Pemahaman Modul

Praktikum Struktur Data.

Kode_Asisten_PJ_:_MTN Kelompok = A11.4301U

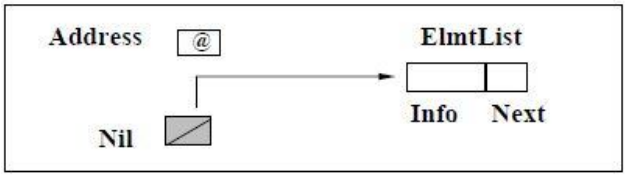
NIM :A11.2012.06758 NIM : A11.2012.06773

Nama :Muhammad Adhi D Nama : Cahyo Dwi Saputro

PSDA – 03 (List Linier Eksplisit).

A. Definisi List Linier

- List Linier merupakan sekumpulan elemen yang bertype sama yang mempunyai criteria tertentu.Setiap Elemen terdiri dari 2 bagian, yaitu :
  - Informasi mengenai Elemen (INFO).
  - Informasi mengenai alamat elemen suksesor (NEXT).
- Sebuah List Linier dapat dikenali beberapa komponennya,yaitu :
  - Elemen pertamanya, didefiniskan sebagai alamat elemen pertama pada List Linier, (FIRST).



  - Alamat Elemen berikutnya, jika alamat sebuah elemen diketahui, maka alamat elemen selanjutnya juga dapat diketahui , (NEXT).
  - Elemen Terakhirnya, perlu dilakukan Traversal untuk mencapai elemen terakhir list.
- Beberapa definisi “Selektor” dan struktur ADT dari List Linier, ditunjukan pada gambar di bawah.

```
#define INFO(P) (P)->Info
#define NEXT(P) (P)->NEXT
#define FIRST(L) (L).FIRST
#define NIL NULL

typedef int InfoType;
typedef struct tElmtList *address;
typedef struct tElmtList
{
    InfoType Info;
    address NEXT;
}ElmtList;
typedef struct
{
    address FIRST;
}List;
```

Variabel buatan bertype Integer, diberi nama InfoType.kemudian mendefinisikan ADT sebagai “Address” dari list Linier, dimana variabel ini aka mengacu suatu alamat dari elemen tertentu, ditandai dengan pointer ‘*’, untuk menunjuk suatu alamat tertentu.

ADT ElmtList untuk tiap elemen terdiri dari 2 komponen utama yaitu, “**INFO**” dan “**NEXT**”.

ADT List untuk mengacu pada Elemen pertama List, komonenya adalah “**FIRST**”, dimana variabel ini bertype “Address”.

B. Definisi dari beberapa fungsi.

```
/*1*/void createList (List *L);
/*2*/boolean IsEmpty (List L);
/*3*/address ALOKASI (InfoType X);
/*4*/void DEALOKASI (address P);
//////////////////////////////////////////
/*5*/void InsVFirst (List *L,InfoType X);
/*6*/void First (List *L,address P);
//////////////////////////////////////////
/*7*/void After (address P,address Prec);
/*8*/void Last (List *L,address P);
/*9*/void InsVLast (List *L,InfoType X);
//////////////////////////////////////////
/*10*/void ShowData (List L);
/*11*/int NbElmt (List L);
/*12*/int MAX (List L);
/*13*/address AdrMAX (List L);
/*14*/int MIN (List L);
/*15*/address AdrMIN (List L);
/*16*/float AVERAGE (List L);
```

```
/*17*/void DellAll (List *L);
/*18*/void InversList (List *L);
/*19*/List FInversList (List L);
/*20*/void CopyList (List *L1, List *L2);
/*21*/List FCopyList (List L);
/*22*/void CPAlokList (List Lin,List *Lout);
/*23*/void CONCAT (List *L1,List *L2, List *L3);
/*24*/void CONCAT_DEL (List *L1,List *L2, List *L3);
//////////////////////////////////////////
/*25*/void Del_First (List *L,address *P);
/*26*/void DelVFirst (List *L,InfoType *X);
/*27*/void Del_Last (List *L,address *P);
/*28*/void DelVLast (List *L,InfoType *X);
/*29*/void DelP (List *L,InfoType X);
/*30*/void DelAfter (address P,address Prec);
//////////////////////////////////////////
/*31*/address Search_Add (List L,InfoType X);
/*32*/address SearchPrec (List L,InfoType X);
/*33*/boolean Search (address P);
/*34*/void PecahList (List *L1, List *L2,List L);
```

1. CreateList (List *L).

Membuat List baru dengan insialisasi yaitu Elemen Pertama diisikan dengan NULL. Parameter bertype List dengan pointer, dimana dapat sebagai Input/output.

I.S (Initial State) : Belum Terdefinisi.

F.S (Final State) : Alamat Elemen Pertama List = NULL.

Algoritma.

```
FIRST(L) <--- NULL
```

2. IsEmpty (List L).

Mengecek kondisi pada List, apakah kosong, atau terisi?, jika kosong maka bernilai TRUE, jika tidak maka FALSE.

I.S (Initial State) : nilai Boolean default = FALSE.

F.S (Final State) : Boolean bisa bernilai TRUE/FALSE.

Algoritma.

```
<--- (FIRST(L) == NULL).
```

3. ALOKASI (InfoType X).

Mengalokasikan sebuah addres baru.fungsi ini berfungsi untuk mendapatkan addres, sekaligus memasukan suatu nilai X kedalam addres tersbut.

I.S (Initial State) : Addres belum terdefinisi.

F.S (Final State) : Addres sudah terdefinisi,sekaligus berisi nilai X.

Algoritma.

P = Malloc of Addres, sizeof (ElmtList).

if (P!=NULL) then, INFO(P) <--- X , NEXT(P) <--- NULL.

4. DEALOKASI (Address P).

Menghapus semua elemen yang ada pada Addres P.semua data dan alamat yang ada pada addres P akan dihapus/didealokasi.

I.S (Initial State) : Addres P masih ada/kosong

F.S (Final State) : Addres P telah didedalokasi, dan kosong.

Algoritma.

Free (P).

5. InsVFirst (List *L, InfoType X).

Menambah nilai X ke dalam elemen list. Dengan posisi penambahan selalu dilakukan pada elemen pertama (FIRST).

I.S (Initial State) : List L kemungkinan kosong.

F.S (Final State) : sebuah elemen dialokasi dan menjadi elemen pertama list tersebut.

Algoritma.

P <--- Alokasi (X)

If (P!=NULL) then ,

*NEXT(P) <--- FIRST(*L)*

*FIRST(*L) = P.*

6. InsVLast (List *L,InfoType X).

Menambah nilai X ke dalam elemen list. Dengan posisi penambahan selalu dilakukan pada elemen terakhir (LAST).

I.S (Initial State) : List L kemungkinan kosong.

F.S (Final State) : sebuah elemen dialokasi dan menjadi elemen terakhir list tersebut.

Algoritma.

P <--- Alokasi (X)

If (P!=NULL) then , Last (L,P)

7. InverseList (List *L).

Membalik urutan list semula, menjadi terbalik.

I.S (Initial State) : List L tidak kosong.

F.S (Final State) : Urutan Elemen dari List Menjadi terbalik.

Algoritma.

```
P <--- FIRST(*l)

Q <--- NIL

While (P!=NIL) , do

R <--- NEXT(P) , NEXT(P) <--- Q , Q <--- P, P <--- R

FIRST(*L) <--- Q
```

8. CopyList (List L).

Menyalin semua elemen List L1, kepada List L2, pada fungsi ini dilakukan alokasi memori baru pada List yang akan di gunakan sebagi tempat salinan list.alur dari fungsi ini adalah, dengan cara memasukan nilai dari elemen List L, ke dalam tiap elemen List yang baru disertai dengan alokasi memori.

I.S (Initial State) : L kemungkinan kosong.

F.S (Final State) : List 2 sudah tersalin dari list 1.

Algoritma.

```
P <--- FIRST(*l)

While (P!=NULL) do ,

InsVLast (&L2,INFO(P))

P = NEXT(P)

<--- L2
```

9. Searching (List L, InfoType X).

Mencari elemen X, pada List L. pencarian dilakukan dengan cara melakukan Traversal (penelusuran) hinnga ketemu dengan nilai X atau mencapai Elemen terakhir.Parameternya adalah list, dan X sebagai nilai yang dicari,nilai baliknya adalah berupa Boolean, dimana bernilai TRUE/FALSE.

I.S (Initial State) : L tidak kosong.

F.S (Final State) : nilai balik Boolean, TRUE jika ketemu, FALSE jika tidak ketemu.

Algoritma.

```
P <--- FIRST(L)
FIND <--- FALSE
While (P!=NIL) do ,
      If (INFO(P)==X) then ,
            FIND <--- TRUE
      P <--- NEXT(P)
<--- FIND
```

10. Concat (List *L1, List *L2, List *L3)

Menggabungkan antara List L1 List L2, dan hasil gabungan antara L1 dan L2 diserahkan pada List 3. Penggabungan dilakukan dengan cara menelusuri L1 hingga mencapai elemen terakhir, kemudian NEXT dari elemen terahir L1, disambungkan dengan FIRST dari L2.

I.S (Initial State) : L1 dan L2 bebas kondisi.

F.S (Final State) : L3 terisi dengan gabungan antar L1 dan L2.

Algoritma.

```
P <--- FIRST(*L1)
While (NEXT(P)!=NULL) do, P <--- NEXT(P)
NEXT(P) <--- FIRST (*L2)
*L3 <--- *L1
```

11. PecahList (List *L1, List *L2, List L)

Memecah List L menjadi 2 bagian yaitu L1 dan L2 , jika jumlah L genap maka, jumlah pemecahan list menjadi setengah bagian, jika jumlah L ganjil, maka jumlah bagian yang 1-nya adalah (jumlah L div 2).Pada prosedur ini dilakukan alokasi memori pada List L1 dan L2.

I.S (Initial State) : L tidak kosong.

F.S (Final State) : L2 dan L3 terisi dngn elemen pecahan dr L.

Algoritma.

```
P <--- FIRST(L)
If (NbElmt(L) mod 2 == 0) then ,
  traversal i...NbElmt(L) div 2
      InsVLast(L1,Info(P))
      P <--- NEXT(P)
  traversal i...NbElmt(L)
      InsVLast(L2,Info(P))
      P <--- NEXT(P)
Else,
      {proses traversal sama dengan kondisi pertama}.
```

12. MIN dan MAX (List L)

Mencari Nilai Max atau Min pada list L. pencarian dilakukan dengan membandingkan antara nilai dari INFO(P) dengan Nilai INFO(P) yang lain.

I.S (Initial State) : L sembarang kondisi.

F.S (Final State) : Nilai Max atau Min dapat didapatkan.

Algoritma.

```
P <--- FIRST(L)
Prec <--- NULL
Min <--- INFO(P)
While (NEXT(P)!=NULL) do,
      Prec <--- P
      P <--- NEXT(P)
      If (Min > INFO(P)) then , Min <--- INFO(P)
<--- Min
```