

04/11/25

## Linear Queue

### Pseudo Code

# define N100

front = -1;

rear = -1;

Queue [N];

(if (front < (N-1) and rear < N)) {  
    void enqueue (x);  
        if (front = N-1)

    {  
        if (front = rear = N-1)

        {  
            point ("Overflow")

        if (front == -1)

            front = rear = 0

            Queue [rear] = x;

    else

        rear ++;

        Queue [rear] = x;

    void dequeue();

    if (front = rear = 0)

        point ("Underflow")

    else  
        point ("Element x is deleted", Queue [front])  
        front ++

```
void display();
```

```
for (int i=front; i <= rear; i++) {
    cout << "d" << queue[i];
}
```

```
int main () {
```

```
int x; int choice; int choice;
print("1. insert 2. deletion 3. display 4. exit");
scanf("ird", &x) : (x) supers
```

(20) ~~1-1~~ ~~re~~ ~~for~~ ~~on~~ ~~break~~ } fi  
(break = ~~while~~ { ~~for~~ ~~on~~ ~~break~~ }) fi

(*cont*) ~~read~~ *input* ("Enter value");  
~~print~~ *print* ("%.d", *val*);  
*scanf* ("%d", *val*);  
(*cont*) *switch* (*x*):

$O = \text{w3re}, O = \text{bavc}$  Case 1:  
 $\mathcal{L} = [\text{w3re}, \text{bavc}]$   $\text{scat}(\underline{\quad})$   
 $\text{enqe}(\underline{n})$

Case 28.

h.t. (rose) = rose deque()

Mr [unclear] would answer Page 3:

display()

Case 4: ( ) <sup>sup3b</sup>  
C <sub>t</sub>

point ("exit"):

(wolfsbach) } do (xe! = 4);

33 : (more = how) 71 56

## Circular Queue

$\{ (i : \text{int} \rightarrow i : \text{bool} = \text{bit}) \text{ so} \}$   
 $\{ [\text{int}] \text{ and } [\text{bit}] \} \text{ true}$

## Pseudo Code :-

#define N 100

Circular Queue[N]

if (true) {  
    ~~if~~ front = -1, rear = -1;

    enqueue(x) :

        if (front == 0 and rear == N-1) (or)  
            ~~if~~ (sliding rear + 1 = front)

            Circular Queue[rear] = x;  
            point ("Overflow")  
            ~~if~~ (true) front = rear

        else if (front == -1)

~~if~~ (true) rear = 0, front = 0

            Circular Queue[rear] = x

        else

~~if~~ (true) rear = (rear + 1) % N

            Circular Queue[rear] = x

    dequeue() :

        if (front == -1) :

~~if~~ (true) point ("Underflow")

        else if (front == rear) :

            point ("Deleted %d", Circular Queue[rear])

            front = rear = -1

else : point ("Deleted v.d", CircularQueue[front])

$$front = (front + 1) \times N$$

insert(v)

( $\leftarrow$  ob)

Display () :

< v. obfz > abulm #

if ( front <= rear ) {  
v. obfz > abulm #

for ( int i = front ; i <= rear ; i++ )  
v. obfz > abulm #  
point (CircularQueue[i])  
i[n] suvD b;

else { } (to thi) suvD bioV

(v. obfz > abulm #) suvD  
for ( int i = front ; i < N ; i++ )  
point (CircularQueue[i])

} (i = rear & front == 0) suvD  
for ( int i = 0 ; i <= rear ; i++ )  
point (CircularQueue[i])  
0 = rear = front

} (i = [rear] suvD)

{

{ ob

x = [front] suvD

{

{

} (v. obfz > bioV

} (i = rear & i == front) ; ;

(v. obfz > abulm #) suvD

{

Hand  
04/11

## Linear Queue

Code =>

```
#include <stdio.h> : () function()
#include <ctype.h> (main() {
#define N 5
int front = -1, rear = -1;
int Queue[N];
void enqueue (int x) {
    if (rear == N-1) {
        printf ("Queue Overflow\n");
    } else if (front == -1 && rear == -1) {
        front = rear = 0;
        Queue[rear] = x;
    } else {
        Queue[++rear] = x;
    }
}
void dequeue () {
    if (front == -1 && rear == -1) {
        printf ("Queue Underflow\n");
    }
}
```

```

else if (front == rear) {
    printf("Deleted element is %d\n", Queue[front]);
    front = rear = -1;
}
else {
    printf("Deleted element is %d\n", Queue[front++]);
    front++;
}
}

```

```

void display() {
    printf("Elements in the Queue are : ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", Queue[i]);
    }
    printf("\n");
}

```

```

int main() {
    int c;
    printf("1. Insert\n 2. Delete\n 3. Display\n 4. Exit\n");
    do {
        printf("Choose an operation : ");
        scanf("%d", &c);
        switch (c) {

```

Case 1:

```

    {
        int x;
        printf("Enter value to be inserted : ");
        scanf("%d", &x);
        enqueue(x);
        break;
    }
}

```

Case 2: {front} is null

else  
if (front == null) break;  
else  
{  
 cout << front;  
 front = front->next;  
 cout << endl;  
}  
break;

Case 3: { } is null

else  
if (front == null) display();  
else  
{  
 cout << front->data;  
 front = front->next;  
 cout << endl;  
}  
break;

Case 4: { } is not null

else  
if (front == null) point("Exiting");  
else  
{  
 cout << front->data;  
 front = front->next;  
 cout << endl;  
}  
break;

default:  
point("choose correct operation");

} else  
{  
 cout << "Enter operation (1-4)";  
}  
}  
while (c != 4);

Output:-  
1. Insert  
2. Deletion  
3. Display  
4. Exit

Enter a operation 1

Enter value to be inserted 12

Enter a operation 1

Enter value to be inserted 2

Enter operation 3

Element in queue are 1 2

Enter operation 2

Deleted element is 1

## Circular Queue

Code  $\Rightarrow$  #include <stdio.h>

#include <ctype.h>

#define N 5

int front = -1, rear = -1;

int Queue[N];

Void enqueue(int x) {

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

printf("Queue Overflow\n");

else if (front == -1 & & rear == -1) {

front = rear = 0;

Queue[rear] = x;

{ (i = 0; i < N - 1; i++) {

else {

rear = (rear + 1) % N;

Queue[rear] = x;

{ (i = 0; i < N - 1; i++) {

{ (i = 0; i < N - 1; i++) {

Void dequeue() {

if (front == -1 & & rear == -1) {

printf("Queue Underflow\n");

{ (i = 0; i < N - 1; i++) {

else if (front == rear) {

printf("Deleted element is %d\n", Queue[front]);

front = rear = -1;

else {  
    printf("Deleted Element is %d", Queue[front]);  
    front = (front + 1) % N;  
}

y

2 in index 4

i = rear, l = front hi

Void display () {

    printf("Elements are : ");

    if (rear == front) {

        printf("Underflow error");

    } else for (int i = front; i < rear; i++) {

        printf("%d", Queue[i]);

    }

y

i = rear, l = front

else {

    for (int i = front; i < N; i++) {

        printf("%d", Queue[i]);

    }

    for (int i = 0; i < rear; i++) {

        printf("%d", Queue[i]);

    }

y

int main () {

    int c;

    printf(" 1. Insert 2. Delete 3. Display ");

```

do {
    printf ("Enter a operation:"); input . L
    scanf ("%d", &c); value . S
    switch (c) { logic . C
        Case 1: { fix . A
            if (x); valence is right
            printf ("Enter the value to be inserted:"); value with
            scanf ("%d", &val); valence is right
            enqueue (x); valence is right
            break; valence is right
        }
        Case 2: { value right
            dequeue(); valence is right
            if (break); break is right
        }
        Case 3: { value right
            display(); break is right
            if (break); break is right
        }
        Case 4: { value right
            printf ("Exiting"); fix
            break; fix
        }
    }
}

```

while (c!=4)

Y

Output:

1. Insert (inserting 0 into 3) Newq
2. Deletion (8, 3) New2
3. Display { (3) New2
4. Exit

Enter a operation 1

Enter value to be inserted 1

Enter a operation 1

Enter value to be inserted 2

Enter a operation 1

Enter value to be inserted 3

Enter a operation 2

Deleted element is 1

Enter a operation 2

Deleted element is 2

Enter a operation 3

Elements in Queue are 3

Enter a operation 4

Exiting.

End of it!!

(4-13) links {