

Data Structures using C

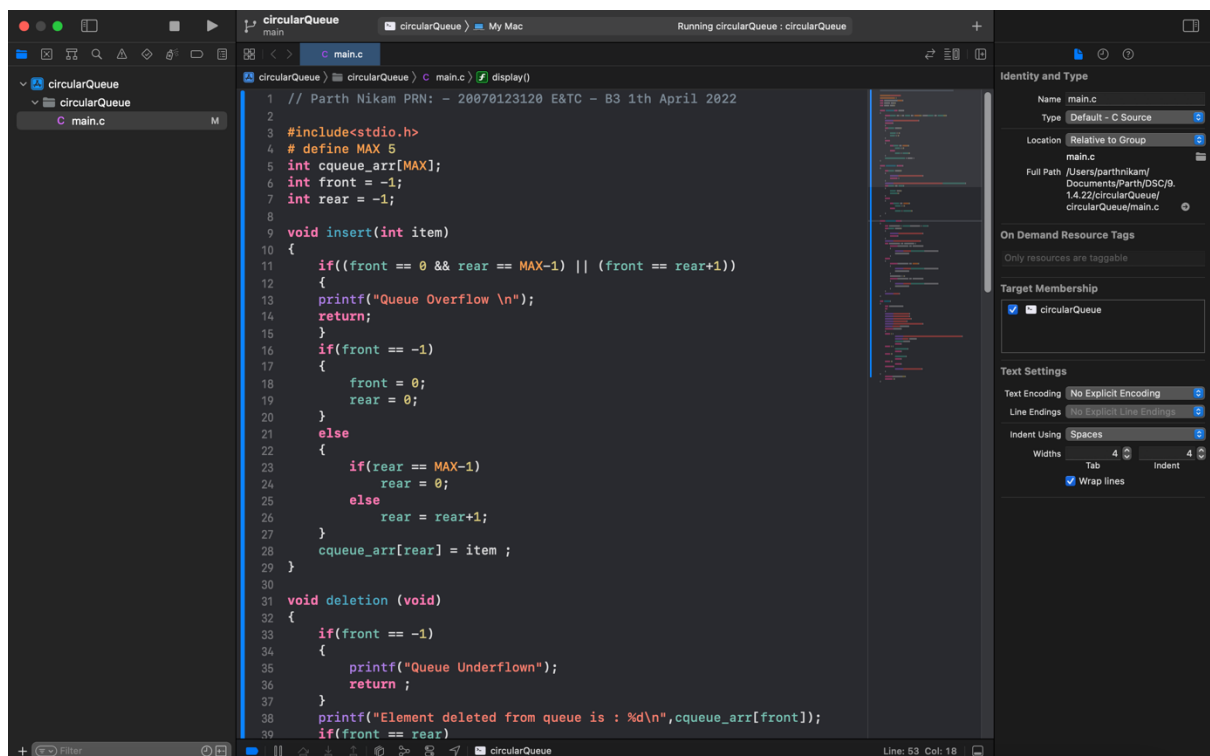
1st April 2022

Parth Nikam
20070123120
E&TC – B3

Aim: - Studying and coding on Circular Queue.

Objective: - To perform enqueue, dequeue, and display operations on Circular Queue in C language

Code: -

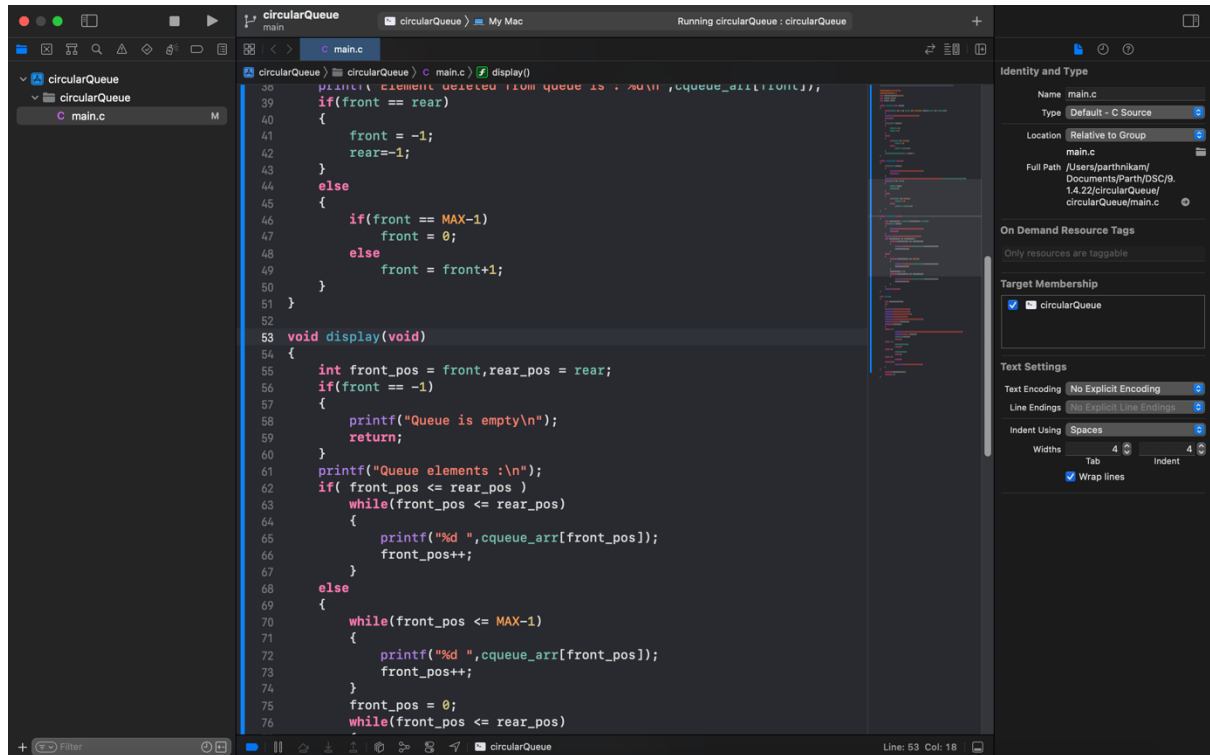


```
1 // Parth Nikam PRN: - 20070123120 E&TC - B3 1th April 2022
2
3 #include<stdio.h>
4 # define MAX 5
5 int cqueue_arr[MAX];
6 int front = -1;
7 int rear = -1;
8
9 void insert(int item)
10 {
11     if(((front == 0 && rear == MAX-1) || (front == rear+1))
12     {
13         printf("Queue Overflow \n");
14         return;
15     }
16     if(front == -1)
17     {
18         front = 0;
19         rear = 0;
20     }
21     else
22     {
23         if(rear == MAX-1)
24             rear = 0;
25         else
26             rear = rear+1;
27     }
28     cqueue_arr[rear] = item ;
29 }
30
31 void deletion (void)
32 {
33     if(front == -1)
34     {
35         printf("Queue Underflown");
36         return ;
37     }
38     printf("Element deleted from queue is : %d\n",cqueue_arr[front]);
39     if(front == rear)
```

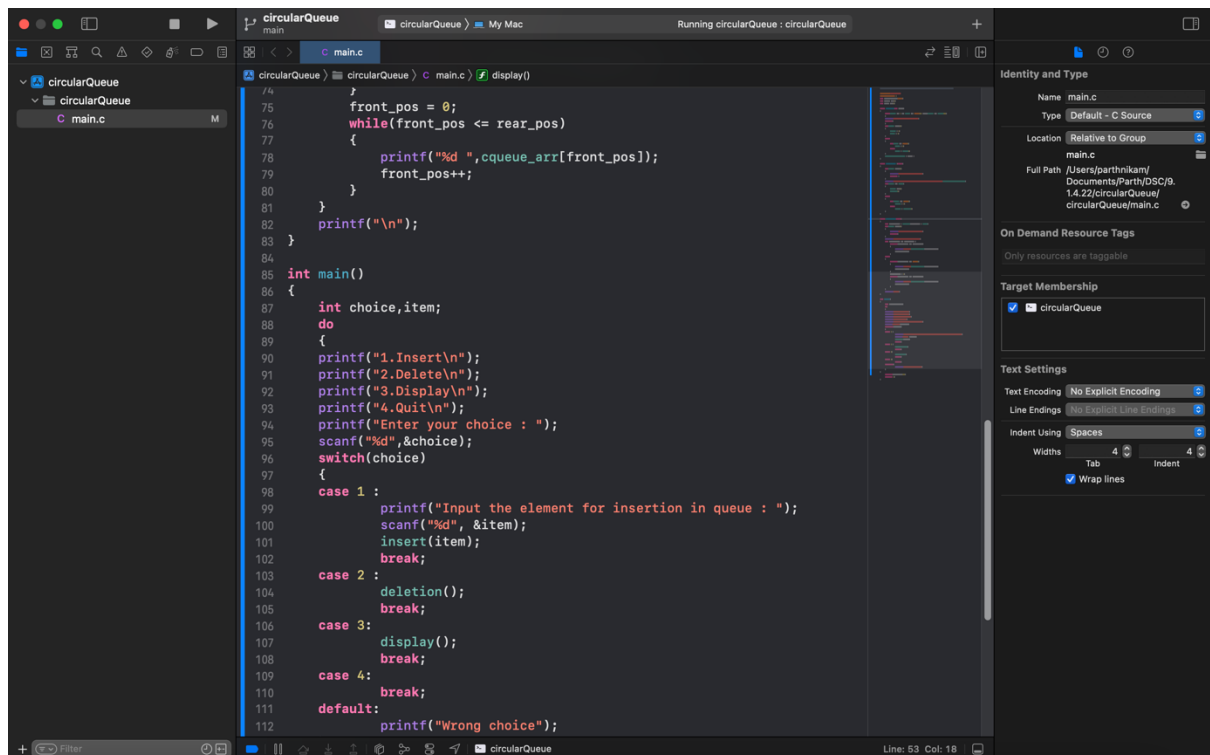
Data Structures using C

1st April 2022

Parth Nikam
20070123120
E&TC – B3



```
38 printf("Element deleted from queue is : %d\n", queue_arr[front]);
39 if(front == rear)
40 {
41     front = -1;
42     rear = -1;
43 }
44 else
45 {
46     if(front == MAX-1)
47         front = 0;
48     else
49         front = front+1;
50 }
51 }
52
53 void display(void)
54 {
55     int front_pos = front, rear_pos = rear;
56     if(front == -1)
57     {
58         printf("Queue is empty\n");
59         return;
60     }
61     printf("Queue elements :\n");
62     if( front_pos <= rear_pos )
63         while(front_pos <= rear_pos)
64         {
65             printf("%d ", queue_arr[front_pos]);
66             front_pos++;
67         }
68     else
69     {
70         while(front_pos <= MAX-1)
71         {
72             printf("%d ", queue_arr[front_pos]);
73             front_pos++;
74         }
75         front_pos = 0;
76         while(front_pos <= rear_pos)
```

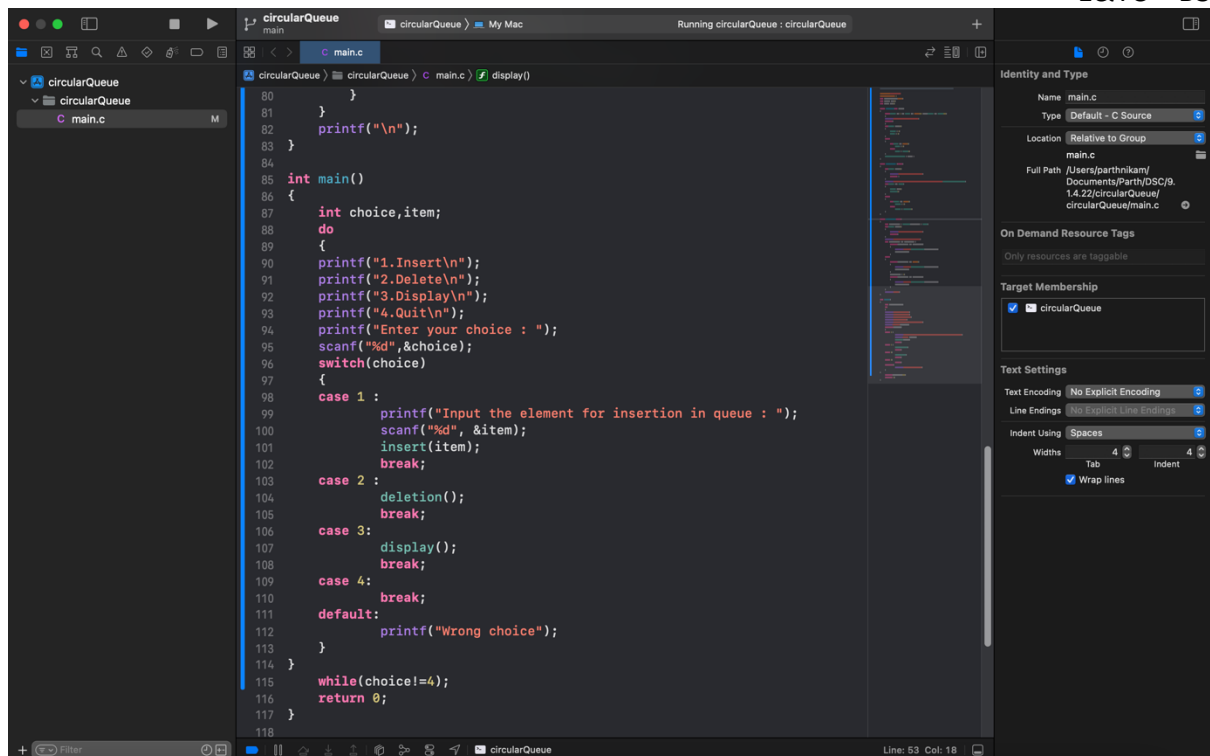


```
75     front_pos = 0;
76     while(front_pos <= rear_pos)
77     {
78         printf("%d ", queue_arr[front_pos]);
79         front_pos++;
80     }
81     printf("\n");
82 }
83
84 int main()
85 {
86     int choice, item;
87     do
88     {
89         printf("1.Insert\n");
90         printf("2.Delete\n");
91         printf("3.Display\n");
92         printf("4.Quit\n");
93         printf("Enter your choice : ");
94         scanf("%d", &choice);
95         switch(choice)
96         {
97             case 1 :
98                 printf("Input the element for insertion in queue : ");
99                 scanf("%d", &item);
100                 insert(item);
101                 break;
102             case 2 :
103                 deletion();
104                 break;
105             case 3 :
106                 display();
107                 break;
108             case 4 :
109                 break;
110             default:
111                 printf("Wrong choice");
112         }
```

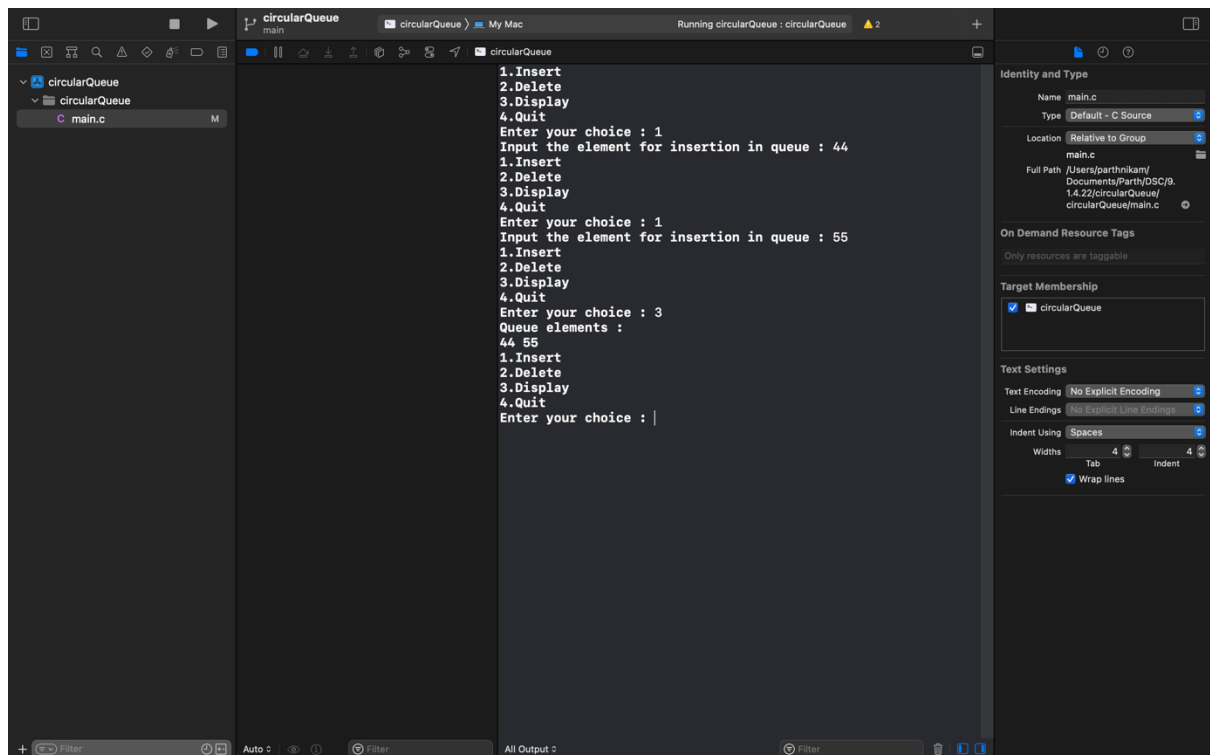
Data Structures using C

1st April 2022

Parth Nikam
20070123120
E&TC – B3



```
80     }
81     }
82     printf("\n");
83 }
84
85 int main()
86 {
87     int choice,item;
88     do
89     {
90         printf("1.Insert\n");
91         printf("2.Delete\n");
92         printf("3.Display\n");
93         printf("4.Quit\n");
94         printf("Enter your choice : ");
95         scanf("%d",&choice);
96         switch(choice)
97         {
98             case 1 :
99                 printf("Input the element for insertion in queue : ");
100                 scanf("%d", &item);
101                 insert(item);
102                 break;
103             case 2 :
104                 deletion();
105                 break;
106             case 3:
107                 display();
108                 break;
109             case 4:
110                 break;
111             default:
112                 printf("Wrong choice");
113         }
114     }
115     while(choice!=4);
116     return 0;
117 }
118
```



```
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 44
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 55
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue elements :
44 55
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : |
```

Result: - A Circular Queue in C is basically a linear data structure to store and manipulate the data elements. It follows the order of First In First Out (FIFO). In Circular Queues, the first element entered into the array is the first element to be removed from the array.