LAPORAN PRAKTIKUM

MODUL VII
QUEUE

Disusun oleh:
Aryo Tegar Sukarno
NIM: 2311102018

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

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

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa memahami perbedaan konsep dan definisi Queue
2. Mahasiswa mampu menerapkan Queue dalam pemrograman

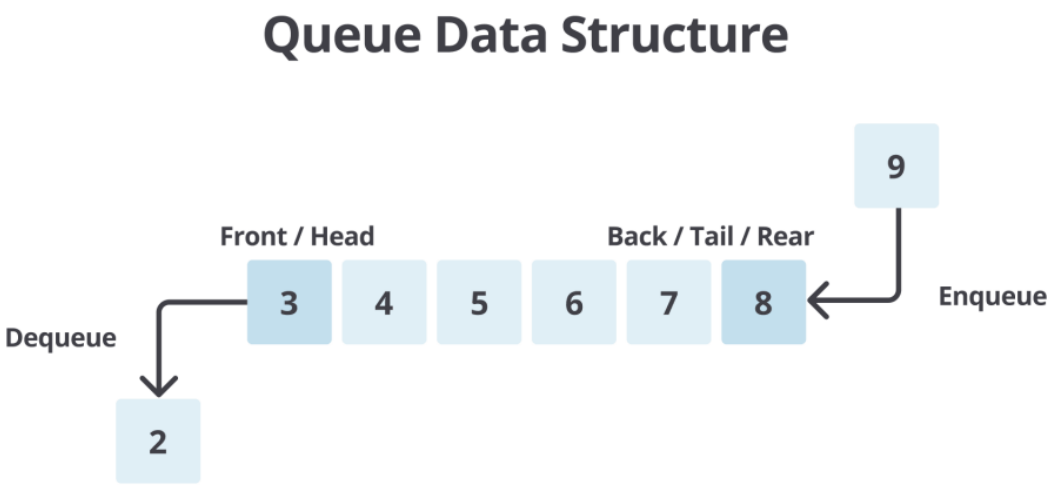
BAB II

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 III

GUIDED

1. Guided 1

Source code

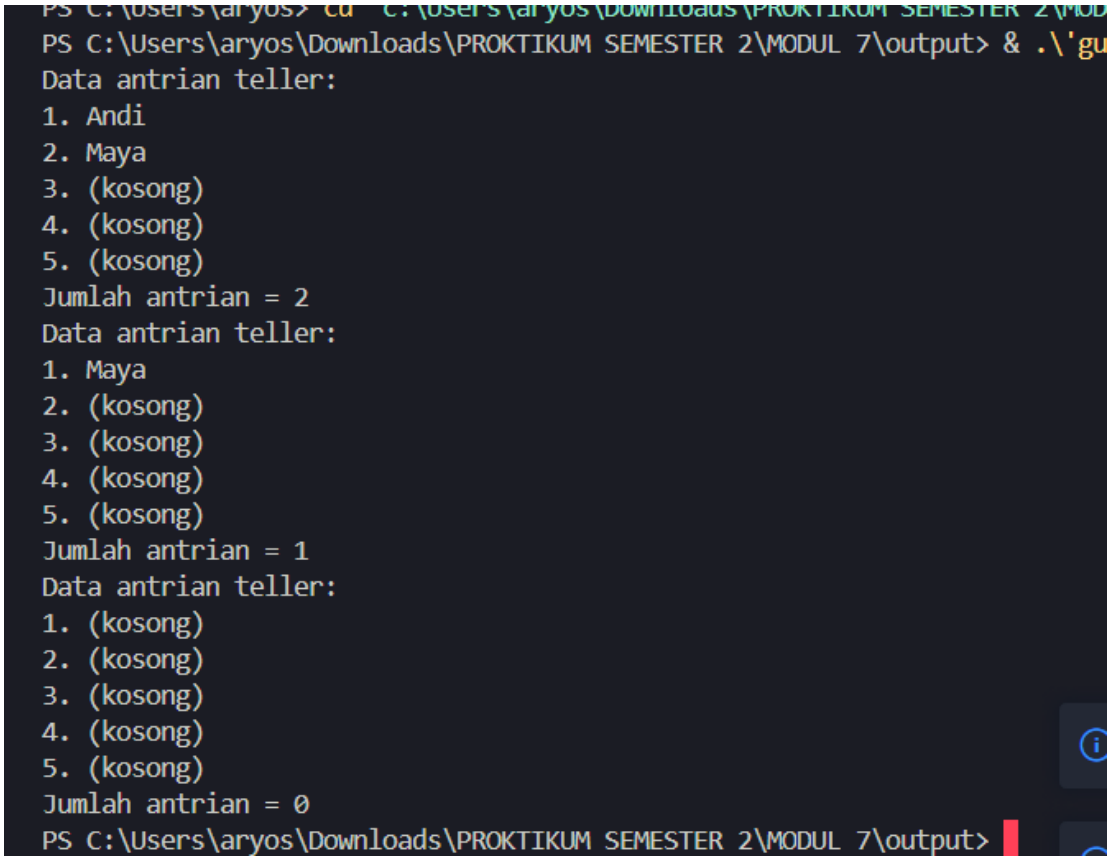
```
#include <iostream>

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 merupakan antrian yang dimana memiliki kapasitas maksimal sebanyak 5 elemen, diwakili oleh array queueTeller. Program ini menyediakan beberapa fungsi untuk mengelola antrian, termasuk pengecekan apakah antrian penuh atau kosong (isFull() dan isEmpty()), menambah elemen ke dalam antrian (enqueueAntrian()), menghapus elemen dari antrian (dequeueAntrian()), menghitung jumlah elemen dalam antrian (countQueue()), mengosongkan seluruh elemen dalam antrian (clearQueue()), dan menampilkan isi antrian (viewQueue()). Pada bagian main(), program ini melakukan beberapa operasi antrian: menambah dua nama ke dalam antrian, menampilkan antrian, mengeluarkan satu elemen dari antrian, menampilkan antrian kembali, dan mengosongkan seluruh antrian, kemudian menampilkan kembali antrian yang telah kosong. Program ini dirancang untuk menunjukkan bagaimana antrian beroperasi dan bagaimana elemen-elemen di dalamnya ditambahkan dan dihapus secara berurutan.

.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

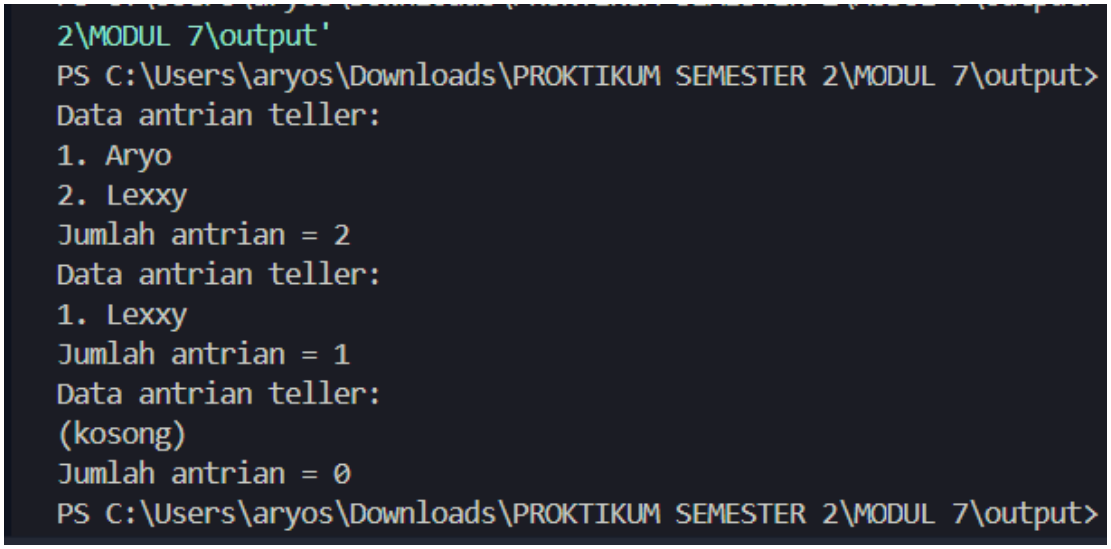
```
#include <iostream>
using namespace std;



//Aryo Tegar Sukarno
//2311102018
//IF-11-A



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

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

public:
    Queue() {
        front = nullptr;
        back = nullptr;
    }

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

```
    }

    void enqueue(string data) {
        Node* temp = new Node();
        temp->data = data;
        temp->next = nullptr;

        if (isEmpty()) {
            front = temp;
            back = temp;
        } else {
            back->next = temp;
            back = temp;
        }
    }

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

    int countQueue() {
        int count = 0;
        Node* current = front;
        while (current != nullptr) {
            count++;
```

```
            current = current->next;
        }
        return count;
    }

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

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

int main() {
    Queue q;
    q.enqueue("Aryo");
    q.enqueue("Lexxy");
    q.viewQueue();
    cout << "Jumlah antrian = " << q.countQueue() << endl;

    q.dequeue();
```

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

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

    return 0;
}
```

Screenshoot program



Deskripsi Program

Program ini merupakan antrian yang dimana memiliki kapasitas maksimal sebanyak 5 elemen, diwakili oleh array queueTeller. Program ini menyediakan beberapa fungsi untuk mengelola antrian, termasuk pengecekan apakah antrian penuh atau kosong (isFull() dan isEmpty()), menambah elemen ke dalam antrian (enqueueAntrian()), menghapus elemen dari antrian (dequeueAntrian()), menghitung jumlah elemen dalam antrian (countQueue()), mengosongkan seluruh elemen dalam antrian (clearQueue()), dan menampilkan isi antrian (viewQueue()). Pada bagian

main(), program ini melakukan beberapa operasi antrian: menambah dua nama ke dalam antrian, menampilkan antrian, mengeluarkan satu elemen dari antrian, menampilkan antrian kembali, dan mengosongkan seluruh antrian, kemudian menampilkan kembali antrian yang telah kosong. Program ini dirancang untuk menunjukkan bagaimana antrian beroperasi dan bagaimana elemen-elemen di dalamnya ditambahkan dan dihapus secara berurutan.

.

2. Unguided 2

Source code

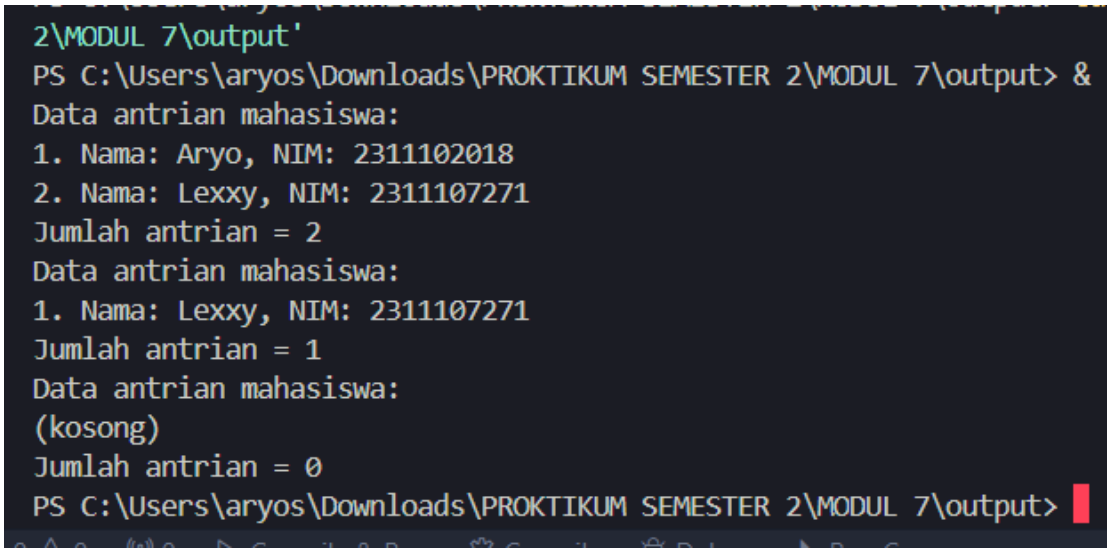
```
#include <iostream>
using namespace std;



//Aryo Tegar Sukarno
//2311102018
//IF-11-A



struct Student {
    string nama;
    string NIM;
    Student* next;
};

class Queue {
```

```
private:
    Student* front;
    Student* back;

public:
    Queue() {
        front = nullptr;
        back = nullptr;
    }

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

    void enqueue(string nama, string NIM) {
        Student* temp = new Student();
        temp->nama = nama;
        temp->NIM = NIM;
        temp->next = nullptr;

        if (isEmpty()) {
            front = temp;
            back = temp;
        } else {
            back->next = temp;
            back = temp;
        }
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            Student* temp = front;
```

```
            front = front->next;
            delete temp;
            if (front == nullptr) {
                back = nullptr;
            }
        }
    }

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

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

    void viewQueue() {
        cout << "Data antrian mahasiswa:" << endl;
        Student* current = front;
        int index = 1;
        while (current != nullptr) {
            cout << index << ". Nama: " << current->nama << ",
NIM: " << current->NIM << endl;
            current = current->next;
            index++;
        }
```

```
        if (isEmpty()) {
            cout << "(kosong)" << endl;
        }
    }
};

int main() {
    Queue q;
    q.enqueue("Aryo", "2311102018");
    q.enqueue("Lexxy", "2311107271");
    q.viewQueue();
    cout << "Jumlah antrian = " << q.countQueue() << endl;

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

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

    return 0;
}
```

Screenshoot program



Deskripsi Program

Program ini merupakan antrian (queue) yang menggunakan linked list untuk menyimpan data mahasiswa. Setiap elemen dalam antrian adalah sebuah objek Student yang berisi nama dan NIM mahasiswa. Kelas Queue mengatur antrian ini dengan dua pointer, front dan back, yang menunjukkan elemen pertama dan terakhir dalam antrian. Metode-metode dalam kelas ini mencakup pengecekan apakah antrian kosong (isEmpty()), menambahkan mahasiswa ke dalam antrian (enqueue()), menghapus mahasiswa dari antrian (dequeue()), menghitung jumlah mahasiswa dalam antrian (countQueue()), mengosongkan seluruh antrian (clearQueue()), dan menampilkan semua mahasiswa yang ada dalam antrian (viewQueue()). Pada fungsi main(), program ini melakukan beberapa operasi antrian: menambah dua mahasiswa ke dalam antrian, menampilkan isi antrian beserta jumlah mahasiswa, menghapus satu mahasiswa dari antrian, dan kemudian mengosongkan seluruh antrian. Program ini menunjukkan bagaimana antrian dapat dikelola menggunakan linked list untuk menyimpan dan mengelola data secara dinamis.

BAB IV

KESIMPULAN

Dari ketiga program diatas dapat disimpulkan bahwa Program pertama menggunakan array statis untuk mengelola antrian, dengan fungsi-fungsi untuk memeriksa apakah antrian penuh atau kosong, menambahkan dan menghapus elemen, menghitung jumlah elemen, mengosongkan, dan menampilkan isi antrian. Pendekatan ini sederhana namun terbatas oleh ukuran tetap dari array. Program kedua menggunakan linked list, di mana setiap elemen adalah sebuah node yang menyimpan data string. Kelas Queue dalam program ini menawarkan fleksibilitas yang lebih besar dibandingkan array statis karena ukuran antrian dapat bertambah secara dinamis. Metode yang disediakan mencakup pengecekan antrian kosong, penambahan dan penghapusan elemen, perhitungan jumlah elemen, pengosongan antrian, dan penampilan isi antrian. Program ketiga juga menggunakan linked list, tetapi lebih kompleks karena setiap elemen antrian adalah objek Student yang menyimpan nama dan NIM mahasiswa. Kelas Queue dalam program ini dikelola dengan metode yang mirip dengan program kedua, namun dengan data yang lebih kaya. Pendekatan ini menunjukkan kemampuan linked list untuk mengelola data yang lebih kompleks secara dinamis. Secara umum, implementasi menggunakan linked list lebih disarankan untuk aplikasi yang memerlukan antrian dengan ukuran dinamis atau data yang bervariasi, sementara array statis cocok untuk kasus yang lebih sederhana dengan ukuran tetap.