LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII

QUEUE

Disusun Oleh :
NAMA : Muhammad Fachri Auravyano Saka
NIM : 103112430180

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

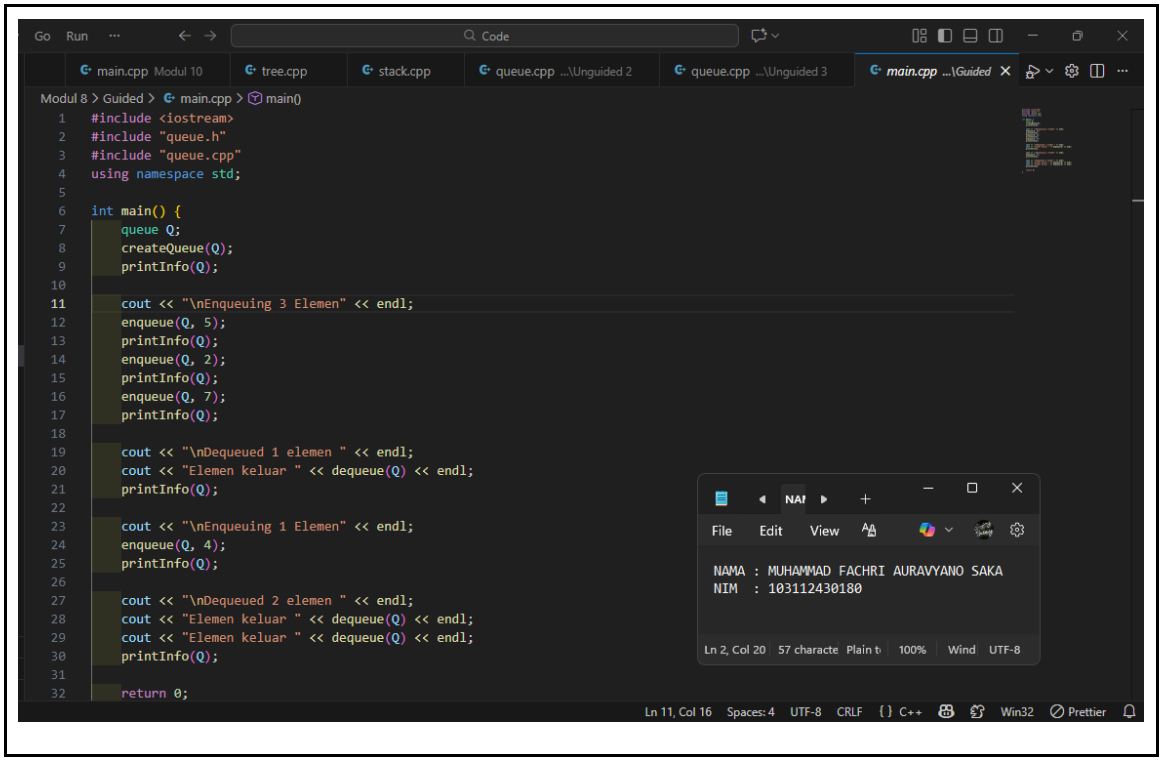
Queue atau antrean adalah struktur data yang bekerja berdasarkan prinsip FIFO (First In, First Out). Konsep ini sangat mirip dengan antrean di dunia nyata, misalnya saat mengantre membeli tiket bioskop: orang yang datang paling awal (masuk pertama) akan dilayani dan keluar paling awal juga. Sebaliknya, orang yang datang terakhir harus menunggu di posisi paling belakang.

Dalam pemrograman (khususnya C++ menggunakan Array), Queue memiliki dua pintu utama: Head (depan) untuk data keluar, dan Tail (belakang) untuk data masuk. Operasi memasukkan data disebut Enqueue, sedangkan mengeluarkan data disebut Dequeue. Tantangan utama menggunakan Array untuk Queue adalah ukuran memori yang tetap.

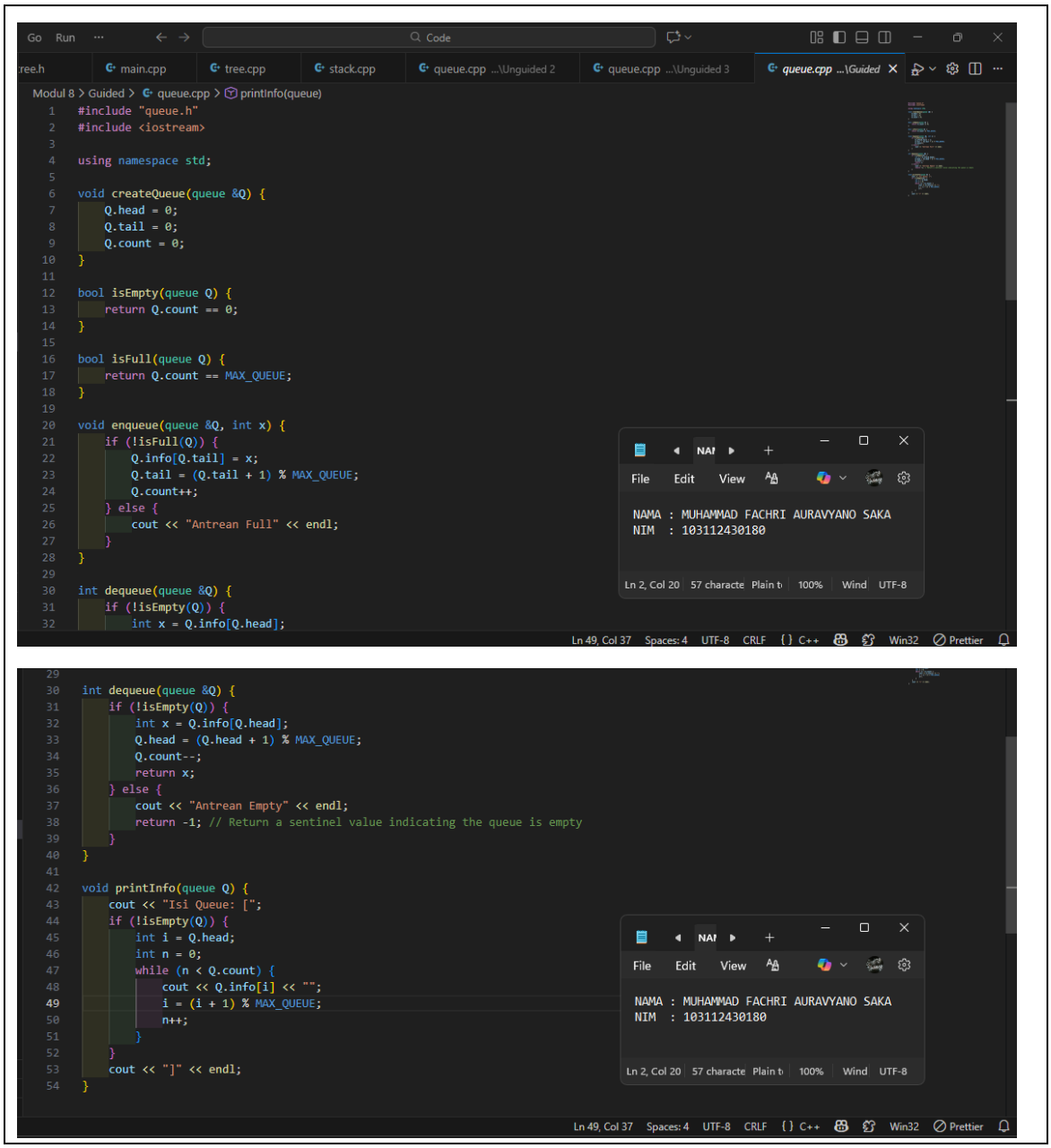
B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

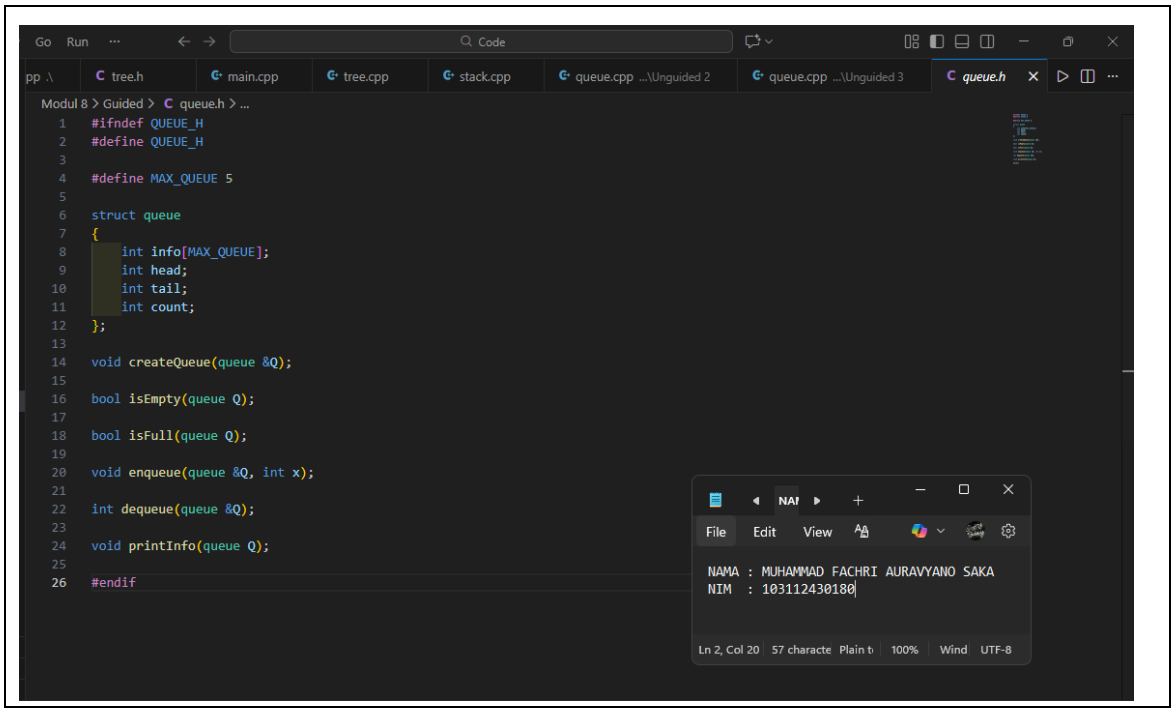
Main.cpp



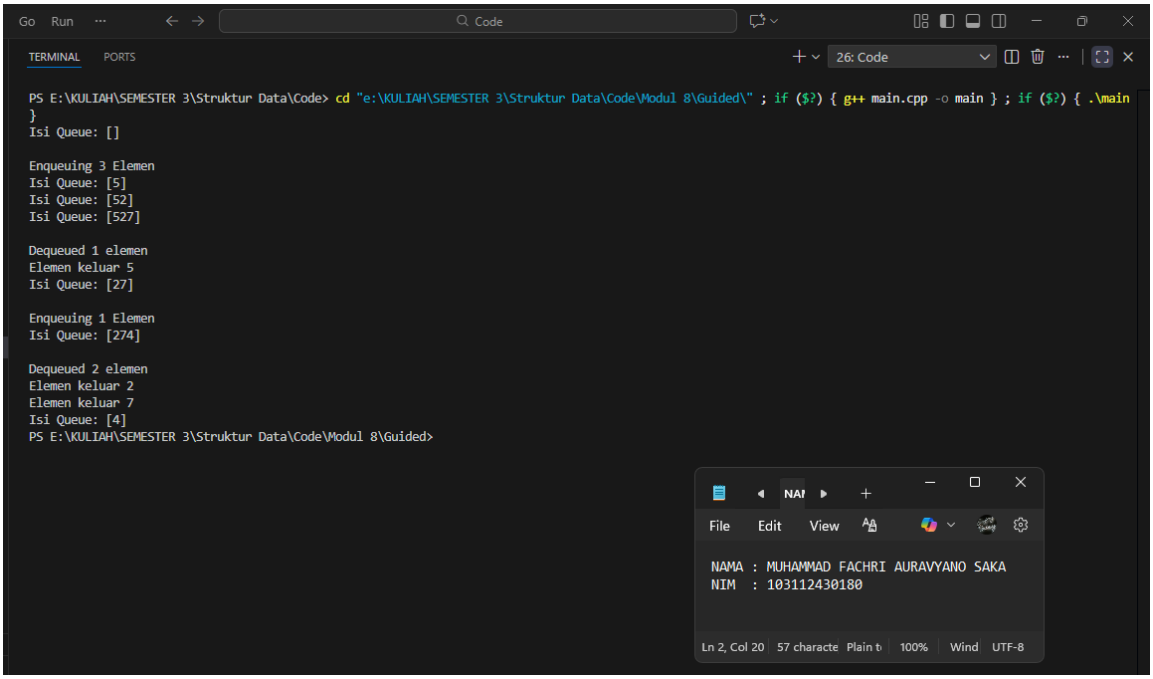
Queue.cpp


Go Run
Code
main.cpp
tree.cpp
stack.cpp
queue.cpp ...\Unguided 2
queue.cpp ...\Unguided 3
queue.cpp ...\Guided
Modul 8 > Guided > 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];
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8
Ln 49, Col 37 Spaces: 4 UTF-8 CRLF { } C++ Win32 Prettier
int dequeue(queue &Q) {
if (!isEmpty(Q)) {
int x = Q.info[Q.head];
Q.head = (Q.head + 1) % MAX_QUEUE;
Q.count--;
return x;
} else {
cout << "Antrean Empty" << endl;
return -1; // Return a sentinel value indicating the queue is empty
}
}
void printInfo(queue Q) {
cout << "Isi Queue: [";
if (!isEmpty(Q)) {
int i = Q.head;
int n = 0;
while (n < Q.count) {
cout << Q.info[i] << " ";
i = (i + 1) % MAX_QUEUE;
n++;
}
}
cout << "]" << endl;
}
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8
Ln 49, Col 37 Spaces: 4 UTF-8 CRLF { } C++ Win32 Prettier

Queue.h



Screenshots Output



Deskripsi:

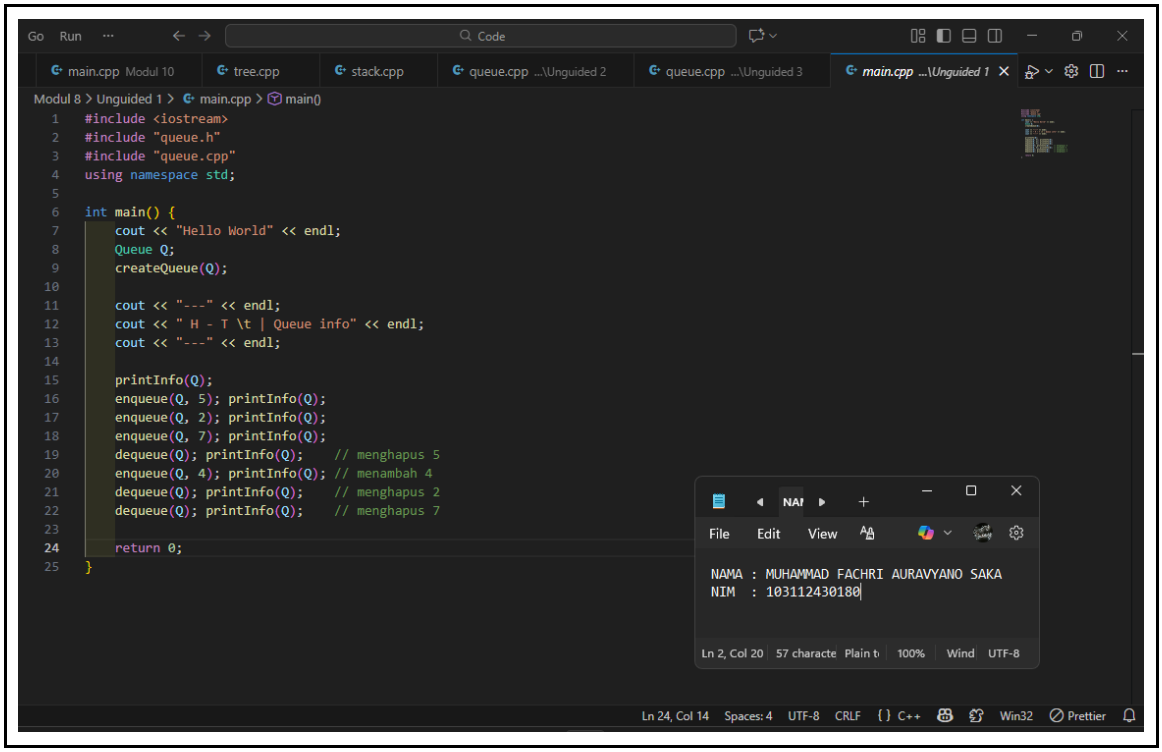
Program ini mengimplementasikan Circular Queue (Antrean Melingkar) menggunakan bantuan variabel penghitung (count). Berbeda dengan antrean biasa yang bisa "mentok" di ujung, program ini menggunakan operasi matematika modulus (%) pada bagian head dan tail. Artinya, jika antrean sudah sampai di posisi terakhir array, penunjuk akan otomatis berputar kembali ke posisi awal (indeks 0) untuk mengisi ruang kosong. Keunikan kode ini terletak pada variabel count, yang berfungsi mencatat jumlah data

secara langsung setiap kali ada yang masuk (enqueue) atau keluar (dequeue). Dengan adanya count, komputer dapat dengan sangat mudah membedakan kondisi antrean penuh atau kosong tanpa perlu menghitung jarak antara depan dan belakang secara manual. File main.cpp kemudian menjalankan simulasi sederhana: memasukkan angka (5, 2, 7), mengeluarkan satu angka, memasukkan angka baru (4), dan mengeluarkan sisanya untuk membuktikan bahwa antrean berjalan lancar secara memutar.

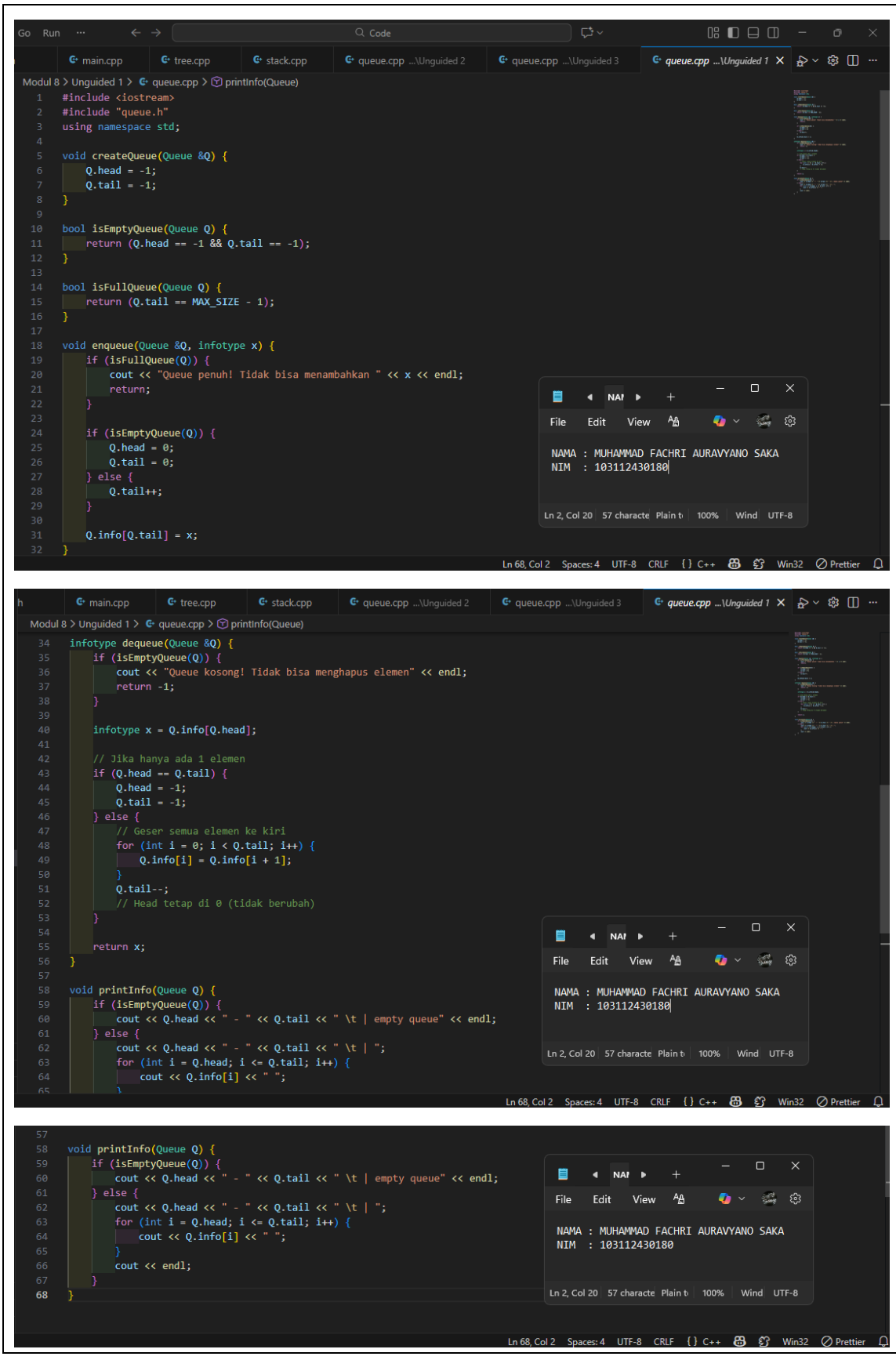
C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

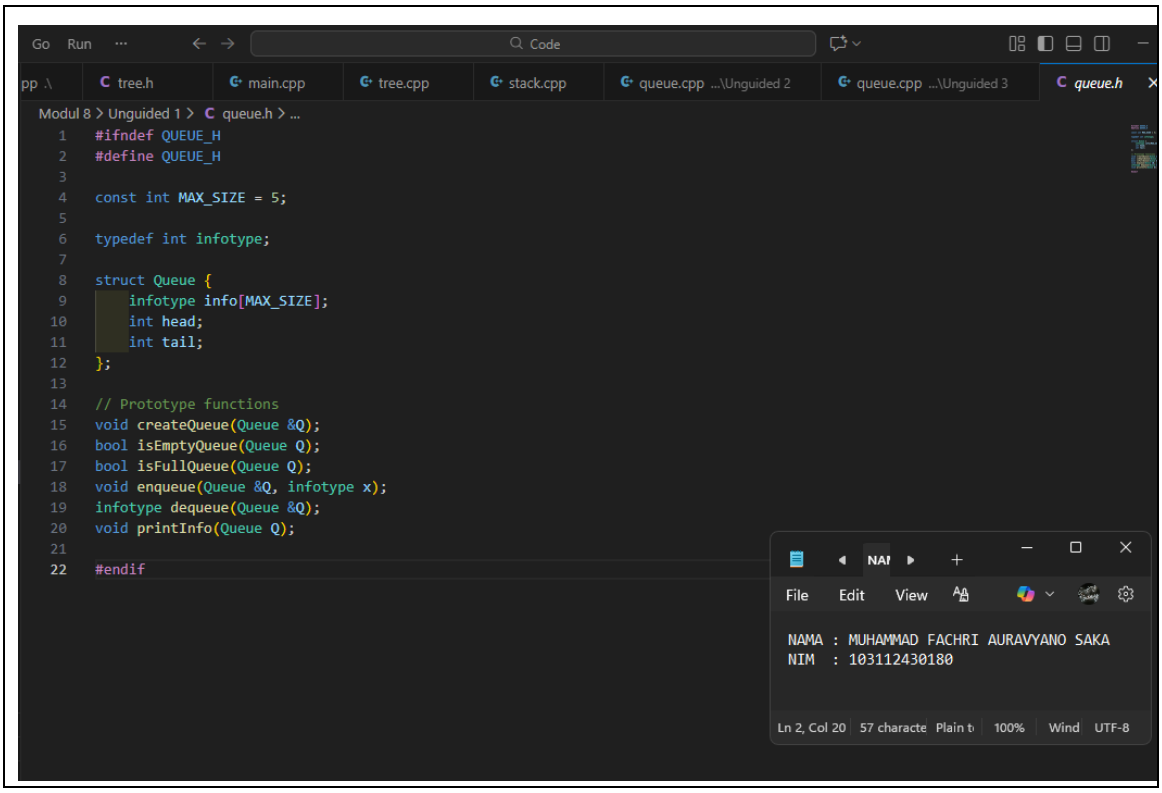
Main.cpp



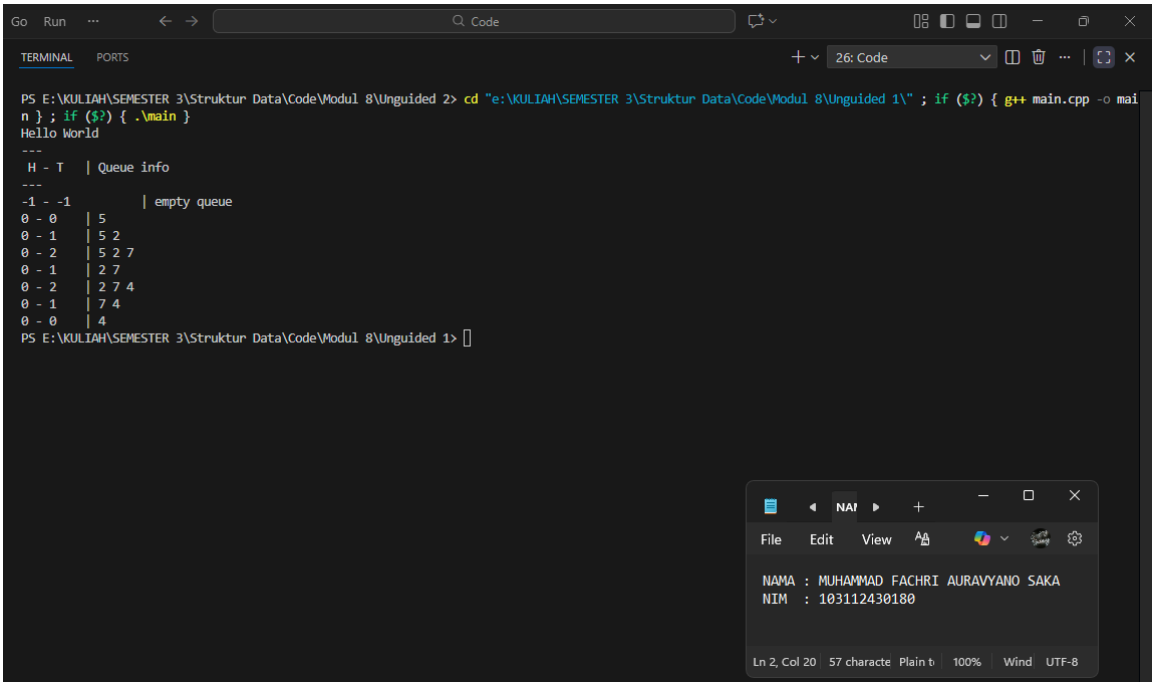
queue.cpp


main.cpp
tree.cpp
stack.cpp
queue.cpp ...\Unguided 2
queue.cpp ...\Unguided 3
queue.cpp ...\Unguided 1
Modul 8 > Unguided 1 > queue.cpp > printInfo(Queue)
#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 && Q.tail == -1);
}
bool isFullQueue(Queue Q) {
return (Q.tail == MAX_SIZE - 1);
}
void enqueue(Queue &Q, infotype x) {
if (isFullQueue(Q)) {
cout << "Queue penuh! Tidak bisa menambahkan " << x << endl;
return;
}
if (isEmptyQueue(Q)) {
Q.head = 0;
Q.tail = 0;
} else {
Q.tail++;
}
Q.info[Q.tail] = x;
}
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
infotype dequeue(Queue &Q) {
if (isEmptyQueue(Q)) {
cout << "Queue kosong! Tidak bisa menghapus elemen" << endl;
return -1;
}
infotype x = Q.info[Q.head];
// Jika hanya ada 1 elemen
if (Q.head == Q.tail) {
Q.head = -1;
Q.tail = -1;
} else {
// Geser semua elemen ke kiri
for (int i = 0; i < Q.tail; i++) {
Q.info[i] = Q.info[i + 1];
}
Q.tail--;
// Head tetap di 0 (tidak berubah)
}
return x;
}
void printInfo(Queue Q) {
if (isEmptyQueue(Q)) {
cout << Q.head << " - " << Q.tail << " \t | empty queue" << endl;
} else {
cout << Q.head << " - " << Q.tail << " \t | ";
for (int i = Q.head; i <= Q.tail; i++) {
cout << Q.info[i] << " ";
}
cout << endl;
}
}
Ln 68, Col 2 Spaces: 4 UTF-8 CRLF C++ Win32 Prettier

queue.h



Screenshot Output

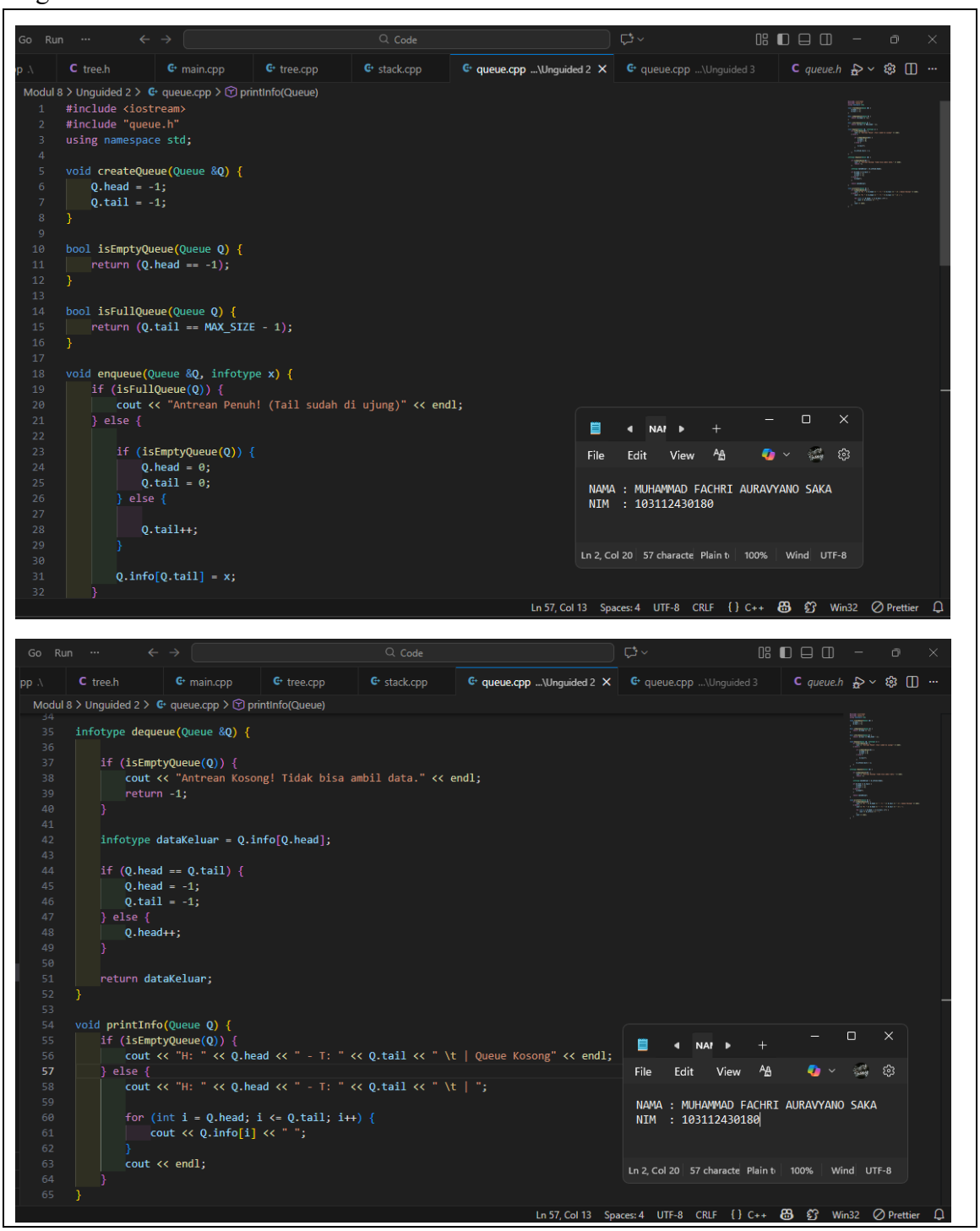


Deskripsi:

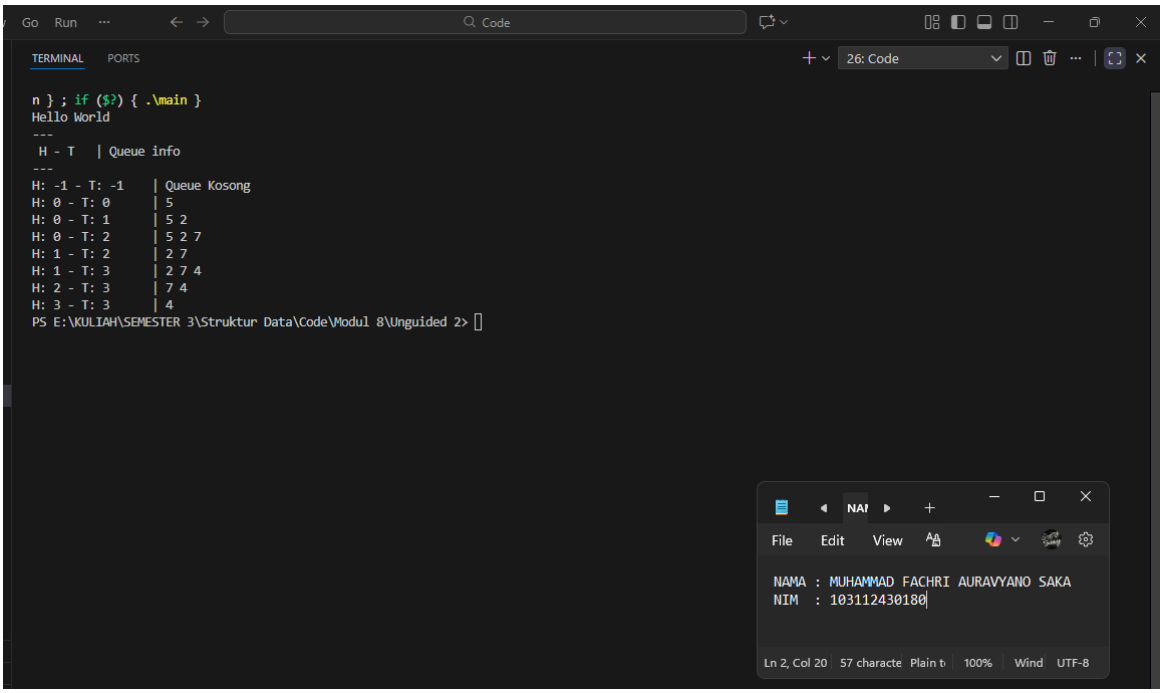
Pada implementasi ini, posisi Head (depan) selalu dikunci di indeks ke-0. Bayangkan antrean orang di kasir; ketika orang paling depan selesai dilayani dan pergi, semua orang di belakangnya harus melangkah maju satu langkah mengisi tempat yang kosong. Dalam bahasa pemrograman, ini berarti setiap kali kita melakukan dequeue (mengeluarkan data),

komputer harus bekerja ekstra untuk menggeser semua data yang tersisa dari indeks belakang ke indeks depannya menggunakan perulangan (looping). Cara ini mudah dibayangkan, tapi kurang efisien karena membebani komputer untuk terus-menerus menggeser data.

Unguided 2



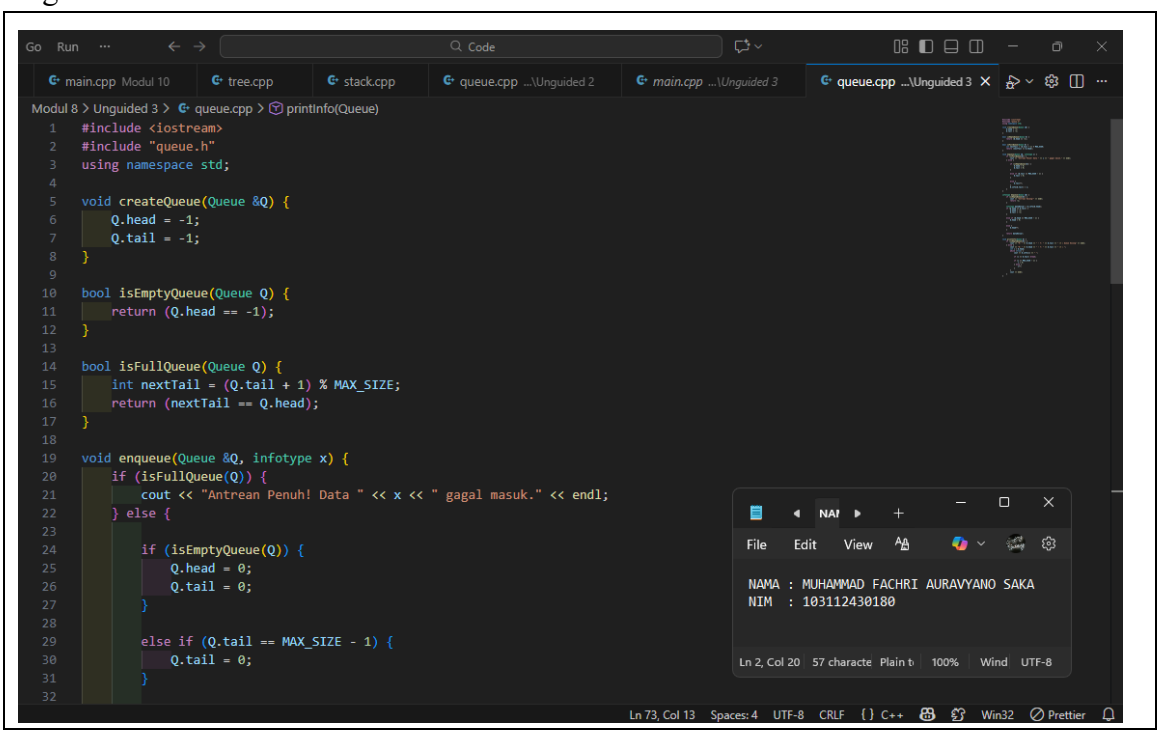
Screenshot Output

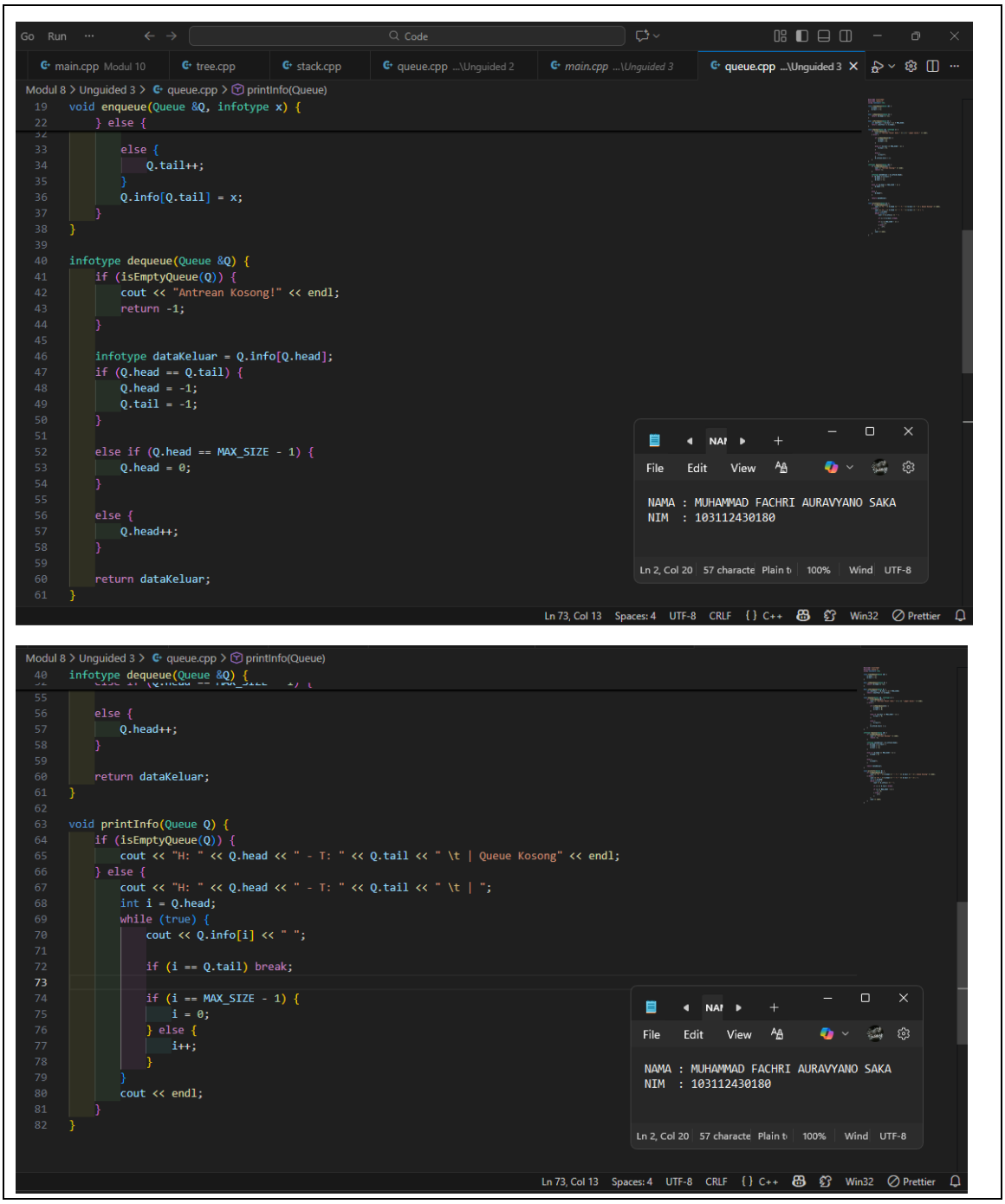


Deskripsi:

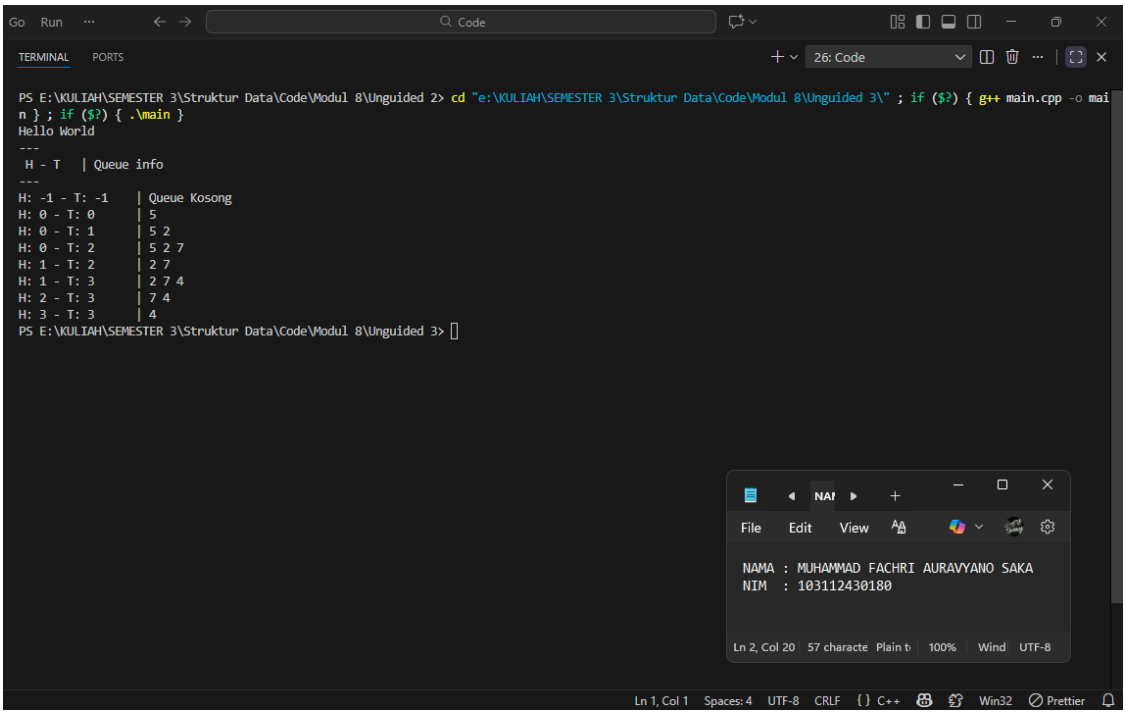
Implementasi ini mencoba mengatasi kelemahan metode pertama agar komputer tidak perlu repot menggeser data. Di sini, Head (depan) dan Tail (belakang) sama-sama bergerak maju. Bayangkan antrean kursi memanjang; ketika orang di kursi nomor 1 pergi, kasir yang pindah melayani orang di kursi nomor 2. Orang-orang dalam antrean tidak perlu bergeser. Prosesnya jadi sangat cepat. Namun, kelemahannya adalah terjadinya "Penuh Semu". Lama-kelamaan, antrean akan mentok di ujung belakang kursi, padahal kursi-kursi di depan yang sudah ditinggalkan kosong dan tidak bisa dipakai lagi (kecuali antrean di-reset total saat kosong).

Unguided 3





Screenshot Output



Deskripsi:

Ini adalah metode yang paling cerdik dan efisien. Konsepnya mirip dengan Alternatif 2, tetapi ujung belakang antrean disambungkan kembali ke ujung depan, seolah-olah antrean berbentuk lingkaran. Jika posisi Tail sudah sampai di ujung array (indeks terakhir) tetapi di bagian depan (indeks 0, 1, dst.) masih ada kursi kosong yang ditinggalkan, maka data baru akan berputar dan masuk ke kursi kosong di depan tersebut. Dengan cara ini, tidak ada memori yang terbuang sia-sia dan komputer tidak perlu lelah menggeser-geser data.

D. Kesimpulan

Berdasarkan praktikum Modul 8 tentang Queue yang telah dilakukan, kita dapat menyimpulkan beberapa hal penting:

1. Pemahaman Prinsip FIFO: Praktikum ini membuktikan bahwa Queue sukses menerapkan prinsip FIFO. Data yang pertama kali dimasukkan (enqueue) terbukti menjadi data yang pertama kali keluar (dequeue) saat program dijalankan.
2. Perbandingan Metode Implementasi: Kita telah mencoba tiga mekanisme berbeda dan melihat kelebihan serta kekurangannya:
    - Alternatif 1 (Geser): Metode ini boros kinerja karena komputer harus melakukan perulangan (*looping*) untuk menggeser data setiap kali ada pengambilan data.
    - Alternatif 2 (Pointer Maju): Metode ini cepat, tetapi boros tempat. Antrean sering dianggap penuh (*Antrean Penuh/Full*) padahal indeks awal array (0, 1, dst) sebenarnya kosong tak terpakai.
    - Alternatif 3 (Circular/Melingkar): Ini adalah metode yang paling efisien.

Dengan menggunakan logika aritmatika modulus (%) atau reset indeks, kita bisa memanfaatkan seluruh kapasitas array tanpa ada ruang kosong yang terbuang sia-sia.

3. Penggunaan Variabel Count: Pada latihan *Guided*, penggunaan variabel tambahan count sangat membantu logika program. Dengan count, kita bisa mengetahui apakah antrean penuh atau kosong secara instan tanpa harus membandingkan posisi Head dan Tail yang rumit. Secara keseluruhan, Circular Queue adalah teknik terbaik untuk menangani antrean dengan alokasi memori tetap (Array).

E. Referensi

Muñoz, D. F. (2024, June). A C++ library for fast simulation of queues and some experimental results. In *AIP Conference Proceedings* (Vol. 3094, No. 1, p. 110002). AIP Publishing LLC.

Goponenko, A., & Carroll, S. (2019). A C++ implementation of a lock-free priority queue based on Multi-Dimensional Linked List. *Link: https://www. researchgate. net/publication/337020321_A_C_Implementation_of_a_Lock-Free _Priority_Queue_Based_on_Multi-Dimensional_Linked_List*.

Malik, D. S. (2010). *Data structures using C++*. USA.