|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No. 05**

|  |  |
| --- | --- |
| Semester | S.E. Semester III – Computer Engineering |
| Subject | Data Structures Lab (CSL301) |
| Subject Professor In-charge | Prof. Swapnil S. Sonawane |
| Assisting Teachers | Prof. Swapnil S. Sonawane |

|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**Title:**

Implement Linear Queue ADT using array

**Objective:**

Students will be able to implement Linear and Non-Linear data structures.

**Explanation:**

In queue, insertion and deletion happen at the opposite ends, so implementation is not as simple as stack.

To implement a queue using array, create an array arr of size n and take two variables front and rear both of which will be initialized to 0 which means the queue is currently empty.

Element rear is the index upto which the elements are stored in the array and front is the index of the first element of the array. Now, some of the implementation of queue operations are as follows:

1) Enqueue: Addition of an element to the queue. Adding an element will be performed after checking whether the queue is full or not. If rear < n which indicates that the array is not full then store the element at arr[rear] and increment rear by 1 but if rear == n then it is said to be an Overflow condition as the array is full.

2) Dequeue: Removal of an element from the queue. An element can only be deleted when there is at least an element to delete i.e. rear > 0. Now, element at arr[front] can be deleted but all the remaining elements have to shifted to the left by one position in order for the dequeue operation to delete the second element from the left on another dequeue operation.

3) Front: Get the front element from the queue i.e. arr[front] if queue is not empty.

4) Display: Print all element of the queue. If the queue is non-empty, traverse and print all the elements from index front to rear.

**Program Code:**

#include<stdio.h>

#define N 10

typedef struct queue

{

int a[N];

int front,rear;

}queue;

void enqueue(queue \*q,int x)

{

if(q->rear==N-1)

{

printf("\nQueue Overflow");

}

else

{

q->rear++;

q->a[q->rear]=x;

}

}

int isempty(queue \*q)

{

if(q->front==q->rear+1)

return 1;

else

return 0;

}

int dequeue(queue \*q)

{

int x;

if(isempty(q))

return -1;

else

{

x=q->a[q->front];

q->front++;

return x;

}

}

void display(queue \*q)

{

int i;

if(isempty(q))

{

printf("\nQueue is empty...");

}

else

{

for(i=q->front;i<=q->rear;i++)

{

printf("\n%d",q->a[i]);

}

}

}

int main()

{

int ch,x;

queue q;

q.front=0;

q.rear=-1;

while(1)

{

printf("\n Menu:\n 1-ENQUEUE\n 2-DEQUEUE\n 3-DISPLAY\n 4-EXIT\n Enter Choice=");

scanf("%d",&ch);

if(ch==4)

break;

switch(ch)

{

case 1:

{

printf("\nEnter element to be inserted=");

scanf("%d",&x);

enqueue(&q,x);

}

break;

case 2:

{

x=dequeue(&q);

if(x==-1)

{

printf("\nQueue Underflow...");

}

else

{

printf("\nDeleted Element=%d",x);

}

}

break;

case 3:

{

display(&q);

}

break;

default:

{

printf("\nInvalid Choice...");

}

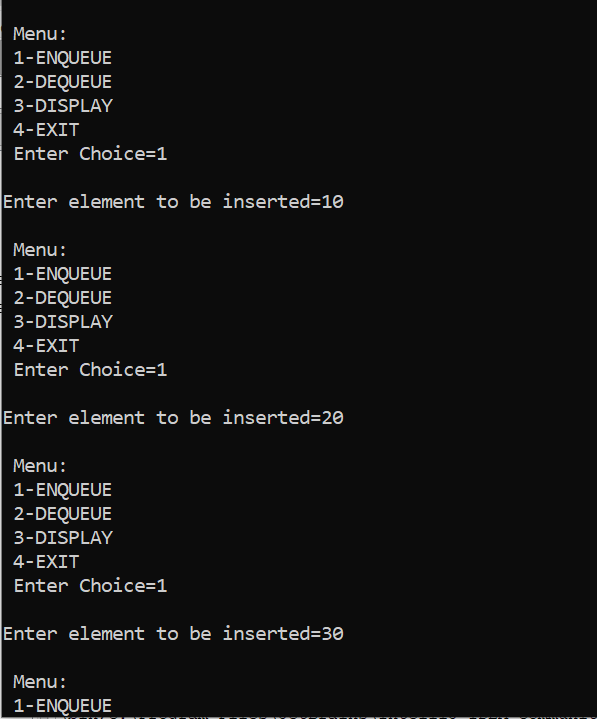
}

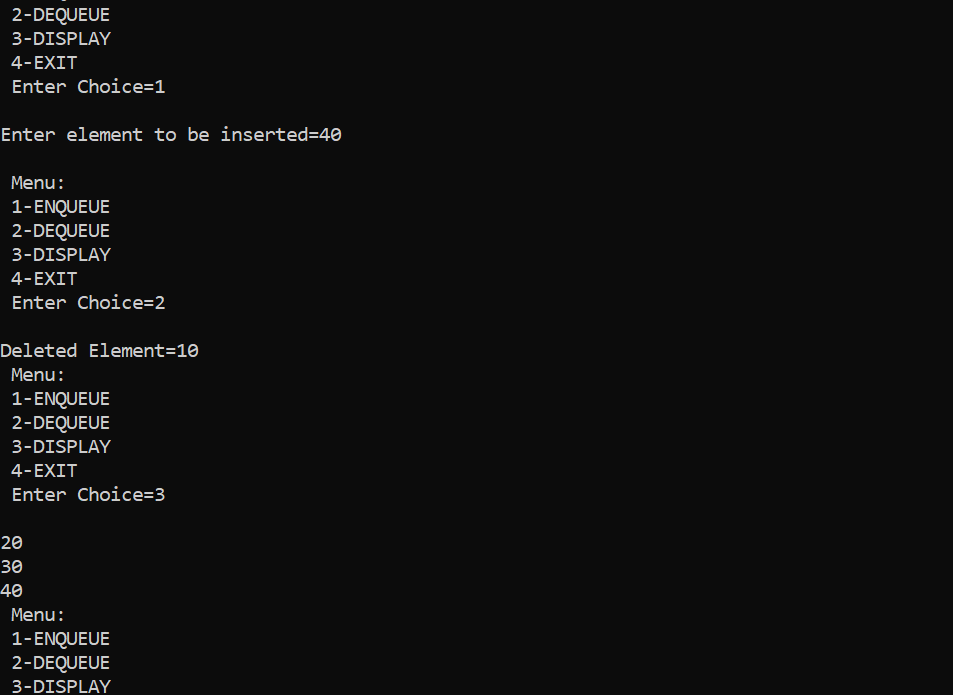
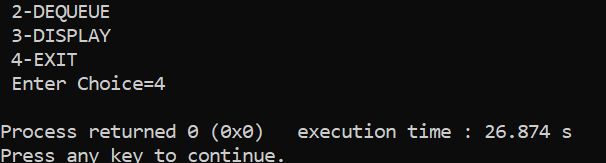
}

return 0;

}

**Output:**

****

****

**Conclusion:**

The students were able to successfully implement a linear queue.