LAPORAN PRAKTIKUM
PERTEMUAN 8
Queue

Nama :

Haza Zaidan Zidna Fann

(2311104056)

Dosen :

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. 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

II. LANDASAN TEORI

Queue adalah struktur data linier yang mengikuti prinsip First In, First Out (FIFO), di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar. Queue digunakan dalam berbagai aplikasi seperti antrian di loket, pengelolaan proses dalam sistem operasi, dan pengolahan data streaming.

Operasi dasar pada Queue:

Enqueue: Menambahkan elemen ke belakang (rear).

Dequeue: Menghapus elemen dari depan (front).

Peek/Front: Menampilkan elemen di depan tanpa menghapusnya.

isEmpty: Memeriksa apakah antrian kosong.

isFull: Memeriksa apakah antrian penuh.

Queue memiliki dua sisi utama:

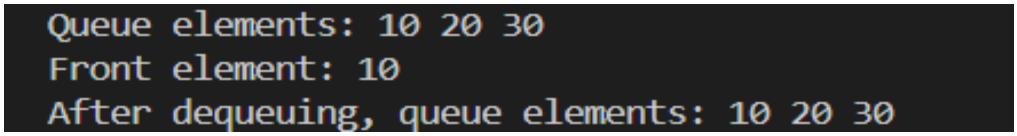
Front: Elemen pertama yang akan keluar.

Rear: Elemen yang terakhir masuk.

Implementasi Queue: Queue dapat diimplementasikan menggunakan array atau linked list. Array lebih sederhana tetapi memerlukan pengelolaan ruang yang efisien, sementara linked list lebih fleksibel dalam ukuran.

III. GUIDED

```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";

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

    return 0;
}
```

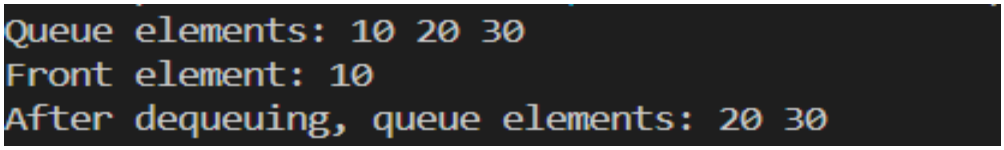


Deklarasi dan Konstruktor:
front dan rear: Mewakili indeks depan dan belakang queue. Kedua variabel ini diinisialisasi dengan nilai -1 yang menandakan bahwa queue kosong saat awal.
Konstruktor Queue(): Mengatur nilai awal front dan rear ke -1.

Metode isFull():
Memeriksa apakah queue sudah penuh dengan membandingkan rear dengan MAX – 1 (ukuran array - 1).
Metode isEmpty():
Memeriksa apakah queue kosong, yang terjadi jika front lebih besar dari rear atau keduanya -1.
Metode enqueue(x):
Menambahkan elemen x ke belakang antrian (dengan memindahkan rear).
Jika queue penuh, mencetak "Queue Overflow".
Metode dequeue():
Menghapus elemen depan antrian dengan meningkatkan nilai front.
Jika queue kosong, mencetak "Queue Underflow".
Metode peek():
Mengembalikan elemen di depan queue tanpa menghapusnya.
Jika queue kosong, mencetak "Queue is empty".
Metode display():
Menampilkan seluruh elemen dalam queue, dimulai dari front hingga rear.

Program Utama (main()):
Menambahkan 3 elemen (10, 20, 30) ke dalam queue menggunakan enqueue().
Menampilkan elemen-elemen dalam queue menggunakan display().
Menampilkan elemen depan menggunakan peek().
Menghapus elemen dari depan dengan dequeue() dan menampilkan elemen-elemen setelahnya.

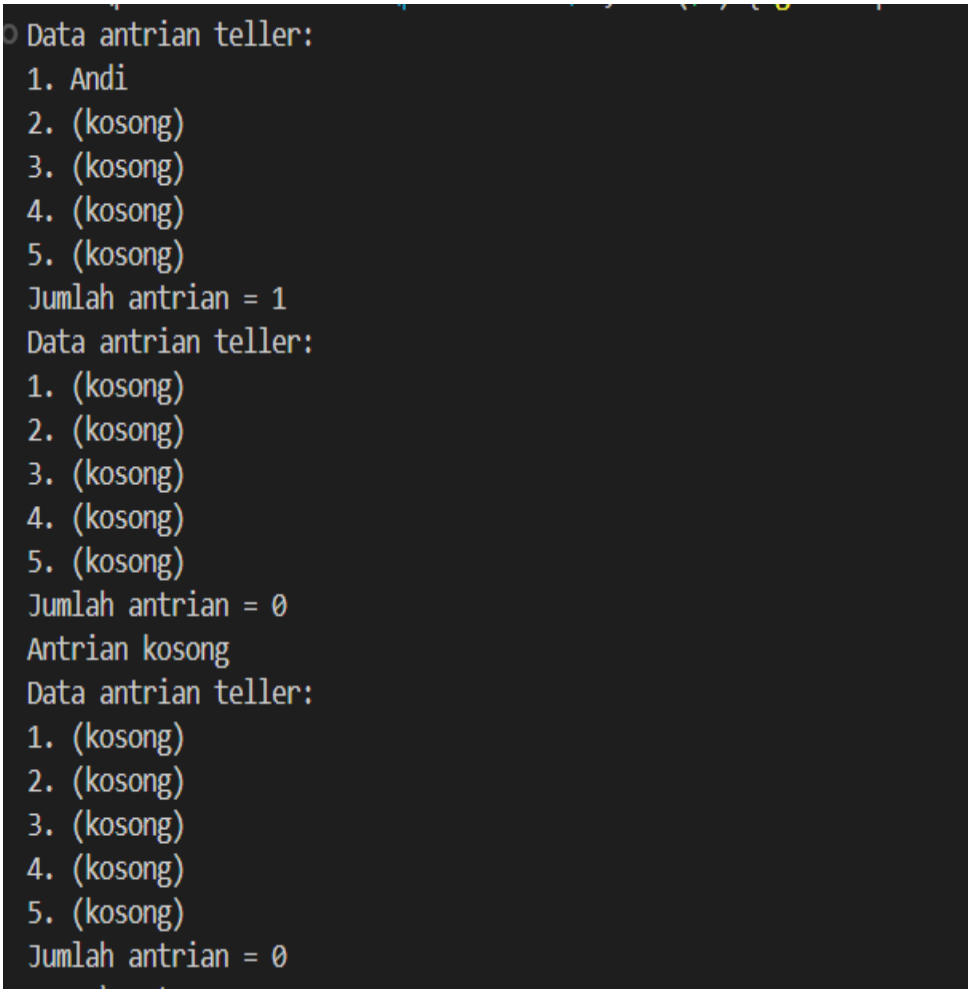
```cpp
#include <iostream>

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;
}
```



Kelas Node:
data: Menyimpan nilai elemen.
next: Menunjuk ke node berikutnya dalam queue.
Konstruktor menginisialisasi data dengan nilai elemen dan next dengan nullptr.
Kelas Queue:
front: Pointer ke elemen depan queue.
rear: Pointer ke elemen belakang queue.
Fungsi:
isEmpty(): Mengecek apakah queue kosong (jika front adalah nullptr).
enqueue(x): Menambahkan elemen baru ke belakang queue.
Jika queue kosong, elemen menjadi front dan rear.
Jika tidak, elemen baru ditambahkan ke node setelah rear, lalu rear diperbarui.
dequeue(): Menghapus elemen depan dari queue.
Jika kosong, mencetak "Queue Underflow".
Jika elemen terakhir dihapus, front dan rear diatur ke nullptr.
peek(): Mengembalikan data dari elemen depan tanpa menghapusnya.
Jika kosong, mencetak "Queue is empty" dan mengembalikan -1.
display(): Menampilkan semua elemen queue dari front ke rear.

Fungsi main:
Menambahkan elemen ke dalam queue: enqueue(10), enqueue(20), enqueue(30).
Menampilkan elemen queue menggunakan display().
Menampilkan elemen depan menggunakan peek().
Menghapus elemen depan menggunakan dequeue(), lalu menampilkan elemen setelah penghapusan.

```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 { // Antrianya 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;
}
```



Konstanta dan Variabel Global:
maksimalQueue: Batas maksimum queue (5 elemen).
front dan back: Penanda awal dan akhir queue.
queueTeller: Array untuk menyimpan elemen queue.
Fungsi Operasi Dasar:
isFull(): Memeriksa apakah queue penuh (jika back == maksimalQueue).
isEmpty(): Memeriksa apakah queue kosong (jika back == 0).
enqueueAntrian(string data): Menambahkan elemen ke belakang queue.
Jika penuh, mencetak "Antrian penuh".
Jika kosong, elemen pertama ditambahkan di indeks 0.
dequeueAntrian(): Menghapus elemen dari depan queue.
Jika kosong, mencetak "Antrian kosong".
Menggeser elemen setelah elemen pertama.
countQueue(): Menghitung jumlah elemen dalam queue (back menunjukkan jumlah elemen).
clearQueue(): Menghapus semua elemen dalam queue dengan mengatur front dan back ke 0.
viewQueue(): Menampilkan elemen queue dan posisi kosong.

Fungsi main():
Menambahkan elemen "Andi" dan "Maya" ke queue.
Menampilkan elemen dalam queue dan menghitung jumlah elemen.

Menghapus elemen depan (dequeue) dan menampilkan queue setelahnya.
Menghapus semua elemen queue dan menampilkan hasilnya.

IV. UNGUIDED

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

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

class Queue {
private:
    Node* front;
    Node* back;
    int size;

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

    bool isFull() {
        return false; // Linked list tidak memiliki batas ukuran tetap
    }

    bool isEmpty() {
        return size == 0;
    }

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

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
        size++;
    }

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

    int count() {
        return size;
    }

    void clear() {
        while (!isEmpty()) {
            dequeue();
        }
    }

    void view() {
        cout << "Isi queue: ";
        Node* current = front;
        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    Queue queue;

    // Input data oleh user
    int jumlahData;
    cout << "Masukkan jumlah nama: ";
    cin >> jumlahData;
    cin.ignore(); // Membersihkan buffer newline

    for (int i = 0; i < jumlahData; i++) {
        string nama;
        cout << "Masukkan nama ke-" << i + 1 << ": ";
        getline(cin, nama);
        queue.enqueue(nama);
    }

    cout << "\nData dalam queue setelah input:\n";
    queue.view();

    return 0;
}
```

```
Masukkan jumlah nama: 2
Masukkan nama ke-1: Haza
Masukkan nama ke-2: Zaidan

Data dalam queue setelah input:
Isi queue: Haza Zaidan
```

Node Structure:
Setiap elemen antrian diwakili oleh Node yang berisi data (nama) dan pointer next ke node berikutnya.
Queue Class:
Attributes:
front: Menunjuk elemen pertama dalam queue.
back: Menunjuk elemen terakhir dalam queue.
size: Menyimpan jumlah elemen dalam queue.
Methods:
enqueue: Menambahkan data ke akhir queue.
dequeue: Menghapus data dari depan queue.
view: Menampilkan semua data dalam queue.
clear: Menghapus semua elemen queue.
isEmpty: Mengecek apakah queue kosong.

Main Function:
Meminta pengguna memasukkan jumlah data dan elemen (nama) untuk dimasukkan ke dalam queue.
Menampilkan isi queue setelah input.

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

struct Mahasiswa {
    string nama;
    long long nim;
};

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

class Queue {
private:
    Node* front;
    Node* back;
    int size;

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

    bool isFull() {
        return false; // Linked list tidak memiliki batas ukuran tetap
    }

    bool isEmpty() {
        return size == 0;
    }

    void enqueue(string nama, long long nim) {
        Node* newNode = new Node();
        newNode->data.nama = nama;
        newNode->data.nim = nim;
        newNode->next = nullptr;

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
        size++;
    }

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

    int count() {
        return size;
    }

    void clear() {
        while (!isEmpty()) {
            dequeue();
        }
    }

    void view() {
        cout << "\nData mahasiswa dalam queue:\n" << 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++;
        }
    }
};

int main() {
    Queue queue;

    // Input data mahasiswa oleh user
    int jumlahMahasiswa;
    cout << "Masukkan jumlah mahasiswa: ";
    cin >> jumlahMahasiswa;
    cin.ignore(); // Membersihkan buffer newline

    for (int i = 0; i < jumlahMahasiswa; i++) {
        string nama;
        long long nim;
        cout << "Masukkan Nama Mahasiswa: ";
        getline(cin, nama);
        cout << "Masukkan NIM Mahasiswa: ";
        cin >> nim;
        cin.ignore(); // Membersihkan buffer newline

        queue.enqueue(nama, nim);
    }

    queue.view();

    cout << "\nJumlah antrian = " << queue.count() << endl;
    return 0;
}
```

```
Masukkan jumlah mahasiswa: 3
Masukkan Nama Mahasiswa: Haza
Masukkan NIM Mahasiswa: 2311104056
Masukkan Nama Mahasiswa: Zidna
Masukkan NIM Mahasiswa: 2311104020
Masukkan Nama Mahasiswa: Fann
Masukkan NIM Mahasiswa: 2311104079

Data mahasiswa dalam queue:

1. Nama: Haza, NIM: 2311104056
2. Nama: Zidna, NIM: 2311104020
3. Nama: Fann, NIM: 2311104079

Jumlah antrian = 3
```

Mahasiswa:
Struct untuk menyimpan data mahasiswa (nama dan NIM).
Node:
Struct untuk menyusun linked list, setiap node berisi Mahasiswa dan pointer ke node berikutnya (next).

Queue Class
Atribut:
front: Pointer ke elemen pertama (kepala) dalam queue.
back: Pointer ke elemen terakhir (ekor) dalam queue.
size: Menyimpan jumlah elemen dalam queue.
Method Utama:
enqueue: Menambahkan data mahasiswa di akhir queue.
dequeue: Menghapus data mahasiswa dari awal queue.
view: Menampilkan semua data mahasiswa dalam queue.
count: Mengembalikan jumlah elemen dalam queue.
clear: Menghapus semua elemen dari queue.
isEmpty: Mengecek apakah queue kosong.

Main Program
Meminta pengguna untuk memasukkan jumlah mahasiswa.
Menginput nama dan NIM mahasiswa, lalu menambahkannya ke dalam queue menggunakan enqueue.
Menampilkan seluruh data mahasiswa menggunakan view.
Menampilkan jumlah elemen dalam queue dengan count.

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

struct Mahasiswa {
    string nama;
    long long nim;
};

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

class Queue {
private:
    Node* front;
    Node* back;
    int size;

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

    bool isFull() {
        return false; // Linked list tidak memiliki batas ukuran tetap
    }

    bool isEmpty() {
        return size == 0;
    }

    void enqueue(string nama, long long nim) {
        Node* newNode = new Node();
        newNode->data.nama = nama;
        newNode->data.nim = nim;
        newNode->next = nullptr;

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            if (newNode->data.nim < front->data.nim) {
                newNode->next = front;
                front = newNode;
            } else {
                Node* current = front;
                while (current->next != nullptr && current->next->data.nim < newNode->data.nim) {
                    current = current->next;
                }
                newNode->next = current->next;
                current->next = newNode;
                if (newNode->next == nullptr) {
                    back = newNode;
                }
            }
        }
        size++;
    }

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

    int count() {
        return size;
    }

    void clear() {
        while (!isEmpty()) {
            dequeue();
        }
    }

    void view() {
        cout << "Data mahasiswa dalam queue:" << 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++;
        }
    }
};

int main() {
    Queue queue;

    // Input data mahasiswa oleh user
    int jumlahMahasiswa;
    cout << "Masukkan jumlah mahasiswa: ";
    cin >> jumlahMahasiswa;
    cin.ignore(); // Membersihkan buffer newline

    for (int i = 0; i < jumlahMahasiswa; i++) {
        string nama;
        long long nim;
        cout << "Masukkan Nama Mahasiswa: ";
        getline(cin, nama);
        cout << "Masukkan NIM Mahasiswa: ";
        cin >> nim;
        cin.ignore(); // Membersihkan buffer newline

        queue.enqueue(nama, nim);
    }

    cout << "\nData mahasiswa setelah diurutkan berdasarkan NIM:\n";
    queue.view();

    cout << "\nJumlah antrian = " << queue.count() << endl;
    return 0;
}
```

```
Masukkan jumlah mahasiswa: 2
Masukkan Nama Mahasiswa: Haza Zaidan
Masukkan NIM Mahasiswa: 2311104056
Masukkan Nama Mahasiswa: Zidna Fann
Masukkan NIM Mahasiswa: 2311104080

Data mahasiswa setelah diurutkan berdasarkan NIM:
Data mahasiswa dalam queue:
1. Nama: Haza Zaidan, NIM: 2311104056
2. Nama: Zidna Fann, NIM: 2311104080
```

Mahasiswa:
Struct untuk menyimpan informasi mahasiswa (nama dan nim).
Node:
Struct untuk membentuk linked list, berisi data (bertipe Mahasiswa) dan pointer next ke node berikutnya.

Queue Class
Atribut:
front: Pointer ke elemen pertama dalam queue.
back: Pointer ke elemen terakhir dalam queue.
size: Menyimpan jumlah elemen dalam queue.
Method Utama:
enqueue:
Menambahkan elemen ke queue berdasarkan prioritas NIM (terkecil di depan).
Jika queue kosong, elemen langsung menjadi front dan back.
Jika NIM elemen baru lebih kecil dari front, elemen baru menjadi front.
Jika NIM elemen baru lebih besar, dicari posisi yang sesuai untuk menyisipkan elemen.
dequeue:
Menghapus elemen dari depan queue (front).
view:
Menampilkan elemen queue dalam urutan dari front ke back.
count:
Mengembalikan jumlah elemen dalam queue.

V. KESIMPULAN

Queue adalah struktur data berbasis FIFO (First In, First Out), di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar, seperti antrean dalam kehidupan sehari-hari. Operasi utama meliputi enqueue (menambahkan elemen di akhir antrian) dan dequeue (menghapus elemen dari awal antrian). Queue dapat diimplementasikan menggunakan array atau linked list, dan sering digunakan dalam sistem komputasi seperti manajemen proses, antrean printer, atau pemrosesan data real-time. Variasi seperti priority queue memungkinkan elemen diproses berdasarkan prioritas, bukan hanya urutan masuk.