

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**MODUL VII**



**DISUSUN OLEH :**

**Nama: Fahrur Rizqi  
Nim : (2311102067)**

**Dosen**

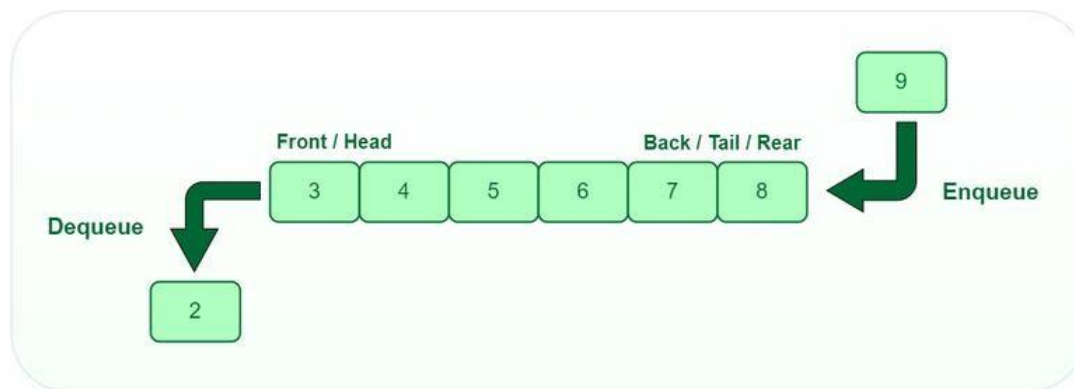
Wahyu Andi Saputra, S.Pd. , M.Eng

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

## A. Dasar Teori

### QUEUE

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

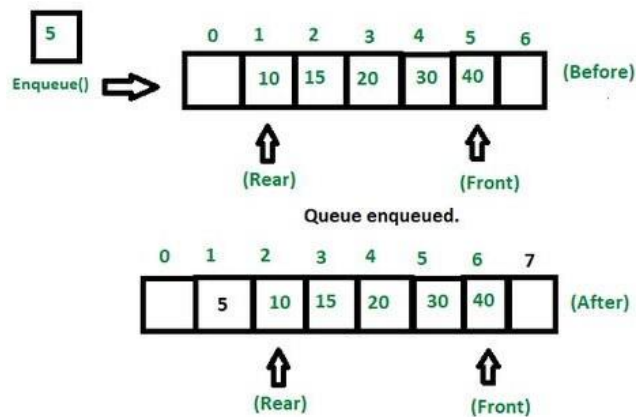
Beberapa operasi dasar Queue dalam Struktur Data adalah:

#### 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;
}
```

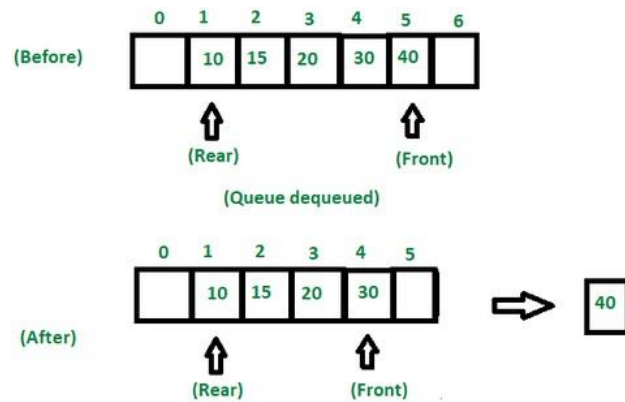
### Operasi 2: dequeue()

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

Sourcode :

```
#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;

```

```

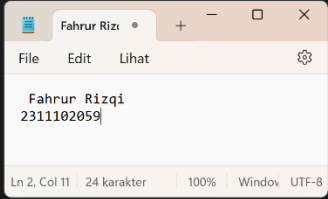
for (int i = 0; i < maksimalQueue; i++)
{
    if (queueTeller[i] != "")
    {
        cout << i + 1 << ". " << queueTeller[i] << endl;
    }
    else
    {
        cout << i + 1 << ". (kosong)" << endl;
    }
}
}

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;
}

```

Output :

```
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 7> cd "c:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 7\" ; if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
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:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 7>
```



Deskripsi :

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

Sourcode

```
#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;

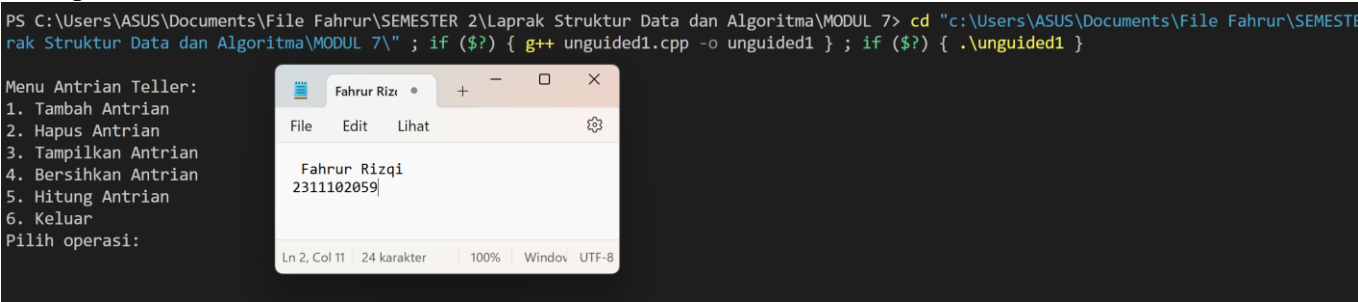
    do {
        displayMenu();
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Masukkan Nama Mahasiswa: ";
                cin.ignore(); // Mengabaikan newline yang tersisa di input buffer
                getline(cin, namaMahasiswa);
                cout << "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;
}
```

Output:



Deskripsi :

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

Sourcode

```
#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;
}

```

Output:

Menu

```

PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 7> cd "c:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 7\" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if ($?) { .\unguided1 }

```

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

Fahrur Rizki  
2311102059

Ln 2, Col 11 | 24 karakter | 100% | Window | UTF-8

## Tambah Antrian

Menu Antrian Mahasiswa:  
1. Tambah Antrian  
2. Hapus Antrian  
3. Tampilkan Antrian  
4. Bersihkan Antrian  
5. Hitung Antrian  
6. Keluar  
Pilih operasi: 1  
Masukkan Nama Mahasiswa: adi  
Masukkan NIM: 231110200

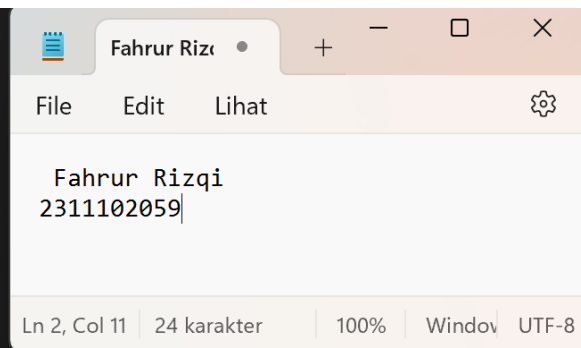


Menu Antrian Mahasiswa:  
1. Tambah Antrian  
2. Hapus Antrian  
3. Tampilkan Antrian  
4. Bersihkan Antrian  
5. Hitung Antrian  
6. Keluar  
Pilih operasi: 1  
Masukkan Nama Mahasiswa: budi  
Masukkan NIM: 231110201

Menu Antrian Mahasiswa:  
1. Tambah Antrian  
2. Hapus Antrian  
3. Tampilkan Antrian  
4. Bersihkan Antrian  
5. Hitung Antrian  
6. Keluar  
Pilih operasi: 1  
Masukkan Nama Mahasiswa: cintia  
Masukkan NIM: 231110202

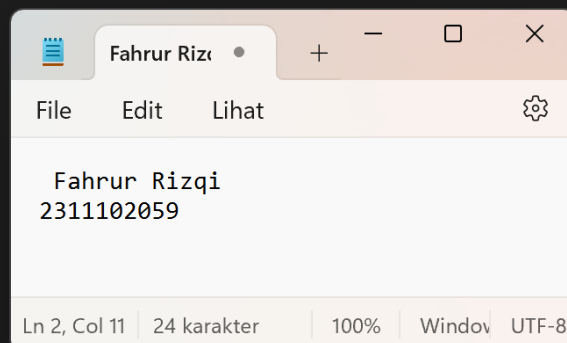
## Hitung Antrian

Menu Antrian Mahasiswa:  
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

Menu Antrian Mahasiswa:  
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: adi, NIM: 231110200  
2. Nama: budi, NIM: 231110201  
3. Nama: cintia, NIM: 231110202



## Hapus Antrian Pertama

Menu Antrian Mahasiswa:

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

Pilih operasi: 2

Antrian Berhasil Dihapus

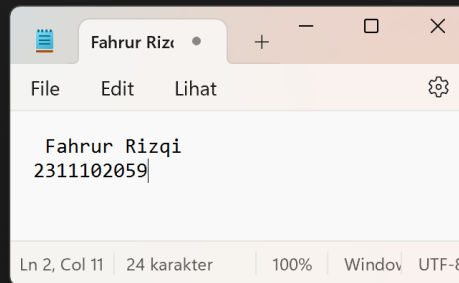
Menu Antrian Mahasiswa:

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: budi, NIM: 231110201
2. Nama: cintia, NIM: 231110202



## Bersihkan Data Antrian

Menu Antrian Mahasiswa:

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

Pilih operasi: 4

Antrian Berhasil Dihapus

Antrian Berhasil Dihapus

Menu Antrian Mahasiswa:

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

Pilih operasi: 3

Antrian kosong.



Keluar

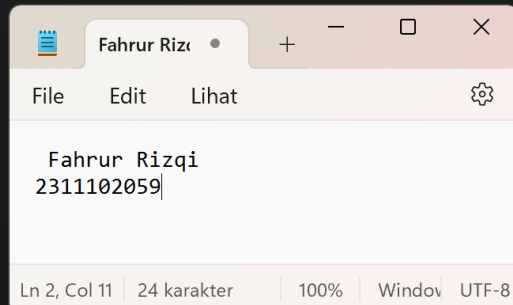
Menu Antrian Mahasiswa:

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

Pilih operasi: 6

Keluar dari program.

PS C:\Users\ASUS\Documents\File Fajar (SEMESTER 2 (Lengkap Struktur Data dan Algoritma)\MODUL 7> [



Deskripsi :

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.

## 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.

## **Referensi**

Asisten Pratikum “Modul 7 Queue”, Learning Management System, 2024.

geeksforgeeks. (14 mei 2024). Queue Data Structure.

<https://www.geeksforgeeks.org/queue-data-structure/#applications-of-queue>