

Tugas 2: Implementasi Struktur Pada Array

Azzam Wildan Maulana NRP 5024201010

October 10, 2021

1 Source Code

Berikut adalah source code dasar yang saya buat

```
#include <bits/stdc++.h>

using namespace std;

float pitagoras(float x1, float y1, float x2, float y2)
{
    return sqrt(((x2 - x1) * (x2 - x1)) + ((y2 - y1) * (y2 - y1)));
}

class DbKota
{
private:
    //properties
    int last_node;
    int banyak_kota;

public:
    //properties
    struct kota_t
    {
        string nama;
        int x;
        int y;
        string hubungan_kota[100];
        int hubungan_kota_pos[100];
        int tanda_akhir_hubungan_kota;
        // int next_node;
    };
    kota_t kota[100];
```

```

//method

DbKota()
{
    last_node = -1;
    banyak_kota = 0;
}
~DbKota()
{
    //desturtor
}

void append(string nama_kota, int x, int y)
{
    last_node++;
    banyak_kota++;
    kota[last_node].nama = nama_kota;
    kota[last_node].x = x;
    kota[last_node].y = y;
    // memset(kota[last_node].hubungan_kota, -1, sizeof(kota[last_node].
    kota[last_node].tanda_akhir_hubungan_kota = -1;
    // kota[last_node].next_node = -1;
}

void remove(int posisi)
{
    posisi++;
    for (int i = posisi - 1; i < banyak_kota; i++)
    {
        kota[i] = kota[i + 1];
    }

    kota[banyak_kota - 1].nama = false;
    kota[banyak_kota - 1].x = false;
    kota[banyak_kota - 1].y = false;
    // memset(kota[last_node].hubungan_kota, -1, sizeof(kota[last_node].
    kota[last_node].tanda_akhir_hubungan_kota = -1;

    banyak_kota--;
}

void removeByName(string nama_kota)
{
    // int posisi;
    // for (int i = 0; i < banyak_kota; i++)
    // {

```

```

        //      if (nama_kota == kota[i].nama)
        //      {
        //          posisi = i;
        //          break;
        //      }
        //  }
        remove(searchByName(nama_kota));
    }

void insert(int posisi, string nama_kota, int x, int y)
{
    banyak_kota++;
    for (int i = banyak_kota; i >= posisi; i--)
    {
        kota[i + 1] = kota[i];
    }

    kota[posisi].nama = nama_kota;
    kota[posisi].x = x;
    kota[posisi].y = y;
    // memset(kota[last_node].hubungan_kota, -1, sizeof(kota[last_node].
    kota[last_node].tanda_akhir_hubungan_kota = -1;
    // kota[posisi].next_node = -1;
}

void show(int more = 0)
{
    int panjang_nama = 0;
    int panjang_terdekat = 0;
    if (!more)
    {
        for (int i = 0; i < banyak_kota; i++)
        {
            // printf("iter-%d\n", i);
            if (kota[i].nama.size() < 0x8)
                cout << kota[i].nama << "\t\t";
            else if (kota[i].nama.size() >= 0x8)
                cout << kota[i].nama << "\t";
            cout << kota[i].x << "\t";
            cout << kota[i].y << "\t";
            if (kota[i].tanda_akhir_hubungan_kota == -1)
                printf("-□");
            else
            {
                for (int v = 0; v < kota[i].tanda_akhir_hubungan_kota; v)
                {

```

```

        cout << kota[i].hubungan_kota[v] << ",_";
    }
    cout << kota[i].hubungan_kota[kota[i].tanda_akhir_hubung
}
    cout << endl;
}
}
else
{
    //header
    for (int i = 0; i < banyak_kota; i++)
    {
        if (kota[i].nama.size() < 0x8)
            panjang_nama = 0;
        else if (kota[i].nama.size() >= 0x8)
            panjang_nama = 1;

        if (cariKotaTerdekatStr(kota[i].nama).size() < 0x8)
            panjang_terdekat = 0;
        else if (cariKotaTerdekatStr(kota[i].nama).size() >= 0x8)
            panjang_terdekat = 1;
    }

    if (panjang_nama == 0)
        printf("Nama_\\t");
    else if (panjang_nama == 1)
        printf("Nama_\\t\\t");

    printf("X\\t");

    printf("Y\\t");

    if (panjang_terdekat == 0)
        printf("Terdekat");
    else if (panjang_terdekat == 1)
        printf("Terdekat\\t");

    printf("Hubungan_");

    printf("\\n");

    //isi
    for (int i = 0; i < banyak_kota; i++)
    {
        // printf("iter-%d\\n", i);
        if (kota[i].nama.size() < 0x8)

```

```

        cout << kota[i].nama << "\t\t";
    else if (kota[i].nama.size() >= 0x8)
        cout << kota[i].nama << "\t";

    cout << kota[i].x << "\t";

    cout << kota[i].y << "\t";

    if (cariKotaTerdekatStr(kota[i].nama).size() < 0x8)
        cout << cariKotaTerdekatStr(kota[i].nama) << "\t\t";
    else if (cariKotaTerdekatStr(kota[i].nama).size() >= 0x8)
        cout << cariKotaTerdekatStr(kota[i].nama) << "\t";
    // cout << cariKotaTerdekatStr(kota[i].nama) << " ";

    if (kota[i].tanda_akhir_hubungan_kota == -1)
        printf("-\u");
    else
    {
        for (int v = 0; v < kota[i].tanda_akhir_hubungan_kota; v++)
        {
            cout << kota[i].hubungan_kota[v] << ",\u";
        }
        cout << kota[i].hubungan_kota[kota[i].tanda_akhir_hubungan_kota] << "\u";
        cout << endl;
    }
}

}

int searchByName(string nama_kota)
{
    for (int i = 0; i < banyak_kota; i++)
    {
        if (nama_kota == kota[i].nama)
        {
            // cout << "Kota " << nama_kota << " ada pada index array ke " << i << endl;
            // break;
            return i;
        }
    }
    return -1;
}

void searchByPos(int posisi)
{
    cout << "Kota\u pada\u posisi\u ke-" << posisi << "\u adalah\u " << kota[posisi].nama << endl;
}

```

```

}
void connect1arah(string src, string dst)
{
    int src_pos = searchByName(src);
    int dst_pos = searchByName(dst);
    if (src_pos >= 0 && dst_pos >= 0)
    {
        kota[src_pos].tanda_akhir_hubungan_kota++;
        kota[src_pos].hubungan_kota[kota[src_pos].tanda_akhir_hubungan_kota] = dst;
        kota[src_pos].hubungan_kota_pos[kota[src_pos].tanda_akhir_hubungan_kota] = dst_pos;
    }
    else
        printf("Invalid name\n");
}
void connect2arah(string src, string dst)
{
    int src_pos = searchByName(src);
    int dst_pos = searchByName(dst);
    if (src_pos >= 0 && dst_pos >= 0)
    {
        kota[dst_pos].tanda_akhir_hubungan_kota++;
        kota[dst_pos].hubungan_kota[kota[dst_pos].tanda_akhir_hubungan_kota] = src;
        kota[dst_pos].hubungan_kota_pos[kota[dst_pos].tanda_akhir_hubungan_kota] = src_pos;

        kota[src_pos].tanda_akhir_hubungan_kota++;
        kota[src_pos].hubungan_kota[kota[src_pos].tanda_akhir_hubungan_kota] = dst;
        kota[src_pos].hubungan_kota_pos[kota[src_pos].tanda_akhir_hubungan_kota] = dst_pos;
    }
    else
        printf("Invalid name\n");
}
//PERCOBAAN, kendala belum menemukan solusi tepat untuk pindah cabang per
void cariHubungan(int dst, int temp[100], int *total_hub, int iter)
{
    //Linear search
    int temp_iter = iter;
    for (int i = 0; i < banyak_kota; i++)
    {
        if (kota[i].tanda_akhir_hubungan_kota > -1)
        {
            for (int j = 0; j <= kota[i].tanda_akhir_hubungan_kota; j++)
            {
                // printf("hub: %d dengan %d, ", kota[i].hubungan_kota_pos[j], kota[kota[i].hubungan_kota_pos[j]].nama_kota);
                if (kota[i].hubungan_kota_pos[j] == dst)
                {

```

```

        temp[temp_iter] = i;
        temp_iter++;
    }
}
// printf("sudah\n");
}
// printf("ier: %d\n", temp_iter);
if (temp_iter == 0)
    printf("Tdk punya hubungan..\n");
else
    *total_hub = temp_iter;
}
void cariRute(string src, string dst)
{
    int temp[100], total_hub = 0, hasil[100];
    int tanda_hasil = 0;
    int src_pos = searchByName(src);
    int dst_pos = searchByName(dst);
    if (src_pos >= 0 && dst_pos >= 0)
    {
        int i = 0;
        // while (temp[i] != src_pos)
        // {
        temp[0] = dst_pos;
        int j = 0;
        while (temp[0] != src_pos)
        {
            // printf("j: %d\n", j);
            if (j >= banyak_kota - 1)
            {
                j = 0;
                i++;
            }
            cariHubungan(temp[i], temp, &total_hub, i);
            j++;
            if (total_hub > 0)
            {
                hasil[tanda_hasil] = temp[i];
                tanda_hasil++;
            }
        }
        // i++;
        // }
    }
    // printf("temp0: %d\n", temp[0]);
}

```

```

        if (temp[0] == searchByName(src))
        { // printf("hasil: %d", temp[0]);
          // printf("Rute: ");
          cout << "Rute_" << src << "_ke_" << dst << ":_:" << endl;
          for (int i = tanda_hasil - 1; i >= 0; i--)
          {
              // printf("awukehhkcuakuaew %d ", hasil[i]);
              // printf("%");
              cout << kota[hasil[i]].nama << "_->_";
          }
          cout << dst << endl;
        }
    }
    int cariKotaTerdekat(string kota_yang_ingin_dicari)
    {
        int pos = searchByName(kota_yang_ingin_dicari);
        float min = 9999999999;
        int min_pos = pos;
        for (int i = 0; i < banyak_kota; i++)
        {
            if (i != pos)
            {
                // printf("min: %f ", min);
                // printf("jarak %s: %f\n", kota[i].nama.c_str(), pitagoras(
                if (min >= pitagoras(kota[pos].x, kota[pos].y, kota[i].x, kota[i].y))
                {
                    min = pitagoras(kota[pos].x, kota[pos].y, kota[i].x, kota[i].y);
                    min_pos = i;
                }
            }
        }
        return min_pos;
    }
    string cariKotaTerdekatStr(string kota_yang_ingin_dicari)
    {
        return kota[cariKotaTerdekat(kota_yang_ingin_dicari)].nama;
    }
};

int main()
{
    DbKota DB;

    DB.append("Jember", 123, 321);
    DB.append("Haven", 234, 823);
    DB.append("Konoha", 0x46, 0x21);
}

```



```

DB.append("Atlantis", -200, -121);
DB.append("Bikini_Bottom", 242, 678);
DB.append("Dimmsdale", 124, 323);

DB.connect1arah("Jember", "Konoha");
DB.connect1arah("Atlantis", "Dimmsdale");

// DB.connect2arah("Jember", "Haven");
// DB.connect2arah("Konoha", "Atlantis");
// DB.connect2arah("Haven", "Bikini Bottom");
// DB.connect2arah("Bikini Bottom", "Dimmsdale");

// DB.insert(2, "sby", 12, 23);
// DB.show();
// DB.searchByName("sby");
// DB.searchByPos(2);
// DB.remove(2);
// DB.removeByName("sby");

// DB.show();
// DB.cariRute("Jember", "Dimmsdale");

// printf("%f", pitagoras(2, 0, 5, 4));
// printf("%d\n", DB.cariKotaTerdekat("Jember"));
// printf("%s\n", DB.cariKotaTerdekatStr("Jember").c_str());
// DB.cariKotaTerdekat("Jember");

DB.show(1);
// int coba = 8;
// if (coba == 0x8)
//     printf("%x\n", coba);
// printf("%x\n", 69);
return 0;
}

```

Disana saya membuat Class seperti Database dengan nama DBKota yang isinya adalah method-method dasar seperti tambah, hapus, tampilkan, dan cari.

1.1 Desain Struktur Data

Struktur data saya simpan pada Class DBKota

```

#include <bits/stdc++.h>
using namespace std;
class DbKota
{

```

```

private:
    //properties
    int last_node;
    int banyak_kota;

public:
    //properties
    struct kota_t
    {
        string nama;
        int x;
        int y;
        int next_node;
    };
    kota_t kota[100];

    //method

    DbKota();
    ~DbKota();

    void append(string nama_kota, int x, int y);
    void remove(int posisi);
    void removeByName(string nama_kota);
    void insert(int posisi, string nama_kota, int x, int y);
    void show();
    int searchByName(string nama_kota);
    void searchByPos(int posisi);
};

```

Dia memiliki method method dasar seperti tambah, hapus, tampilkan, dan cari.

1.2 Fungsi untuk menambahkan data kota ke dalam suatu array

```

void DBKota::append(string nama_kota, int x, int y)
{
    last_node++;
    banyak_kota++;
    kota[last_node].nama = nama_kota;
    kota[last_node].x = x;
    kota[last_node].y = y;
    kota[last_node].next_node = -1;
}

```

Saya membuat fungsi bernama append, Dia menerima variabel yang digunakan untuk mengisi struct kota_t.

1.3 Fungsi untuk menyisipkan dan menghapus kota tersebut kedalam array

```
void DBkota::insert(int posisi, string nama_kota, int x, int y)
{
    banyak_kota++;
    for (int i = banyak_kota; i >= posisi; i--)
    {
        kota[i + 1] = kota[i];
    }

    kota[posisi].nama = nama_kota;
    kota[posisi].x = x;
    kota[posisi].y = y;
    kota[posisi].next_node = -1;
}

void DBkota::remove(int posisi)
{
    posisi++;
    for (int i = posisi - 1; i < banyak_kota; i++)
    {
        kota[i] = kota[i + 1];
    }

    kota[banyak_kota - 1].nama = false;
    kota[banyak_kota - 1].x = false;
    kota[banyak_kota - 1].y = false;

    banyak_kota--;
}
```

Untuk menyisipkan data, saya pakai fungsi insert sedangkan untuk menghapus data saya pakai remove

1.4 Fungsi untuk mencari nama kota, output dari fungsi tersebut adalah index array kota tersebut

```
int DBkota::searchByName(string nama_kota)
{
    for (int i = 0; i < banyak_kota; i++)
    {
        if (nama_kota == kota[i].nama)
```

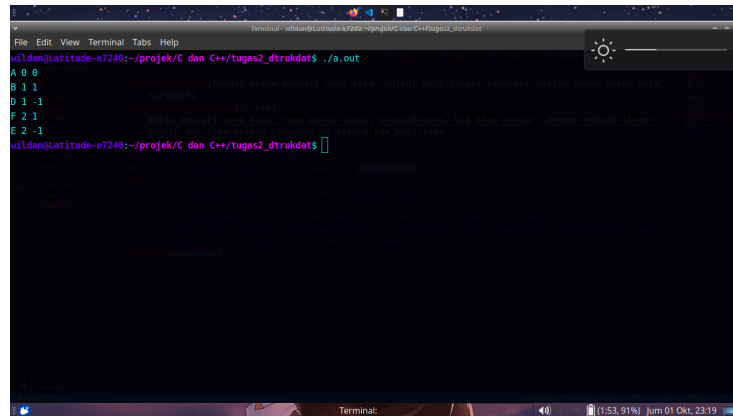
```

        {
            // cout << "Kota " << nama_kota << " ada pada index array ke-" << i;
            // break;
            return i;
        }
    }
}

```

Untuk mencari nama kota, saya pakai fungsi `searchByName`. Dia akan mencari dengan metode linear search dan jika ketemu langsung di-return kan posisinya.

1.5 Output Program



```

wilidan@Latitude-e7240:~/projek/C dan C++/tugas2_dtrukdats$ ./a.out
A 0 0
B 1 1
D 1 -1
F 2 1
E 2 -1
wilidan@Latitude-e7240:~/projek/C dan C++/tugas2_dtrukdats$

```