LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VIII

QUEUE

Disusun Oleh :

Nama : Besthian Guido Rafael Simbolon

NIM : 103112430258

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

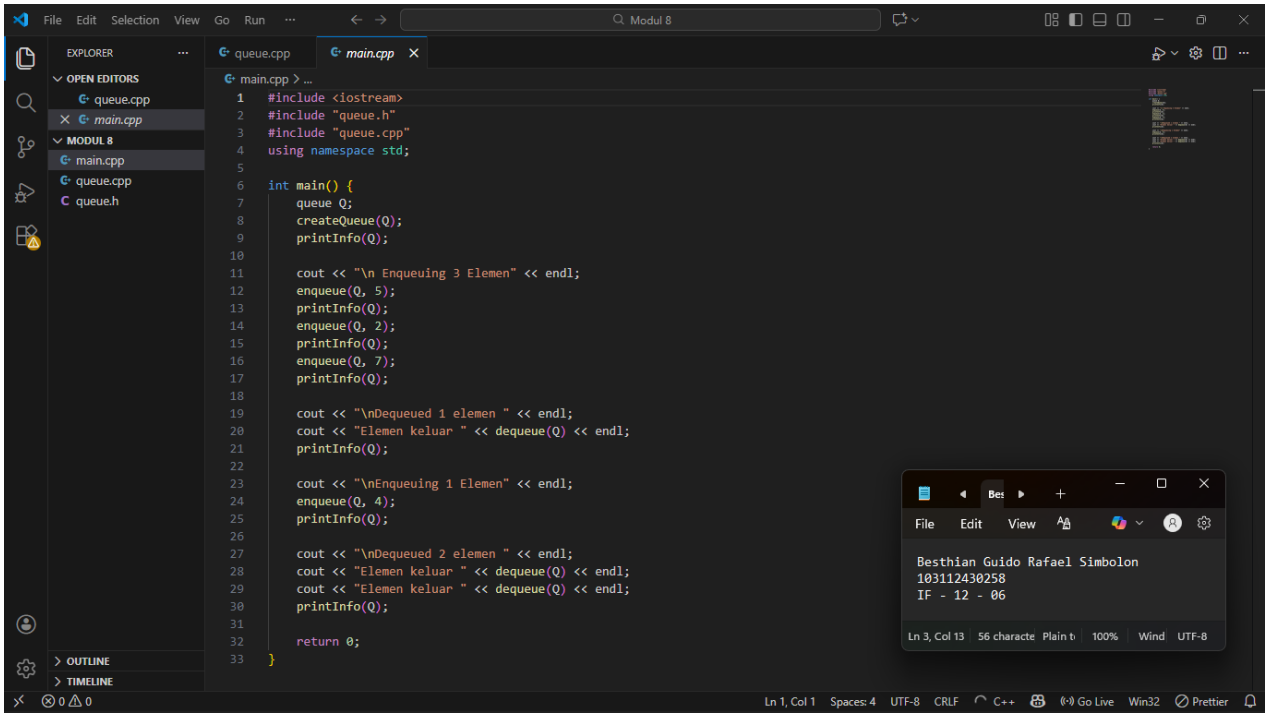
Queue adalah salah satu struktur data linier yang mengatur aliran data berdasarkan urutan kedatangan, sehingga elemen yang lebih dulu dimasukkan akan diproses lebih dulu dibandingkan elemen yang datang belakangan. Pola kerjanya dapat dianalogikan seperti barisan kendaraan di lampu merah, di mana kendaraan paling depan akan bergerak lebih dulu saat lampu hijau menyala.

Dalam penerapannya menggunakan array pada bahasa C++, queue memiliki dua penunjuk posisi, yaitu bagian depan sebagai lokasi pengambilan data dan bagian belakang sebagai tempat penambahan data baru. Penambahan elemen ke dalam queue dilakukan melalui operasi enqueue, sedangkan penghapusan elemen dilakukan melalui operasi dequeue. Karena menggunakan array, ruang penyimpanan queue bersifat terbatas dan tidak dapat bertambah secara dinamis, sehingga pengelolaan indeks harus dilakukan dengan hati-hati agar tidak melebihi kapasitas yang tersedia.

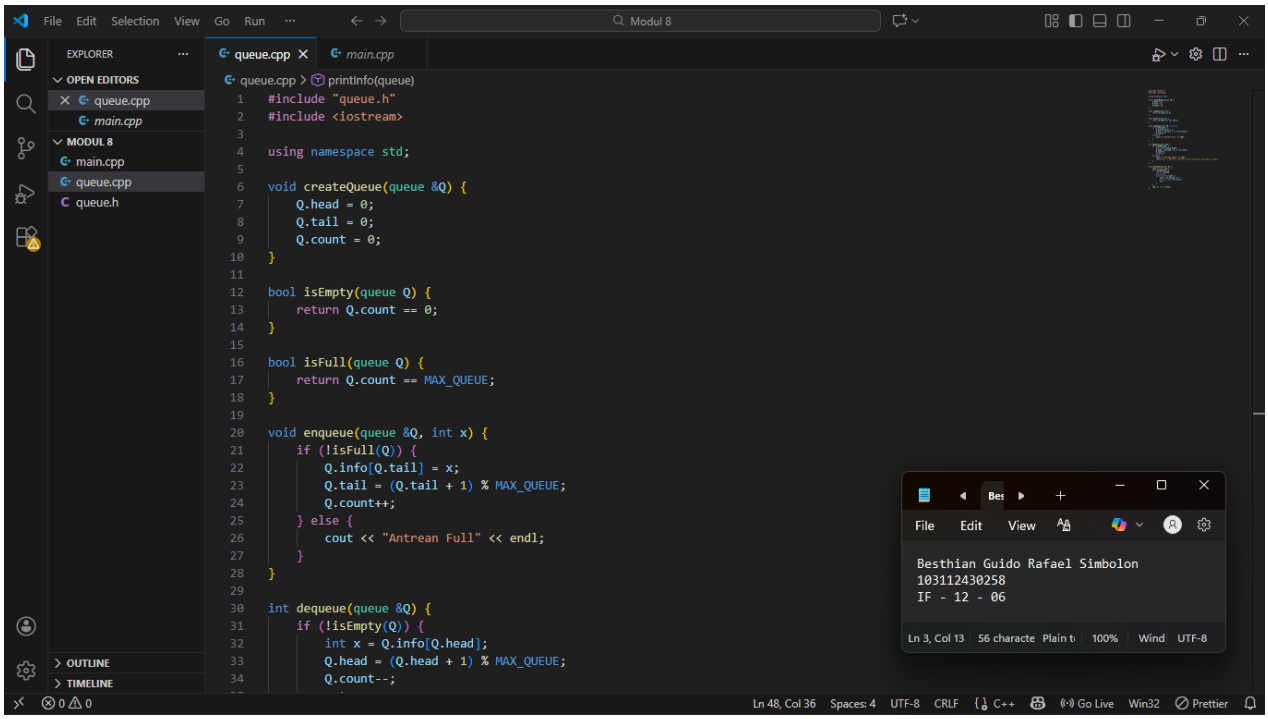
B. Guided

Guided 1

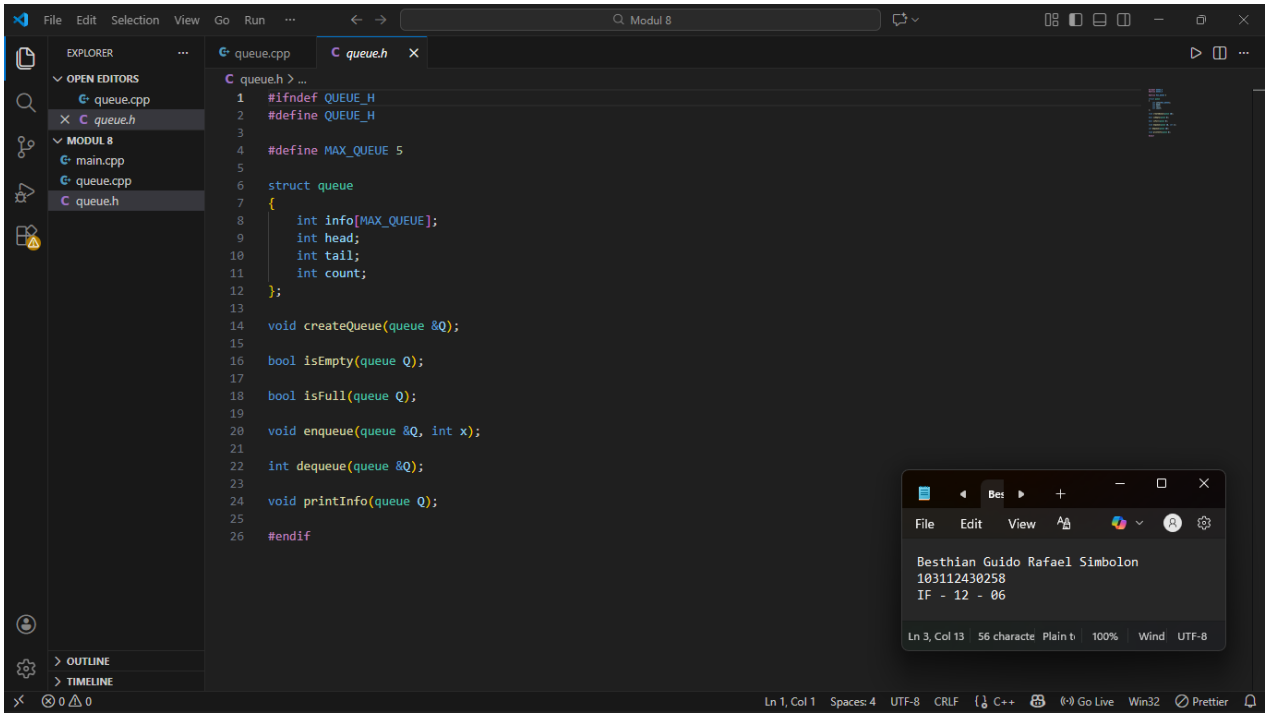
Main.cpp



Queue.cpp


File Edit Selection View Go Run
Modul 8
EXPLORER
OPEN EDITORS
queue.cpp
main.cpp
MODUL 8
main.cpp
queue.cpp
queue.h
OUTLINE
TIMELINE
queue.cpp
main.cpp
queue.cpp > printInfo(queue)
#include "queue.h"
#include <iostream>
using namespace std;
void createQueue(queue &Q) {
Q.head = 0;
Q.tail = 0;
Q.count = 0;
}
bool isEmpty(queue Q) {
return Q.count == 0;
}
bool isFull(queue Q) {
return Q.count == MAX_QUEUE;
}
void enqueue(queue &Q, int x) {
if (!isFull(Q)) {
Q.info[Q.tail] = x;
Q.tail = (Q.tail + 1) % MAX_QUEUE;
Q.count++;
} else {
cout << "Antrean Full" << endl;
}
}
int dequeue(queue &Q) {
if (!isEmpty(Q)) {
int x = Q.info[Q.head];
Q.head = (Q.head + 1) % MAX_QUEUE;
Q.count--;
File Edit View
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 13 56 characte Plain t 100% Wind UTF-8
Ln 48, Col 36 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

Queue.h


File Edit Selection View Go Run
Modul 8
EXPLORER
OPEN EDITORS
queue.cpp
queue.h
MODUL 8
main.cpp
queue.cpp
queue.h
OUTLINE
TIMELINE
queue.cpp
queue.h
queue.h > ...
#ifndef QUEUE_H
#define QUEUE_H
#define MAX_QUEUE 5
struct queue
{
int info[MAX_QUEUE];
int head;
int tail;
int count;
};
void createQueue(queue &Q);
bool isEmpty(queue Q);
bool isFull(queue Q);
void enqueue(queue &Q, int x);
int dequeue(queue &Q);
void printInfo(queue Q);
#endif
File Edit View
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 13 56 characte Plain t 100% Wind UTF-8
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

Screenshots Output

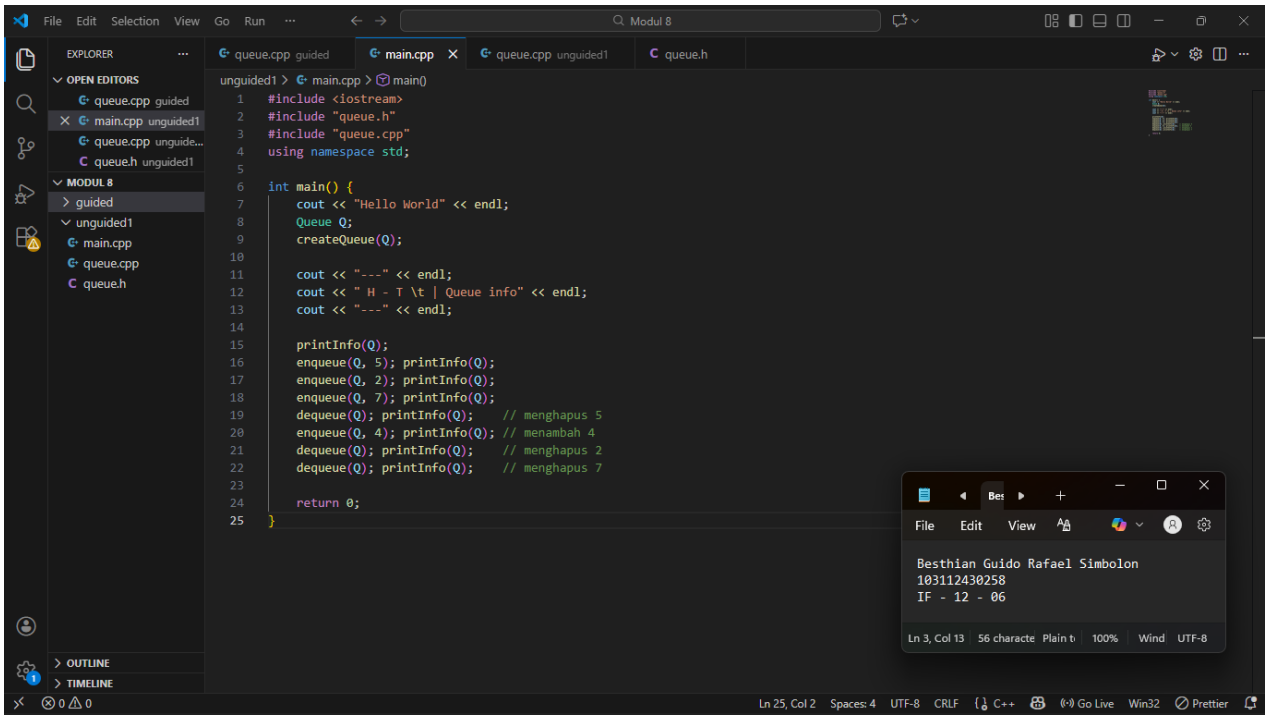
Deskripsi:

Program ini merupakan contoh penerapan **circular queue** yang memanfaatkan sebuah variabel penghitung bernama *count* untuk mengelola data di dalam antrean. Tidak seperti queue biasa yang bisa berhenti saat penunjuk mencapai ujung array, antrean ini dirancang agar penunjuk **head** dan **tail** dapat berputar kembali ke awal array dengan bantuan operasi modulus (%), sehingga ruang kosong yang sudah ditinggalkan dapat digunakan kembali. Variabel *count* berperan penting karena menyimpan jumlah elemen yang sedang ada di dalam queue, nilainya akan bertambah saat proses *enqueue* dan berkurang saat proses *dequeue*. Dengan cara ini, program dapat dengan mudah mengetahui apakah antrean dalam keadaan penuh atau kosong tanpa perlu perhitungan tambahan yang rumit. Pada file *main.cpp*, dilakukan simulasi sederhana dengan memasukkan beberapa angka, menghapus satu elemen, lalu menambahkan elemen baru dan mengeluarkan sisanya, yang menunjukkan bahwa mekanisme antrean melingkar berjalan dengan baik dan konsisten.

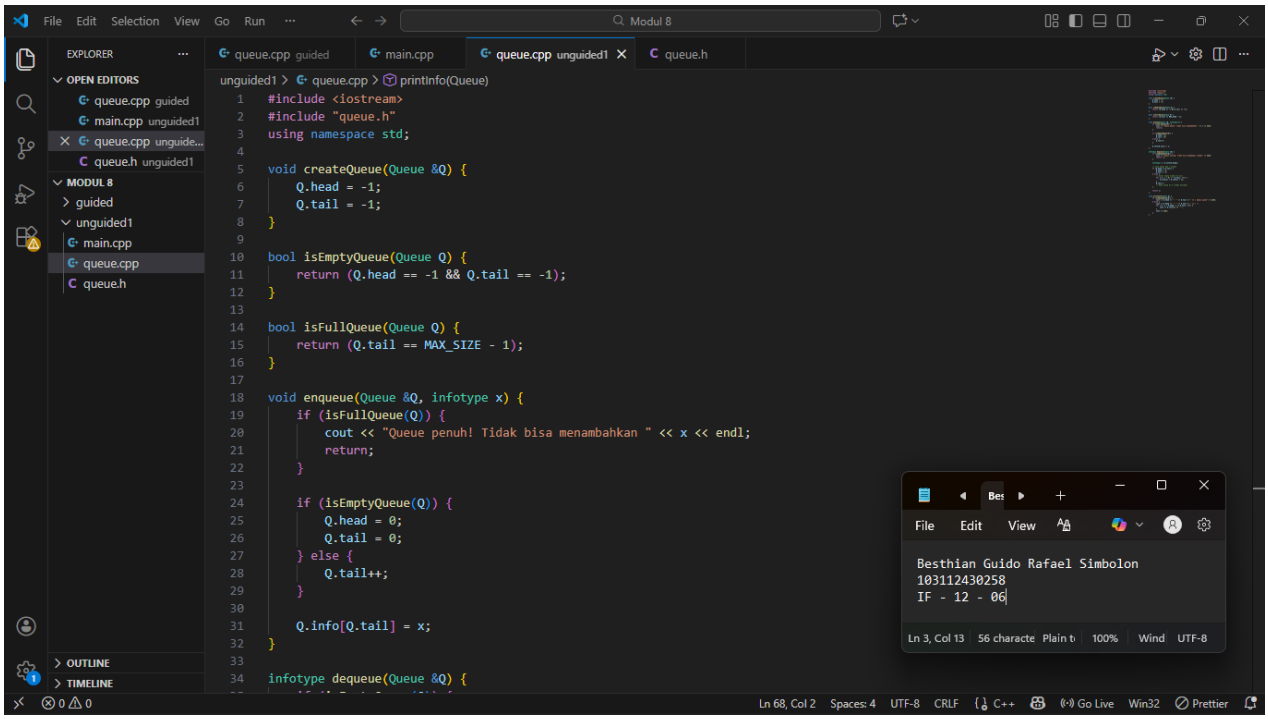
C. Unguided

Unguided 1

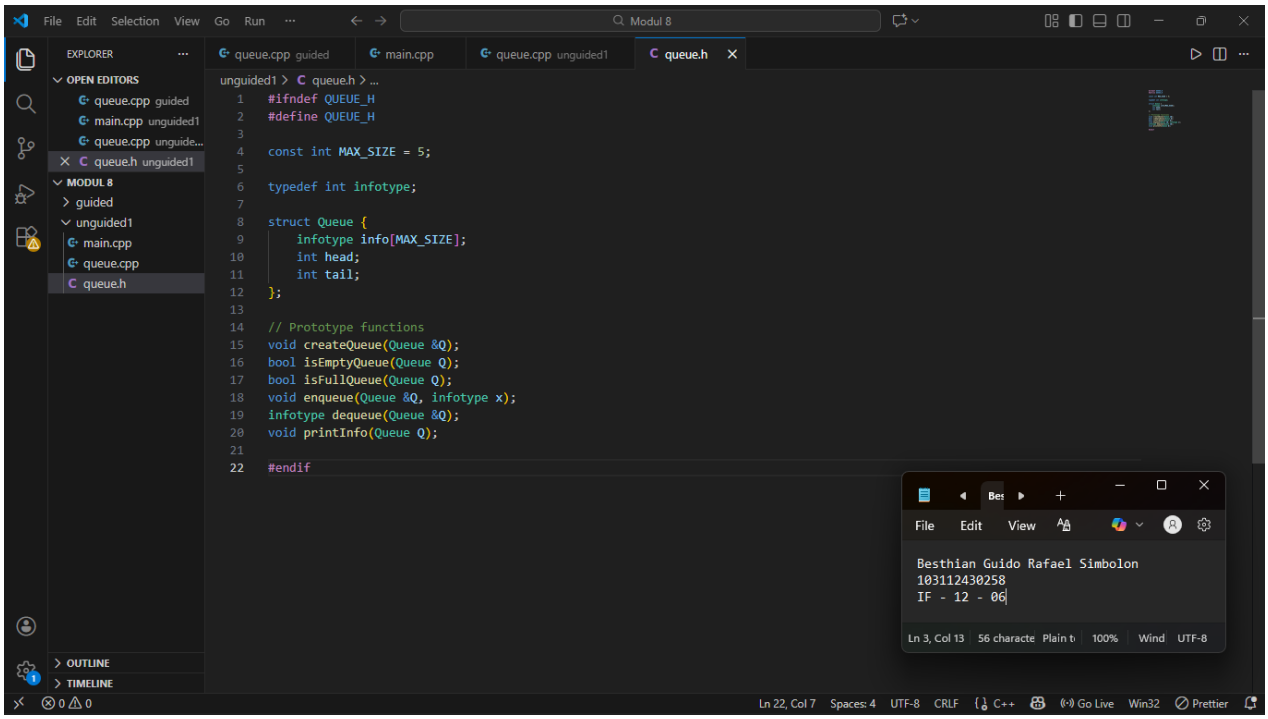
Main.cpp



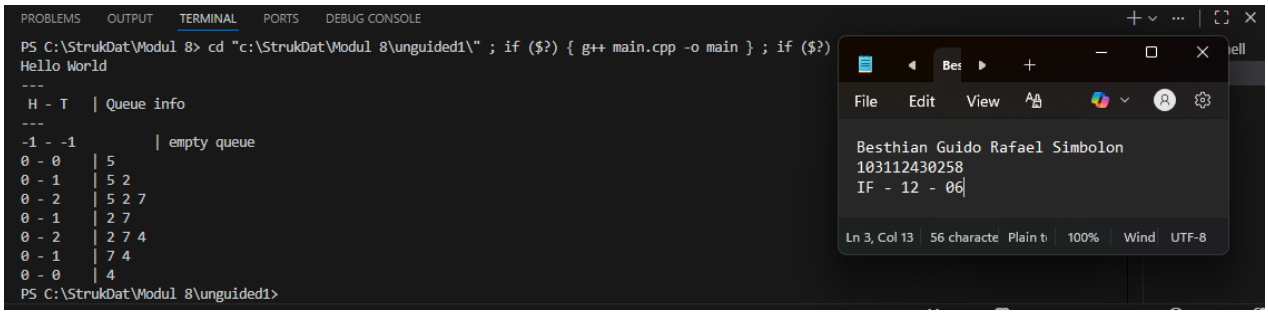
Queue.cpp



Queue.h



Screenshot Output

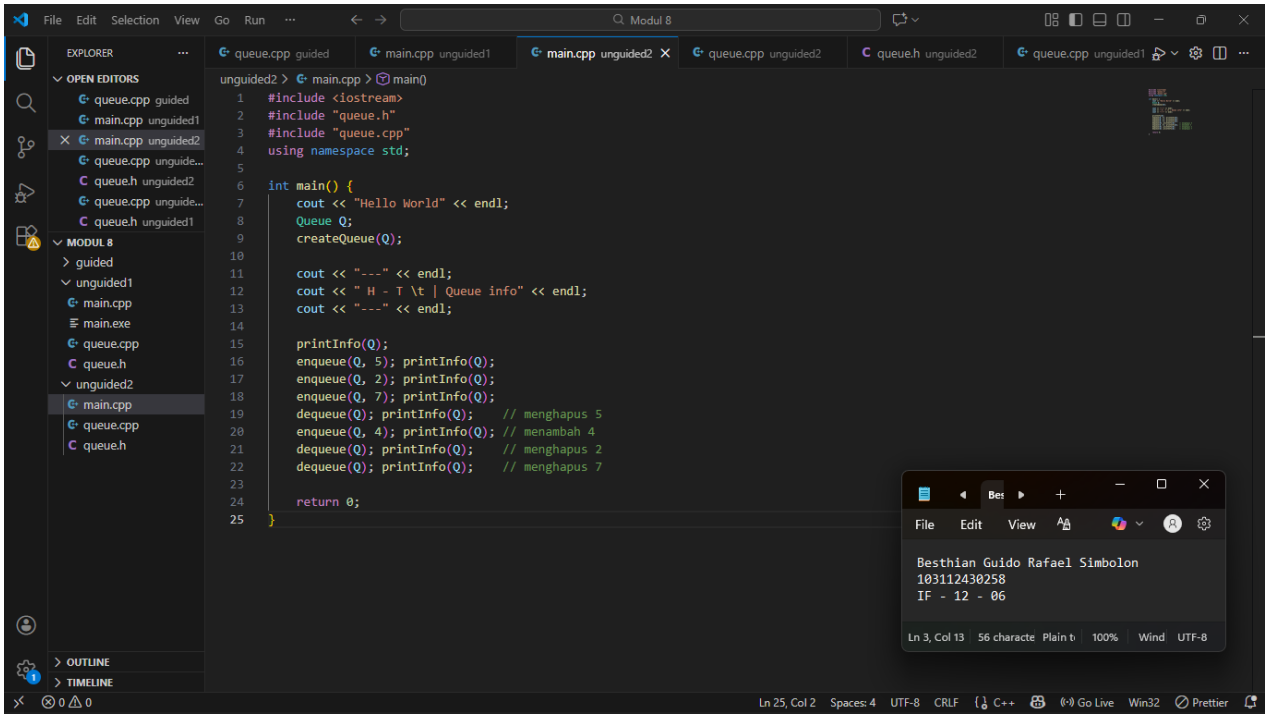


Deskripsi:

Pada program ini, posisi elemen terdepan antrean selalu berada di indeks pertama (indeks 0). Jika dianalogikan dengan antrean di kasir, saat orang paling depan selesai dilayani dan keluar dari antrean, orang-orang di belakangnya harus maju ke depan satu per satu. Hal yang sama terjadi di program: ketika data dikeluarkan dari antrean (dequeue), sistem harus memindahkan semua data yang tersisa agar tetap berurutan mulai dari indeks awal. Proses pemindahan ini dilakukan menggunakan perulangan, sehingga membutuhkan waktu dan kerja tambahan dari komputer. Meskipun konsepnya sederhana dan mudah dipahami, cara ini kurang efisien jika jumlah data dalam antrean cukup banyak..

Unguide 2

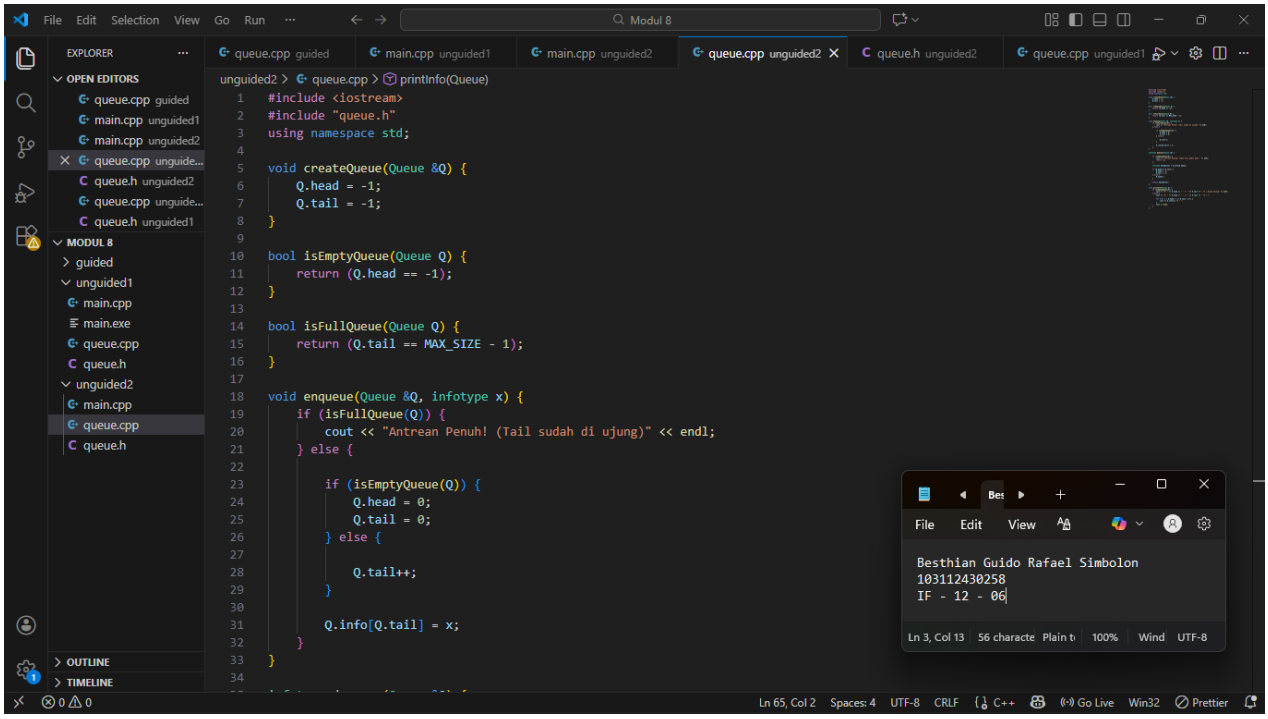
Main.cpp



```cpp
#include <iostream>
#include "queue.h"
#include "queue.cpp"
using namespace std;

int main() {
    cout << "Hello World" << endl;
    Queue Q;
    createQueue(Q);

    cout << "---" << endl;
    cout << " H - T \t | Queue info" << endl;
    cout << "---" << endl;

    printInfo(Q);
    enqueue(Q, 5); printInfo(Q);
    enqueue(Q, 2); printInfo(Q);
    enqueue(Q, 7); printInfo(Q);
    dequeue(Q); printInfo(Q);    // menghapus 5
    enqueue(Q, 4); printInfo(Q); // menambah 4
    dequeue(Q); printInfo(Q);    // menghapus 2
    dequeue(Q); printInfo(Q);    // menghapus 7

    return 0;
}
```

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
```

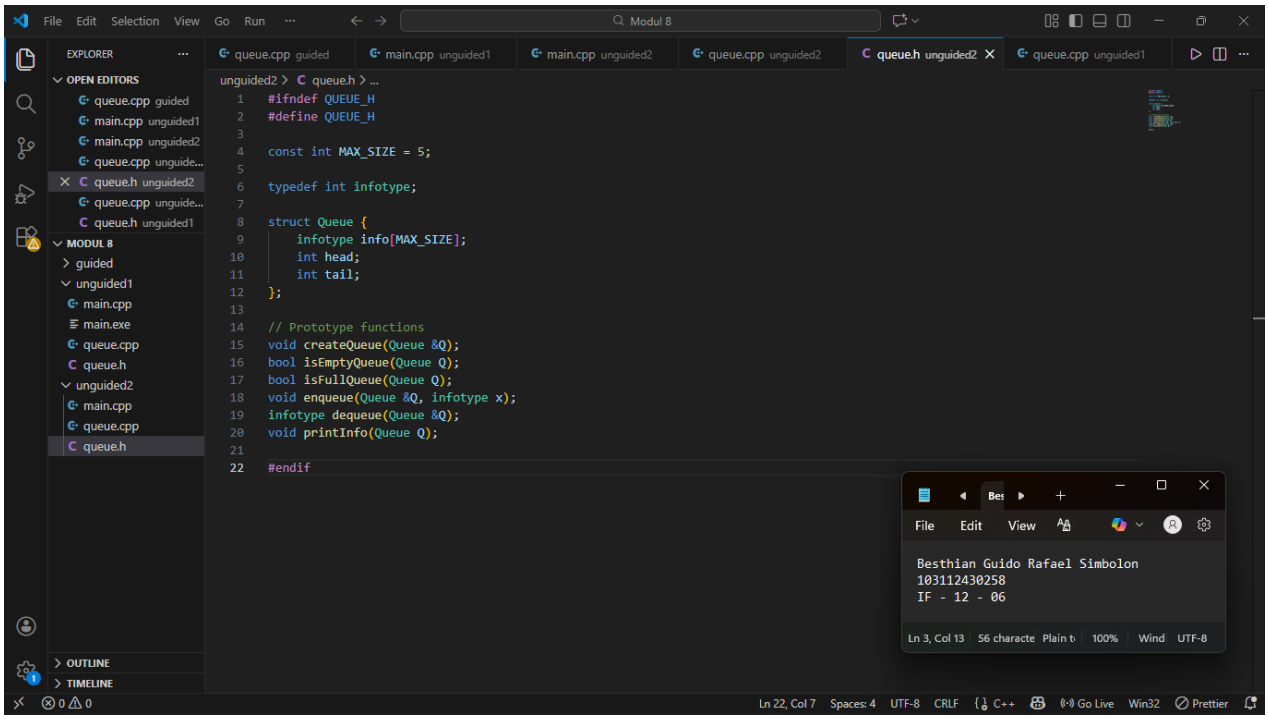
Queue.cpp



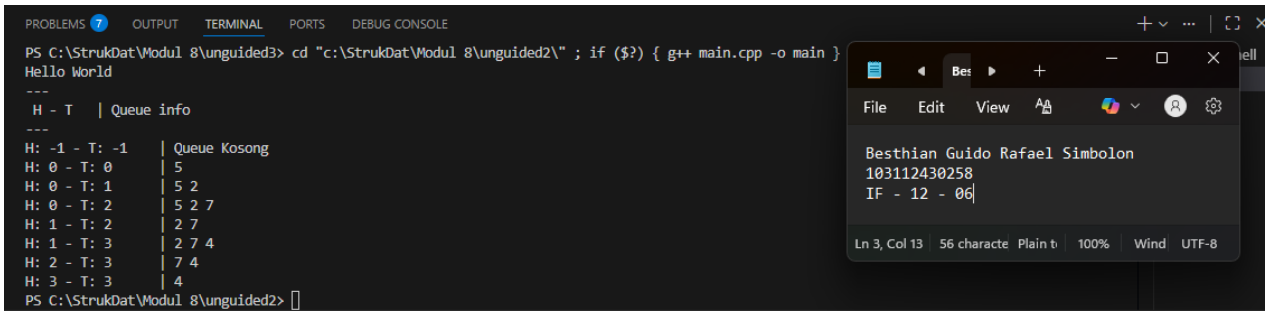
```cpp
#include <iostream>
#include "queue.h"
using namespace std;

void createQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1);
}

bool isFullQueue(Queue Q) {
    return (Q.tail == MAX_SIZE - 1);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Antrean Penuh! (Tail sudah di ujung)" << endl;
    } else {

        if (isEmptyQueue(Q)) {
            Q.head = 0;
            Q.tail = 0;
        } else {

            Q.tail++;
        }

        Q.info[Q.tail] = x;
    }
}
```

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
```

Queue.h



ScreenShoot Output :

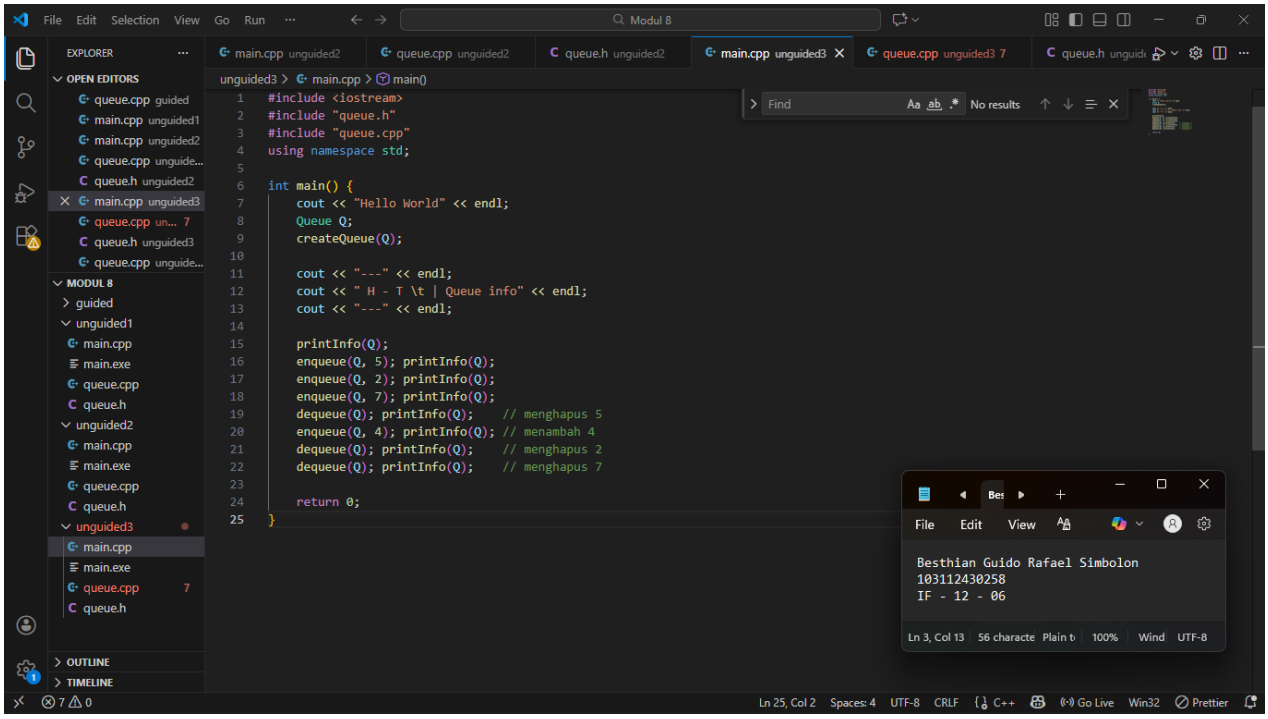


Deskripsi:

Implementasi ini dibuat untuk menghindari proses penggeseran data seperti pada metode sebelumnya, sehingga posisi depan (head) dan belakang (tail) antrean sama-sama bergerak maju. Jika dianalogikan dengan deretan kursi, ketika orang di kursi paling depan pergi, petugas cukup melayani orang di kursi berikutnya tanpa membuat semua orang bergeser. Cara ini membuat proses antrean menjadi lebih cepat dan efisien, namun memiliki kelemahan berupa kondisi *penuh semu*, yaitu saat antrean terlihat penuh karena posisi tail sudah mencapai batas akhir, padahal masih ada ruang kosong di bagian depan yang tidak dapat digunakan kembali kecuali antrean dikosongkan dan diatur ulang.

Unguided 3

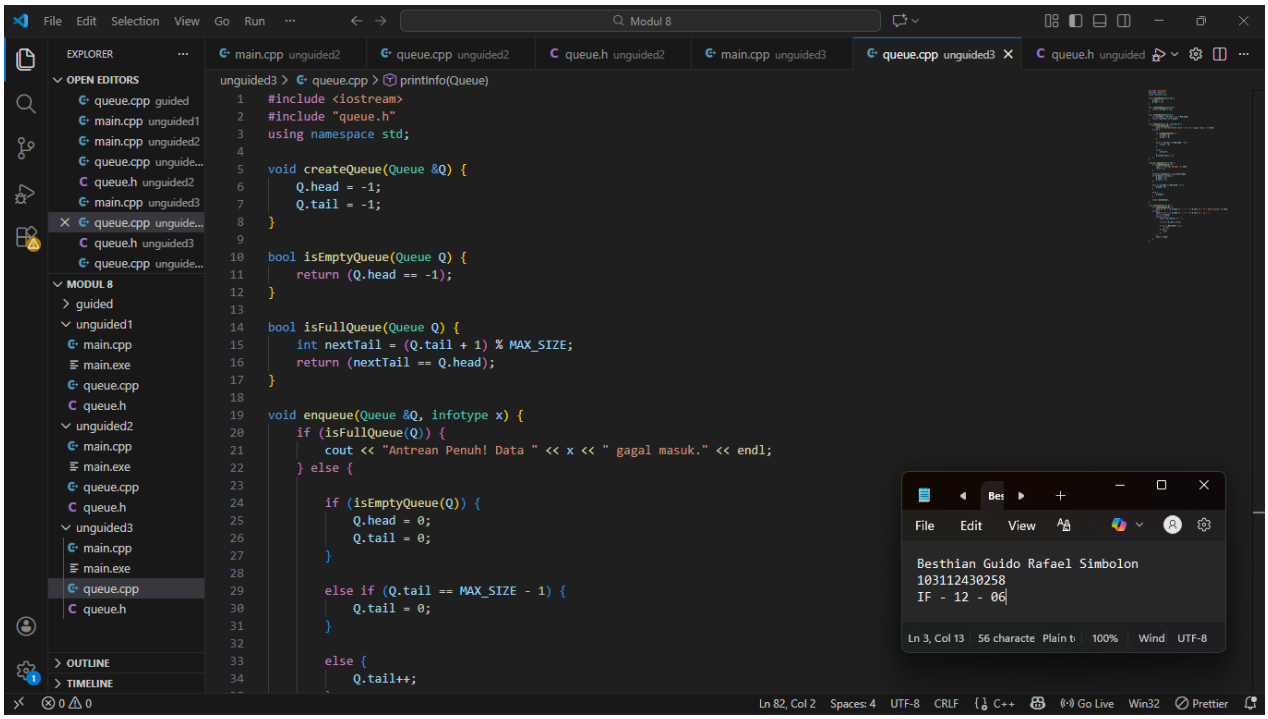
Main.cpp



```cpp
#include <iostream>
#include "queue.h"
#include "queue.cpp"
using namespace std;

int main() {
    cout << "Hello World" << endl;
    Queue Q;
    createQueue(Q);

    cout << "---" << endl;
    cout << " H - T \t | Queue info" << endl;
    cout << "---" << endl;

    printInfo(Q);
    enqueue(Q, 5); printInfo(Q);
    enqueue(Q, 2); printInfo(Q);
    enqueue(Q, 7); printInfo(Q);
    dequeue(Q); printInfo(Q);     // menghapus 5
    enqueue(Q, 4); printInfo(Q); // menambah 4
    dequeue(Q); printInfo(Q);     // menghapus 2
    dequeue(Q); printInfo(Q);     // menghapus 7

    return 0;
}
```

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
```

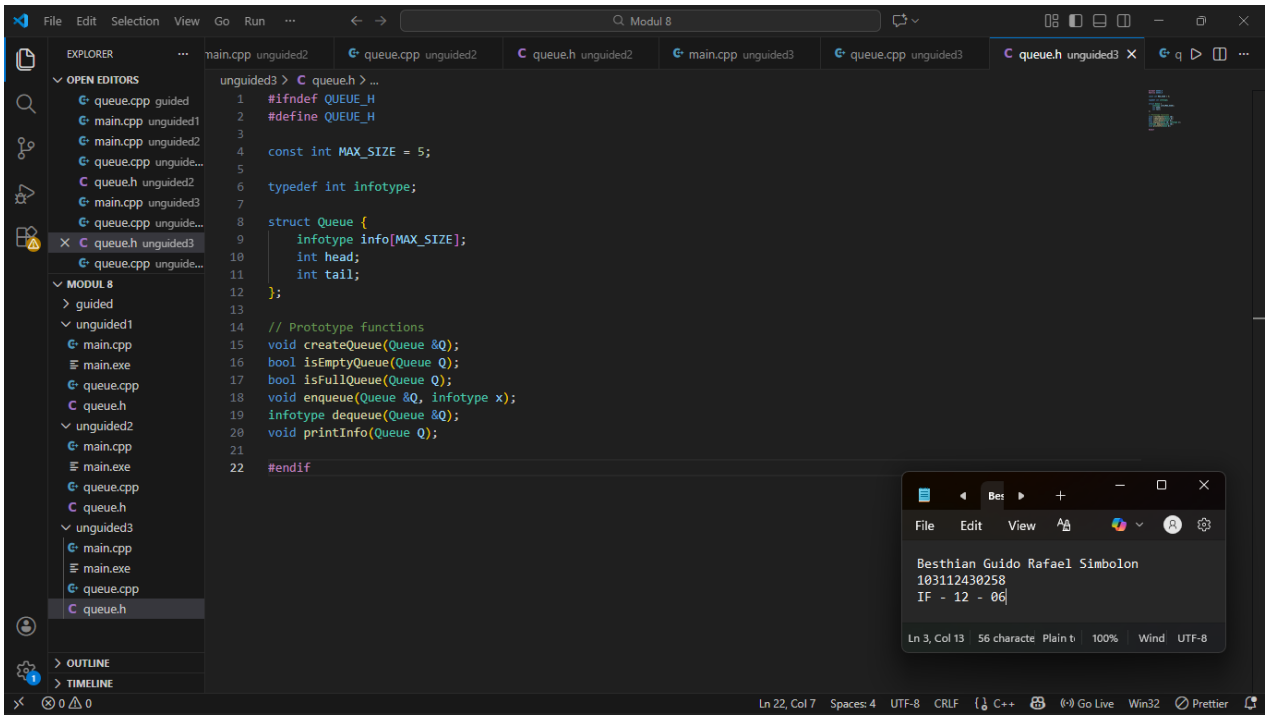
Queue.cpp



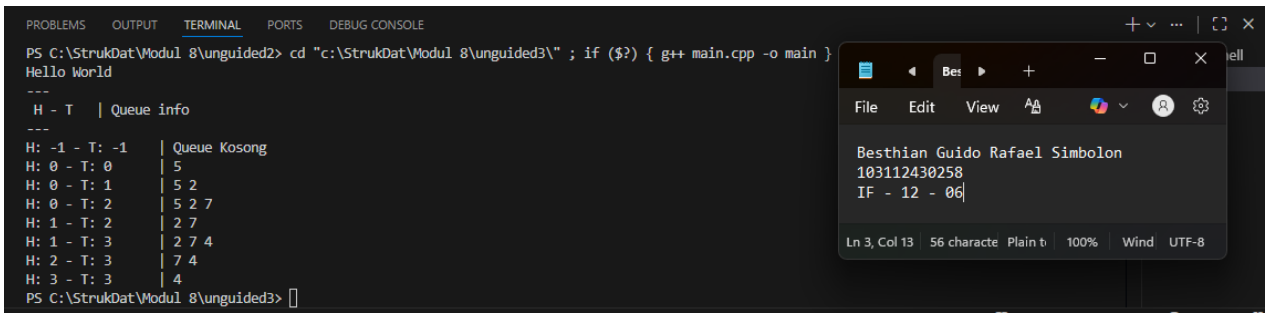
```cpp
#include <iostream>
#include "queue.h"
using namespace std;

void createQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1);
}

bool isFullQueue(Queue Q) {
    int nextTail = (Q.tail + 1) % MAX_SIZE;
    return (nextTail == Q.head);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Antrean Penuh! Data " << x << " gagal masuk." << endl;
    } else {

        if (isEmptyQueue(Q)) {
            Q.head = 0;
            Q.tail = 0;
        }

        else if (Q.tail == MAX_SIZE - 1) {
            Q.tail = 0;
        }

        else {
            Q.tail++;
```

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
```

Queue.h



Screenshot Output



Deskripsi:

Metode ini merupakan pendekatan yang paling efisien karena menghubungkan bagian belakang antrean dengan bagian depannya sehingga membentuk antrean melingkar. Konsepnya mirip dengan metode sebelumnya, tetapi ketika posisi tail sudah mencapai indeks terakhir dan masih terdapat ruang kosong di bagian awal array, data baru akan kembali mengisi posisi kosong tersebut. Dengan mekanisme ini, seluruh ruang memori dapat dimanfaatkan secara maksimal tanpa perlu menggeser data, sehingga proses antrean menjadi lebih cepat dan efektif.

D. Kesimpulan

Berdasarkan praktikum Modul 8 mengenai **Queue** yang telah dilaksanakan, dapat ditarik beberapa kesimpulan penting. Praktikum ini menunjukkan bahwa struktur data Queue bekerja sesuai dengan konsep **FIFO (First In First Out)**, di mana elemen yang pertama kali dimasukkan ke dalam antrean akan menjadi elemen pertama yang dikeluarkan.

Selain itu, telah diuji tiga metode implementasi Queue yang memiliki karakteristik berbeda. Metode geser memerlukan proses pemindahan data setiap kali terjadi penghapusan elemen sehingga kurang efisien dari sisi kinerja. Metode pointer maju menawarkan proses yang lebih cepat, namun menyebabkan pemborosan ruang memori karena sebagian indeks array tidak dimanfaatkan meskipun antrean sudah dianggap penuh. Sementara itu, metode

Circular Queue menjadi solusi paling optimal karena memungkinkan penggunaan seluruh kapasitas array secara maksimal dengan menerapkan konsep indeks melingkar menggunakan operasi modulus atau pengaturan ulang indeks.

Penggunaan variabel tambahan berupa `count` juga terbukti mempermudah proses pengecekan kondisi antrean, baik dalam keadaan kosong maupun penuh, tanpa harus membandingkan posisi *head* dan *tail*. Oleh karena itu, dapat disimpulkan bahwa **Circular Queue** merupakan pendekatan yang paling efektif untuk pengelolaan antrean dengan alokasi memori tetap berbasis array.

E. Referensi

Modul Praktikum Struktur Data. (2023). *Queue dan Variasinya*. Program Studi Teknik Informatika, Fakultas Ilmu Komputer.

Munir, R. (2010). *Algoritma dan Pemrograman*. Bandung: Informatika Bandung.

Nugroho, B. (2017). *Dasar Pemrograman dan Struktur Data*. Yogyakarta: Gava Media.

Suryadi, & Rahayu, S. (2018). *Struktur Data*. Jakarta: Erlangga.