

## SEM 3\Exp6\Queue\_ARR.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MAX 100 // Maximum size of the queue
5
6  // Queue structure using arrays
7  struct QueueArray
8  {
9      int front, rear;
10     int arr[MAX];
11 };
12
13 // Function to create a queue
14 struct QueueArray *createQueue()
15 {
16     struct QueueArray *queue = (struct QueueArray *)malloc(sizeof(struct
QueueArray));
17     queue->front = -1;
18     queue->rear = -1;
19     return queue;
20 }
21
22 // Check if the queue is full
23 int isFull(struct QueueArray *queue)
24 {
25     return queue->rear == MAX - 1;
26 }
27
28 // Check if the queue is empty
29 int isEmpty(struct QueueArray *queue)
30 {
31     return queue->front == -1 || queue->front > queue->rear;
32 }
33
34 // Enqueue an element into the queue
35 void enqueue(struct QueueArray *queue, int value)
36 {
37     if (isFull(queue))
38     {
39         printf("Queue overflow!\n");
40         return;
41     }
42     if (isEmpty(queue))
43     {
44         queue->front = 0; // Initialize front if queue was empty
45     }
46     queue->arr[++queue->rear] = value;
47     printf("%d enqueued to queue\n", value);
48 }
49
50 // Dequeue an element from the queue
51 int dequeue(struct QueueArray *queue)
```

```
52 {
53     if (isEmpty(queue))
54     {
55         printf("Queue underflow!\n");
56         return -1;
57     }
58     return queue->arr[queue->front++];
59 }
60
61 // Peek at the front element of the queue
62 int peek(struct QueueArray *queue)
63 {
64     if (isEmpty(queue))
65     {
66         printf("Queue is empty!\n");
67         return -1;
68     }
69     return queue->arr[queue->front];
70 }
71
72 // Display the queue
73 void display(struct QueueArray *queue)
74 {
75     if (isEmpty(queue))
76     {
77         printf("Queue is empty!\n");
78         return;
79     }
80     printf("Queue elements: ");
81     for (int i = queue->front; i <= queue->rear; i++)
82     {
83         printf("%d ", queue->arr[i]);
84     }
85     printf("\n");
86 }
87
88 int main()
89 {
90     struct QueueArray *queue = createQueue();
91     int choice, value;
92
93     printf("\nQueue Operations (Array Implementation):\n");
94     printf("1. Enqueue\n");
95     printf("2. Dequeue\n");
96     printf("3. Peek\n");
97     printf("4. Display\n");
98     printf("5. Exit\n");
99
100     while (1)
101     {
102
103         printf("Enter your choice: ");
104         scanf("%d", &choice);
105
```

```
106     switch (choice)
107     {
108     case 1:
109         printf("Enter the value to enqueue: ");
110         scanf("%d", &value);
111         enqueue(queue, value);
112         printf("\n");
113         break;
114     case 2:
115         value = dequeue(queue);
116         if (value != -1)
117             printf("Dequeued value: %d\n", value);
118
119         printf("\n");
120         break;
121     case 3:
122         value = peek(queue);
123         if (value != -1)
124             printf("Front value: %d\n", value);
125
126         printf("\n");
127         break;
128     case 4:
129         display(queue);
130         printf("\n");
131         break;
132     case 5:
133         free(queue);
134         exit(0);
135     default:
136         printf("Invalid choice!\n");
137     }
138 }
139
140 return 0;
141 }
142
```