LAPORAN PRAKTIKUM
MODUL 7
QUEUE

Disusun oleh:

Diva Octaviani

NIM : 2311102006

Dosen Pengampu:

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

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

BAB II
DASAR TEORI

Pada struktur data Queue atau antrian adalah sekumpulan data yang mana penambahan elemen hanya bisa dilakukan pada suatu ujung disebut dengan sisi belakang (rear), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain (disebut dengan sisi depan atau front).

Pada Stack atau tumpukan menggunakan prinsip “Masuk terakhir keluar pertama”atau LIFO (Last In First Out), Maka pada Queue atau antrian prinsip yang digunakan adalah “Masuk Pertama Keluar Pertama” atau FIFO (First In First Out).

Queue atau antrian banyak kita jumpai dalam kehidupan sehari-hari, ex: antrian Mobil diloket Tol, Antrian mahasiswa Mendaftar, dll. Contoh lain dalam bidang komputer adalah pemakaian sistem komputer berbagi waktu(time-sharing computer system) dimana ada sejumlah pemakai yang akan menggunakan sistem tersebut secara serempak.

Pada Queue atau antrian Terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung satunya dimana membutuhkan variabel Head dan Tail (depan/front, belakang/rear).

Operasi-operasi Queue :

1. Create()

Untuk menciptakan dan menginisialisasi Queue

2. IsEmpty()

Untuk memeriksa apakah Antrian masih kosong

3. IsFull()

Untuk mengecek apakah Antrian sudah penuh atau belum

4. Enqueue ()

Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang

5. Dequeue()

Digunakan untuk menghapus elemen terdepan/pertama (head) dari Antrian

6. Clear()

Untuk menghapus elemen-elemen Antrian dengan cara membuat Tail dan Head = -1

7. Tampil()

Untuk menampilkan nilai-nilai elemen Antrian.

BAB III

GUIDED

1. GUIDED 1

SOURCE CODE

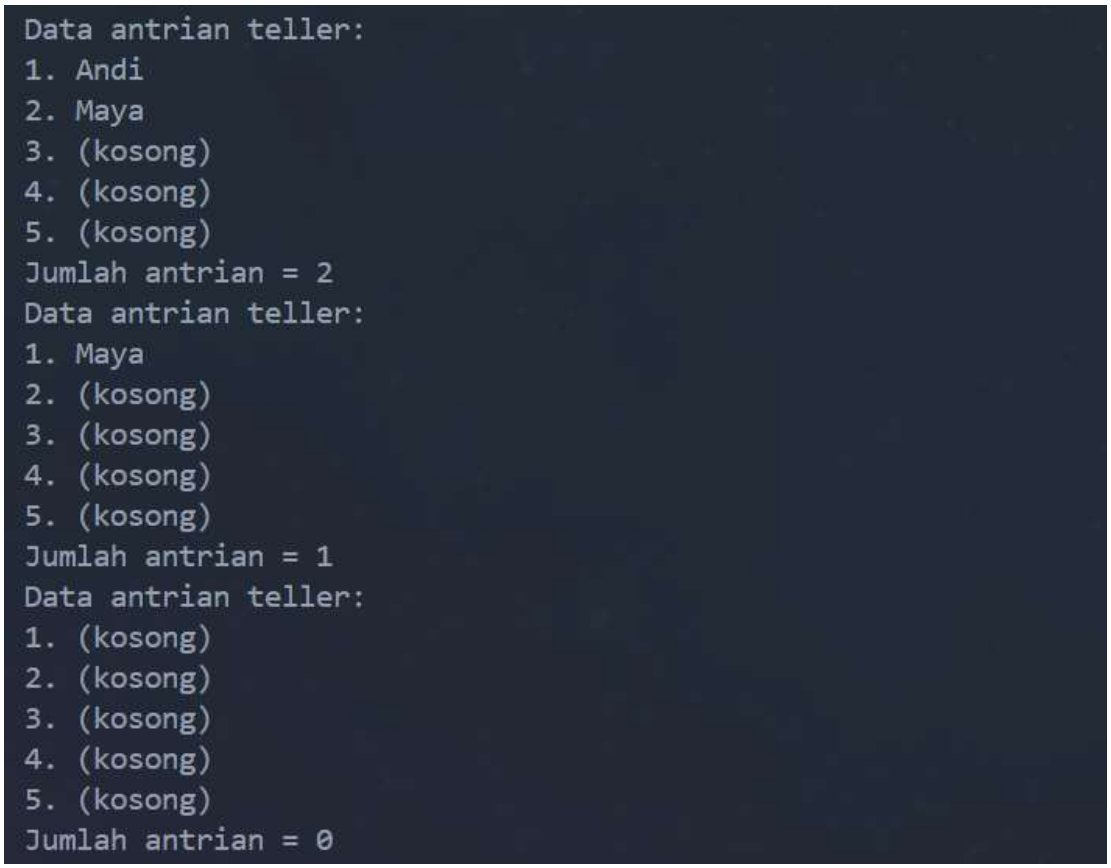
```
#include <iostream>
using namespace std;

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

Pada code diatas menggunakan array untuk menyimpan elemen-elemen antrian. enqueueAntrian(string data): Fungsi ini digunakan untuk menambahkan elemen baru ke dalam antrian. Jika antrian sudah penuh, akan ditampilkan pesan "Antrian penuh". Jika antrian kosong, elemen baru akan ditempatkan di indeks 0. Jika antrian sudah berisi elemen, elemen baru akan ditempatkan di indeks back.

dequeueAntrian(): Fungsi ini digunakan untuk menghapus elemen pertama dari antrian. Jika antrian kosong, akan ditampilkan pesan "Antrian kosong". Jika antrian berisi elemen, elemen-elemen akan digeser ke kiri untuk mengisi posisi yang kosong.

UNGUIDED

1. UNGUIDED 1

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

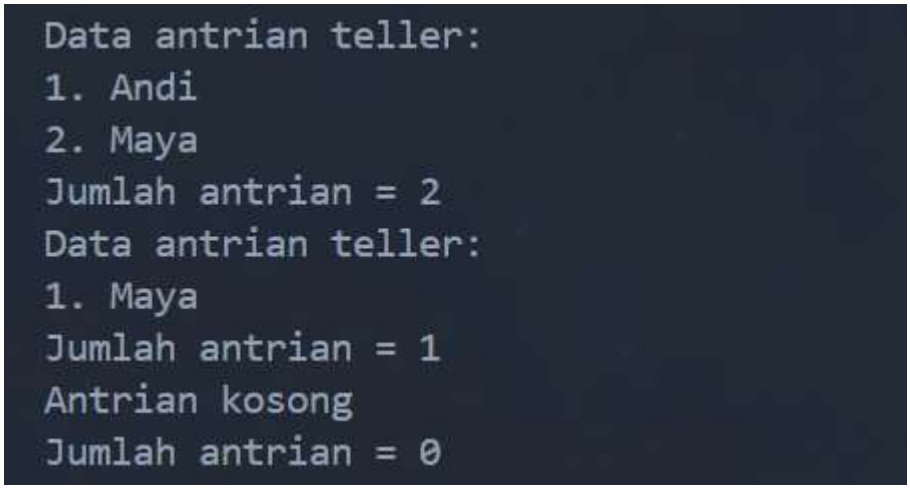
SOURCE CODE

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

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

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

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

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

    void enqueue(string data) {
        Node* newNode = new Node;
        newNode->data = data;
```

```
        newNode->next = NULL;

        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;
            delete temp;
        }
    }

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

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

```
        }
    }

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

int main() {
    Queue queue;
    queue.enqueue("Andi");
    queue.enqueue("Maya");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Pada code di atas mengimplementasikan antrian menggunakan linked list. dequeue(): Metode ini digunakan untuk menghapus elemen pertama dari antrian.
countQueue(): Method ini digunakan untuk menghitung jumlah elemen dalam antrian.
clearQueue(): Method ini digunakan untuk menghapus semua elemen dalam antrian.

2. UNGUIDED 2

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

SOURCE CODE

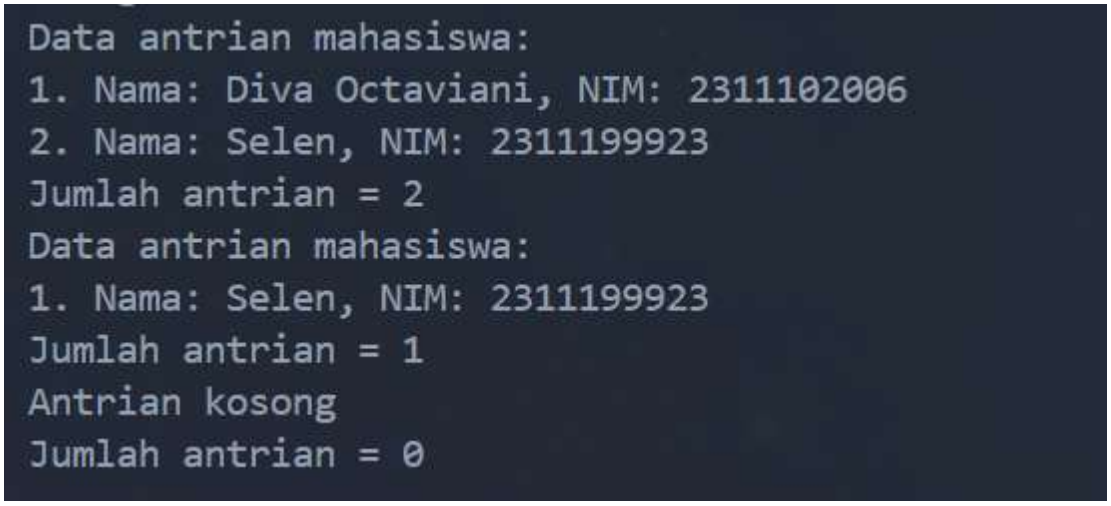
```
#include <iostream>
using namespace std;

struct Node {
    string nama;
```

```
    string nim;
    Node* next;
};
class Queue {
private:
    Node* front;
    Node* back;

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

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

    void enqueue(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = NULL;

        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;
            delete temp; }
    }

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

    void clearQueue() {
        while (!isEmpty()) {
            dequeue();
        }
    }
    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            cout << "Data antrian mahasiswa:" << endl;
            Node* current = front;
            int position = 1;
            while (current != NULL) {
                cout << position << ". Nama: " << current->nama
<< ", NIM: " << current->nim << endl;
                current = current->next;
```

```
                position++;
            }
        }
    }
};
int main() {
    Queue queue;
    queue.enqueue("Diva Octaviani", "2311102006");
    queue.enqueue("Selen", "2311199923");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Pada code diatas terdapat class queue, class Queue adalah implementasi dari antrian. Di dalam class ini, terdapat dua pointer yaitu front dan back, yang menunjukkan ke Node pertama dan terakhir dalam antrian. Metode-metode dalam class ini digunakan untuk melakukan operasi-operasi seperti menambahkan elemen ke dalam antrian (enqueue), menghapus elemen dari antrian (dequeue), menghitung jumlah elemen dalam antrian (countQueue), membersihkan antrian (clearQueue), dan melihat elemen-elemen dalam antrian (viewQueue).

BAB IV
KESIMPULAN

Setelah melakukan pembelajaran mengenai Queue di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Queue dapat diimplementasikan menggunakan berbagai cara, seperti array, linked list, atau custom data structure lainnya.
2. Di C++, queue dapat diimplementasikan dengan menggunakan STL (Standard Template Library) yang menyediakan kelas queue.
3. Queue dapat digunakan untuk mensimulasikan antrian orang di berbagai tempat, seperti kasir supermarket, loket bank, atau layanan pelanggan.

DAFTAR PUSTAKA

Unkown. (2014, 07 Mei) [C++] Konsep Queue. diakses pada 28 Mei 2024 dari https://www.nblognlife.com/2014/05/c-konsep-queue.html