Bahasa Pemrograman :: Functional Programming

Julio Adisantoso
ILKOM IPB

24 Pebruari 2011

Functional Programming

- Program serba fungsi, artinya setiap persoalan diselesaikan dengan menggunakan fungsi.
- Mulai dikembangkan tahun 1960an, dimotivasi oleh peneliti bidang *artificial intelligence*, *symbolic computation*, *theorem proving*, *rule-based system*, dan NLP.
- Bahasa fungsional pertama adalah Lisp (McCarthy, 1960).
- Memodelkan masalah komputasi sebagai suatu fungsi matematika, yang mempunyai input (**domain**) dan hasil atau output (**range**).

Function & Lambda Calculus

- Contoh fungsi matematika: $Kuadrat(n) = n * n$.
- **Kuadrat** adalah fungsi yang memetakan suatu himpunan bilangan real R (domain) ke bilangan real R (range). Atau secara formal didefinisikan:

$$Kuadrat : R \rightarrow R$$

- Suatu fungsi disebut total jika terdefinisi untuk semua elemen dalam domain, dan disebut parsial untuk selainnya.
- Fungsi Kuadrat adalah total karena domainnya seluruh himpunan bilangan real.

Lambda Calculus

- Setiap identifier adalah ekspresi lambda.
- Jika M dan N adalah ekspresi lambda, maka aplikasi dari M dan N, ditulis sebagai $(M\ N)$, adalah ekspresi lambda.
- Abstraksi, ditulis sebagai $(\lambda x.M)$, dimana x adalah identifier dan M adalah ekspresi lambda, juga merupakan ekspresi lambda.
- BNF grammar:

$$\begin{aligned} \mathit{LambdaExpression} &\rightarrow \mathit{variable} \mid (MN) \mid (\lambda \mathit{variable}.M) \\ M &\rightarrow \mathit{LambdaExpression} \\ N &\rightarrow \mathit{LambdaExpression} \end{aligned}$$

Contoh Ekspresi Lambda

- Contoh ekspresi lambda:

$$x$$
$$(\lambda x.x)$$
$$((\lambda x.x)(\lambda y.y))$$

- Aplikasi fungsi:

$$((\lambda x. * xx)5) = (*55)$$

Pemrograman List

- Lisp (List Programming) adalah bahasa pemrograman fungsional yang asli. Saat ini mempunyai dua varian, yaitu:
  - Common Lisp (Steele, 1990)
  - Scheme (Kelsey, 1998) - www.drscheme.org
- Tidak memiliki pernyataan assignment yang tidak sesuai secara matematika, seperti $x = x + 1$.
- Program ditulis sebagai fungsi.

Ekspresi

- Ekspresi disusun dalam notasi Cambridge-prefix.
- Contoh: (+ 2 2)
- Operasi aritmatika:
  - $(+) \rightarrow 0$
  - $(+\ 5) \rightarrow 5$
  - $(+\ 5\ 4\ 3\ 2\ 1) \rightarrow 15$
  - $(*) \rightarrow 1$
  - $(*\ 5) \rightarrow 5$
  - $(*\ 1\ 2\ 3\ 4\ 5) \rightarrow 120$
- Contoh lain: $(+\ (*\ 5\ 4)\ (-\ 6\ 2)$

Variabel

Variabel global

- Didefinisikan dengan menggunakan fungsi define.
- Contoh: (define f 120)

Evaluasi ekspresi

- f → 120
- (+ f 5) → 125
- (f) → error, kenapa?
- 5 → 5
- #f → false
- #t → true

Evaluasi Ekspresi

```
(define warna (quote (merah kuning hijau)))
(define warna '(merah kuning hijau))

(define x f)                120
(define x 'f)               x berisi simbol f
(define warna ' merah)
(define warna merah)        error, kenapa?
```

PLT Scheme

- PLT Scheme memiliki dua tools utama
  - MzScheme : the core compiler, interpreter, and run-time system
  - DrScheme : the programming environment
- DrScheme memiliki beberapa variant. Untuk menggunakan Scheme standar:
  - Pilih Module (Choose Language — Module)
  - Definisikan #lang scheme pada definition area.

Define

- Bentuk:

```
( define <id> <expr> )
( define ( <id> <id>* ) <expr>+ )
```

- Contoh:

```
#lang scheme
(define x 5)                    ; x sbg var bernilai 5
(define (potong st)             ; potong sbg fungsi
   (substring st 0 x))

> x
5
> (potong "hello world")
"hello"
```

Conditionals : if

- Nilai Boolean:
  - #t → true
  - #f → false
- Bentuk IF

```
(if <expr> <expr> <expr>)
```

- Contoh:

```
(define (maks a b)
   (if (> a b) a b))

> (maks 5 10)
10
```

Conditionals : and, or

- Bentuk AND dan OR:

```
(and <expr>*)
(or <expr>*)
```

- Contoh:

```
(define (test n)
   (if (and (>= n 5) (<= n 20))
      "di dalam" "di luar"
   )
)

> (test -4)
"di luar"
```

Conditionals : cond

- Bentuk cond:

```
(cond {[ <expr> <expr>* ]}*)
```

- Contoh:

```
(define (mutu n)
   (cond
      ((= n 4) "A")
      ((= n 3) "B")
      ((= n 2) "C")
      ((= n 1) "D")
      ((= n 0) "E")
      (#t "BL")
   )
)
```

Conditionals : when, unless

- Bentuk:

```
(when <test-expr> <then-expr>)
(unless <test-expr> <then-expr>)
```

- Contoh:

```
(define (bagi a b)
   (when (not (= b 0))
      (/ a b)
   )
)
; atau
(define (bagi a b)
   (unless (= b 0)
      (/ a b)
   )
)
```

Tipe Data

- Booleans, Numbers, Characters, Strings (Unicode)
- Bytes and Byte Strings, Symbols
- Keywords, Pairs and Lists, Vectors
- Hash Tables, Boxes, Void and Undefined

Boolean

- Nilai output : #t dan #f
- Fungsi boolean? untuk memeriksa apakah suatu nilai yang diperiksa adalah boolean.
- Contoh

```
> (= 2 (+ 1 1))
#t
> (boolean? #t)
#t
> (boolean? #f)
#t
> (boolean? "no")
#f
> (if "no" 1 0)
1
```

Numbers

- Bilangan dalam Scheme:
  - Exact
    Integer : 5, 99999999999999999, -17 Ratio : 1/2, 99999999999999999/2, -3/4 Complex number : 1+2i, 1/2+3/4i
  - Inexact : 2.0, 3.14e+87
- Fungsi untuk memeriksa : integer?, rational?, real?, complex?, number?
- Fungsi tipe data: #e (exact), #i (inexact), #b (binary), #o (octal), #x (hexa)

Numbers

```
> 0.5
0.5
>#e0.5
1/2
>#x03BB
955
>(/ 1 2)
1/2
>(/ 1 2.0)
0.5
>(if (= 3.0 2.999) 1 2)
2
> (inexact->exact 0.1)
3602879701896397/3602879701896396
```

Character

- Character dalam Scheme $\simeq$ Unicode scalar value.
- Jenis
  - Printable char : #\ diikuti dgn char
  - Unprintable char : #*u* diikuti dgn hexa
- Beberapa Fungsi:

```
char->integer
integer->char
display
char-alphabetic?, char-numeric?, char-whitespace?
char-downcase, char-upcase
char=?      Bandingkan dua atau lebih char
char-ci=?   Tidak memperhatikan upper/lower case
eqv?        Sama dengan equal? untuk char
```

Character

```
> (integer->char 65)
#\A
> (char->integer #\A)
65
> #\u03BB
#\?
> (integer->char 17)
#\u0011
> (char->integer #\space)
32
> #\A
#\A
> (display #\A)
A
```

Character

```
> (char-alphabetic? #\A)
#t
> (char-numeric? #\0)
#t
> (char-whitespace? #\newline)
#t
> (char-downcase #\A)
#\a
> (char=? #\a #\A)
#f
> (char-ci=? #\a #\A)
#t
> (eqv? #\a #\A)
#f
```

String

- Array of characters
- Contoh:

```
> "Apple"
"Apple"
> (display "Apple")
Apple
> (display "a \"quoted\" thing")
a "quoted" thing
> (display "two\nlines")
Two
lines
```

Latihan

- Buat program membaca nama dan menampilkan seperti contoh.

```
> (halo  "Scheme Nurachem")
Selamat Datang Scheme Nurachem
KOM204
```

- Buat program menjumlahkan tiga bilangan.

```
> (jumlah 4 6 8)
18
```

- Buat program menentukan bilangan terkecil dari tiga bilangan.

```
> (min 6 4 6)
4
```

Latihan

- Buat program menentukan salah satu akar persamaan kuadrat $ax^2 + bx + c = 0$. Program tidak menghasilkan apa-apa bila $a = 0$ atau $b^2 - 4ac < 0$.

```
> (akar 1 1 -6)
2
```

- Buat program menghitung nilai faktorial

```
> (faktorial 4)
24
```

- Buat program menentukan nilai fibonacci ke-n

```
> (fib 10)
55
```