

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN
ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :

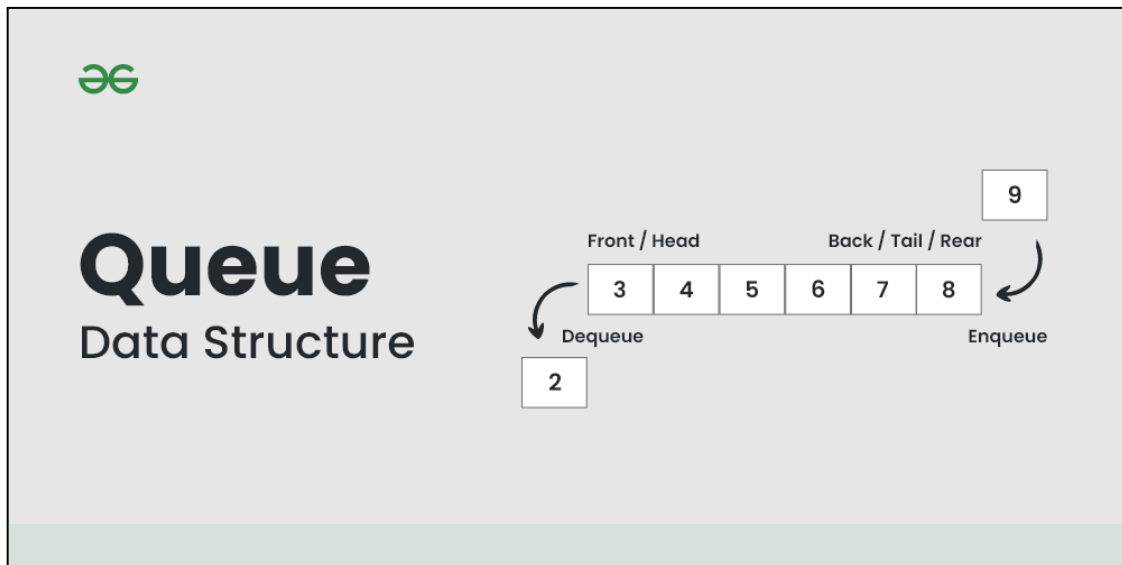
Muhammad Rusdiyanto Asatman
2311102053

Dosen :

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

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

A. Dasar Teori



Gambar 1.0 Ilustrasi Queue

[Sumber : GeeksForGeeks, <https://www.geeksforgeeks.org/queue-data-structure/>]

Queue atau antrian adalah struktur data yang menyusun elemen - elemen data dalam urutan linier. Prinsip dasar dari struktur data ini adalah “First In, First Out” (FIFO) yang berarti elemen data yang pertama dimasukkan ke dalam antrian akan menjadi yang pertama pula untuk dikeluarkan.

Dalam analogi sehari-hari, queue bisa dibayangkan seperti jejeran orang yang sedang menunggu antrian di supermarket. Orang pertama yang datang adalah yang pertama dilayani (First In, First Out). Pada struktur data ini, urutan pertama (data yang akan dikeluarkan) disebut Front atau Head. Sebaliknya, data pada urutan terakhir (data yang baru saja ditambahkan) disebut Back, Rear, atau Tail. Proses untuk menambahkan data pada antrian disebut dengan Enqueue, sedangkan proses untuk menghapus data dari antrian disebut dengan Dequeue. Data yang ditambahkan akan berada di posisi paling akhir/belakang urutan data, sementara data yang akan dihapus adalah data paling depan dari antrian.

Beberapa contoh fungsi dari queue meliputi :

- Antrian: Digunakan untuk menyimpan data antrian, seperti sistem antrian pelayanan konsumen atau pembayaran.
- Web Crawler: Untuk menyimpan URL yang hendak dibuka pada proses crawling.

B. Guided

Guided 1

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

const int maksimalQueue = 5; // Batas maksimal antrian
int front = 0;                // Indeks awal antrian
int back = 0;                 // Indeks akhir antrian
string queueTeller[maksimalQueue]; // array untuk menyimpan elemen antrian

// Fungsi untuk melihat apakah antrian penuh
bool isFull()
{
    return back == maksimalQueue;
}

// Fungsi untuk melihat apakah antrian kosong atau tidak
bool isEmpty()
{
    return back == 0;
}

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string data)
{
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        queueTeller[back] = data;
        back++;
    }
}

// Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian()
{
    if (isEmpty())
```

```

    {
        cout << "Antrian kosong" << endl;
    }
else
{
    for (int i = 0; i < back; i++)
    {
        queueTeller[i] = queueTeller[i + 1];
    }
    queueTeller[back - 1] = ""; // Membersihkan data terakhir
    back--;
}
}

// Fungsi untuk menghitung banyak elemen dalam antrian
int countQueue()
{
    return back;
}

// Fungsi untuk menghapus semua elemen dalam antrian
void clearQueue()
{
    for (int i = 0; i < back; i++)
    {
        queueTeller[i] = "";
    }
    back = 0;
    front = 0;
}

// Fungsi untuk melihat antrian semua elemen dalam antrian
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;
    }
}

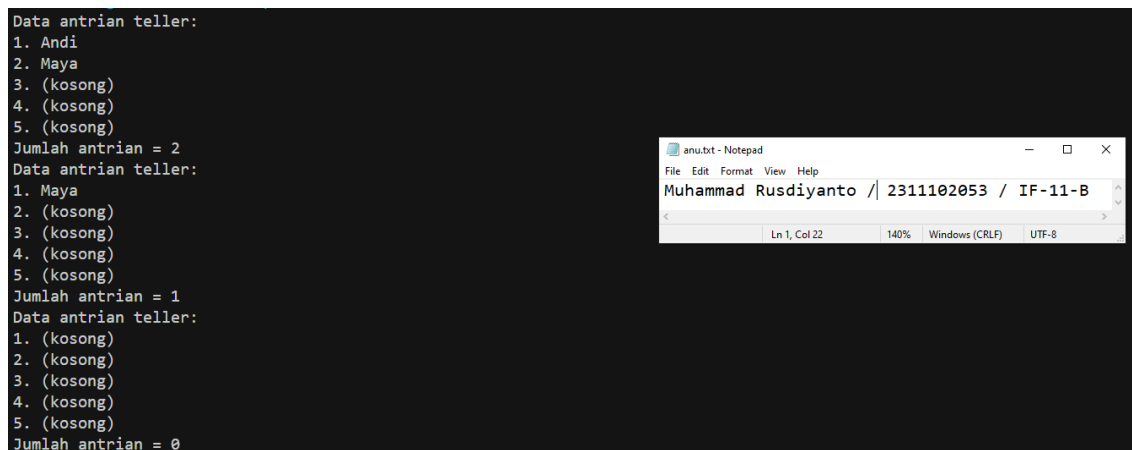
int main()
{
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}

```

Screenshots Output



The screenshot shows the output of the program in a terminal window. The output displays the state of a queue at different stages: initial state with 'Andi' and 'Maya', after dequeuing 'Andi', after clearing the queue, and after re-enqueuing 'Andi' and 'Maya'. A Notepad window titled 'anu.txt - Notepad' is also visible, showing the name 'Muhammad Rusdiyanto' and ID '2311102053 / IF-11-B'.

```

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

```

Deskripsi:

Program di atas adalah program yang mendemonstrasikan penggunaan dari Queue. Dalam program ini data dalam queue disimpan di dalam array yang memiliki size/kapasitas maksimum sebesar 5 elemen. Di dalam program ini juga terdapat beberapa fungsi yang digunakan untuk mengatur data dalam Queue, seperti :

1. isFull : Fungsi yang digunakan untuk mengecek apakah queue sudah penuh atau

belum. Hal ini dilakukan dengan cara membandingkan nilai back yang merepresentasikan jumlah elemen saat ini dengan nilai maksimal queue. Jika kedua nilai sama, dalam kata lain queue sudah penuh, maka akan dikembalikan nilai true, sedangkan jika tidak, maka akan dikembalikan nilai false.

2. isEmpty : Fungsi untuk mengecek apakah queue kosong atau tidak. Proses pengecekan dilakukan dengan membandingkan nilai back dengan 0. Jika nilai back sama dengan 0 atau dalam artian lain kosong, maka akan dikembalikan nilai true. Jika tidak, maka akan dikembalikan nilai false.
3. enqueueAntrian : Fungsi ini adalah fungsi yang digunakan untuk menambahkan elemen dalam antrian. Proses penambahan dimulai dengan mengecek apakah queue sudah penuh atau belum. Jika sudah penuh, maka program akan menampilkan pesan dan proses penambahan dibatalkan. Sementara jika tidak penuh, elemen akan dimasukkan ke indeks/posisi paling belakang/akhir antrian. Lalu, nilai back akan ditambah 1.
4. dequeueAntrian : Merupakan fungsi yang digunakan untuk menghapus elemen paling depan/awal dari antrian. Sebelum elemen dihapus, program akan mengecek apakah queue masih kosong atau belum. Jika kosong, maka program akan menampilkan pesan dan proses penghapusan dibatalkan. Jika tidak, maka program akan menghapus data paling awal dari queue, dilanjutkan dengan melakukan looping untuk mengatur ulang urutan data queue (data indeks 1 dipindah ke indeks 0, data indeks 2 dipindah ke indeks 1, dst.). Lalu, nilai back akan dikurangi 1.
5. countQueue : Berfungsi untuk mengembalikan jumlah elemen saat ini dalam antrian/queue. Fungsi ini hanya mengembalikan nilai dari back tanpa melakukan proses apapun terhadap nilai tersebut.
6. clearQueue : Digunakan untuk menghapus semua data dalam queue. Proses ini dilakukan dengan cara looping ke setiap data dalam antrian, lalu mengubah nilainya menjadi string kosong. Setelah dikosongkan, maka program akan merubah nilai front dan back menjadi 0, sebagai tanda bahwa queue/antrian kosong.
7. viewQueue : Fungsi ini adalah fungsi yang digunakan untuk menampilkan semua data/nilai dalam antrian. Penampilan output data dimulai dari data yang pertama kali dimasukkan, alias dari depan queue, lalu looping sampai indeks terakhir dari queue. Program akan menampilkan output dengan ketentuan jika

data tidak kosong, maka nilai dari data akan ditampilkan. Jika data kosong, maka program akan memberikan keterangan bahwa data pada indeks/posisi tersebut masih kosong.

C. Unguided

Unguided 1

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

const int maksimalQueue = 5; // Batas maksimal antrian
int front = 0;                // Indeks awal antrian
int back = 0;                 // Indeks akhir antrian

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

Mahasiswa *head = nullptr;
Mahasiswa *tail = nullptr;

// Fungsi untuk melihat apakah antrian penuh
bool isFull()
{
    return back == maksimalQueue;
}

// Fungsi untuk melihat apakah antrian kosong atau tidak
bool isEmpty()
{
    return back == 0;
}

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string nama, string nim)
{
    Mahasiswa *baru = new Mahasiswa;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = nullptr;
    if (tail != nullptr) {
        tail->next = baru;
        tail = baru;
    }
    else {
```



```

        head = baru;
        tail = baru;
    }
    back++;
}

// Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Mahasiswa *temp = head;
        if (head == tail) {
            head = nullptr;
            tail = nullptr;
        }
        else {
            head = head->next;
        }
        delete temp;
        back--;
    }
}

// Fungsi untuk menghitung banyak elemen dalam antrian
int countQueue()
{
    return back;
}

// Fungsi untuk menghapus semua elemen dalam antrian
void clearQueue()
{
    Mahasiswa *temp = head;
    head = nullptr;
    tail = nullptr;
    for (int i = 0; i < back - 1; i++)
    {
        Mahasiswa *del = temp;
        temp = temp->next;
    }
}

```

```

        delete del;
    }
    delete temp;
    back = 0;
    front = 0;
}
// Fungsi untuk melihat antrian semua elemen dalam antrian
void viewQueue()
{
    Mahasiswa *temp = head;
    cout << "Data antrian saat ini [Total " << countQueue() << "
antrian]: " << endl;
    for (int i = 0; i < maksimalQueue; i++)
    {
        if (i < back)
        {
            cout << i + 1 << ". " << temp->nama << " : " <<
temp->nim << endl;
            temp = temp->next;
        }
        else
        {
            cout << i + 1 << ". (kosong)" << endl;
        }
    }
}

int main()
{
    string line(15, '=');
    int choice;

    while(true) {
        cout << endl << line << "[ Antrian Mahasiswa ]" << line <<
endl;

        viewQueue();
        cout << "\nPilih menu : \n";
        cout << "1. Tambah antrian\n2. Hapus antrian\n3. Bersihkan
antrian\n0. Keluar\n";
        cout << "\nMasukkan pilihan : ";
        cin >> choice;
        cin.ignore();
    }
}

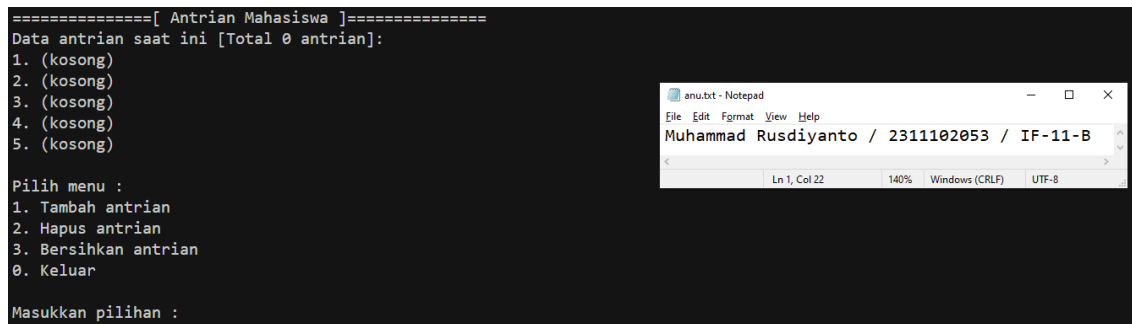
```

```

        switch(choice) {
            case 0 : return 0;
            case 1 : {
                if (isFull()) {
                    cout << "Antrian sudah penuh." << endl;
                }
                else {
                    string nama, nim;
                    cout << "Masukkan nama mahasiswa : ";
                    getline(cin, nama);
                    cout << "Masukkan NIM : ";
                    getline(cin, nim);
                    enqueueAntrian(nama, nim);
                    cout << nama << " berhasil dimasukkan ke
antrian." << endl;
                }
                break;
            }
            case 2 : {
                string temp = head->nama;
                dequeueAntrian();
                cout << temp << " berhasil dikeluarkan dari
antrian." << endl;
                break;
            }
            case 3 : {
                char clarify;
                cout << "Apakah anda yakin untuk menghapus semua
data antrian? [Y/T]";
                cin >> clarify;
                if (tolower(clarify) == 'y') {
                    clearQueue();
                    cout << "Semua data berhasil dihapus.\n\n";
                }
                else {
                    cout << "Penghapusan dibatalkan.\n\n";
                }
                break;
            }
        }
    }
    return 0;
}

```

Screenshots Output



```
===== [ Antrian Mahasiswa ] =====
Data antrian saat ini [Total 0 antrian]:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)

Pilih menu :
1. Tambah antrian
2. Hapus antrian
3. Bersihkan antrian
0. Keluar

Masukkan pilihan :
```

anu.txt - Notepad
File Edit Format View Help
Muhammad Rusdiyanto / 2311102053 / IF-11-B
Ln 1, Col 22 140% Windows (CRLF) UTF-8

Deskripsi:

Program di atas adalah sebuah program modifikasi dari program Guided 1. Program ini menggunakan single linked list non circular sebagai wadah penyimpanan data queue. Hal ini dilakukan dengan mengganti wadah sebelumnya yang merupakan array dengan struct untuk membentuk node linked list. Ada pula pointer head dan tail untuk menunjuk alamat memori dari data pertama dan data terakhir antrian. Di samping itu, program juga dimodifikasi dengan diberikan menu untuk mempermudah penggunaan program. Di dalam menu terdapat beberapa pilihan aksi seperti tambah antrian, hapus antrian, bersihkan antrian, dan keluar. Tidak lupa pula, karena struktur data yang digunakan berbeda, maka implementasi dalam fungsinya juga berbeda. Salah satu fungsi yang paling banyak modifikasinya adalah fungsi dequeue, dimana fungsi tersebut tidak memerlukan looping ke setiap data dalam antrian, karena sudah menggunakan linked list.

Unguided 2

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

const int maksimalQueue = 5; // Batas maksimal antrian
int front = 0;                // Indeks awal antrian
int back = 0;                 // Indeks akhir antrian

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

Mahasiswa *head = nullptr;
Mahasiswa *tail = nullptr;

// Fungsi untuk melihat apakah antrian penuh
bool isFull()
{
    return back == maksimalQueue;
}

// Fungsi untuk melihat apakah antrian kosong atau tidak
bool isEmpty()
{
    return back == 0;
}

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string nama, string nim)
{
    Mahasiswa *baru = new Mahasiswa;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = nullptr;
    if (tail != nullptr) {
        tail->next = baru;
        tail = baru;
    }
    else {
        head = baru;
        tail = baru;
    }
}
```

```

    }
    back++;
}

// Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        Mahasiswa *temp = head;
        if (head == tail) {
            head = nullptr;
            tail = nullptr;
        }
        else {
            head = head->next;
        }
        delete temp;
        back--;
    }
}

// Fungsi untuk menghitung banyak elemen dalam antrian
int countQueue()
{
    return back;
}

// Fungsi untuk menghapus semua elemen dalam antrian
void clearQueue()
{
    Mahasiswa *temp = head;
    head = nullptr;
    tail = nullptr;
    for (int i = 0; i < back - 1; i++)
    {
        Mahasiswa *del = temp;
        temp = temp->next;
        delete del;
    }
}

```

```

    }
    delete temp;
    back = 0;
    front = 0;
}

// Fungsi untuk melihat antrian semua elemen dalam antrian
void viewQueue()
{
    Mahasiswa *temp = head;
    cout << "Data antrian saat ini [Total " << countQueue() << "
antrian]: " << endl;
    for (int i = 0; i < maksimalQueue; i++)
    {
        if (i < back)
        {
            cout << i + 1 << ". " << temp->nama << " : " <<
temp->nim << endl;
            temp = temp->next;
        }
        else
        {
            cout << i + 1 << ". (kosong)" << endl;
        }
    }
}

int main()
{
    string line(15, '=');
    int choice;

    while(true) {
        cout << endl << line << "[ Antrian Mahasiswa ]" << line <<
endl;

        viewQueue();
        cout << "\nPilih menu : \n";
        cout << "1. Tambah antrian\n2. Hapus antrian\n3. Bersihkan
antrian\n0. Keluar\n";
        cout << "\nMasukkan pilihan : ";
        cin >> choice;
        cin.ignore();
    }
}

```

```

        switch(choice) {
            case 0 : return 0;
            case 1 : {
                if (isFull()) {
                    cout << "Antrian sudah penuh." << endl;
                }
                else {
                    string nama, nim;
                    cout << "Masukkan nama mahasiswa : ";
                    getline(cin, nama);
                    cout << "Masukkan NIM : ";
                    getline(cin, nim);
                    enqueueAntrian(nama, nim);
                    cout << nama << " berhasil dimasukkan ke
antrian." << endl;
                }
                break;
            }
            case 2 : {
                string temp = head->nama;
                dequeueAntrian();
                cout << temp << " berhasil dikeluarkan dari
antrian." << endl;
                break;
            }
            case 3 : {
                char clarify;
                cout << "Apakah anda yakin untuk menghapus semua
data antrian? [Y/T]";
                cin >> clarify;
                if (tolower(clarify) == 'y') {
                    clearQueue();
                    cout << "Semua data berhasil dihapus.\n\n";
                }
                else {
                    cout << "Penghapusan dibatalkan.\n\n";
                }
                break;
            }
        }
    }
    return 0;
}

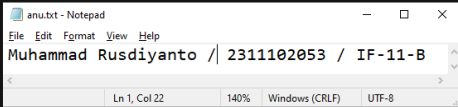
```


Screenshots Output

Menambah data

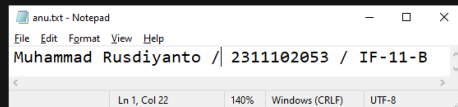
```
Masukkan pilihan : 1
Masukkan nama mahasiswa : Muhammad Rusdiyanto
Masukkan NIM : 2311102053
Muhammad Rusdiyanto berhasil dimasukkan ke antrian.

===== [ Antrian Mahasiswa ] =====
Data antrian saat ini [Total 1 antrian]:
1. Muhammad Rusdiyanto : 2311102053
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
```



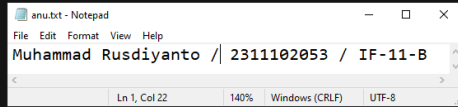
Hapus data

```
===== [ Antrian Mahasiswa ] =====
Data antrian saat ini [Total 3 antrian]:
1. Muhammad Rusdiyanto : 2311102053
2. J. M. K. Adiyasa : 2311102000
3. Chelsy Ambertron : 2211102000
4. (kosong)
5. (kosong)
```



```
Masukkan pilihan : 2
Muhammad Rusdiyanto berhasil dikeluarkan dari antrian.

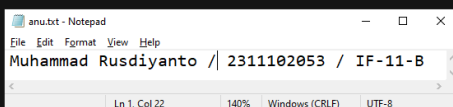
===== [ Antrian Mahasiswa ] =====
Data antrian saat ini [Total 2 antrian]:
1. J. M. K. Adiyasa : 2311102000
2. Chelsy Ambertron : 2211102000
3. (kosong)
4. (kosong)
5. (kosong)
```



Hapus semua data

```
Masukkan pilihan : 3
Apakah anda yakin untuk menghapus semua data antrian? [Y/T]Y
Semua data berhasil dihapus.

===== [ Antrian Mahasiswa ] =====
Data antrian saat ini [Total 0 antrian]:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
```



Deskripsi:

Secara rinci, program ini merupakan sebuah program pendataan antrian mahasiswa yang diimplementasikan menggunakan queue dengan struktur data single linked list. Program ini menyediakan beberapa menu untuk mengolah data dalam queue (seperti yang telah dijelaskan di Unguided 1). Menu tersebut di antaranya adalah :

1. Tambah antrian : Menu ini digunakan untuk menambahkan data antrian mahasiswa dengan cara memasukkan nama dan nim dari mahasiswa terkait. Sebelum data ditambahkan, program akan mengecek apakah antrian sudah penuh atau belum. Jika belum, maka data ditambahkan, jika penuh, maka sebuah pesan akan ditampilkan untuk memperingati pengguna.

2. Hapus antrian : Menu ini berfungsi untuk menghapus satu data di posisi awal antrian.
3. Bersihkan antrian : Berbeda dengan hapus, menu ini digunakan untuk menghapus semua data antrian dari awal sampai akhir. Sebelum semua data dihapus, program akan meminta kepastian dari pengguna. Pengguna perlu mengisikan `Y` (ya) untuk mengonfirmasi penghapusan data. Selain `Y`, maka program akan membatalkan proses penghapusan data antrian.
4. Keluar : Opsi ini digunakan untuk keluar dari program.

D. Kesimpulan

Queue dalam struktur data adalah konsep fundamental dalam ilmu komputer yang digunakan untuk menyimpan dan mengelola data dalam urutan tertentu. Queue mengikuti prinsip “First In, First Out” (FIFO), di mana elemen pertama yang ditambahkan ke dalam queue akan menjadi elemen pertama yang dihapus. Operasi dasar dari queue meliputi Enqueue (menambahkan elemen ke bagian belakang queue), Dequeue (menghapus elemen dari bagian depan queue), dan Peek (melihat elemen di bagian depan queue tanpa menghapusnya). Queue sering digunakan dalam berbagai algoritma dan aplikasi karena kemudahan dan efisiensinya dalam mengatur aliran data.

E. Referensi

Asisten Praktikum, “Modul 7 Queue”

GeeksforGeeks, 2024. Queue Data Structure. Diakses pada 20 Mei 2024 dari <https://www.geeksforgeeks.org/queue-data-structure/>.

Maulana, R., 2023. Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya. Dicoding Blog. Diakses pada 20 Mei 2024 dari : <https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/>.