LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 4
ADT Linear Linked List

DISUSUN OLEH :

AJI KARTIKO HARTANTO - 2350081062

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN INFORMATIKA

UNIVERSITAS JENDERAL ACHMAD YANI

TAHUN 2023

DAFTAR ISI

DAFTAR GAMBAR

BAB I. TUGAS PRAKTIKUM

I.1 Tugas ADT Linear Linked List

I.1.A. Source Code Boolean.h

```
/*
    Program     : boolean.h
    Deskripsi   : Header dari file boolean
*/
#ifndef boolean_H
#define true 1
#define false 0
#define boolean unsigned char
#endif
```

I.1.B. Source Code list.h

```
/*
    Program     : list.h
    Author      : 2350081062, Aji Kartiko Hartanto
    Kelas       : C
    Deskripsi   : Header file dari prototype linked list
    Tanggal     : 03-04-2024
*/



#ifndef _LIST_H
#define _LIST_H

#include <stdio.h>
#include <conio.h>

#include "boolean.h"

#define Nil NULL
#define Info(P) (P)->info
#define Next(P) (P)->next
```

```
#define First(L) (L).First

// Definisi ADT List
typedef struct tElmList *address;
typedef int infoType;

typedef struct tElmList {
    infoType info;
    address next;
} ElmList;

typedef struct {
    address First;
} List;

// Prototype linear list
// konstruktor
void CreateList(List *L);

// Destruktor/Dealokator
address Alokasi(infoType X);

void DeAlokasi(address P);

// {Kelompok operasi cek elemen kosong atau penuh}
boolean ListEmpty(List L);

// {Kelompok interaksi dengan I/O device, Baca/Tulis}
// Penambahan elemen
void InsFirst(List *L, infoType X);

void InsLast(List *L, infoType X);

void InsAfter(List *L, infoType X, infoType Y);

// Penghapusan elemen
void DelFirst(List *L, infoType *X);
```

```
void DelLast(List *L, infoType *X);

void DelAfter(List *L, infoType *X, infoType  Y);

// cetak list
void PrintInfo(List L);

// Kelompok operasi lain terhadap type
int NbElm(List L);

address Search(List L, infoType X);

void InversList(List *L);

List getNewInversList(List L);

#endif
```

I.1.C. Source Code list.c

```
/*
    Program     : list.c
    Author      : 2350081062, Aji Kartiko Hartanto
    Kelas       : C
    Deskripsi   : Prototype file ADT linked list
    Tanggal     : 03-04-2024
*/



#include <stdio.h>
#include <stdlib.h>

#include "list.h"

// Prototype linear list
// konstruktor
void CreateList(List *L) {
```

```
    First(*L) = Nil;
}

// Destruktor/Dealokator
address Alokasi(infoType X) {
    // Kamus
    address NewNode;

    // Alokasi memori menggunakan Malloc
    NewNode = (ElmList *) malloc(sizeof(ElmList));

    // algoritma
    Info(NewNode) = X;
    Next(NewNode) = Nil;

    return NewNode;
}

void DeAlokasi(address P) {
    // Dealokasi memori (membebaskan/menghapus alamat memori)
    free(P);
}

// {Kelompok operasi cek elemen kosong atau penuh}
boolean ListEmpty(List L) {
    if (First(L) == Nil) {
        return true;
    } else {
        return false;
    }
}

// {Kelompok interaksi dengan I/O device, Baca/Tulis}
// Penambahan elemen
void InsFirst(List *L, infoType X) {
    // Kamus
    address NewNode;
```

```
    // Alokasi memori
    NewNode = Alokasi(X);

    // algoritma
    if (NewNode != Nil) {
        Info(NewNode) = X;
        Next(NewNode) = First(*L);

        First(*L) = NewNode;
    }
}

void InsLast(List *L, infoType X) {
    // Kamus
    address NewNode, Current;

    // Alokasi memori
    NewNode = Alokasi(X);

    // algoritma
    Current = First(*L);

    if (NewNode != Nil) {
        while (Next(Current) != Nil) {
            Current = Next(Current);
        }

        Info(NewNode) = X;
        Next(NewNode) = Nil;

        Next(Current) = NewNode;
    }
}

void InsAfter(List *L, infoType X, infoType Y) {
    // Kamus
```

```
    address NewNode, Current;
    int i;

    if (Y <= 1) {
        printf("Data akan ditambahkan ke bagian awal\n");
        InsFirst(L, X);
    } else if (Y >= NbElm(*L)) {
        printf("Data akan ditambahkan ke bagian akhir\n");
        InsLast(L, X);
    } else {
        // alokasi
        NewNode = Alokasi(X);

        // Algoritma
        Info(NewNode) = X;
        Current = First(*L);

        for (i = 1; i < Y - 1; i++) {
            Current = Next(Current);
        }

        Next(NewNode) = Next(Current);
        Next(Current) = NewNode;
    }
}

// Penghapusan elemen
void DelFirst(List *L, infoType *X) {
    // kamus
    address DelNode, Tmp;

    // algoritma

    DelNode = First(*L);
    First(*L) = Next(First(*L));

    // dealokasi memori
```

```
    *X = Info(DelNode);
    Tmp = DelNode;
    DeAlokasi(Tmp);
}

void DelLast(List *L, infoType *X) {
    // kamus
    address DelNode, Current, Before, Tmp;
    int i;

    // algoritma
    i = 1;
    Current = First(*L);

    while (i <= NbElm(*L) && Current != Nil) {
        if (i == NbElm(*L) - 1) {
            Before = Current;
        }

        if (i == NbElm(*L)) {
            DelNode = Current;
        }

        Current = Next(Current);
        i++;
    }

    Next(Before) = Nil;

    *X = Info(DelNode);
    Tmp = DelNode;
    DeAlokasi(Tmp);
}

void DelAfter(List *L, infoType *X, infoType Y) {
    // Kamus
    address DelNode, Current, Before, Tmp;
```

```
    int i;

    // Algoritma
    if (Y <= 1) {
        printf("Data akan dihapus pada bagian awal");
        DelFirst(L, X);
    } else if (Y >= NbElm(*L)) {
        printf("Data akan dihapus pada bagian akhir");
        DelLast(L, X);
    } else {
        i = 1;
        Current = First(*L);

        while (i <= Y) {
            if (i == Y - 1) {
                Before = Current;
            }

            if (i == Y) {
                DelNode = Current;
            }

            Current = Next(Current);
            i++;
        }

        Next(Before) = Current;

        *X = Info(DelNode);
        Tmp = DelNode;
        DeAlokasi(Tmp);
    }
}

// cetak list
void PrintInfo(List L) {
    address Current;
```

```
    Current = First(L);
    while (Current != Nil) {
        printf("[%d] -> ", Info(Current));

        Current = Next(Current);
    }

    printf("Null");
}

// Kelompok operasi lain terhadap type
int NbElm(List L) {
    // kamus
    address Current;
    int i;

    // algoritma
    Current = First(L);

    i = 0;
    while (Current != Nil) {
        Current = Next(Current);
        i++;
    }

    return i;
}

address Search(List L, infoType X) {
    // kamus
    address Current;

    // algoritma
    Current = First(L);
    while (Current != Nil) {
        if (Info(Current) == X) {
```

```
            return Current;
        }

        Current = Next(Current);
    }

    return Nil;
}

void InversList(List *L) {
    address Prev, Current, NextNode;

    Prev = Nil;
    Current = First(*L);
    while (Current != Nil) {
        NextNode = Next(Current);
        Next(Current) = Prev;
        Prev = Current;
        Current = NextNode;
    }

    First(*L) = Prev;
}

List getNewInversList(List L) {
    // kamus
    List NewList;

    // algoritma
    CreateList(&NewList);
    InversList(&L);

    NewList = L;

    return NewList;
}
```

I.1.D. Source Code mlist.c

```
/*
    Program     : mlist.c
    Author      : 2350081062, Aji Kartiko Hartanto
    Kelas       : C
    Deskripsi   : main driver ADT linked list
    Tanggal     : 05-04-2024
*/



#include <stdio.h>
#include "list.c"

int main() {
    List node1;
    infoType tmpDelFirst, tmpDelLast, tmpDelafter, infoElm;
    address searchElm;

    // membuat list
    CreateList(&node1);

    // mengecek apakah list kosong atau tidak
    printf("Apakah list kosong? \n");
    if (ListEmpty(node1)) {
        printf("List Masih Kosong");
    } else {
        printf("List tidak Kosong");
    }

    // menambahkan elemen pada list
    printf("\n\n");
    printf("Insert First List\n");
    InsFirst(&node1, 1);
    InsFirst(&node1, 2);
    InsFirst(&node1, 3);
    InsFirst(&node1, 4);
```

```
// Cetak List
PrintInfo(node1);

// mengecek apakah list kosong atau tidak
printf("\n\n");
printf("Apakah list masih kosong? \n");
if (ListEmpty(node1)) {
    printf("List Masih Kosong");
} else {
    printf("List tidak Kosong");
}

// menambahkan elemen terakhir pada list
printf("\n\n");
printf("Insert last list\n");
InsLast(&node1, 5);
InsLast(&node1, 6);

// Cetak List
PrintInfo(node1);

// menambahkan elemen setelah node ke Y pada list
printf("\n\n");
printf("Insert after list\n");
InsAfter(&node1, 10, 1);
InsAfter(&node1, 20, 2);

// Cetak List
PrintInfo(node1);

// menghapus elemen pertama pada list
printf("\n\n");
printf("Delete first list\n");
DelFirst(&node1, &tmpDelFirst);
DelFirst(&node1, &tmpDelFirst);
```

```
    // Cetak List
    PrintInfo(node1);

    // menghapus elemen akhir pada list
    printf("\n\n");
    printf("Delete Last list\n");
    DelLast(&node1, &tmpDelLast);
    DelLast(&node1, &tmpDelLast);

    // Cetak List
    PrintInfo(node1);

    // menghapus elemen setelah node ke Y pada list
    printf("\n\n");
    printf("Delete After list\n");
    DelAfter(&node1, &tmpDelafter, 3);

    // Cetak List
    PrintInfo(node1);

    // invers list
    printf("\n\n");
    printf("Invers pada list\n");
    PrintInfo(getNewInversList(node1));

    // mencari element list dengan info
    printf("\n\n");
    printf("Masukan Elemen yang dicari: ");
    scanf("%d", &infoElm);

    searchElm = Search(node1, infoElm);
    if (searchElm != Nil) {
        printf("\nElemen %d ada di list dengan alamat %d", infoElm,
searchElm);
    } else {
        printf("Elemen %d tidak ada di list", infoElm);
    }
```

```
    // mengirim banyak nya elemen pada list
    printf("\n\n");
    printf("Banyaknya elemen pada list adalah %d\n", NbElm(node1));

    return 0;
}
```

I.1.E. Hasil

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Apakah list kosong?
List Masih Kosong

Insert First List
[4] -> [3] -> [2] -> [1] -> Null

Apakah list masih kosong?
List tidak Kosong

Insert last list
[4] -> [3] -> [2] -> [1] -> [5] -> [6] -> Null

Insert after list
Data akan ditambahkan ke bagian awal
[10] -> [20] -> [4] -> [3] -> [2] -> [1] -> [5] -> [6] -> Null

Delete first list
[4] -> [3] -> [2] -> [1] -> [5] -> [6] -> Null

Delete Last list
[4] -> [3] -> [2] -> [1] -> Null

Delete After list
[4] -> [3] -> [1] -> Null

Invers pada list
[1] -> [3] -> [4] -> Null

Masukan Elemen yang dicari: 4

Elemen 4 ada di list dengan alamat 12986872

Banyaknya elemen pada list adalah 1
```

Gambar I.1 Output Program Tugas ADT Linear Linked List

I.1.F. Analisa

1. Inisialisasi: Dengan mengatur awalnya ke Nil, CreateList mempersiapkan daftar kosong.

2. Alokasi node: Alokasi menghasilkan node baru dengan data yang telah ditentukan, mengalokasikan memori, dan menginitialisasi bidang untuk penyimpanan dan manajemen link.

3. Dealokasi node: Dealokasi melepaskan memori yang diduduki oleh node tertentu, yang dapat digunakan untuk manajemen memori secara gratis.

4. Periksa daftar kosong: ListEmpty memeriksa elemen pertama untuk memastikan apakah daftar kosong.

5. Menambahkan di Mulai: InsFirst menggunakan Alokasi untuk menggabungkan node baru dengan data yang ditentukan di bagian depan daftar. Kemudian, tautan diperbarui sesuai.

6. Menambahkan pada End: InsLast menggunakan Alokasi untuk menambahkan node baru dengan data yang ditunjuk ke terminus daftar. Ini mengatur elemen pertama secara langsung (jika kosong) atau melalui daftar untuk menemukan node akhir untuk mengubah tautan.

7. Insert After Node: InsAfter mengintegrasikan node baru dengan data setelah node yang ditentukan dan diberi nilai. Ini menangani kasus batas, seperti penyerahan awal atau akhir, dan sebaliknya memanggil Alokasi untuk membangun tautan penyerahan yang tepat.

8. Fungsi untuk menghapus dan mengubah node dari daftar tertaut diatur dalam kode. DelFirst menghapus node awal, menyimpan data, dan memperbarui penunjuk kepala daftar. DelAfter menargetkan node tertentu berdasarkan nilai sebelumnya, menghapus node berikutnya, dan menyimpan data. PrintInfo mencetak data dari setiap node sepanjang daftar. Akhirnya, NbElm menghitung jumlah elemen secara keseluruhan dengan mengulangi daftar dan mengikuti penghitung. Fungsi ini menawarkan operasi

penting untuk mempertahankan dan berinteraksi dengan struktur data daftar yang terhubung.

9. Mencari keberadaan nilai X di dalam list L adalah tugas fungsi Search(L, X). Fungsi ini bekerja dengan iterasi melalui list dan mengembalikan alamat node yang mengandung X jika ditemukan, atau Nil jika tidak ditemukan.

10. Untuk membalikkan urutan node dalam list L secara langsung (in-place), fungsi InversList(L) menggunakan tiga pointer. Ini membalikkan arah link node dan mengubah elemen First dari list ke node terakhir setelah pembalikan.

11. Tujuan fungsi getNewInversList(L) adalah untuk membuat list baru yang merupakan kebalikan dari list asli L. Namun, fungsi ini hanya membalikkan list asli L secara langsung (in-place) menggunakan InversList, sehingga tidak membuat list baru yang terpisah. Fungsi ini juga mengembalikan list asli L yang sudah dibalik.

BAB II. KESIMPULAN

Dengan menggunakan materi ADT linked list linked list, praktikum modul ini memberikan pemahaman mendalam tentang salah satu struktur data dasar dalam ilmu komputer. Praktikum ini mengajarkan kita konsep dasar tentang linked list, seperti cara membuat, menelusuri, menambah, dan menghapus node dari linked list, serta keunggulan dan kelemahan linked list dibandingkan dengan struktur data seperti array.

Linked List adalah salah satu struktur data yang paling sering digunakan dalam pengembangan aplikasi, terutama aplikasi yang membutuhkan operasi insert dan delete yang sering. Memahami Linked List akan membantu kita merancang dan mengimplementasikan algoritma yang efisien untuk memecahkan berbagai masalah komputasi. Selain itu, praktikum ini memberikan dasar yang kuat untuk mempelajari lebih lanjut tentang algoritma dan struktur data yang lebih kompleks.