Tugas 2: Implementasi Struktur Pada Array

Dwi Cahya Ramadani 5024201025

October 1, 2021

1 Listing Program

Berikut adalah Listing 1 yang merupakan jawaban dari ketiga pertanyaan pada tugas 2 kali ini. Pertanyaan nomor 1 mengenai struktur data yang digunakan dalam program terjawab pada struktur data CKota. Pertanyaan nomor 2a, yaitu fungsi untuk menambah data kota kedalam suatu array terjawab pada fungsi tambah. Pertanyaan Nomor 2b, yaitu untuk menyisipkan dan menghapus suatu data kota pada array terjawab pada fungsi sisipkan dan hapus. Dan pertanyaan nomor 2c, yaitu untuk mencari data kota dengan input nama kota dan output indexnya terjawab pada fungsi cari.

Listing 1: Jawaban

```
#include <iostream>

using namespace std;

// M sebagai penunjuk akhir array, posisi untuk menyimpan index
    yang dicari
int M = -1;

struct CKota{
    string nama;
    float x,y;
    int tujuanKiri = -1, tujuanKanan = -1;
};CKota Kota[100];

void tambah(string nama, float x, float y){
    int N = 99;
    if (M < N){
        M = M + 1;
        Kota[M].nama = nama;
        Kota[M].x = x;
        Kota[M].y = y;
    }
}

void cari(string nama,int *posisi){
    *posisi = -1;
    for(int i=0; i<=M; i++){
        if(Kota[i].nama == nama){
            *posisi = i;
```

```
            break;
        }
    }
}

void hubungkan(string asal,string tujuanKiri,string tujuanKanan){
    int indexAsal,indexTujuanKiri,indexTujuanKanan;
    cari(asal,&indexAsal);
    cari(tujuanKiri,&indexTujuanKiri);
    cari(tujuanKanan,&indexTujuanKanan);
    Kota[indexAsal].tujuanKiri = indexTujuanKiri;
    Kota[indexAsal].tujuanKanan = indexTujuanKanan;
}

void sisipkan(string nama,float x,float y,int posisi){
    if (M < 99){
        M = M + 1;
        for(int i = M; i >= posisi+1; i--){
            Kota[i].nama=Kota[i-1].nama;
            Kota[i].x=Kota[i-1].x;
            Kota[i].y=Kota[i-1].y;
            if (Kota[i-1].tujuanKanan >= posisi){
                Kota[i].tujuanKanan=Kota[i-1].tujuanKanan + 1;
            }
            else if (Kota[i-1].tujuanKanan < posisi){
                Kota[i].tujuanKanan=Kota[i-1].tujuanKanan;
            }
            if (Kota[i-1].tujuanKiri >= posisi){
                Kota[i].tujuanKiri=Kota[i-1].tujuanKiri + 1;
            }
            else if (Kota[i-1].tujuanKanan < posisi){
                Kota[i].tujuanKiri=Kota[i-1].tujuanKiri;
            }
        }
        for (int i=0;i<posisi;i++){
            if (Kota[i].tujuanKanan >= posisi){
                Kota[i].tujuanKanan++;
            }
            if (Kota[i].tujuanKiri >= posisi){
                Kota[i].tujuanKiri++;
            }
        }
        Kota[posisi].nama = nama;
        Kota[posisi].x = x;
        Kota[posisi].y = y;
        Kota[posisi].tujuanKanan = -1;
        Kota[posisi].tujuanKiri = -1;
    }
}

void hapus(string nama){
    int posisi;
    cari(nama,&posisi);
    if (posisi!=-1){
        M = M -1;
        for (int i = posisi;i<=M;i++){
            Kota[i].nama=Kota[i+1].nama;
```

```
            Kota[i].x=Kota[i+1].x;
            Kota[i].y=Kota[i+1].y;
            if (Kota[i+1].tujuanKanan >= posisi){
                Kota[i].tujuanKanan=Kota[i+1].tujuanKanan - 1;
            }
            else if (Kota[i+1].tujuanKanan < posisi){
                Kota[i].tujuanKanan=Kota[i+1].tujuanKanan;
            }
            if (Kota[i+1].tujuanKiri >= posisi){
                Kota[i].tujuanKiri=Kota[i+1].tujuanKiri - 1;
            }
            else if (Kota[i+1].tujuanKanan < posisi){
                Kota[i].tujuanKiri=Kota[i+1].tujuanKiri;
            }
        }
        for (int i=0;i<posisi;i++){
            if (Kota[i].tujuanKanan >= posisi){
                Kota[i].tujuanKanan--;
            }
            if (Kota[i].tujuanKiri >= posisi){
                Kota[i].tujuanKiri--;
            }
        }
    }
    else{
        cout << "Kota yang akan dihapus tidak ditemukan !" << endl;
    }
}

void tampilkanSemua(){
    for(int i=0;i<=M;i++){
        if (Kota[i].tujuanKanan != -1 && Kota[i].tujuanKiri != -1 )
    {
            cout << "Kota " << Kota[i].nama << " berhubungan dengan
     Kota " << Kota[Kota[i].tujuanKiri].nama << " dan Kota " <<
    Kota[Kota[i].tujuanKanan].nama << endl;
        }
        else if ((Kota[i].tujuanKanan != -1 && Kota[i].tujuanKiri
    == -1)){
            cout << "Kota " << Kota[i].nama << " berhubungan dengan
     Kota " << Kota[Kota[i].tujuanKanan].nama << endl;
        }
        else if ((Kota[i].tujuanKanan == -1 && Kota[i].tujuanKiri
    != -1)){
            cout << "Kota " << Kota[i].nama << " berhubungan dengan
     Kota " << Kota[Kota[i].tujuanKiri].nama << endl;
        }
        else if (Kota[i].tujuanKanan == -1 && Kota[i].tujuanKiri ==
     -1 ){
            cout << "Kota " << Kota[i].nama << " tidak berhubungan
    dengan kota manapun" << endl;
        }
    }
}

int main()
{
```

```
134     int posisi;
135 
136     // MENAMBAHKAN DATA KOTA
137     tambah("A",0,0);
138     tambah("B",1,1);
139     tambah("D",1,-1);
140     tambah("F",2,1);
141     tambah("E",3,-1);
142 
143     // MENGHUBUNGKAN KOTA SATU DENGAN KOTA LAINNYA
144     hubungkan("A","B","D");
145     hubungkan("B","F","E");
146     hubungkan("D","F","E");
147     tampilkanSemua();
148 
149     // MENCARI DATA KOTA MELALUI NAMA KOTA
150     cari("E",&posisi);
151     cout << "Kota E berada pada index " << posisi <<endl;
152 
153     // MENYISIPKAN DATA KOTA PADA POSISi TERTENTU
154     sisipkan("Z",-2,7,2);
155     tampilkanSemua();
156 
157     // MENGHAPUS DATA KOTA PADA POSISI TERTENTU
158     hapus("Z");
159     tampilkanSemua();
160     return 0;
161 }
```

2 Penjelasan

- Desain Struktur Data
  Desain struktur data untuk tugas ini yaitu data Kota yang memiliki properti nama berupa string,posisi x dan y berupa float, dan tujuan kanan serta tujuan kiri yang bertipe integer. Berikut adalah struktur data yang digunakan dalam program ini yang merupakan jawaban dari soal nomor 1.

  Listing 2: Struktur Data

```
1 struct CKota{
2     string nama;
3     float x,y;
4     int tujuanKiri = -1, tujuanKanan = -1;
5 };CKota Kota[100];
```

  Penjelasan :

  - **Baris 1** : mendefinisikan struktur data CKota
  - **Baris 2** : properti nama bertipe string
  - **Baris 3** : properti x dan y bertipe float
  - **Baris 4** : properti tujuanKiri dan tujuanKanan bertipe integer dengan nilai default -1
  - **Baris 5** : Mendefinisikan variabel Kota sebagai array bertipe data CKota dengan panjang maksimal 100

- Fungsi tambah
  Fungsi ini merupakan fungsi untuk menambahkan data suatu kota ke dalam array. Fungsi ini merupakan jawaban dari soal nomor 2a. Dibawah ini adalah listing fungsi tambah.

  Listing 3: Fungsi tambah

```
1 void tambah(string nama, float x, float y){
2     int N = 99;
3     if (M < N){
4         M = M + 1;
5         Kota[M].nama = nama;
6         Kota[M].x = x;
7         Kota[M].y = y;
8     }
9 }
```

  Penjelasan :

  - **Baris 1** : mendefinisikan fungsi tambah yang bertipe void dengan parameter string nama, float x, dan float y
  - **Baris 2** : mendefinisikan variabel N sebagai batas maksimal dari array Kota

– **Baris 3-8** : percabangan, dimana ketika M (panjang array saat ini kurang dari N-1, maka item masih dapat ditambahkan. Ketika syarat terpenuhi maka M (panjang array saat ini) akan ditambah 1, lalu properti nama dari Kota[M] akan diisi parameter nama yang dikirimkan, begitu juga properti x dan y

- Fungsi sisipkan
  Fungsi ini merupakan fungsi untuk menambahkan data suatu kota ke dalam array pada posisi index tertentu. Fungsi ini merupakan jawaban dari soal nomor 2b. Dibawah ini adalah listing fungsi sisipkan.

Listing 4: Fungsi sisipkan

```
1  void sisipkan(string nama,float x,float y,int posisi){
2      if (M < 99){
3          M = M + 1;
4          for(int i = M; i >= posisi+1; i--){
5              Kota[i].nama=Kota[i-1].nama;
6              Kota[i].x=Kota[i-1].x;
7              Kota[i].y=Kota[i-1].y;
8              if (Kota[i-1].tujuanKanan >= posisi){
9                  Kota[i].tujuanKanan=Kota[i-1].tujuanKanan + 1;
10             }
11             else if (Kota[i-1].tujuanKanan < posisi){
12                 Kota[i].tujuanKanan=Kota[i-1].tujuanKanan;
13             }
14             if (Kota[i-1].tujuanKiri >= posisi){
15                 Kota[i].tujuanKiri=Kota[i-1].tujuanKiri + 1;
16             }
17             else if (Kota[i-1].tujuanKanan < posisi){
18                 Kota[i].tujuanKiri=Kota[i-1].tujuanKiri;
19             }
20         }
21         for (int i=0;i<posisi;i++){
22             if (Kota[i].tujuanKanan >= posisi){
23                 Kota[i].tujuanKanan++;
24             }
25             if (Kota[i].tujuanKiri >= posisi){
26                 Kota[i].tujuanKiri++;
27             }
28         }
29         Kota[posisi].nama = nama;
30         Kota[posisi].x = x;
31         Kota[posisi].y = y;
32         Kota[posisi].tujuanKanan = -1;
33         Kota[posisi].tujuanKiri = -1;
34     }
35 }
```

Penjelasan :

– **Baris 1** : mendefinisikan fungsi sisipkan sebagai fungsi bertipe void yang menerima parameter berupa string nama, float x, float y, dan int posisi

- **Baris 2** : mendefinisikan syarat untuk menyisipkan data baru, yaitu M (panjang array saat ini) harus kurang dari panjang array maksimum
- **Baris 3** : ketika syarat terpenuhi, maka M akan ditambah 1
- **Baris 4** : Perulangan mulai dari i=M hingga i¿=posisi+1, setiap perulangan i dikurangi satu
- **Baris 5-7** : properti nama,x,y dari kota[i] akan diisi properti nama,x,y dari elemen pada index sebelumnya (Menggeser elemen ke kanan)
- **Baris 8-20** : mengatur properti tujuanKanan dan tujuanKiri agar tetap sesuai. Jika tujuanKanan maupun tujuanKiri dari elemen yang akan digeser berada pada index yang akan disisipkan atau setelahnya, maka tujuanKanan dan tujuanKiri nya harus ditambah 1, namun jika tidak maka nilainya tetap
- **Baris 21-28** : Perulangan untuk mengatur tujuanKanan dan tujuanKiri dari elemen pada index sebelum index yang akan disisipkan (mekanismenya sama seperti sebelumnya)
- **Baris 29-33** : mengisikan properti nama,x,dan y dari elemen pada index yang akan disisipkan dengan parameter yang telah diterima. Untuk tujuanKanan dan tujuanKiri diasumsikan -1 karena belum dihubungkan melalui fungsi hubung.

- Fungsi hapus
  Fungsi ini merupakan fungsi untuk menghapus data suatu kota yang ada di dalam array. Fungsi ini merupakan jawaban dari soal nomor 2b. Dibawah ini adalah listing fungsi hapus.

Listing 5: Fungsi hapus

```
1  void hapus(string nama){
2      int posisi;
3      cari(nama,&posisi);
4      if (posisi!=-1){
5          M = M -1;
6          for (int i = posisi;i<=M;i++){
7              Kota[i].nama=Kota[i+1].nama;
8              Kota[i].x=Kota[i+1].x;
9              Kota[i].y=Kota[i+1].y;
10             if (Kota[i+1].tujuanKanan >= posisi){
11                 Kota[i].tujuanKanan=Kota[i+1].tujuanKanan - 1;
12             }
13             else if (Kota[i+1].tujuanKanan < posisi){
14                 Kota[i].tujuanKanan=Kota[i+1].tujuanKanan;
15             }
16             if (Kota[i+1].tujuanKiri >= posisi){
17                 Kota[i].tujuanKiri=Kota[i+1].tujuanKiri - 1;
18             }
19             else if (Kota[i+1].tujuanKanan < posisi){
20                 Kota[i].tujuanKiri=Kota[i+1].tujuanKiri;
21             }
22         }
23         for (int i=0;i<posisi;i++){
24             if (Kota[i].tujuanKanan >= posisi){
25                 Kota[i].tujuanKanan--;
26             }
27             if (Kota[i].tujuanKiri >= posisi){
28                 Kota[i].tujuanKiri--;
29             }
30         }
31     }
32     else{
33         cout << "Kota yang akan dihapus tidak ditemukan !" <<
       endl;
34     }
35 }
```

Penjelasan :

- **Baris 1** : mendefinisikan fungsi hapus yang bertipe void dengan parameter string nama
- **Baris 2** : mendefinisikan variabel posisi yang akan digunakan untuk menyimpan index dari elemen yang akan dihapus yang didapatkan melalui fungsi cari
- **Baris 3** : melakukan pencarian menggunakan fungsi cari dengan parameter nama kota yang akan dihapus dan posisi untuk menyimpan index yang didapatkan

- **Baris 4** : syarat agar elemen dapat dihapus yaitu indexnya tidak sama dengan -1 (elemen ada pada array)
- **Baris 5** : jika elemen yang akan dihapus ditemukan, maka panajng array saat ini akan dikurangi 1
- **Baris 6** : melakukan perulangan mulai i=posisi, hingga i¡=M, setiap perulangan i ditambah 1
- **Baris 7-9** : pada tiap perulangan, properti nama,x,dan y dari Kota[i] akan diisi dengan properti nama,x,dan y dari elemen setelahnya (Menggeser elemen ke kiri)
- **Baris 10-21** : mengatur properti tujuanKanan dan tujuanKiri agar tetap sesuai. Jika tujuanKanan maupun tujuanKiri dari elemen yang akan digeser berada pada index yang akan dihapus atau setelahnya, maka tujuanKanan dan tujuanKiri nya harus dikurangi 1, namun jika tidak maka nilainya tetap
- **Baris 23-30** : Perulangan untuk mengatur tujuanKanan dan tujuanKiri dari elemen pada index sebelum index yang akan dihapus (mekanismenya sama seperti sebelumnya)
- **Baris 32-34** : Jika elemen tidak ditemukan, maka akan dioutputkan kalimat "Kota yang akan dihapus tidak ditemukan"

- Fungsi cari
  Fungsi ini merupakan fungsi untuk mencari data suatu kota di dalam array. Fungsi ini merupakan jawaban dari soal nomor 2c. Dibawah ini adalah listing fungsi cari.

Listing 6: Fungsi cari

```
1 void cari(string nama,int *posisi){
2     *posisi = -1;
3     for(int i=0; i<=M; i++){
4         if(Kota[i].nama == nama){
5             *posisi = i;
6             break;
7         }
8     }
9 }
```

Penjelasan :

- **Baris 1** : mendefinisikan fungsi cari yang bertipe void dengan parameter string nama, dan int *posisi.
- **Baris 2** : posisi diisikan -1 sebagai nilai default ketika data yang dicari tidak ditemukan
- **Baris 3** : perulangan mulai i=0 hingga i¡=M, tiap perulangan i ditambah 1
- **Baris 4** : syarat data tersebut ditemukan, yaitu ketika properti nama dari Kota[i] bernilai sama dengan parameter nama yang diterima

– **Baris 5-6** : jika syarat dipenuhi, artinya data ditemukan, maka posisi akan diisikan nilai i sebagai index dari data tersebut dan perulangan akan dihentikan

- Fungsi hubungkan
  Fungsi ini merupakan fungsi tambahan yang saya buat untuk menghubungkan suatu kota dengan kota lainnya di dalam array. Dibawah ini adalah listing fungsi hubungkan.

Listing 7: Fungsi hubungkan

```
1 void hubungkan(string asal,string tujuanKiri,string
      tujuanKanan){
2     int indexAsal,indexTujuanKiri,indexTujuanKanan;
3     cari(asal,&indexAsal);
4     cari(tujuanKiri,&indexTujuanKiri);
5     cari(tujuanKanan,&indexTujuanKanan);
6     Kota[indexAsal].tujuanKiri = indexTujuanKiri;
7     Kota[indexAsal].tujuanKanan = indexTujuanKanan;
8 }
```

Penjelasan :

– **Baris 1** : mendefinisikan fungsi hubungkan yang bertipe void dengan parameter string asal, string tujuanKiri, dan string tujuanKanan.

– **Baris 2** : mendefinisikan variabel indexAsal, indexTujuanKanan, dan indexTujuanKiri sebagai tipe data integer yang akan digunakan untuk menyimpan index yang akan didapat dari fungsi cari

– **Baris 3-5** : Menggunakan fungsi car untuk mencari index kota asal, tujuanKiri dan tujuanKanan

– **Baris 6-7** : mengisikan properti tujuanKiri dan tujuanKanan dari kota asal dengan index yang telah didapatkan pada proses sebelumnya

- Fungsi tampilkanSemua
  Fungsi ini merupakan fungsi tambahan yang saya buat untuk menampilkan data semua kota beserta hubungannya dengan kota lainnya di dalam array. Dibawah ini adalah listing fungsi tampilkanSemua.

Listing 8: Fungsi tampilkanSemua

```
1 void tampilkanSemua(){
2     for(int i=0;i<=M;i++){
3         if (Kota[i].tujuanKanan != -1 && Kota[i].tujuanKiri !=
       -1 ){
4             cout << "Kota " << Kota[i].nama << " berhubungan
      dengan Kota " << Kota[Kota[i].tujuanKiri].nama << " dan
      Kota " << Kota[Kota[i].tujuanKanan].nama << endl;
5         }
6         else if ((Kota[i].tujuanKanan != -1 && Kota[i].
      tujuanKiri == -1)){
7             cout << "Kota " << Kota[i].nama << " berhubungan
      dengan Kota " << Kota[Kota[i].tujuanKanan].nama << endl;
```

```
8         }
9         else if ((Kota[i].tujuanKanan == -1 && Kota[i].
      tujuanKiri != -1)){
10            cout << "Kota " << Kota[i].nama << " berhubungan
      dengan Kota " << Kota[Kota[i].tujuanKiri].nama << endl;
11        }
12        else if (Kota[i].tujuanKanan == -1 && Kota[i].
      tujuanKiri == -1 ){
13            cout << "Kota " << Kota[i].nama << " tidak
      berhubungan dengan kota manapun" << endl;
14        }
15    }
16 }
```

Penjelasan :

- **Baris 1** : mendefinisikan fungsi tampilkanSemua dengan tipe void.
- **Baris 2** : perulangan mulai index 0 hingga index ke M
- **Baris 3-16** : menampilkan data kota beserta hubungannya dengan kota lain sesuai dengan properti tujuanKanan dan tujuanKiri nya

- Fungsi main
  Fungsi ini merupakan fungsi utama dalam program ini. Dibawah ini adalah listing fungsi main.

Listing 9: Fungsi main

```
1  int main()
2  {
3      int posisi;
4
5      // MENAMBAHKAN DATA KOTA
6      tambah("A",0,0);
7      tambah("B",1,1);
8      tambah("D",1,-1);
9      tambah("F",2,1);
10     tambah("E",3,-1);
11
12     // MENGHUBUNGKAN KOTA SATU DENGAN KOTA LAINNYA
13     hubungkan("A","B","D");
14     hubungkan("B","F","E");
15     hubungkan("D","F","E");
16     tampilkanSemua();
17
18     // MENCARI DATA KOTA MELALUI NAMA KOTA
19     cari("E",&posisi);
20     cout << "Kota E berada pada index " << posisi <<endl;
21
22     // MENYISIPKAN DATA KOTA PADA POSISi TERTENTU
23     sisipkan("Z",-2,7,2);
24     tampilkanSemua();
25
26     // MENGHAPUS DATA KOTA PADA POSISI TERTENTU
27     hapus("Z");
28     tampilkanSemua();
```

```
29      return 0;
30  }
```

Penjelasan :

- **Baris 1** : mendefinisikan fungsi main dengan tipe int.
- **Baris 2** : mendefinisikan variabel posisi sebagai tipe data integer, yang nantinya akan digunakan untuk menampung index hasil dari fungsi cari
- **Baris 6-10** : menambahkan data kota baru melalui fungsi tambah
- **Baris 13-16** : menghubungkan suatu kota dengan kota lainnya menggunakan fungsi hubungkan dan mencetaknya dengan fungsi tampilkanSemua
- **Baris 19-20** : mencari index kota dengan nama tertentu menggunakan fungsi cari dan mencetak hasilnya
- **Baris 23-24** : menyisipkan data kota baru pada index tertentu menggunakan fungsi sisipkan dan mencetaknya menggunakan fungsi tampilkanSemua
- **Baris 27-28** : menghapus data kota dengan nama tertentu dan mencetaknya menggunakan fungsi tampilkanSemua