LAPORAN PRAKTIKUM

MODUL 14

GRAPH

Disusun Oleh:

Rizaldy Aulia Rachman (2311104051)

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

1. Memahami konsep *graph*
2. Mengimplementasikan *graph* dengan menggunakan *pointer*

II. LANDASAN TEORI

2.1 Pengertian

Graph merupakan himpunan tidak kosong dari *node* (*vertec*) dan garis penghubung (*edge*). Contoh sederhana tentang *graph*, yaitu antara Tempat Kost Anda dengan *Common Lab*. Tempat Kost Anda dan *Common Lab* merupakan *node* (*vertec*). Jalan yang menghubungkan tempat Kost dan *Common Lab* merupakan garis penghubung antara keduanya (*edge*).

2.2 Jenis-Jenis Graph

- **Tidak Berarah (Undirected Graph)**: Sisi tanpa arah, hubungan antara simpul bersifat dua arah.
- **Berarah (Directed Graph)**: Sisi memiliki arah tertentu, hubungan antar simpul bersifat satu arah.
- **Berbobot (Weighted Graph)**: Sisi memiliki bobot atau nilai tertentu, seperti jarak atau biaya.
- **Tak Berbobot (Unweighted Graph)**: Sisi tidak memiliki bobot.

2.3 Operasi Dasar pada Graf

- **Menambahkan simpul atau sisi**: Menambahkan simpul baru atau menghubungkan dua simpul dengan sisi.
- **Traversal graf**:
  - **Breadth-First Search (BFS)**: Traversal berdasarkan level, menggunakan queue.
  - **Depth-First Search (DFS)**: Traversal sedalam mungkin sebelum mundur, menggunakan stack atau rekursi.
- **Mencari jalur terpendek**: Menggunakan algoritma seperti Dijkstra, Bellman-Ford, atau Floyd-Warshall.

III. GUIDED

Code:

```cpp
#include <iostream>
#include <queue>

using namespace std;

struct ElmNode;

struct ElmEdge {
    ElmNode *Node;
    ElmEdge *Next;
};

struct ElmNode {
    char info;
    bool visited;
    ElmEdge *firstEdge;
    ElmNode *Next;
};

struct Graph {
    ElmNode *first;
};

void CreateGraph(Graph &G) {
    G.first = NULL;
}

void InsertNode(Graph &G, char X) {
    ElmNode *newNode = new ElmNode;
    newNode->info = X;
    newNode->visited = false;
    newNode->firstEdge = NULL;
    newNode->Next = NULL;

    if (G.first == NULL) {
        G.first = newNode;
    } else {
        ElmNode *temp = G.first;
        while (temp->Next != NULL) {
            temp = temp->Next;
        }
        temp->Next = newNode;
    }
}

void ConnectNode(ElmNode *N1, ElmNode *N2) {
    ElmEdge *newEdge = new ElmEdge;
    newEdge->Node = N2;
    newEdge->Next = N1->firstEdge;
    N1->firstEdge = newEdge;
}

void PrintInfoGraph(Graph G) {
    ElmNode *temp = G.first;
    while (temp != NULL) {
        cout << temp->info << " ";
        temp = temp->Next;
    }
    cout << endl;
}

void ResetVisited(Graph &G) {
    ElmNode *temp = G.first;
    while (temp != NULL) {
        temp->visited = false;
        temp = temp->Next;
    }
}

void PrintDFS(Graph G, ElmNode *N) {
    if (N == NULL) {
        return;
    }
    N->visited = true;
    cout << N->info << " ";
    ElmEdge *edge = N->firstEdge;
    while (edge != NULL) {
        if (!edge->Node->visited) {
            PrintDFS(G, edge->Node);
        }
        edge = edge->Next;
    }
}

void PrintBFS(Graph G, ElmNode *N) {
    queue<ElmNode*> q;
    q.push(N);
    N->visited = true;

    while (!q.empty()) {
        ElmNode *current = q.front();
        q.pop();
        cout << current->info << " ";

        ElmEdge *edge = current->firstEdge;
        while (edge != NULL) {
            if (!edge->Node->visited) {
                edge->Node->visited = true;
                q.push(edge->Node);
            }
            edge = edge->Next;
        }
    }
}

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

    InsertNode(G, 'A');
    InsertNode(G, 'B');
    InsertNode(G, 'C');
    InsertNode(G, 'D');
    InsertNode(G, 'E');
    InsertNode(G, 'F');
    InsertNode(G, 'G');
    InsertNode(G, 'H');

    ElmNode *A = G.first;
    ElmNode *B = A->Next;
    ElmNode *C = B->Next;
    ElmNode *D = C->Next;
    ElmNode *E = D->Next;
    ElmNode *F = E->Next;
    ElmNode *G1 = F->Next;
    ElmNode *H = G1->Next;

    ConnectNode(A, B);
    ConnectNode(A, C);
    ConnectNode(B, D);
    ConnectNode(B, E);
    ConnectNode(C, F);
    ConnectNode(C, G1);
    ConnectNode(D, H);

    cout << "DFS traversal: ";
    ResetVisited(G);
    PrintDFS(G, A);
    cout << endl;

    cout << "BFS traversal: ";
    ResetVisited(G);
    PrintBFS(G, A);
    cout << endl;

    return 0;
}
```

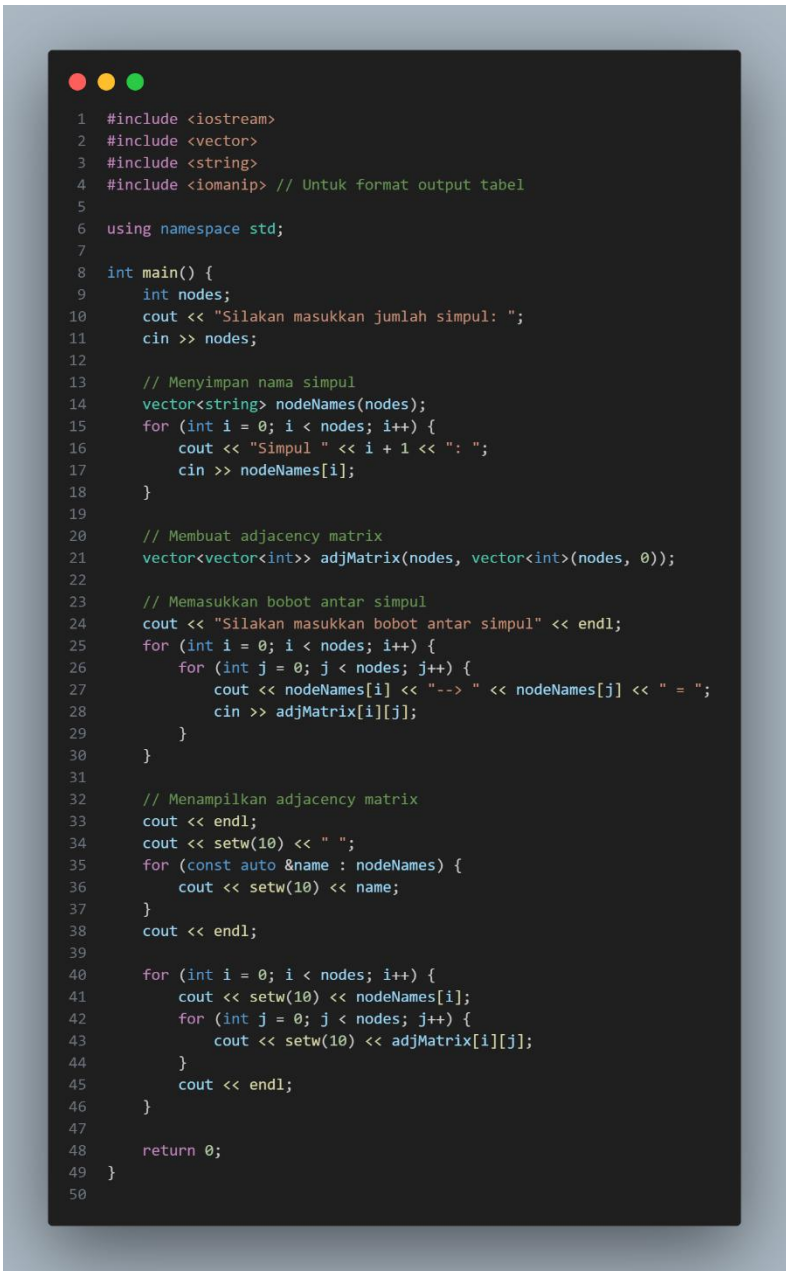
Output:

```
DFS traversal: A C G F B E D H
BFS traversal: A C B G F E D H
PS C:\Praktikum Struktur data\pertemuan11>
```

IV. UNGUIDED

1. Unguided Soal No.1

Code:



```cpp
#include <iostream>
#include <vector>
#include <string>
#include <iomanip> // Untuk format output tabel

using namespace std;

int main() {
    int nodes;
    cout << "Silakan masukkan jumlah simpul: ";
    cin >> nodes;

    // Menyimpan nama simpul
    vector<string> nodeNames(nodes);
    for (int i = 0; i < nodes; i++) {
        cout << "Simpul " << i + 1 << ": ";
        cin >> nodeNames[i];
    }

    // Membuat adjacency matrix
    vector<vector<int>> adjMatrix(nodes, vector<int>(nodes, 0));

    // Memasukkan bobot antar simpul
    cout << "Silakan masukkan bobot antar simpul" << endl;
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            cout << nodeNames[i] << "--> " << nodeNames[j] << " = ";
            cin >> adjMatrix[i][j];
        }
    }

    // Menampilkan adjacency matrix
    cout << endl;
    cout << setw(10) << " ";
    for (const auto &name : nodeNames) {
        cout << setw(10) << name;
    }
    cout << endl;

    for (int i = 0; i < nodes; i++) {
        cout << setw(10) << nodeNames[i];
        for (int j = 0; j < nodes; j++) {
            cout << setw(10) << adjMatrix[i][j];
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
Silakan masukkan jumlah simpul: 2
Simpul 1: Bali
Simpul 2: Palu
Silakan masukkan bobot antar simpul
Bali--> Bali = 0
Bali--> Palu = 3
Palu--> Bali = 4
Palu--> Palu = 0

               Bali      Palu
      Bali        0         3
      Palu        4         0
PS C:\Praktikum Struktur data\pertemuan11>
```

2. Unguided Soal No.2

Code:

```cpp
#include <iostream>
#include <vector>

using namespace std;

void adjacencyMatrix() {
    int nodes, edges;
    cout << "Masukkan jumlah simpul: ";
    cin >> nodes;
    cout << "Masukkan jumlah sisi: ";
    cin >> edges;

    // Membuat adjacency matrix
    vector<vector<int>> adjMatrix(nodes, vector<int>(nodes, 0));

    cout << "Masukkan pasangan simpul: " << endl;
    for (int i = 0; i < edges; i++) {
        int u, v;
        cin >> u >> v;
        adjMatrix[u - 1][v - 1] = 1;
        adjMatrix[v - 1][u - 1] = 1; // Karena graf tidak berarah
    }

    // Menampilkan adjacency matrix
    cout << "Adjacency Matrix:" << endl;
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            cout << adjMatrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    adjacencyMatrix();
    return 0;
}
```

Output:

```
Masukkan jumlah simpul: 4
Masukkan jumlah sisi: 4
Masukkan pasangan simpul:
1 2
1 3
2 4
3 4
Adjacency Matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
PS C:\Praktikum Struktur data\pertemuan11>
```