

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :
DIO GILBRAN PRAMANA
NIM : 2311102062

Dosen
WAHYU ANDI SAPUTRA, S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. 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.

B. GUIDED

GUIDED 1

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

const int maksimalQueue = 5;
int front = 0;
int back = 0;
string queueTeller[maksimalQueue];

bool isFull() {
    return back == maksimalQueue;
}

bool isEmpty(){
    return back == 0;
}

void enqueueAntrian(string data){
    if (isFull()){
        cout << "Antrian penuh" << endl;
    } else {
        queueTeller[back] = data;
        back++;
    }
}

void dequeueAntrian(){
    if (isEmpty()){
        cout << "Antrian kosong" << endl;
    } else {
        for (int i=0; i<back - 1; i++){
            queueTeller[i] = queueTeller[i+1];
        }
        queueTeller[back - 1] = "";
        back--;
    }
}

int countQueue() {
    return back;
}

void clearQueue() {
    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;

    return 0;
}

```

SCREENSHOT OUTPUT

```

PS E:\Praktikum Struktur Data> g++ 'c:\Users\Did...
cher.exe' '-std=Microsoft-MIEngine-In-cc5zxu...
.dec' '-pid=Microsoft-MIEngine-Pid-z3iypq0.u6
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 E:\Praktikum Struktur Data>

```

DESKRIPSI

Program ini memberikan pengertian tentang penggunaan antrian dengan array dalam konteks pemrograman C++. Penggunaannya mencakup operasi dasar antrian seperti enqueue dan dequeue, serta operasi tambahan seperti memeriksa kekosongan atau kepenuhan antrian, menghitung jumlah elemen dalam antrian, membersihkan antrian, dan mencetak isi antrian. Implementasi ini dapat diadaptasi dan dimodifikasi sesuai kebutuhan aplikasi yang memerlukan manajemen data berdasarkan prinsip First In First Out.

C. UNGUIDED

UNGUIDED 1

```
#include <iostream>
using namespace std;
struct Node
{
    string data;
    Node *next;
};
Node *front = nullptr;
Node *back = nullptr;

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

void enqueueAntrian(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 << " berhasil ditambahkan ke antrian."
    << endl;
}

void dequeueAntrian()
{

```

```

    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Node *temp = front;
        front = front->next;
        if (front == nullptr)
        {
            back = nullptr;
        }
        cout << "Data " << temp->data << " berhasil dihapus dari antrian." << endl;
        delete temp;
    }
}

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

void clearQueue()
{
    while (!isEmpty())
    {
        dequeueAntrian();
    }
    cout << "Antrian berhasil dikosongkan." << endl;
}

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++;
    }
    if (isEmpty())
    {
        cout << "(kosong)" << endl;
    }
}

void tampilkanMenu()
{
    cout << "~~ Pengelola Antrian ~~" << endl;
    cout << "1. Menambah Antrian" << endl;
    cout << "2. Menghapus Antrian" << endl;
    cout << "3. Melihat Antrian" << endl;
    cout << "4. Menghitung Jumlah Antrian" << endl;
    cout << "5. Mengosongkan Antrian" << endl;
    cout << "6. Keluar" << endl;
    cout << "Pilih opsi: ";
}

int main()
{
    int choice;
    string data;
    do
    {
        tampilkanMenu();
        cin >> choice;
        switch (choice)
        {
            case 1:
                cout << "Masukkan nama: ";
                cin >> data;
                enqueueAntrian(data);
                break;
            case 2:
                dequeueAntrian();
                break;
            case 3:
                viewQueue();
                break;
            case 4:
                cout << "Jumlah antrian = " << countQueue() << endl;
                break;
            case 5:
                clearQueue();
                break;
            case 6:

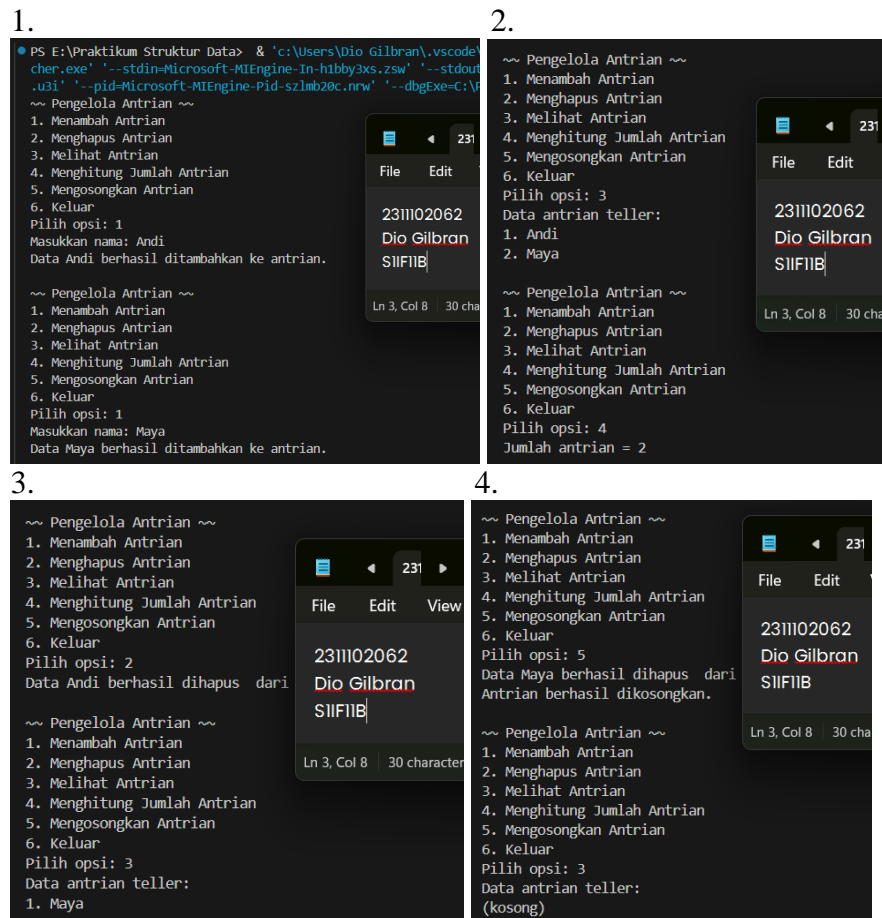
```

```

        cout << "Berhasil keluar dari program." << endl;
        break;
    default:
        cout << "Pilihan salah" << endl;
    }
    cout << endl;
} while (choice != 6);
return 0;
}

```

SCREENSHOT OUTPUT



DESKRIPSI

Program ini mengelola antrian menggunakan struktur data linked list dengan operasi dasar queue seperti enqueue (menambah elemen ke antrian), dequeue (menghapus elemen dari antrian), menghitung jumlah elemen dalam antrian, dan mengosongkan antrian. Program ini memiliki fungsi untuk memeriksa apakah antrian kosong, menambah elemen baru ke antrian, menghapus elemen dari antrian, menghitung jumlah elemen dalam antrian, mengosongkan antrian, dan menampilkan semua elemen dalam antrian. Selain itu, terdapat menu interaktif yang memungkinkan pengguna untuk memilih operasi yang ingin dijalankan.

UNGUIDED 2

```
#include <iostream>
using namespace std;
struct Node
{
    string nama, nim;
    Node *next;
};
Node *front = nullptr;
Node *back = nullptr;

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

void enqueueAntrian(string nama, string nim)
{
    if (nim.length() != 10)
    {
        cout << "NIM harus 10 digit." << endl;
        return;
    }
    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 << "Data mahasiswa" << nama << "dengan NIM " << nim << "
berhasil ditambahkan ke antrian." << endl;
}

void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
```

```

    {
        Node *temp = front;
        front = front->next;
        if (front == nullptr)
        {
            back = nullptr;
        }
        cout << "Data mahasiswa" << temp->nama << "dengan NIM " <<
temp->nim << " berhasil dihapus dari antrian." << endl;
        delete temp;
    }
}

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

void clearQueue()
{
    while (!isEmpty())
    {
        dequeueAntrian();
    }
    cout << "Antrian berhasil dikosongkan." << endl;
}

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

```

```

    {
        cout << "(kosong)" << endl;
    }
}
void tampilkanMenu()
{
    cout << "~~ Pengelola Antrian ~~" << endl;
    cout << "1. Menambah Antrian" << endl;
    cout << "2. Menghapus Antrian" << endl;
    cout << "3. Melihat Antrian" << endl;
    cout << "4. Menghitung Jumlah Antrian" << endl;
    cout << "5. Mengosongkan Antrian" << endl;
    cout << "6. Keluar" << endl;
    cout << "Pilih opsi: ";
}
int main()
{
    int choice;
    string nama, nim;

    do
    {
        tampilkanMenu();
        cin >> choice;
        switch (choice)
        {
            case 1:
                cout << "Masukkan nama: ";
                cin.ignore();
                getline(cin, nama);
                cout << "Masukkan NIM dengan karakter 10 digit: ";
                cin >> nim;
                enqueueAntrian(nama, nim);
                break;
            case 2:
                dequeueAntrian();
                break;
            case 3:
                viewQueue();
                break;
            case 4:
                cout << "Jumlah antrian mahasiswa: " << countQueue()
<< endl;
                break;
            case 5:
                clearQueue();
                break;

```

```

        case 6:
            cout << "Berhasil keluar dari program." << endl;
            break;
        default:
            cout << "Pilihan salah" << endl;
        }
        cout << endl;
    } while (choice != 6);
    return 0;
}

```

SCREENSHOT OUTPUT

1.

```

PS E:\Praktikum Struktur Data> & 'c:\Users\Dio Gilbran.exe' '--stdin=Microsoft-MIEngine-In-epuk12c3.a1
.1u0' '--pid=Microsoft-MIEngine-Pid-0rft2umg.lxf' '-
~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 1
Masukkan nama mahasiswa: dio
Masukkan NIM dengan karakter 10 digit: 2311102062
Data mahasiswa dio dengan NIM 2311102062 berhasil ditambahkan ke antrian.

~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 1
Masukkan nama mahasiswa: tya
Masukkan NIM dengan karakter 10 digit: 2311102063
Data mahasiswa tya dengan NIM 2311102063 berhasil ditambahkan ke antrian.

```

2.

```

~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 3
Data antrian mahasiswa:
1. Nama mahasiswa: dio dengan NIM: 2311102062
2. Nama mahasiswa: tya dengan NIM: 2311102063

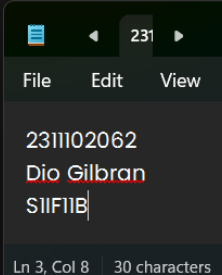
~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 4
Jumlah antrian mahasiswa: 2

```

3.

```
~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 2
Data mahasiswa dio dengan NIM 2311102062 berhasil dihapus dari antrian.

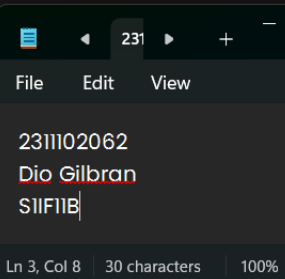
~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 3
Data antrian mahasiswa:
1. Nama mahasiswa: tyu dengan NIM: 2311102063
```



4.

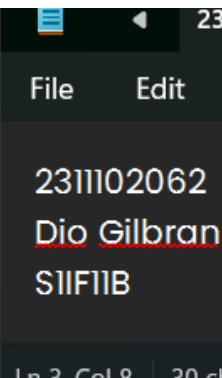
```
~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 5
Data mahasiswa tyu dengan NIM 2311102063 berhasil dihapus dari antrian.
Antrian berhasil dikosongkan.

~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 3
Data antrian mahasiswa:
(kosong)
```



5.

```
~~ Pengelola Antrian ~~
1. Menambah Antrian
2. Menghapus Antrian
3. Melihat Antrian
4. Menghitung Jumlah Antrian
5. Mengosongkan Antrian
6. Keluar
Pilih opsi: 6
Berhasil keluar dari program.
```



DESKRIPSI

Program ini mengelola antrian mahasiswa menggunakan linked list, di mana setiap elemen berisi nama dan NIM mahasiswa. Program menyediakan operasi dasar queue seperti enqueue (menambah elemen ke antrian), dequeue (menghapus elemen dari antrian), menghitung jumlah elemen dalam antrian,

mengosongkan antrian, dan menampilkan semua elemen dalam antrian. Pengguna dapat memasukkan nama dan NIM mahasiswa, dengan validasi bahwa NIM harus berjumlah 10 digit. Menu interaktif memungkinkan pengguna memilih berbagai operasi yang ingin dijalankan, memberikan kemudahan dalam pengelolaan data antrian mahasiswa.

D. KESIMPULAN

Kesimpulan dari praktikum ini adalah bahwa implementasi struktur data queue dapat dilakukan dengan dua pendekatan, yaitu menggunakan array dan linked list. Pada praktikum kali ini, saya mempelajari bagaimana melakukan operasi dasar queue seperti enqueue, dequeue, memeriksa kekosongan atau kepenuhan antrian, menghitung jumlah elemen dalam antrian, dan mengosongkan antrian. Praktikum ini memberikan pemahaman tentang prinsip First-In-First-Out (FIFO) dalam pengelolaan data, serta aplikasi dari struktur data queue dalam program yang interaktif dan efisien.

E. REFERENSI

Asisten Praktikum, "Modul Queue", 2024.

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

Budi Raharjo. 2015. Pemrograman C++. Bandung: Informatika.