Berikut adalah Source Code untuk program yang telah dibuat untuk menampilkan baik Adjacency List dan Graph. Untuk laporan cara pembuatan program dan Analisa program bisa di lihat di file word TugasPembuatanGraph\_BryanYehuda\_05311940000021

#include <iostream>

#include <list>

#include <cmath>

#include <graphics.h>

using namespace std;

class Point {

private:

int x, y;

string name;

public:

Point();

Point(int x, int y, string name);

int getX();

int getY();

string getName();

};

//default

Point::Point() {

this->x = -1;

this->y = -1;

this->name = "";

}

//constructor titik

Point::Point(int x, int y, string name) {

this->x = x;

this->y = y;

this->name = name;

}

int Point::getX() {

return x;

}

int Point::getY() {

return y;

}

string Point::getName() {

return name;

}

//class untuk menampung graph

class Graph {

private:

int V;

list<int> \*adj; //list hubungan antar titik

list<Point> \*coord; //list koordinat dan nama titik

public:

Graph(int v);

void addVertice(int v, int x, int y, string name);

void addEdge(int v, int w);

bool connected(int v, int w);

void printList();

void printEdge();

void printPoint();

};

//constructor graph

Graph::Graph(int V) {

this->V = V;

adj = new list<int>[V];

coord = new list<Point>[V];

}

//menambahkan titik dan update ukuran matriks

void Graph::addVertice(int v,int x,int y, string name) {

Point p(x, y, name);

coord[v].push\_back(p);

}

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

adj[v].push\_back(w);

adj[w].push\_back(v);

}

//mengecek apakah dua titik terhubung

bool Graph::connected(int v, int w) {

list<int>::iterator j;

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

if (\*j == w) return true;

}

for (j = adj[w].begin(); j != adj[w].end(); j++) {

if (\*j == v) return true;

}

return false;

}

//print hubungan dari setiap titik

void Graph::printList() {

list<int>::iterator i;

for (int v = 0; v < V; v++) {

cout << coord[v].begin()->getName();

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

cout << " -> " << coord[\*i].begin()->getName();

}

cout << endl;

}

}

void Graph::printEdge() {

list<Point>::iterator i;

for (int v = 0; v < V; v++) {

for (int val = v + 1; val < V; val++) {

if (!connected(v, val)) continue;

i = coord[v].begin();

int x1 = i->getX();

int y1 = i->getY();

i = coord[val].begin();

int x2 = i->getX();

int y2 = i->getY();

line(x1, y1, x2, y2);

}

}

}

void Graph::printPoint() {

list<Point>::iterator i;

for (int v = 0; v < V; v++) {

i = coord[v].begin();

int point\_x = i->getX();

int point\_y = i->getY();

string name = i->getName();

int n = name.length();

char char\_array[n + 1];

strcpy(char\_array, name.c\_str());

outtextxy(point\_x, point\_y, char\_array);

}

}

int main() {

//membuat objek graph

Graph g(9);

int gd = DETECT, gm;

// initgraph initializes the graphics system

// by loading a graphics driver from disk

initwindow(1920,1080);

// font style

int font = 8;

// font direction

int direction = 0;

// font size

int font\_size = 1;

// for setting text style

settextstyle(font, direction, font\_size);

settextjustify(1, 1);

//menambahkan titik dengan namanya

g.addVertice(0, 100, 100, "Teknik Sipil");

g.addVertice(1, 225, 200, "Arsitektur");

g.addVertice(2, 100, 300, "Manarul Ilmi");

g.addVertice(3, 350, 250, "Rektorat");

g.addVertice(4, 200, 400, "SCC ITS");

g.addVertice(5, 400, 350, "Plaza DR Angka");

g.addVertice(6, 100, 550, "Lapangan Basket");

g.addVertice(7, 300, 550, "Menara Sains");

g.addVertice(8, 200, 700, "K1 Mart");

//menambahkan hubungan titik

g.addEdge(0, 1);

g.addEdge(1, 2);

g.addEdge(1, 3);

g.addEdge(2, 0);

g.addEdge(2, 4);

g.addEdge(3, 5);

g.addEdge(4, 2);

g.addEdge(4, 5);

g.addEdge(4, 7);

g.addEdge(5, 3);

g.addEdge(5, 4);

g.addEdge(6, 4);

g.addEdge(7, 8);

g.addEdge(8, 6);

//print hubungan dan graph

g.printList();

g.printEdge();

g.printPoint();

getch();

// closegraph function closes the graphics

// mode and deallocates all memory allocated

// by graphics system .

closegraph();

return 0;

}