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

using namespace std;

class Titik
{
public:
    int id;
    string nama;
    int x;
    int y;

    Titik(int id, const string &nama, int x, int y)
        : id(id), nama(nama), x(x), y(y) {}
};
class Peta
{
public:
    vector<Titik> titikList;
    vector<vector<int>> adjacencyList;

    void tambahTitik(const Titik &titik)
    {
        titikList.push_back(titik);
        adjacencyList.push_back({});
    }

    void tambahEdge(int from, int to)
    {
        adjacencyList[from].push_back(to);
    }

    void tampilkanAdjacencyList()
    {
        for (size_t i = 0; i < adjacencyList.size(); ++i)
        {
            cout << "Tempat " << titikList[i].nama << " terhubung ke: ";
            for (size_t j = 0; j < adjacencyList[i].size(); ++j)
```

```
            {
                cout << titikList[adjacencyList[i][j]].nama << ", ";
            }
            cout << endl;
        }
    }

    void hapusTitik(int id)
    {
        if (id >= 0 && id < static_cast<int>(titikList.size()))
        {
            // Hapus titik dari adjacency list
            adjacencyList.erase(adjacencyList.begin() + id);
            for (size_t i = 0; i < adjacencyList.size(); ++i)
            {
                for (size_t j = 0; j < adjacencyList[i].size(); ++j)
                {
                    if (adjacencyList[i][j] == id)
                    {
                        adjacencyList[i].erase(adjacencyList[i].begin() +
j);
                    }
                }
            }

            // Hapus titik dari daftar titik
            titikList.erase(titikList.begin() + id);
        }
    }

    void editTitik(int id, const Titik &titik)
    {
        if (id >= 0 && id < static_cast<int>(titikList.size()))
        {
            // Edit informasi titik
            titikList[id] = titik;
        }
    }
};
```

```
int main()
{
    Peta peta;

    peta.tambahTitik(Titik(0, "Rumah", 100, 100));
    peta.tambahTitik(Titik(1, "Masjid", 200, 150));
    peta.tambahTitik(Titik(2, "Sekolah SD", 300, 200));
    peta.tambahTitik(Titik(3, "Pertigaan", 150, 250));
    peta.tambahTitik(Titik(4, "Sekolah SMA", 400, 100));
    peta.tambahTitik(Titik(5, "Restoran", 250, 300));
    peta.tambahTitik(Titik(6, "Masjid 3", 350, 250));
    peta.tambahTitik(Titik(7, "Terminal", 200, 50));
    peta.tambahTitik(Titik(8, "Pertigaan 2", 50, 200));
    peta.tambahTitik(Titik(9, "Rumah Sakit", 300, 100));
    peta.tambahTitik(Titik(10, "Restoran 2", 180, 200));
    peta.tambahTitik(Titik(11, "Kafe 2", 220, 280));
    peta.tambahTitik(Titik(12, "ITS", 120, 180));
    peta.tambahTitik(Titik(13, "Perumahan", 180, 90));
    peta.tambahTitik(Titik(14, "Apartemen 2", 330, 150));
    peta.tambahTitik(Titik(15, "Pasar", 400, 250));
    peta.tambahTitik(Titik(16, "Universitas", 280, 170));
    peta.tambahTitik(Titik(17, "Masjid 2", 320, 210));
    peta.tambahTitik(Titik(18, "Kafe", 120, 250));
    peta.tambahTitik(Titik(19, "Apartemen", 390, 200));

    peta.tambahEdge(0, 1);
    peta.tambahEdge(0, 3);
    peta.tambahEdge(1, 2);
    peta.tambahEdge(1, 0);
    peta.tambahEdge(2, 3);
    peta.tambahEdge(2, 1);
    peta.tambahEdge(2, 19);
    peta.tambahEdge(3, 2);
    peta.tambahEdge(3, 0);
    peta.tambahEdge(3, 4);
    peta.tambahEdge(4, 5);
    peta.tambahEdge(4, 8);
    peta.tambahEdge(4, 3);
    peta.tambahEdge(5, 4);
    peta.tambahEdge(5, 6);
```

```
    peta.tambahEdge(6, 5);
    peta.tambahEdge(6, 7);
    peta.tambahEdge(7, 6);
    peta.tambahEdge(8, 4);
    peta.tambahEdge(8, 17);
    peta.tambahEdge(8, 9);
    peta.tambahEdge(9, 8);
    peta.tambahEdge(9, 10);
    peta.tambahEdge(10, 11);
    peta.tambahEdge(10, 9);
    peta.tambahEdge(11, 10);
    peta.tambahEdge(11, 12);
    peta.tambahEdge(12, 11);
    peta.tambahEdge(12, 15);
    peta.tambahEdge(12, 13);
    peta.tambahEdge(13, 12);
    peta.tambahEdge(13, 16);
    peta.tambahEdge(13, 14);
    peta.tambahEdge(14, 13);
    peta.tambahEdge(14, 15);
    peta.tambahEdge(15, 14);
    peta.tambahEdge(15, 12);
    peta.tambahEdge(16, 13);
    peta.tambahEdge(17, 3);
    peta.tambahEdge(17, 18);
    peta.tambahEdge(18, 17);
    peta.tambahEdge(18, 19);
    peta.tambahEdge(19, 18);
    peta.tambahEdge(19, 2);

    peta.tampilkanAdjacencyList();

    while (true)
    {
        cout << "Menu:\n";
        cout << "1. Tambah Titik\n";
        cout << "2. Hapus Titik\n";
        cout << "3. Edit Titik\n";
        cout << "4. Tambah Edge\n";
        cout << "5. Tampilkan Adjacency List\n";
```

```
        cout << "6. Keluar\n";
        cout << "Pilih menu (1/2/3/4/5/6): ";

        int pilihan;
        cin >> pilihan;

        if (pilihan == 1)
        {
            // Tambah titik
            int id, x, y;
            string nama;
            cout << "ID Titik: ";
            cin >> id;
            cout << "Nama Tempat: ";
            cin.ignore();
            getline(cin, nama);
            cout << "Koordinat X: ";
            cin >> x;
            cout << "Koordinat Y: ";
            cin >> y;
            peta.tambahTitik(Titik(id, nama, x, y));
        }
        else if (pilihan == 2)
        {
            // Hapus titik
            int id;
            cout << "ID Titik yang akan dihapus: ";
            cin >> id;
            peta.hapusTitik(id);
        }
        else if (pilihan == 3)
        {
            // Edit titik
            int id, x, y;
            string nama;
            cout << "ID Titik yang akan diedit: ";
            cin >> id;
            cout << "Nama Tempat baru: ";
            cin.ignore();
            getline(cin, nama);
```

```
            cout << "Koordinat X baru: ";
            cin >> x;
            cout << "Koordinat Y baru: ";
            cin >> y;
            peta.editTitik(id, Titik(id, nama, x, y));
        }
        else if (pilihan == 4)
        {
            // Tambah edge
            int from, to;
            cout << "ID Titik Asal: ";
            cin >> from;
            cout << "ID Titik Tujuan: ";
            cin >> to;
            peta.tambahEdge(from, to);
        }
        else if (pilihan == 5)
        {
            // Tampilkan adjacency list
            cout << "Adjacency List:" << endl;
            peta.tampilkanAdjacencyList();
        }
        else if (pilihan == 6)
        {
            // Keluar dari program
            break;
        }
        else
        {
            cout << "Pilihan tidak valid. Silakan pilih kembali.\n";
        }
    }

    return 0;
}
```

Program di atas merupakan program graph yang dibuat dengan teknik OOP. Tujuan program ini adalah untuk memetakan daerah di sekitar user dan menerjemahkannya ke bahasa yang dimengerti oleh komputer. Berikut penjelasannya:

Class Titik

```
class Titik
{
public:
    int id;
    string nama;
    int x;
    int y;

    Titik(int id, const string &nama, int x, int y)
        : id(id), nama(nama), x(x), y(y) {}
};
```

Class ini bertujuan sebagai lokasi dalam peta. Dengan `id` sebagai id lokasi, `nama` sebagai nama lokasi, `x` sebagai koordinat x lokasi, dan `y` sebagai koordinat y peta.

Class Graph

```
class Peta
{
public:
    vector<Titik> titikList;
    vector<vector<int>> adjacencyList;

    void tambahTitik(const Titik &titik)
    {
        titikList.push_back(titik);
        adjacencyList.push_back({});
    }

    void tambahEdge(int from, int to)
```

```
    {
        adjacencyList[from].push_back(to);
    }

    void tampilkanAdjacencyList()
    {
        for (size_t i = 0; i < adjacencyList.size(); ++i)
        {
            cout << "Tempat " << titikList[i].nama << " terhubung ke: ";
            for (size_t j = 0; j < adjacencyList[i].size(); ++j)
            {
                cout << titikList[adjacencyList[i][j]].nama << ", ";
            }
            cout << endl;
        }
    }

    void hapusTitik(int id)
    {
        if (id >= 0 && id < static_cast<int>(titikList.size()))
        {
            // Hapus titik dari adjacency list
            adjacencyList.erase(adjacencyList.begin() + id);
            for (size_t i = 0; i < adjacencyList.size(); ++i)
            {
                for (size_t j = 0; j < adjacencyList[i].size(); ++j)
                {
                    if (adjacencyList[i][j] == id)
                    {
                        adjacencyList[i].erase(adjacencyList[i].begin() +
j);
                    }
                }
            }

            // Hapus titik dari daftar titik
            titikList.erase(titikList.begin() + id);
        }
    }
```

```
    void editTitik(int id, const Titik &titik)
    {
        if (id >= 0 && id < static_cast<int>(titikList.size()))
        {
            // Edit informasi titik
            titikList[id] = titik;
        }
    }
};
```

Class ini merupakan peta(graph) untuk program ini, berikut penjelasan lebih lanjut untuk fungsi-fungsi di dalamnya:

Fungsi addVertex()

```
    void tambahTitik(const Titik &titik)
    {
        titikList.push_back(titik);
        adjacencyList.push_back({});
    }
```

Fungsi ini bertujuan untuk menambah lokasi ke dalam peta dengan menggunakan fungsi `.push_back` yang tersedia dalam library `vector` c++.Dan juga fungsi ini men-*set* nilai `adjacencyList` sebagai list kosong karena secara default nilainya adalah null.

Fungsi addEdge()

```
    void tambahEdge(int from, int to)
    {
        adjacencyList[from].push_back(to);
    }
```

Fungsi ini bertujuan untuk menambah koneksi antar lokasi dalam peta dengan menambahkan id tujuan ke dalam `adjacencyList` di asal.

Fungsi tampilkanAdjacencyList()

```
    void tampilkanAdjacencyList()
    {
        for (size_t i = 0; i < adjacencyList.size(); ++i)
        {
            cout << "Tempat " << titikList[i].nama << " terhubung ke: ";
```

```
            for (size_t j = 0; j < adjacencyList[i].size(); ++j)
            {
                cout << titikList[adjacencyList[i][j]].nama << ", ";
            }
            cout << endl;
        }
    }
```

Fungsi ini digunakan untuk menampilkan struktur graph pada sistem. Dengan melakukan iterasi terhadap seluruh lokasi yang ada, kemudian jika node tersebut memiliki relasi dengan lokasi lain, akan ditampilkan juga relasi tersebut.

Fungsi hapusTitik()

```
    void hapusTitik(int id)
    {
        if (id >= 0 && id < static_cast<int>(titikList.size()))
        {
            // Hapus titik dari adjacency list
            adjacencyList.erase(adjacencyList.begin() + id);
            for (size_t i = 0; i < adjacencyList.size(); ++i)
            {
                for (size_t j = 0; j < adjacencyList[i].size(); ++j)
                {
                    if (adjacencyList[i][j] == id)
                    {
                        adjacencyList[i].erase(adjacencyList[i].begin() +
j);
                    }
                }
            }

            // Hapus titik dari daftar titik
            titikList.erase(titikList.begin() + id);
        }
    }
```

Fungsi ini digunakan untuk menghapus lokasi dan relasi yang lokasi tersebut miliki dengan lokasi lain.

Fungsi editTitik()

```
void editTitik(int id, const Titik &titik)
{
    if (id >= 0 && id < static_cast<int>(titikList.size()))
    {
        // Edit informasi titik
        titikList[id] = titik;
    }
}
```

Fungsi ini digunakan untuk memodifikasi titik yang sudah ada.