

LAPORAN PRAKTIKUM

MODUL VII QUEUE



Disusun oleh:
Muhammad Agha Zulfadhli
NIM: 2311102015

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

- a. Mampu memahami konsep queue pada struktur data dan algoritma
- b. Mampu mengimplementasikan operasi-operasi pada queue
- c. Mampu memecahkan permasalahan dengan solusi queue

BAB II

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.

BAB III

GUIDED

1. Guided 1

Source code

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

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

bool isFull()
{
    if (back == maksimalQueue)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool isEmpty()
{
    if (back == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void enqueueAntrian(string data)
{
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        if (isEmpty())
        {
            queueTeller[0] = data;
        }
    }
}
```

```

        front++;
        back++;
    }
    else
    {
        queueTeller[back] = data;
        back++;
    }
}

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

int countQueue()
{
    return back;
}

void clearQueue()
{
    if (isEmpty())
    {
        cout << "Antrian Kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

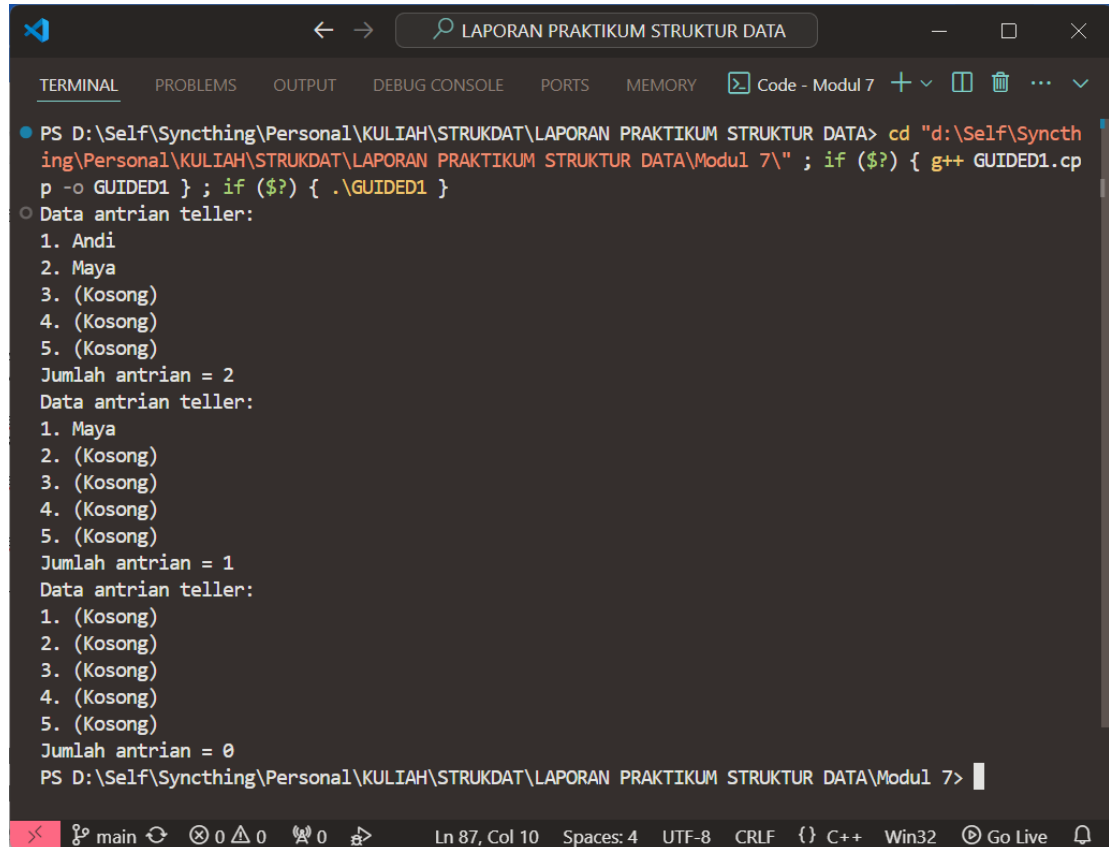
void viewQueue()
{
    cout << "Data antrian teller:" << endl;

```

```
        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;
    return 0;
}
```

Screenshoot program



```
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA> cd "d:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7\" ; if ($?) { g++ GUIDED1.cpp -o GUIDED1 } ; if ($?) { .\GUIDED1 }
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:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7>
```

Deskripsi program

Program di atas merupakan implementasi antrian (queue) menggunakan array. Program ini mendefinisikan ukuran maksimal antrian sebagai 5, serta dua variabel `front` dan `back` untuk melacak elemen terdepan dan elemen terakhir dalam antrian. Antrian tersebut disimpan dalam array `queueTeller` yang dapat menampung hingga lima elemen bertipe `string`.

Program ini menyediakan beberapa fungsi untuk mengelola antrian:

- `isFull()`: Mengecek apakah antrian sudah penuh.
- `isEmpty()`: Mengecek apakah antrian kosong.

- enqueueAntrian(string data): Menambahkan elemen baru ke belakang antrian jika antrian belum penuh.
- dequeueAntrian(): Menghapus elemen terdepan dari antrian jika antrian tidak kosong, dan menggeser semua elemen lainnya ke depan.
- countQueue(): Mengembalikan jumlah elemen dalam antrian.
- clearQueue(): Menghapus semua elemen dalam antrian dan mengatur ulang variabel `front` dan `back` ke 0.
- viewQueue(): Menampilkan seluruh elemen dalam antrian beserta posisi mereka.

Dalam fungsi main(), program melakukan beberapa operasi antrian:

1. Menambahkan dua nama ("Andi" dan "Maya") ke dalam antrian menggunakan enqueueAntrian().
2. Menampilkan isi antrian dan jumlah elemen menggunakan viewQueue() dan countQueue().
3. Menghapus satu elemen dari antrian dengan dequeueAntrian().
4. Menampilkan kembali isi dan jumlah elemen dalam antrian.
5. Mengosongkan antrian menggunakan clearQueue().
6. Menampilkan kembali isi dan jumlah elemen dalam antrian setelah dikosongkan.

Program ini mengilustrasikan dasar-dasar operasi antrian termasuk penambahan, penghapusan, pemeriksaan status, dan penampilan isi antrian.

LATIHAN KELAS - UNGUIDED

Unguided 1

Source code

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

const int maksimalQueue = 5;
int front = 0;
int back = 0;

struct Node {
    string name;
    Node *next;
    Node *prev;
};

bool isFull() {
    if (back == maksimalQueue) return true;
    else return false;
}

bool isEmpty() {
    if (back == 0) return true;
    else return false;
}

void enqueueAntrian(Node *&head, string data) {
    Node *add = new Node;
    add->name = data;
    if (isFull()) cout << "Antrian penuh" << endl;
    else {
        if (isEmpty()) {
            head = add;
            head->next = nullptr;
            head->prev = nullptr;
            front++;
            back++;
        } else {
            Node *current = head;
            for (int i = 0; i < back-1; i++) {
                current = current->next;
            }
            current->next = add;
            add->prev = current;
            add->next = nullptr;
            back++;
        }
    }
}
```

```

    }
}

void dequeueAntrian(Node *&head) {
    if (isEmpty()) cout << "Antrian kosong" << endl;
    else {
        Node *current = head->next;
        // for (int i = 0; i < back; i++) {
        //     queueTeller[i] = queueTeller[i + 1];
        // }
        head = current;
        head->prev = nullptr;
        back--;
    }
}

int countQueue() {return back;}

void clearQueue(Node *&head) {
    if (isEmpty()) {cout << "Antrian Kosong" << endl;}
    else {
        // for (int i = 0; i < back; i++) {
        //     queueTeller[i] = "";
        // }
        head = nullptr;
        back = 0;
        front = 0;
    }
}

void viewQueue(Node *&head) {
    cout << "Data antrian teller:" << endl;
    Node *current = head;
    for (int i = 0; i < maksimalQueue; i++) {
        // cout << i + 1 << ". " << queueTeller[i] << endl;
        if (current != nullptr) {
            cout << i + 1 << ". " << current->name << endl;
            current = current->next;
        } else {
            cout << i + 1 << ". (Kosong)" << endl;
        }
    }
}

int main() {
    Node *mahasiswa = new Node;

    enqueueAntrian(mahasiswa, "Andi");
    enqueueAntrian(mahasiswa, "Maya");
    viewQueue(mahasiswa);
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian(mahasiswa);
}

```

```

enqueueAntrian(mahasiswa, "Muhammad Agha Zulfadhli");
viewQueue(mahasiswa);
cout << "Jumlah antrian = " << countQueue() << endl;
clearQueue(mahasiswa);
viewQueue(mahasiswa);
cout << "Jumlah antrian = " << countQueue() << endl;
return 0;
}

```

Screenshoot program

```

LAPORAN PRAKTIKUM STRUKTUR DATA

TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  MEMORY  Code - Modul 7

PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7> cd "d:\Sel
f\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7\" ; if ($?) { g++ UN
GUIDED1.cpp -o UNGUIDED1 } ; if ($?) { .\UNGUIDED1 }
Data antrian teller:
1. Andi
2. Maya
3. (Kosong)
4. (Kosong)
5. (Kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
2. Muhammad Agha Zulfadhli
3. (Kosong)
4. (Kosong)
5. (Kosong)
Jumlah antrian = 2
Data antrian teller:
1. (Kosong)
2. (Kosong)
3. (Kosong)
4. (Kosong)
5. (Kosong)
Jumlah antrian = 0
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7>

```

Deskripsi program

Program di atas mengimplementasikan antrian (queue) menggunakan struktur data doubly linked list. Program ini mendefinisikan ukuran maksimal antrian sebagai 5, serta dua variabel front dan back untuk melacak elemen terdepan dan elemen terakhir dalam antrian. Antrian

tersebut disimpan dalam bentuk linked list dengan node yang memiliki dua pointer (next dan prev) serta sebuah string name.

Berikut penjelasan fungsi-fungsi dalam program ini:

- `isFull()`: Mengecek apakah antrian sudah penuh berdasarkan variabel `back` yang menunjukkan jumlah elemen dalam antrian.
- `isEmpty()`: Mengecek apakah antrian kosong berdasarkan variabel `back`.
- `enqueueAntrian(Node *&head, string data)`: Menambahkan elemen baru ke belakang antrian jika antrian belum penuh. Jika antrian kosong, elemen baru menjadi head dari linked list. Jika tidak, elemen baru ditambahkan ke akhir linked list.
- `dequeueAntrian(Node *&head)`: Menghapus elemen terdepan dari antrian jika antrian tidak kosong. Head dari linked list diganti dengan node berikutnya.
- `countQueue()`: Mengembalikan jumlah elemen dalam antrian berdasarkan variabel `back`.
- `clearQueue(Node *&head)`: Menghapus semua elemen dalam antrian dan mengatur ulang variabel `front` dan `back` ke 0. Head dari linked list diatur menjadi `nullptr`.
- `viewQueue(Node *&head)`: Menampilkan seluruh elemen dalam antrian beserta posisi mereka. Jika suatu posisi kosong, ditampilkan sebagai "(Kosong)".

Dalam fungsi `main()`, program melakukan beberapa operasi antrian:

1. Menambahkan dua nama ("Andi" dan "Maya") ke dalam antrian menggunakan `enqueueAntrian()`.

2. Menampilkan isi antrian dan jumlah elemen menggunakan `viewQueue()` dan `countQueue()`.
3. Menghapus satu elemen dari antrian dengan `dequeueAntrian()`.
4. Menambahkan satu nama lagi ("Muhammad Agha Zulfadhli") ke dalam antrian.
5. Menampilkan kembali isi dan jumlah elemen dalam antrian.
6. Mengosongkan antrian menggunakan `clearQueue()`.
7. Menampilkan kembali isi dan jumlah elemen dalam antrian setelah dikosongkan.

Program ini mengilustrasikan dasar-dasar operasi antrian termasuk penambahan, penghapusan, pemeriksaan status, dan penampilan isi antrian dengan menggunakan doubly linked list sebagai strukturnya.

Unguided 2

Source code

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

const int maksimalQueue = 5;
int front = 0;
int back = 0;

struct Node {
    string name;
    string nim;
    Node *next;
    Node *prev;
};

bool isFull() {
    if (back == maksimalQueue) return true;
    else return false;
}

bool isEmpty() {
    if (back == 0) return true;
    else return false;
}

void enqueueAntrian(Node *&head, string data, string nim) {
    Node *add = new Node;
    add->name = data;
    add->nim = nim;
    if (isFull()) cout << "Antrian penuh" << endl;
    else {
        if (isEmpty()) {
            head = add;
            head->next = nullptr;
            head->prev = nullptr;
            front++;
            back++;
        } else {
            Node *current = head;
            for (int i = 0; i < back-1; i++) {
                current = current->next;
            }
            current->next = add;
            add->prev = current;
            add->next = nullptr;
            back++;
        }
    }
}
```

```

}

void dequeueAntrian(Node *&head) {
    if (isEmpty()) cout << "Antrian kosong" << endl;
    else {
        Node *current = head->next;
        // for (int i = 0; i < back; i++) {
        //     queueTeller[i] = queueTeller[i + 1];
        // }
        head = current;
        head->prev = nullptr;
        back--;
    }
}

int countQueue() {return back;}

void clearQueue(Node *&head) {
    if (isEmpty()) {cout << "Antrian Kosong" << endl;}
    else {
        // for (int i = 0; i < back; i++) {
        //     queueTeller[i] = "";
        // }
        head = nullptr;
        back = 0;
        front = 0;
    }
}

void viewQueue(Node *&head) {
    cout << "Data Mahasiswa:" << endl;
    Node *current = head;
    for (int i = 0; i < maksimalQueue; i++) {
        // cout << i + 1 << ". " << queueTeller[i] << endl;
        if (current != nullptr) {
            cout << i + 1 << ". " << left << setw(30) << current->name << setw(12) << current->nim << endl ;
            current = current->next;
        } else {
            cout << i + 1 << ". (Kosong)" << endl;
        }
    }
}

int main() {
    Node *mahasiswa = new Node;

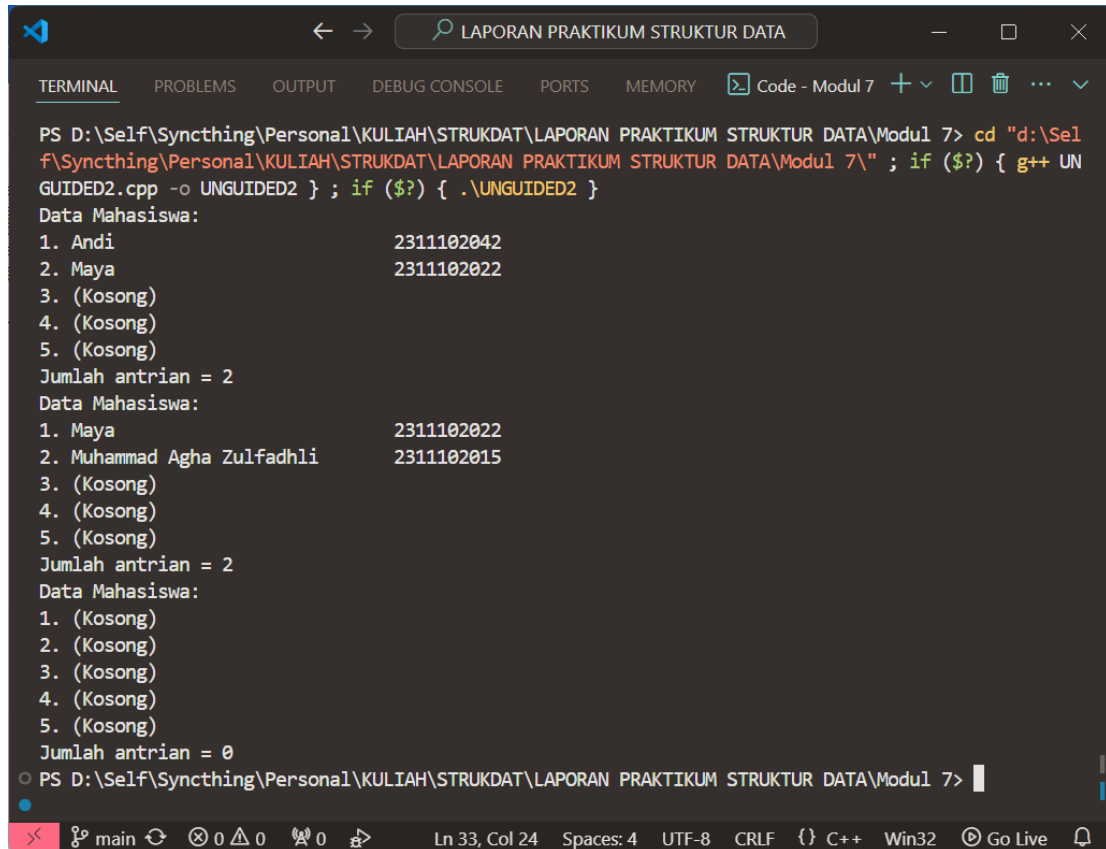
    enqueueAntrian(mahasiswa, "Andi", "2311102042");
    enqueueAntrian(mahasiswa, "Maya", "2311102022");
    viewQueue(mahasiswa);
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian(mahasiswa);
}

```



```
        enqueueAntrian(mahasiswa, "Muhammad Agha Zulfadhli",  
"2311102015");  
        viewQueue(mahasiswa);  
        cout << "Jumlah antrian = " << countQueue() << endl;  
        clearQueue(mahasiswa);  
        viewQueue(mahasiswa);  
        cout << "Jumlah antrian = " << countQueue() << endl;  
        return 0;  
    }
```

Screenshoot program



```
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7> cd "d:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7\" ; if ($?) { g++ UNGUIDED2.cpp -o UNGUIDED2 } ; if ($?) { .\UNGUIDED2 }
Data Mahasiswa:
1. Andi 2311102042
2. Maya 2311102022
3. (Kosong)
4. (Kosong)
5. (Kosong)
Jumlah antrian = 2
Data Mahasiswa:
1. Maya 2311102022
2. Muhammad Agha Zulfadhli 2311102015
3. (Kosong)
4. (Kosong)
5. (Kosong)
Jumlah antrian = 2
Data Mahasiswa:
1. (Kosong)
2. (Kosong)
3. (Kosong)
4. (Kosong)
5. (Kosong)
Jumlah antrian = 0
PS D:\Self\Syncthing\Personal\KULIAH\STRUKDAT\LAPORAN PRAKTIKUM STRUKTUR DATA\Modul 7>
```

Deskripsi program

Program di atas merupakan implementasi antrian (queue) menggunakan struktur data doubly linked list di bahasa pemrograman C++. Program ini mendefinisikan ukuran maksimal antrian sebagai 5, serta dua variabel front dan back untuk melacak elemen terdepan dan elemen terakhir dalam antrian. Antrian tersebut disimpan dalam bentuk linked list dengan node yang memiliki dua pointer (next dan prev) serta dua string name dan nim untuk menyimpan nama dan nomor induk mahasiswa.

Berikut penjelasan fungsi-fungsi dalam program ini:

- isFull(): Mengecek apakah antrian sudah penuh berdasarkan variabel back yang menunjukkan jumlah elemen dalam antrian.

- isEmpty(): Mengecek apakah antrian kosong berdasarkan variabel back.
- enqueueAntrian(Node *&head, string data, string nim): Menambahkan elemen baru ke belakang antrian jika antrian belum penuh. Jika antrian kosong, elemen baru menjadi head dari linked list. Jika tidak, elemen baru ditambahkan ke akhir linked list.
- dequeueAntrian(Node *&head): Menghapus elemen terdepan dari antrian jika antrian tidak kosong. Head dari linked list diganti dengan node berikutnya.
- countQueue(): Mengembalikan jumlah elemen dalam antrian berdasarkan variabel back.
- clearQueue(Node *&head): Menghapus semua elemen dalam antrian dan mengatur ulang variabel front dan back ke 0. Head dari linked list diatur menjadi nullptr.
- viewQueue(Node *&head): Menampilkan seluruh elemen dalam antrian beserta posisi mereka. Jika suatu posisi kosong, ditampilkan sebagai "(Kosong)".

Dalam fungsi main(), program melakukan beberapa operasi antrian:

1. Menambahkan dua mahasiswa ("Andi" dengan NIM "2311102042" dan "Maya" dengan NIM "2311102022") ke dalam antrian menggunakan enqueueAntrian().
2. Menampilkan isi antrian dan jumlah elemen menggunakan viewQueue() dan countQueue().
3. Menghapus satu elemen dari antrian dengan dequeueAntrian().
4. Menambahkan satu mahasiswa lagi ("Muhammad Agha Zulfadhli" dengan NIM "2311102015") ke dalam antrian.
5. Menampilkan kembali isi dan jumlah elemen dalam antrian.
6. Mengosongkan antrian menggunakan clearQueue().
7. Menampilkan kembali isi dan jumlah elemen dalam antrian setelah dikosongkan.

Program ini mengilustrasikan dasar-dasar operasi antrian termasuk penambahan, penghapusan, pemeriksaan status, dan penampilan isi antrian dengan menggunakan doubly linked list sebagai strukturnya, serta menambahkan penggunaan manipulasi output setw untuk mengatur lebar tampilan kolom saat menampilkan data mahasiswa.

BAB IV

KESIMPULAN

Dalam penelitian ini, telah dibahas mengenai implementasi dan konsep dasar dari struktur data queue yang menggunakan metode FIFO (First-In First-Out). Berikut adalah kesimpulan dari studi ini:

1. Metode FIFO pada Queue: Queue adalah struktur data yang mengimplementasikan metode FIFO, di mana data yang pertama dimasukkan akan menjadi data pertama yang dikeluarkan. Hal ini mirip dengan konsep antrian dalam kehidupan sehari-hari, di mana orang yang pertama datang akan dilayani terlebih dahulu.
2. Implementasi dengan Array dan Linked List: Queue dapat diimplementasikan menggunakan array atau linked list. Pada implementasi dengan array, terdapat batasan pada ukuran queue, sementara implementasi dengan linked list memungkinkan ukuran yang dinamis.
3. Penggunaan Pointer Front dan Rear: Struktur data queue menggunakan dua pointer, yaitu front yang menunjuk elemen pertama dan rear yang menunjuk elemen terakhir. Operasi penambahan (enqueue) dilakukan di ujung rear dan operasi penghapusan (dequeue) dilakukan di ujung front.
4. Perbedaan dengan Stack: Perbedaan utama antara queue dan stack adalah pada aturan penambahan dan penghapusan elemen. Stack menggunakan metode LIFO (Last-In First-Out), di mana elemen terakhir yang dimasukkan adalah elemen pertama yang dikeluarkan. Sebaliknya, queue menggunakan metode FIFO, di mana elemen pertama yang dimasukkan adalah elemen pertama yang dikeluarkan.

Penelitian ini menunjukkan bahwa queue merupakan struktur data yang efisien untuk mengelola data dalam urutan yang sesuai dengan kebutuhan aplikasi tertentu yang memerlukan pemrosesan data secara berurutan.

DAFTAR PUSTAKA

[1] Learning Management System ,MODUL 7 Queue.

[2] muh—agha—zul, 「PRAK」 Queue, 2024.

https://www.youtube.com/live/dAJzzPjvoW8?si=v76uhQ_vmHW8-1bw