

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :

NAMA : D'sharlendita Febianda Aurelia
NIM : 2311102069

Dosen :

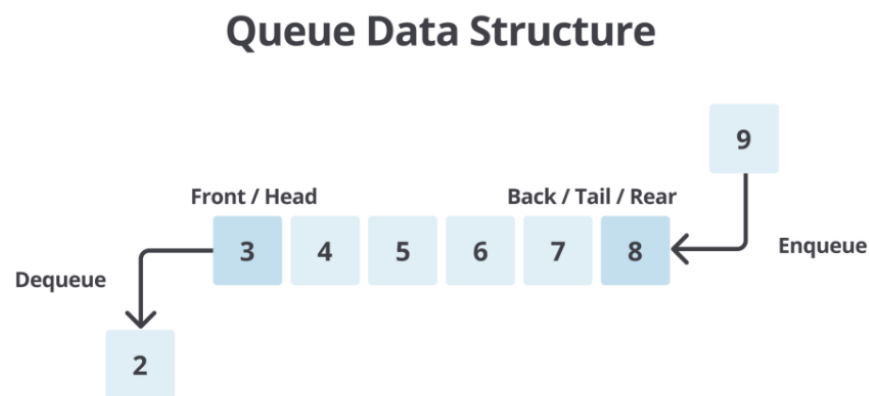
Wahyu Andi Saputra, S.pd., M,Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

1. Pengertian Queue

Queue atau dalam Bahasa Indonesia yang berarti antrian adalah struktur data yang Menyusun elemen-elemen data dalam urutan linier. Prinsip dasar dari struktur data ini adalah “First In, First Out” (FIFO) yang berarti elemen data yang pertama dimasukkan ke dalam antrian akan menjadi yang pertama pula untuk dikeluarkan.



Gambar A1.1 *Queue*

2. Jenis-Jenis Queue dalam Strktur Data

a. Simple Queue

Simple Queue merupakan struktur data linier yang mengikuti prinsip First-In-Firs-Out (FIFO), dimana elemen ditambahkan di bagian belakang (back) dan dikerluarkan dari depan (head).

- Koleksi terurut dari jenis data pembanding
- Struktur antrian adalah FIFO (First In, First Out)
- Ketika elemen baru ditambahkan, semua elemen yang ditambahkan sebelum elemen baru harus dihapus untuk menghapus elemen baru

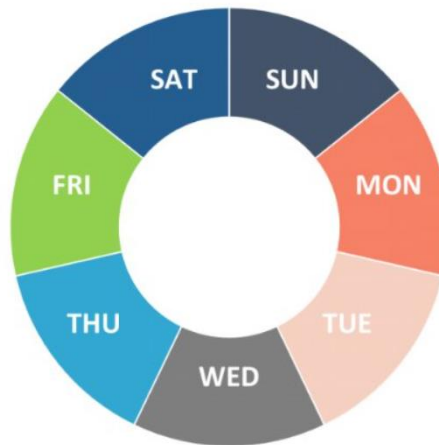


Gambar A2.1 *Simple Queue*

b. Circular Queue

Circular Queue adalah kasus khusus dari antrian sederhana dimana anggota terakhir dihubungkan dengan anggota pertama. Hasilnya, terbentuklah struktur seperti lingkaran.

- Node terakhir terhubung ke node pertama
- Dikenal sebagai Ring Buffer, node terhubung ujung ke ujung
- Penyisipan dilakukan di depan antrian, dan penghapusan dilakukan di akhir antrian



Gambar B2.1 *Circular Queue*

c. Priority Queue

Dalam Priority Queue node akan memiliki beberapa prioritas yang telah ditentukan sebelumnya dalam antrian prioritas. Node dengan prioritas paling kecil akan menjadi yang pertama dikeluarkan dari antrian. Penyisipan dilakukan sesuai urutan kedatangan node.



Gambar C2.1 *Priority Queue*

d. Double-Ended Queue (Deque)

Dalam Double-Ended Queue, penyisipan dan penghapusan dapat terjadi di ujung depan dan belakang antrian.



Gambar D2.1 *Double Ended Queue (Deque)*

3. Operasi Dasar Pada Struktur Data Queue

- Enqueue (): Proses menambahkan atau menyimpan elemen ke akhir Queue
- Dequeue (): Proses menghapus atau mengakses elemen dari depan Queue
- Peek (): Digunakan untuk menempatkan elemen di depan Queue tanpa menghapusnya
- Initialize (): Membuat Queue kosong
- isFull (): Memeriksa apakah Queue sudah penuh
- isEmpty (): Memeriksa apakah Queue kosong
- Size (): Menghitung jumlah elemen yang ada di dalam Queue

B. Guided

Guided 1

Program Queue

Source Code:

```
//D'sharlendita Febianda Aurelia
//2311102069

#include <iostream>

using namespace std;

const int maksimalQueue = 5; // Batas maksimal antrian
int front = 0; // indeks awal antrian
int back = 0; // indeks akhir antrian
string queueTeller[maksimalQueue]; // array untuk menyimpan
elemen antrian

//Fungsi untuk memeriksa apakah antrian penuh
bool isFull() {
    return back == maksimalQueue;
}

//Fungsi untuk memeriksa apakah antrian kosong
bool isEmpty() {
    return back == 0;
}

//Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string data) {
    if (isFull()) {
        cout << "Antrian penuh" << endl;
    } else {
        queueTeller[back] = data;
        back ++;
    }
}

//Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    }
}
```

```

    } else {
        for (int i = 0; i < back; i++) {
            queueTeller[i] = queueTeller[i + 1];
        }
        queueTeller[back - 1] = ""; //membersihkan data terakhir
        back--;
    }
}

//Fungsi untuk menghitung jumlah elemen dalam antrian
int countQueue() {
    return back;
}

//Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    for (int i = 0; i < back; i++) {
        queueTeller[i] = "";
    }
    back = 0;
    front = 0;
}

//Fungsi untuk menampilkan semua elemen dalam antrian
void viewQueue() {
    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;
}

```

Screenshots 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
PS C:\parktikum struktur data\MODUL 7\guided>

```

Inset window content:

```

Nama   |: D'sharlendita Febianda Aurelia
NIM    |: 2311102069
Kelas |: IF - 11 - B

```

Status bar: Ln 1, Col 7 | 77 characters | 100% | Window | UTF-8

Deskripsi:

Program ini mensimulasikan antrian teller di bank. Pengguna dapat menambahkan dan menghapus nasabah dari antrian, serta melihat informasi tentang antrian seperti jumlah nasabah dan urutannya. Program ini menggunakan array queueTeller untuk menyimpan elemen antrian dan memiliki batas maksimal antrian yang ditentukan oleh variabel maksimalQueue. Indeks antrian menggunakan skema "first-in, first-out"

(FIFO), di mana nasabah yang pertama kali masuk antrian akan dilayani terlebih dahulu.

C. Unguided

Unguided 1

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

Source Code :

```
// D'sharlendita Febianda Aurelia
// 2311102069

#include <iostream>
#include <string>

using namespace std;

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

class Queue {
private:
    Node* head;
    Node* tail;
    int maksimalQueue;
    int count;

public:
    Queue(int maxQueue) : head(nullptr), tail(nullptr),
maksimalQueue(maxQueue), count(0) {}

    // Fungsi untuk memeriksa apakah antrian penuh
    bool isFull() {
        return count >= maksimalQueue;
    }

    // Fungsi untuk memeriksa apakah antrian kosong
    bool isEmpty() {
```



```

        return count == 0;
    }

    // Fungsi untuk menambahkan elemen ke antrian
    void enqueueAntrian(string data) {
        if (isFull()) {
            cout << "Antrian penuh" << endl;
            return;
        }
        Node* newNode = new Node{data, nullptr};
        if (isEmpty()) {
            head = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }
        count++;
    }

    // Fungsi untuk menghapus elemen dari antrian
    void dequeueAntrian() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        Node* temp = head;
        head = head->next;
        delete temp;
        count--;
        if (isEmpty()) {
            tail = nullptr;
        }
    }

    // Fungsi untuk menghitung jumlah elemen dalam antrian
    int countQueue() {
        return count;
    }

    // Fungsi untuk mengosongkan semua elemen dalam antrian
    void clearQueue() {
        while (head != nullptr) {

```

```

        Node* temp = head;
        head = head->next;
        delete temp;
    }
    tail = nullptr;
    count = 0;
    cout << "Semua data di Queue telah dihapus" << endl;
}

// Fungsi untuk menampilkan semua elemen dalam antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* current = head;
    int index = 1;
    while (current != nullptr) {
        cout << index << ". " << current->data << endl;
        current = current->next;
        index++;
    }
    for (int i = index; i <= maksimalQueue; i++) {
        cout << i << ". (kosong)" << endl;
    }
}
};

int main() {
    const int maksimalQueue = 5; // Batas maksimal antrian
    Queue antrian(maksimalQueue);

    antrian.enqueueAntrian("Andi");
    antrian.enqueueAntrian("Maya");
    antrian.viewQueue();
    cout << "Jumlah antrian = " << antrian.countQueue() << endl;

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

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

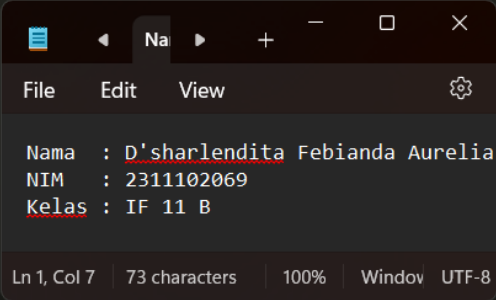
    return 0;
}

```

```
}
```

Screenshots 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
Semua data di Queue telah dihapus
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0
PS C:\parktikum struktur data\MODUL 7\unguided>
```



The screenshot shows a Notepad++ window with a dark theme. The title bar reads 'Na'. The menu bar includes 'File', 'Edit', and 'View'. The text content is as follows:

```
Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF 11 B
```

The status bar at the bottom indicates 'Ln 1, Col 7', '73 characters', '100%', 'Window', and 'UTF-8'.

Deskripsi:

Program ini adalah implementasi antrian (queue) berbasis linked list dalam C++. Program menggunakan struktur `Node` untuk merepresentasikan elemen antrian dan kelas `Queue` untuk mengelola antrian. Dalam fungsi `main()`, program mendemonstrasikan penambahan, penghapusan, penampilan, dan pengosongan elemen antrian.

Unguided 2

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

Source Code:

```
// D'sharlendita Febianda Aurelia
// 2311102069

#include <iostream>
#include <string>

using namespace std;

struct Node {
    string nama;
    Long Long NIM;
    Node* next;
};

class QUEUE {
private:
    Node* head;
    Node* tail;
    int maksimalQueue;
    int count;

public:
    QUEUE(int maxQueue) : head(nullptr), tail(nullptr),
maksimalQueue(maxQueue), count(0) {}

    // Fungsi untuk memeriksa apakah antrian penuh
    bool isFull(){
        return count >= maksimalQueue;
    }

    // Fungsi untuk memeriksa apakah antrian kosong
    bool isEmpty() {
        return count == 0;
    }

    // Fungsi untuk menambahkan elemen ke antrian
    void enqueueAntrian(string nama, Long Long NIM) {
        if (isFull()) {
```

```

        cout << "Antrian penuh" << endl;
        return;
    }
    Node* newNode = new Node(nama, NIM, nullptr);
    if (isEmpty()) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
    count++;
    cout << "Data dengan nama (" << nama << ") ditambahkan"
<< endl << endl;
}

// Fungsi untuk menghapus elemen dari antrian berdasarkan
nama
void dequeueAntrian() {
    cout << "-----"
-----" << endl << endl;
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
        return;
    }
    string namaHapus;
    cout << "Masukkan nama mahasiswa yang ingin dihapus: ";
    cin.ignore();
    getline(cin, namaHapus);

    Node* temp = head;
    Node* prev = nullptr;
    bool found = false;

    while (temp != nullptr) {
        if (temp->nama == namaHapus) {
            found = true;
            if (prev == nullptr) {
                head = temp->next;
            } else {
                prev->next = temp->next;
            }
            if (temp == tail) {

```

```

        tail = prev;
    }
    delete temp;
    count--;
    cout << "Mahasiswa dengan nama (" << namaHapus <<
") telah dihapus" << endl << endl;
    break;
}
prev = temp;
temp = temp->next;
}

if (!found) {
    cout << "Mahasiswa dengan nama '" << namaHapus << "'
tidak ditemukan" << endl << endl;
}
cout << "-----"
-----" << endl << endl;
}

// Fungsi untuk menghitung jumlah elemen dalam antrian
int countQueue() {
    return count;
}

// Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
    tail = nullptr;
    count = 0;
    cout << "Semua data di Queue telah dihapus" << endl <<
endl;
}

// Fungsi untuk menampilkan semua elemen dalam antrian
void viewQueue() {
    cout << "-----"
-----" << endl << endl;
    cout << "Data Mahasiswa:" << endl << endl;
}

```

```

        Node* current = head;
        int index = 1;
        while (current != nullptr) {
            cout << index << ". Nama: " << current->nama << " |
NIM: " << current->NIM << endl;
            current = current->next;
            index++;
        }
        for (int i = index; i <= maksimalQueue; i++) {
            cout << i << ". (kosong)"<< endl;
        }
        cout << endl;
        cout << "Jumlah antrian = " << countQueue() << endl;
        cout << endl;
        cout << "-----"
        -----" << endl << endl;
    }
};

int main() {
    const int maksimalQueue = 5; // Batas maksimal antrian
    QUEUE dita(maksimalQueue);
    int pilih;
    string nama;
    Long Long NIM;

    do {
        cout << endl;
        cout << "Menu:" << endl << endl;
        cout << "-----"
        -----" << endl << endl;
        cout << "1. Input Data Mahasiswa" << endl;
        cout << "2. Lihat Data Mahasiswa Yang Terdaftar Di Queue"
<< endl;
        cout << "3. Hapus Data Mahasiswa berdasarkan nama" <<
endl;
        cout << "4. Hapus Seluruh Data Mahasiswa dari Queue" <<
endl;
        cout << "5. Keluar dari program\n" << endl << endl;
        cout << "-----"
        -----" << endl << endl;
        cout << "Pilih menu: ";
        cin >> pilih;
    }
};

```

```

cout << endl;

switch (pilih) {
    case 1:
        cout << "-----
-----" << endl << endl;
        cout << "Masukan nama: ";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukan NIM: ";
        cin >> NIM;
        dita.enqueueAntrian(nama, NIM);
        cout << "-----
-----" << endl << endl;
        break;
    case 2:
        dita.viewQueue();
        break;
    case 3:
        dita.dequeueAntrian();
        break;
    case 4:
        cout << "-----
-----" << endl << endl;
        dita.clearQueue();
        cout << "-----
-----" << endl << endl;
        break;
    case 5:
        cout << "-----
-----" << endl << endl;
        cout << "Terimakasih" << endl << endl; // keluar
dari program
        cout << "-----
-----" << endl << endl;
        break;
    default:
        cout << "-----
-----" << endl << endl;
        cout << "Pilihan anda tidak tersedia!" << endl <<
endl; // jika pilihan tidak valid
        cout << "-----
-----" << endl << endl;

```



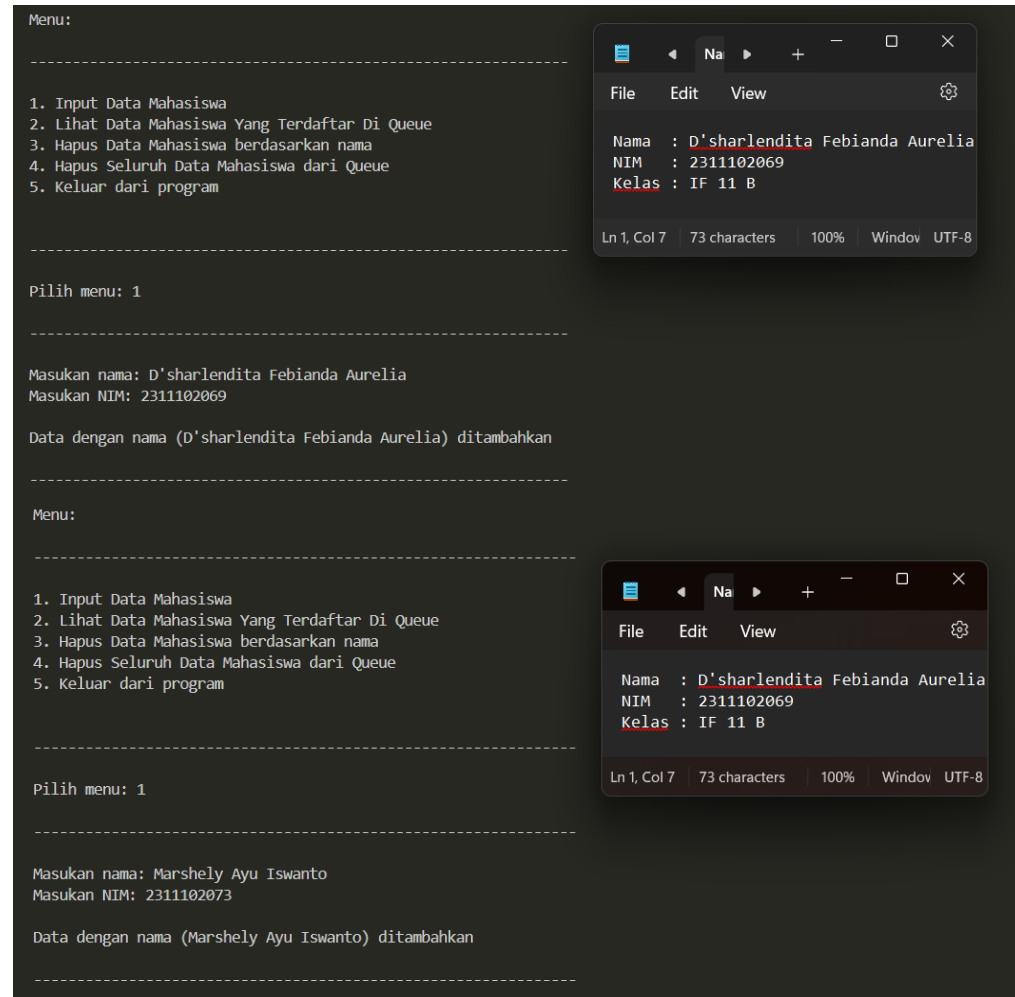
```

    }
} while (pilih != 5); // ulangi sampai pengguna memilih untuk
keluar

return 0;
}

```

Screenshot Output:



Menu:

1. Input Data Mahasiswa
2. Lihat Data Mahasiswa Yang Tendaftar Di Queue
3. Hapus Data Mahasiswa berdasarkan nama
4. Hapus Seluruh Data Mahasiswa dari Queue
5. Keluar dari program

Pilih menu: 2

Data Mahasiswa:

1. Nama: D'sharlendita Febianda Aurelia | NIM: 2311102069
2. Nama: Marshely Ayu Iswanto | NIM: 2311102073
3. (kosong)
4. (kosong)
5. (kosong)

Jumlah antrian = 2

Menu:

1. Input Data Mahasiswa
2. Lihat Data Mahasiswa Yang Tendaftar Di Queue
3. Hapus Data Mahasiswa berdasarkan nama
4. Hapus Seluruh Data Mahasiswa dari Queue
5. Keluar dari program

Pilih menu: 3

Masukkan nama mahasiswa yang ingin dihapus: Marshely Ayu Iswanto

Mahasiswa dengan nama (Marshely Ayu Iswanto) telah dihapus

Menu:

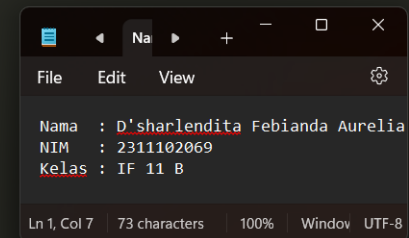
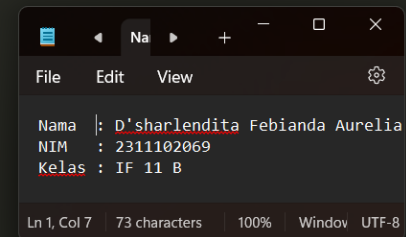
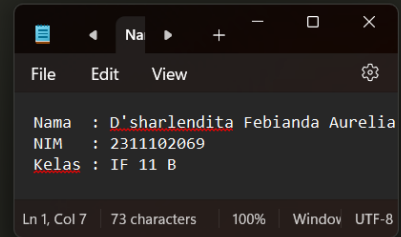
1. Input Data Mahasiswa
2. Lihat Data Mahasiswa Yang Tendaftar Di Queue
3. Hapus Data Mahasiswa berdasarkan nama
4. Hapus Seluruh Data Mahasiswa dari Queue
5. Keluar dari program

Pilih menu: 2

Data Mahasiswa:

1. Nama: D'sharlendita Febianda Aurelia | NIM: 2311102069
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)

Jumlah antrian = 1



Menu:

1. Input Data Mahasiswa
2. Lihat Data Mahasiswa Yang Tendaftar Di Queue
3. Hapus Data Mahasiswa berdasarkan nama
4. Hapus Seluruh Data Mahasiswa dari Queue
5. Keluar dari program

Pilih menu: 4

Semua data di Queue telah dihapus

Menu:

1. Input Data Mahasiswa
2. Lihat Data Mahasiswa Yang Tendaftar Di Queue
3. Hapus Data Mahasiswa berdasarkan nama
4. Hapus Seluruh Data Mahasiswa dari Queue
5. Keluar dari program

Pilih menu: 2

Data Mahasiswa:

1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)

Jumlah antrian = 0

Menu:

1. Input Data Mahasiswa
2. Lihat Data Mahasiswa Yang Tendaftar Di Queue
3. Hapus Data Mahasiswa berdasarkan nama
4. Hapus Seluruh Data Mahasiswa dari Queue
5. Keluar dari program

Pilih menu: 5

Terimakasih

PS C:\parktikum struktur data\MODUL 7\unguided>

```
Na
File Edit View
Nama |: D'sharlendita Febianda Aurelia
NIM  : 2311102069
Kelas : IF 11 B
Ln 1, Col 7 | 73 characters | 100% | Window UTF-8
```

```
Na
File Edit View
Nama : D'sharlendita Febianda Aurelia
NIM  : 2311102069
Kelas : IF 11 B
Ln 1, Col 7 | 73 characters | 100% | Window UTF-8
```

```
Na
File Edit View
Nama |: D'sharlendita Febianda Aurelia
NIM  : 2311102069
Kelas : IF 11 B
Ln 1, Col 7 | 73 characters | 100% | Window UTF-8
```

Deskripsi:

Program ini adalah implementasi antrian (queue) menggunakan struktur data linked list untuk mengelola data mahasiswa. Program ini memiliki fitur untuk menambah, menghapus, menampilkan, dan mengosongkan antrian mahasiswa berdasarkan nama dan NIM.

D. Kesimpulan

Praktikum Modul Queue memberikan pemahaman mendalam mengenai struktur data queue dan penggunaannya dalam pemrograman. Melalui praktikum ini, saya memahami konsep FIFO (First In, First Out) dan operasi dasar queue, seperti enqueue (menambah elemen), dequeue (menghapus elemen), isFull (memeriksa apakah antrian penuh), dan isEmpty (memeriksa apakah antrian kosong).

E. Referensi

- Asisten Praktikum. (2024, 22 Mei). “Modul 7 Queue”. Diakses pada 22 Mei 2024, dari Learning Management System. 2024
- Maulana, Rizki. (2023, 29 November). Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya. Diakses pada 22 Mei 2024, dari <https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/>
- Karan, Rashmi. (2024, 19 Januari). Queue Data Structure: Types, Implementation, Application. Diakses pada 22 Mei 2024, dari <https://www.shiksha.com/online-courses/articles/queue-data-structure-types-implementation-applications/>
- Nurchaliza, Rachmatia. (2024, 5 April). Peran Penting Memahami Struktur Data dalam Ilmu Komputer. Diakses pada 22 Mei 2024, dari <https://dif.telkomuniversity.ac.id/peran-penting-memahami-struktur-data/>