

12/28/2021

**Department Of Computer Science**

**Subject** **Instructor:** Ma’am Zainab Malik

**Assignment:** 3 **Date:** 28-12-2021

**Class:** BSCS-3B

**Submitted by:**

Madina Javed Iqbal Khokhar 2426

**Task**

**Assigned Task:**

**Design and develop menu based program for following data structures with**

**given functions.**

**Linear Queue**

* **Enqueue Function**
* **Dequeue Function**
* **DisplayAll Function**
* **RearValue Function**
* **FrontValue Function**

• **Circular Queue**

* **Enqueue Function**
* **Dequeue Function**
* **DisplayAll Function**
* **RearValue Function**
* **FrontValue Function**

**Priority Queue**

* **Enqueue Function**
* **Dequeue Function**

**Code:**

**Linear Queue:**

**Node Class:**

#include<iostream>

using namespace std;

template <class T>

class node{

public:

T i; // data

node<T> \*next;

node(T j)

{

i = j;

next = NULL;

}

};

**Linear Queue Class:**

#include<iostream>

#include"Node.h"

using namespace std;

template <class T>

class queue{

node<T> \*start;

node<T> \*end;

public:

queue()

{

start = end = NULL;

}

bool empty()

{

return start==NULL;

}

void push(T i)

{

node<T> \*temp = new node<T>(i);

if(empty())

{

start = end = temp;

}

else

{

end->next = temp;

end = temp;

}

}

T front()

{

if(empty())

return NULL;

else

return start->i;

}

void pop()

{

if(empty())

{

cout<<"Queue is Empty"<<endl;

}

else if(start==end)

{

// In case there is only 1 element in the queue

delete start;

start = end = NULL;

}

else

{

// for more one more than 1 element in the queue

node<T> \*temp = start;

start = start->next;

delete temp;

}

}

};

**Main Function:**

#include <iostream>

#include"LinearQueue.h"

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

queue<string> q;

q.push("DSA");

q.push("Assignment");

cout<<"Queue Front: "<<q.front()<<endl;

q.push("I ");

q.push("DO");

q.push("NOT ");

q.push("LIKE");

q.push(" DSA");

while(!q.empty()){

cout<<q.front()<<' ';

q.pop();

}

return 0;

}

**Circular Queue:**

#include <iostream>

using namespace std;

int cqueue[5];

int front = -1, rear = -1, n=5;

void insertCQ(int val) {

if ((front == 0 && rear == n-1) || (front == rear+1)) {

cerr<<” Overflow \n";

return;

}

if (front == -1) {

front = 0;

rear = 0;

} else {

if (rear == n - 1)

rear = 0;

else

rear = rear + 1;

}

cqueue[rear] = val ;

}

void deleteCQ() {

if (front == -1) {

cerr<<" Underflow\n";

return ;

}

cout<<"Deleted elements are : "<<cqueue[front]<<endl;

if (front == rear) {

front = -1;

rear = -1;

} else {

if (front == n - 1)

front = 0;

else

front = front + 1;

}

}

void displayCQ() {

int f = front, r = rear;

if (front == -1) {

cout<<" Our queue is empty"<<endl;

return;

}

cout<<"Our queue elements are :"<<endl;

if (f <= r) {

while (f <= r){

cout<<cqueue[f]<<" ";

f++;

}

} else {

while (f <= n - 1) {

cout<<cqueue[f]<<" ";

f++;

}

f = 0;

while (f <= r) {

cout<<cqueue[f]<<" ";

f++;

}

}

cout<<endl;

}

int main() {

int ch, val;

cout<<"1)Insert\n";

cout<<"2)Delete\n";

cout<<"3)Display\n";

cout<<"4)Exit\n";

do {

cout<<"Enter choice : "<<endl;

cin>>ch;

switch(ch) {

case 1:

cout<<"Inserted data is: "<<endl;

cin>>val;

insertCQ(val);

break;

case 2:

deleteCQ();

break;

case 3:

displayCQ();

break;

case 4:

cout<<"Exit";

break;

default: cout<<"Is not correct!\n";

}

} while(ch != 4);

return 0;

}//circular

**Priority Queue:**

**Node Class:**

#include<iostream>

using namespace std;

template<class T>

Class PriNode{

private:

T info;

PriNode<T> \*next;

int priority ;

public:

PriNode(T i=0, int p=0 , PriNode<T> \*n=0):info(i), priority(p),next(n)

{

// constructor…..

}

void setInfo(T i);

T getInfo();

void setNext(PriNode<T> \*n);

PriNode<T>\* getNext();

void setpriority(int pri);

int getpriority();

};//EOC

template<class T>

void PriNode<T>::setInfo(T i)

{

info=i;

}

template<class T>

void PriNode<T>::setNext(PriNode<T> \*n)

{

next=n;

}

template<class T>

void PriNode<T>::setpriority(int pri)

{

priority=pri;

}

template<class T>

T PriNode<T>::getInfo()

{

return info;

}

template<class T>

int PriNode<T>::getpriority()

{

return priority ;

}

template<class T>

PriNode<T>\* PriNode<T>::getNext()

{

return next;

} // Pirority Node Class

**Link List Class:**

#include<iostream>

#include"PriNode.h"

using namespace std;

template<class T>

LinkedQP{

private :

PriNode<T> \*rear;

PriNode<T> \*front;

public :

LinkedQP() // constructor

{

rear=0;

front=0;

}

void enqueue(T item , int pri);

T Dequeue();

void display();

};

template<class T>

void LinkedQP<T>::enqueue(T item , int pri)

{

PriNode<T> \*n1=new PriNode<T>(item,pri);

if(rear==0&&front==0)

{

rear=front=n1; // addedd first node in pri queue

}

else // already exist some nodes .....

{

if(pri<front->getpriority())

{

n1->setNext(front);

front=n1;

}

else if(pri>=rear->getpriority())

{

rear->setNext(n1);

rear=n1;

}

else

{

PriNode<T> \*ptr=front;

while(ptr->getpriority()<=pri)

{

ptr=ptr->getNext();

}

n1->setNext(ptr->getNext());

ptr->setNext(n1);

}

}

} // enqueue function

template<class T>

T LinkedQP<T>::Dequeue() // it will be same as removefrom head

{

if(front==0) // queue is empty

cout<<"nothing to delete"<<endl;

else if(rear==front) // only one element present

{

PriNode<T> \*temp=front;

T tempinfo=temp->getInfo();

delete temp;

front=rear=0;

return tempinfo;

}

else / / In case more than one element ..

{

PriNode<T> \*temp=front;

front=front->getNext();

T tempinfo=temp->getInfo();

delete temp;

return tempinfo;

}

}

template<class T>

void LinkedQP<T>::display()

{

PriNode<T> \*ptr=front;

while(ptr!=0)

{

cout<<ptr->getInfo()<<" ";

ptr=ptr->getNext();

}

} //traversing

**Main Function:**

#include <iostream>

#include"LinkedQP.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) {

LinkedQP<int> q; // object of priority queue class

q.enqueue(1,0);

q.enqueue(2,1);

q.display();

q.enqueue(3,2) ;

q.enqueue(4,1);

cout<<q.Dequeue()<<endl;

return 0;

}

***THANKS***