Pre Lab dan P-ListRecursive

Silahkan Translasikan List Recursive pada Paradigma Fungsional Menjadi Prosedural.

Referensi : Kuliah Algoritma Dan Pemrograman (Bagian Fungsional)

Slide : List

Modul : List Of Integer

Tipe Rekursif:
LIST

Tujuan

- Mahasiswa memahami definisi type rekursif dan rekurens list
- Berdasarkan definisi yang dipahaminya, mahasiswa mampu membuat ekspresi rekursif untuk manipulasi List
- Mahasiswa mampu mengimplementasi fungsi pemroses list dalam LISP → melalui praktikum

Overview Analisis Rekurens

Overview Analisis Rekurens

- Definisi entitas (type, fungsi) disebut rekursif jika definisi tersebut mengandung terminologi dirinya sendiri (diktat hal 53)
- Ekspresi rekursif direalisasikan dengan membuat fungsi rekursif dan didasari analisis rekurens

Analisis Rekurens

- Teks program rekursif terdiri dari dua bagian:
  - **Basis** (Basis-0 atau Basis-1), yang menyebabkan prosedur/fungsi berhenti
  - Bagian **rekurens** : mengandung call terhadap prosedur/fungsi tersebut (aplikasi dari fungsi), dengan parameter bernilai mengecil (menuju basis).
- Tulislah secara eksplisit dalam teks program anda: mana bagian basis, mana rekurens

Basis Nol atau Satu?

- Jika menangani kasus kosong, maka gunakan basis-0. Karena “kosong” adalah ciptaan kita, maka hati-hati dengan nilai yang dihasilkan oleh kasus kosong.
- Jika persoalan hanya ada artinya kalau tidak kosong, maka harus memakai basis 1.
  - Contoh: mencari nilai maksimum dari sebuah list → tidak bisa menggunakan basis kosong karena pada tabel kosong, nilai maksimum tidak terdefinisi.

Type Rekursif

Type Rekursif

- Type rekursif :
  - Jika teks yang mendefinisikan tipe mengandung referensi terhadap diri sendiri, maka disebut tipe rekursif.
  - Tipe dibentuk dengan komponen yang merupakan tipe itu sendiri.

(diktat fungsional hal 54-55)

Contoh Type Rekursif: Bilangan Integer

- bilangan integer

  Basis : 0 adalah bilangan integer

  Rekurens: if x adalah bilangan integer

  then x+1 adalah bilangan integer

- bilangan integer ganjil

  Basis : 1 adalah bilangan integer ganjil

  Rekurens: if x adalah bilangan integer ganjil

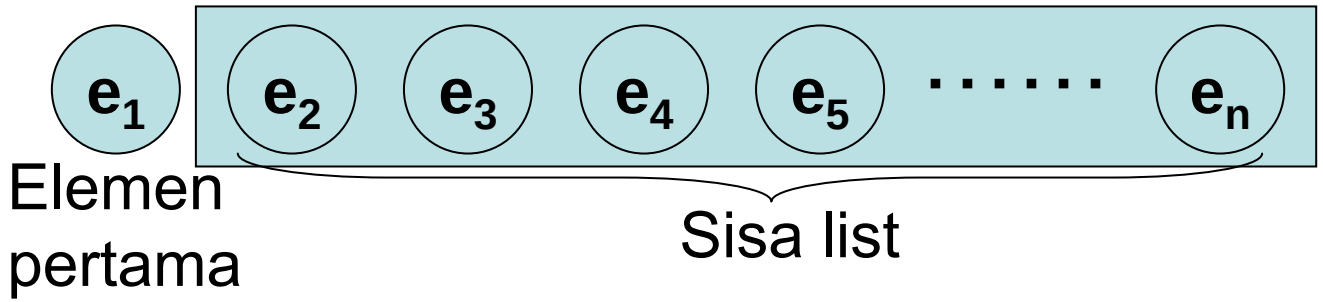
  then x + 2 adalah bilangan integer ganjil

Contoh type rekursif

- List
  - List kosong adalah list
  - List tidak kosong
    - Elemen
    - Sisanya adalah list
- Pohon
  - Pohon biner kosong adalah Pohon biner
  - Pohon biner tidak kosong
    - Akar
    - SubPohon kiri adalah pohon biner
    - SubPohon kanan adalah pohon biner

List

Definisi List



- List adalah sekumpulan elemen yg bertipe sama; disebut juga sequence atau series
- Tipe rekursif
  - Basis 0: list kosong adalah sebuah list
  - Rekurens: list terdiri dari sebuah elemen dan sublist (sublist juga bertipe list)

LIST dlm Kehidupan Sehari-hari

- Dalam kehidupan sehari-hari, list merepresentasi:
  - Teks (list of kata)
  - Kata (list of huruf)
  - Sequential file (list of record)
  - Table (list of elemen tabel, cth utk tabel integer: list of integer)
  - List of atom simbolik (dalam LISP)

LIST dlm Dunia Pemrograman

- Dalam dunia pemrograman
  - Antarmuka basis grafis (GUI): list of windows, list of menu items, list of buttons, list of icons
  - Program editor gambar: list of figures
  - Program pengelola sarana presentasi: list of slides
  - Program pengelola spreadsheet: list of worksheets, list of cells
  - Sistem operasi: list of terminals, list of jobs

JENIS LIST

- LIST dg elemen sederhana
  - LIST dg elemen bilangan integer
  - LIST dg elemen karakter (teks)
  - LIST dg elemen type bentukan, cth: list of point
- LIST dg elemen list (disebut list of list)

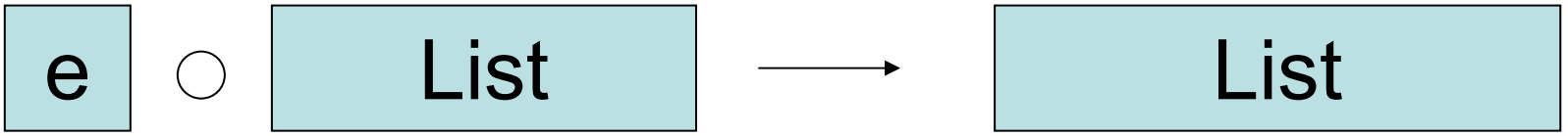
LIST dg ELEMEN SEDERHANA

Definisi rekursif

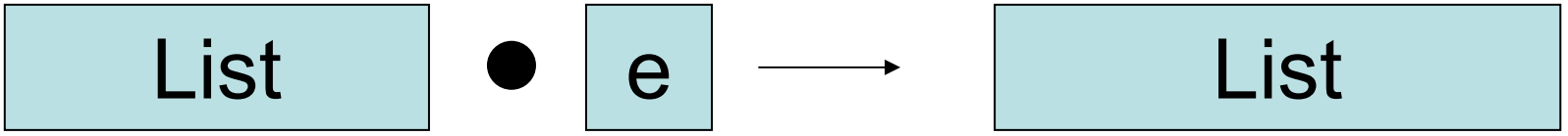
- Basis 0: list kosong adalah sebuah list
- Rekurens: list dapat dibentuk dengan menambahkan elemen pada list (konstruktor), atau terdiri dari sebuah elemen dan sisanya adalah list (selektor)
  - Elemen list: dapat berupa type dasar (integer, character, dll) dan type bentukan (Point, Jam, dll)

DEFINISI & SPESIFIKASI LIST

- Type List: [ ] atau [e o List]

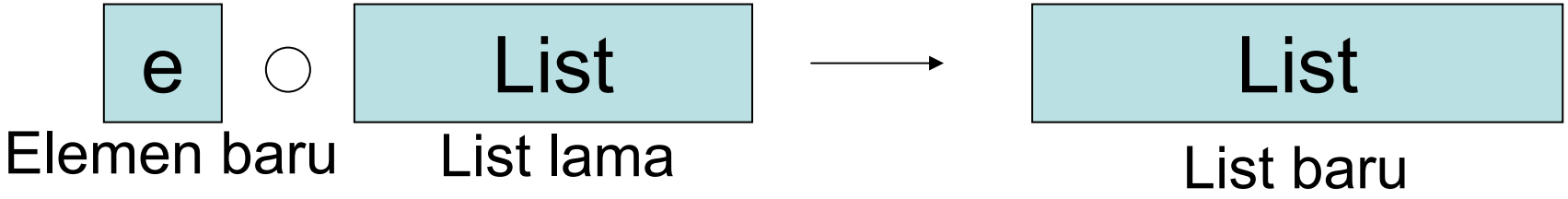


- Type List: [ ] atau [List • e]



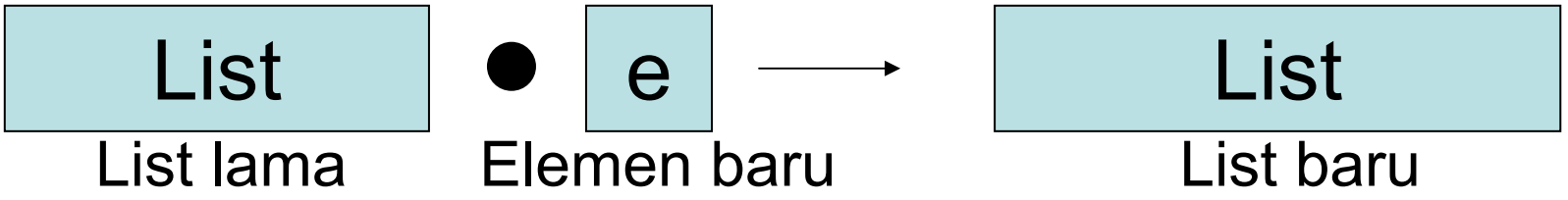
KONSTRUKTOR

- Konso : elemen, List → List



Konso (5,[1,3,6,7]) → [5,1,3,6,7]

- Kons• : List, elemen → List



Kons• ([1,3,6,7],5) → [1,3,6,7,5]

SELEKTOR

- FirstElmt: List tidak kosong → elemen

FirstElmt([5,1,3,6,7]) = 5

- Tail: List tidak kosong → list

Tail([5,1,3,6,7]) = [1,3,6,7]

SELEKTOR

- LastElmt: List tidak kosong → elemen

LastElmt([5,1,3,6,7]) = 7

- Head: List tidak kosong → list

Head([5,1,3,6,7]) = [5,1,3,6]

PREDIKAT DASAR

- {Basis 0}
  IsEmpty: List → boolean
  {benar jika list kosong [ ] }

- {Basis 1}
  IsOneElmt: List → boolean
  {benar jika list hanya berisi 1 element [e]}

Menghitung banyaknya elemen (NbElmt(L), hal 68)

- NbElmt: List → integer
  - Catatan: dengan Konso (FirstElmt dan Tail)
- Cth: NbElmt([ ]) = 0; NbElmt([a, b, c]) = 3
- Rekursif
  - Basis 0: list kosong, NbElmt = 0
  - Rekurens: NbElmt = 1 + NbElmt(Tail(L))
- Realisasi

```
NbElmt(L):
        if IsEmpty(L) then {Basis 0}
                0
        else {rekurens}
                1 + NbElmt(Tail(L))
```

Mengecek keanggotaan sebuah elemen (IsMember(x,L)), hal 68-69

- Fungsi IsMember(x,L): mengecek apakah x adalah member dari L
- IsMember: elemen, List → boolean

  {Benar jika x adalah elemen dari L}
- IsMember(x,[ ])=false; IsMember(x,[a,b,c])=false; IsMember(b,[a,b,c])=true

IsMember(x,L)

- Rekursif
  - Basis: jika list kosong maka nilai keluaran (output) adalah false {basis 0}
  - Rekurens
    - Jika nilai elemen pertama (atau terakhir) dari list adalah x, maka output adalah true. Tapi jika bukan x, maka tail (atau head) harus dicek.

Realisasi IsMember(x,L)

Dgn Konso

```
IsMember(x,L):
    if IsEmpty(L) then {Basis 0}
        false
    else {rekurens}
        if FirstElmt(L)=x then
            true
        else
            IsMember(x,Tail(L))
```

Dgn Kons•

```
IsMember(x,L):
    if IsEmpty(L) then {Basis 0}
        false
    else {rekurens}
        if LastElmt(L)=x then
            true
        else
            IsMember(x,Head(L))
```

Menyalin (copy) list, hal 69

- Proses mengambil satu persatu elemen dari List sumber dan membuat elemen baru untuk List target hasil copy
- Copy: List → List
- Cth: Copy([ ]) = [ ]; Copy([a,b,c]) = [a,b,c]
- Rekursif
  - Basis: IsEmpty(L) → memberi list kosong [ ]
  - Rekurens: mengambil elemen pertama list sumber kemudian memasukkannya sebagai elemen pertama list target ATAU mengambil elemen terakhir dari list sumber kemudian memasukkannya sebagai elemen terakhir list target

```
Copy(L): if IsEmpty(L) then {Basis 0}
                [ ]
         else {rekurens}
              Kons•  (Copy(Head(L)),LastElmt(L))
```

Mengecek apakah 2 list sama atau tidak, IsEqual, hal 70

- IsEqual: 2 List → boolean
- Cth:
  - IsEqual([ ],[ ])=true
  - IsEqual([ ],[a])=false
  - IsEqual([a],[ ])=false
  - IsEqual([a,b,c],[a,b,c])=true
- Rekursif
  - Basis: Jika dua2nya kosong maka true, jika hanya salah satu kosong maka false
  - Rekurens: Cek elemen pertama dari kedua list dan kemudian cek tail kedua list tersebut ATAU cek elemen terakhir dari kedua list dan kemudian cek head kedua list

IsEqual(L1,L2)

IsEqual(L1,L2):
depend on L1,L2
IsEmpty(L1) and isEmpty(L2): true {basis}
IsEmpty(L1) and not isEmpty(L2): false {basis}
not IsEmpty(L1) and isEmpty(L2): false {basis}
not IsEmpty(L1) and not isEmpty(L2): {rekurens}
(
(FirstElmt(L1) = FirstElmt(L2))
and then
IsEqual (Tail(L1),Tail(L2))
)

if (FirstElmt(L1) = FirstElmt(L2)) then
IsEqual (Tail(L1),Tail(L2))
else false

Menggabung (konkatenasi) 2 List, hal 72

- Konkat: 2 List → List
- Cth: Konkat([ ],[ ])=[ ]; Konkat([a],[b,c])=[a,b,c]
- Rekursif terhadap L1
  - Basis 0: L1 adl. list kosong, maka output adl. L2
  - Rekurens: mengambil elemen pertama dari L1 dan menggabungkannya dengan concat terhadap Tail(L1) dan L2.

```
Konkat(L1,L2):
    if  IsEmpty(L1) then {basis}
        L2
    else {rekurens}
        Konso(FirstElmt(L1), Konkat(Tail(L1),L2))
```

Ambil elemen ke-N, hal 71

- ElmtKeN: integer >= 1, List tdk kosong → elemen
- Cth:
  - ElmtKeN(0,[ ]) → tidak terdefinisi, karena dalam spek, list input harus tidak kosong
  - ElmtKeN(1,[a,b,c]) = a
- Rekurens dilakukan terhadap N (dikurangi 1, fungsi *prec*) dan List (diambil tail-nya)

```
ElmtKeN(N,L):
    if N=1 then      {basis 1}
        FirstElmt(L)
    else             {rekurens}
        ElmtKeN( prec(N), Tail(L) )
```

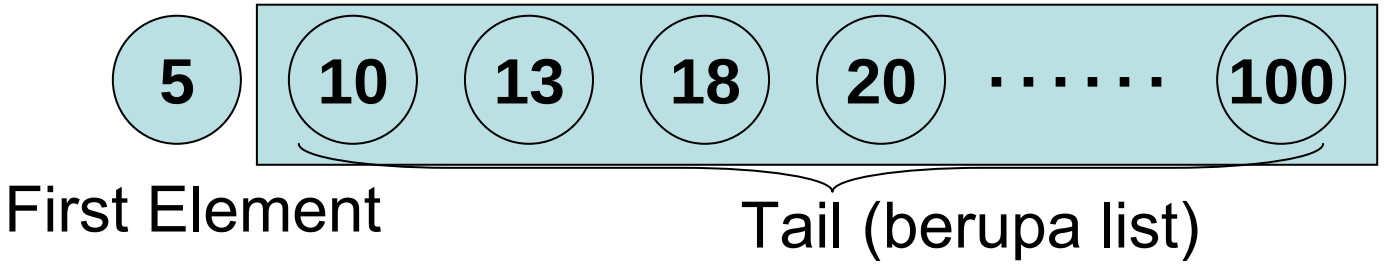
Apakah X adalah elemen ke N?, hal 73-74

- IsXElmtkeN: elemen, integer, list → boolean
- Cara 1 (Rekursif):
  - Basis 0: N=1, dan kemudian mengecek apakah FirstElmt(L)=X
  - Rekurens: N dikurangi 1, X tetap dan L diambil Tail-nya
- Cara 2:
  - Menggunakan fungsi antara ElmtKeN(N,L), mengecek apakah ElmtKeN(N,L) = X ?

LIST of INTEGER

Definisi Rekursif

- Basis 0: List kosong
- Rekurens: list bilangan integer dibuat dengan cara menambahkan sebuah integer pada list bilangan integer



List of Integer

- List of Integer adalah list yang elemennya berupa integer

Element	Integer
List of elemen	List of integer

- Contoh
  - Konso(elemen,list)→list *{untuk list of elemen}*
  - Konso(integer,list of integer)→list of integer *{untuk list of integer}*

Nilai Maksimum (Hal 80)

- Fungsi menghasilkan elemen bernilai maksimum dari list bilangan integer
- maxlist: List of integer tidak kosong→ integer
- Rekursif
  - Basis (Basis 1): jika elemen list berjumlah satu maka ambil nilai terakhir dari list. Basis 1 digunakan karena jika list kosong maka nilai maksimum tidak terdefinisi
  - Rekurens: membandingkan nilai elemen terakhir list dengan nilai maksimum dari head list

```
maxlist(Li):
        if IsOneElmt(Li) then {Basis 1}
                LastElmt(Li)
        else {rekurens}
                max2(LastElmt(Li),maxlist(Head(Li)))
```

Penjumlahan Dua List Integer (Hal 82)

- Fungsinya menjumlahkan dua list integer yang hasilnya disimpan dalam satu list integer. Asumsi: kedua list input memiliki dimensi (jumlah elemen) yang sama.
- Listplus: 2 List of integer ≥ 0 → List of integer ≥ 0
- Rekursif
  - Basis (Basis 0): Jika list kosong maka list output adalah [ ]
  - Rekurens: mengambil elemen pertama dari kedua list, menjumlahkan kedua elemen tadi kemudian memasukkannya sebagai elemen pertama dari list output

```
Listplus(Li1,Li2):
        if IsEmpty(Li) then {Basis 0}
                [ ]
        else {rekurens}
                Konso (FirstElmt(Li1)+FirstElmt(Li2),
                        Listplus(Tail(Li1),Tail(Li2)))
```

Kemunculan Nilai Maks (Hal 84)

- Fungsi menghasilkan nilai maksimum dan jumlah kemunculan nilai maksimum tsb pd list bilangan integer
- maxNb: List of integer → <integer,integer>
- Cth: maxNb([11,3,4,5,11,6,11])= <11,3>
- Rekursif
  - Basis (Basis 1): List dgn satu elemen e menghasilkan <e,1>
  - Rekurens:

Nilai maks: m; Jumlah kemunculan: n

Jika m adalah nilai maksimum dari Tail(Li)
dan n adalah jumlah kemunculan m pada Tail(Li)
maka ketika memeriksa e, ada 3 kemungkinan:
m < e: nilai maksimum diganti yang baru (m ← e), n=1
m = e: nilai maksimum tetap m, nilai kemunculan n ditambah 1
m > e: nilai maksimum tetap m, nilai kemunculan tetap n

Kemunculan Nilai Maks (Hal 84)

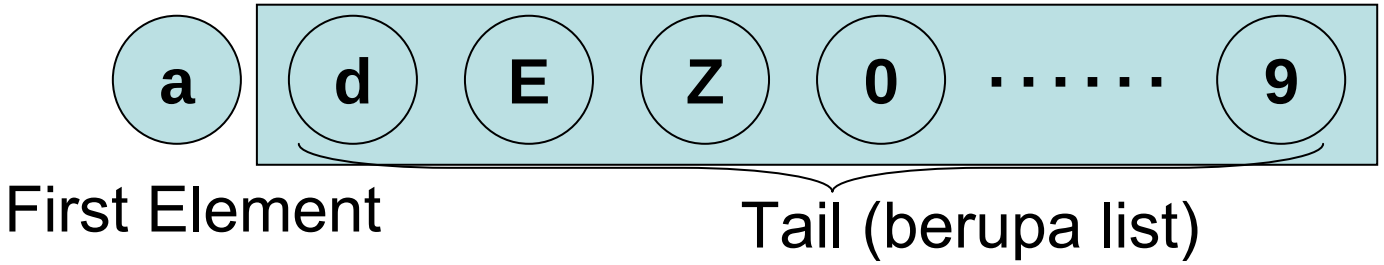
MaxNb(Li): {menghasilkan nilai maks dan kemunculannya}

```
if IsOneElmt(Li) then {basis 1}
   <FirstElmt(Li), 1>
else {rekurens}
   let <m,n> = MaxNb(Tail(Li))
   in     depend on m, FirstElmt(Li)
             m < FirstElmt(Li): <FirstElmt(Li), 1>
             m = FirstElmt(Li): <m,1+n>
             m > FirstElmt(Li): <m,n>
```

LIST of CHARACTER (TEKS)

Definisi Rekursif

- Basis 0: teks kosong adalah teks
- Rekurens: teks dapat dibuat dengan cara menambahkan sebuah karakter pada teks



List of Character

- List of Character adalah list yang elemennya berupa character

Element	Character
List of elemen	Text *{list of character}*

- Contoh
  - Konso(elemen,list)→list *{untuk list of elemen}*
  - Konso(character,text)→text *{untuk list of character}*

Hitung A, hal 76

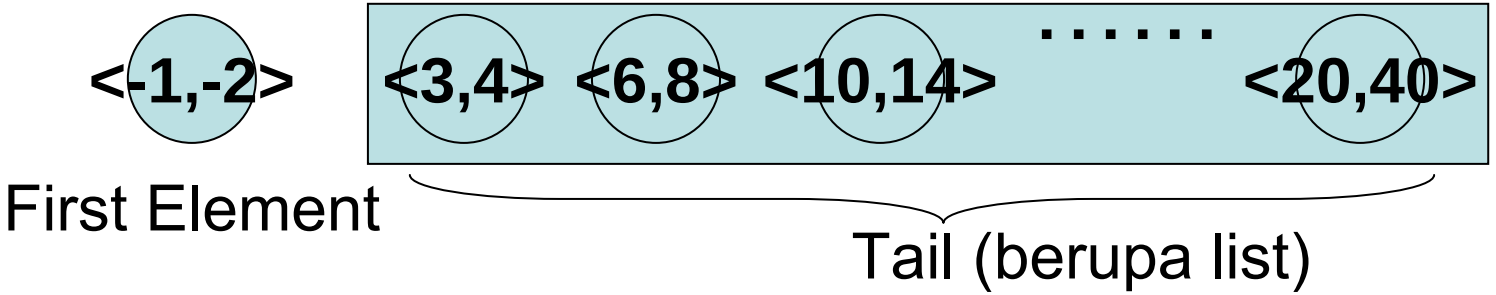
- Nba: Teks → integer >=0
- Cth: Nba([‘b’,‘c’,‘a’,‘d’,‘a’,‘n’,‘a’]) = 3
- Rekursif
  - Basis {basis 0}: teks kosong, output: 0
  - Rekurens: Periksa huruf pertama dari teks, jika ‘a’ maka 1 ditambahkan dengan nilai kemunculan ‘a’ pada tail, jika bukan ‘a’ maka 0 ditambahkan dengan nilai kemunculan ‘a’ pada tail

```
Nba(T): if IsEmpty(T) then {Basis 0}
            0
        else {rekurens}
            (if (FirstElmt(T) = ‘a’) then 1 else 0)
             + Nba(Tail(T))
```

List of type bentukan

Definisi Rekursif

- Basis 0: List kosong
- Rekurens: list of type bentukan dibuat dengan cara menambahkan sebuah elemen bertype bentukan pada list of type bentukan
- Contoh: list of point



List of Type Bentukan

- List of type bentukan adalah list yang elemennya berupa type bentukan

Element	Type bentukan
List of elemen	List of type bentukan

- Contoh
  - Konso(elemen,list)→list *{untuk list of elemen}*
  - Konso(type bentukan,list of type bentukan)→list of type bentukan *{untuk list of type bentukan}*
  - Konso(point,list of point)→list of point *{untuk list of point}*

Pada dasarnya semua jenis list dikelola dengan cara yang sama

- Menghitung kemunculan sebuah elemen pada list
  - List of integer
    - NbXInt: integer, List of integer → integer
  - List of character
    - NbC: character, Teks → integer (diktat hlm.77)
  - List of Point
    - NbPoint: Point, List of Point → integer

Menghitung Kemunculan X

- Rekurens
  - Basis: untuk list kosong, kemunculan adalah 0
  - Rekurens: dicek apakah elemen pertama adalah X, jika iya maka nilai 1 ditambahkan pada hasil penghitungan kemunculan X pada tail dari list, jika tidak maka nilai 0 yang ditambahkan

```
NbX(X,L): if IsEmpty(L) then {Basis 0}
             0
          else {rekurens}
             (if (FirstElmt(L) = X) then 1 else 0)
             + NbX(Tail(L))
```

List of Elemen

```
NbX(X,L):
if IsEmpty(L) then {Basis 0}
   0
else {rekurens}
   (if (FirstElmt(L) = X) then 1 else 0)
   + NbX(X,Tail(L))
```

List of Character

```
NbC(C,T):
if IsEmpty(T) then {Basis 0}
   0
else {rekurens}
  (if (FirstChar(T) = C) then 1 else 0)
   + NbC(C,Tail(T))
```

List of Integer

```
NbXInt(Xi,Li):
if IsEmpty(Li) then {Basis 0}
   0
else {rekurens}
   (if (FirstElmt(Li) = Xi) then 1 else 0)
   + NbXi(Xi,Tail(Li))
```

List of Point

```
NbPoint(P,Lp):
if IsEmpty(Lp) then {Basis 0}
   0
else {rekurens}
   (if (FirstElmt(Lp) = P) then 1 else 0)
   + NbPoint(P,Tail(Lp))
```

TRANSLASI KE LISP

List dalam LISP

- List adalah tipe dasar yang disediakan LISP
- Fungsi manipulasi list pada LISP:
  - Konstruktor: list, cons, append
  - Selektor: car, cdr
  - Predikat: atom, listp, null, equal

Konstruktor

Konso

```
(defun Konso (e L)
        (cons e L))
```

Kons•

```
(defun Kons• (e L)
        (Inverse (cons e (Inverse L))))
```

Selektor

```
(defun FirstElmt (L)
    (car L))

(defun Tail (L)
    (cdr L))
```

```
(defun LastElmt (L)
    (car (Inverse L)))

(defun Head (L)
    (Inverse (cdr (Inverse L)))
)
```

Predikat

```
(defun IsEmpty(L)
    (null L))

(defun IsOneElmt(L)
    (and (not (IsEmpty L)) (IsEmpty (cdr L))))

(defun IsMember (X L)
    (if (IsEmpty L)  nil                        ; basis 0
        (if (equal X (FirstElmt L)) t           ; recc
            (IsMember X (Tail L))
        )
     )
)
```

Fungsi Lain


```
(defun NbElmt(L)
    (length L))

(defun Konkat(L1 L2)
    (append L1 L2))

(defun Inverse (L)
    (reverse L))

(defun Max(L)
    (if (IsOneElmt L) (FirstElmt L)              ;basis 1
        (max2 (FirstElmt L) (Max (Tail L)))   ;recc
    )
)
```

Pekerjaan Rumah

- Translasikan dalam bentuk LISP, fungsi-fungsi yang telah dijelaskan dalam kuliah

Selamat Belajar