

**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**

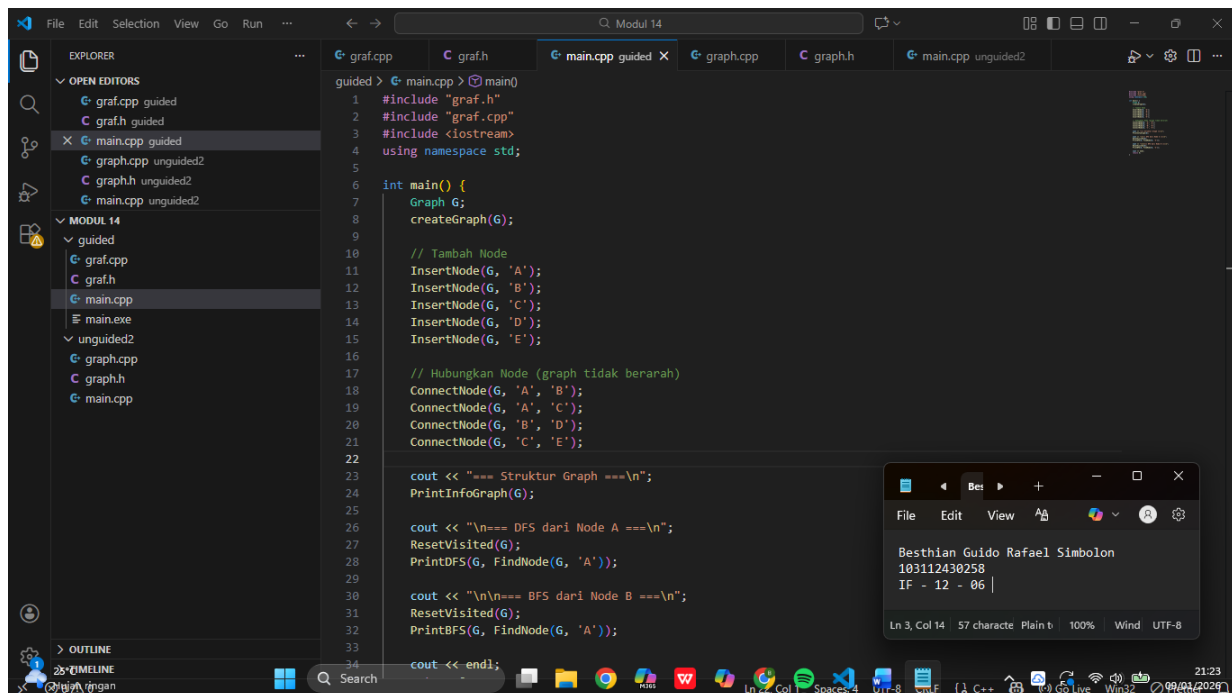
A. Dasar Teori

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

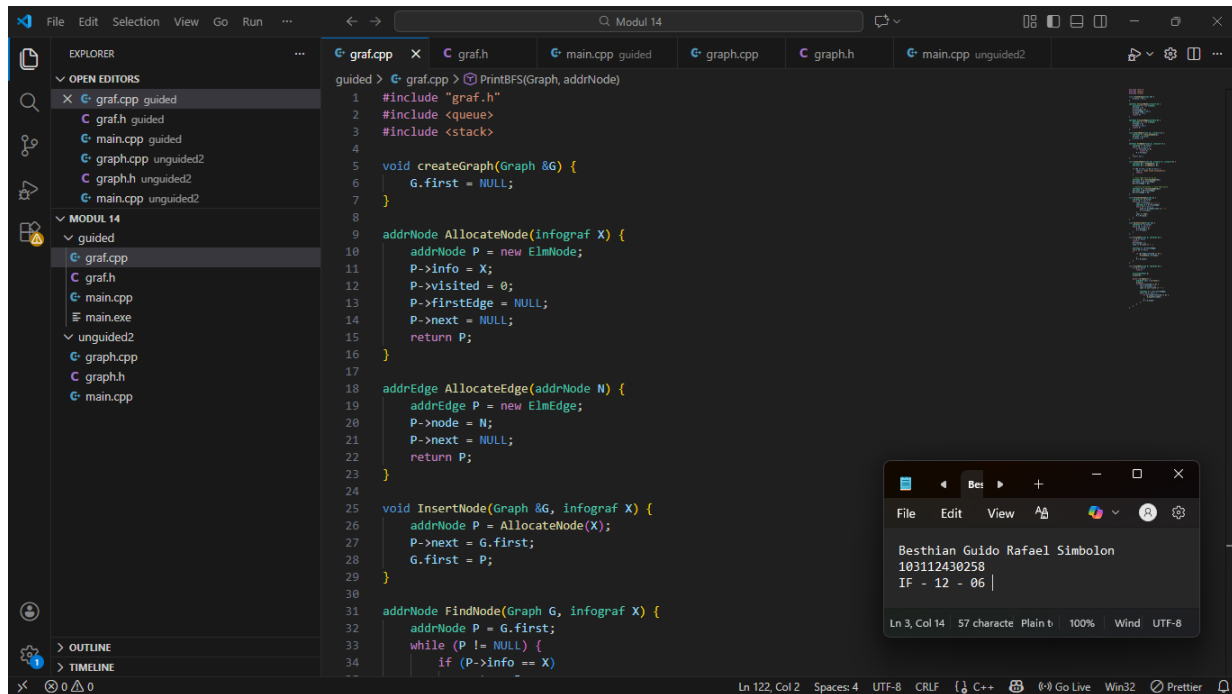
main.cpp



```
1 #include "graf.h"
2 #include "graf.cpp"
3 #include <iostream>
4 using namespace std;
5
6 int main() {
7     Graph G;
8     createGraph(G);
9
10    // Tambah Node
11    InsertNode(G, 'A');
12    InsertNode(G, 'B');
13    InsertNode(G, 'C');
14    InsertNode(G, 'D');
15    InsertNode(G, 'E');
16
17    // Hubungkan Node (graph tidak berarah)
18    ConnectNode(G, 'A', 'B');
19    ConnectNode(G, 'A', 'C');
20    ConnectNode(G, 'B', 'D');
21    ConnectNode(G, 'C', 'E');
22
23    cout << "=== Struktur Graph ===\n";
24    PrintInfoGraph(G);
25
26    cout << "\n=== DFS dari Node A ===\n";
27    ResetVisited(G);
28    PrintDFS(G, FindNode(G, 'A'));
29
30    cout << "\n=== BFS dari Node B ===\n";
31    ResetVisited(G);
32    PrintBFS(G, FindNode(G, 'A'));
33
34    cout << endl;
```

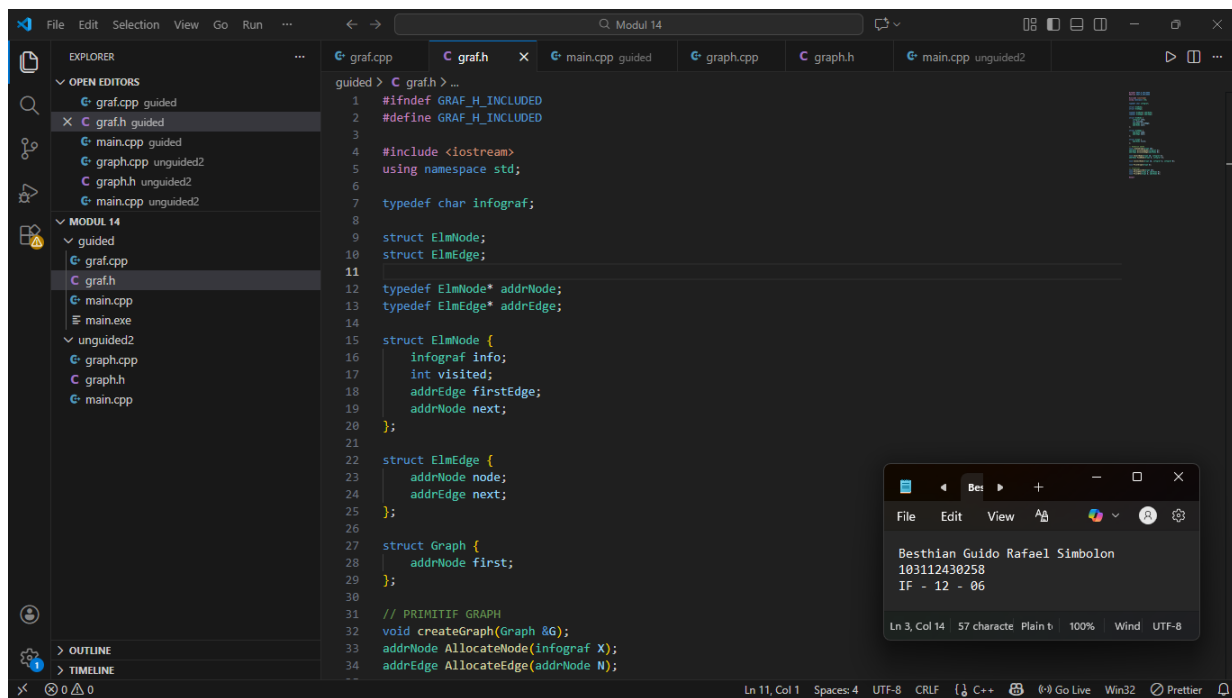
Besthian Guido Rafael Simbolon
183112430258
IF - 12 - 06 |

graf.cpp



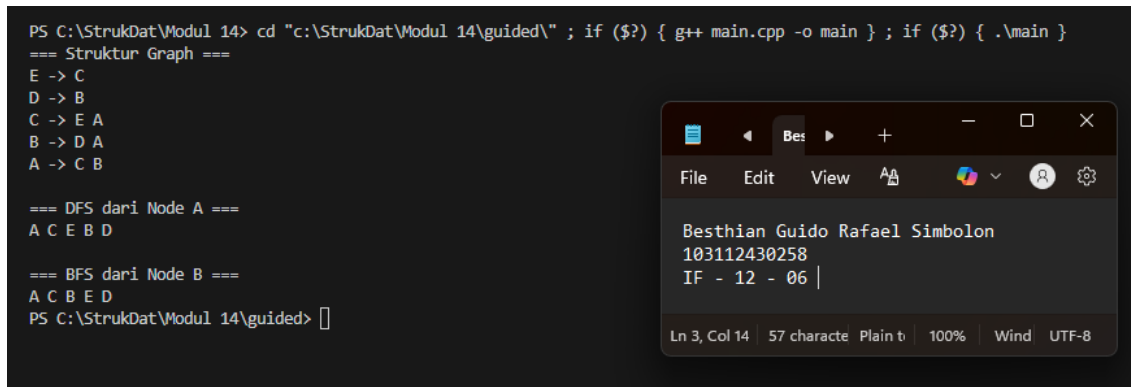
```
1 #include "graf.h"
2 #include <queue>
3 #include <stack>
4
5 void createGraph(Graph &G) {
6     G.first = NULL;
7 }
8
9 addrNode AllocateNode(intograf X) {
10     addrNode P = new ElmNode;
11     P->info = X;
12     P->visited = 0;
13     P->firstEdge = NULL;
14     P->next = NULL;
15     return P;
16 }
17
18 addrEdge AllocateEdge(addrNode N) {
19     addrEdge P = new ElmEdge;
20     P->nnode = N;
21     P->nnext = NULL;
22     return P;
23 }
24
25 void InsertNode(Graph &G, intograf X) {
26     addrNode P = AllocateNode(X);
27     P->next = G.first;
28     G.first = P;
29 }
30
31 addrNode FindNode(Graph G, intograf X) {
32     addrNode P = G.first;
33     while (P != NULL) {
34         if (P->info == X)
```

graf.h



```
1 #ifndef GRAF_H_INCLUDED
2 #define GRAF_H_INCLUDED
3
4 #include <iostream>
5 using namespace std;
6
7 typedef char intograf;
8
9 struct ElmNode;
10 struct ElmEdge;
11
12 typedef ElmNode* addrNode;
13 typedef ElmEdge* addrEdge;
14
15 struct ElmNode {
16     intograf info;
17     int visited;
18     addrEdge firstEdge;
19     addrNode next;
20 };
21
22 struct ElmEdge {
23     addrNode node;
24     addrEdge next;
25 };
26
27 struct Graph {
28     addrNode first;
29 };
30
31 // PRIMITIF GRAPH
32 void createGraph(Graph &G);
33 addrNode AllocateNode(intograf X);
34 addrEdge AllocateEdge(addrNode N);
```

Screenshots Output



```
PS C:\StrukDat\Modul 14> cd "c:\StrukDat\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 C:\StrukDat\Modul 14\guided>
```

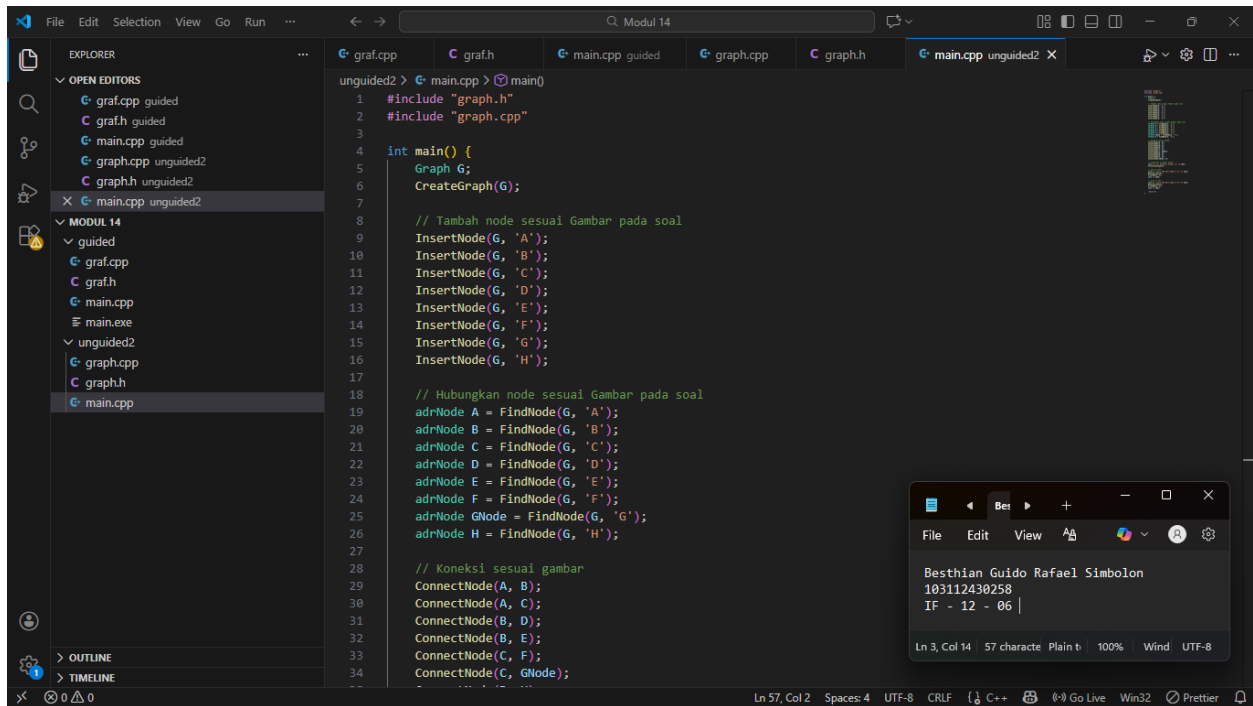
Deskripsi:

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



```
1 #include "graph.h"
2 #include "graph.cpp"
3
4 int main() {
5     Graph G;
6     CreateGraph(G);
7
8     // Tambah node sesuai Gambar pada soal
9     InsertNode(G, 'A');
10    InsertNode(G, 'B');
11    InsertNode(G, 'C');
12    InsertNode(G, 'D');
13    InsertNode(G, 'E');
14    InsertNode(G, 'F');
15    InsertNode(G, 'G');
16    InsertNode(G, 'H');
17
18    // Hubungkan node sesuai Gambar pada soal
19    adrNode A = FindNode(G, 'A');
20    adrNode B = FindNode(G, 'B');
21    adrNode C = FindNode(G, 'C');
22    adrNode D = FindNode(G, 'D');
23    adrNode E = FindNode(G, 'E');
24    adrNode F = FindNode(G, 'F');
25    adrNode GNode = FindNode(G, 'G');
26    adrNode H = FindNode(G, 'H');
27
28    // Koneksi sesuai gambar
29    ConnectNode(A, B);
30    ConnectNode(A, C);
31    ConnectNode(B, D);
32    ConnectNode(B, E);
33    ConnectNode(C, F);
34    ConnectNode(C, GNode);
35    ConnectNode(D, E);
36    ConnectNode(E, F);
37    ConnectNode(F, GNode);
38    ConnectNode(GNode, H);
39    ConnectNode(H, D);
40}
```

Graph.cpp

The screenshot shows the Visual Studio Code editor with the following details:

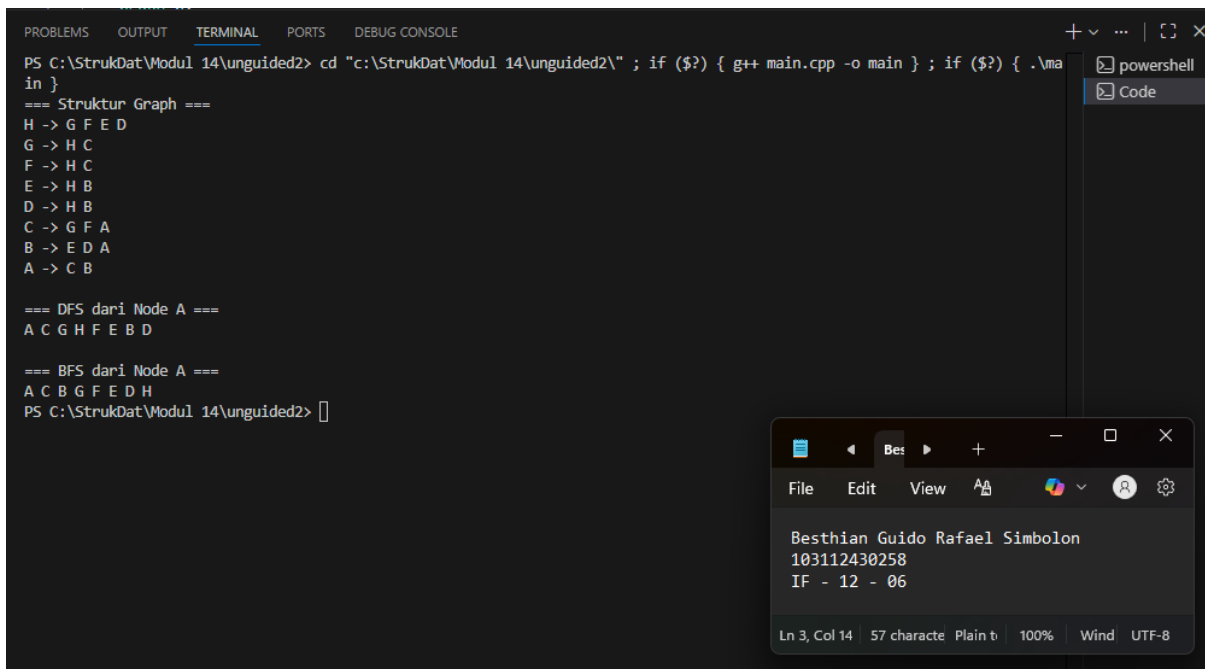
- EXPLORER:** The file explorer on the left shows the project structure. The 'MODUL 14' folder is expanded, showing 'guided' and 'unguided2' subfolders. The 'graph.cpp' file is selected in the 'unguided2' folder.
- OPEN EDITORS:** The top of the editor shows several open files: 'graf.cpp', 'graf.h', 'main.cpp', 'graph.cpp', 'graph.h', and 'main.cpp'. The 'graph.cpp' file is the active editor.
- Code:** The main editor displays the implementation of graph functions in 'graph.cpp'. The code includes headers for 'graph.h' and 'queue', and implements functions like 'CreateGraph', 'AllocateNode', 'AllocateEdge', 'InsertNode', and 'FindNode'. The code is written in C++ and uses a linked list structure for the graph.
- STATUS BAR:** The bottom status bar indicates the current line and column (Ln 120, Col 2), the number of spaces (4), the encoding (UTF-8), and the line ending (CRLF). It also shows the C++ language mode and the Prettier formatter.
- OUTPUT WINDOW:** A small output window is visible on the right side of the editor, displaying the text: 'Besthian Guido Rafael Simbolon 103112430258 IF - 12 - 06'.

Graph.h

The screenshot shows the Visual Studio Code editor with the following details:

- EXPLORER:** The file explorer on the left shows the project structure. The 'MODUL 14' folder is expanded, showing 'guided' and 'unguided2' subfolders. The 'graph.h' file is selected in the 'unguided2' folder.
- OPEN EDITORS:** The top of the editor shows several open files: 'graf.cpp', 'graf.h', 'main.cpp', 'graph.cpp', 'graph.h', and 'main.cpp'. The 'graph.h' file is the active editor.
- Code:** The main editor displays the header file 'graph.h'. The code includes a preprocessor guard for 'GRAPH_H_INCLUDED', includes the 'iostream' header, and defines the 'infoGraph' typedef. It also defines the 'ElmNode' and 'ElmEdge' structures, and the 'Graph' structure. The code is written in C++ and uses a linked list structure for the graph.
- STATUS BAR:** The bottom status bar indicates the current line and column (Ln 47, Col 7), the number of spaces (4), the encoding (UTF-8), and the line ending (CRLF). It also shows the C++ language mode and the Prettier formatter.
- OUTPUT WINDOW:** A small output window is visible on the right side of the editor, displaying the text: 'Besthian Guido Rafael Simbolon 103112430258 IF - 12 - 06'.

Screenshot Output



```
PS C:\StrukDat\Modul 14\unguided2> cd "c:\StrukDat\Modul 14\unguided2\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main.exe }
=== Struktur Graph ===
H -> G F E D
G -> H C
F -> H C
E -> H B
D -> H B
C -> G F A
B -> E D A
A -> C B

=== DFS dari Node A ===
A C G H F E B D

=== BFS dari Node A ===
A C B G F E D H
PS C:\StrukDat\Modul 14\unguided2>
```

Best

Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06

Ln 3, Col 14 | 57 character Plain text | 100% | Window UTF-8

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](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.