

LAPORAN PRAKTIKUM
Modul VIII
“QUEUE”



Disusun Oleh:
Dewi Atika Muthi -2211104042
SE-07-02

Dosen:
Wahyu Andi Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat:

- Menjelaskan definisi dan konsep struktur data queue
- Menerapkan operasi penambahan (**enqueue**) dan penghapusan (**dequeue**) pada queue
- Menerapkan operasi tampil data pada queue

2. Landasan Teori

Queue (antrian) adalah struktur data yang mengikuti **prinsip FIFO (First In First Out)**, di mana elemen pertama yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Konsep ini mirip dengan antrian pada kehidupan sehari-hari, seperti antrian di loket pembelian tiket.

Karakteristik Queue:

- Memiliki dua pointer utama: **HEAD** (depan) dan **TAIL** (belakang)

Operasi dasar:

- **Enqueue** : Menambahkan elemen ke akhir queue
- **Dequeue** : Menghapus elemen dari depan 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.
-



Perbedaan dengan Stack:

- **Stack** menggunakan prinsip **LIFO (Last In First Out)**
- **Queue** menggunakan prinsip **FIFO (First In First Out)**

Implementasi Queue:

Queue dapat diimplementasikan menggunakan dua metode utama:

- Implementasi dengan **Array**
- Implementasi dengan **Linked List**

3. Guided

1) Queue dengan Array Statis

SourceCode Guided1.cpp:

```
#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";

q.dequeue();
cout << "After dequeuing, queue elements(menghapus elemen depan): ";
q.display();

return 0;
}

```

Output:

```

Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements(menghapus elemen depan): 20 30

Process returned 0 (0x0)   execution time : 0.088 s
Press any key to continue.

```

Deskripsi Program:

Program ini merupakan implementasi queue menggunakan array statis dengan ukuran tetap (MAX 100) dengan basis array dengan fixed size, operasi dilakukan dengan menggeser index front dan rear.

Program membuat objek Queue, lalu menambahkan tiga elemen (10, 20, 30) menggunakan enqueue. Kemudian menampilkan seluruh elemen queue, menampilkan elemen depan menggunakan peek, selanjutnya menghapus elemen depan dengan dequeue, dan akhirnya menampilkan kembali isi queue setelah penghapusan.

2) Queue dengan Linked List Dinamis

SourceCode guided2.cpp:

```

#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 null
            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();

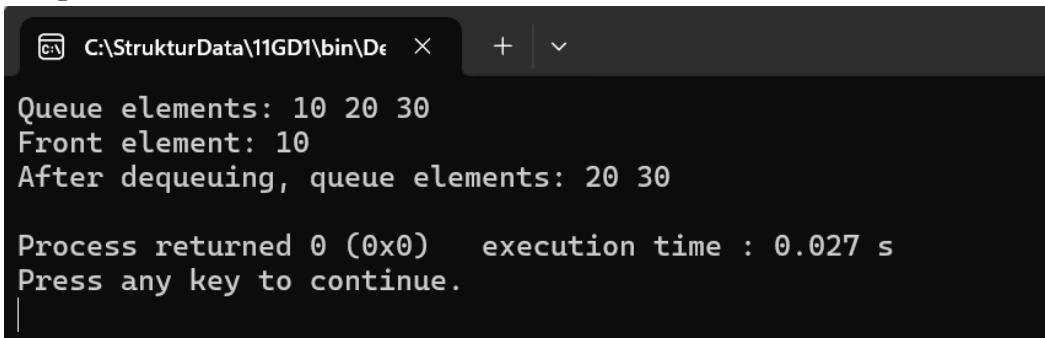
// Menampilkan elemen depan
cout << "Front element: " << q.peek() << "\n";

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

return 0;
}

```

Output:



```

C:\StrukturData\11GD1\bin\De
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30

Process returned 0 (0x0)   execution time : 0.027 s
Press any key to continue.

```

Deskripsi Program:

Program tersebut adalah implementasi queue menggunakan linked list yang setiap elemen dibuat sebagai node yang saling terhubung.

Serupa dengan Guided 1, tetapi menggunakan implementasi Queue dengan Linked List. Program membuat objek Queue, menambahkan tiga elemen (10, 20, 30), menampilkan seluruh elemen, menampilkan elemen depan menggunakan peek, menghapus elemen depan dengan dequeue, dan menampilkan kembali isi queue setelah penghapusan.

3) Queue Antrian Teller

Sourcecode guided3.cpp:

```

#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;

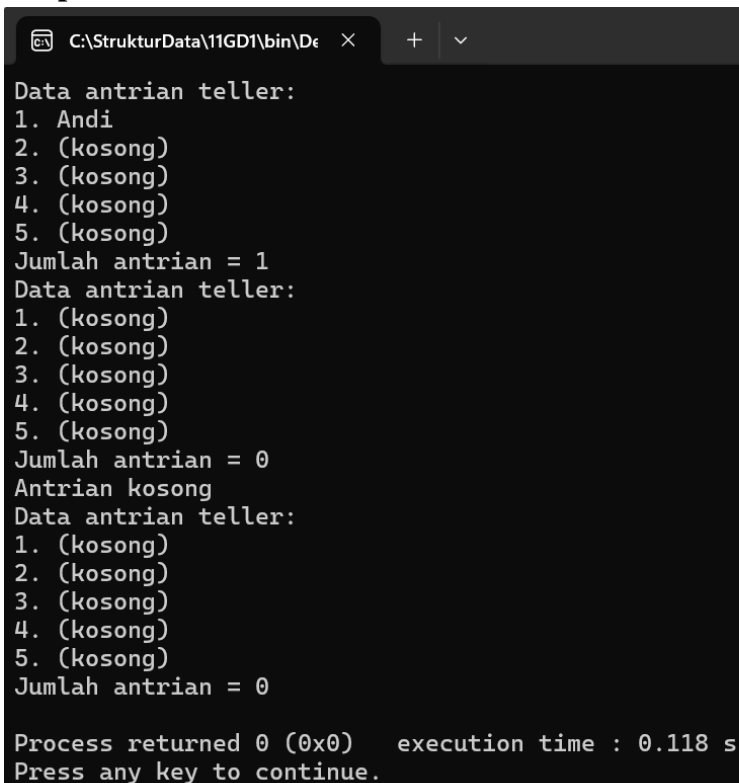
dequeueAntrian();
viewQueue();
cout << "Jumlah antrian = " << countQueue() << endl;

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

return 0;
}

```

Output:



```

C:\StrukturData\11GD1\bin\De X + v
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

Process returned 0 (0x0)   execution time : 0.118 s
Press any key to continue.

```

Deskripsi Program:

Program ini merupakan implementasi queue spesifik untuk simulasi antrian teller menggunakan array dengan fungsi-fungsi khusus antrian

Program fokus pada simulasi antrian teller. Menambahkan dua nama (Andi dan Maya) ke dalam queue, menampilkan isi queue, menghitung jumlah antrian, menghapus satu elemen, menampilkan queue kembali, menghitung ulang jumlah antrian, lalu mengosongkan queue dan menampilkan hasilnya sekali lagi.

4. Unguided

- 1) Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

Sourcecode **unguided1.cpp**:

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

// Kelas Node untuk implementasi Linked List
class Node {
public:
    string data;
    Node* next;
    Node(string value) {
        data = value;
        next = nullptr;
    }
};

class Queue {
private:
    Node* front;
    Node* back;
    int size;
    const int maksimalQueue = 5;

public:
    Queue() {
        front = nullptr;
        back = nullptr;
        size = 0;
    }

    bool isFull() {
        return size == maksimalQueue;
    }

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

    void enqueueAntrian(string data) {
        if (isFull()) {
            cout << "Antrian penuh" << endl;
        } else {
            Node* newNode = new Node(data);
            if (isEmpty()) {
                front = back = newNode;
            } else {
                back->next = newNode;
                back = newNode;
            }
            size++;
        }
    }

    void dequeueAntrian() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            Node* temp = front;
            front = front->next;
            delete temp;
            size--;
        }
    }
};
```

```

        if (front == nullptr) {
            back = nullptr;
        }
    }

    int countQueue() {
        return size;
    }

    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
    }

    void viewQueue() {
        cout << "Data antrian teller:" << endl;
        Node* current = front;
        int index = 1;
        while (current != nullptr) {
            cout << index << ". " << current->data << endl;
            current = current->next;
            index++;
        }

        // Menampilkan slot kosong
        for (int i = index; i <= maksimalQueue; i++) {
            cout << i << ". (kosong)" << endl;
        }
    }
};

int main() {
    Queue q;
    q.enqueueAntrian("Andi");
    q.enqueueAntrian("Maya");
    q.viewQueue();
    cout << "Jumlah antrian = " << q.countQueue() << endl;
    q.dequeueAntrian();
    q.viewQueue();
    cout << "Jumlah antrian = " << q.countQueue() << endl;
    q.clearQueue();
    q.viewQueue();
    cout << "Jumlah antrian = " << q.countQueue() << endl;
    return 0;
}

```

Output:

```

Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
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

Process returned 0 (0x0)   execution time : 0.133 s
Press any key to continue.

```

Penjelasan program:

Program ini mengubah implementasi queue dari array menjadi linked list, dengan membuat kelas Node untuk menyimpan data string, lalu menerapkan fungsi-fungsi dasar queue (enqueue, dequeue, isFull, isEmpty) menggunakan linked list dengan logika yang sama dengan implementasi sebelumnya.

- 2) Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

Sourcecode **unguided2.cpp**:

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

class Mahasiswa {
public:
    string nama;
    string nim;

    Mahasiswa(string _nama = "", string _nim = "") {
        nama = _nama;
        nim = _nim;
    }
};

class Node {
public:
    Mahasiswa data;
    Node* next;
    Node(Mahasiswa value) {
        data = value;
        next = nullptr;
    }
};

class Queue {
private:
    Node* front;
    Node* back;
    int size;
    const int maksimalQueue = 5;

public:
    Queue() {
        front = nullptr;
        back = nullptr;
        size = 0;
    }

    bool isFull() {
        return size == maksimalQueue;
    }

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

    void enqueueAntrian(Mahasiswa data) {
        if (isFull()) {
            cout << "Antrian penuh" << endl;
        } else {
            Node* newNode = new Node(data);
            if (isEmpty()) {
```

```

        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
    size++;
}

void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        delete temp;
        size--;

        if (front == nullptr) {
            back = nullptr;
        }
    }
}

int countQueue() {
    return size;
}

void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

void viewQueue() {
    cout << "Data antrian mahasiswa:" << endl;
    Node* current = front;
    int index = 1;
    while (current != nullptr) {
        cout << index << ". Nama: " << current->data.nama
            << ", NIM: " << current->data.nim << endl;
        current = current->next;
        index++;
    }

    // Menampilkan slot kosong
    for (int i = index; i <= maksimalQueue; i++) {
        cout << i << ". (kosong)" << endl;
    }
}

int getMaksimalQueue() {
    return maksimalQueue;
}

};

int main() {
    Queue q;

    // Input data mahasiswa
    string nama, nim;
    int maksimalQueue = q.getMaksimalQueue(); // Mendapatkan nilai
    maksimalQueue

    for (int i = 0; i < maksimalQueue; i++) {

```

```

        cout << "Masukkan nama mahasiswa ke-" << i+1 << ": ";
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa ke-" << i+1 << ": ";
        getline(cin, nim);

        q.enqueueAntrian(Mahasiswa(nama, nim));
    }

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

    q.dequeueAntrian();
    cout << endl;
    q.viewQueue();
    cout << "Jumlah antrian setelah dihapus 1= " << q.countQueue() << endl;

    q.clearQueue();
    cout << endl;
    q.viewQueue();

    cout << "Jumlah antrian setelah di-clear= " << q.countQueue() << endl;

    return 0;
}

```

Output:

```

C:\StrukturData\ASSES\bin\Di...
Masukkan nama mahasiswa ke-1: Dewi Atika
Masukkan NIM mahasiswa ke-1: 2211104042
Masukkan nama mahasiswa ke-2: Mdhsaf
Masukkan NIM mahasiswa ke-2: 2211104023
Masukkan nama mahasiswa ke-3: Cabine Thither
Masukkan NIM mahasiswa ke-3: 2211104056
Masukkan nama mahasiswa ke-4: Fara
Masukkan NIM mahasiswa ke-4: 2211104021
Masukkan nama mahasiswa ke-5: Dabil
Masukkan NIM mahasiswa ke-5: 2211104020

Data antrian mahasiswa:
1. Nama: Dewi Atika, NIM: 2211104042
2. Nama: Mdhsaf, NIM: 2211104023
3. Nama: Cabine Thither, NIM: 2211104056
4. Nama: Fara, NIM: 2211104021
5. Nama: Dabil, NIM: 2211104020
Jumlah antrian = 5

Data antrian mahasiswa:
1. Nama: Mdhsaf, NIM: 2211104023
2. Nama: Cabine Thither, NIM: 2211104056
3. Nama: Fara, NIM: 2211104021
4. Nama: Dabil, NIM: 2211104020
5. (kosong)
Jumlah antrian setelah dihapus 1= 4

Data antrian mahasiswa:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian setelah di-clear= 0

Process returned 0 (0x0)   execution time : 138.669 s
Press any key to continue.

```

Penjelasan Program:

Dalam program ini kita memodifikasi pada soal sebelumnya dengan menambahkan kelas Mahasiswa dengan atribut nama dan NIM, lalu mengubah Node dan Queue untuk bekerja dengan objek Mahasiswa. Juga membuat program yang memungkinkan input nama dan NIM dari user, setelah memasukkan data, program akan menampilkan data mahasiswa dalam antrian, lalu menghapus dengan dequeue satu kali, setelahnya data ditampilkan Kembali. Selanjutnya

program menghapus semua data dengan clearqueue.

- 3) Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

Sourcecode **unguided3.cpp**:

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

class Mahasiswa {
public:
    string nama;
    string nim;

    Mahasiswa(string _nama = "", string _nim = "") {
        nama = _nama;
        nim = _nim;
    }
};

class Node {
public:
    Mahasiswa data;
    Node* next;
    Node(Mahasiswa value) {
        data = value;
        next = nullptr;
    }
};

class Queue {
private:
    Node* front;
    Node* back;
    int size;
    const int maksimalQueue = 5;

    // Fungsi untuk menyisipkan node berdasarkan prioritas NIM
    void insertSorted(Mahasiswa data) {
        Node* newNode = new Node(data);

        // Jika queue kosong atau NIM baru lebih kecil dari front
        if (isEmpty() || newNode->data.nim < front->data.nim) {
            newNode->next = front;
            front = newNode;
            if (back == nullptr) back = newNode;
            return;
        }

        Node* current = front;
        while (current->next != nullptr &&
            current->next->data.nim <= newNode->data.nim) {
            current = current->next;
        }

        // Sisipkan node baru
        newNode->next = current->next;
        current->next = newNode;

        // Update back jika node baru adalah elemen terakhir
        if (newNode->next == nullptr) {
            back = newNode;
        }
    }
public:
```

```

Queue() {
    front = nullptr;
    back = nullptr;
    size = 0;
}

bool isFull() {
    return size == maksimalQueue;
}

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

void enqueueAntrian(Mahasiswa data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
        return;
    }

    insertSorted(data);
    size++;
}

void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
        return;
    }

    Node* temp = front;
    front = front->next;
    delete temp;
    size--;

    if (front == nullptr) {
        back = nullptr;
    }
}

int countQueue() {
    return size;
}

void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

void viewQueue() {
    cout << "Data antrian mahasiswa (diurutkan berdasarkan NIM):" << endl;
    Node* current = front;
    int index = 1;
    while (current != nullptr) {
        cout << index << ". Nama: " << current->data.nama
            << ", NIM: " << current->data.nim << endl;
        current = current->next;
        index++;
    }
    for (int i = index; i <= maksimalQueue; i++) {
        cout << i << ". (kosong)" << endl;
    }
}

int getMaksimalQueue() {
    return maksimalQueue;
}

```

```

    }
};

int main() {
    Queue q;
    string nama, nim;
    int maksimalQueue = q.getMaksimalQueue(); // Mendapatkan nilai
    maksimalQueue

    for (int i = 0; i < maksimalQueue; i++) {
        cout << "Masukkan nama mahasiswa ke-" << i+1 << ": ";
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa ke-" << i+1 << ": ";
        getline(cin, nim);

        q.enqueueAntrian(Mahasiswa(nama, nim));
    }
    cout << endl;
    q.viewQueue();
    cout << "Jumlah antrian = " << q.countQueue() << endl;

    q.dequeueAntrian();
    cout << endl;
    q.viewQueue();
    cout << "Jumlah antrian setelah dihapus 1= " << q.countQueue() << endl;

    q.clearQueue();
    cout << endl;
    q.viewQueue();

    cout << "Jumlah antrian setelah di-clear= " << q.countQueue() << endl;

    return 0;
}

```

Output:

```

C:\StrukturData\ASSES\bin\Di x + v

Masukkan nama mahasiswa ke-1: Dewi Atika
Masukkan NIM mahasiswa ke-1: 2211104042
Masukkan nama mahasiswa ke-2: Mdhmsaf
Masukkan NIM mahasiswa ke-2: 2211104023
Masukkan nama mahasiswa ke-3: Cabine Thither
Masukkan NIM mahasiswa ke-3: 2211104056
Masukkan nama mahasiswa ke-4: Fara
Masukkan NIM mahasiswa ke-4: 2211104021
Masukkan nama mahasiswa ke-5: Dabil
Masukkan NIM mahasiswa ke-5: 2211104020

Data antrian mahasiswa (diurutkan berdasarkan NIM):
1. Nama: Dabil, NIM: 2211104020
2. Nama: Fara, NIM: 2211104021
3. Nama: Mdhmsaf, NIM: 2211104023
4. Nama: Dewi Atika, NIM: 2211104042
5. Nama: Cabine Thither, NIM: 2211104056
Jumlah antrian = 5

Data antrian mahasiswa (diurutkan berdasarkan NIM):
1. Nama: Fara, NIM: 2211104021
2. Nama: Mdhmsaf, NIM: 2211104023
3. Nama: Dewi Atika, NIM: 2211104042
4. Nama: Cabine Thither, NIM: 2211104056
5. (kosong)
Jumlah antrian setelah dihapus 1= 4

Data antrian mahasiswa (diurutkan berdasarkan NIM):
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian setelah di-clear= 0

Process returned 0 (0x0) execution time : 44.043 s
Press any key to continue.

```


Deskripsi Program:

Dalam program ini ada tambahan fungsi insertSorted() untuk menata ulang queue. Dalam proses enqueue, node disusun berdasarkan NIM dari terkecil ke terbesar. Dalam mekanismenya, berarti NIM terkecil akan selalu berada di depan queue. Lalu program menghapus data dengan dequeue satu kali, setelahnya data ditampilkan Kembali. Selanjutnya program menghapus semua data dengan clearqueue.

5. Kesimpulan

Praktikum queue memberikan pemahaman mendalam tentang:

- Konsep dasar struktur data queue
- Perbedaan implementasi queue menggunakan array dan linked list
- Operasi-operasi dasar pada queue
- Penerapan queue dalam pemecahan persoalan praktis

Melalui praktikum ini, mahasiswa dapat memahami dan mengimplementasikan struktur data queue dengan berbagai variasi dan pendekatan.