```
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
1
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
2
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
3
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
4
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
5
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
```

# Double ENDED QUEUE

```
#include <stdio.h>
#include <process.h>
#define qsize 5
int f=0, r=-1, ch, item, q[qsize]
int isfull ()
{
    return (r==qsize-1)?1:0;
}
int isempty ()
{
    return (f>r)?1:0;
}
void insert_rear ()
{
    if (isfull ())
    {
        printf ("Queue Overflow \n");
        return;
    }
    r=r+1;
    q[r]=item;
}
void delete_front ()
{
    if (isempty ())
    {
        printf ("Queue Empty \n");
        return;
    }
    printf ("Item deleted is %d \n", q[f++]);
}
```



if (f > r)

f = 0;

r = -1;

}

}

void insert\_front()

{

if (f != 0)

{

f = f - 1;

arr[f] = item;

return;

}

else if ((f == 0) && (r == -1))

{

arr[++r] = item;

return;

}

else

printf("insertion not possible\n");

}

void delete\_rear()

{

if (isEmpty())

{

printf("Queue is empty\n");

return;

printf("item deleted is %d\n", arr[r--]);

if (f > r)

{

f = 0;

r = -1;

}

```
void display ()
```

```
{
```

```
int i;
```

```
if (isEmpty())
```

```
{
```

```
printf("Queue empty \n");
```

```
return;
```

```
}
```

```
for (i = 1; i <= n; i++)
```

```
printf("%d \n", q[i]);
```

```
}
```

```
void main ()
```

```
{
```

```
clrscr();
```

```
{
```

```
printf("1. Insert rear \n 2. Insert front \n 3.  
Delete Rear \n 4. Delete Front \n 5. Display \n  
6. Exit \n);
```

```
printf("Enter choice \n);
```

```
scanf("%d", &ch);
```

```
switch (ch)
```

```
{
```

```
case 1: printf("Enter the item \n);
```

```
scanf("%d", &item);
```

```
insert_rear();
```

```
break;
```

```
case 2: printf("Enter the item \n);
```

```
scanf("%d", &item);
```

```
insert_front();
```

```
break;
```

```
case 3: delete_rear();
```

```
break;
```

case 4: delete front ()

break;

case 5: display ()

break;

default: exit(0);

}

}

return 0;

}