LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL XIV

GRAPH

Disusun Oleh :
Nama : Besthian Guido Rafael Simbolon
NIM : 103112430258

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

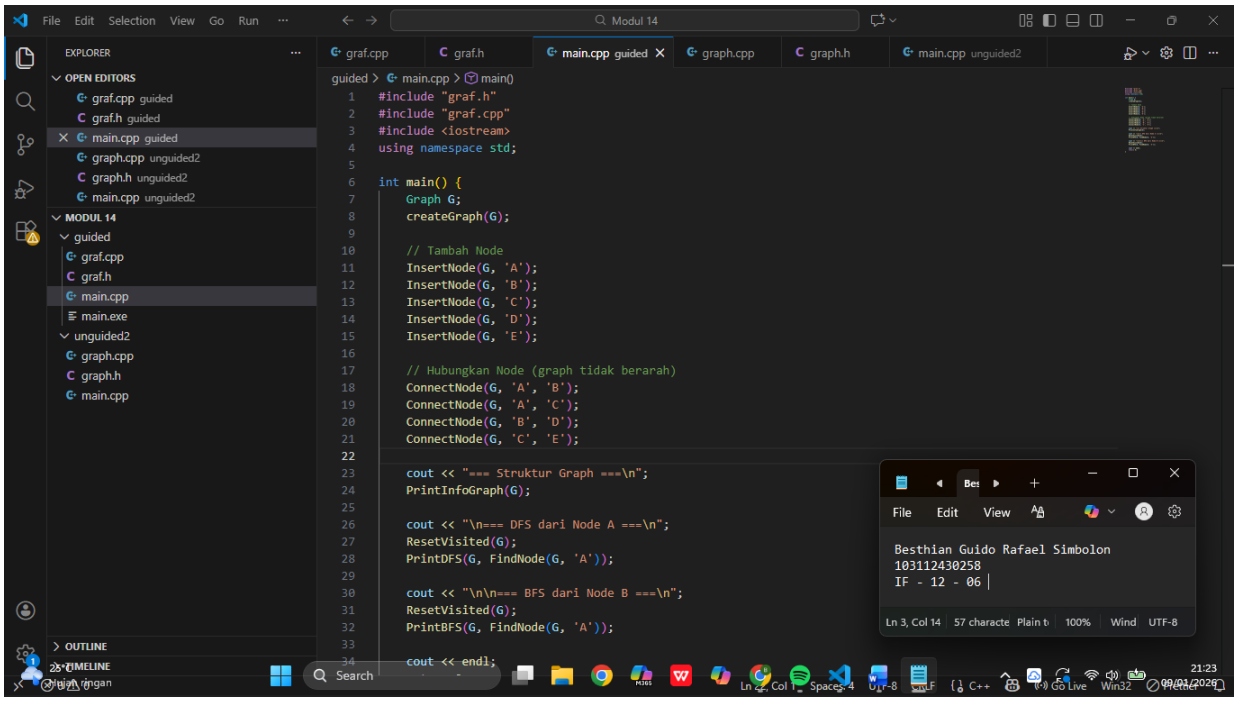
A. Dasar Teori

Graf (Graph) adalah struktur data non-linear yang digunakan untuk menggambarkan hubungan antar objek, yang terdiri dari **simpul (vertex)** dan **sisi (edge)** sebagai penghubung antar simpul. Dalam pemrograman, graf sering direpresentasikan menggunakan **Adjacency List**, yaitu setiap simpul memiliki daftar tetangga yang disimpan dalam bentuk linked list, karena cara ini lebih hemat memori dibandingkan Adjacency Matrix, terutama pada graf yang jumlah sisinya sedikit (sparse). Untuk menelusuri graf, terdapat dua algoritma utama yaitu **Depth First Search (DFS)** dan **Breadth First Search (BFS)**. DFS menelusuri graf secara mendalam dengan mengunjungi satu jalur sampai habis sebelum berpindah ke jalur lain, biasanya menggunakan rekursi atau stack. Sementara itu, BFS menelusuri graf secara bertahap per level dengan mengunjungi semua simpul terdekat terlebih dahulu menggunakan queue. Kedua algoritma ini penting untuk memahami cara menjelajahi dan mengelola data dalam struktur graf.

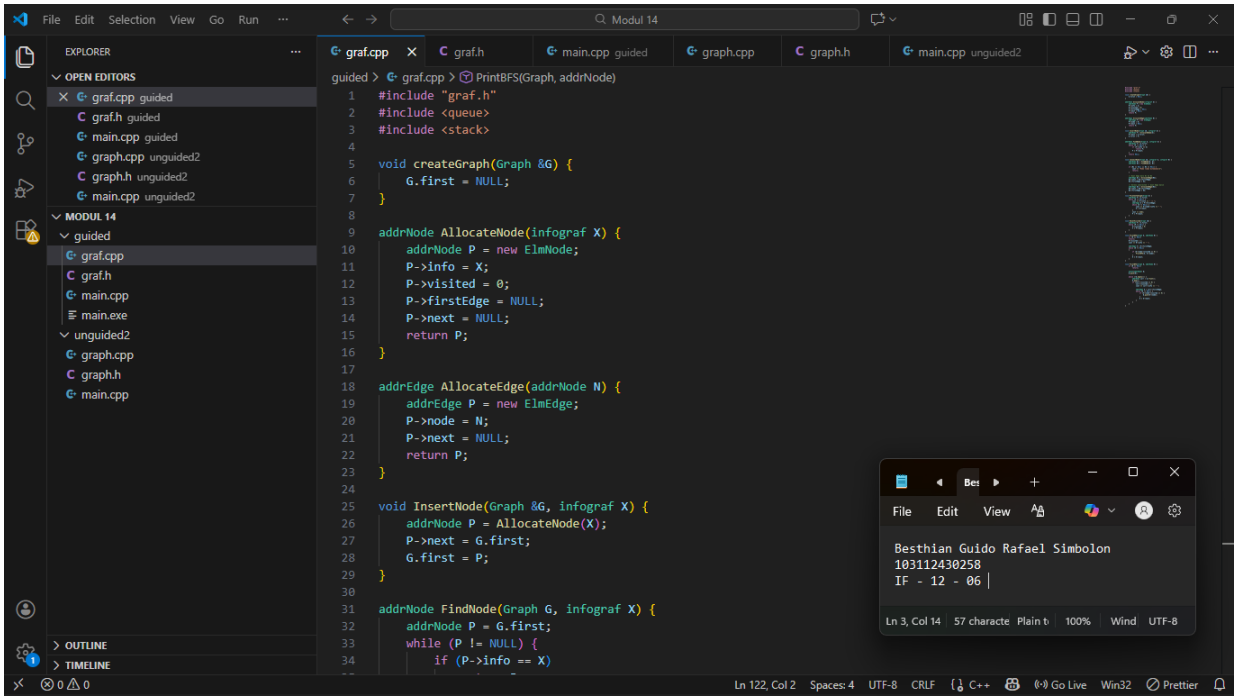
B. Guided

Guided 1

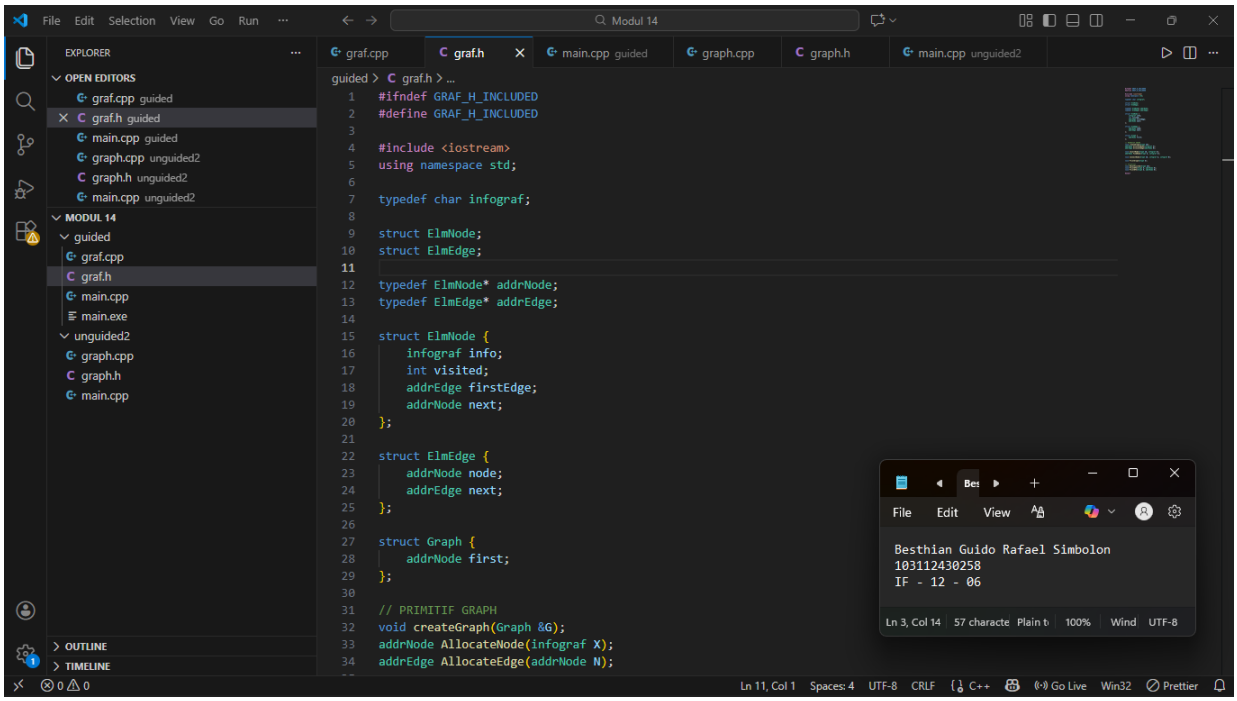
main.cpp



graf.cpp



graf.h



Screenshots Output

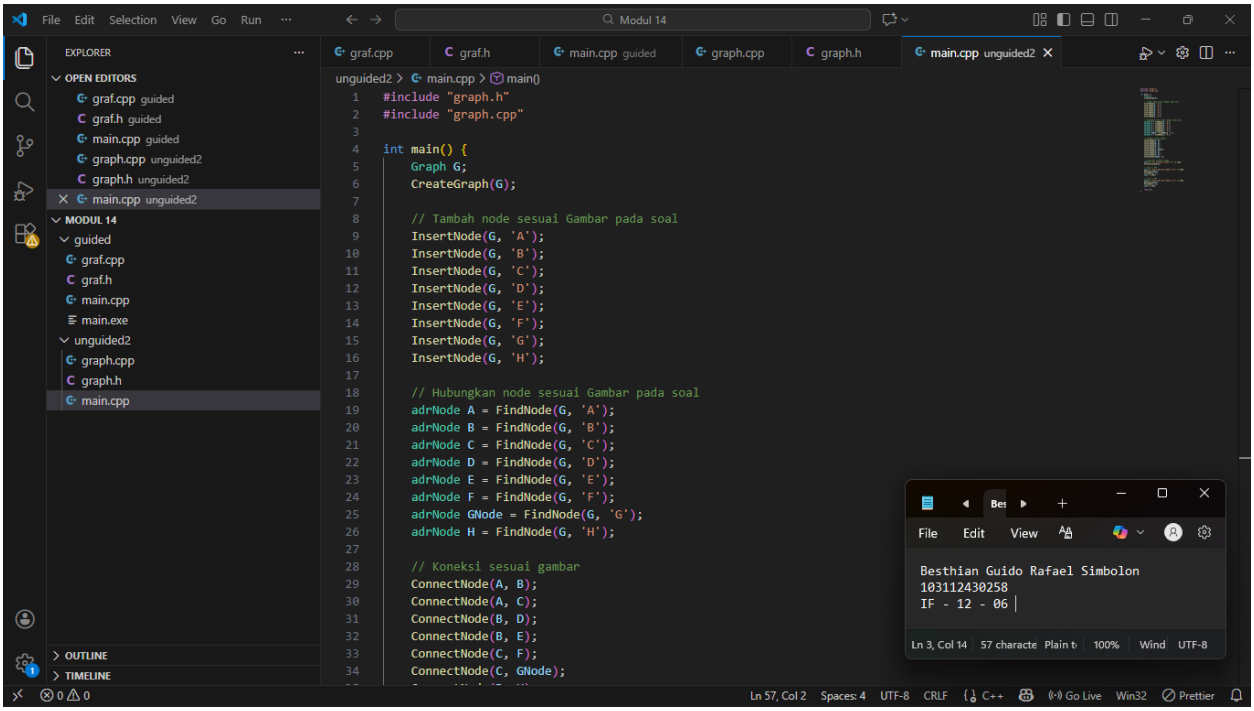
Deskripsi:

Program ini merupakan implementasi struktur data Graph tak berarah (Undirected Graph) menggunakan bahasa C++ dengan representasi adjacency list. Program mendefinisikan tipe data dinamis untuk node (simpul) dan edge (sisi) menggunakan pointer, sehingga setiap node memiliki daftar koneksi ke node-node lain yang terhubung dengannya. Secara fungsional, program memungkinkan pembuatan graph, penambahan node (seperti A, B, C, D, dan E), serta penghubungan antar node tersebut. Setelah struktur graph terbentuk, program menampilkan hubungan atau koneksi antar node secara visual ke layar. Selain itu, program ini juga mendemonstrasikan dua algoritma penelusuran utama, yaitu Depth First Search (DFS) yang melakukan penelusuran secara mendalam dengan menggunakan rekursi, dan Breadth First Search (BFS) yang melakukan penelusuran secara melebar dengan bantuan struktur data queue.

C. Unguided

Unguided 1

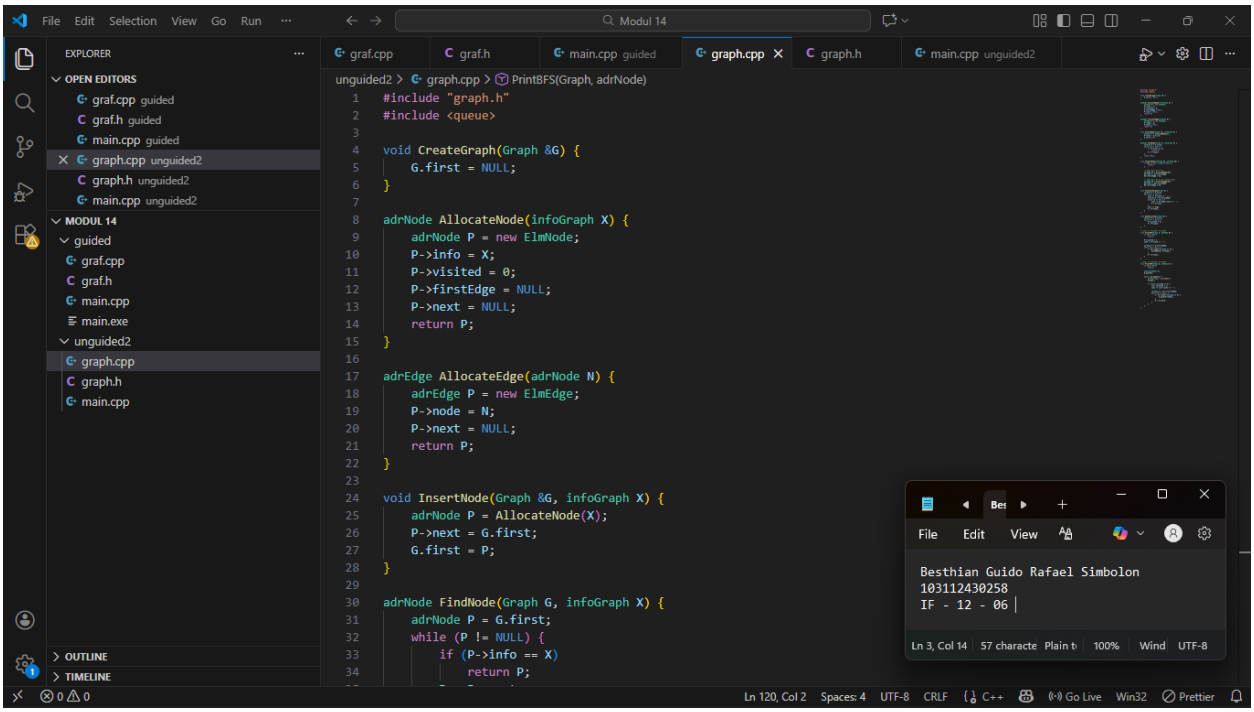
Main.cpp



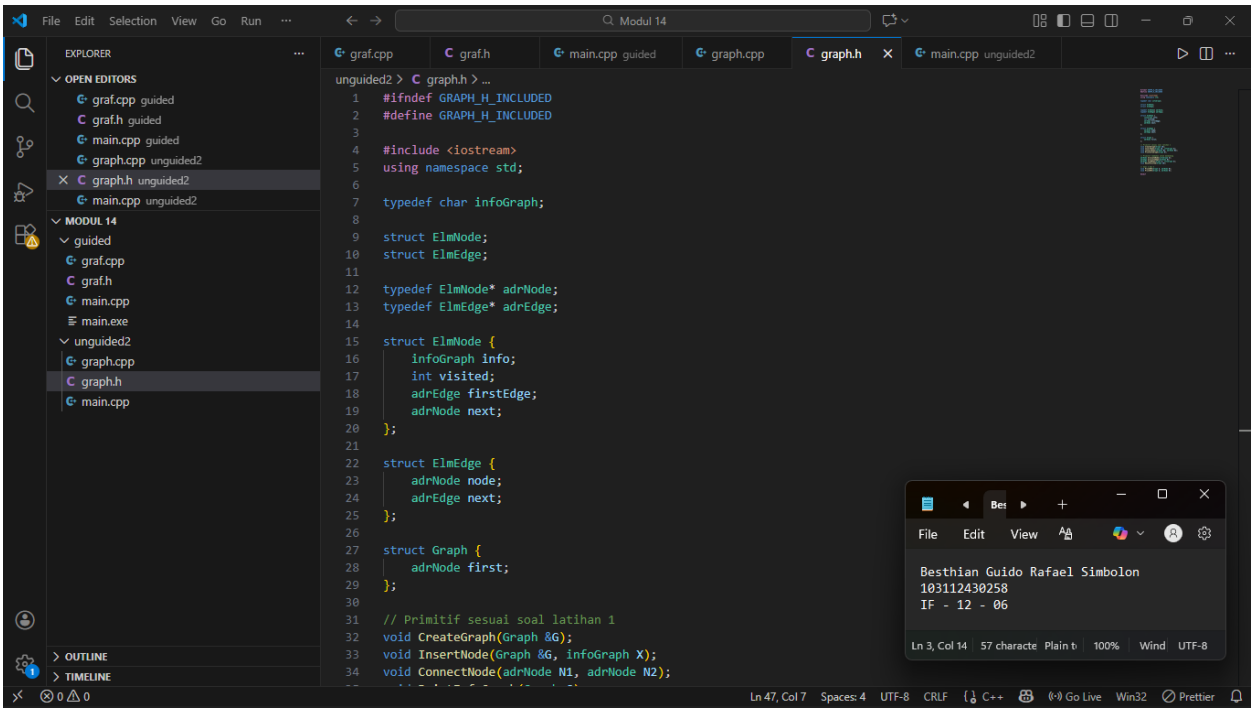
```cpp
#include "graph.h"
#include "graph.cpp"

int main() {
    Graph G;
    CreateGraph(G);

    // Tambah node sesuai Gambar pada soal
    InsertNode(G, 'A');
    InsertNode(G, 'B');
    InsertNode(G, 'C');
    InsertNode(G, 'D');
    InsertNode(G, 'E');
    InsertNode(G, 'F');
    InsertNode(G, 'G');
    InsertNode(G, 'H');

    // Hubungkan node sesuai Gambar pada soal
    adrNode A = FindNode(G, 'A');
    adrNode B = FindNode(G, 'B');
    adrNode C = FindNode(G, 'C');
    adrNode D = FindNode(G, 'D');
    adrNode E = FindNode(G, 'E');
    adrNode F = FindNode(G, 'F');
    adrNode GNode = FindNode(G, 'G');
    adrNode H = FindNode(G, 'H');

    // Koneksi sesuai gambar
    ConnectNode(A, B);
    ConnectNode(A, C);
    ConnectNode(B, D);
    ConnectNode(B, E);
    ConnectNode(C, F);
    ConnectNode(C, GNode);
```

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

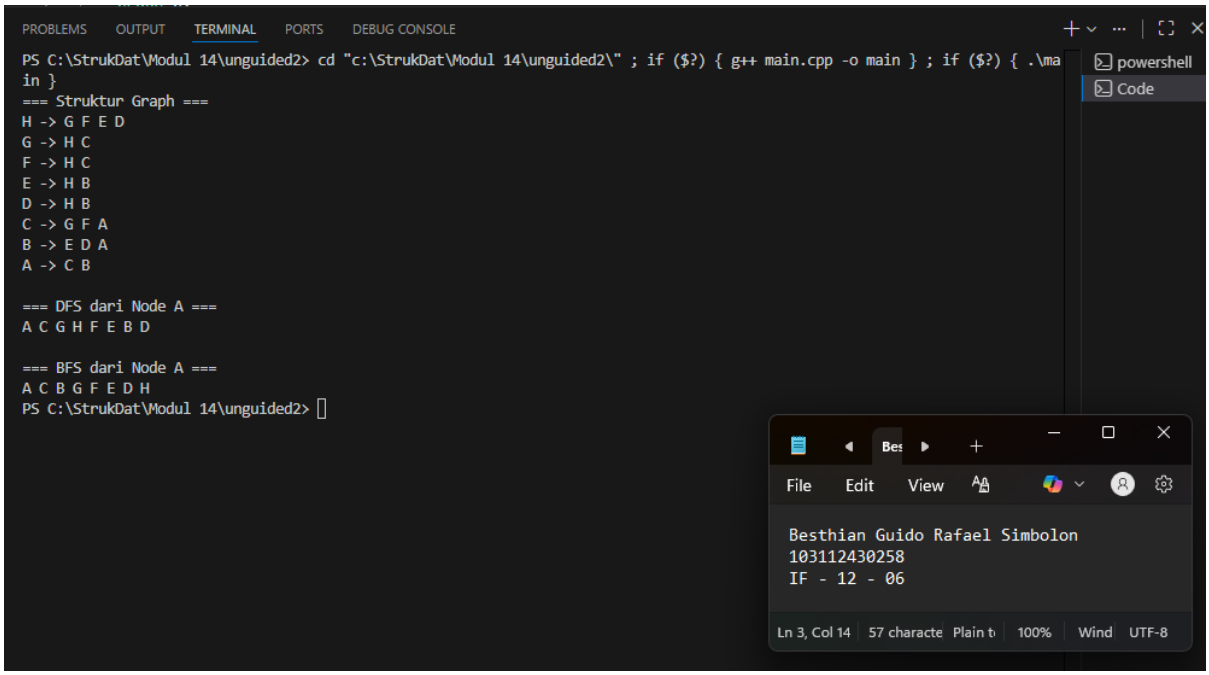
Graph.cpp


File Edit Selection View Go Run
Modul 14
EXPLORER
OPEN EDITORS
graf.cpp guided
graf.h guided
main.cpp guided
graph.cpp unguided2
graph.h unguided2
main.cpp unguided2
MODUL 14
guided
graf.cpp
graf.h
main.cpp
main.exe
unguided2
graph.cpp
graph.h
main.cpp
OUTLINE
TIMELINE
graf.cpp
graf.h
main.cpp guided
graph.cpp
graph.h
main.cpp unguided2
unguided2 > graph.cpp > PrintBFS(Graph, adrNode)
#include "graph.h"
#include <queue>
void CreateGraph(Graph &G) {
G.first = NULL;
}
adrNode AllocateNode(infoGraph X) {
adrNode P = new ElmNode;
P->info = X;
P->visited = 0;
P->firstEdge = NULL;
P->next = NULL;
return P;
}
adrEdge AllocateEdge(adrNode N) {
adrEdge P = new ElmEdge;
P->node = N;
P->next = NULL;
return P;
}
void InsertNode(Graph &G, infoGraph X) {
adrNode P = AllocateNode(X);
P->next = G.first;
G.first = P;
}
adrNode FindNode(Graph G, infoGraph X) {
adrNode P = G.first;
while (P != NULL) {
if (P->info == X)
return P;
File Edit View
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 14 57 characte Plain t 100% Wind UTF-8
Ln 120, Col 2 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

Graph.h


File Edit Selection View Go Run
Modul 14
EXPLORER
OPEN EDITORS
graf.cpp guided
graf.h guided
main.cpp guided
graph.cpp unguided2
graph.h unguided2
main.cpp unguided2
MODUL 14
guided
graf.cpp
graf.h
main.cpp
main.exe
unguided2
graph.cpp
graph.h
main.cpp
OUTLINE
TIMELINE
graf.cpp
graf.h
main.cpp guided
graph.cpp
graph.h
main.cpp unguided2
unguided2 > graph.h > ...
#ifndef GRAPH_H_INCLUDED
#define GRAPH_H_INCLUDED
#include <iostream>
using namespace std;
typedef char infoGraph;
struct ElmNode;
struct ElmEdge;
typedef ElmNode* adrNode;
typedef ElmEdge* adrEdge;
struct ElmNode {
infoGraph info;
int visited;
adrEdge firstEdge;
adrNode next;
};
struct ElmEdge {
adrNode node;
adrEdge next;
};
struct Graph {
adrNode first;
};
// Primitif sesuai soal latihan 1
void CreateGraph(Graph &G);
void InsertNode(Graph &G, infoGraph X);
void ConnectNode(adrNode N1, adrNode N2);
File Edit View
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 14 57 characte Plain t 100% Wind UTF-8
Ln 47, Col 7 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

Screenshot Output



Deskripsi:

Kode ini merupakan implementasi lengkap struktur data **Graph tak berarah (Undirected Graph)** menggunakan bahasa C++ dengan representasi **adjacency list**. Pada program utama (`main.cpp`), dibangun sebuah topologi graph yang terdiri dari **8 node (A hingga H)**, di mana Node A berperan sebagai titik awal yang bercabang ke Node B dan C, kemudian cabang-cabang tersebut terus berkembang hingga akhirnya bertemu kembali pada satu node tujuan, yaitu Node H. Program ini tidak hanya menampilkan struktur koneksi antar node ke layar, tetapi juga bertujuan untuk memperlihatkan perbedaan cara kerja algoritma penelusuran graph. Hal tersebut ditunjukkan melalui penggunaan **Depth First Search (DFS)** yang menelusuri graph secara mendalam hingga satu cabang selesai sebelum berpindah ke cabang lain, serta **Breadth First Search (BFS)** yang menelusuri graph secara melebar berdasarkan level. Kedua algoritma tersebut dijalankan secara berurutan dengan titik awal penelusuran yang sama, yaitu Node A, sehingga perbedaan urutan kunjungan node dapat diamati dengan jelas.

D. Kesimpulan

Berdasarkan implementasi program yang telah dilakukan, dapat disimpulkan bahwa penggunaan representasi graf **adjacency list** dengan pointer memberikan fleksibilitas tinggi dalam membangun dan memodifikasi struktur graf secara dinamis, terutama pada topologi jaringan yang kompleks dengan banyak percabangan dan titik temu. Pengujian terhadap graf menunjukkan perbedaan yang jelas antara algoritma **Depth First Search (DFS)** dan **Breadth First Search (BFS)**, di mana DFS menelusuri graf dengan memprioritaskan kedalaman suatu jalur hingga selesai sebelum berpindah ke jalur lain, sedangkan BFS menelusuri graf dengan memprioritaskan keluasan dengan mengunjungi seluruh node pada level yang sama terlebih dahulu. Hasil ini menegaskan bahwa pemilihan algoritma traversal harus disesuaikan dengan kebutuhan aplikasi, di mana DFS lebih cocok

untuk pencarian solusi berbasis jalur, sedangkan BFS lebih efektif untuk menemukan node terdekat atau jalur terpendek pada graf tak berbobot.

E. Referensi

Rosa, A. S., & Shalahuddin, M. (2018). *Rekayasa perangkat lunak terstruktur dan berorientasi objek*. Bandung: Informatika.

Munir, R. (2018). *Matematika diskrit* (Edisi ke-6). Bandung: Informatika.

Kurniawan, D. (2016). *Struktur data menggunakan C++*. Jakarta: Elex Media Komputindo.

Kadir, A. (2012). *Algoritma dan struktur data*. Yogyakarta: Andi Offset.

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.