`LAPORAN PRAKTIKUM

MODUL VII
QUEUE

Disusun oleh:
Brian Farrel Evandhika
NIM: 2311102037

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

BAB I

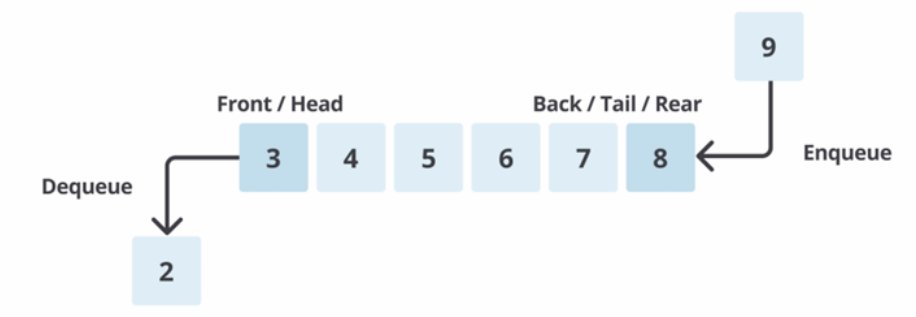
TUJUAN PRAKTIKUM

A. 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

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 maupundelete.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 menjad ielemen selanjutnya

Operasi pada Queue

- enqueue() :menambahkan data kedalam queue.
- dequeue() :mengeluarkan data dari queue.
- peek() :mengambil data dari queue tanpa menghapusnya.
- isEmpty() :mengecek apakah queue kosong atau tidak.
- isFull() :mengecek apakahq ueue penuh atau tidak.
- size() :menghitung jumlah elemen dalamqueue

Guided

Guided 1

```
#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;
}
```

Screenshot Output

```
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITME\Praktikum M
odul 3\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-Struktur-Data> cd "c:\Users\MSI GAMING\Do
cuments\GitHub\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-Struktur-Data\Modul 7\" ; if ($?)
 { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
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 C:\Users\MSI GAMING\Documents\GitHub\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-Struktur
-Data\Modul 7>
```

Deskripsi Program

Program ini mengimplementasikan struktur data queue (antrian) menggunakan array statis di C++ dengan ukuran maksimal 5. Variabel front dan back melacak posisi elemen. Fungsi isFull() dan isEmpty() memeriksa status antrian. Fungsi enqueueAntrian() menambahkan elemen jika antrian tidak penuh, sedangkan dequeueAntrian() menghapus elemen depan dan menggeser elemen lainnya. Fungsi countQueue() mengembalikan jumlah elemen, clearQueue() mengosongkan antrian, dan viewQueue() menampilkan elemen. Di main(), operasi antrian dilakukan: menambahkan elemen ("Andi" dan "Maya"), menampilkan isi dan jumlah antrian, menghapus satu elemen, menampilkan isi dan jumlah terbaru, lalu mengosongkan antrian dan menampilkan kondisi akhir

Unguided

Unguided 1

```
#include <iostream> // Library standar yang digunakan untuk input dan output
using namespace std; // Untuk mempersingkat penulisan kode program

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

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

public:
    Queue() // Membuat Queue untuk menginisialisasi front dan back
    {
        front = nullptr;
        back = nullptr;
    }

    bool isEmpty() // Membuat fungsi isEmpty untuk mengecek apakah antrian
kosong atau tidak
    {
        return front == nullptr;
    }

    void enqueueAntrian(string data) // Membuat fungsi enqueueAntrian untuk
menambahkan data ke dalam antrian
    {
        Node *newNode = new Node;
        newNode->data = data;
        newNode->next = nullptr;

        if (isEmpty())
        {
            front = newNode;
            back = newNode;
        }
        else
        {
            back->next = newNode;
            back = newNode;
        }
```

```
    }

    void dequeueAntrian() // Membuat fungsi dequeueAntrian untuk menghapus
data dari antrian
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
        {
            Node *temp = front;
            front = front->next;
            delete temp;
        }
    }

    int count() // Membuat fungsi count untuk menghitung jumlah data dalam
antrian
    {
        int count = 0;
        Node *current = front;
        while (current != nullptr)
        {
            count++;
            current = current->next;
        }
        return count;
    }

    void clear() // Membuat fungsi clear untuk menghapus semua data dalam
antrian
    {
        while (!isEmpty())
        {
            dequeueAntrian();
        }
    }

    void view() // Membuat fungsi view untuk menampilkan data dalam antrian
    {
        cout << "Data antrian teller:" << endl;
        Node *current = front;
        int position = 1;
        while (current != nullptr)
        {
            cout << position << ". " << current->data << endl;
            current = current->next;
```

```
            position++;
        }
    }
};

int main()
{
    Queue queue;
    queue.enqueueAntrian("Andi");
    queue.enqueueAntrian("Maya");
    queue.view();
    cout << "Jumlah antrian = " << queue.count() << endl;
    queue.dequeueAntrian();
    queue.view();
    cout << "Jumlah antrian = " << queue.count() << endl;
    queue.clear();
    queue.view();
    cout << "Jumlah antrian = " << queue.count() << endl;
    return 0;
}
```

Screenshot Program

```
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITME\Praktikum M
odul 3\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-Struktur-Data> cd "c:\Users\MSI GAMING\Do
cuments\GitHub\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-Struktur-Data\Modul 7\" ; if ($?)
 { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
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 C:\Users\MSI GAMING\Documents\GitHub\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-Struktur
-Data\Modul 7>
```

Deskripsi Program

Program ini mengimplementasikan antrian (queue) menggunakan linked list di C++. Struct Node mendefinisikan elemen antrian dengan data string dan pointer next. Pointer global front dan back melacak elemen pertama dan terakhir antrian, diinisialisasi ke nullptr. Fungsi isEmpty() memeriksa apakah antrian kosong. Fungsi enqueueAntrian(string data) menambahkan elemen baru ke belakang antrian dan memperbarui back. Jika antrian kosong, elemen baru menjadi elemen pertama dan terakhir. Fungsi dequeueAntrian() menghapus elemen depan dan memperbarui front. Jika antrian kosong setelah penghapusan, back juga diatur ke nullptr.

Unguided 2

```cpp
#include <iostream> // Library standar yang digunakan untuk input dan output
#include <string>   // Library standar yang digunakan untuk tipe data string

using namespace std; // Untuk mempersingkat penulisan kode program

struct Node // Membuat struct Node Mahasiswa
{
    string nama;
    long long nim;
    Node *next;
};

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

public:
    Queue() // Membuat Queue untuk menginisialisasi front dan back
    {
        front = nullptr;
        back = nullptr;
    }

    bool isEmpty() // Membuat fungsi isEmpty untuk mengecek apakah antrian kosong atau tidak
    {
        return front == nullptr;
    }

    void enqueueAntrian(string nama, long long nim) // Membuat fungsi enqueueAntrian untuk menambahkan data ke dalam antrian
    {
        Node *newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;

        if (isEmpty())
        {
            front = newNode;
            back = newNode;
        }
        else
        {
            back->next = newNode;
```

```
            back = newNode;
        }
    }

    void dequeueAntrian() // Membuat fungsi dequeueAntrian untuk menghapus
data dari antrian
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
        {
            cout << "Pengunjung dengan nama " << front->nama << " telah
selesai dilayani" << endl;
            Node *temp = front;
            front = front->next;
            delete temp;

            if (front == nullptr)
            {
                back = nullptr;
            }
        }
    }

    int count() // Membuat fungsi count untuk menghitung jumlah data dalam
antrian
    {
        int count = 0;
        Node *current = front;
        while (current != nullptr)
        {
            count++;
            current = current->next;
        }
        return count;
    }

    void clear() // Membuat fungsi clear untuk menghapus semua data dalam
antrian
    {
        while (!isEmpty())
        {
            dequeueAntrian();
        }
    }
```

```
    void view() // Membuat fungsi view untuk menampilkan data dalam antrian
    {
        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
            return;
        }
        else
        {
            cout << "Data antrian mahasiswa:" << endl;
            Node *current = front;
            int position = 1;
            while (current != nullptr)
            {
                cout << position << ". " << current->nama << " (" << current-
>nim << ")" << endl;
                current = current->next;
                position++;
            }
        }
    }
};

int main()
{
    Queue queue;
    int pilihan;
    while (true)
    {
        cout << "========== Menu Antrian Mahasiswa ==========" << endl;
        cout << "1. Tambah antrian" << endl;
        cout << "2. Hapus antrian" << endl;
        cout << "3. Lihat antrian" << endl;
        cout << "4. Hapus semua antrian" << endl;
        cout << "5. Keluar" << endl;
        cout << "============================================" << endl;
        cout << "Masukkan pilihan: ";
        cin >> pilihan;
        switch (pilihan)
        {
        case 1:
        {
            string nama;
            long long nim;
            cout << "Masukkan nama mahasiswa: ";
            cin.ignore();
            getline(cin, nama);
            cout << "Masukkan NIM mahasiswa: ";
```

```
            cin >> nim;
            queue.enqueueAntrian(nama, nim);
            cout << "Mahasiswa " << nama << " telah ditambahkan ke dalam
antrian" << endl;
            break;
        }
        case 2:
        {
            queue.dequeueAntrian();
            break;
        }
        case 3:
        {
            queue.view();
            cout << "Jumlah antrian = " << queue.count() << endl;
            break;
        }
        case 4:
        {
            queue.clear();
            cout << "Semua antrian telah dilayani" << endl;
            break;
        }
        case 5:
        {
            cout << "Terima kasih telah menggunakan program ini" << endl;
            return 0;
        }
        default:
        {
            cout << "Pilihan tidak valid" << endl;
            break;
        }
        }
    }
}
```

Screenshot Program

```
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITME\Praktikum Modul 3\2311102037_Brian-Farrel-Evandhik
_IF-11-A_Praktikum-Struktur-Data> cd "c:\Users\MSI GAMING\Documents\GitHub\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-Struktur-Dat
\Modul 7\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFi { g++ tempCodeRunnerFile.cpp -o te
 { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
========== Menu Antrian Mahasiswa ==========
1. Tambah antrian
2. Hapus antrian
3. Lihat antrian
4. Hapus semua antrian
5. Keluar
=========================================
Masukkan pilihan: 1
Masukkan nama mahasiswa: Brian
Masukkan NIM mahasiswa: 2311102037
Mahasiswa Brian telah ditambahkan ke dalam antrian
========== Menu Antrian Mahasiswa ==========
1. Tambah antrian
2. Hapus antrian
3. Lihat antrian
4. Hapus semua antrian
5. Keluar
=========================================
Masukkan pilihan: 1
Masukkan nama mahasiswa: Farrel
Masukkan NIM mahasiswa: 231110203737
Mahasiswa Farrel telah ditambahkan ke dalam antrian
========== Menu Antrian Mahasiswa ==========
1. Tambah antrian
2. Hapus antrian
3. Lihat antrian
4. Hapus semua antrian
5. Keluar
=========================================
Masukkan pilihan: 1
Masukkan nama mahasiswa: Evandhika
Masukkan NIM mahasiswa: 231110200
Mahasiswa Evandhika telah ditambahkan ke dalam antrian
========== Menu Antrian Mahasiswa ==========
1. Tambah antrian
2. Hapus antrian
3. Lihat antrian
4. Hapus semua antrian
5. Keluar
=========================================
Masukkan pilihan: 3
Data antrian mahasiswa:
1. Brian (2311102037)
2. Farrel (231110203737)
3. Evandhika (231110200)
Jumlah antrian = 3
```

Deskripsi Program

Program ini mengimplementasikan antrian (queue) menggunakan linked list di C++ untuk menyimpan data mahasiswa. Struct Mahasiswa mendefinisikan elemen antrian dengan anggota nama, nim, dan pointer next. Pointer global front dan back melacak elemen pertama dan terakhir, diinisialisasi ke nullptr. Fungsi isEmpty() mengembalikan true jika front adalah nullptr. Fungsi enqueueAntrian(string nama, string nim) menambahkan node baru ke belakang antrian, memperbarui back, dan menjadikan node baru sebagai elemen pertama jika antrian kosong. Fungsi dequeueAntrian() menghapus elemen depan, memperbarui front, dan mengatur back ke nullptr jika antrian kosong setelah penghapusan

KESIMPULAN

Implementasi dengan Array Statis:

Antrian menggunakan array statis memiliki ukuran maksimal 5, dengan dua variabel (front dan back) untuk melacak posisi elemen. Fungsi isFull() dan isEmpty() digunakan untuk memeriksa status antrian. Fungsi enqueueAntrian() menambahkan elemen jika antrian tidak penuh, sedangkan dequeueAntrian() menghapus elemen depan dan menggeser elemen lainnya. Fungsi countQueue() mengembalikan jumlah elemen, clearQueue() mengosongkan antrian, dan viewQueue() menampilkan elemen. Operasi di main() melibatkan penambahan elemen ("Andi" dan "Maya"), menampilkan isi dan jumlah antrian, menghapus satu elemen, dan mengosongkan antrian serta menampilkan kondisi akhirnya.

Implementasi dengan Linked List:

Untuk antrian menggunakan linked list, struct Node atau Mahasiswa mendefinisikan elemen dengan data string (nama dan nim) dan pointer next. Dua pointer global, front dan back, melacak elemen pertama dan terakhir, diinisialisasi ke nullptr. Fungsi isEmpty() memeriksa apakah antrian kosong. Fungsi enqueueAntrian(string nama, string nim) menambahkan node baru ke belakang antrian dan memperbarui back. Jika antrian kosong, node baru menjadi elemen pertama dan terakhir. Fungsi dequeueAntrian() menghapus elemen depan dan memperbarui front. Jika antrian kosong setelah penghapusan, back diatur ke nullptr.

Dengan kombinasi ini, program mencakup implementasi antrian menggunakan dua pendekatan berbeda di C++: satu dengan array statis dan satu lagi dengan linked list, masing-masing dilengkapi dengan fungsi-fungsi untuk memanipulasi dan mengelola elemen-elemen dalam antrian.

Daftar Pustaka

[1] https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-danjenisnya/

[2] https://www.programiz.com/cpp-programming/queue

[3] https://elektro.um.ac.id/wp-content/uploads/2016/04/Struktur-Data-Modul-Praktikum-5-Queue.pdf