

# 1. Write a c program for Implementation of array in Queue.

Programiz  
C Online Compiler

Google Ads ...with Rs.20,000 spend. Get Started Programiz PRO

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #define MAX 100
5 typedef struct {
6     int front, rear, size;
7     int capacity;
8     int* array;
9 } Queue;
10 Queue* createQueue(int capacity) {
11     Queue* queue = (Queue*)malloc(sizeof(Queue));
12     queue->capacity = capacity;
13     queue->front = 0;
14     queue->rear = capacity - 1;
15     queue->size = 0;
16     queue->array = (int*)malloc(queue->capacity * sizeof(int));
17     return queue;
18 }
19 bool isEmpty(Queue* queue) {
20     return (queue->size == 0);
21 }
22 bool isFull(Queue* queue) {
23     return (queue->size == queue->capacity);
24 }
25 void enqueue(Queue* queue, int item) {
26     if (isFull(queue)) {
27         printf("Queue is full!\n");
28         return;
29     }
30     queue->rear = (queue->rear + 1) % queue->capacity;
31     queue->array[queue->rear] = item;
32     queue->size++;
33 }
34 int dequeue(Queue* queue) {
35     if (isEmpty(queue)) {
```

Output

```
/tmp/S4bGNThvW8.o
Dequeued: 10
Front item is: 20

=== Code Execution Successful ===
```

Programiz  
C Online Compiler

Google Ads ...with Rs.20,000 spend. Get Started Programiz PRO


```
main.c
32 queue->size++;
33 }
34 int dequeue(Queue* queue) {
35     if (isEmpty(queue)) {
36         printf("Queue is empty!\n");
37         return -1;
38     }
39     int item = queue->array[queue->front];
40     queue->front = (queue->front + 1) % queue->capacity;
41     queue->size--;
42     return item;
43 }
44 int peek(Queue* queue) {
45     if (isEmpty(queue)) {
46         printf("Queue is empty!\n");
47         return -1;
48     }
49     return queue->array[queue->front];
50 }
51 void freeQueue(Queue* queue) {
52     free(queue->array);
53     free(queue);
54 }
55 int main() {
56     Queue* queue = createQueue(MAX);
57     enqueue(queue, 10);
58     enqueue(queue, 20);
59     enqueue(queue, 30);
60     printf("Dequeued: %d\n", dequeue(queue));
61     printf("Front item is: %d\n", peek(queue));
62     freeQueue(queue);
63     return 0;
64 }
65 }
```

Output


```
/tmp/S4bGNThvW8.o
Dequeued: 10
Front item is: 20

=== Code Execution Successful ===
```

## 2. Write a c program for Implementation of Linked List in Queue.

  
C Online Compiler

Premium Coding  
Courses by Programiz

  
Programiz PRO

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 typedef struct Node {
5     int data;
6     struct Node* next;
7 } Node;
8 typedef struct Queue {
9     Node* front;
10    Node* rear;
11    int size;
12 } Queue;
13 Node* createNode(int data) {
14     Node* newNode = (Node*)malloc(sizeof(Node));
15     newNode->data = data;
16     newNode->next = NULL;
17     return newNode;
18 }
19 Queue* createQueue() {
20     Queue* queue = (Queue*)malloc(sizeof(Queue));
21     queue->front = NULL;
22     queue->rear = NULL;
23     queue->size = 0;
24     return queue;
25 }
26 bool isEmpty(Queue* queue) {
27     return (queue->size == 0);
28 }
29 void enqueue(Queue* queue, int data) {
30     Node* newNode = createNode(data);
31     if (isEmpty(queue)) {
32         queue->front = newNode;
33         queue->rear = newNode;
34     } else {
35         queue->rear->next = newNode;
```

Run

Output

Clear

```
/tmp/OjeS70chJx.o
Front item is: 10
Dequeued: 10
Front item is: 20

=== Code Execution Successful ===
```



```
main.c
35 queue->rear->next = newNode;
36 queue->rear = newNode;
37 }
38 queue->size++;
39 }
40 int dequeue(Queue* queue) {
41     if (isEmpty(queue)) {
42         printf("Queue is empty!\n");
43         return -1;
44     }
45     Node* tempNode = queue->front;
46     int data = tempNode->data;
47     queue->front = queue->front->next;
48     if (queue->front == NULL) {
49         queue->rear = NULL;
50     }
51     free(tempNode);
52     queue->size--;
53     return data;
54 }
55 int peek(Queue* queue) {
56     if (isEmpty(queue)) {
57         printf("Queue is empty!\n");
58         return -1;
59     }
60     return queue->front->data;
61 }
62 void freeQueue(Queue* queue) {
63     while (!isEmpty(queue)) {
64         dequeue(queue);
65     }
66     free(queue);
67 }
68 int main() {
```

```
Output
/tmp/0je570ch.jx.o
Front item is: 10
Dequeued: 10
Front item is: 20

=== Code Execution Successful ===
```

```
main.c
46 int data = tempNode->data;
47 queue->front = queue->front->next;
48 if (queue->front == NULL) {
49     queue->rear = NULL;
50 }
51 free(tempNode);
52 queue->size--;
53 return data;
54 }
55 int peek(Queue* queue) {
56     if (isEmpty(queue)) {
57         printf("Queue is empty!\n");
58         return -1;
59     }
60     return queue->front->data;
61 }
62 void freeQueue(Queue* queue) {
63     while (!isEmpty(queue)) {
64         dequeue(queue);
65     }
66     free(queue);
67 }
68 int main() {
69     Queue* queue = createQueue();
70     enqueue(queue, 10);
71     enqueue(queue, 20);
72     enqueue(queue, 30);
73     printf("Front item is: %d\n", peek(queue));
74     printf("Dequeued: %d\n", dequeue(queue));
75     printf("Front item is: %d\n", peek(queue));
76     freeQueue(queue);
77     return 0;
78 }
79 }
```

```
Output
/tmp/0je570ch.jx.o
Front item is: 10
Dequeued: 10
Front item is: 20

=== Code Execution Successful ===
```

3. Write a c program for Queue Operation of Enqueue and display.

main.c

Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #define MAX 100
5 typedef struct {
6     int front, rear, size;
7     int capacity;
8     int* array;
9 } Queue;
10 Queue* createQueue(int capacity) {
11     Queue* queue = (Queue*)malloc(sizeof(Queue));
12     queue->capacity = capacity;
13     queue->front = 0;
14     queue->rear = capacity - 1;
15     queue->size = 0;
16     queue->array = (int*)malloc(queue->capacity * sizeof(int));
17     return queue;
18 }
19 bool isEmpty(Queue* queue) {
20     return (queue->size == 0);
21 }
22 bool isFull(Queue* queue) {
23     return (queue->size == queue->capacity);
24 }
25 void enqueue(Queue* queue, int item) {
26     if (isFull(queue)) {
27         printf("Queue is full!\n");
28         return;
29     }
30     queue->rear = (queue->rear + 1) % queue->capacity;
31     queue->array[queue->rear] = item;
32     queue->size++;
33     printf("Inserted %d\n", item);
34 }
35 void displayQueue(Queue* queue) {
```

```
/tmp/cALC630G50.o

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 10
Inserted 10

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 20
Inserted 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 2
Queue elements are:
10 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 3
Exiting...

=== Code Execution Successful ===
```

main.c

Share

Run

Output

Clear

```
35 void displayQueue(Queue* queue) {
36     if (isEmpty(queue)) {
37         printf("Queue is empty!\n");
38         return;
39     }
40     printf("Queue elements are:\n");
41     for (int i = 0; i < queue->size; i++) {
42         int index = (queue->front + i) % queue->capacity;
43         printf("%d ", queue->array[index]);
44     }
45     printf("\n");
46 }
47 void freeQueue(Queue* queue) {
48     free(queue->array);
49     free(queue);
50 }
51 int main() {
52     Queue* queue = createQueue(MAX);
53     int choice, item;
54     while (1) {
55         printf("\nQueue Operations Menu:\n");
56         printf("1. Enqueue\n");
57         printf("2. Display\n");
58         printf("3. Exit\n");
59         printf("Enter your choice: ");
60         scanf("%d", &choice);
61         switch (choice) {
62             case 1:
63                 printf("Enter value to enqueue: ");
64                 scanf("%d", &item);
65                 enqueue(queue, item);
66                 break;
67             case 2:
68                 displayQueue(queue);
```

```
Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 10
Inserted 10

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 1
Enter value to enqueue: 20
Inserted 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 2
Queue elements are:
10 20

Queue Operations Menu:
1. Enqueue
2. Display
3. Exit
Enter your choice: 3
Exiting...

=== Code Execution Successful ===
```

```
main.c
46 }
47 void freeQueue(Queue* queue) {
48     free(queue->array);
49     free(queue);
50 }
51 int main() {
52     Queue* queue = createQueue(MAX);
53     int choice, item;
54     while (1) {
55         printf("\nQueue Operations Menu:\n");
56         printf("1. Enqueue\n");
57         printf("2. Display\n");
58         printf("3. Exit\n");
59         printf("Enter your choice: ");
60         scanf("%d", &choice);
61         switch (choice) {
62             case 1:
63                 printf("Enter value to enqueue: ");
64                 scanf("%d", &item);
65                 enqueue(queue, item);
66                 break;
67             case 2:
68                 displayQueue(queue);
69                 break;
70             case 3:
71                 freeQueue(queue);
72                 printf("Exiting...\n");
73                 return 0;
74             default:
75                 printf("Invalid choice! Please try again.\n");
76         }
77     }
78 }
79
```

Output

Queue Operations Menu:  
1. Enqueue  
2. Display  
3. Exit  
Enter your choice: 1  
Enter value to enqueue: 10  
Inserted 10

Queue Operations Menu:  
1. Enqueue  
2. Display  
3. Exit  
Enter your choice: 1  
Enter value to enqueue: 20  
Inserted 20

Queue Operations Menu:  
1. Enqueue  
2. Display  
3. Exit  
Enter your choice: 2  
Queue elements are:  
10 20

Queue Operations Menu:  
1. Enqueue  
2. Display  
3. Exit  
Enter your choice: 3  
Exiting...

=== Code Execution Successful ===

4. Write a c program for Queue operation of Dequeue and display.

main.c

Share

Run

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #define MAX 100
5 typedef struct {
6     int front, rear, size;
7     int capacity;
8     int* array;
9 } Queue;
10 Queue* createQueue(int capacity) {
11     Queue* queue = (Queue*)malloc(sizeof(Queue));
12     queue->capacity = capacity;
13     queue->front = 0;
14     queue->rear = capacity - 1;
15     queue->size = 0;
16     queue->array = (int*)malloc(queue->capacity * sizeof(int));
17     return queue;
18 }
19 bool isEmpty(Queue* queue) {
20     return (queue->size == 0);
21 }
22 bool isFull(Queue* queue) {
23     return (queue->size == queue->capacity);
24 }
25 void enqueue(Queue* queue, int item) {
26     if (isFull(queue)) {
27         printf("Queue is full!\n");
28         return;
29     }
30     queue->rear = (queue->rear + 1) % queue->capacity;
31     queue->array[queue->rear] = item;
32     queue->size++;
33     printf("Inserted %d\n", item);
34 }
35 int dequeue(Queue* queue) {
```

Output

Clear

```
/tmp/fAaKB3tnPx.o

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 323
Inserted 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 123
Inserted 123

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued item: 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue elements are:
```

main.c

Share

Run

```
35 int dequeue(Queue* queue) {
36     if (isEmpty(queue)) {
37         printf("Queue is empty!\n");
38         return -1;
39     }
40     int item = queue->array[queue->front];
41     queue->front = (queue->front + 1) % queue->capacity;
42     queue->size--;
43     return item;
44 }
45 void displayQueue(Queue* queue) {
46     if (isEmpty(queue)) {
47         printf("Queue is empty!\n");
48         return;
49     }
50     printf("Queue elements are:\n");
51     for (int i = 0; i < queue->size; i++) {
52         int index = (queue->front + i) % queue->capacity;
53         printf("%d ", queue->array[index]);
54     }
55     printf("\n");
56 }
57 void freeQueue(Queue* queue) {
58     free(queue->array);
59     free(queue);
60 }
61 int main() {
62     Queue* queue = createQueue(MAX);
63     int choice, item;
64     while (1) {
65         printf("\nQueue Operations Menu:\n");
66         printf("1. Enqueue\n");
67         printf("2. Dequeue\n");
68         printf("3. Display\n");
```

Output

Clear


```
/tmp/fAaKB3tnPx.o

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 323
Inserted 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 123
Inserted 123

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued item: 323

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue elements are:
```

 Edit with WPS Office

main.c

Share

Run

63

int choice, item;

64

while (1) {

65

printf("\nQueue Operations Menu:\n");

66

printf("1. Enqueue\n");

67

printf("2. Dequeue\n");

68

printf("3. Display\n");

69

printf("4. Exit\n");

70

printf("Enter your choice: ");

71

scanf("%d", &choice);

72

switch (choice) {

73

case 1:

74

printf("Enter value to enqueue: ");

75

scanf("%d", &item);

76

enqueue(queue, item);

77

break;

78

case 2:

79

item = dequeue(queue);

80

if (item != -1) {

81

printf("Dequeued item: %d\n", item);

82

}

83

break;

84

case 3:

85

displayQueue(queue);

86

break;

87

case 4:

88

freeQueue(queue);

89

printf("Exiting...\n");

90

return 0;

91

default:

92

printf("Invalid choice! Please try again.\n");

93

}

94

}

95

}

96

Output

Clear

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter value to enqueue: 123

Inserted 123

Queue Operations Menu:

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 2

Dequeued item: 323

Queue Operations Menu:

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 3

Queue elements are:

123

Queue Operations Menu:

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 4

Exiting...

=== Code Execution Successful ===