Lab Manual for Data Structures

Lab 5

Circular and Priority Queues

Table of Contents

Lab7: Circular and Priority Queues

1. Introduction

This lab will introduce concept of circular and priority queues. In standard queues when dequeue operation is performed a problem called de-bufferring occurs, to solve this problem we need to use circular queues. Beside the circular queues we may also use priority queues, in such queues we may delete an element from queue on the basis of some priority. In First in first out queues, the element at start of queue has highest priority to be removed from queue.

1.1 Circular Queues

Circular queuesare linear data structures. It follows FIFO (First In First Out) rule.

- In circular queue the last element is connected back to the first element to make a circle.
- Elements are inserted at the rear end and the elements are removedfrom front end of the queue.
- Both the front and the rear pointers point to the beginning of the array.
- It is also called as “Ring buffer”.

1.2 Priority Queues

Priority queues are used in such situations where you want to remove an element from queue which is not at the moment at front of queue, but its priority is high. For example in daily life if in some hospital there is a queue of patients waiting for checkup/treatment from doctor. If some patient comes whose have light threatening symptoms and need urgent treatment then he should be at high priority in that queue and should be removed from queue on his priority, may be patient at front of queue at the moment has come for general checkup.

A priority queue must at least support the following operations:

- insert_with_priority: add an element to the queue with an associated priority
- pull_highest_priority_element: remove the element from the queue that has the highest priority, and return it.

1.3Relevant Lecture Readings

a) Revise Lecture No. 13 and 14 available at \\dataserver\jinnah$\ in instructor’s folder.
b) From books: C++ Data Structures by Nell Dale (Page 225-245, 530-533) and Data structures using C++ by D. S. Malik (Page 455-462, 471-480).

2. Activity Time boxing

Table 1: Activity Time Boxing

Task No.	Activity Name	Activity time	Total Time
5.1	Design Evaluation	20mins	20mins
6.2	Walk through tasks	20mins	20mins
7	Practice tasks	20mins for task 1, 25mins for task 2 and 3.	70mins
9	Evaluation Task	60mins for all assigned tasks	60mins

3. Objectives of the experiment

- To understand implementation of circular queues in C++.
- To understand the implementation of priority queues in C++.
- To define and understand the ADT (Abstract Data Type) of priority queue in C++.

4. Concept Map

This concept map will help students to understand the main concepts of topic covered in lab.

4.1 Circular Queues in C++

Suppose we have given an array to implement a FIFO queue. We are using queueFront and queueRear two elements referring indices of this array. Now if we perform addition and removal of element on this array in such a way that after addition of three elements we remove one element. On performing addition queueRear will move to index of next element of array. When queueRear will point to last element of array after that it cannot use any other element of array for insertion of further element in this array. Figure 1 shows this scenario.

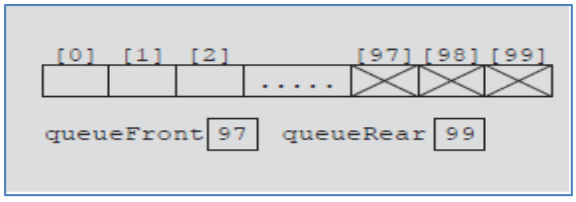

Figure 1: Rear cannot move further in array.

So we may use this array in circular fashion so that rear can move to starting element after reaching last element of array. Figure 2 represents the logical circular array which can be used.

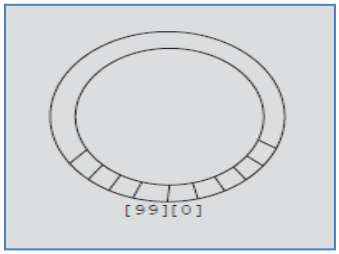

Figure 2: Logical representation of a circular array (Ring Buffer).

Because the array containing the queue is circular, we can use the following statement to advance queueRear (queueFront) to the next array position:

```
queueRear = (queueRear + 1) % maxQueueSize;
```

If `queueRear<maxQueueSize - 1`,
Then`queueRear + 1 <= maxQueueSize - 1`,
so`(queueRear + 1) % maxQueueSize = queueRear + 1.`

If`queueRear == maxQueueSize- 1`

(That is, queueRear points to the last array position),

```
queueRear + 1 ==maxQueueSize,
```

So `(queueRear + 1) % maxQueueSize = 0`. In this case, queueRear will be set to 0, which is the first array position.

4.2Priority Queues in C++

Following is implementation example of a priority queue in C++. Following are some definitions which will be used by this priority queue.

```
const int MAX=5;

struct data
{
char job[MAX];
int prno ;
int ord ;
} ;

struct pque
{
data d[MAX];
int front ;
int rear ;
} ;
```

Following function is used to initialize the queue attributes.

```
void initpque ( pque *pq )
{
int i ;

pq -> front = pq -> rear = -1 ;
for ( i = 0 ; i < MAX ; i++ )
```

```
    {
strcpy ( pq -> d[i].job, '\0' ) ;
pq -> d[i].prno = pq -> d[i].ord = 0 ;
    }
}
```

To add an item in this priority queue, following function is used.

```
void add (pque *pq, data dt )
{
data temp ;
int i, j ;

if ( pq -> rear == MAX - 1 )
    {
cout"\nQueue is full."<<endl;
return ;
    }

pq -> rear++ ;
pq -> d[pq -> rear] = dt ;

if ( pq -> front == -1 )
pq -> front = 0 ;

for ( i = pq -> front ; i <= pq -> rear ; i++ )
    {
for ( j = i + 1 ; j <= pq -> rear ; j++ )
        {
if ( pq -> d[i].prno>pq -> d[j].prno )
            {
temp = pq -> d[i] ;
pq -> d[i] = pq -> d[j] ;
pq -> d[j] = temp ;
            }
else
            {
if ( pq -> d[i].prno == pq -> d[j].prno )
                {
if ( pq -> d[i].ord>pq -> d[j].ord )
                    {
temp = pq -> d[i] ;
pq -> d[i] = pq -> d[j] ;
pq -> d[j] = temp ;
                    }
                }
            }
```

```
        }
    }
}
```

To remove an element from priority queue, following function is used.

```
data  delete (pque *pq )
{
data  t ;
strcpy ( t.job, "" ) ;
t.prno = 0 ;
t.ord = 0 ;

if ( pq -> front == -1 )
    {
cout<<"\nQueue is Empty."<<endl;
return t ;
    }

t = pq ->d[pq -> front] ;
pq -> d[pq -> front] = t ;
if ( pq -> front == pq -> rear )
pq -> front = pq -> rear = -1 ;
else
pq -> front++ ;

return  t ;
}
```

5. Homework before Lab

This homework will help students to study the concepts of topic before start of lab.

5.1 Problem Solution Modeling

After studying the introduction and concept map sections you should be ready to provide the solution of following problems. Design the solutions of the problems in C++ and bring your code to lab so that lab instructor should assess and grade it.

5.2Problem description:

Write a program to implement a priority queue to store the objects of “person” class. “person” class should contain attributes (privately defined) per_ssn (string), per_name (string), per_age (int) and per_priority(int). “person” class should also contain member functions (publicly defined); constructor, input and output functions. You are required to design a priority queue class which should use static array implementation. Your queue class should contain constructor, destructor (if required), enqueue, dequeue, displayElements operations and conditions to check empty or full queue. The member

function dequeue should remove an object of person class which should have high priority number than that of object at front of queue.

5.3Practices from home

5.3.1 Task-1

Compare FIFO circular queue with simple FIFO queue using static array and provide at least three differences between these two implementations.

5.3.2 Task-2

List three real world examples in which priority queue structure is used.

6. Procedure& Tools

This section provides information about tools and programming procedures used for the lab.

6.1 Tools

Microsoft Visual Studio 2017 with Visual C++ compiler configured.

6.2 Walk through Tasks [Expected time = 20mins]

By using Microsoft Visual Studio 2017, we can implement a program for a priority queue using static array. Following figure 3 represent the source code view of this program in visual studio. You are required to type this code and further debug and execute the program.

```
#include<iostream>
using namespace std;
const int MAX=5;
struct data
        {
            int val,p;
        }d[MAX];

class pqueue
{
                  int front,rear;
                public:

                 pqueue()
                 {
                       front=rear=-1;
                 }
                 void insert(data d1);
                 data deletion();
                 void display();
};
        void pqueue :: insert(data d1)
        {
                if(rear==MAX-1)
                   cout<<"Priority Queue is Full";
                else
                {
                   rear++;
                   d[rear]=d1;
                   if(front==-1)
                          front=0;
                   data temp;
                   for(int i=front;i<=rear;i++)
                         for(int j=i+1;j<=rear;j++)
                         {
                                 if(d[i].p > d[j].p)
                                 {
                                        temp=d[i];
                                        d[i]=d[j];
                                        d[j]=temp;
                                 }

                         }
                }
        }
```

Figure 3: Implementation of a priority queue.

```
data pqueue :: deletion()
{
        data d1;
        if(front==-1)
           cout<<"Priority Queue is Empty";
        else
        {
           d1=d[front];
           if(front==rear)
                  front=rear=-1;
           else
                  front++;
        }
        return d1;
}
void pqueue :: display()
{
         if(front==-1)
                cout<<"Priority Queue is Empty";
         else
         {
                for(int i=front;i<=rear;i++)
                {
                        cout<<"Object  :"<<i+1<<endl;
                        cout<<"Value ="<<d[i].val<<endl;
                        cout<<"Priority="<<d[i].p<<endl;
                }
         }
}
```

Figure 4: Implementation of a priority queue.

```
void main()
{
  pqueue p1;

  data d1;
  char op;
  do
  {
        int ch;
        cout<<"----------Menu-------------";
        cout<<"1.Insertion 2.Deletion 3.Display 4.Exit"<<endl;
        cout<<"Enter your Choice<1..4> ?";
        cin>>ch;
        switch(ch)
        {
           case 1 :  cout<<"Enter Value ?";
                                cin>>d1.val;
                                cout<<"Enter Priority?";
                                cin>>d1.p;
                                p1.insert(d1);
                                break;
           case 2 :  d1=p1.deletion();
                                cout<<"Value = "<<d1.val<<endl;
                                cout<<"Priority = "<<d1.p<<endl;
                                break;
           case 3 :  p1.display();
                                break;
          }
          cout<<"Do You Want to Continue <Y/N> ?";
          cin>>op;
        }while(op=='Y' || op=='y');
  }
```

Figure 5: Implementation of a priority queue.

Figure 6 shows output of program in figure 3, figure 4 and figure 5.

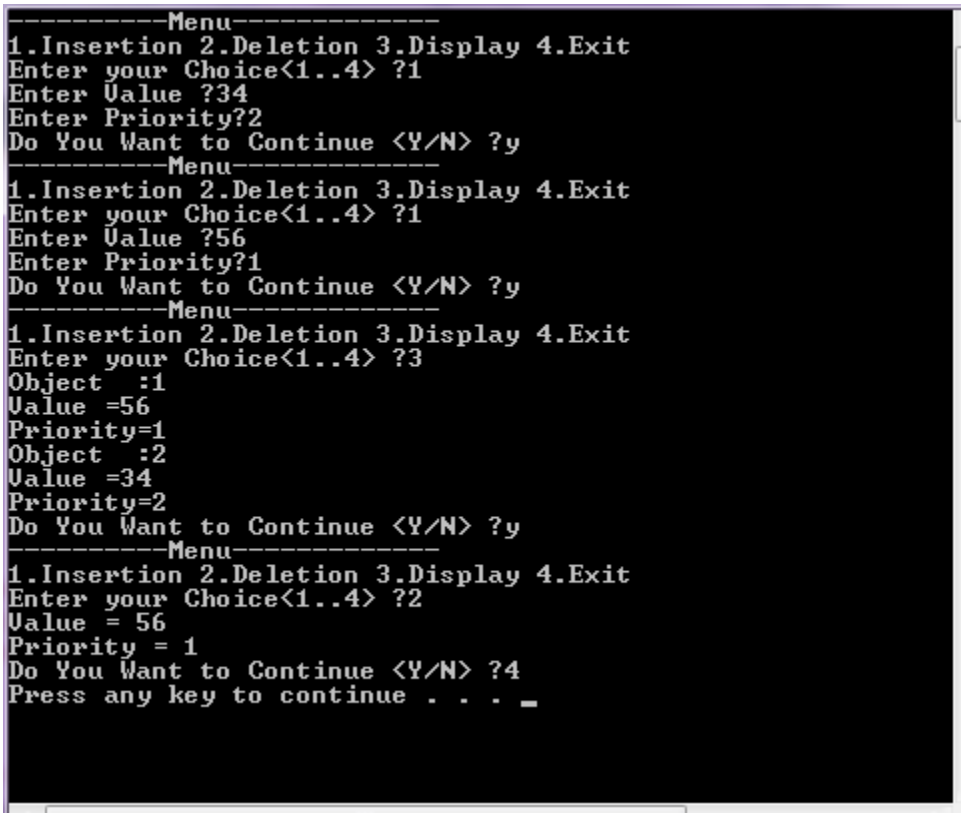

Figure 6: Output of program shown in figure 3, 4 and 5

7. Practice Tasks

This section will provide information about the practice tasks which are required to be performed in lab session. Design solutions of problems discussed in each task and placesolution code in a folder specified by your lab instructor.

Lab instructors are guided to help students learn how ACM problems work and provide students with certain practice/home tasks on the pattern of ACM Problems.

7.1 Practice Task 1 [Expected time = 20 mins]

Write a program which should implement a circular queue using static array of size 10 (10 elements array), elements of array should be of integer (int) type. User will input values to be inserted at rear of circular queue (enqueue) and also number of elements to be removed from front of circular queue (dequeue). Your program should display the value of elements which are being removed from circular queue. Program should also calculate and display the average of elements which have been removed from circular queue. Implement the task using template class.

7.2 Practice Task 2 [Expected time = 30 mins]

Implement a priority queue for interviewing candidates for a job position as Lecturer at CUST. University has taken a written test and candidates have secured marks out of 50 in it. Now candidates give his information (CV) and along with the marks in written test. You have to prioritize the candidates on the basis of Experience (must be more than 3 years), Qualification (must have done MS) and Written_test_marks (More marks high priority). Whenever panel calls a candidate the person whose priority is highest must give the interview. Your program should display all the candidates in the priority queue as well.

7.3 Practice Task 3 [Expected time = 30mins]

Write a program to implement Bill_paying Queue. Your Queue should store the Person Name, CNIC and Age. Note that whenever a senior citizen comes in he/she should be given priority in the queue.

7.4Out comes

After completing this lab, student will be able to understand and develop programs related to circular and priority queues using static arrays in C++ using Microsoft Visual Studio 2017 environment.

7.6Testing

Test Cases for Practice Task-1

Sample Input	Sample Output
Insert elements in rear of circular queue:	
2	
3	
4	
5	
7	
8	
6	
9	
11	
1	
Enter number of elements to be removed from circular queue: 4	2
	3
	4
	5
	Average of removed values is: 3.5

Test Cases for Practice Task-2

Sample input/output:

Press 1 to enter Information
Press 2 to call for interview
Press 3 to view all Candidates
Enter choice: 1
Enter Name: Ali
Enter CNIC: XXXXX-XXXXXXX-X
Enter Qualification: MS
Enter Experience: 4
Enter Your Marks in Written Test: 35

Press 1 to enter Information
Press 2 to call for interview
Press 3 to view all Candidates
Enter choice: 1
Enter Name: Osama
Enter CNIC: XXXXX-XXXXXXX-X
Enter Qualification: MS
Enter Experience: 4
Enter Your Marks in Written Test: 40

Press 1 to enter Information
Press 2 to call for interview
Press 3 to view all Candidates
Enter choice: 1
Enter Name: Hamza
Enter CNIC: XXXXX-XXXXXXX-X
Enter Qualification: BS
Enter Experience: 4
Enter Your Marks in Written Test: 40

Press 1 to enter Information
Press 2 to call for interview
Press 3 to view all Candidates
Enter choice: 2
Osama please come in 5 minutes for interview

Press 1 to enter Information
Press 2 to call for interview
Press 3 to view all Candidates
Enter choice: 3
Osama
MS
4
40

Ali
MS
4
35
Hamza
BS
4
40

Test Cases for Practice Task-3

Sample input/output:

Press 1 to enter Information
Press 2 to call for bill Payment
Press 3 to view all persons
Enter choice: 1
Enter your Name: Firdous
Enter your CNIC: XXXXX-XXXXXXX-X
Enter your Age: 35

Press 1 to enter Information
Press 2 to call for bill Payment
Press 3 to view all persons
Enter choice: 1
Enter your Name: Ahmed Ali
Enter your CNIC: XXXXX-XXXXXXX-X
Enter your Age: 50

Press 1 to enter Information
Press 2 to call for bill Payment
Press 3 to view all persons
Enter choice: 1
Enter your Name: Zulfiqar
Enter your CNIC: XXXXX-XXXXXXX-X
Enter your Age: 40

Press 1 to enter Information
Press 2 to call for bill Payment
Press 3 to view all persons
Enter choice: 2
Ahmed Ali please come for billpayment.

8. Evaluation Task (Unseen) **[Expected time = 60mins for two tasks]**

The lab instructor will give you unseen task depending upon the progress of the class.

9. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 2: Evaluation of the Lab

Sr. No.	Task No	Description	Marks
1	4	Problem Modeling	20
2	6	Procedures and Tools	10
3	7,8	Practice tasks and Testing	35
4	8.1	Evaluation Tasks (Unseen)	20
5		Comments	5
6		Good Programming Practices	10

10. Further Readings

10.1 Web sites related to C++ tutorials related to circular and priority queues

1. http://comsci.liu.edu/~jrodriguez/cs631sp08/c++priorityqueue.html
2. http://www.cplusplus.com/reference/queue/priority_queue/

10.2 Web sites containing supporting material

1. http://web.eecs.utk.edu/~bvz/cs302/notes/PQueues/
2. https://www.tutorialspoint.com/cplusplus-program-to-implement-priority-queue