Lab 07 Data Structures

Date: 16 Sept 2024

Q1) Implementation of Queue using arrays

**CODE:**

#include<iostream>

using namespace std;

#define MAX 5

class queue{

int front=-1;

int rear=-1;

public:

int q[MAX];

void enqueue(int num);

void dequeue();

};

void queue::enqueue(int num)

{

rear=rear+1;

if(rear>=MAX)

{

cout<< "Queue is full \n";

return;

}

else

{

q[rear]=num;

cout<< "Inserted "<<num<<"\n";

return;

}

}

void queue::dequeue()

{

front=front+1;

if(rear==front)

{

cout<< "Queue is empty \n";

return;

}

else

{

int temp=q[front];

cout<< "Deleted "<<temp<<"\n";

return;

}

}

int main()

{

queue q;

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

q.enqueue(4);

q.enqueue(5);

q.enqueue(6);

q.dequeue();

q.dequeue();

q.dequeue();

q.dequeue();

q.dequeue();

q.dequeue();

return 0;

}

**OUTPUT:**

Inserted 1

Inserted 2

Inserted 3

Inserted 4

Inserted 5

Queue is full

Deleted 1

Deleted 2

Deleted 3

Deleted 4

Deleted 5

Queue is empty

Q2) Implement a circular queue of Strings with functions insert, delete and display.

Q3) Write a program to implement the circular queue using arrays

**CODE:**

#include <iostream>

#include <string>

using namespace std;

class CircularQueue {

private:

string\* queue;

int front;

int rear;

int size;

public:

CircularQueue(int s) {

size = s;

queue = new string[size];

front = -1;

rear = -1;

}

~CircularQueue() {

delete[] queue;

}

bool isFull() {

return (rear + 1) % size == front;

}

bool isEmpty() {

return front == -1;

}

void insert(const string& value) {

if (isFull()) {

cout << "Queue is full. Cannot insert " << value << ".\n";

return;

}

if (isEmpty()) {

front = 0;

}

rear = (rear + 1) % size;

queue[rear] = value;

cout << "Inserted: " << value << "\n";

}

void deleteElement() {

if (isEmpty()) {

cout << "Queue is empty. Cannot delete.\n";

return;

}

string deletedValue = queue[front];

if (front == rear) {

front = -1;

rear = -1;

} else {

front = (front + 1) % size;

}

cout << "Deleted: " << deletedValue << "\n";

}

void display() {

if (isEmpty()) {

cout << "Queue is empty.\n";

return;

}

cout << "Queue elements:\n";

int index = front;

while (true) {

cout << queue[index] << "\n";

if (index == rear) {

break;

}

index = (index + 1) % size;

}

}

};

int main() {

CircularQueue cq(5);

cq.insert("A");

cq.insert("B");

cq.insert("C");

cq.display();

cq.deleteElement();

cq.display();

cq.insert("D");

cq.insert("E");

cq.insert("F");

cq.display();

cq.insert("G");

return 0;

}

**OUTPUT:**

Inserted: A

Inserted: B

Inserted: C

Queue elements:

A

B

C

Deleted: A

Queue elements:

B

C

Inserted: D

Inserted: E

Inserted: F

Queue elements:

B

C

D

E

F

Queue is full. Cannot insert G.