Data Structure and Algorithms

Lab Report

Name:	Muhammad Arslan Ishaq
Registration #:	CSU-F16-112
Lab Report #:	02
Dated:	16-04-2018
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. Dev C++

1 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.
Whenever an element is deleted from the queue the value of FRONT is increased by one. This can be implemented by the assignment.
FRONT = 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.
REAR = 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 REAR = 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

Algorithm for insertion into the Queue: -

```
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: -

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.
```

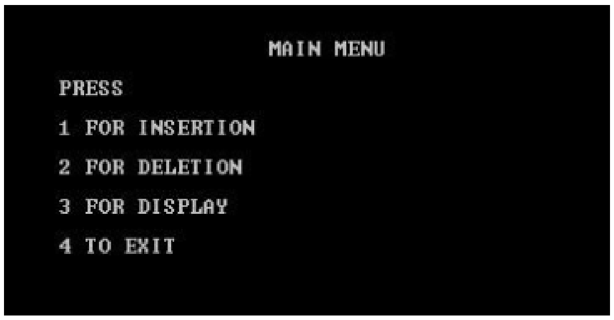
2 Lab Task

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.

2.1 Program

```
#include<iostream.h>
#include<conio.h>
int queue[100],front=-1,end=-1,max;
void insert_element();
void delete_element();
void display_queue();

void main()
{ clrscr();
int option;
cout<<"Implement Queue operations by-Arslan\n\n";
cout<<"Enter the size of Queue : ";
cin>>max;
do
{cout<<"1.Insert an element";
cout<<"\n2.Delete an element";
cout<<"\n3. Display␣queue";
cout<<"\n4. Exit";
cout<<"\nEnter␣your␣choice␣:␣";
cin>>option;
switch(option)
{ case 1: insert_element();
break;
case 2: delete_element();
break;
case 3: display_queue();
break;
case 4: return 0;
}
}while(option!=4);
getch();
}

void insert_element()
{
int num;
cout<<"\nEnter␣the␣item␣to␣be␣inserted␣:␣";
cin>>num;
if(front==0 && end==max-1)
cout<<"\nQueue␣OverFlow␣Occured\n";
else if(front==-1&&end==-1)
{
front=end=0;
queue[end]=num;

}
else if(end==max-1 && front!=0)
{
end=0;
queue[end]=num;
}
else
{
end++;
queue[end]=num;
}
```

```
}
void delete_element()
{
int element;
if(front==-1)
{
cout<<"\nUnderflow\n";
}
element=queue[front];
if(front==end)
front=end=-1;
else
{
if(front==max-1)
front=0;
else
front++;
cout<<"\nThe deleted element is : "<<element;
}

}
void display_queue()
{
int i;
if(front==-1)
cout<<"\nNo elements to display";
else
{
cout<<"\nThe queue elements are :\n";
for(i=front;i<=end;i++)
{
cout<<"\t"<<queue[i];
}
}
}
```



```
                MAIN MENU
PRESS
1 FOR INSERTION
2 FOR DELETION
3 FOR DISPLAY
4 TO EXIT
```

Figure : 1 Output

3 Conclusion

Queues are data structures that follow the First In First Out i.e. the first element that is added to the queue is the first one to be removed. Elements are always added to the back and removed from the front. Think of it as a line of people waiting for a bus. The person who is at the beginning of the line is the first one to enter the bus.