

Folder SEM 3\Exp6

2 printable files

(file list disabled)

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; // Return -1 for underflow
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; // Return -1 if empty
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     while (1)
94     {
95         printf("\nQueue Operations (Array Implementation):\n");
96         printf("1. Enqueue\n");
97         printf("2. Dequeue\n");
98         printf("3. Peek\n");
```

```
99     printf("4. Display\n");
100    printf("5. Exit\n");
101    printf("Enter your choice: ");
102    scanf("%d", &choice);
103
104    switch (choice)
105    {
106    case 1:
107        printf("Enter the value to enqueue: ");
108        scanf("%d", &value);
109        enqueue(queue, value);
110        break;
111    case 2:
112        value = dequeue(queue);
113        if (value != -1)
114            printf("Dequeued value: %d\n", value);
115        break;
116    case 3:
117        value = peek(queue);
118        if (value != -1)
119            printf("Front value: %d\n", value);
120        break;
121    case 4:
122        display(queue);
123        break;
124    case 5:
125        free(queue);
126        exit(0);
127    default:
128        printf("Invalid choice!\n");
129    }
130 }
131
132 return 0;
133 }
134
```

SEM 3\Exp6\Queue_LL.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Node structure for the linked list
5  struct Node
6  {
7      int data;
8      struct Node *next;
9  };
10
11 // Queue structure using linked lists
12 struct QueueLinkedList
13 {
14     struct Node *front;
15     struct Node *rear;
```

```
16 };
17
18 // Function to create a queue
19 struct QueueLinkedList *createQueue()
20 {
21     struct QueueLinkedList *queue = (struct QueueLinkedList *)malloc(sizeof(struct
QueueLinkedList));
22     queue→front = queue→rear = NULL; // Initialize front and rear
23     return queue;
24 }
25
26 // Check if the queue is empty
27 int isEmpty(struct QueueLinkedList *queue)
28 {
29     return queue→front == NULL;
30 }
31
32 // Enqueue an element into the queue
33 void enqueue(struct QueueLinkedList *queue, int value)
34 {
35     struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
36     new_node→data = value;
37     new_node→next = NULL;
38
39     if (isEmpty(queue))
40     {
41         queue→front = queue→rear = new_node; // First node
42         printf("%d enqueued to queue\n", value);
43         return;
44     }
45
46     queue→rear→next = new_node; // Add new node at the end
47     queue→rear = new_node;      // Update the rear pointer
48     printf("%d enqueued to queue\n", value);
49 }
50
51 // Dequeue an element from the queue
52 int dequeue(struct QueueLinkedList *queue)
53 {
54     if (isEmpty(queue))
55     {
56         printf("Queue underflow!\n");
57         return -1; // Return -1 for underflow
58     }
59     struct Node *temp = queue→front;
60     int dequeued_value = temp→data;
61     queue→front = queue→front→next;
62
63     // If the front becomes NULL, set rear to NULL as well
64     if (queue→front == NULL)
65     {
66         queue→rear = NULL;
67     }
68 }
```

```
69     free(temp);
70     return dequeued_value;
71 }
72
73 // Peek at the front element of the queue
74 int peek(struct QueueLinkedList *queue)
75 {
76     if (isEmpty(queue))
77     {
78         printf("Queue is empty!\n");
79         return -1; // Return -1 if empty
80     }
81     return queue->front->data;
82 }
83
84 // Display the queue
85 void display(struct QueueLinkedList *queue)
86 {
87     if (isEmpty(queue))
88     {
89         printf("Queue is empty!\n");
90         return;
91     }
92     struct Node *temp = queue->front;
93     printf("Queue elements: ");
94     while (temp != NULL)
95     {
96         printf("%d ", temp->data);
97         temp = temp->next;
98     }
99     printf("\n");
100 }
101
102 int main()
103 {
104     struct QueueLinkedList *queue = createQueue();
105     int choice, value;
106
107     while (1)
108     {
109         printf("\nQueue Operations (Linked List Implementation):\n");
110         printf("1. Enqueue\n");
111         printf("2. Dequeue\n");
112         printf("3. Peek\n");
113         printf("4. Display\n");
114         printf("5. Exit\n");
115         printf("Enter your choice: ");
116         scanf("%d", &choice);
117
118         switch (choice)
119         {
120             case 1:
121                 printf("Enter the value to enqueue: ");
122                 scanf("%d", &value);
```

```
123         enqueue(queue, value);
124         break;
125     case 2:
126         value = dequeue(queue);
127         if (value != -1)
128             printf("Dequeued value: %d\n", value);
129         break;
130     case 3:
131         value = peek(queue);
132         if (value != -1)
133             printf("Front value: %d\n", value);
134         break;
135     case 4:
136         display(queue);
137         break;
138     case 5:
139         // Free linked list nodes (cleanup)
140         while (!isEmpty(queue))
141         {
142             dequeue(queue);
143         }
144         free(queue);
145         exit(0);
146     default:
147         printf("Invalid choice!\n");
148     }
149 }
150
151 return 0;
152 }
153
```