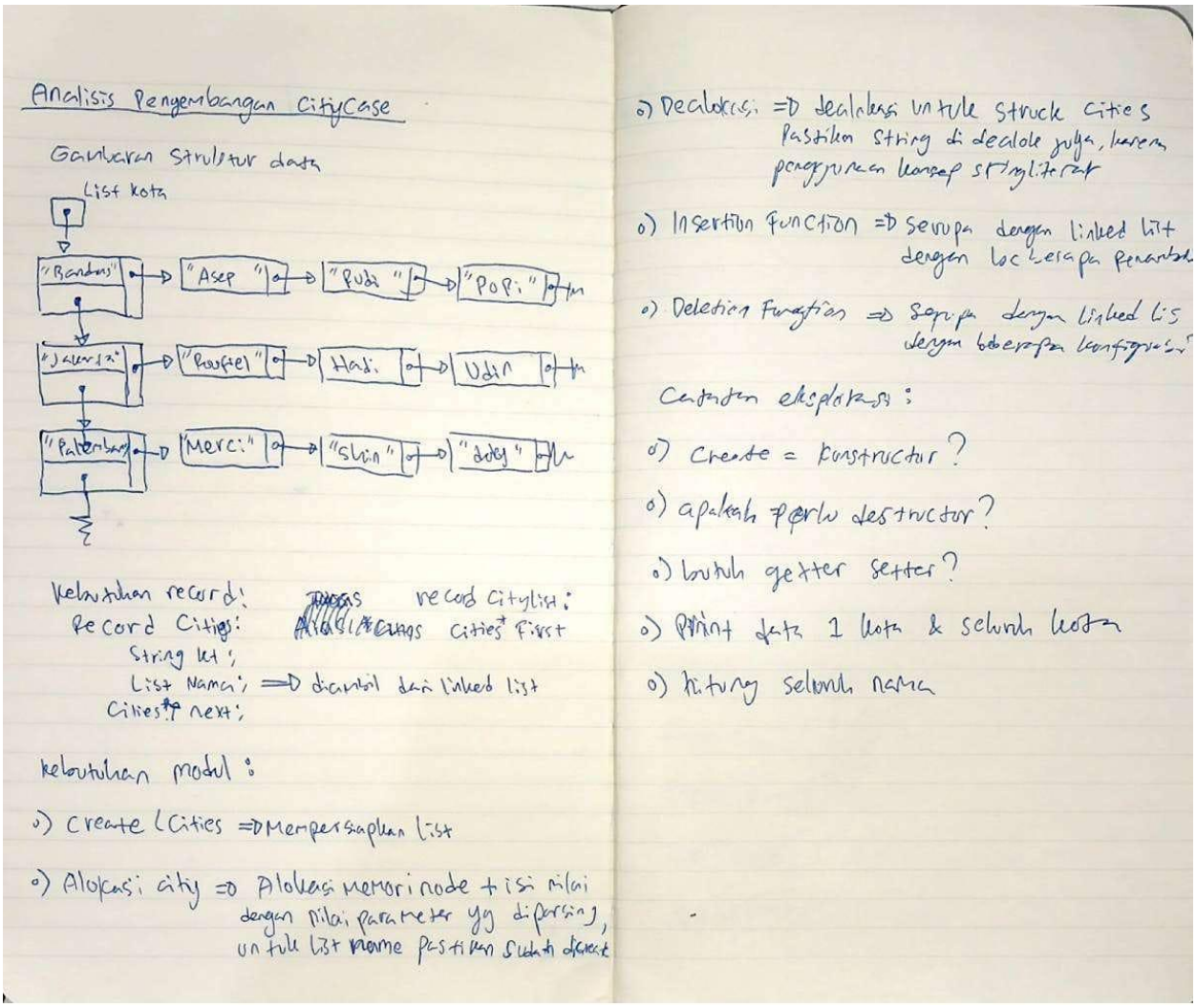
PENJELASAN KASUS 6

Studi Kasus 6: Kembangkan program studi kasus 5 yang sebelumnya berupa array statis menjadi sebuah linked list

Untuk menyelesaikan studi kasus ke-6, saya mengubahnya menjadi enam product backlog sebagai berikut:

1. Meningkatkan dan memperbaiki ADT linked list sehingga dapat digunakan dengan mudah dan flexible
2. Mengubah ADT CaseCity bagian manajemen list kota yang sebelumnya array list menjadi linked list
3. Memperbaiki logic untuk menyesuaikan perubahan struktur data
4. Meningkatkan UI program

1.1 Gambaran Struktur data:



1.2 Komponen ADT yang akan digunakan:

1. Linked.h (sudah dimodifikasi agar bisa menerima char*)
2. Linked.c (implementasi program linked.h) [#include “linked.h”]
3. Main.c (main program) [#include “controller.h”]
4. CaseCity.h (ADT yang akan menampung kebutuhan tambahan untuk array kota dll) [#include “linked.h”]
5. CaseCity.c (implementasi program dari CaseCity.h) [#include “CaseCity.h”]
6. Ui.h (ADT yang akan menyimpan Ui, dipisahkan sebagai bentuk Latihan bila menggunakn GUI)
7. Ui.c (implementasi program dari ui.h) [#include “ui.h”]
8. controller.h (ADT logic untuk memenuhi studi kasus) [#include “ui.h”, #include “CaseCity.h”]
9. controller.c (implementasi program controller.h) [#include “controller.h”]

1.3 Penyelesaian Backlog

1. Meningkatkan dan memperbaiki ADT linked list sehingga dapat digunakan dengan mudah dan flexible

```c
#ifndef LINKED_H
#define LINKED_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "boolean.h"

typedef enum {
    TYPE_INT,
    TYPE_STRING
} DataType;

typedef char* string;
typedef struct tElmtList *address;
typedef struct tElmtList {
    union {
        int intInfo;
        string strInfo;
    };
    address next;
} ElmtList;

typedef struct {
    DataType type;
    address First;
} List;

#define Nil NULL
#define First(L) ((L).First)
#define next(P) ((P)->next)
#define infoInt(P) ((P)->intInfo)
#define infoStr(P) ((P)->strInfo)
```

```c
boolean isEmpty(address P);
boolean ListEmpty(List L);
void CreateAddress(address *P);
void CreateList(List *L, DataType type);
address AlokasiInt(int X);
address AlokasiCharS(char* nama);
void DeAlokasiInt(address P);
void DeAlokasiCharS(address P);
void DeAlokasi(address P);

void InsertFirst(address *P, address newNode);
void InsertLast(address *P, address newNode);
void InsertAfter(address *pBef, address PNew);
void InsertBefore(address *pAft, address *p, address PNew);
void InsertFVInt(address *P, int info);
void InsertFVCharS(address *P, char* info);

#define InsertFirstV(a, b) _Generic((b), \
    int: InsertFVInt, \
    char*: InsertFVCharS \
)(a, b)

void InsertLVInt(address *P, int info);
void InsertLVCharS(address *P, char* info);

#define InsertLastV(a, b) _Generic((b), \
    int: InsertLVInt, \
    char*: InsertLVCharS \
)(a, b)

void InsertAVInt(address *P, int Before, int info);
void InsertAVCharS(address *P, string Before, string info);

#define InsertAfterV(a, b, c) _Generic((b), \
    int: InsertAVInt, \
    char*: InsertAVCharS \
)(a, b, c)

void InsertBVInt(address *P, int After, int info);
void InsertBVCharS(address *P, string After, string info);

#define InsertBeforeV(a, b, c) _Generic((b), \
    int: InsertBVInt, \
    char*: InsertBVCharS \
)(a, b, c)

void DeleteFirstInt(List *L, int *info);
void DeleteFirstCharS(List *L, string *info);

#define DeleteFirst(a, b) _Generic((b), \
    int*: DeleteFirstInt, \
    char**: DeleteFirstCharS \
)(a, b)
```

```c
void DeleteLastInt(List *L, int *info);
void DeleteLastCharS(List *L, string *info);

#define DeleteLast(a, b) _Generic((b), \
    int*: DeleteLastInt, \
    char**: DeleteLastCharS \
)(a, b)

void DeleteByVInt(List *L, int info);
void DeleteByVCharS(List *L, string info);

#define DeleteByValue(a, b) _Generic((b), \
    int: DeleteByVInt, \
    char*: DeleteByVCharS \
)(a, b)

void ClearList(List *L);

void printInfoInt(List L);
void printInfoCharS(List L);

#define PrintList(L) ((L).type == TYPE_INT ? printInfoInt(L) : printInfoCharS(L))

int CountList(List L);
address SearchPrec(List L, int info);
address SearchByVInt(List L, int info);
address SearchByVCharS(List L, char* info);

#define SearchByValue(a, b) _Generic((b), \
    int: SearchByVInt, \
    char*: SearchByVCharS \
)(a, b)

address getReverseList(List L);
void ReverseList(List *L);
void CopyList(List L1, List *L2);
int GetFrontValueInt(List L);
string GetFrontValueCharS(List L);

#define GetFrontValue(L) ((L).type == TYPE_INT ? GetFrontValueInt(L) :
GetFrontValueCharS(L))

int GetTailValueInt(List L);
string GetTailValueCharS(List L);

#define GetTailValue(L) ((L).type == TYPE_INT ? GetTailValueInt(L) : GetTailValueCharS(L))


#endif
```

Peningkatan utama yang saya lakukan pada ADT linked adalah optimalisasi fleksibelitas dengan menerapkan fitur modern C yaitu macro generic untuk mendukun tipe data integer dan string sekaligus. Peningkatan ini dilakukan dengan tujuan agar ADT linked dapat beradaptasi dengan tipe data integer dan String. Untuk keseluruhan perubahan pada ADT linked saya uraikan menjadi butir – butir berikut:

A. Optimalisasi fleksibelitas tipe data
B. Merapihkan struktur penulisan kode
C. Menambahkan penjelasan komentar yang lebih lengkap (pada gambar dihapus)
D. Menambahkan beberapa modul baru (untuk mendukung fleksibelitas)

2. Mengubah ADT CaseCity bagian manajemen list kota yang sebelumnya array list menjadi linked list

```h
#ifndef CASECITY_H
#define CASECITY_H
#include "linked.h"
#include "boolean.h"
#define Head(L) ((L).Head)
#define next(P) ((P)->next)
#define kota(P) ((P)->kt)
#define ListName(P) ((P)->L)

extern int i;

typedef struct kota *Pcity;
typedef struct kota{
  Pcity next;
    char* kt;
    List L;
} CityData;

typedef struct {
    Pcity Head;
} ListCity;

boolean ListCityEmpty(ListCity C);
void CreatePcity(Pcity *A);
void CreateCityList(ListCity *C);
Pcity Constructor(char* cityName);
void Destructor(Pcity A);

void InsertAwal(Pcity *A, Pcity newNode);
void InsertAkhir(Pcity *A, Pcity newNode);
void InsertSetelah(Pcity *pBef, Pcity PNew);
void InsertSebelum(Pcity *pAft, Pcity *p, Pcity PNew);
void InsertAwalV(Pcity *P, char *kota);
void InsertAkhirV(Pcity *P, char *kota);

void DeleteAwal(ListCity *C);
void DeleteAkhir(ListCity *C);
void DeleteByName(ListCity *C, char* cityName);
void ClearListCity(ListCity *C);
```

```
40
41 Pcity FindCity(ListCity C, char* cityName);
42 void PrintAllCityData(ListCity C);
43 void PrintCityData(Pcity cityNode);
44 int countCity(ListCity C);
45 int countNameInCity(Pcity cityNode);
46 int CountTotalNames(Pcity A);
47 void CopyListCity(ListCity C1, ListCity *C2);
48
49 #endif
```

Pada bagian CaseCity perubahan utama yang dilakukan tentu saja pada bagian manajemen struktur data list kota, penggunaan init array sekarang sudah berubah menjadi constructor, destructor dan lain sebagainya. Pengimplementasian konsep linked list diterapkan diseluruh ADT CaseCity. Pembagiannnya secara fungsinya dijabarkan melalui butir – butir berikut:

a. Entry : menggunakan modul insertion yaitu InsertAwal, InsertAkhir, InsertAfter, InsertBefore, InsertAwalV, dan InsertAkhirV
b. Delete : Menggunakan bagaana modul deletion yaitu DeleteAwal, DeleteAkhir, DeleteByName, dan ClearListCity
c. Display : PrintAllCityData dan PribstCityData
d. Count : countCity. countNameInCity, dan countTotalNames

3. Memperbaiki logic untuk menyesuaikan perubahan struktur data
Perubahan pada ADT logic sebenarnya merupakan bentuk adaptasi dari perubahan manajemen list data kota. Penamaan logic saya ubah menjadi controller karena saya rasa lebih tepat untuk menggambaran fungsi ADT tersebut. Perubahan yang banyak terjadi pada ADT ini adalah penggunaan string literal (char*) pada parameter dan sebagai bagian dari proses input user (sebelumnya menggunakan array of char). Penerapan ini memerlukan kehati – hatian karena dapat menyebabkan leak of memory jika tidak ditangani dengan baik. Berikut contoh penerapannya
Sebelum:

```
1  void handleDeleteName(CityData* cities, int cityCount) {
2      char cityName[MAX_CITY_LENGTH];
3      char personName[MAX_NAME_LENGTH];
4
5      readCityName(cityName);
6      int cityIndex = FindCity(cities, cityCount, cityName);
7
8      if (cityIndex >= 0) {
9          readPersonName(personName);
10         DeleteNameFromCity(cities, cityIndex, personName);
11         displayNameDeletedMessage(personName, cityName);
12     } else {
13         displayCityNotFoundMessage(cityName);
14     }
15 }
```

Sesudah:

```c
void handleDeleteName(ListCity *cities) {
    char *cityName = NULL;
    char *personName = NULL;

    system("cls");
    drawbox(50, 9);
    gotoxy(10, 2);
    printf("Delete Person from City");

    gotoxy(5, 4);
    readCityName(&cityName);

    Pcity city = FindCity(*cities, cityName);
    if (city == NULL) {
        gotoxy(5, 6);
        displayCityNotFoundMessage(cityName);
        free(cityName);
        waitForEnter();
        return;
    }

    gotoxy(5, 6);
    readPersonName(&personName);

    if (ListEmpty(ListName(city))) {
        gotoxy(5, 8);
        printf("No people registered in city '%s'!", cityName);
        free(cityName);
        free(personName);
        waitForEnter();
        return;
    }

    address person = SearchByValue(ListName(city), personName);
    if (person == NULL) {
        gotoxy(5, 8);
        printf("Person '%s' not found in city '%s'!", personName, cityName);
        free(cityName);
        free(personName);
        waitForEnter();
        return;
    }

    DeleteByValue(&(ListName(city)), personName);

    gotoxy(5, 8);
    displayNameDeletedMessage(personName, cityName);

    free(cityName);
    free(personName);

    waitForEnter();
}
```

Dari kedua gambar diatas kita dapat melihat penggunaan string literal sebagai penerima input user disini. Pada modul handleDeleteName input user digunakan untuk mencari node yang akan dihapus, setelah kita mendapatkan node yang kita inginkan kita perlu menghapus atau mendealokasikan memori variabel yang sebelumnya menampung string karena sudah tidak digunakan lagi.

4. Meningkatkan UI program

   Pada program kali ini saya menerapkan draw box dan gotoxy sebagai penunjang ui program. Penerapan ini membuat program hasil program lebih baik, berikut hasilnya: