

DATa structures lab

Lab-7



Name: Talib Husian

Roll# 21F-9070

**Task-1**

#include <iostream>

using namespace std;

//Making a Queue ADT

class Queue {

    int \*arr;

    int size;

    int front;

    int rear;

public:

//Constructor to initialze with NULL

    Queue(int size = 10) {

        this->size = size;

        arr = new int[this->size];

        front = -1;

        rear = -1;

    }

    //Functon to Insert a data in a queue

    void enqueue(int x)

    {

        if(!isFull()){

        rear++;

        arr[rear] = x;

        }

        else{

            cout<<"Queue is FUll\n";

        }

    }

    // Function to remove

    int dequeue()

    {

        if(front<size){

        front++;

        return arr[front];

        }

        return -1;

    }

    void display(){

       for (int i=front+1;i<=rear;i++){

        cout<<arr[i]<<" ";

       }

       cout<<endl;

    }

    bool isFull(){

        if(rear==size-1){

            return true;

        }

        return false;

    }

    bool isEmpty(){

        if(rear == -1){

            return true;

        }

        else{

            return false;

        }

    }

};

int main()

{

    Queue q;

    q.enqueue(1);

    q.enqueue(2);

    q.enqueue(3);

    q.enqueue(4);

    q.enqueue(6);

    q.enqueue(7);

    q.enqueue(8);

    q.enqueue(9);

    q.enqueue(10);

    cout<<"Dequeuing Data\n";

    cout << q.dequeue() << endl;

    cout << q.dequeue() << endl;

    cout << q.dequeue()<<endl; cout<<"\nDisplaying a queue\n";

    q.display();

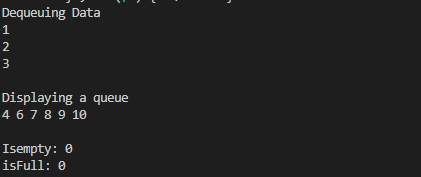
    cout<<"\nIsempty: "<<q.isEmpty();

    cout<<"\nisFull: "<<q.isFull();

    return 0;

}

**Output**

****

**Task-2**

#include <iostream>

using namespace std;

//making a queue ADT

class Queue

{

    int rear;

    int front;

    int\* arr;

    int size;

public:

//constructor

    Queue(int s =5)

    {

        front = -1;

        rear = -1;

        size = s;

        arr = new int[size];

    }

    // Checking if the queue is empty

    bool isEmpty()

    {

        if (front == -1)

            return true;

        else

            return false;

    }

    // checking if the queue is full

    bool isFull()

    {

        if (front == 0 && rear == size - 1)

        {

            return true;

        }

        if (front == rear + 1)

        {

            return true;

        }

        return false;

    }

    // inserting data in the queue

    void enqueue(int element)

    {

        if (isFull())

        {

            cout << "The quueue is full.\n";

        }

        else

        {

            if (front == -1)

                front = 0;

            rear = (rear + 1) % size;

            arr[rear] = element;

        }

    }

    // Removing data

    int dequeue()

    {

        int element;

        if (isEmpty())

        {

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

            return -1;

        }

        else

        {

            element = arr[front];

            if (front == rear)

            {

                front = -1;

                rear = -1;

            }

            else

            {

                front = (front + 1) % size;

            }

            return (element);

        }

    }

    // Displaying all elements in the queue

    void display()

    {

        if (isEmpty()){

            cout << "\nQueue is empty" << endl;

        }

        else{

            int i;

            for (i = front; i != rear; i = (i + 1) % size){

               cout << arr[i]<<" ";

            }

    }

}

};

int main()

{

    Queue q(5);

    q.enqueue(1);

    q.enqueue(2);

    q.enqueue(3);

    q.enqueue(4);

    q.enqueue(5);

    q.enqueue(6);

    cout << "Dequeuing Data\n";

    cout << q.dequeue() << endl;

    cout << q.dequeue() << endl;

    cout << q.dequeue() << endl;

    cout << "\nDisplaying a queue\n";

    q.display();

    q.enqueue(6);

    q.enqueue(7);

    cout << "\nDisplaying a queue\n";

    q.display();

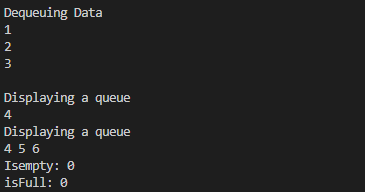
    cout << "\nIsempty: " << q.isEmpty();

    cout << "\nisFull: " << q.isFull();

    return 0;

}

**Output**

****

**Task-3**

#include <iostream>

using namespace std;

//creating a Node for queue

struct Node {

    int data;

    Node\* next;

};

//Making a Queue ADT

class Queue {

    Node\* front;

    Node\* rear;

public:

//Constructor to initialze with NULL

    Queue() {

        front = NULL;

        rear = NULL;

    }

    //Functon to Insert a data in a queue

    void enqueue(int x)

    {

        // Create new node

        Node\* temp = new Node;

        temp->data = x;

        temp->next = NULL;

        // new node is front and rear both If queue is empty

        if (rear == NULL) {

            front = temp;

            rear = temp;

        }

        else {

            // Add the new node at end

            rear->next = temp;

            rear = temp;

        }

    }

    // Function to remove

    int dequeue()

    {

        // If queue is empty, return NULL

        if (front == NULL) {

            cout << "Queue is Empty";

            return NULL;

        }

        else{

        Node\* temp = front;

        front = front->next;

        if (front == NULL)

            rear = NULL;

        int data = temp->data;

        delete temp;

        temp = NULL;

        return data;

        }

    }

    void display(){

        Node \*temp = front;

        while (temp!=NULL)

        {

            cout<<temp->data<<" ";

            temp = temp->next ;

        }

        cout<<endl;

    }

    bool isEmpty(){

        if(front == NULL){

            return true;

        }

        else{

            return false;

        }

    }

};

int main()

{

    Queue q;

    q.enqueue(1);

    q.enqueue(2);

    q.enqueue(3);

    q.enqueue(4);

    q.enqueue(5);

    cout<<"Dequeuing Data\n";

    cout << q.dequeue() << endl;

    cout << q.dequeue() << endl;

    cout << q.dequeue()<<endl;

    cout<<"\nDisplaying a queue\n";

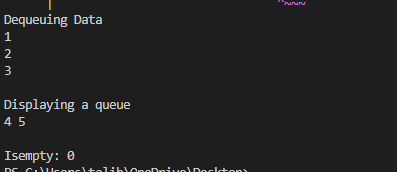
    q.display();

    cout<<"\nIsempty: "<<q.isEmpty();

    return 0;

}

**Output**



**Task-4**

#include <iostream>

#include <windows.h>

using namespace std;

//creating a Node for queue

struct Node {

    int data;

    Node\* next;

};

//Making a Queue ADT

class Queue {

public:

    Node\* front;

    Node\* rear;

public:

    //Constructor to initialze with NULL

    Queue() {

        front = NULL;

        rear = NULL;

    }

    //Functon to Insert a data in a queue

    void enqueue(int x)

    {

        // Create new node

        Node\* temp = new Node;

        temp->data = x;

        temp->next = NULL;

        // new node is front and rear both If queue is empty

        if (rear == NULL) {

            front = temp;

            rear = temp;

        }

        else {

            // Add the new node at end

            rear->next = temp;

            rear = temp;

        }

    }

    // Function to remove

    int dequeue()

    {

        // If queue is empty, return NULL

        if (front == NULL) {

            cout << "Queue is Empty";

            return NULL;

        }

        else {

            Node\* temp = front;

            front = front->next;

            if (front == NULL)

                rear = NULL;

            int data = temp->data;

            delete temp;

            temp = NULL;

            return data;

        }

    }

    void display() {

        Node\* temp = front;

        int time;

        while (temp != NULL)

        {

            cout << temp->data << "  ";

            Sleep(200);

            cout << temp->data - 3 << endl;

            temp = temp->next;

        }

        cout << endl;

    }

    bool isEmpty() {

        if (front == NULL) {

            return true;

        }

        else {

            return false;

        }

    }

};

void displyid(Queue \*q1, Queue \*q2) {

    cout << q1->front << "  " << q2->front;

}

int main()

{

    Queue q;

    bool exit = false;

    int id;

    while (!exit){

        cout << "Enter ID: ";

        cin >> id;

        q.enqueue(id);

        cout << "Exit: 1/0: ";

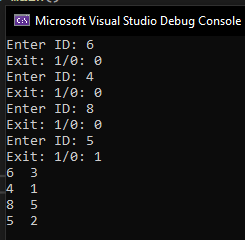
        cin >> exit;

    }

    q.display();

}

**Output**

****