

SIDDHI SINGH
17BIT0028

ITE - 1004
DATA STRUCTURES AND ALGORITHM
EXERCISE - 6

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

struct node
{
    int info;
    struct node *ptr;
}*front, *rear, *temp, *front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
    int no, ch, e;

    printf("\n 1 - Enqueue");
    printf("\n 2 - Dequeue");
    printf("\n 3 - Front element");
    printf("\n 4 - Empty");
    printf("\n 5 - Exit");
    printf("\n 6 - Display");
    printf("\n 7 - Queue size");
    create();
    while (1)
    {
        printf("\n Enter choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                printf("Enter data : ");
                scanf("%d", &no);
                enq(no);
                break;
            case 2:
                deq();
                break;
            case 3:
```

```

        e = fruntelement();
        if (e != 0)
            printf("Front element : %d", e);
        else
            printf("\n No front element in Queue as queue is
empty");
        break;
    case 4:
        empty();
        break;
    case 5:
        exit(0);
    case 6:
        display();
        break;
    case 7:
        queuesize();
        break;
    default:
        printf("Wrong choice, Please enter correct choice  ");
        break;
    }
}

/* Create an empty queue */
void create()
{
    front = rear = NULL;
}

/* Returns queue size */
void queuesize()
{
    printf("\n Queue size : %d", count);
}

/* Enqueing the queue */
void enq(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }
    else
    {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
    }
}

```

```

        temp->ptr = NULL;

        rear = temp;
    }
    count++;
}

/* Displaying the queue elements */
void display()
{
    front1 = front;

    if ((front1 == NULL) && (rear == NULL))
    {
        printf("Queue is empty");
        return;
    }
    while (front1 != rear)
    {
        printf("%d ", front1->info);
        front1 = front1->ptr;
    }
    if (front1 == rear)
        printf("%d", front1->info);
}

/* Dequeueing the queue */
void deq()
{
    front1 = front;

    if (front1 == NULL)
    {
        printf("\n Error: Trying to display elements from empty
queue");
        return;
    }
    else
        if (front1->ptr != NULL)
        {
            front1 = front1->ptr;
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = front1;
        }
        else
        {
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = NULL;
            rear = NULL;
        }
}

```

```

        count--;
    }

    /* Returns the front element of queue */
    int frontelement()
    {
        if ((front != NULL) && (rear != NULL))
            return(front->info);
        else
            return 0;
    }

    /* Display if queue is empty or not */
    void empty()
    {
        if ((front == NULL) && (rear == NULL))
            printf("\n Queue empty");
        else
            printf("Queue not empty");
    }
}

```

The screenshot shows the Code::Blocks IDE with the file `linkedqueue.c` open. The code defines a `node` structure with `info` and `ptr` fields. It includes functions for `frontelement`, `enq`, `deq`, `empty`, `display`, `create`, and `queuesize`. The `main` function tests these operations with a sequence of commands: Enqueue, Dequeue, Front element, Empty, Exit, Display, and Queue size.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct node
5  {
6      int info;
7      struct node *ptr;
8  } *front, *rear, *temp, *front1;
9
10 int frontelement();
11 void enq(int data);
12 void deq();
13 void empty();
14 void display();
15 void create();
16 void queuesize();
17
18 int count = 0;
19
20 void main()
21 {
22     int no, ch, e;
23
24     printf("\n 1 - Enqueue");
25     printf("\n 2 - Dequeue");
26     printf("\n 3 - Front element");
27     printf("\n 4 - Empty");
28     printf("\n 5 - Exit");
29     printf("\n 6 - Display");
30     printf("\n 7 - Queue size");
31     create();

```

linkedqueue.c - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Management

linkedqueue.c

```
30 printf("\n 7 - Queue size");
31 create();
32 while (1)
33 {
34     printf("\n Enter choice : ");
35     scanf("%d", &ch);
36     switch (ch)
37     {
38         case 1:
39             printf("Enter data : ");
40             scanf("%d", &no);
41             enq(no);
42             break;
43         case 2:
44             deq();
45             break;
46         case 3:
47             e = frontelement();
48             if (e != 0)
49                 printf("Front element : %d", e);
50             else
51                 printf("\n No front element in Queue as queue is empty");
52             break;
53         case 4:
54             empty();
55             break;
56         case 5:
57             exit(0);
58         case 6:
59             display();
60             break;
```

C:\Users\Rites\Documents\linkedqueue\linkedqueue.c Windows (CR+LF) default Line 12, Column 12 Insert Read/Write default 6:21 PM 13-Mar-18

linkedqueue.c - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Management

linkedqueue.c

```
56 case 5:
57     exit(0);
58 case 6:
59     display();
60     break;
61 case 7:
62     queueize();
63     break;
64 default:
65     printf("Wrong choice, Please enter correct choice ");
66     break;
67 }
68 }
69 }
70 }
71 /* Create an empty queue */
72 void create()
73 {
74     front = rear = NULL;
75 }
76
77 /* Returns queue size */
78 void queueize()
79 {
80     printf("\n Queue size : %d", count);
81 }
82
83 /* Enqueueing the queue */
84 void enq(int data)
85 {
86     if (rear == NULL)
```

C:\Users\Rites\Documents\linkedqueue\linkedqueue.c Windows (CR+LF) default Line 12, Column 12 Insert Read/Write default 6:22 PM 13-Mar-18

linkedqueue.c - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoryBlocks Settings Help

Management

Projects Symbols Files

Workspace

linkedqueue

```
83  /* Enqueueing the queue */
84  void enq(int data)
85  {
86      if (rear == NULL)
87      {
88          rear = (struct node *)malloc(1*sizeof(struct node));
89          rear->ptr = NULL;
90          rear->info = data;
91          front = rear;
92      }
93      else
94      {
95          temp=(struct node *)malloc(1*sizeof(struct node));
96          rear->ptr = temp;
97          temp->info = data;
98          temp->ptr = NULL;
99
100         rear = temp;
101     }
102     count++;
103 }
104
105 /* Displaying the queue elements */
106 void display()
107 {
108     frontl = front;
109
110     if ((frontl == NULL) && (rear == NULL))
111     {
112         printf("Queue is empty");
113         return;
114     }
115 }
```

C:\Users\Rites\Documents\linkedqueue\linkedqueue.c Windows (CR+LF) default Line 12, Column 12 Insert Read/Write default 6:22 PM 13-Mar-18

linkedqueue.c - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoryBlocks Settings Help

Management

Projects Symbols Files

Workspace

linkedqueue

```
112     printf("Queue is empty");
113     return;
114 }
115 while (frontl != rear)
116 {
117     printf("%d ", frontl->info);
118     frontl = frontl->ptr;
119 }
120 if (frontl == rear)
121     printf("%d", frontl->info);
122 }
123
124 /* Dequeueing the queue */
125 void deq()
126 {
127     frontl = front;
128
129     if (frontl == NULL)
130     {
131         printf("\n Error: Trying to display elements from empty queue");
132         return;
133     }
134     else
135     {
136         if (frontl->ptr != NULL)
137         {
138             frontl = frontl->ptr;
139             printf("\n Dequeued value : %d", front->info);
140             free(front);
141             front = frontl;
142         }
143         else
144         {
145             printf("\n Dequeued value : %d", front->info);
146             free(front);
147             front = NULL;
148         }
149     }
150 }
```

C:\Users\Rites\Documents\linkedqueue\linkedqueue.c Windows (CR+LF) default Line 12, Column 12 Insert Read/Write default 6:22 PM 13-Mar-18

```
139         free(front);
140         front = front1;
141     }
142     else
143     {
144         printf("\n Dequed value : %d", front->info);
145         free(front);
146         front = NULL;
147         rear = NULL;
148     }
149     count--;
150 }
151
152 /* Returns the front element of queue */
153 int frontelement()
154 {
155     if ((front != NULL) && (rear != NULL))
156         return(front->info);
157     else
158         return 0;
159 }
160
161 /* Display if queue is empty or not */
162 void empty()
163 {
164     if ((front == NULL) && (rear == NULL))
165         printf("\n Queue empty");
166     else
167         printf("Queue not empty");
168 }
169
```

OUTPUT

```
1 - Enque
2 - Deque
3 - Front element
4 - Empty
5 - Exit
6 - Display
7 - Queue size
Enter choice : 1
Enter data : 12

Enter choice : 1
Enter data : 28

Enter choice : 1
Enter data : 23

Enter choice : 1
Enter data : 43

Enter choice : 1
Enter data : 67
```

```
Enter choice : 1  
Enter data : 67
```

```
Enter choice : 2
```

```
Dequed value : 12
```

```
Enter choice : 3
```

```
Front element : 28
```

```
Enter choice : 6
```

```
28 23 43 67
```

```
Enter choice : 7
```

```
Queue size : 4
```

```
Enter choice : 4
```

```
Queue not empty
```

```
Enter choice : 5
```

17BIT100V