LAPORAN PRAKTIKUM

MODUL VII
QUEUE

Disusun oleh:
FATTAH RIZQY ADHIPRATAMA
NIM: 2311102019

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

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

BAB I

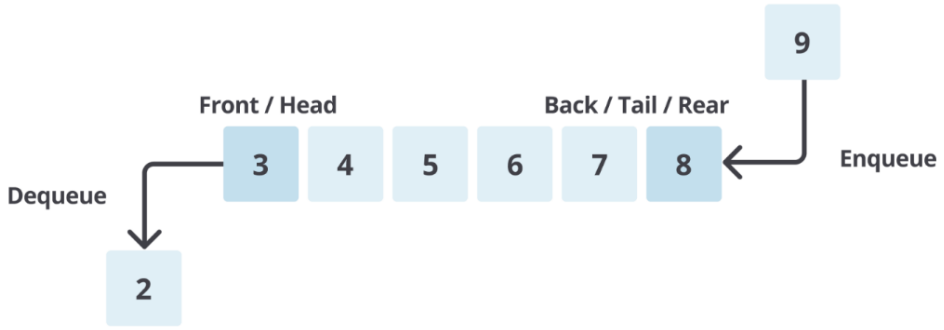
TUJUAN PRAKTIKUM

1. Mahasiswa 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

Pengertian queue adalah antrian. Konsep antrian dalam struktur data sangat berguna. Pada queue, kita hanya bisa menambahkan elemen baru di satu sisi yang disebut rear, sementara untuk menghapus atau mengambil elemen dilakukan dari sisi lain. Caranya bekerja adalah seperti jejeran orang yang sedang menunggu antrean di supermarket di mana orang pertama yang datang adalah yang pertama dilayani (First In, First Out). Pada struktur data ini, urutan pertama (data yang akan dikeluarkan) disebut Front atau Head. Sebaliknya, data pada urutan terakhir (data yang baru saja ditambahkan) disebut Back, Rear, atau Tail. Proses untuk menambahkan data pada antrean disebut dengan Enqueue, sedangkan proses untuk menghapus data dari antrean disebut dengan Dequeue.

Queue memiliki peran yang penting dalam berbagai aplikasi dan algoritma. Salah satu fungsi utamanya adalah mengatur dan mengelola antrean tugas atau operasi secara efisien. Dalam sistem komputasi, ia digunakan untuk menangani tugas-tugas seperti penjadwalan proses, antrean pesan, dan manajemen sumber daya.



Operasi pada Queue :

1. enqueue() : menambahkan data ke dalam queue.
2. dequeue() : mengeluarkan data dari queue.
3. peek() :mengambil data dari queue tanpa menghapusnya.

4. isEmpty() : mengecek apakah queue kosong atau tidak.
5. isFull(): mengecek apakah queue penuh atau tidak.
6. 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

```
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
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 7>
```

Deskripsi program

Program tersebut adalah implementasi sederhana dari sebuah antrian (queue) dalam bahasa pemrograman C++. Program ini dapat digunakan untuk mensimulasikan antrian teller di bank dengan mudah dan efisien. Program ini memungkinkan pengguna untuk menambahkan, menghapus, dan melihat data nasabah dalam antrian.

LATIHAN KELAS - UNGUIDED

1. **Unguided 1**

   Source code

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

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

Node* front = nullptr; // Pointer depan
Node* back = nullptr; // Pointer belakang

void enqueueAntrian(string data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = nullptr;
    if (front == nullptr) { // Jika antrian kosong
        front = newNode;
        back = newNode;
    } else { // Jika antrian tidak kosong
        back->next = newNode;
        back = newNode;
    }
}

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

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

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

```
Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
Data antrian teller:
Jumlah antrian = 0
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 7>
```

Deskrirpsi Program

Program ini adalah implementasi dari struktur data queue yang menggunakan linked list. Program ini mensimulasikan antrian di mana item ditambahkan dan dihapus dalam urutan First In First Out (FIFO). Ini mencakup fungsi untuk mengantrekan (menambah) item, membatalkan antrean (menghapus) item, menghitung jumlah item dalam antrian, menghapus antrian, dan melihat item dalam antrian.

2. Unguided 2

Source Code

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

struct Mahasiswa {
    string nama;
    string nim;
    Mahasiswa* next;
};

Mahasiswa* front = nullptr; // Pointer depan
Mahasiswa* back = nullptr; // Pointer belakang

void enqueueAntrian(string nama, string nim) {
    Mahasiswa* newNode = new Mahasiswa;
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;
```

```
    if (front == nullptr) { // Jika antrian kosong
        front = newNode;
        back = newNode;
    } else { // Jika antrian tidak kosong
        back->next = newNode;
        back = newNode;
    }
}

void dequeueAntrian() {
    if (front == nullptr) {
        cout << "Antrian kosong" << endl;
    } else {
        Mahasiswa* temp = front;
        front = front->next;
        delete temp;
    }
}

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

void clearQueue() {
    while (front != nullptr) {
        Mahasiswa* temp = front;
        front = front->next;
        delete temp;
    }
    back = nullptr;
}

void viewQueue() {
    cout << "Data antrian mahasiswa:" << endl;
    Mahasiswa* current = front;
```

```
    int i = 1;
    while (current != nullptr) {
        cout << i << ". Nama: " << current->nama << ", NIM: "
<< current->nim << endl;
        current = current->next;
        i++;
    }
}

int main() {
    enqueueAntrian("Fattah", "2311102019");
    enqueueAntrian("Tristan", "2311102999");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}
```

Screenshoot Program

```
Data antrian mahasiswa:
1. Nama: Fattah, NIM: 2311102019
2. Nama: Tristan, NIM: 2311102999
Jumlah antrian = 2
Data antrian mahasiswa:
1. Nama: Tristan, NIM: 2311102999
Jumlah antrian = 1
Data antrian mahasiswa:
Jumlah antrian = 0
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 7>
```

Deskripsi Program

Program ini adalah implementasi sederhana dari struktur data queue. Program ini menyediakan fungsi-fungsi seperti enqueueAntrian untuk menambahkan elemen ke dalam antrian, dequeueAntrian untuk menghapus elemen dari depan antrian, countQueue untuk menghitung jumlah total elemen dalam antrian, dan clearQueue untuk mengosongkan seluruh antrian. Selain itu, terdapat fungsi

viewQueue untuk menampilkan isi dari antrian, yang dalam hal ini adalah nama dan NIM mahasiswa.

BAB IV

KESIMPULAN

Queue atau antrian adalah struktur data yang mengatur elemen-elemen data dengan prinsip "First In, First Out" (FIFO). Artinya, elemen data yang pertama masuk akan diproses terlebih dahulu, dan elemen data yang terakhir masuk akan diproses terakhir. Queue sering dianalogikan dengan antrean di kehidupan sehari-hari, di mana orang yang datang pertama kali akan dilayani terlebih dahulu.

Queue dapat diimplementasikan menggunakan berbagai struktur data, seperti array, linked list, dan circular queue. Masing-masing struktur data memiliki kelebihan dan kekurangannya sendiri.

DAFTAR PUSTAKA

[1] Maulana, 2023. Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya. Diakses pada tanggal 25 Mei 2024, dari https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/

[2] Herbi, 2021. Definisi dan Contoh Queue Struktur Data. Diakses pada tanggal 25 Mei 2024, dari https://semutaspal.com/queue/