LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VII

GRAPH

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

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

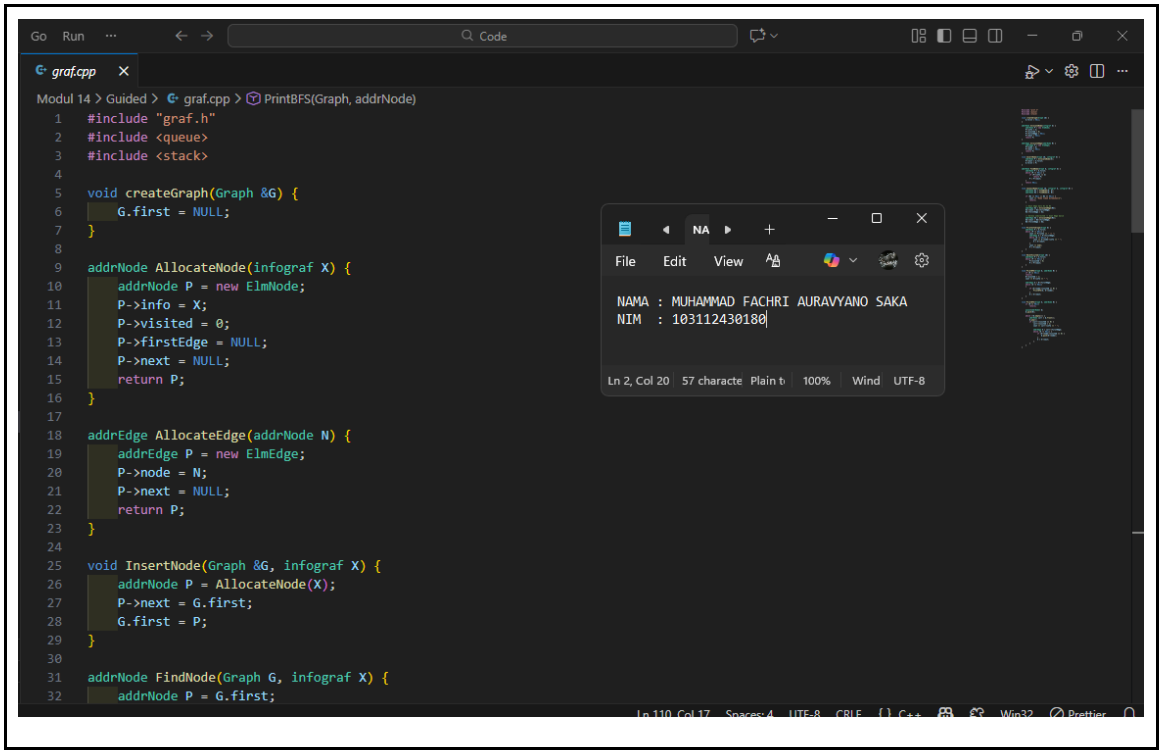
A. Dasar Teori

Graf (Graph) merupakan struktur data non-linear yang merepresentasikan hubungan antar objek, terdiri dari himpunan simpul (node atau vertex) dan himpunan sisi (edge) yang menghubungkan sepasang simpul tersebut. Dalam implementasi pemrograman, graf sering direpresentasikan menggunakan Adjacency List (daftar ketetanggaan), di mana setiap node memiliki senarai berantai (linked list) dinamis yang menyimpan referensi ke node-node tetangganya; metode ini dianggap lebih efisien dalam penggunaan memori dibandingkan Adjacency Matrix, terutama untuk graf yang sparse (jarang memiliki sisi). Untuk menjelajahi seluruh elemen dalam graf, terdapat dua algoritma penelusuran fundamental, yaitu Depth First Search (DFS) dan Breadth First Search (BFS). DFS melakukan penelusuran secara mendalam menggunakan prinsip Last In First Out (LIFO) yang umumnya diimplementasikan melalui rekursi atau struktur data stack, di mana algoritma akan mengunjungi satu cabang sejauh mungkin sebelum melakukan backtracking. Sebaliknya, BFS melakukan penelusuran secara melebar (per level) menggunakan prinsip First In First Out (FIFO) dengan bantuan struktur data queue, yang memastikan semua tetangga terdekat dikunjungi terlebih dahulu sebelum melanjutkan ke node pada lapisan yang lebih dalam.

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

Guided 1

Graf.cpp



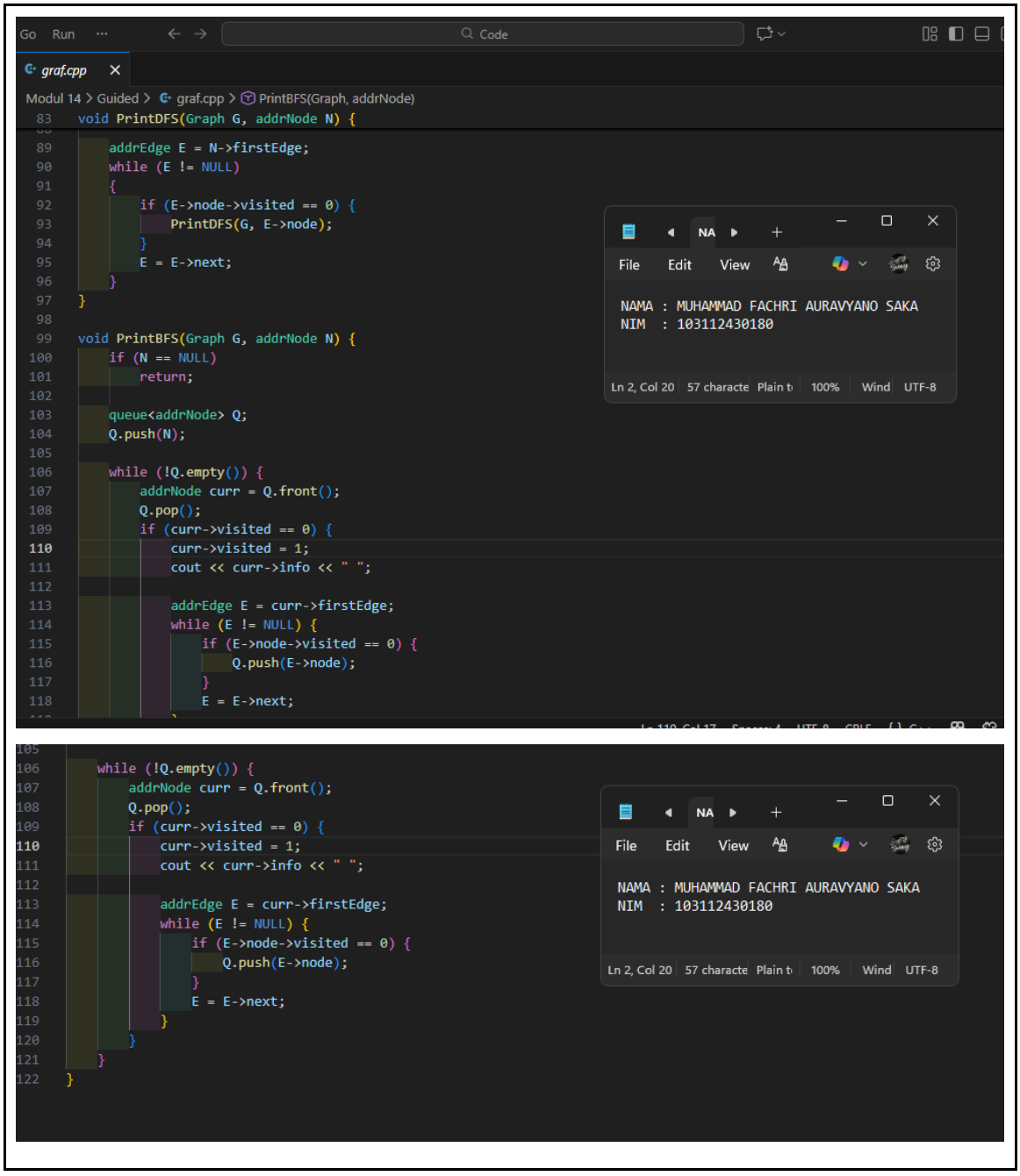
```
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)

addrNode FindNode(Graph G, infograf X) {
    addrNode P = G.first;
    while (P != NULL) {
        if (P->info == X)
            return P;
        P = P->next;
    }
    return NULL;
}

void ConnectNode(Graph &G, infograf A, infograf B) {
    addrNode N1 = FindNode(G, A);
    addrNode N2 = FindNode(G, B);

    if (N1 == NULL || N2 == NULL) {
        cout << "Node tidak ditemukan\n";
        return;
    }

    // Buat Edge daro N1 ke N2
    addrEdge E1 = AllocateEdge(N2);
    E1->next = N1->firstEdge;
    N1->firstEdge = E1;

    // Karena undirected -> buat Edge balik
    addrEdge E2 = AllocateEdge(N1);
    E2->next = N2->firstEdge;
    N2->firstEdge = E2;
}
```

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

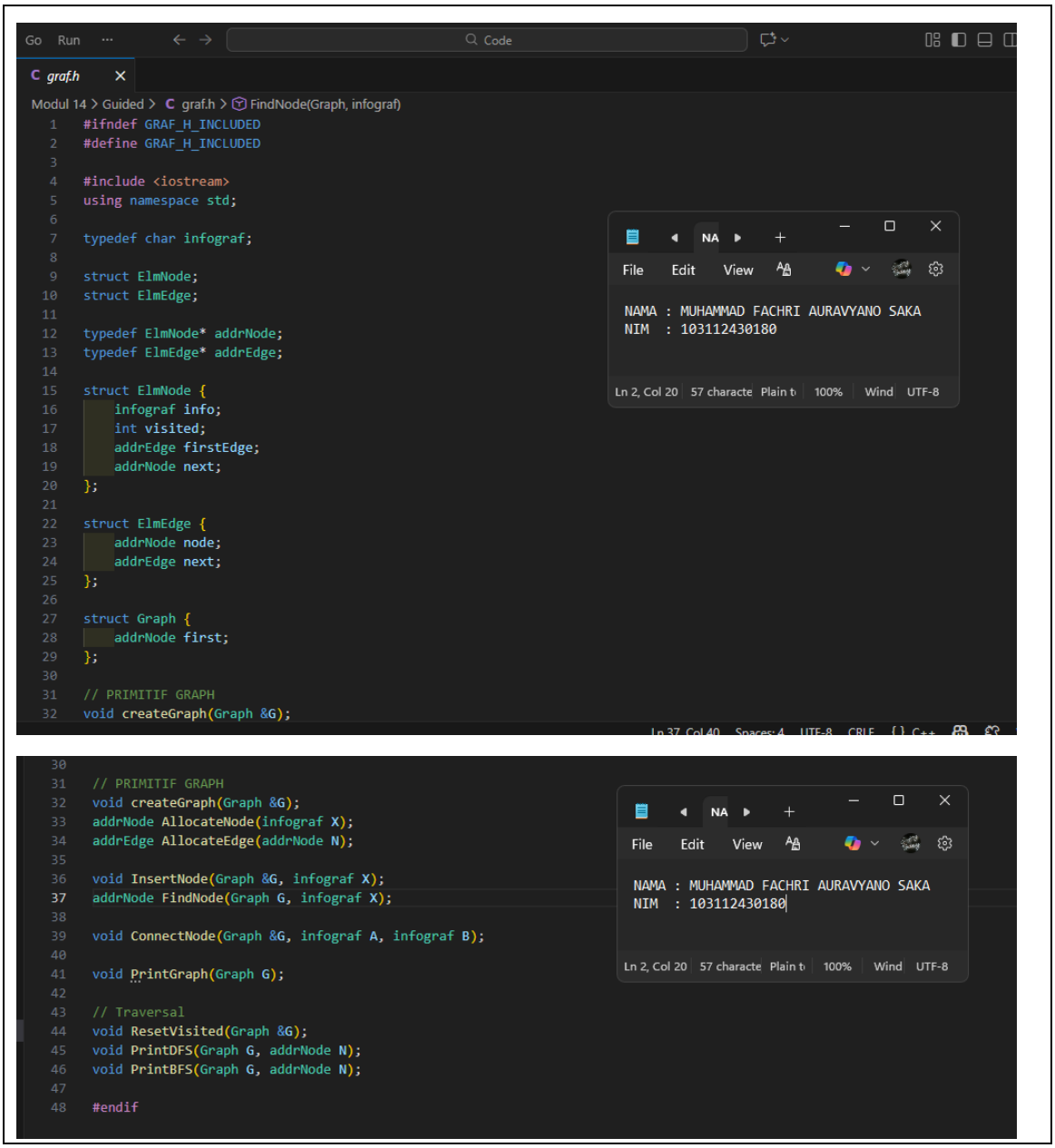


```
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)

void PrintInfoGraph(Graph G) {
    addrNode P = G.first;
    while (P != NULL) {
        cout << P->info << " -> ";
        addrEdge E = P->firstEdge;
        while (E != NULL) {
            cout << E->node->info << " ";
            E = E->next;
        }
        cout << endl;
        P = P->next;
    }
}

void ResetVisited(Graph &G) {
    addrNode P = G.first;
    while (P != NULL) {
        P->visited = 0;
        P = P->next;
    }
}

void PrintDFS(Graph G, addrNode N) {
    if (N == NULL)
    return;
    N->visited = 1;
    cout << N->info << " ";

    addrEdge E = N->firstEdge;
    while (E != NULL)
    {
```

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```


Go Run
Code
graf.cpp
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
void PrintDFS(Graph G, addrNode N) {
addrEdge E = N->firstEdge;
while (E != NULL)
{
if (E->node->visited == 0) {
PrintDFS(G, E->node);
}
E = E->next;
}
}
void PrintBFS(Graph G, addrNode N) {
if (N == NULL)
return;
queue<addrNode> Q;
Q.push(N);
while (!Q.empty()) {
addrNode curr = Q.front();
Q.pop();
if (curr->visited == 0) {
curr->visited = 1;
cout << curr->info << " ";
addrEdge E = curr->firstEdge;
while (E != NULL) {
if (E->node->visited == 0) {
Q.push(E->node);
}
E = E->next;
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8
while (!Q.empty()) {
addrNode curr = Q.front();
Q.pop();
if (curr->visited == 0) {
curr->visited = 1;
cout << curr->info << " ";
addrEdge E = curr->firstEdge;
while (E != NULL) {
if (E->node->visited == 0) {
Q.push(E->node);
}
E = E->next;
}
}
}
}
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

Graf.h


Go Run
Code
graf.h
Modul 14 > Guided > graf.h > FindNode(Graph, infograf)
#ifndef GRAF_H_INCLUDED
#define GRAF_H_INCLUDED
#include <iostream>
using namespace std;
typedef char infograf;
struct ElmNode;
struct ElmEdge;
typedef ElmNode* addrNode;
typedef ElmEdge* addrEdge;
struct ElmNode {
infograf info;
int visited;
addrEdge firstEdge;
addrNode next;
};
struct ElmEdge {
addrNode node;
addrEdge next;
};
struct Graph {
addrNode first;
};
// PRIMITIF GRAPH
void createGraph(Graph &G);
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8
Ln 37, Col 40 Spaces: 4 UTF-8 CRLF C++
// PRIMITIF GRAPH
void createGraph(Graph &G);
addrNode AllocateNode(infograf X);
addrEdge AllocateEdge(addrNode N);
void InsertNode(Graph &G, infograf X);
addrNode FindNode(Graph G, infograf X);
void ConnectNode(Graph &G, infograf A, infograf B);
void PrintGraph(Graph G);
// Traversal
void ResetVisited(Graph &G);
void PrintDFS(Graph G, addrNode N);
void PrintBFS(Graph G, addrNode N);
#endif
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

Main.cpp

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

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

    // Tambah Node
    InsertNode(G, 'A');
    InsertNode(G, 'B');
    InsertNode(G, 'C');
    InsertNode(G, 'D');
    InsertNode(G, 'E');

    // Hubungkan Node (graph tidak berarah)
    ConnectNode(G, 'A', 'B');
    ConnectNode(G, 'A', 'C');
    ConnectNode(G, 'B', 'D');
    ConnectNode(G, 'C', 'E');

    cout << "=== Struktur Graph ===\n";
    PrintInfoGraph(G);

    cout << "\n=== DFS dari Node A ===\n";
    ResetVisited(G);
    PrintDFS(G, FindNode(G, 'A'));

    cout << "\n\n=== BFS dari Node B ===\n";
    ResetVisited(G);
    PrintBFS(G, FindNode(G, 'A'));

    cout << endl;
    return 0;
}
```

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

Screenshots Output

```
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Guided> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Guided\" ; if ($?) { g++ main.cpp -o main } ;
if ($?) { .\main }
=== Struktur Graph ===
E -> C
D -> B
C -> E A
B -> D A
A -> C B

=== DFS dari Node A ===
A C E B D

=== BFS dari Node B ===
A C B E D
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Guided>
```

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

Deskripsi:

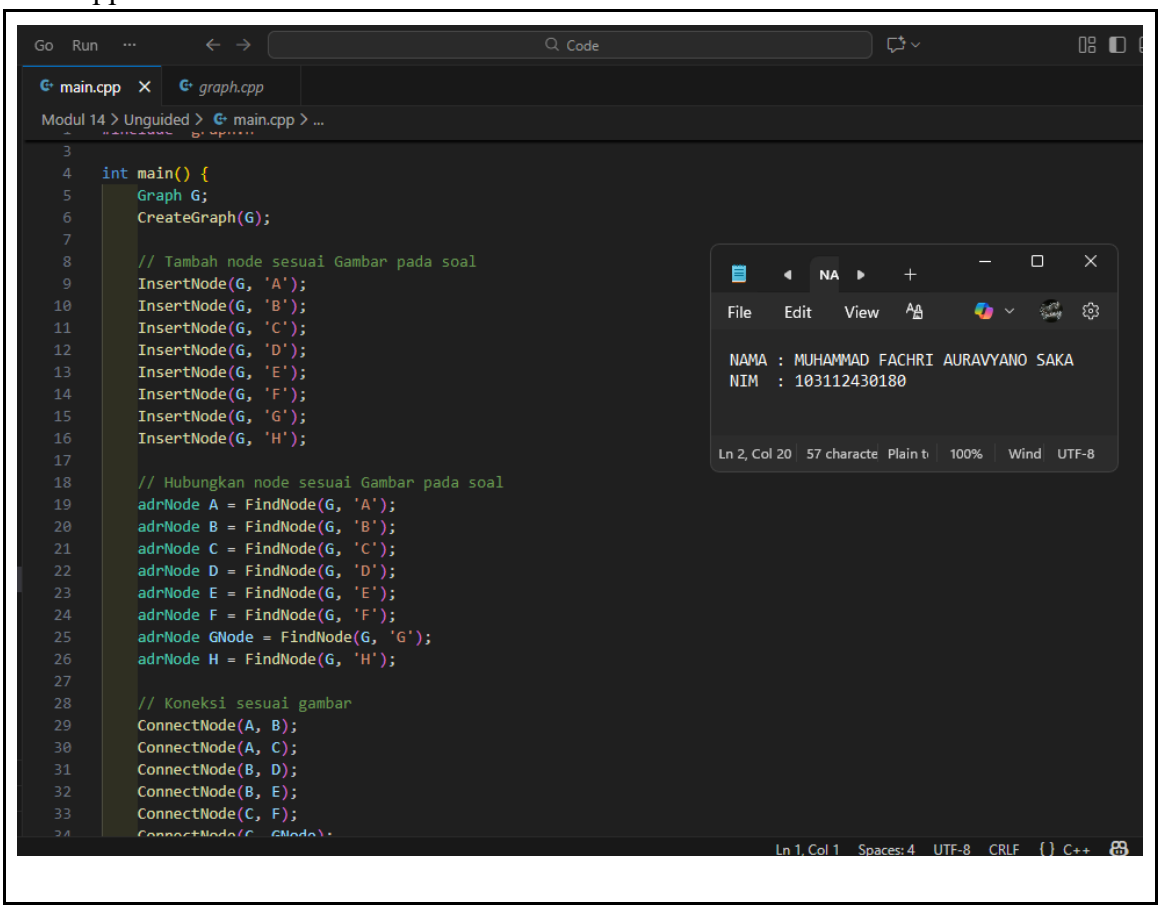
Program ini adalah implementasi struktur data Graph tak berarah (Undirected Graph) menggunakan bahasa C++ dengan representasi daftar ketetanggaan (adjacency list). Program ini mendefinisikan tipe data dinamis untuk Node (simpul) dan Edge (sisi/jalur) menggunakan pointer, di mana setiap node memiliki daftar koneksi ke node lainnya.

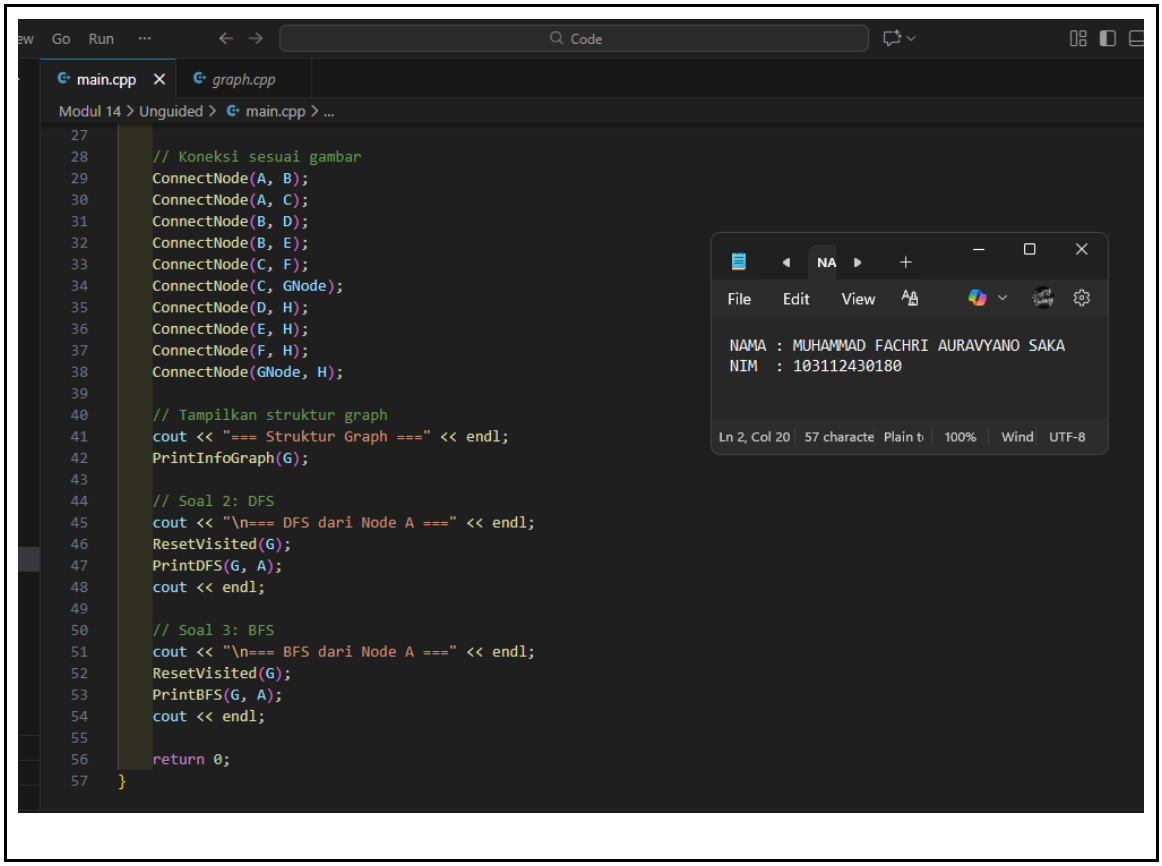
Secara fungsional, program ini memungkinkan pengguna untuk membuat graph, menambahkan node (dalam contoh ini node A, B, C, D, dan E), serta menghubungkan node-node tersebut. Setelah struktur graph terbentuk, program menampilkan representasi visual koneksinya ke layar. Selain itu, program ini juga mendemonstrasikan dua algoritma penelusuran (traversal) utama: DFS (Depth First Search) yang menelusuri graph secara mendalam menggunakan rekursi, dan BFS (Breadth First Search) yang menelusuri secara melebar menggunakan bantuan struktur data queue.

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

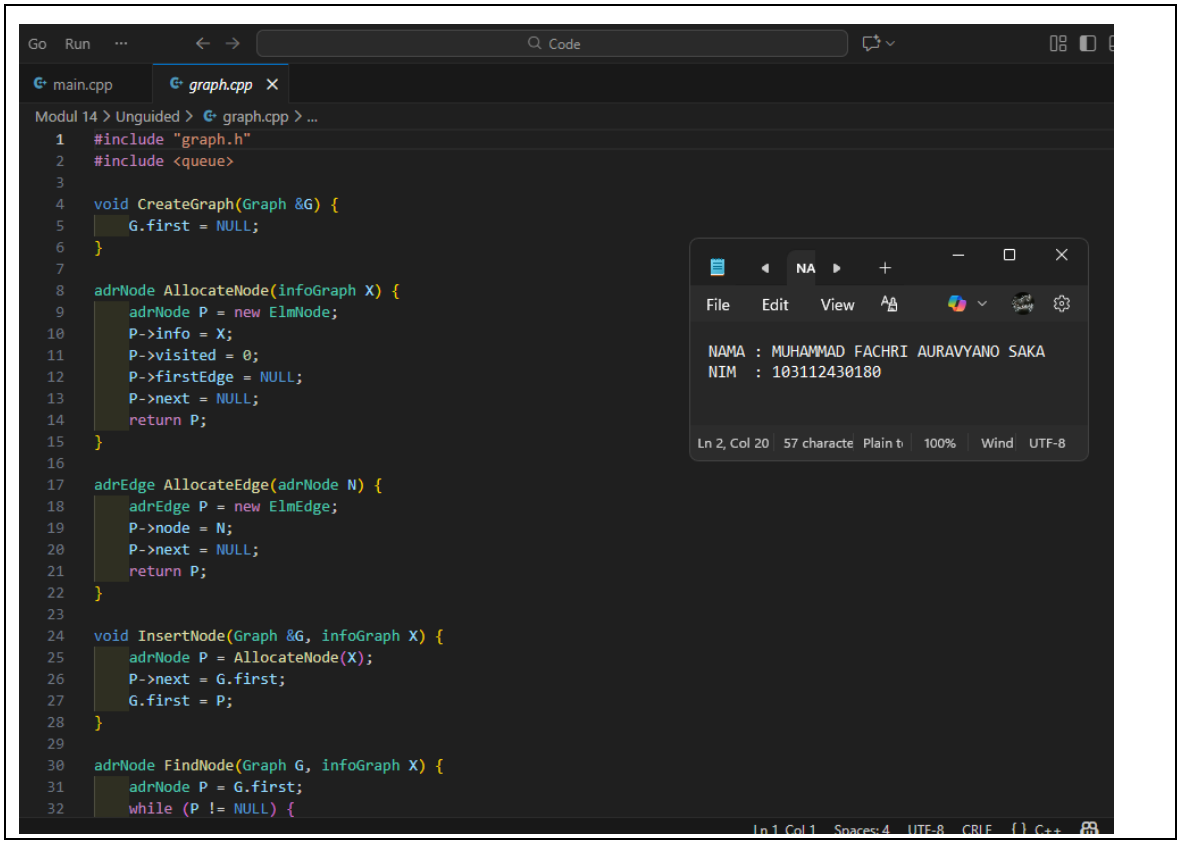
Unguided 1

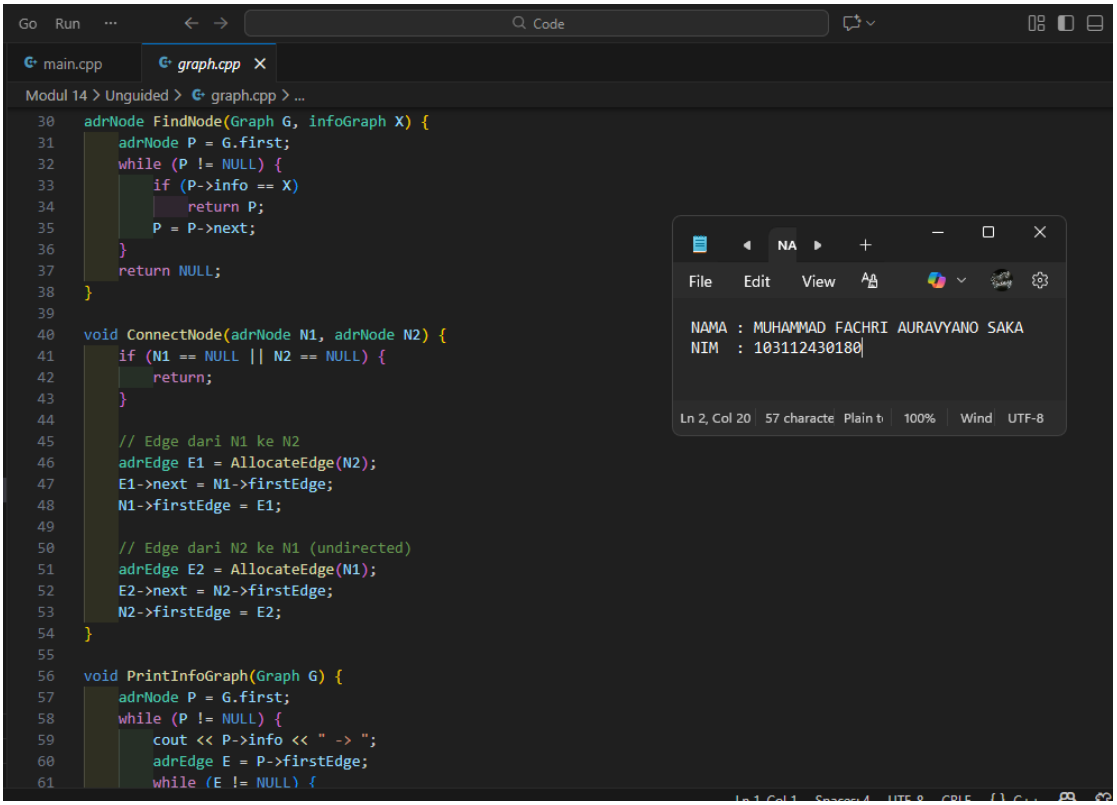
Main.cpp

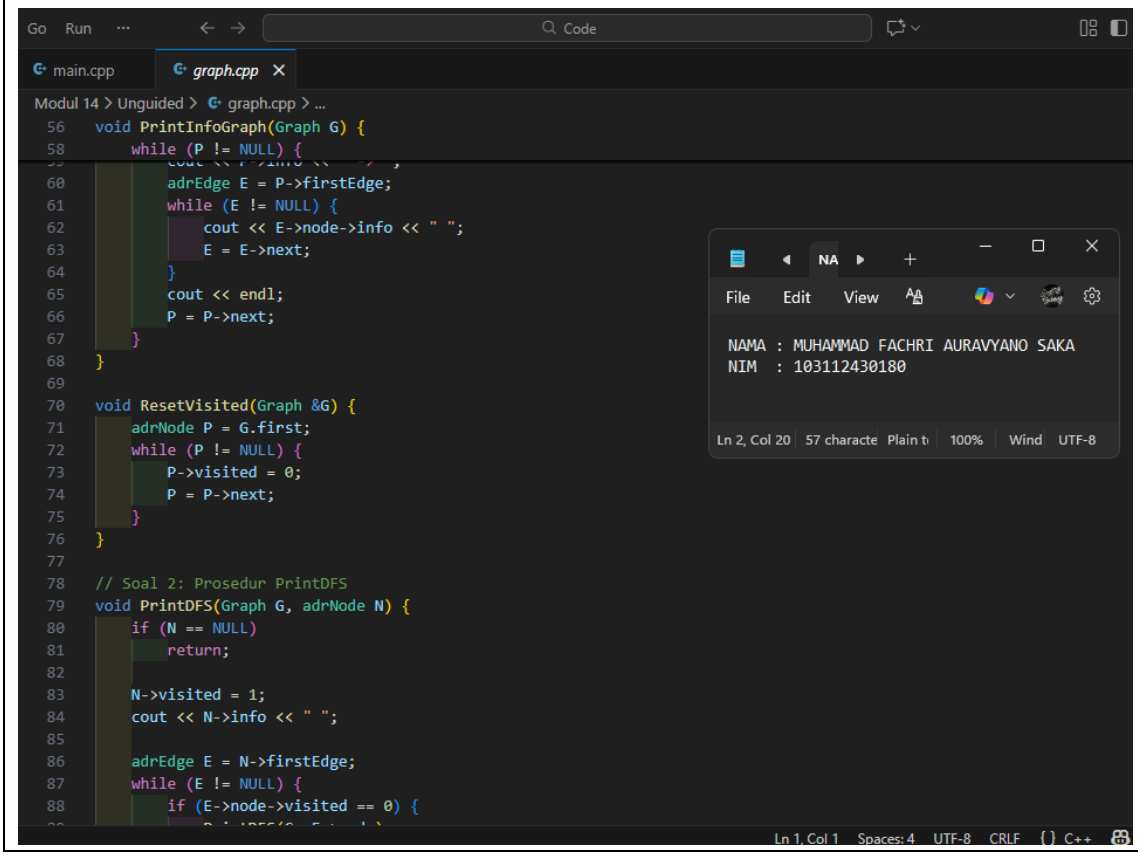


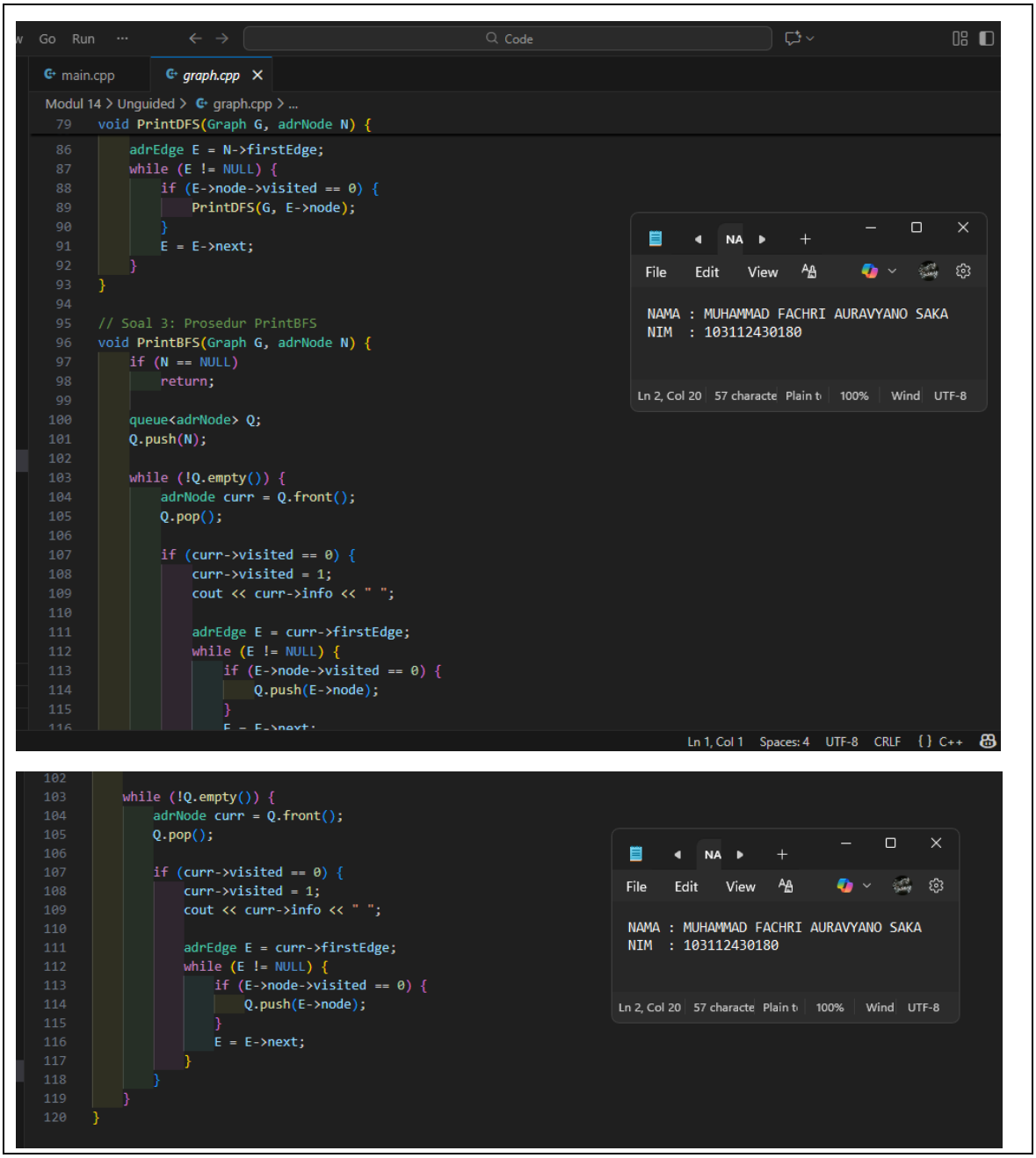

Go Run
Code
main.cpp
graph.cpp
Modul 14 > Unguided > main.cpp > ...
// Koneksi sesuai gambar
ConnectNode(A, B);
ConnectNode(A, C);
ConnectNode(B, D);
ConnectNode(B, E);
ConnectNode(C, F);
ConnectNode(C, GNode);
ConnectNode(D, H);
ConnectNode(E, H);
ConnectNode(F, H);
ConnectNode(GNode, H);
// Tampilkan struktur graph
cout << "=== Struktur Graph ===" << endl;
PrintInfoGraph(G);
// Soal 2: DFS
cout << "\n=== DFS dari Node A ===" << endl;
ResetVisited(G);
PrintDFS(G, A);
cout << endl;
// Soal 3: BFS
cout << "\n=== BFS dari Node A ===" << endl;
ResetVisited(G);
PrintBFS(G, A);
cout << endl;
return 0;
}
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

graph.cpp


Go Run
Code
main.cpp
graph.cpp
Modul 14 > Unguided > graph.cpp > ...
#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) {
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

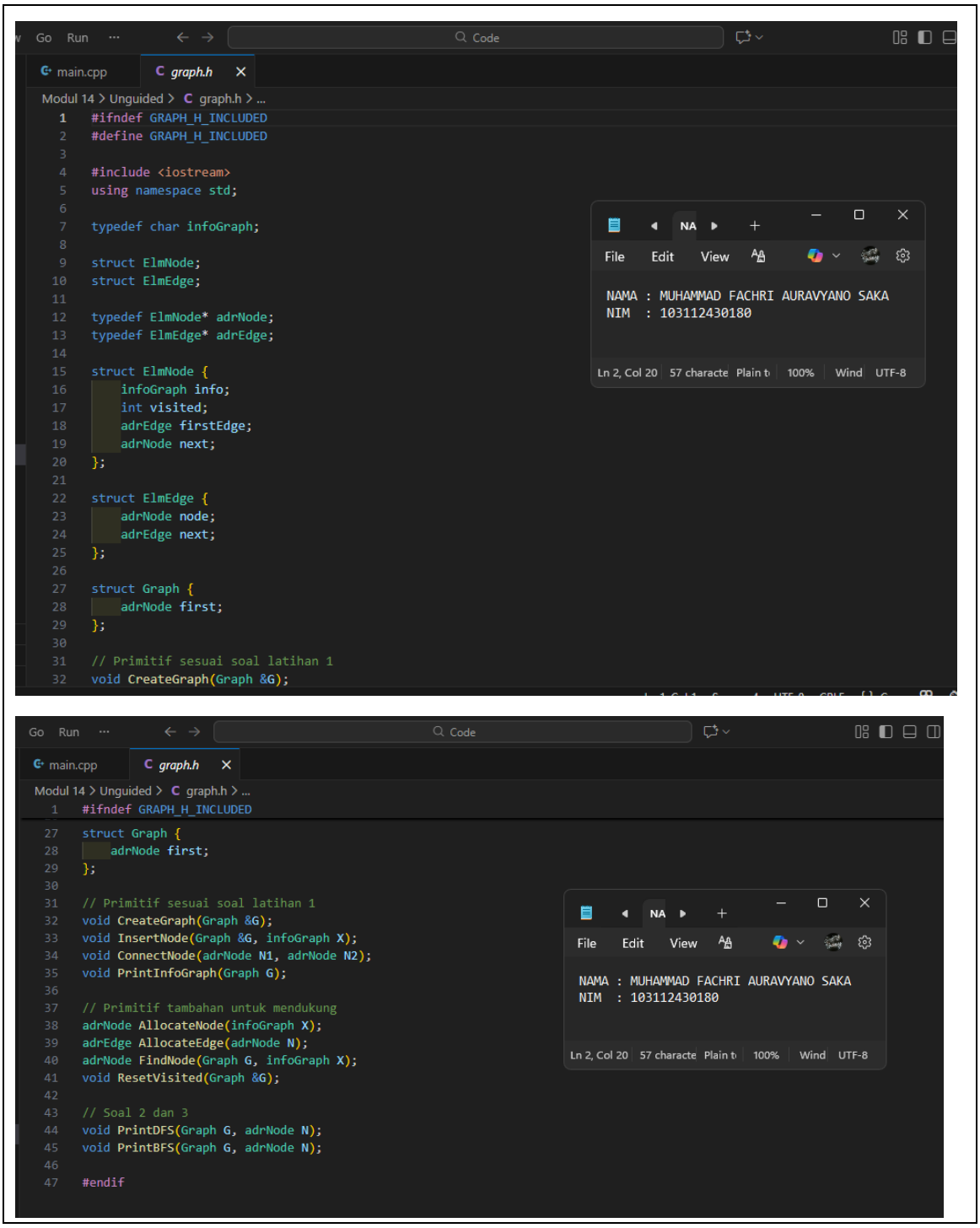

Go Run
Code
main.cpp
graph.cpp
Modul 14 > Unguided > graph.cpp > ...
adrNode FindNode(Graph G, infoGraph X) {
adrNode P = G.first;
while (P != NULL) {
if (P->info == X)
return P;
P = P->next;
}
return NULL;
}
void ConnectNode(adrNode N1, adrNode N2) {
if (N1 == NULL || N2 == NULL) {
return;
}
// Edge dari N1 ke N2
adrEdge E1 = AllocateEdge(N2);
E1->next = N1->firstEdge;
N1->firstEdge = E1;
// Edge dari N2 ke N1 (undirected)
adrEdge E2 = AllocateEdge(N1);
E2->next = N2->firstEdge;
N2->firstEdge = E2;
}
void PrintInfoGraph(Graph G) {
adrNode P = G.first;
while (P != NULL) {
cout << P->info << " -> ";
adrEdge E = P->firstEdge;
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8


Go Run
Code
main.cpp
graph.cpp
Modul 14 > Unguided > graph.cpp > ...
void PrintInfoGraph(Graph G) {
while (P != NULL) {
adrEdge E = P->firstEdge;
while (E != NULL) {
cout << E->node->info << " ";
E = E->next;
}
cout << endl;
P = P->next;
}
}
void ResetVisited(Graph &G) {
adrNode P = G.first;
while (P != NULL) {
P->visited = 0;
P = P->next;
}
}
// Soal 2: Prosedur PrintDFS
void PrintDFS(Graph G, adrNode N) {
if (N == NULL)
return;
N->visited = 1;
cout << N->info << " ";
adrEdge E = N->firstEdge;
while (E != NULL) {
if (E->node->visited == 0) {
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF { } C++

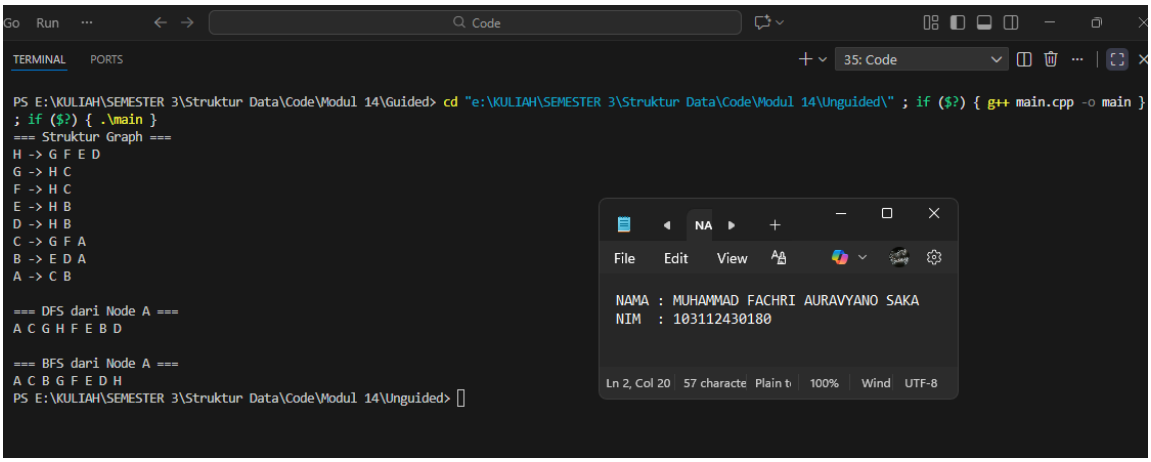

Go Run
Code
main.cpp
graph.cpp
Modul 14 > Unguided > graph.cpp > ...
void PrintDFS(Graph G, adrNode N) {
adrEdge E = N->firstEdge;
while (E != NULL) {
if (E->node->visited == 0) {
PrintDFS(G, E->node);
}
E = E->next;
}
}
// Soal 3: Prosedur PrintBFS
void PrintBFS(Graph G, adrNode N) {
if (N == NULL)
return;
queue<adrNode> Q;
Q.push(N);
while (!Q.empty()) {
adrNode curr = Q.front();
Q.pop();
if (curr->visited == 0) {
curr->visited = 1;
cout << curr->info << " ";
adrEdge E = curr->firstEdge;
while (E != NULL) {
if (E->node->visited == 0) {
Q.push(E->node);
}
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF { } C++
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

while (!Q.empty()) {
adrNode curr = Q.front();
Q.pop();
if (curr->visited == 0) {
curr->visited = 1;
cout << curr->info << " ";
adrEdge E = curr->firstEdge;
while (E != NULL) {
if (E->node->visited == 0) {
Q.push(E->node);
}
E = E->next;
}
}
}
}
NA
File Edit View
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
Ln 2, Col 20 57 characte Plain t 100% Wind UTF-8

graph.h



Screenshot Output



Deskripsi:

Kode ini merupakan implementasi lengkap struktur data Graph Tak Berarah (Undirected Graph) menggunakan bahasa C++ dengan metode representasi daftar ketetanggaan (adjacency list), di mana program utama (main.cpp) membangun sebuah topologi jaringan spesifik yang terdiri dari 8 node (A hingga H). Dalam skenario ini, Node A berfungsi sebagai titik awal yang bercabang ke B dan C, lalu cabang-cabang tersebut terus meluas hingga akhirnya menyatu kembali (konvergen) di satu titik temu yaitu Node H. Selain memvisualisasikan struktur koneksi antar node tersebut ke layar, program ini bertujuan mendemonstrasikan perbandingan logika penelusuran graph dengan menjalankan algoritma Depth First Search (DFS) yang menelusuri kedalaman cabang hingga tuntas terlebih dahulu, dan Breadth First Search (BFS) yang menelusuri node secara melebar per level, yang keduanya dieksekusi berturut-turut dimulai dari sumber yang sama yaitu Node A.

D. Kesimpulan

Berdasarkan implementasi program yang telah dilakukan, dapat disimpulkan bahwa representasi graf menggunakan adjacency list dengan pointer memberikan fleksibilitas tinggi dalam memanipulasi topologi jaringan yang kompleks, seperti penambahan node dan koneksi antar node secara dinamis tanpa batasan ukuran array yang statis. Uji coba pada topologi graf yang memiliki titik percabangan dan titik temu (konvergensi) berhasil memperlihatkan perbedaan karakteristik yang signifikan antara metode DFS dan BFS; DFS terbukti memprioritaskan kedalaman dengan menelusuri satu jalur hingga node tujuan akhir sebelum beralih ke jalur lain, sedangkan BFS memprioritaskan keluasan dengan menyapu bersih semua node pada level yang sama sebelum turun ke level berikutnya. Hal ini menegaskan bahwa pemilihan algoritma traversal sangat bergantung pada tujuan spesifik aplikasi: DFS lebih sesuai untuk simulasi pencarian jalur solusi dalam labirin atau teka-teki, sementara BFS lebih efektif untuk mencari tetangga terdekat atau jalur terpendek dalam graf tak berbobot.

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.