

## DSA\_College\queue\_arrayImplement.cpp

```
1 // Queue - First In First Out
2 // Insertion at tail and Pop from head
3 /*
4 #include<iostream>
5 #include<queue>
6 using namespace std ;
7
8 int main()
9 {
10     // create a queue
11     // use Queue STL
12     queue<int> q ;
13     q.push(10);
14     q.push(20);
15     q.push(40) ;
16
17     cout<<"size of queue is = "<<q.size()<<endl;
18
19     q.pop() ;
20
21     cout<<"size of queue is = "<<q.size()<<endl;
22     cout<<"front of queue is = "<<q.front()<<endl;
23
24     if(q.empty())
25     {
26         cout<<"queue is empty"<<endl;
27     }
28     else{
29         cout<<"queue is not empty"<<endl;
30     }
31
32     return 0;
33 }
34 */
35
36
37
38
39
40 /*
41
42 // Queue Implementation using Arrays
43
44 class Queue
45 {
46     int* arr ;
47     int qfront ; // qfront and rear are initially at 0th index of arr
48     int rear ;
49     int size ;
50
51     public :
```

```
52
53 Queue() // constructor
54 {
55     size = 1000 ;
56     arr = new int(size) ;
57     qfront = 0;
58     rear = 0;
59 }
60
61 void enqueue(int data)    //O(1)
62 {
63     if(rear == size) // queue is full
64     {
65         cout<<"queue is full"<<endl;
66     }
67     else{
68         arr[rear] = data ;
69         rear++ ;
70     }
71 }
72
73 int dequeue()    // O(1)
74 {
75     if(qfront == rear) // empty queue
76     {
77         return -1 ;
78     }
79     else{
80         int ans = arr[qfront] ;
81
82         arr[qfront] = -1 ; // jo value delete ki hai uski jagah -1 krdo and front ko
aage badhao
83         qfront++ ;    // kyuki front se hi delete hua hai
84
85         if(qfront == rear)    // empty queue - return the qfront and rear variable back
to their original positions
86         {
87             qfront = 0;
88             rear = 0;
89         }
90         return ans ;
91     }
92 }
93
94 int front() // O(1)
95 {
96     if(qfront == rear)
97         return -1 ;
98
99     else{
100         return arr[qfront] ;
101     }
102 }
103
```

```
104     bool isEmpty()    // O(1)
105     {
106         return rear == qfront ;
107     }
108 };
109 */
110
111
112
113
114
115
116
117
118
119
120 /*
121 // Circular Queue
122 class circularQueue
123 {
124     int* arr ;
125     int front ;
126     int rear ;
127     int size ;
128
129     public :
130     circularQueue(int n)
131     {
132         size = n ;
133         arr = new int(size) ;
134         front = rear = -1 ; // front and rear initially array ke bahar hai, dono ko ek aage
135         badhane pr vo 0th index pr aajayenge
136     }
137
138     bool enqueue(int value)
139     {
140         if((front == 0 && rear == size -1) || (rear == (front-1)%(size-1)))
141         {
142             cout<<"queue is full"<<endl;
143             return false ;
144         }
145         else if(front == -1)
146         {
147             // first element to push
148             front = rear = 0;
149             arr[rear] = value ;
150         }
151         else if(rear == size-1 && front != 0)
152         {
153             rear = 0; // 0th index(front) khali hai and rear last mai hai, then ab push krne
154             pr
155             // rear 0th index pr aajayega
156             arr[rear] = value ;
157         }
158     }
159 }
```

```
156         else{
157             rear++;
158             arr[rear] = value ;
159         }
160         return true ;
161     }
162
163     int dequeue()
164     {
165         if(front == -1) // to check if queue is empty
166         {
167             cout<<"queue is empty"<<endl;
168             return -1 ;
169         }
170
171         int ans = arr[front] ;
172         arr[front] = -1 ;
173         if(front == rear)
174         {
175             // single element is present
176             front = rear = -1 ;
177         }
178         else if(front == size-1) // to maintain cyclic nature
179         {
180             front = 0 ;
181         }
182         else{ // normal flow
183             front++ ;
184         }
185         return ans ;
186     }
187 };
188 */
```