

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**MODUL VII  
QUEUE**



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

**Disusun oleh:**

**FALAH ASBI PANGESTU**

**(2311102045)**

**IF-11-B**

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

**2024**

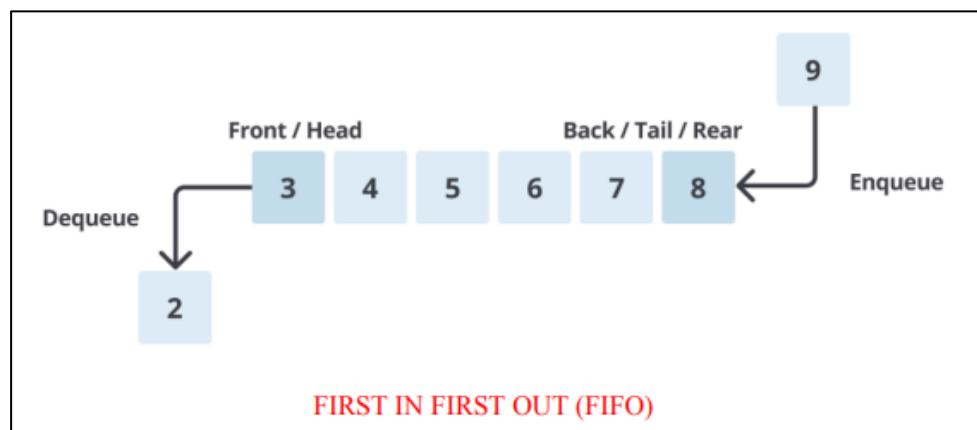
# BAB I

## DASAR TEORI

### A. Dasar Teori

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode FIFO (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep antrian pada kehidupan sehari-hari, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu.

Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. Front/head adalah pointer ke elemen pertama dalam queue dan rear/tail/back adalah pointer ke elemen terakhir dalam queue.



Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir diinputkan akan berada paling dengan dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO.

Pada Queue, operasi tersebut dilakukan ditempat berbeda (melalui salah satu ujung) karena perubahan data selalu mengacu pada Head, maka hanya ada 1 jenis insert maupun delete. Prosedur ini sering disebut Enqueue dan Dequeue pada kasus Queue. Untuk Enqueue, cukup tambahkan elemen setelah elemen terakhir Queue, dan untuk Dequeue, cukup "geser"kan Head menjadi elemen selanjutnya.

### Operasi pada Queue

- enqueue() : menambahkan data ke dalam queue.
- dequeue() : mengeluarkan data dari queue.
- peek() : mengambil data dari queue tanpa menghapusnya.
- isEmpty() : mengecek apakah queue kosong atau tidak.
- isFull() : mengecek apakah queue penuh atau tidak.
- size() : menghitung jumlah elemen dalam queue.

## BAB II

### GUIDED

#### LATIHAN – GUIDED

##### 1. Guided 1

Guided (berisi screenshot source code & output program disertai penjelasannya)

##### Source Code

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

const int maksimalQueue = 5; // Maksimal antrian
int front = 0; // Penanda antrian
int back = 0; // Penanda belakang
string queueTeller[maksimalQueue]; // Array untuk menyimpan data antrian

// Fungsi pengecekan antrian penuh atau tidak
bool isFull() {
    return back == maksimalQueue;
}

// Fungsi pengecekan antriannya kosong atau tidak
bool isEmpty() {
    return back == 0;
}

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

// Fungsi mengurangi antrian
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        for (int i = 0; i < back - 1; i++) {
            queueTeller[i] = queueTeller[i + 1];
        }
        queueTeller[back - 1] = ""; // Clear the last element
    }
}
```

```

        back--;
    }
}

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

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

// Fungsi melihat 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;

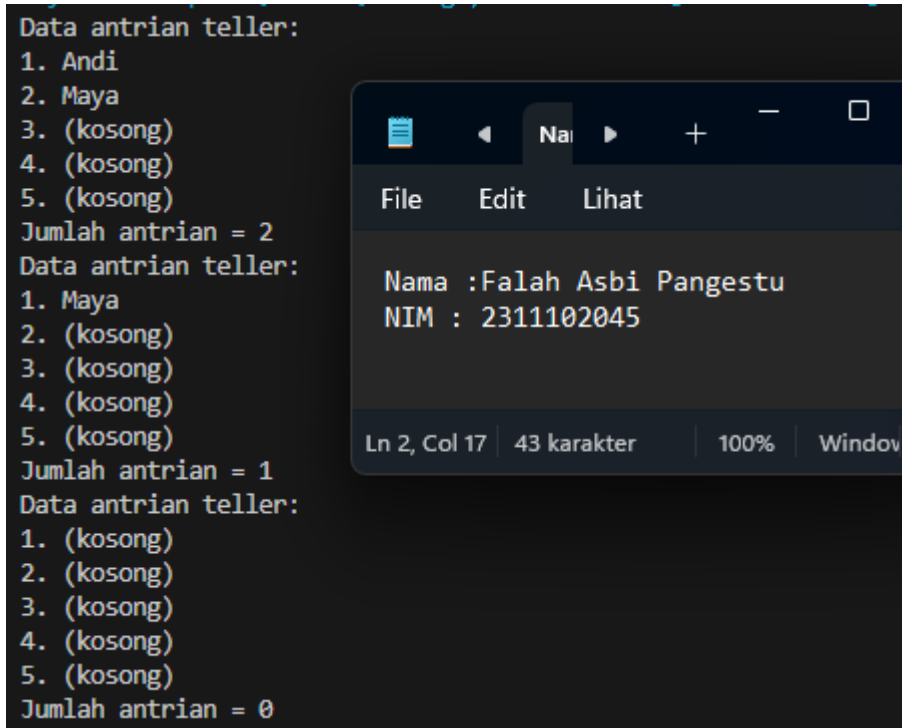
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

```

```
    return 0;
}
```

### Screenshoot program



```
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
```

Context Menu:

- File
- Edit
- Lihat
- Nama :Falah Asbi Pangestu
- NIM : 2311102045
- Ln 2, Col 17 | 43 karakter | 100% | Window

### Deskripsi program

Program menyediakan beberapa fungsi untuk operasi dasar pada antrian, seperti `isFull()` untuk mengecek apakah antrian sudah penuh, `isEmpty()` untuk mengecek apakah antrian masih kosong, `enqueueAntrian()` untuk menambahkan elemen baru ke dalam antrian, `dequeueAntrian()` untuk menghapus elemen terdepan dari antrian, `countQueue()` untuk menghitung jumlah elemen dalam antrian, `clearQueue()` untuk mengosongkan seluruh elemen dalam antrian, dan `viewQueue()` untuk menampilkan seluruh elemen yang ada dalam antrian.

Pada fungsi `main()`, program melakukan demonstrasi penggunaan antrian. Pertama, program menambahkan dua elemen ("Andi" dan "Maya") ke dalam antrian menggunakan `enqueueAntrian()`. Kemudian, program menampilkan isi antrian dengan `viewQueue()` dan menghitung jumlah elemen dalam antrian dengan `countQueue()`. Selanjutnya, program menghapus elemen terdepan dari antrian menggunakan `dequeueAntrian()`, menampilkan isi antrian setelah penghapusan, dan menghitung jumlah elemen dalam antrian setelah penghapusan. Terakhir, program mengosongkan seluruh elemen dalam antrian menggunakan `clearQueue()`, menampilkan isi antrian

setelah dikosongkan, dan menghitung jumlah elemen dalam antrian setelah dikosongkan.

## BAB III

### UNGUIDED

#### TUGAS – UNGUIDED

##### 1. Unguided 1

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

##### Source Code

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

// Struktur node untuk linked list
struct Node {
    string data;
    Node* next;
};

// Penanda antrian
Node* front = nullptr;
Node* back = nullptr;

// Fungsi pengecekan antrian penuh atau tidak (tidak relevan untuk
// linked list karena tidak ada batasan kapasitas)
bool isFull() {
    return false;
}

// Fungsi pengecekan antriannya kosong atau tidak
bool isEmpty() {
    return front == nullptr;
}

// Fungsi menambahkan antrian
void enqueueAntrian(string data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
}

// Fungsi mengurangi antrian
void dequeueAntrian() {
```



```

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

// Fungsi menghitung banyak antrian
int countQueue() {
    int count = 0;
    Node* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

// Fungsi menghapus semua antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi melihat antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int i = 1;
    while (temp != nullptr) {
        cout << i << ". " << temp->data << endl;
        temp = temp->next;
        i++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

int main() {
    enqueueAntrian("PARIS");
    enqueueAntrian("Yusuf");
}

```

```

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

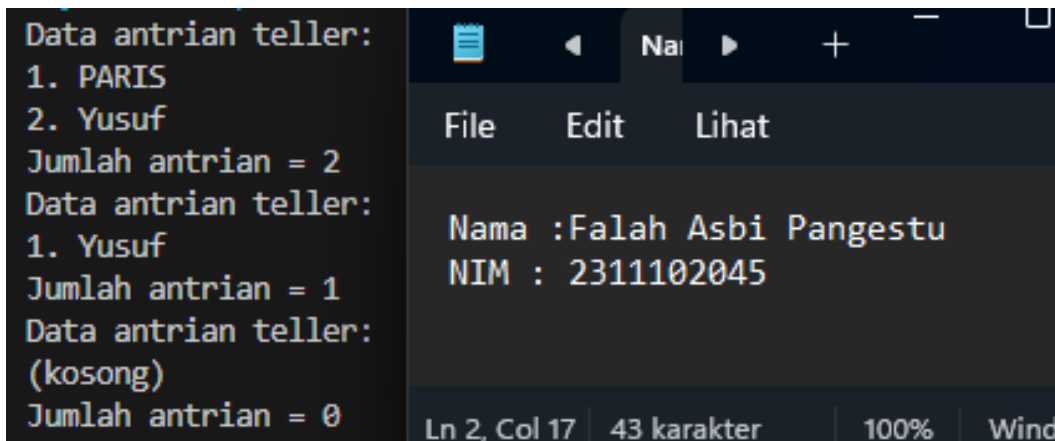
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

    return 0;
}

```

### Screenshot Program



### Deskripsi program

Setiap data dalam antrian disimpan dalam sebuah node yang terhubung satu sama lain membentuk Linked List. Terdapat dua penanda khusus, yaitu `front` yang menunjuk ke node depan antrian (akan dikeluarkan pertama) dan `back` yang menunjuk ke node terakhir antrian (tempat data baru dimasukkan).

Program menyediakan sejumlah fungsi untuk melakukan operasi-operasi pada antrian. Fungsi `enqueueAntrian(data)` digunakan untuk menambahkan data baru ke antrian dengan memasukkannya pada node terakhir. Fungsi `dequeueAntrian()` digunakan untuk mengeluarkan data dari antrian dengan mengambil data pada node depan. Fungsi `isEmpty()` memeriksa apakah antrian dalam keadaan kosong, sedangkan `countQueue()` menghitung jumlah data dalam antrian.

Selain itu, terdapat fungsi `clearQueue()` untuk menghapus semua data dalam antrian dan `viewQueue()` untuk menampilkan seluruh data dalam antrian secara berurutan. Program juga menyediakan fungsi `isFull()` yang selalu mengembalikan nilai false karena pada Linked List, antrian tidak akan pernah penuh.

## 2. Guided 2

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

### Source code

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

// Struktur node untuk linked list dengan Nama dan NIM
struct Node {
    string nama;
    string NIM;
    Node* next;
};

// Penanda antrian
Node* front = nullptr;
Node* back = nullptr;

// Fungsi pengecekan antrian kosong atau tidak
bool isEmpty() {
    return front == nullptr;
}

// Fungsi menambahkan antrian
void enqueueAntrian(string nama, string NIM) {
    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->NIM = NIM;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
}

// Fungsi mengurangi antrian
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    }
}
```

```

    } else {
        Node* temp = front;
        front = front->next;
        if (front == nullptr) {
            back = nullptr;
        }
        delete temp;
    }
}

// Fungsi menghitung banyak antrian
int countQueue() {
    int count = 0;
    Node* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

// Fungsi menghapus semua antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi melihat antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int i = 1;
    while (temp != nullptr) {
        cout << i << ". Nama: " << temp->nama << ", NIM: " << temp->NIM << endl;
        temp = temp->next;
        i++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

int main() {
    enqueueAntrian("Andi", "123456");
    enqueueAntrian("Maya", "654321");
    viewQueue();
}

```

```

    cout << "Jumlah antrian = " << countQueue() << endl;

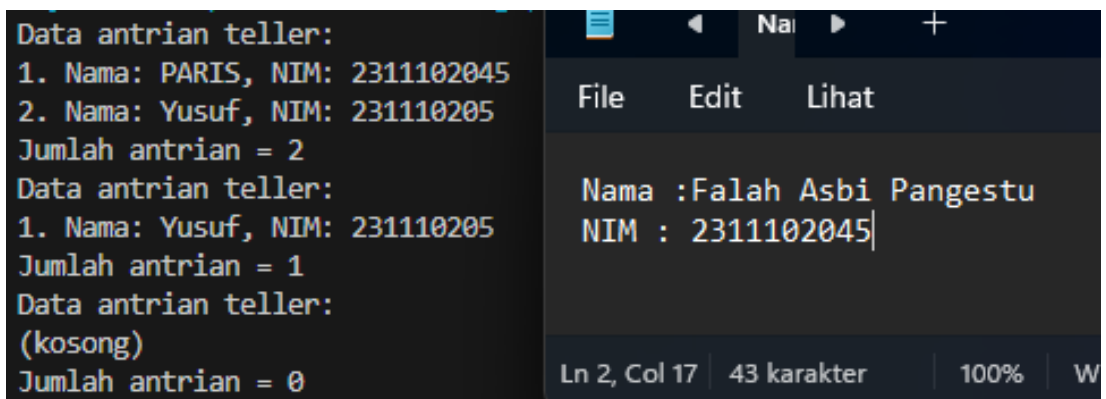
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

    return 0;
}

```

### Screenshoot program



### Deskripsi program

Program ini merupakan implementasi struktur data antrian (queue) menggunakan linked list. Antrian digunakan untuk menyimpan data mahasiswa berupa nama dan NIM. Program mendefinisikan sebuah struktur Node yang menyimpan data nama, NIM, dan pointer ke node berikutnya. Terdapat dua pointer global, yaitu front dan back, yang menunjuk ke node awal dan akhir antrian.

Dalam fungsi main(), program mendemonstrasikan penggunaan fungsi-fungsi tersebut. Pertama, program menambahkan dua data mahasiswa ke dalam antrian menggunakan enqueueAntrian(). Kemudian, program menampilkan isi antrian menggunakan viewQueue() dan menghitung jumlah data dalam antrian menggunakan countQueue(). Selanjutnya, program menghapus satu data dari antrian menggunakan dequeueAntrian(), menampilkan isi antrian yang baru, dan menghitung jumlah data yang tersisa. Terakhir, program menghapus semua data dalam antrian menggunakan clearQueue(), menampilkan isi antrian yang kosong, dan menghitung jumlah data yang tersisa.

## **BAB IV**

### **KESIMPULAN**

Queue merupakan sebuah struktur data linear yang mengikuti prinsip FIFO (First In, First Out), di mana elemen yang pertama kali masuk adalah yang pertama kali keluar. Operasi-operasi dasar pada queue meliputi enqueue (menambahkan elemen di akhir), dequeue (menghapus elemen dari awal), isEmpty (memeriksa apakah queue kosong), isFull (untuk array, memeriksa apakah queue penuh), count (menghitung jumlah elemen), clear (menghapus semua elemen), dan view (menampilkan semua elemen).

Queue dapat diimplementasikan menggunakan array atau linked list. Implementasi dengan array memiliki kapasitas tetap dan memerlukan pengelolaan posisi elemen, sedangkan linked list memungkinkan penambahan dan penghapusan elemen secara dinamis tanpa batasan kapasitas. Setiap metode memiliki kelebihan dan kekurangannya sendiri: array sederhana dan cepat dalam akses elemen tetapi terbatas dalam kapasitas dan memerlukan pergeseran elemen saat dequeuing; sedangkan linked list fleksibel dalam hal kapasitas dan tidak memerlukan pergeseran elemen, namun membutuhkan lebih banyak memori dan sedikit lebih kompleks.

## **DAFTAR PUSTAKA**

- [1] Asisten Praktikum. Learning Management System, Modul 7 Queue. 2024