

**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**

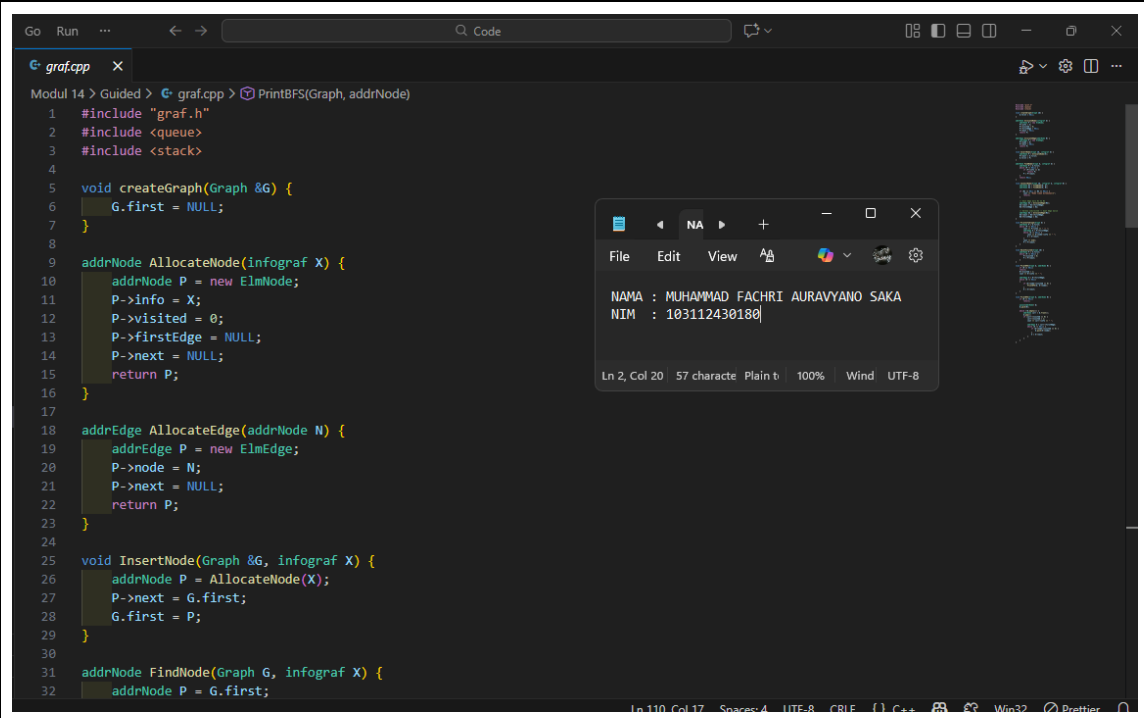
## A. Dasar Teori

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)
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(infograf 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->node = N;
21     P->next = NULL;
22     return P;
23 }
24
25 void InsertNode(Graph &G, infograf X) {
26     addrNode P = AllocateNode(X);
27     P->next = G.first;
28     G.first = P;
29 }
30
31 addrNode FindNode(Graph G, infograf X) {
32     addrNode P = G.first;
```

File Edit View A A

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

```
Go Run ... < -> Q Code
graf.cpp x
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
void InsertNode(Graph &G, infograf A) {
30
31     addrNode FindNode(Graph G, infograf X) {
32         addrNode P = G.first;
33         while (P != NULL) {
34             if (P->info == X)
35                 return P;
36             P = P->next;
37         }
38         return NULL;
39     }
40
41     void ConnectNode(Graph &G, infograf A, infograf B) {
42         addrNode N1 = FindNode(G, A);
43         addrNode N2 = FindNode(G, B);
44
45         if (N1 == NULL || N2 == NULL) {
46             cout << "Node tidak ditemukan\n";
47             return;
48         }
49
50         // Buat Edge daro N1 ke N2
51         addrEdge E1 = AllocateEdge(N2);
52         E1->next = N1->firstEdge;
53         N1->firstEdge = E1;
54
55         // Karena undirected -> buat Edge balik
56         addrEdge E2 = AllocateEdge(N1);
57         E2->next = N2->firstEdge;
58         N2->firstEdge = E2;
59     }
60
61     void PrintInfoGraph(Graph G) {
62         addrNode P = G.first;
63         while (P != NULL) {
64             cout << P->info << " -> ";
65             addrEdge E = P->firstEdge;
66             while (E != NULL) {
67                 cout << E->node->info << " ";
68                 E = E->next;
69             }
70             cout << endl;
71             P = P->next;
72         }
73     }
74
75     void ResetVisited(Graph &G) {
76         addrNode P = G.first;
77         while (P != NULL) {
78             P->visited = 0;
79             P = P->next;
80         }
81     }
82
83     void PrintDFS(Graph G, addrNode N) {
84         if (N == NULL)
85             return;
86         N->visited = 1;
87         cout << N->info << " ";
88
89         addrEdge E = N->firstEdge;
90         while (E != NULL) {
91             if (E->node->visited == 0) {
92                 PrintDFS(G, E->node);
93             }
94             E = E->next;
95         }
96     }
97
98     void PrintBFS(Graph G, addrNode N) {
99         if (N == NULL)
100             return;
101         N->visited = 1;
102         queue<addrNode> Q;
103         Q.push(N);
104         while (!Q.empty()) {
105             addrNode P = Q.front();
106             Q.pop();
107             cout << P->info << " ";
108             addrEdge E = P->firstEdge;
109             while (E != NULL) {
110                 if (E->node->visited == 0) {
111                     Q.push(E->node);
112                     E->node->visited = 1;
113                 }
114                 E = E->next;
115             }
116         }
117     }
118
119     void PrintGraph(Graph G) {
120         PrintInfoGraph(G);
121         ResetVisited(G);
122         PrintDFS(G, G.first);
123         PrintBFS(G, G.first);
124     }
125
126     void main() {
127         Graph G;
128         InsertNode(G, A);
129         InsertNode(G, B);
130         ConnectNode(G, A, B);
131         PrintGraph(G);
132     }
133 }
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

```
Go Run ... < -> Q Code
graf.cpp x
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
61 void PrintInfoGraph(Graph G) {
62     addrNode P = G.first;
63     while (P != NULL) {
64         cout << P->info << " -> ";
65         addrEdge E = P->firstEdge;
66         while (E != NULL) {
67             cout << E->node->info << " ";
68             E = E->next;
69         }
70         cout << endl;
71         P = P->next;
72     }
73 }
74
75 void ResetVisited(Graph &G) {
76     addrNode P = G.first;
77     while (P != NULL) {
78         P->visited = 0;
79         P = P->next;
80     }
81 }
82
83 void PrintDFS(Graph G, addrNode N) {
84     if (N == NULL)
85         return;
86     N->visited = 1;
87     cout << N->info << " ";
88
89     addrEdge E = N->firstEdge;
90     while (E != NULL) {
91         if (E->node->visited == 0) {
92             PrintDFS(G, E->node);
93         }
94         E = E->next;
95     }
96 }
97
98 void PrintBFS(Graph G, addrNode N) {
99     if (N == NULL)
100         return;
101     N->visited = 1;
102     queue<addrNode> Q;
103     Q.push(N);
104     while (!Q.empty()) {
105         addrNode P = Q.front();
106         Q.pop();
107         cout << P->info << " ";
108         addrEdge E = P->firstEdge;
109         while (E != NULL) {
110             if (E->node->visited == 0) {
111                 Q.push(E->node);
112                 E->node->visited = 1;
113             }
114             E = E->next;
115         }
116     }
117 }
118
119 void PrintGraph(Graph G) {
120     PrintInfoGraph(G);
121     ResetVisited(G);
122     PrintDFS(G, G.first);
123     PrintBFS(G, G.first);
124 }
125
126 void main() {
127     Graph G;
128     InsertNode(G, A);
129     InsertNode(G, B);
130     ConnectNode(G, A, B);
131     PrintGraph(G);
132 }
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

```
Go Run ... < -> Code
graf.cpp x
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
83 void PrintDFS(Graph G, addrNode N) {
84     addrEdge E = N->firstEdge;
85     while (E != NULL)
86     {
87         if (E->node->visited == 0) {
88             PrintDFS(G, E->node);
89         }
90         E = E->next;
91     }
92 }
93
94 void PrintBFS(Graph G, addrNode N) {
95     if (N == NULL)
96         return;
97
98     queue<addrNode> Q;
99     Q.push(N);
100
101     while (!Q.empty()) {
102         addrNode curr = Q.front();
103         Q.pop();
104         if (curr->visited == 0) {
105             curr->visited = 1;
106             cout << curr->info << " ";
107
108             addrEdge E = curr->firstEdge;
109             while (E != NULL) {
110                 if (E->node->visited == 0) {
111                     Q.push(E->node);
112                 }
113                 E = E->next;
114             }
115         }
116     }
117 }
118
119 while (!Q.empty()) {
120     addrNode curr = Q.front();
121     Q.pop();
122     if (curr->visited == 0) {
123         curr->visited = 1;
124         cout << curr->info << " ";
125
126         addrEdge E = curr->firstEdge;
127         while (E != NULL) {
128             if (E->node->visited == 0) {
129                 Q.push(E->node);
130             }
131             E = E->next;
132         }
133     }
134 }
```

File Edit View A 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

Graf.h

```
Modul 14 > Guided > C graf.h > FindNode(Graph, infograf)
1  #ifndef GRAF_H_INCLUDED
2  #define GRAF_H_INCLUDED
3
4  #include <iostream>
5  using namespace std;
6
7  typedef char infograf;
8
9  struct ElmNode;
10 struct ElmEdge;
11
12 typedef ElmNode* addrNode;
13 typedef ElmEdge* addrEdge;
14
15 struct ElmNode {
16     infograf 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);
```

```
30
31 // PRIMITIF GRAPH
32 void createGraph(Graph &G);
33 addrNode AllocateNode(infograf X);
34 addrEdge AllocateEdge(addrNode N);
35
36 void InsertNode(Graph &G, infograf X);
37 addrNode FindNode(Graph G, infograf X);
38
39 void ConnectNode(Graph &G, infograf A, infograf B);
40
41 void PrintGraph(Graph G);
42
43 // Traversal
44 void ResetVisited(Graph &G);
45 void PrintDFS(Graph G, addrNode N);
46 void PrintBFS(Graph G, addrNode N);
47
48 #endif
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

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;
35    return 0;
36 }
```

## 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>
```

## Deskripsi:

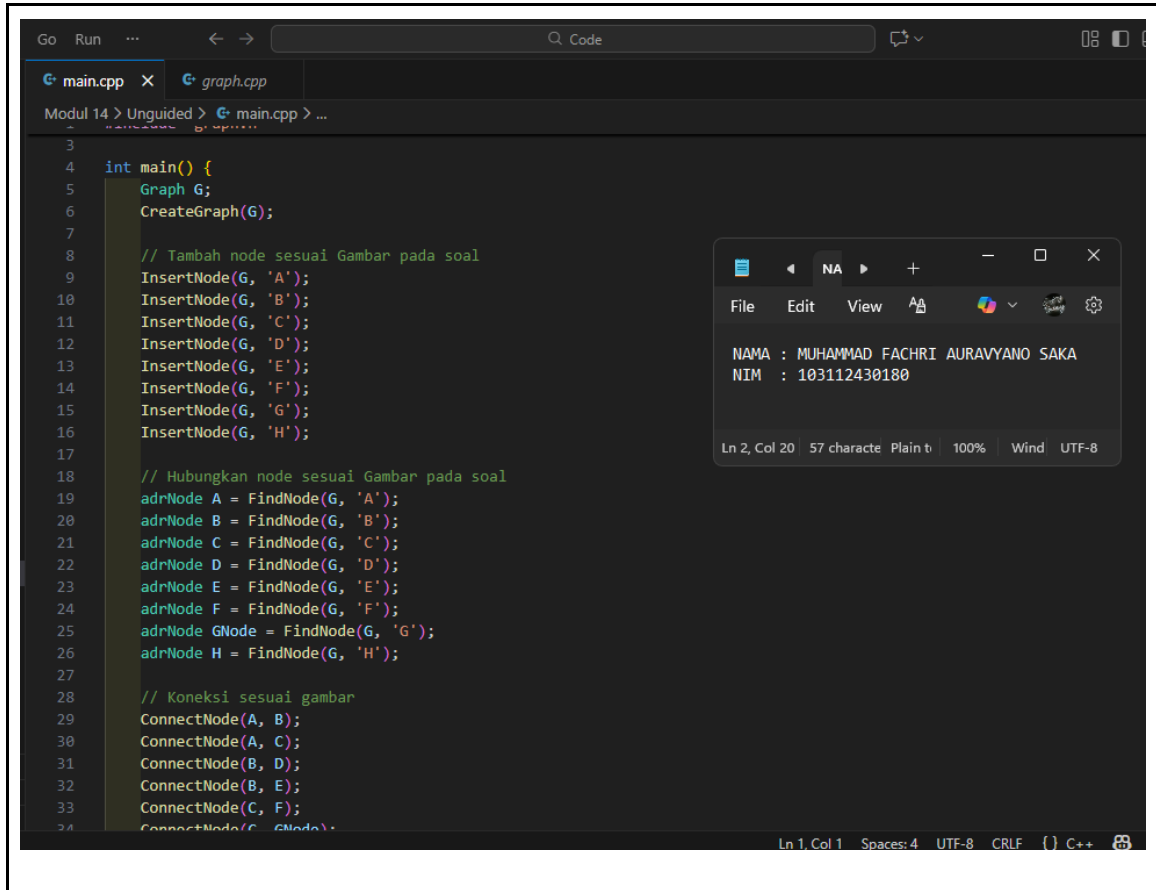
Program ini adalah implementasi struktur data Graph tak berarah (Undirected Graph) menggunakan bahasa C++ dengan representasi daftar ketetanggaan (adjacency list). Program ini mendefinisikan tipe data dinamis untuk Node (simpul) dan Edge (sisi/jalur) menggunakan pointer, di mana setiap node memiliki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



The screenshot shows a C++ IDE with two tabs: `main.cpp` and `graph.cpp`. The `main.cpp` tab is active, displaying the following code:

```
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);
```

On the right side of the IDE, there is a small output window titled "NA" with a menu bar (File, Edit, View) and various icons. It contains the following text:

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

At the bottom of the output window, it shows "Ln 2, Col 20 | 57 character | Plain text | 100% | Window | UTF-8".

```
Go Run ... < -> Code
main.cpp x graph.cpp
Modul 14 > Unguided > main.cpp > ...
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, H);
36 ConnectNode(E, H);
37 ConnectNode(F, H);
38 ConnectNode(GNode, H);
39
40 // Tampilkan struktur graph
41 cout << "=== Struktur Graph ===" << endl;
42 PrintInfoGraph(G);
43
44 // Soal 2: DFS
45 cout << "\n=== DFS dari Node A ===" << endl;
46 ResetVisited(G);
47 PrintDFS(G, A);
48 cout << endl;
49
50 // Soal 3: BFS
51 cout << "\n=== BFS dari Node A ===" << endl;
52 ResetVisited(G);
53 PrintBFS(G, A);
54 cout << endl;
55
56 return 0;
57 }
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

graph.cpp

```
Go Run ... < -> Code
main.cpp x graph.cpp
Modul 14 > Unguided > graph.cpp > ...
1 #include "graph.h"
2 #include <queue>
3
4 void CreateGraph(Graph &G) {
5     G.first = NULL;
6 }
7
8 adrNode AllocateNode(infoGraph X) {
9     adrNode P = new ElmNode;
10    P->info = X;
11    P->visited = 0;
12    P->firstEdge = NULL;
13    P->next = NULL;
14    return P;
15 }
16
17 adrEdge AllocateEdge(adrNode N) {
18     adrEdge P = new ElmEdge;
19    P->node = N;
20    P->next = NULL;
21    return P;
22 }
23
24 void InsertNode(Graph &G, infoGraph X) {
25     adrNode P = AllocateNode(X);
26    P->next = G.first;
27    G.first = P;
28 }
29
30 adrNode FindNode(Graph G, infoGraph X) {
31     adrNode P = G.first;
32     while (P != NULL) {
```

File Edit View

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

Ln 1 Col 1 Spaces: 4 UTF-8 CR LF {} C++



The screenshot shows a C++ IDE with a file named `graph.cpp` open. The code implements a graph structure with nodes and edges. The `FindNode` function searches for a node with a specific info. The `ConnectNode` function adds an edge between two nodes. The `PrintInfoGraph` function prints the graph's structure.

```

30  adrNode FindNode(Graph G, infoGraph X) {
31      adrNode P = G.first;
32      while (P != NULL) {
33          if (P->info == X)
34              return P;
35          P = P->next;
36      }
37      return NULL;
38  }
39
40  void ConnectNode(adrNode N1, adrNode N2) {
41      if (N1 == NULL || N2 == NULL) {
42          return;
43      }
44
45      // Edge dari N1 ke N2
46      adrEdge E1 = AllocateEdge(N2);
47      E1->next = N1->firstEdge;
48      N1->firstEdge = E1;
49
50      // Edge dari N2 ke N1 (undirected)
51      adrEdge E2 = AllocateEdge(N1);
52      E2->next = N2->firstEdge;
53      N2->firstEdge = E2;
54  }
55
56  void PrintInfoGraph(Graph G) {
57      adrNode P = G.first;
58      while (P != NULL) {
59          cout << P->info << " -> ";
60          adrEdge E = P->firstEdge;
61          while (E != NULL) {

```

A terminal window is open on the right, displaying the output of the program:

```

Nama : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180

```

The terminal also shows the status bar: "Ln 2, Col 20 57 caracte Plain t 100% Wind UTF-8".

```
Go Run ... < -> Code
main.cpp graph.cpp x
Modul 14 > Unguided > graph.cpp > ...
56 void PrintInfoGraph(Graph G) {
57     while (P != NULL) {
58         adrEdge E = P->firstEdge;
59         while (E != NULL) {
60             cout << E->node->info << " ";
61             E = E->next;
62         }
63         cout << endl;
64         P = P->next;
65     }
66 }
67
68 void ResetVisited(Graph &G) {
69     adrNode P = G.first;
70     while (P != NULL) {
71         P->visited = 0;
72         P = P->next;
73     }
74 }
75
76 // Soal 2: Prosedur PrintDFS
77 void PrintDFS(Graph G, adrNode N) {
78     if (N == NULL)
79         return;
80
81     N->visited = 1;
82     cout << N->info << " ";
83
84     adrEdge E = N->firstEdge;
85     while (E != NULL) {
86         if (E->node->visited == 0) {
87             PrintDFS(G, E->node);
88         }
89         E = E->next;
90     }
91 }
```

File Edit View A A

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 caractere Plain t 100% Wind UTF-8

```
79 void PrintDFS(Graph G, adrNode N) {
86     adrEdge E = N->firstEdge;
87     while (E != NULL) {
88         if (E->node->visited == 0) {
89             PrintDFS(G, E->node);
90         }
91         E = E->next;
92     }
93 }
94
95 // Soal 3: Prosedur PrintBFS
96 void PrintBFS(Graph G, adrNode N) {
97     if (N == NULL)
98         return;
99
100     queue<adrNode> Q;
101     Q.push(N);
102
103     while (!Q.empty()) {
104         adrNode curr = Q.front();
105         Q.pop();
106
107         if (curr->visited == 0) {
108             curr->visited = 1;
109             cout << curr->info << " ";
110
111             adrEdge E = curr->firstEdge;
112             while (E != NULL) {
113                 if (E->node->visited == 0) {
114                     Q.push(E->node);
115                 }
116                 E = E->next;
117             }
118         }
119     }
120 }
```

File Edit View A 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} C++

```
102
103     while (!Q.empty()) {
104         adrNode curr = Q.front();
105         Q.pop();
106
107         if (curr->visited == 0) {
108             curr->visited = 1;
109             cout << curr->info << " ";
110
111             adrEdge E = curr->firstEdge;
112             while (E != NULL) {
113                 if (E->node->visited == 0) {
114                     Q.push(E->node);
115                 }
116                 E = E->next;
117             }
118         }
119     }
120 }
```

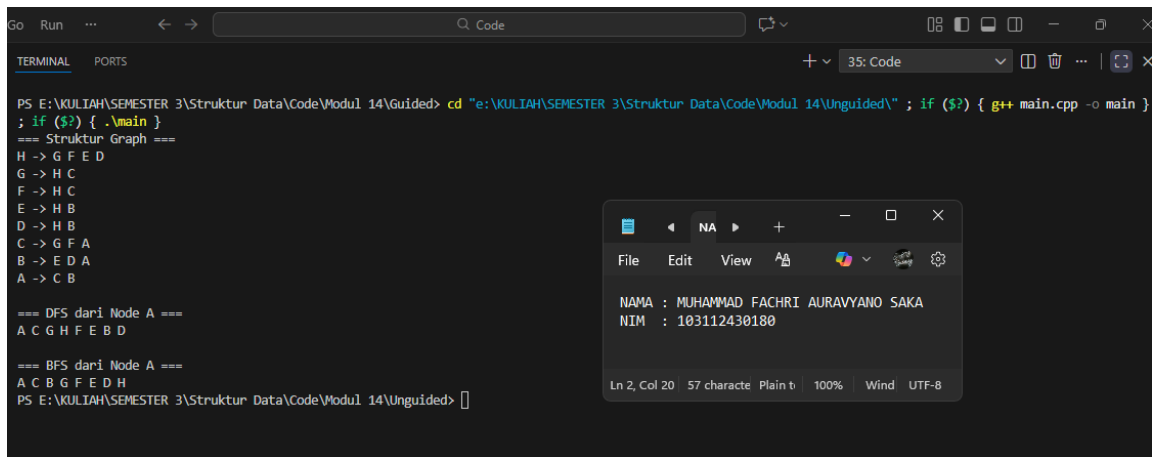
File Edit View A 100% Wind UTF-8

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA  
NIM : 103112430180

Ln 2, Col 20 57 character Plain t 100% Wind UTF-8

graph.h





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

### 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](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.