LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL 7

“QUEUE”

DISUSUN OLEH :

M.Fathiakmal.L.A

18102096

DOSEN

WAHYU ANDI SAPUTRA, S.PD., M.PD.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

A. DASAR TEORI

1. PENGENALAN QUEUE

i. Definisi Queue

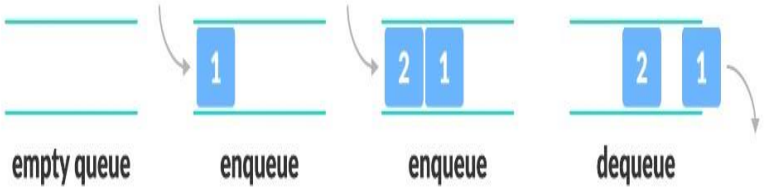
Queue atau dalam bahasa Indonesia yang berarti antrean adalah struktur data yang menyusun elemen-elemen data dalam urutan linier. Prinsip dasar dari struktur data ini adalah “First In, First Out” (FIFO) yang berarti elemen data yang pertama dimasukkan ke dalam antrean akan menjadi yang pertama pula untuk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

Queue adalah struktur data yang mengikuti prinsip First-In-First-Out (FIFO), yang berarti elemen pertama yang masuk ke dalam antrian akan menjadi elemen pertama yang keluar dari antrian. Queue dapat diibaratkan sebagai antrian di mana elemen-elemen baru ditambahkan di satu ujung antrian (rear) dan elemenelemen yang sudah ada dikeluarkan di ujung lainnya (front). Queue mirip dengan antrian nyata yang sering kita temui dalam kehidupan sehari-hari. **ii. Prinsip Queue (FIFO — First-In-First-Out)**

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.

Karakteristik utama dari Queue adalah prinsip FIFO (First-In-First-Out). Elemen pertama yang dimasukkan ke dalam Queue akan menjadi elemen pertama yang diambil atau dihapus dari Queue. Elemen-elemen baru ditambahkan di ujung belakang Queue dan elemen-elemen yang sudah ada dikeluarkan dari ujung depan Queue. Dengan prinsip FIFO ini, Queue dapat membantu mengatur urutan data dan mempertahankan prioritas saat memproses elemen-elemen yang ada di dalamnya

- **Ilustrasi dari Queue**



iii. Perbedaan Stack dan Queue

Perbedaan utama antara stack dan queue terletak pada cara penyisipan dan penghapusan elemen. Pada stack, kedua operasi dilakukan pada ujung yang sama, yaitu ujung atas. Sedangkan pada queue, kedua operasi dilakukan pada ujung yang berbeda, yaitu ujung depan dan ujung belakang.

Pada Stuck Ini mengikuti urutan LIFO (Last In First Out) untuk menyimpan elemen, artinya elemen yang dimasukkan terakhir akan keluar lebih dulu.

Berbedan dengan Stuck pada Queue Mengikuti urutan FIFO (First In First Out) untuk menyimpan elemen, artinya elemen yang dimasukkan terlebih dahulu akan keluar terlebih dahulu.

iv. Operasi Pada Queue

- enqueue() :menambahkandatakedalamqueue.
- dequeue() :mengeluarkandatadariqueue.
- peek() :mengambildatadariqueuetanpamenghapusnya.
- isEmpty() :mengecekapakahqueuekosongatautidak.
- isFull() :mengecekapakahqueuepenuhatautidak.
- size() :menghitungjumlahelemendalamqueue

B. Guided

1. Guided

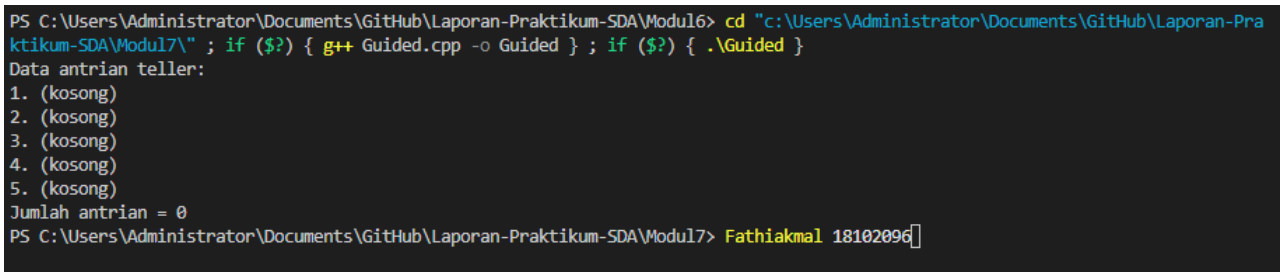
Source Code:

```
#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 koson
            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:



Deskripsi Program:

Program ini mengimplementasikan antrian (queue) sederhana menggunakan array untuk menyimpan data antrian nasabah di teller bank dengan kapasitas maksimum lima antrian. Fungsi isFull() dan isEmpty() digunakan untuk memeriksa apakah antrian penuh atau kosong. Fungsi enqueueAntrian() menambahkan nasabah ke dalam antrian jika tidak penuh, sedangkan dequeueAntrian() menghapus nasabah dari antrian jika tidak kosong. Fungsi countQueue() mengembalikan jumlah nasabah dalam antrian, dan clearQueue() menghapus seluruh antrian. Fungsi viewQueue() menampilkan seluruh antrian beserta statusnya (kosong atau diisi nasabah). Dalam fungsi main(), beberapa operasi antrian dilakukan: menambahkan nasabah "Andi" dan "Maya" ke antrian, menampilkan antrian,

menghitung jumlah antrian, menghapus satu antrian, menampilkan kembali antrian, mengosongkan antrian, dan menampilkan antrian terakhir beserta jumlahnya.

C. UNGUIDED

Unguided 1:

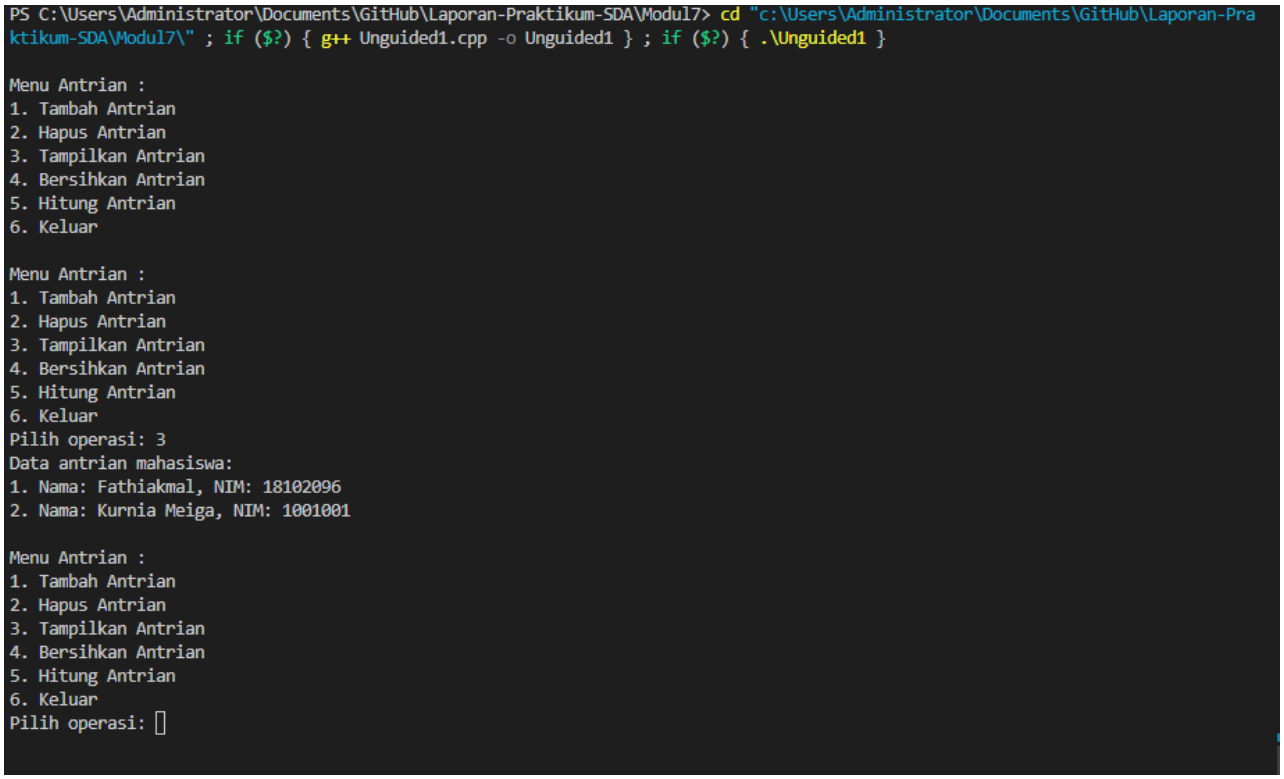
Source Code:

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

struct Node {
    string namaMahasiswa;
    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 namaMahasiswa, string nim) {
        Node* newNode = new Node{namaMahasiswa, nim, nullptr};
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
```

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

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

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

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

void displayMenu() {
    cout << "\nMenu Antrian :" << endl;
    cout << "1. Tambah Antrian" << endl;
    cout << "2. Hapus Antrian" << endl;
    cout << "3. Tampilkan Antrian" << endl;
```

```
    cout << "4. Bersihkan Antrian" << endl;
    cout << "5. Hitung Antrian" << endl;
    cout << "6. Keluar" << endl;
    cout << "Pilih operasi: ";
}

int main() {
    Queue q;
    int choice;
    string namaMahasiswa, nim;

    do {
        displayMenu();
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Masukkan Nama Mahasiswa: ";
                cin.ignore(); // Mengabaikan newline yang tersisa di input buffer
                getline(cin, namaMahasiswa);
                cout << "Masukkan NIM: ";
                cin >> nim;
                q.enqueue(namaMahasiswa, nim);
                break;
            case 2:
                q.dequeue();
                break;
            case 3:
                q.viewQueue();
                break;
            case 4:
                q.clearQueue();
                break;
            case 5:
                cout << "Jumlah antrian = " << q.countQueue() << endl;
                break;
            case 6:
                cout << "Keluar dari program." << endl;
                break;
            default:
                cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
        }
    } while (choice != 6);

    return 0;
}
```

Screenshot Output:



Deskripsi Program:

Program ini mengimplementasikan antrian (queue) mahasiswa menggunakan struktur data linked list, dengan fungsi-fungsi untuk menambah, menghapus, menampilkan, membersihkan, dan menghitung jumlah elemen dalam antrian. Struktur `Node` menyimpan data nama mahasiswa dan NIM, serta pointer ke node berikutnya. Kelas `Queue` memiliki pointer `front` dan `back` untuk menandai awal dan akhir antrian. Fungsi `isEmpty()` mengecek apakah antrian kosong, `enqueue()` menambahkan mahasiswa ke antrian, dan `dequeue()` menghapus mahasiswa dari antrian. Fungsi `viewQueue()` menampilkan seluruh data dalam antrian, `clearQueue()` menghapus semua elemen dalam antrian, dan `countQueue()` menghitung jumlah elemen dalam antrian. Fungsi `displayMenu()` menampilkan menu pilihan operasi. Dalam fungsi `main()`, pengguna dapat memilih untuk menambah antrian, menghapus antrian, menampilkan antrian, membersihkan antrian, atau menghitung jumlah elemen antrian hingga mereka memilih untuk keluar dari program.

Unguided 2:

Source Code:

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

struct Node {
    string namaMahasiswa;
```

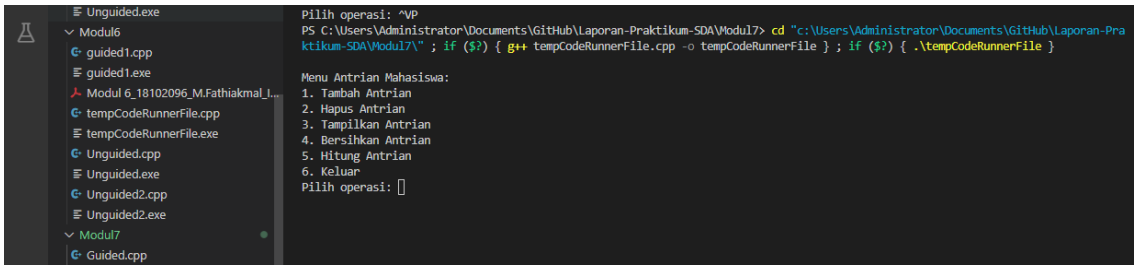
```
    string nim;
    Node* next;
};

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

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

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

    void enqueue(string namaMahasiswa, string nim) {
        Node* newNode = new Node{namaMahasiswa, nim, nullptr};
        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

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

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong." << endl;
        } else {
            cout << "Data antrian mahasiswa:" << endl;
```

```
            Node* current = front;
            int index = 1;
            while (current != nullptr) {
                cout << index << ". Nama: " << current->namaMahasiswa << ", NIM: "
<< current->nim << endl;
                current = current->next;
                index++;
            }
        }
    }

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

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

void displayMenu() {
    cout << "\nMenu Antrian Mahasiswa:" << endl;
    cout << "1. Tambah Antrian" << endl;
    cout << "2. Hapus Antrian" << endl;
    cout << "3. Tampilkan Antrian" << endl;
    cout << "4. Bersihkan Antrian" << endl;
    cout << "5. Hitung Antrian" << endl;
    cout << "6. Keluar" << endl;
    cout << "Pilih operasi: ";
}

int main() {
    Queue q;
    int choice;
    string namaMahasiswa, nim;

    do {
        displayMenu();
        cin >> choice;
        switch (choice) {
            case 1:
```

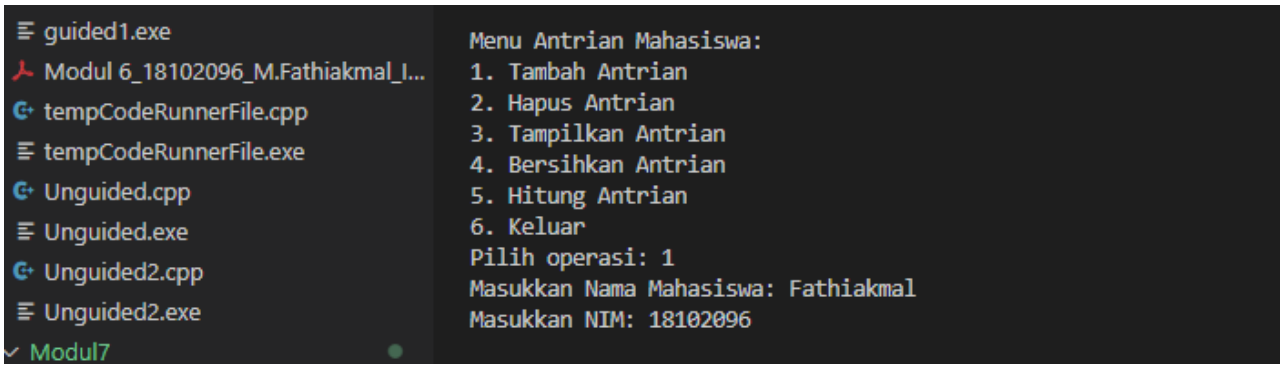
```
                cout << "Masukkan Nama Mahasiswa: ";
                cin.ignore(); // Mengabaikan newline yang tersisa di input buffer
                getline(cin, namaMahasiswa);
                cout << "Masukkan NIM: ";
                cin >> nim;
                q.enqueue(namaMahasiswa, nim);
                break;
            case 2:
                q.dequeue();
                break;
            case 3:
                q.viewQueue();
                break;
            case 4:
                q.clearQueue();
                break;
            case 5:
                cout << "Jumlah antrian = " << q.countQueue() << endl;
                break;
            case 6:
                cout << "Keluar dari program." << endl;
                break;
            default:
                cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
        }
    } while (choice != 6);

    return 0;
}
```

Screenshot Output:

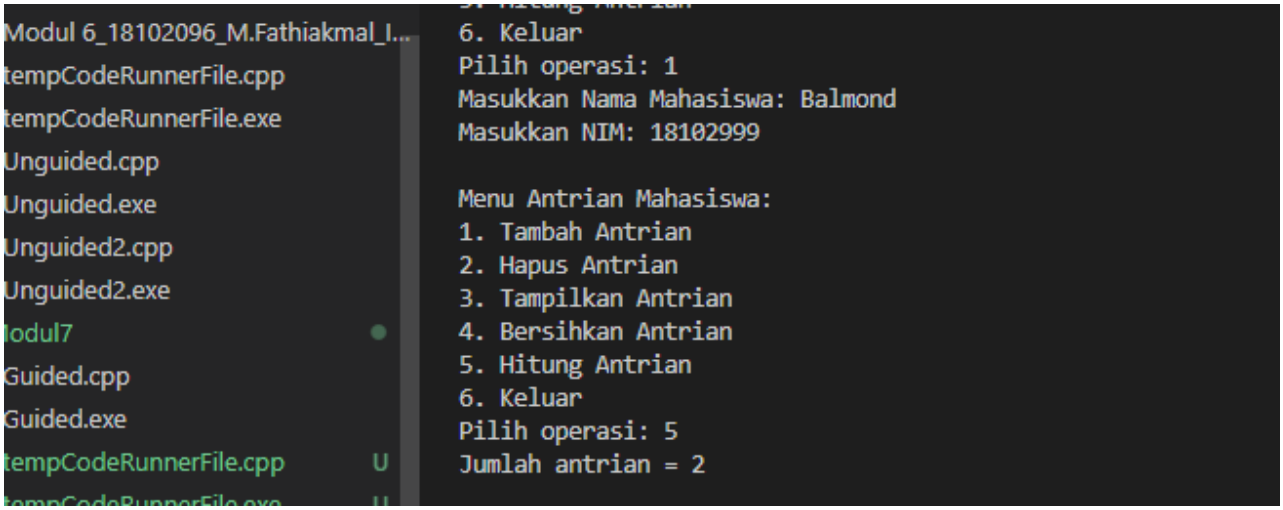
Menu:



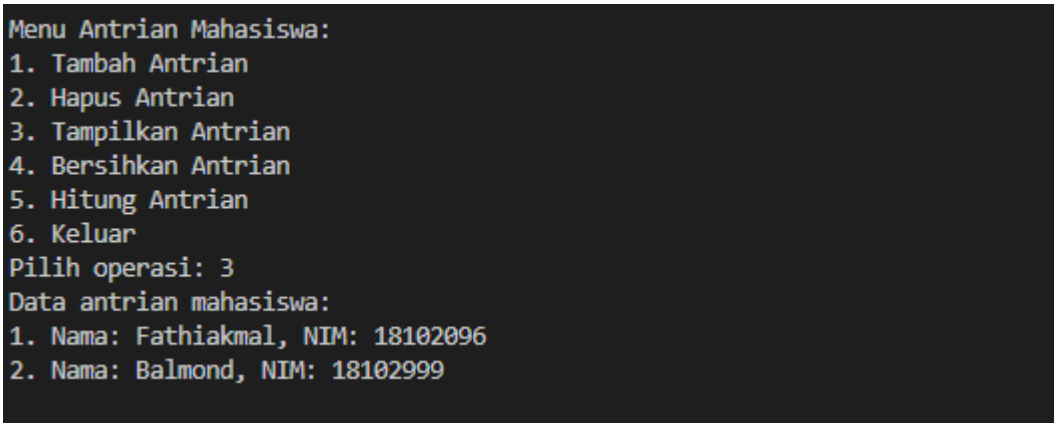
Tambah Antrian:



Hitung antrian:



Tampilkan antrian:



Hapus antrian pertama:

```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar
Pilih operasi: 2
Antrian Berhasil Dihapus

Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar
Pilih operasi: 3
Data antrian mahasiswa:
1. Nama: Balmond, NIM: 18102999
```

Bersihkan data antrian:

```
Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar
Pilih operasi: 4
Antrian Berhasil Dihapus

Menu Antrian Mahasiswa:
1. Tambah Antrian
2. Hapus Antrian
3. Tampilkan Antrian
4. Bersihkan Antrian
5. Hitung Antrian
6. Keluar
Pilih operasi: 3
Antrian kosong.
```

Deskripsi Program:

Program ini mengimplementasikan antrian (queue) menggunakan struktur data linked list untuk menyimpan data mahasiswa, dengan fungsi-fungsi untuk menambah, menghapus, menampilkan, membersihkan, dan menghitung jumlah elemen dalam antrian. Struktur Node menyimpan nama mahasiswa, NIM, dan pointer ke node berikutnya. Kelas Queue memiliki pointer front dan back untuk menandai awal dan akhir antrian. Fungsi isEmpty() mengecek apakah antrian kosong, enqueue() menambahkan mahasiswa ke antrian, dan dequeue() menghapus mahasiswa dari antrian serta menampilkan pesan konfirmasi. Fungsi viewQueue() menampilkan seluruh data dalam antrian, clearQueue() menghapus semua elemen dalam antrian, dan countQueue() menghitung jumlah elemen dalam antrian. Fungsi displayMenu() menampilkan menu pilihan operasi. Dalam fungsi main(), pengguna dapat memilih untuk menambah antrian, menghapus antrian, menampilkan antrian, membersihkan antrian, atau menghitung jumlah elemen antrian hingga mereka memilih untuk keluar dari program.

D. Kesimpulan

1. Pengenalan Antrian (Queue)

Antrian adalah struktur data linear yang mengikuti prinsip First In First Out (FIFO), di mana elemen yang pertama kali dimasukkan akan menjadi elemen yang pertama kali dikeluarkan. Antrian memiliki dua operasi dasar, yaitu enqueue (menambah elemen ke antrian) dan dequeue (menghapus elemen dari antrian).

2. Implementasi Antrian dengan Array Statis

Implementasi antrian menggunakan array statis melibatkan penggunaan array dengan ukuran tetap untuk menyimpan elemen-elemen antrian. Contoh program ini memiliki beberapa fungsi penting:

- **isFull**: Memeriksa apakah antrian sudah penuh.
- **isEmpty**: Memeriksa apakah antrian kosong.
- **enqueueAntrian**: Menambah elemen ke antrian jika belum penuh.

- **dequeueAntrian**: Menghapus elemen dari antrian jika tidak kosong.
- **countQueue**: Menghitung jumlah elemen dalam antrian.
- **clearQueue**: Menghapus semua elemen dalam antrian.
- **viewQueue**: Menampilkan semua elemen dalam antrian.

3. Implementasi Antrian dengan Linked List

Implementasi antrian menggunakan linked list melibatkan penggunaan node-node yang saling terhubung untuk menyimpan elemen-elemen antrian. Contoh program ini juga memiliki beberapa fungsi penting:

- **isEmpty**: Memeriksa apakah antrian kosong.
- **enqueue**: Menambah elemen baru ke antrian dengan membuat node baru dan menambahkannya di akhir antrian.
- **dequeue**: Menghapus elemen dari antrian dengan menghapus node di depan antrian.
- **viewQueue**: Menampilkan semua elemen dalam antrian dengan iterasi melalui node-node.
- **clearQueue**: Menghapus semua elemen dalam antrian dengan menghapus semua node satu per satu.
- **countQueue**: Menghitung jumlah elemen dalam antrian dengan menghitung jumlah node.

4. Kesimpulan

Kedua implementasi antrian memiliki fungsi dasar yang sama, yaitu menambah, menghapus, menampilkan, menghitung, dan membersihkan elemen-elemen antrian. Namun, implementasi dengan array statis memiliki keterbatasan ukuran yang tetap, sedangkan implementasi dengan linked list lebih fleksibel karena ukuran antrian dapat bertambah sesuai dengan kebutuhan. Memahami kedua metode ini memberikan

wawasan yang baik tentang cara kerja antrian dan bagaimana struktur data ini dapat diimplementasikan dalam berbagai situasi.

E. Referensi

[1] Asisten Pratikum “Modul 7 Queue”, Learning Management System, 2024.

[2] Maulana, Rizki, “Struktur data queue: pengertian fungsi dan jenis”. (May 25, 2023), diakses pada 27 mei, dari

Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya (dicoding.com)

[3] cahyasmara, “Antrian / Queue Pemograman C/C++”, (2017), diakses pada 23 mei, dari https://cahyasmara.blogspot.com/2017/06/antrian-queue-pemograman-ccstruktur.html