List Rekursif dalam Konteks Prosedural (1)

IF2110/IF2111 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

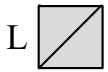
List sebagai Struktur Data Rekursif

Definisi rekursif list linier:

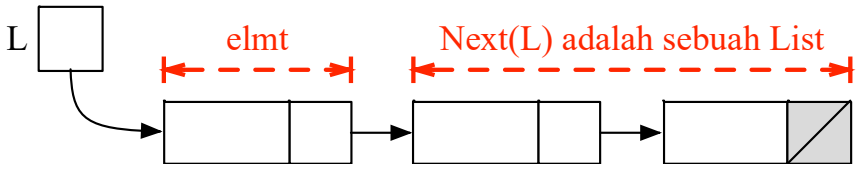
Basis: list kosong adalah list

Rekurens: list tidak kosong terdiri atas sebuah elemen dan sisanya adalah list

List L kosong



List L dengan tiga elemen



Struktur Data List untuk Pemrosesan secara Rekursif (Notasi Algoritmik)

```
KAMUS
{ List direpresentasi dg pointer }
    type  ElType: ... { terdefinisi }
    type Address: ... { terdefinisi }
    type    Node: < info: ElType,
                    next: Address >
    type List: Address


{ Deklarasi  nama untuk variabel  kerja }
    l: List
    p: Address        { address untuk traversal }

    { Maka First(l) adalah l
      Next(p), Info(p) tergantung representasi fisik yang
      digunakan }
```

Struktur Data List untuk Pemrosesan secara Rekursif (Bahasa C, pointer)

```
#define NIL NULL

/* Selektor */
#define INFO(p) (p)->info
#define NEXT(p) (p)->next

typedef int ElType;
typedef struct node* Address;
typedef struct node {
    ElType info;
    Address next;
} Node;
/* Definisi list: */
/* List kosong: l = NIL */

typedef Address List;
```

Primitif Dasar: Pemeriksaan List Kosong

Notasi Algoritmik (rep. berkait)

```
function isEmpty(l: List) → boolean
{ Tes apakah sebuah list l kosong.
  Mengirimkan true jika list kosong,
  false jika tidak kosong }
KAMUS LOKAL
    -
ALGORITMA
    → (l = NIL)
```

Bahasa C (rep. berkait dgn. pointer)

```
boolean isEmpty(List l) {
/* Tes apakah sebuah list l kosong.
   Mengirimkan true jika list kosong,
   false jika tidak kosong */
  /* Kamus Lokal */

  /* Algoritma */
    return (l == NIL);
}
```

Studi Kasus-1: displayList

Notasi Algoritmik

```
procedure displayList(input l: List)
{ I.S. l terdefinisi }
{ F.S. Setiap elemen list diprint }
KAMUS LOKAL
ALGORITMA
    if (isEmpty(l)) then { Basis 0 }
        { tidak melakukan apa-apa }
    else { Rekurens }
        output(l↑.info)
        displayList(l↑.next)
```

Studi Kasus-2a: NbElmtList (1)

Versi **fungsi**

```
function length(l: List) → integer
{ Mengirimkan banyaknya elemen list l, Nol jika l kosong }
KAMUS LOKAL
ALGORITMA
    if (isEmpty(l)) then { Basis 0 }
        → 0
    else { Rekurens }
        → 1 + length(l↑.next)
```

Studi Kasus-2b: NbElmtList (2)

Versi **prosedur**, dengan hasil diletakkan pada parameter output

```
procedure length(input l: List, output n: integer)
{ I.S. l terdefinisi
  F.S. n berisi banyaknya elemen list }
KAMUS LOKAL
    n1: integer
ALGORITMA
    if (isEmpty(l)) then { Basis 0 }
        n ← 0
    else { Rekurens }
        length(l↑.next,n1)
        n ← 1 + n1
```

Studi Kasus-2c: NbElmtList (3) - 1

Versi prosedur, dengan akumulator

```
procedure lengthAcc(input l: List, input acc: integer, output n: integer)
{ I.S. l terdefinisi
  F.S. n berisi banyaknya elemen list }
KAMUS LOKAL
ALGORITMA
    if (isEmpty(l)) then { Basis 0 }
        n ← acc
    else { Rekurens: Next element, Proses }
        length(l↑.next,acc+1,n)
```

Studi Kasus-2c: NbElmtList (3) - 2

Pemanggilan **lengthAcc**

```
procedure length(input l: List, output n: integer)
{ I.S. l terdefinisi
  F.S. n berisi banyaknya elemen list l
  Proses: Memanfaatkan lengthAcc }
KAMUS LOKAL
ALGORITMA
    lengthAcc(l, 0, n)
```

Studi Kasus-3: search

```
function search(l: List, x: ElType) → boolean
{ Mengirim true jika x adalah anggota list, false jika tidak }
KAMUS LOKAL

ALGORITMA
    if (isEmpty(l)) then { Basis 0 }
        → false
    else
        → (l↑.info = x) or search(l↑.next, x)
```

Studi Kasus-3: search, versi 2

```
function search(l: List, x: ElType) → boolean
{ Mengirim true jika x adalah anggota list, false jika tidak }
KAMUS LOKAL

ALGORITMA
    if (isEmpty(l)) then { Basis 0 }
        → false
    else
        if (l↑.info = x) then { Basis 1 }
            → true
        else { Rekurens }
            → search(l↑.next, x)
```

Studi Kasus-4: Delete Elemen

Prosedur menghapus semua elemen list bernilai X

```
procedure deleteX(input/output l: List, input x: infotype)
{ I.S. l dan x terdefinisi }
{ F.S. semua elemen l yang bernilai x dihapus dari l }
KAMUS LOKAL
    p: Address
ALGORITMA
    if (isEmpty(L)) then { Basis 0 }
        {do nothing}
    else { Rekurens }
        deleteX(l↑.next, x)
        if l↑.info=x then
            p ← l
            l ← l↑.next
            dealokasi(p)
```

List Rekursif dalam Konteks Prosedural (2)

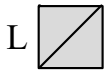
IF2110/IF2111 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

List sebagai Struktur Data Rekursif

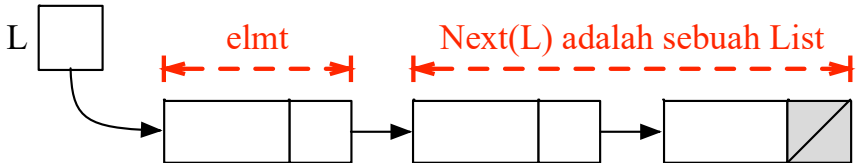
Definisi rekursif list linier:

- **Basis**: list kosong adalah list
- **Rekurens**: list tidak kosong terdiri atas sebuah elemen dan sisanya adalah list

List L kosong

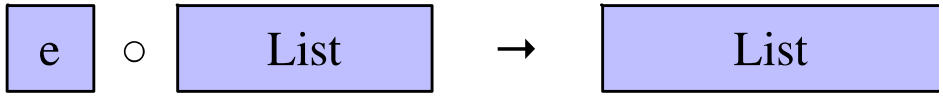


List L dengan tiga elemen



Struktur Data List

type List: [ ] atau [e ○ List]



Primitif dasar (ingat kembali list dalam pemrograman fungsional):

- Selektor: head, tail (≠ head, tail pada queue)
- Konstruktor: kons○, kons●
- Primitif-primitif lain: copy, concat, dll.

Selektor

```
function head(l: List) → ElType
{ Mengirimkan elemen pertama sebuah list l yang tidak kosong }
KAMUS LOKAL
ALGORITMA
    → l↑.info
```

```
function tail(l: List) → List
{ Mengirimkan list l tanpa elemen pertamanya, mungkin yang dikirimkan adalah sebuah list kosong }
KAMUS LOKAL
ALGORITMA
    → l↑.next
```

Konstruktor – Kons○

```
function konso(e: ElType, l: List) → List
{ Mengirimkan list l dengan tambahan e sebagai elemen pertamanya }
{ Jika alokasi gagal, mengirimkan l }
```

```
KAMUS LOKAL
    p: Address
```

```
ALGORITMA
    p ← newNode(e)
    if (p = NIL) then
        → l
    else
        { Insert First }
        p↑.next ← l
        → p
```

Konstruktor – Kons●

```
function kons●(l: List, e: ElType) → List
{ Mengirimkan list l dengan tambahan e sebagai elemen terakhir }
{ Jika alokasi gagal, mengirimkan l }
```

```
KAMUS LOKAL

```

```
ALGORITMA
    if isEmpty(l) then { insert ke list kosong }
        → newNode(e)
    else
        l↑.next ← kons●(tail(l), e)
        → l
```

Konstruktor – Kons●

```
procedure kons●(input/output l: List, input e: ElType)
{ Mengirimkan list l dengan tambahan e sebagai elemen terakhir }
{ Jika alokasi gagal, mengirimkan l }
```

```
KAMUS LOKAL

```

```
ALGORITMA
    if isEmpty(l) then { insert ke list kosong }
        l ← alokasi(e)
    else
        kons●(l↑.next, e)
```

Primitif Lain: Copy – 1 (versi fungsi)

```
function copy(l: List) → List
{ Mengirimkan salinan list l }
{ Jika alokasi gagal, mengirimkan l }
KAMUS LOKAL

ALGORITMA
     if (isEmpty(l)) then { Basis 0 }
         → NIL
     else { Rekurens }
         → konso(head(l), copy(tail(l))
```

Primitif Lain: Copy – 2 (versi procedure)

```
procedure mCopy(input lin: List, output lout: List)
{ I.S. lin terdefinisi }
{ F.S. lout berisi salinan dari lin }
{ Proses: menyalin lin ke lout }
{ Jika alokasi gagal, Lout adalah ??? }
```

```
KAMUS LOKAL
    lTemp: List
```

```
ALGORITMA
    if (isEmpty(lin)) then { Basis – 0 }
        lout ← NIL
    else { Rekurens }
        mCopy(tail(lin), lTemp)
        lout ← konso(head(lin), lTemp)
```

Primitif Lain: Concat – 1 (versi fungsi)

```
function concat(l1, l2: List) → List
{ Mengirimkan salinan hasil konkatenasi list l1 dan l2 }
```

```
KAMUS LOKAL

```

```
ALGORITMA
    if (isEmpty(l1)) then { Basis }
        → copy(l2)
    else { Rekurens }
        → konso(head(l1), concat(tail(l1), l2))
```

Primitif Lain: Concat – 2 (versi procedure)

```
procedure mConcat (input l1, l2: List, output result: List)
{ I.S. l1, l2 terdefinisi }
{ F.S. result adalah hasil melakukan konkatenasi l1 dan l2 dengan cara "disalin" }
{ Proses: Menghasilkan salinan hasil konkatenasi list l1 dan l2 }
```

```
KAMUS LOKAL
    lTemp: List
```

```
ALGORITMA
    if (isEmpty(l1)) then  { Basis – 0 }
        result ← copy(l2)
    else { Rekurens }
        mConcat(tail(l1), l2, lTemp)
        result ← konso(head(l1), lTemp)
```

Potongan header file (1)

```
#ifndef LISTREC_H
#define LISTREC_H

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

#define NIL NULL

typedef int ElType;
typedef struct node* Address;
typedef struct node {
    ElType info;
    Address next;
} Node;

typedef Address List;

/* Selektor */
#define INFO(p) (p)->info
#define NEXT(p) (p)->next
```

Potongan header file (2)

```
/* Manajemen Memori */

Address newNode(ElType x);
/* Mengirimkan address hasil alokasi sebuah elemen */
/* Jika alokasi berhasil, maka address tidak NIL, dan misalnya
   menghasilkan p, maka INFO(p)=x, NEXT(p)=NIL */
/* Jika alokasi gagal, mengirimkan NIL */
```

Potongan header file (3)

```
/* Pemeriksaan Kondisi List */
boolean isEmpty(List l);
/* Mengirimkan true jika l kosong dan false jika l tidak kosong */
int isOneElmt(List l);
/* Mengirimkan true jika l berisi 1 elemen dan false jika > 1 elemen atau kosong */

/* Primitif-Primitif Pemrosesan List */
ElType head(List l);
/* Mengirimkan elemen pertama sebuah list l yang tidak kosong */
List tail(List l);
/* Mengirimkan list l tanpa elemen pertamanya, mungkin mengirimkan list kosong */
List konso(List l, ElType e);
/* Mengirimkan list l dengan tambahan e sebagai elemen pertamanya. e dialokasi terlebih dahulu.
   Jika alokasi gagal, mengirimkan l */

/* Fungsi dan Prosedur Lain */
...

#endif
```

Latihan Soal List Rekursif

IF2110/IF2111 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Soal No. 1

Buatlah fungsi **countPos** yang menghitung banyaknya kemunculan bilangan positif (>0) dari sebuah list of integer l

```
function countPos (l:List) → integer
{ Menghasilkan banyaknya kemunculan bilangan positif (>0)
  pada l }
```

Soal No. 2

Buatlah fungsi **sumPos** yang menghitung penjumlahan semua elemen bilangan positif (> 0) dari sebuah list of integer l. Jika list l kosong, hasilnya adalah 0.

```
function sumPos (l: List) → integer
{ Menghasilkan penjumlahan semua elemen bilangan positif (>0) dari l.
  l kosong menghasilkan 0. }
```

Soal No. 3

Buatlah fungsi **isMember** yang memeriksa apakah sebuah x (integer) merupakan anggota pada sebuah list of integer l

```
function isMember (l:List, x:ElType) → boolean
{ Menghasilkan true jika x adalah salah satu anggota list l,
  false jika tidak }
```

Soal No. 4

Buatlah fungsi **isEqual** yang memeriksa apakah dua buah list of integer l1 dan l2 adalah list yang sama. Sama artinya banyaknya elemen sama dan urutan kemunculan semua elemen juga sama.

```
function isEqual(l1,l2:List) → boolean
{ Menghasilkan true jika l1 dan l2 adalah list yang sama,
  false jika tidak }
```

Soal No. 5

Buatlah procedure **extremes** yang menerima masukan sebuah list of integer yang tidak kosong dan menghasilkan nilai minimum dan maksimum dari list tersebut

```
procedure extremes(input l:List, output min,max:integer)
{ I.S. l terdefinisi, tidak kosong }
{ F.S. min berisi nilai minimum elemen l,
       max berisi nilai maksimum elemen l }
```

Soal No. 6

Buatlah procedure **listPlus** yang menerima masukan dua buah list of integer l1 dan l2 yang mungkin kosong dan memiliki dimensi yang sama serta menghasilkan sebuah list baru yang memiliki dimensi yang sama dan berisi elemen-elemen yang merupakan penjumlahan dari elemen-elemen yang bersesuaian dari l1 dan l2.

```
procedure listPlus(input l1,l2:List; output l3:List)
{ I.S. l1, l2 terdefinisi dengan dimensi yang sama, mungkin kosong. }
{ F.S. l3 berisi elemen-elemen yang merupakan penjumlahan elemen-elemen l1 dan l2
       pada posisi yang bersesuaian. }
```