LAPORAN PRAKTIKUM

MODUL VII

QEUE

Disusun oleh:

Reli Gita Nurhidayati

NIM: 2311102025

Dosen Pengampu:

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

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO

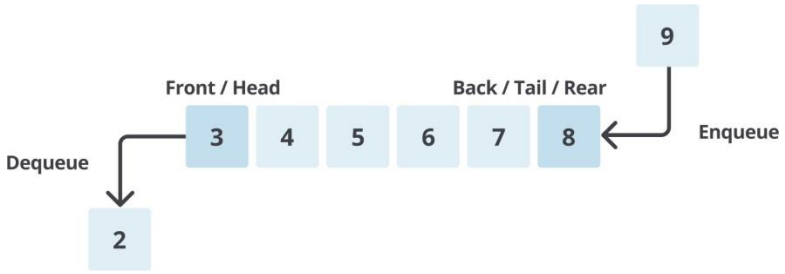
2024

BAB II

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.



FIRST IN FIRST OUT (FIFO)

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 III

GUIDED

Guided 1

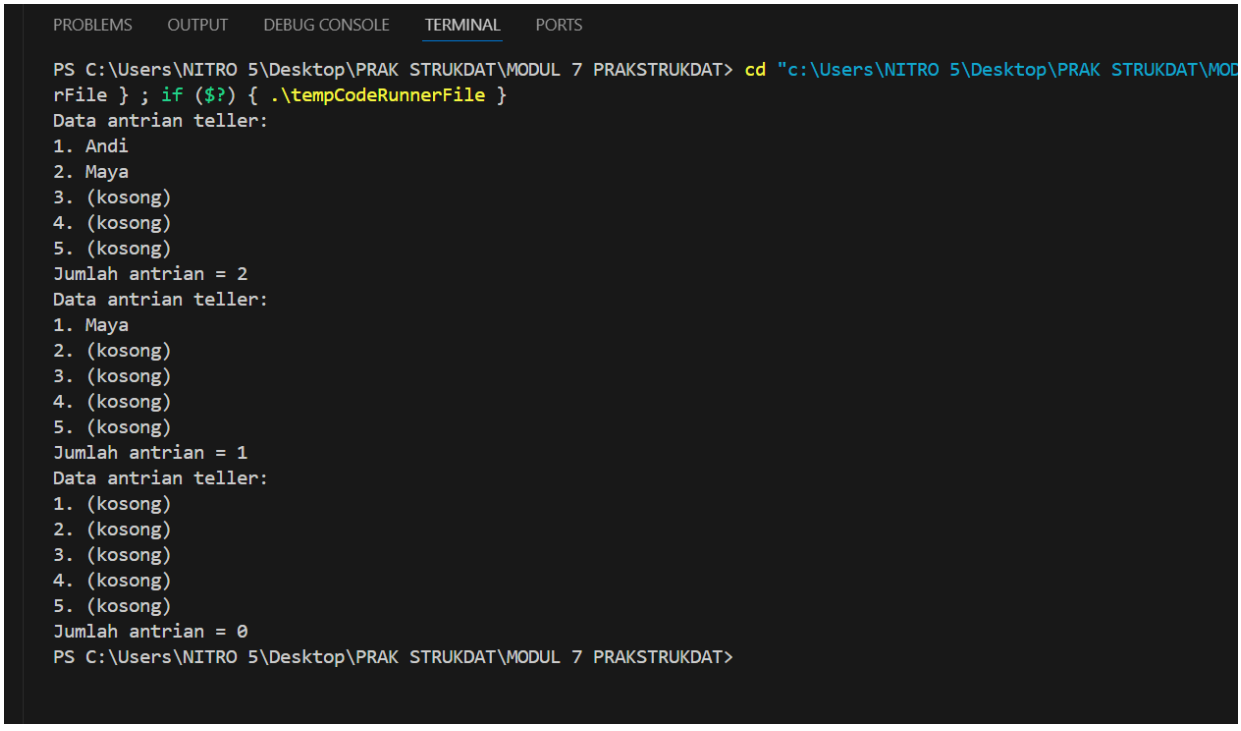
Source Code

```
#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;
}
```

Screenshoot Program



Deskripsi Program

Program ini mengimplementasikan struktur data antrian (queue) menggunakan linked list dalam bahasa pemrograman C++. Antrian direpresentasikan dengan kelas Queue yang memiliki operasi enqueue (menambahkan elemen ke antrian), dequeue (menghapus elemen dari antrian), countQueue (menghitung jumlah elemen dalam antrian), clearQueue (mengosongkan seluruh antrian), dan viewQueue (menampilkan elemen-elemen dalam antrian).

Sebuah simpul (Node) digunakan untuk menyimpan data dan pointer next untuk menunjuk ke simpul berikutnya. Antrian direpresentasikan dengan front (ujung depan antrian) dan back (ujung belakang antrian).

Pada fungsi main, program membuat objek antrian, menambahkan dua elemen ke antrian (Andi dan Maya), menampilkan isi antrian, menghitung jumlah elemen dalam antrian, menghapus elemen dari antrian, menampilkan isi antrian kembali, menghitung jumlah elemen dalam antrian, mengosongkan antrian, menampilkan antrian setelah dikosongkan, dan terakhir menghitung jumlah elemen dalam antrian.

Program ini memberikan contoh penggunaan antrian (queue) menggunakan linked list dalam bahasa pemrograman C++, di mana operasi-operasi dasar seperti enqueue, dequeue, menghitung jumlah elemen, mengosongkan antrian, dan menampilkan isi antrian diimplementasikan dan didemonstrasikan.

BAB IV

UNGUIDED

Unguided 1

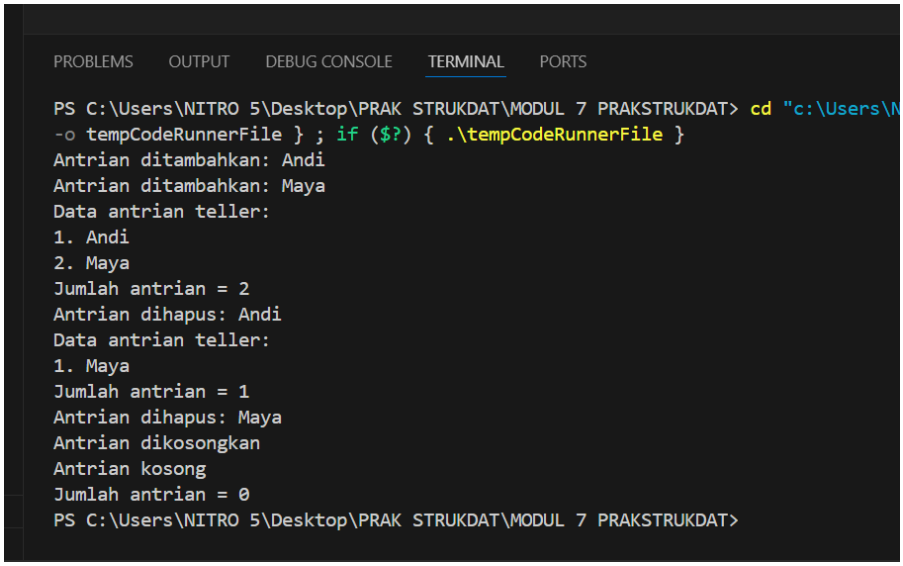
Source Code

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

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

class Queue {
private:
    Node* front;
    Node* back;
public:
    Queue() {
        front = nullptr;
        back = nullptr;
    }

    void enqueueAntrian(string data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = nullptr;
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
        cout << "Antrian ditambahkan: " << data << endl;
    }

    void dequeueAntrian() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        Node* temp = front;
        cout << "Antrian dihapus: " << front->data << endl;
        front = front->next;
```

```
        delete temp;
    }

    int countQueue() {
        int count = 0;
        Node* current = front;
        while (current != nullptr) {
            count++;
            current = current->next;
        }
        return count;
    }

    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
        cout << "Antrian dikosongkan" << endl;
    }

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        cout << "Data antrian teller:" << endl;
        Node* current = front;
        int position = 1;
        while (current != nullptr) {
            cout << position << ". " << current->data << endl;
            current = current->next;
            position++;
        }
    }

    bool isEmpty() {
        return front == nullptr;
    }
};

int main() {
    Queue antrian;
    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;
}
```

Screenshoot Program



Deskripsi Program

Program C++ di atas merupakan implementasi dari struktur data antrian (queue) menggunakan linked list. Dalam program ini, terdapat sebuah struktur Node yang menyimpan data dan pointer ke node selanjutnya. Kelas Queue memiliki metode untuk enqueue (menambahkan data ke belakang antrian), dequeue (menghapus data dari depan antrian), menghitung jumlah antrian, membersihkan antrian, serta melihat isi antrian. Program ini juga dilengkapi dengan fungsi main yang melakukan sejumlah operasi enqueue, dequeue, dan tampilan antrian dengan menampilkan jumlah antrian setiap kali operasi dilakukan.

Unguided 2

Source Code

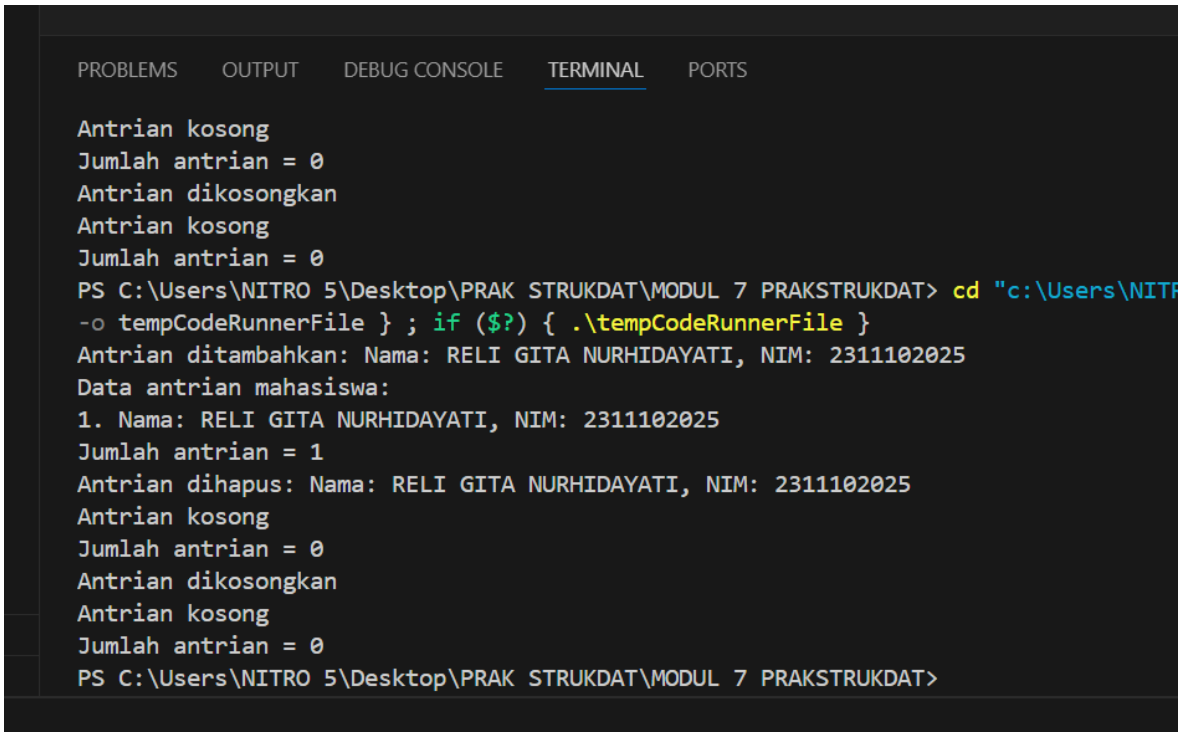
```
#include <iostream>
using namespace std;

// Struktur untuk merepresentasikan setiap elemen dalam linked list
struct Node {
```

```
    string nama;
    string nim;
    Node* next;
};

class Queue {
private:
    Node* front; // Pointer ke depan antrian
    Node* back;  // Pointer ke belakang antrian
public:
    Queue() {
        front = nullptr;
        back = nullptr;
    }

    // Fungsi menambahkan elemen ke belakang antrian (enqueue)
    void enqueueAntrian(string nama, string nim) {
        Node* newNode = new Node();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
        cout << "Antrian ditambahkan: Nama: " << nama << ", NIM: " << nim <<
endl;
    }

    // Fungsi menghapus elemen dari depan antrian (dequeue)
    void dequeueAntrian() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        Node* temp = front;
        cout << "Antrian dihapus: Nama: " << front->nama << ", NIM: " <<
front->nim << endl;
        front = front->next;
        delete temp;
    }

    // Fungsi menghitung banyak elemen dalam antrian
    int countQueue() {
        int count = 0;
```

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

    // Fungsi menghapus semua elemen dari antrian
    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
        cout << "Antrian dikosongkan" << endl;
    }

    // Fungsi untuk melihat antrian
    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
            return;
        }
        cout << "Data antrian mahasiswa:" << endl;
        Node* current = front;
        int position = 1;
        while (current != nullptr) {
            cout << position << ". Nama: " << current->nama << ", NIM: " <<
current->nim << endl;
            current = current->next;
            position++;
        }
    }

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

int main() {
    Queue antrian;
    antrian.enqueueAntrian("RELI GITA NURHIDAYATI", "2311102025");
    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;
}
```

Screenshoot Program



Deskripsi Program

Program C++ ini mengimplementasikan struktur data antrian (queue) menggunakan linked list untuk mengelola data mahasiswa dengan nama dan NIM. Terdapat kelas Queue dengan metode-metode:

1. enqueue() - Menambahkan data mahasiswa ke belakang antrian sedangkan dequeue() untuk Menghapus data mahasiswa dari depan antrian
2. Menghitung jumlah data dalam antrian
3. Membersihkan/mengosongkan antrian
4. Menampilkan data antrian

Di fungsi main:

1. Data mahasiswa ditambahkan ke antrian dengan enqueue()
2. Menampilkan isi antrian dan jumlah data
3. Menghapus data dari depan antrian dengan dequeue()
4. Menampilkan antrian yang diubah dan jumlah data tersisa
5. Membersihkan seluruh antrian
6. Menampilkan antrian kosong dan jumlah data (0)

Jadi program ini mendemonstrasikan operasi dasar antrian seperti enqueue, dequeue, serta menampilkan dan membersihkan isi antrian dalam konteks mengelola data mahasiswa.

BAB V

KESIMPULAN

Queue (antrian) merupakan struktur data yang mengikuti prinsip First In, First Out (FIFO), di mana elemen yang pertama masuk akan menjadi elemen pertama yang keluar. Struktur ini berguna untuk memodelkan situasi di kehidupan nyata seperti antrean kasir, antrean printer, atau antrean pesan yang menunggu untuk diproses.

Manfaat mempelajari queue:

1. Memecahkan masalah FIFO, seperti urutan pemrosesan berdasarkan siapa yang datang pertama.
2. Mengoptimasi proses dengan mengatur urutan eksekusi tugas atau pengelolaan buffer data.
3. Penting untuk menulis program yang dapat menangani antrian data secara efisien.

Konsep penting terkait queue:

1. Operasi dasar: enqueue (menambahkan elemen baru ke belakang antrian) dan dequeue (menghapus dan mengembalikan elemen dari depan antrian).
2. Jenis queue: simple queue (FIFO standar) dan priority queue (elemen diprioritaskan berdasarkan nilai tertentu).
3. Implementasi: queue dapat diimplementasikan menggunakan array atau linked list, tergantung pada kebutuhan dan efisiensi yang diinginkan.

Kesimpulan: Mempelajari queue dalam struktur data memberikan konsep penting untuk menangani urutan pemrosesan data. Penguasaan konsep ini bermanfaat untuk menyelesaikan berbagai permasalahan komputasi dan menulis program yang efektif.

DAFTAR PUSTAKA

[1]A Zein, Eriana ES.2022. Algoritma Dan Struktur Data. Diakses 26 Mei, dari https://repository.unpam.ac.id/10199/1/KB1101_ALGORITMA%20%20DAN%20STRUKTUR%20DATA.pdf