IF1210 Dasar Pemrograman

List sebagai Struktur Data Rekursif

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Type

- Type adalah **himpunan nilai** dan sekumpulan **operator** yang terdefinisi terhadap type tersebut
  - Dalam konteks fungsional: operator dijabarkan dalam bentuk **fungsi**
- Jenis-jenis type:
  - Type dasar → sudah tersedia: integer, real, character, boolean
  - Type bentukan → dibuat sendiri
- Jenis type dari segi banyaknya elemen yang harus dikelola:
  - Type yang mengelola satu objek, contoh: integer, real, point, jam, dll.
  - Type koleksi/kumpulan objek, contoh: array of integer, list of character, dll.

Mendefinisikan Type

- Dalam konteks fungsional, mendefinisikan type adalah mendefinisikan:
  - **Nama dan struktur** type (komponen-komponennya)
  - **Selektor** untuk mengakses komponen-komponen type
  - **Konstruktor** untuk “membentuk” type
  - **Predikat** untuk menentukan karakteristik dan pemeriksaan besaran
  - **Fungsi-fungsi lain** yang didefinisikan untuk type tersebut

Tipe Rekursif

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

Contoh Tipe Rekursif: Bilangan Integer

bilangan integer

Basis : 0 adalah bilangan integer

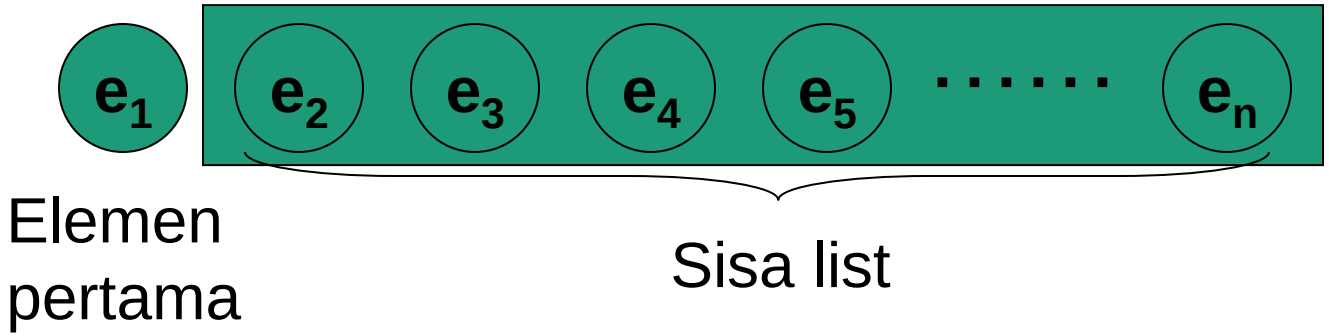
Rekurens : if x adalah bilangan integer
then x+1 adalah bilangan integer

bilangan integer ganjil

Basis : 1 adalah bilangan integer ganjil

Rekurens : if x adalah bilangan integer ganjil
then x+2 adalah bilangan integer ganjil

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 dalam Kehidupan Sehari-hari

- Dalam kehidupan sehari-hari, list merepresentasikan:
  - 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 dalam Dunia Pemrograman

- Dalam dunia pemrograman
  - Antarmuka basis grafis (GUI): list of windows, list of menu items, list of button, list of icon
  - Program editor gambar: list of figures
  - Program pengelola sarana presentasi: list of slides
  - Program pengelola spreadsheet: list of worksheet, list of cell
  - Sistem operasi: list of terminal, list of job

Jenis List

- LIST dengan elemen sederhana
  - LIST dengan elemen bilangan integer
  - LIST dengan elemen karakter (teks)
  - LIST dengan elemen type bentukan,
    - Contoh: list of point (tidak dibahas di kuliah ini)
- LIST dengan elemen list (disebut list of list)
  - (tidak dibahas di kuliah ini)

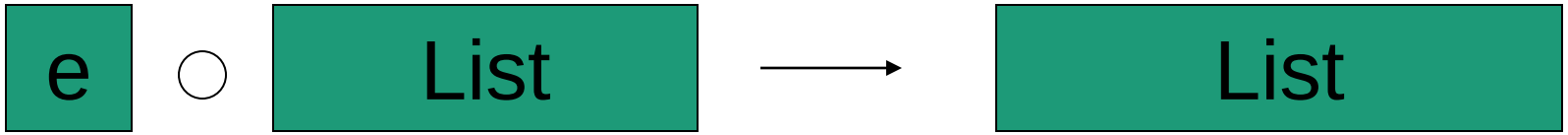
List dengan Elemen Sederhana

Definisi rekursif:

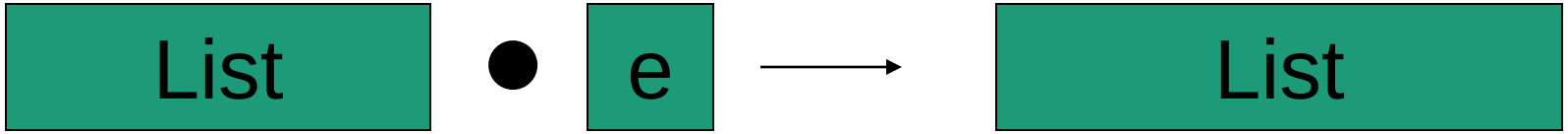
- **Basis**: list kosong adalah sebuah list
- **Rekurens**: list dapat dibentuk dengan menambahkan elemen pada list (**konstruktor**), atau terdiri dari sebuah elemen dan sisanya adalah list (**selektor**)
  - Elemen list: dapat berupa type dasar (integer, character, dll) dan type bentukan (Point, Jam,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

DEFINISI & SPESIFIKASI LIST

- **type** List: [ ] atau [e o List]

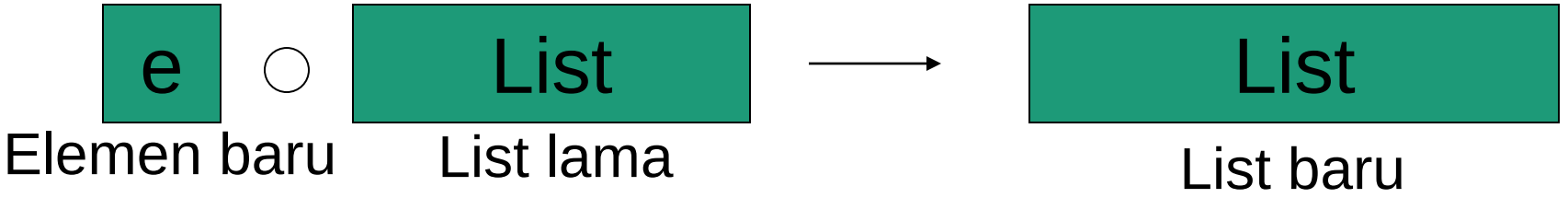


- **type** List: [ ] atau [List • e]



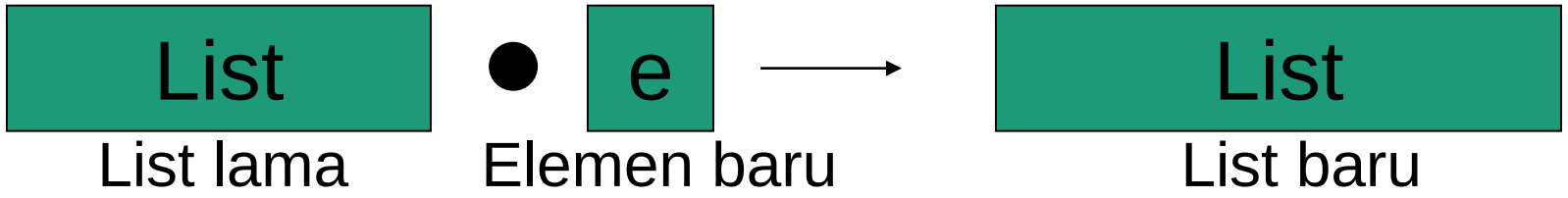
KONSTRUKTOR

- **Konso** : elemen, List → List



Konso (5,[1,3,6,7]) → [5,1,3,6,7]

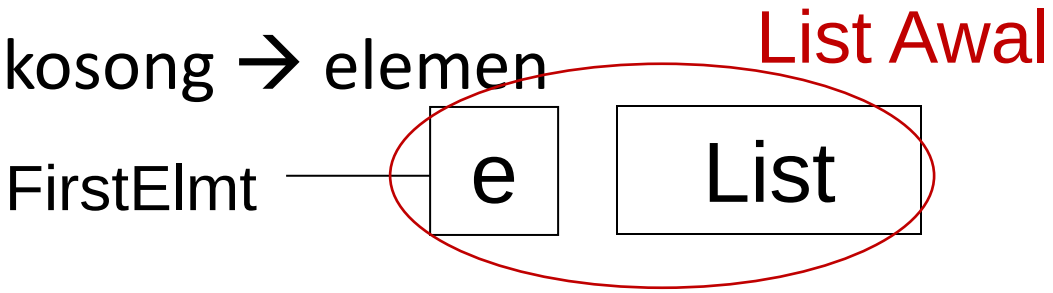
- **Kons•** (dibaca KonsDot): List, elemen → List



Kons• ([1,3,6,7],5) → [1,3,6,7,5]

SELEKTOR

- **FirstElmt**: List tidak kosong → elemen



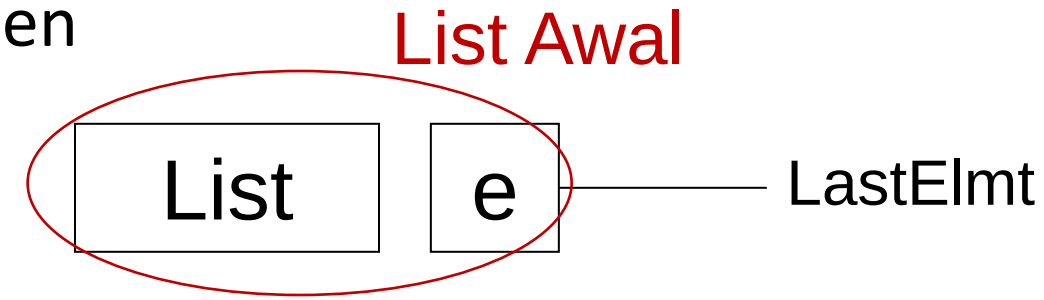
FirstElmt([5,1,3,6,7]) = 5

- **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

- **Pemeriksaan Basis-0**

  IsEmpty: List → boolean

  { IsEmpty(L) menghasilkan true jika list L kosong }

- **Pemeriksaan Basis-1**

  IsOneElmt: List → boolean

  { IsOneElmt(L) menghasilkan true jika list L hanya berisi 1 elemen }

List Pada Haskell

- Contoh list di Haskell
  - List of integer `[Int]`
    `[1,2,3,2,4,1]`
  - List of Char `[Char]`
    `['a','e','2']`
  - List of Point `[(Int,Int)]`
    `[(1,2),(0,0),(-2,9)]`

Deklarasi List of <type_elemen>

- Harus dituliskan sebagai komentar, ganti <type_elemen> dengan type elemen list (integer, character, dll.)

```
-- DEFINISI DAN SPESIFIKASI LIST

-- type List of <type_elemen>: [ ] atau [e o List]
-- Definisi type List of <type_elemen>
-- Basis: List of <type_elemen> kosong adalah list of <type_elemen>
-- Rekurens: List tidak kosong dibuat dengan menambahkan sebuah
-- elemen bertype <type_elemen> di awal sebuah list

-- type List of <type_elemen>: [ ] atau [List • e]
-- Definisi type List of <type_elemen>:
-- Basis: List of <type_elemen> kosong adalah list of <type_elemen>
-- Rekurens: List tidak kosong dibuat dengan menambahkan sebuah
-- elemen bertype <type_elemen> di akhir sebuah list
```

Konstruktor List

```
-- DEFINISI DAN SPESIFIKASI KONSTRUKTOR

konso :: <type_elemen> -> [<type_elemen>] -> [<type_elemen>]
-- konso(e,l) menghasilkan sebuah list dari e (sebuah
-- elemen) dan l (list of elemen),
-- dengan e sebagai elemen pertama: e o l -> l'
-- REALISASI
konso e l = [e] ++ l


konsDot :: [<type_elemen>] -> <type_elemen> -> [<type_elemen>]
-- konsDot(l,e) menghasilkan sebuah list dari l (list of
-- elemen) dan e (sebuah elemen),
-- dengan e sebagai elemen terakhir: l • e -> l'
-- REALISASI
konsDot l e = l ++ [e]
```

Konstruktor List of Integer

```
-- DEFINISI DAN SPESIFIKASI KONSTRUKTOR

konso :: Int -> [Int] -> [Int]
-- konso(e,li) menghasilkan sebuah list dari e (sebuah
-- integer) dan li (list of integer),
-- dengan e sebagai elemen pertama: e o li -> li'
-- REALISASI
konso e li = [e] ++ li


konsDot :: [Int] -> Int -> [Int]
-- konsDot(li,e) menghasilkan sebuah list dari li (list
-- of integer) dan e (sebuah integer),
-- dengan e sebagai elemen terakhir: li • e -> li'
-- REALISASI
konsDot li e = li ++ [e]
```

Konstruktor List of Character (Teks)

```
-- DEFINISI DAN SPESIFIKASI KONSTRUKTOR

konso :: Char -> [Char] -> [Char]
-- konso(e,lc) menghasilkan sebuah teks dari e
-- (sebuah character) dan lc (teks), dengan e sebagai
-- elemen pertama: e o L -> L'
-- REALISASI
konso e lc = [e] ++ lc

konsDot :: [Char] -> Char -> [Char]
-- konsDot(lc,e) menghasilkan sebuah teks dari lc
-- (teks) dan e (sebuah character), dengan e sebagai
-- elemen terakhir: lc • e -> lc'
-- REALISASI
konsDot lc e = lc ++ [e]
```

Selektor List

- Selektor List hanya terdefinisi pada List yang tidak kosong.

Notasi Fungsional	Notasi Haskell
FirstElmt(L)	head l
Tail(L)	tail l
LastElmt(L)	last l
Head(L)	init l

Karena *tail* dan *head* adalah fungsi yang sudah terdefinisi pada Haskell, maka untuk selanjutnya Selektor List yang digunakan sesuai Notasi Haskell

Selektor List of <type_elemen>

- Definisi dan spesifikasi selektor ditulis dalam bentuk komentar, menggunakan nama-nama fungsi manipulasi list Haskell

```
-- DEFINISI DAN SPESIFIKASI SELEKTOR

-- head : [<type_elemen>] -> <type_elemen>
-- head(l) menghasilkan elemen pertama list l, l tidak kosong

-- tail : [<type_elemen>] -> [<type_elemen>]
-- tail(l) menghasilkan list tanpa elemen pertama list l,
-- l tidak kosong

-- last : [<type_elemen>] -> <type_elemen>
-- last(l) menghasilkan elemen terakhir list l, l tidak kosong

-- init : [<type_elemen>] -> [<type_elemen>]
-- init(l) menghasilkan list tanpa elemen terakhir list l,
-- l tidak kosong
```

Contoh Pemanfaatan Selektor

```
> head [1,2,3,4]      -- FirstElmt
1
> tail [1,2,3,4]      -- Tail
[2,3,4]
> last [1,2,3,4]      -- LastElmt
4
> init [1,2,3,4]      -- Head
[1,2,3]
```

Predikat List

```
-- DEFINISI DAN SPESIFIKASI PREDIKAT

isEmpty :: [<type_elemen>] -> Bool
-- isEmpty(l) true jika list of elemen l kosong
-- REALISASI
isEmpty l = null l

isOneElmt :: [<type_elemen>] -> Bool
-- isOneElmt(l) true jika list of integer l hanya
-- mempunyai satu elemen
-- REALISASI
isOneElmt l = (length l) == 1
```

```
> isEmpty [1,2,3,4]
False
> isEmpty []
True
> isOneElmt []
False
> isOneElmt [1]
True
```

Predikat List of Integer

```
-- DEFINISI DAN SPESIFIKASI PREDIKAT

isEmpty :: [Int] -> Bool
-- isEmpty(li) true jika list of integer li kosong
-- REALISASI
isEmpty li = null li

isOneElmt :: [Int] -> Bool
-- isOneElmt(li) true jika list of integer li hanya
-- mempunyai satu elemen
-- REALISASI
isOneElmt li = (length li) == 1
```

Predikat List of Character (Teks)

```
-- DEFINISI DAN SPESIFIKASI PREDIKAT

isEmpty :: [Char] -> Bool
-- isEmpty(lc) true jika list of integer lc kosong
-- REALISASI
isEmpty lc = null lc

isOneElmt :: [Char] -> Bool
-- isOneElmt(lc) true jika list of integer lc hanya
-- mempunyai satu elemen
-- REALISASI
isOneElmt lc = (length lc) == 1
```

Bayangkan kalian duduk berderet di ruang kelas di kampus ITB…

Latihan

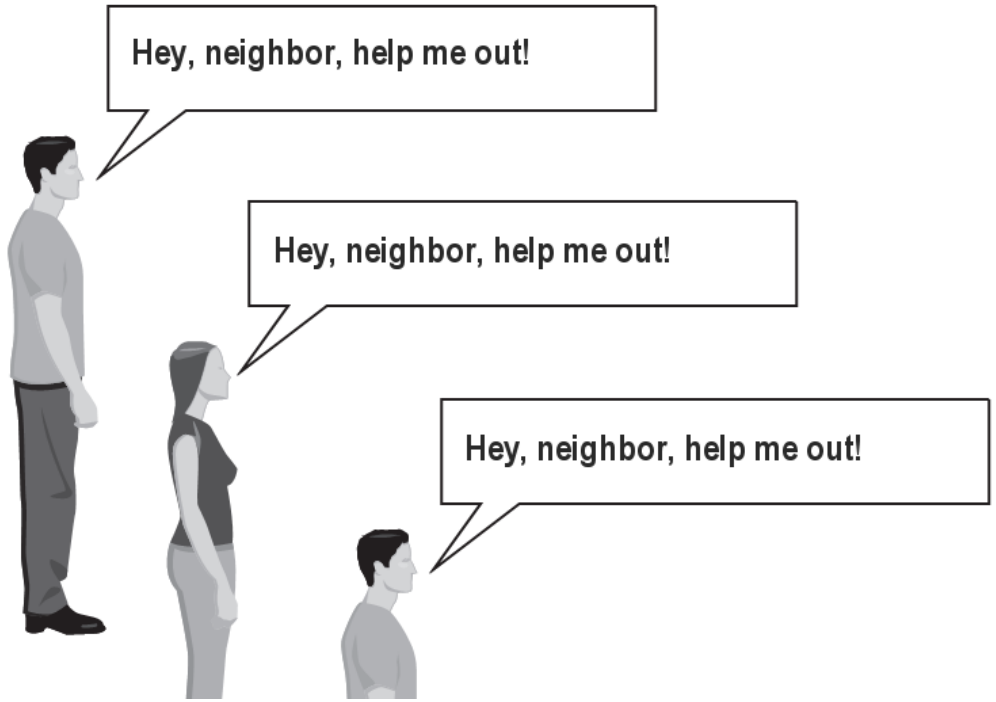
(Bagi mahasiswa yang duduk paling kiri di setiap baris)
Ada berapa mahasiswa yang duduk di baris tersebut?

- Pandanganmu terbatas, sehingga kamu hanya dapat melihat teman yang ada di sebelahmu.
- Tapi kamu boleh bertanya pada teman yang duduk di sampingmu.

Bagaimana cara memecahkan persoalan ini?

Ide penyelesaian

- Setiap orang dapat berpartisipasi untuk menyelesaikan persoalan ini.
  - Bagaimana "versi kecil" dari persoalan yang dapat dibantu penyelesaiannya?
  - Informasi apa dari teman sebelah yang dapat membantu saya?



Recursion is all about breaking a big problem into smaller occurrences of that same problem.

Solusi Rekursif dari Persoalan

Jumlah orang yang ada di baris ini, mulai dari saya hingga orang paling kanan

- Jika ada teman di kanan saya, tanyakan padanya ada berapa orang di kanan saya yang duduk pada baris yang sama
  → Jika dia menjawab ada **N** orang, maka saya akan menjawab ada **N + 1** orang.
- Jika tidak ada teman di kanan saya, maka saya akan menjawab **1** orang.

Menghitung banyaknya elemen (nbElmt)

nbElmt:: [<type_elemen>] -> Int

- Memanfaatkan definisi: list terdiri atas FirstElmt + Tail
- Cth: nbElmt([ ]) = 0; nbElmt([1, 2, 3]) = 3
- Rekursif
  - Basis 0: list kosong, nbElmt = 0
  - Rekurens: nbElmt(L) = 1 + nbElmt(Tail(L))
- Realisasi

```
-- DEFINISI DAN SPESIFIKASI

nbElmt :: [<type_elemen>] -> Int
-- NbElmt(l) menghasilkan banyaknya elemen list, nol
-- jika list kosong

-- REALISASI
nbElmt l = if (isEmpty l) then 0          -- Basis
           else 1 + (nbElmt (tail l)) -- Rekurens
```

Nantikan operasi-operasi pada list pada materi berikutnya…

Sumber

- Diktat “Dasar Pemrograman, Bag. Pemrograman Fungsional” oleh Inggriani Liem, revisi Februari 2014