

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 7
QUEUE**



DISUSUN OLEH:

NAMA : SYARIEF RENDI ADITYA ANTONIUS

NIM : 2311102072

S1 IF-11-B

DOSEN:

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

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

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

B. Guided

GUIDED :

SOURCE CODE

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

const int maksimalQueue = 5; //Batas maksimal antrian
int front = 0;                //Indeks antrian awal
int back = 0;                 //Indeks antrian akhir
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 - 1; 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;

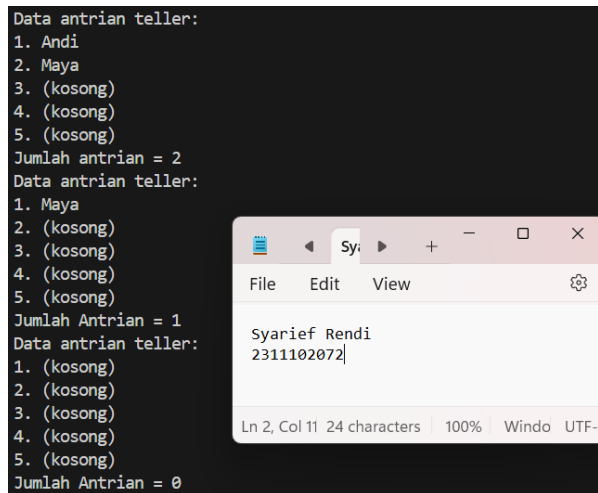
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah Antrian = " << countQueue() << endl;

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

    return 0;
} //2311102072

```

SCREENSHOT OUTPUT



The screenshot shows a terminal window on the left and a text editor window on the right. The terminal window displays the output of a queue simulation. It shows a list of people in a queue, with some positions being empty. The queue is initially populated with 'Andi' and 'Maya', and then 'Andi' is removed, leaving 'Maya' as the only person in the queue. The text editor window shows a file named 'Syarif Rendy' with the text '2311102072'.

```
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
```

DESKRIPSI PROGRAM

Program di atas menggunakan sebuah array `queueTeller` untuk merepresentasikan antrian queue yang menyimpan data berupa nama-nama dalam antrian teller. Program ini memiliki beberapa fungsi untuk mengelola antrian, termasuk menambahkan data ke antrian `enqueue`, menghapus data dari antrian `dequeue`, menghitung jumlah elemen dalam antrian `countQueue`, mengosongkan antrian `clearQueue`, dan menampilkan seluruh isi antrian `viewQueue`. Detail fungsi-fungsi dalam program: `isFull`: Memeriksa apakah antrian sudah penuh. `isEmpty`: Memeriksa apakah antrian kosong. `enqueueAntrian`: Menambahkan elemen ke akhir antrian jika antrian belum penuh. `dequeueAntrian`: Menghapus elemen dari awal antrian jika antrian tidak kosong, dan menggeser elemen-elemen berikutnya ke depan. `countQueue`: Menghitung dan mengembalikan jumlah elemen dalam antrian. `clearQueue`: Mengosongkan semua elemen dalam antrian. `viewQueue`: Menampilkan semua elemen dalam antrian beserta posisinya. Dalam fungsi main, program menambahkan beberapa nama ke dalam antrian, menampilkan isi antrian, menghapus elemen dari antrian, dan mengosongkan antrian, sembari menampilkan perubahan jumlah elemen di setiap langkah.

C. UNGUIDED

UNGUIDED 1 :

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

SOURCE CODE

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

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

Node* front = nullptr;
Node* back = nullptr;

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

// Fungsi untuk menambahkan elemen ke 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;
    }
    cout << "Data " << data << " berhasil ditambahkan ke antrian." << endl;
}

// Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian() {
    if (isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
```

```

        if (front == nullptr) {
            back = nullptr;
        }
        cout << "Data " << temp->data << " berhasil dihapus
dari antrian." << endl;
        delete temp;
    }
}

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

// Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
    cout << "Antrian berhasil dikosongkan." << endl;
}

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

void showMenu() {
    cout << "=== Menu Antrian ===" << endl;
    cout << "1. Tambah Antrian" << endl;
    cout << "2. Hapus Antrian" << endl;
}

```



```

        cout << "3. Lihat Antrian" << endl;
        cout << "4. Hitung Jumlah Antrian" << endl;
        cout << "5. Kosongkan Antrian" << endl;
        cout << "6. Keluar" << endl;
        cout << "Pilih opsi: ";
    }

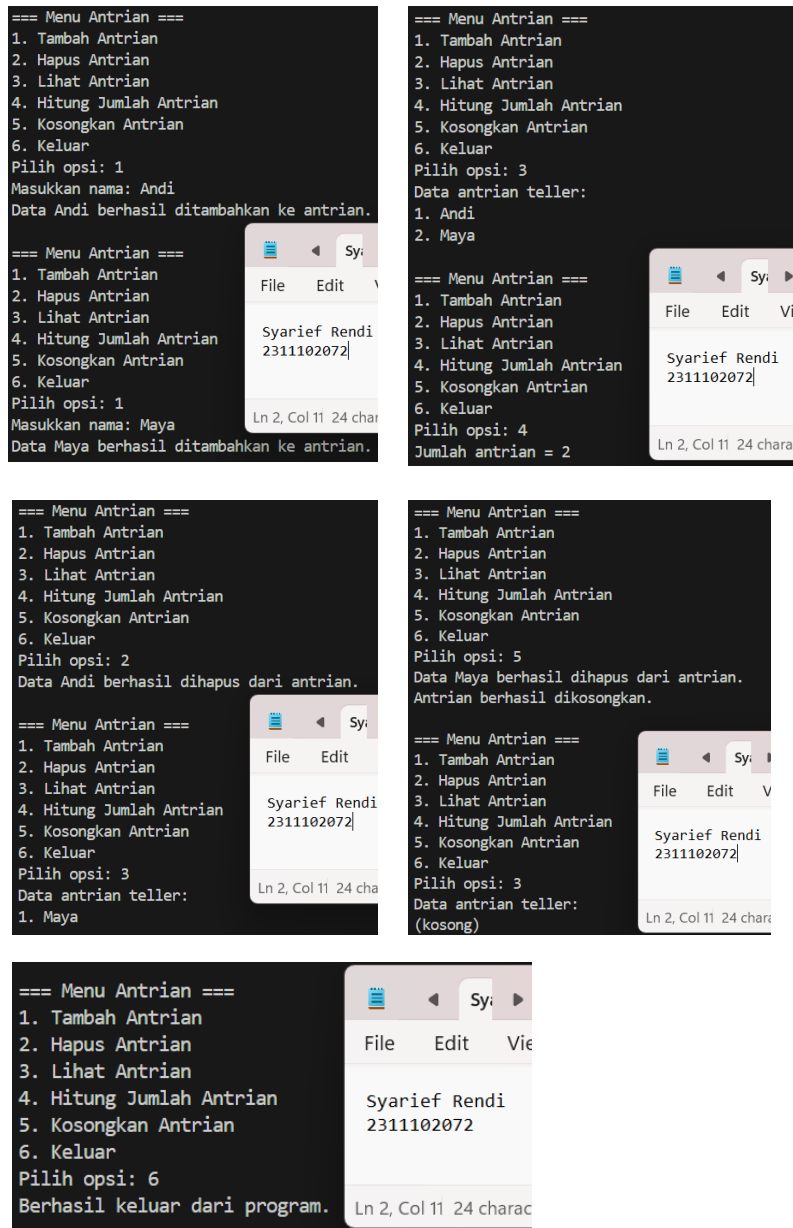
int main() {
    int choice;
    string data;

    do {
        showMenu();
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Masukkan nama: ";
                cin >> data;
                enqueueAntrian(data);
                break;
            case 2:
                dequeueAntrian();
                break;
            case 3:
                viewQueue();
                break;
            case 4:
                cout << "Jumlah antrian = " << countQueue() <<
endl;
                break;
            case 5:
                clearQueue();
                break;
            case 6:
                cout << "Berhasil keluar dari program." <<
endl;
                break;
            default:
                cout << "Opsi salah. Silahkan input menu secara
benar." << endl;
        }
        cout << endl;
    } while (choice != 6);

    return 0;
} //2311102072

```

SCREENSHOT OUTPUT



DESKRIPSI PROGRAM

Program ini menggunakan struktur data linked list untuk merepresentasikan antrian queue yang menyimpan nama-nama dalam antrian teller. Fungsi-fungsi utama dalam program meliputi: isEmpty untuk memeriksa apakah antrian kosong, enqueueAntrian untuk menambahkan elemen ke akhir antrian, dequeueAntrian untuk menghapus elemen dari awal antrian, countQueue untuk menghitung jumlah elemen dalam antrian, clearQueue untuk mengosongkan antrian, dan viewQueue untuk menampilkan seluruh isi antrian. Program ini juga menyediakan menu

interaktif yang memungkinkan pengguna untuk menambahkan nama ke antrian, menghapus nama dari antrian, melihat isi antrian, menghitung jumlah elemen, mengosongkan antrian, atau keluar dari program, sehingga memudahkan pengguna dalam mengelola antrian.

UNGUIDED 2 :

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

SOURCE CODE

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

struct Node {
    string name;
    string nim;
    Node* next;
};

Node* front = nullptr;
Node* back = nullptr;

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

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string name, string nim) {
    if (nim.length() != 10) {
        cout << "NIM harus 10 digit." << endl;
        return;
    }
    Node* newNode = new Node();
    newNode->name = name;
    newNode->nim = nim;
    newNode->next = nullptr;
    if (isEmpty()) {
        front = back = newNode;
    } else {
        back->next = newNode;
        back = newNode;
    }
    cout << "Data " << name << " dengan NIM " << nim << "
    berhasil ditambahkan ke antrian." << endl;
}

// Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian() {
    if (isEmpty()) {
```

```

        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = front;
        front = front->next;
        if (front == nullptr) {
            back = nullptr;
        }
        cout << "Data " << temp->name << " dengan NIM " <<
temp->nim << " berhasil dihapus dari antrian." << endl;
        delete temp;
    }
}

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

// Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
    cout << "Antrian berhasil dikosongkan." << endl;
}

// Fungsi untuk menampilkan semua elemen dalam antrian
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int index = 1;
    while (temp != nullptr) {
        cout << index << ". Nama: " << temp->name << ", NIM: "
<< temp->nim << endl;
        temp = temp->next;
        index++;
    }
    if (isEmpty()) {
        cout << "(kosong)" << endl;
    }
}

```

```

void showMenu() {
    cout << "=== Menu Antrian ===" << endl;
    cout << "1. Tambah Antrian" << endl;
    cout << "2. Hapus Antrian" << endl;
    cout << "3. Lihat Antrian" << endl;
    cout << "4. Hitung Jumlah Antrian" << endl;
    cout << "5. Kosongkan Antrian" << endl;
    cout << "6. Keluar" << endl;
    cout << "Pilih opsi: ";
}

int main() {
    int choice;
    string name, nim;

    do {
        showMenu();
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Masukkan nama: ";
                cin.ignore();
                getline(cin, name);
                cout << "Masukkan NIM (10 digit): ";
                cin >> nim;
                enqueueAntrian(name, nim);
                break;
            case 2:
                dequeueAntrian();
                break;
            case 3:
                viewQueue();
                break;
            case 4:
                cout << "Jumlah antrian = " << countQueue() <<
endl;
                break;
            case 5:
                clearQueue();
                break;
            case 6:
                cout << "Berhasil keluar dari program." <<
endl;
                break;
            default:

```

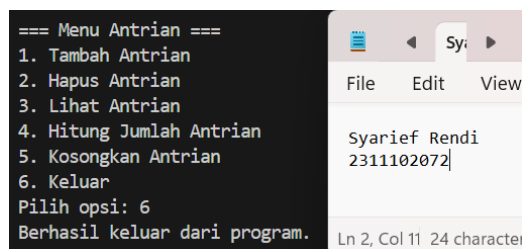
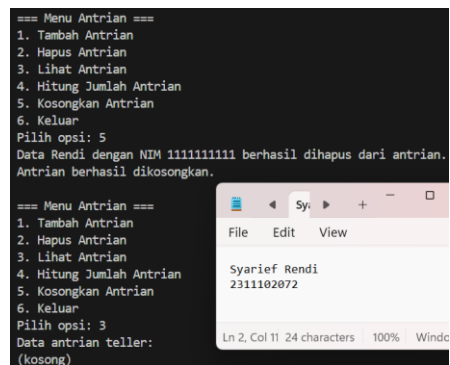
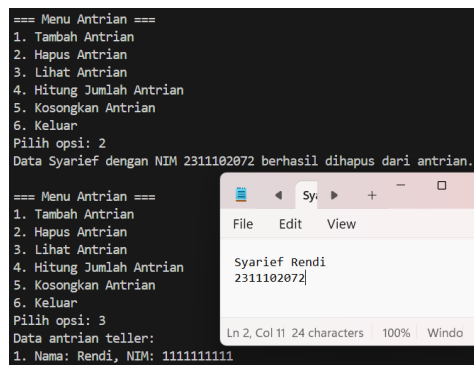
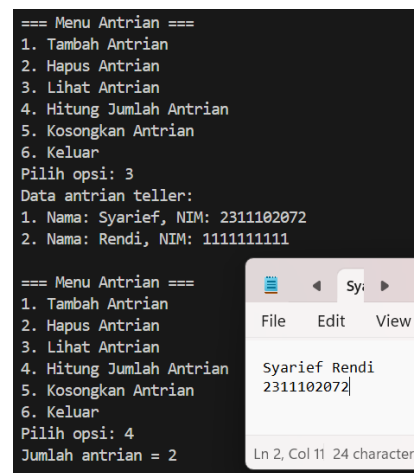
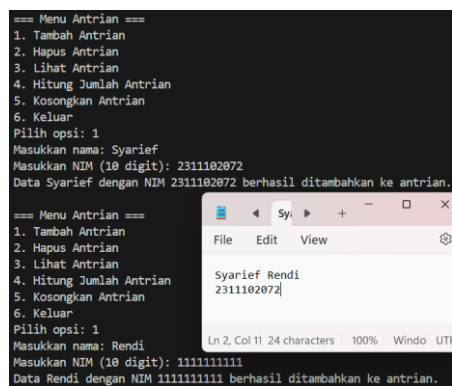
```

        cout << "Opsi salah. Silahkan input menu secara
benar." << endl;
    }
    cout << endl;
} while (choice != 6);

return 0;
} //2311102072

```

SCREENSHOT OUTPUT



DESKRIPSI PROGRAM

Program di atas merupakan implementasi antrian menggunakan linked list untuk menyimpan data mahasiswa yang mencakup nama dan NIM (10 digit). Modifikasi utama meliputi penambahan atribut name dan nim pada struktur Node, serta pembaruan fungsi enqueueAntrian untuk menerima dan memvalidasi input ini. Fungsi viewQueue diubah untuk menampilkan nama dan NIM setiap mahasiswa dalam antrian. Fungsi lain seperti dequeueAntrian, countQueue, dan clearQueue disesuaikan dengan struktur Node yang baru. Program ini menyediakan menu interaktif yang memungkinkan pengguna menambah, menghapus, melihat, menghitung, dan mengosongkan antrian mahasiswa.

D. Kesimpulan

Queue adalah struktur data yang menggunakan prinsip First In, First Out (FIFO), di mana elemen pertama yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Queue diimplementasikan menggunakan array atau linked list dan terdiri dari dua pointer: front untuk elemen pertama dan rear untuk elemen terakhir. Operasi dasar pada queue meliputi enqueue untuk menambah elemen ke belakang queue, dequeue untuk menghapus elemen dari depan queue, peek untuk melihat elemen depan tanpa menghapusnya, isEmpty untuk memeriksa apakah queue kosong, isFull untuk memeriksa apakah queue penuh, dan size untuk menghitung jumlah elemen dalam queue. Queue mirip dengan antrian dalam kehidupan sehari-hari, di mana pelanggan yang datang lebih dulu akan dilayani lebih dulu.

E. Referensi

- [1] Asisten Praktikum, "Modul Queue", 2024.
- [2] Rizki Maulana. 2023. "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/>
- [3] Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.
- [4] Budi Raharjo. 2015. Pemrograman C++. Bandung: Informatika.