Dokumentasi OOP_Graph

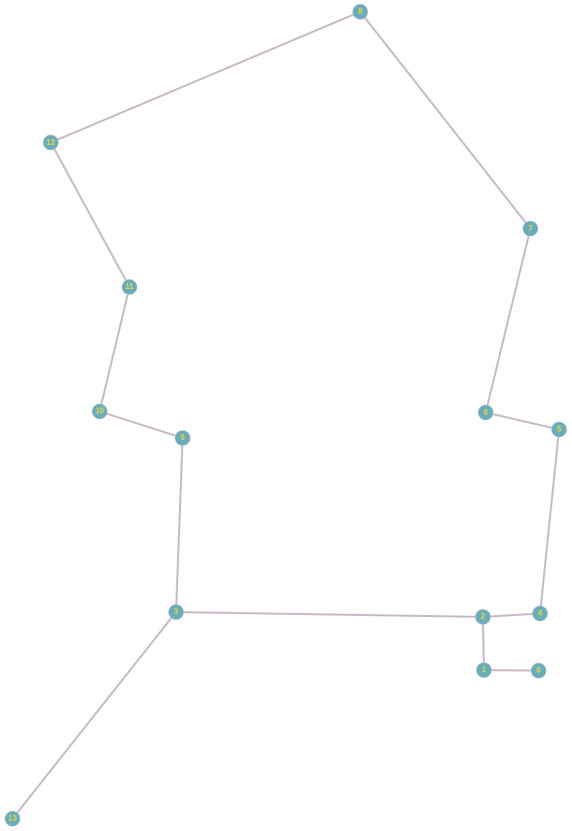
Jonathan Adithya Baswara`- 5027221062

Soal

1. Gambarkan peta sekitar rumah kalian dengan minimal 10 titik dalam bentuk graf berarah **(20 poin)**
2. Dengan menggunakan Class Peta yang telah saya contohkan, implementasikan peta yang telah kalian buat (nomor 1) ke dalam sebuah program dengan representasi graf adjacency list **(20 poin)**
3. Tampilkan hasil adjacency listnya **(5 poin)**
4. Buatkan Class baru bernama "Titik" untuk menyimpan ID titik, nama tempat (misal "Rumah", "Minimarket", "Apotek", dll), titik koordinat x, dan titik koordinat y. Instansiasi class "Titik" untuk menyimpan info titik pada peta. **(20 poin)**
5. Tampilkan hasil adjacency list berupa nama tempat **(15 poin)**
6. Tampilkan hasil graf menggunakan library graphics.h **(10 poin)**
7. Tambahkan modifikasi lain **(1-10 poin)**

Jawaban

1. Gambar Peta



2. Implementasi ke program CPP

```
#include <iostream>
#include <list>
#include <stack>
#include <stdio.h>
using namespace std;

class Peta
{

private:
  // Property
  int jumlah_titik;
  list<int> *adjacency_list;
  int **adjacency_matrix;

public:
  // Constructor
  Peta(int jumlah_titik)
  {
    this->jumlah_titik = jumlah_titik;
    this->inisialisasiAdjList(jumlah_titik);
    this->inisialisasiAdjMatrix(jumlah_titik);
  }

  // Destructor
  ~Peta()
  {
    delete[] adjacency_list;
  }

  // Fungsi untuk inisialisasi adjacency list
  void inisialisasiAdjList(int jumlah_titik)
  {
    adjacency_list = new list<int>[jumlah_titik];
  }

  // Fungsi untuk inisialisasi adjacency matrix
  void inisialisasiAdjMatrix(int jumlah_titik)
  {
    adjacency_matrix = new int *[jumlah_titik];
    for (int i = 0; i < jumlah_titik; i++)
    {
      adjacency_matrix[i] = new int[jumlah_titik];
      for (int j = 0; j < jumlah_titik; j++)
      {
        adjacency_matrix[i][j] = 0; // Inisialisasi matriks dengan nilai 0(tidak ada edge)
      }
    }
  }

  // Fungsi untuk menambahkan koneksi dari titik awal ke tujuan
  void tambahLintasan(int titik_awal, int titik_tujuan)
  {
    // Update adjacency list
    adjacency_list[titik_awal].push_back(titik_tujuan);

    // Update adjacency matrix
    adjacency_matrix[titik_awal][titik_tujuan] = 1;
    adjacency_matrix[titik_tujuan][titik_awal] = 1;
  }

  // Fungsi untuk menampilkan adjacency list
  void tampilkanAdjList()
  {
    list<int>::iterator i;

    for (int v = 0; v < jumlah_titik; v++)
    {
      cout << v << " -> ";
      for (i = adjacency_list[v].begin(); i != adjacency_list[v].end(); ++i)
      {
        cout << (*i);
        if (next(i, 1) != adjacency_list[v].end())
        {
          cout << " -> ";
        }
      }
```

```
      cout << endl;
    }
  }

  // Fungsi untuk menampilkan adjacency matrix
  void tampilkanAdjMatrix()
  {
    for (int i = 0; i < jumlah_titik; i++)
    {
      for (int j = 0; j < jumlah_titik; j++)
      {
        cout << adjacency_matrix[i][j] << " ";
      }
      cout << endl;
    }
  }
};

int main()
{
  cout << "Peta Rumah (Kos)" << endl;
  int jumlah_titik = 14;

  // Ini "Object petaKu"
  // Object adalah instansiasi dari Class
  Peta petaKu(jumlah_titik);

  // Mendefinisikan data untuk graf Peta
  petaKu.tambahLintasan(0, 1); // dari titik 0 ke 1
  petaKu.tambahLintasan(1, 0);
  petaKu.tambahLintasan(1, 2); // dari titik 0 ke 4
  petaKu.tambahLintasan(2, 1);
  petaKu.tambahLintasan(2, 3);
  petaKu.tambahLintasan(2, 4);
  petaKu.tambahLintasan(3, 2);
  petaKu.tambahLintasan(3, 9);
  petaKu.tambahLintasan(3, 13);
  petaKu.tambahLintasan(4, 2);
  petaKu.tambahLintasan(4, 5);
  petaKu.tambahLintasan(5, 4);
  petaKu.tambahLintasan(5, 6);
  petaKu.tambahLintasan(6, 5);
  petaKu.tambahLintasan(6, 7);
  petaKu.tambahLintasan(7, 6);
  petaKu.tambahLintasan(7, 8);
  petaKu.tambahLintasan(8, 7);
  petaKu.tambahLintasan(8, 12);
  petaKu.tambahLintasan(9, 3);
  petaKu.tambahLintasan(9, 10);
  petaKu.tambahLintasan(10, 9);
  petaKu.tambahLintasan(10, 11);
  petaKu.tambahLintasan(11, 10);
  petaKu.tambahLintasan(11, 12);
  petaKu.tambahLintasan(12, 11);
  petaKu.tambahLintasan(12, 8);
  petaKu.tambahLintasan(13, 3);

  cout << endl;
  cout << "Adjacency List" << endl;
  petaKu.tampilkanAdjList();

  cout << endl;
  cout << "Adjacency Matrix" << endl;
  petaKu.tampilkanAdjMatrix();
}
```

Jadi di sini kita tinggal menambahkan vertex dan edges-edgesnnya seperti yang dicontohkan dimodul

3. Adjencancy listnya

```
Peta Rumah (Kos)

Adjacency List
0 -> 1
1 -> 0 -> 2
2 -> 1 -> 3 -> 4
3 -> 2 -> 9 -> 13
4 -> 2 -> 5
5 -> 4 -> 6
6 -> 5 -> 7
7 -> 6 -> 8
8 -> 7 -> 12
9 -> 3 -> 10
10 -> 9 -> 11
11 -> 10 -> 12
12 -> 11 -> 8
13 -> 3
```

4. Class baru

```
class Titik {
public:
    string nama;
    int x;
    int y;
    Titik(string nama, int x, int y) : nama(nama), x(x), y(y) {}
};
```

menambah int x dan y untuk koordinat kartesius dari vertex (tempat) kita

```
#include <iostream>
#include <list>
#include <stack>
#include <stdio.h>
#include <string>
using namespace std;

class Titik {
public:
    string nama;
    int x;
    int y;
    Titik(string nama, int x, int y) : nama(nama), x(x), y(y) {}
};

class Peta {
private:
    int jumlah_titik;
    Titik** titik_list;
    int** adjacency_matrix;

public:
    Peta(int jumlah_titik) {
        this->jumlah_titik = jumlah_titik;
        this->inisialisasiTitikList(jumlah_titik);
        this->inisialisasiAdjMatrix(jumlah_titik);
    }

    ~Peta() {
        for (int i = 0; i < jumlah_titik; i++) {
            delete titik_list[i];
        }
        delete[] titik_list;
        for (int i = 0; i < jumlah_titik; i++) {
            delete[] adjacency_matrix[i];
        }
        delete[] adjacency_matrix;
    }
```

```
    void inisialisasiTitikList(int jumlah_titik) {
        titik_list = new Titik*[jumlah_titik];
        titik_list[0] = new Titik("Rumah kos", 1116, 1406);
        titik_list[1] = new Titik("pertigaan_blokT", 1002, 1402);
        titik_list[2] = new Titik("gapura_perumdosT", 1000, 1296);
        titik_list[3] = new Titik("bunderan_CCWS", 364, 1284);
        titik_list[4] = new Titik("belokan_perumdosT", 1120, 1286);
        titik_list[5] = new Titik("belokan_perumdosU", 1160, 906);
        titik_list[6] = new Titik("belokan_tekpal", 1006, 874);
        titik_list[7] = new Titik("bunderan_daspro", 1098,490);
        titik_list[8] = new Titik("gerbang_its_utara", 748, 44);
        titik_list[9] = new Titik("bunderan_manarul", 382, 922);
        titik_list[10] = new Titik("belokan_manarul", 206, 870);
        titik_list[11] = new Titik("bunderan_lapanganalumni", 272, 612);
        titik_list[12] = new Titik("gerbang_barat", 108, 312);
        titik_list[13] = new Titik("gerbang_selatan", 28, 1712);
    }

    void inisialisasiAdjMatrix(int jumlah_titik) {
        adjacency_matrix = new int* [jumlah_titik];
        for (int i = 0; i < jumlah_titik; i++) {
            adjacency_matrix[i] = new int[jumlah_titik];
            for (int j = 0; j < jumlah_titik; j++) {
                adjacency_matrix[i][j] = 0;
            }
        }
    }

    void tambahLintasan(int titik_awal, int titik_tujuan) {
        adjacency_matrix[titik_awal][titik_tujuan] = 1;
        adjacency_matrix[titik_tujuan][titik_awal] = 1;
    }

    void tampilkanAdjList() {
        for (int v = 0; v < jumlah_titik; v++) {
            cout << titik_list[v]->nama;
            int jumlah_lintasan = 0;
            for (int j = 0; j < jumlah_titik; j++) {
                if (adjacency_matrix[v][j] == 1) {
                    if (jumlah_lintasan == 0) {
                        cout << " -> ";
                    } else {
                        cout << " -> ";
                    }
                    cout << titik_list[j]->nama;
                    jumlah_lintasan++;
                }
            }
            cout << endl;
        }
    }

    void tampilkanAdjMatrix() {
        cout << "Adjacency Matrix:" << endl;
        for (int i = 0; i < jumlah_titik; i++) {
            for (int j = 0; j < jumlah_titik; j++) {
                cout << adjacency_matrix[i][j] << " ";
            }
            cout << endl;
        }
    }
};

int main() {
    cout << "Peta Rumah (Kos)" << endl;
    int jumlah_titik = 14;

    Peta petaKu(jumlah_titik);

    petaKu.tambahLintasan(0, 1);
    petaKu.tambahLintasan(1, 0);
    petaKu.tambahLintasan(1, 2);
    petaKu.tambahLintasan(2, 1);
    petaKu.tambahLintasan(2, 3);
    petaKu.tambahLintasan(2, 4);
    petaKu.tambahLintasan(3, 2);
    petaKu.tambahLintasan(3, 9);
    petaKu.tambahLintasan(3, 13);
```

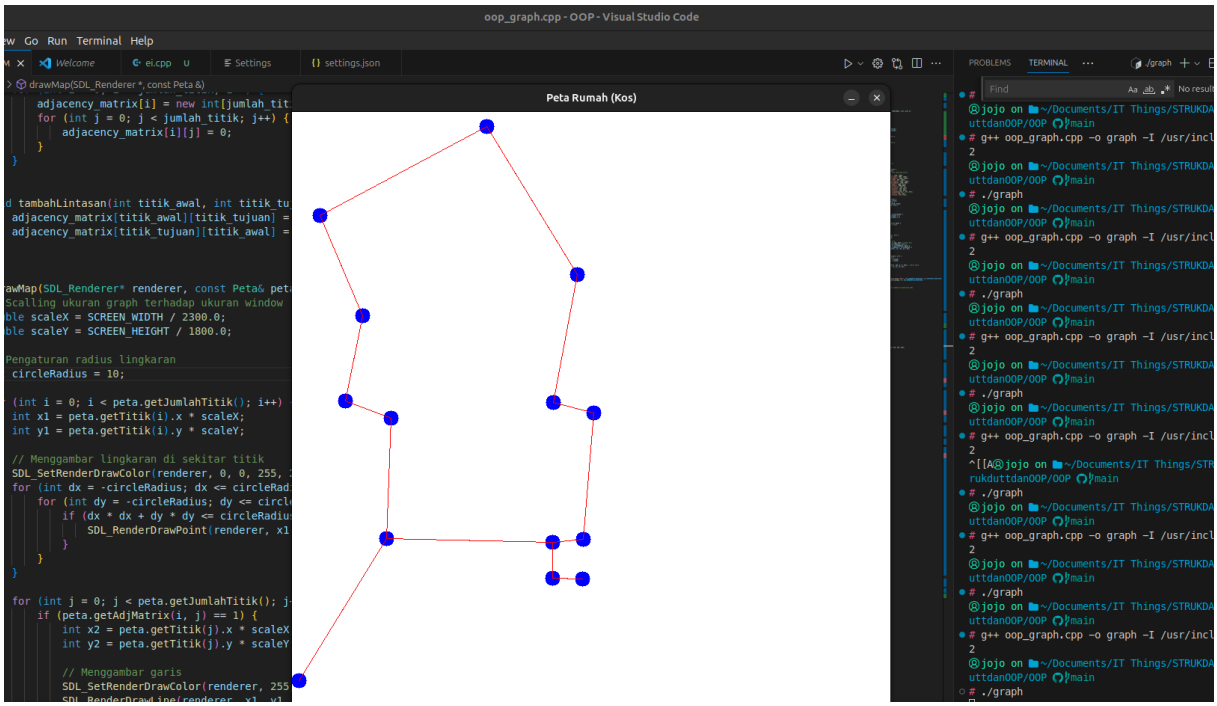
```
    petaKu.tambahLintasan(4, 2);
    petaKu.tambahLintasan(4, 5);
    petaKu.tambahLintasan(5, 4);
    petaKu.tambahLintasan(5, 6);
    petaKu.tambahLintasan(6, 5);
    petaKu.tambahLintasan(6, 7);
    petaKu.tambahLintasan(7, 6);
    petaKu.tambahLintasan(7, 8);
    petaKu.tambahLintasan(8, 7);
    petaKu.tambahLintasan(8, 12);
    petaKu.tambahLintasan(9, 3);
    petaKu.tambahLintasan(9, 10);
    petaKu.tambahLintasan(10, 9);
    petaKu.tambahLintasan(10, 11);
    petaKu.tambahLintasan(11, 10);
    petaKu.tambahLintasan(11, 12);
    petaKu.tambahLintasan(12, 11);
    petaKu.tambahLintasan(12, 8);
    petaKu.tambahLintasan(13, 3);

    cout << endl;
    cout << "Adjacency List (Nama Tempat)" << endl;
    petaKu.tampilkanAdjList();

    cout << endl;
    petaKu.tampilkanAdjMatrix();

    return 0;
}
```

disini kita menginisialisasi nama Titik dengan nama tempat dan juga koordinat kartesiusnya

5. Hasil adjencacy list

```
Adjacency List (Nama Tempat)
Rumah kos -> pertigaan_blokT
pertigaan_blokT -> Rumah kos -> gapura_perumdosT
gapura_perumdosT -> pertigaan_blokT -> bunderan_CCWS -> belokan_perumdosT
bunderan_CCWS -> gapura_perumdosT -> bunderan_manarul -> gerbang_selatan
belokan_perumdosT -> gapura_perumdosT -> belokan_perumdosU
belokan_perumdosU -> belokan_perumdosT -> belokan_tekpal
belokan_tekpal -> belokan_perumdosU -> bunderan_daspro
bunderan_daspro -> belokan_tekpal -> gerbang_its_utara
gerbang_its_utara -> bunderan_daspro -> gerbang_barat
bunderan_manarul -> bunderan_CCWS -> belokan_manarul
belokan_manarul -> bunderan_manarul -> bunderan_lapanganalumni
bunderan_lapanganalumni -> belokan_manarul -> gerbang_barat
gerbang_barat -> gerbang_its_utara -> bunderan_lapanganalumni
gerbang_selatan -> bunderan_CCWS
```

6. Saya menggunakan SDL.h karena graphics.h sudah tidak bisa dipakai di linux versi terbaru.



Dengan memanfaatkan koordinat kartesiusnya, kita bisa menggambarkan graph

```
#include <SDL.h>
#include <iostream>
#include <string>
using namespace std;
const int SCREEN_WIDTH = 800;
const int SCREEN_HEIGHT = 800;

class Titik {
public:
    string nama;
    int x;
    int y;
    Titik(string nama, int x, int y) : nama(nama), x(x), y(y) {}
};

class Peta {
private:
    int jumlah_titik;
    Titik** titik_list;
    int** adjacency_matrix;

public:
    Peta(int jumlah_titik) {
        this->jumlah_titik = jumlah_titik;
        this->inisialisasiTitikList(jumlah_titik);
        this->inisialisasiAdjMatrix(jumlah_titik);
    }

    ~Peta() {
        for (int i = 0; i < jumlah_titik; i++) {
            delete titik_list[i];
        }
        delete[] titik_list;
        for (int i = 0; i < jumlah_titik; i++) {
            delete[] adjacency_matrix[i];
        }
        delete[] adjacency_matrix;
    }

    int getJumlahTitik() const {
        return jumlah_titik;
    }
```

```
    Titik getTitik(int index) const {
        return *titik_list[index];
    }

    int getAdjMatrix(int row, int col) const {
        return adjacency_matrix[row][col];
    }

    void inisialisasiTitikList(int jumlah_titik) {
        titik_list = new Titik*[jumlah_titik];
        // Inisialisasi titik sesuai dengan data yang Anda miliki
        // Contoh inisialisasi titik:
        titik_list[0] = new Titik("Rumah kos", 1116, 1406);
        titik_list[1] = new Titik("pertigaan_blokT", 1002, 1402);
        titik_list[2] = new Titik("gapura_perumdosT", 1002, 1294);
        titik_list[3] = new Titik("bunderan_CCWS", 364, 1284);
        titik_list[4] = new Titik("belokan_perumdosT", 1120, 1286);
        titik_list[5] = new Titik("belokan_perumdosU", 1160, 906);
        titik_list[6] = new Titik("belokan_tekpal", 1006, 874);
        titik_list[7] = new Titik("bunderan_daspro", 1098,490);
        titik_list[8] = new Titik("gerbang_its_utara", 748, 44);
        titik_list[9] = new Titik("bunderan_manarul", 382, 922);
        titik_list[10] = new Titik("belokan_manarul", 206, 870);
        titik_list[11] = new Titik("bunderan_lapanganalumni", 272, 612);
        titik_list[12] = new Titik("gerbang_barat", 108, 312);
        titik_list[13] = new Titik("gerbang_selatan", 28, 1712);
    }

    void inisialisasiAdjMatrix(int jumlah_titik) {
        adjacency_matrix = new int* [jumlah_titik];
        for (int i = 0; i < jumlah_titik; i++) {
            adjacency_matrix[i] = new int[jumlah_titik];
            for (int j = 0; j < jumlah_titik; j++) {
                adjacency_matrix[i][j] = 0;
            }
        }
    }

    void tambahLintasan(int titik_awal, int titik_tujuan) {
        adjacency_matrix[titik_awal][titik_tujuan] = 1;
        adjacency_matrix[titik_tujuan][titik_awal] = 1;
    }
};

void drawMap(SDL_Renderer* renderer, const Peta& peta) {
    // Scalling ukuran graph terhadap ukuran window
    double scaleX = SCREEN_WIDTH / 2300.0;
    double scaleY = SCREEN_HEIGHT / 1800.0;

    // Pengaturan radius lingkaran
    int circleRadius = 10;

    for (int i = 0; i < peta.getJumlahTitik(); i++) {
        int x1 = peta.getTitik(i).x * scaleX;
        int y1 = peta.getTitik(i).y * scaleY;

        // Menggambar lingkaran di sekitar titik
        SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255); // Warna biru
        for (int dx = -circleRadius; dx <= circleRadius; dx++) {
            for (int dy = -circleRadius; dy <= circleRadius; dy++) {
                if (dx * dx + dy * dy <= circleRadius * circleRadius) {
                    SDL_RenderDrawPoint(renderer, x1 + dx, y1 + dy);
                }
            }
        }

        for (int j = 0; j < peta.getJumlahTitik(); j++) {
            if (peta.getAdjMatrix(i, j) == 1) {
                int x2 = peta.getTitik(j).x * scaleX;
                int y2 = peta.getTitik(j).y * scaleY;

                // Menggambar garis
                SDL_SetRenderDrawColor(renderer, 255, 0, 0, 255); // Warna merah
                SDL_RenderDrawLine(renderer, x1, y1, x2, y2);
            }
        }
    }
}
```

```
int main() {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window* window = SDL_CreateWindow("Peta Rumah (Kos)", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, SCREEN_WIDTH, SCREEN_HEI
    SDL_Renderer* renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);

    int jumlah_titik = 14;
    Peta petaKu(jumlah_titik);

    // Menambahkan lintasan seperti yang Anda lakukan di dalam main Anda
    petaKu.tambahLintasan(0, 1);
    petaKu.tambahLintasan(1, 0);
    petaKu.tambahLintasan(1, 2);
    petaKu.tambahLintasan(2, 1);
    petaKu.tambahLintasan(2, 3);
    petaKu.tambahLintasan(2, 4);
    petaKu.tambahLintasan(3, 2);
    petaKu.tambahLintasan(3, 9);
    petaKu.tambahLintasan(3, 13);
    petaKu.tambahLintasan(4, 2);
    petaKu.tambahLintasan(4, 5);
    petaKu.tambahLintasan(5, 4);
    petaKu.tambahLintasan(5, 6);
    petaKu.tambahLintasan(6, 5);
    petaKu.tambahLintasan(6, 7);
    petaKu.tambahLintasan(7, 6);
    petaKu.tambahLintasan(7, 8);
    petaKu.tambahLintasan(8, 7);
    petaKu.tambahLintasan(8, 12);
    petaKu.tambahLintasan(9, 3);
    petaKu.tambahLintasan(9, 10);
    petaKu.tambahLintasan(10, 9);
    petaKu.tambahLintasan(10, 11);
    petaKu.tambahLintasan(11, 10);
    petaKu.tambahLintasan(11, 12);
    petaKu.tambahLintasan(12, 11);
    petaKu.tambahLintasan(12, 8);
    petaKu.tambahLintasan(13, 3);

    bool quit = false;
    SDL_Event e;

    while (!quit) {
        while (SDL_PollEvent(&e) != 0) {
            if (e.type == SDL_QUIT) {
                quit = true;
            }
        }

        SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255);
        SDL_RenderClear(renderer);

        drawMap(renderer, petaKu);

        SDL_RenderPresent(renderer);
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}
```

disini ditambahkan fungsi fungsi untuk mengambar graph untuk libbrari SDL

7. Tambahan
    a. ukuran gambar graph scalling dengan ukuran window SDL
    b. mengatur ukuran Titik
    c. mengatur warna garis dan Titik