**Bahria University, Lahore Campus**

Department of Computer Science

Lab Journal 04

**(Spring 2023)**

|  |  |  |
| --- | --- | --- |
| Course: | **Data Structures and Algorithm - Lab** |  Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Course Code: | CSL-221 |  Max Marks: 10 |
| Faculty’s Name: | Fatima Zulfiqar |  |

Name: Muhammad Hammad Enroll No: 03-134221-024 Class: 3A(BSCS)

Objective(s):

Upon completion of this lab session, learners will be able to:

* Implement queue data structure (enqueue, dequeue, and display)
* Implement Circular queue.
* Use an appropriate data structure to implement the given scenario based problem.

## Lab Tasks:

**Task 1**

Implement queue data structure using arrays and perform following operations.

* enqueue ()
* dequeue ()
* Display ()

**Note:** size of the array should be fixed. Don’t use dynamic array.

Code:

#include<iostream>

using namespace std;

#define N 5

class Queue

{

public:

 Queue()

 {

 front = -1;

 back = -1;

 }

 void Enqueue(int val)

 {

 if (front == 0 && back == N - 1)

 {

 cout << "Queue is Full." << endl;

 return;

 }

 else

 {

 if (front == -1)

 {

 front = 0;

 }

 ++back;

 arr[back] = val;

 }

 }

 void Dequeue()

 {

 if (front > back)

 {

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

 return;

 }

 else

 {

 cout << "Poped value is : " << arr[front]<<endl;

 front++;

 }

 }

 void display()

 {

 cout << "Your Waiting Queue is : ";

 for (int o = front; o <= back; o++)

 {

 cout << arr[o] << " ";

 }

 cout << endl;

 }

private:

 int arr[N], front, back;

};

int main()

{

 Queue a;

 int c;

 do

 {

 cout << "1) Enter data" << endl << "2) Dequeue Data" << endl << "3)Display Data"<<endl<<"Enter Choice : ";

 cin >> c;

 switch (c)

 {

 case 1:

 int val;

 cout << "Enter value : ";

 cin >> val;

 a.Enqueue(val);

 break;

 case 2:

 a.Dequeue();

 break;

 case 3:

 a.display();

 break;

 default:

 break;

 }

 } while (c==1||c==2||c==3);

 system("pause");

 return 0;

}

**Task 2**

Re-do Task 1, and implement circular queue using arrays. Write down the difference between normal queue and circular queue. Apply all the checks in enqueue(), dequeue(), and display() function to successfully implement rules of circular queue.

Code:

#include<iostream>

using namespace std;

#define N 6

class Queue

{

public:

 Queue()

 {

 front = -1;

 back = -1;

 }

 void Enqueue(int val)

 {

 if (back == N - 1)

 {

 back = -1;

 ++back;

 arr[back] = val;

 return;

 }

 else

 {

 if (front == -1)

 {

 front = 0;

 }

 ++back;

 arr[back] = val;

 }

 }

 void Dequeue()

 {

 if (front == -1)

 {

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

 return;

 }

 if (front > back&&front >= N - 1)

 {

 front = -1;

 back = -1;

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

 return;

 }

 else

 {

 cout << "Poped value is : " << arr[front] << endl;

 arr[front] = NULL;

 front++;

 }

 }

 void display()

 {

 int p = front;

 cout << "Your Waiting Queue is : ";

 for (int o = front; o <= N - 1 && arr[o] >= 0; o++)

 {

 if (arr[o] == 0)

 {

 }

 else{

 cout << arr[o] << " ";

 }

 }

 for (int o = 0; o < p; o++)

 {

 if (arr[o] == 0)

 {

 }

 else{

 cout << arr[o] << " ";

 }

 }

 cout << endl;

 }

private:

 int arr[N], front, back;

};

int main()

{

 Queue a;

 a.Enqueue(1); a.Enqueue(2); a.Enqueue(3); a.Enqueue(4); a.Enqueue(5); a.Enqueue(6);

 int c;

 do

 {

 cout << "1) Enter data" << endl << "2) Dequeue Data" << endl << "3)Display Data" << endl << "Enter Choice : ";

 cin >> c;

 switch (c)

 {

 case 1:

 int val;

 cout << "Enter value : ";

 cin >> val;

 a.Enqueue(val);

 break;

 case 2:

 a.Dequeue();

 break;

 case 3:

 a.display();

 break;

 default:

 break;

 }

 } while (c == 1 || c == 2 || c == 3);

 system("pause");

 return 0;

}

OUTPUT:



**Task 3**

Suppose you are working in a hospital as a nurse and you are responsible for managing a queue of patients waiting at the reception area to see a doctor. Consider the following scenario;

At the start of your shift there are 6 patients waiting to see a doctor. The first patient in the queue has minor injury and can wait for up to 30 minutes to see the doctor. The second patient has severe illness and need to be seen by the doctor immediately. The remaining four patient have moderate injuries and can wait up to 15 minutes.

Identify the type of data structure that will be most appropriate to implement the above scenario. Implement the scenario with a choice of your data structure. What is the order in which a patient will be seen by the doctor? Also, print the order in which the doctor will first see a patient along with their illness.

**CODE:**

#include<iostream>

using namespace std;

#define N 6

class Queue

{

public:

 Queue()

 {

 front = -1;

 back = -1;

 }

 void Enqueue(int val)

 {

 if (back == N - 1)

 {

 back = -1;

 ++back;

 arr[back] = val;

 return;

 }

 else

 {

 if (front == -1)

 {

 front = 0;

 }

 ++back;

 arr[back] = val;

 }

 }

 void Dequeue()

 {

 if (front == -1)

 {

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

 return;

 }

 if (front > back&&front >= N - 1)

 {

 front = -1;

 back = -1;

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

 return;

 }

 else

 {

 cout << "Poped value is : " << arr[front] << endl;

 arr[front] = NULL;

 front++;

 }

 }

 void display()

 {

 int p = front;

 cout << "Your Waiting Queue is : ";

 for (int o = front; o <= N - 1 && arr[o] >= 0; o++)

 {

 cout << arr[o] << " ";

 }

 for (int o = 0; o < p; o++)

 {

 if (arr[o] == 0)

 {

 }

 else{

 cout << arr[o] << " ";

 }

 }

 cout << endl;

 }

private:

 int arr[N], front, back;

};

int main()

{

 Queue a;

 a.Enqueue(1); a.Enqueue(2); a.Enqueue(3); a.Enqueue(4); a.Enqueue(5); a.Enqueue(6);

 int c;

 do

 {

 cout << "1) Resequence Patients " << endl << "2) Send Patients to Doctor" << endl << "3)Display Patients List" << endl << "Enter Choice : ";

 cin >> c;

 switch (c)

 {

 case 1:

 for (int i = 0; i < N; i++)

 {

 int val;

 cout << "Enter Number : ";

 cin >> val;

 a.Enqueue(val);

 }

 break;

 case 2:

 a.Dequeue();

 break;

 case 3:

 a.display();

 break;

 default:

 break;

 }

 } while (c == 1 || c == 2 || c == 3);

 system("pause");

 return 0;

}

****

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 02 |  |  |
| 2. | 03 |  |  |
| 3. | 05 |  |  |
| **Total** | **10** |  | **Signature** |

**Note : Attempt all tasks and get them checked by your Lab Instructor.**