

**Manav Rachna International Institute of Research and Studies Bachelor's in
computer applications**

Data Structures using C



Submitted by: Urvashi Pahuja

Department: School Of Computer Applications

Course: Bachelors in Computers Applications

Roll no. : 24/SCA/BCA/087

Semester: 2nd

Subject: Data Structures Using C

DS FILE

AIM1: ENQUE OR THE INSERTION IN QUEUES

```
#include <stdio.h>

#define SIZE 100

int queue[SIZE];
int front = -1, rear = -1;

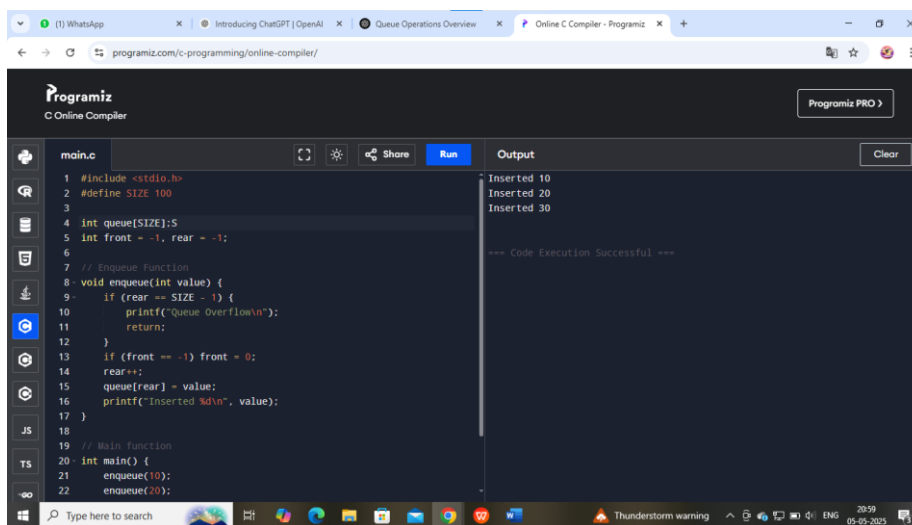
// Enqueue Function
void enqueue(int value) {
    if (rear == SIZE - 1) {
        printf("Queue Overflow\n");
        return;
    }
    if (front == -1) front = 0;
    rear++;
    queue[rear] = value;
    printf("Inserted %d\n", value);
```

```
}
```

```
// Main function
```

```
int main() {  
    enqueue(10);  
    enqueue(20);  
    enqueue(30);  
    return 0;  
}
```

OUTPUT:



The screenshot displays the Programiz Online C Compiler interface. The code editor on the left contains a C program that defines a queue of size 5, enqueues the values 10, 20, and 30, and prints each value as it is inserted. The output window on the right shows the results of the program execution: 'Inserted 10', 'Inserted 20', and 'Inserted 30'. The status bar at the bottom indicates that the code execution was successful.

```
1 #include <stdio.h>  
2 #define SIZE 100  
3  
4 int queue[SIZE];  
5 int front = -1, rear = -1;  
6  
7 // Enqueue Function  
8 void enqueue(int value) {  
9     if (rear == SIZE - 1) {  
10        printf("Queue Overflow\n");  
11        return;  
12    }  
13    if (front == -1) front = 0;  
14    rear++;  
15    queue[rear] = value;  
16    printf("Inserted %d\n", value);  
17 }  
18  
19 // Main function  
20 int main() {  
21     enqueue(10);  
22     enqueue(20);
```

Inserted 10
Inserted 20
Inserted 30

--- Code Execution Successful ---

AIM 2: DEQUEUE OR DELETION IN QUEUES.

```
#include <stdio.h>

#define SIZE 100

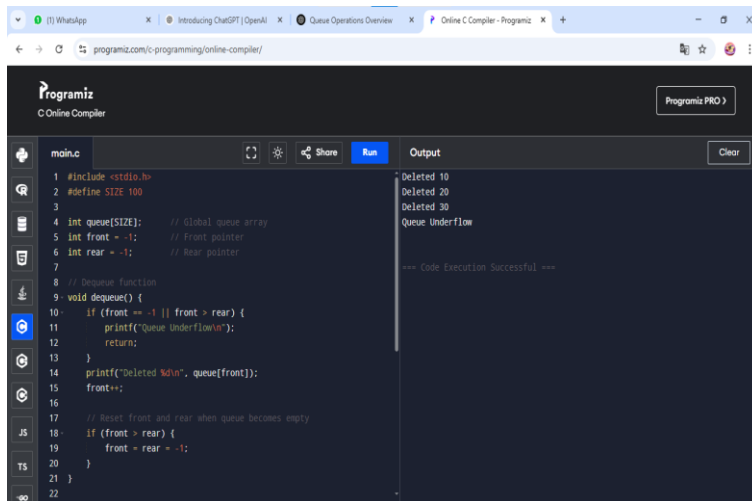
int queue[SIZE];    // Global queue array
int front = -1;     // Front pointer
int rear = -1;      // Rear pointer

// Dequeue function
void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow\n");
        return;
    }
    printf("Deleted %d\n", queue[front]);
    front++;

    // Reset front and rear when queue becomes
    empty
```

```
    if (front > rear) {  
        front = rear = -1;  
    }  
}  
  
// Main function to test  
int main() {  
    // Simulate inserting some elements  
    rear = 2;  
    front = 0;  
    queue[0] = 10;  
    queue[1] = 20;  
    queue[2] = 30;  
  
    dequeue(); // Should delete 10  
    dequeue(); // Should delete 20  
    dequeue(); // Should delete 30  
    dequeue(); // Should print underflow  
  
    return 0;  
}
```

OUTPUT:



```
1 #include <stdio.h>
2 #define SIZE 100
3
4 int queue[SIZE]; // Global queue array
5 int front = -1; // Front pointer
6 int rear = -1; // Rear pointer
7
8 // Dequeue function
9 void dequeue() {
10     if (front == -1 || front > rear) {
11         printf("Queue Underflow\n");
12         return;
13     }
14     printf("Deleted %d\n", queue[front]);
15     front++;
16
17     // Reset front and rear when queue becomes empty
18     if (front > rear) {
19         front = rear = -1;
20     }
21 }
22
```

Deleted 10
Deleted 20
Deleted 30
Queue Underflow

=== Code Execution Successful ===

AIM 3: PEEK FUNCTION IN QUEUES.

```
#include <stdio.h>
```

```
#define SIZE 100
```

```
int queue[SIZE];
```

```
int front = -1;
```

```
int rear = -1;
```

```
void peek() {
```

```
    if (front == -1 || front > rear) {
```

```
        printf("Queue is empty\n");
```

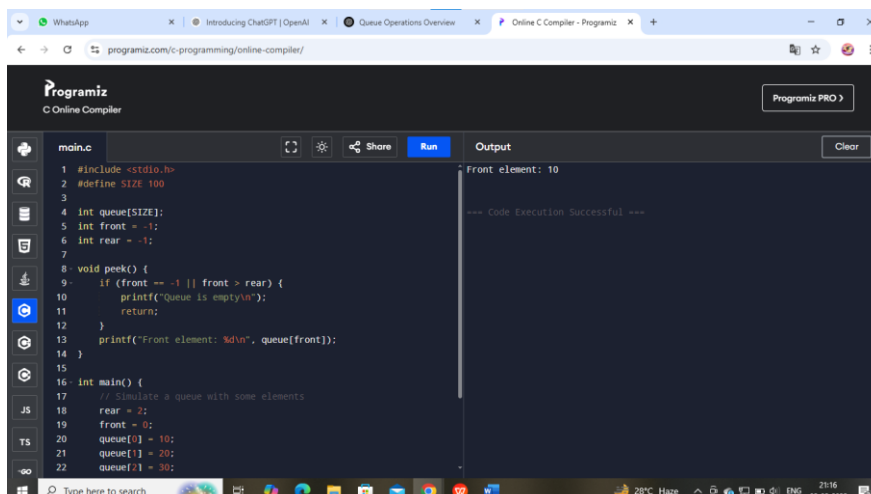
```
        return;
```

```
    }
```

```
printf("Front element: %d\n", queue[front]);  
}
```

```
int main() {  
    // Simulate a queue with some elements  
    rear = 2;  
    front = 0;  
    queue[0] = 10;  
    queue[1] = 20;  
    queue[2] = 30;  
  
    peek(); // Should print: Front element: 10  
  
    return 0;  
}
```

OUTPUT:



```
1 #include <stdio.h>  
2 #define SIZE 100  
3  
4 int queue[SIZE];  
5 int front = -1;  
6 int rear = -1;  
7  
8 void peek() {  
9     if (front == -1 || front > rear) {  
10        printf("Queue is empty\n");  
11        return;  
12    }  
13    printf("Front element: %d\n", queue[front]);  
14 }  
15  
16 int main() {  
17     // Simulate a queue with some elements  
18     rear = 2;  
19     front = 0;  
20     queue[0] = 10;  
21     queue[1] = 20;  
22     queue[2] = 30;  
  
    peek();  
  
    return 0;  
}
```

Output: Front element: 10
--- Code Execution Successful ---

AIM 4: DISPLAY FUNCTION IN QUEUES.

```
#include <stdio.h>
```

```
#define SIZE 100
```

```
int queue[SIZE];
```

```
int front = -1;
```

```
int rear = -1;
```

```
void display() {
```

```
    if (front == -1 || front > rear) {
```

```
        printf("Queue is empty\n");
```

```
        return;
```

```
    }
```

```
    printf("Queue elements: ");
```

```
    for (int i = front; i <= rear; i++) {
```

```
        printf("%d ", queue[i]);
```

```
    }
```

```
    printf("\n");
```

```
}
```



```

int main() {

    // Simulate some queue values

    front = 0;

    rear = 2;

    queue[0] = 10;

    queue[1] = 20;

    queue[2] = 30;

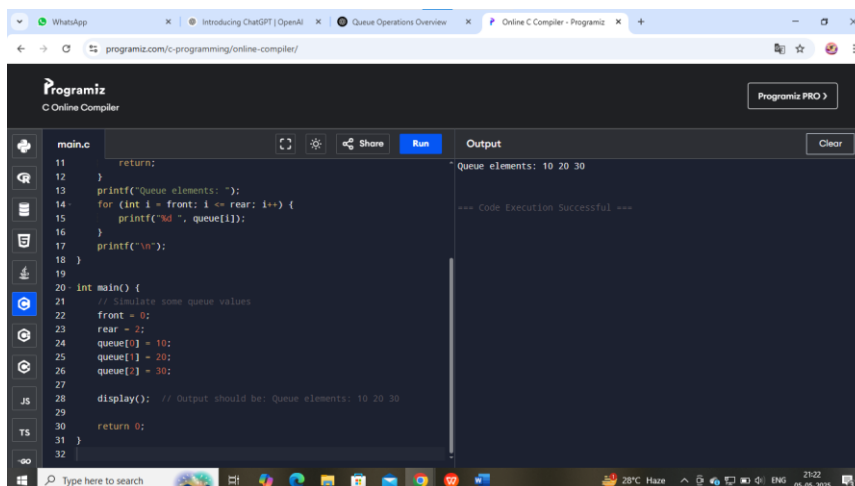

    display(); // Output should be: Queue elements: 10
20 30


    return 0;

}

```

OUTPUT:



The screenshot shows a web browser window with the Programiz C Online Compiler. The code in the editor is as follows:

```

11     return;
12 }
13 printf("Queue elements: ");
14 for (int i = front; i <= rear; i++) {
15     printf("%d ", queue[i]);
16 }
17 printf("\n");
18 }
19
20 int main() {
21     // Simulate some queue values
22     front = 0;
23     rear = 2;
24     queue[0] = 10;
25     queue[1] = 20;
26     queue[2] = 30;
27
28     display(); // Output should be: Queue elements: 10 20 30
29
30     return 0;
31 }
32

```

The output window on the right shows the result of the program execution:

```

Queue elements: 10 20 30
=== Code Execution Successful ===

```