LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL VII
QUEUE

Disusun Oleh :

NAMA : RIFKY DWI MAHARDIKA

NIM : 2311102043

Dosen

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

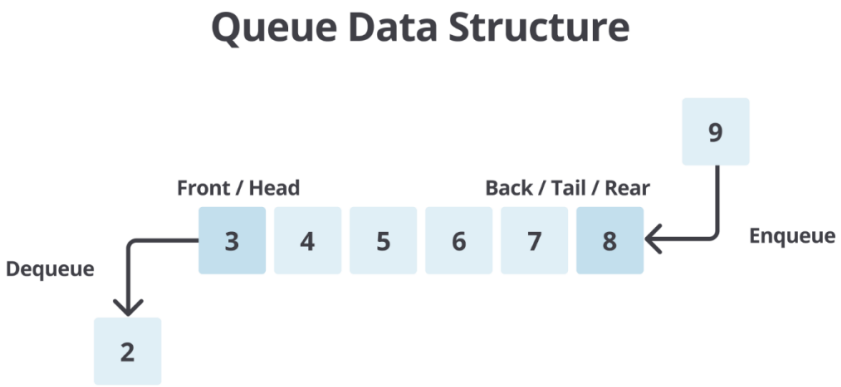
A. Dasar Teori

Queue

Queue atau dalam bahasa Indonesia yang berarti antrean 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 antrean akan menjadi yang pertama pula untuk dikeluarkan.

Cara Kerja FIFO pada Queue

Caranya bekerja adalah seperti jejeran orang yang sedang menunggu antrean di supermarket di mana 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 antrean disebut dengan Enqueue, sedangkan proses untuk menghapus data dari antrean disebut dengan Dequeue.



Queue memiliki peran yang penting dalam berbagai aplikasi dan algoritma. Salah satu fungsi utamanya adalah mengatur dan mengelola antrean tugas atau operasi secara efisien. Dalam sistem komputasi, ia digunakan untuk menangani tugas-tugas seperti penjadwalan proses, antrean pesan, dan manajemen sumber daya.

Kelebihan Queue

Kelebihan queue di antarnya:

- Data dalam jumlah besar dapat dikelola secara efisien.
- Operasi seperti penyisipan dan penghapusan dapat dilakukan dengan mudah karena mengikuti aturan masuk pertama keluar pertama.
- Queue berguna ketika layanan tertentu digunakan oleh banyak konsumen.
- Queue cepat untuk komunikasi antar-proses data.
- Queue dapat digunakan dalam implementasi struktur data lainnya.

Kekurangan Queue

Kelemahan struktur data queue adalah sebagai berikut:

- Operasi seperti penyisipan dan penghapusan elemen dari tengah cenderung banyak memakan waktu.
- Dalam queue konvensional, elemen baru hanya dapat dimasukkan ketika elemen yang ada dihapus dari antrian.
- Ukuran maksimum antrian harus ditentukan sebelumnya.

B. GUIDED

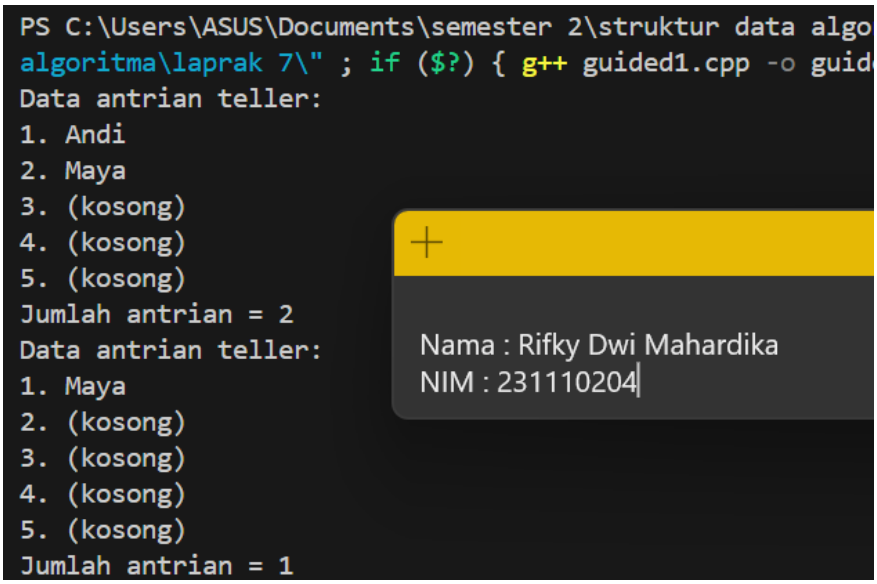
Guided 1

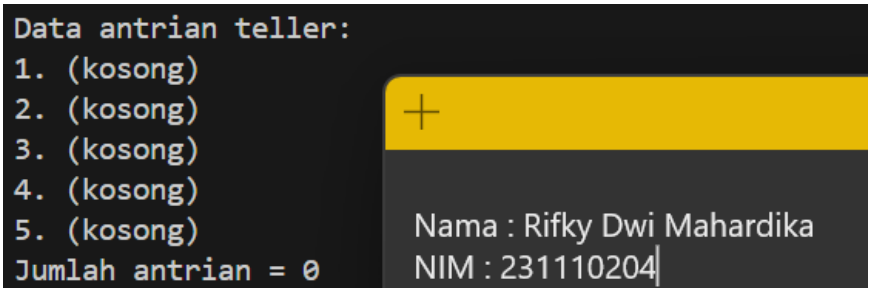
```cpp
#include <iostream>
using namespace std;
const int maksimalQueue = 5; // Maksimal antrian
int front = 0;               // Penanda antrian
int back = 0;                // Penanda
string queueTeller[5];       // Fungsi pengecekan
bool isFull()
{ // Pengecekan antrian penuh atau tidak
    if (back == 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;
}
```

Screenshots Output





Deskripsi

Program tersebut adalah implementasi antrian (queue) menggunakan array statis. Konstanta “maksimalQueue” menentukan kapasitas maksimal antrian (5), dan dua variabel “front” dan “back” sebagai penanda posisi elemen depan dan belakang. Fungsi “isFull()” dan “isEmpty()” mengecek kondisi antrian penuh dan kosong. Fungsi “enqueueAntrian()” menambahkan elemen ke antrian, sedangkan “dequeueAntrian()” menghapus elemen dari antrian. Fungsi “countQueue()” mengembalikan jumlah elemen dalam antrian, “clearQueue()” menghapus semua elemen dalam antrian, dan “viewQueue()” menampilkan isi antrian. Pada fungsi “main()”, beberapa operasi antrian seperti penambahan, penghapusan, dan penampilan elemen antrian dilakukan untuk demonstrasi.

C. UNGUIDED

Unguided 1

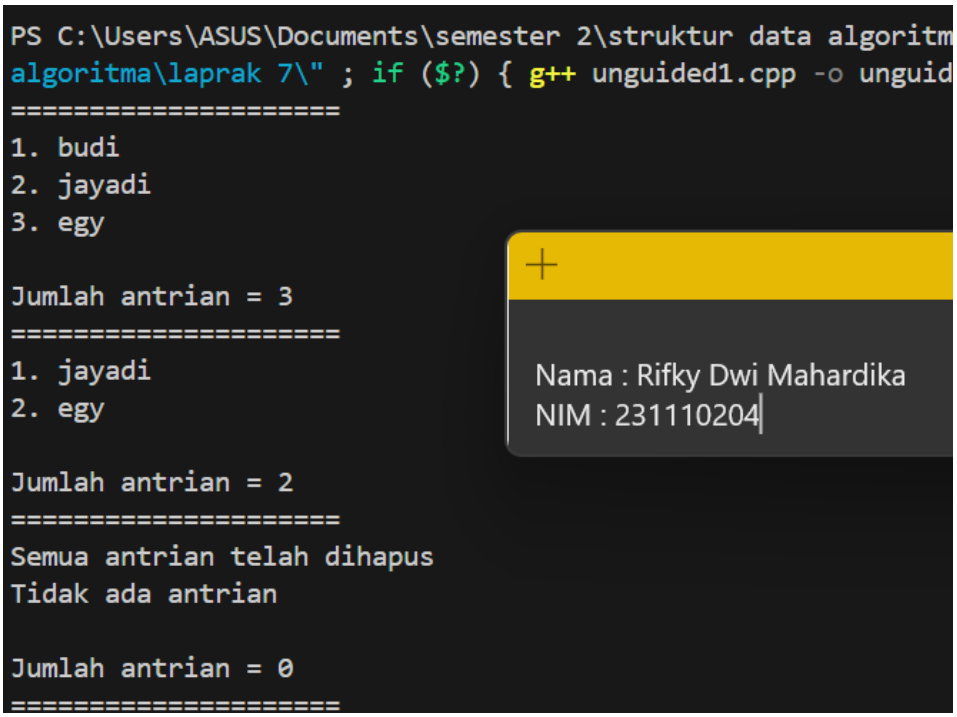
```cpp
#include <iostream>
using namespace std;

const int maximalQueue = 5;
int length = 0;

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

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isFull()
{
    return (length == maximalQueue);
}

bool isEmpty()
{
    return head == NULL;
}

void enqueueAntrian(string nilai)
{
    if (isFull())
    {
        cout << "Antrian Penuh" << endl;
    }
    else
    {
        Node *baru = new Node;
        baru->data = nilai;
        baru->next = NULL;
```

```
        if (isEmpty())
        {
            head = tail = baru;
        }
        else
        {
            tail->next = baru;
            tail = baru;
        }
        length++;
    }
}

void dequeueAntrian()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

void clearQueue()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
```

```
    cout << "Semua antrian telah dihapus" << endl;
}

void viewQueue()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL)
        {
            cout << index << ". " << bantu->data << " " << endl;
            bantu = bantu->next;
            index++;
        }
        cout << endl;
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

int countQueue()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

int main()
{
    init();
    cout << "====================" << endl;
    enqueueAntrian("budi");
    enqueueAntrian("jayadi");
    enqueueAntrian("egy");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    cout << "====================" << endl;
```

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

Screenshots Output



Deskripsi

Program ini adalah implementasi antrian (queue) menggunakan struktur data linked list. Konstanta “maximalQueue” menentukan kapasitas maksimal antrian (5), dan variabel “length” menyimpan jumlah elemen dalam antrian. Struktur Node mendefinisikan elemen antrian dengan data string dan pointer ke node berikutnya. Variabel pointer “head” dan “tail” menunjuk ke elemen pertama dan terakhir dalam antrian. Fungsi “init()” menginisialisasi antrian kosong. Fungsi “isFull()” mengecek apakah antrian penuh, dan

“isEmpty()” mengecek apakah antrian kosong. Fungsi “enqueueAntrian()” menambahkan elemen baru ke akhir antrian jika tidak penuh, dan “dequeueAntrian()” menghapus elemen dari depan antrian jika tidak kosong. Fungsi “clearQueue()” menghapus semua elemen dalam antrian, dan “viewQueue()” menampilkan semua elemen antrian. Fungsi “countQueue()” menghitung dan mengembalikan jumlah elemen dalam antrian. Pada fungsi “main()” beberapa operasi antrian seperti penambahan, penghapusan, pengosongan, dan penampilan elemen antrian dilakukan untuk demonstrasi.

Unguided 2

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

const int maximalQueue = 5;
int length = 0;

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

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isFull()
{
    return (length == maximalQueue);
}

bool isEmpty()
{
    return head == NULL;
}

void enqueueAntrian(string nama, string nim)
{
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        Node *baru = new Node;
        baru->nama = nama;
```

```
        baru->nim = nim;
        baru->next = NULL;
        if (isEmpty())
        {
            head = tail = baru;
        }
        else
        {
            tail->next = baru;
            tail = baru;
        }
        length++;
        cout << endl
             << "Berhasil Masuk Antrian";
    }
}

void dequeueAntrian()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

void clearQueue()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
```

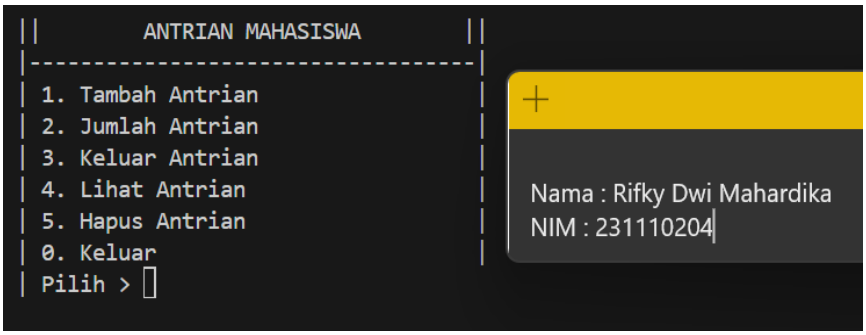
```
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
}

void viewQueue()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL)
        {
            cout << index << ". " << bantu->nama << " - " << bantu->nim
<< endl;
            bantu = bantu->next;
            index++;
        }
        cout << endl;
    }
    else
    {
        cout << "Antrian masih kosong" << endl;
    }
}

int countQueue()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

int main()
{
    init();
    system("cls");
    do
    {
```

```
        int choice;
        string nama, nim;
        cout << "||        ANTRIAN MAHASISWA        ||" << endl;
        cout << "|-----------------------------------|" << endl;
        cout << "| 1. Tambah Antrian                 |" << endl;
        cout << "| 2. Jumlah Antrian                 |" << endl;
        cout << "| 3. Keluar Antrian                 |" << endl;
        cout << "| 4. Lihat Antrian                  |" << endl;
        cout << "| 5. Hapus Antrian                  |" << endl;
        cout << "| 0. Keluar                         |" << endl;
        cout << "| Pilih > ";
        cin >> choice;
        cout << endl;
        switch (choice)
        {
        case 1:
            cout << "Masukkan Nama > ";
            cin.ignore();
            getline(cin, nama);
            cout << "Masukkan NIM > ";
            cin >> nim;
            enqueueAntrian(nama, nim);
            system("pause > nul");
            system("cls");
            break;

        case 2:
            cout << "Jumlah Antrian : " << countQueue() << endl;
            cout << endl;

            system("pause > nul");
            system("cls");
            break;

        case 3:
            dequeueAntrian();
            cout << "Berhasil keluar" << endl;
            cout << endl;

            system("pause > nul");
            system("cls");
            break;

        case 4:
            viewQueue();
```

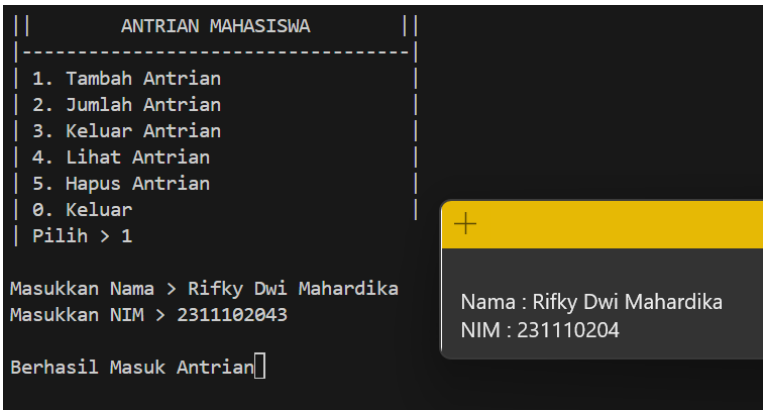
```
            cout << endl;

            system("pause > nul");
            system("cls");
            break;

        case 5:
            clearQueue();
            cout << "Data berhasil dihapus" << endl;
            cout << endl;

            system("pause > nul");
            system("cls");
            break;

        case 0:
            cout << "Anda telah keluar dari program ini. Terimakasih!"
<< endl;
            exit(0);

        default:
            break;
        }

    } while (true);

    return 0;
}
```

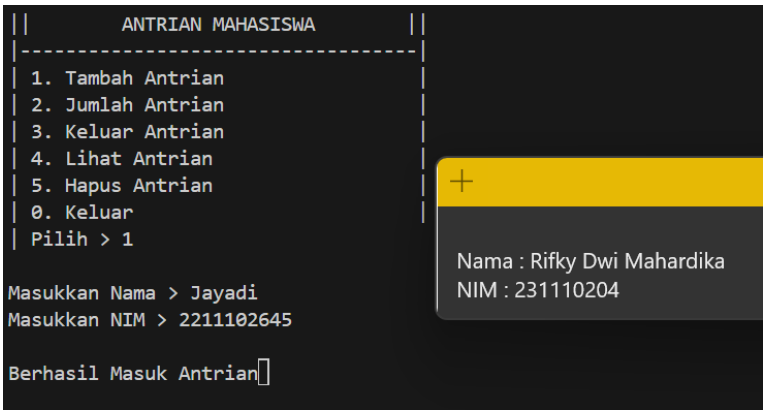
Screenshots Output

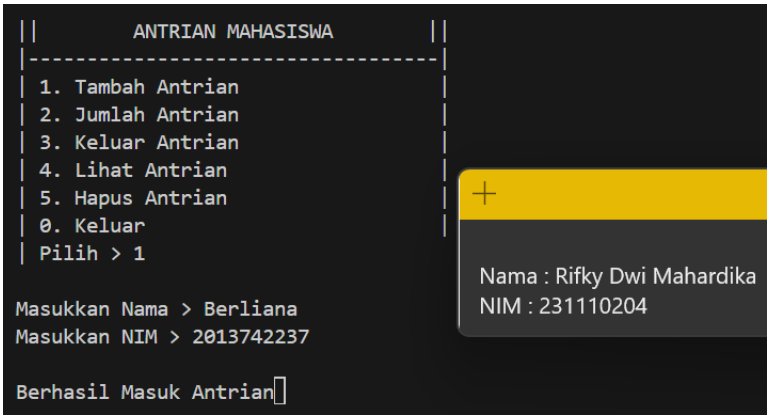
- Menu



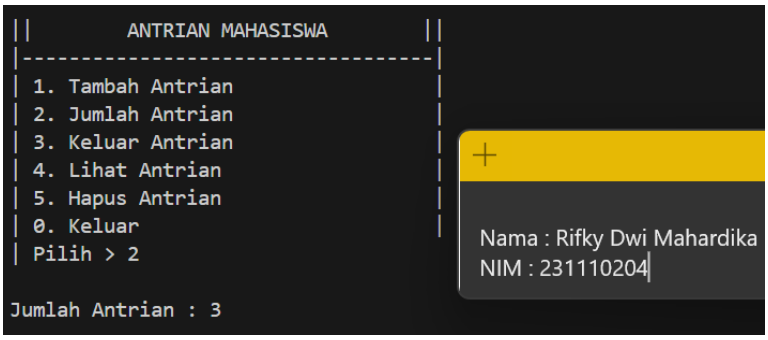
- Tambah Antrian



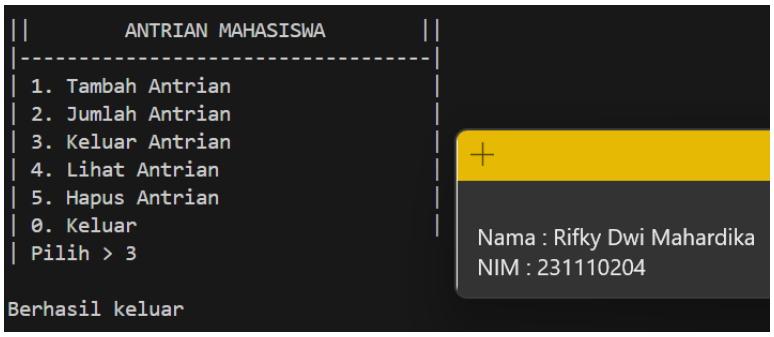




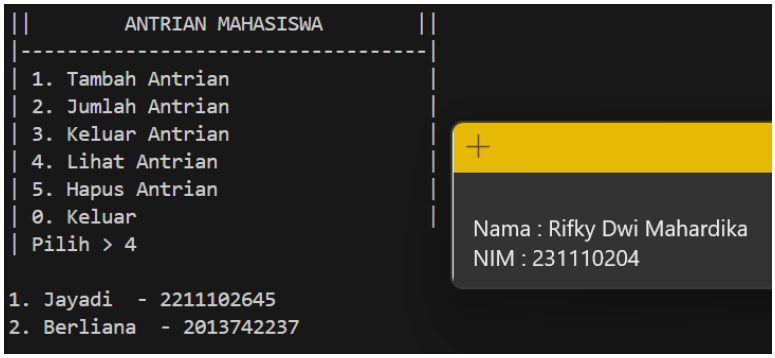
- Jumlah Antrian



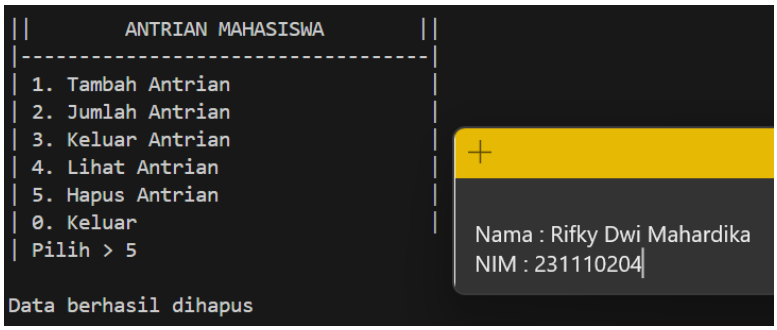
- Keluar Antrian



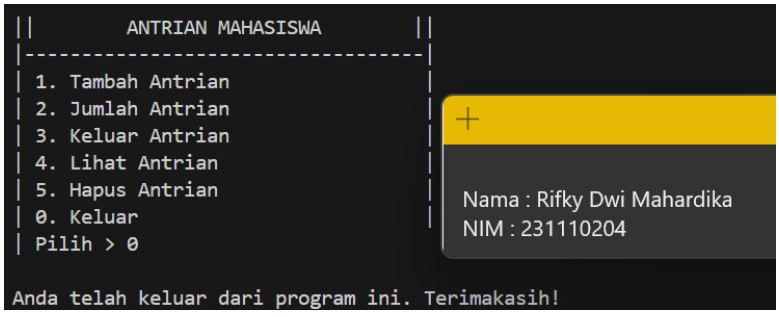
- Lihat Antrian



- Hapus Antrian



- Keluar/Exit



Deskripsi

Program ini adalah implementasi antrian (queue) menggunakan struktur data linked list, khusus untuk mengelola antrian mahasiswa berdasarkan nama dan NIM. Program memiliki konstanta “maximalQueue” yang menentukan kapasitas maksimal antrian (5) dan variabel “length” untuk menyimpan jumlah elemen dalam antrian. Struktur Node mendefinisikan elemen antrian dengan atribut “(nama, nim)” dan pointer ke node berikutnya. Variabel pointer “head” dan “tail” menunjuk ke elemen pertama dan terakhir dalam antrian. Fungsi “init()” menginisialisasi antrian kosong, “isFull()” mengecek apakah antrian penuh, dan “isEmpty()” mengecek apakah antrian kosong. Fungsi “enqueueAntrian()” menambahkan elemen baru ke akhir antrian jika tidak penuh, sementara “dequeueAntrian()” menghapus elemen dari depan antrian jika tidak kosong. Fungsi “clearQueue()” menghapus semua elemen dalam antrian, dan “viewQueue()” menampilkan semua elemen antrian beserta indeksnya. Fungsi “countQueue()” menghitung dan mengembalikan jumlah elemen dalam antrian. Pada fungsi “main()”, terdapat menu interaktif untuk menambah, menghitung, menghapus, melihat, dan membersihkan antrian, serta untuk keluar dari program. Menu ini diimplementasikan dalam loop do-while yang terus berjalan sampai pengguna memilih untuk keluar dari program.

D. Kesimpulan

Queue adalah struktur data yang digunakan untuk menyimpan berbagai elemen yang disusun sesuai prinsip, “first in, first out” (FIFO). Artinya, elemen yang masuk pertama kali pada antrean juga dikeluarkan pertama kali dari antrean. Queue sangat penting dalam berbagai hal misalnya antrean di bank, sistem permintaan jaringan komputer, dan sebagainya.

E. Referensi

[1] Asisten Praktikum. (2024). Modul VII : Queue

[2] Trivusi. 01 juli 2023. Struktur Data Queue: Pengertian, Jenis, dan Kegunaannya. Diakses pada 26 Mei 2024, dari https://www.trivusi.web.id/2022/07/struktur-data-queue.html

[3] Rizki Maulana. 29 November 2023. Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya. Diakses pada 26 Mei 2024, dari https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/