

LAPORAN PRAKTIKUM

MODUL 7

QUEUE



Disusun oleh:

ALDI RANSI

NIM : 2311102328

Dosen Pengampu:

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

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

Pada praktikum kali ini di harapkan mahasiswa dapat mampu

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

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.

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

```
PERKULIAHAN TEKNIK FARMASI (SEMESTER 2) FARMASI  
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

enqueueAntrian(string data)pada fungsi ini di gunakan untuk menambahkan antrian. Jika antrian sudah terpenuh, 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..

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;

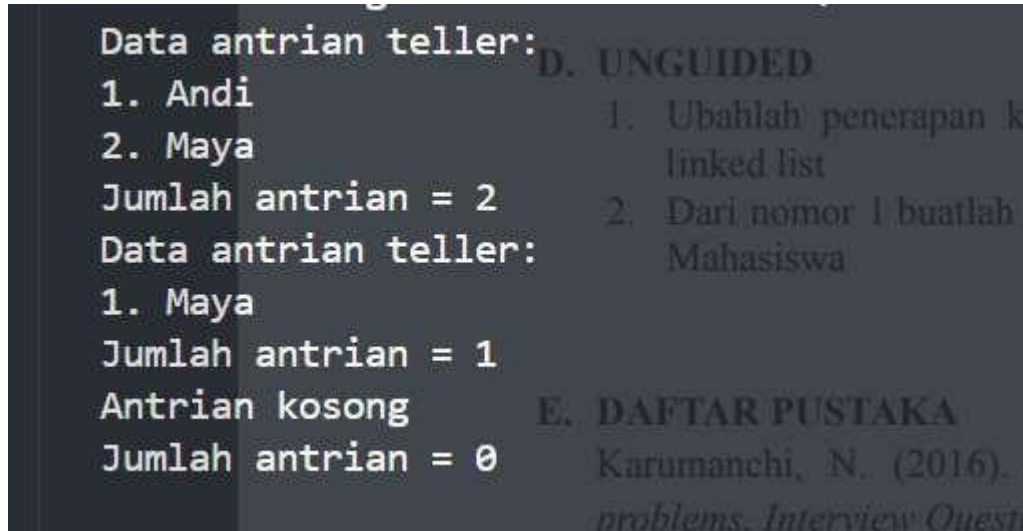
            current = current->next;
            position++;
        }
    }
}

};

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



```
Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
Antrian kosong
Jumlah antrian = 0
```

DESKRIPSI PROGRAM

Pada fungsi main(), objek Queue dibuat dan beberapa operasi antrian dilakukan. Elemen-elemen "Andi" dan "Maya" ditambahkan ke dalam antrian menggunakan metode enqueue(). Kemudian, antrian ditampilkan menggunakan metode viewQueue(), dan jumlah elemen dalam antrian ditampilkan menggunakan metode countQueue(). Selanjutnya, elemen pertama dihapus menggunakan metode dequeue(), dan antrian kembali ditampilkan. Terakhir, semua elemen dalam antrian dihapus menggunakan metode clearQueue(), dan antrian ditampilkan kembali.

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("aldi", "2311102328");
    queue.enqueue("kagak",
"2311102000");

    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

```
1. Nama: aldi, NIM: 2311102328
2. Nama: kakak, NIM: 2311102000
Jumlah antrian = 2
Data antrian mahasiswa:
1. Nama: kakak, NIM: 2311102000
Jumlah antrian = 1
Antrian kosong
Jumlah antrian = 0
```

DESKRIPSI PROGRAM

Stack: Digunakan untuk menyimpan karakter-karakter dari setiap kata sebelum dibalikkan. Stringstream: Digunakan untuk memproses string kata per kata. Fungsi reverseWords: Fungsi utama yang membalikkan huruf-huruf dalam setiap kata.

BAB IV

KESIMPULAN

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

1. Dengan memahami operasi dasar Queue dan implementasinya, mahasiswa dapat menerapkan konsep ini dalam berbagai aplikasi pemrograman, terutama yang memerlukan pengelolaan data dengan metode FIFO.
2. Implementasi menggunakan array memperlihatkan batasan ukuran antrian yang tetap, sedangkan linked list memberikan fleksibilitas ukuran antrian yang dinamis.
3. Melalui praktikum ini, mahasiswa dapat memahami dan mengimplementasikan konsep dasar Queue menggunakan dua pendekatan yang berbeda: array dan linked list.

DAFTAR PUSTAKA

Karumanchi, N. (2016). Data Structures and algorithms made easy: Concepts, problems, Interview Questions. CareerMonk Publications.