LAPORAN PRAKTIKUM
MODUL VII
QUEUE

Disusun oleh:

Raihan Ramadhan

NIM: 2311102040

Dosen Pengampu:

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

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO

2024

BAB I

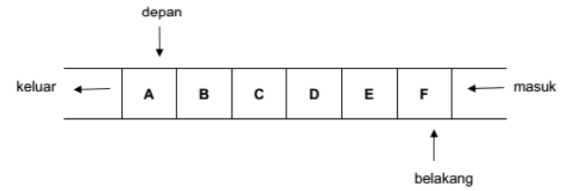
TUJUAN PRAKTIKUM

Praktikum ini bertujuan untuk memastikan bahwa mahasiswa memiliki pemahaman yang baik tentang apa itu Queue, bagaimana cara kerjanya, dan mengapa penting dalam pemrograman. Mahasiswa diharapkan dapat menjelaskan dengan jelas apa itu Queue dan bagaimana ia bekerja sebagai bagian dari struktur data. Praktikum ini memberikan kesempatan kepada mahasiswa untuk mengimplementasikan Queue dalam bahasa pemrograman c++. Melalui latihan konkret, mahasiswa dapat belajar bagaimana mendefinisikan dan menggunakan Queue dalam konteks nyata. Mahasiswa mungkin akan diberikan situasi atau masalah yang memerlukan penggunaan Queue, dan mereka akan belajar bagaimana cara menguji keefektifan solusi mereka menggunakan Queue.

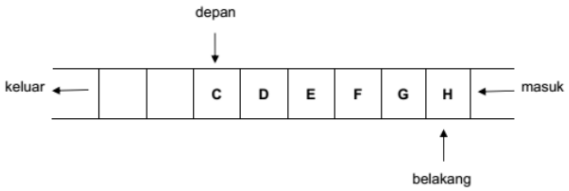
BAB II

DASAR TEORI

Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisi yang lain. Queue bersifat FIFO (First In First Out). Antrian (Queue) merupakan suatu kumpulan data yang penambahan elemennya (masuk antrian) hanya bisa dilakukan pada suatu ujung (disebut dengan sisi belakang/rear) atau disebut juga enqueue yaitu apabila seseorang masuk ke dalam sebuah antrian dan keluar dari antrian adalah dequeue.



Gambar antrian penuh dengan 6 elemen



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 1 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

a. enqueue() : menambahkan data ke dalam queue.
b. dequeue() : mengeluarkan data dari queue.
c. peek() : mengambil data dari queue tanpa menghapusnya.
d. isEmpty() : mengecek apakah queue kosong atau tidak.
e. isFull() : mengecek apakah queue penuh atau tidak.
f. 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;
}
```

Screenshot Output

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 7>
 guided1_queue.cpp -o guided1_queue } ; if ($?) { .\
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:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 7>
```

Deskripsi Program

Program ini menerapkan antrian menggunakan array dengan kapasitas maksimal lima elemen. Fungsi enqueueAntrian berfungsi untuk menambahkan elemen ke dalam antrian, sedangkan fungsi dequeueAntrian digunakan untuk mengeluarkan elemen dari antrian. Program ini juga memiliki fungsi isFul` dan isEmpty untuk memeriksa apakah antrian sudah penuh atau masih kosong. Untuk menampilkan elemen-elemen yang ada di dalam antrian, digunakan fungsi viewQueue, sementara fungsi clearQueue digunakan untuk menghapus semua elemen di dalam antrian. Dalam fungsi main, program melakukan beberapa operasi seperti menambahkan elemen ke dalam antrian, menampilkan isi antrian, menghitung jumlah elemen dalam antrian, menghapus elemen, dan akhirnya mengosongkan antrian, serta menampilkan isi antrian setelah setiap operasi tersebut.

BAB IV

UNGUIDED

1. **Unguided 1**
   Source Code

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

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

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

public:
    Queue()
    {
        front = nullptr;
        back = nullptr;
    }

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

    void enqueue(string data)
    {
        Node *temp = new Node();
        temp->data = data;
        temp->next = nullptr;

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

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

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

void clearQueue()
{
    while (!isEmpty())
    {
        dequeue();
    }
}
```

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

int main()
{
    Queue q;
    int choice;
    string name;

    do
    {
        cout << "\nMenu:\n";
        cout << "1. Tambah ke antrian \n";
        cout << "2. Hapus dari antrian \n";
        cout << "3. Lihat antrian\n";
        cout << "4. Hitung antrian\n";
        cout << "5. Bersihkan antrian\n";
        cout << "0. Keluar\n";
        cout << "Pilih: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            cout << "Masukkan nama: ";
            cin >> name;
            q.enqueue(name);
            break;
        case 2:
            q.dequeue();
            break;
```

```
        case 3:
            q.viewQueue();
            break;
        case 4:
            cout << "Jumlah antrian = " << q.countQueue() << endl;
            break;
        case 5:
            q.clearQueue();
            break;
        case 0:
            cout << "Keluar dari program." << endl;
            break;
        default:
            cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
        }
    } while (choice != 0);

    return 0;
}
```

Screenshot Output

```
Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 1
Masukkan nama: Raihan

Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 1
Masukkan nama: Soib

Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 1
Masukkan nama: Dirwan
```

```
Pilih: 3
Data antrian teller:
1. Raihan
2. Soib
3. Dirwan
```

```
Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 2

Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 3
Data antrian teller:
1. Soib
2. Dirwan
```

```
Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 4
Jumlah antrian = 2
```

```
Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 5

Menu:
1. Tambah ke antrian
2. Hapus dari antrian
3. Lihat antrian
4. Hitung antrian
5. Bersihkan antrian
0. Keluar
Pilih: 3
Data antrian teller:
(kosong)
```

Deskripsi Program

Kode tersebut mendefinisikan struktur antrian menggunakan linked list, dengan metode untuk menambah (enqueue), menghapus (dequeue), menghitung elemen, melihat isi, dan membersihkan antrian. Kelas Queue mengelola dua pointer, front dan back, untuk melacak elemen pertama dan terakhir dalam antrian. Fungsi `isEmpty` digunakan untuk memeriksa apakah antrian kosong, sementara enqueue menambahkan elemen baru di belakang antrian, dan dequeue menghapus elemen dari depan antrian. Fungsi viewQueue`menampilkan semua elemen dalam antrian, dan countQueue menghitung jumlah elemen dalam antrian.

2. **Unguided 2**
   Source Code

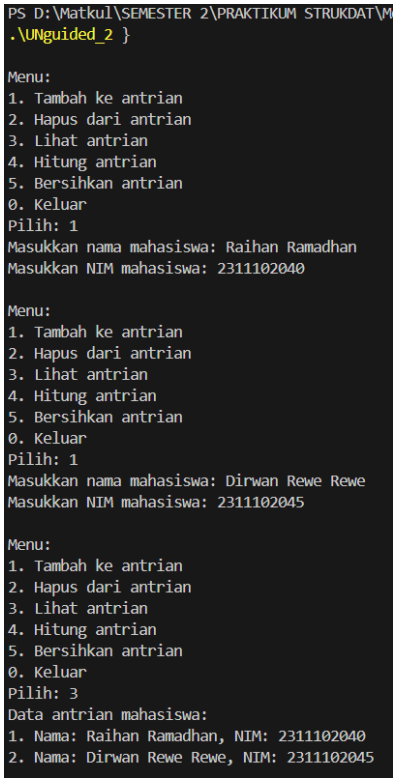
```cpp
#include <iostream>
using namespace std;

struct Node
{
    string nama;
    string NIM;
    Node *next;
};

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

public:
    Queue()
    {
        front = nullptr;
        back = nullptr;
    }

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

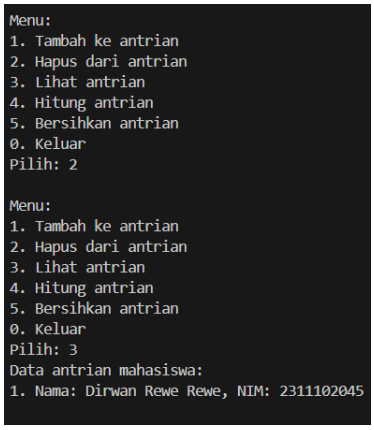
```
void enqueue(string nama, string NIM)
{
    Node *temp = new Node();
    temp->nama = nama;
    temp->NIM = NIM;
    temp->next = nullptr;

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

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

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

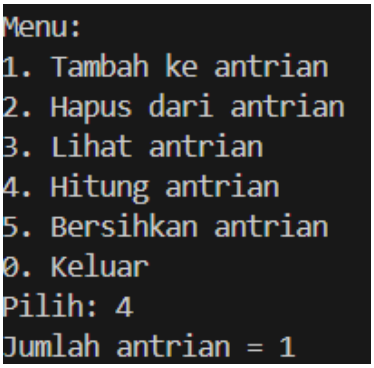
```
    void clearQueue()
    {
        while (!isEmpty())
        {
            dequeue();
        }
    }

    void viewQueue()
    {
        cout << "Data antrian mahasiswa:" << endl;
        Node *current = front;
        int index = 1;
        while (current != nullptr)
        {
            cout << index << ". Nama: " << current->nama << ", NIM: " <<
current->NIM << endl;
            current = current->next;
            index++;
        }
        if (isEmpty())
        {
            cout << "(kosong)" << endl;
        }
    }
};

int main()
{
    Queue q;
    int choice;
    string nama, NIM;

    do
    {
        cout << "\nMenu:\n";
        cout << "1. Tambah ke antrian \n";
        cout << "2. Hapus dari antrian \n";
        cout << "3. Lihat antrian\n";
        cout << "4. Hitung antrian\n";
        cout << "5. Bersihkan antrian\n";
        cout << "0. Keluar\n";
        cout << "Pilih: ";
        cin >> choice;
```

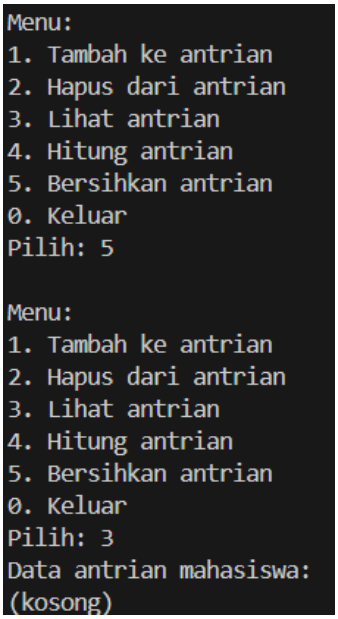
```
        switch (choice)
        {
        case 1:
            cout << "Masukkan nama mahasiswa: ";
            cin.ignore();
            getline(cin, nama);
            cout << "Masukkan NIM mahasiswa: ";
            cin >> NIM;
            q.enqueue(nama, NIM);
            break;
        case 2:
            q.dequeue();
            break;
        case 3:
            q.viewQueue();
            break;
        case 4:
            cout << "Jumlah antrian = " << q.countQueue() << endl;
            break;
        case 5:
            q.clearQueue();
            break;
        case 0:
            cout << "Keluar dari program." << endl;
            break;
        default:
            cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
        }
    } while (choice != 0);

    return 0;
}
```

ScreenShot Output









Deskripsi Program

Program tersebut adalah implementasi antrian (queue) menggunakan struktur data linked list dalam C++. Kode mendefinisikan sebuah struct Node untuk merepresentasikan setiap elemen dalam antrian, yang berisi nama dan NIM mahasiswa serta pointer ke elemen berikutnya. Kelas Queue memiliki metode untuk menambahkan elemen ke antrian (enqueue), menghapus elemen dari antrian (dequeue), menghitung jumlah elemen dalam antrian (countQueue), membersihkan seluruh elemen dalam antrian (clearQueue), dan menampilkan isi antrian (viewQueue). Program utama menyediakan menu interaktif bagi pengguna untuk melakukan operasi-operasi ini, menggunakan perulangan do-while sampai pengguna memilih untuk keluar. Jika antrian kosong, metode dequeue dan viewQueue akan mengeluarkan pesan yang sesuai.

BAB IV
KESIMPULAN

Program pertama menggunakan array statis untuk mengelola antrian dengan fungsi-fungsi dasar seperti pemeriksaan antrian penuh atau kosong, penambahan, penghapusan, perhitungan jumlah elemen, pengosongan, dan penampilan isi antrian. Pendekatan ini sederhana namun terbatas oleh ukuran tetap dari array.

Program kedua menggunakan linked list, di mana setiap elemen direpresentasikan sebagai sebuah node yang menyimpan data string. Kelas Queue pada program ini menawarkan fleksibilitas lebih karena ukuran antrian dapat bertambah secara dinamis. Metode yang disediakan mencakup pengecekan antrian kosong, penambahan dan penghapusan elemen, perhitungan jumlah elemen, pengosongan antrian, dan penampilan isi antrian.

Program ketiga juga menggunakan linked list, namun setiap elemen antrian direpresentasikan sebagai objek Student yang menyimpan nama dan NIM mahasiswa. Meskipun menggunakan pendekatan serupa dengan program kedua, program ini lebih kompleks karena data yang dielola lebih kaya. Pendekatan ini menunjukkan fleksibilitas linked list dalam mengelola data yang lebih kompleks secara dinamis.

Secara keseluruhan, implementasi menggunakan linked list lebih disarankan untuk aplikasi yang memerlukan antrian dengan ukuran dinamis atau data yang bervariasi. Sementara itu, array statis cocok untuk kasus yang lebih sederhana dengan ukuran tetap.

DAFTAR PUSTAKA

Asisten Praktikum “MODUL VII QUEUE”. Learning Management System,27 Mei 2024.

Muliono Rizki,” Pemrograman C++ ALGORITMA & STRUKTUR DATA.Chrome,26 Mei 2024.

https://abdulkadir.blog.uma.ac.id/wp-content/uploads/sites/365/2019/02/Modul-Praktikum-Algoritma-Struktur-Data-Merge.pdf.