

LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL VII QUEUE



Disusun Oleh :

NAUFAL THORIQ MUZHAFAR

2311102078

Dosen

WAHYU ANDI SAPUTRA, S.Pd., M.Eng

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

A. Dasar Teori

Struktur Data **Queue** adalah konsep dasar dalam ilmu komputer yang digunakan untuk menyimpan dan mengelola data dalam urutan tertentu. Ini mengikuti prinsip “Masuk Pertama, Keluar Pertama” (FIFO), di mana elemen pertama yang ditambahkan ke antrian adalah elemen pertama yang dihapus. Antrian biasanya digunakan dalam berbagai algoritma dan aplikasi karena kesederhanaan dan efisiensinya dalam mengelola aliran data.

Operasi Dasar pada Queue

Beberapa operasi dasar Queue dalam Struktur Data adalah:

Operasi 1: enqueue()

Menyisipkan elemen di akhir antrian, yaitu di bagian belakang.

Langkah-langkah berikut harus diambil untuk mengantrekan (memasukkan) data ke dalam antrian:

- Periksa apakah antrian sudah penuh.
- Jika antrian penuh, kembalikan kesalahan overflow dan keluar.
- Jika antrian belum penuh, tambah penunjuk belakang untuk menunjuk ke ruang kosong berikutnya.
- Tambahkan elemen data ke lokasi antrian, di mana bagian belakangnya menunjuk.
- Kembali sukses.

```
void queueEnqueue(int data)
{
    // Check queue is full or not
    if (capacity == rear) {
        printf("\nQueue is full\n");
        return;
    }
    // Insert element at the rear
    else {
        queue[rear] = data;
        rear++;
    }
}
```

```
return;  
}
```

Operasi 2: dequeue()

Operasi ini menghapus dan mengembalikan elemen yang berada di ujung depan antrian.

Langkah-langkah berikut diambil untuk melakukan operasi dequeue:

- Periksa apakah antriannya kosong.
- Jika antrian kosong, kembalikan error underflow dan keluar.
- Jika antrian tidak kosong, akses data yang ditunjuk bagian depan.
- Tingkatkan penunjuk depan untuk menunjuk ke elemen data berikutnya yang tersedia.
- Keberhasilan Kembali.

```
void queueDequeue()  
{  
    // If queue is empty  
    if (front == rear) {  
        printf("\nQueue is empty\n");  
        return;  
    }  
    // Shift all the elements from index 2  
    // till rear to the left by one  
    else {  
        for (int i = 0; i < rear - 1; i++) {  
            queue[i] = queue[i + 1];  
        }  
        // decrement rear  
        rear--;  
    }  
    return;  
}
```

Operasi 3: Front()

Operasi ini mengembalikan elemen di ujung depan tanpa menghapusnya.

Langkah-langkah berikut diambil untuk melakukan operasi depan:

- Jika antrian kosong, kembalikan nilai paling minimum.
- jika tidak, kembalikan nilai depan.

```
// Function to get front of queue
int front(Queue* queue)
{
    if (isempty(queue))
        return INT_MIN;
    return queue->arr[queue->front];
}
```

Operasi 4: Back()

Operasi ini mengembalikan elemen di bagian belakang tanpa menghapusnya.

Langkah-langkah berikut diambil untuk melakukan operasi belakang:

- Jika antrian kosong, kembalikan nilai paling minimum.
- jika tidak, kembalikan nilai belakangnya.

```
// Function to get rear of queue
int back(Queue* queue)
{
    if (isEmpty(queue))
        return INT_MIN;
    return queue->arr[queue->rear];
}
```

Operasi 5: isEmpty():

Operasi ini mengembalikan nilai boolean yang menunjukkan apakah antrian kosong atau tidak.

Langkah-langkah berikut diambil untuk melakukan operasi Kosong:

- Periksa apakah nilai depan sama dengan -1 atau tidak, jika ya maka return true berarti antrian kosong.
- Jika tidak, kembalikan salah, berarti antrian tidak kosong

```
// This function will check whether
```

```
// the queue is empty or not:
```

```
bool isEmpty()
```

```
{
```

```
if (front == -1)
```

```
return true;
```

```
else
```

```
return false;
```

```
}
```

Operasi 6 : isFull()

Operasi ini mengembalikan nilai boolean yang menunjukkan apakah antrian sudah penuh atau tidak.

Langkah-langkah berikut diambil untuk melakukan operasi isFull():

- Periksa apakah nilai depan sama dengan nol dan belakang sama dengan kapasitas antrian jika ya maka kembalikan true.
- Jika tidak, kembalikan salah

```
// This function will check
```

```
// whether the queue is full or not.
```

```
bool isFull()
```

```
{
```

```
if (front == 0 && rear == MAX_SIZE - 1) {
```

```
return true;
```

```
}
```

```
return false;
```

```
}
```

B. Guided

Guided 1

Source Code

```
#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 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 mengkosongkan 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()
{
    // 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;
}

```

Screenshots Output

```

PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_7> cd "m:\Tugas Naufal\Praktikum\Stru
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 M:\Tugas Naufal\Praktikum\Strukdat\Modul_7>

```

Naufal Thoriq Muzhaffar
2311102078
IF-11-B

Ln 3, Col 8 | 42 characters | 100% | Window | UTF-8

Deskripsi Program

Kode program ini merupakan implementasi sederhana dari struktur data antrian (queue), yang menggunakan array untuk menyimpan data antrian. Dalam kode tersebut, fungsi-fungsi dasar seperti penambahan (`enqueueAntrian`) dan penghapusan (`dequeueAntrian`) elemen dari antrian, serta fungsi-fungsi untuk memeriksa apakah antrian penuh (`isFull`) atau kosong (`isEmpty`), menghitung jumlah elemen dalam antrian (`countQueue`), dan mengosongkan antrian (`clearQueue`) telah diimplementasikan. Variabel `front` dan `back` digunakan untuk

menandai posisi depan dan belakang antrian. Operasi-operasi pada antrian, seperti penambahan elemen, penghapusan elemen, pengosongan antrian, dan penampilan antrian, dieksekusi dalam fungsi `main()`.

C. Unguided

Unguided 1

Source Code

```
#include <iostream>
using namespace std;
struct Node
{
    string namaMahasiswa;
    string nim;
    Node *next;
};
class Queue
{
private:
    Node *front;
    Node *back;
public:
    Queue()
    {
        front = nullptr;
        back = nullptr;
    }
    bool isEmpty()
    {
        return (front == nullptr);
    }
    void enqueue(string namaMahasiswa, string nim)
    {
        Node *newNode = new Node{namaMahasiswa, nim, nullptr};
        if (isEmpty())
        {
            front = newNode;
            back = newNode;
        }
        else
        {
            back->next = newNode;
            back = newNode;
        }
    }
}
```

```

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

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

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

int countQueue()
{
    int count = 0;

```

```

        Node *current = front;
        while (current != nullptr)
        {
            count++;
            current = current->next;
        }
        return count;
    }
};

void displayMenu()
{
    cout << "\nMenu Antrian Teller:" << endl;
    cout << "1. Tambah Antrian" << endl;
    cout << "2. Hapus Antrian" << endl;
    cout << "3. Tampilkan Antrian" << endl;
    cout << "4. Bersihkan Antrian" << endl;
    cout << "5. Hitung Antrian" << endl;
    cout << "6. Keluar" << endl;
    cout << "Pilih operasi: ";
}

int main()
{
    Queue q;
    int choice;
    string namaMahasiswa, nim;
    do
    {
        displayMenu();
        cin >> choice;
        switch (choice)
        {
            case 1:
                cout << "Masukkan Nama Mahasiswa: ";
                cin.ignore(); // Mengabaikan newline yang tersisa di input
                getline(cin, namaMahasiswa);
                cout << "Masukkan NIM: ";
                cin >> nim;
                q.enqueue(namaMahasiswa, nim);
                break;
            case 2:
                q.dequeue();
                break;
            case 3:
                q.viewQueue();
                break;
            case 4:
                q.clearQueue();

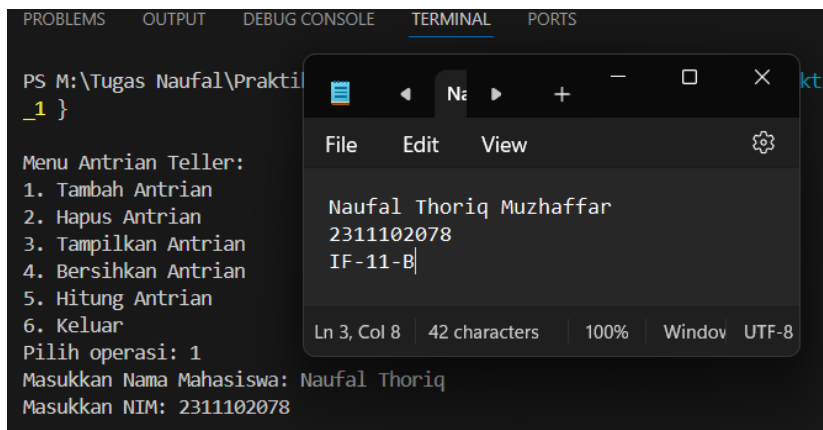
```

```

        break;
    case 5:
        cout << "Jumlah antrian = " << q.countQueue() << endl;
        break;
    case 6:
        cout << "Keluar dari program." << endl;
        break;
    default:
        cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
    }
} while (choice != 6);
return 0;
}

```

Screenshots Output



Deskripsi Program

Program ini menggunakan linked list untuk mengimplementasikan struktur data queue antrian mahasiswa. Setiap mahasiswa direpresentasikan sebagai node dalam linked list, yang menyimpan informasi seperti nama dan NIM serta pointer ke node berikutnya. Operasi-operasi utama termasuk menambah, menghapus, menampilkan, membersihkan, dan menghitung jumlah mahasiswa dalam antrian.

Unguided 2

Source Code

```

#include <iostream>
using namespace std;
struct Node
{
    string namaMahasiswa;
    string nim;
    Node *next;
}

```

```

};
class Queue
{
private:
    Node *front;
    Node *back;

public:
    Queue()
    {
        front = nullptr;
        back = nullptr;
    }
    bool isEmpty()
    {
        return (front == nullptr);
    }
    void enqueue(string namaMahasiswa, string nim)
    {
        Node *newNode = new Node{namaMahasiswa, nim, nullptr};
        if (isEmpty())
        {
            front = newNode;
            back = newNode;
        }
        else
        {
            back->next = newNode;
            back = newNode;
        }
    }
    void dequeue()
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
        {
            Node *temp = front;
            front = front->next;
            if (front == nullptr)
            {
                back = nullptr;
            }
            delete temp;
            cout << "Antrian Berhasil Dihapus" << endl;
        }
    }

```

```

    }
    void viewQueue()
    {
        if (isEmpty())
        {
            cout << "Antrian kosong." << endl;
        }
        else
        {
            cout << "Data antrian mahasiswa:" << endl;
            Node *current = front;
            int index = 1;
            while (current != nullptr)
            {
                cout << index << ". Nama: " << current->namaMahasiswa << ",
NIM: " << current -> nim << endl;
                current = current->next;
                index++;
            }
        }
    }
    void clearQueue()
    {
        while (!isEmpty())
        {
            dequeue();
        }
    }
    int countQueue()
    {
        int count = 0;
        Node *current = front;
        while (current != nullptr)
        {
            count++;
            current = current->next;
        }
        return count;
    }
};

void displayMenu()
{
    cout << "\nMenu Antrian Teller:" << endl;
    cout << "1. Tambah Antrian" << endl;
    cout << "2. Hapus Antrian" << endl;
    cout << "3. Tampilkan Antrian" << endl;
    cout << "4. Bersihkan Antrian" << endl;
    cout << "5. Hitung Antrian" << endl;
}

```

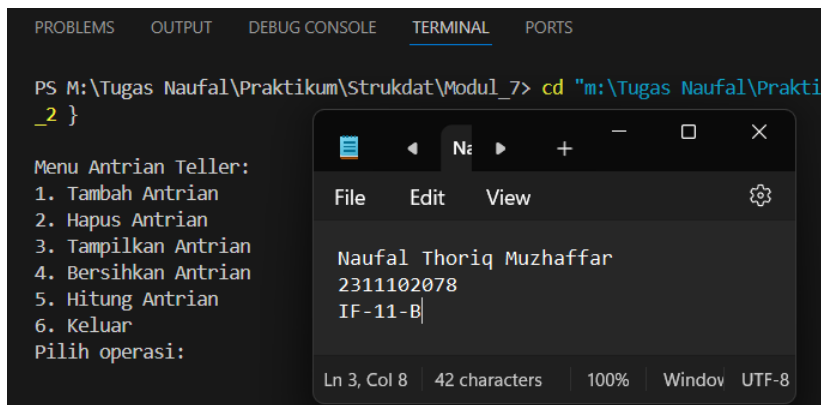
```

        cout << "6. Keluar" << endl;
        cout << "Pilih operasi: ";
    }
int main()
{
    Queue q;
    int choice;
    string namaMahasiswa, nim;
    do
    {
        displayMenu();
        cin >> choice;
        switch (choice)
        {
            case 1:
                cout << "Masukkan Nama Mahasiswa: ";
                cin.ignore(); // Mengabaikan newline yang tersisa di input
buffer
                getline(cin, namaMahasiswa);
                cout << "Masukkan NIM: ";
                cin >> nim;
                q.enqueue(namaMahasiswa, nim);
                break;
            case 2:
                q.dequeue();
                break;
            case 3:
                q.viewQueue();
                break;
            case 4:
                q.clearQueue();
                break;
            case 5:
                cout << "Jumlah antrian = " << q.countQueue() << endl;
                break;
            case 6:
                cout << "Keluar dari program." << endl;
                break;
            default:
                cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
        }
    } while (choice != 6);
    return 0;
}

```

Screenshots Output

Menu



The terminal window displays a menu for a queue system. The menu options are: 1. Tambah Antrian, 2. Hapus Antrian, 3. Tampilkan Antrian, 4. Bersihkan Antrian, 5. Hitung Antrian, 6. Keluar. The user has selected option 2. A notepad window is open in the foreground, showing the text: Naufal Thoriq Muzhaffar, 2311102078, IF-11-B.

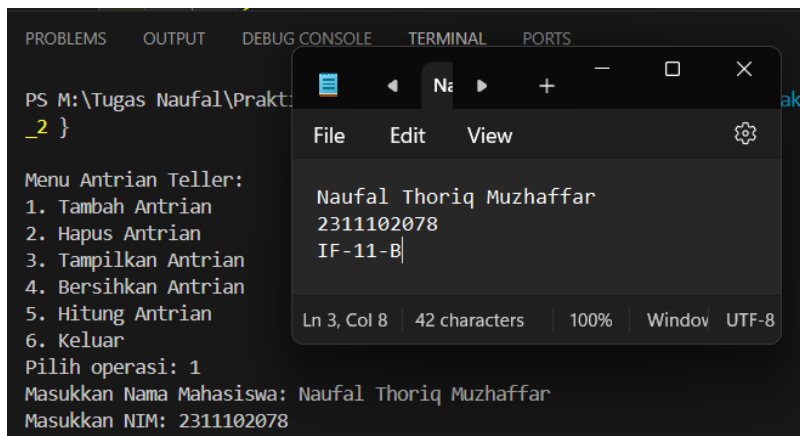
```
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_7> cd "m:\Tugas Naufal\Praktikum\Strukdat\Modul_7"
2 }
```

Menu Antrian Teller:

1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar

Pilih operasi:

Tambah Antrian



The terminal window shows the 'Tambah Antrian' operation. The user has entered the name 'Naufal Thoriq Muzhaffar' and the NIM '2311102078'. The terminal output shows: Masukkan Nama Mahasiswa: Naufal Thoriq Muzhaffar, Masukkan NIM: 2311102078.

```
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_7> cd "m:\Tugas Naufal\Praktikum\Strukdat\Modul_7"
2 }
```

Menu Antrian Teller:

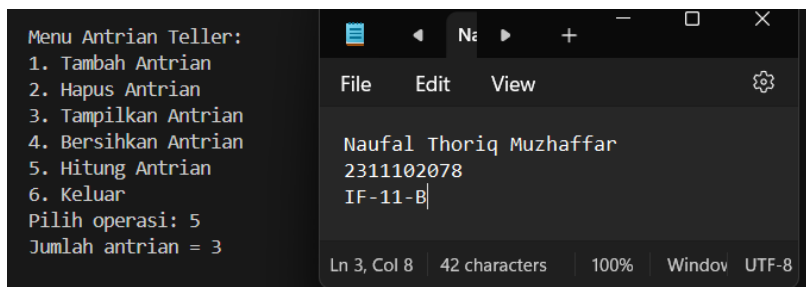
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar

Pilih operasi: 1

Masukkan Nama Mahasiswa: Naufal Thoriq Muzhaffar

Masukkan NIM: 2311102078

Hitung Antrian



The terminal window shows the 'Hitung Antrian' operation. The user has entered the name 'Naufal Thoriq Muzhaffar' and the NIM '2311102078'. The terminal output shows: Masukkan Nama Mahasiswa: Naufal Thoriq Muzhaffar, Masukkan NIM: 2311102078, Jumlah antrian = 3.

```
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_7> cd "m:\Tugas Naufal\Praktikum\Strukdat\Modul_7"
2 }
```

Menu Antrian Teller:

1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar

Pilih operasi: 5

Jumlah antrian = 3

Tampilkan Antrian



The terminal window shows the 'Tampilkan Antrian' operation. The user has entered the name 'Naufal Thoriq Muzhaffar' and the NIM '2311102078'. The terminal output shows: Masukkan Nama Mahasiswa: Naufal Thoriq Muzhaffar, Masukkan NIM: 2311102078, Data antrian mahasiswa: 1. Nama: Naufal Thoriq Muzhaffar, NIM: 2311102078, 2. Nama: Test 1, NIM: 123, 3. Nama: Test 2, NIM: 456.

```
PS M:\Tugas Naufal\Praktikum\Strukdat\Modul_7> cd "m:\Tugas Naufal\Praktikum\Strukdat\Modul_7"
2 }
```

Menu Antrian Teller:

1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar

Pilih operasi: 3

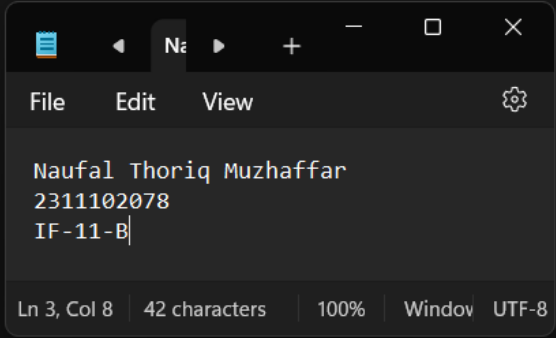
Data antrian mahasiswa:

1. Nama: Naufal Thoriq Muzhaffar, NIM: 2311102078
2. Nama: Test 1, NIM: 123
3. Nama: Test 2, NIM: 456

Hapus Antrian Pertama

```
Pilih operasi: 2
Antrian Berhasil Dihapus

Menu Antrian Teller:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar
Pilih operasi: 3
Data antrian mahasiswa:
1. Nama: Test 1, NIM: 123
2. Nama: Test 2, NIM: 456
```

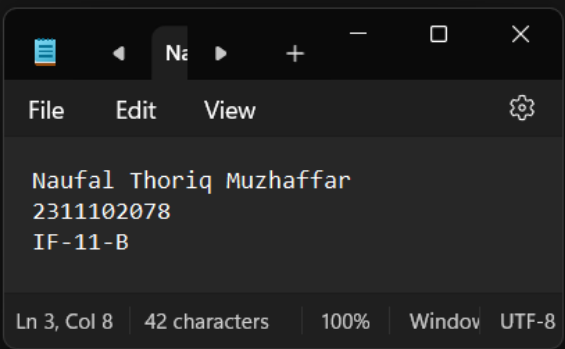


The screenshot shows a text editor window with a dark theme. The title bar reads 'Na'. The menu bar includes 'File', 'Edit', and 'View'. The text content is: 'Naufal Thoriq Muzhaffar', '2311102078', and 'IF-11-B'. The status bar at the bottom indicates 'Ln 3, Col 8', '42 characters', '100%', 'Window', and 'UTF-8'.

Bersihkan Data Antrian

```
Pilih operasi: 4
Antrian Berhasil Dihapus
Antrian Berhasil Dihapus

Menu Antrian Teller:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar
Pilih operasi: 3
Antrian kosong.
```



The screenshot shows a text editor window with a dark theme. The title bar reads 'Na'. The menu bar includes 'File', 'Edit', and 'View'. The text content is: 'Naufal Thoriq Muzhaffar', '2311102078', and 'IF-11-B'. The status bar at the bottom indicates 'Ln 3, Col 8', '42 characters', '100%', 'Window', and 'UTF-8'.

Deskripsi Program

Program ini merupakan implementasi sederhana dari struktur data queue sistem antrian untuk manajemen mahasiswa. Menggunakan struktur data queue, program memungkinkan pengguna untuk menambah, menghapus, menampilkan, membersihkan, dan menghitung jumlah mahasiswa dalam antrian.

D. Kesimpulan

Queue adalah struktur data linear yang mengikuti prinsip FIFO (First In, First Out). Queue dapat diimplementasikan menggunakan array atau linked list. Implementasi dengan array memiliki kelebihan berupa akses elemen yang cepat, namun memiliki kekurangan seperti ukuran yang tetap, sehingga bisa menyebabkan overflow atau pemborosan memori.

Sebaliknya, implementasi dengan linked list menawarkan fleksibilitas ukuran karena memori dialokasikan secara dinamis, meskipun memerlukan memori tambahan untuk pointer dan akses elemen yang lebih lambat. Kedua metode ini memiliki kelebihan dan

kekurangan masing-masing, sehingga pemilihan implementasi bergantung pada kebutuhan spesifik aplikasi serta batasan memori atau performa yang diinginkan.

E. Referensi

- [1] Asprak “Modul 7 Queue”. Learning Management System 2024.
- [2] Geeksforgeeks – Queue Data Structure. Diakses pada 26 Mei 2024
<https://www.geeksforgeeks.org/queue-data-structure/>