Lab Report 08

Submitted by

Muhammad Awais / 2883

Submitted To

Ma'am Zanaib Malik

Department of Computer Science and Engineering

National University of modern language, Islamabad

List of Tasks

1. Class PNode
2. Class PQueue
3. Enqueue
4. Dqueue
5. isEmpty
6. rearValue
7. front Value
8. Delete all in one Go

Source Code PNode.h

```
#include <iostream>
#include "E:\BSCS 3rd Semester\DSA\lab code\L3P3 Singular link List\node.h"
using namespace std;
template <class t>
class PNode : public Node<t>
{
private:
    int prio;

public:
    PNode(int prio, t data, Node<t> *next) : Node<t>(data,next)
    {
        this->prio = prio;
    }

    void setprio(int prio);
    int getprio();
    void displayPNode();

}; // End of PNode class

template <class t>
void PNode<t>::setprio(int prio)
{
    this->prio = prio;
} // End

template <class t>
int PNode<t>::getprio()
{
    return this->prio;
} // end

template <class t>
void PNode<t>::displayPNode()
{
    cout << this->data << "\t";
    cout << this->next << "\t";
    cout << this->prio;
}
```

Description of PNode

- The code defines a class **PNode** that is derived from the **Node** class.
- **PNode** is a template class that takes a generic data type **t**.
- The class has a private member variable **prio** of type **int** to store the priority value.
- The constructor of **PNode** takes three parameters: **prio**, **data**, and **next**. It calls the base class (**Node**) constructor to initialize the data and next pointers.
- The **setprio()** function is a member function that sets the priority value of the **PNode** object.
- The **getprio()** function is a member function that returns the priority value of the **PNode** object.
- The **displayPNode()** function is a member function that displays the data, next pointer, and priority value of the **PNode** object.

Source Code PQueue.h

```
#include <iostream>
#include "PNode.h"
using namespace std;
template <class t>
class PQueue
{
private:
    PNode<t> *front; // head
    PNode<t> *rear;  // tail
```

```
public:
    PQueue()
    {
        rear = 0;
        front = 0;
    }

    void Enqueue(int prio, t element);
    t Dqueue();
    bool isEmpty();
    t rearvalue();
    t frontValue();
    t allDelete();
}; // End of Class

template <class t>
void PQueue<t>::Enqueue(int prio, t element)
{
    PNode<t> *n = new PNode<t>(prio, element, 0);

    if (rear == 0 && front == 0)
    {
        rear = front = n;
    }

    else if (n->getprio() < rear->getprio()) // rear/tail modify or rear has
greater priority than n
    {
        rear->setNext(n);
        rear = n;
    }

    else if (n->getprio() > front->getprio()) // front/head modify or front has
less priority than n
    {
        rear->setNext(n);
        rear = n;
    }

    else // in between case
    {
        PNode<t> *ptr = front;
        while (((PNode<t> *)ptr->getNext())->getprio() >= n->getprio())
        {
            ptr = (PNode<t> *)ptr->getNext();
        }
        ptr->setNext(n);
        n->setNext(ptr->getNext());
    }
} // End

template <class t>
t PQueue<t>::Dqueue()
{
     t element;
    if (rear == 0 && front == 0)
    {
        cerr << "Queue is empty \n";

    }

    else if (rear == front)
    {
        element = front->getInfo();
        delete front;
        front = rear = 0;

    }
    else
    {
        PNode<t> *temp = front;
         element =front->getInfo();
        front = (PNode<t> *)front->getNext();
        delete temp;

    }
  return element;
} // end Dequeue

template<class t>
bool PQueue<t>::isEmpty()
{
    if(rear==0 && front==0)
    return true;

    else
    return false;
}

template<class t>
```

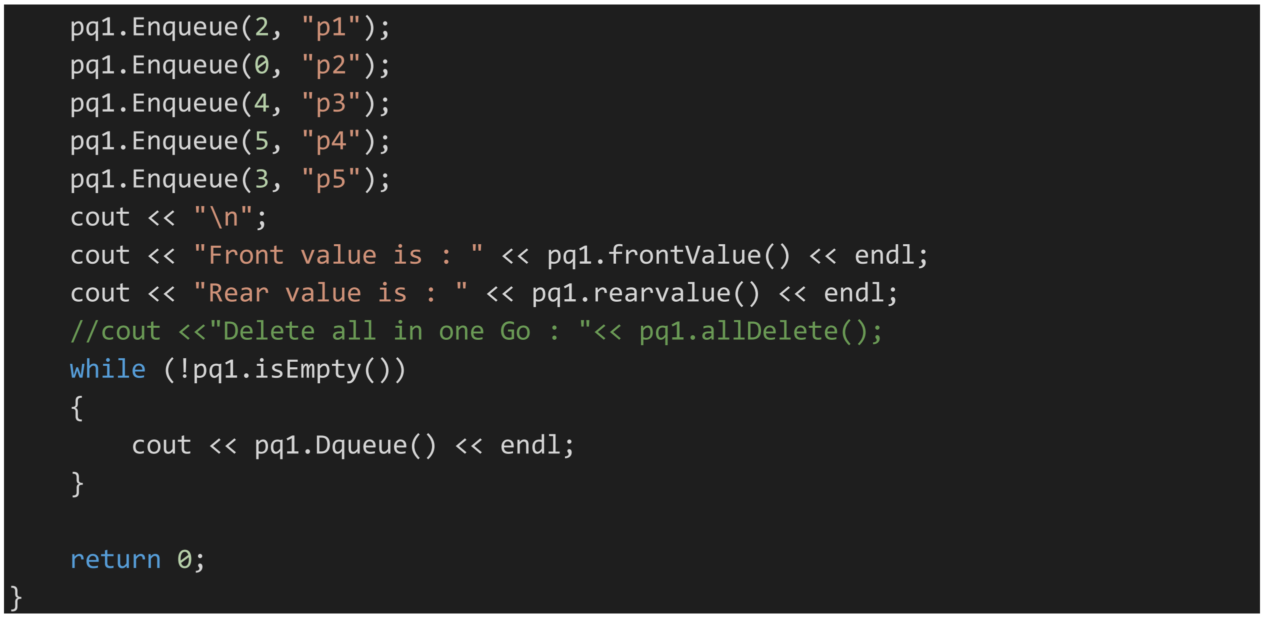
```
t PQueue<t>::rearvalue()
{
    if(rear==0 && front==0)
{
        cerr<<"Queue is empty \n";
}
    else
    {
        t element = rear->getInfo();

    return element;
    }
}

template<class t>
t PQueue<t>::frontValue()
{
    if(rear==0 && front==0)
    {
    cerr<<"Queue is empty \n";
}
    else
    {
        t element = front->getInfo();
     return element;
    }
}

template<class t>
t PQueue<t>::allDelete()
{
    if(rear==0 && front==0)
    {
    cerr<<"Queue is empty \n";
  }

    PNode<t> *ptr=front;
    t element;
    while(ptr!=NULL)
    {
        element+=Dqueue();
        ptr=(PNode<t>*)ptr->getNext();
    }
    return element;
}
```

Description of PQueue.h

- The code defines a class **PQueue** which represents a priority queue.
- The class uses a linked list implementation using **PNode** objects.
- The **PQueue** class is a template class that takes a generic data type **t**.
- The class has two private member variables: **front** and **rear**, which represent the head and tail of the priority queue, respectively.
- The constructor initializes **front** and **rear** to 0, indicating an empty priority queue.
- The **Enqueue()** function adds an element with a given priority to the priority queue.
- The **Dequeue()** function removes and returns the element with the highest priority from the priority queue.
- The **isEmpty()** function checks if the priority queue is empty and returns a boolean value indicating its status.
- The **rearvalue()** function returns the value of the element with the highest priority without removing it from the priority queue.
- The **frontValue()** function returns the value of the element with the lowest priority without removing it from the priority queue.
- The **allDelete()** function removes all elements from the priority queue and returns their sum.

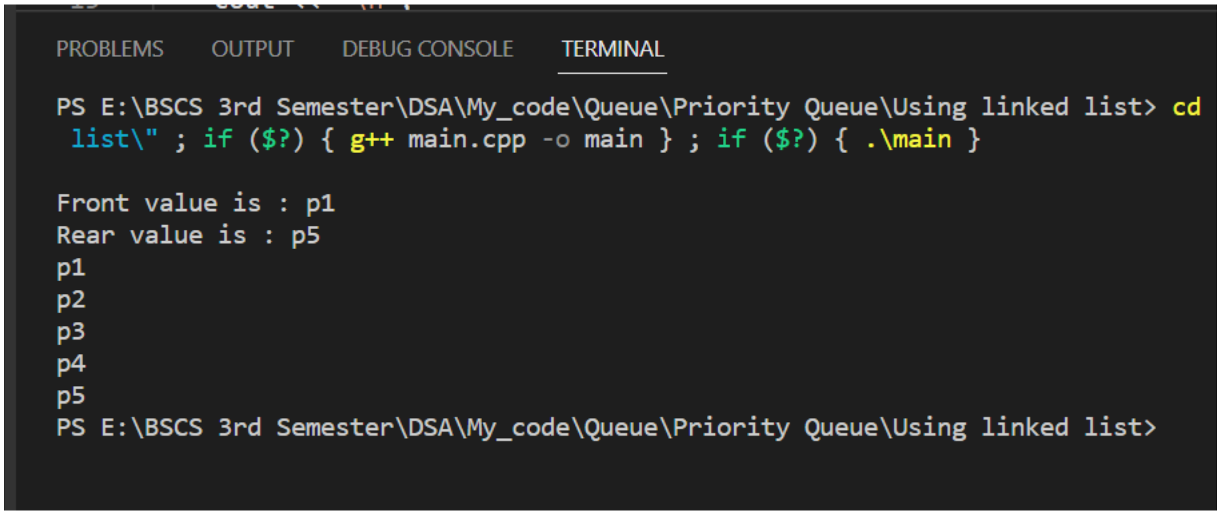
Source Code main class

```
#include <iostream>
#include "PQueue.h"
using namespace std;
/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

int main(int argc, char **argv)
{

    PQueue<string> pq1;
```



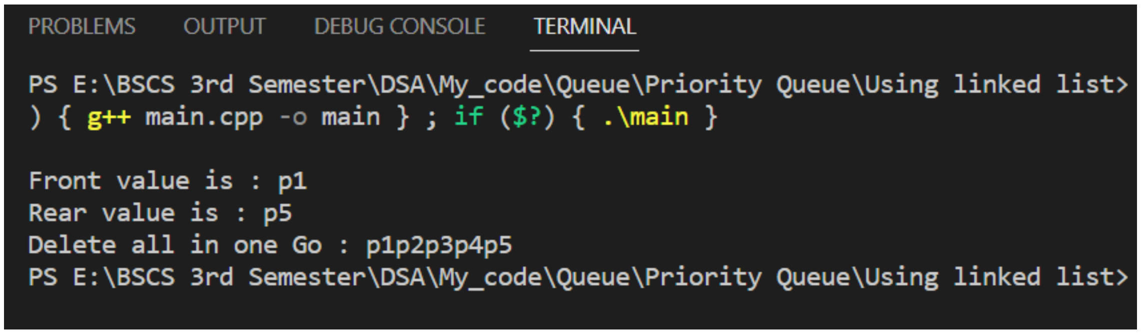
```
    pq1.Enqueue(2, "p1");
    pq1.Enqueue(0, "p2");
    pq1.Enqueue(4, "p3");
    pq1.Enqueue(5, "p4");
    pq1.Enqueue(3, "p5");
    cout << "\n";
    cout << "Front value is : " << pq1.frontValue() << endl;
    cout << "Rear value is : " << pq1.rearvalue() << endl;
    //cout <<"Delete all in one Go : "<< pq1.allDelete();
    while (!pq1.isEmpty())
    {
        cout << pq1.Dqueue() << endl;
    }

    return 0;
}
```

Output



```
PS E:\BSCS 3rd Semester\DSA\My_code\Queue\Priority Queue\Using linked list> cd
 list\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }

Front value is : p1
Rear value is : p5
p1
p2
p3
p4
p5
PS E:\BSCS 3rd Semester\DSA\My_code\Queue\Priority Queue\Using linked list>
```

If Call allDelete();



```
PS E:\BSCS 3rd Semester\DSA\My_code\Queue\Priority Queue\Using linked list>
) { g++ main.cpp -o main } ; if ($?) { .\main }

Front value is : p1
Rear value is : p5
Delete all in one Go : p1p2p3p4p5
PS E:\BSCS 3rd Semester\DSA\My_code\Queue\Priority Queue\Using linked list>
```

End of Lab 08