**Mutia Karimah 140810170002**

**Felia Sri Indriyani 140810170018**

**Ilham Muharram 140810170046**

**Ikbal Antoni 140810170054**

#include<iostream>

#include<list>

using namespace std;

// Graph class merepresentasikan graf berarah menggunakan representasi adjacency list

class Graph

{

int V; // No. simpul

// Pointer ke array yang memiliki adjacency lists

list<int> \*adj;

// Fungsi rekursif yang digunakan DFS

void DFSUtil(int v, bool visited[]);

public:

Graph(int V); // Constructor

// fungsi untuk menambah tepian ke graf

void addEdge(int v, int w);

// DFS traversal dari simpul yang terjangkau dari v

void DFS(int v);

};

Graph::Graph(int V)

{

this->V = V;

adj = new list<int>[V];

}

void Graph::addEdge(int v, int w)

{

adj[v].push\_back(w); // Menambah w ke list v.

}

void Graph::DFSUtil(int v, bool visited[])

{

// Menandakan node bersangkutan sudah dikunjungi lalu cetak

visited[v] = true;

cout << v << " ";

// Ulang simpul berdekatan ke node ini

list<int>::iterator i;

for (i = adj[v].begin(); i != adj[v].end(); ++i)

if (!visited[\*i])

DFSUtil(\*i, visited);

}

// DFS traversal dari simpul terjangkau dari v.

// Menggunakan rekursif DFSUtil()

void Graph::DFS(int v)

{

// Menandakan semua simpul belum dikunjungi

bool \*visited = new bool[V];

for (int i = 0; i < V; i++)

visited[i] = false;

// Memanggil fungsi rekursif pembantu untuk mencetak DFS traversal

DFSUtil(v, visited);

}

int main()

{

// Membuat graf di diagram

Graph g(8);

g.addEdge(1, 2);

g.addEdge(1, 3);

g.addEdge(2, 5);

g.addEdge(2, 4);

g.addEdge(5, 6);

g.addEdge(3, 7);

g.addEdge(3, 8);

g.addEdge(7, 8);

cout << "Depth First Traversal"

" (dimulai dari node 1) \n";

g.DFS(1);

return 0;

}

