

PRN: 2020BTEIT00041

Double Ended Queue:

```

1  /*
2  |   PRN: 2020BTEIT00041
3  |   Name: Om Vivek Gharge
4  | */
5
6  /*
7  |   Double Ended Queue - Implementation using linked list
8  | */
9
10 #include <bits/stdc++.h>
11 using namespace std;
12
13 // Node class
14 class Node{
15 public:
16     int data;
17     Node *next;
18
19     Node(){
20         this->data = 0;
21         this->next = NULL;
22     }
23
24     Node(int data){
25         this->data = data;
26         this->next = NULL;
27     }
28 };
29
30 // Double Ended Queue class
31 class Deque{
32 public:
33     Node* front;
34     Node* rear;
35
36     Deque(){
37         this->front = NULL;
38         this->rear = NULL;
39     }
40
41     // Operations on Deque
42     void insertFront(int data);
43     void insertRear(int data);
44     void deleteFront();
45     void deleteRear();
46     int getFront();
47     int getRear();
48     void Display();

```

```

49     };
50
51     // Inserts element at front of Deque
52     void Deque::insertFront(int data){
53         // Create new node
54         Node *newNode = new Node(data);
55
56         // If Deque is empty
57         if(this->front == NULL){
58             this->front = newNode;
59             this->rear = newNode;
60         }
61         // If Deque is not empty
62         else{
63             // Make next of new node as front
64             newNode->next = this->front;
65
66             // Move front to point to new node
67             this->front = newNode;
68         }
69     }
70
71     // Inserts element at rear of Deque
72     void Deque::insertRear(int data){
73         // Create new node
74         Node *newNode = new Node(data);
75
76         // If Deque is empty
77         if(this->front == NULL){
78             this->front = newNode;
79             this->rear = newNode;
80         }
81         // If Deque is not empty
82         else{
83             // Make next of rear as new node
84             this->rear->next = newNode;
85
86             // Move rear to point to new node
87             this->rear = newNode;
88         }
89     }
90
91     // Deletes element from front of Deque
92     void Deque::deleteFront(){
93         // If Deque is empty
94         if(this->front == NULL){
95             cout << "Deque is empty" << endl;

```

```

96         return;
97     }
98     // If Deque has only one element
99     else if(this->front == this->rear){
100         delete this->front;
101         this->front = NULL;
102         this->rear = NULL;
103     }
104     // If Deque has more than one element
105     else{
106         // Store pointer to old front
107         Node *old_front = this->front;
108
109         // Move front to point to next of old front
110         this->front = this->front->next;
111
112         // Delete old front
113         delete old_front;
114     }
115 }
116
117 // Deletes element from rear of Deque
118 void Deque::deleteRear(){
119     // If Deque is empty
120     if(this->front == NULL){
121         cout << "Deque is empty" << endl;
122         return;
123     }
124     // If Deque has only one element
125     else if(this->front == this->rear){
126         delete this->front;
127         this->front = NULL;
128         this->rear = NULL;
129     }
130     // If Deque has more than one element
131     else{
132         // Store pointer to last element
133         Node *last = this->front;
134
135         // Find second last element
136         while(last->next != this->rear){
137             last = last->next;
138         }
139
140         // Delete last element
141         delete this->rear;

```

```

142
143     // Move rear to point to last element
144     this->rear = last;
145
146     // Change next of last element to NULL
147     this->rear->next = NULL;
148 }
149 }
150
151 // Returns element from front of Deque
152 int Deque::getFront(){
153     // If Deque is empty
154     if(this->front == NULL){
155         cout << "Deque is empty" << endl;
156         return -1;
157     }
158     // If Deque has only one element
159     else if(this->front == this->rear){
160         return this->front->data;
161     }
162     // If Deque has more than one element
163     else{
164         return this->front->data;
165     }
166 }
167
168 // Returns element from rear of Deque
169 int Deque::getRear(){
170     // If Deque is empty
171     if(this->front == NULL){
172         cout << "Deque is empty" << endl;
173         return -1;
174     }
175     // If Deque has only one element
176     else if(this->front == this->rear){
177         return this->front->data;
178     }
179     // If Deque has more than one element
180     else{
181         return this->rear->data;
182     }
183 }
184
185 // Displays Deque
186 void Deque::Display(){
187     // If Deque is empty
188     if(this->front == NULL){

```

```

189         cout << "Deque is empty" << endl;
190         return;
191     }
192     // If Deque has only one element
193     else if(this->front == this->rear){
194         cout << this->front->data << endl;
195     }
196     // If Deque has more than one element
197     else{
198         // Store pointer to last element
199         Node* first = this->front;
200         Node *last = this->front;
201
202         // Find second last element
203         while(last->next != this->rear){
204             last = last->next;
205         }
206
207         // Display elements from front to second last
208         while(first != last){
209             cout << first->data << " ";
210             first = first->next;
211         }
212
213         // Display last element
214         cout << this->rear->data << endl;
215
216         // Move front and rear to point to NULL
217         first = NULL;
218         last = NULL;
219     }
220 }
221
222 int main(){
223     Deque d;
224
225     // Menu driven program
226     int choice;
227
228     do{
229         cout<<"-----Menu-----\n";
230         cout << "1. Insert at front" << endl;
231         cout << "2. Insert at rear" << endl;
232         cout << "3. Delete from front" << endl;
233         cout << "4. Delete from rear" << endl;
234         cout << "5. Get front element" << endl;

```

```

235     cout << "6. Get rear element" << endl;
236     cout << "7. Display" << endl;
237     cout << "8. Exit" << endl;
238
239     cout << "Enter your choice: ";
240     cin >> choice;
241
242     switch(choice){
243     case 1:
244         int dataFront;
245         cout << "Enter data: ";
246         cin >> dataFront;
247         d.insertFront(dataFront);
248         break;
249     case 2:
250         int dataRear;
251         cout << "Enter data: ";
252         cin >> dataRear;
253         d.insertRear(dataRear);
254         break;
255     case 3:
256         d.deleteFront();
257         break;
258     case 4:
259         d.deleteRear();
260         break;
261     case 5:
262         cout << "Front element is: " << d.getFront() << endl;
263         break;
264     case 6:
265         cout << "Rear element is: " << d.getRear() << endl;
266         break;
267     case 7:
268         d.Display();
269     case 8:
270         cout<<"Exiting..."<<endl;
271         break;
272     default:
273         cout << "Wrong choice" << endl;
274     }
275     }while(choice != 8);
276
277     return 0;
278 }

```

## OUTPUT:

```
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 1
Enter data: 1
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 2
Enter data: 2
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 2
Enter data: 3
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 2
Enter data: 4
```



```
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 7
1 2 3 4
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 3
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 7
2 3 4
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 4
```

```
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 7
2 3
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 5
Front element is: 2
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 6
Rear element is: 3
-----Menu-----
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 8
Exiting...
05-5-2024 10:41:04 AM
```