

# **LAPORAN PRAKTIKUM**

## **Modul 8**

### **Queue**



**Disusun Oleh :  
Fauzan Rofif  
Ardiyanto/2211104036  
S1SE06-02**

**Asisten Praktikum :  
Aldi Putra  
Andini Nur Hidayah**

**Dosen Pengampu :  
Wahyu Andi Saputra**

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2024**

## 1. Tujuan

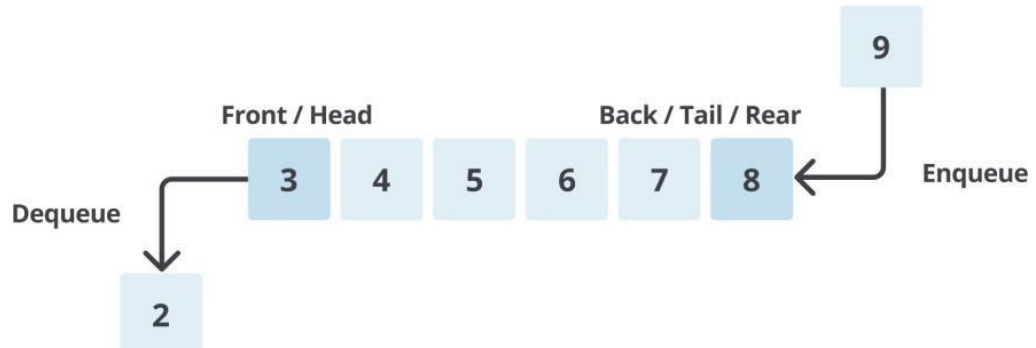
- Mahasiswa mampu menjelaskan definisi dan konsep dari queue.
- Mahasiswa mampu menerapkan operasi tambah, menghapus, pada queue.
- Mahasiswa mampu menerapkan operasi tampil data pada queue.

## 2. Landasan Teori

### a. Queue

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode **FIFO** (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep **antrian** pada kehidupan sehari-hari, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu.

Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. **Front/head** adalah pointer ke elemen pertama dalam queue dan **rear/tail/back** adalah pointer ke elemen terakhir dalam queue.



FIRST IN FIRST OUT (FIFO)

Perbedaan antara *stack* dan *queue* terletak pada aturan penambahan dan penghapusan elemen. Pada *stack*, operasi penambahan dan penghapusan elemen dilakukan di satu ujung yang disebut *top* (ujung atas). Elemen yang terakhir kali dimasukkan ke dalam *stack* akan berada di posisi paling atas dan akan menjadi elemen pertama yang dihapus. Sifat ini dikenal dengan istilah *LIFO* (Last In, First Out). Contoh analogi sederhana dari *stack* adalah tumpukan piring, di mana piring terakhir yang ditambahkan berada di posisi paling atas dan akan diambil atau dihapus terlebih dahulu.

Sebaliknya, pada *queue*, operasi penambahan dan penghapusan elemen dilakukan di dua ujung yang berbeda. Elemen baru ditambahkan di ujung belakang (*rear* atau *tail*), dan elemen dihapus dari ujung depan (*front* atau *head*). Proses ini mengikuti prinsip *FIFO* (First In, First Out), yang berarti elemen pertama yang dimasukkan ke dalam *queue* akan menjadi elemen pertama yang dikeluarkan. Dalam konteks *queue*, operasi penambahan elemen dikenal sebagai ***Enqueue***, dan operasi penghapusan elemen disebut ***Dequeue***.

Pada *Enqueue*, elemen ditambahkan di belakang *queue* setelah elemen terakhir yang ada, sementara pada *Dequeue*, elemen paling depan (*head*) dihapus, dan posisi *head* akan bergeser ke elemen berikutnya. Contoh penggunaan *queue* dalam kehidupan sehari-hari adalah antrean di kasir, di mana orang pertama yang datang adalah yang pertama dilayani.


#### **Operasi pada Queue:**

- `enqueue()` : menambahkan data ke dalam queue.
- `dequeue()` : mengeluarkan data dari queue.
- `peek()` : mengambil data dari queue tanpa menghapusnya.
- `isEmpty()` : mengecek apakah queue kosong atau tidak.
- `isFull()` : mengecek apakah queue penuh atau tidak.
- `size()` : menghitung jumlah elemen dalam queue.

### **3. Guided**

#### **a. Guided 1**

Source Code



```
#include <iostream>
#define MAX 100

using namespace std;

class Queue {
private:
    int front, rear;
    int arr[MAX];
public:

    Queue() {
        front = -1;
        rear = -1;
    }

    bool isFull() {
        return rear == MAX - 1;
    }

    bool isEmpty() {
        return front == -1 || front > rear;
    }

    void enqueue(int x) {
        if (isFull()) {
            cout << "Queue Overflow\n";
            return;
        }
        if (front == -1) front = 0;
        arr[++rear] = x;
    }
}
```

```
void dequeue() {
    if (isEmpty()) {
        cout << "Queue Underflow\n";
        return;
    }
    front++;
}

int peek() {
    if (!isEmpty()) {
        return arr[front];
    }
    cout << "Queue is empty\n";
    return -1;
}

void display() {
    if (isEmpty()) {
        cout << "Queue is empty\n";
        return;
    }
    for (int i = front; i <= rear; i++) {
        cout << arr[i] << " ";
    }
    cout << "\n";
}

};

int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    cout << "Queue elements: ";
    q.display();

    cout << "Front element: " << q.peek() << "\n";

    cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}
```

## Output

```
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT
'OT1.exe'
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT
P\Pra> zcd 'c:\Users\LENOVO\Documents\ITTP\TUGAS SEMES
TER 5 ITTP\Praktikum STD\week8\output'
```

## Deskripsi

Program tersebut adalah implementasi struktur data *queue* (antrian) menggunakan array dengan batas maksimum elemen sebanyak 100. Queue memiliki operasi dasar seperti **enqueue** (menambahkan elemen ke belakang antrian), **dequeue** (menghapus elemen dari depan antrian), dan **peek** (melihat elemen di depan antrian tanpa menghapusnya). Program juga menyediakan fungsi **isFull** untuk mengecek apakah antrian penuh, **isEmpty** untuk mengecek apakah antrian kosong, serta **display** untuk menampilkan semua elemen antrian. Dalam fungsi main, queue diuji dengan menambahkan tiga elemen (10, 20, 30), menampilkan elemen queue, melihat elemen depan, dan

mencoba menghapus elemen depan sambil memeriksa perubahan dalam antrian. Program menangani kondisi overflow (penambahan pada antrian penuh) dan underflow (penghapusan dari antrian kosong) dengan menampilkan pesan kesalahan.

b. Guided 2

Source Code

```

#include <iostream>

using namespace std;

// Node untuk setiap elemen Queue
class Node {
public:
    int data;        // Data elemen
    Node* next;      // Pointer ke node berikutnya

    // Konstruktor untuk Node
    Node(int value) {
        data = value;
        next = nullptr;
    }
};

// Kelas Queue menggunakan linked list
class Queue {
private:
    Node* front; // Pointer ke elemen depan Queue
    Node* rear;  // Pointer ke elemen belakang Queue

public:
    // Konstruktor Queue
    Queue() {
        front = rear = nullptr;
    }

    // Mengecek apakah Queue kosong
    bool isEmpty() {
        return front == nullptr;
    }

    // Menambahkan elemen ke Queue
    void enqueue(int x) {
        Node* newNode = new Node(x);
        if (isEmpty()) {
            front = rear = newNode; // Jika Queue kosong
            return;
        }
        rear->next = newNode; // Tambahkan node baru ke belakang
        rear = newNode;      // Perbarui rear
    }

    // Menghapus elemen dari depan Queue
    void dequeue() {
        if (isEmpty()) {
            cout << "Queue Underflow\n";
            return;
        }
        Node* temp = front; // Simpan node depan untuk dihapus
        front = front->next; // Pindahkan front ke node berikutnya
        delete temp;        // Hapus node lama
        if (front == nullptr) // Jika Queue kosong, rear juga harus
            rear = nullptr;
    }
};

```



```

// Mengembalikan elemen depan Queue tanpa menghapusnya
int peek() {
    if (!isEmpty()) {
        return front->data;
    }
    cout << "Queue is empty\n";
    return -1; // Nilai sentinel
}

// Menampilkan semua elemen di Queue
void display() {
    if (isEmpty()) {
        cout << "Queue is empty\n";
        return;
    }
    Node* current = front; // Mulai dari depan
    while (current) {      // Iterasi sampai akhir
        cout << current->data << " ";
        current = current->next;
    }
    cout << "\n";
}

};

// Fungsi utama untuk menguji Queue
int main() {
    Queue q;

    // Menambahkan elemen ke Queue
    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    // Menampilkan elemen di Queue
    cout << "Queue elements: ";
    q.display();

    // Menampilkan elemen depan
    cout << "Front element: " << q.peek() << "\n";

    // Menghapus elemen dari depan Queue
    q.dequeue();
    cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}

```

## Output

```
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\week8\output> & .\OT2.exe
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\Praktikum STD\week8\output> █
```

## Deskripsi

Program tersebut adalah implementasi struktur data *queue* (antrian) menggunakan pendekatan *linked list*. Elemen *queue* diwakili oleh kelas **Node** yang memiliki atribut data (nilai elemen) dan next (penunjuk ke elemen berikutnya). Kelas **Queue** memiliki pointer front untuk menunjuk elemen depan dan rear untuk elemen belakang. Program menyediakan operasi dasar seperti **enqueue** (menambahkan elemen ke belakang antrian), **dequeue** (menghapus elemen dari depan antrian), dan **peek** (melihat elemen depan tanpa menghapusnya). Fungsi tambahan seperti **isEmpty** digunakan untuk memeriksa apakah antrian kosong, dan **display** untuk menampilkan semua elemen dalam antrian. Di fungsi main, *queue* diuji dengan menambahkan tiga elemen (10, 20, 30), menampilkan elemen antrian, melihat elemen depan,

menghapus elemen depan, dan menampilkan elemen yang tersisa. Program ini menangani kondisi antrian kosong dengan menampilkan pesan kesalahan.

- c. Guided 3  
Source Code

```
#include<iostream>

using namespace std;

const int maksimalQueue = 5; // Maksimal antrian
int front = 0; // Penanda antrian
int back = 0; // Penanda
string queueTeller[5]; // Fungsi pengecekan

bool isFull() { // Pengecekan antrian penuh atau tidak
    if (back == maksimalQueue) { return true; // =1
    } else {
        return false;
    }
}

bool isEmpty() { // Antriannya kosong atau tidak
    if (back == 0) { return true;
    } else {
        return false;
    }
}

void enqueueAntrian(string data) { // Fungsi menambahkan antrian
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    } else {
        if (isEmpty()) { // Kondisi ketika queue kosong
            queueTeller[0] = data; front++;
            back++;
        } else { // Antriannya ada isi queueTeller[back] = data; back++;
        }
    }
}

void dequeueAntrian() { // Fungsi mengurangi antrian
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        for (int i = 0; i < back; i++) { queueTeller[i] = queueTeller[i + 1];
        }
        back--;
    }
}

int countQueue() { // Fungsi menghitung banyak antrian
    return back;
}
```

```

void clearQueue() { // Fungsi menghapus semua antrian
if (isEmpty()) {
cout << "Antrian kosong" << endl;
} else {
for (int i = 0; i < back; i++) { queueTeller[i] = "";
}
back = 0;
front = 0;
}
}

void viewQueue() { // Fungsi melihat antrian
cout << "Data antrian teller:" << endl; for (int i = 0; i < maksimalQueue; i++) {
if (queueTeller[i] != "") {
cout << i + 1 << ". " << queueTeller[i] <<

endl;

} else {
cout << i + 1 << ". (kosong)" << endl;

}
}
}

int main() {
enqueueAntrian("Andi");

enqueueAntrian("Maya");

viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

dequeueAntrian();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

clearQueue();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

return 0;
}

```

Output

```

PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT
P\Praktikum STD\week8\output> & .\'OT3.exe'
Data antrian teller:
1. Andi
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 1
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
Antrian kosong
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT
P\Praktikum STD\week8\output>

```

### Deskripsi

Program ini adalah implementasi struktur data *queue* (antrian) menggunakan array dengan kapasitas tetap (maksimal 5 elemen). Program ini mensimulasikan antrian teller dengan operasi dasar sebagai berikut: **enqueueAntrian** untuk menambahkan elemen ke antrian, **dequeueAntrian** untuk menghapus elemen dari antrian, **isFull** untuk memeriksa apakah antrian penuh, **isEmpty** untuk memeriksa apakah antrian kosong, **countQueue** untuk menghitung jumlah elemen dalam antrian, **clearQueue** untuk mengosongkan seluruh antrian, dan **viewQueue** untuk menampilkan isi antrian.

Di dalam fungsi main, program menambahkan dua nama ke dalam antrian ("Andi" dan "Maya"), menampilkan isi antrian beserta jumlahnya, menghapus

satu elemen dari antrian, lalu menampilkan kembali isi dan jumlah antrian. Program juga mengosongkan seluruh antrian menggunakan **clearQueue** dan


menampilkan keadaan antrian setelahnya. Kondisi penuh atau kosong ditangani dengan menampilkan pesan kesalahan yang sesuai.

#### **4. Unguided**

##### **a. Soal 1**

Source Code





```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* rear;

public:
    Queue() {
        front = nullptr;
        rear = nullptr;
    }

    bool isEmpty() {
        return front == nullptr;
    }

    void enqueue(int x) {
        Node* newNode = new
Node(); newNode->data = x;
        newNode->next = nullptr;

        if (rear == nullptr) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
    }
}
```

```
void display() {
    if (isEmpty()) {
        cout << "Queue is empty\n";
        return;
    }

    Node* temp = front;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

};

int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    cout << "Queue elements: ";
    q.display();

    q.dequeue();

    cout << "After dequeuing, queue elements: ";
    q.display();

    return 0;
}
```

## Output

```
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT  
P\Praktikum STD\week8\output> & .\'UN1.exe'  
Queue elements: 10 20 30  
After dequeuing, queue elements: 20 30  
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT  
P\Praktikum STD\week8\output> █
```

## Deskripsi

Program ini mengimplementasikan struktur data **queue** menggunakan **linked list**, di mana elemen-elemen dimasukkan ke dalam antrian (enqueue) dan dikeluarkan dari antrian (dequeue) sesuai prinsip **FIFO** (First In, First Out). Node dalam linked list hanya menyimpan data sederhana, yaitu nilai integer. Program ini menyediakan operasi untuk memeriksa apakah antrian kosong, menambahkan elemen ke antrian, menghapus elemen dari antrian, dan menampilkan seluruh elemen dalam antrian. Struktur ini fleksibel karena tidak bergantung pada batasan ukuran seperti array.

## b. Soal 2

### Source Code

```
#include <iostream>
#include <string>
using namespace std;

struct Node {
    string nama;
    string nim;
    Node* berikut;
};

class Queue {
private:
    Node* depan;
    Node* belakang;

public:
    Queue() {
        depan = nullptr;
        belakang = nullptr;
    }

    bool kosong() {
        return depan == nullptr;
    }

    void enqueue(string nama, string nim) {
        Node* nodeBaru = new Node();
        nodeBaru->nama = nama;
        nodeBaru->nim = nim;
        nodeBaru->berikut = nullptr;

        if (belakang == nullptr) {
            depan = belakang = nodeBaru;
        } else {
            belakang->berikut = nodeBaru;
            belakang = nodeBaru;
        }
    }
};
```

```

    }
}

void dequeue() {
    if (kosong()) {
        cout << "Antrian kosong (Underflow)\n";
        return;
    }

    Node* sementara = depan;
    depan = depan->berikut;

    if (depan == nullptr) {
        belakang = nullptr;
    }

    delete sementara;
}

void tampilkan() {
    if (kosong()) {
        cout << "Antrian kosong\n";
        return;
    }

    Node* sementara = depan;
    while (sementara != nullptr) {
        cout << "Nama: " << sementara->nama << ", NIM: " << sementara-
>nim << "\n";
        sementara = sementara->berikut;
    }
}

};

int main() {
    Queue antrian;
    int jumlah;

    cout << "Masukkan jumlah mahasiswa yang ingin diinput: ";
    cin >> jumlah;

    for (int i = 0; i < jumlah; i++) {
        string nama, nim;
        cout << "Masukkan nama mahasiswa: ";
        cin >> ws;
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa: ";
        cin >> nim;
        antrian.enqueue(nama, nim);
    }

    cout << "\nData antrian mahasiswa:\n";
    antrian.tampilkan();
}

```

```
    antrian.dequeue();  
    cout << "\nSetelah 1 mahasiswa keluar dari antrian:\n";  
    antrian.tampilkan();  
  
    return 0;  
}
```

## Output

```
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT
P\Praktikum STD\week8\output> & .\UN2.exe'
Masukkan jumlah mahasiswa yang ingin diinput: 2
Masukkan nama mahasiswa: fauzan
Masukkan NIM mahasiswa: 2211104036
Masukkan nama mahasiswa: ziva
Masukkan NIM mahasiswa: 2211104039

Data antrian mahasiswa:
Nama: fauzan, NIM: 2211104036
Nama: ziva, NIM: 2211104039

Setelah 1 mahasiswa keluar dari antrian:
Nama: ziva, NIM: 2211104039
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITT
P\Praktikum STD\week8\output> █
```

## Deskripsi

Program ini merupakan pengembangan dari program nomor 1 dengan menambahkan atribut tambahan pada node, yaitu **nama mahasiswa** dan **NIM mahasiswa**. Antrian masih menggunakan linked list dengan konsep FIFO, namun data yang dimasukkan lebih kompleks karena setiap node menyimpan informasi nama dan NIM mahasiswa. Pengguna dapat memasukkan data melalui terminal, dan program akan menampilkan elemen-elemen dalam antrian sesuai urutan yang dimasukkan. Operasi enqueue, dequeue, dan tampilkan tetap tersedia, namun disesuaikan dengan data nama dan NIM.

c. Soal 3

Source Code

```
#include <iostream>
#include <string>
using namespace std;

struct Node {
    string nama;
    string nim;
    Node* berikut;
};

class PriorityQueue {
private:
    Node* depan;

public:
    PriorityQueue() {
        depan = nullptr;
    }

    bool kosong() {
        return depan == nullptr;
    }

    void enqueue(string nama, string nim) {
        Node* nodeBaru = new Node();
        nodeBaru->nama = nama;
        nodeBaru->nim = nim;
        nodeBaru->berikut = nullptr;

        if (kosong() || nim < depan->nim) {
            nodeBaru->berikut = depan;
            depan = nodeBaru;
        } else {
            Node* sementara = depan;
            while (sementara->berikut != nullptr && sementara->berikut->nim
<= nim) {
                sementara = sementara->berikut;
            }
            nodeBaru->berikut = sementara->berikut;
            sementara->berikut = nodeBaru;
        }
    }

    void dequeue() {
        if (kosong()) {
            cout << "Antrian kosong (Underflow)\n";
            return;
        }
    }
}
```



```

        Node* sementara = depan;
        depan = depan->berikut;
        delete sementara;
    }

    void tampilkan() {
        if (kosong()) {
            cout << "Antrian kosong\n";
            return;
        }

        Node* sementara = depan;
        while (sementara != nullptr) {
            cout << "Nama: " << sementara->nama << ", NIM: " << sementara-
>nim << "\n";
            sementara = sementara->berikut;
        }
    }
};

int main() {
    PriorityQueue antrian;
    int jumlah;

    cout << "Masukkan jumlah mahasiswa yang ingin diinput: ";
    cin >> jumlah;

    for (int i = 0; i < jumlah; i++) {
        string nama, nim;
        cout << "Masukkan nama mahasiswa: ";
        cin >> ws;
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa: ";
        cin >> nim;
        antrian.enqueue(nama, nim);
    }

    cout << "\nData antrian mahasiswa (berdasarkan prioritas NIM):\n";
    antrian.tampilkan();

    antrian.dequeue();
    cout << "\nSetelah 1 mahasiswa keluar dari antrian:\n";
    antrian.tampilkan();

    return 0;
}

```

## Output

```
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 IT
P\Praktikum STD\week8\output> & .\'UN3.exe'
Masukkan jumlah mahasiswa yang ingin diinput: 2
Masukkan nama mahasiswa: fauzan
Masukkan NIM mahasiswa: 2211104036
Masukkan nama mahasiswa: zivva
Masukkan NIM mahasiswa: 2211104039

Data antrian mahasiswa (berdasarkan prioritas NIM):
Nama: fauzan, NIM: 2211104036
Nama: zivva, NIM: 2211104039

Setelah 1 mahasiswa keluar dari antrian:
Nama: zivva, NIM: 2211104039
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 IT
P\Praktikum STD\week8\output> |
```

## Deskripsi

Program ini memodifikasi konsep queue dari program nomor 1 dan 2 dengan menambahkan **prioritas berdasarkan NIM mahasiswa**. Saat elemen baru ditambahkan ke antrian, elemen tersebut dimasukkan ke posisi yang sesuai dengan urutan NIM secara **ascending**. Dengan kata lain, mahasiswa dengan NIM lebih kecil akan berada di depan antrian. Operasi enqueue menggunakan perbandingan nilai NIM untuk menyisipkan elemen pada posisi yang tepat, memastikan antrian tetap teratur. Program ini tetap menyediakan fungsi dequeue dan tampilkan untuk mengelola dan melihat isi antrian.