



Data Structure and Algorithm (CS09203)

Lab Report

Name: Muhammad Umer
Registration #: CSU-F16-104
Lab Report #: 01
Dated: 26-03-2018
Submitted To: Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 2

Queue with Array implementation

Objective

The objective of this session is to understand the various operations on queues using array structure in C++.

Software Tool

1. Window 7 (32-bit)
2. Sublime Text Editor
3. Dev C++

Theory

Queue using Array:-

This manual discusses an important data structure, called a queue. The idea of a queue in computer science is the same as the idea of the queues to which you are accustomed in everyday life. There are queues of customers in a bank or in a grocery store and queues of cars waiting to pass through a tollbooth. Similarly, because a computer can send a print request faster than a printer can print, a queue of documents is often waiting to be printed at a printer. The general rule to process elements in a queue is that the customer at the front of the queue is served next and that when a new customer arrives, he or she stands at the end of the queue. That is, a queue is a First In First Out data structure.

A queue is a set of elements of the same type in which the elements are added at one end, called the back or rear, and deleted from the other end, called the front. For example, consider a line of customers in a bank, wherein the customers are waiting to withdraw/deposit money or to conduct some other business. Each new customer gets in the line at the rear. Whenever a teller is ready for a new customer, the customer at the front of the line is served.

The rear of the queue is accessed whenever a new element is added to the queue, and the front of the queue is accessed whenever an element is deleted from the queue. As in a stack, the middle elements of the queue are inaccessible, even if the queue elements are stored in an array.

Queue: A data structure in which the elements are added at one end, called the rear, and deleted from the other end, called the front; a First-In-First-Out (FIFO) data structure.

Queues may be represented in the computer in various ways, usually by means at one-way list or linear arrays. Unless otherwise stated or implied each of our queues will be maintained by a linear array QUEUE and two pointer variable FRONT containing the location of the front element of the queue and REAR containing the location of the rear element of the queue. The condition $FRONT = NULL$ will indicate that the queue is empty.

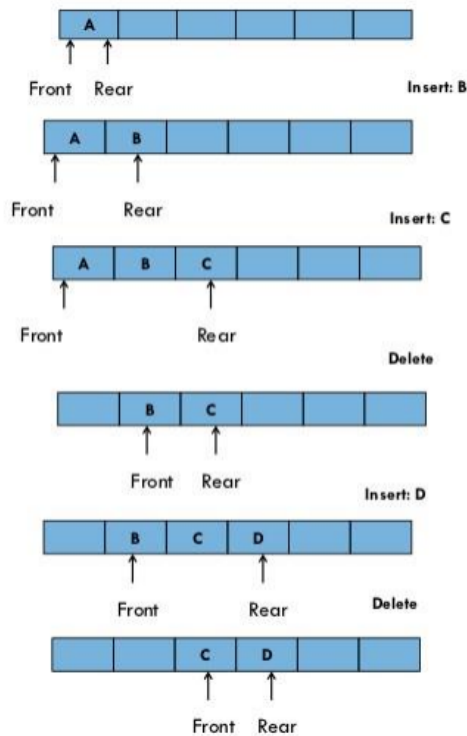


Figure 1:

Whenever an element is deleted from the queue the value of FRONT is increased by one. This can be implemented by the assignment.

$$\text{FRONT} = \text{FRONT} + 1$$

Similarly, whenever an element is added to the queue the value of REAR is increased by one. This can be implemented by the assignment.

$$\text{REAR} = \text{REAR} + 1$$

This means that after N insertions the rear element of the queue will occupy QUEUE [N] or in other words eventually the queue will occupy the last part of the array. This occurs even though the queue itself may not contain many elements.

Suppose we want to insert an element ITEM into a queue at the time the queue does occupy the last part of the array i.e. when $\text{REAR} = \text{N}$. One way is to do this simply move the entire queue to the beginning of the array changing FRONT and REAR accordingly, and then inserting ITEM as above. This procedure may be very expensive. The procedure we adopt is to assume that the array QUEUE is circular that is that QUEUE [1] comes after QUEUE [N] in the array. With this assumption, we insert ITEM into the queue by assigning ITEM to QUEUE [1]. Specifically, instead of increasing REAR to N+1 we reset $\text{REAR}=1$ and then assign

$$\text{QUEUE}[\text{REAR}] = \text{ITEM}$$

Similarly, if $\text{FRONT}=\text{N}$ and an element is deleted then we reset $\text{FRONT}=1$ instead of increasing

$$\text{FRONT to N+1.}$$

Suppose that our queue only contains one element i.e. suppose that

$$\text{FRONT} = \text{REAR} \quad \text{NULL}$$

And suppose that the element is deleted. Then we assign $\text{FRONT} = \text{NULL}$ and $\text{REAR} = \text{NULL}$ to indicate that the queue is empty.

Algorithm for insertion:-

QINSERT (QUEUE, N, FRONT, REAR, ITEM)

This procedure inserts an element ITEM into a queue.

1.[Queue already filled?]

If FRONT = 1 and REAR = N or if FRONT = REAR + 1 then

Write OVERFLOW and Return.

2.[Find new value of REAR]

If FRONT = NULL then [Queue initially empty] Set FRONT = 1
and REAR = 1.

else if REAR = N then

Set REAR = 1.

else

Set REAR = REAR + 1. [End of if Structure]

3.Set QUEUE[REAR] = ITEM [This insert new element]

4.Return.

Algorithm for Deletion from Queues:-

QDELETE (QUEUE, N, FRONT, REAR, ITEM)

This procedure deletes an element from a queue and assigns it to the variable ITEM.

1.[Queue already empty?]

If FRONT = NULL then Write Underflow and Return.

2.Set ITEM = QUEUE[FRONT]

3.[Find new value of FRONT]

If FRONT = REAR then [Queue has only one element to start]

Set FRONT = NULL and REAR = NULL

else if FRONT = N then

Set FRONT = 1

else

Set FRONT = FRONT + 1 [End of If Structure]

4.Return.

Lab Task:-

Write a C++ code to perform insertion and deletion in queue using arrays applying the algorithms given in the manual. Create a menu shown below.



Figure 2: The output of the program

Solution:-

```
#include<iostream>
#include<conio.h>
using namespace std;

void insert(int queue[], int n, int& front, int& rear){
    int item;
    if((front == 1 && rear == n) || front == rear + 1){
        cout<<"\n\nError: Queue Overflow!";
        cout<<"\n\nPress any key to continue....";
        getch();
        return;
    }
    cout<<"\n\nEnter a value: ";
    cin>>item;
    if(front == 0){
        front = 1;
        rear = 1;
    }
    else if(rear == n)
        rear = 1;
    else
        ++rear;
```

```

        queue[rear] = item;
        cout<<"\n\nValue inserted Successfully";
        cout<<"\n\nPress any key to continue ....";
        getch();
    }

    void deletion(int queue[], int n, int& front, int& rear){
        if(front == 0){
            cout<<"\n\nError: Queue Underflow!";
            cout<<"\n\nPress any key to continue ....";
            getch();
            return;
        }
        int item;
        item = queue[front];
        if(front == rear){
            front = 0;
            rear = 0;
        }
        else if(front == n)
            front = 1;
        else
            front++;
        cout<<"\n\nValue deleted Successfully";
        cout<<"\n\nPress any key to continue ....";
        getch();
    }

    void display(int queue[], int n, int& front, int& rear){
        if(front == 0 && rear == 0){
            cout<<"\n\nQueue is Empty!\n\n";
            cout<<"\n\nPress any key to continue ....";
            getch();
            return;
        }
        cout<<"\n\nItems in the queue\n\n";
        for(int i=front; i<=rear; i++){
            cout<<queue[i]<<" ";
        }
        cout<<"\n\nPress any key to continue ....";
    }

```

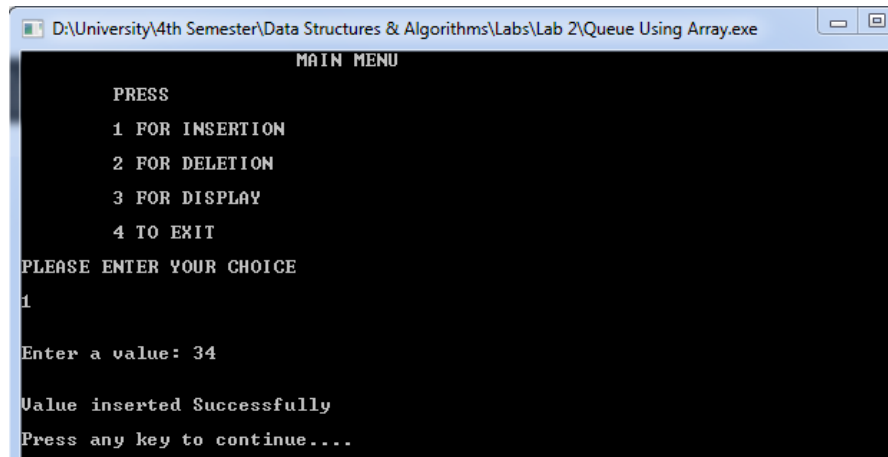
```

        getch ();
    }

    int main()
    {
        int choice , n = 15, queue[n], front = 0, rear = 0,value;
        up:
        system("cls");
        cout<<"\t\t\tMAIN_MENU\n\n";
        cout<<"\tPRESS\n\n";
        cout<<"\t1_FOR_INSERTION\n\n";
        cout<<"\t2_FOR_DELETION\n\n";
        cout<<"\t3_FOR_DISPLAY\n\n";
        cout<<"\t4_TO_EXIT\n\n";
        cout<<"PLEASE_ENTER_YOUR_CHOICE\n\n";
        cin>>choice;
        if(choice == 1){
            insert(queue, n, front, rear);
            goto up;
        }
        else if(choice == 2){
            deletion(queue, n, front, rear);
            goto up;
        }
        else if(choice == 3){
            display(queue, n, front, rear);
            goto up;
        }
        else if(choice == 4)
            exit(0);
        else{
            cout<<"\n\nWRONG_CHOICE!";
            cout<<"\n\nPRESS_ANY_KEY_TO_CHOOSE_AGAIN...";
            getch();
            goto up;
        }
        return 0;
    }

```


Output:-

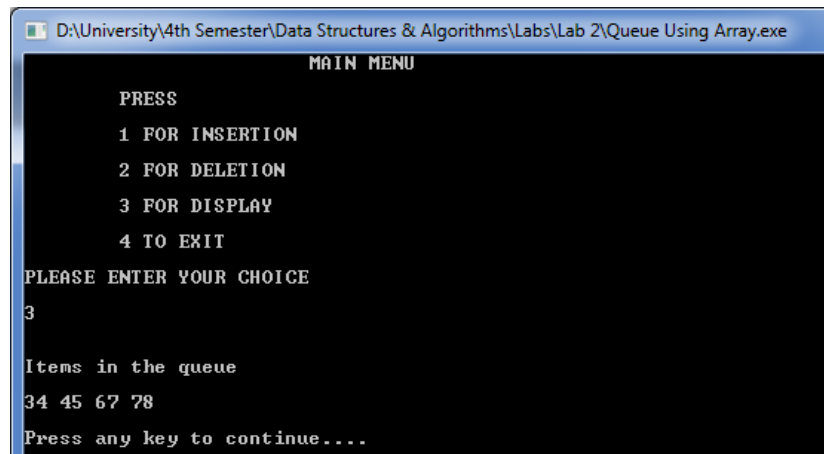


```

D:\University\4th Semester\Data Structures & Algorithms\Labs\Lab 2\Queue Using Array.exe
MAIN MENU
PRESS
1 FOR INSERTION
2 FOR DELETION
3 FOR DISPLAY
4 TO EXIT
PLEASE ENTER YOUR CHOICE
1
Enter a value: 34
Value inserted Successfully
Press any key to continue....

```

Figure 3: Main menu and insertion operation



```

D:\University\4th Semester\Data Structures & Algorithms\Labs\Lab 2\Queue Using Array.exe
MAIN MENU
PRESS
1 FOR INSERTION
2 FOR DELETION
3 FOR DISPLAY
4 TO EXIT
PLEASE ENTER YOUR CHOICE
3
Items in the queue
34 45 67 78
Press any key to continue....

```

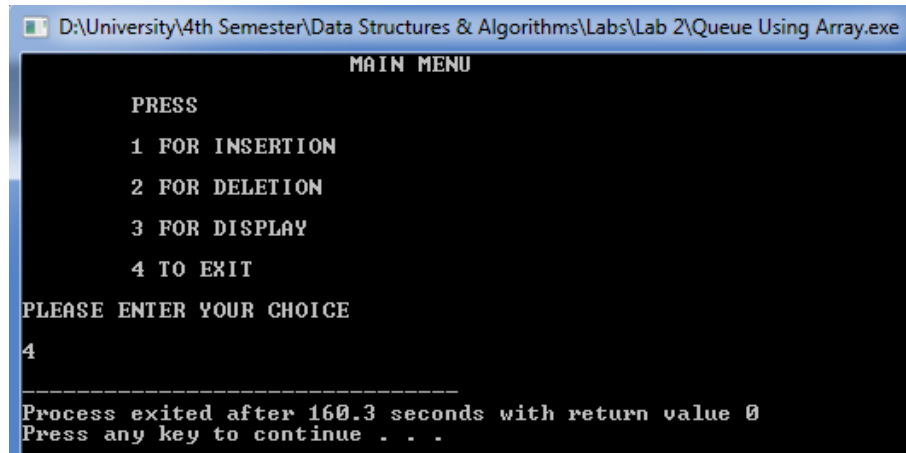
Figure 4: Displaying after insertion

```
D:\University\4th Semester\Data Structures & Algorithms\Labs\Lab 2\Queue Using Array.exe
MAIN MENU
PRESS
1 FOR INSERTION
2 FOR DELETION
3 FOR DISPLAY
4 TO EXIT
PLEASE ENTER YOUR CHOICE
2
Value deleted Successfully
Press any key to continue....
```

Figure 5: Deleting operation

```
D:\University\4th Semester\Data Structures & Algorithms\Labs\Lab 2\Queue Using Array.exe
MAIN MENU
PRESS
1 FOR INSERTION
2 FOR DELETION
3 FOR DISPLAY
4 TO EXIT
PLEASE ENTER YOUR CHOICE
3
Items in the queue
45 67 78
Press any key to continue....
```

Figure 6: Displaying after deletion



```
D:\University\4th Semester\Data Structures & Algorithms\Labs\Lab 2\Queue Using Array.exe
MAIN MENU
PRESS
1 FOR INSERTION
2 FOR DELETION
3 FOR DISPLAY
4 TO EXIT
PLEASE ENTER YOUR CHOICE
4
-----
Process exited after 160.3 seconds with return value 0
Press any key to continue . . .
```

Figure 7: Exit

Source Code:- <https://github.com/umerayan/Semester-4-Labs.git>

Conclusion:- Queue in computer science is same as we accustomed in everyday life. There are many examples of queue in everyday life i.e. in Banks, grocery store, and queues of cars waiting to pass through a toll-booth. In computers one of the best example of queues that computer sends a print request to the printer in a queue page by page. A queue is a First in First out data structure.

We implemented the queue program using the algorithm given in the manual. The program performs the every task related to queue in daily life or in computer. This program can be use to understand and fulfill the problem of queue in your software. The program is coded in C++ language and open source for everyone at my guthub account (The link mentioned above).

(Concerned Teacher/Lab Engineer)