Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHub yang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
    a. Asisten Praktikum: AndiniNH
    b. Asisten Praktikum : 4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folder dalam repository akan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Penganalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. **Jam Praktikum:**
    - Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwal praktikum adalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
        - **06.30 - 07.30**: Waktu ini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
        - **07.30 - 0G.30**: Sesi ini mencakup **tutorial, diskusi, dan kasus problem-solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
            - **60 menit pertama**: Tugas terbimbing.
            - **60 menit kedua**: Tugas mandiri.
6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

    nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

    Sebagai contoh:

    STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

8. **Struktur Laporan Praktikum**
    1. **Cover :**

LAPORAN PRAKTIKUM
Modul 8
QUEUE

Disusun Oleh:
KAFKA PUTRA RIYADI - 2311104041

Kelas:
SE 07-02

Dosen :
Wahyu Andi Saputra, S.Pd., M.Eng,

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

2. Tujuan

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

3. Landasan Teori

Queue adalah struktur data yang bekerja dengan prinsip First In, First Out (FIFO), di mana elemen pertama yang masuk adalah elemen pertama yang keluar. Operasi utama pada queue meliputi:

- Enqueue: Menambahkan elemen ke belakang antrian.
- Dequeue: Menghapus elemen dari depan antrian.
- Peek: Melihat elemen di depan tanpa menghapusnya.

Jenis Queue:

- Simple Queue: FIFO standar.
- Circular Queue: Antrian melingkar untuk efisiensi.
- Priority Queue: Elemen diproses berdasarkan prioritas.
- Deque: Elemen bisa ditambah/hapus dari kedua ujung.

4. Guided

Code Program guided1;

Screenshoot codingan guided1 saya bagi menjadi 2 bagian agar tidak blur gambarnya

```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++;
    }
```

Lanjutan dari codingan diatas:

```cpp
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;
}
```

Outputannya:

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
PS D:\STRUKTUR DATA P9\output>
```

Code Program Guided2:

Sama seperti guided1, SS an nya saya bagi menjadi 2 agar tidak blur

```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;
    }
```

Untuk lanjutan Codingannya ada pada halaman selanjutnya.

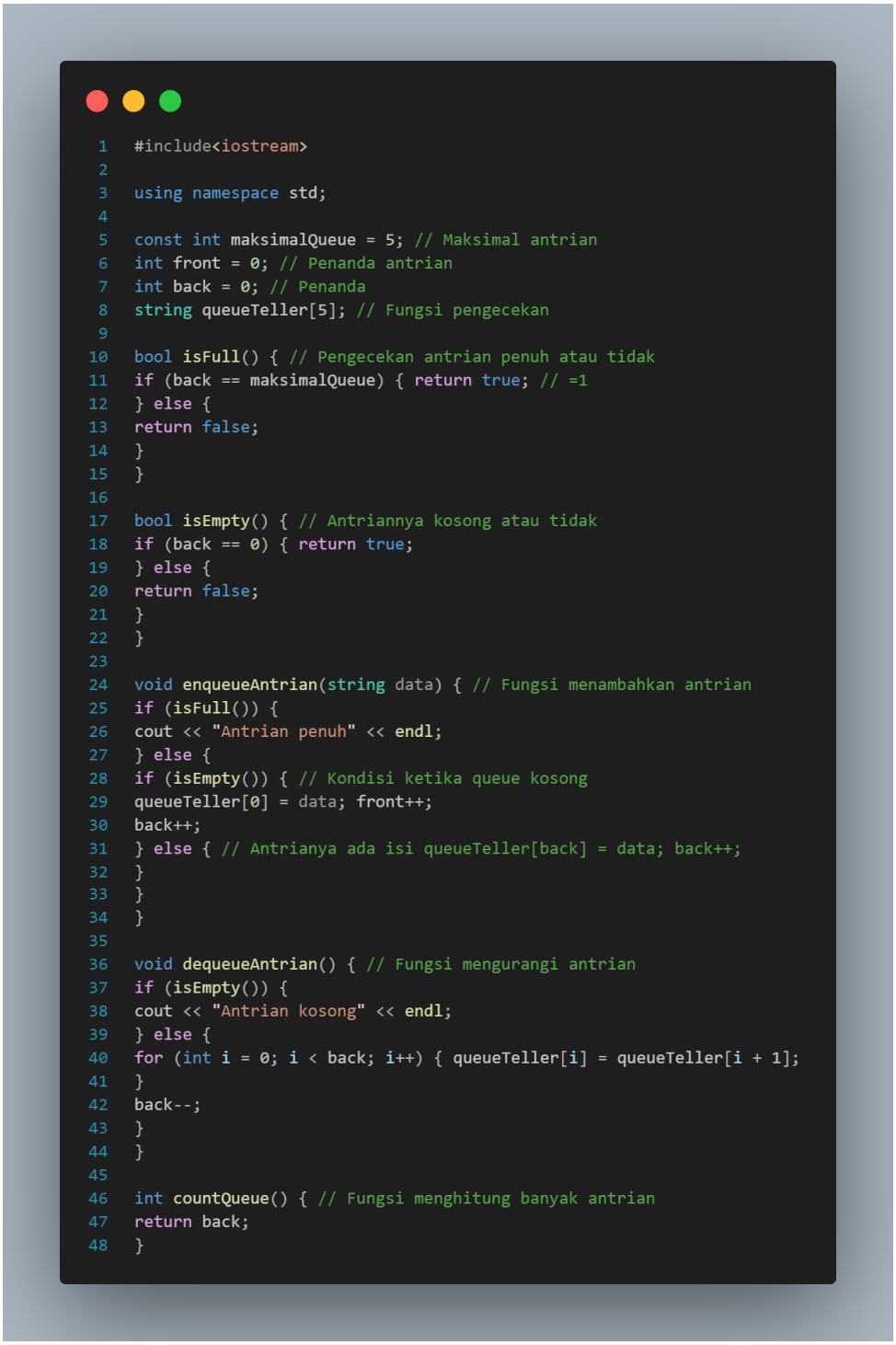
```
    // Mengembalikan elemen depan Queue tanpa menghapusnya
    int peek() {
        if (!isEmpty()) {
            return front->data;
        }
        cout << "Queue is empty\n";
        return -1; // Nilai sentinel
    }

    // 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;
}
```

Outputannya:

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
PS D:\STRUKTUR DATA P9\output>
```

Code Program Guided3:

Sama seperti guided1dan 2, SS an nya saya bagi menjadi 2 agar tidak blur



```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;
}
```

Lanjutan dari code program guided3 ada pada halaman selanjutnya

```
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;
}
```

Outputannya:

```
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 D:\STRUKTUR DATA P9\output>
```

5. **Unguided**

1.

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

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

struct Queue {
    Node* front;
    Node* back;
    int size;

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

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

    void enqueueAntrian(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++;
        cout << data << " ditambahkan ke antrian." << endl;
    }

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

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

        cout << temp->data << " dikeluarkan dari antrian." << endl;
        delete temp;
        size--;
    }
```

Codingan Unguided1 saya bagi menjadi 2 bagian agar tidak blur

Lanjutan dari codingan diatas ada pada halaman selanjutnya

```
    int countQueue() {
        return size;
    }

    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
        cout << "Antrian telah dikosongkan." << endl;
    }

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

        cout << "Data antrian teller:" << endl;
        Node* current = front;
        int position = 1;

        while (current != nullptr) {
            cout << position++ << ". " << current->data << endl;
            current = current->next;
        }
    }
};

int main() {
    Queue queue;

    queue.enqueueAntrian("Andi");
    queue.enqueueAntrian("Maya");

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

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

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

    return 0;
}
```

Outputannya:

```
Andi ditambahkan ke antrian.
Maya ditambahkan ke antrian.
Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Andi dikeluarkan dari antrian.
Data antrian teller:
1. Maya
Jumlah antrian = 1
Maya dikeluarkan dari antrian.
Antrian telah dikosongkan.
Antrian kosong.
Jumlah antrian = 0
PS D:\STRUKTUR DATA P9\output>
```

Telkom University Purwokerto

2.

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

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

struct Queue {
    Node* front;
    Node* back;
    int size;

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

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

    void enqueueAntrian(string nama, string nim) {
        Node* newNode = new Node();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
        size++;
        cout << "Mahasiswa " << nama << " (NIM: " << nim << ") ditambahkan ke antrian." << endl;
    }

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

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

        cout << "Mahasiswa " << temp->nama << " (NIM: " << temp->nim << ") dikeluarkan dari antrian." << endl;
        delete temp;
        size--;
    }
```

Lanjutan dari codingan diatas:

```cpp
int countQueue() {
        return size;
    }

    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
        cout << "Antrian telah dikosongkan." << endl;
    }

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

        cout << "Data antrian mahasiswa:" << endl;
        Node* current = front;
        int position = 1;

        while (current != nullptr) {
            cout << position++ << ". Nama: " << current->nama << ", NIM: " << current->nim << endl;
            current = current->next;
        }
    }
};

int main() {
    Queue queue;
    queue.enqueueAntrian("KAFKA PUTRA RIYADI", "2311104041");
    queue.enqueueAntrian("RonaldoWati", "2311104040");

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

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

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

    return 0;
}
```

Outputannya:

```
Mahasiswa KAFKA PUTRA RIYADI (NIM: 2311104041)
ditambahkan ke antrian.
Mahasiswa RonaldoWati (NIM: 2311104040) ditambahkan ke
antrian.
Data antrian mahasiswa:
1. Nama: KAFKA PUTRA RIYADI, NIM: 2311104041
2. Nama: RonaldoWati, NIM: 2311104040
Jumlah antrian = 2
Mahasiswa KAFKA PUTRA RIYADI (NIM: 2311104041)
dikeluarkan dari antrian.
Data antrian mahasiswa:
1. Nama: RonaldoWati, NIM: 2311104040
Jumlah antrian = 1
Mahasiswa RonaldoWati (NIM: 2311104040) dikeluarkan dari
antrian.
Antrian telah dikosongkan.
Antrian kosong.
Jumlah antrian = 0
```

Telkom University Purwokerto

3.

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

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

struct Queue {
    Node* front;
    Node* back;
    int size;

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

    bool isEmpty() {
        return size == 0;
    }
    void enqueueAntrian(string nama, string nim) {
        Node* newNode = new Node();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            Node* current = front;
            Node* previous = nullptr;

            while (current != nullptr && current->nim < nim) {
                previous = current;
                current = current->next;
            }

            if (previous == nullptr) {
                newNode->next = front;
                front = newNode;
            } else {
                previous->next = newNode;
                newNode->next = current;
                if (current == nullptr) {
                    back = newNode;
                }
            }
        }
        size++;
        cout << "Mahasiswa " << nama << " (NIM: " << nim << ") ditambahkan ke antrian." << endl;
    }

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

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

        cout << "Mahasiswa " << temp->nama << " (NIM: " << temp->nim << ") dikeluarkan dari antrian." << endl;
        delete temp;
        size--;
    }
```

Lanjutan dari codingan diatas

```cpp
    int countQueue() {
        return size;
    }

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

        cout << "Data antrian mahasiswa:" << endl;
        Node* current = front;
        int position = 1;

        while (current != nullptr) {
            cout << position++ << ". " << current->nama << " (NIM: " << current->nim << ")" << endl;
            current = current->next;
        }
    }
};

int main() {
    Queue queue;
    string nama, nim;

    for (int i = 0; i < 5; i++) {
        cout << "Masukkan nama mahasiswa ke-" << i+1 << ": ";
        getline(cin, nama);
        cout << "Masukkan NIM mahasiswa ke-" << i+1 << ": ";
        getline(cin, nim);
        queue.enqueueAntrian(nama, nim);
    }

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

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

    return 0;
}
```

Outputannya:

```
Masukkan nama mahasiswa ke-1: AGUS
Masukkan NIM mahasiswa ke-1: 2311104040
Mahasiswa AGUS (NIM: 2311104040) ditambahkan ke antrian.
Masukkan nama mahasiswa ke-2: KAFKA
Masukkan NIM mahasiswa ke-2: 2311104041
Mahasiswa KAFKA  (NIM: 2311104041) ditambahkan ke antrian.
Masukkan nama mahasiswa ke-3: ASEP
Masukkan NIM mahasiswa ke-3: 2311104042
Mahasiswa ASEP (NIM: 2311104042) ditambahkan ke antrian.
Masukkan nama mahasiswa ke-4: CECEP
Masukkan NIM mahasiswa ke-4: 2311104039
Mahasiswa CECEP (NIM: 2311104039) ditambahkan ke antrian.
Masukkan nama mahasiswa ke-5: DADANG
Masukkan NIM mahasiswa ke-5: 2311104038
Mahasiswa DADANG (NIM: 2311104038) ditambahkan ke antrian.
Data antrian mahasiswa:
1. DADANG (NIM: 2311104038)
2. CECEP (NIM: 2311104039)
3. AGUS (NIM: 2311104040)
4. KAFKA  (NIM: 2311104041)
5. ASEP (NIM: 2311104042)
Jumlah antrian = 5
Mahasiswa DADANG (NIM: 2311104038) dikeluarkan dari antrian.
Data antrian mahasiswa:
1. CECEP (NIM: 2311104039)
2. AGUS (NIM: 2311104040)
3. KAFKA  (NIM: 2311104041)
4. ASEP (NIM: 2311104042)
Jumlah antrian = 4
```

6. Kesimpulan

Kesimpulannya, Queue adalah struktur data yang menerapkan prinsip First In, First Out (FIFO), di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar. Operasi utama dalam queue meliputi menambahkan elemen ke belakang

antrian (enqueue), menghapus elemen dari depan antrian (dequeue), dan melihat elemen depan tanpa menghapusnya (peek). Ada beberapa jenis queue, antara lain Simple Queue yang mengikuti aturan FIFO standar, Circular Queue yang lebih efisien dengan antrian melingkar, Priority Queue yang memproses elemen berdasarkan prioritas, dan Deque yang memungkinkan penambahan dan penghapusan elemen dari kedua ujung antrian.