

LAPORAN PRAKTIKUM

Modul 14

“Graph”



Disusun Oleh:

**Benedictus Qsota Noventino Baru - 2311104029
S1SE07A**

Dosen :

Yudha Islami Sulistya, S.Kom., M.Cs

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO
2024**

Latihan Soal Modul

Nomor 1

File graph.h

```
1  #ifndef GRAPH_H
2  #define GRAPH_H
3
4  typedef char infoGraph;
5
6  // Struktur untuk elemen Edge (sisi)
7  struct ElmEdge {
8      struct ElmNode* node;
9      ElmEdge* nextEdge;
10 };
11
12 // Struktur untuk elemen Node (simpul)
13 struct ElmNode {
14     infoGraph info;
15     int visited;
16     ElmEdge* firstEdge;
17     ElmNode* nextNode;
18 };
19
20 // Struktur untuk Graph
21 struct Graph {
22     ElmNode* firstNode;
23 };
24
25 // Prototipe fungsi
26 void CreateGraph(Graph &G);
27 void InsertNode(Graph &G, infoGraph X);
28 ElmNode* FindNode(Graph G, infoGraph X);
29 void ConnectNode(ElmNode* N1, ElmNode* N2);
30 void PrintInfoGraph(Graph G);
31 void PrintDFS(Graph G, ElmNode* N);
32 void PrintBFS(Graph G, ElmNode* N);
33
34 #endif
35
```

snappify.com

File **graph.h** berfungsi sebagai header file yang mendeklarasikan struktur data dan prototipe fungsi untuk digunakan dalam program. Dalam konteks ADT Graph, file ini mendeklarasikan struktur seperti Graph, ElmNode, dan ElmEdge, yang merepresentasikan elemen-elemen dari graph. Selain itu, file ini juga berisi prototipe fungsi-fungsi penting seperti CreateGraph, InsertNode, ConnectNode, PrintInfoGraph, PrintDFS, dan PrintBFS. Dengan deklarasi ini, compiler mengetahui fungsi-fungsi yang tersedia dan bagaimana menggunakannya tanpa harus melihat implementasinya.

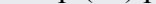
Fungsi lainnya adalah untuk modularisasi program, di mana logika implementasi disimpan di file terpisah (**graph.cpp**), sedangkan deklarasi terpusat di satu tempat, yaitu **graph.h**. Hal ini membuat program lebih terstruktur, memudahkan pemeliharaan, dan menghindari deklarasi ulang di beberapa file. Dengan adanya header guard (**#ifndef**, **#define**, **#endif**), file ini juga memastikan tidak ada deklarasi ganda yang dapat menyebabkan error. Header file ini memungkinkan komunikasi yang efisien antara file implementasi seperti **main.cpp** dan **graph.cpp**.

Kesimpulan

Fungsi utama **graph.h** adalah:

1. Mendeklarasikan struktur data.
2. Mendeklarasikan prototipe fungsi.
3. Membuat program lebih modular, terorganisir, dan mudah dikelola.
4. Memungkinkan komunikasi antar file implementasi.

File graph.cpp

Gambar lebih jelas bisa akses di link ini : 

File **graph.cpp** berfungsi sebagai implementasi dari fungsi-fungsi yang telah dideklarasikan di file **graph.h**. File ini merealisasikan logika operasional dari struktur data graph, seperti membuat graph baru (**CreateGraph**), menambahkan node (**InsertNode**), menyambungkan node dengan edge (**ConnectNode**), mencetak informasi graph (**PrintInfoGraph**), serta melakukan penelusuran graph menggunakan metode DFS (**PrintDFS**) dan BFS (**PrintBFS**). Dengan menyimpan implementasi di file ini, program menjadi lebih modular karena memisahkan deklarasi dan implementasi.

File ini bekerja sama dengan **graph.h** untuk memastikan kode tetap terorganisasi. Semua fungsi didefinisikan secara terpisah, sehingga detail implementasi tersembunyi dari file lain seperti **main.cpp**. Ini meningkatkan keterbacaan dan memudahkan debugging, karena perubahan pada logika hanya perlu dilakukan di file ini. Selain itu, penggunaan file **graph.cpp** memungkinkan pengembang untuk fokus pada logika spesifik tanpa mencampuradukkan dengan program utama.

```

1 #include <iostream>
2 #include <queue>
3 #include <graph.h>
4 using namespace std;
5 // Memulai proses
6 void CetakGraph(Graph G) {
7     G.firstNode = nullptr;
8 }
9 // Memeriksa dua simpul apakah
10 // satu terhubung dengan sisi, infoGraph X {
11     Elomode nextNode = new Elomode;
12     nextNode->info = X;
13     nextNode->visited = 0;
14     nextNode->firstEdge = nullptr;
15     nextNode->nextNode = G.firstNode;
16     G.firstNode = nextNode;
17 }
18 // Memulai mode BreadthSearch Informasi
19 Elomode ElomodeNext(G, infoGraph X) {
20     Elomode current = G.firstNode;
21     while (current != nullptr) {
22         if (current->info == X) {
23             return current;
24         }
25         current = current->nextNode;
26     }
27     return nullptr; // Mode tidak ditemukan
28 }
29 // Memeriksa dua mode apakah satu
30 // Elomode Connect(Elomode N1, Elomode N2) {
31     if (N1 == nullptr && N2 == nullptr) {
32         return nullptr;
33     }
34     Elomode nextNode = new Elomode;
35     nextNode->info = N1;
36     nextNode->nextNode = N1->firstEdge;
37     N1->firstEdge = nextNode;
38 }
39 // Memeriksa info dari N2 ke N1 (sistem graph ini berarah)
40 Elomode nextNode = new Elomode;
41 nextNode->info = N1;
42 nextNode->nextNode = N1->firstEdge;
43 N1->firstEdge = nextNode;
44 }
45 // Memeriksa informasi simpul
46 void PrintInfoGraph(Graph G) {
47     Elomode currentNode = G.firstNode;
48     while (currentNode != nullptr) {
49         cout << "Mode : " << currentNode->info << " ";
50         while (currentNode->nextNode != nullptr) {
51             cout << "currentNode->nextNode->info << " ";
52             currentNode->nextNode->info << " ";
53             currentNode = currentNode->nextNode;
54         }
55         cout << endl;
56         currentNode = currentNode->nextNode;
57     }
58 }
59 // Mengisi informasi untuk DFS
60 void Cetak(Elomode N) {
61     if (N == nullptr || N->visited == 1) {
62         return;
63     }
64 }
65 // Menjalankan algoritma rekursif dan menjajal
66 N->visited = 1;
67 // Menjalankan algoritma rekursif
68 cout << N->info << " ";
69 // Menjalankan algoritma rekursif dan menjajal
70 Elomode currentNode = N->firstEdge;
71 while (currentNode != nullptr) {
72     if (currentNode->visited == 0) {
73         currentNode->nextNode->info << " ";
74     }
75 }
76 // Menjalankan algoritma rekursif dan menjajal
77 void PrintDFS(Graph G) {
78     // Menjalankan algoritma rekursif dan menjajal
79     Elomode currentNode = G.firstNode;
80     while (currentNode != nullptr) {
81         currentNode->visited = 0;
82         currentNode = currentNode->nextNode;
83     }
84 }
85 // Mengisi fungsi DFS untuk mencari mode N
86 cout << "hasil penelusuran DFS : ";
87 DFS(N);
88 cout << endl;
89 }
90 // Menjalankan algoritma rekursif dan menjajal
91 void PrintDFS(Graph G) {
92     if (N == nullptr) {
93         cout << "Mode tidak ditemukan" << endl;
94         return;
95     }
96 }
97 // Menjalankan algoritma rekursif dan menjajal
98 void PrintDFS(Graph G) {
99     Elomode currentNode = G.firstNode;
100     while (currentNode != nullptr) {
101         currentNode->visited = 0;
102         currentNode = currentNode->nextNode;
103     }
104 }
105 // Menjalankan algoritma rekursif dan menjajal
106 queue<Elomode> Q;
107 // Menjalankan algoritma rekursif dan menjajal
108 N->visited = 1;
109 Q.push(N);
110 cout << "hasil penelusuran BFS : ";
111 while (!Q.empty()) {
112     Elomode current = Q.front();
113     Q.pop();
114 }
115 // Menjalankan algoritma rekursif dan menjajal
116 cout << current->info << " ";
117 // Menjalankan algoritma rekursif dan menjajal
118 // Menjalankan algoritma rekursif dan menjajal
119 while (currentNode != nullptr) {
120     if (currentNode->nextNode->visited == 0) {
121         Q.push(currentNode->nextNode);
122     }
123     currentNode->nextNode->info << " ";
124 }
125 cout << endl;
126 }
127 // Menjalankan algoritma rekursif dan menjajal
128 // Menjalankan algoritma rekursif dan menjajal
129 // Menjalankan algoritma rekursif dan menjajal
130 // Menjalankan algoritma rekursif dan menjajal
131 // Menjalankan algoritma rekursif dan menjajal
132 // Menjalankan algoritma rekursif dan menjajal
133 // Menjalankan algoritma rekursif dan menjajal
134 // Menjalankan algoritma rekursif dan menjajal
135 // Menjalankan algoritma rekursif dan menjajal
136 // Menjalankan algoritma rekursif dan menjajal
137 // Menjalankan algoritma rekursif dan menjajal
138 // Menjalankan algoritma rekursif dan menjajal
139 // Menjalankan algoritma rekursif dan menjajal
140 // Menjalankan algoritma rekursif dan menjajal
141 // Menjalankan algoritma rekursif dan menjajal
142 // Menjalankan algoritma rekursif dan menjajal
143 // Menjalankan algoritma rekursif dan menjajal
144 // Menjalankan algoritma rekursif dan menjajal
145 // Menjalankan algoritma rekursif dan menjajal
146 // Menjalankan algoritma rekursif dan menjajal
147 // Menjalankan algoritma rekursif dan menjajal
148 // Menjalankan algoritma rekursif dan menjajal
149 // Menjalankan algoritma rekursif dan menjajal
150 // Menjalankan algoritma rekursif dan menjajal
151 // Menjalankan algoritma rekursif dan menjajal
152 // Menjalankan algoritma rekursif dan menjajal
153 // Menjalankan algoritma rekursif dan menjajal
154 // Menjalankan algoritma rekursif dan menjajal
155 // Menjalankan algoritma rekursif dan menjajal
156 // Menjalankan algoritma rekursif dan menjajal
157 // Menjalankan algoritma rekursif dan menjajal
158 // Menjalankan algoritma rekursif dan menjajal
159 // Menjalankan algoritma rekursif dan menjajal
160 // Menjalankan algoritma rekursif dan menjajal
161 // Menjalankan algoritma rekursif dan menjajal
162 // Menjalankan algoritma rekursif dan menjajal
163 // Menjalankan algoritma rekursif dan menjajal
164 // Menjalankan algoritma rekursif dan menjajal
165 // Menjalankan algoritma rekursif dan menjajal
166 // Menjalankan algoritma rekursif dan menjajal
167 // Menjalankan algoritma rekursif dan menjajal
168 // Menjalankan algoritma rekursif dan menjajal
169 // Menjalankan algoritma rekursif dan menjajal
170 // Menjalankan algoritma rekursif dan menjajal
171 // Menjalankan algoritma rekursif dan menjajal
172 // Menjalankan algoritma rekursif dan menjajal
173 // Menjalankan algoritma rekursif dan menjajal
174 // Menjalankan algoritma rekursif dan menjajal
175 // Menjalankan algoritma rekursif dan menjajal
176 // Menjalankan algoritma rekursif dan menjajal
177 // Menjalankan algoritma rekursif dan menjajal
178 // Menjalankan algoritma rekursif dan menjajal
179 // Menjalankan algoritma rekursif dan menjajal
180 // Menjalankan algoritma rekursif dan menjajal
181 // Menjalankan algoritma rekursif dan menjajal
182 // Menjalankan algoritma rekursif dan menjajal
183 // Menjalankan algoritma rekursif dan menjajal
184 // Menjalankan algoritma rekursif dan menjajal
185 // Menjalankan algoritma rekursif dan menjajal
186 // Menjalankan algoritma rekursif dan menjajal
187 // Menjalankan algoritma rekursif dan menjajal
188 // Menjalankan algoritma rekursif dan menjajal
189 // Menjalankan algoritma rekursif dan menjajal
190 // Menjalankan algoritma rekursif dan menjajal
191 // Menjalankan algoritma rekursif dan menjajal
192 // Menjalankan algoritma rekursif dan menjajal
193 // Menjalankan algoritma rekursif dan menjajal
194 // Menjalankan algoritma rekursif dan menjajal
195 // Menjalankan algoritma rekursif dan menjajal
196 // Menjalankan algoritma rekursif dan menjajal
197 // Menjalankan algoritma rekursif dan menjajal
198 // Menjalankan algoritma rekursif dan menjajal
199 // Menjalankan algoritma rekursif dan menjajal
200 // Menjalankan algoritma rekursif dan menjajal
201 // Menjalankan algoritma rekursif dan menjajal
202 // Menjalankan algoritma rekursif dan menjajal
203 // Menjalankan algoritma rekursif dan menjajal
204 // Menjalankan algoritma rekursif dan menjajal
205 // Menjalankan algoritma rekursif dan menjajal
206 // Menjalankan algoritma rekursif dan menjajal
207 // Menjalankan algoritma rekursif dan menjajal
208 // Menjalankan algoritma rekursif dan menjajal
209 // Menjalankan algoritma rekursif dan menjajal
210 // Menjalankan algoritma rekursif dan menjajal
211 // Menjalankan algoritma rekursif dan menjajal
212 // Menjalankan algoritma rekursif dan menjajal
213 // Menjalankan algoritma rekursif dan menjajal
214 // Menjalankan algoritma rekursif dan menjajal
215 // Menjalankan algoritma rekursif dan menjajal
216 // Menjalankan algoritma rekursif dan menjajal
217 // Menjalankan algoritma rekursif dan menjajal
218 // Menjalankan algoritma rekursif dan menjajal
219 // Menjalankan algoritma rekursif dan menjajal
220 // Menjalankan algoritma rekursif dan menjajal
221 // Menjalankan algoritma rekursif dan menjajal
222 // Menjalankan algoritma rekursif dan menjajal
223 // Menjalankan algoritma rekursif dan menjajal
224 // Menjalankan algoritma rekursif dan menjajal
225 // Menjalankan algoritma rekursif dan menjajal
226 // Menjalankan algoritma rekursif dan menjajal
227 // Menjalankan algoritma rekursif dan menjajal
228 // Menjalankan algoritma rekursif dan menjajal
229 // Menjalankan algoritma rekursif dan menjajal
230 // Menjalankan algoritma rekursif dan menjajal
231 // Menjalankan algoritma rekursif dan menjajal
232 // Menjalankan algoritma rekursif dan menjajal
233 // Menjalankan algoritma rekursif dan menjajal
234 // Menjalankan algoritma rekursif dan menjajal
235 // Menjalankan algoritma rekursif dan menjajal
236 // Menjalankan algoritma rekursif dan menjajal
237 // Menjalankan algoritma rekursif dan menjajal
238 // Menjalankan algoritma rekursif dan menjajal
239 // Menjalankan algoritma rekursif dan menjajal
240 // Menjalankan algoritma rekursif dan menjajal
241 // Menjalankan algoritma rekursif dan menjajal
242 // Menjalankan algoritma rekursif dan menjajal
243 // Menjalankan algoritma rekursif dan menjajal
244 // Menjalankan algoritma rekursif dan menjajal
245 // Menjalankan algoritma rekursif dan menjajal
246 // Menjalankan algoritma rekursif dan menjajal
247 // Menjalankan algoritma rekursif dan menjajal
248 // Menjalankan algoritma rekursif dan menjajal
249 // Menjalankan algoritma rekursif dan menjajal
250 // Menjalankan algoritma rekursif dan menjajal
251 // Menjalankan algoritma rekursif dan menjajal
252 // Menjalankan algoritma rekursif dan menjajal
253 // Menjalankan algoritma rekursif dan menjajal
254 // Menjalankan algoritma rekursif dan menjajal
255 // Menjalankan algoritma rekursif dan menjajal
256 // Menjalankan algoritma rekursif dan menjajal
257 // Menjalankan algoritma rekursif dan menjajal
258 // Menjalankan algoritma rekursif dan menjajal
259 // Menjalankan algoritma rekursif dan menjajal
260 // Menjalankan algoritma rekursif dan menjajal
261 // Menjalankan algoritma rekursif dan menjajal
262 // Menjalankan algoritma rekursif dan menjajal
263 // Menjalankan algoritma rekursif dan menjajal
264 // Menjalankan algoritma rekursif dan menjajal
265 // Menjalankan algoritma rekursif dan menjajal
266 // Menjalankan algoritma rekursif dan menjajal
267 // Menjalankan algoritma rekursif dan menjajal
268 // Menjalankan algoritma rekursif dan menjajal
269 // Menjalankan algoritma rekursif dan menjajal
270 // Menjalankan algoritma rekursif dan menjajal
271 // Menjalankan algoritma rekursif dan menjajal
272 // Menjalankan algoritma rekursif dan menjajal
273 // Menjalankan algoritma rekursif dan menjajal
274 // Menjalankan algoritma rekursif dan menjajal
275 // Menjalankan algoritma rekursif dan menjajal
276 // Menjalankan algoritma rekursif dan menjajal
277
```

File main.cpp

```
1  #include <iostream>
2  #include "graph.h"
3
4  using namespace std;
5
6  int main() {
7      Graph G;
8      CreateGraph(G);
9
10     // Tambahkan 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
18     ElmNode* A = FindNode(G, 'A');
19     ElmNode* B = FindNode(G, 'B');
20     ElmNode* C = FindNode(G, 'C');
21     ElmNode* D = FindNode(G, 'D');
22     ElmNode* E = FindNode(G, 'E');
23
24     ConnectNode(A, B);
25     ConnectNode(A, C);
26     ConnectNode(B, D);
27     ConnectNode(C, E);
28     ConnectNode(D, E);
29
30     // Cetak graph
31     PrintInfoGraph(G);
32
33     // Penelusuran DFS
34     PrintDFS(G, A);
35
36     // Penelusuran BFS
37     PrintBFS(G, A);
38
39     return 0;
40 }
41
```

snappify.com

Kode ini membuat sebuah graf tak terarah, menambahkan lima node ('A', 'B', 'C', 'D', 'E'), dan menghubungkannya dengan edge menggunakan fungsi InsertNode dan ConnectNode. Setelah graf terbentuk, program mencetak informasi graf menggunakan PrintInfoGraph.

Selanjutnya, program melakukan penelusuran graf dengan algoritma Depth First Search (DFS) dan Breadth First Search (BFS) mulai dari node 'A' menggunakan fungsi PrintDFS dan PrintBFS.

Output :

```
noven@NOVEN MINGW64 ~/Desktop/Prak SD/14_Graph/Guided Modul
$ ./program
Node E: D C
Node D: E B
Node C: E A
Node B: D A
Node A: C B
Hasil penelusuran DFS: A C E D B
Hasil penelusuran BFS: A C B E D
```

Latihan Soal dari Asprak

Soal 1 (File Soal1.cpp)

```
1  #include <iostream>
2  #include <vector>
3  #include <iomanip> // Untuk format output tabel
4
5  using namespace std;
6
7  int main() {
8      int n;
9
10     // Meminta jumlah simpul dari pengguna
11     cout << "Silakan masukkan jumlah simpul: ";
12     cin >> n;
13
14     vector<string> nodes(n);
15
16     // Meminta nama simpul
17     for (int i = 0; i < n; i++) {
18         cout << "Simpul " << i + 1 << ": ";
19         cin >> nodes[i];
20     }
21
22     // Membuat matriks bobot dengan ukuran n x n
23     vector<vector<int>> weights(n, vector<int>(n, 0));
24
25     cout << "\nSilakan masukkan bobot antar simpul\n";
26     for (int i = 0; i < n; i++) {
27         for (int j = 0; j < n; j++) {
28             if (i != j) { // Jika bukan simpul ke diri sendiri
29                 cout << nodes[i] << " → " << nodes[j] << " = ";
30                 cin >> weights[i][j];
31             } else {
32                 weights[i][j] = 0; // Bobot ke diri sendiri adalah 0
33             }
34         }
35     }
36
37     // Menampilkan matriks bobot
38     cout << "\nMatriks Bobot:\n";
39     cout << setw(10) << " "; // Header baris kosong
40     for (const auto& node : nodes) {
41         cout << setw(10) << node;
42     }
43     cout << endl;
44
45     for (int i = 0; i < n; i++) {
46         cout << setw(10) << nodes[i]; // Nama simpul di sisi kiri
47         for (int j = 0; j < n; j++) {
48             cout << setw(10) << weights[i][j];
49         }
50         cout << endl;
51     }
52
53     return 0;
54 }
55
```

Soal 2 (File Soal2.cpp)

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6      int n, m;
7
8      // Meminta jumlah simpul dan jumlah sisi
9      cout << "Masukkan jumlah simpul: ";
10     cin >> n;
11     cout << "Masukkan jumlah sisi: ";
12     cin >> m;
13
14     // Inisialisasi adjacency matrix dengan nilai 0
15     vector<vector<int>> adjMatrix(n, vector<int>(n, 0));
16
17     // Meminta pasangan simpul untuk setiap sisi
18     cout << "Masukkan pasangan simpul:\n";
19     for (int i = 0; i < m; i++) {
20         int u, v;
21         cin >> u >> v;
22         // Karena graf tidak berarah, simetri harus dijaga
23         adjMatrix[u - 1][v - 1] = 1;
24         adjMatrix[v - 1][u - 1] = 1;
25     }
26
27     // Menampilkan adjacency matrix
28     cout << "\nAdjacency Matrix:\n";
29     for (int i = 0; i < n; i++) {
30         for (int j = 0; j < n; j++) {
31             cout << adjMatrix[i][j] << " ";
32         }
33         cout << endl;
34     }
35
36     return 0;
37 }
38
```