LAPORAN PRAKTIKUM
MODUL 7
QUEUE

Disusun oleh:

Reza Alvonzo

NIM : 2311102026

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

Queue adalah konsep yang mendasar dan sering digunakan dalam berbagai aplikasi komputer. Dalam dunia nyata, kita sering menjumpai antrian di berbagai tempat seperti toko, bank, atau layanan pelanggan. Konsep antrian ini diadaptasi ke dalam pemrograman untuk mengatur dan mengelola data dengan prinsip First-In-First-Out (FIFO). Dalam artikel ini, kita akan menjelajahi konsep dasar Queue, implementasinya menggunakan array dan linked list, serta operasi dasar yang terkait. Mari kita mempelajari secara mendalam tentang Queue dan memahami mengapa ini adalah struktur data yang penting untuk dipahami oleh para pengembang perangkat lunak.

Queue adalah struktur data yang mengikuti prinsip First-In-First-Out (FIFO), yang berarti elemen pertama yang masuk ke dalam antrian akan menjadi elemen pertama yang keluar dari antrian. Queue dapat diibaratkan sebagai antrian di mana elemen-elemen baru ditambahkan di satu ujung antrian (rear) dan elemen-elemen yang sudah ada dikeluarkan di ujung lainnya (front). Queue mirip dengan antrian nyata yang sering kita temui dalam kehidupan sehari-hari.

Karakteristik utama dari Queue adalah prinsip FIFO (First-In-First-Out). Elemen pertama yang dimasukkan ke dalam Queue akan menjadi elemen pertama yang diambil atau dihapus dari Queue. Elemen-elemen baru ditambahkan di ujung belakang Queue dan elemen-elemen yang sudah ada dikeluarkan dari ujung depan Queue. Dengan prinsip FIFO ini, Queue dapat membantu mengatur urutan data dan mempertahankan prioritas saat memproses elemen-elemen yang ada di dalamnya.

BAB III

GUIDED

1. GUIDED 1

SOURCE CODE

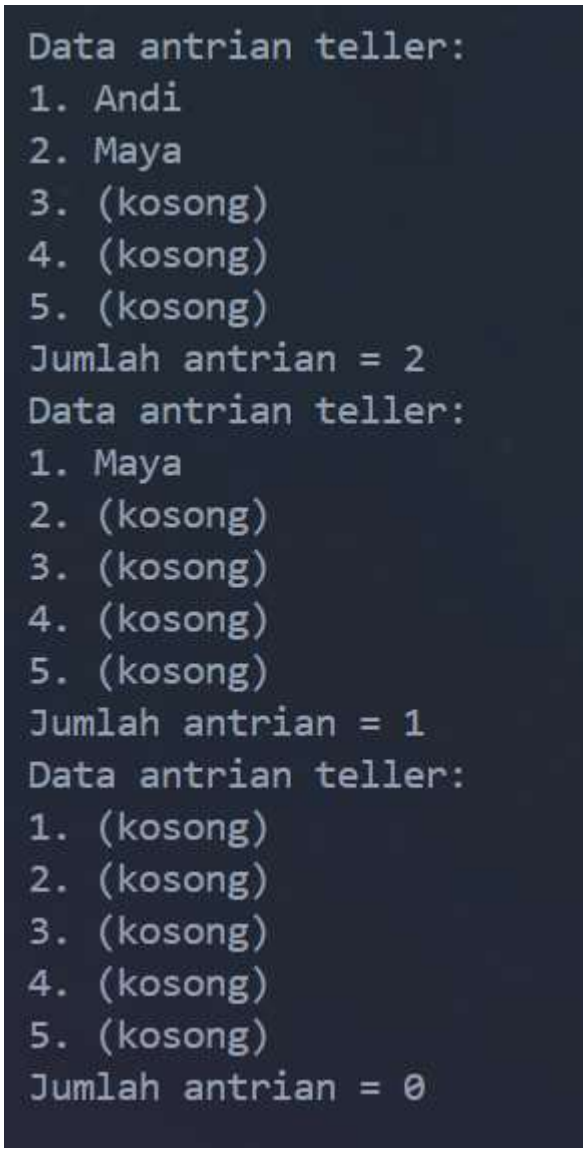
```
#include <iostream>
using namespace std;
string arrayBuku[5];
int maksimal = 5, top = 0;
bool isFull()
{
    return (top == maksimal);
}
bool isEmpty()
{
    return (top == 0);
}
void pushArrayBuku(string data)
{
    if (isFull())
    {
        cout << "Data telah penuh" << endl;
    }
    else
    {
        arrayBuku[top] = data;
        top++;
    }
}
void popArrayBuku()
{
    if (isEmpty())
    {
```

```
        cout << "Tidak ada data yang dihapus" << endl;
    }
    else
    {
        arrayBuku[top - 1] = "";
        top--;
    }
}
void peekArrayBuku(int posisi)
{
    if (isEmpty())
    {
        cout << "Tidak ada data yang bisa dilihat" << endl;
    }
    else
    {
        int index = top;
        for (int i = 1; i <= posisi; i++)
        {
            index--;
        }
        cout << "Posisi ke " << posisi << " adalah " <<
arrayBuku[index] << endl;
    }
}
int countStack()
{
    return top;
}
void changeArrayBuku(int posisi, string data)
{
    if (posisi > top)
    {
        cout << "Posisi melebihi data yang ada" << endl;
```

```
    }
    else
    {
        int index = top;
        for (int i = 1; i <= posisi; i++)
        {
            index--;
        }
        arrayBuku[index] = data;
    }
}
void destroyArraybuku()
{
    for (int i = top; i >= 0; i--)
    {
        arrayBuku[i] = "";
    }
    top = 0;
}
void cetakArrayBuku()
{
    if (isEmpty())
    {
        cout << "Tidak ada data yang dicetak" << endl;
    }
    else
    {
        for (int i = top - 1; i >= 0; i--)
        {
            cout << arrayBuku[i] << endl;
        }
    }
}
int main()
```

```
{

    pushArrayBuku("Kalkulus");
    pushArrayBuku("Struktur Data");
    pushArrayBuku("Matematika Diskrit");
    pushArrayBuku("Dasar Multimedia");
    pushArrayBuku("Inggris");
    cetakArrayBuku();
    cout << "\n";
    cout << "Apakah data stack penuh? " << isFull() << endl;
    cout << "Apakah data stack kosong? " << isEmpty() << endl;
    peekArrayBuku(2);
    popArrayBuku();
    cout << "Banyaknya data = " << countStack() << endl;
    changeArrayBuku(2, "Bahasa Jerman");
    cetakArrayBuku();
    cout << "\n";
    destroyArraybuku();
    cout << "Jumlah data setelah dihapus: " << top << endl;
    cetakArrayBuku();
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

fungsi enqueueAntrian() untuk menambahkan data baru ke antrian. Fungsi ini pertama-tama memeriksa apakah antrian penuh dengan menggunakan fungsi isFull(). Jika antrian penuh, fungsi akan menampilkan pesan "Antrian penuh".

UNGUIDED

1. UNGUIDED 1

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

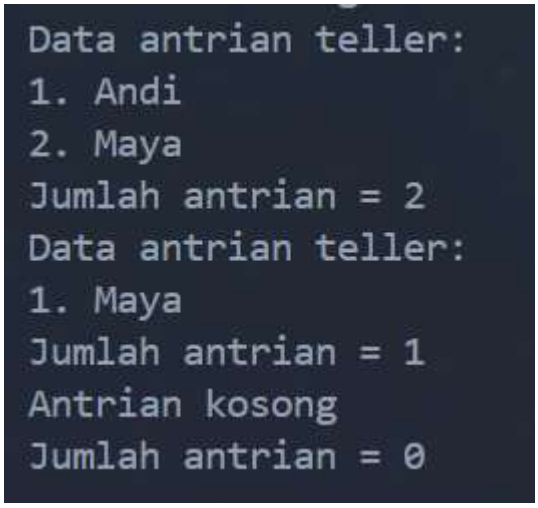
SOURCE CODE

```
#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

Implementasi fungsi enqueue(string data) untuk menambahkan data baru ke antrian. Fungsi ini membuat node baru (newNode) dengan data yang diberikan dan menunjuk ke NULL (karena node baru ini menjadi node terakhir).

2. UNGUIDED 1

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

SOURCE CODE

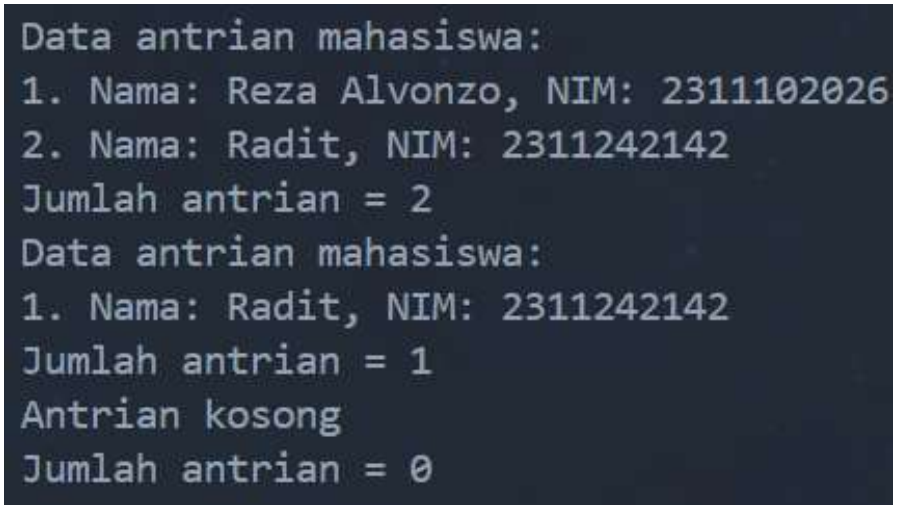
```
#include <iostream>
using namespace std;



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("Reza Alvonzo", "2311102026");
    queue.enqueue("Radit", "2311242142");
    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

program antrian mahasiswa menggunakan linked list. Mahasiswa dapat ditambahkan dengan fungsi enqueue, dikeluarkan dengan fungsi dequeue, dan jumlah mahasiswa dapat dilihat dengan fungsi countQueue.

BAB IV
KESIMPULAN

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

1. Queue adalah struktur data yang memungkinkan operasi tambah (enqueue) dan ambil (dequeue) elemen secara berurutan.
2. Queue memiliki dua sisi: depan (front) dan belakang (rear). Elemen yang ditambahkan pertama kali berada di belakang, sedangkan elemen yang diambil pertama kali berada di depan.
3. tetapi memiliki kekurangan dalam menghapus elemen secara acak dan menambahkan elemen di tengah.

DAFTAR PUSTAKA

Rizky, Pratama. (2023, 25 Mei) Queue: Pengenalan, Implementasi, Operasi Dasar, dan Aplikasi. diakses pada 28 Mei 2024 dari https://medium.com/@furatamarizuki/queue-pengenalan-implementasi-operasi-dasar-dan-aplikasi-c5eed7e871a3