Fungsi Rekursif dan List

Tim Pengajar
IF1210 Dasar Pemrograman
Updated: 23 Februari 2015

Keterangan

- Seluruh kode ditulis pada file **<nama-file>.hs**
- Gunakan **:load <nama-file>** untuk me-load script
- Setelah di-load, fungsi pada script dapat diaplikasikan

MORE ON FUNGSI REKURSIF

Contoh-1: Factorial

```
-- Definisi dan Spesifikasi

fac0 :: Int -> Int

-- fac0(n) = n! sesuai dengan definisi rekursif factorial

-- Realisasi

fac0 x = if (x==0) then 1      -- basis 0

         else fac0(x-1)*x      -- rekurens
```

Contoh-2: GCD

```
-- Definisi dan spesifikasi

fpb :: Int -> Int -> Int

-- fpb(m,n) menghasilkan bilangan integer terbesar

-- yang dapat membagi m dan n tanpa sisa

-- Realisasi

fpb m n =  if ((mod m n)==0) then n     -- basis

           else fpb(n,(mod m n))        -- rekurens
```

Contoh-3a: Fibonacci

```
-- DEFINISI DAN SPESIFIKASI
fib :: Int -> Int
-- Definisi rekursif fungsi fibonacci:
-- fib (n) = sesuai dengan definisi deret fibonacci:
--       n = 0 : fib(0) = 0
--       n = 1 : fib(1) = 1
--       n > 1 : fib(n) = fib(n-1) + fib(n-2)

-- REALISASI(versi 1)
fib n = if n == 0 then 0           -- basis 0
        else if n == 1 then 1      -- basis 1
        else fib(n-1) + fib(n-2)   -- rekurens
```

Contoh-3b: Fibonacci

```
-- DEFINISI DAN SPESIFIKASI
fib2 :: Int -> Int
-- Definisi rekursif fungsi fibonacci:
-- fib2(n) = sesuai dengan definisi deret fibonacci:
--       n = 0 : fib2(0) = 0
--       n = 1 : fib2(1) = 1
--       n > 1 : fib2(n) = fib2(n-1) + fib2(n-2)

-- REALISASI(versi 2)
fib2 n
   | n == 0 = 0
   | n == 1 = 1
   | otherwise = fib(n-1) + fib(n-2)
```

Latihan Soal (untuk dikerjakan mandiri)

Buatlah definisi, spesifikasi, dan realisasi dari fungsi-fungsi berikut (menggunakan pendekatan rekursif):

1. **DeretSegitiga**, merupakan fungsi untuk mencari nilai bilangan ke-n pada deret segitiga.
   Deret segitiga: 1, 3, 6, 10, 15, ...
2. **IsGanjil**, merupakan predikat untuk memeriksa apakah sebuah bilangan integer (≥0) merupakan bilangan ganjil.
3. **LuasBS**, merupakan fungsi untuk menghitung luas bujur sangkar dengan panjang sisi tertentu.

Latihan Soal (untuk dikerjakan mandiri)

4. **SumOfDigits**, menghitung hasil penjumlahan dari setiap bilangan tunggal yang terdapat di dalam sebuah bilangan integer positif.

   Misalnya:

   - SumOfDigits(234) = 2 + 3 + 4 = 9
   - SumOfDigits(38) = 3 + 8 = 11
   - SumOfDigits(5) = 5

   Apabila didefinisikan bahwa SumOfDigits dari bilangan negatif dilakukan dengan mengabaikan tanda '-', buatlah fungsi **SumOfDigitsPosNeg** yang menangani hal ini.

   Misalnya: SumOfDigitsPosNeg(-45) = 4 + 5 = 9

Latihan Soal (untuk dikerjakan mandiri)

5. **IsOdd?** dan **IsEven?** yang saling *mutually recursive*. IsOdd? digunakan untuk memeriksa apakah sebuah bilangan integer (>0) merupakan bilangan ganjil, sedangkan IsEven? sebaliknya.

   Hint: IsOdd?(n) akan memberikan hasil yang sama dengan IsEven?(n-1), IsOdd?(n-2), IsEven?(n-3), dst.

6. **GCD** dengan menggunakan **pendekatan kedua**. Fungsi Min2(a,b) yang menghasilkan nilai terkecil dari a dan b dapat langsung digunakan.

LIST OF ELEMEN SEDERHANA

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]
```

Predikat List of integer

```
-- DEFINISI DAN SPESIFIKASI PREDIKAT

isEmpty(l) :: [<type_elemen>] -> Bool
-- isEmpty(l) true jika list of integer 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
```

Predikat untuk list of <type_elemen>

- Gunakan pola pada realisasi predikat isEmpty dan isOneElmt pada type list of integer dengan cara mengganti integer menjadi <type_elemen>

Fungsi Lain terhadap List of <type_elemen>

- NbElmt
- IsMember?
- Copy
- IsEqual
- Konkat
- ElmtKeN
- isXElmtKeN

nbElmt - List of integer

```
-- DEFINISI DAN SPESIFIKASI

nbElmt :: [Int] -> 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
```

isMember – List of Integer

```
isMember1 :: Int -> [Int] -> Bool
-- isMember1(x,l) true jika x adalah elemen list l
-- REALISASI isMember1 menggunakan head dan tail
isMember1  x l  = if (isEmpty l) then False  -- Basis
                  else if (head l) == x then True
                       else (isMember1 x (tail l)) -- Rekurens

isMember2 :: Int -> [Int] -> Bool
-- isMember2(x,l) true jika x adalah elemen list l
-- REALISASI isMember2 menggunakan init dan last
isMember2  x l  = if (isEmpty l) then False  -- Basis
                  else if (last l) == x then True
                       else (isMember2 x (init l))  -- Rekurens
```

IsEqual – List of integer

```
-- DEFINISI DAN SPESIFIKASI
isEqual :: [Int] -> [Int] -> Bool
-- isEqual(l1,l2) true jika semua elemen list l1 sama
-- dengan l2: sama urutan dan sama nilainya

-- REALISASI
isEqual  l1 l2
   | (isEmpty l1) && (isEmpty l2) = True          -- Basis
   | (isEmpty l1) && not (isEmpty l2) = False   -- Basis
   | not (isEmpty l1) && (isEmpty l2) = False   -- Basis
   | not (isEmpty l1) && not (isEmpty l2) =   -- Rekurens
     ((head l1)==(head l2)) && (isEqual (tail l1) (tail l2))
```

Copy – List of integer

```
-- DEFINISI DAN SPESIFIKASI
copy :: [Int] -> [Int]
-- copy(l) menghasilkan list yang identik dengan list
-- asal

-- REALISASI (versi 1: menggunakan konso)
copy l = if (isEmpty l) then []                  -- Basis
         else (konso (head l) (copy (tail l)) -- Rekurens

-- REALISASI (versi 2: menggunakan konsDot)
copy l = if (isEmpty l) then []                      -- Basis
         else (konsDot (copy (init l)) (last l)) -- Rekurens
```

Konkat – List of integer

```
-- DEFINISI DAN SPESIFIKASI
konkat :: [Int] -> [Int] -> [Int]
-- konkat(L1,L2) menghasilkan konkatenasi 2 list, dengan
-- list l2 "sesudah" list l1

-- REALISASI
konkat l1 l2  = if (isEmpty l1) then l2  -- Basis
                else -- Rekurens
                   (konso (head l1) (konkat (tail l1) l2))
```

Latihan

- Lanjutkan untuk:
  - ElmtKeN
  - isXElmtKeN
  - listPlus
- Bagaimana untuk list of elemen bertype lain?

maxList – List of integer

```
max2 :: Int -> Int -> Int
-- max2(a,b) menghasilkan bilangan yang terbesar antara a
-- dan b
-- REALISASI
max2 a b = if a>=b then a else b

maxList :: [Int] -> Int
-- maxList(Li) : menghasilkan elemen Li yang bernilai
-- maksimum
-- REALISASI
maxList l = if (isOneElmt l) then head l          -- Basis
            else (max2 (head l) (maxList (tail l))) -- Recc
```

maxNb – List of integer

```
-- DEFINISI DAN SPESIFIKASI
maxNb :: [Int] -> (Int,Int)
-- maxNb(li) menghasilkan <nilai max, kemunculan nilai max>
-- dari suatu list of integer li: <m,n> dengan m adalah
nilai
-- maksimum di li dan n adalah jumlah kemunculan m dalam li
-- REALISASI
maxNb li = if (isOneElmt l) then (head l,1) -- Basis
           else -- Rekurens
              let (m,n) = (maxNb (tail l)) in
                 if (m < (head l)) then (head l,1)
                 else if (m>(head l)) then (m,n)
                      else (m,n+1)
```

nbA – List of character/Teks

```
-- DEFINISI DAN SPESIFIKASI
nbA :: [Char] -> Int
-- nbA(lc) menghasilkan banyaknya kemunculan huruf
-- 'A' di teks lc

-- REALISASI
nbA lc = if (isEmpty lc) then 0  -- Basis
         else -- Rekurens
            (nbA (tail lc)) + (if (head lc)=='A'
                               then 1 else 0)
```