Tugas 2: Implementasi Struktur Pada Array

Akhmad Thoriq Afif NRP 5024201028

10 Oktober 2021

1 Listing Program

Berikut ini merupakan source code dari tugas 2 Implementasi Struktur pada Array. Program ini dibuat dengan bahasa pemrograman C++.

Listing 1: Implementasi Struktur pada Array

```
#include <iostream>
#include <string>
#include <utility>

using namespace std;

struct Coordinate
{
    Coordinate() {}
    Coordinate(int x, int y) : x(y), y(y) {}

    int x;
    int y;
};

struct Node
{
    Node() {}
    Node(string city_name, int x, int y) : mCityName(std::move(city_name)),

    bool operator==(const Node &A) const
    {
        if (A.mCityName == mCityName)
        {
            return true;
        }
        return false;
    }
```

```
    void connect(int index)
    {
        mCurrentConnected++;
        mConnected[mCurrentConnected] = index;
    }

    void remove(int index)
    {
        bool alreadyDeleted = false;

        for (int i = 0; i < mCurrentConnected + 1; i++)
        {
            if (mConnected[i] == index && !alreadyDeleted)
            {
                for (int j = i; j < mCurrentConnected + 1; j++)
                {
                    mConnected[j] = mConnected[j + 1];
                }
                mCurrentConnected--;
                alreadyDeleted = true;
            }

            if (mConnected[i] >= index)
            {
                mConnected[i]--;
            }
        }
    }

    void insert(int index)
    {
        for (int i = 0; i < mCurrentConnected + 1; i++)
        {
            if (mConnected[i] >= index)
            {
                mConnected[i]++;
            }
        }
    }

    string mCityName;
    int mConnected[100]{-1};
    int mCurrentConnected = -1;
    Coordinate mCoordinate;
};
```

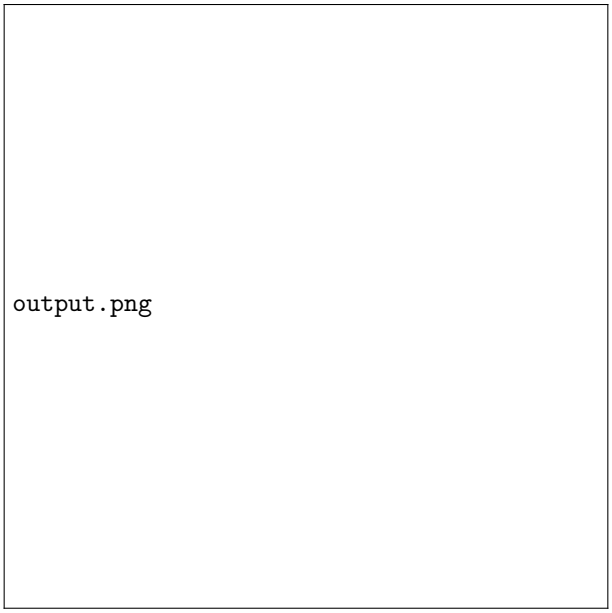
```
class Maps
{
public:
    Maps(int size)
    {
        mSize = size;
        mArray = new Node[size];
    }

    ~Maps()
    {
        delete[] mArray;
    }

    void print()
    {
        cout << "| Nama Kota |"
             << " Kota Tersambung |" << endl;
        for (int i = 0; i < mCurrSize + 1; i++)
        {
            cout << mArray[i].mCityName << " (" << mArray[i].mCoordinate.x <
                 << "\t";
            for (int j = 0; j < mArray[i].mCurrentConnected + 1; j++)
            {
                cout << mArray[mArray[i].mConnected[j]].mCityName << (j == m

            }

            cout << endl;
        }
    }

    void insert(const string &city_name, int x, int y, int index)
    {
        if (mCurrSize + 1 < mSize && index <= mCurrSize + 1)
        {
            const Node newCity(city_name, x, y);
            int back = mCurrSize;
            while (back >= index)
            {
                mArray[back + 1] = mArray[back];
                back--;
            }

            for (int i = 0; i < mCurrSize + 1; i++)
            {
                mArray[i].insert(index);
```

```
        }
        mArray[index] = newCity;

        mCurrSize++;
    }
}

Maps &addCity(const string &city_name, int x, int y)
{
    const Node newCity(city_name, x, y);

    if (mCurrSize + 1 < mSize)
    {
        mCurrSize++;
        mArray[mCurrSize] = newCity;
    }
    else
    {
        cout << "Maps is full\n";
    }

    return *this;
}

void remove(const string &city_name)
{
    const int index = findCityByName(city_name);

    for (int i = index; i < mCurrSize; i++)
    {
        mArray[i] = mArray[i + 1];
    }

    for (int i = 0; i < mCurrSize + 1; i++)
    {
        mArray[i].remove(index);
    }
    mCurrSize--;
}

Maps &connect(const string &A, const string &B)
{
    const int AIndex = findCityByName(A);
    const int BIndex = findCityByName(B);

    mArray[AIndex].connect(BIndex);
```

```
        return *this;
    }

    int findCityByName(const string &city_name) const
    {
        for (int index = 0; index <= mCurrSize; index++)
        {
            if (mArray[index].mCityName == city_name)
            {
                return index;
            }
        }
        return -1;
    }

private:
    Node *mArray;
    int mSize;
    int mCurrSize = -1;
};

int main()
{
    Maps Maps(5);

    Maps.addCity("A", 0, 0)
        .addCity("B", 1, 1)
        .addCity("D", 1, -1)
        .addCity("F", 2, 1)
        .addCity("E", 2, -1);

    Maps.connect("A", "B")
        .connect("A", "D")
        .connect("B", "F")
        .connect("D", "E")
        .connect("B", "E")
        .connect("D", "F");

    Maps.print();

    Maps.remove("B");

    cout << "\nDaftar␣kota␣seletah␣B␣dihapus" << endl;
    Maps.print();
```

```
    Maps.insert("C", 2, -2, 3);

    cout << "\nDaftar kota C diinsert di index 3" << endl;
    Maps.print();

    return 0;
}
```

OUTPUT PROGRAM:



Gambar 1: Output Program

2 Penjelasan Program

2.1 Desain struktur data

Pada program ini terdapat struct Node yang merupakan representasi dari Kota. Dalam Node ini terdapat atribut nama kota, kota yang tersambung, dan koordinat kota. Selain Node, terdapat class Maps. Class Maps menyimpan Node-Node yang ada. Node tersebut disimpan kedalam sebuah array yang dapat ditentukan ukurannya.

2.2 Fungsi untuk menambah kota

Listing 2: Fungsi untuk menambah kota

```
            {
                cout << mArray[mArray[i].mConnected[j]].mCityName << (j == m
            }

            cout << endl;
        }
    }

    void insert(const string &city_name, int x, int y, int index)
    {
        if (mCurrSize + 1 < mSize && index <= mCurrSize + 1)
        {
            const Node newCity(city_name, x, y);
            int back = mCurrSize;
```

Fungsi ini digunakan untuk menambahkan kota. Dibutuhkan 3 parameter yaitu nama kota dan koordinat kota. Langkah-langkah dalam menambah kota adalah sebagai berikut:

1. Membuat object Node baru bernama newCity.
2. Melakukan pengecekan ukuran Maps. Jika ukuran Maps masih cukup maka akan dilakukan penambahan kota.

2.3 Fungsi untuk menghapus kota

Listing 3: Fungsi untuk menghapus kota

```
            {
                mArray[back + 1] = mArray[back];
                back--;
            }

            for (int i = 0; i < mCurrSize + 1; i++)
```

```
            {
                mArray[i].insert(index);
            }
            mArray[index] = newCity;

            mCurrSize++;
        }
    }
```

Fungsi ini digunakan untuk menghapus kota. Dibutuhkan 1 parameter yaitu nama kota. Langkah-langkah dalam menghapus kota adalah sebagai berikut:

1. Melakukan pencarian index kota yang akan dihapus dengan menggunakan fungsi findCityByname.
2. Memindahkan elemen sebelah kanan kota yang akan dihapus ke index kota yang akan dihapus.
3. Mengurangi ukuran Maps.

2.4 Fungsi untuk mencari kota

Listing 4: Fungsi untuk mencari kota

```
    }
    else
    {
        cout << "Maps is full\n";
    }

    return *this;
}

void remove(const string &city_name)
```

Fungsi ini digunakan untuk mencari kota. Dibutuhkan 1 parameter yaitu nama kota. Langkah-langkah dalam mencari kota adalah sebagai berikut:

1. Melakukan transversal pada array maps. Jika nama kota yang dicari sama dengan nama kota pada array maka akan dikembalikan index kota.
2. Jika tidak ada kota yang sama maka akan dikembalikan nilai -1.