LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA

MODUL 7
"QUEUE"

DISUSUN OLEH :
BINTANG YUDHISTIRA
2311102052

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.

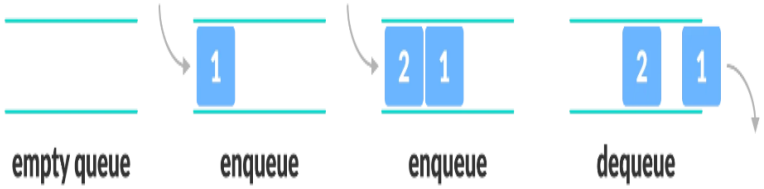
Queue adalah struktur data yang mengikuti prinsip First-In-First-Out (FIFO), yang berarti elemen pertama yang masuk ke dalam antrian akan menjadi elemen pertama yang keluar dari antrian. Queue dapat diibaratkan sebagai antrian di mana elemen-elemen baru ditambahkan di satu ujung antrian (rear) dan elemen-elemen 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

Guided 1.

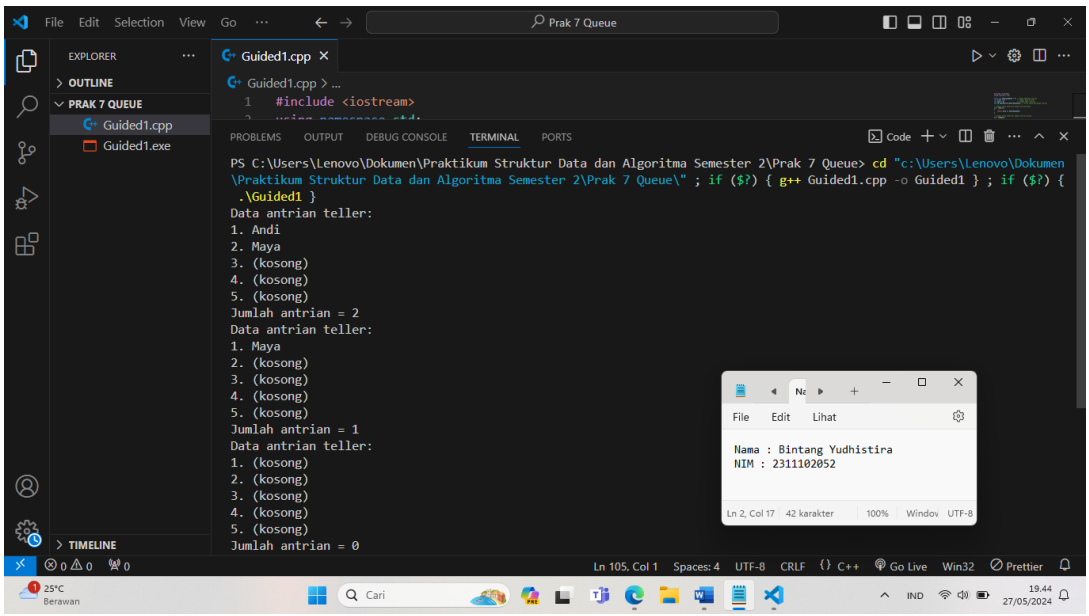
Source Code:

```
#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 antrian

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

// Fungsi untuk memeriksa apakah antrian kosong
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 jumlah elemen dalam antrian
int countQueue()
{
    return back;
}

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

// Fungsi untuk menempilkan semua elemen dalam antrian
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 diatas adalah struktur data queue (antrian), Menggunakan sebuah array untuk menyimpan data antrian. Didalam kode diatas terdapat fungsi-fungsi seperti penambahan(enqueue antrian) dan penghapusan (dequeue antrian). Dalam output

codingan diatas terlihat 2 data nama yaitu maya dan andi. Ketika kita menambahkan 3 data nama lagi maka nama andi dan maya akan tetap berada di posisi 1 dan 2 sebaliknya nama terakhir yg diinputkan akan akan terakhir keluar.

Program ini menetapkan kapasitas maksimum antrian (maksimalQueue) sebanyak 5 elemen. Terdapat beberapa fungsi yang bertujuan untuk memanipulasi antrian: isFull() untuk memeriksa apakah antrian penuh, isEmpty() untuk memeriksa apakah antrian kosong, enqueueAntrian() untuk menambahkan elemen ke antrian, dequeueAntrian() untuk menghapus elemen dari antrian, countQueue() untuk menghitung jumlah elemen dalam antrian, clearQueue() untuk mengosongkan semua elemen dalam antrian, dan viewQueue() untuk menampilkan semua elemen dalam antrian. Fungsi main() digunakan untuk menguji fungsi-fungsi tersebut dengan menambahkan, menampilkan, menghapus, dan mengosongkan antrian, serta menampilkan jumlah elemen yang ada di antrian. Contoh data yang digunakan dalam antrian adalah nama-nama seperti "Andi" dan "Maya".

C. UNGUIDED

Unguided 1:

Source Code:

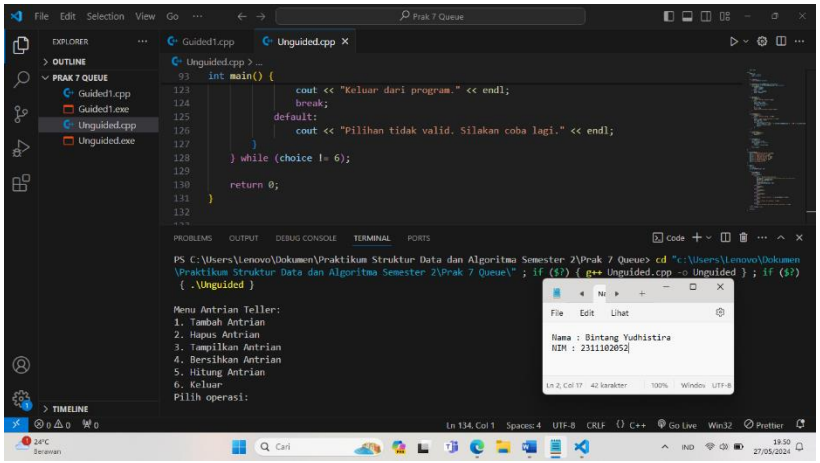
```
#include <iostream>
using namespace std;

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 Teller:" << 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 menggunakan linked list untuk mengimplementasikan struktur data queue antrian mahasiswa. Setiap mahasiswa direpresentasikan sebagai node dalam linked list, yang menyimpan informasi seperti nama dan NIM serta pointer ke node berikutnya. Operasi-operasi utama termasuk menambah, menghapus, menampilkan, membersihkan, dan menghitung jumlah mahasiswa dalam antrian.

Unguided 2:

Source Code:

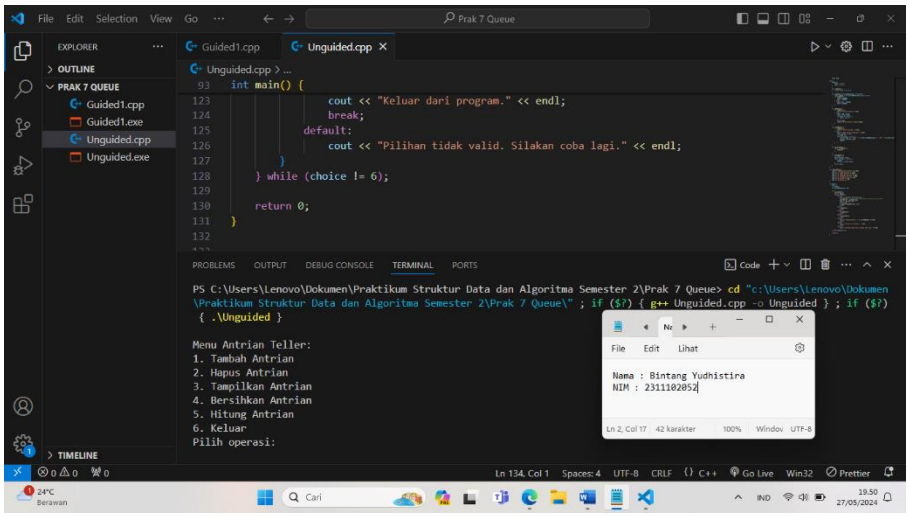
```
#include <iostream>
using namespace std;

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;
```

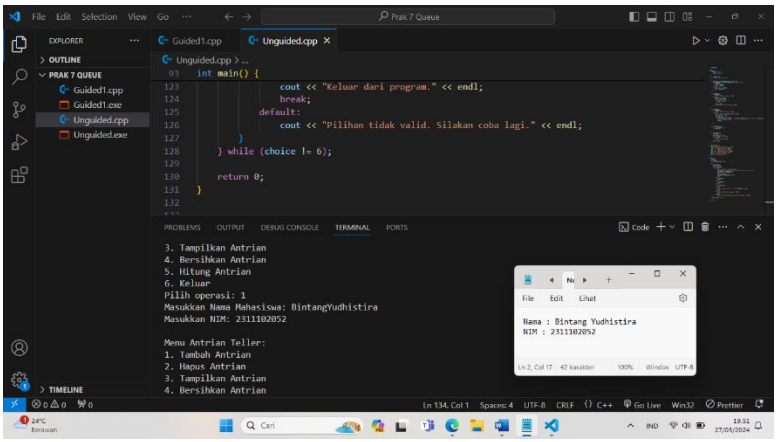
```
    int choice;
    string namaMahasiswa, nim;

    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:

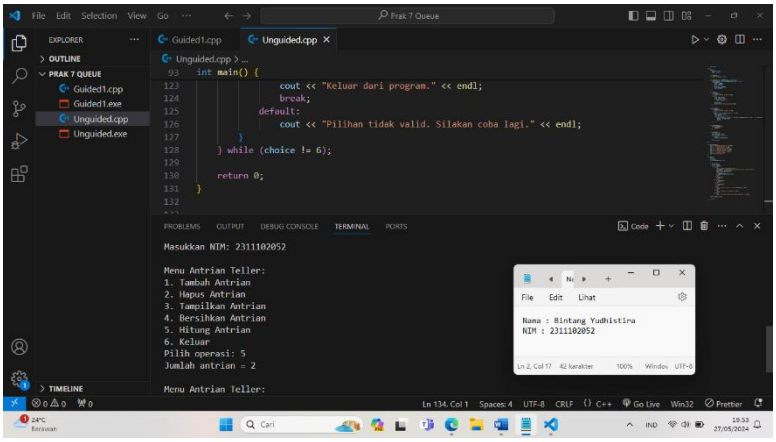
Menu:



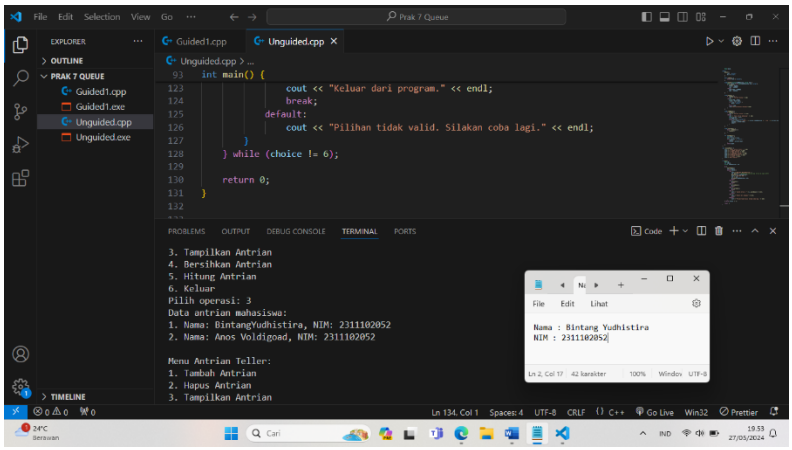
Tambah Antrian:



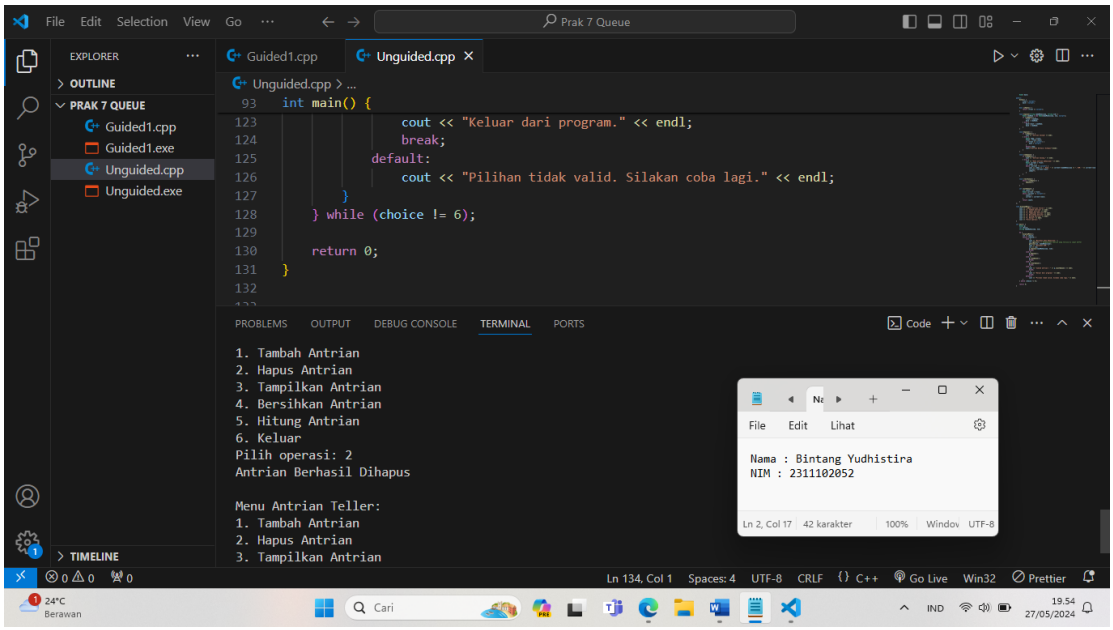
Hitung antrian:



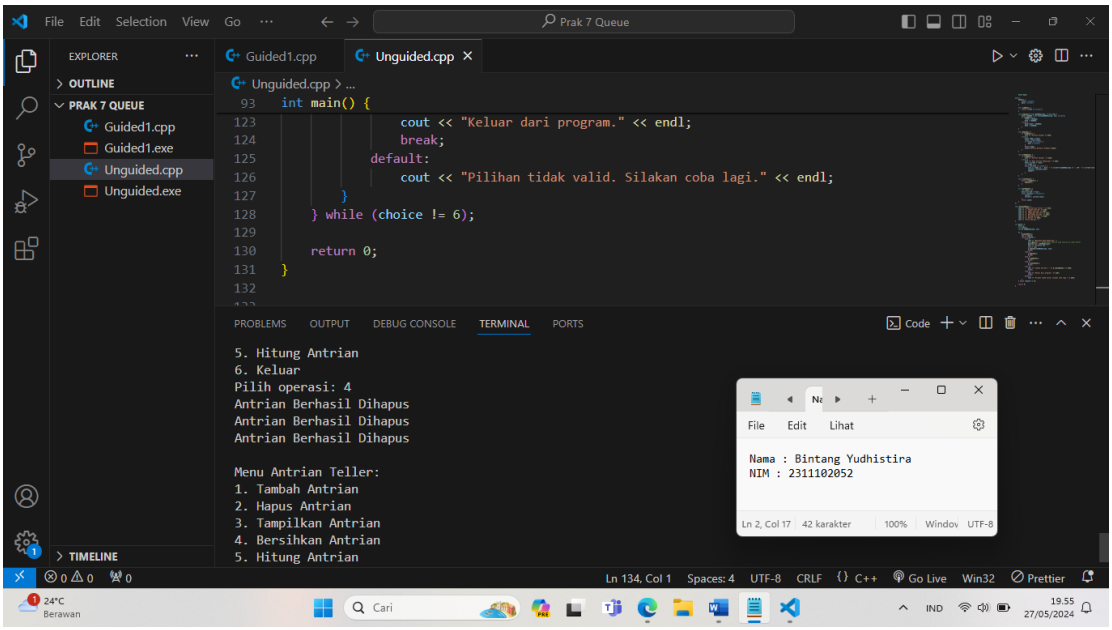
Tampilkan antrian:



Hapus antrian pertama:



Bersihkan data antrian:



Deskripsi Program:

Program ini merupakan implementasi sederhana dari struktur data queue sistem antrian untuk manajemen mahasiswa. Menggunakan struktur data queue, program memungkinkan pengguna untuk menambah antrian, menghapus antrian, menampilkan antrian, bersihkan antrian, hitung antrian, dan keluar.

Fungsi isEmpty() memeriksa apakah antrian kosong dengan memeriksa apakah front adalah nullptr. Fungsi enqueue() menambahkan elemen baru (node) ke antrian. Jika antrian kosong, elemen baru menjadi front dan back. Jika tidak, elemen baru ditambahkan di belakang dan back di-update. Fungsi dequeue() menghapus elemen dari depan antrian. Jika antrian kosong, pesan ditampilkan. Jika tidak, elemen depan dihapus dan front di-update. Jika setelah penghapusan antrian menjadi kosong, back juga di-set ke nullptr. Fungsi viewQueue() menampilkan semua elemen dalam antrian. Jika antrian kosong, pesan ditampilkan. Jika tidak, semua elemen ditampilkan satu per satu. Fungsi clearQueue() menghapus semua elemen dalam antrian hingga kosong dengan memanggil dequeue berulang kali. Fungsi countQueue() menghitung dan mengembalikan jumlah elemen dalam antrian. Fungsi displayMenu() menampilkan pilihan menu untuk operasi antrian. Fungsi main() mengelola input dari pengguna dan memanggil fungsi yang sesuai berdasarkan pilihan menu. Pengguna dapat menambah, menghapus, menampilkan,

membersihkan, dan menghitung elemen dalam antrian, serta keluar dari program dengan memilih opsi yang tersedia. Input dan output ditangani dengan menggunakan cin dan cout. Program berulang hingga pengguna memilih untuk keluar.

D. Kesimpulan

Queue (antrian) adalah struktur data yang mengikuti prinsip FIFO (First In, First Out), di mana elemen pertama yang masuk adalah elemen pertama yang keluar. Queue digunakan untuk mengelola data dalam urutan yang teratur, misalnya dalam pemrosesan antrian pelanggan atau tugas. Operasi dasar pada queue meliputi menambah elemen ke belakang antrian (enqueue), menghapus elemen dari depan antrian (dequeue), memeriksa apakah antrian kosong (isEmpty), dan menghitung jumlah elemen dalam antrian (countQueue). Queue dapat diimplementasikan menggunakan array atau linked list, masing-masing dengan kelebihan dan kekurangannya sendiri terkait efisiensi penggunaan memori dan kompleksitas operasi. Struktur data ini sangat penting dalam berbagai aplikasi pemrograman dan sistem komputer, termasuk manajemen proses, penjadwalan, dan buffering.

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-cc-struktur.html