LAPORAN PRAKTIKUM

MODUL VII
QUEUE

Disusun oleh:
Ilhan Sahal Mansiz
2311102029

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

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

Modul 7 Kali ini membahas tentang Queue dalam C++ berfokus pada pemahaman dan implementasi struktur data queue, yang merupakan salah satu struktur data fundamental dalam ilmu komputer. Queue, atau antrian, adalah struktur data yang mengikuti prinsip First In, First Out (FIFO), artinya elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan. Struktur data ini sangat penting dalam berbagai aplikasi seperti manajemen antrian di sistem operasi, penjadwalan tugas, dan komunikasi data dalam jaringan. Dalam C++, queue dapat diimplementasikan menggunakan array, linked list, atau dengan memanfaatkan kelas template dari pustaka standar seperti std::queue dari header <queue>.

Operasi dasar yang dapat dilakukan pada queue mencakup enqueue (atau push), dequeue (atau pop), front, back, dan empty. Operasi enqueue menambahkan elemen ke akhir queue, sedangkan dequeue menghapus elemen dari depan queue. Operasi front mengakses elemen di depan queue tanpa menghapusnya, back mengakses elemen di belakang queue, dan empty memeriksa apakah queue kosong. Dengan menggunakan kelas std::queue, pengguna dapat dengan mudah mengelola antrian tanpa harus memikirkan detail implementasi yang rumit, karena pustaka standar C++ menyediakan semua fungsi dasar ini dengan antarmuka yang sederhana dan efisien.

Pemahaman tentang implementasi manual queue dalam C++ juga sangat penting. Salah satu metode umum adalah menggunakan array melingkar (circular array) untuk mengelola antrian, yang memungkinkan pemanfaatan ruang memori secara efisien dengan menghindari perpindahan elemen saat antrian diisi dan dikosongkan. Alternatif lainnya adalah menggunakan linked list, di mana setiap elemen dihubungkan oleh pointer, yang memungkinkan antrian berkembang secara dinamis sesuai kebutuhan tanpa batasan ukuran tetap. Modul ini tidak hanya memperkenalkan teori dasar tentang queue tetapi juga memberikan latihan praktis untuk mengimplementasikan dan memodifikasi queue sesuai kebutuhan spesifik, sehingga mahasiswa dapat mengatasi berbagai masalah pemrograman yang melibatkan antrian dengan lebih efektif dan efisien.

BAB III

GUIDED

1. Guided 1

Source code

```
#include<iostream>

using namespace std;

const int maksimalQueue = 5; //Maksimal antrian
int front = 0; //Pennanda 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(){//Antrian 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{// Antriannya 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 untuk 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(){
    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;
}
```

Screenshoot program

Deskripsi program

Program C++ ini adalah implementasi dasar dari struktur data queue yang berfungsi untuk mengelola antrian dengan kapasitas maksimum 5 elemen. Program ini memiliki beberapa fungsi penting yang mengatur operasi dasar queue, seperti menambahkan elemen ke antrian (enqueueAntrian), menghapus elemen dari antrian (dequeueAntrian), memeriksa apakah antrian penuh (isFull), memeriksa apakah antrian kosong (isEmpty), menghitung jumlah elemen dalam

antrian (countQueue), mengosongkan antrian (clearQueue), dan menampilkan elemen-elemen dalam antrian (viewQueue). Fungsi-fungsi ini saling bekerja sama untuk memastikan bahwa antrian dikelola dengan benar sesuai dengan prinsip First In, First Out (FIFO).

Dalam main(), program memulai dengan menambahkan dua nama ke antrian, yaitu "Andi" dan "Maya", kemudian menampilkan isi antrian beserta jumlah elemen yang ada. Setelah itu, program menghapus satu elemen dari antrian, menampilkan kembali isi antrian, dan mencetak jumlah elemen yang tersisa. Akhirnya, program mengosongkan antrian sepenuhnya dan menampilkan keadaan antrian setelah dikosongkan, serta jumlah elemen yang ada. Fungsi viewQueue menampilkan keadaan antrian dengan mencetak semua posisi dalam antrian, menunjukkan elemen yang ada atau menandainya sebagai kosong jika tidak ada elemen di posisi tersebut. Program ini memberikan gambaran praktis tentang bagaimana antrian dapat diimplementasikan dan dikelola dalam C++.

LATIHAN KELAS - UNGUIDED

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

Source code

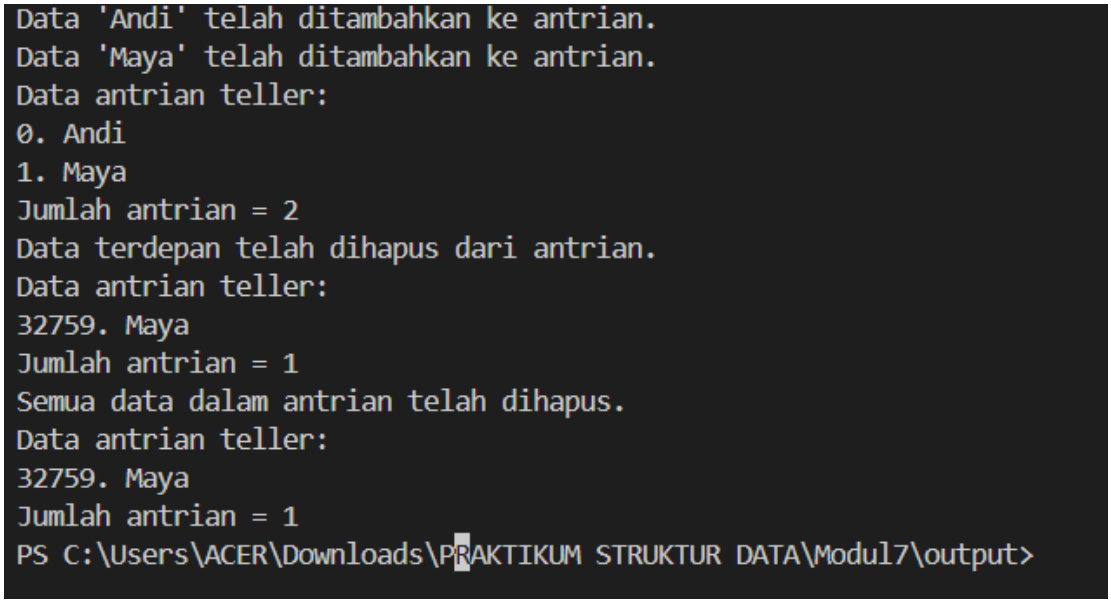
```
#include <iostream>
using namespace std;

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

Node *front = nullptr;
Node *back = nullptr;

bool isEmpty()
{
    return front == nullptr;
}

void enqueAntrian(string data)
{
    Node *newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;

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

```
    {
        back->next = newNode;
        back = newNode;
    }
    cout << "Data '" << data << "' telah ditambahkan ke antrian."
<< endl;
}

void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian Kosong" << endl;
    }
    else
    {
        Node *temp = front;
        front = front->next;
        if (front == nullptr)
        {
            back = nullptr;
        }
        delete temp;
        cout << "Data terdepan telah dihapus dari antrian." <<
endl;
    }
}

int countQueue()
{
    int count = 0;
    Node *temp = front;
    while (temp != nullptr)
    {
```

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

void clearQueue()
{
    while (isEmpty())
    {
        dequeueAntrian();
    }
    cout << "Semua data dalam antrian telah dihapus." << endl;
}

void viewqQueue()
{
    if (isEmpty())
    {
        cout << "Antrian Kosong" << endl;
    }
    else
    {
        Node *temp = front;
        cout << "Data antrian teller:" << endl;
        int i = i;
        while (temp != nullptr)
        {
            cout << i << ". " << temp->data << endl;
            temp = temp->next;
            i++;
        }
    }
}
```

```
int main ()
{
    enqueAntrian("Andi");
    enqueAntrian("Maya");
    viewqQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    dequeueAntrian();
    viewqQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    clearQueue();
    viewqQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    return 0;

} //2311102029 ILHAN SAHAL MANSIZ
```

Screenshot Program



Deskripsi Program

Program ini merupakan implementasi struktur data queue menggunakan linked list dalam bahasa C++. Queue adalah struktur data yang mengikuti prinsip First In, First Out (FIFO), yang berarti elemen yang pertama kali ditambahkan akan menjadi elemen pertama yang dihapus. Dalam program ini, struktur Node digunakan untuk menyimpan data dan pointer ke node berikutnya, memungkinkan antrian untuk tumbuh dan menyusut secara dinamis sesuai dengan elemen yang dimasukkan atau dihapus.

Fungsi-fungsi utama dalam program ini mencakup enqueAntrian untuk menambahkan elemen ke akhir queue, dequeueAntrian untuk menghapus elemen dari depan queue, isEmpty untuk memeriksa apakah queue kosong, countQueue untuk menghitung jumlah elemen dalam queue, clearQueue untuk menghapus semua elemen dalam queue, dan viewQueue untuk menampilkan elemen-elemen dalam queue. Saat menambahkan elemen baru, fungsi enqueAntrian membuat node baru dan menghubungkannya ke node belakang antrian. Jika queue kosong, node baru ini menjadi node pertama dan terakhir sekaligus. Fungsi dequeueAntrian menghapus node dari depan antrian dan menyesuaikan pointer front untuk menunjuk ke node berikutnya.

Program ini juga menyediakan fungsionalitas untuk mengosongkan seluruh queue dengan clearQueue, yang memanggil dequeueAntrian berulang kali sampai queue

kosong. Fungsi viewQueue menampilkan semua elemen dalam antrian, memungkinkan pengguna untuk melihat status terkini dari antrian. Dengan implementasi berbasis linked list, program ini tidak dibatasi oleh ukuran tetap seperti pada array, sehingga lebih fleksibel dan efisien dalam memori, terutama untuk aplikasi di mana jumlah elemen dalam antrian dapat sangat bervariasi.

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

Source Code

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

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

Mahasiswa *front = nullptr;
Mahasiswa *back = nullptr;

bool isEmpty()
{
    return front == nullptr;
}

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

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

void dequeueMahasiswa()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Mahasiswa *temp = front;
        front = front->next;
```

```
            if (front == nullptr)
            {
                back = nullptr;
            }
            delete temp;
        }
    }

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

    void clearQueue()
    {
        while (isEmpty())
        {
            dequeueMahasiswa();
        }
    }

    void viewQueue()
    {
        if (isEmpty())
        {
            cout << "Antrian Mahasiswa: kosong" << endl;
        }
        else
        {
            cout << "\nAntrian Mahasiswa:" << endl;
            Mahasiswa *temp = front;
            while (temp != nullptr)
            {
                cout << "Nama: " << temp->nama << ", NIM: " <<
    temp->nim << endl;
                temp = temp->next;
            }
        }
    }

    int main()
    {
        enqueueMahasiswa("Ilhan Sahal Mansiz", "2311102029");
        enqueueMahasiswa("Asfi", "654321");
        viewQueue();
```

```
        cout << "Jumlah Mahasiswa dalam antrian: " <<
    countQueue() << endl;

        dequeueMahasiswa();
        viewQueue();
        cout << "Jumlah Mahasiswa dalam antrian: " <<
    countQueue() << endl;

        clearQueue();
        viewQueue();
        cout << "\nJumlah Mahasiswa dalam antrian: " <<
    countQueue() << endl;

        return 0;
} // 2311102029 ILHAN SAHAL MANSIZ
```

Screenshot Program

```
Antrian Mahasiswa:
Nama: Ilhan Sahal Mansiz, NIM: 2311102029
Nama: Asfi, NIM: 654321
Jumlah Mahasiswa dalam antrian: 2

Antrian Mahasiswa:
Nama: Asfi, NIM: 654321
Jumlah Mahasiswa dalam antrian: 1

Antrian Mahasiswa:
Nama: Asfi, NIM: 654321

Jumlah Mahasiswa dalam antrian: 1
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul7\output>
```

Deskripsi Program

Program ini adalah implementasi dari struktur data queue menggunakan linked list khusus untuk mengelola antrian mahasiswa. Dalam program ini, struktur Mahasiswa didefinisikan untuk menyimpan informasi tentang setiap mahasiswa, yaitu nama dan NIM (Nomor Induk Mahasiswa), serta pointer next yang menunjuk ke node mahasiswa berikutnya dalam antrian. Pointer front dan back digunakan untuk melacak

awal dan akhir antrian, masing-masing diinisialisasi ke nullptr untuk menunjukkan bahwa antrian pada awalnya kosong.

Fungsi-fungsi utama dalam program ini mencakup enqueueMahasiswa untuk menambahkan mahasiswa baru ke akhir antrian, dequeueMahasiswa untuk menghapus mahasiswa dari depan antrian, isEmpty untuk memeriksa apakah antrian kosong, countQueue untuk menghitung jumlah mahasiswa dalam antrian, clearQueue untuk menghapus semua mahasiswa dalam antrian, dan viewQueue untuk menampilkan daftar mahasiswa yang sedang antri. Fungsi enqueueMahasiswa membuat node baru untuk setiap mahasiswa dan menambahkannya ke akhir antrian, sementara dequeueMahasiswa menghapus node dari depan antrian dan membebaskan memori yang digunakan oleh node tersebut.

Dalam fungsi main(), program mendemonstrasikan penggunaan fungsi-fungsi tersebut dengan menambahkan dua mahasiswa ke antrian, menampilkan daftar mahasiswa dalam antrian, dan mencetak jumlah mahasiswa yang sedang antri. Kemudian, satu mahasiswa dihapus dari antrian, dan daftar antrian serta jumlah mahasiswa diperbarui. Akhirnya, seluruh antrian dihapus menggunakan fungsi clearQueue, dan keadaan akhir dari antrian ditampilkan. Program ini mengilustrasikan bagaimana antrian dinamis dapat diimplementasikan dan dikelola secara efisien menggunakan linked list dalam konteks data mahasiswa.

BAB IV

KESIMPULAN

Modul 7 tentang queue dalam C++ memberikan pemahaman mendalam tentang salah satu struktur data fundamental yang mengikuti prinsip First In, First Out (FIFO). Dengan menggunakan queue, kita dapat mengelola data secara efisien di berbagai aplikasi seperti manajemen antrian, penjadwalan tugas, dan komunikasi data. Implementasi queue dapat dilakukan menggunakan array atau linked list, dengan masing-masing metode memiliki kelebihan dan kekurangannya. Implementasi menggunakan array cenderung lebih sederhana tetapi terbatas oleh ukuran tetap, sementara linked list menawarkan fleksibilitas dinamis dalam ukuran tetapi memerlukan manajemen memori yang lebih kompleks.

Melalui modul ini, mahasiswa tidak hanya mempelajari teori dasar queue tetapi juga mempraktikkan implementasi konkret menggunakan C++. Mereka belajar membuat fungsi-fungsi dasar seperti enqueue, dequeue, isEmpty, countQueue, clearQueue, dan viewQueue, baik menggunakan array maupun linked list. Dengan pemahaman ini, mahasiswa siap untuk menerapkan konsep queue dalam berbagai konteks pemrograman nyata, meningkatkan efisiensi dan efektivitas pengelolaan data dalam aplikasi yang mereka kembangkan.

BAB V

REFERENSI

- https://kotakode.com/pertanyaan/8724/queue-dalam-c%2B%2B
- https://www.kaskus.co.id/thread/5ec54d20facb95558a5496e1/pengertian-queue-dalam-c
- https://www.programiz.com/cpp-programming/queue