LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL 7
QUEUE

Disusun Oleh :

Muhammad Hamzah Haifan Ma'ruf

231102091

Dosen :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

A. **Dasar Teori**

Pada struktur data Queue atau antrian adalah sekumpulan data yang mana penambahan elemen hanya bisa dilakukan pada suatu ujung disebut dengan sisi belakang (rear), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain (disebut dengan sisi depan atau front).

Pada Stack atau tumpukan menggunakan prinsip "Masuk terakhir keluar pertama"atau LIFO (Last In First Out), Maka pada Queue atau antrian prinsip yang digunakan adalah "Masuk Pertama Keluar Pertama" atau FIFO (First In First Out).

Queue atau antrian banyak kita jumpai dalam kehidupan sehari-hari, ex: antrian Mobil diloket Tol, Antrian mahasiswa Mendaftar, dll. Contoh lain dalam bidang komputer adalah pemakaian sistem komputer berbagi waktu(time-sharing computer system) dimana ada sejumlah pemakai yang akan menggunakan sistem tersebut secara serempak.

Pada Queue atau antrian Terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung satunya dimana membutuhkan variabel Head dan Tail (depan/front, belakang/rear).

Operasi-operasi Queue

1. Create()

Untuk menciptakan dan menginisialisasi Queue

2. IsEmpty()

Untuk memeriksa apakah Antrian masih kosong

3. IsFull()

Untuk mengecek apakah Antrian sudah penuh atau belum

4. Enqueue ()

Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang

5. Dequeue()

Digunakan untuk menghapus elemen terdepan/pertama (head) dari Antrian

6. Clear()

Untuk menghapus elemen-elemen Antrian dengan cara membuat Tail dan Head = -1

7. Tampil()

Untuk menampilkan nilai-nilai elemen Antrian

B. **Guided**

Guided 1

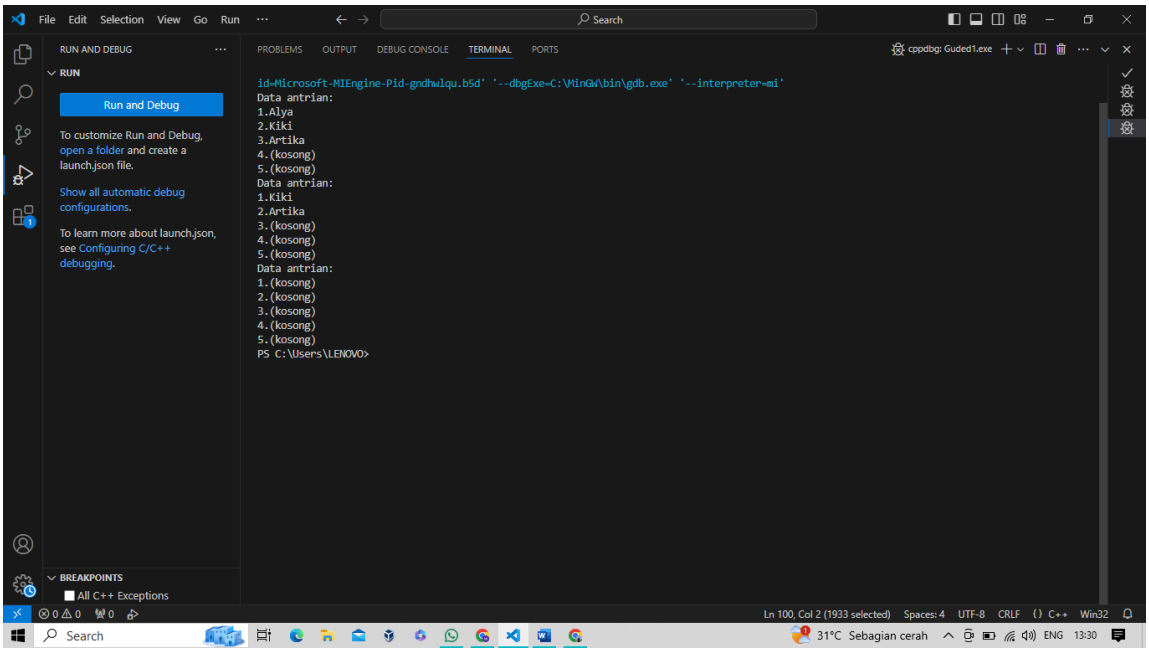
Source Code

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

//mengecek antrian apakah sudah penuh
const int maksimalAntrian = 5;
int front = 0;
int back = 0;
string queueTeller[5];

bool isFull(){
    if (back == maksimalAntrian){
        return true;
    } else {
        return false;
    }
}

//mengecek antrian apakah masih kosong
bool isEmpty(){
    if(back == 0){
        return true;
    } else {
        return false;
    }
}

//menambah antrian
void tambahData(string nama){
    if (isFull ()){
        cout<<"Antrian sudah penuh."<<endl;
    } else {
        if (isEmpty ()){
            queueTeller[0] = nama;
            front++;
            back++;
        } else{
            queueTeller[back] = nama;
            back++;
        }
    }
}

//mengurangi antrian
void kurangAntrian (){
    if (isEmpty ()){
```

```
        cout<<"Antrian kosong."<<endl;
    } else {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        back --;
    }
}

//menghitung banyak antrian
int count () {
    return back;
}

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

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

//main fungsi
int main (){
    tambahData("Alya");
    tambahData("Kiki");
    tambahData("Artika");
    viewQueue();

    //mengurangi antrian
    kurangAntrian();
```

```
    viewQueue();

    //menghapus seluruh antrian
    clearQueue();
    viewQueue();
}
```

Screenshots Output :



Deskripsi :

Program di atas adalah implementasi dari sistem antrian menggunakan array di bahasa C++. Program ini menggunakan variabel global maksimalAntrian, front, back, dan array queueTeller untuk mengelola antrian. Fungsi isFull mengecek apakah antrian penuh dengan membandingkan back dengan maksimalAntrian, sedangkan fungsi isEmpty mengecek apakah antrian kosong dengan melihat apakah back bernilai 0. Fungsi tambahData menambahkan elemen baru ke antrian, dan jika antrian penuh, ia menampilkan pesan. Jika antrian kosong, elemen baru ditempatkan pada indeks 0 dan kedua nilai front dan back ditingkatkan, sebaliknya, elemen baru ditempatkan di posisi back dan back ditingkatkan. Fungsi kurangAntrian mengeluarkan elemen dari antrian dengan menggeser elemen-elemen lainnya ke depan dan menurunkan back. Fungsi count mengembalikan jumlah elemen dalam antrian, dan clearQueue mengosongkan seluruh antrian dengan mengatur ulang back dan front ke 0. Fungsi viewQueue menampilkan semua elemen dalam antrian atau (kosong) jika tidak ada elemen pada posisi tertentu. Fungsi main menambahkan tiga elemen ke antrian, menampilkan antrian, mengurangi satu elemen dari antrian, menampilkan antrian lagi, menghapus seluruh antrian, dan menampilkan antrian yang kosong.

C. **Unguided**

Unguided 1

Source Code

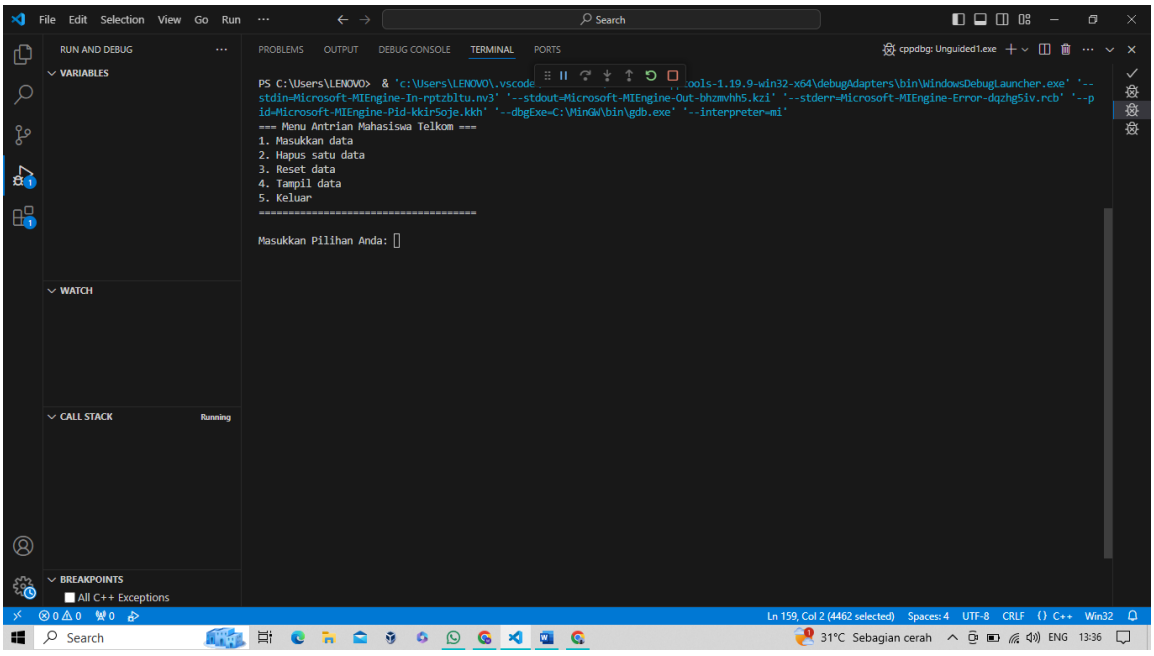
```
#include <iostream>
using namespace std;

//simpul untuk node dalam linked list
struct Mahasiswa {
    string nama_091;
    string nim_091;
    Mahasiswa* next_091;
};

//kelas queue yang menggunakan linked list
class Queue {
private:
    //pointer ke depan antrian
    Mahasiswa* front;
    //pointer ke belakang antrian
    Mahasiswa* back;

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

    //fungsi untuk mengecek apakah antrian kosong
    bool isEmpty() {
        return front == nullptr;
    }

    //fungsi untuk menambahkan data ke antrian
    void enqueue(string nama_091_, string nim_091_) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama_091 = nama_091_;
        newNode->nim_091 = nim_091_;
        newNode->next_091 = nullptr;

        //jika antrian kosong, node baru menjadi front dan back
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            //jika antrian tidak kosong, tambahkan node baru ke belakang
dan update back
            back->next_091 = newNode;
            back = newNode;
```

```
        }
        cout << "Data berhasil dimasukkan ke dalam antrian" << endl;
    }

    //untuk menghapus data dari antrian
    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            //jika antrian tidak kosong, hapus node pertama dan update
front
            Mahasiswa* temp = front;
            front = front->next_091;
            delete temp;
            cout << "Data berhasil dihapus dari antrian" << endl;
        }
    }

    //untuk menghitung jumlah data dalam antrian
    int countQueue() {
        int count = 0;
        Mahasiswa* temp = front;
        while (temp != nullptr) {
            count++;
            temp = temp->next_091;
        }
        return count;
    }

    //untuk menghapus semua data dari antrian
    void clearQueue() {
        while (!isEmpty()) {
            dequeue();
        }
        cout << "Data berhasil di-reset" << endl;
    }

    //untuk menampilkan data dalam antrian
    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            cout << "Data antrian Mahasiswa: " << endl;
            Mahasiswa* temp = front;
            int pos = 1;
            while (temp != nullptr) {
                cout << pos << ". Nama: " << temp->nama_091 << " || NIM:
" << temp->nim_091 << endl;
                temp = temp->next_091;
```

```
                pos++;
            }
        }
    }
};

int main() {
    Queue queue;
    int choice;

    do {
        cout << "=== Menu Antrian Mahasiswa Telkom ===" << endl;
        cout << "1. Masukkan data" << endl;
        cout << "2. Hapus satu data" << endl;
        cout << "3. Reset data" << endl;
        cout << "4. Tampil data" << endl;
        cout << "5. Keluar" << endl;
        cout << "====================================\n" << endl;
        cout << "Masukkan Pilihan Anda: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                string nama, nim;
                cout << "Masukkan Nama Mahasiswa : ";
                cin.ignore();
                getline(cin, nama);
                cout << "Masukkan NIM Mahasiswa  : ";
                cin >> nim;
                queue.enqueue(nama, nim);
                break;
            }
            case 2: {
                if (queue.isEmpty()) {
                    cout << "Antrian kosong" << endl;
                } else {
                    queue.dequeue ();
                }
                break;
            }
            case 3: {
                if (queue.isEmpty()) {
                    cout << "Antrian kosong" << endl;
                } else {
                    queue.clearQueue();
                }
                break;
            }
            case 4: {
```

```
                queue.viewQueue();
                break;
            }
            case 5: {
                cout << "Terima kasih telah menggunakan layanan kami ;)"
<< endl;
                break;
            }
            default: {
                cout << "Pilihan yang Anda masukkan tidak valid" << endl;
                break;
            }
        }

        cout << endl;

    } while (choice != 5);

    return 0;
}
```

Screenshots Output :



Deskripsi :

Program di atas adalah struktur data linked list digunakan dengan membuat sebuah simpul yang terdiri dari data mahasiswa (nama dan NIM) serta pointer ke simpul selanjutnya. Kelas Queue mengelola antrian dengan dua pointer, yaitu front untuk menunjukkan elemen terdepan dan back untuk elemen terbelakang. Fungsi-fungsi yang ada mencakup pengecekan apakah antrian kosong, penambahan data ke antrian,

penghapusan data dari antrian, penghitungan jumlah data dalam antrian, penghapusan seluruh data dari antrian, dan penampilan data dalam antrian. Program utama memberikan menu interaktif untuk mengoperasikan antrian, termasuk masukkan data, penghapusan satu data, penghapusan seluruh data, dan penampilan seluruh data dalam antrian. Pengguna dapat memilih opsi yang diinginkan sampai memilih untuk keluar dari program.

Unguided 2

Source Code

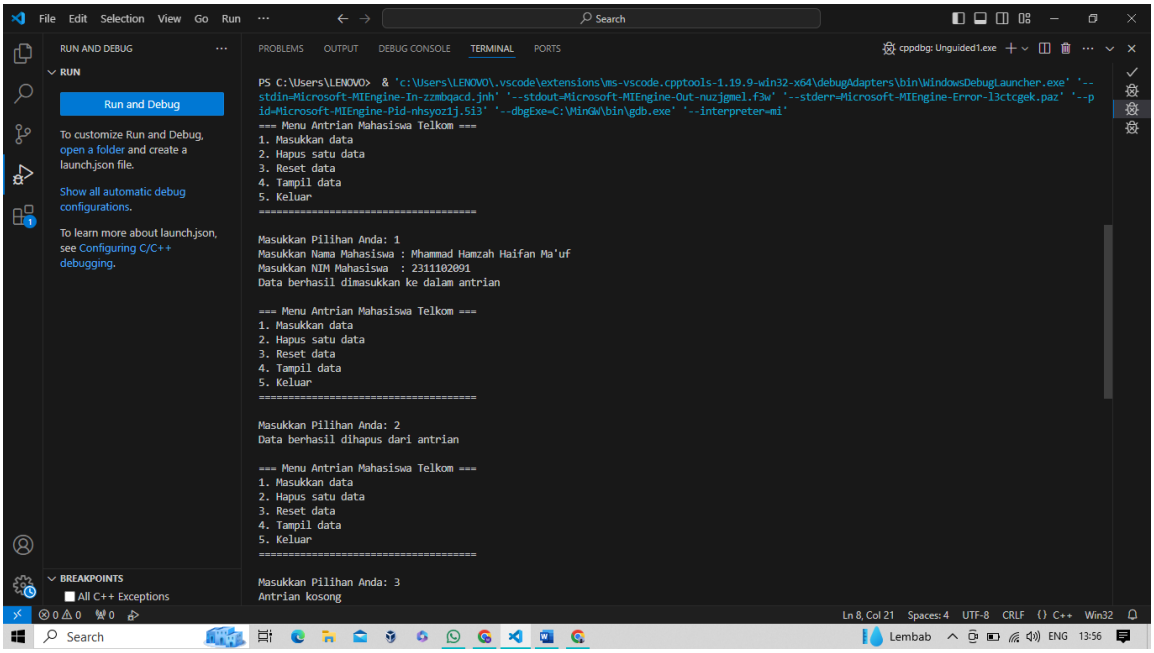
```
#include <iostream>
using namespace std;

//simpul untuk node dalam linked list
struct Mahasiswa {
    string nama_091;
    string nim_091;
    Mahasiswa* next_091;
};

//kelas queue yang menggunakan linked list
class Queue {
private:
    //pointer ke depan antrian
    Mahasiswa* front;
    //pointer ke belakang antrian
    Mahasiswa* back;

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

    //fungsi untuk mengecek apakah antrian kosong
    bool isEmpty() {
        return front == nullptr;
    }

    //fungsi untuk menambahkan data ke antrian
    void enqueue(string nama_091_, string nim_091_) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama_091 = nama_091_;
        newNode->nim_091 = nim_091_;
        newNode->next_091 = nullptr;

        //jika antrian kosong, node baru menjadi front dan back
        if (isEmpty()) {
```

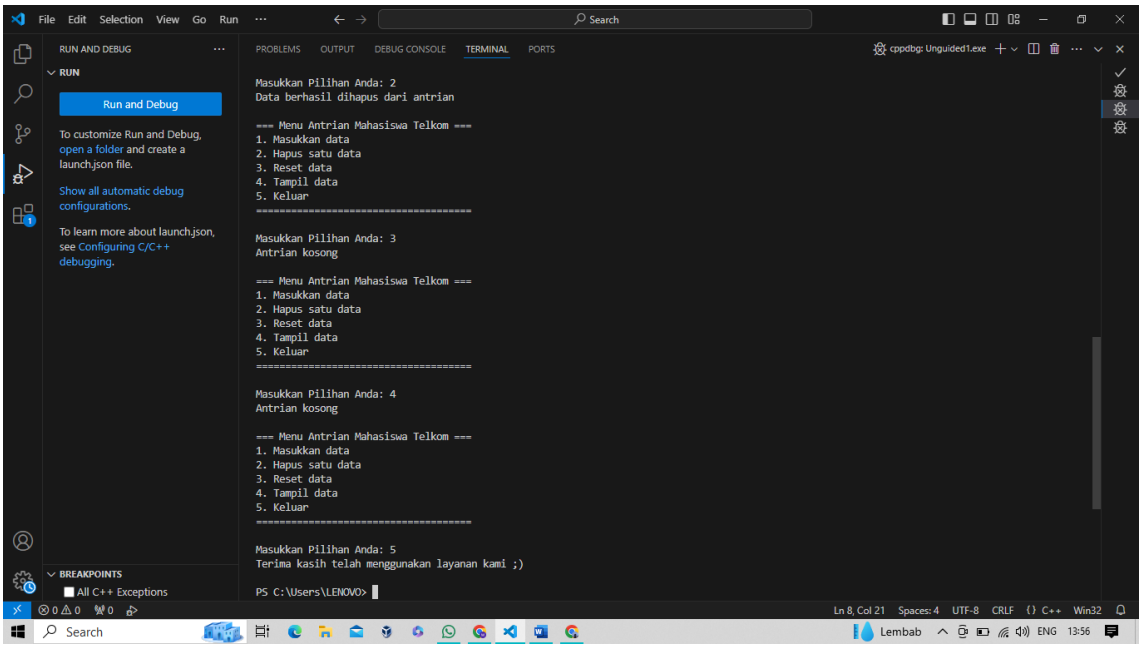
```
            front = newNode;
            back = newNode;
        } else {
            //jika antrian tidak kosong, tambahkan node baru ke belakang
dan update back
            back->next_091 = newNode;
            back = newNode;
        }
        cout << "Data berhasil dimasukkan ke dalam antrian" << endl;
    }

    //untuk menghapus data dari antrian
    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            //jika antrian tidak kosong, hapus node pertama dan update
front
            Mahasiswa* temp = front;
            front = front->next_091;
            delete temp;
            cout << "Data berhasil dihapus dari antrian" << endl;
        }
    }

    //untuk menghitung jumlah data dalam antrian
    int countQueue() {
        int count = 0;
        Mahasiswa* temp = front;
        while (temp != nullptr) {
            count++;
            temp = temp->next_091;
        }
        return count;
    }

    //untuk menghapus semua data dari antrian
    void clearQueue() {
        while (!isEmpty()) {
            dequeue();
        }
        cout << "Data berhasil di-reset" << endl;
    }

    //untuk menampilkan data dalam antrian
    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
```

```
            cout << "Data antrian Mahasiswa: " << endl;
            Mahasiswa* temp = front;
            int pos = 1;
            while (temp != nullptr) {
                cout << pos << ". Nama: " << temp->nama_091 << " || NIM:
" << temp->nim_091 << endl;
                temp = temp->next_091;
                pos++;
            }
        }
    }
};

int main() {
    Queue queue;
    int choice;

    do {
        cout << "=== Menu Antrian Mahasiswa Telkom ===" << endl;
        cout << "1. Masukkan data" << endl;
        cout << "2. Hapus satu data" << endl;
        cout << "3. Reset data" << endl;
        cout << "4. Tampil data" << endl;
        cout << "5. Keluar" << endl;
        cout << "=====================================\n" << endl;
        cout << "Masukkan Pilihan Anda: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                string nama, nim;
                cout << "Masukkan Nama Mahasiswa : ";
                cin.ignore();
                getline(cin, nama);
                cout << "Masukkan NIM Mahasiswa  : ";
                cin >> nim;
                queue.enqueue(nama, nim);
                break;
            }
            case 2: {
                if (queue.isEmpty()) {
                    cout << "Antrian kosong" << endl;
                } else {
                    queue.dequeue ();
                }
                break;
            }
            case 3: {
                if (queue.isEmpty()) {
```

```
                    cout << "Antrian kosong" << endl;
                } else {
                    queue.clearQueue();
                }
                break;
            }
            case 4: {
                queue.viewQueue();
                break;
            }
            case 5: {
                cout << "Terima kasih telah menggunakan layanan kami ;)"
<< endl;
                break;
            }
            default: {
                cout << "Pilihan yang Anda masukkan tidak valid" << endl;
                break;
            }
        }

        cout << endl;

    } while (choice != 5);

    return 0;
}
```

Screenshots Output :





Deskripsi :

Program di atas adalah implementasi data linked list digunakan untuk merepresentasikan setiap elemen dalam antrian, yang pada kasus ini adalah data mahasiswa yang terdiri dari nama dan NIM. Setiap elemen dalam antrian direpresentasikan oleh simpul (node) yang memiliki dua atribut, yaitu nama_091 dan nim_091 untuk menyimpan informasi nama dan NIM mahasiswa, serta pointer next_091 untuk menunjukkan ke simpul berikutnya dalam antrian.

Kelas Queue memiliki dua pointer, yaitu front untuk menunjukkan elemen terdepan dalam antrian, dan back untuk menunjukkan elemen terbelakang. Fungsi-fungsi yang disediakan dalam kelas Queue mencakup pengecekan apakah antrian kosong (isEmpty), penambahan data ke dalam antrian (enqueue), penghapusan data dari antrian (dequeue), penghitungan jumlah data dalam antrian (countQueue), penghapusan seluruh data dari antrian (clearQueue), dan penampilan seluruh data dalam antrian (viewQueue).

Program utama memberikan menu interaktif untuk mengoperasikan antrian, termasuk masukkan data mahasiswa baru, penghapusan satu data mahasiswa, penghapusan seluruh data dalam antrian, dan penampilan seluruh data dalam antrian. Pengguna dapat memilih opsi yang diinginkan sampai memilih untuk keluar dari program.

D. Kesimpulan

Pertama, program nomor 1 adalah konsep dasar dari antrian dengan menggunakan array, di mana setiap elemen dalam antrian direpresentasikan oleh indeks array. Program ini memanfaatkan variabel global dan fungsi untuk mengelola antrian, termasuk pengecekan apakah antrian penuh atau kosong, penambahan elemen baru,

pengurangan elemen, penghitungan jumlah elemen, penghapusan seluruh elemen, dan penampilan seluruh elemen dalam antrian.

Kedua, program nomor 2 adalah peningkatan dari program sebelumnya dengan mengganti struktur data array menjadi linked list. Dalam program ini, setiap elemen dalam antrian direpresentasikan oleh simpul (node) dalam linked list, yang memiliki atribut nama dan NIM mahasiswa serta pointer ke simpul berikutnya. Konsep antrian dipertahankan dengan menggunakan pointer front dan back untuk menunjukkan elemen terdepan dan terbelakang dalam antrian.

Ketiga, kesimpulan dari kedua penjelasan tersebut adalah bahwa penggunaan linked list memberikan fleksibilitas dan efisiensi yang lebih baik dalam pengelolaan antrian dibandingkan dengan array, terutama dalam hal penambahan atau penghapusan elemen. Struktur data linked list memungkinkan penambahan atau penghapusan elemen dengan kompleksitas waktu konstan, yang membuatnya lebih cocok untuk aplikasi yang membutuhkan operasi antrian yang sering dan dinamis. Selain itu, kedua program menyediakan antarmuka yang interaktif untuk mengoperasikan antrian, yang memungkinkan pengguna untuk dengan mudah memasukkan, menghapus, dan menampilkan data dalam antrian.

E. **Referensi**

Guest (no date) 'Pengertian queue dalam C++,' *KASKUS*.

https://www.kaskus.co.id/thread/5ec54d20facb95558a5496e1/pengertian-queue-dalam-c.

nblognlife (no date). https://www.nblognlife.com/2014/05/c-konsep-queue.html