LAPORAN PRAKTIKUM STRUKTUR DATA DAN PEMROGRAMAN

MODUL VII :

QUEU

Disusun Oleh :

BAHARUDDIN BARKAH PRATAMA

2311102321

IF-11-A

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

A. TUJUAN PRAKTIKUM

a. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
b. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
c. Mahasiswa mampu menerapkan operasi tampil data pada queue

BAB II
DASAR TEORI

B. 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.

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

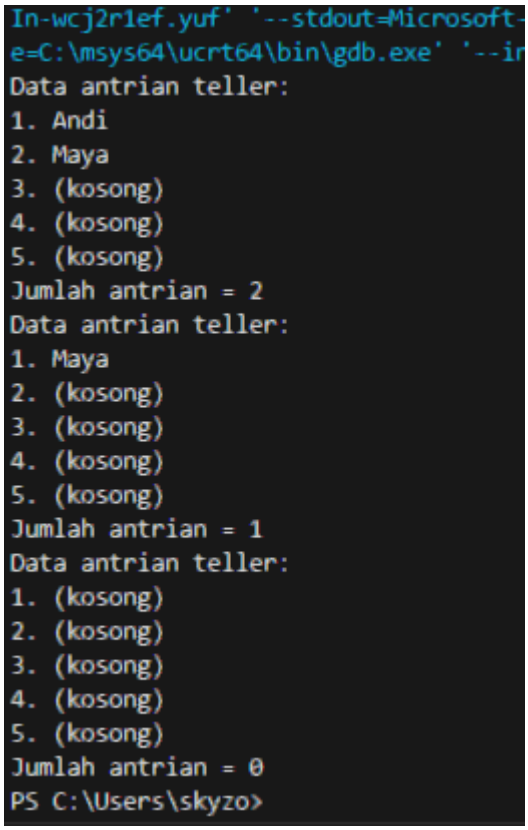
C. GUIDED

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 {
```

```
    } 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



DESKDRIPSI PROGRAM

Program ini mengimplementasikan antrian (queue) menggunakan array statis dalam bahasa C++ dengan kapasitas maksimal lima elemen. Fungsi `isFull` dan `isEmpty` memeriksa apakah antrian penuh atau kosong. Fungsi `enqueueAntrian` menambah elemen baru jika antrian tidak penuh, dan `dequeueAntrian` menghapus elemen terdepan jika antrian tidak kosong. Fungsi `countQueue` menghitung jumlah elemen, `clearQueue` mengosongkan antrian, dan `viewQueue` menampilkan isi antrian. Dalam fungsi `main`, program menambah dua elemen ke antrian, menampilkan isi dan jumlah elemen, menghapus satu elemen, menampilkan isi kembali, dan mengosongkan antrian lalu menampilkannya lagi.

BAB IV

UNGUIDED

D. UNGUIDED

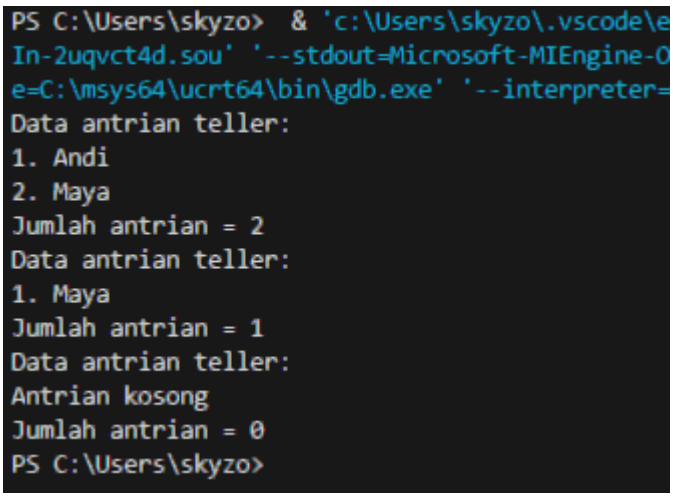
UNGUIDED 1

SOURCE CODE

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


// Definisi struktur Node
struct Node {

    string data;

    Node* next;

};


// Deklarasi variabel untuk front dan back
Node* front = nullptr;

Node* back = nullptr;


// Fungsi untuk mengecek apakah queue penuh (tidak relevan
untuk linked list)

bool isFull() {

    return false; // Linked list tidak terbatas ukuran

}


// Fungsi untuk mengecek apakah queue kosong
bool isEmpty() {

    return front == nullptr;

}


// Fungsi untuk menambahkan elemen ke dalam queue
void enqueueAntrian(string data) {

    Node* newNode = new Node();
    newNode->data = data;
    newNode->next  =  nullptr;
    if (isEmpty()) {

        front = back = newNode;
```

```
        delete temp;
    }
}

// Fungsi untuk menghitung jumlah elemen dalam queue
int countQueue() {
    int count = 0;
    Node* temp = front;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

// Fungsi untuk menghapus semua elemen dalam queue
void clearQueue() {
    while (!isEmpty()) {
        dequeueAntrian();
    }
}

// Fungsi untuk menampilkan elemen dalam queue
void viewQueue() {
    cout << "Data antrian teller:" << endl;
    Node* temp = front;
    int index = 1;
    while (temp != nullptr) {
        cout << index << ". " << temp->data << endl;
        temp = temp->next;
        index++;
    }
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

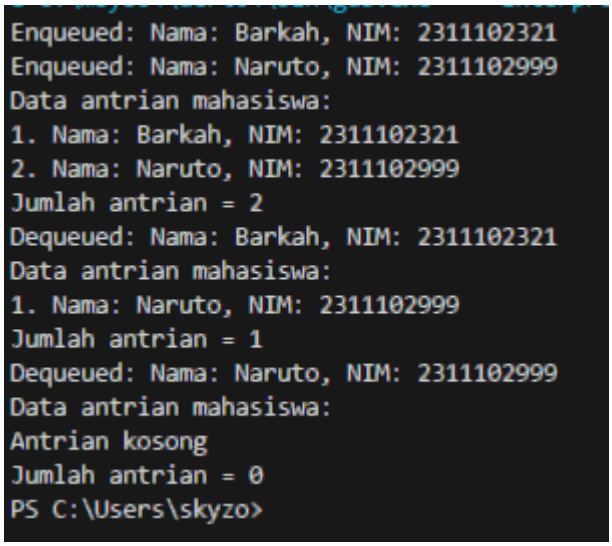
Program ini memiliki beberapa fungsi utama untuk mengelola antrian yaitu enqueueAntrian untuk menambahkan elemen baru ke akhir antrian, dequeueAntrian untuk menghapus elemen pertama dari antrian, isEmpty untuk memeriksa apakah antrian kosong atau tidak, countQueue untuk menghitung dan mengembalikan jumlah elemen dalam antrian, clearQueue untuk menghapus semua elemen dari antrian, dan viewQueue untuk menampilkan semua elemen yang ada dalam antrian. Fungsi main dalam program ini melakukan beberapa operasi pada antrian: menambahkan beberapa elemen ke antrian, menampilkan semua elemen yang telah ditambahkan, menghitung dan menampilkan jumlah elemen dalam antrian, menghapus satu elemen pertama, menghapus semua elemen yang tersisa, dan akhirnya menampilkan antrian setelah semua elemen dihapus.

UNGUIDED 2

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


// Definisi struktur Node dengan atribut nama mahasiswa dan
NIM mahasiswa

struct Node {

    string nama;
    string nim;
    Node* next;

};


// Deklarasi variabel untuk front dan back
Node* front = nullptr;

Node* back = nullptr;


// Fungsi untuk mengecek apakah queue kosong
bool isEmpty() {

    return front == nullptr;

}


// Fungsi untuk menambahkan elemen ke dalam queue
void enqueueAntrian(string nama, string nim) {

    Node* newNode = new Node();
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next  =  nullptr;
    if (isEmpty()) {

        front = back = newNode;

    } else {

        back->next = newNode;
        back = newNode;

    }

    cout << "Enqueued: Nama: " << nama << ", NIM: " << nim <<
endl;

}


// Fungsi untuk menghapus elemen dari queue
void dequeueAntrian() {

    if (isEmpty()) {
```

```
            count++;
            temp = temp->next;
        }
        return count;
    }

    // Fungsi untuk menghapus semua elemen dalam queue
    void clearQueue() {
        while (!isEmpty()) {
            dequeueAntrian();
        }
    }

    // Fungsi untuk menampilkan elemen dalam queue
    void viewQueue() {
        cout << "Data antrian mahasiswa:" << endl;
        Node* temp = front;
        int index = 1;
        while (temp != nullptr) {
            cout << index << ". Nama: " << temp->nama << ", NIM: "
    << temp->nim << endl;
            temp = temp->next;
            index++;
        }
        if (index == 1) {
            cout << "Antrian kosong" << endl;
        }
    }
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program ini mengimplementasikan antrian (queue) menggunakan linked list untuk menyimpan informasi mahasiswa, di mana setiap node berisi nama, NIM, dan pointer ke node berikutnya. Variabel global front dan back menunjuk ke node pertama dan terakhir dalam antrian. Fungsi isEmpty memeriksa apakah antrian kosong. Fungsi enqueueAntrian menambahkan node baru ke akhir antrian, sementara dequeueAntrian menghapus node pertama. Fungsi countQueue menghitung jumlah node dalam antrian, dan clearQueue menghapus semua node. Fungsi viewQueue menampilkan semua node dalam antrian. Dalam fungsi main, program menambahkan beberapa node ke antrian, menampilkan isi antrian, menghitung dan menampilkan jumlah node, menghapus node pertama, menghapus semua node yang tersisa, dan menampilkan antrian setelah setiap operasi dengan informasi tambahan tentang node yang ditambahkan atau dihapus.

BAB V
KESIMPULAN

E. KESIMPULAN

Modul ini menjelaskan cara mengimplementasikan antrian (queue) menggunakan linked list, khususnya untuk menyimpan informasi mahasiswa yang terdiri dari nama dan NIM. Antrian adalah struktur data yang mengikuti prinsip First-In First-Out (FIFO), di mana elemen yang pertama kali masuk akan menjadi elemen pertama yang keluar. Tujuan praktikum mencakup pemahaman tentang konsep double queue, serta operasi penambahan, penghapusan, dan penampilan data dalam queue. Teori dasar modul ini membahas perbedaan antara stack dan queue, serta operasi dasar pada queue seperti `enqueue` (menambah elemen), dequeue (menghapus elemen), peek (mengambil elemen tanpa menghapusnya), isEmpty (memeriksa apakah queue kosong), isFull (memeriksa apakah queue penuh), dan size (menghitung jumlah elemen dalam queue). Bagian guided memberikan contoh implementasi queue menggunakan array, lengkap dengan fungsi untuk menambah, menghapus, menampilkan, dan mengosongkan queue. Bagian unguided menginstruksikan untuk mengubah implementasi dari array menjadi linked list serta menambahkan atribut nama dan NIM pada queue. Kesimpulannya, modul ini memberikan pemahaman mendalam mengenai konsep dan implementasi antrian dalam struktur data, baik menggunakan array maupun linked list.

DAFTAR PUSTAKA

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

https://www.geeksforgeeks.org/queue-data-structure/