

Experiment 5 Lab 1: Queue Implementation Using Arrays

main.c	Output
<pre>1 #include <stdio.h> 2 #include <stdlib.h> 3 4 #define MAX 5 // Define the maximum size of the queue 5 6 // Queue structure 7 struct Queue { 8 int arr[MAX]; 9 int front; 10 int rear; 11 }; 12 13 // Function prototypes 14 void enqueue(struct Queue *queue, int value); 15 int dequeue(struct Queue *queue); 16 int peek(struct Queue *queue); 17 int isEmpty(struct Queue *queue); 18 int isFull(struct Queue *queue); 19 20 int main() { 21 struct Queue queue; 22 queue.front = -1; // Initialize front and rear 23 queue.rear = -1; 24 25 int choice, value; 26 27 while (1) { 28 // Menu 29 printf("\nQueue Operations:\n"); 30 printf("1. Enqueue\n"); 31 printf("2. Dequeue\n"); 32 printf("3. Peek\n"); 33 printf("4. Check if Queue is Empty\n"); 34 printf("5. Check if Queue is Full\n"); 35 printf("6. Exit\n"); 36 printf("Enter your choice: "); 37 scanf("%d", &choice); 38 39 switch (choice) { 40 case 1: 41 // Enqueue operation 42 printf("Enter the value to enqueue: "); 43 scanf("%d", &value); 44 enqueue(&queue, value); 45 break; 46 47 case 2: 48 // Dequeue operation 49 value = dequeue(&queue); 50 if (value != -1) { 51 printf("Dequeued value: %d\n", value);</pre>	<pre>/tmp/iYXJ95s5QT.o Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice: 3 Queue is empty. Nothing to peek. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice: 4 The queue is empty. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice: 5 The queue is not full. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice:</pre>
<pre>50- if (value != -1) { 51- printf("Dequeued value: %d\n", value); 52- } 53- break; 54- 55- case 3: 56- // Peek operation 57- value = peek(&queue); 58- if (value != -1) { 59- printf("Front value: %d\n", value); 60- } 61- break; 62- 63- case 4: 64- // Check if queue is empty 65- if (isEmpty(&queue)) { 66- printf("The queue is empty.\n"); 67- } else { 68- printf("The queue is not empty.\n"); 69- } 70- break; 71- 72- case 5: 73- // Check if queue is full 74- if (isFull(&queue)) { 75- printf("The queue is full.\n"); 76- } else { 77- printf("The queue is not full.\n"); 78- } 79- break; 80- 81- case 6: 82- // Exit program 83- printf("Exiting...\n"); 84- exit(0); 85- 86- default: 87- printf("Invalid choice! Please try again.\n"); 88- } 89- } 90- 91- return 0; 92- } 93- 94- // Function to enqueue an element to the rear of the queue 95- void enqueue(struct Queue *queue, int value) { 96- if (isFull(queue)) { 97- printf("Queue Overflow! Cannot enqueue %d.\n", value); 98- } else { 99- if (queue->front == -1) {</pre>	<pre>/tmp/iYXJ95s5QT.o Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice: 3 Queue is empty. Nothing to peek. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice: 4 The queue is empty. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice: 5 The queue is not full. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Check if Queue is Full 6. Exit Enter your choice:</pre>

main.c

Share

Run

```
95 - void enqueue(struct Queue *queue, int value) {
96 -     if (isFull(queue)) {
97 -         printf("Queue Overflow! Cannot enqueue %d.\n", value);
98 -     } else {
99 -         if (queue->front == -1) {
100 -             queue->front = 0; // Set front to 0 if inserting the first element
101 -         }
102 -         queue->rear++;
103 -         queue->arr[queue->rear] = value;
104 -         printf("Enqueued %d to the queue.\n", value);
105 -     }
106 - }
107
108 // Function to dequeue an element from the front of the queue
109 int dequeue(struct Queue *queue) {
110 -     if (isEmpty(queue)) {
111 -         printf("Queue Underflow! Cannot dequeue.\n");
112 -         return -1;
113 -     } else {
114 -         int value = queue->arr[queue->front];
115 -         if (queue->front == queue->rear) {
116 -             // If the queue becomes empty after this operation
117 -             queue->front = -1;
118 -             queue->rear = -1;
119 -         } else {
120 -             queue->front++;
121 -         }
122 -         return value;
123 -     }
124 - }
125
126 // Function to peek at the front element without removing it
127 int peek(struct Queue *queue) {
128 -     if (isEmpty(queue)) {
129 -         printf("Queue is empty. Nothing to peek.\n");
130 -         return -1;
131 -     } else {
132 -         return queue->arr[queue->front];
133 -     }
134 - }
135
136 // Function to check if the queue is empty
137 int isEmpty(struct Queue *queue) {
138 -     return queue->front == -1;
139 - }
140
141 // Function to check if the queue is full
142 int isFull(struct Queue *queue) {
143 -     return queue->rear == MAX - 1;
144 - }
```

Output

/tmp/iYXJ9Ss5QT.o

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 3
Queue is empty. Nothing to peek.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 4
The queue is empty.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 5
The queue is not full.

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: |

Experiment 5 Lab 2: Queue Implementation Using Linked Lists

main.c	Output
<pre>1 #include <stdio.h> 2 #include <stdlib.h> 3 4 // Define a node structure for the linked list 5 struct Node { 6 int data; 7 struct Node* next; 8 }; 9 10 // Queue structure with front and rear pointers 11 struct Queue { 12 struct Node* front; 13 struct Node* rear; 14 }; 15 16 // Function prototypes 17 void enqueue(struct Queue* q, int value); 18 int dequeue(struct Queue* q); 19 int peek(struct Queue* q); 20 int isEmpty(struct Queue* q); 21 22 int main() { 23 struct Queue q; 24 q.front = q.rear = NULL; // Initialize front and rear 25 26 int choice, value; 27 28 while (1) { 29 // Menu 30 printf("\nQueue Operations:\n"); 31 printf("1. Enqueue\n"); 32 printf("2. Dequeue\n"); 33 printf("3. Peek\n"); 34 printf("4. Check if Queue is Empty\n"); 35 printf("5. Exit\n"); 36 printf("Enter your choice: "); 37 scanf("%d", &choice); 38 39 switch (choice) { 40 case 1: 41 // Enqueue operation 42 printf("Enter the value to enqueue: "); 43 scanf("%d", &value); 44 enqueue(&q, value); 45 break; 46 47 case 2: 48 // Dequeue operation 49 value = dequeue(&q); 50 if (value != -1) { 51 printf("Dequeued value: %d\n", value);</pre>	<pre>/tmp/wbfxmpl11.o Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Exit Enter your choice: 3 Queue is empty. Nothing to peek. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Exit Enter your choice: 4 The queue is empty. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Exit Enter your choice: </pre>
<pre>50 51 if (value != -1) { 52 printf("Dequeued value: %d\n", value); 53 } 54 break; 55 56 case 3: 57 // Peek operation 58 value = peek(&q); 59 if (value != -1) { 60 printf("Front value: %d\n", value); 61 } 62 break; 63 64 case 4: 65 // Check if queue is empty 66 if (isEmpty(&q)) { 67 printf("The queue is empty.\n"); 68 } else { 69 printf("The queue is not empty.\n"); 70 } 71 break; 72 73 case 5: 74 // Exit program 75 printf("Exiting...\n"); 76 exit(0); 77 78 default: 79 printf("Invalid choice! Please try again.\n"); 80 } 81 82 return 0; 83 } 84 85 // Function to enqueue an element to the rear of the queue 86 void enqueue(struct Queue* q, int value) { 87 struct Node* newNode = (struct Node*)malloc(sizeof(struct Node)); 88 if (!newNode) { 89 printf("Memory allocation error. Unable to enqueue %d.\n", value); 90 return; 91 } 92 newNode->data = value; 93 newNode->next = NULL; 94 if (q->rear == NULL) { 95 q->front = q->rear = newNode; 96 } else { 97 q->rear->next = newNode; 98 q->rear = newNode; 99 } 100 }</pre>	<pre>/tmp/wbfxmpl11.o Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Exit Enter your choice: 3 Queue is empty. Nothing to peek. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Exit Enter your choice: 4 The queue is empty. Queue Operations: 1. Enqueue 2. Dequeue 3. Peek 4. Check if Queue is Empty 5. Exit Enter your choice: </pre>

main.c



Share

Run

Output

```
04
85 // Function to enqueue an element to the rear of the queue
86 void enqueue(struct Queue* q, int value) {
87     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
88     if (!newNode) {
89         printf("Memory allocation error. Unable to enqueue %d.\n", value);
90         return;
91     }
92     newNode->data = value;
93     newNode->next = NULL;
94     if (q->rear == NULL) {
95         q->front = q->rear = newNode;
96     } else {
97         q->rear->next = newNode;
98         q->rear = newNode;
99     }
100     printf("Enqueued %d to the queue.\n", value);
101 }
102
103 // Function to dequeue an element from the front of the queue
104 int dequeue(struct Queue* q) {
105     if (isEmpty(q)) {
106         printf("Queue Underflow! Cannot dequeue.\n");
107         return -1;
108     }
109     struct Node* temp = q->front;
110     int dequeuedValue = temp->data;
111     q->front = q->front->next;
112
113     if (q->front == NULL) {
114         q->rear = NULL; // If queue becomes empty, set rear to NULL
115     }
116     free(temp);
117     return dequeuedValue;
118 }
119
120 // Function to peek at the front element without removing it
121 int peek(struct Queue* q) {
122     if (isEmpty(q)) {
123         printf("Queue is empty. Nothing to peek.\n");
124         return -1;
125     } else {
126         return q->front->data;
127     }
128 }
129
130 // Function to check if the queue is empty
131 int isEmpty(struct Queue* q) {
132     return q->front == NULL;
133 }
```

/tmp/wbfxmpl11.o

Queue Operations:

1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Exit

Enter your choice: 3

Queue is empty. Nothing to peek.

Queue Operations:

1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Exit

Enter your choice: 4

The queue is empty.

Queue Operations:

1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Exit

Enter your choice: |